UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

REFLEXIONES FILOSÓFICAS SOBRE LA DEMOSTRACIÓN MATEMÁTICA A LA SOMBRA DE THOMAS TYMOCZKO PROYECTADAS DESDE ALGUNAS DEMOSTRACIONES COMPUTARIZADAS

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

M A T E M Á T I C O

P R E S E N T A:

M A U R I C I O T O R R E S V I L L A

FACULTAD DE CIENCIAS
UNAM

DIRECTOR DE TESIS: DR. BEMJAMÍN MACÍAS PIMENTEL

2006

A mis padres, Jaime y Yolanda,

por su incompre(n)sible paciencia, ayuda y confianza.

A mis hermanos, Axell y Emmanuel,

por la tautología de la sangre, el tatuaje del querer.

A mi abuela, Sebastiana,

porque el convencimiento también comienza en la panza.

A mi hada malandrina,

lechosa luz dispersora de sombras mortecinas.

A la bandera de ciencias,

por su asistencia en la remoción de varias neuronas tontas y necias.

A Benjamín y Menelao,

por su apoyo continuo y desinteresado.

Por último y desde un principio, a mis amigos reales e imaginarios,

sus complejas relaciones conmigo raíces me han dejado.

Agradecimientos

Quiero agradecer a Benjamín por enseñarme una visión menos ingenua y fundamentalista de las matemáticas, aún quedan algunos dogmas por derruir y en el olvido hacerlos explotar (veneno mata veneno); a Favio, por revelarse como el eslabón perdido cuyo hallazgo permitió cerrar este ciclo, a Julio César porque a César lo que es del César (gracias por reafirmar en mí el interés por la historia), a Carlos por su incondicional respaldo y sobretodo por presentarme a la lógica matemática, disciplina de apariencia rígida pero con perseverancia brindadora de experiencias exquisitas y a Liliana por su dedicación al revisar este trabajo.

Por desgracia el acto esencial de dar las gracias tropieza con mi austero espíritu falto de gracia, así entonces con torpeza prosigo terminando súbitamente este apartado.

ÍNDICE

INTRODUCCIÓN 11

Capítulo I: EL DESPERTAR DE UN NUEVO SIMIO 19

The Logic Theory Machine.....19

Hacia la mecanización.....27

Postdata: The Geometry Machine.....31

Capítulo II: LA PINCELADA DE LOS CUATRO COLORES 39

Nociones Previas.....39

I believe I can prove it 42

Heaven is angered by my arrogance 44

Although his" proof" turned out not to be complete it contained most of the basic ideas 47

There is a price for this sort of knowledge: it cannot be absolute.....56

Postdata.....76

Capítulo III: LA RESOLUCIÓN INDUSTRIAL 92

Disolución.....92

Resolución.....117

Soluciones.....145

Absolución.....155

Postdata: Laberinto semántico.....180

Capítulo IV: LA INDUCCIÓN Y LA INCERTIDUMBRE 211

Principio.....211

Intermedio.....221

Fin.....237

PRELUDIOS Y FUGAS 264

Fugas.....264

Preludios.....267

BIBLIOGRAFÍA 277

INTRODUCCIÓN

Queremos examinar con todo cuidado aquellas construcciones conceptuales y aquellos métodos de investigación que enriquezcan a nuestra disciplina, queremos cultivarlos, apoyarlos y servirnos de ellos siempre que se presente la más ligera posibilidad de obtener un resultado.

D. Hilbert, Acerca del infinito

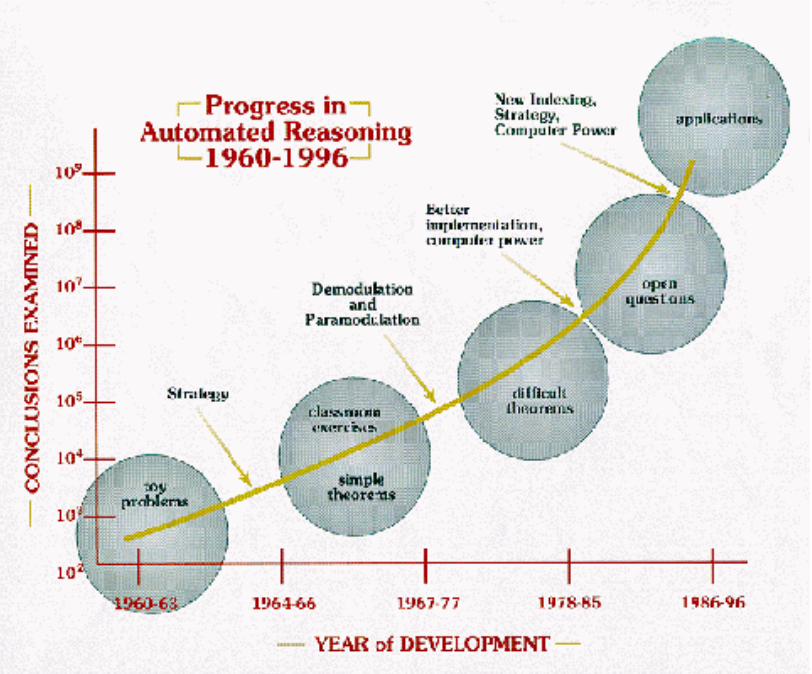
Sin lugar a dudas, la demostración matemática busca trasladar a las conjeturas a un lugar sin dudas, a un santuario donde las nombran teoremas. La autoridad de la razón capacitada determina a los candidatos merecedores del revestimiento real. Autoridad no implica autoritarismo, mucho menos arbitrariedad. Las matemáticas se han sometido a la revisión crítica de su proceder en reiteradas ocasiones, ajustándose a veces a los resultados de dichos análisis. Algunos defenderán la inmutabilidad del objeto de estudio, pero nadie con la mínima consciencia histórica avalará la de la actividad. La demostración es un elemento esencial de las matemáticas ya que es fuente y garantía de conocimiento. En consecuencia la necesidad de inquirir el concepto de "demostrar" ha sido, es y será imperiosa mientras nos propongamos hacer matemáticas.

Los sistemas computacionales desde mediados del siglo XX comenzaron a ser una herramienta de uso insistente y hasta indispensable en varias actividades científicas e intelectuales, las matemáticas no han sido la excepción. De hecho, el empleo de las computadoras ha alcanzado al corazón deductivo de dicha disciplina, existiendo desde hace algunas décadas programas para asistir en la elaboración y verificación de demostraciones (en el sentido inverso, las ciencias de la computación han recibido frecuentemente la colaboración de las matemáticas en su desarrollo; es más, su centro neurálgico, la teoría de la computación, en gran medida está escrita en caracteres matemáticos). A los objetos deductivos producidos por medio de programas los llamaré demostraciones computarizadas, epíteto agregado por sus variopintas peculiaridades congénitas, características tan singulares en algunos casos como para sospechar si el adjetivo tiene un sustantivo a quien acompañar. Estos esperpentos digitales pudieron deambular inadvertidos para la comunidad matemática (con la posible excepción de lógicos-matemáticos y aquellos involucrados en el área) a no ser de sus éxitos en algunas conjeturas no despejadas por los matemáticos de carne y hueso. Tras las victorias de los monstruos deductivos de origen binario emergieron patrones de recelo y escepticismo. Donde las vacilaciones suenan los ríos filosóficos resuenan, aun si no llevan agua. La noción de la demostración por culpa de las demostraciones computarizadas fue esculcada y manoseada en pos de incluir o desterrar a estas demostraciones de apariencia morbosa. La enfermedad enseña sobre la sanidad o al menos sobre lo que consideramos ser saludable. De este modo, el agente patológico digital provocó y sigue ocasionando reflexiones sobre lo que es una demostración, reflexiones muchas veces escoltadas por una caracterización de ese extraño objeto del conocimiento. Quienes están saludables raramente reparan en la enfermedad, mejor aún, ni siquiera les

preocupa pensar acerca de la salud. La mayoría de los matemáticos pueden proseguir con su trabajo demostrativo sin detenerse a meditar sobre lo que están haciendo, han sido entrenados para escribir, leer, comprender demostraciones pertenecientes a sus campos de especialización y menos que esporádicamente se detienen en seco a sopesar su trabajo fuera del contexto en donde se sitúa. La anterior opinión dista de ser una crítica corrosiva, al contrario, es un síntoma del funcionamiento saludable y vigoroso de la actividad matemática poder dispensar de una conciencia de sus actos. Venenosa advertencia escupida desde el Eclesiastés y cuajada por Dostoyevski: el dolor es la causa de la consciencia y la conciencia, solo, origina sufrimiento. Sólo un pequeño grupo constituido principalmente por lógicos-filósofos trasnochados está interesado en soportar estas espinosas inquisiciones aunque este conjunto de masoquistas ha crecido con la incorporación de algunos computólogos y la adquisición de nuevos fustes binarios, dicha medra ha estado motivada principalmente por la intromisión de la computadora en las labores demostrativas y ha sido auspiciada por las preguntas con sus posibles respuestas planteadas a partir de las demostraciones computarizadas. Objetivo de este ensayo es retomar algunas de ellas y batirlas con fuerza para despertar del estupor autocomplaciente a sacudidas por medio de las caricias de los flagelos.

Aunque la discusión primaria ronde alrededor de la noción de la demostración, algunas trampas con herrumbre filosófica y óxidos de desidia son infranqueables. ¿Cómo conoceríamos a las bacterias sin el microscopio o las estrellas sin el telescopio? *Así como el físico examina sus aparatos, el astrónomo su punto de referencia y el filósofo lleva a cabo una crítica de la razón, también el matemático se ve obligado a asegurar sus teoremas, y para ello requiere de una teoría de la demostración* (D.H., La nueva fundamentación de las matemáticas). Debido a las demostraciones computarizadas, ¿podemos evitar agregar a la lista de Hilbert a las computadoras dentro del rango a inspeccionar por los matemáticos? El instrumentalismo en las matemáticas no depende exclusivamente de la aparición de las demostraciones computarizadas; la historia de los aparatos utilizados en el quehacer matemático es rica, extensa y talvez un poco menospreciada. La participación activa de las computadoras en la creación y validación de demostraciones puede convertirse en la chispa adecuada para la ignición de una nueva representación de la actividad matemática menos idealista, una representación donde los objetos materiales tengan papeles no tan secundarios. La inflamabilidad está contenida en la siguiente pregunta: ¿conoceríamos al teorema x sin el instrumento computadora? Vaticino apresurado, algunas demostraciones computarizadas han sido aceptadas por la comunidad matemática aun cuando su validación únicamente por otros programas ha sido efectuada, peor todavía, esas demostraciones son las únicas en existencia. Así pues, la pregunta previa pertenece por el momento a los planteamientos característicos de la ciencia ficción, tiene bases sólidas aunque adolece de una exagerada extrapolación. Por sanidad mental evito la versión zen-ontológica ¿existe "a" sin el agente "b"? Tampoco voy a confundir ontología con epistemología, si los procesos relacionados con la génesis del conocimiento son contingentes, como lo evidenciaría el advenimiento de los oráculos binarios, esta propiedad de nuestro entendimiento no constituye (i) un alegato suficiente contra el platonismo, (ii) ni siquiera se consolida como un argumento en contra de la

esquizofrenia ideal [(i) Cada cuervo por mí percibido es negro, entonces ¿"Todos los cuervos son negros" es la única posibilidad (antiposibilidad-necesidad) dentro de la baraja de las posibilidades? ¿Quién dentro de un marco deductivo se conduce de esta manera? No obstante, el camino inductivo no está clausurado de antemano; bienvenidos sean quienes enumeren el conocimiento, hallen el caso base y el conozco contingentemente x implica el conocimiento contingente de x+1 (ii)Equivaldría a equiparar a la epistemología con la ontología, anacronismo del régimen absolutista de la Verdad] . Empero, todo parece indicar la venida de una etapa de palpitaciones tumultuosas sino discernimos sobre la acción de las computadoras en la deductiva bomba central de las matemáticas, sino se aclara las consecuencias del marcapasos en los latidos de las demostraciones afectadas. El panorama de las demostraciones computarizadas debe ser desmenuzado para ser procesado. Hay de varios tamaños y colores, según el gusto y las intenciones de sus creadores-programadores. En el actual trabajo se presenta una pequeña muestra significativa. El criterio de selección se sustenta en una ponderación de su relevancia (i)matemática (ii)filosófica (iii) computacional (iv) histórica. No son todas las que están, pero si están todas las que son básicas para conformar el esqueleto del trabajo, la exégesis de la demostración dimanada por las pulsaciones del on/off. La gráfica presentada después de este párrafo proviene de la página de internet de Larry Wos [http://www-unix.mcs.anl.gov/~wos/index.html], eminente figura del razonamiento automático-izado deductivo (fuente y subtipo de algunas demostraciones computarizadas). Con un rango de error amplio pero tolerado sirve para incrustar cronológicamente la curva del desarrollo de las demostraciones computarizadas. Literalmente comenzaron con párvulas proposiciones hasta eventualmente sacar a la venta en la década de los setenta su primer gran éxito: la demostración computarizada del Teorema de los Cuatro Colores, teorema insigne de la teoría de gráficas. Con reniegos y discrepancias, la comunidad finalmente la compró a pesar de que su validación también recayó en otros programas. Se abrió la computadora de Pandora, brotaron improperios, encomios, églogas, elegías, odas etc. distribuidos en libros y artículos (me uno al club con el formato tesis). ¿La revolución digital triunfó en el deductivo corazón de las matemáticas? Sí y no (!). El impacto en la empresa deductiva ha sido mesurado; se han producido algunas otras demostraciones binarias de problemas no resueltos por el sacrosanto intelecto de algun(os) matemático(s) como la demo stración computarizada de McCune del teorema de Robbins (relativa al álgebra booleana), no obstante la práctica de apelar a la computadora para fabricar demostraciones no se ha extendido sustancialmente en los practicantes de las matemáticas. En este escr ito se exhibirán algunas razones de este desdén, algunas de ellas íntimamente relacionadas con un área donde causaron revuelo: la filosofía de las matemáticas, en particular en la noción de la demostración. De una vez aclaro, revuelo no es sinónimo de novedad.



No es la meta de este ensayo definir de una vez por todas qué es una demostración, en general esta empresa peca por pretenciosa y en particular siendo un aspirante a matemático parece irrisoria. Con mi escasa experiencia comprendo más no comparto la inclinación al hermetismo sobre la noción de la demostración, encasillándola dentro de las matemáticas como un argumento cifrado donde sólo los miembros del área pueden decodificarlo. Posición sugerida por el Wittgenstein de "Observaciones sobre los Fundamentos de la Matemática", nada más se puede decir acerca de las demostraciones: son ilaciones convincentes para los matemáticos. La reducción de la demostración a la prueba formal aunque la cimienta dista de ser concluyente y como descripción no trasciende del ámbito de la representación insípida. Al revisar la vida de las demostraciones computarizadas – ejercicio similar puede hacerse con otras demostraciones– emergen procesos de índole social partícipes en la creación/verificación/adopción de éstas, algunos de ellos se señalarán a lo largo de la tesis según sea el caso. El tema propicia diversas perspectivas de análisis, imposible abarcarlas todas si no se quiere sacrificar la cohesión. A esta sección compete especificar los límites de la amplitud de la revisión, la profundidad en los capítulos será escarbada, capítulos a continuación delineados históricamente a paso veloz:

Capítulo I: EL DESPERTAR DE UN NUEVO SIMIO. Este capítulo es un recuento del nacimiento de los programas demostradores de teoremas aparecidos en el ombligo del siglo XX: la Máquina Lógica Teórica (*Logic Theory Machine* - LTM) de Simon, Newell y Shaw y su posterior pariente cercano, la Máquina Geométrica (*Geometry Machine* -GM) de Gelernter, Hansen y Loveland. Obedeciendo una postura histórica ortodoxa, me parece pertinente remembrar al origen, pero tampoco he de sacralizarlo ni simplificarlo a mi beneficio. La motivación detrás del proyecto de la Máquina Lógica Teórica no era la construcción de una herramienta matemática para la consecución de demostraciones, la

meta de Simon et al era mucho más ambiciosa: buscaron infundirle "inteligencia" a la computadora emulando cómo posiblemente maniobraba un matemático en sus deducciones. Antes de caminar erguido se aprende a gatear; los teoremas demostrados por la LTM eran sencillos al igual que el lenguaje de la prueba, eran teoremas del cálculo proposicional extraídos de la monumental obra de Russell y Whitehead *Principia Mathematica*. Simon et al identificaron y propusieron algunos métodos relacionados con el descubrimiento de una demostración, para luego reproducirlos y programarlos. La LTM fue una pieza primordial dentro de la incipiente área de la inteligencia artificial (IA) y de rebote inauguró la investigación sobre la automatización de la generación de demostraciones. Los objetos deductivos maquilados por la LTM estaban incrustados en un marco lógico, por lo cual llamó sobretodo la atención de los estudiosos de esta disciplina. Hao Wang, eminente lógico del siglo XX, se preocupó más por la mecanización de la producción de demostraciones que por la inteligencia artificial, enfocándose en el primer problema y desprendiéndose del lastre del segundo. No fue el único. Además, se puso en la mira a un sistema formal más rico, el cálculo de predicados de primer orden. Más aún, Wang fue de los primeros en vislumbrar la potencialidad de las computadoras en la actividad matemática, así como también, algunas de sus ideas referentes a la demostración atañen a las demostraciones computarizadas residentes en un sistema formal y en este primer capítulo algunas de esas observaciones serán discutidas. Por último, como exhibición de otras gracias programables en una computadora, se describe brevemente a la Máquina Geométrica y su habilidad para guiar sus derivaciones deductivas "semánticamente".

Capítulo II: LA PINCELADA DE LOS CUATRO COLORES. En 1976 Appel, Haken y Koch sorprendieron a la comunidad matemática por dos razones:

(i) Clamaban haber demostrado una rancia conjetura reacia a doblegase.

(ii) Recurrieron a la computadora, convirtiéndose ésta en una protagonista "indispensable" en su candidato a demostración

El primer capítulo fue el proemio, a partir de aquí comienza la trama en serio. Appel, Haken y Koch con la ayuda de algunos programas demostraron el Teorema de los Cuatro Colores, sin embargo, su demostración era todo menos convencional. A diferencia de los programas demostradores generales restringidos a un sistema formal mencionados en el primer capítulo, programas cuyo objetivo es encontrar la prueba de cualquier teorema perteneciente al sistema formal de su dominio, los programas elaborados por Koch servían exclusivamente para demostrar un lema clave en la polémica demostración del famoso teorema. Desgraciadamente, debido a la exorbitante cantidad de operaciones realizadas por estos programas, la revisión de sus resultados abruma a cualquier inspector humano por lo que fue relegada a otros programas. El hedor de la anterior peculiaridad atrajo a varios buitres, entre ellos al filósofo Thomas Tymoczko quien después de enunciar una caracterización poco arriesgada de las demostraciones (y muy conveniente para sus intenciones discriminatorias) atacó a la demostración computarizada del crucial lema indicando las violaciones a su caracterización y de paso preparando el terreno para sus digresiones filosóficas. Las réplicas no se hicieron esperar, resaltan las de Paul Teller y Edward Swart. En este capítulo se recopilan algunos ecos de la discusión y se propone-defiende una modificación a la caracterización de Tymoczko, una reformulación con la

ventaja de albergar directamente a la demostración computarizada del conflictivo lema y a las demostraciones computarizadas similares.

Capítulo III: LA RESOLUCIÓN INDUSTRIAL. Las demostraciones artesanales computarizadas (LTM, GM), al pretender simular el estilo humano no aprovechaban el poder bruto computacional. A finales de los sesenta, Alan Robinson diseñó una técnica innovadora, la resolución, la cual ajustó los engranes optimizando el procedimiento computacional, al mismo tiempo se alejó de las usuales rutas derivativas transitadas por el *homo sapiens*. Las expectativas de la prometedora resolución sedujeron a varios investigadores porque prometía la conquista del cálculo de predicados de primer orden. Los sueños de opio de traducir las maneras racionales del simio erecto a la máquina se habían difuminado, el medio tiene peso mientras de resultados. Desgraciadamente la resolución tampoco los dio, al menos no proporcionalmente a las altas expectativas levantadas por ella. Graves limitantes teóricas minaron –y todavía minan- su camino, proviniendo el obstáculo más explosivo de la teoría de la complejidad computacional. No obstante algunos necios diligentes (como L. Wos) no abandonaron la vía de la resolución, la modificaron y mejoraron. Su determinación empezó a rendir frutos y saltaron a la fama mundial en 1996 gracias a la demostración de McCune de la Conjetura de Robbins conseguida con el socorro del programa EQP, conjetura reticente a convertirse en teorema hasta la conjunción hombre-máquina citada. En este capítulo se examina bajo la sombra de la caracterización modificada de Tymoczko a las demostraciones computarizadas salidas de sistemas demostradores resolutivos y sistemas afines, respaldando algunas exigencias promulgadas por esta caracterización mediante la evidencia histórica y la argumentación filosófica. Por otro lado también se reseñará otra atractiva aplicación de los asistentes digitales deductivos: la verificación de pruebas, útil en pequeña escala en la validación de la demostraciones computacionales habitantes de un sistema formal y a gran escala para soñar –preciso: soñar despiertos- con la formalización digital de las matemáticas, obra faraónica cuyos principales granitos de arena hasta el tiempo presente han sido colocados por el proyecto Mizar.

Capítulo IV: LA INDUCCIÓN Y LA INCERTIDUMBRE. Espoleados por el distanciamiento de los demostradores resolutivos con respecto a los medios deductivos de su alma mater de carbono, Robert Boyer y J Strother Moore en la década de los setenta incorporaron a la inducción y finalmente a su madre nutricia a su familia de programas demostradores de teoremas. Boyer y Moore desecharon a la automatización total debido a la mejora de la eficiencia provista por la pericia de los usuarios cabalmente preparados en el manejo de los programas y con una sagaz experiencia en el hábitat del problema, dando origen a un sistema híbrido de demostradores mecánicos interactivos. Para evitar confusiones desde aquí lo aclaro, tampoco los sistemas demostradores resolutivos e hijos del capítulo anterior se conforman con recibir como entrada a un teorema más un conjunto de axiomas mientras que el usuario tras picar un botón nada más se pica los ojos en la espera de la prueba manufacturada. La tarea no es tan sencilla como se explicará en el capítulo correspondiente. Sin embargo el nivel de interactividad de esos programas no es equiparable con el proporcionado por los demostradores de la familia Boyer-Moore,

sistemas demostradores en donde literalmente la prueba es construida paulatinamente por el usuario y por el programa. En este capítulo también se especifica con más palabras al motor primario del desarrollo de los programas demostradores: esta rama de investigación se impulsó y todavía en parte se sigue alentando debido a sus aplicaciones y repercusiones en la verificación de software/hardware. La meta de dicha rama de estudio es asegurar el correcto funcionamiento de los sistemas computacionales, siendo su aspiración extrema corroborarlo de manera deductiva e incluso para algunos de sus desarrolladores aseverarlo de forma "absoluta". ¿Este último objetivo es asequible? Los polos de la respuesta a esta pregunta pueden generar un movimiento pendular entre la certeza y la incertidumbre adheridas a los objetos deductivos digitalmente manufacturados, en especial aquellos cuya revisión fue realizada por otros programas; si la incertidumbre se impone entonces su extensión hacia el resto de las demostraciones inquietante se ofrece. Entre otros logros en las matemáticas de la conspicua familia del demostrador Boyer-Moore resaltan los conseguidos por uno de sus miembros de nombre Nqthm , quien de la mano de Shankar Natarajan en 1986 construyó una demostración del teorema de incompletud de Gödel y en 1994 en complicidad con Kenneth Kunen hicieron lo propio para el teorema Paris-Harrington-Ramsey. Las demostraciones computarizadas maquiladas por ACL2, integrante más reciente de la familia Boyer-Moore, también se someterán a la caracterización mutada de Tymoczko para no descuidar su calibración.

Situado el contexto del objeto de estudio, las propiedades ópticas de la lupa también deben ser indicadas. Este no es un escrito de sociología del conocimiento, aunque se puntualicen reiteradamente algunos factores sociales inmiscuidos en la génesis/verificación/adopción de las demostraciones. Son elementos presentes e inevitables –el hombre mientras sea hombre es un organismo social- pero me parece peligroso e insostenible entronarlos como los factores preponderantes. Deseo evitar a toda costa a las abruptas simplificaciones recortadas en los extremos. Tampoco es un ensayo de filosofía pura ni dura, es un ensayo de filosofía aplicada cuyo hilo central es la caracterización de Tymoczko de la demostración. Si bien aparecerán asuntos filosóficos de prístina prosapia, v.gr. la epistemología en varias ocasiones se asoma entre líneas, para bien del escrito no me propongo abocarme enteramente a ellos ni dar por concluido algunas de sus añejas disputas. En esta tesis no regirán la afirmaciones del estilo "el conocimiento es tal o cual cosa" y aunque las haya, deben ser comprendidas como modelos cuya grado de precisión trataré de solventar. Por lo tanto quiero y debo mantener el contacto entre la representación y lo representado, así entonces desde el primer capítulo desfilarán demostraciones computarizadas y las explicaciones de cómo fueron generadas. Obedeciendo a esa honestidad crítica, también son atrapadas y relatadas algunas de las reacciones suscitadas en la comunidad matemática por algunos de las demostraciones computarizadas en pos de ponderar a la caracterización de la demostración propuesta desde el segundo capítulo. Insisto, es una evaluación histórico-filosófica de una caracterización erigida a su vez sobre un examen histórico-filosófico cuyo punto de partida fue la demostración computarizada de Haken, Appel y Koch del Teorema de los Cuatro Colores, examen prolongado hacia otras demostraciones computarizadas cuya intención final es abarcar con una exactitud tolerable a la demostración en general restringida al contexto histórico actual. Precaución desde aquí

emitida: en ningún momento Tymoczko pregona por la imposición de una serie de normas para regular la aceptación o rechazo de una demostración ni yo tampoco lo haré. Tengo una animadversión hacia las falsas cenicientas, aquellas quienes buscan ajustar el pie al zapato a la fuerza. El modelo propuesto es un instrumento de análisis, en su poder explicativo reside la calidad del calzado. No me apresuro, sería demasiado prematuro enunciar aquí a la susodicha caracterización, quien quiera darse una idea, lo mando a la sección postrera. En la última parte de la tesis además de señalar otros rumbos para las indagaciones emanadas de las demostraciones computarizadas, aprovecho la inercia de los capítulos previos para defender y alimentar a la caracterización mentada.

Si son de las personas cuya celeridad existencial necesita de la aburrida concisión y de la hipócrita revelación de los objetivos últimos desde el principio, esta tesis es una apología de la demostración computarizada, acoto, es una apología de algunas de ellas. Examinemos con todo cuidado aquellas herramientas que enriquezcan a nuestra disciplina. Así sea.

CAPÍTULO I: EL DESPERTAR DE UN NUEVO SIMIO

El primer paso de este ensayo nos catapulta al origen del objeto nodal, las demostraciones generadas por medio de una computadora. Aprovecho el estruendo explicativo del big-bang de la demostración digital, la Máquina Lógica Teórica, para explicitar algunas nociones lógicas necesarias para seguir una ruta de la evolución demoscósmicabinaria. La profundidad es dispensable, sin embargo una familiaridad con los sistemas formales es ineludible. El cálculo proposicional es el punto de partida, no obstante el cálculo de predicados empieza a vislumbrarse desde este capítulo, felizmente el arribo se realizará en el capítulo tercero. Por otro lado, en la sección dedicada a las profecías de Wang, comienza el análisis de las demostraciones computarizadas inmersas en un sistema formal bajo un umbral amplio, desde lo técnico (lógica, poder de cómputo) hasta lo filosófico e incluso lo social (pedagogía). La sección terminal de este capítulo, el fugaz delineamiento de la Máquina Geométrica, viene como complemento, es un postre. La Máquina Geométrica esta endulzada con una semántica, en el capítulo tercero se discutirá de nuevo esa incorporación azucarada en nuestros asistontos binarios.

THE LOGIC THEORY MACHINE

Logic is a subject taught in college courses, and is difficult enough for most humans.
Newell, Shaw & Simon

Herbert A. Simon se formó académicamente con la arcilla de las ciencias sociales (B.A,& Ph.D en ciencias políticas, Universidad de Chicago) y complementó su preparación con el molde de las ciencias "duras"(v.gr. estudió lógica con Carnap, Russell y economía matemática con Shultz). Bajo el auspicio de esta fértil combinación su obra fue fructífera y profusa, incluso lo hizo merecedor del Premio Nobel en Economía:

...la Academia Real de Ciencias ha decidido otorgarte en 1978 el premio Nobel en las Ciencias Económicas por tu investigación pionera en el proceso de toma de decisiones en las organizaciones económicas [1]*...*

Su idea principal en torno a la toma de decisiones es un alegato en contra de la visión clásica de la compañía como un ser omnisciente, racional y maximizador de ganancias. Simon parte de su apreciación sobre la manera en cómo los individuos se enfrentan a los problemas y la extiende a través de una estructura relacional hasta el nivel organizativo. Quienes toman decisiones poseen un conocimiento limitado de las consecuencias de sus fallos, ergo en lugar de buscar la mejor de las alternativas se deben conformar con opciones localmente satisfactorias. En palabras de Simon:

Es imposible para el comportamiento de un individuo solo e aislado alcanzar cualquier grado alto de racionalidad. El número de alternativas que él debe explorar es tan grande,

la información que él necesitaría para evaluarlas tan inmensa que incluso una aproximación a una racionalidad objetiva es difícil de concebir[2].

La curiosidad inquisitiva de Simon lo impulsó hacia otras áreas propicias para explotar sus ansias exploradoras en nuestras capacidades cognoscitivas. Sus hipótesis como la finitud de nuestra base de conocimiento o la búsqueda de soluciones parciales las aplicará constantemente en las investigaciones en el tópico donde fue pionero, la inteligencia artificial (IA). La retroalimentación y cooperación con el físico-matemático (actualmente lo designaríamos computólogo) Allen Newell fue determinante para sacar a la luz en 1956 al padre de los probadores mecanizados de teoremas, la *Logic Theory Machine* (LTM), también considerada en su momento como el primer peldaño de la escalera de la inteligencia artificial hacia las regiones superiores del intelecto. La lógica le está negada a la mayoría de los hombres, cita textual de Newell y Simon . Ofrece un par de ventajas adoptarla en la IA: (i) Estar por encima de la media intelectual del homo sapiens con respecto a una actividad racional en concreto (ii) La formalización de la lógica desarrollada vigorosamente por los lógicos-matemáticos desde la postrimería del siglo XIX facilita su implementación en la computadora. El mérito (i) tambalea altivo e ingenuo, sin embargo modificándolo drásticamente se transformará en uno de los polos de la IA: (i.a) La lógica es el mecanismo tras bambalinas del intelecto del homo sapiens; atenuémoslo : (i.a.i) La lógica al menos está presente en algunos procesos racionales enfocados a resolver problemas. Bajo el manto de (1.a.2) se cobija la LTM y en general también lo hace el área de la IA cualificada como "dura" o "limpia".

La interacción con Newell nutrió a Simon con cuestiones técnicas sobre las computadoras. El equipo se completó con el programador Cliff Shaw. Las siguientes líneas extraídas de su famoso reporte sobre la LTM revelan el objetivo de su investigación:

La investigación reportada aquí está dirigida al entendimiento de los procesos complejos (heurísticos) que son eficaces en la resolución de problemas. Sin embargo no estamos interesados en métodos que garantizan las soluciones pero que requieren inmensas cantidades de cómputo. Más bien, nosotros deseamos entender cómo un matemático, por ejemplo, puede demostrar un teorema aun cuando él no sabe cómo empezar ni si va a tener éxito.[3]

El anterior párrafo es un efluvio de las ideas de Simon sazonadas con el concepto "heurístico". Newell fue alumno de George Polya, quien popularizó este apelativo en su libro *How to solve it*. Simon, Newell y Shaw (S&N&S) definen el término "heurístico" contrastándolo con "algorítmico". Parafraseo: un procedimiento con posibilidades de resolver un problema sin la garantía de hacerlo se le nombra "heurístico", mientras que si asegura solucionarlo se llama "algorítmico". El matrimonio, divorcio o amorío entre este par de ideas marcarán como a los hombres y a las vacas el desarrollo (o embrollo) y la pertenencia (o penitencia) de las demostraciones computarizadas. En este capítulo se comentarán algunas quejas del lógico chino Wang sobre el enfoque "heurístico" de la LTM,

aunque Wang consciente de sus gracias no vacila en proponer y sostener un affaire con la estrategia criticada.

Previo a zambullirnos aL Tenebroso pozo de deMócrito, una lista aportación del trabajo de S&N&S a las ciencias de la computación aboga por una breve mención: el lenguaje de programación de la LTM es el primer lenguaje de procesamiento de listas. Bautizado como *Information Processing Language* (IPL), es descrito por Newell y Shaw en un anexo al reporte de su LTM. Esta clase de lenguajes de programación es ampliamente usada en la IA y en particular también ha sido empleado en los demostradores automáticos. Incluso el sucesor inmediato de IPL, LISP, diseñado por John McCarthy en 1958 fue catalogado como EL lenguaje de programación de los investigadores de IA por algún tiempo (algo de LISP se expondrá en el cuarto capítulo).

Las demostraciones de la LTM están estrictamente apegadas a las leyes lógicas. Su fuente y censor es el primer volumen del *Principia Mathematica.* El LTM es un programa diseñado para demostrar teoremas en el cálculo proposicional (CP). El lenguaje manejado por la LTM consta de:

- variables: p,q,r, ..., A,B,C, ...
- conectivos: “or”, “not-”, “implies”
- signos ortográficos: “(“,”)”,”[“,”]”

Evitando complicaciones y distracciones algunas consideraciones lógicas y físicas serán obviadas. Una descripción meticulosa nos remitiría hasta la era cavernaria de las tarjetas perforadas y hasta la perforación de la caverna pertrechada por un curso de lógica. Por el momento simplifiquemos y supongamos. Las fórmulas manipuladas por la LTM están bien formadas. La LTM posee la base axiomática del cálculo proposicional provista por Russell y Whitehead, es más, conserva la numeración de los axiomas y los teoremas contenida en *Principia*:

(1.2) (p or p) implies p
(1.3) p implies (q or p)
(1.4) (p or q) implies (q or p)
(1.5) [p or (q or r)] implies [q or (p or r)]
(1.6) (p implies q) implies [(r or p) implies (r or q)]

Además la LTM tiene programadas las siguientes reglas de derivación:

1. Regla de sustitución: Cualquier expresión (variable o fórmula bien formada) puede sustituir a cualquier variable en todas sus apariciones en una fórmula.
   Ej: Sustitución de “p” por “p or q” en (1.3)
   (p or q) implies [q or (p or q)]

2. Regla de reemplazo: Un conectivo puede ser remplazado por su equivalente lógico conformado por los otros conectivos del sistema en cualquiera de sus apariciones en una fórmula.
   Ej: Reemplazo de "implies" por "or","not-" en (1.3)
   not-p or (q or p)

3. Regla de separación (modus ponens): Si entre mis hipótesis tengo "A" y "A implies B" entonces puedo inferir "B".

A la LTM se le suministra como entrada un teorema A_e y ésta busca una demostración de éste. Las demostraciones de la LTM son pruebas en el severo sentido de la palabra para los lógicos: es una secuencia finita de fórmulas $A_1,.....,A_n$ donde p.c. 1=i=n:

A_i es un axioma
o
$\exists j((1 \leq j < i)$ & A_i es el resultado de la regla de sustitución aplicada a $A_j)$
o
$\exists j((1 \leq j < i)$ & A_i es el resultado de la regla de reemplazo aplicada a $A_j)$
o
$\exists k,j(1 \leq k,j < i$ & $j \neq k$ & A_i es el resultado de la regla de separación efectuada sobre A_j, $A_k)$

El teorema demostrado es A_n. ¿Cómo le hacemos para generar esta secuencia cuya A_n sea igual al teorema de entrada A_e? S&N&S en primer lugar imitan al conocimiento previo de un matemático, al agregar la opción de clavar en la memoria de la LTM los teoremas ya demostrados. Por lo tanto debemos adicionar la disyuntiva:

o
A_i es un teorema

La primera respuesta con sarna propuesta por S&N&S es la algorítmica. El cálculo proposicional es demasiado afable, para todo teorema del CP existe una prueba[4] a partir del conjunto de axiomas (1.2)..(1.6). Además podemos enumerar las pruebas-teoremas. El Algoritmo del Museo Británico estimula a la paciencia del espectador disfrutando una historia interminable hasta encontrar el tesoro de su agrado. El algoritmo construye todas los posibles pruebas de una forma sistemática, en cada instancia k (i)elimina duplicados (ii) corrobora si $A_n(k)=A_e$. Recomendación desprendida de las observaciones de S&N&S con respecto al rendimiento de dicho algoritmo: mejor dedícate a jugar a la lotería toda tu vida, tienes más probabilidades de ganar, además esta ociosa actividad tiene expectativas mucho más redituables. Trampa denostada por Wang, ante la efectividad rayana a la nulidad del algoritmo del Museo Británico casi cualquier otro procedimiento será mejor, en particular los heurísticos propuestos por S&N&S (bautizados como "métodos"). Antes de proseguir con la crítica de Wang, desglosemos a la contraparte de la ventajosa comparación y sean a continuación explicados:

- Método de sustitución: La prueba se busca aplicando las reglas de sustitución y reemplazo a los axiomas y teoremas ya demostrados por la LTM

- Método de separación: Si la meta es demostrar A_n, este método busca un axioma o un teorema de la forma "A implies A_n". En caso de encontrarlo, A se convierte en un nueva submeta.

- Métodos de encadenamientos: Se dividen en dos subtipos, hacia adelante y hacia atrás, aunque ambos están fundamentados en la transitividad de la implicación. Los dos se aplican si A_n es de la forma "A implies C". El encadenamiento hacia adelante busca un axioma o un teorema de la forma "A implies B"; en caso de encontrarlo "B implies C" se considera una nueva submeta. El encadenamiento hacia atrás es similar, si se encuentra un teorema o axioma de la forma "B implies C" entonces "A implies B" se define como nuevo objetivo secundario.

El método de sustitución es el encargado de probar teoremas, el otro par trazan el mapa de subproblemas con la posibilidad de marcar el rumbo hacia la prueba prometida. Los árboles, gráficas conexas sin ciclos (definiciones básicas de teoría de gráficas en los albores del capítulo segundo), sirven para representar el espacio de soluciones (en este caso de subproblemas) y con base a éste se pueden definir estrategias para explorar las distintas ramas del problema. Es decir, existe un vértice raíz (el teorema de entrada A_e), si el método de sustitución no tiene éxito, se genera un conjunto de hijos (subproblemas), vértices adyacentes a la raíz, mediante los métodos de separación y encadenamientos. Puede seguir creciendo el árbol al agregar subproblemas a los subproblemas siempre y cuando los métodos de separación y encadenamientos tengan éxito y no se halla encontrado prueba de algún subproblema (no es un ejercicio de reforestación). La búsqueda en árboles constituye uno de los tópicos clásicos de la IA, la estrategia del recorrido influye drásticamente en los resultados obtenidos. A grandes rasgos la LTM sigue una búsqueda por amplitud, sean $A^1(k),....A^m(k)$ los subproblemas nivel k en el árbol, para los cuales el método de sustitución no tuvo resultado, entonces para i=1 hasta i=m se aplica a $A^i(k)$ los métodos de separación y encadenamientos y para cada uno de los subproblemas nivel k+1 generados por éstos se emplea el método de sustitución. La LTM para cuando (a)se encuentra una prueba para $A^m(n)$ o (b)se acabe el tiempo permitido o (c) se satura la memoria o (d) ya no se hallen más subproblemas. La somera inmersión en el funcionamiento de la LTM sacó a flote a los incisos indecisos (b) y (c). Nuestro instrumento tiene limitantes, profundicemos en sus consecuencias a continuación.

La computadora donde se ejecutó la LTM , la JOHNIAC, pertenece a la era de los dinosaurios de la computación, enormes cerebros de mente estrecha. Además en la etapa mesocomputarozoica el tiempo de uso valía sus segundos en oro debido la escasez y el elevado costo de estos gigantes. Por lo anterior la restricción (b) encuentra una advenediza explicación de su adición, en cuanto que (a), (c) y (d) son condiciones suficientes para la terminación de la LTM[5]. Abordemos y actualicemos (c) con un concomitante y modificado (b). La complejidad temporal y espacial, es decir, la cantidad de recursos de

tiempo y memoria requeridos para la ejecución de un programa, producen obstáculos insoslayables. Aun cuando las computadoras ya no son esas cortesanas con corpachones cuyo costo por segundo de interacción con ellas era proporcional al tamaño de sus cuerpos, el tiempo se yergue todavía como una barrera a franquear. Por ejemplo, el algoritmo del Museo Británico ni siquiera en las actuales supercomputadoras valdría la pena implementarlo, incluso si las dotáramos con memoria y energía infinitas, la humanidad probablemente se extinguiría antes de localizar algunas valiosas y añoradas pruebas en el museo. La escasa productividad del anterior algoritmo radica en su diseño, la falla tiene un carácter teórico. Además los muros poseen una manifestación física, v.gr. la cantidad de memoria de la JOHNIAC y la restricción temporal sólo le permitieron a la LTM encontrar una prueba cuyo árbol de subproblemas constara con cuatro niveles[6]. S&N&S proclamaron su triunfo y vaticinaban mejores resultados en caso de trabajar con computadoras más poderosas: de los 52 teoremas del capítulo segundo de *Principia*, 38 fueron probados por la LTM y de los 14 restantes, según los tres gayos científicos, 12 podrían haber sido probados en caso de contar con la suficiente memoria y tiempo.

En computación la complejidad de un algoritmo normalmente se mide con base al número de operaciones básicas ejecutadas con relación a la longitud de la entrada. Suelen someterse a revisión el mejor, promedio y peor caso. En varias ocasiones la complejidad puede modelarse por una función, así entonces el instrumento ejecutante "computadora" suele cargar de antemano la tara de las limitantes teóricas. En términos prosaicos cuando la función de la complejidad tiene un orden polinomial, i.e. cuando esta acotada por algún polinomio, el algoritmo es aceptable, asequible para nuestros sistemas de cómputo (más información sobre complejidad en el capítulo tercero). Examinemos un problema afín a la LTM y una manera trivial, certera y tediosa de resolverlo: Determinar si la fórmula A del CP es una tautología por medio de tablas de verdad. Este método es un algoritmo pues siempre decide si A es o no es una tautología, sólo se debe verificar el resultado de cada renglón de la tabla (la cual es finita). Sin embargo el simple procedimiento de la construcción de la tabla es al menos de orden exponencial: Si el número de componentes atómicas de A es n, entonces la tabla tendrá 2^n renglones. Cuando el orden de un algoritmo es mayor al polinomial, aunque no puede ser tajantemente expelido de los algoritmos programables en una computadora, si puede trasladar al programa a zonas de bajo y posiblemente nulo rendimiento. En las matemáticas menos interesadas en la viabilidad de sus métodos, las cuales no son pocas, basta con elaborar los algoritmos. En cambio en la computación, sobra decir la importancia de la eficiencia de éstos. La aparición de híbridos trasmina una necesidad de desarrollo en sus progenitores enfocado en las relaciones de sus partes integrantes. Los programas probadores establecieron un vínculo y un área de estudio común entre matemáticas, lógica y ciencias de la computación. El instrumento "computadora" instigó una retroalimentación entre algunas ramas del conocimiento. Impacto de la tecnología en la teoría, tal vez mesurado y canalizado pero innegable. En la siguiente sección, "Hacia la mecanización", se retomará el curso de este párrafo.

S&N&S conscientes de su importancia incorporaron en el reporte sobre la LTM un análisis del esfuerzo, definido a partir del número de operaciones primitivas efectuadas en las

pruebas. En conjunción con sus métodos, la LTM utilizó otros mecanismos heurísticos secundarios[7] para incrementar la posibilidad de hallar una prueba, los cuales fueron evaluados por el esfuerzo provocado por su implementación. Su exégesis incluyó dos facetas, una a posteriori y otra cuasi a priori (supusieron el esfuerzo proporcional al número total de teoremas examinados por la LTM)[8]. S&N&S heurísticamente catalogaron a una heurística como buena, si reducía al esfuerzo real comparándolo con el estimado. Es una valoración intuitiva, atinada y desinhibida: al diablo con la teoría rígida. En el trabajo de S&N&S resalta su consistencia experimental, un poco blanda para los cánones matemáticos. Pragmatismo alambicado. La casualidad encausada da resultados, el lirismo irrumpe en el encomio de su criatura y en la vituperación del anticuado algoritmo para anticuarios realizadas por sus orgullosos progenitores:

... el método que genera subproblemas trabaja "hacia atrás", desde el teorema deseado hacia los axiomas o los teoremas conocidos en lugar de "hacia delante" como lo hace el algoritmo del Museo Británico. Ya que sólo nos interesa demostrar un teorema dentro del infinito número de teoremas existentes, la eficacia de trabajar hacia atrás puede ser análoga a la facilidad con la que una aguja puede averiguar su manera de salir del pajar, comparada con la dificultad padecida por alguien que se propone encontrar a la solitaria aguja en el pajar[9]*...*

El lógico Hao Wang también tiene metáforas en la lengua y arremete:

No hay necesidad alguna de matar un pollo con el cuchillo de un carnicero. Es más, la impresión neta es que Newell-Shaw-Simon ni siquiera mataron al pollo con su cuchillo de carnicero. Ellos no desean usar los algoritmos normales como el método de tablas de verdad... Sostener la superioridad de lo "heurístico" escogiendo un algoritmo particularmente ineficaz [Museo Británico] escasamente parece justo [10]

Retórica vehemente, imprecisa pero contundente. ¿Eficacia en el algoritmo de las tablas de verdad? Si el conjunto de proposiciones a probar está integrado por los primeros 52 teoremas enunciados en el capítulo segundo de *Principia.*, la respuesta es afirmativa. Las fórmulas son cortas, no constan con un número elevado de componentes atómicas. La ventaja de lo "heurístico" sobre lo "algorítmico" no debe ni puede señalarse categóricamente. Si S&N&S lo hicieron, me parece un mero artilugio persuasivo, sus anzuelos estaban preparados para otros fines. El pez gordo era la simulación de la inteligencia *humana* (¿redundancia vana o vanidad redundante?), la inoculación de la simiente de la razón en la computadora. Empresa hierótica, de la orgía del azar brotarán resplandecientes razonamientos deductivos alumbrando al nacimiento de un nuevo simio pensante, pensaban los animales erectos de S&N&S. Sus primeros pasos en los terrenos de la lógica fueron un comienzo espectacular, pues es un tópico *lo suficientemente difícil para la mayoría de los hombres.* Sin embargo para Wang la lógica era su pan nuestro de casi todos los días. Insatisfecho por el desapego de S&N&S hacia sus venerados métodos, contraataca. Tampoco poseerá una posición equilibrada. Le importa un comino la inteligencia artificial. La lógica provee y prevé. Los algoritmos prevalecen. Los métodos de

decisión se envanecen, ¿por qué nos habían abandonado? Wang no enmudece, no calla, no se retuerce. La lógica permanece a su lado. La lógica incrustará su legado. No obstante debe modificarse, debe ceder ante lo animado.

Antes de vanagloriar o repudiar anticipadamente a Wang, ¿restregada contra sus objetivos la LTM a su favor padece o a su temor acierta de/en qué? Frecuentemente los matemáticos al enfrentar a un teorema en abanico despliegan proposiciones (lemas, teoremas, corolarios) de donde se desprende la demostración, acierto. A menudo sacan de la manga lemas y los demuestran en aras de la prueba del teorema en revisión, acierto. Hacen uso de teoremas previamente demostrados, acierto. Escogen teoremas-lemas-corolarios relacionados (en la LTM semejantes morfológicamente) con el teorema a demostrar, acierto. La línea de investigación-emulación de S&N&S sobre las demostraciones ejecutadas por los matemáticos no estaba tan desatinada. Sin embargo la metáfora sobre la angustia de la aguja en pos de escapar del pajar es torpe y poco agraciada. El sentido carece de importancia, sólo subiste y persiste la calidad de la guía, la eficacia del derrotero. Además exuda un bucólico platonismo trasnochado, como si las demostraciones ya existieran y esperaran pacientemente a ser rescatadas de la paja ideal o lucharan descarnadamente por escapar hacia el electrizante cerebro humano. (aunque a su favor tienen que se atuvieron nada más a teoremas ya demostrados) Los resultados positivos los respaldaron, ¿para qué mostrarnos quisquillosos sobre tonterías? Sin embargo S&N&S al tratar de generalizarlos se expusieron a la burla cáustica y al socarrón escarnio, al sonido meloso de la risa asintótica del escepticismo[11]. No es un tratado sobre inteligencia artificial, por lo tanto prudentemente me detengo. Concentrémonos en lo que tiñe a nuestra curiosidad y cadenciosamente nos incumbe. ¿Cómo eran las pruebas de la LTM? Contestémoslo y constatémoslo con un par de ejemplos extremosos (el fácil e inicial y el más demandante con respeto al esfuerzo):

2.01 (p implies not-p) implies not-p P.D.

1. (A or A) implies A (Ax.1.2)
2.(not-A or not-A) implies not-A (sus. de A por not-A)
3.(A implies not-A) implies not-A (remp. de “or” por “implies”)
4.(p implies not-p) implies not-p (sus. de p A por p) QED.

2.17 (not-q implies not-p) implies (p implies q) P.D.

1. A implies not-not-A (teorema 2.12)
2. p implies not-not-p (sus. de A por p)
3. (A implies B) implies [(B implies C) implies (A implies C)] (teorema 2.06)
4. (p implies not-not-p) implies [(not-not-p implies q) implies (p implies q)] (sus. A por p, B por not-not-p, C por por q en 3)
5. (not-not-p implies q) implies (p implies q) (sep. 2,4)
6. (not-A implies B) implies (not-B implies A) (teo. 2.15)
7. (not-q implies not-p) implies (not-not-p implies q) (sus. A por q, B por not-p en 6)
8. (not-q implies not-p) implies (p implies q) (encad. 7,5) QED.

No hay mucho por apostillar. Son demostraciones con todas las de la ley. Para el buen entendedor pocas palabras: son pruebas. Poco deslumbrantes por sí mismas, su generación constituye un logro empero. Esas cosas llamadas computadoras no son sólo ábacos sofisticados. También pueden manipular símbolos, son capaces de escupir pruebas. Sorprendente cómo lo consigue la LTM gracias a sus fantásticas y aventuradas estrategias. Giró la ruleta. ¡Eureka! –no gritó una, sino 38 veces la LTM. Varios la escucharon, entre ellos el lógico en chino Wang y el equipo Gelernter-Hansen-Loveland.

HACIA LA MECANIZACIÓN

Decidible o no decidible, esa era la pregunta. En específico, sea A cualquier fórmula dentro de un sistema formal, ¿existe un método mecánico para determinar si A es o no un teorema de éste?[12] Con relación a lo expuesto en este capítulo, traduzco aunque se pierda información, ¿hay un algoritmo para decidir si A es un teorema? La respuesta es relativa al sistema en cuestión, por ejemplo para el CP es afirmativa. Rápidamente, en el CP todo teorema es una tautología (y viceversa), por lo cual mediante las tablas de verdad podemos determinar si una fórmula es un teorema o no del CP. No obstante existen varias teorías indecidibles, es más, el hilo esencial de las matemáticas, el cálculo de predicados de primer orden es indecidible (ya lo veremos en el capítulo tercero)[13]. Aun arropados por algoritmos de decisión, el inclemente clima de las limitaciones teóricas y físicas azotan inexorablemente a los programas probadores[14]. La sorna invade sin cuartel a la globalidad de la tarea: la misma teoría derivada dentro de un sistema formal se puede construir algorítmicamente mediante el algoritmo del Museo Británico aunque ese recinto no es un sitio de esparcimiento cultural para las tristes criaturas perecederas con sus juguetes mecánicos. Brindador de visiones del conocimiento lúbrico, el Museo es una ventana donde se exhiben impúdicamente verdades sin fin, es el aparador y nosotros siempre afuera, la inmortalidad es la única moneda que acepta (a la muerte como cambio entrega). Ni la lógica ni la computación claudican. Hace falta adaptarse a las circunstancias. Los lógicos no acostumbrados al pragmatismo y los programadores-computólogos no versados en la lógica deben modificar sus patrones de investigación, por el bien común, deben cooperar. Wang sugirió en una serie de seminarios sobre las aplicaciones no numéricas de las computadoras y la teoría de los sistemas formales realizados en 1961 a los interesados entre otras cosas lo siguiente:

(a) He examinado al teóricamente indecidible dominio del cálculo de predicados y logré hacer que una IBM 704 demostrara todos los teoremas (más de 350) de Principia Mathematica en este dominio en menos de 9 minutos; esto sugiere que normalmente no usamos cabalmente el poder de los métodos matemáticos y que no debemos abstenernos de atacar una área a causa de las estimaciones abstractas pesimistas de los casos más difíciles en la región.

(b) Es esencial tener cuidado en el diseño teórico de los procedimientos y tener una cierta cantidad de sofisticación en la lógica matemática es indispensable, porque la mayoría de los métodos existentes no es inmediatamente aplicable en las máquinas
(c) En particular, uno tiene a menudo que reformular los métodos disponibles o incluso inventar nuevos métodos
...
(e) Aunque se puede hacer más con máquinas más poderosas, el diseño y la elección de los métodos es más crucial por lo menos en la fase presente porque estamos lejos de haber empleado al máximo el poder de cómputo de una IBM 704 o 7090
(d) Desconfía de la suerte, por ejemplo no uses métodos oscuros con la esperanza que algo maravilloso podría pasar ya que no sabemos lo que pasará; la probabilidad de consecuencias indeseables es más grande. [15]

Se viró de rumbo hacia la mecanización de las matemáticas, la conjunción de la lógica-computación-matemáticas. La primera escala seria sería la llegada y la conquista del cálculo de predicados. Casi en su totalidad las matemáticas pueden escribirse en lenguajes de primer orden. No basta con dominar las técnicas de navegación lógica, hace falta conocer las capacidades del *barcordenador.* Si el mar de la suerte es propicio, la empresa puede ir por buena ruta. Sin embargo Wang propone confeccionar métodos más robustos sin olvidar las características del navío para sortear las dificultades y depender menos de los favores del océano del azar:

Una buena organización de las propiedades elementales más una formulación exacta de los métodos familiares de búsqueda de pruebas probablemente rendiría en cada disciplina algo similar a los principios obtenidos de lo que se llama a menudo el acercamiento "heurístico", [16]

No sólo hacen piruetas en el aire sus recomendaciones, las aterriza. Para la teoría de números, el probador debe incorporar la inducción matemática; para la teoría de conjuntos, al axioma de comprensión, permitiendo definir conjuntos a partir de propiedades. Tampoco emprende el vuelo hacia las nubes tóxicas de la ostentación. Para finales de 1958, los programas de Wang en menos de 30 segundos demostraron todos los teoremas probados por la LTM. Abismal diferencia con S&N&S, no enarbola la bandera de la “inteligencia” ni nada por el estilo. Para Wang, las computadoras son *talacheros* persistentes (*plodders*). Sus palabras exhuman mesura:

En las áreas más avanzadas de la matemática, el éxito es poco probable si buscamos hacer que la máquina imite al hombre completamente. [17]

Aduce la dificultad de reproducir en las máquinas algunas facultades mentales como la intuición o la experiencia. Son temas espinosos, sobretodo el de la intuición. En su lugar se pueden obtener ventajas al enfocarse en las propiedades de los *talacheros* binarios, por su claridad meridional continúo citando a Wang:

La incapacidad humana para manipular con precisión cualquier masa grande de detalles causa una limitación intrínseca en el tipo de cosas que se hacen en las matemáticas y en la manera como se hacen. La superioridad de las máquinas en este respecto, mientras sigan exhaustivamente los caminos bosquejados por los especialistas, podría conducir a sorprendentes nuevos resultados haciendo muchos nuevos giros que el hombre no está acostumbrado a tomar.[18]

Su profecía se cumplirá cabalmente una década y media después, para ser precisos en 1976, sin embargo quien traerá la buena nueva no provendrá de las tierras santas de la lógica. Las máquinas abrirán nuevas brechas hacia las demostraciones, irremediablemente los senderos recién abiertos despiertan dudas, provocan desconfianza. Aunque la demostración computarizada de la Conjetura de los Cuatro Colores hecha por un equipo sui géneris encabezado por Haken y aparecida en 1976 (capítulo segundo) es diametralmente distinta a las pruebas mecánicas (derivaciones sintácticas guarnecidas de la ambigüedad bajo el techo de un sistema formal) en este capítulo abordadas, disfrutan de problemas comunes. El instrumento ejecutor "computadora" desde su encarnación concreta hasta la dimensión abstracta del programa será puesto en tela de juicio. Inocentemente Wang le da la bienvenida a las computadoras en la actividad matemática, las recibe prediciendo su rol cada vez más protagónico. Prematuro ponerse puntilloso sobre el portador del futuro, aun el inminente. Para Wang además de la investigación, las matemáticas mecánicas tendrán otro par de novedosos puntos de injerencia: el pedagógico y el filosófico.Leamos entre líneas:

Sin embargo, los problemas interesantes de las matemáticas mecánicas sólo se presentan cuando nos ocupamos de las tareas que requieren a un agente activo para dar respuestas, consejos y críticas. Siendo el caso más simple la corrección de ejercicios y de exámenes. [19]

La computadora puede asistir a los estudiantes (en general a los matemáticos) a revisar sus ejercicios, a corregir sus demostraciones[20]. Este uso fue explotado y explorado por N.G. de Brujin a finales de los 60's con AUTOMATH, un lenguaje para escribir demostraciones diseñado para poder ser verificadas por medio de la computadora[21]. Han habido otros intentos siguiendo esta línea, en general de bajo impacto en la comunidad aunque de presencia cada vez menos invisible (Mizar será discutido desde el capítulo tercero). Para aumentar su poder de convocatoria y la dimensión de su influencia se requiere de un cambio de hábitos, se debe incorporar a la computadora como una herramienta deductiva cotidiana; es más, haría falta una reforma cultural para fomentar la adopción de lenguajes técnicos, más propios de lógicos y computólogos. Lenguajes secos, hoscos, para los no diletantes marchitos. Cuando las caranto ñas y melifluas proposiciones se frustran la imposición en seducción violenta se consolida. Pero, ¿qué ofrece a cambio? ¿Qué ganan los matemáticos con aprender y utilizar un idioma aunque no ajeno para la mayoría ajado? ¿La garantía del romeo binario de la plena validación de las pruebas o la devolución total de sus dolores de cabeza? ¿Acaso los matemáticos van a confiar más en una computadora que en sí mismos y en sus congéneres? Wang convenientemente no transfiere las interrogantes al instrumento (discutidas con mayor amplitud en el capítulo cuarto), las atiende

perspicazmente en el ámbito lógico. Arrastradas hacia la filosofía de las matemáticas, las arpa y sutilmente vibran los ecos de respuestas inducidas:

Cuando entendamos con mayor exactitud al rango de las matemáticas mecánicas, conseguiremos una vista más clara de la relación entre la complejidad y la dificultad conceptual en las matemáticas, ya que probablemente desearíamos estipular que los pedazos mecanizables, aunque sean muy complejos, son conceptualmente sencillos. Cuando las demostraciones alternativas son totalmente mecanizables, también obtendríamos una medida cuantitativa de la simplicidad de las demostraciones matemáticas, complementando nuestro vago pero más rico concepto de la simplicidad. Con el poder creciente de la formalización y la mecanización nos libramos de los detalles tediosos y podemos inspeccionar más eficazmente el contenido y núcleo conceptual de una demostración matemática [22]

La ha dado al clavo. Aunque, ¿qué otra cosa se podría esperar si utiliza martillos? Wang conoce a fondo los usos y costumbres de la lógica matemática, está enterado de las causas subyacentes a los estudios de los sistemas formales. Acertadamente aboga por la "simplicidad" y no por la "certeza". Wang está consciente de la aspiración de asentar a las matemáticas por medio de los sistemas formales, para desterrar sin compasión a los disfraces de lo incierto. El necrófilo sueño del rigor es inhumar en el olvido a la vaguedad de lo intuitivo (como abono será bien recibido). También sabe de todos los contratiempos de la faena, de su tediosa y poco prometedora ejecución. Ya asistió a la obra "Esperando a Gödel", a diferencia de Godot, si llegó en los treinta y trajo la salvación con la condena de la empresa (el primer Teorema de la Incompletud será mencionado en el capítulo tercero, el segundo igual de virulento desgraciadamente no será reseñado). Reventaron las ínfulas de las grandes letras (Consistencia, Verdad) pero el trabajo en el frente sigue en pie. En escena aparecen los esclavos digitales, poderosos, incansables; poseen defectos y carencias pero son más aptos que sus amos para el trabajo requerido. Al fin y al cabo deben realizar operaciones simples, comprensibles para casi cualquier humano, aunque en cantidades exorbitantes, en un número apabullante para los individuos. Inquieta cómo termina el párrafo citado, pues ¿cuál es el contenido y núcleo conceptual de una demostración matemática? Al menos para Wang la parte neurálgica no reside nada más en la sintaxis, en los detalles técnicos y tediosos de las pruebas. Postergo la discusión, por el momento varios elementos no han sido convocados por lo que se desenvolvería de manera precoz, talvez procaz. Mejor evito dejar sin atender otro pendiente, ¿qué beneficios otorga a los practicantes de las matemáticas el manejo de los lenguajes estudiados por la lógica y la computación? ¿El beneficio recae en la simplicidad de las pruebas? ¿Acaso es más complicada una demostración apoyada en una figura de un teorema básico de la geometría euclidiana escrita en unas cuantas líneas que su correspondiente prueba formal normalmente desplegada en muchas más de ellas? Wang es hábil, no revuelve a la complejidad con la dificultad conceptual. Aunque la prueba sea extensa, sus componentes gracias a la formalización ya han sido diligentemente identificados y descritos. Por ejemplo el algoritmo enseñado desde la primaria para multiplicar enteros es conceptualmente llano (basta con memorizar las tablas de multiplicación). Sólo como castigo o como acto extremo

de relajación y meditación un hombre multiplicaría 100 números de 100 dígitos cada uno. Una computadora con la resolución necesaria no tendría mayor problema en hacerlo. La ejecución en cierto sentido es irrelevante, lo importante es comprender a los métodos, discernir sobre los mecanismos. La lógica otorga un entrenamiento adecuado para hacerlo. La justificación posee connotaciones pedagógicas. Ordenar clarifica. Eso no implica la derogación de las demostraciones sin la forma de pruebas. Jamás. Si son traducibles a pruebas es un indicador que vamos en el camino correcto, nada más. Exigir su formalización es pregonar para que lo pavimentemos y no para suspender la construcción de nuevas vías. Guardamos nuestro tesoro, nuestro conocimiento en una bóveda reforzada por lógica, ninguna bóveda es inviolable, empero.

En ningún momento descuida Wang el aspecto de la complejidad, pregona por atenernos a lo accesible para una computadora. Stephen Cook, discípulo de Wang, aprendió bien la lección. En 1971 de su fusca manga sacará un teorema de consecuencias a primeras luces desalentadoras, relacionado con determinar si una fórmula del CP es una tautología. Este famoso teorema al cual Cook le presta su nombre será degustado en el tercer capítulo. Para concluir con esta sección trascribo un pronóstico de Wang:

Mientras el objetivo ulterior es usar a las máquinas para ayudar a la investigación matemática con la asistencia de la lógica, también pueden usarse para ayudar a nuestra investigación teórica en la lógica en la fase presente. Las computadoras pueden tener el buen uso de la producción masiva de ejemplos concretos, los cuales constantemente los necesitamos como medios para clarificar nuestros conceptos y así apresurar los resultados teóricos generales. [23]

Cuando Wang trabajó con la mecanización de procedimientos lógicos, al implementarlos los resultados obtenidos fueron insistentemente inesperados. Por este motivo Wang visualizó las potencialidades heurísticas de la máquina y pregonó por experimentar con las computadoras. La computadora desenrolla sus tentáculos con sus ventosas de unos y ceros; no sólo prueba, asiste a los matemáticos brindando ejemplos, exhibe situaciones no previstas (el caso de Lorenz, la computadora y la teoría del caos es un clásico[24]). El artefacto binario sucita desarrollo.

POSTDATA: THE GEOMETRY MACHINE

H. Gelertner, J.R. Hansen y D. Loveland (G&H&L) acuciados por la LTM acogen al enfoque heurístico en su Máquina Geométrica (MG), elaborada para probar proposiciones de la geometría euclidiana. Al haberse fijado como meta teoremas cuyo nivel es el de la preparatoria, el sistema axiomático de la MG es elemental y apegado a los libros de texto, ergo está desprovisto de la sofisticación de las axiomatizaciones de Tarski o Hilbert. Su mayor innovación radica en la incorporación de la interpretación semántica, pues:

En el caso de geometría euclidiana, la interpretación semántica es tan útil que virtualmente nadie intentaría probar un teorema en ese sistema sin dibujar un diagrama primero; si no físicamente, entonces en el ojo de la mente.[25]

La GM está organizada en tres submáquinas: la sintáctica y la diagramática coordinadas por la heurística. La componente sintáctica es un vástago de la LTM, hereda sus métodos de separación y sustitución; la diagramática contiene una representación en coordenadas cartesianas de la proposición a probar además de contar con una serie de rutinas para producir una descripción cualitativa del diagrama. Por último la función primaria de la componente heurística consiste en comparar las subproposiciones generadas por la submáquina sintáctica con su interpretación en el diagrama, desechando aquellas no soportadas por el modelo. La submáquina heurística también controla los métodos de encadenamientos y reconoce simetrías sintácticas[26].

La semántica entró en la casa. La geometría le abrió la puerta, albergó al intruso. Un momento, ¿semántica? Al reconocerla en la GM, ¿acaso argüimos a favor de la inteligencia artificial? La investigación de la GM compartía con la de la LTM su objetivo titánico y brumoso:

Ellos [los psicólogos] todavía no caracterizan satisfactoriamente a dicha conducta [inteligente] en los humanos y raramente han considerado el concepto abstracto de inteligencia con independencia de su agente. En el último análisis, las personas ocasionalmente hacen cosas que al ser observadas pueden ser descritas en el mejor de los casos como inteligentes, sean cual sean las connotaciones vagas del apelativo. En general, estas tareas involucran procesos complejos de decisión en un potencialmente infinito e incontrolable ambiente. Debemos estar muy complacidos porque nuestra máquina duplica esta clase de conducta, cualquiera que sea la etiqueta pegada a ésta.[27]

Los "ismos" (conductismo, funcionalismo, etc,) estallan en grito de guerra, la presea en disputa es la inteligencia. Me circunscribo a los fines menos sublimes aunque igual de nebulosos de este escrito, sin desperdiciar la oportunidad de aprovechar la opinión de G&H&L. El modelo de la caja negra sirve para recolectar indicadores sobre la simulación de algunas cualidades cognoscitivas. La semántica involucra el establecimiento de relaciones entre lenguajes, entre los palabras del lenguaje, entre las palabras empalmadas en las sentencias de éste, entre las sentencias, entre muchos entres. Una instancia del significado (una palabra puede poseer múltiples significados, distintos significados de sus significados, etc.) a veces desciende hasta la experiencia sensible, hacia lenguajes de bajo nivel (como el visual); en otras a partir del mismo lenguaje queda recluido. En los "objetos matemáticos", aunque asistido por diagramas, figuras o configuraciones espaciales, su significado se esculpe de la materia prima del lenguaje natural, depurándolo hacia un lenguaje lógico-matemático. Independientemente de si el lenguaje construye o representa a los objetos, su significado reside en la red relacional torpemente bosquejada. Los procesos para generar / reconocer las relaciones están encerrados en la caja negra, bajo este enfoque el cómo es ignorado, es relegado. Importan los resultados, las relaciones. La GM a su manera las construye, la GM está provista de semántica.

La semántica ayuda a los matemáticos en sus derivaciones deductivas, es más, bajo la terminología de Wang, pertenece al contenido y núcleo de una demostración. En la GM se emplea como guía, tiene una función heurística. La computadora, aunque de forma confinada y sobresimplificada, tiene la capacidad de involucrar a la semántica en la manipulación de cadenas de caracteres (fórmulas). Los probadores mecánicos de teoremas gracias a la GM, adquirieron una nueva dimensión. Intrigante, aunque el entremés del tercer capítulo debe darse por concluido.

NOTAS

(1) Sune Calson, Discurso de presentación del premio , *Nobel Lectures, Economics 1969-1980*, Editor Assar Lindbeck, World Scientific Publishing Co., Singapore, 1992

"the Royal Academy of Sciences has decided to award you the 1978 Alfred Nobel Memorial Prize in Economic Sciences for your pioneering research into the decision-making process in economic organization."

(2) Herbert Simon, *Administrative Behavior*, Macmillan, 1947

" It is impossible for the bahavior of a single, isolated individual to reach any high degree of rationality. The number of alternatives he must explore is so great, the information he would need to evaluate them so vast that even an approximation to objetctive rationality is hard to conceive" pág. 79

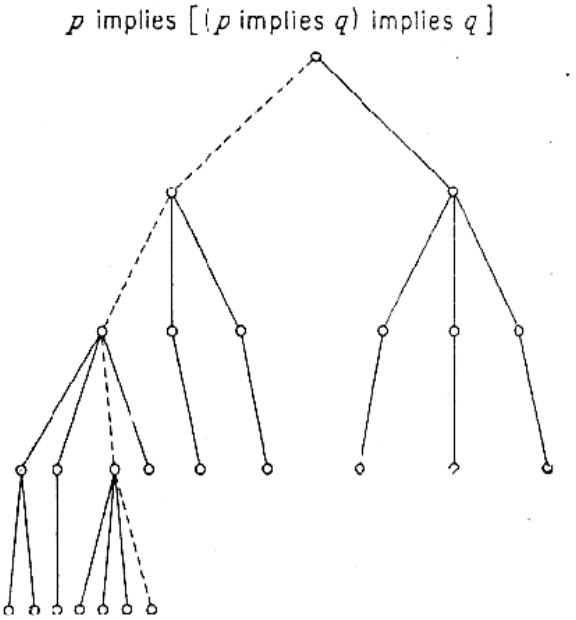
(3)Newell, Shaw, Simon,. *Empirical explorations with the logic theory machine: a case study in heuristics,* en Feigenbaum, EA y Feldman J.(Eds.), *Computers and Thought,* McGraw-Hill, 1963, págs. 109 a 133

The research reported here is aimed at understanding the complex processes (heuristics) that are effective in problem-solving. Hence, we are not interested in methods that guarantee solutions, but which require vast amounts of computation. Rather, we wish to understand how a mathematician, for example, is able to prove a theorem even though he does not know when he starts how, or if he is going to succeed pág. 109

(4) Escindo a la redundancia, de las puntas la extiendo en linea recta , directa, entiendo sus puntos. La fórmula A es un teorema en un sistema formal (en este caso CP) si existe una prueba de A. Por lo cual "todo teorema de CP tiene una prueba..." a primera instancia dice tanto como "A es A". Aclaremos, una fórmula en CP es verdadera (tautología) si es satisfecha bajo cualquier asignación (verdadero,falso) de sus componentes atómicas. Por ejemplo, "A implies A" es una tautología.No importa si A es verdadera o falsa, "A implies A" es verdadera. En el CP toda tautología es un teorema bajo una elección cuidadosa del conjunto de axiomas, a éste se le denomina base. Por lo tanto (1.2)...(1.6) es una base de CP, para mayor información revísese *Principia MAthematica.*

(5) Saber si un programa para en particular puede ser bastante dificultoso, en general imposible. Apelo a la ingenua transparencia irradiada por la displicencia a las regiones turbias teóricas no aptas para este escrito.

(6) Teorema 2.27 :

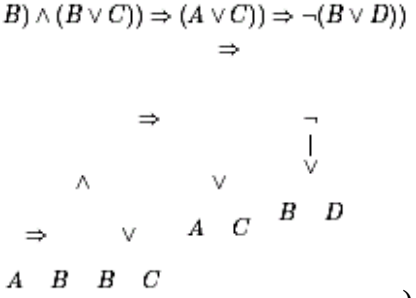


(7) Los mecanismos se dividen en dos clases:

- Apareamiento: Para aparear A con B (donde A, B son fórmulas moleculares del CP) se deben cumplir y seguir los siguientes pasos: (1) los conectivos principales de A y B son iguales (2) las variables a la izquierda del conectivo principal son las mismas (3) los conectivos en las subfórmulas en paréntesis a la derecha del conectivo principal son diferentes (4) se modifica (regla de reemplazo) la subfórmula a la derecha del conectivo principal de A para que tenga el mismo conectivo principal de la subfórmula a la derecha del conectivo principal de B. Si la fórmula resultante es exactamente B, A y B están felizmente apareados. Este mecanismo es el corazón del método de sustitución, además también se usa en los métodos de encadenamientos y separación. Si mi meta es probar B, el método de separación requiere expresiones del tipo "A implies B". Se amplia el horizonte de búsqueda y las probabilidades de éxito si nos conformamos con hallar fórmulas del estilo "A implies C", donde C se puede aparear con B.

- Similitud: Para optimizar los intentos de apareamientos se hace un preprocesamiento entre las fórmulas a aparear (A con B), se verifica si las partes izquierdas y derechas de A,B son iguales correspondientemente con respecto a : (1) el número de variables distintas (2) el número de lugares para las variables (3) el máximo número de niveles del conectivo principal hacia cualquier variable (El nivel se deriva del árbol de formación de una fórmula. Ejemplo de un árbol de formación con nivel cuatro:

$((((A \Rightarrow B) \wedge (B \vee C)) \Rightarrow (A \vee C)) \Rightarrow \neg(B \vee D))$



)

En caso de satisfacerse (1),(2) y (3) A y B son similares. S&N&S ajustaron el criterio a la LTM y sólo exigían (2).

(8)He aquí la tabla con el mecanismo de similitud y excluyendo a los teoremas probados directamente por sustitución:

Total primitives, thousands		
Theorem	Actual	Estimate
2.06	3.2	0.8
2.07	4.3	4.4
2.08	3.5	3.3
2.11	2.2	2.2
2.13	24.5	24.6
2.14	3.3	3.2
2.15	15.8	13.6
2.18	34.1	35.8
2.25	11.1	11.5

Los esfuerzos estimados también contemplaron diferentes ponderaciones, conforme al método aplicado al teorema revisado (v.gr separación = 3 sustituciones).

(9) *...the method that generates subproblems work "backward", from the desired theorem to axioms or known theorems rather than "forward" as did the British Museum algorithm. Since there is only one theorem to be proved, but a number of known true theorems, the efficacy of working backward may be analogous to the ease with which a needle can find its way out of a haystack, compared with the difficulty of someone finding the lone needle in the haystack...* pág, 118-119 de (3)

(10) Wang Hao, *Toward Mechanical Mathematics*, en *IBM Journal of Research and Development 4* , 1960 págs. 2-21.

There is no need to kill a chicken with a butcher's knife. Yet the net impression is that Newell-Shaw-Simon failed even to kill the chicken with their butcher´s knife. They do not wish to use standard algorithms such as the method of truth tables...[But] to argue the

superiority of "heuristic" by choosing a particularly inefficient algorithm ["British Museum"] seems hardly just ... pág.4.

(11) Se ridiculiza(aba) a los "duros" , i.e. a quienes en mayor o menor medida reducen a la inteligencia a sistemas lógicos, con situaciones estilo "cruzar la calle sino vienen carros". Si la inteligencia implica una secuencia correcta de inferencias, ¿debo "demostrar la ausencia de coches" para cruzar la calle sin temor a convertirme en una estampa del pavimento? ¿Debo transformarme en una estatua de la acera concentrada en las engorrosas deducciones?

(12) Por si alguien no leyó la advertencia inicial – Sistema formal := lenguaje (variables, conectivos, cuantificadores, letras relacionales, signos funcionales, constantes, signos de puntuación) + reglas para construir fórmulas + reglas para construir términos + reglas de inferencia + definición de prueba. De la exclusión-omisión-modificación de los componentes de los sumandos germinan los distintos tipos (v.gr. en CP no hay cuantificadores) .
La decidibilidad es un tema clave para la lógica, tiene diversas vertientes, por ejemplo, el problema de la satisfacibilidad (¿dada una fórmula, existe un(a) método efectivamente calculable- función recursiva para determinar si existe una asignación que la satisfaga) o su dual, el problema de la validez (¿dada una fórmula, existe un mecanismo para decidir si bajo cualquier interpretación es verdadera?)

(13) Que no cunda el pánico, si L es un lenguaje de primer orden (i.e. la cuantificación abarca sólo un nivel , v.gr. del i.e. cuantifico sobre los números pero no sobre los conjuntos de números) , usualmente se estudia sobre la decidibilidad de los $T \subset L$. Se restringe el dominio de decidibilidad, por ejemplo la Teoría Elemental de la Geometría Euclideana o la Teoría de los Grupos Abelianos son decidibles.

(14) En 1929 Presburger diseñó un algoritmo para determinar si cualquier enunciado en la teoría de la adición en la aritmética de los enteros es o no verdadero. En 1954 Martin Davis lo programó en una computadora de tubos al vacío (spooky) en Princeton, sin embargo debido a la complejidad del algoritmo (no polinomial) la fortuna no le sonrió. Su mayor triunfo fue demostrar que la suma de dos pares es par y es ser considerado por algunos (como él mismo) el padre de los probadores mecanizados de teoremas (*The Early history of Automated Deduction*, www.cs.nyu.edu/cs/faculty/davism/early.ps)

(15) Algunas intervenciones fueron recopiladas en el libro: *Computer programming and formal systems*, editado por P. Braffort y D. Hirschberg. Amsterdam : North-holland, 1963

La ponencia de Wang se tituló *Mechanical Mathematics and inferential analysis*, encontrada en las páginas 1-20

(a) I have examined the theoretically undecidable domain of the predicate calculus and managed to make an IBM 704 prove all theorems (over 350) of Principia Mathematica in

this domain in less than 9 minutes; this suggests that we usually do not use the full power of strong mathematical methods and should not be prevented from trying to handle an area on account of pessimistic abstract estimates of the more difficult cases in the region.
(b) Care in the theoretical design of the procedures is essential and a certain amount of sophistication in mathematical logic is indispensable, because most existing methods are not immediately applicable on machines
(c) In particular, one often has to reformulate available methods or even invent fundamentally new ones
...
(e) While more can be done with larger machines, the design and choice of methods is at least at the present stage, more crucial because we are far from having made full use of an IBM 704 or 7090 yet.
(d) Distrust luck and do not, for example, use obscure methods with the hope that something wonderful might happen since we do not know what will happen; the chances of undesirable consequences are much bigger. págs.5-6

(16) *A good organization of elementary properties plus an exact formulation of familiar methods of trying to find a proof would presumably yield in each discipline something similar to the principles obtained from what is often called the "heuristic approach.*
pág.9 de (15)

(17) *In the more advanced areas of mathematics, we are not likely to succeed in making the machine imitate the man entirely,* pág.2 de (15)

(18) *The human inability to command precisely any great mass of details sets an intrinsic limitation on the kind of thing that is done in mathematics and the manner in which it is done. The superiority of machines in this respect indicates that machines, while following the broad outline paths drawn up by people, might yield surprising new results by making many new turns which man is not accustomed to taking.* pág.2 de (15)

(19) *However, interesting problems of mechanical mathematics arise only when we come to tasks which call for an active agent to give answers, advices, and criticisms. The simplest being the correction of exercises and examination papers.* pág.3 de (15)

(20) La cita (19) continua:*Psychologically, the pupil has many reasons for preferring a patient machine teacher when the problem is, as in the majority of situations, a matter of drill rather than inspiration*. Me imagino a uno que otro pedagogo pegando un grito en el cielo y a toda una legión de maestros suspirando por ese maravilloso asistente, dispensador de la labor sucia y rutinaria de la enseñanza, la parte más apta para ser mecanizada.

(21) Aquí se puede descargar una versión moderna: http://www.cs.ru.nl/~freek/aut/

(22) *As we understand more fully the range of mechanical mathematics, we get a clearer view of the relation between complexity and conceptual difficulty in mathematics, since we*

would probably wish to say that mechanizable pieces, even highly complex, are conceptually easy. When alternative proofs are fully mechanizable, we obtain also a quantitative measure of the simplicity of mathematical proofs, to supplement our vaguer but richer intuitive concept of simplicity. With the increasing power to formalize and mechanize we are freed from tedious details and can more effectively survey the content and conceptual core of a mathematical proof.
pág.4 de (15)

(23) *While the ulterior aim is to use machines to aid mathematical research with the assistance of logic, machines can also be used to aid our theoretical research in logic at the present stage. Computers can be put to good use in the quantify production of concrete examples, which we constantly need as a means of clarifying our concepts and so expediting general theoretical results.* pág.7 de (15)

(24) La bibliografía sobre el tema es numerosa, un libro recomendable para quienes busquen una lectura ligera de amplio panorama es *¿Does God plays dice?* escrito por I. Stewart.

(25) Gelertner, Hansen, Loveland, *A geometry-theorem proving machine,* en *Feigenbaum, EA y Feldman J.(Eds.), Computers and Thought, McGraw-Hill, 1963,* págs.135-152

In the case of Euclidean geometry, the semantic interpretation is so useful that virtually no one would attempt the proof of a theorem in that system without first drawing a diagram; if not physically, then in the mind's eye. pág.138

(26) *The latter function produces behavior equivalent to that of the human mathematician who, when A and B are syntactically symmetric, and both must be established, will merely prove A, and say "Similarly, B"*[18].

(27) Galertner, Hansen, Loveland. *Empirical explorations of the geometry-theorem proving machine,* en *Feigenbaum, EA y Feldman J.(Eds.), Computers and Thought, McGraw-Hill, 1963* , págs.153-163

They [the psychologists] have yet to satisfactorily characterize such behavior [intelligent] in humans, and have rarely considered the abstract concept of intelligence independent of its agent. In the final analysis, people are occasionally observed to do things that may best be described as intelligent, however vague the connotations of the word. These are, in general, tasks involving highly complex decision processes in a potentially infinite and uncontrollable environment. We should be most happy to have our machine duplicate this kind of behavior, whatever label is affixed to it. pág.154

CAPÍTULO II: LA PINCELADA DE LOS CUATRO COLORES

El capítulo arranca con un breve repaso de los conceptos de la teoría de gráficas requeridos para leer sin mayor problema las demostraciones desplegadas en esta parte de la tesis. La primera de ellas es la "demostración" actualizada de Kempe de la conjetura de los cuatro colores, la segunda es una demostración computarizada de dicho "teorema" elaborada por Haken, Appel, Koch y su asistontos digitales. La irrupción de las computadoras en la labor demostrativa acarreó una serie de críticas (positivas y negativas) sobre el papel del ayudante binario y disquisiciones de especial interés acerca del concepto de demostración en las matemáticas. La última sección del presente capítulo es una pequeña red donde se recolectan algunos de los comentarios filosóficos provocados por la llamativa demostración, llamativa por lo famoso de la conjetura y lo peculiar de su ejecución. En la malla además de respetables atunes han quedado atrapados algunos delfines cuestionables provenientes de mis reflexiones. Enredada está la caracterización de tres ojos de las demostraciones aleteada por Tymoczko. Según él las demostraciones comunes en las matemáticas satisfacen tres propiedades esenciales: son convincentes, son revisables y son formalizables. La demostración computarizada expuesta en este capítulo tiene serios conflictos con la segunda propiedad, por lo que esa demostración es un nuevo animal matemático o la caracterización necesita modificarse (la novedad o falta de ella radica en el grado de adherencia a la tradición filosófica matemática). Tymoczko se inclina más por la primera opción en pos de ampliar la variedad de los objetos matemáticos buscando derruir la hegemonía de lo a priori en la verdad de los teoremas. En cambio yo transitaré más por la segunda, aunque la demostración de Haken, Appel, Koch sus asistentes digitales impele a deambular al menos un rato por la primer vereda.

NOCIONES PREVIAS

Para aquellos no versados sobre los conceptos básicos de teoría de gráficas empleados en el capítulo y no encuentren satisfactorio este breve repaso les sugiero leer *Graph theory with applications*[1], fuente de las siguientes párrafos.

- Una gráfica G es una tripleta ordenada $<V(G), A(G), \psi_G>$ constituida por un conjunto no vacío V(G) de vértices, un conjunto A(G) de aristas disjunto de V(G) y una función de incidencia ψ_G que asocia cada arista de G con una pareja no ordenada de vértices de G (no necesariamente distintos). Si e es una arista y u,v un par de vértices tal que $\psi_G(e)=uv$ entonces e se dice que une a u y v; los vértices u.v son llamados los extremos de e. Los extremos de una arista se dice que son incidentes con la arista y viceversa. Dos vértices que son incidentes con una arista común se denominan adyacentes, así como también dos aristas que inciden un mismo vértice. Una arista con extremos iguales se le llama lazo, cuando son distintos enlace. Una gráfica es finita si sus conjuntos de vértices y aristas son finitos. Una gráfica es simple si no contiene lazos ni enlaces múltiples (i.e. varias aristas que unan al mismo par de vértices). A partir de aquí "gráfica" equivale a "gráfica simple y finita".

Ejemplo:

$G^*=<V(G), A(G), \psi_G>$
donde $V(G^*)=\{v_1, v_2, v_3, v_4\}$
$A(G^*)= \{e_1, e_2, e_3, e_4\}$

ψ_{G^*} está definida de la siguiente forma:
$\psi_{G^*}(e_1)=v_1v_2$, $\psi_{G^*}(e_2)=v_2v_3$, $\psi_{G^*}(e_3)=v_3v_4$, $\psi_{G^*}(e_4)=v_4v_1$

El nombre "gráfica" de estos objetos matemáticos indica a su directa (e incluso intuitiva) interpretación visual en forma de diagramas. Es más, gracias a su representación gráfica podemos comprender muchas de sus propiedades.

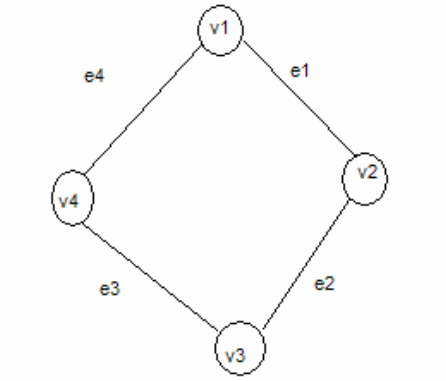


Diagrama de la gráfica G*

- Una gráfica H es subgráfica de G ($H \subseteq G$) si $V(H) \subseteq V(G)$, $A(H) \subseteq A(G)$ y $\psi_H = \psi_G|_{A(H)}$
  Ejemplos: Para toda G gráfica $G \subseteq G$;
  si $V(H^*) = \{v_1, v_2\}$, $A(H^*) = \{e_1\}$, $\psi_{H^*}(e_1)=\psi_{G^*}(e_1)=v_1v_2$ entonces $H^* \subseteq G^*$

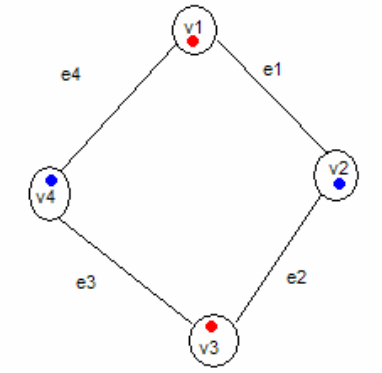
- El grado de un vértice en G (δ_G) es el número de aristas distintas que inciden en él.
  Ejemplo: $\delta_{G^*}(v_1)=2$, $\delta_{G^*}(v_2)=2$, $\delta_{G^*}(v_3)=2$, $\delta_{G^*}(v_4)=2$

- Un camino en G es una secuencia finita (no nula) de vértices $C=\{v_0, v_1,...,v_{k-1},v_k\}$ en donde para todo $1=i=k$ v_{i-1} es adyacente en G a v_i. Si todos los vértices son distintos C es una trayectoria. Si únicamente el primer y último vértices son iguales C es un ciclo.
  Ej: $T=\{v_1,v_2, v_3 , v_4\}$ es una trayectoria en G*, $C=\{v_1,v_2, v_3 , v_4, v_1\}$ es un ciclo de G*.

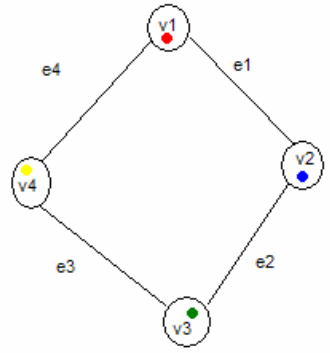
- G es una gráfica conexa si para cualquier par de vértices u,v pertenecientes a V(G) existe un camino que los une (a éstos se les bautiza uv-caminos).

- Una k-coloración de los vértices de G es una asignación de k colores a los vértices de G; la coloración es propia si no existen vértices adyacentes con el mismo color, en caso de ser posible dicha coloración a la gráfica G se le llama k-coloreable. Una gráfica G es k-cromática si k es el mínimo número de colores distintos requeridos para colorearla propiamente. Ejemplos:

G* es 2-cromática



G* es 4-coloreable

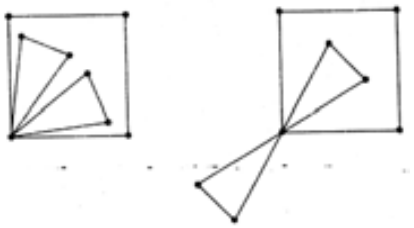


- Una gráfica es plana si puede dibujarse en un plano de tal forma que cada par de aristas a lo sumo intersequen en un vértice. Una gráfica plana define a una partición del plano en un cierto número de regiones conexas, las cerraduras de dichas regiones son llamadas las caras de G. La gráfica G* tiene dos caras, la rodeada por el ciclo C y la exterior de tal ciclo. Cada gráfica plana posee exactamente una cara sin aristas que la delimiten, ésta es llamada la cara exterior.

- Una gráfica es triangular si es plana y a cada una de sus caras le pertenecen tres aristas. G* puede convertirse en una gráfica triangular si se le agregan las aristas con extremos v_2v_4 trazada en la cara interior y con extremos v_1v_3 dibujada en la cara exterior. las indicaciones donde trazar las aristas obedecen a las precauciones para no violar la planaridad de G.

Los mapas planares, v.gr. los planisferios vendidos en las papelerías, pueden delinearse por medio del lenguaje de la teoría de gráficas. El libro escrito por Saaty y Kainen, *The Four Color Problem*[2], es el origen del par de párrafos por advenir y dicho sea de paso es fuente de las primeros apartados de este capítulo; confieso de una vez las no tan ominosas omisiones de algunos detalles técnicos, por lo cual a quien le provoque interés la lectura de las siguientes tres secciones, revisarlo es una opción bastante recomendable.

- Un mapa (mapa planar) consiste en una gráfica plana conexa G y en la manera como está dibujada en el plano. A la gráfica G se le llama la gráfica subyacente del mapa (G=U(M)). Las caras de G corresponden a los países. Por ejemplo la gráfica que consiste de un cuadrado y dos triángulos que intersecan en un vértices pueden generar los siguientes dos mapas distintos:



- A todo mapa M se le puede asociar una gráfica (D(M)) que representa con vértices y aristas las relaciones de vecindad entre los países. A cada país p_i se le asocia un vértice v_i. Si p_1 comparte una arista con p_2 entonces v_1 es adyacente en D(M) a v_2. D(M) es la gráfica dual del mapa. Para evitar distracciones técnicas ningún país es vecino de sí mismo y si dos países comparten varias aristas solo agregamos una arista en D(M) entre dichos países. Es decir, vamos a trabajar con gráficas simples (estas modificaciones pueden ser justificadas si nuestra meta es demostrar la conjetura protagonista de este capítulo) Si M es un mapa entonces D(M) es una gráfica plana conexa. Basta con dibujar D(M) en el plano de la siguiente manera: para cada cara r de M se escoge un punto r* en el interior de ella, r* representa al correspondiente vértice en D(M). Si r,s son dos regiones de M con una arista común, entonces se traza una línea de r* a s* que pase por susodicha arista. Como M es plana, ninguna de sus aristas intersecan en otra cosa que no sea un vértice de M, por lo cual nuestras aristas trazadas de D(M) tampoco van a intersecarse más que en un vértice de D(M).

- Adaptación del teorema de Cauchy (Euler [Descartes]): Para toda gráfica plana conexa se cumple: p+r-q=2; donde p es el número de vértices, q de aristas y r de caras de la gráfica (para quien quiera una demostración la encontrará en la sección 9.3 de *Graph theory with applications*)
    - Corolario: $q \leq 3p-6$
        - Corolario: En todo mapa M existe un país con no más de cinco vecinos.

I BELIEVE I CAN PROVE IT

H. Minkowski (refiriéndose a la Conjetura de los Cuatro Colores)

Conjetura de los Cuatro Colores (**4CC**): Todo mapa inmerso en un plano puede ser coloreado propiamente a lo sumo con cuatro colores.

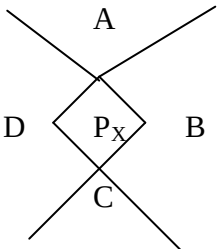
OBSERVACIÓN: La 4CC equivale a que toda gráfica plana conexa es a lo sumo 4-cromática.

Según cuenta la leyenda al estar coloreando un mapa de Inglaterra, Francis Guthrie en 1852 intuyó que cuatro colores eran suficientes para iluminar un mapa sin países vecinos de un

mismo color. Por otro lado un mapa puede estar inmerso en distintas superficies, la conjetura original en este capítulo discutida sólo estudia el caso cuando la superficie es un plano.

DEMOSTRACIÓN
(Síntesis-paráfrasis de la prueba de Kempe publicada en 1879[3] hecha por Haken y Appel[4])

Asumamos la falsedad de la conjetura, entonces existe un mapa M cuya coloración propia requiere de cinco colores y posee el mínimo número de vértices para los cuales cuatro colores no bastan (llamémoslo mapa **minimal**). M contiene a un país (P_x) cuyos vecinos son a lo sumo cinco. Si el número de vecinos de P_x no son más de tres, borremos de la faz del mapa a P_x. Por hipótesis M-P_x puede colorearse propiamente con cuatro colores, sea C dicha coloración. Entonces los vecinos de P_x en M-P_x están pintados a los sumo con tres colores distintos; asignémosle a P_x el color de C no empleado en iluminar a sus vecinos. De esta forma hemos coloreado a M propiamente con cuatro colores. CONTRADICCIÓN.
Si P_x tiene cuatro vecinos (A, B, C, D), procedamos de manera análoga considerando la coloración C de cuatro colores en M-P_x. Para evitar el absurdo anterior los vecinos de P_x deben estar iluminados con cuatro colores distintos (R, Bl, G, Y respectivamente). Sean s.p.g A opuesto por P_x con C y B opuesto con D :



La disyuntiva siguiente aparece: (i) existe un camino en M-P_x de países adyacentes que va de A a C colo reados alternadamente con R-G o (ii) las regiones de colores R-G que contienen a A y a C no intersecan (la **región C_1-C_2 del país Y** es la unión de países X para los cuales existe un camino de países adyacentes pintados alternadamente con C_1,C_2 que van de Y a X). En caso de (ii) intercambiemos los colores de la región R-G de A, entonces A ahora tiene asignado el color G. Si a P_x lo pintamos con R, tenemos una coloración propia de M con cuatro colores. CONTRADICCIÓN. Por lo tanto sea ahora (i), entonces el R-G camino de A a C aísla cromáticamente a D de B en M-P_x, i.e.no existe un Bl-Y camino de B a D. Intercambiemos los colores de la Bl-Y de D, entonces D está pintado de Bl, liberando al color Y y encadenándolo a P_x tenemos una coloración propia de cuatro colores de M. CONTRADICCIÓN. En honor a su creador a los ingeniosos caminos de dos colores alternantes se les denomina **cadenas de Kempe**.

Por último si P_x tiene cinco vecinos de nuevo iluminemos con cuatro colores a M-P_x ($A \approx Green. B \approx Red, C \approx Blue, D \approx Yellow, E \approx Red$) . Transcribo (salvo los colores) el trabalenguas de Kempe:

Si A y C pertenecen a distintas regiones verde-azules, intercambiando los colores en cualquiera de ellas, A y C se convierten ambos en verdes o ambos en azules. Si A y C pertenecen a la misma región verde-azul, vea si A y D pertenecen a diferentes regiones amarilla-azules; en caso de que así sea, intercambiando los colores en cualquier región, A y D se vuelven los dos amarillos o los dos azules. Si A y C pertenecen a la misma región verde-azul, y A y D pertenecen a la misma región amarilla-azul, las dos regiones separan B de E, por lo que la región roja-amarilla a la que B pertenece es diferente de aquella en donde están D y E, y la región roja-verde a la que E pertenece es diferente a aquella en donde están B y C. Así, intercambiando los colores en la región roja-amarilla a la que B pertenece, y en la región roja-verde a la que E pertenece, B torna en amarillo y E torna en verde, mientras que A, C y D permanecen inalterados. En cada uno de los tres casos el número de colores se reduce a tres [5]. CONTRADICCIÓN

Recapitulando, todo mapa normal inevitablemente contiene a un país con menos de seis vecinos. Gracias a éste se pueden reducir a cuatro los colores requeridos para colorear propiamente al caso crítico del mapa minimal de cinco colores (!). Por lo tanto son cuatro los colores suficientes para colorear propiamente a cualquier mapa. ¿QED?

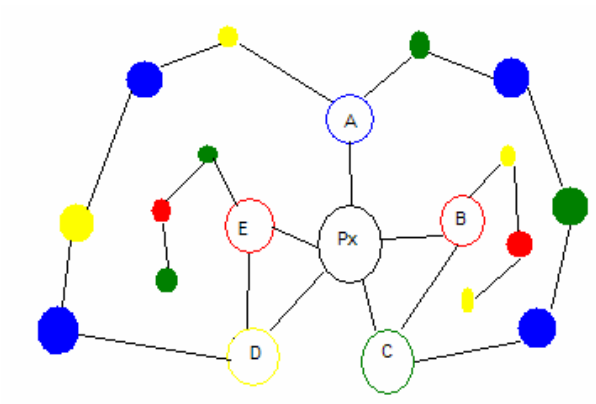
HEAVEN IS ANGERED BY MY ARROGANCE; MY PROOF IS ALSO DEFECTIVE.

palabras de Minkowsky inspiradas por su fracaso

La demostración de Kempe vivió por once años. Fue publicada en 1879 y refutada por P. Heawood en 1890. En el ínterin gozó de la credibilidad de la comunidad matemática, la demostración fue *convincente*[6]. La "falla de Kempe" derrumbó al teorema de los cuatro colores y sacudió a los procesos de confirmación dentro de las matemáticas. Los análisis sísmicos pueden agruparse bajo dos miramientos: el de la sociología y el de la filosofía de las matemáticas. La visión polar del primero abogaría por el término "certificación" en lugar de "ratificación", ateniéndose sólo a lo social y despreciando al contenido matemático. Como toda posición extrema se sostiene por obra y gracia de la obstinación ciega, peca por ser deletérea (!). Análogo descrédito merece aquella postura testarudamente reticente a los aspectos sociales como factor positivo en la generación/conservación de conocimiento. Releguemos a los absolutismos a una institución con una mayor sagacidad para manejarlos y manipular a su antojo a quienes se atengan a ellos. La actividad matemática está incrustada en una estructura social, lo cual no implica que las matemáticas sean sociales. Nuestro conocimiento matemático ha crecido y se ha diversificado gracias al desarrollo y estructuración de la comunidad matemática, pero de la anterior premisa no se infiere que sin los matemáticos ya no hay matemática, equivaldría a asegurar que sin oyentes el ruido de una abstracción al caer en el agua de la razón desaparece. Más aún, lo social corre por las venas y las arterias de las matemáticas, fluye por sus lenguajes, vibra en el "splash". Por once años las instituciones y los individuos aceptaron la demostración de Kempe. Ninguna actividad humana está exenta de errores. La arrogancia preludia a la

decepción. ¿En dónde estuvo la falla y por qué pasó desapercibida por tanto tiempo? Debido al rubro de este ensayo asumamos principalmente la perspectiva desde la filosofía de las matemáticas.

La paráfrasis de la demostración de Kempe difiere notoriamente de las pruebas dentro de un sistema formal descritas en el capítulo uno. Abundan las deducciones implícitas, los atajos de dudosa procedencia. Sin la intuición en primera instancia y en un sentido más rigorista sin nociones matemáticas previas resulta *ininteligible.* La estrategia de la reducción al absurdo se emplea desde la matemática griega clásica y está basada en los principios lógicos clásicos de la no contradicción y el tercero excluido. Suponemos la falsedad de la 4CC y se construye un absurdo. Luego se utiliza un resultado ya demostrado (confiemos o si dudamos se puede verificar) sobre la existencia de un país P_x cuyo número de vecinos es menor a seis. Asumimos la existencia de un mapa M con el menor número de países y en el cual no se cumple la 4CC (mapa minimal). Intuitivamente como los mapas tienen una cantidad finita de países debe existir dicho M (se puede formalizar la existencia de M por el buen orden de los números naturales, ya que el conjunto de naturales correspondientes al número de países en los mapas rebeldes es un subconjunto de los naturales). Cuando el número de vecinos de P_x es menor o igual a tres, se apela más al sentido común, a la intuición llana; sobra al menos un color para P_x. Si el número de vecinos es igual a cuatro se utiliza el truco clave: las cadenas de Kempe. La percepción espacial asiste a la creencia en la validez de los argumentos empleados, al menos en los mapas en un PLANO. Por ejemplo, cuando la AC cadena de Kempe separa a B de D confiamos en este hecho pues nuestra intuición espacial nos conmina a aceptar que dos países en un plano no pueden traslaparse y por la comprensión de coloración propia tampoco pueden sobreponerse cromáticamente. También se le puede dar más rigor, v.gr. empleando el Teorema de la Curva de Jordan[7].
En la postrimería del último caso la demostración trastabilla. Un posible panorama alentador lo exhibe la siguiente gráfica dual:

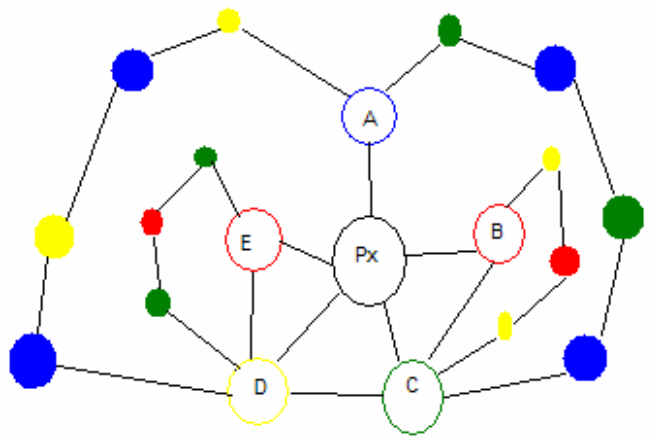


Se puede verificar de forma similar al caso anterior que las cadenas AC y AD separan a B de E, por lo cual las regiones R-Y de E y R-Y de B son ajenas y de igual forma las regiones R-G de E y R-G de B están aisladas. En la figura anterior la transposición de colores de la

región R-G de E y de la R-Y de B hace su cometido borrando al color rojo de los vecinos de P_x. Sin embargo las regiones R-Y de B y R-G de E pueden tener problemas al intercambiar los colores, los candados de las cadenas no son indestructibles. Heawood se percató de este inconveniente:

El Sr. Kempe dice que las transposiciones de colores en la región rojo-verde de E y en la región rojo-amarilla de B removerán cada una a un rojo, y lo que es requerido se cumple. Si esto fuera así, llevaría en seguida a una prueba de la proposición en cuestión.... Pero desgraciadamente, es concebible que aunque cualquier transposición quita un rojo, las dos a lo mejor no quitan ambos rojos. [8]

Las regiones R-G de E y R-Y de B pueden ser vecinas. Una breve modificación de la gráfica dual anterior constituye un contraejemplo de la erradicación de los dos rojos (fig. siguiente). Al intercambiar los colores en el par de regiones los adyacentes C y D quedan pintados con el mismo color (rojo), por lo cual no puede llevarse a cabo la doble transposición. El truco de Kempe queda invalidado. El absurdo no puede ser edificado.



Heawood empuñó al último de los casos e hirió de muerte al trabajo de Kempe, la demostración entró en una fase terminal. No la pudo corregir Kempe ni nadie más de su época. La muerte del teorema de los cuatro colores había sido proclamada. No obstante la esperanza y los esfuerzos por revivirlo no pararon. La fe precede a los milagros.

Encarrilados por el caso para cuatro vecinos y por los encadenamientos de Kempe con una efectiva acción aislante, la voluble intuición podría aceptar la consumación de la liberación de los dos rojos. Para los escépticos la búsqueda de un contraejemplo[9] no era tarea tan sencilla, pues independientemente de la claridad de la representación del problema, una búsqueda a ciegas difícilmente hubiera conducido al contraejemplo de Heawood. De un conjunto ? de hipótesis ("las regiones R-G de B y de E están aisladas", etc) Kempe y muchos más concluyeron a, i.e. "el intercambio de colores de las regiones R-G de E y R-Y de B libera dos rojos" .Por once años a fue consecuencia de ?. El recelo farfulla la pregunta espinosa: ¿De ? cómo se deduce a? Peor todavía, ¿cómo inferimos todos los pasos intermedios? En la precedente breve exposición se señalaron dos vías: la tornadiza intuición y al anclaje en resultados matemáticos más sólidos, i.e. con una menor

intervención de la intuición y con un mayor apego a la rigidez de la lógica. El segundo camino en los tiempos de gloria de la demostración de Kempe apenas empezaba a recobrar vigor[10]. La intuición aunque bien intencionada fracasó, no contempló al caso cuando son vecinas las regiones que intercambian sus colores. Las creativas cadenas de Kempe pesaron demasiado sobre sus seguidores, les exigieron pensar sólo en dos colores. Whitney no lo hizo, las puso en entredicho y rompió al bicolor maleficio.

El rigor no es una terquedad obtusa, es una necesidad para solventar a la confusa intuición, tal como fue puntualizado en la sección dedicada a Wang en el primer capítulo. Aquí ha sido ejemplificado. Por otro lado, las ideas de la demostración de Kempe no murieron por completo, en su agonía siguieron inspirando trabajos matemáticos posteriores en pos de la demostración de la 4CC. Sin embargo haría falta un aparato desfibrilador para que las estrategias intuitivas de Kempe bombearan de nuevo y volviera a la vida el teorema de los cuatro colores. La intuición reclamaría su triunfo después de sucesivas mutaciones y por la intervención de un dispositivo inesperado. Moraleja, la intuición y el rigor aunque a veces enraizados en disputas neuróticas, en las demostraciones matemáticas son una fructífera pareja.

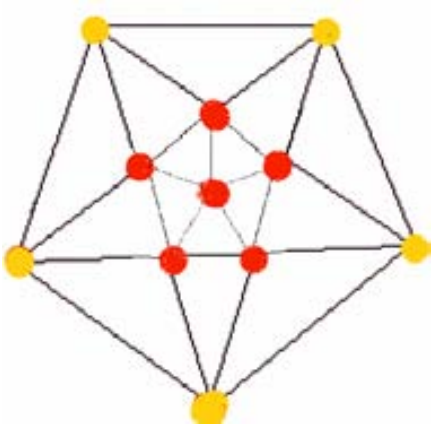
Tamizo lo hasta aquí expuesto: las demostraciones pueden *convencer* porque al *revisarlas* encontramos razones intuitivas (v.gr. intuición espacial) y resultados más rigurosos (v.gr. teoremas) que la respalden y porque su forma lógica, epónima del rigor, es reconociblemente válida (v.gr. reducción al absurdo). No siempre concuerdan la intuición y el rigor. Cuando la balanza se inclina marcadamente a favor de alguna de ellas, la demostración entra en un peligroso desequilibrio. Podemos caer en demostraciones falaces o en demostraciones ilegibles. Irónicamente, gracias a la computadora las ingeniosas e intuitivas ideas de Kempe fueron resarcidas pero también gracias a ella el sopor en nuestro entendimiento puede instalarse. Las demostraciones computarizadas normalmente son una larga o pesada cadena de derivaciones/operaciones que magullan a nuestra intuición y asfixian a nuestra comprensión. Después de la sumaria presentación de la demostración de Appel, Haken y Koch retomaré este punto. Adelanto optimista previsto por la información de esta sección, la caída ofrece la posibilidad de un refortalecido ascenso.

ALTHOUGH HIS "PROOF" TURNED OUT NOT TO BE COMPLETE IT CONTAINED MOST OF THE BASIC IDEAS THAT EVENTUALLY LED TO THE CORRECT PROOF ONE CENTURY LATER.
Haken & Appel

Haken y Appel (H&A) proceden de manera similar a su paráfrasis de la demostración de Kempe. Suponen la negación de la 4CC y postulan la existencia de una gráfica minimal que no puede ser coloreada propiamente con cuatro colores. En caso de existir dicha gráfica, tendría que ser triangular[11]. Así entonces Haken y Appel se limitan a trabajar con gráficas triangulares.

Por los resultados de Kempe la gráfica minimal no puede poseer vértices con un grado menor a cinco, pues por ejemplo la existencia de un vértice de grado cuatro induciría una cuatro-coloración de la gráfica minimal, i.e. reduciría a cuatro el número de colores empleados para la coloración propia. Una **configuración** B de una gráfica G es una subgráfica conexa de ésta ($B \subset G$) más la manera en la que está incrustada en ella. Las configuraciones se determinan por los vértices pertenecientes a B y los grados de éstos en G. Por ejemplo, sea B la configuración de un vértice de grado cinco cuyos vértices adyacentes también tienen grado cinco. Por una parte B sería la subgráfica inducida por estos vértices (i.e. la subgráfica de G integrada por los vértices de B y las aristas entre ellos que estén presentes en G):

Además B debe considerar a todos los vértices adyacentes a B, como G es triangular la subgráfica generada por la adición de estos vértices (llamados vecindad primera de B) será única (salvo isomorfismos). Es más, los vértices de la vecindad primera de toda configuración formarán un ciclo alrededor de ésta, dicho ciclo es denominado **anillo** de la configuración. Una **configuración n-anular** es aquella rodeada por un ciclo conformado por n vértices. B es una configuración 5-anular:



Alentados por el trabajo de Kempe vale la pena estudiar las posibles configuraciones cuya presencia implique una disminución de colores en la coloración, a éstas se les denomina **configuraciones reducibles**. Varios matemáticos exploraron esta idea, por ejemplo Birkhoff en 1913 nos regaló el siguiente teorema:

Una configuración de un vértice de grado cinco con tres vecinos consecutivos de grado cinco es reducible.

En la demostración de Kempe se evidenciaba otro hecho fundamental: en cualquier gráfica plana existe un vértice con menos de seis aristas. De donde se deduce la inevitable existencia de la configuración constituida por un vértice de grado cinco en todo mapa candidato a ser el contraejemplo de la 4CC. Ampliando la anterior idea podemos hablar de un conjunto U de configuraciones donde si G es una gráfica triangular con aspiraciones a ser el contraejemplo entonces existe B$\in$ U tal que B es configuración de G. U fue bautizado con el nombre trágico de **conjunto inevitable**.

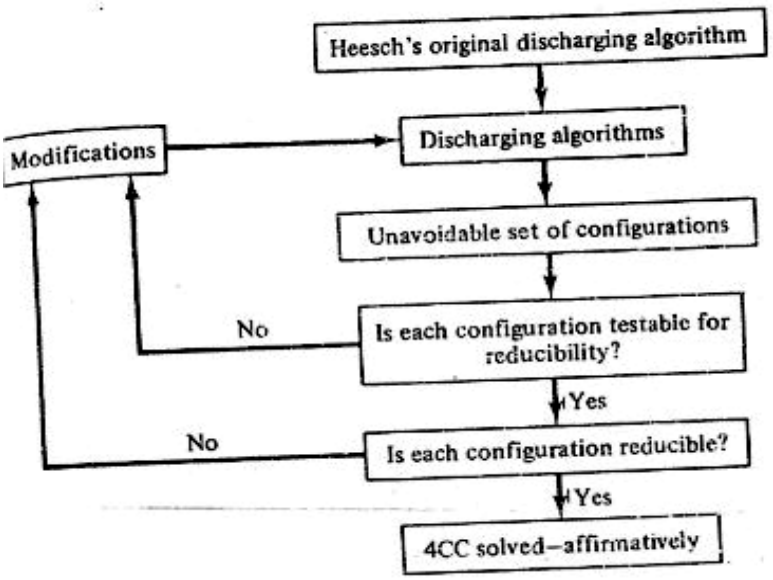
La reducibilidad y el conjunto inevitable son el par de pilares donde se apoya la demostración de H&A. Si podemos construir un conjunto inevitable U cuyos elementos sean todos reducibles habremos elaborado al absurdo de la negación de la 4CC. Suena muy bien, pero ¿cómo ejecutar tan melodiosa estrategia? H&A lo hicieron recurriendo a un instrumento binario. Y su atrevida interpretación no pasó inadvertida, provocando variopintas críticas.

Para la construcción del conjunto U Haken y Appel adaptaron al algoritmo propuesto por el matemático Heinrich Heesch, quien encontró una manera para edificar de forma mecánica a un conjunto inevitable. La idea es simple, si tomamos prestados conceptos sencillos de la electricidad como lo sugieren H&A en (4). A cada vértice v de la gráfica G se le asigna la carga inicial $q(v)=6-\delta_G(v)$. Si G es una gráfica minimal, entonces todos los vértices tienen una carga inicial menor o igual a 1; de hecho los únicos vértices con carga positiva son aquellos de grado cinco. Más aun, como G es triangular la sumatoria de las cargas iniciales es igual a doce[12].
Ahora bien, se pueden definir diversos procedimientos de descarga deferentes con la electricidad para mover las cargas de tal forma que no quede ningún vértice con carga positiva. Si v es un vértice de grado cinco la cantidad de su carga perdida debe incrementar la carga de algunos de los vértices negativos aledaños a v, afín a la teoría de la electricidad la sumatoria de las cargas debe ser constante. Entonces un **algoritmo de descarga** es aquel que redistribuye las cargas manteniendo la sumatoria total constante y permitiendo únicamente el flujo de positivo a negativo (o a neutro) con el fin de eliminar a todos los vértices con carga positiva. Cuando G es triangular cualquier procedimiento de descarga es inefectivo ya que la suma de las cargas siempre será igual a doce. Por lo tanto en G deben existir ciertas configuraciones que inhiban a los procesos de descarga. Además, dado un algoritmo de descarga se pueden enumerar las configuraciones causantes de su malogro. El conjunto de las configuraciones obstructoras del algoritmo es un conjunto inevitable[13].

Resalta la utilización de ideas del ámbito físico (electricidad) en un problema matemático, aunque no es un práctica novedosa[14] tampoco deja de ser sorprendente. La yuxtaposición de teorías en matemáticas es algo común, no obstante es nada corriente si una de las participantes de la fiesta proviene de una ciencia natural. Objeción supina, la posibilidad de *materializar* a los objetos matemáticos implica una evidencia en contra de su existencia

ideal pura y libre de manchas físicas. No a lugar. El tribunal es epistemológico. Heesch a lo mejor importó nociones de la electricidad y las incorporó en la teoría de gráficas; descargando sus cualidades físicas descriptivas pero conservándolas en el nivel del entendimiento. Pensar en cargas ayuda a comprender el algoritmo de Heesch y también nos permite vislumbrar al posible proceso heurístico detrás de la creación de éste. Otra observación importante, el algoritmo debe ser modificado para los fines buscados por H&A: la construcción de un conjunto inevitable de configuraciones reducibles. La suciedad metodológica amaga con inficionar a la inmaculada matemática, H&A por medio de la experimentación diseñaron el algoritmo requerido. ¿Experimentación en las matemáticas? Peor, el instrumento llamado computadora brindó la posibilidad de dicha experimentación. Mucho peor, la computadora hizo el trabajo sucio de verificar la reducibilidad de las configuraciones del conjunto inevitable. El acabose, la llama de la esperanza en el éxito de la empresa fue un argumento probabilístico. Después de este breve boceto, a nadie debe coger desprevenido las reacciones encontradas de hostilidad y adulación suscitadas por las demostración de H&A de la 4CC.

El siguiente diagrama de flujo[15] presenta de manera concisa las etapas de la manufactura de la demostración de H&A :



Con respecto a la reducibilidad se le deben también a Heesch algunos pasos clave para el diseño de los algoritmos encargados de revisar dicha cualidad en las configuraciones sometidas al escrutinio. En particular fue él quien por vez primera señaló algunos óbices a evitar si anhelamos encontrar la reducción de una configuración. Una configuración es **geográficamente buena** si ningún vértice dentro ella es adyacente a más de tres vértices del anillo y cuando se cumple la negación de lo anterior entonces los tres vértices del anillo deben ser adyacentes. Por lo cual el algoritmo de descarga debe producir configuraciones

geográficamente buenas, configuraciones con una mayor probabilidad de ser reducibles. Por otro lado fue también Heesch quien definió la D-reducibilidad, la cual ocupa un lugar primordial dentro de la demostración pues los algoritmos para verificarla pueden ser implementados sin tantas dificultades en la computadora. Sea R una configuración, A_R el anillo de dicha configuración y G una gráfica minimal. Entonces G-R (a G se le quitan los vértices de la configuración y las aristas que inciden en R) es cuatro-coloreable y su cuatro-coloración propia c induce una coloración propia c' de A_R. Si la coloración inducida c' en A_R puede directamente extenderse a R o si mediante el intercambio de colores en las cadenas de Kempe de A_R se puede inducir una nueva coloración c'' extensible a R entonces dicha configuración es **D-reducible**. Por ejemplo sea R la configuración conformada por un vértice de grado cuatro. Entonces R es cuatro-anular. Siguiendo la demostración de Kempe si la coloración propia c de A_R es de menos de cuatro colores directamente se puede definir una coloración propia de R de menos de cinco colores. Si c tiene cuatro colores entonces utilizando el truco de las cadenas de Kempe podemos definir una cuatro-coloración propia de $R \cup A_R$. Como el número de posibles coloraciones propias de cuatro colores de un anillo con n vértices es finito, podemos definir un algoritmo para corroborar la D-reducibilidad de la siguiente forma:

Para toda coloración propia c de cuatro colores de A_R:

(1) Si c es directamente extensible a R termina
si no
(2) Busca un intercambio de colores en las cadenas de Kempe tal que la coloración propia pueda ser extendida a R.

Si para toda coloración de A_R se satisface (1) o (2) entonces R es D-reducible. El algoritmo hace uso de la fuerza bruta de las computadoras. Por ejemplo un anillo de 13 vértices tiene 66,430 cuatro-coloraciones diferentes. Con respecto a la extensión directa hay varios algoritmos para cuatro-colorear una gráfica plana, en particular el empleado por H&A es de orden polinomial (bicuadrático). Por lo tanto el poder computacional restringe el tamaño de los anillos analizables, lo hacía en la década de los setenta y todavía con nuestras ufanas supercomputadoras lo sigue haciendo. Por estos motivos se justifica la exigencia de la generación de configuraciones inevitables propicias a ser reducibles, i.e. configuraciones geográficamente buenas y cuyos tamaños de sus anillos no sean muy grandes. El escenario estaba listo para la experimentación, para sostener el diálogo hombre-máquina[16]. Cierro este párrafo con otra importante contribución de Heesch para la demostración de H&A. Heesch con base al trabajo sobre reducibilidad de Birkhoff definió la C-reducibilidad. A grosso modo la C-reducibilidad consiste en remplazar a la configuración en cuestión con una subgráfica de la configuración de menos vértices llamada reductor. Si G es minimal entonces la gráfica que resulta de remplazar a la configuración original B con el reductor R es cuatro-coloreable. Si todas las cuatro-coloraciones del anillo compatibles con R (i.e. las que inducen coloraciones propias del reductor) pueden ser extendidas a la configuración original (directamente o por medio de las transposiciones cromáticas de las cadenas de Kempe) entonces la configuración B es **C-reducible**. Hace falta verificar la definición, i.e.

comprobar que la C-reducibilidad también nos conduce hacia el absurdo. Dentro de las coloraciones propias de cuatro colores de G-B como (G-B)∪R es cuatro coloreable entonces debe existir una coloración c compatible con R. Luego como B es C-reducible la coloración c compatible con R puede ser extendida a B. Por lo cual G tiene una coloración propia de cuatro colores. CONTRADICCIÓN. La automatización de la C-reducibilidad aunque resulta más complicada (el número de reductores puede ser muy grande) puede llevarse a cabo. Si B es D-reducible entonces B es C-reducible (recordemos que para la D-reducibilidad todas las cuatro coloraciones propias del anillo deben ser extensibles a la configuración). El recíproco no siempre se cumple. La C y D-reducibilidad serán las piedras de toque en cuanto a la reducibilidad utilizadas por Haken y Appel en su demostración computarizada.

La primera fase del trabajo de H&A consistió en refinar al procedimiento de construcción del conjunto inevitable deseado. Experimentaron con la asistencia de la computadora diferentes algoritmos de descarga y los conjuntos inevitables por ellos obtenidos. La computadora enumeraba las configuraciones resultantes de un algoritmo de descarga; H&A las revisaban y hacían las correcciones pertinentes. El algoritmo mejorado se programaba y empezaba de nuevo el periodo de pruebas. Cuando H&A consideraron lo suficientemente maduro a su algoritmo lo abordaron desde una perspectiva deductiva, probando la existencia del conjunto U de configuraciones geográficamente buenas[17] construido gracias a éste. Sin embargo en esa fase no se restringió el tamaño de los anillos, por lo cual aunque de antemano se garantizaba la existencia de U muchas de las configuraciones pertenecientes a éste podrían no permitir la revisión sobre su reducibilidad con los medios disponibles. No obstante confirmar la existencia de ese U confirió a la gesta una mayor probabilidad de éxito, espantando por el momento a los fantasmas de la claudicación.

Ahora era prioridad para H&A disminuir el tamaño de los anillos, por lo cual la experimentación siguió en pie. Cabe señalar un tercer obstáculo de la reducibilidad señalado por Heesch: un par de vértices adyacentes de grado cinco donde todos sus demás vecinos están contenidos en el anillo[18]. A partir de aquí H&A se enfocaron principalmente en configuraciones libres de los tres obstáculos y cuyos anillos no fueran de un gran tamaño. Efusivos por los resultados positivos del empleo de la computadora, escribieron:

A estas alturas el programa, que había absorbido nuestras ideas y mejoras durante dos años, empezó a sorprendernos. Al principio verificábamos sus argumentos a mano para que siempre pudiéramos predecir el curso que él seguiría en cualquier situación; sin embargo de repente empezó a actuar como las máquinas que juegan ajedrez... empezó a enseñarnos cosas sobre cómo proceder que nosotros nunca esperábamos. En cierto sentido había superado a sus creadores en algunos aspectos "intelectuales" así como también en las partes mecánicas de la tarea. [19]

Los experimentos generan resultados no previstos, los resultados proporcionan una retroalimentación. Con paternal orgullo, H&A halagan a su creación. El instrumento ganó un protagonismo definitivo, una incidencia cada vez mayor en el proceso de la

demostración. Donde la acuidad racional flaquee la potencia computacional acude al rescate. Si la participación de la máquina no hubiera superado el nivel de interacción-experimentación, la demostración de H&A no hubiera sido tan transgresora. Sin embargo H&A prosiguieron con la explotación de la computadora, con la asistencia de un nuevo cómplice de nombre John Koch.

El siguiente escalón a subir era el de la reducibilidad. Koch fue el encargado de programar los algoritmos de verificación de la C,D-reducibilidad . Se estableció de nuevo un canal de retroalimentación para los procedimientos de la construcción del conjunto inevitable. Gracias a la computadora se podía revisar si una configuración era reducible en un adecuado periodo de tiempo. Si una configuración después de reiterados intentos se resistía a reducirse (configuración crítica), el procedimiento para la construcción del conjunto inevitable debía ser modificado. Por otro lado a estas alturas H&A ya habían optado por la construcción manual del conjunto inevitable, i.e. prefirieron la flexibilidad humana ante la fastidiosa tarea de rescribir el programa por las cuitas de las configuraciones críticas (en busca de superar a las más rejegas diseñaron reglas de descarga particulares). Dato técnico detonante de la polémica: más de mil horas de trabajo-máquina fueron requeridas en tres computadoras para verificar la reducibilidad de las configuraciones, tiempo requerido para establecer el **lema de reducibilidad** (todas las configuraciones del conjunto inevitable U son reducibles). Si un *homo sapiens* ejecuta a los algoritmos implementados por Koch para cerciorarse sobre la supuesta reducibilidad de las configuraciones del conjunto inevitable NO se le augura mucho éxito, aunque se le promete una muerte tranquila inmerso en operaciones inanes. Por otro lado no existen técnicas adaptadas a las capacidades de un ser humano para decidir si cualquier configuración es o no es reducible. La demostración de Haken, Appel, Koch y programas (H&A&K&1&0) parece acorralarnos en una exclusiva elección en la siguiente disyuntiva: confías o no lo haces en el instrumento computadora. Más adelante nos adentraremos en estas sinuosas coyunturas.

Por otro lado como se ha expuesto hasta aquí el esfuerzo fue titánico. Una cantidad colosal de horas-máquinas y horas-hombre fue invertida en la demostración. Nunca hubo certeza sobre el buen término de la empresa, aunque farios probabilísticos favorables la guiaron y la mantuvieron en marcha. Hace algunos párrafos se mencionó la confirmación por medios más ortodoxos de la existencia de un conjunto inevitable de configuraciones geográficamente buenas. Similarmente se puede asegurar un U conformado por configuraciones sin obstáculos. No obstante todavía ahora bajo los métodos tradicionales de las matemáticas no se puede afirmar un U tal que todos sus miembros sean reducibles. El panorama se complica con la insidiosa existencia de un mapa sin configuraciones reducibles de tamaño anular menor a once[20]. H&A no se amilanaron y de acuerdo a la naturaleza de su demostración fabricaron un amuleto-faro probabilístico:

Los argumentos probabilísticos [de la demostración de H&A&K&1&0] son reconocidamente crudos y hacen varias asunciones injustificadas. ... Quizás la analogía más atinada sea compararlos con la intuición que es requerida para obtener un nuevo resultado experimental. Una vez que el resultado ha sido encontrado, uno ya no necesita

defender la probabilidad de que el experimento tenga éxito en demostrarlo. Debido a que el "experimento" que deseaban realizar Appel y Haken tomaría años de valioso tiempo de cómputo, sin mencionar su propio tiempo y esfuerzo, aun un argumento crudo era mejor que ninguno. [21]

A grandes rasgos haciendo uso de la probabilidad H&A pronostican la existencia de un conjunto inevitable cuyos elementos son todos reducibles. Más aún, auguran que las configuraciones pertenecientes al divisado conjunto inevitable son de un tamaño conveniente para ser despachados por los programas verificadores de la reducibilidad. Su análisis muestra lo siguiente : Sea B una configuración n-anular donde n está fija; entonces para esa n existe un valor crítico x tal que si m es mayor a x (m es el número de vértices de B) entonces B goza de una alta probabilidad de ser reducible. Está respaldada la primera suposición. Para complacer a la segunda debemos recordar la misión de generar configuraciones cuyo tamaño anular no fuera muy grande dentro de los procedimientos de construcción del conjunto inevitable. Si el número de vértices del anillo de una de las configuraciones producidas excedía al valor catorce, entonces H&A alteraban al algoritmo de descarga. No obstante el valor crítico para configuraciones 14-anulares podía ser muy grande. Con el **Lema-m** H&A despejaron los miedos: si $m>(3/2)n-6$ y B está libre de obstáculos entonces B casi seguramente es reducible. Por lo tanto el valor crítico para los elementos de U era inferior a 15. Para aquellos quienes no regalan su aceptación como una simple tarjeta de presentación, les recomiendo la lectura de (2) para revisar muchos de los detalles de los argumentos probabilísticos (en general de la demostración) aquí pasados por alto. La probabilidad atenúa a las dudas, no las desvanece por completo. Sin embargo inficionados por la descarada seguridad de conocer el "fin" de la historia, podemos concluir con el enunciado elidido en la cita 21:

... En retrospectiva, sin embargo, sabemos que [los argumentos probabilísticos] son válidos ya que desembocaron en una demostración matemática correcta....[22]

Bajo las recetas experimentales del azar H&A cocinaron su conjunto U, cuya cardinalidad original en 1976 era igual a 1936 aunque lograron posteriormente disminuirla hasta 1482. La aderezaron con alrededor de 500 reglas de descarga para cocer a las configuraciones críticas. Cito la descripción física de H&A de su platillo de la demostración de la 4CC:

El lector debe enfrentar 50 páginas que contienen texto y diagramas, 85 páginas llenas con casi 2500 diagramas adicionales, y 400 páginas en microfichas que contienen más diagramas y miles de comprobaciones individuales de las suposiciones hechas en los 24 lemas de las secciones principales del texto. Además, el lector debe enterarse que se han verificado ciertos hechos con el uso de aproximadamente mil doscientas horas de tiempo de cómputo y que demandaría una cantidad enorme de tiempo verificarlos a mano. Los papeles son en cierta madera intimidantes debido a su estilo y su longitud por lo cual sólo pocos matemáticos los han leído con detalle. [23]

El relatar y explicar sumariamente el proceso de elaboración de la demostración de H&A también tenía como finalidad revelar los ingredientes esenciales de su contenido. Sin duda es un guiso exótico, sui géneris al menos en las matemáticas. Ésta no es una disciplina donde se acostumbre proceder de forma experimental en una de sus actividades primarias, la hechura de las demostraciones. Sobra recalcar la rareza de la utilización de un instrumento para la realización de la demostración de H&A. Basta con la escritura y reescrituras. Su contraparte, la lectura (revisión, comprensión, refutación o ratificación, etc.) de una demostración también por el trabajo de H&A sufrió un drástico revuelo. Al cerrar en una de las instancias las vías ortodoxas de la verificación, la demostración de H&A obligó a los matemáticos a recurrir al instrumento generador del ruido. Sumergirse en los aspectos creativos-metodológicos de la manufactura de la demostración puede alejarnos del motivo principal de este ensayo, por lo cual mejor disfrutemos o padezcamos la degustación del producto dado como terminado. No obstante para cerrar con broche de silicio esta sección transcribo el motivo desencadenador del triunfo de la estrategia experimental-exhaustiva según sus desarrolladores:

La razón fundamental por la que funcionó el argumento del conjunto inevitable mientras otros acercamientos a la Conjetura de Cuatro colores no lo hicieron es que dichos acercamientos necesitan herramientas teóricas algo más fuertes para que sus métodos sean aplicables. Mientras éstos son posibles de crear, hay ninguna garantía que sean realmente asequibles; y si lo son, hay ninguna manera obvia para encontrarlos. [24]

Pragmatismo. Como fue expuesto H&A se pararon sobre los hombros de Heesch, Kempe y muchos más así como también lo hicieron sobre el lomo de la computadora. La ausencia de sofisticadas herramientas teóricas la compensaron con la enorme capacidad de realizar operaciones simples de la computadora. Sustituyeron a la elegancia teórica por la potencia mecánica. La computadora suplió los medios para la experimentación, los argumentos probabilísticos la respaldaron y en parte la dirigieron. Por último la computadora hizo la tarea sucia de revisar la reducibilidad de las configuraciones del conjunto inevitable. ¿H&A establecieron un nuevo paradigma en el desarrollo y confección de las demostraciones? Talvez, siempre y cuando su fardo de más de 100 páginas, 400 microfichas y un montón de horas del trabajo de una computadora sean considerados una demostración. La comunidad matemática después de un periodo de cavilación así lo hizo y lo sigue haciendo, el Teorema de los Cuatro Colores volvió a nacer y sigue vivo (**4CT**). No obstante nunca faltan filósofos metiches sembradores de cizaña.

THERE IS A PRICE FOR THIS SORT OF KNOWLEDGE. IT CANNOT BE ABSOLUTE. BUT THE LOSS OF INNOCENCE HAS ALWAYS ENTAILED A RELATIVISTIC WORLD VIEW; THERE IS NO PROGRESS WITHOUT THE RISK OF ERROR.
Saaty & Kainen

Conmoción, pavor, optimismo, fervor, desconcierto. Alguien más osado con una sola palabra incluiría algunos de los significados de los términos anteriores: revolución. La demostración de A&H&K&1&0 por la celebridad de la conjetura y por las especiales características de ella no podía pasar desapercibida. De nuevo podemos escoger dos circuitos de circulación: el social (procesos de ratificación-aceptación dentro de la comunidad matemática) o el filosófico (elementos necesarios en una demostración). Advertencia: los caminos en algunos recovecos intersecan.

Deambulemos brevemente por la historia de la aceptación de la demostración de la 4CC. Las dos partes de la demostración exigían árbitros distintos. La sección de la inevitabilidad al haber sido escrita por seres humanos también podía ser *revisada* por ellos. Algunos especialistas, entre ellos el francés Jean Mayer[25], "corroboraron" esta parte de la demostración. Sin embargo debido a la tediosa longitud de esta sección nadie presumiblemente la ha verificado por completo. Este fue uno de los dos motivos detrás de la reelaboración en 1996 de la demostración de la 4CC llevada a cabo por Robin Thomas, Paul Seymour, Daniel Sanders y Neil Robertson[26]. El prefijo "re" se aplica pues esta nueva demostración a grandes rasgos es una actualización de la de H&A&K&1&0 al estar cimentada también en el conjunto inevitable de configuraciones reducibles. Además la acción computarizada prevalece y se expande, Thomas et al acataron la recomendación y voto de desconfianza emitido por el especialista Edward Swart:

Me atrevería a decir que cualquier falta de fiabilidad de las pruebas presentes del 4CT reside menos en el uso de una computadora para las pruebas de la reducibilidad y más en el hecho que una computadora no se usara en la creación del conjunto inevitable de configuraciones generado por los procedimientos de descarga.[27]

No obstante los lineamientos generales de la construcción del conjunto inevitable pueden ser y fueron *revisados*, sin recelo se puede afirmar la presencia en esta primera fase de la demostración de los elementos indispensables para que sea *convincente*. Un refuerzo en el grado de *convencimiento* lo proporcionan los ecos de otras demostraciones en progreso de similar manufactura (en el siguiente párrafo se menciona una). Por lo que el problema mayor, se esconde detrás del lema de reducibilidad, lema respaldado por una fantasmal demostración computarizada. Antes de abordar al binario dilema cabe señalar la identificación de un par de errores en la primera parte, errores diligentemente atendidos por H&A en la modificación del algoritmo de descarga[28]. Temblores innocuos, reforzadores de la confianza y el convencimiento.

La inspección del lema de reducibilidad fue encargada entre otros a Frank Allaire, quien también andaba al acecho de la demostración de la 4CC con los mismos medios. Allaire ya contaba con programas para verificar a la reducibilidad para beneplácito de H&A, aunque para desmérito de Allaire su inconclusa demostración se vio opacada por la anticipación de H&A. Allaire comparó los resultados de los programas de Koch con los arrojados por los suyos y coincidieron. En 1977 publicó Allaire[29] su demostración, basada también en la estrategia del conjunto inevitable de configuraciones reducibles pero sólo requería de 50 horas de trabajo computacional. Tanto sus programas para examinar la reducibilidad como su algoritmo de descarga eran diferentes a los de H&A. Citando a Swart:

Menospreciando su esfuerzo, al menos la demostración de Allaire debe considerarse como una corroboración independiente de la verdad de la conjetura de los cuatro colores, y ya hay muy pocas dudas, aun cuando la prueba de Hakken/Appel tenga fallas, el teorema no obstante es verdadero. [30]

Al discrepar los algoritmos de descarga el conjunto inevitable de Allaire era distinto al de H&A. No obstante el plano de prueba era similar, la demostración de Allaire ofrecía análogos cimientos persuasivos para dudar de su invalidez y también se cimbraba por su inusual establecimiento del lema de reducibilidad. Para molestia de algunos como Swart y el mismo Allaire[31], la parte de la inevitabilidad de la demostración de H&A salió avante, aun sin el completo escrutinio del conjunto U. La existencia de otro U totalmente distinto en otra demostración con apariencia de ser correcta (la ironía la sufrió Allaire) acallaron a las voces escépticas de esa parte de la demostración (no puedo silenciar al siguiente tendencioso comentario: "además esa tediosa y técnica sección de la demostración fue cuidadosamente realizada por seres humanos"). El debate se centró entonces en el segundo eje de la demostración, el lema de la reducibilidad.

La evaluación de la parte conflictiva recayó en programas de reducibilidad independientes, otra computadora avaló el trabajo de su congénere. Sobra mencionar la aceptación de los algoritmos, los cuales pueden ortodoxamente ser examinados para concederles el visto bueno. El meollo del asunto esta encajado en nuestra incapacidad para revisar los resultados de la implementación de los algoritmos. Para ver de soslayo a la inexpugnable tarea recordemos que en nuestro sencillo algoritmo de la D-reducibilidad se inspeccionan TODAS las coloraciones de cuatro colores propias de los anillos. El mero número de subcasos a revisar en una configuración 14-anular firma nuestra carta de renuncia. De nada sirve si se imprimen todas las operaciones efectuadas por el computador, al menos si nuestra intención no es tapizar con esas hojas las paredes de la universidad. El escenario ha sido montado, es tiempo del diálogo de los actores en torno a la trama principal de esta tesis.

El plano de la demostración de Wolfgang Haken y Keneth Appel es conservador. Es una demostración por reducción al absurdo, montada sobre las ideas de Heesch del conjunto inevitable y de las configuraciones reducibles. Desconcertantemente los albañiles utilizaron maquinaria pesada para levantar a la contradicción. Los métodos artesanales fueron

vejados, pues la única forma para verificar a la obra implica de nuevo al uso de la computadora. El edificio pende del lema de la reducibilidad, el cual según sus ingenieros está cimentado por programas computacionales. La pregunta inmediata parece de antemano oscilar entre el rechazo y el aval, ¿el lema está demostrado por medio de una computadora? Si la respuesta es afirmativa, entonces la siguiente disyuntiva se presenta: nuestro concepto de demostración incluye a este tipo de demostración o lo debemos expandir para adoptarlo. Por otro lado, si la negación es nuestra opción, podemos ir a contracorriente negando tajantemente el estatuto de teorema de la 4CC o con bríos divinos podemos ampliar a la taxonomía matemática, incorporando una nueva clase vecina a los teoremas. Bajo cualquiera de las dos respuestas la sombra de la experimentación (¿en las matemáticas?) se proyecta. Como sucedió en el área de la inteligencia artificial, la afirmación sobre si las computadoras tienen tal o cual propiedad (capacidad) nos remite a la revisión de la definición de la propiedad (capacidad) atribuida. En nuestro caso aunque la tarea es nada fácil, al menos es mucho más sencilla ¿Qué es una demostración? Que nos responda uno de los protagonistas de la tragicomedia filosófica.

Thomas Tymoczko (Tym) caracteriza a las demostraciones de la siguiente forma:

(a) Las demostraciones son convincentes. Este hecho es la clave para entender a las matemáticas como una actividad humana. Porque las pruebas son convincentes para un matemático arbitrario pueden asumir el papel como árbitros del juicio en la comunidad matemática.... ¿por qué son convincentes?... [porque las demostraciones] son revisables y son formalizables .

(b) Las demostraciones son revisables. Las demostraciones son las garantías del conocimiento matemático y por eso deben poder ser comprendidas por los matemáticos. Una demostración es una construcción que puede ser ojeada, revisada, verificada por un agente racional.

(c) Las demostraciones son formalizables. Una prueba, como está definida en la lógica, es una sucesión finita de fórmulas de una teoría formal que satisface ciertas condiciones. Es una deducción de la conclusión desde los axiomas de la teoría por medio de los axiomas y reglas de la lógica. La mayoría de los matemáticos y filósofos cree que cualquier demostración aceptable puede formalizarse... [32]

(PRECAUCIÓN: la *formalización* aludida por Tym es local, no alude a un sistema formal madre de todas las pruebas cuya existencia ha sido denegada por el Teorema de Incompletud de Gödel , imposibilidad en el capítulo tercero explicada con un poco más de palabras)

Tymoczko a cada una de las tres caracterizaciones la asocia con un disciplina para su estudio, ayudando a clarificar el sentido de éstas y sobretodo sus intenciones: (a)antropología (bien podría ser sociología), (b)epistemología y (c) lógica; a todas favor de agregar las tres palabras “de las matemáticas”. Precavido evita arriesgarse demasiado, el

inciso (a) lo explica con (b) y (c). Una demostración es *convincente* por sus propiedades estructurales (lógica) y por su revestimiento semántico (epistemología), recubrimiento perceptible por los matemáticos, al menos por los especialistas del área matemática en cuestión. Hace ya varios párrafos mencioné dos polos generadores del conocimiento de una demostración, el conocimiento asistido por la intuición y conocimiento asentado en el rigor lógico. Es hora de acoplarlos con (b) y (c), En otro artículo Tym hace patente las metas de (b) en conjunción con (c):

"Verificar una demostración" se resuelve en dos componentes: el acto epistemológico de verificar una prueba [gulp! ref.circ.] y la distinción lógico-matemática en las pretendidas demostraciones entre las demostraciones (reales) y las falsificaciones.. [33]

Revisar a una demostración conlleva a su conocimiento:

La habilidad de los matemáticos de revisar las demostraciones proporciona un panorama idealizado sobre el conocimiento matemático de los teoremas. ¿Cómo un matemático llega a saber un teorema? Revisando una demostración de él. [34]

¿Se debe considerar a la intuición como un elemento perteneciente al marco epistemológico de los matemáticos? Haciendo uso del sobado argumento aristotélico contra el descenso argumentativo infinito, en algún punto debemos parar en conocimiento no demostrado y talvez arribemos en una variante del *doblepensar*[35] sufrido por los habitantes de Oceanía en 1984: una verdad dimanada por nuestra intuición pero que gracias a la razón se borra la fuente y nos quedamos con la emisión. Por lo tanto en un sentido estricto la intuición debe incluirse en el costal epistemológico pero debe acatar a las modificaciones pertinentes del tipo de conocimiento aludido (intuición no es igual a conocimiento). La intuición matemática conviene representarla como una memoria tipo EPROM (Erasable Programable Read Only Memory), puede venir programada de fábrica pero permite reprogramarse bajo la luz de la educación y el entrenamiento. Para evitar posibles querellas a partir de aquí ya no haré uso del término "intuición", el término *"surveyable"* tornó en castellano en *revisable,* intentando mantenerme fiel a su significado original. Sin embargo, *"inteligible"* es una mejor alternativa pues denota a las componentes epistemológicas (incluida la intuición matemática) que facilitan la comprensión de una demostración y esta opción más adelante será la escogida después de haber realizado los cambios pertinentes.

Por otro lado si la demostración cumple con (c)ser *formalizable*, entonces la demostración para mí será *computable*. En el primer capítulo se mencionó hasta el cansancio el grado de mecanización de los sistemas formales, lo cual le permite a la versión tamaño bolsillo de una máquina de Turing (dudas sobre M. de Turing en el capítulo tercero) que responde por el nombre de computadora manipular a las fórmulas ya sea para fabricar o verificar pruebas (recordemos prueba=demostración formal). El bautizo no es mero capricho, es un símbolo de las ideas de Wang. Una demostración será más sencilla y segura si hasta los *talacheros* binarios pueden crearla o verificarla, es decir si satisface al inciso (c)la demostración es *computable* .

Los enunciados de la forma "α ser(conjugado en presente) Σ" son impecablemente sucios desde la burbuja de la filosofía (!). El verbo ser es una porquería, tanto en su acción pringosa como en su resbaladizo significado. O bien nos conformamos con relaciones de sinonimia, disfrutando el remolino concupiscente de las referencias circulares (frenado de vez en cuando por los templanza ostensiva-esotérica) o vomitamos ciertas propiedades de α que casi con seguridad no son α. Regado a escupitinas, el ser brota con el estiércol del lenguaje, ser no es sino escatología en reciclaje. Desde un punto de vista filosófico concuerdo con (c) y un poco con (b) y me conformo con (a) como practicante de las matemáticas, en suma aserto a una versión más débil de la caracterización de Tymoczko: si x es una demostración entonces (a)x es convincente o (b)x es revisable* o (c)x es computable –el asterisco receloso será explicado posteriormente-. Por otro lado el recíproco me parece lo suficientemente insulso como ser aquí discutido, x es (a) $\vee$ x es (b) $\vee$ x es (c) no implica que x es una demostración. Cambiar la disyunción por la conjunción periclita a la noción de la demostración, con ánimos sediciosos así procede Tym.

Ser *revisable* para Tymoczko encubre una revelación, él hace hincapié en que la legibilidad precede a la lectura de comprensión. Las siguientes oraciones fueron recortadas del inciso (b):

Decimos a menudo que una demostración debe ser clara o capaz de ser verificable a mano. Es una exhibición, una derivación de la conclusión, y no necesita nada fuera de sí misma para ser convincente. El matemático inspecciona a la demostración en su integridad y por eso llega a saber la conclusión. [36]

Sorpresa mojada, ninguna exclamación de asombro detonan las siguientes líneas:

...decir que pueden revisarse [las demostraciones] es decir que pueden verificarse definitivamente por los miembros de la comunidad matemática. [37]

La característica (a) en cuanto retórica exige un público a quien persuadir. Según Tymoczko si x es una demostración entonces x es (a) *convincente* y x es (b) *revisable* y x es (c)*computable*, además simplifica la caracterización explicando su convencimiento con base a su *revisabilidad* y *computabilidad* , así si x es (a) entonces x es (c) & x es (b). Por lo tanto una demostración es *convincente* porque es legible y puede ser comprendida y asentida por el quórum matemático. Más aún, si su presentación no es la de una prueba formal entonces la demostración es un simulacro de aquella otra, al menos tiene una estructura lógica lo suficientemente resistente para soportar a la demostración y alimentar a la fe de su posible formalización. A partir de este razonamiento con apariencia de borrego esponjado de redundancias Tym ataca a dentelladas al "Teorema" de los Cuatro Colores. Midamos la fuerza de sus mordidas. De ahora en adelante cuando traduzca *surveyable* como "*revisable*" será para remarcar a la legibilidad total –humana- y al chauvinismo epistemológico –*no necesita nada fuera de sí misma* para comprenderse-, propiedades propias de las demostraciones según Tym .

Para Tymoczko el lema de la reducibilidad no es un lema bajo su caracterización de las demostraciones porque no es *revisable*. Por lo tanto la demostración de H&A no es una demostración en el sentido normal de la palabra (bajo la taimada caracterización de Tym.). Poco importa si el resto de la demostración de H&A sí puede ser inspeccionada por los seres humanos, el lema "todas las configuraciones del conjunto inevitable son D o C-reducibles" esta cerrado en una caja negra. La computadora después de varias horas y millones de operaciones confirmó el lema, por lo cual o lo aceptamos sin revisarlo o por no poder revisarlo lo desechamos. De nada sirve si otros programas independientes concuerdan con los programas de Koch, pues tampoco son *revisables* los resultados de dichos programas. Una alternativa consistiría en eliminar la exigencia antropológica de la *revisabilidad*. Aunque al hacerlo ¿de paso no prescindimos del sentido epistemológico del inciso (b)?. La respuesta es ambivalente. Por un lado podemos comprender perfectamente los algoritmos de la D,C-reducibilidad y su finalidad dentro de la demostración. Desgraciadamente la demostración es por casos y como sucedió con la demostración de Kempe, basta con uno falaz para destruir a toda la demostración. Haken y Appel pueden rebatir al escepticismo con base a la flexibilidad de su demostración. Pues si una configuración etiquetada por los programas de Koch como reducible para otro programa con la función de árbitro después de varias corridas no lo es, entonces se puede construir un nuevo conjunto inevitable con mayor probabilidad de que sus configuraciones sean todas reducibles. El lema es potencialmente cierto (si fuera un griego clásico renegaría tanto del infinito como de la validez del lema). La ejecución de la demostración del lema para su entendimiento global es secundaria, el plano de la demostración preside a las huestes de su convencimiento y entendimiento, el plano es *inteligible* pero el proceso de construcción llevado a cabo por los albañiles digitales no es *revisable*. Para alivio de H&A &K esta situación no se presentó, los resultados de los programas independientes coincidieron con los de sus programas. No obstante importa un comino si n programas distintos evalúan de manera positiva al dichoso lema, mientras no pueda ser *revisado* no podrán despejarse a la niebla de la duda. Y cuando hay recelos la comprensión los resiente, la acomplejan con un epíteto no bien recibido en las matemáticas. Si el lema no es *revisable* aunque sí es *inteligible*, la comprensión se ve cercada por el adjetivo "relativa". Ian Stewart, matemático de Warwick mordazmente escribió:

... [la demostración de H&A] no brinda una explicación satisfactoria sobre por qué el teorema es verdadero. Esto es en parte porque la prueba es tan larga que es difícil de asir (incluyendo los cálculos de la computadora,¡imposible!).... La respuesta aparece como un tipo de coincidencia monstruosa. ¿Por qué existe un conjunto inevitable de configuraciones reducibles? La mejor respuesta en la actualidad es: porque sí. La demostración: aquí está, véalo usted mismo. La búsqueda del matemático por estructuras ocultas y su impulso por ligar patrones han sido frustrados.[38]

H&A construyeron un conjunto inevitable de configuraciones reducibles, ergo existe U o ¿acaso no lo ven? Más aún, haciendo uso de herramientas probabilísticas se puede respaldar la existencia de dicho U. Sin embargo mientras la parte de la reducibilidad no pueda inspeccionarse por completo, ¿cómo esperan que veamos si U realmente es el elegido? En

la demostración de H&A los mecanismos probabilísticos no son más que suspiros; la comprensión se mantiene en vilo. ¿Como sortear tal predicamento? Una manera intrépida para hacerlo es elevar a las computadoras hacia la sacrosanta infalibilidad de un matemático.

La segunda oleada del ataque de Tymoczko consiste en desacreditar a las computadoras y de una vez a la imagen idílica de un matemático. Las computadoras en cuanto máquinas cometen fallas ocasionales y mientras más baja sea la calidad del sistema computacional, mayor será su tasa de error. El hardware, el organismo electrónico donde se ejecuta el programa, puede errar por n razones distintas (hasta la radiación cósmica pueden alterar el estado de uno o varios bits). Además como cualquier sistema físico tiene limitantes, las cuales pueden generar errores (v.gr. en los números el redondeo). A nivel software, los códigos de un programa no suelen ser el vil reflejo de los algoritmos, la traducción no esta libre de discrepancias y eventuales equivocaciones. Bajándonos otra vez de nivel, aun cuando el código fuente siga al pie de la letra al algoritmo, el proceso para convertirlo en algo ejecutable por una computadora no esta libre de errores[39]. Para evitar complicaciones atengámonos nada más a la traslación de los algoritmos a un lenguaje de programación. Los programas para evaluar la reducibilidad fueron escritos por Koch en lenguaje ensamblador[40] ya que prefirió optimizar los recursos sacrificando la claridad de la traducción. Ante la enorme cantidad de operaciones a realizar, Koch optó por la eficiencia, su código no tiene la estructura recomendada por el área de la computación especialista en estos litigios: el área de la verificación formal.

La verificación formal es tema del capítulo cuarto, así pues será tratada a ras de piel y la sumersión quedará pendiente. La verificación formal de un programa consiste en demostrar que en todo momento el programa cumple con sus especificaciones, expresado prosaica y deontológicamente, el programa hace lo que debe de hacer. El apelativo “formal” no es fortuito, las demostraciones de la verificación tienen la guisa de pruebas formales (por claridad incurro en pleonasmos). Si los programas de Koch pudieran ser verificados formalmente, entonces las dudas ceñidas sobre el lema de la reducibilidad en gran medida se disiparían. En ese caso la computadora tendría medio pase para ingresar en el paraíso de la infalibilidad. La otra mitad aunque agujereada puede adherírsele, el diseño del hardware y los programas encargados de transformarlo en algo ejecutable por la computadora también son sujetos a una verificación formal. Los hoyos de la realidad impiden la certeza absoluta, ninguna máquina es perfecta. Aunque con el blindaje de la verificación formal y bajo los efectos de un romanticismo que se debate entre lo ideal y lo cínico, la afirmación siguiente retumba en el escrito: sería más fácil cambiar de lugar a los océanos por medio de una concha que descubrirle una impureza al santísimo lema. A pesar de esta garantía, el pesar no recula e impasible discrimina. Las pruebas de la verificación formal son abstrusas y de una longitud indómita. La solución está en el problema, las computadoras son empleadas para construir (automática o automatizadamente) a las pruebas de la verificación. No hay escape, el círculo conduce a ninguna parte.

Las consideraciones del anterior párrafo están ausentes presencialmente en la argumentación de Tymoczko, para él las computadoras fallan y en ese tiempo no había forma de evitarlo:

La tarea de evaluar a los programas es un tema de las ciencias de la computación, pero en el presente no hay métodos generales para lograrlo. Los programas son escritos en "lenguajes" especiales, y muchos de ellos pueden ser bastante complejos. Los programas pueden contener "bugs", o fallas que pueden permanecer inadvertidas durante mucho tiempo. La fiabilidad de cualquier apelación a las computadoras debe descansar finalmente en tales difusas consideraciones*. [41]

Además dichas consideraciones tampoco están presentes en los programas de Koch; funcionan correctamente por obra y gracia del espíritu electrónico y de la acertada traducción de Koch. Swart replica contra Tymoczko y explica las dolencias del programa resanándolas:

Es verdad que los programas a veces tienen "bugs", pero también los tienen los intentos de demostraciones realizados a través del lápiz y el papel. También es verdad, como Tymockzco dice, que las fallas en los programas a veces pueden permanecer durante mucho tiempo inadvertidas. Pero también pueden subsistir las fallas en las demostraciones que tienen nada que ver con las computadoras. La primer "demostración" publicada de la 4CT, por Kempe, era errónea, y no fue hasta después de 10 años que Heawood destapó la falla...

Es verdad que en algunos casos los algoritmos con implementaciones lógicamente correctas que son ellos mismos lógicamente correctos no siempre producen los resultados que deben generar cuando corren en una computadora. Sucede porque la computadora es finita y esto nos obliga a truncar o a redondear a los números irracionales o a los números recurriendo a una representación binaria..... Pero tal situación infeliz no sucede con el algoritmo para probar a la reducibilidad, ya que se limita a efectuar operaciones que involucran enteros o valores booleanos... Aunque el propio algoritmo es muy simple, su implementación en la computadora es cualquier cosa menos trivial; los programas reales para las pruebas fueron escritos en lenguaje ensamblador para hacerlos tan eficaces como fuera posible. Esto levanta problemas para la verificación formal del programa, ya que la verificación formal en la actualidad se aplica a los lenguajes de alto nivel en lugar del lenguaje ensamblador de los programas**. No obstante los resultados de la reducibilidad se han inspeccionado indirectamente de una manera escasamente diferente de las revisiones a mano de otras demostraciones que involucran una gran cantidad de casos a comprobar; a saber, por medio de programas independientes ejecutados en distintas computadoras . De hecho, la prueba por casos de la reducibilidad en el 4CT probablemente se ha inspeccionado con más cuidado que cualquier otra demostración por casos no dependiente de la computadora.* [42]

Como la verificación formal no es aplicable a los programas de Koch, la táctica defensiva torna en ofensiva: la computadora es falible, pero también lo son los humanos. Además los problemas típicos de la representación numérica en una computadora hacen ninguna mella en cuanto sólo enteros son empleados. Más aún los programas evaluadores de la reducibilidad han sido ¿*revisados*? hasta el cansancio por árbitros binarios independientes, mucho más eficaces en este tipo de tarea como lo señalan Haken y Appel que los evaluadores humanos (en el caso en cuestión ni siquiera hay competencia):

Cuando las pruebas son largas y altamente computacionales, puede defenderse que incluso cuando la comprobación a mano es posible, la probabilidad de error en los humanos es considerablemente más alta que la de un error de la máquina.[43]

El desprestigio es una artimaña muy socorrida y a veces efectiva. Sin embargo los argumentos de Tymoczko, contrario a las intenciones de Swart, no son inhabilitados. No hay programas perfectos, o cuando menos, los programas de Koch con seguridad no lo son. Por otro lado el careo hombre-máquina propuesto por Haken y Appel es ridículo, tanto como comparar la velocidad entre un coche, un caballo y un guepardo en una carrera de 100 metros sobre una pista de asfalto. Al defender a la máquina denostando a la bestia humana, Haken, Appel y Swart cooperan con la causa de Tym, pues para él la falibilidad de los matemáticos es una de las cargas filosóficas por eximir[44]. Para conservar a la demostración de H&A&K&1&0 o se ignora a la *revisabilidad* –seppuku- o se aminoran las peticiones del inciso (b) o participamos en el alumbramiento de un nuevo objeto deductivo cuya labor de parto sucede debajo del inquietante árbol de la contingencia. Yo argumentaré a favor de la segunda alternativa, persiguiendo al albor vislumbrando por Tym (quien escoge el destino de Ícaro al desear volar más alto para alcanzar a derretir a las alas de lo a priori). En el horizonte se alza la demostración computarizada, dando a luz a una nueva caracterización de las demostraciones.

Si asumimos crudamente la *revisión* del lema por parte de programas independientes, ¿donde queda el conocimiento? Hábilmente la digresión de Tymoczko parte de su definición de la *revisabilidad*, definición cercana a las prácticas matemáticas usuales. Las máquinas manipulan unos y ceros, no gráficas ni configuraciones ni coloraciones ni nada por el estilo. Tratar de equiparar "las fallas de la implementación computacional con los errores de lógica de un matemático" es una tergiversación acomodaticia de Swart. Más de alguna tribu podrá quejarse, pero la postura dominante en la filosofía de las matemáticas tiene un sabor al dualismo clásico mente/cerebro[45]. No cometemos el error de aceptar la demostración de Kempe porque un cúmulo de neuronas tuvo problemas en las sinapsis debido a la interferencia de las ondas de radio de la Z, en lugar de esta explicación laceramos nuestro orgullo con el látigo de la lógica o con una tralla epistemológica. Ni siquiera los tipos de errores psicológicos (estados depresivos) o fisiológicos (cansancio) son tomados en cuenta, son situaciones ajenas a las matemáticas; como humanos las padecemos pero como filósofos de las matemáticas deben ser dispensadas. En cambio las equivocaciones de una computadora pueden ser explicadas desde los elementos físicos (uno de los circuitos integrados de la ALU se sobrecargó por un pico de corriente) hasta el nivel

software (errores en la programación), incluso la cota de las causas del error puede ser el usuario. Claro, las fallas de la implementación podríamos equipararlas con los errores lógicos cometidos por los matemáticos, pero tales fallas son propiedad de los programadores quienes trasladaron el algoritmo al código y no de las máquinas. Para bien y para mal la epistemología matemática en el instrumento goza de una participación nula, las máquinas *revisan* nada. Son los matemáticos-computólogos quienes perciben el resultado positivo del "experimento" llevado a cabo en la computadora después de una interpretación teórica de los datos arrojados por la máquina. ¿La demostración del lema es algo parecido a un experimento? Para Tym la respuesta es afirmativa, pero antes de proseguir analicemos la opción de la adecuación del inciso (b). Los candados de Tym "*no necesita nada fuera de sí misma para ser convincente" (it needs nothing outside of itself to be convincing)* y "*debe ser verificable a mano... el matemático inspecciona a la demostración en su integridad" (capable of being checked by hand... the mathematician surveys the proof in its entirety)* deben ser transgredidos. Lo *revisable* debe ser sustituido por lo *inteligible*, propiedad definida por el inciso (b) con la supresión del anterior par de exigencias (mañosamente al transcribir por primera vez a (b) no las incluí). Ningún miembro de la comunidad va a inspeccionar por completo el lema, sin embargo puede comprender su validez por la acción concomitante de dos núcleos epistemológicos: el matemático (entendimiento de los algoritmos, etc.) y el computacional (verificación de software, hardware, etc.). El segundo de los anteriores centros, el extranjero, es imprescindible para sostener a la demostración del lema o con menos pretensiones para aumentar nuestra confianza en su validez. Aunque estar versados en computación en sí no impulsa el conocimiento sobre la demostración del teorema, sí es necesario para el entendimiento de este tipo de demostración. Paul Teller no solo demeritó al segundo nodo epistemológico, en pos de salvar el honor de la demostración de H&A&K&1&0 decidió tajar de la caracterización al abultado inciso (b) para después decapitar a la cabeza inspectora del hombre de la *revisabilidad*:

La revisabilidad se necesita, no porque sin ella una demostración no sea en cualquier sentido una demostración (o sólo una demostración en algún nuevo sentido), pero porque sin la revisabilidad nosotros parecemos no ser capaces de verificar si una demostración es correcta. Así que la revisabilidad no es parte de lo que es una demostración. Es una característica que algunas demostraciones tienen y que queremos que nuestras demostraciones tengan para poder asegurarnos razonablemente acerca de lo que nosotros tomamos como una demostración correcta. En particular, podemos aprovechar nuevos métodos de revisión con tal de que éstos nos permitan satisfacer las demandas sensatas de la revisión de las demostraciones, y el cambio en los medios de revisión sólo implica un cambio en los métodos de revisión, no un cambio en nuestra concepción de la demostración.... La novedad está en los medios de la revisión, no en la noción de demostración.[46]

Más allá del espiritismo de las demostraciones fantasmas contadas por Teller(aquellas de las cuales sabemos y no sabemos su cualidad de ser demostraciones), en el más acá existe una situación a favor de la supresión de la *revisabilidad* en la conjunción caracterizadora

de una demostración: la aparición de los oráculos. Tymoczko henchido de una vasta imaginación maliciosa describió la siguiente situación de ciencia ficción:

Supongamos que los avances en las ciencias de la computación nos guían hacia las circunstancias siguientes. Podemos programar una computadora para comenzar una búsqueda a través de varios procedimientos demostrativos, con subprogramas que pueden modificar y combinar los procedimientos en las circunstancias apropiadas, hasta que encuentre una prueba de la proposición A. Después de un largo intervalo de tiempo, la computadora arroja una prueba de A, aunque nosotros no podemos reconstruir la forma general de la prueba más allá de lo mínimo (por ejemplo el esquema de inducción). Quizás nosotros podríamos describir este ejemplo hipotético diciendo que la supercomputadora encontró una prueba asistida por los humanos.... la pregunta es si los matemáticos tendrían la fe suficiente en la fiabilidad de las computadoras para aceptar este resultado.[47]

En los capítulos tercero y cuarto el escenario se preparará para que la profecía sea cumplida; tenía poco de ficción y mucho de científica. La *revisabilidad* se desvanece pero también la *inteligibilidad* tiende a enmudecer. Las demostraciones formales (recordemos a las pruebas del capítulo primero) son un territorio yermo para nuestro entendimiento aunque con el entrenamiento y los medios adecuados algunos desiertos pueden ser irrigados. Para Tymoczko, lo ficticio provenía de la negación de demostraciones con (c) y sin (b), para él las demostraciones preceden a sus pruebas formales, tal como ocurre frecuentemente aunque con el advenimiento de las demostraciones computarizadas exhibidas en el capítulo tercero, la ficción puede ser superada por la realidad. Materialización sencilla de la pesadilla de Tym, por medio de una computadora se puede construir una prueba formal (cuyo teorema será la última fórmula de la secuencia) ilegible para un ser humano por la extenuante longitud de sus fórmulas y el escandaloso número de éstas, sin importar que la fórmula demostrada sea una tontería. Este panorama tiene una ventaja sobre la demostración del lema de la reducibilidad: la contundencia de un sistema formal, las pruebas son la forma más depurada de las demostraciones. Los escrúpulos para el convencimiento disminuyen, aún cuando la confianza en el instrumento siga en la cuerda floja. Las mentes perezosas encontrarán más fácil en este caso el contraste sobre la naturaleza de los errores entre las computadoras y los seres humanos, ya que aquí los fallas en la derivación cometidas por el programa pueden recibir la identificación de "errores lógicos" (sean remitidas a la inatención las causas físicas o de la programación). Esta monstruosa prueba para su validación (¿quién en la comunidad matemática la va a aceptar si no fue sometida a una inspección?) debido a su ilegibilidad al igual que la demostración del lema de reducibilidad requiere de la asistencia de árbitros digitales. Las reglas de inferencia están explicitadas, los programas de ratificación/refutación sólo deben asegurar que la supuesta prueba no los viole, es decir, en este caso los programas si "cuasi*revisan*" a la prueba (prescindiendo claro de las componentes epistemológica y antropológica de la *revisabilidad*). Continuaremos con estas digresiones en el tercer capítulo, por el momento conformémonos con saber sobre la tácita existencia de "demostraciones" *computables* (c) y no *revisables* (b fuerte). La acepción débil de (b) análogamente se despacha, las pruebas aquí barruntadas son in*teligibles*. Las pruebas por definición para ser aceptadas deben ser

leídas por completo. Los saltos y atajos en las demostraciones nos los proporcionaba precisamente su cualidad de ser *inteligibles*, la *inteligibilidad* de las pruebas depende de su legibilidad total. La prueba es *convincente* por motivos lógicos, la forma subyuga al fondo. Basta con darle el visto bueno a las reglas de inferencia y la definición de prueba en el sistema formal en cuestión además de asimilar a la computadora como extensión de un matemático (avalando de este modo al trabajo del autor y los árbitros digitales) para convencerse sobre la validez de nuestra odiosa prueba de ciencia no tan ficticia. Teller escribe:

Si una computadora se programa para usar los mismos métodos de demostración que nosotros usamos, la demostración que produzca será una demostración en nuestro viejo sentido de la palabra. Las preocupaciones legítimas por si un error pudo haberse cometido sólo muestran que puede haber preocupaciones legítimas de que la demostración sea correcta, no de que la demostración sea una demostración. Para establecer el punto en una forma ligeramente diferente, el hecho de que yo no pueda seguir una demostración compleja producida por un matemático bueno no muestra que la demostración compleja de tal matemático sea una demostración en un sentido diferente de la palabra de las demostraciones que yo pueda seguir. [48]

La fe en el instrumento resulta fundamental, debe asemejarse a la fe en nuestros colegas matemáticos (Tymoczko dio en el clavo al escribir "*faith*"). Se puede asegurar la baja probabilidad tanto de un error por parte de la computadora como de una falla de la comunidad matemática, pero donde la probabilidad no es cero intervienen los asuntos de fe. Regresemos a la homilía principal de este apartado, todavía no es el tiempo del calvario. En un principio no importa ni siquiera comprender el contenido de las fórmulas, la interpretación, i.e. una asignación semántica de los objetos de nuestro sistema formal (términos, constantes, letras relacionales, etc.), viene después. Las pruebas no requieren para ser demostraciones de la *inteligibilidad*, sin embargo los matemáticos sí la necesitan para conocer al teorema. La información de la prueba es invaluable, nos dice como se vincula al susodicho teorema con algunos de los axiomas y otros teoremas de la teoría. Lo epistemológico en los sistemas interviene en la construcción de éstos (en la elaboración de la base axiomática, en la selección de letras funcionales, etc). A otro nivel lo epistemológico también actúa en la comprensión de las características (lógicas) del sistema formal en cuestión, v.gr.¿el sistema es completo? Cesen los desvíos, la discusión se centra en la caracterización de una demostración. Concentrémonos en la existencia de x demostración tal que x es *computable* y x es in*inteligible.* En el capítulo tercero se explicarán algunas variantes para arreglar la in*inteligibilidad*, en pos de salvar el derrotero de Tym: el conocimiento de un teorema es conducido por la verificación (lógica y epistemológica) de éste.

¿La computadora como una extensión de un matemático? Es la alternativa pregonada por Teller e incluso por Swart. El instrumento es secundario, la computadora genera tanto ruido como la tiza y el pizarrón. ¿Por qué escamarse si el ábaco en lugar de ser operado por un matemático es operado por un niño? ¿Y si al niño lo sustituimos por una computadora?

Las operaciones son tan simples que hasta una computadora puede realizarlas, es más, es el agente hasta ahora con menor probabilidad de errar en este tipo de tareas cuando su número es gigantesco. Al menguar la inquietud sobre la acción de la computadora, tanto para Swart como para Teller ya no cabe ninguna nueva definición de demostración. Es una típica demostración por casos y se acabó. Si alguien todavía osa dudarlo puede llevar a cabo la siguiente empresa: contrate a 10 millones de chinos y entrénelos para demostrar obedeciendo las tácticas de Haken y Appel al lema de reducibilidad. Ahora contrate a otros veinte millones para formar un par de equipos de inspección. Aunque en China la esclavitud es asistida por el estado, la empresa tal vez fracase por presiones económicas o sociales. Seamos ingenuos y optimistas, supongamos su éxito después de unos cuantos años. Los tres equipos coinciden en los resultados, todas las configuraciones del conjunto inevitable son reducibles. Las cajas mágicas digitales ya no expelen los resultados, son millones de hombres diligentemente capacitados quienes la respaldan. ¿Seguimos ondeando las pirrónicas banderas o por fin acariciamos al consenso con la bandera blanca de la aceptación? Siempre hay salidas para el escepticismo, es más, el escepticismo es siempre una salida. La tormenta de la duda puede azotar a los elementos sociales de nuestra nueva demostración del lema. Podemos renegar de la capacitación (equivalente a la implementación del algoritmo), de la calidad de nuestros chinitos (equivalente al hardware, por ejemplo la desnutrición de nuestros esclavos puede ocasionar fallas) o hasta del sistema político de China. La demostración sigue sin ser *revisable* ya que no preparamos a cada uno de nuestros chinitos para ser matemáticos y estar calificados para *revisar* su labor, sólo se les enseña lo básico, ellos obedecen o son enviados a campos de adoctrinamiento. Entonces siguiendo los pasos de Tymoczko, ¿debemos llamar a esta demostración un "experimento" social? ¿Constituye también un nuevo tipo de demostración? Al menos conserva una similitud clave con la demostración computarizada, independientemente de la confiabilidad de los ejecutores para obtener los resultados se requiere forzosamente la interacción con el mundo material. Tym no ataca a la computadora por sí misma sino a la necesidad de utilizar un sistema (social-electrónico) dependiente de las contingencias de la realidad, arremete contra la filosofía clásica de las matemáticas y su apoteosis de la mente con acceso directo al Valhalla de las ideas. En concreto atenta contra la sacralización el conocimiento de los teoremas, profana al hierático *a priori* con el errático *a posteriori.*

Empecemos con la definición de Tymoczko:

Tradicionalmente, las verdades a priori son aquellas que pueden conocerse por la razón independientemente de cualquier experiencia más allá del pensamiento puro. [49]

Con su definición podemos anticipar su conclusión: como la demostración del lema de reducibilidad no es *revisable*, la razón pura sufre un atropello y paralítica emplea a la silla de ruedas digital. Entonces si conocemos la verdad del lema, ésta no puede ser *a priori.* Por otro lado, aunque la demostración del lema sea *inteligible*, la comprensión va a estar supeditada a la interacción con un sistema físico. Sea K el agente racional quien sabe lo suficiente de teoría de gráficas para programar correctamente los algoritmos e interpretar los resultados arrojados por los sus programas inspectores de reducibilidad. Además K sabe

lo necesario en el área de la computación para tener plena confianza en sus programas y en el instrumento ejecutor. Es decir, existe un agente racional K quien realiza un "experimento" en el instrumento computadora y recopila el resultado:"el lema de reducibilidad es válido". El supuesto cambio en la *revisabilidad* propuesto por Teller es un taimado fraude. Los programas independientes revisan-comprenden nada. En primer lugar los programas independientes de la reducibilidad son sólo una repetición del "experimento" llevada a cabo por otros agentes racionales. Por ejemplo, Allaire con la misma base teórica (Keesch) preparó el "experimento" a su manera (i.e. escribió sus programas) y ¡oh sorpresa! obtuvo resultados análogos. Siempre puede haber discrepancias en los resultados, como bien lo saben aquellos enfrascados en las batallas experimentales. Sin embargo mientras mejor estén diseñados los experimentos (programación de los algoritmos) y el efecto de las contingencias físicas sea reducido (v.gr. fallas de hardware) entonces los resultados con mayor probabilidad se corresponderán. Conclusión, los programas independientes no revisan el lema, tampoco lo comprenden. Lo inteligible es captado por los experimentadores, no por el "experimento". Situación distinta la ofrecería la verificación formal de los programas de reducibilidad. Si existiera un programa P capaz de demostrar que la implementación de los algoritmos es correcta entonces P "cuasi*revisaría*" indirectamente a la demostración del lema. Recordemos, en la "cuasi*revisión*" hemos desvencijado casi por completo al inciso (b) haciéndolo colindar con el (c). El agente P (no lo llamo racional para evitar complicaciones innecesarias) examina por completo al programa, por medio de una asignación semántica del código (capítulo cuarto) prueba que en todo momento el programa cumple con sus objetivos. Si los resultados de los "experimentos" son de antemano asegurados deductivamente, entonces podría descascararse el adjetivo "experimental". Pero como ya se mencionó, el círculo erosiona toda esperanza de fuga hacia el Valhalla pues P casi con seguridad será un programa y también debe ser verificado. En algún punto se debe parar la exigencia de la verificación, concediéndonos un grado de confianza muy alto aunque no absoluto.
Por otro lado cuando se delinearon a las pruebas con (c) y sin (b), se denostaron por su ausencia de *inteligibilidad*, en ellas nuestra vanidosa razón nada entre la fe bien fundamentada y la incredulidad abyecta. En los dos escenarios, el real y el hipotético, nuestra mente inevitablemente debe ser asistida por un sistema físico, una extensión llave de nuevas puertas de conocimiento. Si no son verdades a priori bajo la definición (49), entonces pensando en blanco y negro, ¿son a posteriori? ¿Corresponde entonces escribir "experimento" sin las comillas? Tymoczko siguió esa ruta accidentada e ipso facto lo bombardearon Swart y Teller. Inicio con la réplica menos explosiva, lanzadas desde el B-52 Teller:

... nos hemos anticipado a la ayuda futura de las computadoras en el descubrimiento de nuevas demostraciones con el uso de programas que incluyen métodos demostrativos por nosotros usados. Pero esto de ninguna manera desencadena un cambio en los fundamentos epistemológicos de las matemáticas, como tampoco el desarrollo de nuevos maneras de construir microscopios electrónicos trae consigo un cambio en las fundamentos epistemológicos de las ciencias naturales. [50]

Cuando los microscopios asuman un papel tan activo como el de la computadora en la demostración del lema de la reducibilidad, entonces temblará la epistemología de las ciencias naturales y lloverán ranas. Adaptemos la situación descrita por Teller de manera más acertada; las mejores progresivas de los ábacos en nada afectan a los fundamentos epistemológicos de la aritmética.

Teller apunta con más tino hacia la dirección de su defensa disparando las siguientes palabras:

Un experimento en las ciencias naturales determina un hecho espaciotemporal, que entonces puede o no puede generalizarse más allá del lugar y tiempo locales donde y cuando se aplica directamente. Una prueba matemática correcta establece un tipo de hecho que no es espaciotemporal. [51]

La inmutabilidad de los objetos y teoremas matemáticos sería una evidencia de su dimensión espacio-temporal nula. Cambio implica tiempo. Si el concepto número ha variado desde el periodo griego clásico hasta nuestros días, ¿entonces el ente matemático "número" no es eterno ni inalterable? El conocido caso del teorema de Cauchy(Euler[Descartes])-utilizado en la demostración de Kempe y retomado por la de H&A&K&1&0- y sus constantes transformaciones sagazmente fue documentado por Lakatos en su bestteller "Pruebas y refutaciones". Aunque sean disuasivas las lecciones de historia, son todo menos conclusivas. Además las saetas discursivas de Teller apuntan hacia otro blanco. Los experimentos en las ciencias naturales manipulan objetos atados al espacio-tiempo donde-cuando se realizan, es decir, carecen de independencia de la realidad adscrita. En el sentido común son verdades (si lo son) *a posteriori*. Objeción velada, ¿los físicos acaso no experimentan con abstracciones? La masa, velocidad, etc. son propiedades mentales atribuidas a los objetos materiales, tampoco gozan de una existencia en el espacio-tiempo. De una bala en movimiento no se deriva la existencia de su velocidad a menos de que un agente racional le adose esa propiedad, así Arquímedes Newton –agitador social imaginario- previo a su fusilamiento calcula la velocidad de las balas para soportar los segundos previos antes de su ajusticiamiento, sus últimas palabras serán "las balas tienen una velocidad x". Nuestro agente K interpretaba en la teoría de gráficas los resultados arrojados por el sistema electrónico, así como también Arquímedes Newton interpreta su fusilamiento con la cinemática. Para Teller, Swart y casi cualquier filósofo hay una marcada diferencia entre el "experimento" de K y el experimento de un físico. Swart expresa directamente el mensaje entrelíneas de las tres líneas previas de Teller:

Una verdad a priori es una verdad que posee validez universal y necesaria; es decir, una verdad que es verdadera en los todos los mundos posibles. [52]

La demostración de la 4CC, asentida por la comunidad matemática, debe establecer una verdad a priori bajo la filosofía clásica de las matemáticas. Los mapas a primera vista parecen objetos empíricos aunque pueden ser definidos mediante un lenguaje matemático. Para evitar desorientaciones enfoquémonos en la versión del teorema sobre gráficas planas.

Si los vértices de una gráfica plana pueden colorearse propiamente a lo sumo con cuatro colores, aceptando el carácter necesario de las verdades matemáticas debemos concluir como Tymoczko[53] que el Teorema de los Cuatro Colores encierra una propiedad esencial de las gráficas planas. De acuerdo a Swart la confusión de Tym estriba en cómo desnudamos (al menos en las matemáticas) a las verdades a priori:

Si para develar su verdad, es necesario usar lápiz y papel o una calculadora o incluso una computadora, esto en nada afecta a la naturaleza de su verdad. [54]

Consciente de la definición inapelable de Tymoczko de las verdades a priori (aquellas que pueden ser establecidas sin el uso de la experiencia empírica de nuestro mundo, i.e. las hijas de la razón pura), Swart la redefine conforme a su postura y prosigue con su crítica:

[una verdad a priori] es una verdad cuya validez puede en principio establecerse sin recurrir a la experiencia sensible del mundo físico. [55]

La anterior definición se atiene a su definición de *a priori* previamente transcrita(52) y Swart las enlaza con las siguientes palabras:

... una verdad a priori es aquella que puede descubrirse por un ser sensible, con un cerebro lo suficientemente grande para tal propósito, sin recurrir a cualquier experimento. Pero lo anterior no implica que un tipo particular de seres sensibles que por algún motivo se irguieron de la tierra, los cuales escogimos llamar Homo sapiens, no puedan necesitar acudir a un experimento para saber la verdad sobre una verdad a priori específica. [56]

Maniobra astuta, en lugar de perder el tiempo y de paso perder la discusión en la distinción sobre si la demostración del lema es o no es un experimento mejor afirma que los medios de adquisición no equivalen al tipo de verdad adquirida. En contra de nuestra argumentación basada en la ausencia de *revisabilidad* del lema, si nuestras capacidades mentales son limitadas y deben ser asistidas por un instrumento no implica la expulsión de la categoría "a priori" del lema de la reducibilidad. Swart tiene a su favor la opinión de algunos matemáticos (entre ellos claro Haken y Appel) acerca de la inexistencia de una demostración no computarizada del Teorema de los Cuatro Colores[57]. Tymoczko confiesa no ser un asesino, en el ruedo capotea pero no mata y en su ensayo sobre el problema de los cuatro colores precisa:

... esta investigación debe ser completamente filosófica, ya que la pregunta matemática puede considerarse definitivamente resuelta. No es mi objetivo interferir con los derechos de los matemáticos para determinar lo que es y lo que no es un teorema. Yo sugeriré, sin embargo, que si aceptamos al 4CT estamos comprometidos a cambiar... el sentido del concepto subyacente de "demostración". [58]

La Conjetura de los Cuatro Colores es un teorema, su demostración es válida. Ni se puede nadar a contracorriente con aletas filosóficas rechazando al 4CT ni conviene hacerla de

Adán nombrando nuevas criaturas del universo matemático[59] pues primero se debe clarificar a las clasificaciones existentes. Enfoquémonos en la pregunta guía de los últimos párrafos, la inquietud ceñida sobre el tipo de verdad del lema. Swart no la responde, sólo la traslada a regiones más tradicionales de la filosofía de las matemáticas, la transporta al reino de lo a priori en cuanto a la supuesta cualidad de los teoremas matemáticos de ser verdaderos en todos los mundos posibles. En la anterior afirmación me parece se esconde un platonismo recalcitrante pues afirma que las posibilidades de la materia en nada influyen sobre la inmutabilidad de las ideas. Por precaución mi incursión por esos lugares filosóficos será corta y no resolutoria. En primer lugar, ¿qué sentido tiene postular verdades a priori inaprensibles al intelecto? Desde un punto de vista plenamente filosófico, su importancia es incuestionable. Por ejemplo, serviría para desinflar de una vez por todas a nuestras vanas presunciones y vanidosas aseveraciones sobre nuestras capacidades mentales, al reventarlas con los supuestos alfileres de las verdades fuera de nuestro alcance. Por otro lado también obligaría a atenuar la inferencia inmediata "son aquellas verdades hijas de la razón pura" derivada de su característica de ser necesariamente verdaderas e independientes de la experiencia sensible. La razón deberá mezclarse y fortalecerse con sistemas físicos para acceder a ciertas verdades a priori vedadas a nuestra devaluada mente. Sin embargo hay un punto intrincado hasta ahora no mencionado, ¿cómo saber si son independientes del mundo sensible si la única forma durante un tiempo indefinido de descubrirlas es por medio de la interacción con la realidad material? La denominación a priori parece en este caso más hereditaria que necesaria, i.e. como todos los teoremas matemáticos anteriores a este bicho raro eran considerados verdades a priori (sean cual sean las razones) entonces la 4CC al haber sido declarada teorema también debe ser una verdad a priori. O procediendo de una manera más rigorosa, debería someterse a revisión la clasificación "a priori" de las verdades reveladas por los teoremas matemáticos como sucedió con la noción de la demostración de éstos. Ineludiblemente en nuestro análisis debe incluirse una componente empírica para poder abarcar a la demostración del lema de la reducibilidad y no será el único con la necesidad de ese manto. En el sentido inverso, si consideramos como a posteriori el conocimiento del Teorema de los Cuatro Colores debido a la dependencia de nuestras demostraciones con el mundo físico, entonces la nefasta posibilidad de la aparición de una demostración normal (i.e. una demostración *revisable*) podría invalidar a nuestra clasificación. En el 2004 salió a la luz una aspirante a demostración normal del 4CT, obra de I. Cahit, poseedora de atributos favorables para ser convincente[60]. Aunque la historia tapizada de ideas novedosas y sonados fracasos de la 4CC nos ha enseñado a ser poco complacientes, no importa si la demostración de Cahit traerá o no la salvación y reivindicación de la ortodoxia, las vías usuales no pueden ser declaradas como clausuradas (al menos no en este caso). El apelativo "a posteriori" de la verdad del 4CT pende de un hilo. La clasificación del 4TC como "a priori" tampoco ofrece mayor alivio. La flexibilidad del lenguaje acude a nuestro rescate, cuando las palabras no se ajustan a nuestras expectativas podemos crear un nuevo término. Podemos llamar "cuasiempírica" a la verdad desprendida de la demostración computarizada de H&A. Conservemos las comillas, es una demostración "experimental". Acuñemos sus posibles y necesarias definiciones: una verdad "cuasiempírica" es una verdad necesaria a posteriori –denominación alegada por Tymoczko[61]- o una verdad a priori contingente. Sin duda he

incurrido en una licencia literaria asaz inútil, el híbrido contradictorio de este bellaco nominalismo satisface a quienes de antemano les importaba poco las distinciones rígidas. Bajo esta tónica, el lema de la reducibilidad puede ser utilizado como un argumento contra la ¿espuria e impositiva? división entre lo a priori y lo a posteriori; sin embargo aunque es una mecha atractiva nos aleja de los objetivos pactados. El tema principal es la demostración matemática, no el tipo de verdad generadas por ellas. Ya es tiempo de cerrar este capítulo.

La demostración computarizada del lema de reducibilidad provocó un desbarajuste en la caracterización usual de las demostraciones. Mañosamente quien preconizó un arreglo del desajuste fue quien también proporcio nó la caracterización estándar. Sin embargo recoge las ideas tradicionales manejadas por la filosofía de las matemáticas, en ningún momento sus detractores (Teller, Swart) renegaron sobre los incisos canónicos. La propiedad de una demostración de ser *revisable* (i.e. puede ser inspeccionada por completo a través de lentes epistemológicos por algun(os) miembros de la comunidad matemática y comprendida con base exclusivamente de su contenido matemático) no es aplicable a la demostración de H&A&K&1&0, en su lugar debe tomarse en cuenta la propiedad de ser *inteligible* (puede ser inspeccionada por algún agente, su contenido epistemológico puede ser succionado y pueden intervenir otras fuentes epistemológicas además de la matemática). Pero para ser *inteligible* la demostración de H&A&K&1&0, la computadora debe ser aceptada en nuestra biblioteca de instrumentos y procedimientos demostrativos. La nueva caracterización está bien definida al conservar y ampliar el alcance de la primera, pues si una demostración es *revisable* entonces es *inteligible*, pero por su mayor especificación cuando se estudien a las demostraciones ortodoxas conviene más utilizar a la propiedad original de la *revisabilidad*. Como avance del capítulo tercero y continuación del primero se señaló la existencia de demostraciones *computables* e in*inteligibles*. Por otro lado en un sentido estricto si la demostración es *inteligible* entonces se puede construir un sistema formal a la medida para que sea *computable*. Sin embargo esta artimaña carece de valor pues en las matemáticas trabajamos con teorías, por eso la búsqueda –base axiomática, reglas de inferencia- y el correcto diseño –debe obedecer restricciones lógicas- del sistema formal de donde se pueda derivar el teorema bajo lupa y a los teoremas importantes de la teoría aunado a la de por si difícil tarea de construir la prueba hace del trabajo un reto atroz. No obstante siguiendo las enseñanzas de Wang las demostraciones *inteligibles* se deben convertir en *computables*, bienvenida sea la ayuda de donde provenga. En pos de hacer *computable* a la demostración de H&A&K&1&0, las siguientes indicaciones según Swart deben acatarse:

1. *El procedimiento de descarga y la creación resultante del conjunto inevitable de configuraciones debería ser computarizado.*
2. *Los programas para ambos tareas, la descarga y las pruebas de reducibilidad deberían escribirse en un lenguaje de alto nivel y adecuadamente verificarse para asegurar que ellos llevan a cabo los algoritmos que ellos pretenden llevar a cabo.* [62]

La demostración de Thomas, Seymour, Sanders y Robertson siguen el par de lineamientos de Swart, la construcción del conjunto inevitable fue computarizada y ambas acciones

digitales –la verificación de la reducibilidad y la edificación del conjunto inevitable- fueron escritas en un lenguaje de alto nivel (lenguaje C). La verificación formal de los programas sigue pendiente, sus resultados también se avalaron por medio de programas independientes. Sin embargo, el trabajo de Thomas et al hace lucir como viable la formalización a la manera sugerida por Swart–una formalización digital- de las demostraciones de la 4CC al estilo H&A&K&1&0. La fe en la *computabilidad* de la demostración de H&A&K&1&0 está bien sustentada, siempre y cuando confiemos en el instrumento computadora.

Recapitulando, en la caracterización propuesta por Tymoczko se intercambiaron *revisable* por *inteligible* y *formalizable* por *computable*, además la conjunción "si x es una demostración entonces x es (a)*convincente* y x es (b)*inteligible* y x es (c)*computable*" es insostenible ante la existencia de demostraciones in*inteligibles* y *computables*. Indistintamente a la anterior invalidación, la propiedad de ser *formalizable* exige una alta cuota de fe si no nos conformamos con formalizaciones vanas. Ahora bien, ¿toda demostración debe ser *convincente*? Mientras nos interese reflejar el desarrollo matemático humano, la respuesta es afirmativa. Sin embargo para satisfacer a quienes su ímpetu los arroja hasta los santuarios filosóficos libres de la contaminación social/psicológica, opto por la flexibilidad de la caracterización disyuntiva: "si x es una demostración entonces x es (a)*convincente* o x es (b)*inteligible* o x es (c)*computable*". Ahora bien, al explicar Tym al *convencimiento* con base a la conjunción de las otras dos propiedades da un paso arriesgado afín a su danza corrosiva contra lo provecto en la filosofía de las matemáticas pero quien mete el pie factual a semejante traba será sometido y la realidad lo hará caer de bruces, pues reducir el *convencimiento* enteramente a cuestiones lógico-matemáticas no baila con todo el proceder matemático (este tropezón será discutido en el capítulo tercero). No obstante sin la pretensión por la doble implicación, aspiración común de toda caracterización, el *convencimiento* de una demostración implica al menos una de las otras dos propiedades, i.e. si x es *convincente* entonces x es *inteligible* o x es *computable*. De no ser así, ¿de que estaríamos hablando entonces? Con respecto a la alborotadora demostración de H&A&K&1&0 culpable de los filosóficos quebrantos, han sido expuestos su *convencimiento* a pesar de la ausencia de su *revisabilidad* y su *computabilidad* con todo y la falta de su total *formalización*. De este modo, la demostración de H&A&K&1&0 es (a)*convincente*, (b)*inteligible* y (c)*computable*.

Al tipo de demostración por casos asistida por la computadora lo denotaré con el nombre de demostración computarizada-exhaustiva. La demostraciones del 4CT de H&A&K&1&0, Allaire y Thomas et al no son los únicos miembros del anterior club. Por citar otra en 1988 se unió a la fiesta la demostración de Clement Lam acerca de la inexistencia de un plano proyectivo de orden diez[63].

Por último, emitiré un par de observaciones vendadas hasta el final de esta sección; fueron cubiertas para eludir distracciones antes inoportunas. Una ya había sido mencionada en este capítulo, aunque vale la pena ser retomada. El inciso (a) huele a sociología del conocimiento versión ultraligera, ese olor es rebajado por los incisos (b) y (c) pero la

fragancia no se evapora totalmente. Las demostraciones comunican algo y ese algo a veces sólo puede ser descifrado por un pequeño grupo de especialistas; sin embargo la comunidad matemática confía en ellos y absorbe su trabajo después del periodo de validación de manera incondicional, el convencimiento es fomentado por la creencia en la solidez de la estructura social matemática. Tymoczko sugiere a los interesados en la filosofía de las matemáticas:

Borra la imagen de un matemático aislado y admite que el conocimiento matemático es el conocimiento público de una comunidad de matemáticos. Reconociendo esta comunidad nos ayudará a reconocer la comunicación entre matemáticos que constituye el hacer matemático y cimienta al conocimiento matemático. [64]

Según Tymoczko para conocer un teorema se debe verificar su demostración (pregunta sin premio al lector: ¿"conocer es demostrar" se la atribuye A?). El proceso de verificación se lleva a cabo en un par de niveles, el lógico y el epistemológico. Sin embargo, ¿es la demostración la única vía para el conocimiento de un teorema? Respondo con un ejemplo, luego profundizaremos. La demostración de Thomas et al del Teorema de los Cuatro Colores estaba espoleada no sólo por mejorar la demostración de H&A&K&1&0. Les interesaba verificar esa demostración porque lograron relacionar a la 4CC con otra conjetura sobre coloración en la teoría de gráficas, la Conjetura de Hadwiger[65]. Debido al pesado y poco prometedor panorama optaron por la reelaboración de la demostración:

De hecho tratamos de verificar la demostración de Appel y Haken, pero pronto nos rendimos. Verificar la parte de la computadora no sólo requeriría mucha programación, sino también el ingreso de las descripciones de 1476 gráficas, y ésa ni siquiera es la parte más polémica de la demostración [procedimiento de descarga]. Entonces decidimos que sería más provechoso elaborar nuestra propia demostración. [66]

Conozco una palabra si entiendo su definición localizada en un diccionario. Pero, ¿no la conozco también cuando la empleo correctamente en un enunciado? El conocimiento no se limita a la compre(n)sión, debe abarcar a la operación inversa, a la extensión.

Me despido con una frase de Tymoczko y conmino a los lectores a regresarse hasta la cita de Hilbert localizada al principio de la tesis:

Lo que hace del uso de las computadoras en el 4CT tan dramático es que lleva a una extensión genuina de nuestro conocimiento de las matemáticas puras. No son meramente cálculos, se ha producido una demostración de un nuevo resultado sustancial.[67]

POSTDATA

La etiqueta “experimental” atribuida a la demostración computarizada-exhaustiva de 4CT puede ser pegada debido a las características de la estrategia planeada y seguida por H&A. Imaginemos a un biólogo cuya meta es corroborar la siguiente aseveración: todos los cuervos del zoológico de Chapultepec son negros. El biólogo recurre a la experiencia sensible y verifica la población del zoológico para confirmar su hipótesis. Haken y Appel proceden de manera análoga, metafóricamente revisan todas las plumas de cada uno de los cuervos de su conjunto inevitable para verificar su fusco color. Ahora bien, hasta aquí aunque la etiqueta “experimental” ya se ha recubierto con algunas capas de pegamento, tampoco son las suficientes para fijar definitivamente a la etiqueta en la demostración. Otro caso, imaginemos que la demostración del teorema de Nozdriov en la teoría de números nos remite a verificar que todos los números pertenecientes a un conjunto finito S son primos. Además como dichos números son tan grandes la asistencia de las computadoras es más que bienvenida. Sin embargo a diferencia del conjunto U de H&A, la construcción del conjunto S fue llevada a cabo mediante procedimientos deductivos, v.gr. por medio del delineamiento de una de las estructuras ocultas del teorema en cuestión. Entonces la demostración del teorema de Nozdriov, aunque parecida a la demostración computarizada-exhaustiva de H&A, tiene vínculos más fuertes con las demostraciones usuales de las matemáticas. Más allá del empleo polémico de las computadoras en la demostración de ambas, los métodos de creación de los conjuntos U y S son tan diametralmente opuestos como para considerar a dichas demostraciones miembros de distintas fa milias. En contra de la postura tradicionalista de la filosofía de las matemáticas, que busca separar al proceso de la manufactura del producto terminado (menospreciando al primero en favor del segundo), la demostración de H&A temeraria atenta contra este estado de la balanza. ¿Por qué en el último de los casos la demostración de H&A es “experimental”? Porque la construcción del conjunto U fue, es y seguirá siendo hasta que alguien se digne a revertir al flujo de la historia, “experimental”. La miasma de palabras previas a la sección dedicada al impacto filosófico de la demostración computarizada-exhaustiva, para los no interesados palabras hediondas, resulta necesaria. Me parece pertinente hacérselo notar al lector no gustoso de la teoría de gráficas al final de su suplicio.

NOTAS

(1) Bondy J.A. y Morty U.S., *Graph theory with applications,* North-Holland, 1976

(2) Saaty, Thomas y Kainen, Paul , *The four-color problem.* Dover, 1977

(3) Kempe, A *"On the Geographical Problem of Four-Colors."* Amer. J. Math. 2, págs.193-200, 1879

(4)Appel, Kenneth y Wolfgang, Haken *"The Four-Color Problem"*, Mathematics Today, ed. L.A. Stern , Springer 1978

(5) Kempe, p.195 de (3)
If A and C belong to different green and blue region , interchanging the colours in either, A and C become both green or both blue. If A and C belong to the same green and blue region, see if A and D belong to different yellow and blue regions; if they do, interchanging the colours in either region, A and D become both yellow or both blue. If A and C belong to the same green and blue region, and A and D belong to the same yellow and blue region, the two regions cut off B from E, so that the red and yellow region to which B belongs is different from that to which D and E belong, and the red and green region to which E belong is different form that to which B and C belong. Thus, interchanging the colours in the red and yellow region to which B belongs, and in the red and green region to which E belongs, B becomes yellow and E green , A, C and D remaining unchanged . In each of the three cases the number of colours is reduced to three

(6) J. Short, editor del naciente *American Journal of Mathematics* (fundado en 1878) en 1879 no sólo avaló a la demostración de Kempe sino le hizo unos agregados. El matemático Charles Sanders Peirce en 1880 refinó la demostración. La Sociedad Matemática de Londres le dio también el visto bueno. El problema de los cuatro colores después de 26 años parecía haber sido resuelto.

(7) Cualquier curva simple cerrada en un plano lo divide en dos regiones conexas ajenas. Uniendo a la AC cadena el país Px se forma una "curva simple cerrada". Luego la disyunción exclusiva acaece: o B está en el interior o D está en el interior de esa "curva".

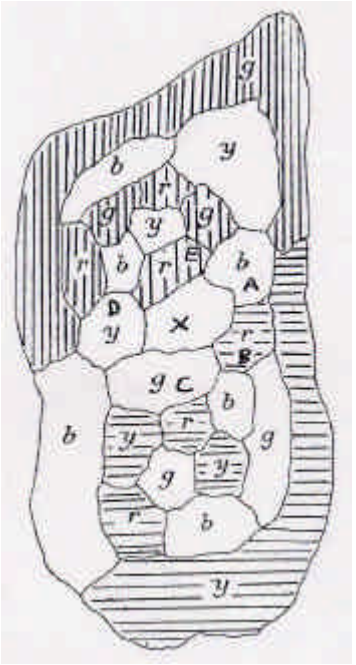
(8) Heawod, P.J., *Map-Color Theorem,* Quaterly Journal of Mathematics 24 , 1890, págs. 332-339.

La cita original etiqueta a los países con números:

Mr. Kempe says-the transposition of colours throughout 5's red-green and 2's red-yellow regions will each remove a red, and what is required is done-. If this were so, it would at once lead to a proof of the proposition in question.... But unfortunately, it is conceivable

that though either transposition would remove a red, both may not remove both reds. pág. 337-338

(9)Heawood culmina la demolición con su contrajemplo (cambie de nuevo los números por letras) encontrado en la página 338 de (8):



Las regiones r-g y r-y son adyacentes, por lo cual no se liberan los dos rojos.

(10) En 1859/1860 Karl Theodor Wilhelm Weierstrass por vez primera abordó los fundamentos del análisis, haciendo hincapié en el rigor. Esta exigencia en 1861 lo llevaría a descubrir una función continua aunque sin derivada en todos los puntos de su dominio, asestando un duro golpe contra la intuición. En el ocaso decimonónico y los albores del siglo XX en la tarea por fundamentar a las matemáticas se le otorgó a la lógica un papel principal dentro de la obra, recuperando su importancia en las demostraciones sin agobiarlas por completo.

(11) Sea G minimal y G no triangular. Entonces existe una región R para la cual existen más de tres aristas, por lo cual existe un par de vértices $x,w \in R$ tal que no son adyacentes. Agreguemos a G la {x.w} arista y contraigamos ese par de vértices (hagamos de los dos uno conservando todas las adyacencias de ambos). Entonces esta nueva gráfica tiene menos vértices que G, por lo que tiene una cuatro-coloración C. Fácilmente se puede extender esa coloración a G asignándole a x y a w el color del vértice contraído.

(12) Sea p es el número de vértices y q el de aristas de G ($|V(G)| = p$, $|A(G)| = q$) Como resultado básico de la Teoría de Gráficas tenemos:

$$\sum_{v \in V(G)} \delta_G(v) = 2q$$

Entonces:

$$\sum_{v \in V(G)} q\,(v) = \sum_{v \in V(G)} 6 - \delta_G(v) = 6p - 2q$$

Como G es triangular como corolario del T. de Cauchy (Euler[Descartes]) tenemos:
$q = 3p - 6$

Entonces $12 = 6p - 2q = \sum_{v \in V(G)} q\,(v)$

(13) El siguiente llano algoritmo de descarga fue extraído de (4):

> Si v es un vértice de grado cinco transferir 1/5 de su carga a todo vértice con carga negativa (i.e. toda vértice con grado mayor a seis) .

Después de aplicar el algoritmo a la carga inicial la presencia inevitable de un vértice con carga positiva implica que la triangulación debe contener:

Una configuración de dos vértices de grado cinco adyacentes

o

Una configuración de un vértice de grado cinco adyacente a uno de grado seis

Se puede justificar la inevitable presencia de este par de configuraciones barriendo las posibilidades o revisando (2); la elección la dejo al gusto y estado de ánimo del lector. Así hemos construido un conjunto U para el algoritmo propuesto.

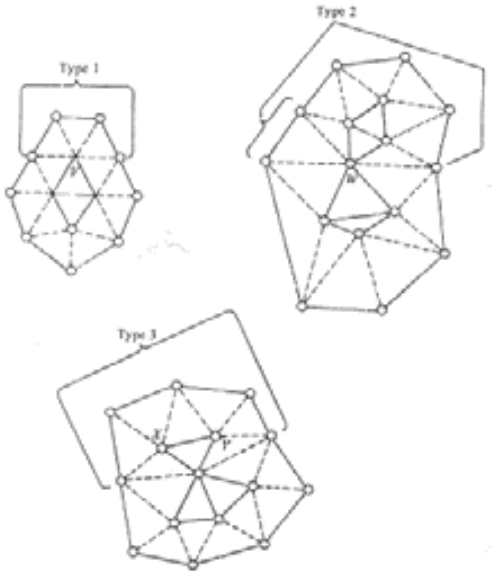
(14) El Método de Arquímedes talvez sea el primer caso documentado de la materialización abstracta de entidades matemáticas. Arquímedes le otorgo cualidades físicas a algunos objetos geométricos y con base a principios mecánicos dedujo y *"demostró"* varias proposiciones con relación a áreas y volúmenes de éstos.

(15) Saaty y Kainen, *The four-color problem*. pág 75 de (2).

(16) *the man-machine dialogue,* pp 173-174 de (4)

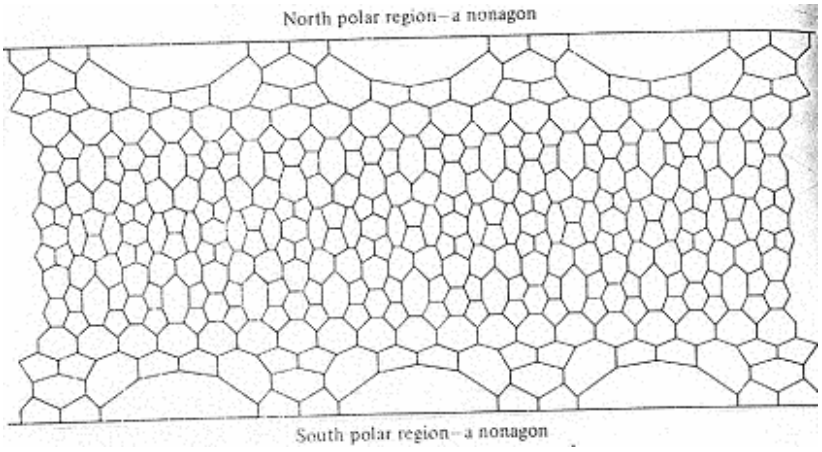
(17) Casi simultáneamente Walter Stromquist (*Some Aspects of the Four Color Problem,* 1975) también demostró la existencia de un conjunto inevitable de configuraciones geográficamente buenas *in a rather elegant way* . pág 174 de (4)

(18) Aunque no hay una demostración sobre la imposibilidad de una configuración reducible que contenga a los obstáculos, hasta el momento ninguna conocida ha podido ser reducida. La siguiente figura muestra los tres obstáculos encontrados por Heesch:



(19) *At this point the program, which had now absorbed our ideas and improvements for two years, began to surprise us. At the beginning we would check its arguments by hand so we could always predict the course it would follow in any situation; but now it suddenly started to act like a chess-playing machines...it began to teach us things about how to proceed that we never expected. In a sense it had surpassed its creators in some aspects of the "intellectual" as well as the mechanical parts of the task*.
página 175 de (4)

(20) Edward Moore, de la Univesidad de Wisconsin, construyó el mapa sin configuraciones n-anulares reducibles si $n<11$.



En el sentido contrario, la existencia de estos pérfidos mapas sugieren que todos los caminos trazados por las ideas de Heesch para llegar a la demostración de la conjetura de los cuatro colores conducen a la computadora.

(21) Saaty y Kainen, página 84 de (2)

The probability arguments are admittedly crude and make several unjustified assumption. ... Perhaps the best analogy is with the intuition which is required to obtain a new experimental result. Once the result has been found, one no longer needs to defend the likelihood that the experiment will succeed in demonstrating it. Since the "experiment" which Appel and Haken wished to perform would take years of valuable computer time, not to mention their own time and effort, even a crude argument was better than none.

(22) *In retrospect, however, we know that they [the probability arguments] are valid in that they lead to a mathematically correct proof*
Saaty y Kainen, página 84 de (2)

(23) K. Appel and W. Haken, *Every planar map is four colorable*, Contemp. Math. 98, 1989

This leaves the reader to face 50 pages containing text and diagrams, 85 pages filled with almost 2500 additional diagrams, and 400 microfiche pages that contain further diagrams and thousands of individual verifications of claims made in the 24 lemmas in the main sections of text. In addition, the reader is told that certain facts have been verified with the use of about twelve hundred hours of computer time and would
be extremely time-consuming to verify by hand. The papers are somewhat intimidating due to their style and length and few mathematicians have read them in any detail.

(24) Appel y Haken, pág. 178 de (2)

The fundamental reason that the unavoidable set argument worked whereas other approaches to the Four-Color Conjecture did not is that all other approaches need somewhat stronger theoretical tools to make their methods apply. While these might be possible to create, there is no guarantee that they are actually possible; and if they are, there is no obvious way to go about finding them.

(25) Jean Mayer profesor de literatura, obseso y erudito de la conjetura de los cuatro colores. Entre sus contribuciones referentes a ella demostró que todo mapa de menos de 96 países es cuatro-coloreable.

(26) Thomas,Robin & Robertson, Neil & Sanders,Paul. *The Four color Theorem*. http://www.math.gatech.edu/~thomas/FC/fourcolor.html. Copio y pego :

There are two reasons why the Appel-Haken proof is not completely satisfactory.

- *Part of the Appel-Haken proof uses a computer, and cannot be verified by hand, and*
- *Even the part that is supposedly hand-checkable is extraordinarily complicated and tedious, and as far as we know, no one has verified it in its entirety.*

(27) Swart, E.R, *The philosophical implications of the Four-Color problem*. American Mathematical Monthly, 697-707;

I would go so far as to say that any lack of reliability of the present proofs of the 4CT resides less in the use of a computer for the reducibility testing and more in the fact that a computer is not used to create the unavoidable set of configurations arising from the discharging procedure pág.700

Edward Swart, matemático perteneciente al grupo de investigadores con un andamio común (Heesch y computadora) al de Appel y Haken para pintar a la capilla de los cuatro colores. En ese momento de su carrera trabajaba en la Universidad de Rhodesia y como a los demás miembros del conspicuo grupo la demostración de Appel y Haken les robó la comisión y la gloria.

(28) En una actualización de Robin Thomas (AN UPDATE ON THE FOUR-COLOR THEOREM, Notices Amer. Math. Soc. 45 (1998), no. 7, 848-859; http://www.math.gatech.edu/~thomas/PAP/update.pdf) se continua con la desacreditación:

To my knowledge, the most comprehensive effort to verify the A&H proof was undertaken by Schmidt...during the one-year limitation imposed on his master's thesis, Schmidt was able to verify about 40% of part I of the A&H proof.

Escarbando en *The four color proof suffices* (Appel, K., and W. Haken. 1986.. *The Mathematical Intelligencer* 8(No. 1), págs. 19-20) se relata el descubrimiento del error por Schmidt (1981) y su posterior rectificación.

De la fuente [(23), p 27] extraje la notificación del segundo error, localizado por el nipón Saeki en su tesis de maestría (1985). El final feliz se impuso al también ser corregido.

(29) Allaire, Frank, *Another proof of the Four-Color Theorem*, Proceedings of the Seventh Manitoba Conference on Numerical Mathematics and Computing, 1977, págs 3-72.

(30) *At the very least Allaire`s proof must rank as an independent corroboration of the truth of the four-color conjecture, and there can be little doubt that even if the Haken/Appel proof is flawed the theorem is nevertheless true.* pág. 698 de (27)

(31) *I cannot express any confidence in their [H&A] discharching scheme*.
Allaire, *On Reducible Configurations*, pág. 94

(32) Tymoczko, Thomas. *The Four-Color problem and its philosophical significance.* The Journal of Philosophy, vol LXXVI, no.2 págs 57-83,

(a) Proofs are convincing. This fact is key to understanding mathematic as a human activity. It is because proofs are convincing to an arbitrary mathematician that they can play their role as arbiter of judgment in mathematical community...
...¿why they are convincing?... [because they] are surveyable and...formalizable...
b) Proofs are surveyable. Proofs are the guarantees of mathematical knowledge and so they must be comprehended by mathematicians. A proof is a construction that can be looked over, reviewed, verified by a rational agent.
(c) Proofs are formalizable. A proof, as defined in logic, is a finite sequence of formulas of a formal theory satisfying certain conditions. It is a deduction of the conclusion from the axioms of the theory by means of the axioms and rules of logic. Most mathematicians and philosophers believe than any acceptable proof can be formalized... págs.59 y60.

(33) Tymoczko, Thomas, *Computers, Proofs and Mathematicians: A Philosophical Investigation of the Four-Color Proof* págs.131-138.

Thus "checking a proof" resolves into two components: the epistemological act of checking a proof and the logico-mathematical distinction among purported proofs between (real) proofs and counterfeits. pág.132

(34) *Mathematicians` ability to check proofs provides a simple idealized account of mathematical knowledge of theorems. How does a mathematician come to know a theorem? By checking a proof of it.* pág.131 de (33)

(35) El doblepensar es algo parecido a inhumar conscientemente a las mentiras en la inconsciencia, para luego creer en las fantasmales verdades y por último inconscientemente borrar las paladas de la consciencia.

(36) *We often say that a proof must be perspicuous or capable of being checked by hand. It is an exhibition, a derivation of the conclusion, and it needs nothing outside of itself to be convincing. The mathematician surveys the proof in its entirety and thereby comes to know the conclusion.* pág.59 de (32)

(37) Tymoczko, pág. 60 de (32)
...to say that they can be surveyed is to say that they can be definitively checked by members of the mathematical community.

(38) Stewart, Ian. *Concepts of Modern Mathematics,* Penguin, 1981, pág.304.

...[H&A proof] does not give a satisfactory explanation why the theorem is true. This is partly because the proof is so long that it is hard to grasp (including the computer calculations, impossible!).... The answer appears as a kind of monstrous coincidence. Why is there an unavoidable set of reducible configurations? The best answer at the present time is: there just is. The proof: here it is, see for yourself. The mathematician's search for hidden structure, his pattern-binding [sic] urge, is frustrated.

(39) Compilación es el proceso por el cual se traducen programas en código fuente a programas en código objeto. El programa que realiza esta traducción se llama compilador. El archivo de código objeto que se obtiene con la compilación está representado normalmente en código de máquina (ceros y unos).

Para conseguir el programa ejecutable final a partir de todos los archivos de código objeto se debe utilizar un programa llamado montador y un enlazador (linker). Este proceso de montaje tiene como resultado un archivo ejecutable que contiene el programa en código máquina listo para ser ejecutado con la ayuda del sistema operativo.

De Wikipedia, la enciclopedia libre. http://es.wikipedia.org

(40) El lenguaje ensamblador o código simbólico (en inglés Assembly language) es una notación del lenguaje de máquina que es legible por personas y es específica de cada arquitectura de computadoras. El lenguaje de máquina, un simple patrón de bits, es hecho legible reemplazando valores crudos por símbolos denominados mnemónicos. Se inventó para facilitar la tarea de los primeros programadores que hasta ese momento tenían que escribir directamente en código binario.

De Wikipedia, la enciclopedia libre. http://es.wikipedia.org

(41) *The task of evaluating programs is a topic of computer science, but at present there are no general methods for accomplishing it at this level. Programs themselves are written in special "languages", and many of them can be quite complex. They can contain "bugs*", or flaws that can go unnoticed for a long time. The reliability of any appeal to computers must ultimately rest on such diffuse grounds as these*.
pág.74 de (32)

* Un error de software o *computer bug*, es el resultado de una falla de programación introducida en el proceso de creación de programas de computadora. El término *bug* fue acreditado erróneamente a Grace Murray Hopper, una pionera en la historia de la computación, pero Thomas Edison ya lo empleaba en sus trabajos para describir defectos en sistemas mecánicos por el año 1870.

Wikipedia, la enciclopedia libre.

(42) *It is true that computer programs do sometimes have "bugs", but so do attempts to prove theorems by means of pencil and paper. It is also true, as Tymoczko says, that the "flaws in programs may sometimes go unnoticed for a long time". But so do flaws in proofs that have nothing to do with computers. The first published "proof" of the 4CT, by Kempe, was flawed, and it was not until 10 years later that Heawood uncovered the flaw...It is true that in some cases algorithms with logically sound implementations that are themselves logically sound* do not always produce the results they should when run on a computer.*

Thus us because computer are finite and we are thus forced either to truncate or to round off irrational number or number with a recurring binary representation.....But such an unhappy situation does not apply to the algorithm for reducibility testing, which is restricted to operations involving integer or Boolean truth values... Although the algorithm itself is very simple, its computer implementation is anything but trivial; the actual programs for the proofs were written in assembler language in order to make them as efficient as possible. This raises problems as regards any formal check on the correctness of the programs, since the formal checks used at present apply to high-level language programs rather than assembler language programs[**]*. Nevertheless the reducibility results have been indirectly surveyed in a manner very little different form the hand surveying of other proofs involving a large amount of case testing-namely, by means of independently written computer programs run on different computers. In fact, the case testing for the reducibility results in the 4CT has probably been more thoroughly surveyed than the case testing in some other theorems that are not dependent on the computer.*

págs. 703-704 de (27).

* La verificación formal comienza desde el diseño del algoritmo, normalmente aquí de manera ortodoxa se puede demostrar si el algoritmo es correcto, i.e. es una representación precisa de un procedimiento efectivo para resolver el problema (*logical soundness*). Por ejemplo Swart en (27) menciona el algoritmo de Kruskal (*greedy algorithm, pág 702*-703) para encontrar el árbol generador de peso máximo de una gráfica conexa G con aristas ponderadas. El árbol T producto de dicho algoritmo es un árbol generador de peso máximo, este hecho se demuestra (v.gr. en (3)).

** La programación en el lenguaje ensamblador tiene ciertas ventajas:

- Mayor adaptación al equipo.
- Posibilidad de obtener la máxima velocidad con mínimo uso de memoria.

También tienen importantes inconvenientes:

- Imposibilidad de escribir código independiente de la máquina.
- Mayor dificultad en la programación y en la comprensión de los programas.

Por esta razón, a finales de los años 1950 surgió un nuevo tipo de lenguaje que evitaba los inconvenientes, a costa de ceder un poco en las ventajas.

Estos lenguajes se llaman "de tercera generación" o "de alto nivel", en contraposición a los "de bajo nivel" o "de nivel próximo a la máquina" (ensamblador). Los lenguajes de alto nivel tienen una gramática mucho más rica y asequible para los humanos. Ejemplos de lenguajes de alto nivel son Java, C y C++.

De Wikipedia, la enciclopedia libre.

(43)*When proofs are long and highly computational, it may be argued that even when hand checking is possible, the probability of human error is considerably higher than that of machine error*
pág 179 de (4)

(44) *Drop the image of mathematicians as infallible. Recognizing the fallible nature of mathematical knowledge will help us to recognize the mechanism by which mathematicians individually and in the community work to minimize error.*
pág. 136-37 de (32)

(45) Aunque la gran distinción filosófica entre la mente y el cuerpo en el pensamiento occidental puede ser rastreada desde los griegos, es en la obra fecunda de René Descartes (1596-1650), matemático, filósofo y fisiólogo francés, al que debemos la primera explicación sistemática de las relaciones entre la mente y el cuerpo. Según la concepción de Descartes, el alma racional, una entidad distinta del cuerpo y puesta en contacto con el mismo por la glándula pineal, puede o no puede darse cuenta de las emanaciones diferenciales que los espíritus animales traían a su alrededor a través de la reordenación de los espacios interfibrilares. Cuando tales percepciones ocurren, sin embargo, el resultado es la sensación consciente -el cuerpo afecta a la mente. A su vez, en la acción voluntaria, el alma puede por sí misma iniciar una emanación diferencial de espíritus animales. La mente, en otras palabras, puede también afectar al cuerpo.
. La esencia del cuerpo es la extensión; mientras la del alma o mente es el pensamiento. El cuerpo es espacial, el alma no tiene extensión. Al localizar el punto de contacto entre el alma y el cuerpo en la glándula pineal, Descartes había planteado la cuestión de las relaciones de la mente con el cerebro y el sistema nervioso. Pero al mismo tiempo, al trazar una radical distinción ontológica entre el cuerpo como extensión y la mente como puro pensamiento, Descartes, en búsqueda de la certidumbre, había creado, paradójicamente, un caos intelectual.

Wozniak, Robert, *MENTE Y CUERPO: DE RENÉ DESCARTES A WILLIAM JAMES*,
Traducción al castellano de la versión original en inglés:
Miguel Angel de la Cruz Vives.
http://platea.pntic.mec.es/~macruz/mente/descartes/descartes.html

(46) Teller, Paul , *Computer proof,* The Journal of Philosophy, pp 797-803

Surveyability is needed, not because without it a proof is in any sense not a proof (or only a proof in some new sense), but because without surveyability we seem not to be able to verity that a proof is correct. So surveyability is not a part of what it is a proof. It is a characteristic which some proofs have and which we want our proofs to have so that we may reasonably assure ourselves what we take to be a correct proof is so. In particular, we may take advantage of new methods of surveying as long as these enable us to meet sensible demands on checking proofs, and a shift in the means of surveying actually used means only

a shift in methods of checking proofs, not a shift in our conception....The novelty is in the means of surveying. not in the means of proof.
pág. 798

(47) Tymoczko, págs.74 y 75 de (32)

Suppose that advances in computer science lead to the following circumstances. We can program a computer to initiate a search through various proof procedures, with subprograms to modify and combine procedures in appropriate circumstances, until it finds a proof of statement A. After a long time, the computer reports a proof of A, although we can't reconstruct the general shape of the proof beyond the bare minimum (e.g. by induction). Perhaps we could describe this hypothetical example by saying that the supercomputer found a human assisted-proof....the question is whether mathematicians would have sufficient faith in the reliability of computers to accept this result.

(48) Teller, pág. 796 de (46)

If a computer is programmed to use the same methods of proof we use, a proof that it produced would be a proof in our old sense. Legitimate worries whether an error might have been made only show that there may be legitimate worries whether the proof is correct, not that the proof is a proof.. To put the point in a very slightly different way, the fact that I cannot follow a complex proof produced by a good mathematician does not show that such a mathematician's complex proof is a proof in a different sense of the word from a proof that I can follow.

(49) *Traditionally, a priori truths are those truths which can be known by reason independently of any experience beyond pure thought.*
pág.77 de (32)

(50) *...we have anticipated future help from computers in discovering new proofs by use of programs embodying methods of proof we use ourselves. But this no mere entails a shift in the epistemological foundations of mathematics than would new ways of checking the proper functioning of electron microscopes, or new ways of building electron microscopes, entail a shift in the epistemological foundations of the natural sciences.*
pág. 801 de (46)

(51) Teller, pág 802 de (46)
An experiment in the natural sciences determines a spatiotemporal fact, which then may or may not be generalized beyond the local place and time to which it directly applies. A correct mathematical proof establishes a kind of fact which is non-spatiotemporal.

(52) Swart , pág. 700 de (27)
A priori truth is a truth that possesses universal and necessary validity; that is to say, a truth that is true in all possible worlds.

(53) *...we can argue that al mathematical truths, even the 4CT, are necessary, or true in all possible worlds. The 4CT, we might say, records an essential property of planar graphs.* pág.78 de (31)

(54) Swart, pág. 701 de (27)
If, in order to uncover their truth, it is necessary to use pencil and paper or a calculator or even a computer, thus can make no difference to the nature of their truth.

(55) Swart, pág. 701 de (27)
[a priori truth] is a truth whose validity can in principle be established without recourse to sense experience of the physical world.

(56) *...a priori truth is one that can be uncovered by a sentient being, with a brain sufficiently large for the purpose, without recourse to any experiment. It does not imply that a particular type of sentient beings that happen to have arisen on earth, which we choose to call Homo sapiens, may not need to resort to an experiment to know the truth of a specific a priori truth.* pág. 701 de (27)

(57)
Appel y Haken pág. 153 de (4)
One can never rule out the chance that a short proof of the Four-Color Theorem might some day be found, perhaps by the proverbial bright high-school student. But is also conceivable that no such proof is possible. In this case a new and interesting type of theorem has appeared, one which has no proof of the traditional form

The current consensus among mathematicians is that the present proof [H&A] is reasonably close to the simplest proof. If this is so, then the appeal to computers would be essential to any mathematical justification of the 4CT.
Tymoczko, pág. 69 de (32)

(58) *...this investigation should be purely philosophical, since the mathematical question can be regarded as definitively solved. It is not my aim to interfere with the rights of mathematicians to determine what is and what is not a theorem. I will suggest, however, that, if we accept the 4CT we are committed to changing...the sense of the underlying concept of "proof".* pág. 57-58 de (32)

(59) La única forma para derrocar al real teorema convirtiéndola en una mundana conjetura es por las vías de la matemática, la demostración debe ser refutada con argumentos matemáticos y no con escaramuzas filosóficas.

Por otro lado Swart es seducido por la tentación del nominalismo adánico, aunque fiel a su postura el 4TC conserva su majestad, la demostración de H&A reptando insinúa un nuevo tipo entidad en el edén donde conviven los lemas, conjeturas, teoremas, etc. llamada "agnograma":

...agnograms, meaning thereby theoremlike statements that we have verified as best we can but whose truth is not known with the kind of assurance we attach to theorems and about which we must thus remain, to some extent, agnostic.
pág. 705 de (27)

Hasta donde puede llegar una persona con tal de no abandonar sus creencias, en lugar de desembarazarse de la sacralización de las entidades matemáticas. Swart prefiere crear una nueva creencia afín a sus sueños de pureza en lugar de despertar.

(60) *ABSTRACT: Acceptable but due to extensive usage of a computer rather unpleasant proof of the famous four color map problem of Francis Guthrie were settled eventually by W. Appel and K. Haken in 1976. Using the same method but shortening the proof twenty years later by another team, namely N. Robertson, D.P. Sanders, P.D. Seymour and R. Thomas would not improve considerably the readability of the proof either. Thus it has been widely accepted the need of more elegant and readable proof. There are considerable number of equivalent formulations of the problem but none of them promising for a possible non-computer proofs. On the other hand known proofs are used the concept of Kempe chain and reducibility of the configurations which were a century old ideas. With these in mind we have introduced a new concept which we call "spiral chains" in the maximal planar graphs. We have shown that for any maximal graph as long as spiral chains are being used we do not need the fifth color. Henceforth this paper offers another proof to the four color theorem which is not based on deep and abstract theories from the other branches of mathematics or using computing power of computers, but rather completely on a new idea in graph theory.*

Cahit, http://arxiv.org/abs/math.CO/0408247

(61)*...the 4CT would be an important example of an a posteriori necessary truth and, a fortiori, a counterexample to the claim that all known necessary truths are known a priori.*
pág.78 de (32)

(62) *The discharging procedure and the resulting creation of the unavoidable set of configurations could be computarized.*
The programs for both the discharging and the reducibility testing could be written in a high-level language and adequately checked to ensure that they do implement the algorithms they purport to implement.
pág. 704 de (27)

(63) Lam, Clement, *The Search for a Finite Projective Plane of Order 10****
http://www.cecm.sfu.ca/organics/authors/lam/index.htm

Abstract:

Projective planes are special cases of a class of combinatorial objects called symmetric block designs. We are not going to discuss block designs, except to mention that Chowla and Ryser have generalized the Bruck-Ryser theorem to symmetric block designs [10], which it is now known as the Bruck-Ryser-Chowla theorem. Here again, a partial converse exists, providing more credence to the hope that the conditions in the Bruck-Ryser-Chowla theorem are both necessary and sufficient. This hope is now shattered by the non-existence of the finite projective plane of order 10.

***Previously appeared in the American Mathematical Monthly 98, (no. 4) 1991, 305 - 318.

El epílogo aunque intrascendente por emotivo lo transcribo:

Epilogue

While we were tracing the origin of the existence problem of the plane of order 10, we asked Dan Hughes, who has worked in this area for a long time and is famous for the Hughes planes which are named after him. He recounted the following story. In about 1957, at a Chinese restaurant in Chicago, Reinhold Baer, another mathematician well known for his work in group theory and projective planes, was trying to impress the younger Hughes by remarking that if the plane of order 10 was settled by a computer, he hoped not be alive to see it. Baer got his wish but I do not think Herb Ryser shared this opinion. Ryser was happy that the weight 12 case was settled by a computer. I can only extrapolate and hope that he would also be happy that the whole problem has been ``settled'', even if by a computer.

(64) *Drop the image of an isolated mathematician and admit that mathematical knowledge is public knowledge of a community of mathematicians. Recognizing this community will help us to recognize the communication among mathematicians that constitutes the doing of mathematics and grounds mathematical knowledge*
pág 136 de (32)

(65) Thomas,Robin & Robertson, Neil & Sanders,Paul, *HADWIGER'S CONJECTURE FOR K_6-FREE GRAPHS*, September 1992, revised April 1993. Published in Combinatorica 14 (1993), 279-361.
http://www.math.gatech.edu/~thomas/PAP/hadwiger.pdf

Abstract
In 1943, Hadwiger made the conjecture that every loopless graph not contractible to the complete graph on t+1 vertices is t-colourable. When t = 3 this is easy, and when t = 4, Wagner's theorem of 1937 shows the conjecture to be equivalent to the four-colour conjecture (the 4CC). However, when t =5 it has remained open. Here

we show that when t = 5 it is also equivalent to the 4CC. More precisely, we show (without assuming the 4CC) that every minimal counterexample to Hadwiger's conjecture when t = 5 is "apex", that is, it consists of a planar graph with one additional vertex. Consequently, the 4CC implies Hadwiger's conjecture when t = 5, because it implies that apex graphs are 5-colourable.

La conjetura de Hardwiger escrita de una manera más formal:

For every t ³ 0, every loopless graph with no K_{t+1}-minor is t-colourable.

(All graphs [considered] are finite; K_n is the complete graph with n vertices; a graph H is a minor of a graph G if H can be obtained from a subgraph of G by contracting edges; an H-minor of G is a minor isomorphic to H; a loopless graph does not have vertices adjacent to themselves)

(66) *We have in fact tried to verify the Appel-Haken proof, but soon gave up. Checking the computer part would not only require a lot of programming, but also inputing the descriptions of 1476 graphs, and that was not even the most controversial part of the proof [discharging procedure] We decided that it would be more profitable to work out our own proof.*
http://www.math.gatech.edu/~thomas/FC/fourcolor.html

(67) *What makes the use of computers in the 4CT so dramatic is that it leads to a genuine extension of our knowledge of pure mathematics. It is not merely calculation, but yields a proof of a substantial new result.*
pág. 69 de (32)

CAPÍTULO III: LA RESOLUCIÓN INDUSTRIAL

La primera sección relata la capacitación de los binarios obreros deductivos para incorporarse a las fábricas de pruebas para el cálculo de predicados de primer orden, haciendo un recuento histórico de las primeras técnicas y procedimientos desarrollados para pasar del mercado laboral del cálculo proposicional de la LTM hacia este sector lógicamente más redituable. Se hará especial énfasis en los principales problemas teóricos de la producción, saboteadores activos en teoría pero en la práctica nihilistas pasivos -aunque su fantasmal presencia no debe ni puede olvidarse pues determinan los límites últimos de la manufactura-, quienes son el teorema de la indecidibilidad, la catalogación NP y en menor medida el teorema de la incompletez. En la segunda sección se describirán a los procesos de producción de la industria deductiva más lucrativa de la actualidad, representante del paradigma del lenguaje clausular y cuyas instalaciones primarias se localizan en Argonne. El motor histórico de este modelo ha sido la regla de inferencia desarrollada por Alan Robinson de nombre resolución, regla enfocada a las habilidades de los obreros digitales en detrimento de la comprensión de sus patrones; al final de la tercera sección se enunciarán algunos de los trabajadores de Argonne, en donde figura el obrero más productivo llamado Otter. La tercera sección exhibirá un par de pruebas sobre dos problemas abiertos maquiladas por los obreros bajo el régimen resolutivo, a saber la determinación de la base axiomática XCB del cálculo de equivalencias y la conjetura de Robbins del álgebra Booleana.
Una vez preparado el terreno, en la sección final se evaluará la calidad de los objetos deductivos manufacturados por la fábrica resolutiva, analizando si cumplen con las tres máximas de una demostración: ser *convincente-computable-revisable* (*inteligible*). Paralelo a estos puntos, se reseñará el segundo aire del descomunal proyecto de la formalización de las matemáticas, rescatado del baúl de los sueños devaluados gracias a la fuerza laboral de nuestros trabajadores de silicio y por el momento resguardado en la caja de las esperanzas en el futuro construidas desde el pasado.

DISOLUCIÓN

Todas las demostraciones son convincentes. $\forall x(DEM(x) \Rightarrow CON(x))$
La “H&A&K&1&0 demostración” de la 4CC es una demostración $DEM(H+A+K+1+0)$
La “H&A&K&1&0 demostración” de la 4CC es convincente $CON(H+A+K+1+0)$

Como pacto demoníaco, el lenguaje concierta lo infinito con lo finito, la lógica reglamenta al sulfuroso amasiato. Los cuantificadores sellan al concubinato, la fecundidad representativa explota pero un precio debe ser pagado. La incorporación de los cuantificadores ofrece nuevas vías, incluso hacia el infinito, aunque también nos proporciona nuevas dificultades. La complejidad de lo expresado y de los medios para hacerlo correctamente se dispara con el “para todos” o el “existe”. En las matemáticas resulta indispensable ese despegue.

En el capítulo primero se explicaron dos maneras para generar pruebas en el sistema del cálculo proposicional, el algoritmo del Museo Británico y los métodos heurísticos de la LTM. Recordando, los métodos subyacentes a la LTM buscaban emular algunas estrategias

seguidas por los matemáticos cuando se disponen a elaborar una demostración mientras que el algoritmo del Museo Británico construía sistemáticamente a todas las pruebas del cálculo proposicional. Pregunta inmediata: ¿Pueden adaptarse estos paradigmas demostrativos al sistema del cálculo de predicados? Pregunta-respuesta: ¿vale la pena hacerlo? Para poder contestar adecuadamente debemos saber sobre el extraño objeto de la duda, por lo cual esbocemos al sistema donde la cuantificación (sólo en un nivel) hace acto de presencia.

El sistema del cálculo de predicados de primer orden (CPPO) es el entorno donde marcharán nuestras inminentes exploraciones. Todo sistema formal que se respete consta de un lenguaje y sus reglas de construcción de fórmulas, términos, etc. junto con un cálculo, i.e. sus reglas para derivar fórmulas, edificar sus pruebas, etc. La semántica es el tercer componente esencial de cualquier sistema formal. Prosaicamente el CPPO es una extensión del cálculo proposicional, conserva todas las gracias de las operaciones con conectivos (conjunción, disyunción, etc.) aunadas a la cuantificación sobre los símbolos individuales. Sea CPPOpat un bosquejo de ejemplo de un CPPO:

Lenguaje CPPOpat

- Símbolos: conectivos lógicos ("$\neg$", "$\Rightarrow$"), signos de puntuación ("(", ")"), signos de relación ("D", "C"), cuantificadores ("$\forall$"), variables individuales ("x_i" donde $i \in N$), constantes ("c")
- Términos: Las variables y las constantes son términos
- Fórmulas: Si r,t son términos entonces "D(t)" y "C(t)" son fórmulas ; si A,B son fórmulas y x una variable individual entonces $\neg A$, $A \Rightarrow B$, $\forall xA$ son fórmulas

Cálculo CPPOIpat

- Axiomas: (G1) $A \Rightarrow (B \Rightarrow A)$, (G2) $\forall xA(x) \Rightarrow A(t)$ con t libre para x en A[1]
- Reglas de inferencia: MP $\dfrac{A \Rightarrow B \quad A}{B}$ GEN $\dfrac{A}{\forall xA}$
- Pruebas,derivaciones, teoremas: No se permite GENeralizar aquellas variables que ocurren libres en las hipótesis. Una prueba es una secuencia finita de fórmulas $A_1,.....,A_n$ donde p.c. $1=i=n$:

$$A_i \text{ es un axioma}$$
$$\text{o}$$
$$\exists j((1 \leq j < i) \ \& \ A_i \text{ es el resultado de GEN } A_j$$
$$\text{o}$$
$$\exists k,j(1 \leq k,j < i \ \& \ j \neq k \ \& \ A_i \text{ es el resultado de MP de } A_j, A_k$$

  A_n es un teorema de CPPOpat , si A es un teorema se indican así: $+_{\text{CPPOIpat}} A$

- Semántica: Un dominio de interpretación de CPPOPat son las aspirantes a demostraciones; "c" es la supuesta demostración de H&A&K&1&0 de la 4CC; "D(t)" significa "t es una demostración"; "C(t)" significa "t es convincente".

Ahora estamos en posición de intentar demostrar C(c) con la hipótesis especiales $\forall x(D(x) \Rightarrow C(x))$, D(c) en CPPOIPat, es decir $\forall x(D(x) \Rightarrow C(x)), D(c) +_{CPPOIpat} C(c)$

PRUEBA:

(1) $\forall x(D(x) \Rightarrow C(x))$ HIP
(2) $\forall x(D(x) \Rightarrow C(x)) \Rightarrow (D(c) \Rightarrow C(c))$ AXG2
(3) $D(c) \Rightarrow C(c)$ MP 1,2
(4) D(c) HIP
(5) C(c) MP 3,4

El teorema patito de CPPOpat bajo las hipótesis especiales sirve para reafirmar lo obvio: cuando están estipuladas las reglas del lenguaje y del cálculo de un sistema, tal como lo están en los sistemas formales, tanto los enfoques algorítmicos como heurísticos son aplicables. Por un lado, un Algoritmo del Museo Británico modificado puede construir el inacabable legado demostrativo del sistema del cálculo de predicados con igualdad. Observación trivial y reiterativa: el museo ofrece una belleza inalcanzable para los mortales matemáticos. En el otro polo, las técnicas heurísticas aunque de antemano no descartables por sí solas prometen poco éxito si no se acatan las sugerencias de Wang, al menos si nuestro objetivo es demostrar teoremas. Hagamos uso de nuestra memoria, Wang propone moldear las herramientas lógicas según las capacidades del esclavo binario para explotarlo óptimamente en la pizca de pruebas. En 1958 el capataz Wang hizo caso de si mismo y elaboró un programa para recolectar las pruebas de todos los teoremas (alrededor de 400) del cálculo de predicados de primer orden con igualdad (CPPOI) localizados en la monumental obra de Rusell-Whitehead, *Principia Mathematica*[2]. Wang implementó un procedimiento de decisión para un subdominio del CPPOI, en donde ningún cuantificador existencial gobierna a uno universal en la forma *prenex* (cuantificadores al principio) de las fórmulas[3]. La fortuna le sonrió a Wang, todos los teoremas del CPPOI en *Principia Mathematica* pueden transformarse a una forma *prenex* con el prefijo simple $\forall \ldots \forall \exists \ldots \exists$. Intuitivamente, un **proceso de decisión** es aquel que siempre determina correctamente si un objeto matemático (v.gr. una fórmula o un conjunto fórmulas) cumple o no con cierta propiedad. Ejemplo ya mencionado en el primer capítulo, las tablas de verdad son un método de decisión sobre la satisfactibilidad de una fórmula en el cálculo proposicional (Sea A una fórmula del CP, ¿existe una asignación de los valores {falso,verdadero} a las componentes atómicas de A para hacerla verdadera?). Más aun, mediante las tablas de verdad podemos decidir si una fórmula es una tautología y por consecuencia un teorema en el CP. La aventura binaria por demostrar dragones de primer orden se podría dar por terminada (al menos teóricamente pues en la realidad pululan incrédulamente los obstáculos materiales) de contar con la espada de la decidibilidad. Es momento para sumir de una vez por todas en la piedra de la negación a la respuesta.

TEOREMA: El cálculo de predicados de primer orden es **indecidible**.
Traducción: No hay un "procedimiento mecánico" para decidir si cualquier fórmula A es o no es un teorema, i.e. si existe o no una prueba de A en el cálculo de predicados de primer orden

Alonso Church y Alan Turing a mediados de los 1930's independientemente confirmaron a la indecidibilidad del CPPO, en la referencia (4) se encuentra un boceto detallado de la demostración *à la Turing* del teorema, demostración inconclusa pero con las explicaciones pertinentes para acabarla. Reproducirlo enteramente (no se diga completarla) retrasaría la marcha hacia los objetivos del escrito debido a su carácter excesivamente técnico. En lugar de la total trascripción sólo delinearé el plan maestro de la demostración. Abuso de la función comunicativa de las demostraciones para hacer patente de una vez una situación común en ellas: la creativa interacción entre las teorías. En este caso el elemento epistemológico ensancha sus requerimientos para cumplir con su función: hacer la demostración *inteligible*. Para poder leer al plan se necesita importar objetos y resultados de la teoría de la computación, hijos de la mente de Turing.

NOCIONES PREVIAS

Una máquina de Turing es una abstracción elaborada por la matemática A. Turing para definir rigurosamente lo que es un "procedimiento mecánico"; intuitivamente éste es una secuencia de instrucciones seguidas por un agente para completar correctamente una tarea específica (procedimiento mecánico~algoritmo). In/formalmente una máquina de Turing consiste en[5]:

- Una cinta de longitud infinita dividida en celdas adjuntas.
- Un conjunto finito de símbolos (G), el abecedario de la cinta. Cada celda de la cinta puede contener sólo uno de los símbolos del abecedario. Existe un símbolo notable perteneciente a todo abecedario, el símbolo nulo ($0 \in G$), tabula rasa de las celdas de la cinta (las celdas en las cuales no se ha escrito ningún otro símbolo contienen al símbolo nulo).
- Una cabeza lectora/escritora con la capacidad de moverse una celda tanto a la derecha (R) como a la izquierda (L) sobre la cinta.
- Un conjunto finito de estados (Q), al cual siempre pertenece un estado inicial ($q_0 \in Q$), estado inicial de toda secuencia de acciones de la máquina.
- Una tabla de acciones (función de transición δ) en donde se establecen el símbolo a escribir, el estado al cual debe cambiar y el desplazamiento de la cabeza con base al estado y el símbolo de la celda actuales. Cuando no hay una acción determinada en la tabla para alguna combinación de símbolo y estado en ese momento presente, entonces la máquina para. Dichos estados donde se especifica ninguna acción son nombrados estados finales.

  Formalmente $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ es la función parcial de transición (es parcial pues no está definida en todo su dominio). La función δ caracteriza a la máquina de Turing a la cual corresponde.

Ejemplo $M_{cuaccuac}$:

$Q=\{q_0, q_1, q_2\}$
$G=\{0,1\}$

	0	1
q_0	$(1, q_2, R)$	$(1, q_1, R)$
q_1	$(1, q_2, R)$	$(1, q_1, R)$
q_2		

Si como entrada de la máquina tenemos únicamente n 1´s adyacentes de izquierda a derecha, entonces $M_{cuaccuac}$ calcula su sucesor n+1.

Un problema clásico acerca de las máquinas de Turing es otro famoso probema de decisión: ¿existe un algoritmo/procedimiento mecánico para indicar si para cualquier máquina de Turing con una entrada dada ésta para? La respuesta del "Problema de la Detención" (*Halting Problem*) de nuevo es negativa. La primera demostración acerca de su indecidibilidad fue publicada por Turing en 1937[6] . Una vía refutar la existencia de dicho procedimiento mecánico es mediante la autoreferencia y el posterior absurdo. Esbozo. Con base a las funciones de transición podemos enumerar a las máquinas de Turing ($M_1, M_2, ...$). Simplifiquemos sin perder generalidad, toda entrada es numérica y asociemos a las máquinas de Turing el modelo de la caja negra, para todo número dado como entrada la máquina calcula un número de salida cuando la máquina se detiene. Sea D la máquina de Turing definida del siguiente modo: $D(i)=T_i(i)+1$ si $T_i(i)$ se detiene; $D(i)=0$ si no lo hace. Luego existe $k \in N$ tal que $T_k=D$. ¿Qué valor tiene D(k)? El valor de la contradicción.

Ahora ya podemos atacar el plan de la demostración de la indecidibilidad del cálculo de predicados de primer orden. La idea es simple y elegante, su ejecución larga y tediosa. Las máquinas de Turing junto con su accionar (el registro de los estados, valores presentes en la cinta y posición de la cabeza lectora/escritora en cada instante de ejecución de la función de transición) pueden ser descritos en el lenguaje del cálculo de predicados del primer orden, el proceso para hacerlo aunque no es complicado si exige meticulosidad y mucha atención en varios detalles técnicos; por lo tanto aquí no será explicado. En particular para cualquier máquina de Turing M, se puede construir algorítmicamente una fórmula S_M que es válida si y sólo si M se detiene. En caso de contar con un procedimiento mecánico para determinar si una fórmula del cálculo de predicados de primer orden es válida, lo que equivale a decidir si S_M es o no es un teorema debido a la completud del sistema[7], entonces habríamos resuelto el Problema de la Detención. Contradicción.

El plan vaticina una demostración demandante de conocimientos en lógica y en la teoría de la computación. Aunque dicho sea de paso, tampoco son ramas distantes, al contrario, han tenido una relación bastante fructífera, cercana y la seguirán teniendo, el tema de la tesis es sólo uno de sus puntos de contacto. Hago hincapié en la componente epistemológica de la demostración puesto que contrasta nítidamente con la característica de las demostraciones de ser formalizables, bautizada en el Capítulo II como *computable.* El término computable se utiliza en las ciencias de la computación para denominar a las tareas resolubles de

manera efectiva, mecánica, algorítmica. Siendo más precisos, se les llama Turing-computables a los problemas aptos para ser despachados por una máquina de Turing. Si en la demostración se recurre a la capacidad de representar con el lenguaje de primer orden a las máquinas de Turing con sus idas y vueltas, el recíproco también sucede, con diligencia se puede "programar" una máquina de Turing para generar o validar pruebas del cálculo de predicados de primer orden. Desde un punto de vista simplista aunque no tan desatinado, nuestras poderosas computadoras digitales son una burda materialización de una máquina de Turing. Las actuales computadoras en comparación con las máquinas de Turing son menos poderosas pues les está negado el privilegio de la infinitud de la memoria y la independencia del tiempo, es decir, padecen del mal de la complejidad temporal y espacial. En el terreno de la computabilidad *per se* resulta intrascendente cuánto tiempo y memoria se requiera para la ejecución de una función de transición, basta con la existencia de una máquina de Turing capaz de resolver el problema en cuestión. No obstante aún en el limitante ámbito de la realidad nuestras computadoras tienen la facultad de manejar y producir pruebas en los sistemas formales, tal como fue ejemplificado en el capítulo uno. El cálculo de predicados gracias a su explícita construcción facilita su implementación en la computadora. Es más, los siervos mecánicos de silicio sobrepasan por mucho nuestros talentos para la manipulación en un sistema formal; pueden trabajar con fórmulas increíblemente largas y con una cantidad terrorífica de ellas y lo hacen en un tiempo infinitamente menor al empleado por nuestros pusilánimes cerebros de carbono. Renombrar a la característica "ser formalizable" de las demostraciones por "ser computable" presagió mi postura con respecto a la computadoras y la demostración. En retrospectiva, la sección de Wang indicaba precozmente una de las direcciones del ensayo. Una demostración debe ser en su última instancia formalizada, debe depurarse incansablemente para dispensarla de las mondaduras intuitivas para delimitar en su corazón puro las nervaduras de la intuición. No importa si el encargado de hacerlo sea una insensible y lerda máquina. La discusión puede tornar en un análisis más teórico al abordar la ¿computabilidad en abstracto de las demostraciones? (estilo Turing-computabilidad) pero no transmigrará debido al protagonista material del ensayo. "C*omputabilidad"* se atendrá con las computadoras en sus labores de asistencia para conseguir la formalización de las demostraciones. Por lo tanto objetivo esencial de este capítulo y el próximo es delimitar las habilidades de las computadoras aprovechables para la monumental obra de la formalización de las demostraciones. Restrinjo todavía más la mira: escudriñaré las capacidades de la computadora para la verificación y sobretodo para la generación de pruebas. La primera acción es más fácil de cumplir, para determinar si una prueba en verdad lo es en un principio basta con programar los axiomas, teoremas conocidos y reglas de inferencia del sistema examinado. Lo simple a veces también esta cubierto por los nubarrones de la adversidad. Aunque en apariencia sea sencilla la programación de la verificación, sufre la presión de ser la parte más crítica del proceso. Debe ASEGURARSE hasta la medida de lo imposible el correcto funcionamiento de los programas encargados de la verificación de las pruebas. Recordando, la "incertidumbre" en el instrumento era causa de la retahíla de dudas sobre la demostración de Haken y Appel. La computadora debe ser depósito de la total confianza de la comunidad matemática, mientras la sombra del recelo persista la empresa estará eclipsada. Imposible lograr este estado de armonía mientras el

estatus de la computadora siga siendo menospreciado y relegado por varios miembros de la comunidad. De una vez aviso, NO está a nuestro alcance la elaboración de programas libres de error ni máquinas perfectas en donde se ejecuten, por lo tanto los programas maestros de verificación carecen de una grado de certeza del 100% (CAP IV). No tiene nada de malo, al contrario, los absolutismos son más perniciosos, la fe siega a la visión crítica y paulatinamente ciega al conocimiento. Los avances dentro de la computación minimizan drásticamente la posibilidad de fallas, poco a poco y constantemente se solidifica la confianza en el instrumento. Para lograr una mayor adopción de la computadora por parte de los matemáticos, debe ponerse en marcha la campaña del convencimiento. Pues aunque haya garantía sobre el funcionamiento del cuerpo de transistores y el alma codificada en instrucciones, la computadora debe "de mostrar" su utilidad en las procesos de evaluación de las demostraciones, eslabón esencial en la cadena de producción del conocimiento matemático. Los programas verificadores deben ser provistos de trabajo. El éxito es un imán terriblemente efectivo. Si las máquinas triunfan donde los matemáticos reiteradas veces han fracasado, entonces el interés en ellas crecerá drásticamente. La demostración de Haken y Appel y ya afinada con el tema la estrella del actual capítulo, la demostración de McCune de la conjetura de Robbins, son sólo la punta del iceberg disuasivo. El segundo eje del apoyo de las computadoras en la formalización, le generación de pruebas, puede socorrer a la aceptación de las computadoras en la demostrativa vida matemática. En este capítulo se le dedicará especial atención a diversas maneras de llevarlo a cabo. Conforme se avanza hacia la inasequible infalibilidad de los programas, podríamos suponer prescindibles a los programas verificadores. Si los programas encargados de tejer pruebas tienen una nimia probabilidad de equivocarse en su zurcido, ¿para qué someterlos al escrutinio de otros programas? Inclusive se puede integrar a los programas generadores rutinas verificadoras para desde el momento de la creación de la prueba inspeccionar su buena hechura. Bajo semejante tónica, si nos obstinamos por imponer a los matemáticos de carne y hueso como los únicos árbitros acreditados para la verificación, de entrada los programas destinados a dicha tarea se sitúan en la puerta de la salida. Desde el capítulo anterior se disparó el alegato contra la discriminación de las espaldas binarias, aunque a muchos les desagraden, con el aumento de la producción de demostraciones matemáticas nuevas y difícilmente *revisables*, las computadoras se están volviendo indispensables. Aclaro, tampoco son desechables los usuarios pues son ellos los responsables de interpretar y revestir epistemológicamente a las pruebas manufacturadas. Por otro lado la búsqueda de pruebas es tan demandante que agregar a los programas destinados a realizarla la carga de la verificación es poco recomendable. La trasgresión alcanzaría una masa crítica si optamos por ni siquiera molestarnos en ratificar a las pruebas a través de evaluadores (programas o seres humanos), se pertrecharía un atentado en contra de la usanza vigente de las matemáticas. Aunque la acusada de ser una prueba goce de un amparo otorgado por el grado de sofisticación y la escasa probabilidad de falla de su programa progenitor, dicha "prueba" no puede escapar del juicio de su validación. A menos de que desmantelemos al andamio de las matemáticas, los respetables miembros del jurado, hombres y/o máquinas, no pueden ser excluidos. El error fielmente acompañará siempre a los matemáticos en su andar, más vale tomar toda precaución a la mano. Abandonarse por completo a las acciones de un solitario agente del conocimiento acarrea la posibilidad de una a falla de dimensiones

titánicas, una falla con posibilidad de generar un terremoto destructor de todo lo conocido gracias al agente. Además de la disminución de la probabilidad de error, el arbitraje externo ofrece el respaldo de los lazos de confianza sustentados por terceros. En teoría los miembros de la comunidad más experimentados son quienes examinan la demostración del teorema X, si la aprueban entonces los demás matemáticos confían en su validez y pueden hacer uso de X a su antojo. Ante cualquier conato de duda, el interesado Y revisa la demostración si su preparación se lo permite y en el genial extremo Y puede escribir una nueva demostración (más sencilla a lo mejor). Sin embargo Y tiene la enorme ventaja de estar enterado sobre el visto bueno de la demostración de X concedido por varios especialistas, por lo cual ahondar en la demostración de X probablemente le reportará una ganancia en lugar de una pérdida de tiempo. La confianza antecede al conocimiento, me temo. La confianza se pacta mediante terceros. Siendo más objetivos, en la precedente situación la verificación independiente de la demostración no murió, anda de parranda junto a otra área del conocimiento. Pues, ¿cómo se garantiza la casi nula posibilidad de error del programa generador de las pruebas? ¿Acaso el programa no debe forzosamente ser sometido al análisis de especialistas para sustentar su certificado de ser casi divino? Monstruoso panorama, si X es un teorema de topología algebraica y la garantía sobre su validez reposa en una base de conocimiento atestada de cuestiones computacionales. Según recuerdo Frankestein no fue muy popular dentro de la comunidad de aldeanos aunque hizo nada en bienestar del pueblo para ganarse su aprecio.
Catalizador de la aprobación de las computadoras en el núcleo de las matemáticas es su papel preponderante en la ambiciosa empresa de la formalización de las demostraciones. Con excepción del segundo capítulo, a lo largo de la tesis se vislumbra el ¿cómo? y a partir de aquí se tratará de sustentar el ¿por qué? Anticipo, la justificación puede ser plenamente filosófica al estilo de Wang: las pruebas son conceptualmente más sencillas y menos propensas al error. La desmesurada génesis de conocimiento matemático acaecida desde el siglo XX brinda un motivo de mayor peso, el ordenamiento y confección del cuerpo del conocimiento matemático, la construcción de la biblioteca de Alejandría del siglo XXI. Independientemente de la obra faraónica de la formalización, el estatus de "ser demostración" en las demostraciones computarizadas merece sin recular ser defendido, no hacerlo sería un símbolo del regreso al ignominioso oscurantismo y a la ignorante egolatría. PERO la defensa debe sostenerse con sumo cuidado, todo lo escrito por ella puede ser usado en su contra.

Aunque debido a las líneas finales del párrafo anterior parezca partidario de un recalcitrante y vacío formalismo, NO lo soy. La componente epistemológica, las andanzas intuitivas, son imprescindibles. Hay otros medios además de la fuerza bruta y a menudo hacemos uso de ellos: intuición, experiencia, conocimiento. El deber no agosta a nuestro intelecto, las demostraciones deber ser formalizadas pero el fondo también germina afuera de la reclusión en recipientes. Ejemplo ad hoc, la demostración del teorema de la indecidibilidad del CPPO. En ella sacamos provecho de nuestro don epistemológico de colocarnos en y sobre el sistema formal del cálculo de predicados de primer orden. En el CPPO podemos especificar las fórmulas caracterizadoras de cualquier máquina de Turing. Sobre el sistema formal aplicamos el resultado del Problema de la Detención perteneciente

a otra clase de estructura formal: las máquinas de Turing. El CPPO permite reflejar a las máquinas de Turing, con diligencia y conocimiento aparece un íntimo vínculo entre la imagen, el modelo y el espejo, dicha relación nos indica un límite especular. Emprender la formalización sobre el teorema no parece a primera vista cosa fácil ni útil. En general las demostraciones sobre sistemas formales apelan a la intuición, experiencia, conocimiento. Una de las razones es directa: debemos escapar del sistema para poder hablar del sistema. Hay propiedades de algunos sistemas imposibles de probarse en ellos mismos; en general hay proposiciones verdaderas en varios sistemas formales cuya prueba no puede ser formulada dentro de ellos, a partir de mediados del siglo pasado lo supimos gracias al trabajo del Kurt Gôdel. Su famoso y manoseado teorema de la **incompletud**[8] rompe el enlace entre el estatus de verdad y su pedigrí proporcionado por una prueba; Gödel exhibe cómo construir una proposición verdadera no demostrable en algunos sistemas formales (si le añadimos al CPPOI algunas cuantas gracias habremos logrado su conversión hacia la incompletud[9]). Sucintamente, su demostración lleva al borde de la potencialidad la reflexividad de algunos sistemas formales, en dichos sistemas se refleja a la aritmética recursiva pero la imagen en el espejo reflecta las propiedades del espejo. Las demostraciones sobre las propiedades de los sistemas formales nos conducen hacia la frontera del rigor y la intuición, a un inhóspito territorio intelectual donde sólo el conocimiento evita la perdición y el ofuscamiento. Desde una perspectiva más prosaica, la formalización de los teoremas sobre los sistemas sufren un cota natural, si bien a primera vista tal vez en otro sistema formal se pueda formular v.gr. la prueba del teorema de la incompletud, dicho sistema a su vez puede tener propiedades no demostrables dentro de él . O englobamos ad infinitum los sistemas formales o paramos en algún lugar en donde se pueda enunciar y demostrar características sobre los sistemas sin sufrir las restricciones potencialmente problemáticas de los sistemas formales. La última opción ha sido la escogida por los matemáticos interesados en estos menesteres. El lugar llevará el nombre de metamatemáticas y será un compendio epistemológico (lógica, matemáticas, filosofía, t. de la computación...) donde la intuición controlada no sólo es bienvenida, es imprescindible. Las demostraciones de este agreste territorio han tenido una clara inclinación hacia la *inteligibilidad* y cierto distanciamiento a la llana *computabilidad,* aunque como contraejemplo mediante la interacción pertinaz entre un usuario con su asistente binario algunos inconvenientes fueron zanjados y N. Shankar en conjunción con el NQTHM elaboraron una demostración formal del Teorema de Incompletud (CAP IV). Vale la pena respaldar y resguardar a los preciados teoremas de las matemáticas en la bóveda de los sistemas formales, algunas de las características de la bóveda pueden estar fuera de ella empero. De manera análoga varias de las garantías no absolutas sobre la validez de las demostraciones computarizadas pueden estar estipuladas fuera de las matemáticas.

Ya se mencionaron un par de contratiempos en la empresa de la elaboración de programas demostradores de teoremas: explícitamente al teorema de la indecidibilidad del CPPO e implícitamente y con menor detalle al teorema de la incompletud de Gödel. La integridad del teorema de incompletud de Gödel ha sido forzada por varias extrapolaciones arriesgadas, abriendo la puerta a conclusiones aventuradas y a veces descabelladas; es el precio de su fama. Por respeto a Gödel poco habré de agregar a lo enunciado por el

teorema. Sólo menciono una consecuencia inmediata y su implicación cercana: es imposible recoger en un sistema formal TODOS los enunciados aritméticos verdaderos[10], luego entonces cualquier estructura refugio de la aritmética no puede ser explotada completamente en cuanto a su riqueza verdadera por los sistemas formales. Es decir, en las áreas de las matemáticas donde habite la aritmética, el enfoque de los sistemas formales no acapara todo el trabajo, así deja un hueco para la intuición y la labor epistemológica. Sin embargo aun en esas regiones los exabruptos de la incompletud han sido pálidas aprensiones, los temidos enunciados indemostrables son bastante tímidos y generalmente se han presentado ex profeso. No obstante en el siguiente capítulo se exhibirá un teorema indemostrable no tan artificioso en la "aritmética", teorema poseedor de una demostración computarizada (Teorema Paris-Harrington y la demostración de Kunen-NQTHM). Debe quedar asentado, el teorema de Gödel establece un límite teórico y no práctico. Impide el éxito de la formalización global de las matemáticas pero no el de las demostraciones tratadas en particular. Similar a una de las consecuencias del teorema de la indecidibilidad, la incompletud sostiene la inexistencia de un sistema formal maestro en donde todas las demostraciones matemáticas pueden ser derivadas, así como tampoco hay un método mecánico para determinar si cualquier proposición del CPPO es o no es un teorema. Tirando el fardo de las panaceas, los problemas accesibles a nuestra disposición sucumben. Con metas menos ambiciosas se obtienen buenos resultados. Wang lo expresó contundentemente, incluso en las zonas de la indecisión con las herramientas lógicas adaptadas a la computadora se pueden conseguir las pruebas anheladas. Él hace más de cuarenta años predicó con el ejemplo, su programa probó todos los teoremas del CPPOI del *Principia.* Laureado por su honestidad, a raíz de su éxito al respecto confesó lo siguiente:

La lección más interesante de estos resultados es quizás que incluso en un dominio bastante rico, los teoremas realmente demostrados son principalmente aquellos que hacen uso de una porción muy pequeña de los recursos disponibles del dominio.[11]

Las proposiciones del CPPOI del *Principia* se prestaban a ser atendidas por un proceso de decisión gracias a su parvedad. Si nuestra meta es programar a las computadoras para probar teoremas, cabe estudiar y elaborar herramientas lógicas más poderosas aunque sin descuidar su finalidad de ser implementadas en la máquina. Los trabajos de investigación de notables lógicos matemáticos fueron la veta de algunos mecanismos lógicos para la binaria tarea de la demostración de teoremas. En especial destacan los trabajos de Herbrand y Skolem de la primera mitad del siglo XX. En la tesis doctoral de Herbrand[12] resalta su conspicuo teorema, base del desarrollo de varios programas demostradores de teoremas del CPPO (incluidos los de Wang). A grandes rasgos el teorema de Herbrand brinda una manera de lidiar con las fórmulas del CPPO reduciéndolas a una versión de ellas en el cálculo proposicional, en las cuales haciendo uso de procesos de decisión (determinar si son satisfacibles, insatisfacibles, tautologías) se puede dilucidar mecánicamente si son o no son válidas, si son o no son teoremas. Simplifico abruptamente para la brevedad y claridad de la explicación. Sea S un conjunto de fórmulas (axiomas, hipótesis, proposiciones a probar...) del CPPO universalmente cuantificadas. El universo de Herbrand de S está constituido por todos los términos que pueden ser compuestos con las funciones y

constantes encontrados en las fórmulas de S. Si no existen constantes en S el universo se genera con una constante arbitraria. La base de Herbrand de S se integrará por las fórmulas atómicas resultantes de realizar las sustituciones de las variables en las componentes atómicas de fórmulas de S por todos los elementos de H; si hay componentes atómicas en S sin términos variables entonces también ellas forman parte de la base. Una fórmula está aterrizada si no contiene variables. Una interpretación de Herbrand es una asignación funcional de la verdad sobre las fórmulas que pueden ser compuestas con las fórmulas de S y los términos del universo de Herbrand de la siguiente manera: a las fórmulas de la base de Herbrand empleadas en la composición se les fija arbitrariamente un valor de verdad y con base a las tablas de verdad de los conectivos lógicos se extiende la asignación funcional. Por ejemplo, supongamos F(a) perteneciente a la base de Herbrand empleada en la interpretación, si bajo ésta F(a) es verdadera entonces ¬F(a) debe ser falsa. Teorema de Herbrand versión ultraligera, S es insatisfacible sii existe un conjunto finito de instancias aterrizadas (en el universo de Herbrand) de S imposibles de satisfacer bajo toda interpretación de Herbrand. Es decir, S es insatisfacible si y sólo si existe un conjunto finito de instancias aterrizadas de S donde para cualquier asignación funcional de verdad sobre ese conjunto existe una fórmula en él que es falsa bajo esa interpretación de Herbrand. Ejemplo inocuo, sea $S=\{\forall x(P(x)\Rightarrow A(x)), P(c), \neg A(c)\}$. Demostrar la insatisfacibilidad de S equivale a demostrar $\forall x(P(x)\Rightarrow A(x)), P(c)+ A(c)$ (reducción al absurdo). El universo de Herbrand de S sólo estará conformado por la constante c; P(c) y A(c) forman la base de Herbrand de S. Debemos encontrar un conjunto de instancias aterrizadas de S insatisfacible bajo toda interpretación de Herbrand. Felizmente para nuestra causa la única composición de S con su universo $(P(c)\Rightarrow A(c), P(c),\neg A(c))$ es el conjunto buscado. Para corroborarlo, supongamos lo contrario y sea dicho conjunto satisfacible, es decir existe una asignación funcional de verdad donde P(c), ¬A(c) y $P(c)\Rightarrow A(c)$ son todos verdaderos. Pero como ¬A(c) es verdadera entonces A(c) es falsa; luego P(c) es verdadero y como "verdadero implica falso" es falso entonces $P(c) \Rightarrow A(c)$ es falsa. CONTRADICCIÓN. Por el teorema ultraligero de Herbrand S es insatisfacible.

Hace falta abordar muchísimos detalles, v.gr. ¿qué hacer cuando hay cuantificadores existenciales? pero en este instante serán ignorados. Basta con saber que existe una manera para demostrar teoremas del CPPO a primera vista apta para ser programada. Señalo con el dedo índice su dependencia en los procedimientos de decisión (satisfacibilidad,...) del cálculo proposicional. Para no dar lugar a dudas (aunque debe haberlas dada su precaria exposición), el camino marcado por la tesis de Herbrand no nos dirige al callejón de la contradicción rodeado por el teorema de la indecidibilidad del CPPO. Posibilita dar respuestas parciales y no la decisión total sobre la naturaleza da las fórmulas del CPPO. Si casualmente A es un teorema, entonces eventualmente llegaremos al aviso "A es un teorema" (¿cuándo? sólo el fantasma de Herbrand lo sabe); si no lo es; el pasmo procede a la ausencia de la información decisiva (**semidecidibilidad** del CPPO).

En los albores de los 1960's Hao Wang[13] junto con sus coetáneos Paul Gilmore[14], Dag Prawitz y Martin Davis-Hillary Putnam -entre otros- siguieron la guía del faro Herbrand para el desarrollo de programas demostradores en el cálculo de predicados. Todos ellos así

como cualquiera atraído por esa deductiva luz se enfrentaron y peligrarán por un par de puntiagudos escollos: la explosiva expansión del Universo de Herbrand y la ineficacia de los procedimientos de decisión cuando son implementados en las máquinas. Dag Prawitz se concentró con ahínco en la erosión del primer obstáculo y diseñó restricciones para la búsqueda en el Universo de Herbrand[15]. La idea impulsora de la mejora de Prawitz es concisa y eficaz: no generaras elementos del Universo de Herbrand en vano. Para llevarla a cabo Dag propone trabajar con la forma normal disyuntiva (FND) de las fórmulas (una fórmula tiene la FND si es la unión de fórmulas [sumandos] y cada una de ellas es la conjunción de fórmulas atómicas o su negación). El procedimiento de Prawitz va generando al conjunto finito buscado representándolo mediante una fórmula en FND, es decir va produciendo una fórmula en FND de longitud creciente aglutinando con el pegamento de la disyunción a las fórmulas de S (en su FND*) compuestas con el universo hasta conseguir una fórmula insatisfacible. Para que una fórmula en su FND sea insatisfacible todos sus sumandos deben ser falsos bajo cualquier interpretación, esto se cumple si en cada uno de ellos aparece una fórmula atómica (literal positiva λ) y su negación (literal negativa $\neg\lambda$) ya que los sumandos están constituidos por conjunciones y para que una conjunción bajo cualquier interpretación sea falsa debe contener a λ y $\neg\lambda$. Por lo anterior los elementos del universo generados serán aquellos mediante los cuales se puedan componer en cada uno de los sumandos alguna literal positiva y su negación, para lo grarlo se puede plantear un sistema de ecuaciones con base a los parámetros de la expansión. La generación de elementos innecesarios del universo ha cesado gracias al procedimiento de Prawitz. Sin embargo como lo apunta Martin Davis, dicho procedimiento no disminuye la tasa de crecimiento desmedido:

Desgraciadamente, las enormes expansiones de la fórmula en la forma normal disyuntiva generadas por todos los problemas con excepción de los más sencillos dejan claro que, al menos como es presentado, este procedimiento [de Prawitz] permanece siendo insatisfactorio.[16]

Intentando salvar el otro escollo, Davis-Putnam diseñaron un algoritmo más eficaz para revisar la satisfacibilidad de una fórmula del cálculo de proposicional[17]. Se le conoce como el procedimiento Davis-Putnam. Se aplica a la forma normal conjuntiva (FNC) de las fórmulas (conjunción de disyunciones [factores]). Posee las siguientes reglas:

- Regla de la cláusula unitaria: Si la literal λ es factor (cláusula [en la siguiente sección será definido este término]) de la fórmula (factor unitario) y dicha fórmula tiene otros factores distintos a éste se eliminan todos los factores que contengan a λ y además se borra $\neg\lambda$ en los factores donde aparezca. (aplicable si no existe un factor con λ y $\neg\lambda$). Si además λ la fórmula tiene como factor a la negación de λ, i.e. $\neg\lambda$ es factor unitario, entonces la fórmula $\{\varnothing\}$, fórmula cuyo único elemento es el vacío, se obtiene. A diferencia de la fórmula vacía, la cual por vacuidad es satisfacible, la fórmula $\{\varnothing\}$ es insatisfacible pues como carece de literales es imposible darle una asignación funcional de verdad.

- Regla de la literal pura: Si la fórmula contiene a la literal λ (en factores no unitarios) pero nunca a su negación y además existen factores que no contienen a λ entonces se puede eliminar todos los factores en donde aparezca λ.
- Regla de eliminación atómica: Si una literal está presente de forma positiva (es decir una fórmula atómica) en un factor y negativamente en otro (la negación de dicha fórmula atómica) elimínese ese par de literales en dichos factores y hágase la disyunción de ellos para crear el factor resultante de la eliminación atómica. Si no hay más literales en el par de factores donde se aplicó el borrado entonces el factor resultante será la fórmula $\{\varnothing\}$.

Si aplicando las reglas llegamos a producir el factor $\{\varnothing\}$ entonces la fórmula original es insatisfacible; de lo contrario, si ya no se puede simplificar más la fórmula obtenida por medio de las reglas entonces la fórmula es satisfacible. Ejemplo:

Revisar la satisfacibilidad de:

0. $(a\vee\neg b\vee c\vee d)\wedge(\neg a\vee b\vee\neg c)\wedge d\wedge(a\vee c)\wedge(a\vee b\vee c\vee\neg d)\wedge(a\vee\neg c)$
1. $(\neg a\vee b\vee\neg c)\wedge(a\vee c)\wedge(a\vee b\vee c)\wedge(a\vee\neg c)$ Regla de la cláusula unitaria sobre d
2. $(a\vee c)\wedge(a\vee\neg c)$ Regla de la literal pura sobre b
3. $(a\vee a)$ Regla de la eliminación atómica sobre c
4- FIN (ya no se puede aplicar ninguna regla)

Por lo tanto $(a\vee\neg b\vee c\vee d)\wedge(\neg a\vee b\vee\neg c)\wedge d\wedge(a\vee c)\wedge(a\vee b\vee c\vee\neg d)\wedge(a\vee\neg c)$ es satisfacible. Es más, una asignación de verdad que la satisface es a=V, b=V, d=V .y c=*.

El método funciona pues la fórmula simplificada por una de las reglas es satisfacible sii la fórmula original es satisfacible. Por lo tanto si mediante las reglas arribamos al factor $\{\varnothing\}$, la insatisfacibilidad se propaga como enfermedad hereditaria a la inversa hasta la fórmula original. De hecho con la regla de la eliminación atómica en solitario se puede indagar sobre la satisfacibilidad de una fórmula, el otro par de reglas sirven para aumentar la eficiencia del procedimiento.

Sin ánimos de recaer en una incisiva redundancia, otra vez lo menciono: los impedimentos físicos del instrumento influyen en la concepción de los algoritmos cuya implementación tienda a ser eficaz. Donald Loveland y George Logeman al programar las reglas del procedimiento DP constataron el excesivo uso de la memoria principal desencadenado por la regla de la eliminación atómica. En su lugar propusieron la siguiente regla dando origen al DPLL[18]:

- Regla de separación: Si la fórmula X contiene a la fórmula atómica p, sea X_p la fórmula obtenida de borrar todas los factores que contienen a p y removiendo $\neg$p de los factores donde aparezca. Sea $X_{\neg p}$ la fórmula conseguida al eliminar a los factores donde esté $\neg$p y borrar a p en los factores donde se encuentre. X es satisfacible si una de los dos fórmulas construidas ($X_{\neg p}$, X_p) lo es. Ejemplo:

0.$(a \vee b) \wedge (\neg a \vee \neg b) \wedge (a \vee \neg b) \wedge (\neg a \vee b)$

X_a $X_{\neg a}$

1.1 $(\neg b) \wedge (b)$ 1.2 $(b) \wedge (\neg b)$

Regla de la cláusula unitaria

2.1 $\{\varnothing\}$ 2.2 $\{\varnothing\}$

Ninguna de las dos fórmulas X_a, $X_{\neg a}$ es satisfacible, por lo tanto la fórmula original tampoco lo es.

¿Para qué dedicarle más de un párrafo al procedimiento DP /DPLL? Tres motivos impelieron la extensión poco profusa y nada profunda de su exposición. Con el DPLL se ejemplifica la materialidad impositiva del instrumento sobre el desarrollo de herramientas teóricas, repetido hasta el cansancio este punto merecía ser respaldado con hechos. La finalidad práctica de la regla de separación era hacer uso de unidades de almacenamiento secundario (cintas magnéticas) para guardar las fórmulas en espera de ser examinadas, liberando así espacio de la memoria principal[19]. Segunda intención, la regla de la eliminación atómica junto con la generación de literales con signos opuestos en el procedimiento de Prawitz son un preludio a la técnica resucitadora de la investigación acerca de la automatización de las demostraciones. Enfatizo y especifico, los logros adquiridos por los programas demostradores hasta los 1960's desde la perspectiva matemática fueron pequeñitos. El fastuoso comienzo con la Máquina Lógica (criticada por Wang como "mucho ruido y pocas nueces"), la conquista del Principia Mathematica de Wang (no tan glorioso pues ya evidenciamos la debilidad prenex del Principia) y muchos otros éxitos entran dentro la clasificación de "problemas fáciles" realizada por Larry Wos (gráfica en la introducción de la tesis), aquellos con un nulo aporte al desarrollo de las matemáticas aunque vitales para el avance de la automatización de la demostración de teoremas. Hacía falta una técnica más poderosa, capaz de impulsar a la empresa para confrontarse con problemas de un mayor nivel de dificultad, portadores de una relativa importancia –o no tan hartos de tanta insignificancia- dentro de las matemáticas. Sin embargo, dicha técnica traedora de buenas nuevas también chocó contra la infausta tercera causa de la longitud de la exposición del procedimiento DP/DPLL. El problema de la decisión de la satisfacibilidad, objetivo del misil DP/DPLL, resultó ser un bunker de gruesas paredes teóricas. Los métodos de este capítulo hasta aquí expuestos (el camino de Herbrand, procedimientos de Prawitz y DP/DPPL) y las técnicas venidas a continuación (res...) al final de cuentas buscan determinar la satisfacibilidad (o insatisfacibilidad) de un conjunto de fórmulas; si dicho problema de decisión presenta trabas inherentes entonces los mecanismos del actual capítulo sufrirán complejos apuros.

¿Por qué tanto barullo alarmista? ¿Acaso no es trivial determinar si una fórmula del CP en su FND es satisfacible? En primer lugar, aunque no lo clarifiqué en el justo momento (procedimiento de Prawitz), mecánicamente se puede traducir cualquier fórmula del CP a su forma normal disyuntiva. Luego, una fórmula en su FND es insatisfacible si en cada uno sus sumandos aparece una fórmula atómica y su negación; basta con un sumando libre de esta penuria contradictoria para que la fórmula sea satisfacible. Por lo tanto, en menos de lo

que canta un gallo una computadora competentemente programada puede recibir de entrada a cualquier fórmula en su FND y barrer con los sumandos para decidir si es o no satisfacib le. En términos un poco más precisos, la complejidad del problema de decisión de la satisfacibilidad para fórmulas en su FND es polinomial. Entonces, ¿por qué trabajar en la FNC como lo exigen los procedimientos de DP/DPLL? Porque aunque siempre se pueda convertir una fórmula a su FND, la complejidad de los algoritmos de traducción no es polinomial. En cambio, gracias al trabajo de G. Tseitin (1970) se conocen métodos eficientes para pasar a cualquier fórmula α del CP a una fórmula β en FNC donde α es satisfacible sii β es satisfacible[20]. Entonces, ¿el procedimiento DP/DPLL es la llave hacia el reino de la efectividad para los procedimientos de decisión de la satisfacibilidad? A pesar de los resultados positivos sobre la viabilidad de trabajar en la FNC conseguidos por Tseitin unos cuantos años después de la génesis de los procedimientos DP/DPLL, el optimismo teórico duró poco tiempo En fechas aledañas (1971) el aguafiestas de Cook entronó al problema de la satisfacibilidad para las fórmulas en su FNC como el padre completo de los problemas de la clase NP. Los programas demo stradores basados en la decisión sobre la satisfacibilidad miraron la solidificación de su peor amenaza.

Antes de disfrutar la opresión del teorema de Cook, puntualizo algunos conceptos mencionados a medias en el primer capítulo.
Sea f : N ([0,+() Se define el conjunto de funciones de orden O(f) como el conjunto de todas las funciones g : N ([0,+() para las cuales existen c(R y n0(N tales que g(n)(cf(n) p.t. n(n0. El tiempo de ejecución de un algoritmo alude a una función para estimar el número de operaciones básicas realizadas por el algoritmo con base a las características de los datos de entrada (usualmente su tamaño, i.e. el número de datos)
El orden de (la complejidad de) un algoritmo es O(f) si su tiempo de ejecución para el peor caso (aquel para el cual se requieren más operaciones) pertenece al orden O(f). Los algoritmos cuyo orden puede ser acotado por un polinomio se llaman algoritmos de tiempo polinomial. Los algoritmos de orden O(ln(n)), O(x), O(x2),....,O(xk) son de tiempo polinomial. Un problema de decisión pertenece a la clase P si mediante un algoritmo de tiempo polinomial puede ser solucionado. Equivalentemente, ((P si existe una máquina de Turing M(cuyo accionar toma la decisión en un tiempo polinomial. El problema pertenece a la clase NP si las supuestas respuestas de tono afirmativo pueden ser corroboradas en un tiempo polinomial. Ejemplo: el problema de decisión de la satisfacibilidad de las fórmulas en su FNC (bautizado con cariño como SAT). Si cualquier asignación de vabres de verdad se presenta como testigo y confiesa satisfacer a la fórmula entonces su testimonio puede ser sencillamente verificado (en todo factor debe existir una literal verdadera bajo la asignación) finiquitando así el litigio de la decisión. Cabe recalcar la indefinición en el sentido inverso, la verificación sólo asiente a los candidatos afirmativos, si no la pasan no se deriva su pertenencia al polo contrario de la decisión. Ejemplo de esto el SAT, la asignación candidata a satisfacer a una fórmula puede no adjudicarle un valor de verdad a todas la componentes atómicas de ella; luego si dicha asignación no pasa la prueba de la verificación entonces no se puede concluir la insatisfacción de la fórmula por la parcial asignación .

Ahora bien, dar con el testigo clave no es tan simple como validar su testimonio. De hecho la clase NP también es caracterizada como la clase de problemas resolubles en un tiempo polinomial por máquinas de Turing no-deterministas (o por algoritmos no-deterministas). Las máquinas de Turing no-deterministas -a diferencia de las anteriormente explicadas- operan con base a una relación de transición en lugar de una función de transición; i.e. ante una combinación de estado y símbolo leído en la cinta la máquina puede efectuar simultánea e independientemente diversas acciones en lugar de sólo una; cuando uno de las múltiples rutas de acción (llamadas caminos de cómputo) llega a un estado final entonces hay una detención global de la máquina. Su símil menos formal muestra una definición parecida, un algoritmo no-determinista es aquel donde se presente la ubicuidad en los múltiples saltos a las subrutinas, permitiendo realizar varias operaciones al mismo tiempo. Conceptualmente un algoritmo no-determinista podría ejecutarse por un sistema de cómputo en paralelo con un número ilimitado de procesadores. No obstante, la noción de Turing-computabilidad permanece inalterada, todo problema al alcance de una máquina de Turing no-determinista también puede ser resuelto por una máquina de Turing normal. Donde si hay diferencia es en el campo de la complejidad, la brecha entre las capacidades de las máquinas deterministas con respecto a las no-deterministas sirve para segregar a la clase P dentro de la NP. Dentro, ya que usando la segunda caracterización de la clase NP directamente se identifica la contención de P en NP ($P \subseteq NP$); la contención en el otro sentido es una pregunta abierta y su respuesta es acreedora a un premio de un millón de dólares[21] junto con la veneración de toda la pandilla de computólogos y legos agregados. Aclaro, la catalogación en la clase NP no es definitiva, si un problema carece del salvoconducto P esto nada más indica nuestro desconocimiento de algoritmos de orden polinomial para resolverlo y no afirma categóricamente la inexistencia de ellos. La opinión general alimentada por años de experiencia en el asunto se inclina por la contención estricta $P \subset NP$ y $NP \not\subset P$, pero lo general poco importa mientras continúe siendo una opinión. La pregunta escrita con términos más comunes plantearía la siguiente interrogante estilo zen: si las respuestas son sencillas, ¿entonces las vías para llegar a ellas también lo son? Expresado con más palabras y menos claridad: si para un problema de decisión todo elemento portador del certificado “Sí, yo soy la solución” puede ser fácil(polinomial)mente verificado, entonces ¿siempre existe un método sencillo (polinomial) para solucionar al problema en cuestión? ¿NP=P? Antes de proyectar nuestras inquietudes desde el teorema de Cook necesitamos extender el área donde jugarán las sombrías dudas inquisitivas. Un problema de decisión A se reduce polinomialmente a un problema de decisión B sii:

- Existe un algoritmo polinomial para responder a A que tiene como subrutina a un algoritmo de B.
- Cada ejecución de dicha subrutina cuenta como un paso.
- El algoritmo es de tiempo polinomial.

Por la última condición el número de llamadas al algoritmo de B está acotado por un polinomio; cuando sólo se llama una vez la reducción se llama transformación polinomial. Observación inmediata: Si A se reduce polinomialmente a B y $B \in P$ entonces $A \in P$ (la composición de polinomios es un polinomio). Observación más amplia: Si A se reduce

polinomialmente a B entonces B es al menos tan complejo como A; para aseverarla se requiere analizar los dos casos $A \notin P$ y $A \in P$. Para el primero procedamos por reducción al absurdo, supongamos $A \notin P$ y $B \in P$, luego por hipótesis A se transforma polinomialmente a B y por la observación anterior entonces $A \in P$ ¡CONTRADICCIÓN!; por lo tanto B es tan complejo como A ($B \notin P$). Si $A \in P$, B es al menos tan complejo como A pues $B \in P$ o $B \notin P$.

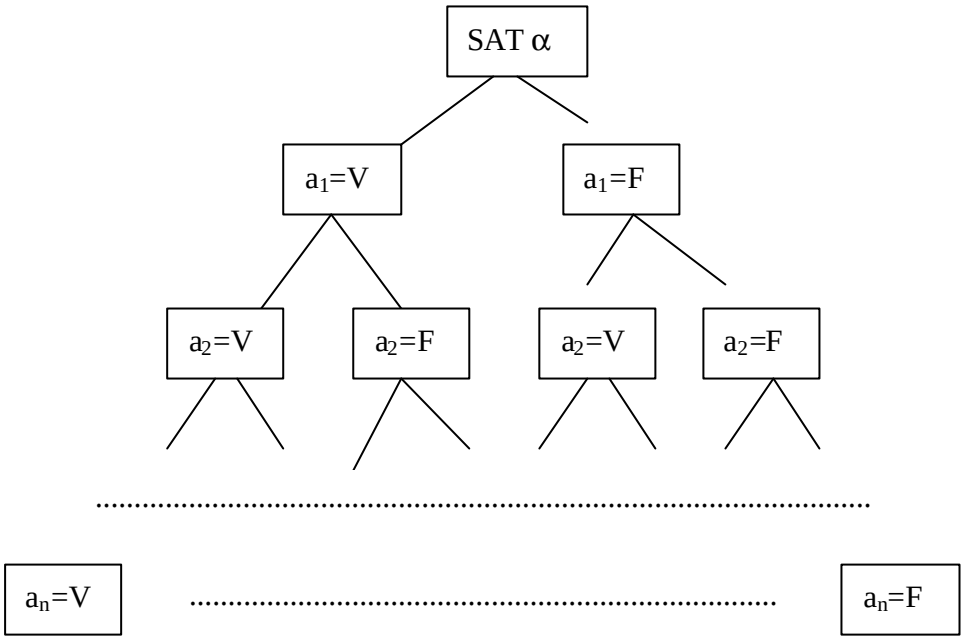
Delimitación de la clase clave: Un problema A es **NP-completo** si $A \in NP$ y todo problema perteneciente a la clase NP se transforma polinomialmente a A. Bajo la pregunta ¿NP=P? la clase NP-completos recibe especial atención pues con la elaboración de un algoritmo polinomial para uno de sus miembros se interrumpen las trémulas opiniones al borrarse los signos de interrogación, P=NP. La construcción de la clase NP es oligárquica, la membresía normalmente se adquiere por medio de relaciones con algún problema perteneciente al club; para agregar a B a la clase NP-completo usualmente se hace por medio de la transformación polinomial de alguno de los problemas ya establecidos de la clase NP-completo a B. Por fin estamos listos para enunciar el teorema de Cook.

TEOREMA DE COOK (1971)[22]: Sea A un problema de decisión. Son equivalentes:

(i) $A \in NP$
(ii) Existe un algoritmo no-determinista para solucionar A.
(iii) A se transforma polinomialmente a SAT.

BREVE BOSQUEJO DE LA DEMOSTRACIÓN[23].

(i)$\Rightarrow$(ii) Como A pertenece a NP entonces existe un algoritmo polinomial para verificar a toda respuesta certificada de A. Con base a este algoritmo de verificación se puede construir un algoritmo no-determinista capaz de resolver a A. ¿Cómo? Verificando en paralelo a las respuestas certificadas. ¿Cómo? Sacando provecho del poder demoníaco concedido por los algoritmos no-deterministas. Conviene representar al problema de decisión como un árbol de búsqueda donde cada rama agrupe a las respuestas certificadas y en sí la rama pueda constituir una de éstas (sin contar a la raíz). Gracias a los algoritmos no-deterministas el número de elementos por nivel del árbol puede crecer infernalmente a ritmos exponenciales, revelando así el verdadero poder de la sulfurosa no-determinación. Además como cada verificación conlleva un tiempo polinomial entonces en sí la creación y ejecución del árbol del conocimiento en un principio necesita un tiempo polinomial para dar con la decisión. Para la demostración hacen falta precisar muchos puntos, pero la estrategia general aquí ha sido definida y aquí también será ejemplificada con el SAT. Sea α una fórmula en FNC. Sean $(a_1, ..., a_n)$ el conjunto de componentes atómicas de α. Entonces un algoritmo no-determinista polinomial para decidir si α es satisfacible se representa por medio del siguiente árbol:



En cada k nivel del árbol se verifica si la fórmula es satisfecha por la asignación $\{a_1=V/F,...a_k=V/F\}$. El algoritmo no-determinista propuesto en el peor de los casos requiere n pasos para decidir la insatisfacibilidad de α. Entonces el algoritmo es una composición lineal de algoritmos de verificación polinomiales. Por lo tanto el algoritmo no-determinista propuesto es polinomial y resuelve el problema SAT.

(ii) $\Rightarrow$ (iii) Otra vez nos movemos en los linderos de las teorías y acudimos al truco de relacionar a las máquinas de Turing (en este caso no-deterministas) con las fórmulas de nuestro sistema formal (CP en FNC), Veamos por dónde va la táctica aunque sólo sea de lejos.

Haciendo uso del modelo abstracto y riguroso de algoritmo podemos afirmar la existencia de una máquina de Turing no-determinista (M_A) capaz de resolver a A en un tiempo n^k, es decir, para toda entrada la máquina para y arriba a un estado perteneciente al conjunto de estados finales determinado por las coyunturas de la decisión A ($A(x)=q_{si}$, $\neg A(x)=q_{no}$) en a lo sumo n^k pasos. Para toda entrada x, se puede construir una tabla $|x|^k \times |x|^k$ construida con base a los caminos de cómputo definidos por la relación de transición de M_A. A partir de esa tabla se puede construir una fórmula φ en FNC satisfacible sii en la tabla se mapea el camino de cómputo exitoso, i.e. aquel que nos conduce a uno de los estados finales de M_A a partir de la entrada x. En dicha tabla el cómputo inicial debe estar presente ($C_0=q_{in}x$), la relación de transición debe ser respetada ($C_i \rightarrow C_{i+1}$ sii $C_{i+1} \in \partial_A(C_i)$) y el estado final debe aparecer ($C_{|x|^k}=yq_{si/no}z$). Cada C_i representa una instancia de cómputo de la máquina, las cuales se integran por el estado actual (q_{si} en $C_{|x|^k}$), la cadena de símbolos presentes en la

cinta (yz en $C_{|x|^k}$) y la posición de la cabeza lectora/escritora ($yq_{si}z$ en $C_{|x|^k}$). La fórmula φ se atiene a las condiciones anteriores y las representa mediante una conjunción de disyunciones, además la longitud de φ es una función polinomial con respecto a |x|. De esta forma el algoritmo no-determinista de A puede transformarse polinomialmente al problema de satisfacibilidad para una fórmula en su FNC.

(iii) ⇒ (i) Recordemos, si A se transforma polinomialmente a SAT entonces para establecer la decisión se recurre una sola vez a SAT y listo. Entonces, cada respuesta candidata a consolidarse en una afirmación cuenta con un certificado de su validez, a saber, la fórmula en FNC generada por la transformación ocasionada por tal candidata. De nuevo, verificar si la supuesta respuesta afirmativa satisface o no a la fórmula puede llevarse a cabo en un tiempo polinomial. Por lo tanto, $A \in NP$.

Por el teorema de Cook y por definición SAT es miembro de la clase NP-completos. Es más, SAT es el padre de los problemas NP-completos[24]. A partir de él desde 1971 la fila de los afiliados a los NP-completos se ha ido engrosando aunque dicho sea de paso ninguno de estos tercos problemas a accedido a ser resuelto por un algoritmo de tiempo polinomial, respaldando la conjetura de la desigualdad $P \neq NP$. Los efectos del optimismo desmedido provocado por la igualdad también han frenado la creencia en ella. Asustémonos por las delicias del dulce porvenir. Pr(x,y) denote la relación "x es prueba de y" en el CPPO. Dicha relación es verificable en un tiempo polinomial, pues de ser cierta x debe ser una secuencia de fórmulas atenidas a las reglas derivativas del CPPO y la última fórmula de x debe ser igual a y. Por lo cual, el problema de decisión "¿x es teorema del CPPO?" (T(x)) presumiblemente pertenece a la clase NP. Si P=NP, entonces los matemáticos podrían preocuparse o aliviarse de tener un serio rival o competente asistente apto para hallar a las pruebas ansiosamente buscadas. Atención, la hipotética existencia de un algoritmo polinomial para T no origina un absurdo conflicto con el teorema de la indecidibilidad del CPPO, siempre nos conformaremos con algoritmos polinomiales gregarios o incompletos, es decir acogerán a cualquier fórmula del CPPO pero para algunas la decisión será errada o siempre su jucio acerca de si una fórmula dada es o no es un teorema será correcto pero existirán fórmulas para las cuales no puede ser establecida la respuesta. O siendo más ambiciosos, si P=NP podemos diseñar algoritmos segregacionistas y asintóticamente completos de T. Por ejemplo sea x una fórmula y nuestra intuición o suerte grande por lo que x es un teorema en el CPPO, entonces existiría un maravilloso algoritmo polinomial productor de una demostración de x en un tiempo relativamente pequeño –v.gr siguiendo la vía de Herbrand-. Cuando falla este algoritmo, o más bien cuando no acaba, es en la antípoda, si x no es un teorema entonces el algoritmo se queda trabajando como Sísifo. No obstante, con base a la complejidad polinomial del algoritmo se puede definir una cota adecuada para delimitar su jornada laboral, acercándonos a una efectividad muy alta en el procedimiento de decisión. La efectividad iría en aumento conforme se refinen los algoritmos y aumente la capacidad del hardware (esta relación se apreciará en un tabla más adelante). Lazo epistemológico en sentido inverso al previamente establecido, si pudiéramos resolver el Problema de la Detención entonces el CPPO sería decidible (sea M_H una máquina de Turing que pueda seguir la vía de Herbrand o cualquier otro algoritmo

segregacionista correcto; si $M_H(\alpha)$ se detiene entonces α es un teorema). Cuestionamientos similares se localizan en una carta fechada de 1956, para muchos este comunicado epistolar marcó el nacimiento silencioso de la teoría de la complejidad computacional. En ella, Gödel después exhibir diplomáticamente su pesar por la terrible enfermedad de su destinatario, lo aborda con una serie de preguntas en busca de su opinión. El destinatario era nada menos que John Von Newman, y la primera pregunta versaba sobre la complejidad de una máquina de Turing bastante especial:

Obviamente uno puede construir fácilmente una máquina de Turing que para cada fórmula F en el cálculo de predicados de primer orden y cada natural n, permita decidir si existe una prueba de F de longitud n. Sea $y(F,n)$ el número de pasos que la máquina requiere para hacerlo y sea $f(n)=max_F y(F,n)$. La pregunta importante es que tan rápido $f(n)$ crece en una máquina óptima. Uno puede mostrar que $f(n) \geq kn$. Si realmente existiera una máquina con $f(n) \sim kn$ (o incluso $f(n) \sim kn^2$), esto tendría consecuencias de suma relevancia. A saber, obviamente implicaría a pesar de la indecisión del cálculo de predicados del primer orden que el trabajo mental de los matemáticos concerniente a las preguntas Sí/No podría completamente ser remplazado por una máquina. Después de todo, uno simplemente podría escoger un número natural n tan grande que cuando la máquina no entrega un resultado, tiene poco sentido pensar más sobre el problema. Por el momento me parece, sin embargo, que está completamente dentro del reino de las posibilidades que $f(n)$ crezca lento.[25]

Al menos en la época de la carta, Gôdel estimaba posible la sustitución de los matemáticos en las tareas Sí/No por las máquinas. Curiosamente, años de batallar con los programas demostradores de teoremas han derruido esa predicción optimista y como consecuencia muchos expertos rechazan la igualdad P=NP. Aún en la actualidad, los programas provistos de sofisticadas técnicas lógicas y heurísticas pueden trabajar sin descanso por varios días en poderosos sistemas en paralelo sin encontrar la solución al problema de decisión "x es un teorema del sistema formal X (el cual engloba al CPPOI)". Cito a Steven Rudich, especialista en complejidad computacional:

Con P=NP, podríamos generar, no sólo una prueba, sino le prueba más corta de F [F es cualquier teorema en lógica de primer orden] en un tiempo polinomial que dependa de la longitud de la prueba... Esto podría dejar sin trabajo a los matemáticos, aunque P=NP tiene consecuencias más amplias. Por ejemplo, es fácil de verificar si un diseño de un reactor de fusión fría funciona. Bueno, [si P=NP] entonces podríamos generar un diseño de un reactor de fusión fría. Pero, ¿por qué parar aquí? Si se cumpliera la esplendorosa igualdad, la sociedad podría producir aviones, puentes, edificios, cohetes, etc. óptimos. Todo esto suena increíble, y quizás esta es la causa por la que la mayoría de los teóricos cree que P simplemente no puede ser igual a NP[26].

Extraña argumentación proveniente de uno de esos teóricos escépticos de la igualdad P=NP, la experiencia física menoscaba la posibilidad del país de las maravillas. Rara pues a esos teóricos no los invade recelo material cuando hablan de máquinas de memoria infinita,

máquinas casi omniscientes (máquinas de Turing donde en su cinta viene de fábrica la solución del Problema de la Detención) y otras muchas quimeras teóricas con atributos parecidos a los divinos. Además desde un punto de vista teórico, hasta ahora no se puede sostener ni la igualdad ni la desigualdad. Aunque, nadando a favor de la corriente dominante alimentada por la intuición y el conocimiento cuasiempírico ("cuasi" pues en honor a la verdad tiene varios retazos teóricos) como borrego elijo $P \neq NP$[27]. ¿Qué efectos nocivos provoca esta desigualdad para el tema que nos atañe? ¿Cuáles son las consecuencias negativas del teorema de Cook para los programas demostradores de teoremas de los sistemas formales montados en el CPPO?

Observaciones elementales: la complejidad de un problema de decisión A y la complejidad de un algoritmo de A pueden no coincidir, no obstante la complejidad del problema es la cota teórica para todo algoritmo que lo soluciona mientras no se demuestre la existencia de uno con un orden menor; sobra decir que el problema de decisión de la satisfacibilidad de una fórmula en el CP y el SAT son dos caras del mismo problema y ambos tienen canas complejas. A primeras luces y como ya fue explicado, decidir si φ es o no es un teorema del CPPO (¿$T(\varphi)$?) y llevarlo a cabo mediante SAT comparten la misma clase NP.
Aunque, siendo más precisos, la vía de Herbrand marca una opción frecuente de los demostradores automatizados, en donde se opta por la certera determinación de T para uno de los polos de la decisión y para la otra simplemente no se responde (este subdominio del problema lo denotaré $T|_T$). Cuando α es un teorema del CPPO, $T|_T(\alpha)$ lo confirma, cuando no lo es, $T|_T(\alpha)$ contesta con su silencio. T y $T|_T$ son problemas aparentados pero distintos, por lo cual uno podría sospechar si $T|_T$ tiene una complejidad menor. Pero si $T|_T$ fuera polinomial, ¿acaso esto no contravendría a la catalogación NP de T? Pues, si $T|_T$ perteneciera a P, tendríamos uno de los casos ya explicado cuando supusimos P=NP, y disfrutaríamos de un algoritmo polinomial para T, claro, incompleto pero todo algoritmo de T es portador de esta discapacidad. Sabiendo el final de la historia, ningunos de los algoritmos/métodos diseñados hasta la época actual para atender a $T|_T$ es polinomial. Coincidencia descartada, en casi todos (al menos los dos descritos en este capítulo) bombean por sus reglas el problema de la satisfacibilidad. Así pues, avocarse a las cuitas complejas de SAT es un paso infranqueable.

SAT pertenece a los camorreros NP. Los algoritmos DP/DPLL no conceden sorpresas agradables, su orden no es polinomial. De hecho su eficiencia depende considerablemente del orden de la elección de la letras proposicionales (o literales) sobre las cuales se apliquen las reglas. Antes de proceder por la senda de las superaciones particulares, el marco global del inexorable pesimismo teórico (no práctico) debe quedar asentado. ¿Por qué se evitan a toda costa los problemas NP en la computación? La respuesta frecuente hace uso de la intimidación, arrojándonos a la peor situación posible para uno de tantos problemas NP y un nefando algoritmo para resolverlo. Por ejemplo, para el problema de la satisfacibilidad el método de las tablas de verdad puede tornar en una opción desastrosa. Imaginemos una fórmula no satisfacible compuesta por un número apabullante N de letras proposicionales, Nuestro pobre algoritmo propuesto debe barrer con 2^N asignaciones para dar con la insatisfactoria –en muchos sentidos- respuesta, desastre inminente si también consideramos

la longitud de la fórmula y todo el tiempo requerido para evaluar cada renglón ¿Qué tan grande debe ser N para alcanzar el colapso? Además, gracias al avance a alzados trancos del hardware, ¿acaso no es ocioso perturbarnos por las N's cuando en algún momento cercano esa barrera será saltada? En esos ratos donde los niños duermen y los hombres sueñan en el insomnio cuentos de hadas terroríficos para despertarlos con pesadillas, uno podría calcular algunas de las ignominiosas N's. Por ejemplo (quien no me quiera creer adelante, únase a legión de los desvelados) si N=100 entonces 2^N= 1,267,650,600,228,229,401,496,703,205,376 y se requerirían mmm... ¡hokus pokus ! 4×10^{13} años de cómputo [28]. Continuando con el teatro macabro de los números enormes, si N=1000 entonces 2^N rebasa por dos centenas de dígitos el número estimado de partículas en el universo[29]. Que baje el telón, esta función puede prolongarse sin mucha razón. Las generalizaciones deben aparecer adustas en el escenario, el peligro de la sobreactuación y el desasosiego de livianas conclusiones sólo en los ensayos son tolerados; las generalizaciones son una mina peligrosa, en cualquier instante por el aire salen despedazadas sus aseveraciones (!). Si bien podemos hablar de la enfermedad NP, en la práctica tratamos con los enfermos NP porque cada uno de ellos exhibe síntomas propios. El mal NP provoca ineficacia e incluso imposibilidad, la gravedad de los padecimientos y cuándo se activan depende en gran medida de los enfermos y su trabajo empero. A ver, ¿que tan grandes (número de sus diferentes letras proposicionales) y largas son las fórmulas en FNC empleadas por los matemáticos, lógicos, computólogos, ingenieros,...? Las respuestas pueden variar, pero al menos en las matemáticas y en la lógica las fórmulas no suelen ser tan colosales para llegar a la imposibilidad aunque si lo suficiente para sufrir de ineficacia y su posterior parálisis. Ejemplo relacionado con el tema, la vía de Herbrand hacia la demostración de teoremas en CPPO. Las enseñanzas de Herbrand imponen un uso exhaustivo de procedimientos de decisión sobre la satisfacibilidad de las fórmulas del CP (de hecho se busca la insatisfacibilidad de un conjunto de ellas [o una de ellas conjunción de las fórmulas del conjunto]), a lo mejor éstas no cuentan con un monstruoso número de componentes atómicas pero si son tantas fórmulas generadas por la gran expansión del universo de Herbrand como para conducir a los programas demostradores al páramo de la efectividad casi nula (i.e. el del trabajo interrumpido sin recompensas).Por este motivo Davis y Putnam decidieron mejorar los métodos de decisión, tornando al problema en SAT y diseñando su famoso procedimiento. Optimistas pensaron haber dado con el clavo [30], en la práctica se dieron cuenta de su error y fue confirmado por Cook en 1973. SIN EMBARGO, pertenecer a la clase NP (en particular SAT) no es sinónimo de la excomunión de la gracia de las computadoras. El trabajo de Davis y Putnam sembró desarrollo: desde su nacimiento en 1958 hasta la fecha se han apilado mejoras del procedimiento y aunque ninguna ha logrado la conversión a P el desempeño notablemente se ha incrementado. Antes de proseguir la defensa, debe ser enunciada una consecuencia peor de la membresía NP, asentando lo que se había quedado en el aire.

La siguiente tabla diluye a la esperanza en el desarrollo del hardware de las computadoras como aplacamiento de los gamberros NP:

Orden de complejidad	Mayor tamaño de la entrada resoluble efectivamente		
	Computadoras actuales	Computadoras 100 veces más poderosas	Computadoras 1000 veces más poderosas
n	N1	100N1	1000N1
n^2	N2	10 N2	31.6 N2
n^3	N3	4.64 N3	10 N3
2^n	N4	N4+6.64	N4+9.97
3^n	N5	N5+4.19	N5+6.29

¿Quién exclamó algún signo de asombro? La carrera del poder del hardware contra los NP's asemeja a una tortuga compitiendo contra Aquiles y en este caso ella partió después de él. Sigamos instigando a SAT y su quelonio algoritmo de las tablas de verdad. Entonces nos ubicamos cuando menos en el renglón de la idealizada tabla del orden 2^n. El tamaño máximo de la entrada para que la máquina escupa un resultado en un tiempo sensato (arbitrariedad concedida para los fines del análisis) es igual a N4; es decir la máquina a lo sumo realiza 2^{N4} operaciones. Si la computadora es 100 veces más poderosa entonces la máquina puede hacer $100(2^{N4})$ operaciones en el periodo permitido de tiempo, por lo que el tamaño máximo de la entrada será $\log_2(100(2^{N4}))=\log_2(100)+\log_2(2^{N4})=6.64+N4$. Observación inmediata: para hacer competitiva a la tortuga la velocidad de desarrollo del hardware debe ser al menos exponencial. Según la mítica ley de Moore[31], ley de carácter ambiguo entre lo normativo y lo descriptivo, la capacidad de los procesadores se duplica cada 18 meses. Fue pronunciada en 1965 por el pionero del Valle del Silicio Gordon Moore, cofundador del gigante Intel, y a muy grandes rasgos se aproxima al desarrollo real de la industria. O el desarrollo de los procesadores tiende a obedecerla, pues la ley de Moore se convirtió en un estatuto-meta de la rama. Sin embargo, su reinado parece próximo a expirar[32]. De todos modos, bañándonos con el rocío del diáfano optimismo lúdico, si se conservara la tasa exponencial de Moore entonces (simplificación exagerada, sin considerar otros factores vitales como la memoria) cada año y medio aumentaría el tamaño de los datos de entrada tratables a $\log_2(2^x(2^{N4}))=x+N4$. O sea cada 18 meses le sumamos un uno a nuestra anterior cota para el método de las tablas de verdad. La tortuga tiene rotas tres patas. *Mens sana in corpore sano*, en cambio los problemas polinomiales son enormemente beneficiados por el perfeccionamiento del hardware. A menos de que haya un salto cuántico en el desarrollo de los órganos de la computadora, el hardware no es la cura definitiva para los NP's.

Los contratiempos de la indecidibilidad y la incompletud son más cordiales, el campo de acción de los programas demostradores de teoremas de sistemas montados sobre el CPPO

se ve escasamente afectado por ellos. No hay un procedimiento de decidibilidad para el CPPO, no importa, trabajemos a sabiendas de que existe la posibilidad de toparse con una fórmula, falso profeta de la verdad o profeta de un futuro remoto, para la cual no funcione nuestro método diseñado. Los matemáticos sufren de esa indecisión también, luchan contra los supuestos teoremas, a veces triunfan e inscriben sus logros parciales o totales en los libros de texto y artículos pero otras tantas fracasan consagrando su esfuerzo en la memoria de lo perdido. Si nuestro sistema formal engloba a la aritmética, a lo mejor tenemos la fortuna de convivir con un indemostrable y agregar otro miembro a la ilustre familia. SIN EMBARGO, si la tarea de antemano tiene la marca NP, ¡compadézcanse pues sus programas a lo mejor sólo sirven para probar teoremas de bolsillo o conjeturas de juguete! El empeño y pujanza de quienes creían y creen en la empresa empezó a rendir frutos desde los 1970's, aunque las recompensas fueron más jugosas en los ochentas y en los 1990's por fin dieron el salto a la fama. ¿Cómo pasó? ¿La marca de la bestia desapareció? No, aprendieron a vivir con ella.

Algunos problemas NP en la práctica han sido resueltos efectivamente por una computadora. Ocurre como con la indecidibilidad y el trabajo de Wang, se puede trabajar un dominio del problema NP donde los algoritmos sean efectivos, es más, polinomiales. Nuestro conejillo de indias, el problema de la satisfacibilidad del CP, bajo ciertas condiciones es exonerado de la carga NP. Ya se mencionó una, si la fórmula están en FND entonces determinar su satisfacibilidad es una cuestión polinomial (a expensas de la traducción exponencial a la FND). Si el número de letras proposicionales de la fórmula es dos, el cobijo polinomial cubre la decisión (dato curioso, a partir de tres el problema descubre su feo rostro NP). Para las fórmulas corniformes (*horn clauses,* llamadas en honor y memoria del matemático Alfred Horn aunque tras una socarrona simplicidad abuso de la licencia literaria en la traducción) su satisfacibilidad o la falta de ella puede ser examinada en un tiempo polinomial (las fórmulas corniformes están en FNC y cada factor a lo sumo tiene una fórmula atómica no negada). Aunque alentadoras, estas condiciones no alivian las necesidades de los programas demostradores del CPPO. Ni modo, no hay remedio, la única manera de vencer al mal es encarándolo. Los algoritmos deben revisarse hasta el cansancio, para mejorar a los conocidos y elaborar algunos nuevos. Los algoritmos deben renunciar a su rigidez y abrazar a la fe heurística, sin ella difícilmente tendrán la fuerza para acceder al reino de los teoremas. En las estrategias heurísticas se localiza un paliativo de los NP's. (el padre LTM no estaba tan perdido). Debe ser enunciado, las técnicas heurísticas (a diferencia de la definición de Simon-Newell del capítulo I) no rompen forzosamente con el contrato algorítmico de obtener resultados. Por ejemplo, los procedimentos sencillos DP/DPLL descansan sobre brújulas heurísticas -¿qué variables se escogen para aplicar las reglas?- cuya calibración ha mejorado gradualmente. En la siguiente sección se discutirá el trascendente caso de la estrategia SOS dentro del área de los demostradores y cómo puede preservar el don de la completez refutacional. Las estrategias heurísticas consisten en una serie de reglas, desprovistas a lo mejor de justificaciones teóricas sólidas pero respaldadas usualmente por la experiencia, diseñadas para gobernar el rumbo de los procedimientos (algoritmos) en pos de incrementar su efectividad. En ocasiones sí rompen con la completud algorítmica: ofrecen mejoras al costo

de no contestar siempre. No obstante con diligencia se pueden ir aglutinando estrategias heurísticas con procedimientos algorítmicos para sopesar el estigma NP de los programas demostradores de teoremas. Tampoco son una panacea, las estrategias heurísticas mal diseñadas pueden aumentar la complejidad sin regalar nada a cambio, pues a veces exageran en sus demandas de cómputo causando una pérdida en lugar de una ganancia. Tampoco son meras corazonadas, las estrategias pueden tener ciertas bases teóricas. Por cierto, muchas innovaciones de los algoritmos pueden provenir de otras disciplinas además de las ciencias de la computación, en especial han sido bastantes fructíferas las intervenciones de la probabilidad y de la estadística. Es hora de poner un ejemplo: el algoritmo aleatorio UnitWalk de Edward Hirsch y Arist Kojevnikov para SAT[33]. Sus prótesis heurísticas lo hacen incompleto: cuando logra hallar una asignación que satisfaga a la fórmula, entonces responde correctamente "la fórmula es satisfacible". Sin embargo en caso de no encontrar tal asignación, no se puede concluir con total certeza la insatisfacibilidad de la fórmula.

A grandes rasgos el algoritmo genera una asignación aleatoria y en cada iteración hace los cambios pertinentes para acercarse (probabilísticamente) a una asignación que satisfaga a la fórmula (si ésta es satisfacible). De hecho incorpora y adapta la añeja idea la Regla de la cláusula unitaria del DP, pues los factores unitarios fuerzan una asignación de verdad. Si bien es incompleto en trueque da mayor rapidez. Además el algoritmo UnitWalk tiende probabilísticamente a ser completo, como ya se mencionó, si la fórmula es satisfacible conforme se aumentan las iteraciones la probabilidad de dar con la asignación buscada para cualquier asignación inicial aleatoria tiende a uno. Por otro lado en la práctica no le ha ido nada mal, desde 2002 anualmente se lleva a cabo un concurso de procedimientos-programas para SAT y la implementación de UnitWalk ganó el primer lugar en la categoría de aleatorios en el 2003. Por cierto en la dirección electrónica de la competencia SAT http://www.satcompetition.org/ destaca la elevada afluencia de participantes (55 en 2004), indicador inobjetable sobre la ausencia del marasmo NP para el desarrollo de nuevos algoritmos. Asombrosos datos pueden ser recopilados de los resultados de la competencia, por ejemplo el número de variables proposicionales y factores manejables para algunos programas ya tienen 6 ceros (información del 2004, ejemplo Zchaff[34]). NP is not the end. Tampoco desencadenará espasmos de incredulidad la saludable cantidad de programas demostradores del CPPO (en www.qpq.org aparecen 58), aunque en este capítulo tan solo se abordará a ras de agua al más "popular": OTTER. Para no dejar cabos sueltos, el orden de complejidad de SAT (versión completa) es $p(n)2^n$, el de UnitWalk $p(n)1.363^n$, luego entonces el orden de los demostradores del CPPO y agregados al menos debe rondar por $p(n)2^n$. También debe especificarse, si SAT ha recibido tanto interés no es únicamente por su utilización en los procedimientos para demostrar teoremas redactados en CPPO, tiene otros usos como el diseño de circuitos electrónicos y uno muy importante, motor de la investigación en su arranque no solo de SAT sino de los programas demostradores en general, la verificación formal de software/hardware, tema del capítulo cuatro.

Ser NP por el momento arroja a la vitrina de las rarezas de la ciencia ficción la situación especulada por Gödel sobre la instauración del efectivo régimen absoluto demostrativo del silicio y el fin del período de carbono en la región del CPPO y aquellos territorios aptos

para ser importados a ésta. No obstante la ficción tiene una amplia colindancia con la realidad, las espaldas electrizadas cuentan con una excelente capacitación para pizcar demostraciones dentro de los sistemas formales y cada vez ocupan más plazas. Por cierto, tampoco estaban copadas pues una minoría dentro de la comunidad matemática (lógicos sobretodo) trabajaba con los sistemas formales y lo hacía normalmente acerca y no dentro de ellos. Aunque el sello NP anule la garantías de éxito, la computadora puede asistir en la búsqueda de demostraciones de teoremas de interés para la comunidad matemática, escritos convenientemente en el CPPO pueden ser atacados por los programas demostradores y a su favor tienen varios laureles por sus conquistas de problemas abiertos, algunas cuantas líneas adelante se enunciarán victoriosos ejemplos. Asistencia es la denominación correcta, pues en la práctica para lograr el asalto sobre los problemas el arma requiere de un tirador perspicaz, la dupla hombre-máquina puede ser una combinación asaz eficaz. Suficiente con los contratiempos, es tiempo de ver el cómo. Adrede postergo hasta el final la pregunta más relevante para este ensayo sobre la naturaleza de las demostraciones manufacturadas.

RESOLUCIÓN

Un hombre, William F. Miller, plantó una solitaria semilla de investigación. De esa sola semilla florecieron numerosas y gloriosas flores: la introducción de estrategia, la formulación de una poderosa regla deductiva, el diseño del programa de razonamiento más eficaz del mundo, la respuesta a una pregunta que estuvo abierta durante sesenta años, y mucho, mucho más... Si fue la suerte, la coincidencia, previsión, o el plan específico de Miller, yo no lo he determinado todavía. La historia puede nunca revelar la naturaleza precisa del principio. Lo que puedo decir con certeza es que (con una alta probabilidad) la flor se habría muerto en la vid de no ser que en Argonne, tantos o quizás casi todos los descubrimientos claves en que los que el razonamiento automatizado ahora se basan hubieran eludido a nuestra investigación.[35]

Encomio de Larry Wos, actor estelar de la historia del razonamiento automatizado deductivo, sobre William Miller, impulsor de esa nueva área de investigación en el mítico laboratorio nacional de Argonne. Fue Miller quien promocionó en el amanecer de los 1960's el uso de las computadoras para demostrar teoremas en la división de Matemáticas Aplicadas del laboratorio, juntando y azuzando a los miembros originales de la nueva línea de investigación: Wos, George Robinson y Dan Carson. Fue Miller quien en 1960 le entregó a Alan Robinson el artículo de Davis y Putnam y le instigó para que programara el método DP en la computadora IBM 704 de Argonne. No obstante Alan Robinson fue más allá del mero encargo y sacó a escena pública en 1965 en un artículo de culto[36] a la técnica revolucionaria de la resolución binaria, publicación de inflexión en la historia del razonamiento automatizado. Enterado de los intentos de Davis, Putnam y Prawitz por hacer del camino de Herbrand una vereda transitable por una computadora, Robinson inspirado por ellos diseñó un atajo para evitar la infinidad de recovecos del universo de Herbrand y así llegar más rápido a las demostraciones buscadas. Además, en el artículo original no

publicado (el cual data de 1963) aparece otra regla de inferencia cuya importancia se explicará más adelante: la factorización. Ya es tiempo de escribir con el lenguaje de la rama (o de la nutria) y aclarar ciertos asuntos con definiciones concisas. La fuente de esta sección es el amigable libro de texto de Wos y Pieper: *A Fascinating Country in the World of Computing and Your Guide to Automated Reasoning* y si el lector gusta de la precisión lógica le recomiendo consultar los *Fundamentos lógicos del programa de razonamiento automático Otter* escrito por Favio Miranda Perea[37].

El **lenguaje de las cláusulas** (*clausal language*) es el de las fórmulas en FNC con cuantificación universal implícita, es decir, toda variable se asume bajo el alcance del "para todo" dentro del factor al cual pertenezca. Una **literal** es una fórmula atómica (literal positiva) o su negación literal (negativa). Una **cláusula** es un conjunto de literales en disyunción (cláusula equivale a factor con cuantificación universal implícita). La disyunción se representa con el signo "|". La conjunción se representa con un salto de línea. La negación se denota con el signo "-", si dos negaciones aparecen seguidas inmediatamente se considera su cancelación. Cada cláusula termina con un punto (".").
Por otro lado para representar al cuantificador existencial se hace uso de las funciones de Skolem; de este modo $\exists x\, \varphi(x)$ se sustituye por $\varphi(c)$ (c es una constante de Skolem) y $\forall x_1 ... \forall x_n\, \exists y\, \varphi(x_1,...,x_n,y)$ se remplaza por $\varphi(x_1,...,x_n,f(x_1,...x_n))$ (f es una función de Skolem). En el lenguaje clausular no se permite la repetición de literales idénticas (letras predicativas iguales con los mismos términos y el mismo orden de aparición de ellos) en una cláusula, pues por definición ésta es un conjunto de aquellas. Una cláusula sin literales negativas se llama **cláusula positiva** y viceversa (**cláusula negativa**). A una cláusula conformada sólo por una literal se le llama **cláusula unitaria**. Redactemos nuestras primeras frases en el lenguaje de las cláusulas para asentar lo escrito, frases provenientes de la teoría de grupos.

P(x,y,z) denota mediante una relación la operación binaria del grupo x*y=z.
"e" es la identidad del grupo.

Los axiomas de grupo se escribirían de la siguiente manera:

IDENTIDAD DERECHA $\forall x(x*e=x)$: P(x,e,x).
IDENTIDAD IZQUIERDA $\forall x(e*x=x)$: P(e,x,x).

INVERSO IZQUIERDO $\forall x \exists y(x*y=e)$: P(g(x),x,e).
INVERSO DERECHO $\forall x \exists y(y*x=e)$: P(x,g(x),e).

"g(x)" es la función de Skolem para representar el "existe y en función de x" ($\forall x \exists y$) en los axiomas de los inversos.

ASOCIATIVIDAD $\forall x \forall y \forall z\, (x*(y*z)=(x*y)*z)$
Por la notación relacional del producto necesitamos el uso de variables secundarias:
x*y=u, y*z=v , u*z=w.

Utilizando a estas variables la asociatividad del grupo se expresaría:
u*z=w sii x*v=w

Luego, se desdobla la doble implicación en la conjunción de un par de implicaciones tomando en cuenta la definición de las variables secundarias
((x*y=u ∧ y*z=v ∧ u*z=w) ⇒ x*v=w) ∧ ((x*y=u ∧ y*z=v ∧ x*v=w) ⇒ u*z=w)

Finalmente, en lenguaje clausular la asociatividad se escribiría:
-P(x,y,u) | -P(y,z,v) | -P(u.z.w) | P(x,v, w).
-P(x,y,u) | -P(y,z,v) | -P(x,v,w) | P(u,z,w).

A primera vista de la traducción de la asociatividad se evidencia la poca afabilidad del lenguaje clausular con los lectores humanos, trato peor reciben aquellos no acostumbrados a la rigidez sintáctica propia de sistemas lógicos o lenguajes de programación. El lenguaje clausual está diseñado para ser manejado por los programas de razonamiento deductivo, es un lenguaje hecho a la medida de las aptitudes de las máquinas y es un traje estrecho para los seres humanos; este punto será retomado más adelante y sobretodo en la última sección del presente capítulo.

Sigamos, un **conjunto** de **cláusulas** es **insatisfacible** si la conjunción de ellas es una fórmula insatisfacible. Toda fórmula α del CPPO puede traducirse a un conjunto de cláusulas β del lenguaje de las cláusulas conservando su (in)satisfacibilidad, i.e. α es (in)satisfacible sii β es (in)satisfacible. Para hacerlo se debe rescribir α en su forma prenex, luego se convierte la parte de la fórmula resultante en el interior de la cuantificación a su forma normal conjuntiva. Notemos que la cuantificación universal se distribuye en la conjunción ($\forall x(A(x)\wedge...\wedge B(x))\equiv\forall x(A(x))\wedge...\wedge\forall z(B(z))$) y la cuantificación existencial se distribuye en la disyunción ($\exists x(A(x)\vee...\vee B(x))\equiv\exists x(A(x))\vee...\vee\exists z(B(z))$. Los existenciales son sustituidos por funciones de Skolem, v.gr. si un existencial afecta a una misma variable en varias cláusulas entonces la misma función de Skolem debe sustituir a esa variable en todas las cláusulas sujetas al alcance del existencial[38]. Remembrando, gracias a Tseitin la traducción al lenguaje de las cláusulas puede implementarse efectivamente en un programa.

Por enésima vez, un término es una variable o una constante o una función cuyos argumentos son todos términos. Una sustitución se define con base a un conjunto de pares ordenados t_i/v_i, donde cada t_i es un término y cada v_i una variable distinta de la cláusula sobre la cual se aplica la sustitución, acción de remplazar todas las v_i`s de la cláusula por t_i. La cláusula D es una **instancia** de C sii D puede ser obtenida de C mediante una sustitución. La instanciación puede entenderse como una regla de derivación sustentada por el axioma del CPPO $\forall xA(x)\Rightarrow A(t)$ modus ponens con $\forall xA(x)$ y/o por equivalencias nominales ($\forall x\, A(x)\equiv\forall y(A(y))$). Aunque utilizada al por mayor por los programas de razonamiento automatizado deductivo, la instanciación no se emplea como regla de inferencia *per se* sino como regla intermedia para llevar a cabo el mecanismo indispensable y básico de la unificación. La **instancia común más general** (MCGI) de un par de literales

K, M es, si existe, una literal L tal que L es una instancia de ambas K, M y para toda L' instancia tanto de K como de M se tiene que L' es una instancia de L. Dos literales K,M se pueden unir sii existe una sustitución que aplicada tanto a K como a M produce K`, M` (ignorando los signos) iguales. Si lo anterior sucede entonces las literales se denominan **unificables**. Cuando una sustitución sobre un par de cláusulas genera un MGCI para un par de literales en cada una de ellas (sin considerar su signo) el reemplazo se llama **unificador más general** (MGU). Por fin, la **unificación** es el proceso de encontrar un MGU, si existe tal. Por definición, existe MGCI (sin signo) sii existe MGU. También de manera directa se deriva la unicidad del MGCI (sin signo) y MGU (salvo variantes alfabéticas).

Ahora bien, para muchas de las reglas a continuación expuestas y en particular la regla dorada de la resolución binaria se busca unificar literales con signos contrarios (¿se acuerdan de Prawitz?). La regla de inferencia de la **resolución binaria** deriva la cláusula C de las cláusulas A y B (las cuales asumimos no tienen variables en común) cuando A contiene a la literal K, B a la literal M, K y M son unificables y difieren en el signo. La cláusula C es obtenida encontrando un MGU de K y M, aplicando MGU tanto a A como a B produciendo A' y B' respectivamente, para finalmente hacer la disyunción de A'-K' con B'-M' (donde K'= -M'= MCGI, A'/B'-K'/M' denota la remoción de sólo la K'/M' originada por MGU en la K/M original y no la eliminación de cuanta K'/M´ aparezca en A'/B'). C es bautizado como **resolvente** de sus padres o ancestros inmediatos A y B.

De las cláusulas
P(y)| Q(y) | Q(a).
-Q(a)| R(z).

Se pueden obtener los siguientes resolventes,
cuando Q(y) se une con –Q(a): P(a)|Q(a) | R(z).
cuando Q(a) se une con –Q(a): P(y)| Q(y) | R(z).

De la definición se observa la absorción de la regla de la eliminación atómica del procedimiento DP en la resolución binaria. Además, el método de Prawitz fue la simiente del procedimiento de unificación. La generación espontánea es tan común en el conocimiento como en la vida. Ahora bien, ¿que buscamos con la resolución binaria? Demostrar por contradicción, algo similar a la localización del conjunto de fórmulas insatisfacibles sin la contrariedad de la explosión demográfica del universo de Herbrand. Antes de seguir la explicación de las metas varias nociones más merecen ser esclarecidas.

Debido a la subsecuente enunciación de nuevas reglas de inferencia, donde la resolución binaria ya hizo acto de presencia, debe indicarse como puede ser justificada su solidez lógica. Una regla de inferencia es lógicamente correcta si el conjunto de fórmulas derivadas (conclusiones) son consecuencia lógica del conjunto de fórmulas progenitoras de la deducción (premisas). Por ejemplo la instanciación, sustrato de la unificación, de manera no explicita ya fue defendida su solidez lógica y por consecuencia también la de la

unificación. Pues ser consecuencia lógica conlleva la herencia de la veracidad de las premisas hacia las conclusiones, es decir, si las premisas son verdaderas forzosamente las conclusiones también lo son. En el caso de la resolución binaria, si las cláusulas –P|...|Q. y P|...|R. son verdaderas entonces las instancias producidas por la unificación –P'| ...|Q'. y P'|...|R' también lo son; además como cada cláusula resultante de la MGU son disyunciones una de las literales en cada una de ellas debe de ser verdadera. Sin embargo, sólo una entre P' y –P' puede ser verdadera, s.p.g. P' es verdadera , y como por hipótesis -P'|,,,|Q' es verdadera entonces existe una literal S' en dicha cláusula distinta a –P' tal que S' es verdadera, lo cual asegura la veracidad del resolvente S'|..|Q'|..|R'. De esta manera, con más cuidado por supuesto, se puede demostrar la solidez lógica de la resolución binaria. A partir de aquí ya no se va a instigar a las demás reglas introducidas, son correctas lógicamente y este hecho puede ser revisado para quien ande con el humor para hacerlo.

Hay varios algoritmos para la unificación porque es el procedimiento de ignición de las reglas de inferencia descendientes de la resolución binaria e incluso en otras cuyo parentesco con la resolución se distingue precisamente por encender de igual forma. En todos ellos en primera instancia las cláusulas deben ser sometidas a un rebautizo general de sus variables para evitar repeticiones inadecuadas para su tratamiento algorítmico. Como la cuantificación universal se distribuye en la conjunción, podemos renombrar a las variables en las cláusulas y asegurar variables exclusivas para cada una de ellas. Ahora ya podemos seguir las indicaciones de uno de tantos algoritmos de unificación, el más sencillo. El algoritmo va de izquierda a derecha, atendiendo a cada símbolo conforme aparece en ese sentido en las literales a unificar. Se construye una tabla de sustitución para las variables de las cláusulas donde queremos unificar el par de literales. Para edificarla se aplican las siguientes reglas: si los dos símbolos son variables, entonces a una de éstas se le asigna la otra. De donde si ambas son idénticas, ninguna modificación de la tabla es requerida. Si los símbolos en cuestión no son ambos variables, entonces termina con fracaso si son constantes o funciones distintas. En caso de ser iguales, sigue el barrido de las literales. Cuando se parea una variable contra una función o una constante, a dicha variable se le debe asignar en la tabla a tal término completo. Y se acabó el algoritmo, simple y correcto ya que siempre encuentra la MGU si las literales son unificables. Se le pueden hacer modificaciones y mejoras (por cierto existen algoritmos de unificación de orden lineal[39]), en este capítulo mucho más adelante se mencionará una de ellas (AC-unificación).

Para poseer los elementos necesarios para erigir al teorema de la completez refutacional explico brevemente la otra regla de inferencia estipulada por Robinson en su artículo original: la factorización. La cláusula D, llamada **factor**, es generada de la cláusula C por factorización cuando C contiene dos literales K,M unificables con el mismo signo. La cláusula D es obtenida al aplicar a C un MGU que una a K y a M y removiendo a K' (o a M', da igual mientras sólo se borre uno de los dos) de C'. Ejemplo:

De la cláusula: P(a)| P(x)| Q(x). por factorización se puede obtener las cláusula: P(a)|Q(a).

Últimos detalles previos al teorema: padecemos un **conflicto unitario** si están presentes un par de las cláusulas unitarias unificables integradas respectivamente por una literal y su negación. El conflicto unitario allana a un conjunto de cláusulas con una contradicción y la satisfacibilidad de dicho conjunto queda vejada. También ya fue señalado en el procedimiento DP el impedimento de la satisfacibilidad por la cláusula $\{\varnothing\}$.
Por fin emerge el objetivo primordial de la resolución binaria, establecer pruebas por reducción al absurdo. Una prueba será una secuencia de fórmulas donde cada una de ellas será un axioma o un teorema anterior o el resultado de la aplicación de nuestras reglas de inferencia (resolución, etc.) a otras fórmulas anteriores en la secuencia. En un sistema formal si a partir de los axiomas $\sum$ buscamos probar $\Omega ?\ \alpha$, el camino de la reducción al absurdo a partir de Ω y $\neg\alpha$ nos arrastra mediante las reglas de inferencia hacia la derivación de una contradicción β y $\neg\beta$, al ser aburridos y consistentes esto equivale a demostrar a $\Omega ?\ \alpha$. Por otro lado también equivale a establecer la insatisfacibilidad del conjunto $\{\sum,\Omega,\neg\alpha\}$. En nuestro lenguaje clausular, la demostración queda establecida deduciendo un conflicto unitario o a la cláusula $\{\varnothing\}$. Un conjunto de reglas de inferencia R goza de la **completez refutacional** si todo conjunto insatisfacible de fórmulas del sistema (cláusulas) después de un número de aplicaciones de R conduce a una contradicción (en nuestro caso a un conflicto unitario o a la cláusula $\{\varnothing\}$). Listo, ahora enunciemos lo prometido:

TEOREMA DE LA COMPLETEZ REFUTACIONAL: El conjunto de reglas de inferencia integrado por la resolución binaria y la factorización tiene la propiedad de la completez refutacional.

BOSQUEJO DE LA DEMOSTRACIÓN: Primer acto en el libro de Wos-Pieper, demostrar la completez refutacional para el caso de las cláusulas aterrizadas (aquellas sin variables). Basta con demostrar la llegada a la cláusula $\{\varnothing\}$ después de aplicar un número de veces finito nuestras reglas de inferencia si nuestro conjunto Ω de cláusulas es insatisfacible. De una vez se debe indicar por qué el teorema depende de la presencia de la factorización: esta regla es requerida pues sin ella a la solitaria resolución binaria se le escapa la insatisfacibildad del siguiente conjunto de cláusulas:

P(x)|P(y).
-P(z)|-P(w).

Así pues, gracias a la factorización la completez refutacional equivale en su primera batalla deductiva a demostrar la completud algorítmica de la regla de la eliminación atómica del procedimiento DP. Para extender el resultado cuando en las cláusulas hay variables se hace uso de un lema cuya morosa demostración no será desplegada, aunque fiándonos ciegamente del pulso intuitivo, rumor alimentado por las negras olas de líneas de esta sección, tampoco debe provocar mayor ruido:

LEMA DEL LEVANTAMIENTO: Si A' y B' son, respectivamente, instancias aterrizadas de las cláusulas A y B (asumimos la intersección vacía de sus variables), y si C' es el

resolvente de A' y B', entonces existen cláusulas E y F tales que son los ancestros inmediatos del resolvente C, donde C' es una instancia de C, E=A o E es un factor de A y F=B o F es un factor de B.

Después de anclar deductivamente el caso de las cláusulas aterrizadas el levantamiento ratifica al teorema cuando en las cláusulas hay variables.

Obedeciendo a inquietudes menos rigurosas esbozo en este párrafo un intento de demostración con el fin de sintetizar el contenido del teorema. Sean A, B las cláusulas padres del resolvente C. Afirmación: la conjunción de A,B es satisfacible sii C también lo es. Además, si el conjunto de cláusulas al cual pertenecen A y B es satisfacible entonces A y B son satisfacibles. Con la transitividad de las contrapuestas se concluye que si C es insatisfacible entonces el conjunto original de cláusulas es insatisfacible, de donde se atisba el arribo a la cláusula $\{\varnothing\}$ por medio de la resolución binaria y la factorización cuando el conjunto manoseado por dichas reglas sea insatisfacible. Diluyo deductivamente un sentido de la doble implicación para fortalecer a la comprensión; sea C satisfacible. Entonces C contiene al menos a una literal α depositaria del valor verdadero de C. Sea $\beta \in A, -\beta \in B$ el par de literales desencadenadoras de la resolución, s.p.g. $\alpha \in A$. Entonces si tanto a α como a $-\beta$ se les asigna funcionalmente el valor de verdadero habremos satisfecho a la conjunción de los padres. ¿Cómo garantizar la derivación de la cláusula $\{\varnothing\}$ a partir de Ω? Por reducción al absurdo, supongamos Ω' insatisfacible tal que $\{\varnothing\} \notin \Omega$' y en el cual ya no podemos aplicar la resolución binaria. Para que toda asignación funcional genere al menos una cláusula falsa, entonces debe existir al menos una letra proposicional en una de las cláusulas y su negación en otra de ellas, es decir, literales unificables de signos opuestos. Pues de lo contrario, todas las literales en las cláusulas serían independientes y trivialmente se podría dar con una asignación funcional adecuada para satisfacer a Ω' (OBSERVACIÓN: esta argumentación fallaría si no contásemos con la factorización). Por lo anterior podemos seguir aplicando la regla de la resolución binaria en Ω'. CONTRADICCIÓN.

Para variar tenemos un teorema de aspecto positivo, aunque como pasó con los tres teoremas "negativos", las apariencias engañan; no todos los perros que ladran muerden ni todos los que no ladran lamen la mano de los viandantes. La resolución binaria conjuga las ventajas ofrecidas por los trabajos de Davis, Putnam y Prawitz, armoniza lo mejor de lo mejor; por un lado la explosión del universo de Herbrand se retrae y evita, objetivo buscado por Prawitz, por el otro la insatisfacibilidad se establece por medio de uno de los métodos conocidos más eficientes, vástago de la regla de eliminación atómica de Davis y Putnam. Además, la unificación para la posterior resolución recoge las ideas de Prawitz de generar instancias útiles para derivar la insatisfacibilidad. Por si fuera poco, si a la resolución binaria le adicionamos la factorización entonces dicho conjunto de reglas de inferencia ofrecen la misma garantía que la vía de Herbrand, i.e. si de casualidad la fórmula a probar es un teorema entonces ambos métodos prometen y de contar con recursos ilimitados cumplirían con el hallazgo de la demostración. Demasiado bello para no ser complejo. La contribución resolutiva de A. Robinson causó mucha expectación y

esperanza, tanta como darle un nuevo empuje a la investigación sobre la automatización de la demostración de teoremas e incluso al área de la inteligencia artificial confiada y confinada en los sistemas deductivos. Más acá de todas las ambiciosas aplicaciones, la resolución guardaba un complejo secreto a voces, el cual por fin fue revelado por Armin Haken (¡oh no, otro alborotador apellidado Haken!). Hace algunos líneas se mencionó la relación entre el método DP con la resolución binaria. De dicha relación podrían incriminarse al método demostrativo de la resolución binaria y a su socio la factorización con el delito de pertenecer a la banda NP. Aunque basta con inculpar a la resolución binaria para revelar los nexos NP delictivos. El soplón de A. Haken en 1985 presentó una prueba acerca de la asociación de la resolución con la pandilla NP[40]. A resumidas cuentas denunció un caso donde el algoritmo de SAT basado en la resolución (DP con solo la regla de eliminación atómica) incurre en el delito exponencial. O sea, encontró un conjunto de cláusulas aterrizadas para el cual la resolución debe efectuarse muchas[muchas] veces en pos de determinar la insatisfacibilidad del conjunto. La fórmula en FNC detonadora del estallido exponencial representa el Principio de las Casillas (*Pigeonhole principle*), principio clásico dentro de las matemáticas enunciado por Dirichlet en el siglo XIX; coloquialmente expresa lo siguiente: si tengo m palomas y m-1 nidos entonces al menos un nido debe contener dos palomas (*pigeon*~ paloma). Otra vez fue el complejo S. Cook en conjunción con Reckhow quien tradujo por vez primera el PC al cálculo proposicional[41]. Haken demostró el orden exponencial de la decisión de la satisfacibilidad del PC expresado en FNC usando a la resolución de por medio. Turán y Buss en 1988[42] refinaron el trabajo de Haken al hacer uso del Principio Generalizado de las Casillas: si tengo m objetos y n casillas (n>0), entonces al menos existe una casilla con $\lceil m/n \rceil$ objetos. Ejemplo del principio, si la población estudiantil de la Facultad de Ciencias es igual a 4000 entonces al menos existen: 334 alumnos con el mismo signo zodiacal (n=4000, k=12, n/k=333.333... $\lceil n/k \rceil$=334), 10 alumnos cuya primer berrido ocurrió el mismo día, 145 con la misma primera letra de su apellido paterno, 1000 con el mismo tipo de sangre ... Sea $PGC_{m,n}$ la siguiente fórmula en FNC:

$$\left(\bigwedge_{i=1}^{m}\bigvee_{j=1}^{n} x_{i,j}\right) \wedge \left(\bigwedge_{j=1}^{n}\bigwedge_{1\le i_1<i_2\le m} (\overline{x}_{i_1,j} \vee \overline{x}_{i_2,j})\right)$$

donde m denota al número de objetos, n al de casillas, $x_{i,j}$ al predicado “el objeto i está en la casilla j” y la barrita encima del predicado previo a su negación. Así pues, la primera parte de la conjunción principal se podría leer como “todo objeto ocupa al menos una casilla” y la segunda parte “ningún casilla tiene más de un objeto”. Si m>n la insatisfacibilidad de $PGC_{m,n}$ equivale al Principio generalizado de las casillas. Turán y Buss demostraron el siguiente teorema:

Toda prueba por resolución de la insatisfacibilidad de $PGC_{m,n}$ tiene una longitud de al menos

$$\frac{1}{2} \cdot \left(\frac{3}{2}\right)^{\frac{1}{50} \cdot \frac{n^2}{m}}$$ líneas.

La demostración es demasiado técnica, tanto como para ni siquiera hacer un esbozo de ella. (Dato curioso, S. Cook y A. Haken algunos años después trabajarían juntos y de su real e imaginaria unión nacería un artículo cuyo marco es nada complejo de adivinar - *An Exponential Lower Bound for the Size of Monotone Real Circuit*, Journal of Computer and System Sciences 58 , 1999, pp. 326-335) Nada nuevo bajo el SAT. La resolución tampoco huye del hoyo negro NP. Sin embargo, cuando las cláusulas contienen variables la resolución y la factorización superan por mucho la efectividad de la vía de Herbrand, aunque esta noticia es poco novedosa pues precisamente las reglas surgieron de los arreglos de esa vía. Drásticamente se contiene la expansión del universo de Herbrand, al no requerir aterrizar todas las cláusulas desde las alturas universales se aprecia mejor el camino hacia la contradicción. Pero, la teoría de la complejidad deshace cualquier pretensión de la resolución a convertirse en una escalera al cielo de las demostraciones. Para bien o para mal, parece imposible contar con oráculos, máquinas capaces del despido masivo de los matemáticos. Aunque la guerra deductiva no se pueda ganar, numerosas victorias han sido alcanzadas por los demostradores automatizados. Cada batalla remite a una perenne revisión de nuestros triunfos y derrotas, la crítica ha sido impulsora del cambio constante: las medras en los algoritmos, las adecuaciones de las reglas a los problemas, la creación de nuevas técnicas, el desarrollo de estrategias heurísticas, el aprovechamiento de los avances del hardware, la prueba y el error para ya no errar tanto en la consecución de las pruebas. El paradigma de investigación del departamento de Razonamiento Automatizado Deductivo de Argonne sintetiza, en palabras de su miembro conspicuo Larry Wos, una ruta viable para el desarrollo del área:

Brevemente, el paradigma consiste en hallar a un teorema cuya prueba está más allá del alcance del programa actual, inventando una estrategia, regla de la inferencia, o mecanismo para traer a la prueba dentro del rango, para luego evaluar el propuesto avance. La evaluación se enfoca primero en los teoremas estrechamente relacionados, luego en teoremas distantemente relacionados, para finalmente probarlo con teoremas no relacionados. En otras palabras, el adelanto propuesto debe ofrecer generalidad, no debe funcionar sólo para el teorema con el que se comenzó el estudio.[43]

A continuación se explicarán sólo algunas de las reglas y estrategias fruto de este paradigma acumulativo y un par de los problemas abiertos arrastrados al alcance de los programas demostradores de teoremas (hay más de dos, pero esos dos son representativos y con ellos basta). Antes de proseguir, conviene desmitificar a las restricciones emanadas desde la teoría y desvanecer su apariencia de inefables e inquebrantables. Si bien Haken confirmó el confinamiento NP de las técnicas resolutivas, debe establecerse la existencia de países teóricos menos coercitivos y con mayores libertades polinomiales. Tsietsin, aquel hijo de la

madre Rusia mencionado en la conversión de las fórmulas a su FNC, en el mismo trabajo[44] cimentó un sistema de pruebas resolutivo en donde la satisfacibilidad de la fórmula del Principio de las Casillas tiene una prueba de longitud polinomial. Aquel paraíso lleva el nombre de sistema resolutivo extendido, el último participio denota la inclusión de un truco similar al usado en su conversión a FNC por medio del cual aumenta su potencia demostrativa. En este sistema se permite la creación de nuevas variables (sobre literales) con base a las literales existentes en la teoría (base axiomática y teoremas conocidos) y a las variables en el sistema extendido; así si x,y son literales de la teoría (o variables en la extensión) entonces en estos sistemas se puede introducir la variable $w \equiv x \wedge y$. La regla de inferencia principal de estos sistemas extendidos sigue siendo la resolución. Gracias a esta pequeña licencia de adicionar nuevas variables muchos de los problemas fuera del rango polinomial para los sistemas sencillos resolutivos tienen pruebas de longitud polinomial en el sistema extendido[45]. Desgraciadamente no está tipificada la adición de las nuevas variables en pos de la reducción de la longitud de las pruebas, por lo cual nadie hasta el momento ha implementado estos sistemas. La teoría aprisiona a nuestras aspiraciones aunque podemos caminar con sus cadenas, la teoría también ofrece libertades pero no siempre podemos gozar de ellas.

La resolución binaria ha servido como barro a partir del cual se han moldeado nuevas reglas y estrategias, cuyo diseño obedece distintas necesidades y persigue fines concisos. Aunque la fuente no sea muy fidedigna, pues se remite a una serie de conversaciones entre Larry Wos y George Robinson cuyo registro es la memoria de Wos[46], George se anticipó y propició el nacimiento de la resolución binaria pues él le enseñó a Alan Robinson en su visita a Argonne en 1962 la regla de la resolución unitaria. La resolución unitaria exige a uno de los padres ser una cláusula unitaria. Más allá de los cotilleos y los supuestos, el mismo Wos obtuvo la patente en 1963 de la estrategia unitaria, es decir, el fomento a aplicar la resolución binaria en un par de cláusulas cuando una de ellas sea unitaria. La justificación es transparente, nuestras derivaciones buscan ocasionar una contradicción, representada en el lenguaje clausual a partir de aquí por un conflicto unitario (por comodidad ya no descenderemos hasta la cláusula $\{\varnothing\}$), por lo tanto nuestra absurda meta impele a despedir a los unitarios progenitores incapaces de procrear al absurdo. Fue la primera estrategia sugerida por Wos, una entre tantas. Por cierto, Larry Wos ha sido uno de los principales promotores del uso de estrategias en los programas de razonamiento deductivo automatizado, es más, fue él quien bautizó esta área de investigación donde se mezclan la computación y las matemáticas como “razonamiento deductivo automatizado” al final de los 1970’s; antes ésta era conocida como “demostración mecánica de teoremas” (exhibiendo este nombre el espíritu de Wang y muchos otros lógicos) , también fue denominada “demostración automática de teoremas” (exudando un abotargado optimismo en las habilidades de las computadoras) cuando Wos acuñó un nombre irradiador del estado actual del área, donde la automatización parece distante a ser implementada por completo dada la naturaleza compleja del problema y ya no sólo se busca demostrar teoremas, muestras de otros usos son la verificación formal de programas (CAP IV) y el diseño de circuitos . Para despejar la noción de **estrategia**, merece ser transcrita la definición y clasificación redactada por uno de los máximos exponentes en la creación de ellas:

Una estrategia es una regla o un conjunto de reglas que gobiernan el uso de las reglas de inferencia. Si una estrategia constriñe la aplicación de una regla de inferencia, entonces es una estrategia de restricción. Si una estrategia dicta dónde sea llevada la siguiente aplicación de una regla de inferencia, la estrategia es una estrategia de dirección. Ambos tipos de estrategia son necesarios para que un programa de razonamiento deductivo automatizado sea efectivo.[47]

La estrategia unitaria puede colorearse con ambas tonalidades no excluyentes, si sólo permite la resolución unitaria entonces es una estrategia restrictiva, si favorece la resolución binaria cuando una de las cláusulas es unitaria entonces es una estrategia de dirección.

Otra regla resolutiva famosa lleva el nombre de hiperresolución, fue formulada por el mismísimo Alan Robinson en 1965[48]. Sean $A_1,...A_n$ un conjunto de cláusulas positivas, N una cláusula con sólo una literal positiva, el **hiperresolvente** B es obtenido a partir de N (núcleo) y de los A_i's (satélites) si existe un MGU tal que simultáneamente en cada A_i existe una literal positiva unificable con una literal negativa en N ; después de haber aplicado la MGU B se construye con la disyunción de todas las literales encontradas en los satélites (A'_i) y el núcleo (N') no unificadas. Si intercambiamos los positivos por los negativos definimos a la hiperresolución negativa.

Veamos en práctica el poder de la hiperresolución con la demostración del siguiente sencillo teorema de la teoría de grupos:
$\forall x(x*x=e) \Rightarrow \forall y \forall z(y*z=z*y)$, es decir si todo elemento del grupo elevado al cuadrado resulta la identidad entonces el grupo es abeliano.

Axiomas:
1)P(e,x,x).
2)P(x,e,x).
3)-P(x,y,u) | -P(y,z,v) | -P(u,z,w) | P(x,v,w).
4)-P(x,y,u) | -P(y,z,v) | -P(x,v,w) | P(u,z,w).

Hipótesis
5)P(x,x,e).

La negación de la conclusión, $\exists y \exists z(y*z \neq z*y)$, en lenguaje clausular utilizando constantes de Skolem la escribiríamos:
6) P(a,b,c).
7) -P(b,a,c).

Hiperresolución de 4 con 6,5,2
8) P(c,b,a)
La primera literal de 4 se unifica con 6, la segunda con 5 y la tercera con 2 bajo el siguiente MGU:

x:=a, y:=b, u:=c, z:=y, v:=e, w:=x

Hiperresolución de 3 con 5,8,1
9)P(c,a,b).
x:=y, y:=c, u:=e, z:=b, v:=a, w:=z es MGU empleado

Hiper de 4 con 9,5,2
10) P(b,a,c).

Conflicto unitario entre 7 y 10 !

La hiperresolución en un solo brinco de varias premisas llega a una conclusión, es más, la hiperresolución puede ser entendida como la aplicación simultánea de la resolución binaria sobre un conjunto de cláusulas todas menos una positivas y esta otra con solamente una literal positiva, rehuyendo de esta manera a la generación innecesaria de cláusulas intermedias. Cada vez queda más clara la orientación industrial de las reglas de inferencia, en general el lenguaje busca sacar provecho del instrumento en detrimento de la comprensión del usuario sobre el producto manufacturado. En la demostración previa, no fue tan complicado desglosar a las hiperresoluciones y sus correspondientes unificaciones (por cierto, ¿cuál es el MGU para el paso 10?), conforme las pruebas son más extensas y cada cláusula ha sido engordada con varias literales y términos la inferencia se enturbia para el ojo humano empero. No hace falta recurrir a una regla en donde en un paso se aplican varias derivaciones para provocar el ofuscamiento, con cláusulas henchidas de símbolos de donde se desprendan unificaciones difusas y con la aridez inherente del lenguaje clausular nuestro intelecto puede perderse en la forma y sus espejismos. Además, después de la laboriosa validación de las inferencias todavía hace falta una tarea igual de importante y no menos dificultosa, el vertido epistemológico sobre el recipiente de la forma y sus derivaciones. En el fondo, ¿qué representan las deducciones de nuestro ejemplo? Con alevosía y ventaja cambié el orden original de la exposición, en el libro de Wos primero enuncia una demostración arquetipo de la realizada por cualquier matemático promedio y luego con base a ésta determina la regla de inferencia recomendada en combinación con los axiomas e hipótesis útiles para producir a la subsiguiente (en nuestro caso anterior) prueba, sombra de la demostración del matemático común y corriente. Así, si un matemático supone la negación de la conclusión probablemente seguiría el siguiente patrón deductivo:

hipótesis: a*b=c , b*a≠c, x*x=e
por la buena definición de la operación binaria: (a*b)*b=c*b
por asociatividad , por instanciación de la hipótesis b*b=e y por axioma de identidad: a*(b*b)=a*e=a=c*b
por la buena definición de la operación binaria: c*a=c*(c*b)
por asociatividad , por instanciación de la hipótesis c*c=e y por axioma de identidad: c*a=(c*c)*b=e*b=b
por la buena definición de la operación binaria (c*a)*a=b*a

por asociatividad , por instanciación de la hipótesis a*a=e , por axioma de identidad y por hipótesis c=a*b :
c*(a*a)=c*e=a*b=b*a !

Por comodidad, elegancia o simplemente para facilitar la consumación de la comunicación no suelen apostillarse con mucho detalle todos los pasos deductivos, es más, podría procederse al nivel de las formas manipulando algebraicamente sin tanto burbujeo epistemológico y cualquiera con las mínimas nociones de álgebra podría dar con las justificaciones de las inferencias efectuadas. Cuando comienzan las omisiones en las demostraciones al *revisarlas* empieza el resanado epistemológico. Sin embargo, la *revisabilidad* de la demostración desde una perspectiva ideal se basa en la propiedad de las demostraciones de ser formalizables (*computables* bajo la notación propuesta), pues la forma depurada es el fondo del último estado representativo del conocimiento, acrisolado en la medida de los posible de cuanta ambigüedad se esconda en el lenguaje natural o en nuestra intuición matemática y explicitando con mucho cuidado todos los pasos tomados en la demostración. Así pues, las demostraciones bajo un ingenuo idealismo son sólo pruebas incompletas. Sin embargo, en nuestros sistemas formales las deducciones individuales tienden a ser lo más sencillas que se pueda pues las reglas de inferencia son el resultado de un exhaustivo proceso de destilación epistemológica; en este delicado punto las derivaciones efectuadas por las computadoras discrepan notablemente con las nuestras. El mismo (¿mismo con respecto a qué marco de referencia? tartamudo entre la forma dura y su asepsia de nosocomio o el blando fondo y su fertilidad de prostíbulo) Alan Robinson puntualizó la acción industrial de sus reglas resolutivas:

Tradicionalmente, las deducciones se desarrollan en pasos simples, por razones pragmáticas y psicológicas, pues los pasos sencillos, hablando en un sentido general, facilitan la aprehensión de lo correcto para un ser humano realizada en actos intelectuales simples. Sin duda esta costumbre se originó en el deseo de que cada paso elemental de una deducción debe ser indubitable, aunque la deducción en conjunto puede consistir en una cadena larga de esos pasos...[en comparación, la regla de la resolución] está orientada a las máquinas porque para obtener un mayor poder deductivo de lo que ha sido hasta aquí la norma, se exige mucho más esfuerzo computacional para aplicarla que el requerido típicamente por las reglas orientadas a los hombres.[49]

Con el mismo espíritu misántropo, Wos escribió acerca del enfoque de Argonne:

...nuestro acercamiento a la automatización del razonamiento difiere grandemente de los que uno a menudo encuentra en la inteligencia artificial; de hecho, nuestro paradigma confía en tipos de razonamiento y otros procedimientos que no son fácilmente o naturalmente aplicados por una persona.[50]

Ya no hay vuelta atrás. La automatización amplifica la inhospitalidad de los sistemas formales, las reglas ya no obedecen los manuales de cortesía epistemológica para con sus creadores. Lo importante es conseguir resultados sacando provecho de las capacidades de

las máquinas; ingenuo aquel con esperanzas de una manufactura artesanal cuando la mayor ventaja de las computadoras reside en su fuerza bruta. Talvez se llegue a un estado de desarrollo donde nuestros hornos industriales produzcan lindas o al menos siempre *revisables* demostraciones como las supuestamente elaboradas por los matemáticos, pero para eso falta mucho tiempo aunque el esfuerzo si se está haciendo (más adelante se citará un ejemplo). Pues, ¿quién va a querer consumir las demostraciones maquiladas sin un sello de calidad ortodoxo emitido por jueces humanos? Otra vez el panorama de desasosiego desatado por la demostración de Haken-Appel-Koch-computadora se vislumbra en el horizonte del razonamiento automatizado deductivo, al menos desde la ventana de aquellos interesados en sacar provecho de las características del instrumento y en canje relegan al entendimiento humano (he de decir que no son pocos). Sin embargo el problema todavía no ha explotado por completo como con la demostración de H&A&K&1010, por el momento las mercancías se encuentran en el estante fronterizo donde todavía los cubre la luz de lo *revisable*, aunque a veces sólo lo haga tenuemente. Dicho sea de paso, tampoco se ha logrado despedir a los inspectores binarios, ni parece sensato hacerlo. Ya empiezan a colocarse los elementos de la escenografía para montar la farsa de la sección postrera, pero todavía hace falta seguir diseccionando algunas de las reglas, estrategias y productos terminados; nada se discute si nada se sabe acerca del objeto de la disputa.

Fugazmente, hago mención de otra regla resolutiva. Similar a la hiperresolución por sus trancos, la UR-resolución genera el resolvente B a partir de las cláusulas unitarias $A_1 .... A_n$ al unificarlas de un solo MGU con n literales de la cláusula C y remover las n literales unificadas de B' (la formulación estándar además exige B unitaria ya que C debe estar integrada por n+1 literales). Esta regla fue una contribución a comienzos de los 1970's de otro miembro de Argonne de nombre Ross Overbeek, autor de una exitosa estrategia más adelante descrita. La UR-resolución puede considerarse una aplicación masiva de resoluciones unitarias, por lo cual además del ahorro de todas las cláusulas intermedias tiene la ventaja de generar cláusulas cada vez más cortas y próximas -si la suerte es próspera- a la contradicción. Por cierto, ante cada regla o estrategia nueva cabe cuestionar si posee o conserva la propiedad de la completez refutacional, pero no entra dentro de los alcances del escrito contestar propiamente esa pregunta para cualquiera regla o estrategia se cite. Como dato cultural la resolución unitaria junto con la factorización goza de la completez refutacional cuando sólo se manipulan cláusulas corniformes.

Pregunta desprendida de una de las líneas del anterior párrafo, ¿el uso de estrategias puede poner en riesgo la completez refutacional? La respuesta es afirmativa, para muestra un trivial ejemplo. Sean las reglas de inferencia la resolución binaria y la factorización actuando bajo la guía de la estrategia unitaria de manera restrictiva, i.e. sólo se permite aplicar la resolución cuando al menos una de las cláusulas progenitoras es unitaria. Entonces el siguiente conjunto de fórmulas insatisfacible desnuda la incompletud refutacional de nuestras reglas y estrategia escogidas:

P(a)|Q(b).
-P(a)| Q(b).

P(a)|Q(b).
-P(a)| Q(b).´

Aunque si la estrategia esta bien planeada entonces la completez refutacional puede salvarse. Así sucede con la estrategia de mayor abolengo, hija de la mente de Larry Wos con asistencia del parto por parte G. Robinson y Carson. Obedeciendo su paradigma de investigación, Wos y compañía decidieron poner a prueba un programa hecho por Carson con el teorema aquí ya probado de la teoría de grupos. Para su pesar el programa cayó fulminado ante un teorema de apariencia inerme, en consecuencia efectuaron una autopsia del fracaso de donde extirparon observaciones valiosas. En el cadáver destacaba la gangrenosa generación de cláusulas (alrededor de 2000) inútiles, su ponzoñosa cantidad incluso mató al programa por saturación de memoria. A partir de ese análisis el doctor Wos ideó una estrategia para generar cláusulas con mayores posibilidades de ser conductos hacia la contradicción. La llamó la estrategia del conjunto de soporte y saldría a la luz en el fecundo año de 1965[51].

Sea R un conjunto de reglas de inferencia que contiene a la factorización, S un conjunto no vacío de cláusulas. La estrategia del **conjunto de soporte** requiere la elección de un subconjunto no vacío T de S. Sea T_0 el conjunto de todas las cláusulas D tales que D está en T o D es factor de una cláusula C perteneciente a T. Sea T_1 el conjunto de cláusulas D tales que D es deducida aplicando alguna de las reglas de R a las cláusulas $C_1,....C_n$ pertenecientes a S o factores de cláusulas en S con al menos una C_i perteneciente a T_0 y una C_k que no esté en T_0 o D es un factor de una cláusula en T_1. T_2 es obtenido de manera similar a partir T_1, y de manera inductiva se puede construir T_n a partir de T_{n-1}. Una deducción T-soportada es una secuencia de cláusulas tal que toda cláusula en ella pertenece a alguna T_k ($k\in N$) o es una cláusula de S o un factor de una cláusula de S. A las cláusulas en T_i ($i\in N$) se les considera con T-soporte.

Por ejemplo, la estrategia del conjunto de soporte para la resolución binaria exige que al menos una de las dos cláusulas padres tenga T-soporte. En particular la estrategia prohíbe entonces la aplicación de una regla de inferencia a un conjunto de cláusulas subconjunto de S-T . Estamos entonces ante una estrategia de restricción, pero, ¿qué ganamos con ella? Evitar deambular por la teoría. Pues sea $\sum$ el conjunto de axiomas de nuestra teoría y $\Omega\Rightarrow\beta$ el teorema a demostrar; para hacerlo debemos derivar a partir de $\sum,\Omega,\neg\beta$ mediante nuestras reglas de inferencia en R una contradicción, lo que equivale a demostrar la insatisfacibilidad del conjunto de cláusulas integrado por las cláusulas de $\sum,\Omega$ y $\neg\beta$. Ahora bien, intuitivamente es una pérdida de tiempo deducir cláusulas únicamente a partir de $\sum$ porque si nuestra teoría está bien hecha los axiomas y las cláusulas derivadas de ellos son verdaderos, así pues damos un paseo por las nubes impolutas de la teoría cuando en realidad queremos retozar en el fango de la contradicción. La cláusula $\{\varnothing\}$ o el conflicto unitario deben esconderse en las derivaciones donde se involucren las cláusulas de Ω y $\neg\beta$. Formalizando el anterior argumento tenemos el siguiente teorema fundamental:

TEOREMA DEL CONJUNTO DE SOPORTE. Sea R el conjunto conformado por la resolución binaria y la factorización; además sea S un conjunto no vacío de cláusulas

insatisfacible. Sea $T \subset S$. Si S-T es satisfacible entonces la estrategia del conjunto de soporte conserva la completez refutacional para R.

BREVE ALEGATO PARA EL CONVENCIMIENTO: La herramienta del convencimiento esculpida con más detalle se encuentra en el libro de Wos o con más rigor en el opúsculo del Doctor Miranda[52]. Prefiero aquì emprender otra artimaña discursiva menos sólida pero con algunos visos de ser correcta; apelo más a la intuición y acepto su propensión al error. Sea S un conjunto insatisfacible de cláusulas aterrizadas. En primer lugar, si la cláusula $\{\varnothing\}$ o algunos conflictos unitarios de antemano pertenecen a S (o mediante la depuración inmediata de las cláusulas por medio de la factorización) entonces la única forma para que S-T sea satisfacible es incluir en T a dichas cláusulas de la contradicción. De donde se sigue trivialmente sin hacer realmente nada la derivación de la contradicción, manteniendo así la completez refutacional. Así pues, sea S insatisfacible donde no esté presente desde el principio la deshonrosa marca de la contradicción y las cláusulas de S hayan sido ya podadas por la factorización. Sea $T \subset S$ tal que S-T es satisfacible, de aquí se infiere $T \neq \varnothing$. Si de casualidad T es insatisfacible, entonces mediante la resolución podremos derivar la contradicción y por consecuencia la conservación de la completez refutacional. El otro caso es cuando la conjunción T y S-T es insatisfacible. Ahora bien, el truco radica en considerar a la resolución binaria como una pseudoperación binaria asociativa y conmutativa, me explico, si de la cláusulas σ_1 y σ_2 se obtiene el resolvente σ_{12} y con éste en combinación con σ_3 se resuelve σ_{123} entonces el último resolvente puede obtenerse también alterando el orden de las resoluciones, por ejemplo se puede resolver σ_2 con σ_3 para generar σ_{23} y éste a su vez se puede resolver con σ_1 para llegar de nuevo a σ_{123} (la resolución no es una operación binaria pues el resolvente de σ_1 y σ_2 no es único). Ya que S-T es un conjunto satisfacible de cláusulas aterrizadas, la regla de la resolución binaria sólo puede aplicarse un número finito de veces con sus cláusulas hasta generar (S-T)'. Ahora bien, como la conjunción de T y S-T es insatisfacible, deben existir $\sigma_1', \ldots, \sigma_n'$ cláusulas en (S-T)' que se resuelvan con $\tau_1, \ldots, \tau_m$ cláusulas en T para generar un conjunto de cláusulas T' a partir del cual se deriva la contradicción. Recurriendo a la asociatividad y conmutatividad de la resolución binaria, se puede afirmar que T' también puede ser generado desde un principio si resolvemos a $\tau_1, \ldots, \tau_m$ con $\sigma_1, \ldots, \sigma_k$ cláusulas de S-T y luego realizamos las resoluciones correspondientes con algunas de las cláusulas en S-T para al fin obtener de nuevo a T'. Bajo este orden, nuestras deducciones tendrán soporte y respetarán la completez refutacional de R. Para el otro caso, cuando S tiene variables, entonces simplemente hacemos uso del lema de levantamiento. (;o))

Se puede intentar sacar provecho al teorema del conjunto de soporte desde una perspectiva sintáctica: si T está constituido por todas las cláusulas negativas o por todas las cláusulas positivas de S cumpliremos con los requisitos del teorema, v.gr. para la primera alternativa S-T se satisface si asignamos el valor de verdadero a todas las componentes atómicas de S ya que por hipótesis en toda cláusula de S-T al menos debe haber una literal positiva (está bien definido T pues si S=T entonces S no es insatisfacible). Sin embargo, las sugerencias óptimas para explotar las ventajas del teorema provienen de la semántica. Como ya fue mencionado, si en verdad $\Omega \Rightarrow \beta$ es un teorema entonces intuitivamente los conjuntos de

soporte que incluyan a -β, Ω o simplemente a -β son fuertes candidatos a obedecer las exigencias del teorema del conjunto de soporte. Más aun, escoger la construcción sintáctica de T puede volver inoperantes a algunas reglas de inferencia. En particular, si nuestra regla de inferencia además de la factorización es la hiperresolución la elección de la constitución de T por todas las cláusulas negativas de S inhibe toda posible derivación con tal regla. Dicho sea de paso, la hiperresolución con la factorización portan el estandarte de la completez refutacional, corroborarlo no es cosa tan difícil y si lo es se puede acudir a la abnegada asistencia de los libros (como (35)).

Mediante el teorema del conjunto de soporte queda constatado que las estrategias no son un mero capricho, pueden tener un sustento teórico firme. Sin embargo para no perder la costumbre la teoría predica y la práctica acata cuando le conviene. La completez refutacional es una propiedad deseada para todo conjunto de reglas de inferencias bajo cualquier estrategia, sin embargo tampoco segrega a aquellas reglas ni estrategias no agraciadas por esa propiedad. Sin deseo también se procrea. En palabras de Wos:

Yo siempre he afirmado que nuestro campo presta demasiada atención a las propiedades como la completez refutacional. Tales propiedades son en verdad estéticamente placenteras, pero, en general, su presencia derrama poca o ninguna luz sobre la efectividad. Fuera algún individuo tácito a pedirme que le explicara cómo un programa puede tener éxito y puede tener éxito a menudo ante la ausencia de completez refutacional, yo sugeriría que la respuesta descansa en las increíblemente numerosas y distintas pruebas que pueden encontrarse para casi cualquier teorema dado. Yo sé que, cuando descubrí esta fecundidad, mi preparación matemática no me había preparado para ella; es más, sospecho que muchos matemáticos podrían sorprenderse ante tal fecundidad.[53]

Ante tal confesión, tampoco debe sorprender la identidad del autor del teorema del conjunto de soporte y su demostración: Alan Robinson, filósofo interesado en esas propiedades agradables para el espíritu aunque relevantes no revelan por completo las sendas de la efectividad. Debe mesurarse el comentario de Wos, quien ha estado más interesado en diseñar mecanismos para conseguir pruebas en lugar de toda la descripción meticulosa de la maquinaria subyacente. Con toda razón, la delineación precisa de los elementos lógicos involucrados en la demostración automatizada es un trabajo más apropiado para los lógicos-matemáticos o filósofos. Sin embargo, todo ese trabajo de apariencia inútil en algún momento puede tornar en fuente invaluable para medrar a la maquinaria. Acaso, ¿no fue el ocioso trabajo sobre lógica de un joven matemático de apellido Herbrand quien marcó el camino para los demostadores automatizados de teoremas del CPPO? Si conocemos las propiedades de los procedimientos, reglas, estrategias y en general de los sistemas deductivos, entonces tal vez podamos elaborar mejoras. Wos tampoco se desentiende del fundamento lógico en pos de un pragmatismo desinteresado en esas cuestiones, en su libro de texto *A Fascinating* ... dedica un capítulo completo a la discusión de algunos aspectos lógicos donde se inmerge el razonamiento deductivo automatizado (capítulo donde la mayor parte de esta sección se basa). Su comentario no impele a la desavenencia total contra la lógica inmiscuida, sino pregona

centrarse en el objetivo del área: la automatización del razonamiento deductivo y su nodo central, la demostración automatizada de teoremas. Por cierto, gracias a la implementación de la estrategia del conjunto de soporte por Carson en un programa ejecutado en una IBM 704, después de tres segundos se trenzó la demostración del hasta entonces inexpugnable teorema de la teoría de grupos. A favor de la opinión de Wos se ha evidenciado en este par de secciones del capítulo la falta de un veredicto definitivo emitido por cuanto teorema aquí ha sido enunciado. Ni los resultados a favor ni los resultados en contra determinan por completo el destino del desarrollo del razonamiento deductivo automatizado. Marcan límites, señalan fortalezas, indican direcciones, pero no apuntan hacia el fin ni explican cabalmente el latir de la historia. Es bueno saber que con un pincel y pinturas no se hace una escultura, o que con tres colores primarios puedo generar a los demás colores. Sin embargo no hay manual donde esté totalmente reglamentado como realizar un retrato de las demostraciones, se idean técnicas, se diseñan métodos, se mejoran las herramientas, pero el acto creativo ni con las computadoras puede ser sustituido, al menos no por el momento. La confesión de Wos es una advertencia y un consejo, el movimiento se demuestra haciéndolo. No obstante, para los objetivos de este ensayo resulta imposible emitir un juicio por más precario que sea si se ignora el proceso de producción de las demostraciones, pues el instrumento y las técnicas diseñadas en gran medida con base a éste imponen un sello en el producto maquilado. En concreto, en lugar de sólo desmenuzar demostraciones particulares se debe analizar al sistema de manufactura en conjunto, incluidas las placenteras y ociosas especificaciones lógicas. Además, los chapuzones en los ojos de agua teórico-prácticos son refrescantes y la mirada es bañada por sus gotas de historia. Prosigamos con el buceo por la sangre teórica y la corriente práctica, zambullámonos en el estanque de la memoria.

Otra anécdota del paradigma en acción: a partir de la revisión de un intento fallido para demostrar el imberbe teorema de la teoría de anillos xy=(-x)(-y), Wos se percató sobre la redundancia de las cláusulas derivadas; de 2000 verificadas, 1937 podrían obviarse si se rescribieran las expresiones del tipo x+0, x+0+0,..,x+0+...+0 como x. Para facultar a los programas con dicha capacidad de destilación de las redundancias se gestó el procedimiento de la demodulación en 1967 entre Wos y compañía[54]. La **demodulación** no es una regla de inferencia ni una estrategia, es un procedimiento para simplificar a los términos a una forma canónica; así se aumenta nuestro arsenal deductivo con un nuevo tipo de armamento. Por otro lado, introduce la unificación aplicada a términos en lugar de literales. Un **demodulador** es una cláusula positiva unitaria con un predicado de igualdad de la forma EQUAL(r,s). Sea t un término de una cláusula A. Si t es una instancia de r, cuando aplicamos un demulador a la cláusula A se elimina dicha cláusula y en su lugar queda el demodulado B, obtenido al remplazar t por s`, donde r'=t y EQUAL(r',s') es la correpondiente instancia de EQUAL(r,s), es decir, EQUAL(r',s') es la instancia de EQUAL(r,s) ocasionada por la unificación de t con r. En la práctica se puede optar también por la demodulación derecha-izquierda, buscando unificar t con s y sustituirlo por t=r', la convención común es la demodulación izquierda-derecha empero. La demodulación únicamente puede ser efectuada si al unificar t con alguno de los argumentos del demodulador no se afectan a los demás términos de la cláusula al realizar la sustitución.

Por ejemplo:

GREATERTHAN(SUC(x),x).
ninguno de sus términos puede ser demodulado con
EQUAL(SUC(1),2).
pues siempre se modifica al otro argumento de la cláusula.

En cambio
EQUAL(sum(sum(x,0),sum(y,z)),sum(sum(x,y),z)).
con el demodulador
EQUAL(sum(x,0),x).
genera la cláusula demodulada
EQUAL(sum(x,sum(y,z)),sum(sum(x,y),z)).

Cuando se permite agregar demoduladores en el curso de una demostración y son aplicados a las cláusulas retenidas en ese momento, dicho proceso se conoce como demodulación hacia atrás.

Con la demodulación se puso sobre la mesa de análisis un par de elementos claves: la igualdad y la unificación. La igualdad es la más cercana a la ubicuidad de todas las relaciones, el juego de reflejos irradiado por el espejo de un par de segmentos paralelos se proyecta por casi todos los rincones de las matemáticas. Intuitivamente transparente, teóricamente intrigante, la igualdad emerge a la vista del intelecto cuando el vacío se ve poblado por elementos, cuando puede uno dilucidar con la pócima de un marco espacial absoluto o elucidar bajo algún maleficio de lo atemporal que un objeto es igual a sí mismo; a partir de ahí empieza el festín de la simetría y la verbena transitiva. Incluso después de haber visto se puede afirmar que en la ceguera, el vacío es igual al vacío. No por nada Alan Robinson en un influyente artículo publicado en 1967[55] señalaba dos rutas a ser exploradas si esperamos que el razonamiento deductivo automatizado nos lleva a algún paraje matemático importante: el rumbo de la igualdad y el paso de la teoría de los conjuntos. Por otro lado, al trasladar a la unificación a los términos se consigue la llave de una ley de acceso a la teoría de la igualdad:

$$\forall x \forall y \, (x=y \Rightarrow P(x,x) \Leftrightarrow P(x,y))$$

El anterior axioma lógico responde al nombre de ley de Leibniz para el CPPO. “Todos los solteros son flacos” y “Todos los no-casados son flacos” son predicados equivalentes ya que “soltero”es una condición igual a ”no-casado”. Es más, explicando la notación, la sustitución en el predicado puede ser no uniforme, “si eres niño entonces tus pesadillas parecen espeluznantes desde tu perspectiva de niño” equivale a “si eres infante entonces tus pesadillas parecen espeluznantes desde tu perspectiva de niño” y también a “si eres niño entonces tus pesadillas parecen espeluznantes desde tu perspectiva de infante” pues “niño”=”infante”. Los ingredientes estaban listos para cocinar una nueva regla de inferencia.

La paramodulación (cuyo nombre deriva de la demodulación) es una regla de inferencia abocada a la ley de Leibniz. Fue formulada por Wos con ayuda de George Robinson en 1968[56] y ha sido una de las reglas más exitosas sacadas al mercado de los demostradores, al menos ha brillado en una de las direcciones puntualizadas por A. Robinson.
La **paramodulación** genera la cláusula C de las cláusulas A y B, cláusulas sin variables comunes, cuando A contiene una literal positiva cuyo predicado es el de la igualdad (EQUAL(r,s)) y B contiene un término que se unifica con uno de los argumentos de dicho predicado. Sin pérdida de la generalidad, el término t de B se unifica con r, gracias al hallazgo del MGU correspondiente y aplicándolo se obtienen A',B'. Sea K' la literal que represente a EQUAL(r',s'). En B' se remplaza t' por s', la cláusula resultante B'' se une disyuntivamente con A'-K' (i.e. se remueve la literal K' de la cláusula A') para finalmente conformar a C. A es la cláusula "desde" (*from clause*), B la cláusula "hacia" (*into clause*) y C es el paramodulante.

La paramodulación va más allá de la mera ley de Leibniz, pues las unificaciones pueden modificar a los términos tanto en la cláusula "desde" como en la cláusula "hacia". Por ejemplo:

Desde
EQUAL(sum(0,x),x).
Hacia
EQUAL(sum(y,minus(y)),0).
Se obtiene el paramodulante
EQUAL(minus(0),0).
con el MGU
y:=0, x:=minus(y)

A diferencia de la demodulación, la modificación de los demás argumentos está permitida, en el ejemplo de la demodulación fallida

Desde
EQUAL(SUC(1),2).
Hacia
GREATERTHAN(SUC(x),x).
se obtiene
GREATERTHAN(2,1).

Otra diferencia con la demodulación, las cláusula "hacia" no se borra. La retención de todas las cláusulas derivadas es una de las características de los programas diseñados por los miembros de Argonne, aunque no todos siguen esta táctica memoriosa (a quien le interesa saberlo PROLOG y sus secuaces no lo hacen). Guardar todas las cláusulas aumenta el poder deductivo pero en caso de no contar con equipos de cómputo con la suficiente memoria el desastre es inminente. Otra vez evidenciado, las características del instrumento hacen sentir su peso.

Para hacer de la paramodulación una regla efectiva, debe venir de la mano del axioma de la reflexividad (EQUAL(x,x)). Por un lado con la anexión del axioma caracterizamos a la igualdad (la simetría y la transitividad vienen implícitas) y además agregamos una cláusula potencial para generar el conflicto unitario, pues muchas veces el resultado de nuestras derivaciones nos conduce a una cláusula del tipo -EQUAL(t,t). Como información agregada, la paramodulación posee la completez refutacional (claro debemos trabajar con cláusulas donde se incluya el predicado de la igualdad) aunque bajo la estrategia del conjunto de soporte puede perder esta bella propiedad. Para conservarla al axioma de reflexividad debe tonificarse con la reflexividad funcional, i.e. debe incluir toda instancia del axioma elaborable con letras funcionales presentes en el conjunto de cláusulas iniciales EQUAL(f(x),f(x)), EQUAL(g(f(x),x),g(f(x),x)),...si g y f aparecen en alguna literal de nuestro conjunto de cláusulas. No entro en detalles, pues a la Wos, en honor a la efectividad no vale la pena incluir el axioma de reflexividad funcional ya que desencadena mucho trabajo computacional y a cambio da pocos resultados.

Con esta regla se hace patente de nuevo la obstrucción de la revisabilidad de las derivaciones para la mirada escrutiñadora de un ser humano. Aunque no es imposible verificarlas, distan de ser diáfanas las deducciones por paramodulación. Para que no me acusen de cometer vanas difamaciones, presento una prueba acarreando el ejemplo de la teoría de grupos traduciendo a cláusulas con el predicado de la igualdad:

Sea f(x,y) la función que represente la operación binaria del grupo f(x,y)=x*y
g(x) la función que represente el inverso $g(x)=x^{-1}$
“e” es la identidad del grupo

AXIOMAS

Identidad
1) EQUAL(f(e,x),x).
2) EQUAL(f(x,e),x).
Inverso
3) EQUAL (f(x,g(x)),e).
4) EQUAL (f(g(x),x),e).
Asociatividad
5) EQUAL(f(f(x,y),z),f(x,f(y,z)).

Reflexividad
6)EQUAL(x,x).

CONJUNTO DE SOPORTE

Hipótesis
7)EQUAL(f(x,x),e).

Conclusión negada
8)-EQUAL(f(a,b),f(b,a)).

DERIVACIONES

PARAM desde 5 hacia 7:
9) EQUAL(f(x,f(y,f(x,y))),e).
PARAM desde 7 hacia 5, con DEMOD por 1:
10) EQUAL(f(x,f(x,z)),z).
PARAM desde 9 hacia 10, con DEMOD por 2:
11) EQUAL(f(y,f(x,y)),x).
PARAM desde 11 hacia 10:
12) EQUAL(f(x,y),f(y,x)).

Conflicto unitario entre 12 y 8 !

Ejercicio para el lector, desenredar todas las paramodulaciones y el par de demodulaciones infiltradas. Debe encontrar el término en la cláusula "desde" unificado con uno de los argumentos del predicado EQUAL para después emprender la divertida tarea de hallar el MGU. De paso debe despachar a las dos demodulaciones colgadas de un par de paramodulaciones. Si tiene éxito se hará acreedor del título de verificador honorario, aunado con la ingrata satisfacción de haber superado un reto inútil, un desafío propio para computadoras, calculadoras, refrigeradores,...pero no para seres humanos. La respuesta al acertijo se localiza en la nota (57).

Ahora es el turno de una estrategia de dirección, la ponderación (*weightening*). Diseñada por Ross Overbeek en los setentas, la **ponderación** es la asignación de un valor de prioridad a las variables, letras predicativas, términos y letras funciones, en general a las cláusulas, para indicar un orden de preferencia al momento de aplicar las reglas de inferencia. Esta estrategia tal vez sea el epítome del enfoque heurístico, pues normalmente es el usuario quien reparte los pesos con base a su conocimiento o intuición, indicándole al programa en dónde debe enfocarse cuando haga las derivaciones. En el ejemplo del teorema de grupos, se le puede otorgar por medio del peso apropiado una mayor relevancia a la función del producto (f(x.y)) en comparación con la función del inverso (g(x)), pues de antemano sabemos o podemos suponer la nula o diezmada participación de g(x) en la demostración. Usualmente ante menor peso se tiene mayor prioridad. También se pueden atribuir pesos de manera automática, v.gr. peso (cláusula) = #signos (cláusula); no obstante para sacarle mayor provecho a la ponderación el papel del usuario resulta preponderante pues le infunde intuición al programa. La ponderación también tiene una cara restrictiva, activada cuando se define un PESOMÁXIMO, así toda cláusula con un peso mayor o igual a este se ignora en las inferencias. La completez refutacional se pierde con la ponderación, pues la obesidad puede discriminar a las cláusulas requeridas para derivar a la contradicción.

La ponderación ha tenido una fructuosa vida desde su nacimiento, en la demostración estelar del capítulo –dicho sea de paso es la estrella de toda el área-se recurre a una estrategia que la engloba. La susodicha es la estrategia de la **proporción**(*ratio*), en la cual se define una proporción m×n, la cual le ordena al programa seguir m iteraciones la estrategia de peso y n iteraciones una estrategia de fila (FIFS), i.e se escogen a las cláusulas respetando al orden de su generación (First In First Served). La estrategia de fila tiene la peculiaridad de instar a la búsqueda de pruebas más cortas, en términos precisos ordena una búsqueda por amplitud. El nivel de profundidad (η) de una cláusula –en el primer capítulo ya explicado- se define de la siguiente manera:

$\eta(c_0)=0$ donde c_o es una cláusula de entrada (axiomas, hipótesis, ...)
$\eta(c_n)= \eta(c_{n-1})+1$ para las cláusulas derivadas , donde c_{n-1} es el padre de c_n con mayor nivel de profundidad

La búsqueda por amplitud se concentra en todas las cláusulas de nivel 1, luego 2, y así nos vamos; de donde se aprecia la equivalencia entre la amplitud y la estrategia de fila. La estrategia de proporción hereda esta persecución de la cortedad, mientras más grande sea n mayor será el ímpetu por las pruebas cortas.

La ponderación puede ser aplicada tendiendo a fines afines; posibilitando el razonamiento por analogía en los programas. Sea Π_T una prueba del teorema τ, entonces con base a las cláusulas en Π_T podemos definir una plantilla de pesos para indicarle un rumbo a las inferencias cuando se quiera encontrar otra prueba de τ, o incluso de otro teorema similar a éste. Una consecuencia afortunada del razonamiento por analogía es la posible consecución de pruebas menos largas, objetivo apreciado por varias razones, en particular es un grato regalo para la *revisabilidad.* Con la misma tónica puede ser usado el procedimiento de la subsunción, otra aportación del gentil inglés Alan Robinson en la sexta década del siglo XX.

La **subsunción** es un procedimiento para desechar cláusulas con un contenido igual o menos general al retenido por una cláusula almacenada. La cláusula A subsume a la cláusula B si existe una instancia de A en donde algunas de sus literales están contenidas en B. Ejemplos:

EQUAL(f(x,0),0). subsume a EQUAL(f(1,0),1)
P(x). subsume a P(a)|-Q(b).

P(a,x) | P(y,b) subsume P(a,b). Este ejemplo exhibe un defecto de la subsunción cuando no se es precavido, ya que sin restricciones la subsunción puede desechar cláusulas unitarias, valiosas para alcanzar a la contradicción.

Si una nueva cláusula subsume a alguna de las retenidas, se conserva la primera y se elimina a la segunda; a esta variante se le conoce como subsunción hacia atrás. La subsunción hacia adelante procede al revés. La completez refutacional en general se conserva, pues en teoría si A subsume a B y B es empleada para deducir la contradicción

entonces A también puede jugar el papel de B. No obstante si aunada a la subsunción seguimos la estrategia del conjunto de soporte entonces la completez corre un serio peligro, porque una cláusula con soporte puede ser subsumida por una sin éste.

La demodulación junto con la subsunción son procedimientos esenciales para administrar las redundancias en las derivaciones, herramientas indispensables si se desea conservar todas las cláusulas útiles (táctica de los demostradores de Argonne). La subsunción también se emplea para construir pruebas más cortas, a través de la subsunción ancestral. La cláusula A **subsume propiamente** a la cláusula B sii A subsume a B pero B no subsume a A. La **longitud** de **derivación** de una cláusula es el número de aplicaciones de las reglas de inferencia para llegar a ella. Finalmente, la cláusula A **subsume ancestralmente** a la cláusula B sii A subsume propiamente a B ó A=B pero A tiene una menor longitud derivativa que B. Gracias a la subsunción ancestral, se cuenta con una poderosa herramienta para encontrar pruebas más digeribles para el intelecto humano. Retomaré este alivio en algunos cuantos párrafos.

Existen varias modificaciones de las reglas, estrategias y procedimientos expuestos, por supuesto los hay también nuevos, sin embargo el recetario de esta sección satisface las necesidades del ensayo. Contamos con el conocimiento básico para poder disecar a las pruebas manufacturadas y disertar sobre su naturaleza. Es más, ya se puede definir un algoritmo sencillo para producirlas.

ALGORITMO SECUENCIAL para PROBAR TEOREMAS

Al comienzo la lista usable (USA) contiene a los axiomas y la lista del conjunto de soporte (SOS) a las hipótesis más consecuencias nefastas de la negación del teorema proclives a provocar la contradicción. Así, el objetivo es demostrar la insatisfacibilidad de la segunda lista debido a la verdad de las cláusulas en la primera. Adicionalmente hay otra lista auxiliar (DEMO) donde se guardan las cláusulas demoduladoras seleccionadas por el usuario.

Mientras no se localice un conflicto unitario o una cláusula $\{\varnothing\}$:

1) Selecciónese una cláusula de SOS y llámese G (*focal clause*). Muévase G de SOS a USA
2) Aplíquese cada una de las reglas de inferencia seleccionadas por el usuario deduciendo a las cláusulas que tengan a G como padre en conjunción con cualquiera de las otras cláusulas en USA. Bautícese a este conjunto de claúsulas derivadas como NEW. Así, mientras NEW no esté vacío;
    a. Selecciónese una cláusula C_i de NEW.
    b. Simplifique C_i demodulando todos sus términos, así se produce una nueva cláusula C_{i+1} y se elimina C_i.
    c. Si C_{i+1} es subsumida por cualquier otra cláusula de USA o SOS, adiós a C_{i+1}.

d. Si C_{i+1} no es subsumida pero es demasiado pesada, hasta nunca C_{i+1}.
e. Si C_{i+1} superó a los filtros c,d; úsese para subsumir a las cláusulas en USA y en SOS (subsunción hacia atrás).
f. Si C_{i+1} es una cláusula unitaria con un predicado de igualdad, agréguese a DEMO y utilícese C_{i+1} para simplificar cuando sea posible a cada cláusula de SOS, DEMO y USA, i.e. para cada una de esas cláusulas (1) simplifíquesen los términos correspondientes en ella mediante C_{i+1} (2) bórrese de la lista donde estaba (3) añádese la cláusula simplificada a NEW.
g. Si C_{i+1} no ha sido borrado, añadir C_{i+1} a SOS.

Es un algoritmo básico, intuitivamente cumple con lo querido, busca la derivación de una contradicción y prueba por reducción al absurdo. No asegura terminar, pero ya sabíamos desde la sección pasada la inexistencia de un algoritmo con tal garantía; además debido a la indecidibilidad del CPPO, la pregunta usual sobre si el algoritmo es o no correcto sale sobrando, al menos con respecto al esqueleto pues las partes (reglas de inferencia, unificación, ...) si pueden ser objeto de tal cuestionamiento. Ahora bien, evaluar formalmente (CAP IV) la implementación de cualquier algoritmo para probar teoremas *á la Robinson* puede ser más una distracción y menos una necesidad, al final de cuentas dada la complejidad del problema sería ridículo sacrificar la eficiencia por una implementación correcta útil para demostrar únicamente teoremas de preescolar. Recordando, una argumentación similar fue dada para justificar los enrevesados programas en ensamblador para examinar la reducibilidad de las configuraciones en la demostración de 4CT. Sin embargo, en contraste con los programas de Koch y compañía, algunos de los programas demostradores de teoremas (al menos los de este capítulo) arrojan un producto deductivo completo apto para ser verificado e incluso no tan distante para ser *revisado*. El lema de la reducibilidad tenía el inconveniente de pertenecer a una demostración carente de una justificación deductiva causal, a saber, ¿por qué existe el conjunto de configuraciones inevitables reducibles? Pues aunque pudiera suponerse cada determinación de reducibilidad (C o D) como una secuencia de operaciones elementales y lógicamente válidas, operaciones que pueden ser verificadas por una máquina esta consideración -aún ignorando la violación a los derechos antropológicos de la *revisabilidad*- no disiparía el recelo sobre la enigmática existencia de tal conjunto, existencia impulsada por una vaharada probabilística y sostenida en el aire por las computadoras de vuelo. En cambio las pruebas resolutivas (o paramodulativas) son hijas de la lógica con un cuerpo deductivo entero; no exentas del error pero libres del estigma de la ambigüedad (ni por equivocación se asoma la noción de "experimentación"). Se puede dispensar la urgencia por elaborar programas demostradores cuasinfalibles y en su lugar enfocarnos en impulsar su eficacia, poder y elegancia derivativa; relegando la verificación a terceros –computadora no excluida-.

PROGRAMAS

Debido al paradigma de investigación de Argonne esta exposición se confinaría a la paraplejia de la incompletud sin un fugaz recuento de los programas desarrollados o emparentados con la investigación del razonamiento automatizado deductivo en el laboratorio de Illinois. Por otro lado la responsabilidad histórica condenaría dicha ausencia.

- P1 (Program 1). Elaborado por Carson con la cooperación de Wos en lenguaje ensamblador para la IBM 704, fue el primogénito de los demostradores de Argonne. En su caja de herramientas deductivas estaban las reglas de la resolución binaria y la factorización, además de la estrategia unitaria y el procedimiento de la subsunción hacia adelante. La estrategia del conjunto de soporte y la demodulación fueron desarrolladas durante la experimentación con este programa. Entre sus trofeos de pesca deductiva figuran el mentado teorema de la teoría grupos y el teorema de la teoría de anillos causa de la génesis de la demodulaciòn: (-x)(-y)=xy. A raíz de la experimentación con P1 Wos diseñó a la paramodulación desatando la creación de un nuevo programa.

- RW1 (Robinson-Wos 1). La inseminación dactilar de Robinson de sus ideas e innovaciones de Wos dieron a luz al caballo de batalla del laboratorio durante la segunda mitad de los 1960's. La regla de inferencia principal era la paramodulación, gracias a RW1 esta regla junto con la demodulación fueron sometidas a varias pruebas para someter a más pruebas. Los problemas al alcance de P1 con la debida traducción a predicados de igualdad entraban dentro del rango de RW1. Uno de los trofeos obtenidos por RW1 fue la prueba de la dependencia del cuarto y quinto axioma con respecto a los anteriores en la base axiomática de Grau de las álgebras booleanas ternarias[58]. Ante los frustrados intentos por demostrar el siguiente teorema de la teoría de grupos:
  para todo $x \in G$ se tiene $x^3=e$ entonces $[[x,y],y]=e$, donde $[x,y]$ es el conmutador de x,y ($[x,y]=x*y*x^{-1}*y^{-1}$)
  se hizo patente la necesidad de la demodulación hacia atrás.

- FTP (Functionless Theorem Prover). Programado por Ross Overbeek por los albores de los 1970's cuando todavía no pertenecía a Argonne (aunque si era vecino pues estaba en la Northern Illinois University), marcó un hito en la historia de los binarios esclavos demostradores –al menos los de Argonne- pues originó y obedeció al algoritmo secuencial anteriormente descrito –base de muchos de los algoritmos posteriores- y a su vez fue el primero en incluir la estrategia de la ponderación. Además, exhibía una eficiente implementación de la regla de la hiperresolución así como también incorporó a los procedimientos de la subsunción hacia adelante y hacia atrás. Aunque incapaz de manipular signos funcionales, fue lo suficientemente poderoso para probar al teorema sobre los conmutadores previamente mencionado y para demostrar que todo anillo Booleano es conmutativo.

- WOS1. Fruto de la apertura del canal de cooperación de Overbeek con Wos en 1971, heredó los dones de sus progenitores FTP y RW1, inaugurando la tradición en Argonne del diseño acumulativo de sus programas –todo lo bueno de los ancestros se transmite a sus vástagos-. Escrito en ensamblador, contaba con las reglas de la paramodulación y la hiperresolución, con las estrategias del conjunto de soporte y ponderación, con las subsumisiones, demodulaciones y su lenguaje manejaba símbolos funcionales.

- NIUTPx. La pandilla de Illinois se reforzó con Steve Winker y Ewing Lusk y juntos desarrollaron la caja de herramientas denominada NIUTPx (x indica la versión y llegó a valer siete). Poseía una serie de sistemas deductivos flexibles, pues permitía elegir variaciones del algoritmo demostrador de teoremas o ajustar las reglas de inferencia (debut de la UR-resolución) y estrategias entre otras concesiones. NIUTP con la asistencia de Winker y Wos (o al revés) adquirió la primera presea importante para los demostradores automatizados al resolver un problema abierto sobre álgebras booleanas ternarias, probando la independencia de los tres primeros axiomas de la base axiomática de Grau -ver nota (58)-. NIUTPx dominó la escena en Argonne y derredores durante los setentas.

    - AURA. Programa consanguíneo de NIUTP elaborado por los miembros de Argonne Bob Veroff y Brian Smith hacia los ochentas. Programa ambiental, i.e. actuaba como un conjunto de programas de razonamiento deductivo; el usuario en cada miembro del conjunto podía definir ciertos parámetros y opciones, además los elementos se comunicaban entre sí. Como anécdota, en una conversación de Wos con Smith de esta época se acuñó el nombre de "razonamiento automatizado deductivo".

- LMA (Logic Machine Architecture). William McCune dijo presente en el lustro temprano de los 1980's, en equipo con Lusk y Overbeek confeccionaron en Pascal (lenguaje de alto nivel) su biblioteca deductiva LMA. Al romper las cadenas del ensamblador con su hardware, la portabilidad de la LMA hacía posible incrementar su distribución entre otros grupos de investigación. La ingeniería de software alentó y enseñó la construcción planificada de un programa, haciendo de la programación un acto eficiente y ordenado. Así, la LMA consistía en tres capas distintas con funciones específicas: en la primera se encontraban los procesos elementales como la unificación, en la segunda se implementaron las reglas de inferencia, procedimientos y estrategias y en la tercera los algoritmos demostradores, algoritmos controlados interactivamente por el usuario.

    - ITP (Interactive Theorem Prover). Manifestación de la LMA, aumentaba el grado de interacción con el usuario. Los programas anteriores a la LMA estaban orientados a los archivos (el usuario especificaba todos los datos pertinentes en un archivo de entrada y esperaba pacientemente los

resultados) mientras que en ITP el control se especificaba mediante menús en cada ejecución del programa. ITP y LMA fueron los primeros programas de Argonne copiados y utilizados por otros investigadores a nivel mundial.

- OTTER (Organized Theorem-proving Techniques for Effective Research). Rey de los programas demostradores, surgido de los golpeteos en el teclado de los dedos de McCune en 1987. Escrito en C, en busca de portabilidad y eficiencia, regresó al control orientado a archivos. Comparte el espíritu ambientalista de Aura, en general, conserva y mejora muchas de las cualidades de sus antecesores: sus reglas de inferencia están basadas en la resolución y en la paramodulación; dispone de las estrategias de proporción, conjunto de soporte, ponderación,...; cuenta con variaciones optimizadas de los procedimientos para administrar las redundancias. Ningún otro programa hasta el momento –mediados del 2005- ha conseguido domar a tantos problemas abiertos; destaca un momento cumbre: la labor en equipo a mediados de los noventa del matemático hindú R. Padmanabhan con el computólogo McCune para probar con ayuda de Otter una cantidad de nuevos teoremas mayor a todos los problemas abiertos solucionados por todos los programas del área[59] acumulados hasta la fecha del ménage à trois. Ningún otro programa de razonamiento deductivo automatizado ha tenido una distribución tan extensa.

    - ROO. Versión en paralelo de Otter diseñada por Lusk y J. Slaney con la participación de McCune en los inicios de los noventas. Aunque ofrece un tentador aumento en la velocidad y capacidad de almacenamiento, también regala un serio obstáculo para la recopilación de datos experimentales pues una misma entrada en varias ejecuciones debido al paralelismo produce dispares salidas. Desgraciadamente, ante la ausencia de resultados gordos la balanza se sigue inclinando hacia OTTER .

    - EQP. Hijo de Otter procreado también por McCune, enfocado a las cláusulas donde reinan los predicados de igualdad, ergo ofrece más opciones para la paramodulación e implementa un tipo de unificación en donde la conmutatividad y la asociatividad son inherentes a ella, la AC-unificación.

? PROVER9. Estrenado en julio 7 del 2005, diseñado y designado por McCune como el sucesor de Otter.

Ya es tiempo de ver a los productos con más detenimiento, soluciones a problemas donde los humanos sin la computadora demostraban nada más abatimiento.

SOLUCIONES

El cálculo de equivalencias (*Equivalential Calculus*) es el campo de la lógica responsable del estudio de la noción de equivalencia. Las fórmulas de EC son el conjunto de expresiones construidas a partir de la función distinguida con dos argumentos e(x,y), la "e" obviamente proviene de la primera letra de "equivalente" donde x,y,z... son variables proposicionales. La regla de inferencia del EC es la separación condensada[60] (*condensed detachment*): sean e(A.B) (premisa mayor) y C fórmulas de EC, entonces se deriva D al aplicarle a B el MGU que unifique a C con A . Ejemplo:

```
                A              B
        e(  e(x,     e(y,z)),   e(x,z)  )

C           e(e(u,w),e(u,u))
                                        MGU y:=u , z:=u . x:=e(u,w)
D       e(e(u,w),u)
```

La separación condensada en el lenguaje clausular se representa así:

-P (e(x,y)) | -P (x) | P(y).
donde P denota "deducible"

La axiomatización de EC más intuitiva debe representar la caracterización estándar de una relación de equivalencia:

P(e(x,x)).	reflexividad
P(e(e(x,y),e(y,x))).	simetría
P(e (e(x,y), e(e(y,z),e(x,z)))).	transitividad

Las fórmulas en donde cada una de sus variables aparece un número par de veces son los teoremas del EC. El trío previo de fórmulas forma una base axiomática de EC, es decir, toda fórmula con un número par de cada una de sus variables puede ser generada con los axiomas y la aplicación de la separación condensada. Por cierto, dada la representación clausular de la separación condensada aunada a la presentación unitaria de las fórmulas de EC la regla de inferencia a primera vista óptima es la hiperresolución, de hecho es la empleada en la consecución binaria (hombre-máquina) de los resultados de este apartado.

En 1933 Lukasiewicz descubrió las primeras tres fórmulas de once signos (sin contar los de puntuación) con la peculiaridad de ser una base axiomática de EC cada una por separado, además demostró la ausencia de fórmulas con tal peculiaridad de menos símbolos[61]. A partir de entonces arrancó la cruzada por hallar a las santas fórmulas de once símbolos provistas del don de ser una base de EC. Despues de Lukas, Meredith encontró otras siete. Sin embargo a partir de las investigaciones de John Kalman y su estudiante Peterson en los

setentas con la ayuda de un asistente computarizado la búsqueda exhaustiva comenzó. Existen 630 fórmulas distintas de EC con 11 símbolos, Kalman agregó una más al costal base y Peterson demostró la imposibilidad de 612 para meterse al saco[62]. Las siete fórmulas restantes quedaron como preguntas abiertas. Kalman disponía de su propio esclavo demostrador de teoremas, pero a sabiendas de que podría tener éxito en la tierra santa de la equivalencia si se nutría con los adelantos del razonamiento deductivo automatizado llegó a Argonne en 1978, dando origen a una fecunda relación para ambas partes, fruto de ella fue el sometimiento de las siete fórmulas ocurrido desde los finales de los 1970's y culminado en el 2002 (fue Kalman quien les presento a los de Argonne a las siete bellas señoritas). Miremos ahora el rostro clausular de las fórmulas mencionadas, cubierto por el velo de sus nombres convencionales diseñados por Kalman y acompañadas por su suegro:

Lukas:

P(e(e(x,y),e(e(z,y),e(x,z)))). YQL
P(e(e(x,y),e(e(x,z),e(z,y)))). YQF
P(e(e(x,y),e(e(z,x),e(y,z)))). YQJ

Meredith:

P(e(e(e(x,y),z),e(y,e(z,x)))). UM
P(e(x,e(e(y,e(x,z)),e(z,y)))). XGF
P(e(e(x,e(y,z)),e(z,e(x,y)))). WN
P(e(e(x,y),e(z,e(e(y,z),x)))). YRM
P(e(e(x,y),e(z,e(e(z,y),x)))). YRO
P(e(e(e(x,e(y,z)),z),e(y,x))). PYO
P(e(e(e(x,e(y,z)),y),e(z,x))). PYM

Kalman:

P(e(x,e(e(y,e(z,x)),e(z,y)))). XGK

Los siete pilares de la duda:

P(e(x,e(y,e(e(e(z,y),x),z)))). XJL
P(e(x,e(y,e(e(x,e(z,y)),z)))). XKE
P(e(x,e(e(e(e(y,z),x),z),y))). XAK
P(e(e(e(e(x,e(y,z)),z),y),x)). BXO
P(e(x,e(e(y,z),e(e(x,z),y)))). XHK
P(e(x,e(e(y,z),e(e(z,x),y)))), XHN
P(e(x,e(e(e(x,y),e(z,y)),z))). XCB

Las siete columnas fueron demolidas por los programas y miembros de Argonne. Las primeras cuatro resultaron ser muy débiles para sostener al EC. La metodología para demostrar su pusilanimidad en su mayoría es obra de Wos. Al inspeccionar las derivaciones arrojadas por la computadora a partir de XJL Larry encontró un patrón en ellas, a saber todas contenían al menos una copia de su progenitor en sus entrañas, por

ejemplo las siguientes dos fórmulas se obtienen por separación condensada a partir de e(x,e(y,e(e(e(z,y),x),z))) :

e(v,e(e(e(w,v), **e(x,e(y,e(e(e(z, y),x),z)))**) ,w))
e(e(e(v, **e(x,e(y,e(e(e(z,y),x),z)))**) , *e(x*$_0$*,e(y*$_0$*,e(e(e(e(z*$_0$*,y*$_0$*),x*$_0$*),z*$_0$*)))*), v)

Si K=XJL, entonces las dos fórmulas anteriores se rescribirían
e(y,e(e(e(z,y),K),z))
e(e(e(z,K),K),z)

Si toda fórmula derivada desde XJL contiene a K, entonces la reflexividad no puede ser demostrada con la base axiomática propuesta pues e(x,x) no contiene a K. Con el apoyo de AURA , Wos y compañía esquematizaron a las derivaciones con base XJL y mediante una análisis por casos se demostró la hipótesis de la implicación previa. Un esquema típico de las deducciones a partir de XJL es:

f(A)= e(y,e(e(e(z,y),A),z)) para las fórmulas A

La inspección de los casos fue en su totalidad llevada a cabo por AURA: El trabajo en conjunto brindó una caracterización de los teoremas deducibles de XJL, y e(x,x) no encajaba en ella. Una prueba por casos montada a partir de la información de la interacción de los usuarios con AURA; una prueba por casos ejecutada por bichos electrónicos e interpretada por animales erguidos, ¿un deja vu del capítulo anterior? Al menos en la metodología ligeramente se parecen, sin embargo la comparación se establece con mayor acuidad al exhibir alguno de los productos terminados producidos por los programas de razonamiento deductivo automatizado, aquí se presentarán dos -uno de ellos en EC-.
Del mismo modo cayeron XKE, XAK y BXO, su demolición ocurrió en el primer lustro de los 1980[61]. Tal vez la estereotipada imagen del matemático de pizarrón ya deba incluir en la escena a una computadora. El instrumento binario es un asistente tenaz en la elaboración de demostraciones; el coche no hace al piloto empero.

Winker fue el primer artífice en el fortalecimiento deductivo de XHK y XHN como base axiomática de EC. La grúa de AURA con el operador Winker erigió una prueba de 84 pasos condensados con fórmulas de hasta 71 –sin contar comas ni paréntesis- símbolos para XHK y una prueba de 159 pasos separatistas con fórmulas de hasta 103 símbolos –sin contar comas ni paréntesis- para XHN después de varias corridas a lo largo de semanas. La prueba de XHK terminó con la derivación de YRO, la de XHN con UM, bases unitarias de EC demostradas por Meredith. Winker fue quien atisbó esta ruta, el programa lo condujo hacia ella en los párvulos ochentas. Para corresponder a la definición de prueba se agregan las consecuencias de la negación de YRO y UM a los correspondientes archivos de entrada para determinar el conflicto unitario. Lamento desilusionar a quien haya imaginado el proceso de la demostración automatizada como una línea de producción industrial, en donde después de preparar la materia prima e indicarle las instrucciones a la máquina –el archivo de entrada con axiomas, sos, estrategias, ...-uno oprime el botón y espera

pacientemente a la salida del producto terminado. La maquila puede ser gradual, puede modificarse con la retroalimentación de los resultados hasta el momento conseguidos, v.gr. se pueden incorporar lemas obtenidos a la lista de cláusulas usables, de donde se siga la dificultad de exhibir un integro archivo de entrada desencadenador de la prueba. Sin embargo, una vez obtenida una prueba se puede depurar –de hecho es una práctica común de las pruebas procedentes de Argonne- hasta reducir su longitud o conseguir con un archivo de entrada la generación de una prueba del teorema bajo lupa en una sola ejecución. Las rechonchas pruebas de Winker en los albores de los 1980's fueron las más grandes hasta entonces encontradas por los programas de razonamiento deductivo automatizado. En palabras de Wos:

Tal complejidad podría haber estado más allá de lo que cualquier investigador sin ayuda encontraría agradable o, quizás, posible. Quizás una década o más después de los triunfos de Winker, mi investigación (en que la NUTRIA tuvo un papel vital) en el área culminó con el hallazgo de una prueba de 23-pasos que establece XHK como una base axiomática y una prueba del 19-pasos para XHN. Así, de nuevo (tanto en los tempranos 1980s como en los tempranos 1990s) el valor y poder de un programa del razonamiento automatizado fue demostrado.[63]

En 1983 fueron recopiladas y publicadas[64] las respuestas a las primeras seis interrogantes planteadas por Kalman-Peterson. XCB se resistió hasta el 2002. Arduo fue el camino, no voy a describir todas las dificultades y logros parciales para desembocar finalmente en la prueba a continuación transcrita, prueba destilada de las pruebas antecesoras más largas. El proceso tiene un valor histórico, pero al ser todavía relativamente fresco, también es una refrescante fuente para los investigadores del área. Semejante a las pruebas de Winker, XCB se estableció como axioma unitario pues a partir de él se puede generar otra base axiomática conocida, a saber la constituida por la triple alianza de la reflexividad, asociatividad y simetría. Esta búsqueda fue motivada por el éxito de la derivación en Otter de la reflexividad a partir de XCB[65] en el 2001. Sin embargo Wos y compañía simplificaron la búsqueda al demostrar la dependencia de la reflexividad de los otros dos axiomas, contando con la asistencia de la Nutria[66]. Todas las estrategias, alicientes, ideas, para llegar a la prueba de la simetría y la transitividad de 25 pasos con nivel 19 están documentados en la referencia (67), aquí la aviento desnuda de esa exégesis:

A 25-Step Proof from *XCB*

```
The command was "otter". The processID is 24362.

-----> EMPTY CLAUSE at   0.15 sec ----->
   134 [hyper,52,132,128] $ANSWER(all_s_t_indep).

Length of proof is 25.  Level of proof is 19.

---------------- PROOF ----------------
```

```
51 [] -P(e(x,y)) | -P(x) | P(y).
52 [] -P(e(e(a,b),e(b,a))) | -P(e(e(a,b),e(e(b,c),e(a,c)))) |
   ANSWER(all_e_t_indep).
53 [] P(e(x,e(e(e(x,y),e(z,y)),z))).
105 [hyper,51,53,53] P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),
   u),e(v,u)),v)).
106 [hyper,51,105,53] P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),
   u),v),e(u,v))).
107 [hyper,51,53,106] P(e(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),
   z)),u),v),e(u,v)),w),e(v6,w)),v6)).
108 [hyper,51,106,53] P(e(x,e(e(e(e(e(y,e(e(e(y,z),e(u,z)),
   u)),x),v),e(w,v)),w))).
109 [hyper,51,107,53] P(e(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),
   z)),u),v),e(u,v)),w),v6),e(w,v6))).
110 [hyper,51,106,108] P(e(x,e(e(e(e(e(y,e(e(e(y,z),e(u,z)),
   u)),e(e(v,e(e(e(v,w),e(v6,w)),v6)),x)),v7),e(v8,v7)),v8))).
111 [hyper,51,53,108] P(e(e(e(e(x,e(e(e(e(e(y,e(e(e(y,z),
   e(u,z)),u)),x),v),e(w,v)),w)),v6),e(v7,v6)),v7)).
112 [hyper,51,105,110] P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),
   e(e(u,e(e(e(u,v),e(w,v)),w)),e(e(v6,e(e(e(v6,v7),
   e(v8,v7)),v8)),v9))),v10),e(v9,v10))).
113 [hyper,51,109,111] P(e(e(x,e(y,e(e(e(e(e(z,e(e(e(z,u),
   e(v,u)),v)),e(e(w,e(e(e(w,v6),e(v7,v6)),v7)),y)),v8),
   e(v9,v8)),v9))),x)).
114 [hyper,51,107,112] P(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),
   z)),e(u,e(e(e(u,v),e(w,v)),w))),v6),v7),e(v6,v7))).
115 [hyper,51,53,113] P(e(e(e(e(e(x,e(y,e(e(e(e(e(z,
   e(e(e(z,u),e(v,u)),v)),e(e(w,e(e(e(w,v6),e(v7,v6)),
   v7)),y)),v8),e(v9,v8)),v9))),x),v10),e(v11,v10)),v11)).
116 [hyper,51,114,106] P(e(x,e(e(y,e(e(e(e(y,z),e(u,z)),
   u)),x))).
117 [hyper,51,53,116] P(e(e(e(e(x,e(e(y,e(e(e(e(y,z),
   e(u,z)),u)),x)),v),e(w,v)),w)).
118 [hyper,51,112,117] P(e(e(e(e(e(x,
   e(e(y,e(e(e(e(y,z),e(u,z)),u)),x)),v),w),e(v,w)),
   e(v6,e(e(e(e(v6,v7),e(v8,v7)),v8))))).
119 [hyper,51,112,118] P(e(e(e(e(e(e(x,e(e(y,e(e(e(e(y,z),
   e(u,z)),u)),x)),e(v,e(e(e(v,w),e(v6,w)),v6))),v7),
   v8),e(v7,v8)),e(v9,e(e(e(v9,v10),e(v11,v10)),v11)))).
120 [hyper,51,115,119] P(e(e(e(x,e(y,e(e(e(e(y,z),
   e(u,z)),u)))),v),e(x,v))).
122 [hyper,51,120,105] P(e(e(e(x,e(e(e(e(x,y),e(z,y)),
```

```
   x)),e(e(e(u,v),e(w,v)),w)),u)).
123 [hyper,51,106,122] P(e(e(e(e(x,y),e(z,y)),z),x)).
124 [hyper,51,53,123] P(e(e(e(e(e(e(e(x,y),e(z,y)),
   z),x),u),e(v,u)),v)).
125 [hyper,51,124,123] P(e(e(e(x,y),x),y)).
127 [hyper,51,124,108] P(e(e(e(e(x,e(e(e(x,y),
   e(z,y)),z)),e(e(e(e(e(u,v),e(w,v)),w),u),
   v6)),v7),e(v6,v7))).
128 [hyper,51,127,123] P(e(e(x,y),e(e(y,z),e(x,z)))).
130 [hyper,51,128,125] P(e(e(x,y),e(e(e(z,x),z),y))).
131 [hyper,51,128,130] P(e(e(e(e(e(x,y),x),z),u),
   e(e(y,z),u))).
132 [hyper,51,131,123] P(e(e(x,y),e(y,x))).
```

En la línea 51 está tipificada la separación condensada, en la 52 la negación de la simetría y la transitividad, en la 53 la fórmula XCB. Luego "`[hyper,51,n,m]" indica la aplicación de la regla de la hiperresolución con núcleo 51 y los satélites n,m, es decir la separación condensada con las fórmulas n y m. En la línea 128 está derivada la asociatividad, y en la 132 la conmutatividad. Si observamos detenidamente, la cláusula 52 nunca se usa en las inferencias, sólo el axioma XCB y sus derivaciones. Para lograr la inactividad de una cláusula en las inferencias pero su intromisión en la búsqueda de conflictos unitarios existe otra lista además de la usable y la del conjunto de soporte, llamada la lista pasiva y a ésta pertenece 52. La lectura para desentrañar a las hiperresoluciones no es imposible, sin embargo es lo suficientemente tediosa para espantar los ánimos de quien esté interesado en hacerlo. Bajo esta perspectiva desalentadora, emergen las computadoras como el tonto asistente con la disposición y la habilidad para verificar esta cadena de caracteres y confirmar la validez de todos los pasos. No parece pertinente atacar el estatuto de "demostración" a este engendro, ni en general a los monstruos liberados por los programas de razonamiento automatizado deductivo en donde se exhiban todos los pasos derivativos de la prueba normativizados por una sintaxis y un cálculo; incluso las abominables pruebas de Winker están protegidas por el manto de la lógica y años de estudio de los sistemas formales; pueden ser demostraciones equivocadas pero no dejan de tener el mote de ser una demostración. Sin embargo, ¿qué conocimiento se puede exhumar de esa procesión aplastante de signos? Una posibilidad factible es revisar todo el proceso de la producción de la prueba, si existe claro, paso no común para quienes no están acostumbrados a vagar por la historia e husmear por los datos anexos de las demostraciones. Además, la información ahí encontrada aunque útil probablemente sea incompleta y no explique tanto embrollo deductivo. Me detengo, estas disquisiciones son propias de la última sección. Veamos otro ejemplo, la estrellita de la prueba del teorema de Robbins y algunas de las alternativas seguidas para su desentierro epistemológico.

Un álgebra booleana es una estructura algebraica montada sobre un conjunto X con un par de operaciones binarias en X ("+", "*"), un par de elementos notables en X ("0" y "1") y una función de X a X ("n(x)"). Las operaciones son asociativas, conmutativas y cada una se distribuye sobre la otra. Existe un neutro para cada operación, "0" para "+" y "1" para "*". Para todo elemento x de X existe n(x) en X tal que x+n(x)=1 y x*n(x)=0. El álgebra booleana captura la esencia de las operaciones lógicas ("*"-conjunción, "+"-disyunción, "n()"→negación) y de las operaciones sobre conjuntos ("*"→intersección, "+"→unión, "n()"→complemento). Existen innumerables bases axiomáticas para las álgebras booleanas, en 1933 E.V. Huntington edificó la siguiente[68]:

```
x + y = y + x.                      [conmutatividad]
(x + y) + z = x + (y + z).          [asociatividad]
n(n(x) + y) + n(n(x) + n(y)) = x.   [ecuación de Huntington]
```

A partir de la base de Huntington. se puede demostrar la existencia de "0","1" y definir a la otra operación binaria con sus correspondientes propiedades. Poco tiempo después, Herbert Robbins cambió la ecuación de Huntingon por una con menos signos y conjeturó la permanencia de la tripleta como base axiomática:

```
n(n(x + y) + n(x + n(y))) = x.      [ecuación de Robbins]
```

Ni Huntington ni Robbins ni Tarski ni alguno de los interesados pudo demostrar la conjetura de Robbins hasta antes de McCune-EQP, aunque si hubieron progresos parciales. Por lo tanto durante mucho tiempo la base de Robbins caracterizaba a las álgebras de Robbins. Se puede demostrar sin tanto problema la contención de las álgebras booleanas en las álgebras de Robbins; i.e. se puede derivar la ecuación de Robbins con los axiomas de Huntington. La contención en el otro sentido -¿toda álgebra de Robbins es un álgebra booleana?- fue por mucho tiempo una pregunta abierta. En 1979 el problema fue abordado por Winker, quien a sugerencia de Larry Wos desentrañó algunas condiciones adicionales para hacer del conjunto de fórmulas de Robbins una base axiomática del álgebra booleana. Winker encontró varias, y dicho sea de paso algunas de ellas fueron probadas mediante un programa de razonamiento deductivo automatizado, v.gr $\forall x(x+x=x)$. Winker localizó un par de condiciones atractivas por su simpleza y demostró que su anexión individual hace del conjunto de fórmulas de Robbins más la correspondiente fórmula agregada una base axiomática de las álgebras booleanas [69]:

```
existe C existe D, C+D=C          [primera condición de  Winker]
existe C existe D, n(C+D)=n(C)    [segunda condición de  Winker]
```

McCune proporcionó las siguientes indicaciones y datos de entrada a EQP:

```
% This input file is for EQP 0.9.
% Theorem.  Robbins algebras satisfy
%
%     exists C exists D, n(C+D)=n(C).
%
```

```
% This is the theorem that solves the Robbins problem.
% It was first proved by EQP.
% It takes about a week on an RS/6000 processor.

assoc_comm(+).
op(500, xfy, +).
assign(max_weight, 70).
assign(max_mem, 32000).
assign(report_given, 5000).
set(para_pairs).
clear(print_given).
set(basic_paramod).
assign(pick_given_ratio, 1).
assign(ac_superset_limit, 0).

end_of_commands.

list(sos).
n(n(n(y)+x)+n(x+y))=x.               % Robbins axiom
end_of_list.
list(passive).
x+y != x.         % denial of Winker condition 1
n(x+y) != n(x).   % denial of Winker condition 2
end_of_list.
```

Con estas especificaciones le llevó aproximadamente una semana a un equipo con un procesador RS/6000 usando alrededor de 30 megabytes de memoria encontrar la demostración del siguiente lema:

Toda álgebra de Robbins satisface a la segunda condición de Winker.

Con base al anterior lema la Conjetura de Robbins queda demostrada gracias al trabajo de Winker, aunque EQP por su cuenta siguiendo las instrucciones de McCune encontró las pruebas para las bases axiomáticas de las álgebras booleanas constituidas por el conjunto de axiomas de Robbins más las dos condiciones de Winker respectivamente, estas pruebas están incluidas en el reporte de la demostración de la Conjetura[70]. Antes de transcribir la prueba algunos puntos de las especificaciones de entrada y sobre los procesos efectuados por EQP (en particular la AC-unificación) deben ser aclarados.

- La **AC-unificación** incorpora a la asociatividad y a la conmutatividad de una operación binaria en la unificación de términos, ahorrando así la acción de explicitar a estas propiedades. Dos términos son AC-idénticos si mediante la reasociación y conmutación de sus subtérminos pueden convertirse en idénticos. Un AC-unificador es una substitución que hace AC-idénticos a dos términos. La AC-unificación es el proceso de hallar para dos términos sus AC-unificadores más generales (el equivalente a MGU). Sin embargo por la conmutatividad y la asociatividad pueden existir un gran número de AC-unificadores más generales (a diferencia de MGU) , por lo que se requieren heurísticas para controlar a esa multitud. En la prueba de la Conjetura de Robbins McCune recurrió a la estrategia **super-0**, prosaicamente

dicha estrategia se encarga de eliminar a los unificadores más complicados (mayor información en la referencia 70) y se le ordena a EQP en al archivo de entrada con la instrucción `assign(ac_superset_limit, 0).` Un AC-apareamiento es un caso especial de la AC-unificación en donde solo uno de los dos términos es instanciado, con el fin de simplificar a las ecuaciones derivadas mediante la demodulación.

- La estrategia de **proporción** de **apareamientos** (m×n) marca una dirección sobre las cláusulas a escoger para la aplicación de las inferencias. Sea el peso de un par de ecuaciones la suma del número de signos que contengan, y su edad la suma de sus correspondientes edades (la edad de una ecuación es determinada a partir de su posición en la secuencia de las ecuaciones retenidas). Así, m veces se elegirán a las ecuaciones más ligeras previamente no seleccionadas y n veces las más viejas no seleccionadas con anterioridad. Para la prueba del teorema la proporción es igual a uno: *`assign(pick_given_ratio, 1).`*

Así entonces, con el conjunto de soporte, la lista pasiva y demás estrategias especificadas el programa EQP consiguió la siguiente prueba del lema a través de la paramodulación con la depuración de los términos mediante su demodulación (AC-apareamiento):

```
----- EQP 0.9, June 1996 -----
The job began on eyas09.mcs.anl.gov, Wed Oct  2 12:25:37 1996
UNIT CONFLICT from 17666 and 2 at 678232.20 seconds.

---------------- PROOF ----------------

2 [] -(n(x + y) = n(x)).
3 [] n(n(n(x) + y) + n(x + y)) = y.
5 [para(3,3)] n(n(n(x + y) + n(x) + y) + y) = n(x + y).
6 [para(3,3)] n(n(n(n(x) + y) + x + y) + y) = n(n(x) + y).
24 [para(6,3)] n(n(n(n(x) + y) + x + y + y) + n(n(x) + y)) = y.
47 [para(24,3)] n(n(n(n(n(x)+y)+x+y+y)+ n(n(x) + y) + z) + n(y + z)) = z.
48 [para(24,3)] n(n(n(n(x)+y)+n(n(x)+y) + x + y + y) + y) = n(n(x) + y).
146 [para(48,3)] n(n(n(n(x)+y)+n(n(x)+y)+x+y + y + y) + n(n(x) + y)) = y.
250 [para(47,3)] n(n(n(n(n(x)+y)+x+y+y)+n(n(x)+y)+n(y+z)+z)+z)= n(y + z).
996 [para(250,3)]
     n(n(n(n(n(x)+y)+x+y+y)+n(n(x)+y)+n(y+z)+z)+z+u)+n(n(y+z)+u)) = u.

16379 [para(5,996),demod([3])] n(n(n(n(x)+x)+x+x+x) + x) = n(n(x) + x).
16387 [para(16379,3)] n(n(n(n(n(x)+x)+x+x+x)+x+y)+n(n(n(x)+ x) + y)) = y.
16388 [para(16379,3)] n(n(n(n(x)+x)+x + x + x + x) + n(n(x) + x)) = x.
16393 [para(16388,3)] n(n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)+x) = n(n(x) + x).
16426 [para(16393,3)]
     n(n(n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)+x + y)+n(n(n(x) + x) + y)) = y.
17547 [para(146,16387)]
     n(n(n(n(n(x)+x)+n(n(x)+x)+x+x+x+x)+n(n(n(x)+x)+x+x+x)+x)+x) =
                          n(n(n(x) + x) + n(n(x) + x) + x + x + x + x).
17666 [para(24,16426),demod([17547])]
     n(n(n(x)+x)+n(n(x)+x)+ x + x + x + x) = n(n(n(x) + x) + x + x + x).

------------ end of proof -------------
```

Donde “para(m,n)” significa paramodulación desde m hacia n y “demod(n)” denota demodulación con n (AC-apareamiento). Así, después de 15 inferencias se arriba a la ecuación cabalística 17666, la cual afirma la existencia de una c, a saber c=n(n(x)+x)+x+x+x, y de una d , d=n(n(x)+x)+x, tal que n(c+d)=n(c), es decir, causa un conflicto unitario con la negación de la segunda condición de Winker (ecuación 2, la 3 es la ecuación de Robbins) para así demostrar el lema.

Si era difícil dilucidar las inferencias con la unificación simple, ahora con la AC-unificación la tarea se turbia y más pesada se hace. ¿Cómo convalidar a la prueba? De las variopintas reacciones y reflexiones provocadas por la demostración computarizada de Haken y compañía se pueden aprehender algunas moralejas y asi lo hizo McCune:

Las pruebas encontradas por los programas siempre son cuestionables. Nuestro acercamiento [de Argonne] a este problema es hacer que el demostrador de teoremas construya un objeto de prueba detallado y tener un programa muy simple (escrito en un lenguaje de alto nivel) que verifique que el objeto de prueba es correcto. El programa de verificación es lo bastante simple para ser escrutado por los humanos, y la verificación formal es probablemente factible.
EQP no puede todavía construir los objetos de prueba, así la prueba de EQP fue usada para guiar a Otter (usando los axiomas de asociatividad y conmutatividad en lugar de la AC-unificación) a una prueba del mismo teorema. Otter produjo un objeto de prueba que se validó entonces por el programa verificador[71].

La prueba de Robbins además de su relevancia matemática y su fulgor histórico –también mediático pues hasta la difundió el NYTimes en un artículo aparecido el 10 de diciembre de 1996- ofrece una veta para la filosofía abocada a la noción de demostración. Desde un punto de vista meramente lógico la interrogante sobre la clasificación de los productos obtenidos por los programas aquí descritos ni siquiera está sitiada por el par de signos de la duda; son pruebas en sistemas formales, aunque no infalibles tampoco son demostraciones mutantes. Sin embargo, todavía queda pendiente establecer los mecanismos de su validación, el actor recomendado para hacerlo no sólo por McCune sino por las características de estos objetos deductivos es otra vez el instrumento que los ensambló. Por cierto, la prueba generada por Otter tiene una longitud igual a 34 y es de nivel 17; mientras que el objeto de prueba tiene una longitud igual a 171 pues se especifican con un poco más de detalle algunas de las sustituciones realizadas. Además, en el objeto de prueba se utiliza una notación prefija para facilitar la “lectura” para el programa verificador. Las siguientes líneas fueron copiadas del susodicho objeto de prueba [72]:

```
(1 (input) ((= (+ v0 v1) (+ v1 v0))))
(2 (input) ((= (+ (+ v0 v1) v2) (+ v0 (+ v1 v2)))))
(3 (input) ((= (+ (+ v0 v1) v2) (+ v0 (+ v1 v2)))))
(4 (input) ((= (n (+ (n (+ (n v0) v1)) (n (+ v0 v1)))) v1)))
(5 (input) ((not (= (n (+ v0 v1)) (n v0)))))
(6 (instantiate 1 ((v0 . v64)(v1 . v65))) ((= (+ v64 v65) (+ v65 v64))))
(7 (instantiate 3 ((v0 . v64)(v1 . v65)(v2 . v66))) ((= (+ (+ v64 v65)
v66) (+ v64 (+ v65 v66)))))
```

```
(8 (paramod 6 (1 1) 7 (1 1 1)) ((= (+ (+ v65 v64) v66) (+ v64 (+ v65
v66)))))
```

La línea 1 corresponde a la conmutatividad, las 2,3 a la asociatividad, la 4 a la ecuación de Robbins y la 5 a la negación de la condición de Winker. En la 6 se indica las sustituciones v0 por v64 (v0 . v64) y v1 por v65 (v1 . v65) en la cláusula 1 (instantiate 1); en la 8 la paramodulación desde 6 hasta 7 (paramod 6 (1 1) 7 (1 1 1)) en donde el término del primer argumento de la primera literal de igualdad (1 1) de 6 se une con el término en el primer argumento del primer argumento de la primera literal de igualdad (1 1 1) de 7, derivando en notación normal a la ecuación (v65+64)+v66=v64+(v65+v66) , i.e. se aplica la conmutatividad v64+v65=v65+v64 de 6 al primer sumando de lado izquierdo de la igualdad 7. Así, aunque están mejor delineadas los pasos deductivos en el objeto de prueba tampoco se explicitan a las unificaciones.

Por otro lado, rememorando la caracterización de Tymoczko de las demostraciones, las pruebas maquiladas por los programas estilo Otter ¿son *revisables*? ¿*inteligibles*? ¿*convincentes*? La prueba de McCune provocó una serie de trabajos cuyo objetivo fue facilitar su *revisabilidad*; evidenciando cierta dependencia de las demostraciones matemáticas en esta característica (o al menos en la *integibilidad*) aún en los objetos deductivos excéntricos, objetos aledaños a la perpleja región deductiva donde la existencia formal precede a la esencia epistemológica, al sitio donde las demostraciones son por nacimiento *computables* pero su *revisabilidad* debe desarrollarse para finalmente conseguirse (contrario a la mayoría de las demostraciones, en las cuales su formalización si llega, llega hasta el final).

ABSOLUCIÓN

Dos son los principales frentes de batalla para la defensa y ataque de las pruebas computarizadas completas, nombre otorgado al tipo de demostraciones por computadora de este capítulo en donde el objeto producido por la máquina satisface la caracterización de una prueba en un sistema formal sin omitir ninguno de los pasos de la deducción, su validación y su *revisabilidad-inteligibilidad*. Los dos campos de guerra colindan, pueden recibir metralla del mismo calibre y parapetarse con barreras similares. Escojo el menos accidentado para comenzar, a sabiendas que los refuerzos vendrán del capítulo cuarto. ¿Cómo verificar a las pruebas computarizadas completas?

En contraste con las demostraciones computarizadas exhaustivas –tipo Haken y compañía del 4CT- la validación de las pruebas computarizadas completas es independiente a su *revisabilidad* o *inteligibilidad*, la verificación es un procedimiento mecánico libre de especulaciones epistemológicas. Después de haber firmado el pacto epistemológico con un sistema formal, al aceptar sus reglas sus pruebas son una mera secuencia de derivaciones estipuladas por ellas y la validación se reduce a corroborar el cumplimiento de las normas.

Lo irónico del asunto radica en la tergiversación aparente del sueño de Wang y muchos otros lógicos sobre la simpleza acarreada por los sistemas formales. En el 2003 el afortunado de Thiele rescató de las notas de Hilbert a su problema 24, no incluido en su famosa lista redactada en el despertar del siglo XX de los 23 forajidos que debieran ser los más perseguidos por los alguaciles matemáticos. El problema desconocido alude a la búsqueda de la simpleza de las demostraciones, en las hasta hace poco invisibles palabras del por mucho tiempo invencible Hilbert:

El problema 24 en mi ponencia en París iba a ser: criterio de simpleza, o demostrar que ciertas demostraciones están en su forma más simple. Desarrollar una teoría sobre la demostración en las matemáticas en general. Bajo un conjunto de condiciones dadas no puede haber más que una demostración simple.[73]

Hilbert en parte también pavimentó una posible avenida de la respuesta:

El rigor matemático que es esencial en el tratamiento un problema no requiere demostraciones complicadas; sólo requiere que el resultado se obtenga en un número finito de pasos lógicos de un número finito de hipótesis dispuestas por el propio problema; buscando el rigor podemos encontrar la simplicidad.[74]

Las pruebas son la forma más simple de las demostraciones, así lo señaló Wang entre otros desde la sombra proyectada por Hilbert y su teoría de la demostración. Los pasos lógicos de las pruebas en consecuencia deben ser simples y de fácil captación. Desgraciadamente como lo puntualizó Alan Robinson, los sistemas de prueba resolutivos e hijos ignoraron esta petición, sus reglas de inferencia realizan varios pasos sencillos en una sola aplicación. Sin embargo, esta desobediencia de los sistemas resolutivos es una falta menor, pues se pueden desdoblar –aunque no sea una tarea menos que tediosa- las reglas en sus componentes elementales, como lo hace en parte Otter al construir los objetos de prueba; se aumenta la longitud de la prueba pero se tiende hacia la simpleza. Además, ¿acaso se vuelve terriblemente complejo e inaceptable para la criba de la simplicidad que en lugar de efectuar una suma hagamos diez de un solo golpe?

Desglosar a las inferencias resolutivas se complica ante el aumento de la cantidad de signos en las fórmulas y la extensión de la prueba, así un trabajo simple puede convertirse en un castigo digno de la pluma de Dante. La pulsión de Wos y otros investigadores por pruebas mas cortas sigue el deseo de Hilbert hacia la austeridad, al satisfacerlo se acercan las pruebas al rango de nuestras capacidades mentales para su lectura, su validación-refutación y sobretodo su comprensión. Con pruebas no tan largas puede desenredarse el contenido epistemológico depositado (o insuflado) en ellas como posteriormente se expondrá. Aún así, la validación de las pruebas computarizadas completas no puede dejar a un lado la asistencia de programas y abandonarse al intelecto de los monos erguidos. ¿Por qué? Porque nos sacan una amplia ventaja en la ejecución de este tipo de tareas mecánicas, son más poderosas y tienen una menor propensión al error. Aferrarse a la antropoverificación es un prejuicio de animales en peligro de extinción. Atenerse a la

verificación por computadora reduce a cenizas binarias el cuento de hadas sobre la certeza absoluta en las matemáticas, pues la única certeza sobreviviente es aquella que niega la seguridad total en las operaciones efectuadas por las computadoras. Reconocer de antemano la falibilidad en las pruebas y sobretodo en las verificaciones computarizadas libera a las matemáticas de los rancios absolutismos, manumite a las demostraciones en general al destrozar los bretes de la razón divina, aquella condenada a errar sin errar por toda la eternidad. La historia (real) de por sí clamaba romper con los regímenes despóticos en las matemáticas, evidenciando las grietas en los procesos de validación de las demostraciones (como la "demostración" de Kempe de la 4CT); es más, la historia registra a las diversas fluctuaciones de la noción "demostración" acaecidas desde los griegos, aquí se relata una de ellas. Sin embargo los hechos históricos por sí solos tienen poco poder de convencimiento en una disciplina que presume ser ajena al ruido de los sucesos del mundo; más aun, las equivocaciones en las matemáticas pueden ser explicadas a partir de las mundanas interferencias de la realidad en el quehacer matemático, somos criaturas minúsculas incrustadas en estructuras sociales deficientes con un estólido asombro ante la plenitud de la verdad y la belleza infinita de la inmarcesible matemática; aunque nuestro intelecto sea torpe y nuestros métodos de validación precarios nuestras matemáticas son un reflejo de aquella matemática eterna, un reflejo sometido a una purificación gradual pero constante. Toda la anterior perorata idealista es un credo difícil de tolerar cuando nuestro único puente hacia esa escala superior del conocimiento en su garantía tenga en letras pequeñas aun la nimia pero existente posibilidad de equivocación, tema del capítulo cuarto. Esfuerzo ridículo tratar de solventar la separación de teorema y demostración, ser teorema equivale a contar con el salvoconducto de una demostración. Si las demostraciones son falibles, como está "demostrado" para las demostraciones computarizadas, entonces la contingencia en las matemáticas tiene un papel mayor en el relato de fantasía de la matemática ideal. Consecuencia inmediata del reconocimiento de la falibilidad de las demostraciones computarizadas –ya expuesta en el segundo capítulo- es la extensión de la falibilidad a las demás demostraciones, si nos resistimos a hacerlo, la única ridícula alternativa es coronar a la mente con nimbadas facultades suprahumanas. Lo que provenga del hombre, en duda estará. No obstante, el polo de las matemáticas entendidas meramente como una actividad social, tampoco se alza imponente e inquebrantable si confesamos nuestros límites intelectuales. Las críticas circulares del tipo "si todo esta cubierto por la vacilación social entonces esta afirmación también lo está" reiteradamente se lanzan como cohetes en contra de aquellos díscolos sociólogos-historiadores con pretensiones altas y argumentos pobres. Imposible negar la remota existencia del universo ideal en donde forniquen las líneas con los círculos perfectos y que con un poco de suerte nuestras matemáticas logran penetrarse en ese insensible coito. Imposible dejar de sospechar lo contrario, ante la recopilación de votos a favor de la contingencia, algunos de ellos solventados teóricamente como la falibilidad de las demostraciones computarizadas. Entonces ¿en dónde estamos situados? Parados en la indeterminación, aunque gracias a la sensata falta de atención o preocupación en esta parada las matemáticas siguen su marcha, y como lo dijo el gato de risa evanescente a la perpleja Alicia, siempre que te muevas eventualmente llegarás a alguna parte. Por otro lado, para los filósofos, sociólogos y demás bichos interesados, la indeterminación es un manantial a partir del cual fluyen numerosos

ensayos que desembocan en ninguna conclusión final; condición favorable para la filosofía, pues según me parece, en esta disciplina cuando una discusión se puede dar como terminada entonces no valió la pena empezarla.

El temor por la longitud abrumadora de las pruebas completas computarizadas aunque tenga una manifestación palpable (como las de Winker) raramente se materializa en los seguidores del paradigma de Argonne. Los programas adeptos a tal paradigma almacenan todas las cláusulas útiles, por lo cual el número y profundidad de las ramas generadas para pizcar a la demostración puede saturar a la memoria de la computadora o llegar al límite de tiempo. Por esta razón los procedimientos para administrar redundancias (demodulación, subsunción) tienen un rol preponderante en los programas estilo Otter, aunque también por este motivo la longitud y nivel de las pruebas producibles está limitado por las capacidades del instrumento y los lineamientos del algoritmo demostrativo. Sin embargo conforme se incremente la memoria y poder de procesamiento y se depuren las estrategias y procedimientos entonces podemos esperar pruebas más grandes. Por ejemplo, Otter en febrero del 2005 gracias a las indicaciones de Bob Veroff encontró una prueba para establecer una fórmula de 25 símbolos como base axiomática de las ortolatices modulares con una longitud igual a 356 y un nivel igual a 140; Otter tuvo éxito en la generación de esta gigantesca prueba porque se recurrió a la novedosa estrategia llamada "bosquejos de prueba" [75] (*proof sketches*). Por cierto, en el 2002 Veroff con la excavadora Otter halló una nueva prueba de la conjetura de Robbins de longitud 142 y nivel 70, en la cual se deduce a partir de la base de Robbins la ecuación de Huntington[76]. Así, paulatinamente las pruebas computarizadas completas pueden alejarse de los linderos del trato amable con respecto a nuestro intelecto. No obstante, aunque la pregunta suene entre vana y juguetona –sobretodo mal formulada-, ¿cuáles son los límites de nuestras lentes mentales empleadas en la lectura-validación de una demostración? Si las fronteras se trazaran por la longitud, la demostración computarizada de H&A&K&1&0 sobre el lema de la reducibilidad sería un buen ejemplo de una demostración con una longitud inasequible para los ojos humanos. Pero, ¿las demostraciones de Veroff o Winker acaso coquetean con la imposibilidad de su lectura? Previo a esta tribulación, vale la pena preguntarse si nos conviene dedicarle tanto tiempo a la inspección de ese torrente de caracteres que parece anegar a nuestro intelecto escupido por las fauces de la computadora desde sus entrañas electrónicas. Afortunadamente contamos con un lacayo con el talante para verificar a esas pruebas, para luego informarnos si éstas merecen ser atendidas por sus amos. Los programas verificadores no son un lujo, su utilidad está más que justificada. Por otro lado, aunque seamos ineptos en la lectura del lenguaje en el cual están escritas las pruebas computarizadas completas, contamos con otras artimañas de tintes epistemológicos para facilitarnos la tarea. Para bien o para mal, no somos máquinas de miras estrechas sin inventiva ni argucias creativas; ante las veredas repletas de opresiones o penurias podemos trazar atajos o inventar salidas. Por ejemplo ¿podremos traducir a las pruebas resolutivas e hijos a un lenguaje más comprensible para el ser humano? Suficiente con la verificación, es trabajo de las máquinas en primera instancia aunque paulatinamente gracias a la reducción del tamaño o a la propuesta traducción de la prueba pueda ser llevada a cabo por los hombres, no obstante, la obtención de una apariencia presentable no es un objetivo propio

de la verificación sino un meta para lograr la *revisabilidad.* Corresponde entonces recorrer el camino de la traducción.

En la página de internet (http://www.unix.mcs.anl.gov/~mccune/papers/robbins/) donde se encuentra la presentación de las pruebas de McCune-EQP-Otter de la conjetura de Robbins aparecen una serie de hipervínculos hacia documentos con títulos reveladores:

- *The Robbins Problem - Computer Proofs and Human Proofs*, por Louis H. Kauffman
- *A Mathematician's Translation of the EQP Proof* por Stan Burris
- *A Complete Proof of the Robbins Conjecture*, by Allen Mann
- *Using Mathematica to Understand the Computer Proof of the Robbins Conjecture* por Branden Fitelson

Después de hurgar en ellos se hace patente la urgencia por transformar al ristre de signos de la prueba computarizada a un lenguaje donde se desdoblen los nudos deductivos. El testimonio de Fitelson es representativo y educativo:

Aunque la prueba de EQP de la conjetura de Robbins (que se ha verificado tanto automáticamente como a mano por varios investigadores diferentes) firmemente establece la verdad de la conjetura de Robbins, dicha prueba hace poco para ayudar a los seres humanos a entender cómo demostrar a la conjetura de Robbins. De hecho, la prueba de EQP es bastante difícil de seguir (¡o incluso de sólo verificar la validez de las inferencias!). Después de varias frustrantes e infructuosas semanas de intentar desglosar a mano la prueba de EQP, empecé a experimentar con Mathematica.
¡Un par de días después, ya tenía una reconstrucción completa de la prueba en Mathematica![77]

La intromisión del instrumento no cesa ni siquiera en la labor de la traducción. Mathematica es un programa especializada en análisis numérico y cálculo simbólico con un lenguaje de programación propio. En particular permite la manipulación simbólica con una presentación visual normal en las matemáticas, así la base de Robbins se escribiría en la notación de Mathematica (pude haber escrito también en "la notación usual matemática") así:

$x \oplus y = y \oplus x$ [Commutativity of $\oplus$]

$(x \oplus y) \oplus z = x \oplus (y \oplus z)$ [Associativity of $\oplus$]

$\overline{\overline{x \oplus y} \oplus \overline{x \oplus \overline{y}}} = x$ [Robbins equation]

Además, en Mathematica se pueden definir funciones de gran utilidad para descifrar a la prueba de EQP, como la expansión de Robbins de un término x con respecto a cualquier otro término y:

RobbinsExpand[x,y] = $\overline{\overline{x \oplus y} \oplus \overline{x \oplus \overline{y}}}$

También se puede definir la simplificación de una expresión con la forma de Robbins:

RobbinsSimplify[$\overline{\overline{x \oplus y} \oplus \overline{x \oplus \bar{y}}}$]=x

Fitelson señala a la unidimensionalidad notacional del operador de la negación "n(x)" como el mayor obstáculo para la lectura de la prueba de EQP. En un sentido similar, Kauffman identifica al complicado abanico de paréntesis, resultado de esa notación unidimensional, como la cause detonante del intrincado patrón sintáctico. En consecuencia, ambos dos arrastrarán hacia el plano las operaciones estipuladas en la prueba, buscando con el incremento de la dimensión el decremento del ofuscamiento. Por ejemplo, en la notación de Mathematica el primer par de líneas de la prueba se escribiría:

1. $\overline{\overline{x \oplus y} \oplus \overline{y \oplus x}} == y$ [Assumption: the Robbins Equation]

2. $y \oplus \overline{y \oplus \bar{x} \oplus \overline{\overline{x} \oplus \bar{y}}} == \overline{x \oplus y}$ [1]

De alguna manera 2 se deriva de 1. La prueba de EQP poco indica, menciona una paramodulación de `3 con 3` (recordemos, la ecuación `2` es la negación de Winker2 y la `3` la ecuación de Robbins en la prueba EQP) pero no especifica ni los términos unificados ni el AC-MGU ni nada. Ante la ausencia de pistas emergen dos posibles alternativas, verificar el objeto de prueba creado por Otter o intentar decodificar 2 sabiendo que es un espejismo de 1. Para evitar hurgar entre tantas líneas concentrémonos en la prueba de la Nutria en donde también se especifican los términos unidos en las paramodulaciones (activando una bandera llamada *detailed_history*) y donde se brindan más pistas sobre las AC-unificaciones pues ante la discapacidad Otter de hacerlas directamente se agregan los axiomas y las inferencias paramodulativas a través de la conmutatividad y la asociatividad.
Por medio del axioma de la conmutatividad **2** se generan otras dos cláusulas (**71,93**) intermedias a la consecución de 2 (`5` en la de EQP, **132** en la de Otter):

```
2 [] x+y=y+x.                          %conmutatividad
5 [] n(n(n(x)+y)+n(x+y))=y.            %robbins
...
71 [para_into,5.1.1.1.2.1,2.1.1] n(n(n(x)+y)+n(y+x))=y.
...
93 [para_into,71.1.1.1,2.1.1] n(n(x+y)+n(n(y)+x))=x.
```

Así, paramodulando desde la conmutatividad **2** hacia la ecuación de Robbins **5** se rescribe ésta en su forma **71**, conseguida remplazando en **5** n(x+y) por n(y+x). Luego se paramodula desde **2** hacia **71** para conseguir **93** unificando n(n(x)+y) + n(y+x) con x+y, de donde en 93 conmuta los sumandos dentro de la negación –n(n(...+x))- del lado izquierdo de la ecuación **71** intercambiando en todo **71** las y con las x.

```
...
132 [para_into,93.1.1.1.2,5.1.1] n(n(n(x+y)+n(x)+y)+y)=n(x+y).
```

La ecuación **132** se genera por la paramodulación desde **5** hacia **93.**
Rescribimos en el lenguaje de Otter a las cláusulas progenitoras de **132** para desentrañar este paso, donde f(x,y) equivale a x+y:

Desde
5 EQUAL(n(f(n(f(n(x),y)) , n(f(x,y)))) , y).
Hacia
93 EQUAL(n(f(n(f(x,y)) , n(f(n(y),x)))) , x).

`93.1.1.1.2,5.1.1 señala la` unificación de
n(f(n(f(n(x),y)) , n(f(x,y))))
con n(f(n(y) , x))
de donde:
n(y)=n(f(n(x),y)) luego y= f(n(x),y), x= n(f(x,y))

traduciéndolo y=n(x)+y, x=n(x+y), luego en **93** como lo indica la paramodulación el sumando n(n(y)+x) se remplaza por y, así haciendo el resto de las sustituciones en 93 se obtiene **132**:

n(n(n(x+y)+n(x)+y) + y) = n(x+y)

De esta manera, se puede ir corroborando la secuencia de derivaciones de la prueba de EQP a través de la más "amigable" prueba de Otter, no obstante si seguimos esta ruta imitaríamos como enajenados el proceder de la máquina y para alimentar a la humillación además tendríamos una más alta probabilidad de equivocación. Obedecer esta táctica otorgaría la ratificación de la prueba, pero quedaría pendiente todavía la interpretación de la prueba. Por lo cual, la alternativa del desentierro epistemológico, seguida por Fitelson, parece más atractiva.

La pregunta a la cual ayuda a responder Mathematica se plantearía de la siguiente manera:

¿Qué término z en la expansión de Robbins de $\overline{x \oplus y}$,z genera a $\overline{y \oplus \overline{y \oplus \bar{x} \oplus \overline{x \oplus y}}}$ tomando en consideración a las transformaciones concedidas por la asociatividad, conmutatividad, ecuación de Robbins, etc. manejables por Mathematica ?
Fitelson con base a la táctica de la prueba y error habilitada por Mathematica dio con la respuesta:

```
In[7]:= RobbinsExpand[x⊕y, x̄⊕y]

Out[7]= x⊕y⊕y⊕x̄⊕y⊕x⊕x⊕y
```

Los dos sumandos principales de la salida pueden rescribirse como una ecuación de Robbins sabiendo las leyes de DeMorgan además de la conmutatividad, así:

$\overline{\overline{x \oplus y} \oplus \overline{y \oplus \bar{x}}} = y$, $\overline{y \oplus \bar{x} \oplus \overline{x \oplus y}} = \overline{\overline{x \oplus y} \oplus \overline{x \oplus \bar{y}}} = x$

Mathematica puede manejar las transformaciones necesarias para establecer la igualdad

```
In[9]:= ProofLine[2] == (RobbinsExpand[x⊕y, x̄⊕y] == x⊕ȳ) // RobbinsSimplify

Out[9]= True
```

de manera automática tras definir las funciones correspondientes; por lo cual la experimentación anuncia esperanzas de éxito, más aún, conforme se avanza en el desenredo de la prueba se pueden codificar nuevas funciones en Mathematica para acelerar al descifrado. Así lo hizo Fitelson en el documento citado, hasta el último paso donde se deriva:

$$\overline{x \oplus x \oplus x \oplus x \oplus \overline{x \oplus \bar{x}} \oplus \overline{x \oplus \bar{x}}} = \overline{x \oplus x \oplus x \oplus \overline{x \oplus \bar{x}}}.$$

si $C = x \oplus x \oplus x \oplus \overline{x \oplus \bar{x}}$ y $D = x \oplus \overline{x \oplus \bar{x}}$ podemos visualizar en la ecuación previa a la segunda condición de Winker.

Es un método con una enorme dependencia en el conocimiento - intuición del usuario, pero acaso, ¿no era esto lo que estábamos buscando? Aunque es una metodología aplicable a las demostraciones donde se involucren manipulaciones algebraicas cada prueba ofrecería un reto distinto. Se abre el telón para el problema de la interpretación cuando se escuchan los aplausos finales de la ratificación de una prueba. En palabras de Kauffman:

Las computadoras demostrarán teoremas con pruebas que son muy largas y enrevesadas. Estas pruebas necesitarán traducirse a un lenguaje que los seres humanos puedan entender. Estamos en el principio de un nuevo campo de las matemáticas, amasijo entre la lógica, la comunicación y la epistemología. Las máquinas nos impulsarán hacia nuevas y poderosas maneras de explorar el terreno matemático.[78]

Con esta preocupación entre manos, Kauffman moldeó una notación donde se percibieran y realizaran con mayor facilidad las operaciones booleanas. En lugar de la asistencia del siervo electrónico, intenta rescatar a la devaluada mente con una notación cuya presentación visual fomenta el reconocimiento de patrones vinculados con operaciones algebraicas. Como bien lo señala, para aprender a razonar en un lenguaje nuevo se requiere de práctica, así también pregona al lector a entrenarse con la notación de cajas. Expliquemos con un ejemplo, la ecuación de Robbins en la notación de cajas se escribiría así:

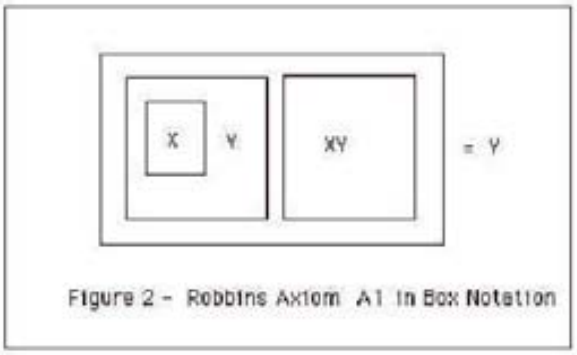
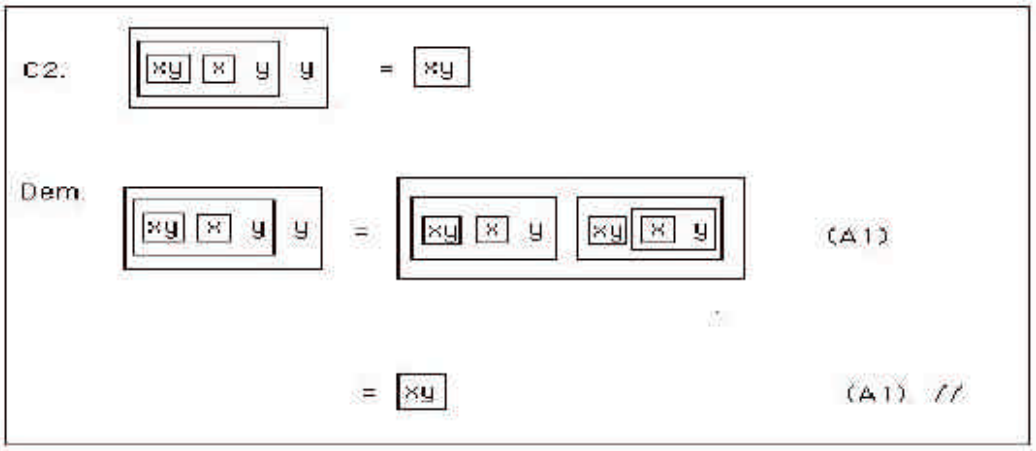


Figure 2 - Robbins Axiom A1 in Box Notation

Estar encerrado en una caja denota la negación de los términos de su interior. Además, ahora la operación binaria manejada es el producto "*" (intersección, conjunción), señalado por la vecindad de las términos. Por lo cual Kaufmann de antemano convierte e las operaciones con "+" de la prueba de EQP a operaciones con "*". Así, sucumbiendo ante la misma ansia de Fitelson por desnudar a las derivaciones, logra descargar su obsesión a través de la presentación en dos dimensiones de la notación de cajas. Por ejemplo, la primera deducción de la prueba la obtiene en el lenguaje de cajas así:



Experiencia en el manejo de esta notación es la clave para sacarle provecho y rellenar a esas cajas con conocimiento. Gracias a su pericia, Kauffman además de decodificar a la prueba logró disminuir en uno el número de pasos de la prueba de EQP[79]. La exigencia de Kauffman por una traducción de la prueba computarizada hacia un lenguaje donde se puede establecer la comunicación entre la demostración y su lector puede considerarse como una petición -casi obligación- de la *revisabilidad* en las demostraciones. Con lirismo K. se explaya:

EQP produjo una prueba que pudo haber sido inaccesible para los seres humanos pero de hecho se hizo comprensible para nosotros con un poco de esfuerzo en la comunicación... Yo entendí la prueba de EQP con un regocijo muy parecido al goce que me provoca una demostración elaborada por un ser humano. Entender una demostración es como escuchar

a una pieza de música - o como improvisar un pieza de música a partir de una partitura incompleta-.[80]

Dos acotaciones: si la prueba fuera inaccesible entonces ni siquiera podría realizarse la traducción, si sé nada de chino, estaría en chino que tradujera un libro en ese idioma al castellano. Sin embargo para conseguir una traducción eficaz –aquella que le transmita en la medida de lo posible al lector de diferente lengua el sentido del texto original- no basta con hablar el idioma remitente; se debe compenetrar en lo que dicen los enunciados para luego expresar ese entendimiento en el lenguaje destinatario y para lograr ese acoplamiento también se debe dominar este último lenguaje (v.gr.lenguaje de cajas). De este modo, la traducción puede ser más que un mero paso para lograr la comprensión y fungir como la comprensión en sí (v.gr. las traducciones de Fitelson y Kauffman). Si se alcanza esta meta el texto traducido es un bocado ya masticado epistemológicamente. La prueba de EQP no cierra los portones al entendimiento en su forma bruta pues no es un producto cuya deslumbrante novedad ciega; por formación los matemáticos disponen de mucha experiencia en la manip ulación algebraica pero no están(aban) acostumbrados a manejar ecuaciones tan rebuscadas, derivaciones simbólicas que con sarna desafían a nuestro intelecto. Para compensar esta debilidad se puede recurrir al apoyo de un instrumento con mayores (y distintas) capacidades –como Fitelson- o podemos representar el problema en un lenguaje –Kaufmann- más aprehensible para dar finalmente con la traducción anhelada. Segunda acotación: ni la prueba de EQP y tampoco las pruebas computarizadas aquí reseñadas son una partitura incompleta; son una partitura escrita para músicos con más manos, es música diseñada para otro tipo de audiencia. A lo largo de las primeras dos secciones del capítulo se han exhibido partituras inconclusas, bosquejos de demostración donde se invita al lector a culminarlas bajo su interpretación epistemológica; quien las lea –porque se pueden leer- podrá ejecutarlas si tiene el entrenamiento matemático y la preparación en el área; así como un músico se especializa en tocar música barroca o clásica. Las partituras de la prueba XCB o de la conjetura de Robbins son obras terminadas, sin embargo para poder ser interpretadas por/para los humanos requieren una trascripción, el trabajo de Kauffman y Fitelson constituyen un par de esos ejercicio s de composición. Antes de continuar por esta línea evito dejar cabos sueltos. Stan Burris optó por la sustitución en lugar del incremento de la dimensión; es decir, asociando los términos de la prueba de EQP a nuevas variables edita la redacción de ésta facilitando su lectura de comprensión[81]. Sin ánimos de menospreciar el trabajo de Burris esta idea ya había sido explorada por McCune en el artículo donde expone la sorprendente prueba de EQP[82]. Por otro lado, Allen Mann es quien muestra un trabajo más acabado y completo, su escrito es un tratado sobre álgebras booleanas redactado en una notación análoga a la de Mathematica donde la demostración de la Conjetura de Robbins es sólo una entre varias demostraciones relacionadas con el tema[83]. Su demostración del teorema de Robbins está organizada en subdemostraciones de lemas y teoremas auxiliares, de este modo exhibe una presentación estructurada arquetípica, respetando la trayectoria deductiva de la prueba de EQP.

Hasta ahora se han estudiado diferentes traducciones sobre una prueba específica, la pregunta general infame se asoma: ¿toda prueba computarizada completa puede ser

traducida efectivamente a un lenguaje para los *homo sapiens*? O equivalentemente: ¿toda prueba computarizada completa es *revisable*? Si recordamos en el capítulo anterior para huir del apuro se acuñó a la *inteligibilidad* con las cenizas de las pretensiones antropológicas y de la lectura total albergadas por la *revisabilidad*. Podemos entender la prueba del lema de la reducibilidad porque comprendemos a los algoritmos que determinan si una configuración es C-D-reducible, confiamos en la implementación de éstos porque conocemos las características y propiedades línguísticas y materiales del instrumento para sustentar su poca probabilidad de falla. En el fondo la demostración del lema de la reducibilidad se reduce a un número gigantesco de operaciones sencillas, ladrillos comprensibles al igual que el plano del algoritmo y el edificio del programa. En contraste, por medio de un número gigantesco de operaciones sencillas se ensambla una prueba computarizada completa; pistas vacías se nos revelan al conocer acerca de las reglas de inferencia, estrategias, algoritmos. etc. cuando perseguimos a la comprensión del producto terminado, aunque por otro lado, ese conocimiento sí nos permite verificar la validez de las pruebas para ratificar o rechazar a los teoremas –ganancia nada despreciable-. El rodeo epistemológico del lema de reducibilidad aquí es inoperante, se debe encarar directamente a la prueba computarizada para exprimir su contenido. Incauto abogar por panaceas después de tanto resultado negativo, la existencia de un método general para traducir efectivamente cualquier prueba computarizada completa constituiría un buen argumento para una novela de ciencia ficción sobre la inteligencia artificial en lugar de un argumento bien fundamentado a favor de la *revisabilidad* de este tipo de pruebas. Aunque compartan el lenguje clausular y las reglas de inferencia, las pruebas computarizadas completas versan sobre distintas áreas de las matemáticas, a menos de que la semántica sea prescindible, la existencia de un método general para construir traducciones en donde el contenido epistemológico –expresado en forma coloquial- se dé peladito y a la boca del lector contravendría al sentido común. Es más, la identificación de la traducción de Kauffman y de Fitelson como la comprensión en sí de la prueba de EQP de la conjetura de Robbins descansa más en el proceso y menos en el resultado final. Haciendo uso de nuevo de la metáfora de Kauffman, la comprensión radica en la interpretación de las partituras y no en éstas, sean completas, incompletas, inhumanas o suprahumanas. Entonces cabe reformular la pregunta sobre la existencia de los elìxires epistemágicos, ¿puede haber un método para traducir a toda prueba computarizada completa a un lenguaje donde sea viable su comprensión?

La respuesta oscila en una afirmación parcial, sí existen programas para transladar a las pruebas computacionales hacia sitios intermedios donde a veces es más fácil excarbar y extraer el contenido epistemológico. Por ejemplo Derrick Coetzee – ¿tendrá algún parentesco con el laureado por el Nobel?- elaboró el programa "po2hr"[84] que a partir de los objetos de prueba producidos por Otter genera un documento en Latex donde:

- Se traduce el lenguaje clausular al lenguaje implicacional de primer orden, i.e. las cláusulas se convierten en fórmulas con una implicación principal y se explícita a los cuantificadores universales.
- Eliminación de pasos redundantes u obvios.

- Reorganización de la prueba para disminuir la distancia entre los padres y fórmula derivada.
- Se especifican los términos, literales unificados en la aplicación de las reglas de inferencia.

Veamos un ejemplo con la demostración de un teorema sencillo de álgebra:

La función identidad (0) es unívoca.

Para facilitar las cosas, le regalamos de entrada a Otter el teorema más general :
Si f es una función y su inverso f^{-1} es una función bien definida entonces f es uno a uno.
Continuando con el espíritu caritativo, a Otter le damos como entrada la hipótesis $0=0^{-1}$.
El objeto de prueba construido por Otter con nuestros presentes es el siguiente:

```
(1 (input) ((not (FUNCTION v0)) (not (FUNCTION (inverse v0))) (ONEONE
v0)) (1))
(2 (input) ((not (ONEONE (0)))) (2))
(3 (input) ((FUNCTION (0))) (3))
(4 (input) ((equal (inverse (0)) (0))) (4))
(5 (instantiate 1 ((v0 . (0)))) ((not (FUNCTION (0))) (not (FUNCTION
(inverse (0)))) (ONEONE (0))) NIL)
(6 (paramod 4 (1 1) 5 (2 1 1)) ((not (FUNCTION (0))) (not (FUNCTION (0)))
(ONEONE (0))) NIL)
(7 (propositional 6) ((not (FUNCTION (0))) (ONEONE (0))) (7))
(8 (resolve 7 (1) 3 (1)) ((ONEONE (0))) (8))
(9 (resolve 2 (1) 8 (1)) () (9))
```

Po2hr redacta la siguiente traducción:

Assume the theorem is false. We know the following:

$$(\forall v_0)\ \text{FUNCTION}(v_0) \wedge \text{FUNCTION}\left(v_0^{-1}\right) \Longrightarrow \text{ONEONE}(v_0)$$

We then substitute 0 for v_0 to obtain

$$\text{FUNCTION}(0) \wedge \text{FUNCTION}\left(_0{}^{-1}\right) \Longrightarrow \text{ONEONE}(0)$$

which together with $0^{-1} = 0$ yields

$$\text{FUNCTION}(0) \wedge \text{FUNCTION}(0) \Longrightarrow \text{ONEONE}(0)$$

which reduces to

$$\text{FUNCTION}(0) \Longrightarrow \text{ONEONE}(0)$$

which together with FUNCTION (0) yields

$$\text{ONEONE}(0)$$

but this contradicts $\neg$ ONEONE (0).

Todavia admite muchas mejoras el programa, una de ellas sería la conversión correcta de las funciones de Skolem a cuantificadores existenciales o el diseño de métodos de reescritura para simplificar a los términos –estilo McCune o Burris-. Para poco serviría la traducción de po2hr de la prueba de EQP sobre la conjetura de Robbins pues sus cláusulas son unitarias y tienen una cantidad y disposición espacial de signos inhibidora de las ventajas de esta traducción.

No obstante, debe señalarse la inquietud de las ciencias computacionales –en particular IA- por el problema de la traducción, lo cual ha impulsado el desarrollo de técnicas para lograr traducir textos de un lenguaje natural –v.gr. chino- a otro –v.gr. castellano-. Ya hasta en los buscadores de internet (google, yahoo) se ofrecen herramientas de traducción de idiomas de una rudimentaria calidad. En nuestro caso se tiene la enorme ventaja de la reglamentación puntual de la sintaxis del documento original, la prueba computarizada, además los lenguajes destino, apegados a los lenguajes lógico-simbólicos de los matemáticos, tampoco ofrecen en un principio sintaxis tan complicadas como la de los lenguajes naturales. Así, aunque el sentido común abogue por el escepticismo, probablemente se desarrollen nuevas técnicas capaces de entregar mejores transcripciones para habilitar la interpretación epistemológica de los músicos de carbono de las pruebas computarizadas completas.

Resumiendo, el problema de la *revisabilidad* posee al menos dos mecanismos para solucionarlo, los cuales si actúan en conjunto pueden aminorar las dolencias o incluso aliviar al problema epistemológico. Los métodos para la reducción de la longitud de las pruebas y para la generación de traducciones aprehensibles deben depurarse bajo una investigación continua si aspiramos a comprender a las pruebas completas manufacturadas por las computadoras, de su avance depende la respuesta al problema. Para beneplácito de la caracterización disyuntiva sobre el *convencimiento* de las demostraciones enunciado en el capítulo segundo (si x es *convincente* entonces x es *computable* o *revisable*), la existencia de demostraciones *computables* no *revisables* es más que probable, al menos su existencia temporal es casi un hecho seguro, y se requerirían habilidades proféticas para determinar por cuanto tiempo carguen con esa cruz. Es un problema de alcance, por un lado los programas demostradores conforme se perfeccionen –incluyendo el aumento del poder del hardware- pueden maquilar pruebas más largas y más enrevesadas, mientras que los mecanismos de traducción y recorte de longitud luchan por arrastrarlas hacia zonas navegables para nuestra comprensión. En el peor de los casos, las medicinas actuales pueden no lograr curar la *irrevisabilidad* de la prueba sometida a tratamiento. Bajo este escenario, ¿dichas pruebas son *convincentes*? Vale la pena desviar un poco la atención hacia otras situaciones similares donde no hizo falta la intervención de la computadora diseminadora de la discordia. A lo mejor por pereza intelectual se suele olvidar el proceso de absorción de nuevas –para su época- técnicas, principios, estrategias, etc. utilizadas en las demostraciones que ahora consideramos normales pues desde el comienzo de nuestra carrera nos amamantan con ellas. Remover el polvo del olvido histórico sobre los avatares de la asimilación de esas técnicas, principios, etc. que forman ahora parte del arsenal manejado por cualquier matemático/especialista –inducción, reducción al absurdo,

principio de las palomas, diagonalización, "axioma del continuo", etc.- podría desvelar el desarrollo real de nuestras matemáticas. Algunas demostraciones hechas por matemáticos de carne y hueso no fueron totalmente *convincentes* para sus coetáneos porque rompìan o extendían las costumbres demostrativas de su tiempo, introduciendo esas técnicas, principios, etc. Los cambios exigen cierto tiempo para el reajuste, poco ha de sorprendernos esto. Sin embargo la novedad no es el único gatillo para disparar la controversia sobre el *convencimiento*, por ejemplo en la actualidad existen demostraciones ortodoxas cuya longitud y demanda de conocimiento sobre distintas ramas dificultan aunque no invalidan la caracterización mencionada en el capítulo pasado en donde se establece el *convencimiento* con base a la *revisabilidad (inteligibilidad)* o a la *computabilidad* de la demostración. Como ejemplo de una demostración de tamaño abominable y tarifas epistemológicas elevadas se tiene a la demostración de Andrew Wiles sobre la Conjetura Shimura-Taniyama y su corolario, el último teorema de Fermat[85]. Su lectura de comprensión para su posterior ratificación fue una obra titánica a varias voces, divide y vencerás, su *revisión* global fue la suma de las *revisiones* parciales de sus distintos módulos efectuada por varios especialistas. La interpretación epistemológica que abarque a toda la demostración está vedada para la mayoría de los matemáticos, sin embargo al superar el escrutinio de los especialistas y después de las correcciones pertinentes se concedió el visto bueno a la demostración y la consolidación de la fama de Wiles. Así, dicha demostración es *convincente* –la comunidad ya la aceptó- pues después del calvario de su *revisabilidad* resucitó en una demostración convalidada, sin embargo para el grueso de los matemáticos su *revisabilidad* es un acto prestado (aunque dicho sea de paso después de haber sido aceptada la pavimentación del camino hacia su entendimiento está en marcha con el asfaltado de los libros de texto, de este modo tarde o temprano su comprensión ya no será un bien tan escaso). Siguiendo el llamado de pesimismo gritado por este ejemplo, se puede plantear la existencia de una demostración tan larga cuya cuota de conocimiento sea tan alta y extensa para situarla fuera del alcance de lo *revisable* para los matemáticos excluyendo a su creador humano, además dicha demostración carezca de la depurada presentación de una prueba de un sistema formal. Así, la única manera para leerla por completo, entenderla y confirmar su validez sería someterse a un arduo y tardado entrenamiento en las artes epistemológicas utilizadas por el autor de tal esperpento. Hasta aqui el panorama no difiere tanto al caso de Wiles, sin embargo si nadie se anima a descifrar a la entrecomillada demostración por razones sociales (v.gr. son demasiados años de sacrificio y pocas expectativas de éxito o en caso de triunfar la gloria se la llevará otro) entonces nuestro terrible conejillo de indias, ¿es o no es una demostración? Recordemos la fórmula de las habichuelas mágicas: si x es una demostración entonces x es *convincente* o *computable* o *revisable (inteligible)*. Si recordamos, la demostración de H&A&K&1&0 provocó recelos sobre su condición de demostración pues no es *revisable* y poseía una dudosa *computabilidad*. Remarco mi malicia, estamos suponiendo válida a la hipotética demostración aunque sólo su creador lo sepa pues es el único quien puede *revisarla*. La demostración sin quórum carece de *convencimiento* y sin éste accede al limbo de la indeterminación. Me explico, por hipótesis es una demostración correcta y por conclusión nadie excepto uno la puede ver como tal. Quizás años después de su penitencia algunos matemáticos tras conseguir los conocimientos requeridos para *revisarla* la desentierren del

olvido y la lleven hacia el cielo donve vuelan las demostraciones con alas de moscardón, quizás los años pasen y sólo los gusanos que devoraron a su creador por compasión hacia su comida consideren válida a la demostración. Posible realización de este escenario fatalista lo representa la demostración de Louis de Branges sobre la Conjetura de Riemann[86] (la realidad es más terrible que la ficción). El factor social influye mucho más de lo que como matemáticos – o aspirantes a serlo - nos gustaría difundir públicamente, a veces para bien como con el caso de Wiles, a veces para mal como el caso de Louis de Branges. De este modo queda invalidado el recíproco de premisa reforzada, si x es *revisable y x es válida* no implica que x sea *convincente.* El caso de Louis de Branges muestra semejanzas con el objeto de nuestra discusión, pues aunque su demostración se suponga *revisable* se requiere un esfuerzo de comunicación para extender a tal propiedad. ¿Qué tanto cambia el panorama cuando nos situamos en los nubarrones causantes de esta lluvia de devaneos, presagiada desde el segundo capítulo, cuando tenemos una demostración *computable* indefinidamente no *revisable*? El escenario puede ser menos dramático con las pruebas computarizadas completas, pues –mencionado ya hasta el cansancio- están arropadas por la experiencia de la comunidad con los sistemas formales, ya nadie pone en duda la validez de una prueba una vez pasada su verificación (humana). Convenientemente si igualamos convencimiento con aceptación, entonces las pruebas computarizadas completas conquistan al convencimiento con su *computabilidad*. La única barrera para la aceptación de las pruebas computarizadas completas se alzaría cuando el agente encargado de la verificación sea un programa, pero su verticalidad es más una cuestión de costumbre que un muro teórico amenazador. Las computadoras se equivocan, pero dentro nuestro mundo lleno de imperfecciones, son mejores que nosotros en la tarea de la verificación de las pruebas. Talvez algún filósofo trasnochado reniegue de la verificación por computadora bajo el supuesto de la incorruptibilidad del conocimiento matemático, yo le recomendaría paciencia para lograr su lanzamiento a través de la muerte hacia ese mundo ideal donde todas las almas son iguales y todos los teoremas son verdades puras. Sin embargo, ¿es correcto reducir al *convencimiento* a la mera aceptación? En el capítulo pasado recomendé cautela en la conjunción en la caracterización del *convencimiento* propuesta por Tymoczko, incluso en su postrimería postulé únicamente una implicación disyuntiva: si x es *convincente* entonces x es *computable* o x es *revisable* ¿Por qué evito el regreso? Por las demostraciones a la de Branges, demostraciones hipotéticamente *válidas* y *revisables* (conjeturar su *formalización* sería el colmo de los supuestos) pero carentes de convocatoria. Porque me parece insuficiente la justificación lógica, porque no se puede entender a las matemáticas sin considerar a los procesos involucrados en la gestación del conocimiento y algunos de ellos son de índole social.

¿Por qué una demostración puede ser convincente? Porque existen lectores (plural forzoso) interesados y capaces de *revisarla*, o como lo sugerí, porque esos zancudos de alguna manera pueden extraer la sangre epistemológica que da vida a ese cuerpo de signos. ¿Por qué confiamos en el conocimiento succionado de ella? Porque ese conocimiento puede destilarse con el alambique del rigor, porque en última instancia la demostración es formalizable –*computable*-. Porqué obedeciendo a Hilbert y toda la pandilla de lógicos, médicos especialistas en las enfermedades deductivas, cuando nuestro conocimiento se ha

convertido en axiomas y nuestras demostraciones en pruebas, el sano ambiente de los sistemas formales ha fortalecido hasta su punto máximo al sistema inmunológico de nuestras pruebas, fagocitando a la mayoría de los gérmenes de la equivocación y bombardeando con la radiación formal al cáncer de la duda (una duda lleva a otra, su metástasis es inminente). Aunque como sucede en la vida diaria, pocos hacen caso de las sugerencias de sus médicos, así las *computabilidad* es un ideal aunque no una necesidad para el quehacer demostrativo. La negligencia no es el principal motivo para esa desatención, ponerse en forma consume mucho tiempo y esfuerzo aprovechable en la recolección de nuevo conocimiento. Formalizar es nada sencillo. Además, existe otra causa a continuación expuesta. ¿Por qué una prueba computarizada completa por el momento *no revisable* es *convincente*? Porque es *computable,* porque ya está sujeta a la presentación de una prueba en un sistema formal y a su subsecuente validación. ¿Por qué entonces aunque *convincente* puede ser no tomada en cuenta? Porque los matemáticos difícilmente sacarán a bailar a las demostraciones recatadas y feas. Porque aunque esté encerrada en sus entrañas la respuesta, de nada sirve sino pueden ser sus puertas abiertas. Nuestro conocimiento matemático es una construcción social –cuidado no escribí "el conocimiento..."-. Sin comunicación no hay sociedad. Las pruebas computarizadas *computables* y no *revisables* pueden quedarse indefinidamente en la zona de las demostraciones despechadas, sin pretendientes para conducirlas a la pista de la labor matemática, al menos sin pretendientes carentes de piernas cibernéticas. Ser *convincente* engloba también propiedades comunicativas, disuasivas; no basta con la imposición formal de la validez de una prueba; se necesita seducir al lector para que trabaje en ella, para que la desnude epistemológicamente y como consecuencia de ese coqueteo la mejore, generalice el teorema demostrado o lo relacione con otros teoremas de la teoría o fuera de ella. La demostración es el fin del comienzo del teorema. Somos criaturas superfluas, la superficialidad es un requisito para la profundidad, la presentación exterior de una demostración determina si podamos o no llegar hasta su médula epistemológica. El convencimiento pleno de una demostraciòn –así como el desarrollo de ésta- dependen de su *revisabilidad*. Sorpresa sardónica si las demostraciones nacen con una presentación de escaso rigor, la flexibilidad fomenta a la comunicación y alienta al desarrollo, al crecimiento y reproducción del conocimiento. Tampoco debe extrañarnos que la formalización si alguna vez llega normalmente lo haga en una etapa madura, cuando la demostración del teorema, el teorema y la teoría donde se incrusta han alcanzado un nivel en donde convenga asegurar su posteridad. No obstante, la formalización no es la muerte de una demostración ni de una teoría, hechos mostrados en este capítulo, las pruebas se depuran, la teoría crece con nuevos teoremas probados en el sistema. Las pruebas computarizadas completas hacen imperioso revertir el proceso, no tanto en la pérdida del rigor, sino en la ganancia de la afabilidad de la representación. Toda la anterior alegación se ha saltado una pregunta elemental, ¿qué tanto se pierde con la ausencia de *revisabilidad* en las pruebas computarizadas completas? Trataré de arreglar este brinco en el próximo párrafo.

¿Para qué nos sirve la *revisabilidad* en una demostración? Para ratificar su condición de demostración y para deshilvanar a los hilos epistemológicos enredados en ella y entre

otras cosas así tejer al conocimiento del teorema demostrado. Con la *computabilidad* basta para satisfacer el primer uso de la *revisabilidad,* en el segundo se esconde una pregunta crucial. ¿Qué aprendemos de una demostración gracias a su *revisabilidad*? Respuesta ancestral, tan vieja como lo es la necesidad de demostrar en las matemáticas, comprender a la demostración es conocer al teorema, al menos en parte. ¿Conocer al teorema? Si, por medio de la demostración podemos identificar las relaciones de ésta con el resto de la teoría (definiciones, axiomas, teoremas) en donde vive o incluso con teorías extranjeras, saber ésto puede ocasionar simplificaciones, nuevas relaciones y nuevas definiciones, restricciones, generalizaciones, en suma, más conocimiento. Por otro lado a partir de la demostración se pueden recoger técnicas, estrategias, procedimientos demostrativos aplicables a otros teoremas cuando nuestra habilidad para hacer analogías con acuidad nos lo aconseje. Primera acotación, a veces las demostraciones son muy parcas y no ofrecen muchas de estas maravillas epistemológicas. Existen demostraciones muy técnicas que a lo sumo nos indican la validez del teorema y en el mejor de los casos nos entrenan para demostrar parientes cercanos. Como sea, por medio de la *revisabildad* nos percatamos del alma epistemológica encerrada en los signos de la demostración independientemente de su belleza y su potencia. Además, para quien no esté ciego como un topo de nombre Edipo, la *revisabilidad* habilita la comunicación y ergo el desarrollo del conocimiento, conocimiento construido por el trabajo de varios matemáticos sobre la demostración, el teorema, la teoría, las teorías. Aclarado esto, ¿cómo puede compensar la *computabilidad* de las pruebas computacionales completas la temporal (el optimismo es una cara feliz del cinismo) ausencia de la visión de las entrañas epistemológicas facultada por la *revisabilidad*? Vamos por partes, una prueba computarizada completa exhibe con precisión mecánica la relación del teorema demostrado con los axiomas, teoremas y objetos de la teoría o las teorías involucradas en ella. Mientras que, al estilo del paradigma de Argonne, una prueba exitosa puede distribuir el empleo de las estrategias, reglas de inferencia, procedimientos triunfales en otros teoremas cercanos y con suerte bajo las pertinentes modificaciones se encontrarán sus pruebas. Los canales de comunicación no están clausurados cuando los oyentes además de sus dos orejas simiescas cuentan con el amplificador llamado computadora. McCune-EQP en el 96 emitieron la prueba de la conjetura de Robbins, el receptor Veroff-Otter en este siglo hicieron lo suyo. Cuando la *revisabilidad* anda de parranda en las pruebas computacionales completas el conocimiento no está muerto, aunque para explotar cabalmente los remanentes epistemológicos se hace forzoso otra vez recurrir a las computadoras. ¿Se avecina una revolución en los usos y costumbres en el epónimo de las disciplinas deductivas? Corresponde ahora demarcar en dónde puede ser y ha sido aplicado exitosamente el razonamiento deductivo automatizado, ubicar en dónde no le ha ido tan bien, puntualizar sus perspectivas de expansión en las matemáticas; quien mejor que Wos para empezar a hacerlo:

[en matemáticas] las áreas de álgebra abstracta son particularmente dóciles para estudiar con la ayuda de un programa de razonamiento automatizado, áreas que incluyen la teoría de grupos, la teoría de anillos, la teoría de látices, semigrupos y la geometría algebraica. La docilidad descansa principalmente en el tipo de axiomatización que frecuentemente se

encuentra, a saber, un conjunto de ecuaciones que se pueden representar con cláusulas unitarias.[87]

[en la lógica] sobre todo cuando la separación condensada es la regla de inferencia (desde el punto de vista de la lógica), varias áreas son muy dóciles de estudiar con el ayudante de razonamiento automatizado.
Las áreas incluyen al cálculo de equivalencias, al cálculo proposicional multivaluado y también al bivalente. [88]

Con respecto al álgebra Ken Kunen enfático señala:

Hay dos eras en la historia de los axiomas individuales para grupos y variedades de grupos. Los resultados tempranos, por Neumann y otros, en donde lo usual era producir axiomas individuales mucho más largos que los mínimos posibles, pero que, eran construidos bajo algún esquema que facilitaba su verificación a mano. Aunque no óptimos, estos resultados tenían la virtud de que cualquier matemático pudiera conceptualmente captar a estas demostraciones. La segunda era inició con el advenimiento del programa de razonamiento automatizado Otter, hecho por McCune, ahora gracias a la Nutria podemos construir axiomas individuales más simples y más cortos, aunque a cambio normalmente requieren verificaciones más intrincadas. [89]

En general, el poder actual del razonamiento deductivo automatizado ha impulsado a la consecución de resultados en teorías basadas en la igualdad o en teorías de pequeña escala. En una de las dos áreas críticas propuestas por Alan Robinson se han obtenido y se seguirán cosechando logros. ¿Qué pasa con la otra joya de Robinson? Contesta Wos:

En contraste con el álgebra, el área de la teoría de los conjuntos ha provocado estragos con el razonamiento automatizado en general. Claro, el progreso está ocurriendo en tal área, vea (Quaife 1992; Belinfante 1998a; Belinfante 1998b). No obstante, el área citada todavía presenta muchas asperezas. Más en general, si la representación incluye cláusulas no-corniformes o cláusulas que simultáneamente contienen literales de igualdad y literales con algún otro predicado, entonces usted puede esperar problemas. [90]

¿Qué pasa con la teoría de los conjuntos? Con una relación –la pertenencia- y con menos de una decena de axiomas en un sistema de primer orden puede ser construida ¿Cuál es el problema entonces? Tratémoslo de aislar con una axiomática usual de la teoría de conjuntos escrita en primer orden, la de Zermelo-Fraenkl. El axioma de comprensión de ZF es un esquema infinito, con una instancia-axioma para cada fórmula del lenguaje, de donde la explosión infita de axiomas no puede representarse con nuestro pobre lenguaje clausular de manera finita[91]. El primer problema son los sistemas axiomáticos infinitos intratables. La otra causa principal de la catatonia –según Wos[92]- recae en la ausencia de una regla de inferencia especial para el razonamiento con conjuntos, similar a la paramodulación con respecto al razonamiento con la igualdad. Los progresos de Quaife y Belinfante inhiben al primer catalizador del fracaso, al recurrir a la axiomática finita de Gödel-Bernays (GB), la

cual caracteriza a la teoría de conjuntos sin salirse del primer orden ni tentar al infinito. Belinfante manejando a Otter con la axiomática GB pudo probar más de 1000 teoremas de la teoría de conjuntos[93], ninguno relevante para esa teoría pero en conjunto con el trabajo de Quafie aunque hayan dado un pequeño paso para la teoría de conjuntos, han dado un gran paso para el razonamiento deductivo automatizado enfocado a los conjuntos, acercándose al menos un poco a la conquista del otra área crucial presagiada por Alan Robinson. Además, en teoría si se consiguiera dominar a la teoría de conjuntos entonces las puertas de casi todos los departamentos de las matemáticas estarían abiertas para los programas de razonamiento automatizado deductivo, pues "esencialmente" la mayoría de las matemáticas puede ser caracterizada en términos de esta teoría. Sin embargo en la práctica, como lo señala otro investigador del área de apellido Beeson, manejar por medio de la teoría de conjuntos a las funciones, números, etc. es demasiado complicado, tanto que casi ningún matemático no interesado en esa teoría lo hace en su trabajo diario, por lo cual tampoco debiera hacerlo hasta nuevo aviso un programa de razonamiento deductivo automatizado si la meta es probar nuevos teoremas[94].

Si se columbra una revolución demostrativa digital por parte de los programas aquí descritos, entonces la sedición por el momento se ve de escala pequeña, de tamaño similar a las teorías al alcance del poder de dichos esclavos digitales. ¿Cuando valdrá la pena preocuparnos por los rumores del levantamiento de las computadoras demostradoras? Cuando sus acciones salgan de las teorías chiquitas y tomen por asalto a teorías más amplias y potentes. ¿Cómo? Dominando al razonamiento deductivo automatizado enfocado a los conjuntos y esperando que Beeson se trague las palabras de su comentario (94). Para los matemáticos humanos hacer su trabajo desde los términos de la teoría de conjuntos resulta difícil, es un camino largo y penoso. Sin embargo tal vez no sea así para los programas de razonamiento deductivo automatizado, poseen ventajas –mayor memoria, mayor velocidad y precisión en las operaciones deductivas mecánicas- sobre nosotros que tal vez invaliden las palabras de Beeson, ya el tiempo lo dirá. Otra manera para diseminar las semillas de la revolución sería conseguir programas de razonamiento automático deductivo de órdenes superiores. Aunque la matemática –por medio de la teoría de los conjuntos- pueda ser en su mayoría redactada en un lenguaje de primer orden, los matemáticos en su labor cotidiana hacen uso de lenguajes de orden superior, especialmente de lenguajes de segundo orden. Existen programas demostradores de teoremas en el mercado especializados en órdenes superiores, v.gr. *Theorema*[95], pero ninguno iguala los triunfos de Otter ni en general buscan hacerlo, en palabras de Beeson[96], el desarrollo de estos programas parece no aspirar a la demostración de nuevos teoremas. Por este motivo, Beeson busca desarrollar una versión de Otter capaz de manejar lenguajes de segundo orden, sin interferir con sus esplendorosas facultades de primer orden[97]. Sintetizando, aunque la presencia de los programas de razonamiento deductivo automatizado en las matemáticas sea cada vez más conspicua, su notoriedad abarca un muy restringido ámbito, por lo cual ninguna revolución a gran escala se presagia en un futuro próximo en la labor demostrativa. Sin embargo por todos sus logros demostrados, más vale estar pendientes de las pequeñas pero continuas victorias de los programas de razonamiento deductivo automatizado.

Consecuencia desalentadora de la conclusión del párrafo anterior, la majestuosa obra de la formalización de nuestro conocimiento matemático carece indefinidamente de la maquinaria pesada para la extracción de nuevas pruebas, maquinaria útil para la construcción de esta biblioteca de Alejandría. Los programas demostradores actuales abarcan una nimia porción de las matemáticas. Siendo más exactos con respecto a los programas aquí analizados, Otter y los demás demostradores de Argonne ni siquiera cumplen con los requerimientos básicos para la digital edificación formal de las matemáticas. Los programas de Argonne han sido punta de lanza en el desarrollo del razonamiento deductivo automatizado, han sido pioneros en la cacería de pruebas de conjeturas. Otter, aun contando con la sintaxis de un sistema formal y con su asombroso poder deductivo limitado a la lógica clásica de primer orden, adolece la falta de un verificador y de una biblioteca (base de datos de los teoremas-pruebas conocidos, axiomas, definiciones, teorías) integrados en él. Se pueden compensar estas ausencias, por ejemplo considerando el binomio Otter-Ivy (Ivy[98] es el programa verificador de pruebas usado actualmente en Argonne). No obstante, desde hace varios años rielan tímidamente algunos sistemas computacionales diseñados para satisfacer –al menos parcialmente- a los deseos grandilocuentes de la formalización de las matemáticas. El más añejo de los activos (el más viejo sería Automath de 1967, ver primer capítulo) y poseedor de la biblioteca más grande, comenzó a formalizar a las matemáticas desde 1973 y se llama Mizar[99]. El lenguaje de Mizar está basado en el lenguaje de la teoría de conjuntos (axiomatización de Tarski Grothendieck en lenguaje de primer orden), las pruebas son escritas por los usuarios y al programa le corresponde su verificación. De este modo, con el sistema Mizar la construcción de las grandes pirámides del conocimiento matemático depende de la mano de obra humana corregida por los capataces de la verificación automática. Ante esta perspectiva, afín a todos los proyectos computarizados en pos de la construcción de una de las siete maravillas del conocimiento, se hace forzoso contar con una gran infraestructura, se requieren un número elevado de matemáticos, computólogos y un solvente presupuesto para mantener a todos estos alienados más los equipos de cómputo. Con una inversión tan elevada no será difícil adivinar el armado en cámara superlenta de la colosal estructura, la velocidad podría incrementarse y los costos reducirse si contásemos con electrónicos asistentes incansables aptos para manufacturar pruebas de teoremas en cada vez más pisos de las matemáticas. Además en las empresas de esta envergadura, emergen otras problemáticas externas a su núcleo como son el diseño de una interfase eficaz, la estandarización del lenguaje, la divulgación-distribución para atraer a más clientes-esclavos (en este sentido internet es el canal de distribución idóneo). Como sucede con las heurísticas, aunque el proyecto de la formalización mediante los sistemas de cómputo ofrezca las ventajas ya mencionadas, como lo son la solidificación-ordenamiento del conocimiento matemático, su ubicuidad gracias a internet o sus beneficios pedagógicos, debe evaluarse si vale la pena cubrir el elevado costo del proyecto. ¿Acaso se colapsaría o suspendería el desarrollo matemático sin esta obra titánica? La respuesta negativa cimbra toda esperanza de la culminación cercana de esta nueva biblioteca de Alejandría, las matemáticas sin ella pueden continuar su vida con sus errores y sus aciertos -algunos de los cuales podrían ser corregidos y fomentados respectivamente con la guía de este faro-. La

posteridad es la más sugestiva de todas las tentaciones, por lo que eventualmente siempre arribarán moscas a la miel –o hiel- de este monumental proyecto. En la década de los noventas (1993) algunos de los más prestigiosos investigadores del campo del razonamiento automatizado deductivo –Wos, Kunen, Beeson, McCune, etc.- encabezados por Robert Boyer conformaron al colectivo QED[100], manifestando la necesidad e impulsando el desarrollo de la formalización de las matemáticas por medio de las computadoras. Incluso sus perspectivas eran más amplias, instaban a construir una base de conocimiento matemático en donde también deberían incluirse programas de manipulación simbólica, software para modelar y simular y sofware de análisis numérico. Tuvieron un par de congresos (1994 en Argonne, 1995 en Varsovia) y hasta ahí llegaron, ya por 1996 la cohesión del grupo se disolvió. En el coloquio de 1995, Boyer vaticinó –según lo reporta Roman Matuszewski- a la desbandada y al atolladero a la vuelta de la esquina:

[según Boyer] Mientras que un número grande de matemáticos y computólogos están marcadamente a favor del proyecto de QED y esperan su construcción en el siguiente siglo o dos, las esperanzas en el progreso actual del proyecto están oscurecidas debido a problemas sociológicos y políticos. Al menos por el momento hay algo de apoyo económico y hay oportunidades para la publicación (pulso del estado de un área científica) de trabajos acerca de nuevas maneras para intentar realizar un razonamiento automatizado efectivo y, claro, para la publicación de resultados matemáticos nuevos logrados a través de él. Pero persiste el temor que hay muy pocos fondos y escasas oportunidades de publicación en la empresa de la construcción colectiva del enhiesto edificio de las pruebas de teoremas verificadas por computadora. Así, la mayor esperanza de Boyer en el congreso de QED es dilucidar cómo superar estos problemas sociológicos y políticos, para establecer motivaciones pragmáticas o personales que fomenten la cooperación con este esfuerzo colectivo.[101]

El desolador presente lleno de escasez, fue el regalo de despedida de QED y sigue siendo el presente de aquellos obsesionados con la formalización computarizada de las matemáticas. No obstante con paciencia, austeridad y terquedad se pueden enfrentar las carencias, Mizar predica con el ejemplo. De la mano de su fundador Andrzej Trybulec, desde 1973 la biblioteca Mizar mantiene su ritmo (tal vez pausado pero insistente) de crecimiento, actualmente cuentan con alrededor de dos decenas de colaboradores primarios concentrados en tres universidades del mundo (nota (99)). En la dirección electrónica http://merak.pb.bialystok.pl/ se exhibe y actualiza a la biblioteca Mizar, hasta junio del 2005 han contribuido 171 autores con 909 artículos, tiene 40923 teoremas y 7721 definiciones. Algunos de los teoremas más renombrados a su disposición son:

"Baire Category Theorem for Continuous Lattices"
"Banach Fix Point Theorem for Compact Spaces"
"Binomial Theorem"
"Birkhoff Variety Theorem "
"Brouwer Fixed Point Theorem for Disks on the Plane"
"Brouwer Fixed Point Theorem for Intervals"
"Cantor Theorem"

"Caratheodory's Theorem"
"Carmichael's Theorem on Prime Divisors "
"Cauchy Theorem"
"Chinese Remainder Theorem"
"Contraction Lemma"
"Deduction Theorem"
"Euler's Theorem"
"First isomorphism theorem for groups"
"First isomorphism theorem for universal algebras"
"Fundamental Theorem of Algebra "
"Fundamental Theorem of Arithmetic "
"Goedel Completeness Theorem "
"Hahn-Banach Theorem (complex spaces)"
"Hahn-Banach Theorem (real spaces)"
"Heine-Borel Theorem for intervals"
"Hilbert Basis Theorem"
"Jordan Curve Theorem for special polygons"
"Koenig Lemma"
"Koenig Theorem"
"Kuratowski-Zorn Lemma"
"Zorn Lemma"
"Lagrange Theorem"
"Lagrange Theorem for Groups"
"Lebesgue's Covering Lemma"
"Rolle Theorem"
"Second isomorphism theorem for groups"
"Small Fermat's Theorem"
"Third isomorphism theorem for groups"
"Weierstrass Theorem"
"Zermelo Theorem"
"l'Hospital Theorem"

Sigue la yunta andando, los jornaleros en su mayoría polacos y japoneses siguen cultivando al campo de Mizar, sistema que lleva la delantera en la carrera de la formalización, pues hay otros desperdigados en algunas universidades del mundo[102] (como el ya mencionado por otros asuntos Theorema). Si usted conoce algún millonario excéntrico con la mira de plasmar su nombre en la inmortalidad por medio de una obra faraónica, propóngale donar su fortuna al proyecto del Genoma Matemático (quizás a algún narcotraficante ya le haya aburrido su apoteosis cantada en corridos y prefiera lavar dinero con este proyecto donde además en unos cientos de años –expectativa de Boyer- se ganará la adulación de toda la fauna matemática y anexos).

En fin, el fin del capítulo ya llegó. Reflexiones y acotaciones terminales. El área de investigación del razonamiento deductivo automatizado es vigorosa y pujante. Cada dos

años se monta un congreso (CADE-*Conference on Automated Deduction*) para mostrar sus avances y en donde además se divierten con la competencia de los demostradores automáticos (CASC-*CADE Automated Theorem Proving System Competition*) mediante una biblioteca de problemas (TPTP-*Thousands of Problems for Theorem Provers*) diseñados para retar a los talentos de los programas[103]. Existe también una publicación mensual especializada en el área, llamada en inglés *Journal of Automated Reasoning*[104]. Hay vida fuera de Argonne, de hecho los últimos campeones de CASC provienen de otros centros de investigación. La variedad de los demostradores de teoremas en el mercado debe también recalcarse[105], en Argonne sólo han trabajado con una lógica clásica de primer orden montada en un sistema resolutivo, pero los hay con lógicas intuicionistas (constructivistas), de órdenes superiores (teoría de categorías), basados en la aritmética primitiva recursiva, en el cálculo lambda de Church, no tan automatizados, sin tantas estrategias, etc., ya en el próximo capítulo se explicará brevemente otra escuela –la del demostrador Boyer-Moore- centro de pruebas del tema del capítulo cuarto, la verificación formal del software-hardware. Sin embargo son los sistemas afiliados (o cercanos) al paradigma clausular defendido por Argonne quienes han aportado más nuevos resultados matemáticos y quienes además producen los objetos deductivos mejor acabados, objetos con el derecho a ser considerados demostraciones, demostraciones bautizadas con el nombre de pruebas computarizadas completas. Recapitulando, el paradigma clausular está conformado por los siguientes incisos:

(i) Uso de lenguaje clausular para representar a las fórmulas
(ii) Sistema resolutivo (e hijos) para la derivación de las pruebas (por contradicción)
(iii) Uso de estrategias (dirección y restricción), retención de cláusulas y procedimientos para administrar las redundancias

Las pruebas producidas bajo ese paradigma obedecen a la caracterización disyuntiva propuesta con base a la sugerida por Tymoczko en el segundo capítulo:

$$\forall x(\text{DEMostración}(x) \Rightarrow \text{CONvincente}(x) \vee \text{COMputable}(x) \vee \text{REVisable}(x))$$

ya que satisfacen al segundo sumando de la conclusión, son *computables;* además la caracterización llave para abrir la caja negra de la labor demostrativa matemáticas:

$$\forall \text{dem}(\text{CON}(\text{dem}) \Rightarrow \text{COM}(\text{dem}) \vee \text{REV}(\text{dem}))$$

es decir las demostraciones son convincentes debido a su cimentación lógica o a su concentrado epistemológico, también es satisfecha por las pruebas computarizadas completas ya que son *computables* desde su origen por lo que estas demostraciones siempre satisfacen a la disyunción en el miembro derecho de la implicación caracterizadora del CONvencimiento. Además, al tener el marco de un sistema formal la *aceptación* de este tipo de pruebas radica en la validación de sus pasos derivativos, los procesos de verificación pueden ser motivos de duda empero. No obstante con insistencia se ha argumentado a favor de la incorporación de los programas de computadora para

realizar esta tarea pues son más eficientes que nosotros en esos menesteres. Y si son mejores para ese trabajo que sus patrones humanos, entonces el hombre medida de todas las cosas no tiene otra opción más que *aceptar* esas validaciones con sus respectivas pruebas computarizadas completas. Y ser *aceptado* es una instancia de ser *convincente.*
Para este clase de demostraciones la *inteligibilidad* no puede suplantar a la *REVisabilidad* pues por el momento la única manera de extraer el contenido epistemológico de la prueba computarizada completa es a través de una inspección total de ella y para que la exégesis puede llevarse a cabo se necesitan pruebas más cortas y métodos de traducción a lenguajes más afables con la lectura humana. Por otro lado, entendidas las matemáticas como una actividad social se debe tender a la conjunción en la previa caracterización:

$$\forall dem(CON(dem) \Rightarrow COM\ (dem) \wedge REV(dem))$$

Dicho sea de paso, por motivos de comunicación, el *convencimiento* usual del resto de las demostraciones, a diferencia de las pruebas computarizadas completas, radica más en su *revisabilidad* (o *inteligibilidad*). Conforme se aumenta las cuotas de rigor para adquirir el *convencimiento* de los matemáticos, a las demostraciones se les exige tender hacia su presentación formal, aunque casi nunca se llegará a pedir el extremo de la prueba formal (la mayoría se conforma con saber que en teoría toda demostración debe poder ser formalizada, aunque haya algunos locos –Mizar,QED- con el ímpetu de corroborarlo).

Aunque nuestra caracterización de ser *convincente* invita a la simplificación de la caracterización de una demostración, si nos interesa mantener el contacto con la realidad del desarrollo matemático, ante la posible existencia (la demostración de DeBranges sobre la Conjetura de Rienman) de demostraciones *"revisables"* con un nulo poder de *convencimiento* no podemos eliminar impunemente a ese predicado Hay otros factores involucrados en el *convencimiento* además de la *revisabilidad* y la *computabilidad*, para bien y para mal algunos son factores de caracter social. Bajo este enfoque histórico-sociológico la caracterización tornaría en la siguiente formulación:

$$\forall x(DEM(x) \Rightarrow CON(x) \wedge (COM\ (x) \vee INT(x)))$$

De hecho los radicales la escribirían así:

$$\forall x(DEM(x) \Rightarrow CON(x))$$

pero eso dice tanto como A implica A, tanto como el conocimiento matemático es aquella base de datos avalada y alimentada por los matemáticos. La anterior caracterización renuncia a desentrañar las características de la demostración y presumiblemente alienta a cotillear sobre los contextos. Esfuerzo enriquecedor, pero a la vez paupérrimo, si explicamos a la Tercera Sinfonía de Beethoven únicamente a partir de las condiciones políticas y sociales reinantes en la Europa cuando fue compuesta, olvidándonos de la música y concentrándonos en la explicación del borrado de la dedicatoria a Napoleón de la partitura. Esfuerzo coherente, pero a la vez incongruente, si nos encerramos en la trinchera

filosófica-matemática sin considerar a los elementos sociales partícipes en los procesos demostrativos. Así, si nos importa un comino la interferencia social entonces redactaríamos a la caracterización del siguiente modo:

$$\forall x(DEM(x) \Rightarrow (COM\ (x) \vee REV(x))$$

Es más, ya encarrerados, por qué no escribirla así:

$$\forall x(DEM(x) \Leftrightarrow (COM\ (x))$$

Aparte de su intermitente contacto histórico, la última caracterización fue diezmada con el análisis de las pruebas computarizadas completas no *revisables* (temporalmente), sin comunicación el desarrollo se coagula, el avance se bloquea, el conocimiento se solidifica pero su endurecimiento lo hace menos maleable. Con la asistencia de alguna maquinaria capaz de derretir a esas rígidas demostraciones, se puede rescatar una parte del intercambio epistemológico empero. Pero tal maquinaria, la computadora, todavía está en un estado embrionario, actualmente ni los programas ni el hardware ni los usuarios tienen la fuerza para sostener a la última caracterización. Si bien están en marcha los proyectos de la construcción formal computarizada –Mizar- o están dando resultados los programas demostradores de teoremas –Otter-, ninguno muestra una fase tan avanzada (ni la tendrá en un futuro cercano) como para atreverse a defender a la idílica doble implicación, hacerlo, sería promover el fin de las matemáticas como las conocemos.

Me despido con una cita de Wos para asistir a la cita de los matemáticos con los programas de razonamiento deductivo automatizado:

Si algunas preguntas abiertas pudieron ser contestadas por investigadores con tan poco conocimiento en las áreas de los problemas, confiándose al poder del programa de razonamiento automatizado, entonces los matemáticos con credenciales sólidas en esos campos podrían producir resultados sorprendentes de verdad.[106]

POSDATA: LABERINTO SEMÁNTICO

En el primer capítulo se prometió relatar de nuevo a la incorporación de la semántica en los programas demostradores de teoremas, para denotar su potencialidad no restringida a la mera derivación sintáctica. Ya fue cumplida esta promesa, con la explicación de la vía de Herbrand. Después se reseñó el arribo de la resolución y la era de la dominación sintáctica (con un tenue rastro semántico en el proceso de la unificación). No obstante la semántica al estilo Herbrand ha resurgido con los trabajos de Mark Stickel, Zhang, McCune entre otros[107]. Es una semántica de producción industrial, nada que ver con la guía semántica, resemblanza de la derivación diagramática humana de la Máquina Geométrica de Galernter. En sentido contrario a la vía de Herbrand, los constructores-enumeradores de modelos de primer orden buscan desesperadamente un dominio de interpretación finito donde se satisfaga el conjunto de hipótesis y a la conclusión negada, encontrando de este modo un modelo contraejemplo. Adentrémonos en MACE, programa escrito por McCune para complementar a Otter.

Por el momento hay dos versiones de MACE (2,4), diseñadas con metas y técnicas distintas. Los dos manejan un lenguaje y obedecen instrucciones similares a las de Otter, de donde los archivos de entrada son parecidos. MACE 4 está diseñado para cláusulas con predicados de igualdad, MACE 2 para problemas relacionales sin un número elevado de términos anidados. Por derecho de antigüedad, perdámonos en MACE 2.

El enunciado a ser modelado debe estar escrito en un lenguaje clausular. MACE 2 convierte a cada cláusula en una cláusula relacional (las funciones n-arias se traducen en relaciones n+1-arias), luego con base a la cardinalidad definida del modelo se construyen las instancias aterrizadas. Después se recurre al procedimiento DPLL para determinar la satisfacibilidad del conjunto de cláusulas aterrizadas, si resulta satisfacible entonces se traducen a un modelo de primer orden. Transcribo al ejemplo bonito del manual[108]:

¿De la identidad e inverso izquierdos más la asociatividad se puede deducir la conmutatividad de un grupo?
group.in

```
set(auto).
list(usable).
  e * x = x.                    % left identity
  g(x) * x = e.                 % left inverse
  (x * y) * z = x * (y * z).    % associativity
  a * b != b * a.               % denial of commutativity
end_of_list.
```

Dos son las alternativas usuales desprendidas de una conjetura, para seguir a la primera podemos poner a trabajar a Otter con las miras de su validación:

% otter < group.in > group.out

Y a MACE lo ponemos a buscar su refutación, en este caso con un modelo de cuatro elementos:

% mace -n4 -p -c < group.in > group4.out

Los dos fallan, se les acaba el tiempo. A raíz del fracaso de Otter y después de inspeccionar su salida, la intuición nos impele a dudar sobre la conjetura, por lo que aumentamos la cardinalidad del modelo a seis:

% mace -n6 -p -c < group.in > group6.out

¡Eureka! MACE genera la salida:

```
======================= Model #1 at 1.13 seconds:
e: 0
a: 1
b: 2
*:   | 0 1 2 3 4 5
   --+------------
   0 | 0 1 2 3 4 5
   1 | 1 0 3 2 5 4
   2 | 2 4 0 5 1 3
   3 | 3 5 1 4 0 2
   4 | 4 2 5 0 3 1
   5 | 5 3 4 1 2 0

g:     0 1 2 3 4 5
   ---------------
       0 1 2 4 3 5
```

El anterior modelo es el contrajemplo que refuta a la conjetura.

MACE 2 ha asistido a la resolución de problemas existenciales de cuasigrupos, problemas de latices y álgebras booleanas[109].

NOTAS

(1)El alcance de un cuantificador es la fórmula donde en ella ejerce su acción dicho cuantificador. La correcta escritura de los paréntesis facilita determinar el alcance. Por ejemplo el alcance de $\forall x_1$ en
$\forall x_1(\forall x_2(A(x_1)\Rightarrow\neg B(x_2)))\Rightarrow C(x_1)$ es $\forall x_2(A(x_1)\Rightarrow\neg B(x_2))$; en este caso conviene hacer la sustitución de x_1 por otra variable no usada , v.gr. x_3 para señalar que x_3 ocurre libremente en la fórmula, es decir, x_3 está bajo el alcance de ningún cuantificador.

(2)Wang, Hao, *Computer Theorem Proving and Artificiall Intelligence,* Contemporary Mathematics 29, pág. 49-70

(3)Una fórmula está en su forma prenex si todos sus cuantificadores están al principio, i.e. no hay cuantificadores bajo el alcance de un conectivo lógico. Las siguientes reglas permiten arropar a una fórmula con su forma prenex. P es una fórmula, la variable x no ocurre libremente en P, "(x)" es para todo x, "-" es la negación:

(1)	$-(x)Fx$	$(\exists x)-Fx$
(2)	$-(\exists x)Fx$	$(x)-Fx$
(3)	$P\,\&\,(x)Fx$	$(x)(P\,\&\,Fx)$
(4)	$P\vee(x)Fx$	$(x)(P\vee Fx)$
(5)	$P\,\&\,(\exists x)Fx$	$(\exists x)(P\,\&\,Fx)$
(6)	$P\vee(\exists x)Fx$	$(\exists x)(P\vee Fx)$
(7)	$P\rightarrow(\exists x)Fx$	$(\exists x)(P\rightarrow Fx)$
(8)	$P\rightarrow(x)Fx$	$(x)(P\rightarrow Fx)$
(9)	$(\exists x)Fx\rightarrow P$	$(x)(Fx\rightarrow P)$
(10)	$(x)Fx\rightarrow P$	$(\exists x)(Fx\rightarrow P)$

(4) R,*Mathematical reasoning and automated theorem proving*,

Los artículos originales donde Church y Turing por aislado contestaban negativamente al *Entscheidungsproblem* fueron:
Alonzo Church, *"An unsolvable problem of elementary number theory"*, *American Journal of Mathematics*, 58 (1936), pp 345–363
Alonzo Church, *"A note on the Entscheidungsproblem"*, Journal of Symbolic Logic, 1 (1936), pp 40–41.

Alan Turing, *"On computable numbers, with an application to the Entscheidungsproblem"*, Proceedings of the London Mathematical Society, Series 2, 42 (1936), pp 230–265.

(5)Existe numerosas variaciones sobre la definición de una máquina simple de Turing, aquí sigo la encontrada en Wikipedia (http://en.wikipedia.org/wiki/Turing_machine). Sin embargo a grandes rasgos todas convergen en el poder de cómputo de la máquina descrita.

(6) Turing, Alan, *On Computable Numbers, with an application to the Entscheidungsproblem*, Proceedings London Mathematical Society (series 2) vol 42, 1936-7, pp.230-265.

(7) Teorema de la completud (Gödel, 1930) del CPPO: Si $\Gamma \models \varphi$ entonces $\Gamma \vdash \varphi$.

(en particular $\Gamma=\varnothing$, traduciendo si φ es válida (i.e. verdadera para toda posible interpretación, ej. los axiomas del CPPO) entonces existe una prueba de φ en el CPPO).

(8) Gôdel *aritmetiza* la sintaxis de un sistema formal, traduce las reglas de formación y deducción del sistema a la aritmética recursiva; cada relación, propiedad u operación sintáctica induce una relación, propiedad u operación entre números.
La aritmética recursiva es la parte de la teoría de los números que se construye con base a:

- Operaciones lógicas elementales como la disyunción, negación, conjunción, implicación y la cuantificación restringida a dominios finitos.
- La relación de igualdad y el principio de substitución de iguales por iguales
- El principio de inducción finita y métodos equivalentes
- Las definiciones recursivas

La teoría de la división, la del máximo común divisor y la de la descomposición en factores primos forman parte de la aritmética recursiva.

Gödel construye enunciados aritméticos que describen la estructura de sistemas formales, en particular elabora el enunciado bautizado en su honor (G(g)) afín a la paradoja del mentiroso cuyo significado metateórico sería algo así como "G(g) no es demostrable en la aritmética recursiva" . Dicho enunciado es verdadero, Gödel demuestra que no existe la prueba de G(g) en algunos sistemas capaces de representar a la aritmética recursiva, evidenciando los límites de tales sistemas.

Gracias al trabajo de Gödel se pude construir un enunciado cuyo significado metateórico aproximadamente sería " Si existe una prueba en el sistema de su consistencia entonces G(g) no es demostrable en la aritmética recursiva "; por lo cual si existe una prueba sobre la consistencia del sistema podríamos probar en el sistema vía modus ponens al enunciado de Gödel. CONTRADICCIÓN. De esta manera Gödel mostró una propiedad del sistema imposible de demostrarse dentro de él.

Fuente: Torres, Carlos. Teorema de Incompletud de Gödel, Tesis para obtener el título de maestro en ciencias, Facultad de Ciencias, UNAM. 1982

(9) El teorema de la incompletud de Gödel es aplicable a un gran número de sistemas formales basados en el CPPO, si cumplen entre otras cosas con:

- Las nociones de axioma y prueba del sistema son recursivas
- El sistema representa a la aritmética recursiva.
- La operación de sustituir todas las ocurrencias libres de una variable por un término es recursiva.

pág 63 de (8)

(10) pág. 60 de (8)

(11) *The most interesting lesson from these results is perhaps that even in a fairly rich domain, the theorems actually proved are mostly one which call on a very small portion of the available resources of the domain.*
Wang, Hao, *The mechanization of mathematical arguments*, Proc. Symp. in Applied Math., Vol. XV, 1963 págs. 31-40 , p. 32

(12) J. Herbrand, *Recherches sur la theorie de la demonstration*, PhD thesis, University of Paris, 1930.

(13) La versión débil del teorema de Herbrand trabaja exclusivamente con fórmulas en su forma prenex, tal como lo hizo Wang. El teorema se puede formular de la siguiente manera:

Let T be a theory axiomatized by purely universal formulas. Suppose that $T \vDash (\forall \boldsymbol{x})(\exists y_1, \ldots, y_k) B(\boldsymbol{x}, \boldsymbol{y})$ *with* $B(\boldsymbol{x}, \boldsymbol{y})$ *a quantifier-free formula. There there is a finite sequence of terms* $t_{i,j} = t_{i,j}(\boldsymbol{x})$, *with* $1 \leq i \leq r$ *and* $1 \leq j \leq k$ *so that*

$$T \vdash (\forall \boldsymbol{x}) \left(\bigvee_{i=1}^{r} B(\boldsymbol{x}, t_{i,1}, \ldots, t_{i,k}) \right).$$

(14) Paul Gilmore se incorporó a la plantilla de investigación de la IBM y en 1959 hizo un programa demostrador de teoremas en una IBM 704 para el cálculo de predicados.

Gilmore, P.C., *A Proof Method for Quantification Theory: Its Justification and Realization*, IBM Journal of Research and Development 4 , 1960, págs. 28-35

(15) Prawitz, Dag, *An Improved Proof Procedure*, Theoria 26, 1960. págs. 102-139

(16)

> Prawitz's procedure was a great improvement over what had been done previously because no spurious elements of the Herbrand universe were generated. Unfortunately, the huge expansions into disjunctive normal form that would be generated by all but the simplest problems made it clear that, at least as presented, this was still an unsatisfactory procedure. However, it contained the seminal idea of

Davis, Martin, *The Early History of Automated Deduction,* HANDBOOK OF AUTOMATED REASONING, Ed. Alan Robinson & Andrei Voronkov, Vol I, MIT Press, págs.3-15, pág.10

(17) Davis,Martin-Putnam, Hillary, *A computing procedure for quantification theory,* Journal of the ACM 7(3), 1960, págs. 201-215

(18) Davis, Logemann, Loveland. *A machine program for theorem proving,* Communications of the ACM 5, 1962, págs,394-397

(19)

> The idea was that a stack for formulas to be tested could be kept in external storage (in fact a tape drive) so that formulas in RAM never became too large.[2]

pág. 9 de (16)

(20) Para convertir cualquier fórmula a su FND o a su FNC en primer lugar se debe traducir a una forma donde solo aparezcan literales (fórmulas atómicas o su negación) y los conectivos de la disyunción y la conjunción. Para lograrlo se aplican las siguientes reglas de equivalencia:
Sean a,b,c fórmulas del CP

$a \Rightarrow b \equiv \neg a \vee b$

$a \Leftrightarrow b \equiv (a \Rightarrow b) \wedge (b \Rightarrow a)$

$\neg(a \vee b) \equiv \neg a \wedge \neg b$

$\neg(a \wedge b) \equiv \neg a \vee \neg b$

$\neg\neg a \equiv a$

Dependiendo de la conversión deseada se aplican las reglas de distributividad:

$a \wedge (b \vee c) \equiv (a \wedge b) \vee (a \wedge c)$
$a \vee (b \wedge c) \equiv (a \vee b) \wedge (a \vee c)$

Las reglas de distributividad generan un crecimiento exponencial del número de sumandos/factores. Corroborémoslo con el caso cuando convertimos de FNC a FND.

Queremos transformar $A \wedge B$ a su FND, donde $A=(a_1 \vee a_2 \vee \ldots a_n)$, $B=(b_1 \vee b_2 \vee \ldots b_m)$
Luego $A \wedge B = (a_1 \vee a_2 \vee \ldots a_n) \wedge (b_1 \vee b_2 \vee \ldots b_m)$

$\equiv (a_1 \wedge b_1) \vee (a_1 \wedge b_2) \vee \ldots \vee (a_1 \wedge b_m) \vee (a_2 \wedge b_1) \vee (a_2 \wedge b_2) \vee \ldots (a_n \wedge b_m)$

La tasa de crecimiento para la conversión de $A \wedge B$ es de orden cuadrático, se generan (m*n) sumandos. ¿Cuántos sumandos se generarán para $(a_1 \vee a_2 \vee \ldots a_n) \wedge (b_1 \vee b_2 \vee \ldots b_m) \wedge (c_1 \vee c_2 \vee \ldots c_l)$? ¿Y para $A \wedge B \wedge C \wedge ... \wedge \Omega$?
Como se puede apreciar, la tasa de crecimiento provocado por las reglas de distributividad es exponencial. Por lo tanto la complejidad de la transformación de las fórmulas a su FND/FNC por la vía descrita al menos es exponencial.

Sin embargo, el método de la variable conmutadora planteado por Tseitin mejora drásticamente la efectividad de la traducción conservativa de la satisfacibilidad de cualquier fórmula hacia una fórmula en FNC. Con la técnica de Tseitin la tasa de crecimiento de factores es cuadrática y ya no exponencial. Apreciemos en uso el truco de intercambio para el conflictivo caso de la conversión de una fórmula en FND a la FNC.

Queremos convertir $(b_1 \wedge b_2 \wedge b_3) \vee (c_1 \wedge c_2 \wedge c_3)$ a su FNC.
sea z la variable conmutadora:
$\neg z := (c_1 \wedge c_2 \wedge c_3)$
$z := (b_1 \wedge b_2 \wedge b_3)$
Luego como $A \wedge A \equiv A$, rescribimos la fórmula con la variable conmutadora:
$(b_1 \wedge b_2 \wedge b_3) \vee (c_1 \wedge c_2 \wedge c_3) \equiv (\neg z \vee (b_1 \wedge b_2 \wedge b_3)) \wedge (z \vee (c_1 \wedge c_2 \wedge c_3))$
Aplicando la regla distributiva:
$= (\neg z \vee b_1) \wedge (\neg z \vee b_2) \wedge (\neg z \vee b_3) \wedge (z \vee c_1) \wedge (z \vee c_2) \wedge (z \vee c_3)$
Nótese que la última fórmula es satisfacible sii la original lo es. Además, en lugar de los 9 sumandos producidos bajo la conversión normal sólo se generaron 6.
Cualquier fórmula en el peor de los casos (FND→FNC) sufre una explosión de los factores cuadrática, ya que cada una de las literales de dicha fórmula se combina sólo con las variables conmutadoras que la afectan.
El truco no funciona a la inversa (FNC→FND) pues la asignación del par de factores a la variable conmutadora no propaga la satisfacibilidad. La fórmula original previa a la traducción FND→FNC es satisfacible si z es verdadera o $\neg z$ es verdadera, ya que la variable conmutadora z representa al par de sumandos. En la fórmula traducida, si z es verdadera entonces todos los factores que la contengan serán verdaderos, pero como s.p.g. $z := (b_1 \wedge b_2 \wedge ... \wedge b_n)$ entonces cada b_i es verdadera, por lo cual el resto de los factores donde aparezca $\neg z$ también lo serán; siguiendo esta argumentación se concluye la doble implicación de la satisfaciblidad con respecto a las fórmulas original y traducida. No obstante la fórmula conseguida de seguir los mismos pasos en el caso FNC→FND es trivialmente satisfacible, mientras que la original a lo mejor ni siquiera lo es.

La investigación de Tseitin data aproximadamente de 1968, aunque fue publicada hasta 1970:
Tseitin, G., *On the Complexity of Derivation in Propositional Calculus*. In Studies in Constructive Mathematics and Mathematical Logic, Part 2, pp. 115-125, edited by A. O. Slisenko, 1970 (reprinted in Automation of Reasoning, vol. 2, edited by J. Siekmann and G. Wrightson, Springer- Verlag, New-York, 1983, pp. 466-483).

(21) Un millón de dólares es la recompensa ofrecida por el Instituto Clay a quien logre capturar la demostración de siete forajidos revoltosos llamados “los problemas de milenio”. Entre ellos está el facineroso ¿P=NP? Según sus nobles desplegados el Instituto busca fomentar el desarrollo matemático aunque parezca más una campaña de mercadotecnia http://www.claymath.org/millennium/

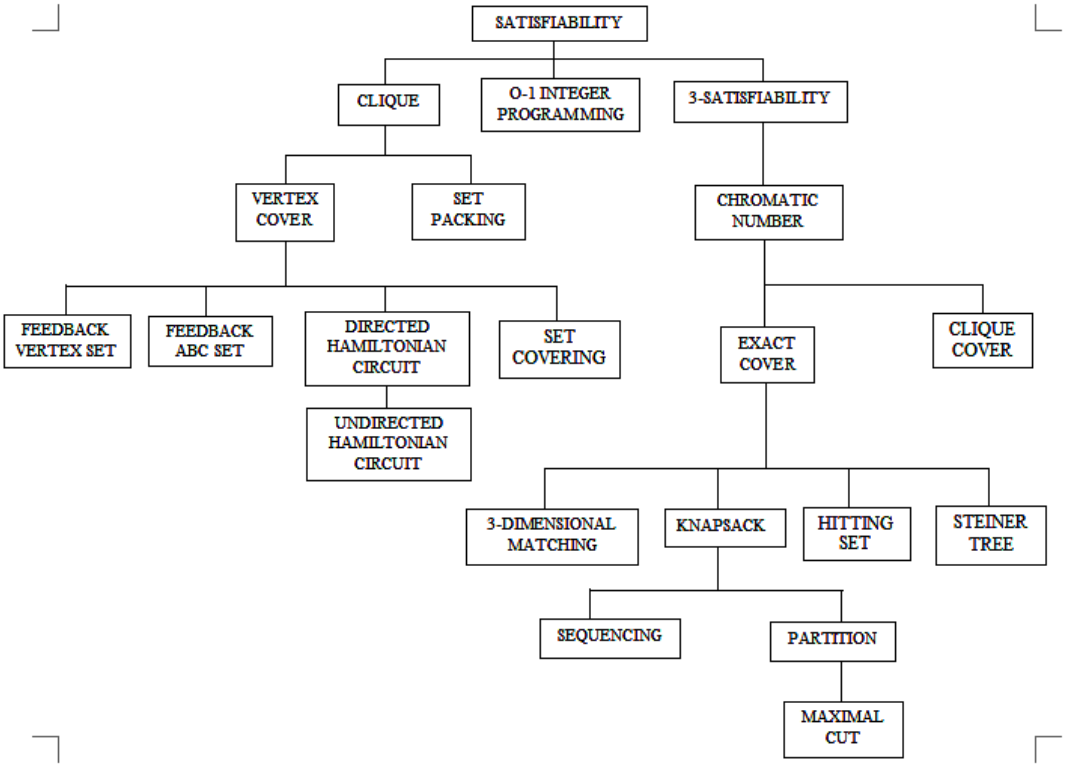
(22) S. Cook (1971) ,*The Complexity of Theorem Proving Procedures*, Proc. 3rd ACM Symp. on the Theory of Computing, ACM, 151-158

(23) La demostración puede ser consultada en el original de Cook si se tienen intereses históricos; si no se tienen esas metas copiosos son los libros donde aparece ya que el teorema de Cook es fundamental dentro del área de complejidad computacional. Algunas sugerencias:

M. Garey and D. Johnson (1979) ,*Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, New York

C. Papadimitriou and K. Steiglitz (1982) ,*Combinatorial Optimization; Algorithms and Complexity*, Prentice-Hall, New Jersey Una buena introducción (buena por su afable asistencia en la elaboración de esta parte del escrito) para los temas de complejidad aquí tratados se encuentra en la siguiente página mantenida por Kimon Spiliopoulos : http://users.forthnet.gr/ath/kimon/CC/CCC1b.htm

(24) Postal turística del zoológico de los NP-completos:



(25) *Princeton, 20 March 1956*
Dear Mr. von Neumann:

With the greatest sorrow I have learned of your illness. The news came to me as quite unexpected. Morgenstern already last summer told me of a bout of weakness you once had, but at that time he thought that this was not of any greater significance. As I hear, in the last months you have undergone a radical treatment and I am happy that this treatment was successful as desired, and that you are now doing better. I hope and wish for you that your condition will soon improve even more and that the newest medical discoveries, if possible, will lead to a complete recovery. Since you now, as I hear, are feeling stronger, I would like to allow myself to write you about a mathematical problem, of which your opinion would very much interest me: One can obviously easily construct a Turing machine, which for every formula F in first order predicate logic and every natural number n, allows one to decide if there is a proof of F of length n (length = number of symbols). Let $\psi(F,n)$ *be the number of steps the machine requires for this and let* $\phi(n) = \max_F \psi(F,n)$ *. The question is how fast* $\phi(n)$ *grows for an optimal machine. One can show that* $\phi(n) \geq k \cdot n$ *. If there really were a machine with* $\phi(n) \sim k \cdot n$ *(or even* $\sim k \cdot n^2$*), this would have consequences of the greatest importance [Tragweite]. Namely, it would obviously mean that in spite of the undecidability of the Entscheidungsproblem, the mental work of a mathematician concerning Yes-or-No questions could be completely replaced by a machine. After all, one would simply have to choose the natural number n so large that when the machine does not deliver a result, it makes no sense to think more about the problem. Now it seems to me, however, to be completely within the realm of possibility that* $\phi(n)$ *grows that slowly.*

La carta completa en inglés así como el original en alemán se encuentra en
http://www.complexitytheory.com/
John Von Newman padecía el mal del cangrejo y murió sólo dos años después del año de la carta. Para los necrófilos transcribo una descripción de su agonía:

... his mind, the amulet on which he had always been able to rely, was becoming less dependable. Then came complete psychological breakdown; panic, screams of uncontrollable terror every night. His friend Edward Teller said, "I think that von Neumann suffered more when his mind would no longer function, than I have ever seen any human being suffer." Von Neumann's sense of invulnerability, or simply the desire to live, was struggling with unalterable facts. He seemed to have a great fear of death until the last... No achievements and no amount of influence could save him now, as they always had in the past. Johnny von Neumann, who knew how to live so fully, did not know how to die.

S J Heims, *John von Neumann and Norbert Wiener: From mathematics to the technologies of life and death* ,Cambridge, MA, 1980.

(26)

We could let R be the relation between a formula F in first order logic, and a proof that the F is true. With P = NP, we could generate, not only a proof, but the **shortest** proof for F in time polynomial in the length of the proof. This means you could solve all mathematical problems in time polynomial in the length of the answer! This would of course leave all mathematicians without work, but P = NP has far wider consequences. It is, for example, easy to verify if a design for a cold fusion reactor works. Well, then use the same technique as before to generate a design for a cold fusion reactor. But why stop there? A society with access to this wonderful algorithm could produce optimal airplanes, bridges, buildings, rockets, etc. All of this sounds just incredible, and perhaps that is why a majority of theoreticians believe P just cannot be equal to NP.

Steven Rudich, *Notas sobre complejidad computacional,* www.complexitytheory.com, curso impartido en la Universidad Carnegie Mellon en el 2000, transcrito por Ke Yang, pág.30.

(27) Hasta encuestas han levantado al respecto, una más o menos reciente (2002) realizada por William Gasarch entre especialistas arrojó lo siguiente:
61 votos a favor de P≠NP, 9 para P=NP, 4 votos para "es independiente" (en un sistema axiomático Σ [si no se especificó por default ZFC] la fórmula ψ [en este caso P=NP] es independiente si existe un modelo tanto para su afirmación como para su negación], 22 se abstuvieron de opinar, 1 votó por "depende el modelo" y 3 sólo afirmaron que la igualdad/desigualdad "no es independiente de la aritmética recursiva" (w.t.m).
En la encuesta hay varios resultados curiosos y opiniones valiosas (vacas sagradas incluidas), la pueden consultar en http://www.win.tue.nl/~gwoegi/P-versus-NP.htm.

(28) Steven Skiena, *Analysis of Algorithms: Lecture Notes* , Department of Computer Science ,SUNY Stony Brook, Lecture 19
http://www2.toki.or.id/book/AlgDesignManual/LEC/LECTURES/ALL.HTM

(29) 2^{1000} = 10,715,086,071,862,673,209,484,250,490,600,018,105,614,048,117,
055,336,074,437,503,883,703,510,511,249,361,224,931,983,788,
156,958,581,275,946,729,175,531,468,251,871,452,856,923,140,
435,984,577,574,698,574,803,934,567,774,824,230,985,421,074,
605,062,371,141,877,954,182,153,046,474,983,581,941,267,398,
767,559,165,543,946,077,062,914,571,196,477,686,542,167,660,
429,831,652,624,386,837,205,668,069,376
(302 digits)

El estimado total de partículas en el universo fluctúa de 10^{72} a 10^{87}. En 1931 Eddington lo situó en 10^{80}.

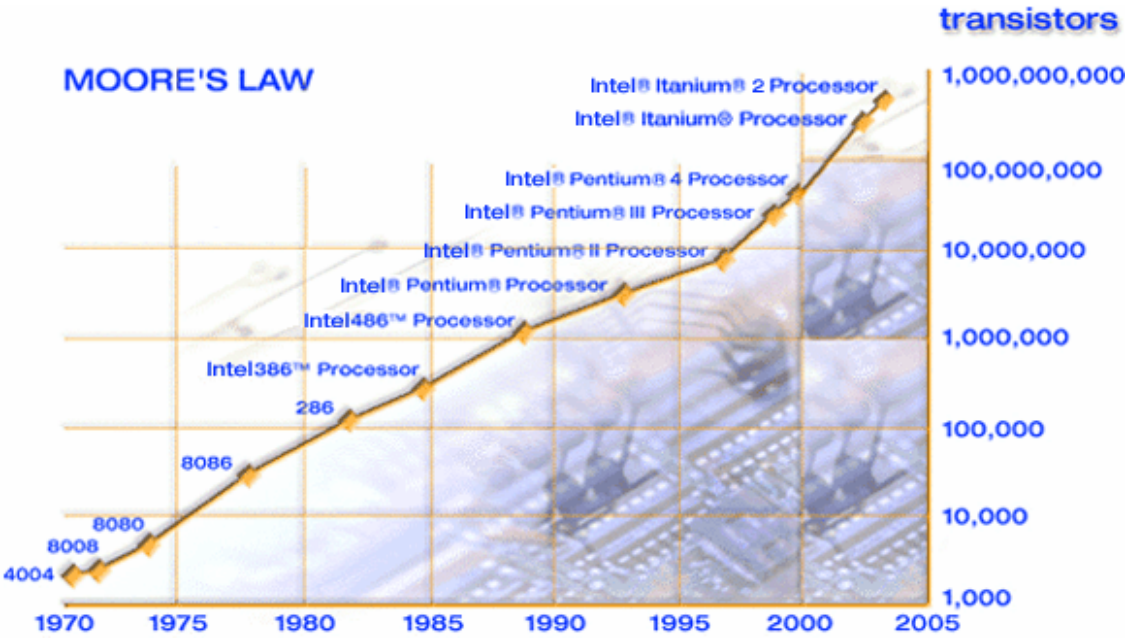
(30)

Martin Davis and Hilary Putnam noted that Gilmore's program failed on some rather simple examples because of its reliance on expansions into disjunctive normal form for satisfiability testing. This led them to the optimistic (and in retrospect rather naive) conclusion that the lack of effective methods for testing large formulas of the propositional calculus for satisfiability was the main obstacle to be surmounted. Although their interest in algorithms for what came to be known as

pág. 8 de (16)

(31) Respetando a la casa-cuna del autor de la ley, importo la información de ésta de la página de Intel (http://www.intel.com/research/silicon/mooreslaw.htm)

Moore observed an exponential growth in the number of transistors per integrated circuit and predicted that this trend would continue. Through Intel's relentless technology advances, Moore's Law, the doubling of transistors every couple of years, has been maintained, and still holds true today.



(32) El 13 de abril del presente año, el mismísimo Gordon Moore declaró en una entrevista que su ley no va a continuar siendo válida por mucho tiempo, ya que la miniaturización de los transistores va a alcanzar el límite atómico:

In terms of size [of transistor] you can see that we're approaching the size of atoms which is a fundamental barrier, but it'll be two or three generations before we get that far—but that's as far out as we've ever been able to see. We have another 10 to 20 years before we reach a fundamental limit. By then they'll be able to make bigger chips and have transistor budgets in the billions.

De Techworld, the UK's Infrastructure and network knowledge centre (http://www.techworld.com/opsys/news/index.cfm?NewsID=3477)
En la referencia (30) también se pronostica la muerte de la ley de Moore :
Intel expects that it will continue at least through the end of this decade.

(33) Hirsch Edward A., Kojevnikov Arist, *UnitWalk: a new SAT solver that uses local search guided by unit clause elimination,*
gauss.ececs.uc.edu/Conferences/ SAT2002/Abstracts/hirsch.ps

(34) *zChaff is designed with performance and capacity in mind. We have success stories of using zChaff to solve problems with more than one million variables and 10 million clauses. (Of course, it can't solve every such problem!).*
http://www.princeton.edu/~chaff/zchaff.html

(35) Wos , Larry, *Reflections on Automated Reasoning at Argonne: A personalized fragment of history,* en el CD-ROM incluído en (37 Wos), pág. 3

One man, William F. Miller, planted a single seed of research. From that single seed blossomed numerous and glorious flowers: the introduction of strategy, the formulation of a powerful rule fo quality-oriented reasoning, the design of the world's most effective reasoning program, the answer to a question that was open for sixty years, and far, far more. All of these topics will be detailed here. Whether serendipity, coincidence, foresight, or Miller's specific plan, I have not yet ascertained. History may never reveal the precise nature of the beginning. What I can say with certainty is that (most likely) the flower would have died on the vine were it not for Argonne, that many or perhaps almost all of the key discoveries on which automated reasoning now rests would have eluded research.

(36) Robinson, J. A., *A machine oriented logic based on the resolution principle,*
JACM 12, 23-41, 1965.

(37) Wos, L., and Pieper, G., *A Fascinating Country in the World of Computing,*
World Scientific Press, Singapore (1999).

Miranda, Favio, *FUNDAMENTOS LÓGICOS DEL PROGRAMA DE RAZONAMIENTO AUTOMÁTICO OTTER,* tesis para obtener el grado académico de maestro en ciencias, 1999, UNAM http://abulafia.fciencias.unam.mx/~favio

(38) *It is natural to question whether the given actions of restricting the logical connectives, of rewriting statements into a normal form, and of replacing existential quantifiers with Skolem functions preserve the logical meaning of a statement. They do not. However, as will be seen, the actions preserve enough of the logical properties, most important of which are the properties of satisfability and unsatisfability... The precise formulation of what occurs is that the resulting set of clauses is not necessarily logically equivalent to the original statementor formula. The loss of logical equivalence is due to the use of Skolem functions. The essence of the situation is, however, that, if the original statement or formula in first-order predicate calculus is unsatisfiable (inconsistent or self-contradictory),then the resulting set of clauses is also, and conversely.*
pág. 269 de (37 Wos)

(39) Paterson, M.S., and Wegman, M.N. *Linear Unification*, J. Comput. Syst. Sci. 16 (1978) pp 158-167

(40) A. Haken, *The intractability of resolution,* Theoretical Computer Science, 39 (1985), pp. 297-308.

(41) S. A. Cook and R. A. Reckhow, *The relative eficiency of propositional proof systems,* Journal of Symbolic Logic, 44 (1979), pp. 36{50.

(42) Samuel R. Buss and György Turán, *Resolution proofs of generalized pigeonhole principles.* Theoretical Computer Science 62 (1988) 311-317.

(43) *Briefly, that paradigm consists of finding a theorem whose proof is beyond the reach of the current program, devising a strategy, inference rule, or mechanism to bring the proof within range, and then testing the proposed advance. The test first focuses on closely related theorems, then on distantly related theorems, then on unrelated theorems.*
In other words, the proposed advance must offer generality, must not be tuned to the theorem that initiated the study
págs. 2-3 de (35)

(44) Tseitin, G., *On the Complexity of Derivation in Propositional Calculus*. In Studies in Constructive Mathematics and Mathematical Logic, Part 2, pp. 115-125, edited by A. O. Slisenko, 1970 (reprinted in Automation of Reasoning, vol. 2, edited by J. Siekmann and G. Wrightson, Springer- Verlag, New-York, 1983, pp. 466-483).

(45) *An* extended resolution proof *for a theory* T *is one where* T *is first augmented by a collection of groups of axioms, each group of the form*

$$\neg x \vee \neg y \vee w$$

$$x \vee \neg w$$

$$y \vee \neg w$$

where x *and* y *are literals in the (possibly already extended) theory* T *and* w *is a new variable. Following this, derivation proceeds using conventional resolution on the augmented theory.*

As we have stressed, however, the fact that a proof system is strong does not mean that it works well in practice. We know of no *implementation of extended resolution for the simple reason that virtually nothing is known about how to select new variables so as to shorten proof length.*

Understanding why *the introduction of these new variables can reduce proof length is considerably simpler. As an example, suppose that during a resolution proof, we have managed to derive the nogood* $a \vee x$*, and that we have also derived* $a \vee y$*. In order to complete the proof, we need to perform lengthy - but identical - analyses of each of these nogoods, eventually deriving simply* x *from the first and* y *from the second (and then resolving against* $\neg x \vee \neg y$*, for example).*

If we could replace the pair of nogoods $a \vee x$ *and* $a \vee y$ *with the single nogood* $a \vee w$ *using the augmented group of axioms the two proofs from* $a \vee x$ *and* $a \vee y$ *could be collapsed into a single proof, potentially halving the size of the proof in its entirety. Introducing still more variables can repeat the effect, resulting in exponential reductions in proof size.*

Another way to look at this is as an improvement in expressivity. There is simply no way to write $(a \vee x) \wedge (a \vee y)$ or the equivalent $a \vee (x \wedge y)$ as a single Boolean axiom in disjunctive normal form. The power of extended resolution rests on the fact that subexpression substitution makes it possible to capture expressions such as $a \vee (x \wedge y)$ in a single axiom.

NOTAS: si tenemos las cláusulas unitarias x. , y. por convención tenemos su conjunción, por lo cual aplicando dos veces la resolución binaria con x. , y. , –x|-y|w. obtenemos a nuestra nueva variable w asociada con x∧y. Si tenemos -x, entonces debemos trabajar con -w (pues w ≡x∧ y) por lo cual se requiere x|-w. en los axiomas extendidos para obtener el resolvente –w.; de igual manera se puede justificar la inserción y|-w en los axiomas extendidos.

Nogoods: escala en el camino hacia la determinación de la insatisfacibilidad de un conjunto de cláusulas, i.e. cláusula a partir de la cual se debe seguir aplicando resolución hasta llegar a $\{\varnothing\}$ o a un conflicto unitario.

Información trascrita de Dixon, Ginsberg, Parks, *Generalizing Boolean Satisfability: Backgrounds*, University of Oregon
http://www-2.cs.cmu.edu/afs/cs/project/jair/pub/volume21/dixon04a-html/node9.html

(46)
Specifically, in 1962, during Alan's second summer visit to Argonne, George (in effect) presented unit resolution to him, the rule that restricts binary resolution to requiring that one of the two parents be a unit clause, a statement consisting of one literal . My years of research conducted jointly with one or more other colleagues provides me with a general understanding of how George's contribution was lost, even to Alan. (My main source for George's role in the history of binary resolution was George himself, in many, many conversations; the continuity of those conversations still persuades me of their accuracy. Also, my deep knowledge of George as a person gathered over many, many years convinces me that he did not exaggerate his contribution.)
pág. 7 de (35)

(47) *A strategy is a rule or set of rules that governs the use of inference rules. If a strategy constrains application of an inference rule, then it is a restriction strategy. If a strategy dictates where next to apply an inference rule, the strategy is a direction strategy. Both types of strategy are necessary for an automated reasoning program to be efective.*
pág. 285 de (37 Wos)

(48) Robinson, J. A., 1965, *Automatic Deduction with Hyperresolution*, Internat. J. Comput. Math., *Vol. 1, pp. 227-234.*

(49) *Traditionally, a single step in a deduction has been required, for pragmatic and psychological reasons, to be simple enough, broadly speaking, to be apprehended as correct by a human being in a single intellectual act. No doubt this custom originated in the desire that each single step of a deduction should be indubitable, even though the deduction as a whole may consist of a long chain of such steps...[by comparison, the resolution rule] is machine-oriented because in order to obtain the much greater deductive power than has hitherto been the norm, it requires much more computational effort to apply it than traditional human-oriented rules typically requires.*

Robinson, Alan , *Computational Logic: Memories of the Past and Challenges for the Future,* J. Lloyd et al. (Eds.): CL 2000, LNAI 1861, pp. 1-24, 2000.
 pág.10

(50)
...our approach to the automation of reasoning differs sharply form approaches one often finds in artificial intelligence; indeed, our paradigm relies on types of reasoning and other procedures that are not easily or naturally applied by a person.

Wos, Larry , *Automated reasoning Answers Open Questions,* Notices of the American Mathematical Society 40. 1993,15-26, pág.16

(51) L. Wos, G. Robinson, and D. Carson. *Efficiency and completeness of the set of support strategy in theorem proving.* J. ACM, 12(4):536-541, 1965.

(52) págs. 288-289 de (37 Wos), págs. 42-44 de (37 Miranda)

(53) *I have always asserted that our field paid far too much attention to properties such as refutation completeness. Such properties are indeed aesthetically pleasing, but, in the main, their presence sheds little or no light on effectiveness. Were some mythical individual to ask me to explain how a program can succeed and succeed often in the face of the absence of refutation completeness, I would suggest that the answer rests with the incredibly numerous and distinct proofs that can be found or almost any given theorem. I know that, when I discovered this fecundity, my background in mathematics had not prepared me for it; further, I suspect many mathematicians might be surprised at such fecundity.*
pág. 10 de (35)

(54) Larry Wos, George A. Robinson, Daniel F. Carson, Leon Shalla: *The Concept of Demodulation in Theorem Proving,* J. ACM 14(4): 698-709 (1967)

(55) Robinson, J., ''*A review of automatic theorem-proving*'', pp. 1-18 in Proceedings of the Annual Symposia in Applied Mathematics, Vol. 19, American Mathematical Society, Providence, Rhode Island (1967).

(56) Wos, L., and Robinson, G., ''*Paramodulation and Set of Support*'', in Lecture Notes in Mathematics, Springer-Verlag, New York, pp. 276-310. (1968)

Robinson, G., and Wos, L., ''*Paramodulation and theorem-proving in first-order theories with equality*'', pp. 135-150 in Machine Intelligence 4, ed. B. Meltzer and D. Michie, Edinburgh University Press, Edinburgh (1969).

(57)
9) se unifica f(f(x,y),z) desde 5) con f(w,w) hacia 7) –deben renombrarse las variables de las cláusulas para que no comportan variables- MGU: w=f(x,y), w=z
10) se unifica f(w,w) desde 7) con f(x,y) hacia 5), MGU: w=x , x=y ; además se utiliza la simetría de la igualdad pues la paramodulación generaría la cláusula EQUAL(f(e,z), f(x,f(x,z))).,en la demodulación se unifica f(e,x) de 2) con f(e,z)

11)se unifica f(x,f(y,f(x,y))) desde 9) con f(w,u) hacia 10), MGU: w:=x u:=f(y,f(x,y)), sustituyo en 10) e intercambio los argumentos de EQUAL y en la demodulación se unifica f(x,e) con 1) f(x,e)
12) Para aumentar la confusión ahora no renombro las variables, se unifica f(y,f(x,y)) desde 11) con f(x,z) hacia 10); MGU x:=y, z:=f(x,y), sustituyo en 10) y de nuevo intercambio los argumentos de la literal EQUAL (simetría)

(58) La base axiomática propuesta por A. Grau es la siguiente:

1) $f(f(v,w,x),y,f(v,w,z)) = f(v,w,f(x,y,z))$
2) $f(y,x,x) = x$
3) $f(x,y,g(y)) = x$
4) $f(x,x,y) = x$
5) $f(g(y),y,x) = x$

donde f representa el producto y g el inverso.

La fuente principal de la enumeración histórica de los programas de Argonne es :
Lusk, Ewing, *Controlling Redundancy in Large Search Spaces: Argonne-Style Theorem Proving Through the Years*

(59) Los nuevos resultados abarcan las áreas de la geometría algebraica, la lógica ecuacional y el álgebra moderna .

McCune, W., and Padmanabhan, R., *Automated Deduction in Equational Logic and Cubic Curves, Lecture Notes in Computer Science, Vol. 1095*, Springer-Verlag, Heidelberg (1996), http://www.mcs.anl.gov/home/mccune/ar/monograph/ .

(60) La separación (modus ponens) y la sustitución (instanciación) eran las reglas de inferencia de EC cuando en los sesentas Meredith las fusionó en la separación condensada, concediéndole una mejor presentación al cálculo y de paso le abrió las puertas a los programas de razonamiento deductivo automatizado pues la instanciación es una regla de inferencia fuera de su rango y hasta el momento se desconoce un control eficaz para su aplicación.
Meredith C.A, Prior, *Notes on the axiomatics of the propositional calculus*, Notre Dame J. Formal Logic 4, no,3 (1963), 171-187

(61)Lukasiewicz, J. , *Selected Works*, North Holland Amsterdam 1970.

(62) Jeremy George Peterson, *Shortest single axioms for the classical equivalential calculus.* Notre Dame J. Formal Logic 17, no. 2 (1976), 267–271.

(63) *Such complexity might have been beyond that which any unaided researcher would find pleasant or, perhaps, possible. Perhaps a decade or so after Winker's triumphs, my*

research (in which OTTER played a vital role) in the area culminated in finding a 23-step proof establishing XHK to be a single axiom and a 19-step proof for XHN. Thus, again (both in the early 1980s and in the early 1990s) the value and power of an automated reasoning program were demonstrated.
págs. 220-221 de (37 Wos)

(64) L. Henschen, B. Smith, R. Veroff, S. Winker and L. Wos, *Questions concerning possible shortest single axioms for the equivalential calculus: an application of automated theorem proving to infinite domains.* Notre Dame J. Formal Logic 24, no. 2 (1983), 205–223.

(65)Caro adelanto a la vista de aquellos quienes se toman la molestia de revisar las notas, aquí y en la próxima nota se transcriben las pruebas de Otter, en este apartado sobre la deducción de la reflexividad a partir de XCB:

```
----> UNIT CONFLICT at  77.13 sec ----> 8514 [binary,8513.1,3.1]
$ANSWER(reflex).
Length of proof is 10.  Level of proof is 8.
---------------- PROOF ----------------
1 [] -P(e(x,y))| -P(x)|P(y).
2 [] P(e(x,e(e(e(x,y),e(z,y)),z))).
3 [] -P(e(a,a))|$ANSWER(reflex).
10 [hyper,2,1,2] P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v)).
11 [hyper,10,1,2]
P(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v),w),e(v6,w)),
v6)).
12 [hyper,10,1,2]
P(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),v),e(u,v))).
14 [hyper,11,1,2]
P(e(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(v,u)),v),w),v6),e(w,v
6))).
19 [hyper,12,1,2]
P(e(x,e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),x),v),e(w,v)),w))).
61 [hyper,19,1,14]
P(e(e(e(e(e(x,e(e(e(x,y),e(z,y)),z)),e(e(e(e(e(e(u,e(e(e(u,v),e(w
,v)),w)),v6),e(v7,v6)),v7),v8),v9),e(v8,v9))),v10),e(v11,v10)),v11)
).
220 [hyper,61,1,14]
P(e(e(x,e(e(e(e(e(e(y,e(e(e(y,z),e(u,z)),u)),v),e(w,v)),w),e(e(e(v6
,e(e(e(v6,v7),e(v8,v7)),v8)),v9),e(v10,v9))),v10)),x)).
849 [hyper,220,1,10]
P(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),e(e(e(e(v,e(e(e(v,w),e(v6,w)),v
6)),v7),v7),u))).
2722 [hyper,849,1,10] P(e(e(e(x,e(e(e(x,y),e(z,y)),z)),u),u)).
8513 [hyper,2722,1,12] P(e(x,x)).
8514 [binary,8513.1,3.1] $ANSWER(reflex).
```

(66)The theorem states that the axioms of symmetry and associativity completely axiomatize equivalential calculus; in particular, their use leads to the deduction of

```
      e(e(e(x,y),e(z,x)),e(y,z)),
```

```
which, with symmetry, forms a complete axiomatization of the calculus.
```

The proof presented here (obtained June 21, 1995) has length **6.**

Reference: For a detailed description of this theorem, see L. Wos, ``Meeting the Challenge of Fifty Years of Logic," *Journal of Automated Reasoning,* **6** (1990) 213-232.

```
----> UNIT CONFLICT at   1.61 sec ----> 297 [binary,296.1,4.1] $F.
Length of proof is 6.  Level of proof is 4.
---------------- PROOF ----------------

1 [] -P(e(x,y))| -P(x)|P(y).
2 [] P(e(e(x,y),e(y,x))).
3 [] P(e(e(e(x,y),z),e(x,e(y,z)))).
4 [] -P(e(e(e(a,b),e(c,a)),e(b,c))).
5 [hyper,1,3,3] P(e(e(x,y),e(z,e(x,e(y,z))))).
6 [hyper,1,2,3] P(e(e(x,e(y,z)),e(e(x,y),z))).
8 [hyper,1,5,5] P(e(x,e(e(y,z),e(e(u,e(y,e(z,u))),x)))).
14 [hyper,1,5,6] P(e(x,e(e(y,e(z,u)),e(e(e(y,z),u),x)))).
30 [hyper,1,2,8] P(e(e(e(x,y),e(e(z,e(x,e(y,z))),u)),u)).
296 [hyper,1,30,14] P(e(e(e(x,y),e(z,x)),e(y,z))).
297 [binary,296.1,4.1] $F.
```

(67) Wos, Fitelson, Ulrich, *Vanquishing the XCB Question: The Methodological Discovery of the Last Shortest Single Axiom for the Equivalential Calculus*, Journal of Automated Reasoning **29** (2): 107-124,2002

(68) E. V. Huntington, *New sets of independent postulates for the algebra of logic*, Trans. AMS 35, 274--304 (1933).

(69) S. Winker, `*Robbins Algebra: Conditions That Make a Near-Boolean Algebra Boolean*, J. Automated Reasoning 6(4), 465--489 (1990).

(70) W. McCune, *Solution of the Robbins Problem*, JAR 19(3), 263--276 (1997)

(71) *Proofs found by programs are always questionable. Our approach to this problem is to have the theorem prover construct a detailed proof object and have a very simple program (written in a high-level language) check that the proof object is correct. The proof checking program is simple enough that it can be scrutinized by humans, and formal verification is probably feasible.*

EQP is not yet able to construct proof objects, so the EQP proof was used to guide Otter (using AC axioms instead of AC unification) to a proof of the same theorem. Otter produced a proof object, which was then checked by the proof checker,
http://www-unix.mcs.anl.gov/~mccune/papers/robbins/

(72) Localizado en http://www-unix.mcs.anl.gov/~mccune/papers/robbins/otter-theorem.object.txt

(73) Rüdiger Thiele ,*HILBERT'S TWENTY-FOURTH PROBLEM,* THE AMERICAN MATHEMATICAL MONTHLY, enero 2003, www.maa.org/news/Thiele.pdf

The 24th problem in my Paris lecture was to be: Criteria of simplicity, or proof of the greatest simplicity of certain proofs. Develop a theory of the method of proof in mathematics in general. Under a given set of conditions there can be but one simplest proof.
pág. 2, traducción del alemán al inglés del historiador Thiele

(74) *The mathematical rigor that is essential in the treatment of a problem does not require complicated demonstrations; it requires only that the result be obtained by a finite number of logical steps from a finite number of hypotheses furnished by the problem itself; in seeking this rigor we may find simplicity.*

pág. 6 del intermediario (73), cuya fuente es la entrevista de Hilbert con Scott:
C. A. Scott, *The International Congress of Mathematicians in Paris*, Bull. Amer. Math. Soc. **7** (1900) 57–79, págs. 67-68.

(75) *A 6-basis for modular ortholattices (MOL) in terms of join (v), meet (^), and complement (c).*

```
x v (y v z) = y v (x v z).  % AJ
x v (x ^ y) = x.            % B1
x ^ y = c(c(x) v c(y)).     % DM
c(c(x)) = x.                % CC
x v c(x) = y v c(y).        % ONE
x v (y ^ (x v z)) = x v (z ^ (x v y)).  % MOD
```

A 4-basis for modular ortholattices in terms of the Sheffer stroke .

```
f(x,f(f(y,z),f(y,z))) = f(y,f(f(x,z),f(x,z))).  % A_SS
f(f(x,x),f(x,y)) = x.                           % B_SS
f(x,f(x,x)) = f(y,f(y,y)).                      % ONE_SS
f(x,f(y,f(x,f(z,z)))) = f(x,f(z,f(x,f(y,y)))).  % MOD_SS
```

Nota: *'Sheffer stroke"* `(x | y)` equivale a una NAND ($\neg(x \wedge y)$) y en Otter se representa así f(x,y).

Here http://www.cs.unm.edu/~veroff/LT/c6_mod.pf is *a proof that a length 25 Sheffer equation named C6 (MOL-25A-785) implies modularity. This is a key step in proving that C6 is a short single axiom for modular ortholattices. This proof is 356 steps long! We found it using our method of proof sketches.*

Nota: *"proof sketches"* es una estrategia donde el usuario le indica al programa un conjunto de cláusulas que marcan rutas deductivas por donde posiblemente se localice la prueba.

Veroff, Robert , http://www.cs.unm.edu/~veroff/LT/mol.html

(76) *To the best of our knowledge, presentations of the proof that Robbins algebras are Boolean have remained in the form of sequences of proofs, eventually leading to a proof of Huntington's equation. Here http://www.cs.unm.edu/~veroff/ROBBINS/r2h.pf is a single Otter derivation of Huntington's axiom. Here* http://www.cs.unm.edu/~veroff/ROBBINS/r2h.in *is an Otter input file that can be used to check the derivation.* Veroff, Robert, http://www.cs.unm.edu/~veroff/ROBBINS/

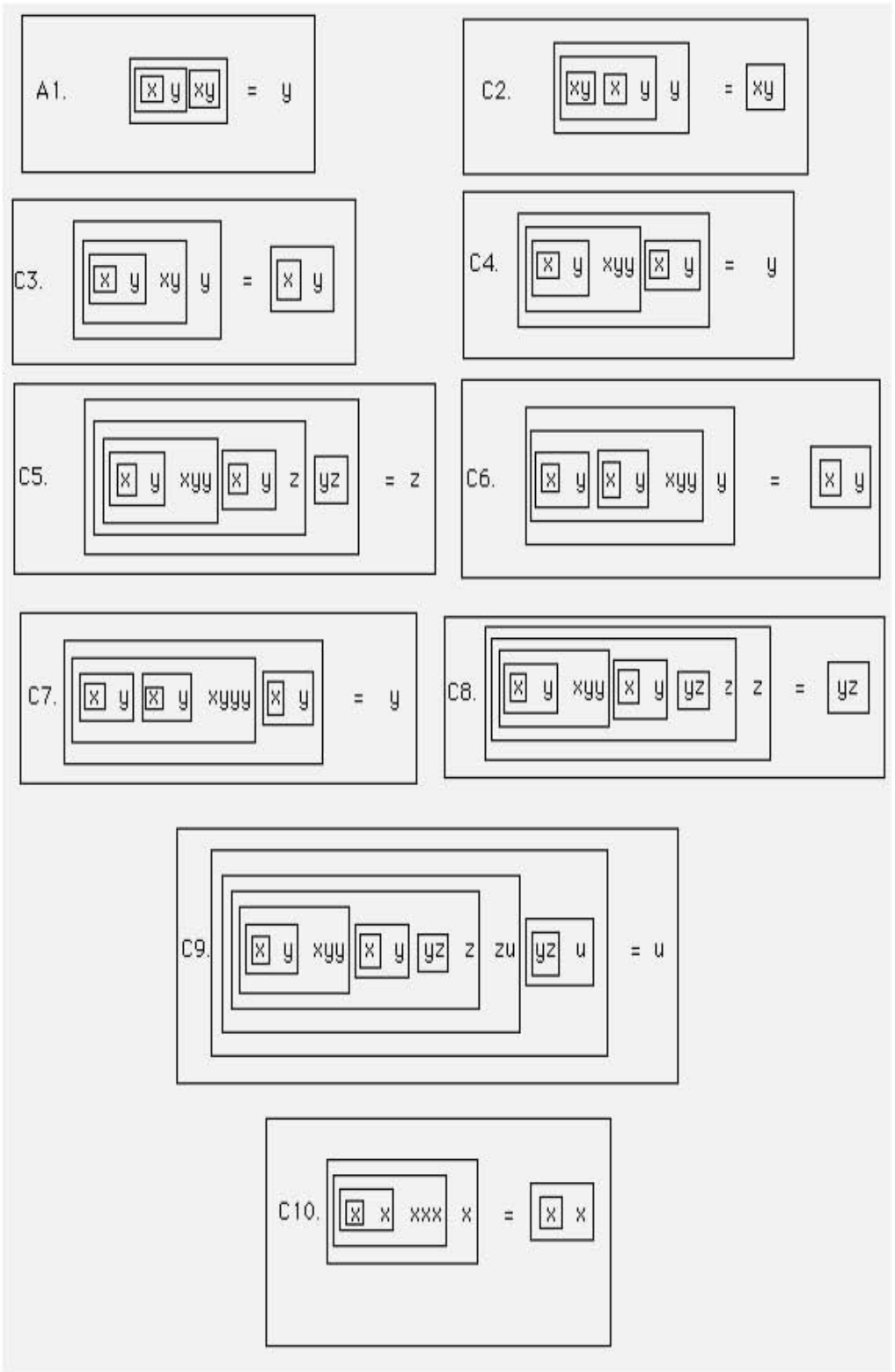
(77) *Although the EQP proof of the Robbins conjecture (which has now been verified both automatically and by hand by several different researchers) firmly establishes the truth of the Robbins conjecture, the EQP proof object does little to help human beings understand how to prove the Robbins conjecture. In fact, the EQP proof object is quite difficult to follow (or even parse!). After several frustrating and fruitless weeks of trying to work through the EQP proof object by hand, I started experimenting with Mathematica. Just a couple of days later, I had a complete Mathematica reconstruction of the proof!*

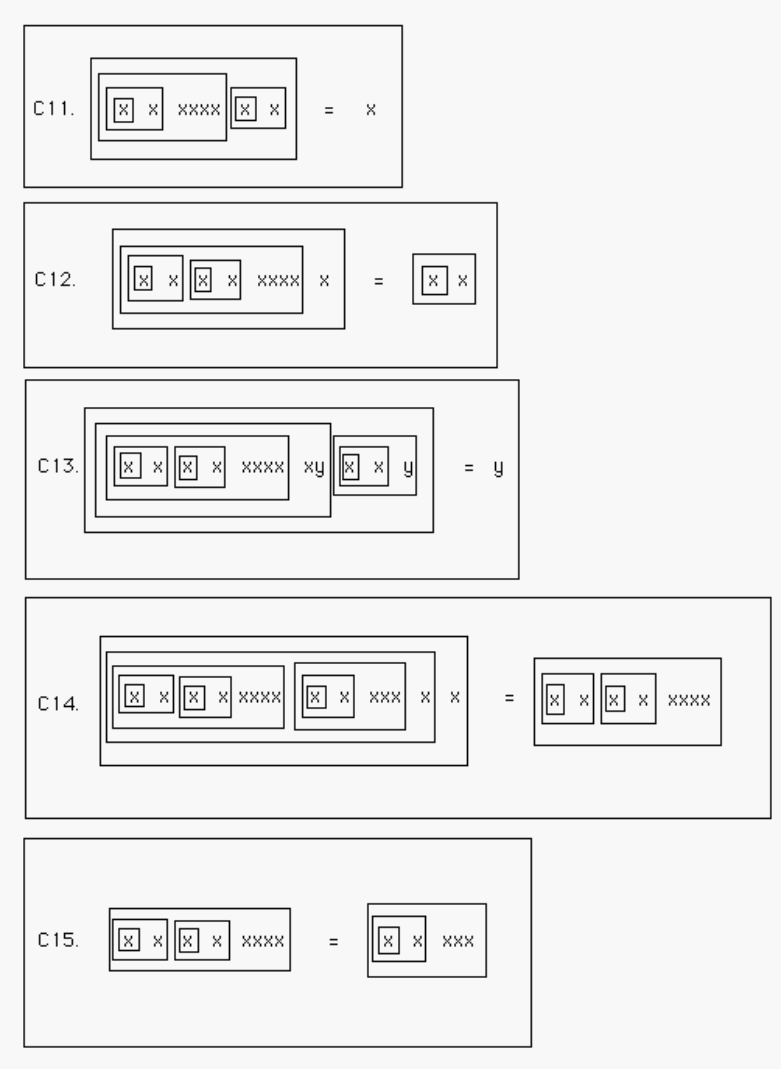
Fitelson, Branden, *Using Mathematica to Understand the Computer proof of the Robbins Conjecture,* Mathematica in Education and Research, Vol *7,* 1998 , págs. 17-26

(78)
Computers will prove theorems with proofs that are very long and complex. These proofs will need translations into language that human beings can understand. Here is the beginning of a new field of mathematics in the interface between logic, communication and the power to reason and understand. The machines will help us move forward into new and powerful ways to explore mathematical terrain.

Kauffman, *The Robbins Problem: computer proofs and human proofs,* Kybernetes - The International Journal of Systems and Cybernetics, Gordon Pask Remembered and Celebrated: Part I, edited by Bernard Scott and Ranulph Glanville, Volume 30, Number 5/6 (2001) http://www2.math.uic.edu/~kauffman/Robbins.htm

(79) Cabe señalar que la notación de caja es obra de G. Spencer-Brown y del famoso filósofo Charles Sanders Peirce. He aquí la demostración completa:


A1. x y xy = y
C2. xy x y y = xy
C3. x y xy y = x y
C4. x y xyy x y = y
C5. x y xyy x y z yz = z
C6. x y x y xyy y = x y
C7. x y x y xyyy x y = y
C8. x y xyy x y yz z z = yz
C9. x y xyy x y yz z zu yz u = u
C10. x x xxx x = x x



(80) *EQP produced a proof that could have been inaccessible to human beings but is in fact understandable to us with a little effort in communication...*
*I understood EQP's proof with an enjoyment that was very much the same as the enjoyment that I get from a proof produced by a human being. Understanding a proof is like listening to a piece of music - or like improvising a piece of music from an incomplete score.*de (78)

(81) Ejemplo de estas substituciones:

$A := x + y$

$B := \mathrm{N}(x + y)$

$C := \mathrm{N}x + y$

$D := \mathrm{N}(\mathrm{N}x + y)$ ...

Burris, Stan, *An anthropomorphised version of the McCune's machine proof thatRobbins algebras are Boolean*, preprint available in
http://www-unix.mcs.anl.gov/~mccune/papers/robbins/.

(82) Por ejemplo:

STEP 8871

With (8870), let y be $2x$:

$$n(n(n(n(3x)+x)+n(3x)+2x)+n(3x)) = 2x.$$

Use the preceding equation to simplify (8865):

$$n(n(3x)+x)+2x = 2x. \qquad [8865:8870]\ (8871)$$

Q.E.D.

donde 8870 $n(n(n(n(3x)+x)+n(3x)+y)+n(x+y)) = y.$

págs. 11-12 de (70)

(83) Demuestra entre otras cosas que toda álgebra Booleana es un álgebra de Robbins, también demuestra los lemas de Winker (toda algebra de Robbins más las condiciones de Winker forman un álgebra Booleana).

Mann, Allen, *A Case Study in Automated Theorem Proving: Otter and EQP*
Tésis de Maestría, Universidad de Colorado, 2003
http://ucsu.colorado.edu/~almann/MA/robbins_complete.pdf

(84) Derrick Coetzee, *po2hr - Otter Proof Object to LaTeX converter*
http://moonflare.com/code/po2hr.php

(85) Andrew John Wiles (Cambridge, Inglaterra, 11 de abril de 1953 -) alcanzó la fama mundial en 1995 por demostrar el último teorema de Fermat. En realidad Wiles demostró que toda curva elíptica semiestable definida sobre los racionales es una forma modular. Ken Ribet años antes demostró que de lo anterior (versión de la conjetura Shimura-Taniyama) se deriva el último teorema de Fermat. A la primera demostración de Wiles de 1994 le pizcaron errores, tuvo que pasar un año y contar con la ayuda de su exalumno Richard Taylor para corregirla y publicarla, sobra decir que la cuota para entenderla es muy alta pues además de su larga longitud se combinan conocimientos matemáticos de diversas áreas, como la topología y la teoría de números.

Wiles, Andrew. *Modular elliptic curves and Fermat's last theorem*. Ann. of Math. (2) 141 (1995), no. 3, 443—551 con el anexo:

Wiles , Taylor , *Ring-theoretic properties of certain Hecke algebr*as, (Annals of Mathematics 141 (1995), p. 553-572)

El libro de divulgación *Fermat's Last Theorem* de Simon Singh es un relato ameno de todas las peripecias y viscicitudes para dar con la demostración.

(86) Sabbagh, *The Strange Case of Louis de Branges,* LRB.Vol 26, 2004
La historia novelada escrita por Karl Sabbagh puede ser consultada en
http://www.lrb.co.uk/v26/n14/sabb01_.html

Si se quiere otear a la supuesta demostración de la conjetura de Riemann de Louis de Branges "publicada" en la red en el 2004 visitar:
www.math.purdue.edu/~branges

En seis líneas, Louis de Branges es un matemático que ha encarnado varias veces la fábula de Pedro y el Lobo con respecto a sus supuestas demostraciones correctas sobre algunos problemas abiertos, entre ellos la conjetura de Riemann; no obstante todavía le hacen un poco de caso los aldeanos de la comunidad matemática porque el pastorcillo de Louis halló a la oveja perdida de la demostración de la conjetura de Bieberbach. Para empeorar las cosas, utiliza herramientas matemáticas manejadas por un puñado, y el tiempo para entrenarse en ellas asusta a cualquiera.

(87) *Areas of abstract algebra are particularly amenable to study with the assistance of an automated reasoning program, areas that include group theory, ring theory, lattice theory, semigroups, and algebraic geometry. The amenability rests mainly with the type of axiomatization that is frequently encountered, namely, a set of equations that nicely map into unit clauses.*
pág.505 de (37 Wos)

(88) *Especially when condensed detachment is the rule of inference (from the viewpoint of logic), various areas are most amenable to study with your automated reasoning assistant.*

The areas include equivalential calculus, many-valued sentential calculus, and two valued sentential (or propositional) calculus.
pág.510 de (37 Wos)

(89)

§0. Introduction. There are two eras in the history of single axioms for groups and varieties of groups. The early results, by Neumann and others [7], often produced single axioms which were larger than the minimal possible size, but which were constructed via some scheme which made them easy to verify by hand. Although not optimal, these results had the virtue that a person could conceptually grasp their proofs. The second era began with the advent of McCune's automated reasoning system OTTER [4]; now one could produce shorter and simpler single axioms, which often required much more complex verifications. Short single axioms for groups and some varieties of groups were found by McCune and Wos [5,6], and by Kunen [2,3]; in many cases, these axioms were shown to be of optimal size. The results in these latter four papers had the defect that no insight was given into *why* the single axiom worked. The proofs often consisted of instructions on how to set up the OTTER input, and the reason given for considering a particular axiom was often that it survived an exhaustive computer search.

In this paper, we have our cake and eat it too – at least in the case of single axioms for odd exponent groups. We produce a large class of such axioms of minimal size, and we give conceptual proofs that these axioms are correct. We, too, used OTTER as a reasoning assistant, as advocated by Larry Wos, but found that by examining the output from our assistant, we could provide conceptual proofs which a human could also understand. This conceptual understanding, in turn, led us to discover more axioms.

Quien tenga la curiosidad de seguir las referencias numéricas hasta las notas encontrará que transcribí más de lo citado, pues el resto de la cita de Kunen también documenta la necesidad de la *revisabilidad* y como puede obtenerse ésta con la colaboración entre el programa y el usuario.

Kunen, Ken , Hart, Joan, *Single Axioms for Odd Exponent Groups,*

Las citas a las que alude Kunen:

[2] Kunen, K., Single Axioms for Groups, *J. Automated Reasoning*, 9:291 – 308, 1992.

[3] Kunen, K., The Shortest Single Axioms for Groups of Exponent 4, Technical Report UW #1134, University of Wisconsin, 1993; to appear, Computers and Mathematics and Applications.

[4] McCune, W. W., OTTER 3.0 Reference Manual and Guide, Technical Report ANL-94/6, Argonne National Laboratory, 1994.

[5] McCune, W. W., Single Axioms for Groups and Abelian Groups with Various Operations, *J. Automated Reasoning*, 10:1 – 13, 1993.

[6] McCune, W. W. and Wos, L., Applications of Automated Deduction to the Search for Single Axioms for Exponent Groups, in *Logic Programming and Automated Reasoning*, Springer-Verlag, 1992, pp. 131 – 136.

[7] Neumann, B. H., Another Single Law for Groups, *Bull. Australian Math. Soc.*, 23:81 – 102, 1981.

(90) *In contrast to algebra, areas such as set theory and Tarskian geometry play havoc with automated reasoning in general. Of course, progress is occurring in*

such areas; for the former, see (Quaife 1992; Belinfante 1998a; Belinfante 1998b), and for the latter, see (Quaife 1992). Nevertheless, the cited areas are still proving quite troublesome. More generally, if the representation includes non-Horn clauses (clauses with more than one positive literal) or clauses that simultaneously contain both equality literals and literals in some other predicate, then you can expect trouble.
pág.506 de (37 Wos)

(91) Axioma del subconjunto de comprensión de ZF: sea u un conjunto y $\varphi(v)$ una fórmula conjuntista, entonces existe un subconjunto w de u tal que $v \in w$ sii $\varphi(v)$.
De este modo, nuestro pobre lenguaje clausular no puede lidiar de manera finita con este axioma.
Otro axioma de ZF resaltante por sus amoríos con el infinito, es el axioma del conjunto inductivo, axioma donde se establece por medio de cuantificadores la existencia de conjuntos infinitos (¿recuerdan como hace poco más de un centenar de páginas empezó este capítulo?)
AxIoma del conjunto inductivo: existe un conjunto i tal que $\varnothing \in$ i y para todo v que pertenezca a i entonces su sucesor -$v \cup \{v\}$- pertenece a u .

(92) *The helplessness was derived from two sources. First, there existed no reasonably effective inference rule to capture set-theoretic reasoning (such an inference rule still does not exist).*
Wos, Larry , pág, 21 de (35)

(93) Belinfante, J., *Computer proofs in Gödel's class theory with equational definitions for composite and cross*, J. Automated Reasoning 22, No. 3, pp. 311-339.

Belinfante, J., *On computer-assisted proofs in ordinal number theory*, J. Automated Reasoning 22, No. 3, pp. 341-378, 1988.

(94) *...it is too complex to regard numbers and functions as built out of sets. No mathematician does so in everyday practice, and neither should automated deduction when the aim is to someday prove new theorems.*
Además el cáustico de Beeson, cita la opinión de Belinfante al respecto y de paso rememora una de tantas y tontas anécdotas bellas que inspiran el amor a la ciencia :

For the record, Belinfante agrees with this statement. His aim, however, is foundational. As a boy, he took Principia Mathematica from his physicist father'sbookshelf and said to himself, "Someday I'm going to check if all these proofs are really right!". That spirit still animates his work.

Beeson, M., The *Mechanization of Mathematics*, in Teuscher, C. (ed.) *Alan Turing: Life and Legacy of a Great Thinker*, Springer-Verlag, Berlin Heidelberg NewYork, 2003.
pág. 45 http://www.mathcs.sjsu.edu/faculty/beeson/Papers/turing2.pdf

(95) Theorema
Dirección electrónica: <http://www.theorema.org/>
Lenguaje de programación: Mathematica
Persona principal detrás del sistema: Bruno Buchberger

(96) *...second-order and (why not?) higher-order theorem proving has been in existence for a long time, and there are even whole conferences devoted to the subject, e.g. TPHOL (Theorem Proving in Higher-Order Logic). It seems that most of this research is not directed towards proving new theorems,*

Beeson, pág. 44 de (94)

(97) Besson ha trabajado con la implementación (y modificaciones) del algoritmo de Pietrzykolski* para la unifiicación en lenguajes de segundo orden. Reportes de sus avances se localizan en:

Beeson, M., *A second-order theorem prover applied to circumscription*, in: Gor, R., Leitsch, A., and Nipkow, T. (eds.), Automated Reasoning, First International Joint Conference, IJCAR 2001, Siena, Italy, June 2001, Proceedings, Lecture Notes in Artificial Intelligence 2083, Springer-Verlag (2001).

Beeson, M., *Solving for functions*, Journal of Symbolic Computation. A preliminary version appeared in: LMCS 2002, Logic, Mathematics, and Computer Science: Interactions, Symposium in Honor of Bruno Buchberger's 60th Birthday, pp. 24–38, RISC-Linz Report Series

* Pietrzykowski, T., and Jensen, D., A complete mechanization of second order logic, *J. Assoc. Comp. Mach.* 20 (2) pp. 333-364, 1971.

(98) *Ivy is a preprocessor and proof checker for resolution/paramodulation theorem provers. It is coded in ACL2 and proved sound for finite interpretations.*
ACL2 fue diseñado por J. Moore y Matt Kaufmann, y es el suceso de Nqthm, el demostrador de teoremas Boyer-Moore (capítulo cuarto)
http://www-unix.mcs.anl.gov/~mccune/acl2/ivy/

(99) Mizar
Dirección electrónica: <http://mizar.org/>
Líder del proyecto desde 1973: Andrzej Trybulec
El verificador (escrito en Pascal ¡aics!) fue elaborado por: Grzegorz Bancerek , Czeslaw Bylinski , Adam Grabowski, Artur Kornilowicz, Robert Milewski, Adam Naumowicz, Andrzej Trybulec , Josef Urban

Actualmente son tres las universidades en donde Mizar (sus peones) encuentra(n) sustrato:: Shinshu University in Nagano, University of Alberta de Edmonton, University of Bialystok (base).

Investigadores activos que cargan y acomodan ladrillos en la construcción de Mizar además de los albañiles polacos previos (más un checo de polizón, adivinen quién) :

Yasushi Fuwa , Mariusz Giero,Pauline N. Kawamoto,Jaroslaw Kotowicz ,Roman Matuszewski,Yatsuka Nakamura,Krzysztof Retel ,Piotr Rudnicki,Christoph Schwarzweller, Yasunari Shidama ,Bartlomiej Skorulski,Katsumi Wasaki,Freek Wiedijk,Mariusz Zynel.

(100) La dirección electrónica de Q*uod* E*rat* D*emonstrandum* es:
http://www-unix.mcs.anl.gov/qed/

El manifiesto QED, escrito a varias manos publicado en la red de redes, explicaba las pretensiones, motivaciones, contexto, boceto de plan de trabajo, objeciones y disoluciones de éstas, de la formalización computarizada de las matemáticas. Por su valor histórico vale la pena leerlo: http://www.rbjones.com/rbjpub/logic/qedres00.htm
Como muchos manifiestos a veces tiende más al lirismo llamativo, llama para prender a la inconsciencia, combustible de las revoluciones, que a una adusta objetividad promotora de la pasividad reflexiva. Por ejemplo entre los motivos para justificar la formalización de las matemáticas mencionaban:

to provide a cultural monument to "the fundamental reality of truth"
(sin comentarios)

Algo menos poético pero más determinante era su apertura con respecto al agente que elaborara a las pruebas:

there is great aesthetic, philosophical, scientific, educational, and technological value in the construction of the QED system, regardless of whether its construction is or is not largely done `by hand' or largely automatically.

(101) *It is Boyer's view that while a sufficient number of mathematicians and computer scientists are sufficiently in favor of the QED project to build it in the next century or two, hopes for much progress on it at the current time are currently dim because of sociological and political problems. There is currently at least some funding, and certainly reasonable opportunities for publication (which leads to scientific status, including tenure) for work on new ways of trying to do automated reasoning and, of course, for new mathematics. But he fears that there is very little funding or publication opportunity for the cooperative building of up an extremely large proof-checked edifice. Thus, Boyer's highest personal hope for the QED workshop is some insight into how to overcome these sociological and political problems, to establish practical, personal motivation for large scale cooperation on this endeavor.*

Opinión de Boyer transcrita por Roman Matuszewski pronunciada en el congreso QED de Varsovia, 1995
ftp://ftp.mcs.anl.gov/pub/qed/workshop95/intrpeople.txt

(102) Freek Wiedijk de la Universidad de Nijmegen comparó las características de los candidatos actuales más fuertes (encabezados por Mizar) para correr en la carrera de resistencia de la formalización de las matemáticas por medio de sistemas de cómputo. Por cierto Wiedijk tras realizar un análisis más o menos rigoroso estima que hasta dentro de 140 años se contará con una biblioteca formal-computarizada digna de las matemáticas. Ya Matusalén nos lo dirá. Algunos de los compañeros de Mizar y Otter/Ivy son:

HOL
Dirección electrónica: <http://www.cl.cam.ac.uk/Research/HVG/HOL/>
Lenguaje de programación: ML
Persona principal detrás del sistema: Mike Gordon

PVS
Dirección electrónica: <http://pvs.csl.sri.com/>
Lenguajes de programación: Lisp, ML
Personas principales encargadas del sistema: John Rushby, Natarajan Shankar, Sam Owre

Coq
Dirección electrónica: <http://pauillac.inria.fr/coq/>
Lenguaje de programación: ML
Personas principales encargadas del sistema: Gerard Huet, Thierry Coquand, Christine Paulin

ACL2
Dirección electrónica: <http://www.cs.utexas.edu/users/moore/acl2/>
Lenguaje de programación: Lisp
Persona principal detrás del sistema: J Strother Moore

Metamath
Dirección electrónica: <http://metamath.org/>
Lenguaje de programación: C
Persona principal detrás del sistema: Norman Megill

NuPRL
Dirección electrónica: <http://www.cs.cornell.edu/Info/Projects/NuPrl/nuprl.html>
Lenguajes de programación: ML, Lisp
Persona principal detrás del sistema: Robert Constable

Theorema
Dirección electrónica: <http://www.theorema.org/>
Lenguaje de programación: Mathematica
Persona principal detrás del sistema: Bruno Buchberger

Freek Wiedijk, *Comparing mathematical provers*,
http://www.cs.ru.nl/~freek/comparison/index.html

(103) La primera CADE se organizó en 1974 en –¿dónde más?- Argonne, normalmente se celebran cada dos años.
http://www.cs.albany.edu/~nvm/cade.html
En la CADE celebrada en Miami en el 2003, el máximo ganador de CASC (CADE ATP System Competition) fue el programa Vampire, otro probador resolutivo de lógica clásica de primer orden diseñado por Alexandre Riazanov y Andréi Voronkov en la Universidad de Manchester.
La biblioteca TPTP (Thousands of Problems for Theorem Provers) es administrada por Christian Suttner y Geoff Sutcliffe en la universidad de Miami.
http://www.cs.miami.edu/~tptp/

(104) http://www-unix.mcs.anl.gov/JAR/

(105) En `http://www.qpq.org/` se puede encontrar la dirección de algunos otros programas de razonamiento deductivo.

(106) *If open questions can be answered by researchers with so little knowledge, then mathematicians with solid credentials in a field might produce truly startling results*.

Wos pone como ejemplo de matemático con excelentes credenciales a Ken Kunen y su investigación asis tida por un programa de razonamiento deductivo en el álgebra abstracta, también podría ponerse sobre la mesa de la denuncia positiva al matemático Padmanabhan con su asociación deductiva McCune-Otter (59)
pág. 189 de (37 Wos)

(107) Zhang, J., Zhang, H., *SEM, a System for Enumerating finite Models of first-order many-sorted theories*, http://www.cs.uiowa.edu/~hzhang/sem.html

John Slaney, FINDER (Finite Domain Enumerator*)*,
http://users.rsise.anu.edu.au/~jks/finder.html

(108) http://www-unix.mcs.anl.gov/AR/mace2/

(109) W. McCune. *A Davis-Putnam program and its application to finite first-order model search: Quasigroup existence problems*. Tech. Report ANL/MCS-TM-194, Argonne National Laboratory, Argonne, IL, May 1994.
ref(59) con el anexo
http://www.mcs.anl.gov/home/mccune/ar/monograph/, 1996.

CAPÍTULO IV: LA INDUCCIÓN Y LA INCERTIDUMBRE

En la primera sección del capítulo cuarto se discutirá qué es la verificación formal y cómo puede llevarse a cabo a partir del germinal artículo de C.A. Hoare: *"An axiomatic basis for computer programming"*. La verificación formal es relevante para el tema de esta tesis porque es un método cuya meta es corroborar deductivamente el correcto funcionamiento de los sistemas computacionales. Dada su importancia desde la primera sección del presente capítulo se emitirán algunas observaciones sobre sus alcances, medios y limitaciones. En la segunda sección se escudriñará a una conspicua familia de programas empleada para realizar verificaciones pero también utilizada para demostrar teoremas y verificar pruebas, familia denominada como "demostrador Boyer-Moore". ACL2 es el miembro más reciente de la familia famosa entre otras cosas por haber incorporado con éxito un mecanismo deductivo bastante popular en las matemáticas: la inducción. Con no mucho detenimiento se inspeccionarán los objetos deductivos producidos por este programa. En la sección postrera se recrearán las almas con sed de pugna en los extractos de la discusión sobre los mitos y realidades de la verificación formal avivada por la tensión entre aquellos quienes consideran a la programación una actividad matemática pura –Hoare, Dijikstra, Wulf- y los otros para quienes dicha actividad acarrea incertidumbre ya sea por factores sociales –Lipton, De Millo, Perlis- o por la injerencia de la realidad física en los sistemas computacionales –Smith, Cohn, Fetzer- El destino de los programas pertinentes para la tesis –demostradores de teoremas/ verificadores de pruebas- apuntará hacia las direcciones resultantes de esta polémica.

PRINCIPIO

Las soluciones usualmente no vienen solas, suelen estar acompañadas por otras preguntas y nuevos problemas. Las soluciones digitales no son la excepción, han estado escoltadas por inquietudes y conflictos desde su aparición. En particular, una pregunta resuena con el engranaje de las respuestas computacionales y se amplifica conforme se extiende su asistencia y aumenta nuestra dependencia en ellas, a saber, ¿son confiables? O en el extremo, ¿pueden ser infalibles? ¿Qué nos garantiza que los programas cumplan con sus cometidos sin cometer algún error grave o en la exageración de nuestra exigencia sin incurrir en cualquier falla?

La pregunta sobre la confiabilidad con sus distintas gradaciones ha sido el origen de una copiosa y variada bibliografía, desde escritos técnicos-teóricos propios de las ciencias de la computación hasta ensayos con reflexiones éticas, económicas, jurídicas, etc. pertenecientes a otras actividades del intelecto; desde la definición de los parámetros de confiabilidad y las maneras para satisfacerlos hasta las repercusiones sociales causadas por las equivocaciones de un programa. Incluso la pregunta ha cimbrado a la estructura de las ciencias de la computación, cimentándolas según la respuesta con fundamentos deductivos o con argumentos anejos a las ciencias experimentales. Desde la década de los 1960's el programa de la *matematización* de las ciencias de la computación ha luchado por la hegemonía en la caracterización de dicha disciplina, siendo su motor principal la

verificación formal de los programas. Hoare en 1969 emitió el siguiente grito de guerra, repetido y repudiado en un sinnúmero de ocasiones:

La programación es una ciencia exacta ya que todas las propiedades de un programa y todas las consecuencias al ejecutarlo en cualquier ambiente dado pueden, en un principio, ser obtenidas del texto del propio programa por medio de un razonamiento completamente deductivo[1].

Escuchando la llamarada de Hoare, si a partir del código de un programa se pudiera corroborar deductivamente el cumplimiento puntual del programa al ser ejecutado de los objetivos para los que fue diseñado, entonces las demostraciones computarizadas podrían acendrarse y tendrían todo el derecho a formar parte de la familia demostrativa de las matemáticas. En caso de ser la declaración de Hoare un lamento cantado por quimeras, entonces el sello de la duda por más tenue que sea estará impreso en TODAS las demostraciones generadas por las computadoras. Si la hipótesis de Hoare con respecto a la programación implica que las demostraciones computarizadas son válidas, de la refutación del supuesto sobre la exactitud de la programación no se desprende la negación de la conclusión; las demostraciones computarizadas pueden ser tan válidas como el resto de las demostraciones matemáticas y su validación puede ser sometida a procesos análogos a los empleados en las demás criaturas deductivas. Sin embargo, en el segundo y tercer capítulos se señalaron a los sistemas computaciones como los agentes idóneos para la ratificación de las demostraciones computarizadas, por lo cual muchas de las demostraciones digitales tienen y tendrán resquicios donde se escondan los recelos. Peor aún, la propagación de la duda es inmediata, pues ¿de que privilegios gozamos los humanos para afirmar nuestra infalibilidad? Quien ignora entre comillas sabe, pues dice saber sin decir lo que ignora. Si estamos hechos para le verdad, ¿en verdad de qué estamos hechos?

En este capítulo se despejará la disyunción entre la visión clara o psicotrópica de Hoare et al (Dijkstra, Wulf, etc.) sobre la solidez deductiva de la programación restringida a los programas demostradores –generales (CAPIII) o particulares (CAPII)- y sus consecuencias en los productos demostrativos manufacturados. Narrador terrible cual soy, al darle mayor peso al fin del relato descarto casi todo el desarrollo histórico-teórico y me limito a describir prosaicamente en esta sección con base al papel seminal de Hoare qué podemos entender como verificación formal y cómo podemos llevarla a cabo.

¿Para qué se elabora un programa? Para satisfacer ciertos requerimientos en aras de la resolución de algún problema. De este modo, primero se debe establecer concisamente dichos requerimientos, esta enumeración en el lenguaje técnico se llama especificación del problema. La *matematización* comienza desde la especificación, pues puede ser escrita en un lenguaje formal, i.e. en un lenguaje lógico-matemático. Los programas de nuestro interés tienen dos metas primarias: la generación y la verificación de demostraciones (pruebas). El ejemplo más sencillo se aboca a la segunda finalidad, sea PVCP un programa cuya especificación informal y general sería la siguiente:

PVCP recibe como entrada una secuencia de fórmulas del CP y su tarea es reconocer si la secuencia es o no es una prueba. Una especificación formal de PVCP debe desglosar y restringir a las nociones incluidas en la especificación anterior, entre otras cosas se debe precisar qué cadenas de símbolos constituyen fórmulas bien formadas del CP, qué reglas de inferencia son aceptadas y qué secuencias de fórmulas son consideradas una prueba, todo lo anterior redactado en un lenguaje formal. La especificación indica qué debe hacer un programa, el cómo es enunciado por los algoritmos. Aquí algoritmo debe ser entendido como una secuencia finita de operaciones enfocadas a la resolución de un problema particular, sin la obligación de la terminación ni del éxito. En pos de la verificación formal, después de la etapa de la especificación empieza la corroboración de la definición correcta de los algoritmos, es decir, se debe demostrar que satisfagan a las especificaciones. Si la especificación así lo ordena, tras demostrar su correcta definición entonces un algoritmo puede obedecer a la connotación rigurosa del término, en la cual se pide imperiosamente la satisfacción de los objetivos en un tiempo finito con un número finito de operaciones. Retomemos nuestro ejemplo previo:

ESPECIFICACIÓN PVCP

Sea a_1, a_2, ...a_n una secuencia de fórmulas bien formadas del CP
PVCP(a_1, ..., a_n)=T si a_1,..., a_n es una prueba de a_n
sino PVCP(a_1, ... , a_n)=F

ALGORITMO PVCP en pseudocódigo

```
respuesta:= T ;
i:=1;
MIENTRAS (respuesta = T y i≤n)
HAZ{
      Si a_i es una instancia de un axioma o existen 1≤k,j≤i tal que a_k,a_j∈ {a_1, ...a_n}  y
      MP(a_k,a_j)= a_i
                  entonces i:=i+1;
      sino  respuesta:= F;
    }
```

Pagando una cuota baja de atención percibimos el entrelazamiento del algoritmo y la especificación del PVCP; en el ciclo del MIENTRAS se determina la noción de prueba y regla de inferencia (modus ponens) aceptadas. Aunque en este ejemplo la mezcla sea una maña y su separación pueda establecerse de antemano, para algunos problemas existe una dependencia dinámica adhesiva entre la especificación y los algoritmos, pues comúnmente conforme se avanza en la elaboración de los éstos surgen otros aspectos importantes no considerados en la especificación original. En el siguiente nivel también es frecuente la revoltura entre programa y especificación, situación a evitar si buscamos realizar una verificación formal como más adelante se explicará. Por otro lado, si queremos demostrar la correcta definición del algoritmo PVCP podemos hacerlo por reducción al absurdo o por

inducción, aunque en este caso el algoritmo sea una traducción transparente de la definición de prueba a la cual nos atenemos y su corroboración parezca innecesaria. ¿De qué maneras puede fallar el algoritmo? Adjudicándole el estatus de prueba a una secuencia de fórmulas cuando no siguen nuestra definición de prueba o viceversa., v.gr. sea b1,...bm una secuencia de fórmulas bien formadas de CP que satisfagan nuestra definición de prueba a la cual nuestro algoritmo le otorgue el valor F. Entonces existe r∈ {1 ,..,m} tal que b_r provoca la asignación resultado=F, de donde se deriva que b_r no es una instancia de un axioma ni una fórmula obtenida vía modus ponens de otro par de fórmulas previas pertenecientes a la secuencia, contradiciendo a nuestra definición de prueba. Pobre es la demostración porque demasiada rica es la definición del algoritmo, ostentosa pero poco meticulosa, reluciente para quien puede determinar si una fórmula es una instancia de un axioma o el resultado de un modus ponens pero parva para una agente de naturaleza más mecánica carente de las capacidades de decisión anteriores. Si buscamos facultar a las computadoras con las habilidades para resolver problemas, no bastan los algoritmos, hace falta una traducción a un lenguaje más cercano a las acciones realizables por una computadora. El acto de programar (escritura de los algoritmos en lenguajes aptos para ser decodificados y ejecutados por una máquina) es el acto sexual en el lecho de la computación. La abundancia y diversidad sintáctica de los lenguajes de programación contenidas están en una clasificación escalonada, la cual arranca en el lenguaje de ceros y unos manejado por la máquina y llega hasta los lenguajes de alto nivel aledaños al lenguaje lógico-matemático-natural hablado por los programadores en su discurso algorítmico. Su basto vocabulario, su portabilidad, su presentación propiciadora de una lectura y escritura no tan complicadas, son algunas de las ventajas de los lenguajes de alto nivel; facilidades apreciadas por los programadores y correspondidas por su preferencia en la redacción de programas en ellos. No obstante estos lenguajes deben transformarse en algo ejecutable por una máquina, siendo la compilación uno de los procesos típicos involucrados en la conversión de las instrucciones de alto nivel a instrucciones en lenguaje binario, proceso sujeto a fallas y a una posible disminución en el desempeño del programa. Las traducciones difícilmente son óptimas debido al drástico cambio de un texto en un lenguaje profuso hacia un lenguaje rudimentario, transformación provocadora de redundancias en muchas ocasiones y de un aumento de la demanda de recursos del sistema. Es más, si se necesita una eficacia alta lo recomendado es programar en lenguajes más cercanos al lenguaje máquina, como el lenguaje ensamblador. Desgraciadamente si vamos tras la verificación formal, LA opción son los lenguajes de alto nivel. Después de haber demostrado la correcta definición de los algoritmos, en pos de la certeza debemos corroborar la correcta implementación ellos en los lenguajes de programación. ¿Cómo hacerlo? Incluyendo en nuestro análisis a los elementos inherentes al lenguaje de programación cuya validez es incuestionable en un principio y a partir de los cuales junto con los axiomas lógicos-matemáticos atenidos a las operaciones lógicas-matemáticas al alcance del lenguaje y de su ejecutor puede deducirse la total satisfacción de la especificación por parte del programa o la correcta implementación de los algoritmos en el programa. Los lenguajes de alto nivel gracias a su proximidad con los lenguajes lógicos-matemáticos permiten enunciar su estructuración con una interpretación semántica de tonos formales, es decir colorear a los enunciados del código con tintes lógico-matemáticos y matices propios de las características de dichos lenguajes; en cambio

los lenguajes cercanos al lenguaje máquina –ensamblador- exhiben una estructuración menos abundante y más rústica, poseen una primitiva sintaxis ocasionadora de enrevesadas redacciones reacias a derretirse a la claridad de las interpretaciones formales y los subsecuentes razonamientos deductivos. Además los lenguajes vecinos al lenguaje máquina se expresan a expensas del diseño de la computadora en donde viven, no tienen la generalidad prometida por la portabilidad de los lenguajes de alto nivel. Por tales motivos los lenguajes de alto nivel son los elegidos para ponerse bajo la lupa de la evaluación deductiva, evidenciando el trueque normalmente acaecido entre efectividad y confiabilidad, coyuntura presente incluso cuando sólo trabajamos con lenguajes de alto nivel (las redacciones pueden ser rebuscadas y extensas con tal de mejorar el rendimiento dificultando su inspección). En lugar de examinar el buen funcionamiento del programa infiriendo su correcta implementación a partir de la comparación entre el programa y sus algoritmos previamente avalados, sin tanto rodeo podemos establecer deductivamente la satisfacción de las especificaciones con base al texto del programa y su caracterización lógico-matemática-endémica del lenguaje en cuestión. Sin embargo tener la seguridad del correcto funcionamiento de los algoritmos es un paso hacia la confianza en el programa, no tan pequeño ya que la caminata en la vía a continuación trazada puede ser larga y cansina, tanto como para requerir el uso de un vehículo más veloz. El trabajo de Hoare marcó el rumbo de de los defensores del paradigma de la formalización lógico-matemática de la programación, indicando los primeros pasos de la verificación formal de los programas. Aterricemos lo escrito con las ideas primigenias de Hoare encontradas en su germinal artículo *Axiomatic basis for computer programming.*

La meta de Hoare es la construcción de un sistema formal donde se incluyan operaciones y propiedades esenciales de casi cualquier lenguaje de alto nivel, para edificar en este sistema las pruebas del correcto funcionamiento de los programas escritos en ese lenguaje modélico. Como lo indica el título del artículo, uno de los primeros objetivos es la determinación de los axiomas. En primer lugar, las operaciones lógicas-matemáticas deben acotarse a las capacidades del instrumento y a los límites marcados por los lenguajes de programación. Por ejemplo una computadora tiene un límite en la representación de números enteros o algunos lenguajes de programación de alto nivel pueden no trabajar directamente con números reales (punto flotante). Tomando esto en consideración, Hoare enuncia entre otros a los siguientes axiomas aritméticos:

A1 $x + y = y + x$
A2 $x \times y = y \times x$
A3 $(x + y) + z = x + (y + z)$
A4 $(x \times y) \times z = x \times (y \times z)$
A5 $x \times (y + z) = x \times y + x \times z$
A6 $y \leq x \Rightarrow (x-y) + y = x$
A/ $x + 0 = x$
A8 $x \times 0 = 0$
A9 $x \times 1 = x$
$A10_F$ $\forall x\,(x \leq MAX)$

$A11_B$ MAX + 1 = MAX
$A11_M$ MAX + 1 = 0

Los axiomas del A1 al A9 son propiedades básicas de las operaciones suma, multiplicación en Z y un axioma para la relación de orden simétrica (A6) ; $A10_F$ señala la existencia del máximo entero positivo representable por la computadora (MAX) y $A11_{B,}$ $A11_M$ fijan dos alternativas para manejar el *overflow,* la aritmética módulo M ($A11_M$) o la aritmética con un máximo firme ($A11_B$).

Hoare prosigue internándose de lleno en los terrenos de la programación, indicando a la relación de ciertas condiciones (valores) iniciales con los resultados del cómputo como base para establecer la validez de un programa (o parte de éste como una instrucción). Esta observación de manera más precisa se escribe en la clásica notación de Hoare así:

$$P \{Q\} R$$

donde P son las precondiciones, Q el programa y R los resultados de la ejecución de Q; de este modo , Hoare afirma "Si P es verdadero antes de la inicialización de Q entonces R debe ser verdadera después de su terminación". Puede no haber precondiciones, rescribiéndose así la aserción:

$$true \{Q\} R$$

Terminado el preámbulo, se puede comenzar a enunciar la base axiomática propia del lenguaje de programación refiriéndose a sus componentes y derivaciones elementales.

AXIOMA DE ASIGNACIÓN

La instrucción "x:= f", i.e. la asignación del valor f a la variable identificada como x, debe respetar que si la afirmación P(x) es verdadera después de la ejecución de la asignación entonces P(f) también debe serlo antes de ésta:

$$D0 + P_0 \{x:=f\} P$$

donde x es el apuntador (una variable en un programa en realidad identifica a una localidad de memoria o a un registro de la computadora), f es una expresión (puede ser un término donde esté presente x), P_0 es obtenida sustituyendo a las x's con f's en P(x). Por lo cual, el axioma de asignación es un esquema debido a las múltiples afirmaciones P's posibles. Compañeras de los axiomas, las reglas de inferencia también deben ser estipuladas.

REGLAS DE INFERENCIA

- Reglas de consecuencia: Si la ejecución de Q asegura la verdad de la afirmación R, entonces también avala la verdad de cualquier afirmación que sea consecuencia lógica de R. Por otro lado, si P es una precondición del programa Q productor de R,

entonces cualquier otra afirmación de la cual P sea consecuencia lógica es también una precondición.

D1 Si + $P\{Q\}R$ y $R \Rightarrow S$ entonces + $P\{Q\}S$
Si + $P\{Q\}R$ y $S \Rightarrow P$ entonces + $S\{Q\}R$

- Reglas de composición: Un programa usualmente esta constituido por una sucesión de instrucciones dependientes, por lo cual es forzoso representar esta secuenciación mediante una regla de composición::

D2 Si + $P\{Q_1\}R_1$ y + $R_1\{Q_2\}S$ entonces+ $P\{Q_1;Q_2\}S$

- Regla de iteración: Existen varias instrucciones en los lenguajes de la programación para ordenar la repetición de una serie de instrucciones hasta que se cumpla una condición predeterminada, representemos una muy sencilla con el esquema "while B do S" (mientras B haz S), así si B es verdadero entonces debe ejecutarse otra vez S. Sea P una afirmación que siempre es verdadera al terminar S cuando P es verdadera antes de hacerlo; de esto modo en el ciclo del "while..." P siempre será verdadera antes y después de cada iteración de S. La condición de parada del ciclo es la falsedad de B. Tomando en consideración las anteriores observaciones, la regla de iteración se escribiría:

D4 Si + $P \wedge B\{S\}P$ entonces + P {while B do S}$P \wedge \neg B$

Con el axioma de asignación y las tres reglas de inferencia, Hoare definió una parte de un sistema formal presente en casi todos los lenguajes de alto nivel (donde puede variar ligeramente es en la regla de iteración), además decidió aumentar el poder de dicho sistema con la incorporación de los axiomas aritméticos y uno de orden enunciados en A1,...$A11_x$, perdiendo muy poca generalidad pues casi todo lenguaje de alto nivel debe incluir instrucciones primitivas para las operaciones aritméticas de la suma, resta y multiplicación

Continuando con la edificación de la sintáctica, las fórmulas del sistema tienen la forma "P{Q}R" , donde las afirmaciones P,R estarán escritas en un lenguaje lógico-aritmético o la forma de un enunciado aritmético con conectivos lógicos. Para completar la construcción del sistema formal hace falta una definición: prueba será una secuencia de fórmulas donde cada una de ellas es un axioma (axioma de asignación) o resultado de aplicar las tres reglas de inferencia a fórmulas previas en la sucesión o axiomas-teoremas del sistema axiomático A1-$A11_x$. Hoare para ejemplificar una prueba recurre a un problema de juguete cuya especificación es encontrar el cociente q y residuo r de la división x/y, con $x \geq y$, $x, \in N$, $y>0$. El programa Q propuesto para satisfacerla es:

r:=x; q:=0;
while ($y \leq r$) do (r:=r-y; q:=1+q)

Escrita la especificación de Q en manera formal, el programa debe hallar un par de enteros q,r tal que x=q×y+r con r<y , por lo cual se debe demostrar:

true{Q}¬y≤r ∧ x=r+q×y

El guión de la prueba elaborado por Hoare es el siguiente:

Line number	Formal proof	Justification
1	$\mathbf{true} \supset x = x + y \times 0$	Lemma 1
2	$x = x + y \times 0\{r := x\}x = r + y \times 0$	D0
3	$x = r + y \times 0\ \{q := 0\}\ x = r + y \times q$	D0
4	$\mathbf{true}\ \{r := x\}\ x = r + y \times 0$	D1 (1, 2)
5	$\mathbf{true}\ \{r := x;\ \ q := 0\}\ x = r + y \times q$	D2 (4, 3)
6	$x = r + y \times q \wedge y \leqslant r \supset x =$ $(r-y) + y \times (1+q)$	Lemma 2
7	$x = (r-y) + y \times (1+q)\{r := r-y\}x =$ $r + y \times (1+q)$	D0
8	$x = r + y \times (1+q)\{q := 1+q\}x =$ $r + y \times q$	D0
9	$x = (r-y) + y \times (1+q)\{r := r-y;$ $q := 1+q\}\ x = r + y \times q$	D2 (7, 8)
10	$x = r + y \times q \wedge y \leqslant r\ \{r := r-y;$ $q := 1+q\}\ x = r + y \times q$	D1 (6, 9)
11	$x = r + y \times q\ \{\mathbf{while}\ y \leqslant r\ \mathbf{do}$ $(r := r-y;\ \ q := 1+q)\}$ $\neg y \leqslant r \wedge x = r + y \times q$	D3 (10)
12	$\mathbf{true}\ \{((r := x;\ \ q := 0);\ \ \mathbf{while}\ y \leqslant r\ \mathbf{do}$ $(r := r-y;\ \ q := 1+q))\}\ \neg y \leqslant r \wedge x =$ $r + y \times q$	D2 (5, 11)

El conectivo "⊃" equivale a "⇒", el Lema 1 puede ser probado de los axiomas A7 y A8 y el Lema 2 de los axiomas A3, A5, A6 y A9[2].

El trabajo pionero de Hoare sólo indicó la dirección a seguir, derrotero caminado por muchos investigadores después de él para desarrollar a la verificación formal, la demostración (formal) del cumplimiento de las especificaciones (formales) lograda por un programa, en particular à la Hoare a partir de la interpretación formal del lenguaje de programación. Mientras que verificar a un programa denota una actividad más global, una

labor cuyo objetivo es respaldar a la confiabilidad del programa sin ofrecer necesariamente refuerzos deductivos tan fuertes. La verificación y su dominio formal son áreas demasiado frondosas y amplias para ser cubiertas en pocas cuartillas donde además es fácil perderse si no se cuenta con la experiencia suficiente. Por lo tanto poco he de agregar a lo explicado, con lo reseñado del artículo de Hoare se vislumbra qué es y cómo puede realizarse la verificación formal, en la siguiente sección de también manera superficial se delineará un programa reciente (ACL2-demostrador Boyer-Moore) utilizado para verificar formalmente tanto software como hardware. Más aun, únicamente nos interesa la verificación con respecto a cierto tipo de programas (generadores/verificadores de demostraciones) No obstante algunas observaciones hechas por el profeta Hoare deben ser escuchada para evitar especular a la deriva:

- Con acuidad señala Hoare la imposibilidad de decidir si el programa termina o no lo hace con su metodología formal, conservando la consistencia con el irresoluble problema de la parada. Por lo tanto, la interpretación de " P{Q} R" debe apostillarse con "bajo el supuesto de la terminación de Q". Berg delimita la diferencia entre la verificación total y la verificación parcial con base a poseer o no la garantía de la terminación del programa[3]. Hoare en cambio sugiere trabajar con verificaciones condicionales, sujetas a la definición en el programa de una cota máxima de iteraciones permitidas para no vagar indefinidamente en ciclos infinitos.

- *La práctica de suplir pruebas a programas no triviales se va a difundir cuando se hagan disponibles técnicas para generar pruebas considerablemente más poderosas, y todavía contando con ellas no va a ser tarea fácil*[4]. Esas técnicas anheladas por Hoare encarnaron en un instrumento acostumbrado a trabajar mecánicamente, con un poder atroz para almacenar y manipular símbolos en un tiempo mínimo. En los capítulos primero y tercero se develó la identidad de dicho instrumento y su ventaja con respecto a los seres humanos para trabajar con sistemas formales y encontrar pruebas en ellos. James King fue el primero en elaborar programas destinados a verificar a otros programas, siendo la estrella de estos últimos un programa para calcular potencias enteras de números enteros[5]. Desde los 1970's el asistente común de la verificación formal es la computadora ejecutando otros programas diseñados ex profeso, aunque la advertencia de Hoare sigue vigente, ni con el apoyo de los monstruos de silicio es una tarea fácil, Los programas útiles y no de utilería tienen miles de líneas de código, por ejemplo la primera versión de Otter de 1987 consistía en alrededor de 20000 líneas escritas en C. Por otro lado las expectativas de pruebas muuuuy cortas es sólo un mugido pronunciado por vacas optimistas, ya que en las pruebas formales por definición está prohibido saltarse pasos. No obstante, la estrategia del divide y vencerás ha alentado a quienes están involucrados en la rama de la verificación formal, tal como lo expresó Hoare: *el método general más poderos para solucionar problemas complicados es dividirlo en problemas más sencillos, los cuales pueden ser resueltos de manera independiente*[6]. La programación estructurada, aquella en donde el programa se constituye con bloques principales compuestos por distintas

subprogramas es altamente recomendada no sólo para la verificación formal, sino para la verificación en general pues gracias al ordenamiento claro de un programa es más fácil localizar errores o reutilizar otras subprogramas ya probados.

- La verificación formal es aplicable a las demás elementos de software y hardware involucrados en la ejecución de un programa, Hoare escribió: *Cuando el correcto funcionamiento de un programa, de su compilador y el hardware en donde se ejecute sean establecidos con rigor matemático, entonces se puede depositar casi toda nuestra confianza en los resultados del programa y predecir sus propiedades con un grado de certeza limitado únicamente por la fiabilidad de los componentes electrónicos*[7]. La llave para la verificación formal es la construcción de un sistema formal para el cual el lenguaje de programación o el estrato del hardware enfocado sean un modelo de éste, de este modo, la metodología formal no está restringida nada más al software.

- Por último, una frase lapidaria de Hoare: *Sin embargo, la demostración de programas [verificación formal],... va a ser difícil hasta para los programadores de alto calibre, y talvez sea sólo aplicable a los programas con un diseño sencillo. Como en otras áreas, la confiabilidad sólo puede comprarse pagando el precio la simplicidad*[8]. Pueden pagarse otros precios, como el de la paciencia casi infinita. Al intentar verificar formalmente sistemas no tan sencillos suele dedicarse una considerable cantidad de tiempo, período crítico para ciertas aplicaciones pues pueden aparecen en el mercado académico o industrial otros programas con mayores aptitudes y aunque no estén libre de fallas -...- la ausencia de éstas no es el único parámetro de evaluación, por ejemplo la tolerancia a las fallas –acciones responsivas de un sistema ante su ocurrencia- en muchas aplicaciones es igual o más importante que su ausencia . Para el contexto de la tesis, esta observación apunta la mira de la verificación hacia los programas más sencillos requeridos, aquellos capaces de ratificar/refutar a las supuestas pruebas computarizadas completas. Ni modo demostración exhaustiva computarizada, tu caso está atrapado en la telaraña de la dificultad descrita por Hoare donde la araña de la imposibilidad busca devorarte. Peor aun, si recordamos los programas de Koch en aras de las optimización de recursos sacrificaron su ascenso a otro escaño de la escalera de la confiabilidad y fueron escritos en lenguaje ensamblador.

Para concluir, el paradigma de la *matematización* de las ciencias de la computación encabezado por la verificación formal circula en su forma dura por los cuatro puntos bombeados desde el formal corazón de Hoare:

1. *Las computadoras son máquinas matemáticas.*
2. *Los programas son expresiones matemáticas.*
3. *Un lenguaje de programación es una teoría matemática que incluye conceptos, notaciones, definiciones, axiomas y teoremas.*
4. *La programación es una actividad matemática.*[9]

Si este paradigma reflejara a las ciencias de la computación, entonces nuestras demostraciones computarizadas podrían adquirir el boleto de ingreso al paraíso de los serafines deductivos alabados por los fervientes practicantes de la actividad matemática. En la última sección del presente capítulo se atenderán las pugnas teológicas incitadas por los demonios digitales.

INTERMEDIO

Sea Q(n)=n!=m , cuando $n \in N\backslash\{0\}$:

m:=n;
while (n>1) do (n:=n-1; m:=m*n)

¿De qué maneras podemos demostrar el correcto funcionamiento de Q? Una ya expuesta sería cargar con la abultada metodología semántica de Hoare, sin embargo resalta otra para quien ha recibido un poco de entrenamiento en las matemáticas y conoce uno de los trucos demostrativos más difundidos cuando trabajamos con ordinales. El método de las aserciones inductivas[10] busca demostrar la validez de las afirmaciones de salida (postcondiciones) sobre un programa con base a ciertas afirmaciones de entrada (precondiciones) -hasta aquí parece calca de P{Q}R- recurriendo a ciertas relaciones constantes entre las variables, cuando estamos trabajando con ciclos a dichas relaciones se les llama invariantes del ciclo (*loop invariants*). En el caso de los ciclos, se debe intentar demostrar inductivamente la existencia de una invariante mediante la cual se sostengan las afirmaciones de salida. Las invariantes del ciclo típicamente establecen una relación entra variables mantenida después de un número arbitrario de iteraciones, mientras que las especificaciones son el candidato natural a elegir como afirmaciones de salida.

Regresemos a nuestro ejemplo Q:

Precondición : $n \in N\backslash\{0\}$

Postcondición: m=n!

¿Cuál es la invariante del ciclo requerida? Para ejemplos triviales la respuesta no rompe con la armonía de la simpleza. Después de i iteraciones, la invariante propuesta es:

$m_i(n_i-1)! = n!$

Por observación en el programa Q se infiere lo siguiente:
$n_i = n_{i-1}-1 = n-i$, $m_0 = n_0 = n$, $m_i = m_{i-1}(n_i)$ y el número de iteraciones del ciclo while es igual a $n-1$, de donde $m=m_{n-1}$.

Al concluir el ciclo while obtendríamos:
$n! = m_{n-1}(n_{n-1}-1)! = m_{n-1}(n-(n-1)-1)! = m_{n-1}(0!) = m$

Corroborada la utilidad de la invariante sugerida, falta demostrar su calidad de invariante en el ciclo. Para hacerlo obedecemos el nombre del método, haciendo inducción sobre i.

Caso base: i=0, de donde n=1, por lo cual $m_0(n_0-1)! = 1(1-1)! = 1 = 1! = n!$

H.I.: Para i=k se cumple $m_k(n_k-1)! = n!$

Paso inductivo: Para i=k+1, tenemos

$$m_{k+1}(n_{k+1}-1)! = m_k\, n_{k+1}\,(n_k-1-1)! = m_k\,(n_k-1)(n_k-2)! = m_k(n_{k-1})!$$
$$\text{por H.I.} = n!$$

Por medio de una interpretación formal de Q (estructura aritmética con inducción como regla de inferencia) se demuestra el correcto funcionamiento de Q, a saber, Q(n)=n! si $n \in N\backslash\{0\}$. Para verificar programas menos párvulos haciendo uso de las aserciones inductivas, se señalan las precondiciones, condiciones invariantes y postcondiciones para cada uno de los módulos constituyentes del programa para después demostrar que en toda posible trayectoria de ejecución del programa entre sus módulos y sus correspondientes condiciones se satisfacen a las especificaciones del programa. El enfoque de las aserciones inductivas fue adoptado por varios investigadores, de hecho en un artículo de 1985 los especialistas Robert Boyer y Strother Moore recalcaban:

Porque este acercamiento ha recibido por mucho la mayoría de la atención, ha producido los resultados más impresionantes hasta la fecha[11].

Desafortunadamente ninguno de los programas demostradores del capítulo tercero manejan a la deducción por inducción, en general ningún sistema resolutivo lo hace. Ante esta omisión, aquellos quienes en el 85 remarcaban los logros de las aserciones inductivas una década antes desarrollaban un programa demostrador con inducción incluida, marcando el inicio de uno de los programas más influyentes con sus diversas reencarnaciones en las áreas particular de la verificación y global de la demostración automatizada de teoremas, programa identificado con el nombre Demostrador Boyer-Moore.

Para incorporar a la inducción hizo falta cambiar de sistema lógico hacia uno en donde se tuviera una estructura más propicia para que el agente digital la llevara a cabo. Se le dio la espalda al CPPO y se viró hacia la aritmética recursiva primitiva, sistema estrechamente relacionado con el lenguaje de programación LISP.

En la aritmética recursiva[12] sólo se utilizan métodos constructivos, especialmente dos: la recursión y la inducción. Por hacer uso exclusivamente de esta clase de métodos, dicha aritmética en principio puede trasladarse transparentemente hacia los sistemas computacionales (inclusive se puede definir a la computabilidad con base a las funciones

recursivas en lugar de las Máquinas de Turing). La aritmética recursiva primitiva (ARP) consta de las siguientes funciones iniciales:

- funciones constantes de la forma $k_{nc}(x_1,...,x_n) = c$ p.a. constante c
- funciones de proyección de la forma $p_{ni}(x_1,...,x_n) = x_i$ p.a. $1 \leq i \leq n$
- función sucesor de la forma s(x)=x+1

Por otro lado la ARP posee dos reglas para generar nuevas funciones:

- composición de funciones recursiva primitivas g, h_1,.....,h_m de la forma:
  $f(x_1,...,x_n) = g(h_1(x_1,...,x_n),...,h_m(x_1,...,x_n))$

- recursión de las funciones recursiva primitivas g,h de la forma:
  $f(x_1,...,x_n, 0) = g(x_1,...,x_n)$
  $f(x_1,...,x_n, s(x)) = h(x_1,...,x_n , s(x) , f(x_1,...,x_n,x))$

Una función recursiva primitiva es una función inicial o es generada a partir de las funciones iniciales aplicando el par previo de reglas. Ejemplos:

Definición de suma (x,y)=x+y en ARP

$suma(x.0)=p_{11}(x)=x$
$suma(x,s(y))=s(p_{33}(x, s(y), suma(x,y)))=s(suma(x,y))$

prod(x,y)=x×y
$prod(x,0)=k_{10}(x)=0$
$prod(x,s(y))=suma(x, prod(x,y))$

fact(x)=x!
$fact(0)=k_{01}=1$
$fact(s(x))=prod(s(x),fact(x))$

Cuando la inducción es la bujía de la demostración y queremos encender a la hipótesis inductiva en el paso inductivo debemos vincular P(s(x)) con P(x), de donde las definiciones recursivas favorecen el chispazo. LISP es un lenguaje de programación para procesar listas creado por John McCarthy en los 1960's -¿quién desarrolló el primero? respuesta en el primer capítulo- . El núcleo de LISP tiene características afines a la aritmética recursiva primitiva:

- uso de expresiones simbólicas como datos
- las funciones son la unidad primaria de los programas
- recursión como el mecanismo primario para describir los cómputos

Las expresiones simbólicas (*s-expressions*) incluyen numerales para representar naturales -0,1,2,…-, constantes booleanas -T , F-, .el átomo NIL para denotar el fin de las listas y los pares ordenados (s . t) donde s, t son cualquiera de las expresiones simbólicas aquí mencionadas. Por ejemplo la lista (1,2,3) se escribe en LISP (1 .(2. (3. NIL))) aunque se puede abreviar '(1 2 3). Algunas de las operaciones primitivas en LISP son:

- (cons x y): construye el par ordenado (x.y).
- (car x): regresa el primer elemento de x si x es un par ordenado, F cualquier otra cosa.
- (cdr x) regresa el segundo elemento de x si x es un par ordenado, F cualquier otra cosa.
- (consp x): regresa T si x es un par ordenado, F cualquier otra cosa.
- (endp x): regresa T si x es un átomo, i.e. T si x no es un par ordenado.
- `(booleanp x) regresa T si x es un dato booleano, F si no lo es.`
- `(numberp x) regresa T si x es un número, F si no lo es.`

La última versión del demostrador Boyer-Moore responde al nombre de ACL2 y fue elaborado por J. Moore y Matt Kaufman[12]. El sistema formal donde habita ACL es una lógica de primer orden para las funciones recursivas totales (funciones recursivas primitivas definidas para todos sus argumentos). La sintaxis de ACL2 es la de versión de LISP llamada Common Lisp[13]. ACL2 es un sistema lógico compuesto por un compendio de reglas para definir (o axiomatizar) funciones recursivas, para declarar propiedades acerca de ellas y para demostrarlas, todas estas acciones tienen cierto nivel de automatización. ACL2 maneja otros tipos de datos como los caracteres, cadenas de caracteres –normalmente utilizados para regresar comentarios sobre los errores- , en lugar de la constante booleana F utiliza NIL y sus números son racionales o incluso complejos racionales. El operador lógico principal de ACL2 es `(if x y z)`, el cual regresa z si x es igual a NIL y si difiere entonces regresa y (if x then y else z).

Los demás operadores lógicos pueden construirse a partir de if, por ejemplo:
(NOT x) = `(IF x NIL T)`
(IMPLIES x y)= `(if x (if y T NIL) T)`
(AND x y) = `(if x (if y T NIL) NIL)`

El operador básico de la igualdad es (EQUAL x y), cuyo valor es T sii x=y. Cabe señalar la posibilidad de definir otras relaciones de equivalencia en ACL2, mediante la instrucción (defequiv equiv), atajo para indicar en ACL2 la reflexividad, simetría y transitividad de la relación equiv:

```
(and (booleanp (equiv x y))
     (equiv x x)
     (implies (equiv x y) (equiv y x))
     (implies (and (equiv x y)
                   (equiv y z))
              (equiv x z)))
```

Los axiomas de ACL2 esencialmente especifican el valor de cada término evaluable. Algunos axiomas de los operadores/constantes anteriormente descritas:

AXIOM 1 T ? NIL
AXIOM 2 x = y ? (EQUAL x y) = T
AXIOM 3 x ? y ? (EQUAL x y) = NIL
AXIOM 4 x = NIL ? (IF x y z) = z
AXIOM 5 x ? NIL ? (IF x y z) = y
...
AXIOM 45 (CONSP (CONS x y) = T
AXIOM 47 (EQUAL (CAR (CONS x y)) x)
AXIOM 48 (EQUAL (CDR (CONS x y)) y)

Para extender el número de operaciones evaluables se pueden definir nuevas funciones en ACL2. Una definición de una función f_n es válida si f_n está definida por recursión y existe un buen orden para alguna medida de los argumentos donde dicha medida decrece en cada paso de la recursión (esto último sirve para garantizar la terminación de la recursión) o f_n está construida con funciones previamente aceptadas. Una instrucción para definir nuevas funciones es:

(DEFUN nombrefuncion (arg1 ,,, arg2) cuerpo)

Como ejemplo de libro de texto definamos a la función `app` (*append*) cuya finalidad es concatenar dos listas x,y:

```
(defun app (x y)
 (if (consp x)
    (cons (car x) (app (cdr x) y))
       y))
```

Es decir, si x es un par ordenado entonces construye paulatinamente a la concatenación con el primer elemento de x `-(car x)-` y con el resto de x concatenado con y `-(app (cdr x) y)-` en cada llamada recursiva; de lo contrario regresa y. Si x,y son ambas listas, la función definida cumple con los cometidos de la concatenación.

ACL2 automáticamente avala la definición:

```
The admission of APP is trivial, using the relation O< (which
is known to be well-founded on the domain recognized by O-P)
and the measure (ACL2-COUNT X).  We observe that the type of APP is
described by the theorem (OR (CONSP (APP X Y)) (EQUAL (APP X Y) Y)).
We used primitive type reasoning.

Summary
Form:  ( DEFUN APP ...)
Rules: ((:FAKE-RUNE-FOR-TYPE-SET NIL))
Warnings:  None
```

```
Time:  0.03 seconds (prove: 0.00, print: 0.00, other: 0.03)
 APP
```

El buen orden es el establecido por la desigualdad estricta (`O<`) y la medida corresponde al número de elementos del argumento x `(ACL2-COUNT X)` de la función `app`. En cada llamada recursiva se remueve al primer elemento de x `(cdr x)` por lo que la medida va decreciendo conforme se aplica la recursión. .El tipo de dato del resultado de la función app es reconocido automáticamente por ACL2:

`(APP X Y)` es un par ordenado `(CONSP (APP X Y))`
ó
`(APP X Y)` es igual a y

Entre los mecanismos deductivos de ACL2 se cuentan con procedimientos de decisión, reescritura condicional –similar a la demodulación restringida a ciertas condiciones lógicas pero con más relaciones de equivalencia además de la igualdad-, simplificaciones proposicionales e inducción matemática hasta el ordinal ε_0 (aclaración más adelante). El principal comando para demostrar es `defthm`, veámoslo en acción al pedirle a ACL2 la demostración de la asociatividad de `app`:

```
(defthm associativity-of-app
        (equal (app (app a b) c)
               (app a (app b c))))
```

ACL2 responde:

```
Name the formula above *1.
Perhaps we can prove *1 by induction.  Three induction schemes are
suggested by this conjecture.  Subsumption reduces that number to two.
However, one of these is flawed and so we are left with one viable
candidate.

We will induct according to a scheme suggested by (APP A B).  If we
let  (:P A B C) denote *1 above then the induction scheme we'll use
is
(AND
   (IMPLIES (AND (NOT (ENDP A))
                 (:P (CDR A) B C))
            (:P A B C))
   (IMPLIES (ENDP A) (:P A B C))).

This induction is justified by the same argument used to admit APP,
namely, the measure (ACL2-COUNT A) is decreasing according to the
relation
O< (which is known to be well-founded on the domain recognized
by O-P).  When applied to the goal at hand the above induction
scheme produces the following two nontautological subgoals...
```

Antes de proseguir con la arenga deductiva de ACL2 desenredemos al nudo trascrito:
Los tres esquemas de inducción referidos son planteados a partir de la definición recursiva de app aplicada a los términos de la conjetura:

```
(app a b)
(app b c)
(app a (app b c))
```

donde la variable subrayada es sujeta a ser descompuesta recursivamente.

Sin embargo, `(app a b)` subsume a `(app a (app b c))`. Se descarta `(app b c)` pues en el otro término donde aparece b -`(app a (app b c))`- no se puede hacer la inducción sobre esta variable. Por lo cual el esquema escogido para la inducción es `(app a b)`. De este modo, se debe demostrar:

PASO INDUCTIVO Si A no es la lista vacía -`(NOT (ENDP A))`- ,
`( app (app (CDR A) B) C) = (app (CDR A) (app B C))` implica que `(app (app (A B) C) = (app A (app B C))`

CASO BASE: Si A es la lista vacía -`(ENDP A)`- entonces
`(app (app (A B) C)=(app A (app B C))`

ACL2 se cerciora del buen planteamiento de la inducción de manera análoga a la aprobación de las definiciones recursivas de funciones, encontrando un buen orden y una medida decreciente -`(ACL2-COUNT A)`-. La reseña de ACL2 sobre sus cabriolas derivativas aunque meticulosa omite algunos pasos, elude por ejemplo aludir los brincos deductivos triviales como lo son las tautologías. Sigamos con los galimatías de ACL2.

```
..induction scheme produces the following two nontautological subgoals

Subgoal *1/2
(IMPLIES (AND (NOT (ENDP A))
              (EQUAL (APP (APP (CDR A) B) C)
                     (APP (CDR A) (APP B C))))
         (EQUAL (APP (APP A B) C)
                (APP A (APP B C)))).

By the simple :definition ENDP we reduce the conjecture to

Subgoal *1/2'

(IMPLIES (AND (CONSP A)
              (EQUAL (APP (APP (CDR A) B) C)
                     (APP (CDR A) (APP B C))))
         (EQUAL (APP (APP A B) C)
                (APP A (APP B C)))).

But simplification reduces this to T, using the :definition APP, the
:rewrite rules CDR-CONS and CAR-CONS and primitive type reasoning.
```

ACL2 nos informa la demostración del paso inductivo esbozando cómo lo hizo. En primer lugar haciendo uso de la definición de `(ENDP x)` sustituye en *1/2 `(ENDP A)` por `(NOT (CONSP A))`, luego simplifica `(NOT (NOT (CONSP x))= (CONSP x)` produciendo así la submeta *1/2'.

Luego, a partir de la definición recursiva de `(APP A B)` se puede rescribir `(APP (APP A B) C)` por:

```
(APP (IF (CONSP A)
         (CONS (CAR A) (APP (CDR A) B))
         B)
     C)
```

Pero por hipótesis, `(CONSP x)=T por lo cual obedeciendo las reglas de (if x y z)` tenemos :

```
(APP (CONS (CAR A) (APP (CDR A) B))
     C)
```

Expandiendo la definición del `app`:

```
(IF (CONSP (CONS (CAR A) (APP (CDR A) B)))
           (CONS (CAR (CONS (CAR A) (APP (CDR A) B)))
                 (APP (CDR (CONS (CAR A) (APP (CDR A) B))) C))
    C)
```

Pero aplicando el axioma 45, `(CONSP (CONS (CAR A) (APP (CDR A) B))) = T.` por lo que tenemos respetando al IF:

```
(CONS (CAR (CONS (CAR A) (APP (CDR A) B)))
 (APP (CDR (CONS (CAR A) (APP (CDR A) B))) C))
```

Debido al axioma 47, `(CAR (CONS (CAR A) (APP (CDR A) B)))= (CAR A),` entonces podemos rescribirlo así:

```
(CONS (CAR A) (APP (CDR (CONS (CAR A) (APP (CDR A) B))) C))
```

Por causa del axioma 48, `(CDR(CONS(CAR A)(APP(CDR A) B)))= (APP(CDR A) B),` de nuevo rescribimos el término:

```
(CONS (CAR A) (APP (APP (CDR A) B) C )
```

Pero por hipótesis deductiva, `(APP (APP(CDR A)B) C)=(APP (CDR A)(APP B C)),` por lo cual otra vez rescribimos acercándonos al objetivo:

```
(CONS (CAR A) (APP (CDR A)(APP B C)))
```

Recapitulando, queremos demostrar la igualdad:

```
(EQUAL (CONS (CAR A) (APP (CDR A)(APP B C))) (APP A (APP B C))))
```

Disminuyendo la nitidez derivativa e imitando la celeridad de ACL2, `(APP A (B C)))` equivale a `(CONS (CAR A) (APP (CDR A) APP (B C)) )` por la definición de `app` y la hipótesis `(CONSP A)`. Queda así establecida la igualdad:

```
Subgoal *1/2'
(IMPLIES (AND (CONSP A)
              (EQUAL (APP (APP (CDR A) B) C)
                     (APP (CDR A) (APP B C))))
         T)
```

Finalmente, x implica T siempre es T:

```
Subgoal *1/2'
T
```

ACL2 termina la demostración con el caso base:

```
Subgoal *1/1
(IMPLIES (ENDP A)
         (EQUAL (APP (APP A B) C)
                (APP A (APP B C)))).

By the simple :definition ENDP we reduce the conjecture to

Subgoal *1/1'
(IMPLIES (NOT (CONSP A))
         (EQUAL (APP (APP A B) C)
                (APP A (APP B C)))).

But simplification reduces this to T, using the :definition APP and
primitive type reasoning.

That completes the proof of *1.

Q.E.D.

Summary
Form:  ( DEFTHM ASSOCIATIVITY-OF-APP ...)
Rules: ((:REWRITE CDR-CONS)
        (:REWRITE CAR-CONS)
        (:DEFINITION NOT)
        (:DEFINITION ENDP)
        (:FAKE-RUNE-FOR-TYPE-SET NIL)
        (:DEFINITION APP))
Warnings:  None
Time:  0.27 seconds (prove: 0.10, print: 0.05, other: 0.12)
 ASSOCIATIVITY-OF-APP
```

El caso base `-Subgoal *1/1-` se infiere directamente de la definición de `app`, ya que `(ENDP A)` equivale a `NOT(CONSP A)`, por lo cual `(APP A B)=B` y `(APP A (APP B C))= (APP B C)`. Por lo tanto:

```
(APP (APP A B) C)= (APP B C)= (APP A (APP B C))
```

Los ejemplos sencillos aunque inyectan entendimiento también pueden inficionar confusión. Son escasos los casos de éxito automático del demostrador, su eficacia reside más en la simbiosis del usuario y el programa. El esquema del comando `defthm` tiene otros parámetros para admitir la asistencia del usuario:

```
(defthm nombre términoconjeturado

        ...
        : hints (“subconjeturameta”
                : sugerencias)
        ...)
```

De este modo podemos brindarle una gama de sugerencias al demostrador:

- Señalar lemas anteriormente demostrados o fácilmente demostrables útiles para la demostración de la conjetura meta.
  ```
  :hints (("[1]Subgoal *1/1.2'" :use lemma23))
  ```

- Inhibir la inducción para la demostración de la conjetura especificada.
  ```
  :hints (("Goal"
   :do-not-induct t))
  ```

- Desactivar la reescritura para algunas funciones
  ```
  :hints (("Goal"
  :hands-off (length binary-append))
  ```

- Activar la inducción sin considerar otras reglas –v.gr.simplificación-
  ```
  :hints (("Goal"
  :induct t)
  ```

- Ordenar la aplicación inmediata de la inducción sobre una función (ACL2 corrobora si es posible y además genera el esquema de inducción) con apariencia de servir a la demostración de la conjetura meta
  ```
  :hints (("Goal"
  :induct (EQUAL (app b NIL) b))
  ```

- Asumir como demostrado la conjetura indicada y proceder con el resto de la demostración., si el triunfo bendice a la gesta entonces convenientemente ACL2 lo relativiza recordando nuestras deudas deductivas `-the proof would have succeeded had the indicated goals been proved-`, aunque ahora puede enfocarse `defthm` para saldarlas.

```
:hints (("Goal"
:BY nil)
```

- Señalar que la submeta equivale o es subsumida por otro término.

```
:hints (("Goal"
:by (:instance associativity-of-app (x a) (y b) (z c))
```

Si el valor obtenido por `defthm` es `T` y si en el término demostrado aparece una relación de equivalencia (como la igualdad) entonces por defecto se agrega al flamante nuevo teorema a la base de datos de las reglas de reescritura –el equivalente a los demoduladores-. Por lo que en nuestro ejemplo, cada vez que ACL2 encuentre un término con la forma `(app(app x y)z)` va a ser rescrito como `(app x(app y z))`. Esta consecuencia de `defthm` es un arma de dos filos, su utilidad depende de las circunstancias, por lo cual si únicamente deseamos demostrar a la conjetura debemos agregar a `defthm` el parámetro `:rule-classes nil`.

Las demostraciones de ACL2 se van construyendo paulatinamente con la interacción del usuario y el programa. Las conjeturas a demostrar se dividen en lemas ya sea de antemano por el usuario o por la inspección de alguna demostración fallida de ACL2. Los creadores advierten:

No es fácil conseguir que ACL2 demuestre teoremas no triviales. Para lograrlo, el usuario debe comprender:
** el modelo,*
** ACL2 como una lógica matemática, y*
** debe poder entrenarse en la construcción interactiva de demostraciones con ACL2*[14].

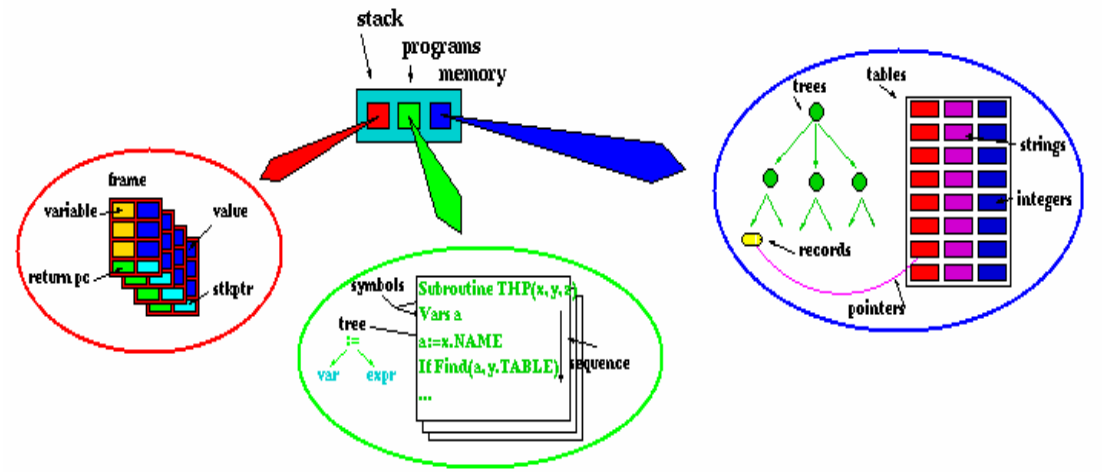
Representar adecuadamente al problema a través del lenguaje de ACL2 constituye el primer reto. Gracias a su tendencia constructivista, i.e. todo de cuanto se habla en ACL2 debe ser efectivamente calculable, algunas gracias matemáticas son despedidas: adiós a la cuantificación no acotada, hasta luego cuantificación explícita –ACL2 aunque la permite no ofrece mecanismos para explotarla-, auf Wiedersehen a los reales y au revoir conjuntos infinitos. La ausencia explícita de los cuantificadores es aprovechada para aumentar el nivel de automatización en la demostración de teoremas. Aunque a primera vista parezca débil el poder de representabilidad de ACL2, en palabras de Natarajan Shankar , avezado usuario del demostrador Boyer-Moore:

Es sorprendente el gran fragmento de las matemáticas discretas y las ciencias de la computación expresables en esta lógica libre de cuantificación[15].

Las estructuras matemáticas deben modelarse con las listas expresiones recursivas del demostrador Boyer-Moore. Algunos teoremas se prestan a la traducción, por ejemplo el teorema de incompletud de Gödel –donde la aritmética recursiva dijo presente- fue trasladado y demostrado en 1986 por Natarajan Shankar en Nqthm, predecesor de ACL2[16]. En 1995 Ken Kunen cazó otro indemostrable con el rifle Nqthm , colgando en el muro de su

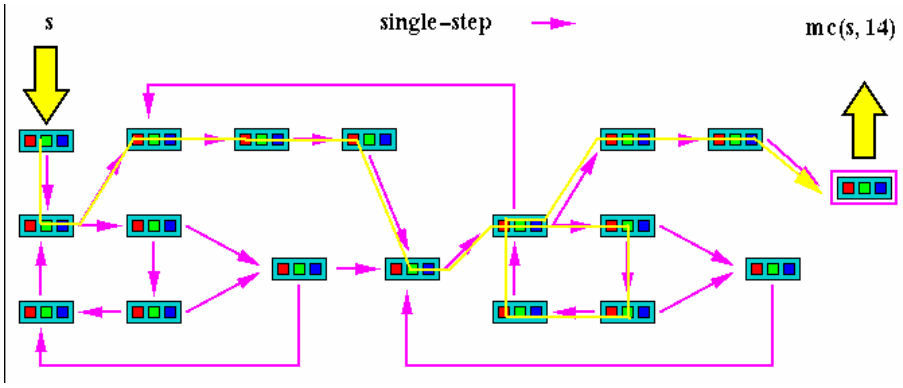
vanidad el trofeo del teorema Paris-Harrington-Ramsey gracias a la mira de la inducción transfinita hasta el ordinal límite $\epsilon_0 = \omega^{\omega^{\omega^{\cdots}}}$.[17]

El software/hardware en ACL2 pueden ser modelados como una máquina de estados, compuesta por los estados y sus relaciones de transición. Los estados en los sistema de cómputo normalmente involucran los siguientes objetos: booleanos, enteros, vectores, arreglos, pilas, tablas, cadenas, directorios, archivos, caracteres, páginas etc. En general son objetos discretos y pueden servir como ladrillos en una construcción (recursiva) de los estados del sistema. El siguiente diagrama representa un estado típico:



Las funciones de transición y la simulación de su ejecución también pueden someterse a su formalización en ACL2. Por ejemplo, sea `(single-step s)` la función de transición de un solo paso, i.e. para un estado `s` `(single-step s)` calcula el siguiente estado del sistema. Definamos una función cuya finalidad `sea` calcular la sucesión de transición de longitud n:

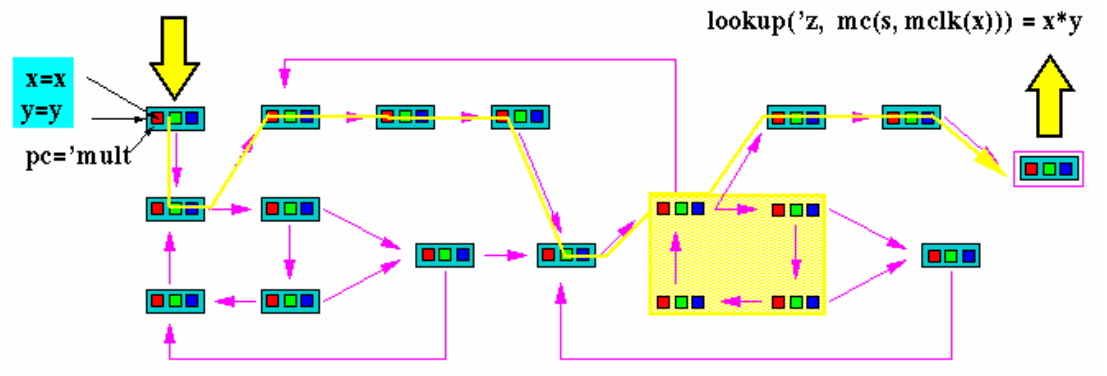
```
(defun mc (s n)                          ; avanzar desde s n veces
 (if (zp n)                              ; si n es igual 0
     s                                   ;   entonces regresa s
     (mc (single-step s) (- n 1))))      ; sino step single-step(s)
                                                    n-1 veces.
```



Imaginemos al sistema modelado como una especie de unidad aritmética y el estado inicial indique la instrucción para la multiplicación de x por y, siendo el resultado almacenado en z. Más aún, debido al diseño de la unidad aritmética modelada y cómo en ella la multiplicación está implementada podemos definir una función para evaluar el número de pasos requeridos en la multiplicación de x por y, v.gr. la unidad aritmética descomponga a la multiplicación x×y en una suma con x sumandos y
-y+y+...+y- y `(mclk x)` calcule el número de pasos requeridos para evaluar dicha operación. El propósito detrás de las acciones formalizadoras es demostrar teoremas del modelo, por ejemplo se puede trabajar en conjunto con ACL2 para demostrar que nuestra hipotética unidad aritmética multiplica correctamente:

```
(defthm  MC_mult_is_a_multiplier
       (implies (and (natp x)
                     (natp y))
                (equal (lookup 'z (mc (s 'mult x y) (mclk x)))
                       (* x y))))
```

Donde `natp` es una función para reconocer números naturales y lookup importa el valor de la variable z después de la ejecución de un programa.



Este ejemplo de papel exhibe la estrategia general para utilizar ACL2 en la verificación de software/hardware. En esta rama el demostrador Boyer-Moore ha sido una herramienta bastante socorrida, su última encarnación ha asistido entre otras a la verificación del algoritmo de división para punto flotante del procesador AMD5k86
y a la verificación de algunas funciones en la máquina virtual de Java[18].

Más entonado con la tesis, ACL2 es una alternativa atractiva para elaborar los programas verificadores de pruebas computacionales completas. El grupo de Argonne, en particular William McCune, así lo contempló y su programa Ivy –verificador de pruebas por resolución/paramodulación- fue escrito en ACL2. Afín a este afán, el programa para validar la demostración de Otter de la conjetura de Robbins fue elaborado con el antecesor de ACL2. Programar en un lenguaje inmerso por defecto en un sistema formal fomenta una construcción lógicamente sólida y apresta la verificación de dichos programas, además si está redactado en ACL2/Nqthm se puede recurrir a ACL2/Nqthm para llevarla a cabo. Sin

embargo algunas incompatibilidades deben ser arregladas. Otter y en general los programas adscritos al paradigma clausal deben preprocesar las conjeturas a diferencia de ACL2 (transformación a la forma normal conjuntiva y skolemización); no obstante la traducción puede modelarse mediante funciones y demostrarse correcta en ACL2. Por otro lado el sistema formal de primer orden debe representarse en ACL2, desde su sintaxis hasta sus reglas de inferencia y nociones de prueba. De este modo fue construido Ivy, programa portador de la garantía de ser correcto bajo dominios de interpretación finitos otorgada por McCune-Matlin-ACL2[19]. Es una garantía contundente pero parcial, ya que hay teoremas verdaderos para todo dominio de interpretación finito pero falsos para algún dominio infinito, v.gr. sea f:X? X, si p.t. x,y f(x)=f(y) implica x=y, entonces p.t. x existe y tal que f(y)=x. No obstante, ACL2 es una opción recomendada y recomendable por su reforzada cimentación lógica para construir los programas aludidos. Además, la quisquillosa pregunta planteada desde hace muchos párrafos sigue sumergida en el suspenso, ¿aun cuando se pudiera verificar formalmente la solidez de Ivy/programa verificador de teoremas, esto garantiza su infalibilidad?

Por último, la cuestión primigenia se asoma con curiosidad de gato. ¿Cómo son las demostraciones de ACL2? Contesto con lo obvio en el sentido contrario, sus demostraciones no son pruebas formales, como ya se percibió en el ejemplo de la asociatividad de `app`. Las demostraciones individuales, aquellas construidas con `defthm` y comandos similares, son bosquejos de una demostración, terminarlas aunque sea una labor tediosa tan poco es tan complicada, para muestra falta la exposición previa de la demostración `ASSOCIATIVITY-OF-APP`. No obstante, las conjeturas demostradas individualmente normalmente son lemas usados para concertar la demostración de la(s) conjetura(s) originales. De este modo, las demostraciones suelen ser un registro secuencial de eventos, i.e. extensiones de la lógica de ACL2/Nqthm. Entre esos eventos se incluye la importación de una teoría (axiomas y teoremas demostrados), las definiciones de funciones (`defun`) avaladas por ACL2 o las demostraciones de lemas (`defthm`) para su posterior empleo o desempleo como reglas de reescritura. Sobra decir que las demostraciones sobre teoremas no tan simples o verificaciones de software/hardware no de utilería son extremadamente largas. Por ejemplo, la "demostración" de Kunen-Nqthm del teorema de Paris-Harrington es un aluvión de cientos de páginas[20]. El tamaño se puede reducir drásticamente si omitimos las demostraciones individuales, aunque el orden todavía araña a las centenas[21]. Reconstruir minuciosamente la demostración acabando todos los bocetos de las demostraciones individuales es una manda demasiado opresora. Aunque de manera similar a lo sucedido con las pruebas computacionales completas, con un poco de esfuerzo se puede extraer el núcleo epistemológico de la demostración y compactarse en una demostración más corta y más ortodoxa. En este caso no es tan difícil el desentierro pues el usuario en la construcción de la demostración fue cómplice del programa y sabe donde hundir las palas del intelecto. Ken Kunen publicó un artículo de 16 páginas con una demostración más común aunque poco corriente -difiere de la clásica- de la demostración del teorema Paris-Harrington-Ramsey inspirada por Nqthm –ver nota (17)-. No obstante, la forma sugerida por la forma de estas demostraciones para masticarlas es emplear las mandíbulas mecánicas de Nqthm/ACL2 y seguir/ejecutar el registro de

eventos/instrucciones integrantes de la susodicha demostración, para así procesar su comprensión. El libro *Metamathematics, Machines and Gödel's Proof* (15) fue fruto de la demostración Shankar/Nqthm del primer teorema de incompletud de Gödel; en él se explica detalladamente la construcción del modelo en la lógica Boyer-Moore y la edificación secuencial de la demostración interactiva entre el programa-usuario, acreditando este texto la *inteligibilidad* de esta demostración computarizada. Es más, sabedor de la caracterización de Tymoczko, Shankar en su libro declara:

... las demostraciones aquí descritas son, en principio, revisables incluso sin asistencia mecánica. Se deja al lector decidir si son convincentes, pero claramente son formalizables.[22]

En principio, un ser humano a pie puede viajar desde Alaska hasta la Patagonia. En la práctica, conviene utilizar medios de transporte mecánicos. La preocupación inquietante y manifiesta desde el segundo capítulo es cuando se llegan a zonas deductivas inalcanzables sin la propulsión computarizada para los matemáticos de carne y hueso, perdiéndose de este modo el contacto con las sendas usuales de la verificación. Sin embargo, en principio cualquier cálculo deductivo efectuado por una computadora puede realizarlo un ser humano, al fin y al cabo, fue otro ser humano quien diseñó los algoritmos y los instaló mediante programas en una computadora, programas compuestos por operaciones tan sencillas que hasta una computadora las puede realizar. En la práctica, la carga de trabajo agota a nuestras capacidades, carecemos de la fuerza bruta de las máquinas. Aunque un hombre sea un nadador experimentado, si decide cruzar el Atlántico haciendo sólo uso de sus habilidades de tritón tarde o temprano se ahogará. No está fuera del alcance de nuestro intelecto repetir la miríada de operaciones de la demostración del lema de reducibilidad en la demostración de H&A&K&1&0 del Teorema de los Cuatro Colores. No es imposible sino inoperante, es potencialmente imposible. Los itinerarios deductivos elaborados por ACL2/Nqthm-usurario se asemejan bastante a las estructuradas demostraciones estándares, demostraciones enlazadas con escalas secundarias (definiciones, lemas) donde el intelecto reposta para llegar a la demostración de los teoremas principales. Por otro lado fue un compañero de especie el lazarillo del programa, por lo cual la extensión de la demostración aunque sea extenuante por su origen está dentro del rango de lo *revisable*, al menos en el aspecto global. Además, cada demostración individual a su vez puede ser leída, comprendida y completada con un poco de paciencia y dedicación, obsequiando esta característica la llave para abrir su rigurosa *formalización.* No obstante, desmenuzar cada demostración individual para luego explicitar sus pasos faltantes sin apoyo mecánico es una tarea propensa al error y propagadora de la cerrazón. Por enésima vez arremeto con el argumento ad hominen o pro machina, las computadoras son mejores que los hombres en los cálculos mecánicos -¿serán infalibles?-. Si bien Shankar sostiene la hipotética *revisabilidad* hecha por entero por los humanos, algunas páginas antes menciona los requisitos para creer (*believe* [sic]) en su prueba:

1. Creer en la solidez lógica del demostrador de teoremas Boyer-Moore y

2. Estar convencido de que el teorema de la incompletud ha sido enunciado correctamente.[23]

Shankar pregona la confianza en el programa -¿será equiparable a la mantenida entre colegas matemáticos?-. Por eso deja al gusto o disgusto del lector decidir si es o no convincente su demostración –en general las demostraciones edificadas con la ayuda de Nqthm/ACL2- aunque para los escépticos recalcitrantes les informa que en principio pueden *revisar* por sus propios miedos –perdón, medios- a su demostración –en general a cualquier demostración edificada con la ayuda de Nqthm/ACL2 aún cuando de soslayo se vislumbre el infierno de las tareas posibles precursoras de lo imposible- . Por otra parte si el ímpetu por la formalización meticulosa nos conmina a terminar todos los esbozos demostrativos, ¿acaso no es otra vez el instrumento generador el candidato idóneo para llevarlo a cabo? Así como Otter produce objetos de prueba, no es tan difícil agregarle un módulo a ACL2 cuya función sea expulsar a las enormes serpientes de las demostraciones con sus escamas derivativas meticulosamente detalladas para así con la manzana de la completud convencer a los duros adoradores de la forma pura. Dicho sea de paso, los saltos deductivos en la demostración de Shankar-Nqthm brincan a los ojos de los observadores humanos únicamente en el nivel de las demostraciones individuales, por lo cual esta demostración exhibe una de las presentaciones más rigurosas –salvo el detalle mencionado y sólo quienes rechacen al inciso primero lo considerarán intolerable- de la demostración del primer teorema de incompletud de Gödel.

Para dejar de entrever y empezar a ver el entretejido derivativo con mayor claridad deben ensuciarse las manos con el teclado. Para compenetrarse con un lenguaje el aspirante a respirarlo debe incesantemente practicarlo. ACL2/Nqthm está montado sobre un lenguaje de programación, LISP, empleado como marco lógico para los cuadros deductivos. La enunciación del modelo en la lógica Boyer-Moore exige un dominio en la lengua meta de la traducción., no se pide menos para su entendimiento. LISP tiene la enorme ventaja de ser un lenguaje altamente *matematizado*, un lenguaje fuertemente emparentado con la aritmética recursiva. Las flechas de las dudas difícilmente penetrarán la coraza del sistema formal donde reside ACL2, el convencimiento del segundo inciso puede anclarse teóricamente. Además, los computarifóbicos pondrán ninguna objeción pues es otro miembro de su especie quien realiza la traducción. A sazón del capítulo anterior se puede discurrir con mayor detenimiento sobre los mecanismos deductivos de ACL2, promoviendo el convencimiento del primer inciso. Sin embargo, no basta con dar pruebas de papel sobre el correcto diseño de los algoritmos/mecanismos subyacentes a los programas en esta tesis discutidos. Hace falta escudriñar al agente en acción, evaluar a las garantías del buen funcionamiento de los programas en ejecución. Es hora de retomar una ajada discusión, la polémica del paradigma de le verificación formal.

FIN

La verificación formal suele contrastarse contra la usual exterminación experimental de los errores, erradicados conforme aparecen en el programa al someterse a un exhaustivo y representativo número de situaciones de evaluación, mientras una alternativa presumiblemente posee una prueba sobre el correcto funcionamiento del programa la otra tras distintas cualidades –v. gr. la velocidad- se reconoce acreedora de la incertidumbre prestada por la estrategia de la prueba y el error. Quienes defiendan con vehemencia al paradigma de la *matematización* entrarán en resonancia con las palabras de Dijkstra:

El probado experimental puede descubrir la presencia de errores en los programas, pero nunca podrá enseñar su ausencia[24].

La efervescencia de la verificación formal esparció la fiebre de la *matematización* de las ciencias de la computación cuyos síntomas están contenidos en los cuatro puntos de Hoare. Contagiado por el frenesí William Wulf en 1979 afirmaba:

La cosa irritante sobre la normalmente baja calidad de gran parte del software actual es que no hay ninguna razón extrínseca para justificarla; la perfección es, en principio, posible. A diferencia de los dispositivos físicos:
(i) Hay ninguna ley natural que limite el grado de depuración de los programas producibles; un programa puede ser construido exactamente como fue especificado.
(ii) No hay algún principio de incertidumbre de Heisenberg operatorio; una vez construido, un programa se comportará exactamente como fue prescrito.
(iii) Hay ninguna fricción por el uso; la exactitud y el correcto funcionamiento de un programa no se deteriorarán con el paso del tiempo.[25]

El febril anhelo de la *matematización* tampoco provocaba por sí solo delirios esquizofrénicos. El aparato físico receptor del alma codificada en programas, es un objeto material asolado por las leyes naturales, situación no ignorada por los defensores del paradigma. Parafraseando a Hoare, los componentes electrónicos son la última –y quizás única- barrera para la perfección Wulfiana. Pero ese muro puede desmoronarse en gran medida ya que el hardware y los demás procesos involucrados para convertir el código de un programa en algo ejecutable por una máquina también pueden ser verificados formalmente. Un rechoncho magnate y un escuálido campesino tardan el mismo tiempo en caer desde el último piso de la Torre Latinoamericana sin importar las marcadas disparidades en su corpulencia y opulencia, es un hecho con cierto grado de exactitud tolerado por los físicos. Aunque si ambos individuos caricaturescos emprendieran el vuelo en picada en el "vacío", tardarían exactamente lo mismo, otro físico -probablemente obsesionado con su eminencia Galileo- precisa. Un programa verificado formalmente satisface sus especificaciones, siempre y cuando el compilador, el sistema operativo, el hardware hagan bien su trabajo. Aunque si todos los agentes involucrados en la ejecución del programa han sido verificados formalmente, entonces las computadoras son máquinas

matemáticas que interpretan puntualmente las correctas traducciones matemáticas efectuadas por el compilador a partir de las bien definidas expresiones matemáticas de los programas, algún defensor duro del paradigma de la *matematización* precisa y sostiene a la escalera hacia la materialización de la perfección wulfiana.

El anterior párrafo barrunta una línea de ataque contra la verificación formal dura, aunque respetando su orden cronológico, la relego. Dicho sea de paso, algunas de las críticas en contra de la verificación formal serán un poco ignoradas al no corresponder con la situación de los programas relevantes para la tesis. Justo en el año cuando Wulf luchaba por el asentamiento de la perfección en el software, un trío de computólogos con la publicación de *"Social processes and proof of theorems and programs"* [26] enguantaron al primer gran golpe asestado por ellos un par de años antes contra la magnificencia de la verificación formal. El entonces desconocido par de Richards, R. Lipton y R. DeMillo, en complicidad con el renombrado investigador Alan Perlis cometieron el delito de lesa majestad en el Cuarto Simposio sobre los Principios de los Lenguajes de Programación organizado por ACM (Association for Computing Machinery, longeva asociación de enorme prestigio en el mundo de la computación). Recopilo algunos puntos de la ofensiva en su presentación definitiva de 1979, ideas cáusticas de especial relevancia para la tesis pues su fuerza –o falta de ella- descansa en la noción de lo que es una demostración matemática advocada por el acusatorio trío integrado por Lipton, DeMillo y Perlis (L&D&P), noción adivinable en el título de su artículo.

Discutidores precoces, en el primer párrafo sus palabras enfilan el grueso de sus macanazos:

... una prueba es sólo un paso en la dirección del convencimiento. Nosotros creemos que, en la última instancia, es un proceso social el que determina si los matemáticos se sienten seguros sobre un teorema - y nosotros creemos que, porque ningún proceso social comparable puede tener lugar entre los verificadores de programas, la verificación formal de programas está ligada al fracaso.[27]

Al darle ese inaugural acogimiento a lo "social", es de esperarse un sobrecogimiento –por no decir encogimiento- de su argumentación filosófica y nuestra expectativa no se ve defraudada. La coronación de los procesos sociales suele estar respaldada por los argumentos provenientes de disciplinas enfrascadas en esas pestilentes constelaciones. La historia, y sobre todo la historia reciente, aquella donde todavía se huele al hedor del presente, son la principal fuente –y también inconveniente- de las disquisiciones de L&D&P. Pero su posición es todavía más excluyente, pues al despreciar a los hijos de la verificación formal, parientes consanguíneos de las pruebas formales, inexorablemente reniegan de la postura clásica de la filosofía de las matemáticas con respecto a lo que debiera ser una demostración. Impelidos por transferir su contemplación del desarrollo matemático a la definición de demostración, declaman:

... la demostración de un teorema es un mensaje. Una demostración no es un bello objeto abstracto con una existencia independiente.[28]

Siguiendo las brechas de L&D&P, una demostración es un mensaje y como tal requiere de varios canales de comunicación. Debe ser legible y se lectura interesante para un corpúsculo de matemáticos en su primera fase de vida. En ese periodo se depura normalmente por la interacción social entre matemáticos mediante sugerencias y correcciones. De este modo el mensaje se edita para su publicación. Después llega el momento de la validación, encargada a un comité de especialistas en el tema. Tras recibir el aval de los árbitros, la vida social de la demostración puede aumentar drásticamente ya que una vez aprobada, su difusión aumenta y su absorción se cataliza, pues la ratificación no es la muerte del mensaje, al contrario, su *verdadera* vida comienza después de ésta. Los mensajes no trascienden porque una academia de la lengua revise su correcta escritura, sino porque los practicantes de la lengua lo incorporan a su bagaje cultural, extraen su contenido y lo parafrasean, generalizan y lo conectan con otros mensajes o conjuntos de ellos. ¿Cómo respaldar esta construcción social del convencimiento? Con ejemplos históricos sobre la falibilidad del proceso de validación de una demostración, v.gr. L&D&P mencionan el ya citado caso de la demostración de Kempe de la 4CC, fisuras que exhiben su trasfondo social. O con ejemplos recopilados de la historia aún candente, sucesos situados en el pasado colindante con el presente, localizándonos en la salvaje frontera de la expansión del conocimiento matemático. L&D&P reseñan la extraña historia de terror epistemológico acaecida en la topología en fechas cercanas a la publicación de su artículo. Dos grupos de investigadores independientes, uno norteamericano y el otro japonés, anunciaron de manera aislada resultados contradictorios sobre grupos de homotopía. Para solucionar la polémica, ambos grupos intercambiaron sus demostraciones con la esperanza de refutar la demostración de sus contrincantes. Sobra decir que ambas demostraciones eran demasiado intrincadas y complejas como para esperar una solución rápida de la disputa. Para colmo de las extravagancias, ni los norteamericanos ni los japoneses hallaron algún error en la demostración de sus adversarios. La discusión se dio por terminada cuando un tercer grupo elaboró una nueva demostración a favor de lo conjeturado/demostrado por los norteamericanos. El mensaje norteamericano encontró un eco en el mensaje del tercer grupo, fortaleciendo su poder de convencimiento. Evitando al amarillismo y para no depender exclusivamente del escándalo, L&D&P venden boletos para ser espectadores de la dinámica diaria ocurrida en los departamentos de matemáticas, donde las conjeturas nacen envueltas con el humo de los cigarrillos y el café hirviente, los problemas son rumores en los pasillos, las demostraciones se gestan a varios manos y se plasman tanto en pizarrones como en papel higiénico. ¿Garantías formales? ¿Subsisten las demostraciones porque son *formalizables*? Cuento de hadas para dormir inquietudes. Las pruebas formales tienen una rudimentaria presentación inhibidora de la comunicación. Son toscas, gélidas, rígidas, frígidas, aunque de manera ideal, son un hipotético sustrato de la validez de las demostraciones:

Después de la suficiente absorción, luego de bastantes transformaciones, generalizaciones y vinculaciones, después del suficiente uso, la comunidad matemática decide finalmente que los conceptos centrales en el teorema original, ahora quizás grandemente cambiados, han alcanzado una última estabilidad. Si las diversas demostraciones parecen correctas y los resultados son examinados desde bastantes ángulos, entonces se considera finalmente

que la verdad del teorema ha quedado establecida. Se piensa que el teorema es verdadero en el sentido clásico - es decir, en el sentido que sostiene que podría demostrarse mediante una lógica deductiva, formal, aunque para casi todos los teoremas tal deducción no haya sido construida ni nunca será elaborada.[29]

La *sociabilité* de la(s) demostración(s) labran la confianza en el teorema con quien están emparentadas. Las pruebas formales, las demostraciones avaladas y alabadas por la corriente filosófica clásica, no son solamente escasas y de lectura repugnante, también son confidentes del error y en cualquier momento pueden revelar sus secretas equivocaciones. En contraposición con la postura clásica, L&D&P elogian a la visión probabilística: las demostraciones irradian una certeza no absoluta, mientras más largas o rebuscadas, titilan con mayor intensidad las dudas. De donde el convencimiento de la sucesión de demostraciones y la confianza en su teorema dependa irremediablemente de procesos sociales. Las verificaciones formales por su larga, tediosa y ampulosa conversación no pueden ser personalidades del *jetset*, es más, usualmente a lo sumo gozan de la amistad de un lector, quien también funge como su autor o coautor (v.gr. si es asistida por programas):

Las verificaciones no son mensajes; una persona quien con entusiasmo vaya a la sala de reuniones para comunicar su última verificación rápidamente recibirá como respuesta el desinteresado silencio reservado para los parias sociales. Realmente no pueden leerse las verificaciones; un lector puede rebanarse los sesos y librar los escollos de una de las más cortas a fuerza de un esfuerzo heroico, pero ese acto difícilmente puede considerarse como una lectura. Siendo ilegibles y - literalmente - inexpresables, las verificaciones no pueden ser absorbidas, transformadas, generalizadas, usadas, conectadas a otras disciplinas, y finalmente incorporadas a la conciencia comunal. Ellas no pueden adquirir la credibilidad gradualmente, como un teorema matemático lo hace; la creencia en ellas es ciega, como una pura manifestación de fe, o es visiblemente inexistente[30].

La avanzada de D&L&P busca no descuidar su retaguardia ni desaprovechar otros flancos débiles de la posición enemiga. Nunca acusan a la compleja profundidad de lo expresado como causa de la ilegibilidad de las verificaciones formales. Son un texto con un fondo simple pero con una forma indigesta, larga y confusa. Por lo tanto, a partir del principio del divide y vencerás se puede montar un poderoso contraataque. Si en el fondo su gigantesca extensión es la causa de su exclusión social, hacer de las verificaciones formales suma de las verificaciones parciales de los módulos constituyentes de los programas facilitaría su lectura, comprensión y aceptación. Para esquivar estas balas D&L&P apelan de nuevo a la observación de los usos y costumbres de los programadores de grasa y huesos:

... nosotros argüimos que el mundo de la producción de software es discontinuo. Ningún programador afirmaría que los sistemas grandes están compuestos nada más por algoritmos y programas pequeños. Los parches, las construcciones ad hoc, vendas y torniquetes, campanillas y silbatos, pegamento, escupir y pulir, el código firmado, la sangre-sudor-y-lágrima. y, claro, el fregadero de la cocina - la jerga vívida del

programador practicante parece estar diciendo algo sobre la naturaleza de las estructuras con las que trabaja; quizá los teóricos deberían escucharlo[31].

Por otro lado también se cuidan de las zarpadas de las quimeras. Si la mecanización de la verificación formal llegara a producir verificadores automáticos, talvez la exigencia por la legibilidad y sociabilidad de las verificaciones saldría sobrando. Los programas verificadores se contraerían en un centro gravitacional masivo, atrayendo toda la confianza hacia sus indescifrables veredictos. Sin embargo, una sola falla en el centro gravitacional podría provocar un cataclismo. Este fenómeno catastrófico fue referido por D&L&P como el efecto Titanic: la tragedia es proporcional a la intensidad de la creencia en la infalibilidad del sistema. Bajo una perspectiva menos novelesca, siempre quedará en duda la reputación de la solidez de los verificadores automáticos, pues ¿quiénes juzgan y convalidan a los jueces? ¿Ellos mismos? Yo soy infalible, yo soy incuestionable. Su confianza se arrodillará ante mis designios, santificarán a mis verificaciones formales. Soy su salvador y tirano, soy el sí/no divino. Para perpetrar el deicidio, D&L&P matan al dios verificador con los pecados de la criatura. El ser humano hereda sus defectos a sus creaciones, lo que del hombre procede, tendrá equivocaciones. Supongamos al verificador programado por un benigno alienígena como el tierno y faloforme E.T. Aunque fuera infalible, ningún programa hecho por un humano llevado al altar del verificador automático pasaría la prueba. Un dios sin utilidad condenado está al olvido. ¿Cuáles son los ineludibles manantiales del error donde se baña todo programador? En la cita previa se encuentran uno, la programación está plagada de código *ad-hoc,* cuya presencia está justificada porque atiende o corrige algo en el programa sin contar con un respaldo riguroso de su inclusión, funciona intuitiva-experimentalmente y se acabó. Por ejemplo, D&L&P pescaron unas palabras adversas a la verificación originadas por su eminencia Hoare:

En muchas aplicaciones, los algoritmos son casi prescindibles y este elusión origina casi ningún problema.[32]

D&L&P se filtran por otro resquicio de la muralla defensora de la verificación formal, exhibiendo otra fuente de la imperfección: el proceso de la especificación. Por un lado los requisitos a ser cumplidos por un programa nacen informales y su traslado a un lenguaje formal puede ocasionar errores por su carácter informal. Por otra parte, hay otro problema vaticinado desde la primera sección de este capítulo:

La demostración formal de la consistencia de un programa con sus especificaciones sólo tiene valor si se derivan la especificación y el programa independientemente. En el anaquel de los programas de juguete administrado por los teóricos de la verificación, este criterio se cumple fácilmente. Pero en la vida real, si durante el proceso de diseño un programa falla, se cambia, y los cambios están basados en el conocimiento de sus especificaciones; si las especificaciones se cambian, esos cambios están basados en el conocimiento del programa ganado a través del fracaso. En cualquier caso, el requisito de tener una dualidad independiente para establecer la comparación contra cada uno de sus polos ya no se satisface.

Nosotros esperamos que nadie sugiriera que las especificaciones y los programas no deban modificarse repetidamente durante el proceso de diseño. Esa sería una posición de pobreza extrema - la clase de pobreza ocasionada, según tememos, por la infatuación de la lógica formal-.[33]

Más allá de su acerbo rechazo al despotismo de la lógica formal –paradójicamente su desprecio tiene visos fundamentalistas- el problema de le especificación puede desarrollarse todavía más. El computólogo danés Peter Naur en el artículo *"Formalization in Program Development"*[34] –artículo tres años más joven que *"Social.."*- aconseja tener cuidado con el arrogante ímpetu formalizador de las especificaciones, porque:
(a) La formalización se construye y adquiere su significado por medio de su vinculación con un ambiente informal –intuitivo, experimental- y la transición entre ambos mundos es más bien informal.
(b) La amplia gama de problemas en cuya solución puede intervenir algún programa o provoca un aumento desmedido en el vocabulario del lenguaje formal de la especificación o explota el desorden de Babel con la multiplicidad de lenguajes a la medida o por conservar la economía y la uniformidad las redacciones se hacen largas y enredadas
(c) Los lenguajes formales aceptados no incluyen diagramas, tablas, etc. modos de expresión útiles en varias áreas.

Del inciso (a) se infiere la posibilidad de error presente en cualquier acto informal, de los incisos (b) y (c) fluye la siguiente crítica:

Enfrentados con las dificultades de expresar los caminos hacia las soluciones requeridas usando los lenguajes de programación establecidos, los abogados de la llamada especificación formal claman ofrecer una ayuda a los turbados programadores. Según sus ideas el programador primero debe especificar la solución en un lenguaje formal, segundo, debe expresar cómo conseguir esa solución en un lenguaje de programación, y tercero, demostrar que los resultados del programa correspondan con esa solución. De la discusión anterior [incisos (b), (c)] debe estar claro que este acercamiento ofrece escasa ayuda al programador y en cambio le agrega nuevas cargas a su trabajo.[35]

Brian Cantwell Smith hurga con mayor profundidad en la llaga y esparce la sal de la duda en la estructura origen de las especificaciones, ya sean formales o informales, en su ensayo titulado *"Limits of Correctness in Computers"*[36].
¿De donde surgen las especificaciones? Responde Smith: Éstas se desprenden de un modelo del problema a resolver. Si el modelo invoca al mundo material, irremediablemente deberemos conformarnos con aproximaciones de la basta
–impráctico sino es que imposible abarcarla toda en la representación- y misteriosa –bajo su manga esconde varias sorpresas- realidad circundante. De este modo, toda verificación a lo sumo tendrá una veracidad relativa a su modelo, por lo cual ante situaciones no previstas en éste los programas pueden fallar aun cuando posean un certificado formal. Todavía peor, Smith cuestiona la variedad de especificaciones y los varios niveles de error presentes en cualquier problema y en su solución digital. Por un

lado son diversas y numerosas las personas involucradas en un proyecto de software/hardware, desde los clientes, diseñadores, ingenieros –en electrónica, en mecatrónica, sistemas, etc-, administradores, programadores, etc., cada uno de los grupos involucrados pueden apreciar de manera distinta las intenciones del sistema construido de manera conjunta. Aun si se lograra homogenizar las especificaciones de todos los sectores partícipes de la elaboración del sistema, todavía falta el veredicto final, el emitido por el usuario. Si ser bonito y fácil es sinónimo de ser correcto, para varios chimpancés –me incluyo- el sistema operativo Windows cumple con su especificación; las temidas pantallas azules mensajeras del error, son motivos para la degustación estética y no para la preocupación patética Por otro lado los errores aparecen en varios niveles, desde el hardware hasta el software producido. Aun si se verificaran todas las capas del sistema, hay un elemento fuera del alcance de la verificación formal, a saber, los usuarios. En un principio podría reducirse la participación del usuario hasta casi la nulidad, pero en la práctica el *correcto* funcionamiento de la mayoría de los programas depende de la experiencia y hasta del sentido común de quienes lo manejan. La concepción y la utilización de los sistemas computacionales son el fruto de una interacción teórica/técnica pero también social, ni el hardware ni el software pueden ser reducidos a la mera aplicación de teorías electrónicas ni algoritmos. La verificación formal si en verdad quiere ganarse su mote de infalible, le corresponde solucionar otra clase de problemas relacionados con la fundamentación de la solidez de los modelos o con la inclusión de factores sociales y según lo sugiere Smith, de estos nuevos problemas surgen algunos límites insuperables.

Alto. Ni a los programas para generar demostraciones ni a los programas para verificar pruebas les quedan todos los sacos curtidos con las críticas hasta aquí expuestas. En especial y en general, las especificaciones no son semillas de discordias, tampoco lo son los modelos. Nos interesa generar sucesiones de cadenas de signos con algunas propiedades mediante ciertas reglas (o series de operaciones) o inspeccionar que ciertas cadenas de signos obedezcan las reglas de derivación aceptadas y los planos de construcción tolerados. No hay lugar para sorpresas ni entredichos, todo ha sido dicho, dicha implícita, de manera explícita. Por lo tanto, basta con concentrarnos en tres bombas de la ofensiva de D&L&P:

- La sociabilidad de una demostración ocasiona su convencimiento.
- Las verificaciones formales son antisociales, ergo carecen de convencimiento.
- Los programas no son la suma de programas más pequeños pretendientes de verificaciones formales más amenas y por lo tanto más sociables porque en la práctica los programas están minados por instrucciones *ad-hoc*.

Empiezo desarmando al último dispositivo explosivo. El problema atendido por los programas verificadores de pruebas es sencillo, por lo que su implementación en el lenguaje adecuado puede ser relativamente corta y puede estar constituida por módulos libres de los injertos de código temidos. Es decir, estos programas son los candidatos mejor posicionados para recibir la purificación de la verificación formal. Es decir, la pieza clave del armacabezas de las pruebas computarizadas completas puede ser colocada. Los

programas demostradores del estilo del capítulo tercero pueden concentrarse en la difícil búsqueda de pruebas y relegar la decisión sobre la veracidad de sus hallazgos a otros agentes mejor capacitados y en el mejor de los casos certificados deductivamente. El foco donde han irradiado la mayoría de las soluciones a problemas abiertos mediante sistemas resolutivos/paramodulativos, Argonne, ha escogido esta opción. El programa Ivy de McCune, ya cuenta con una demostración sobre su correcto funcionamiento para modelos finitos (ref. 19). Mizar y proyectos afines podrían seguir el mismo camino. Algunos demostradores de alto nivel, v.gr. ACL2, podrían construir circunstanciadamente pruebas aptas para ser verificadas por otros programas. Cuidado, indico posibilidades viables y no deberes inevitables, aunque si la confianza de la comunidad matemática es la meta, el deber persevera. El argumento global de la discontinuidad en la programación está basado en los usos y costumbres de los programadores, pero estás prácticas constituyen uno de los objetivos mejorables por medio de la *matematización* de esta actividad. Maurer le concede a D&L&P que si bien una pequeña modificación puede hacer de un programa correcto una aberración inexpugnable para la verificación formal, eso no afecta el principio crucial: las verificaciones formales de un programa preservan su veracidad al incrustar este programa en otro de manera controlada y cuidadosa. Por lo que, si el rigor y el orden son las máximas obedecidas al elaborar un programa:

Si el programa se divide en un programa principal y n subrutinas, tendremos n+1 verificaciones que hacer, y eso es todo lo que tenemos que hacer para demostrar el correcto funcionamiento del programa.[37]

Si los hombres se amasen los unos a los otros, el paraíso en la tierra podría amasarse con la suma de nuestros nobles actos. Aunque es un bello ideal, la supervivencia es sorda, talvez sea más fácilmente escuchado en la existencia suspendida en el formol de los conventos, los monasterios o los manicomios (aunque su audición sea sólo el preludio de la burla y el escarnio). De igual modo, algunos académicos acatarán las sugerencias de Maurer, pero en las aplicaciones comerciales e industriales, la supervivencia también impera. Lo cual nos remite otra vez a focalizar los esfuerzos en los programas más simples, los programas encargados de verificar a las pruebas.

Corresponde desactivar al segundo argumento detonante. ¿Qué tan inadaptados, marginales, antisociales son los esperpentos deductivos digitales? De los tres agresores, Richard de Millo era quien contaba con mayor experiencia en el área de la verificación formal, su tesis de doctorado versaba sobre la construcción de una semántica de los lenguajes de programación por medio un acercamiento algebraico a la lógica e hizo algunas verificaciones de programas escritos en el lenguaje Madcap[38]. Testigo y actor del "calvario" de la verificación, junto con Perlis y Lipton confesaba:

Las verificaciones formales son largas y enredosas pero poco profundas; ése es lo que está mal con ellas. La verificación de incluso un programa trivial puede extenderse en docenas de páginas, y no hay un momento de alivio o una chispa de ingenio en cualquiera de esas páginas.[39]

Para tener éxito en los atentados terroristas mediáticos, basta con el correcto funcionamiento de uno de los mecanismos explosivos. Si la continuidad en la programación ceba al tercer argumento de nitroglicerina, la cuenta regresiva del segundo inalterada sigue su marcha. Aunque la verificación formal de un programa pudiera descomponerse en la verificación de n subprogramas triviales, cada una de éstas probablemente sea una secuencia de derivaciones horripilante. En el artículo *"Towards Inferential Programming"* de William Scherlis y Dana Scott, se encuentra una frase de semblante análogo a la cita previa:

...incluso los programas relativamente elementales tienden a ser más complicados que los teoremas elementales. ¿Por qué? Porque en un cierto sentido una mayor parte de su estructura tiene que ser hecha explícita. Una demostración es a menudo más una guía turística que un conjunto de instrucciones para encontrar un tesoro. Los programas, por otro lado, no sólo operan con datos altamente estructurados sino deben hacerlo de maneras nada obvias para ser eficaces.[40]

Si se quiere realizar una verificación formal, deben clarificarse y formalizarse esas maneras nada obvias presentes hasta en los programas más cándidos, el fin del misterio puede dar inicio a la peor de las pesadillas, aquella en donde los alaridos del miedo han sido sofocados con los bostezos del tedio. En 1976 se asomó a la luz pública una gigantesca encarnación de los temores digitales de D&H&P y contrario a sus aprensivos estimados, el monstruo tuvo una agitada vida culminada con los cuernos del convencimiento, tal como lo mencionan Scott y Sherlis varios párrafos después de aquél donde habita su cita previa:

- aunque ellos mencionan la turbia historia temprana de las "soluciones" de la Conjetura de los Cuatro Colores- De Millo, Lipton y Perlis no discuten la demostración subsecuente asistida por una máquina de esta conjetura de fama mundial. (Su papel fue recibido por los editores en el octubre de1978 y probablemente escrito mucho antes, el primer anuncio de la solución de Haken, Appel y Koch de la conjetura fue hecho en septiembre de 1976 y el posterior artículo se publicó en septiembre de 1977)[41]

Posteriormente Scott y Sherlis transitan fugazmente por algunas opiniones de Tymoczko y Swart –parcialmente reproducidas en el segundo capítulo de esta tesis- aunque dejan asentado que la demostración de H&A&K&1&0 tuvo una ajetreada agenda social hasta llegar a su baile de iniciación, en donde la sexagenaria conjetura dejo de ser una virginal duda para convertirse en un teorema de la vida pública. Con sarna Scott y Danna recalcan las fechas de publicaciones para dejar sin coartada a los tres ofensores de la verificación formal, agresores cuyas principales armas argumentativas son dardos históricos y sobretodo aquellos remojadas con el fresco veneno de la historia viviente. Veneno mata veneno. Aunque difieran las verificaciones formales de la demostración computarizada del lema de reducibilidad, dicha demostración muestra la peor cara posible de los males atribuidos por D&L&P a las verificaciones formales. Es un objeto deductivo libre de una lectura larga y pesada pues ni siquiera es legible para un ser humano; son miles y miles de operaciones en una carrera frenética laureada en la meta por el vaivén de la bandera con la

siguiente leyenda: “la configuración dada como entrada si es C-D-reducible”. Con maña podemos cambiar la frase de bienvenida al vencedor por “el programa dado como entrada ha sido verificado formalmente” para transportarnos a la sima del horror descrita por D&L&P. En aquel abismo donde la mente sin ayuda de la computadora se asfixia, existe la vida social. En aquellas profundidades donde reptan las criaturas con escamas de ceros y unos cuyas colas tienen una longitud potencialmente infinita, el convencimiento puede desarrollarse socialmente tal y como lo postulan D&L&P. A menos de que la respiración artificial sea rechazada –decisión ni siquiera tomada por la comunidad matemática, organismo que D&L&P tanto se jactan de conocer– las verificaciones formales tienen acceso a la vida, pueden aspirar al convencimiento.

Donde debí comenzar termino. En las matemáticas el convencimiento (se quedaron a menos dos dedazos de cegar a la sílaba “ven” con el coyuntural “o”) es una construcción social según De Millo y compañía. Leamos la objeción de un filósofo de profesión:

En definitiva, aunque los procesos sociales son cruciales para determinar qué teoremas la comunidad matemática considera verdaderos y qué demostraciones considera válidas, estos procesos por sí solos no los hace verdaderos ni válidas respectivamente [42]

El autor de la oración anterior es James Fetzer, quien alrededor de diez años después de la punzada de D&L&P abrió de nuevo la herida con el ensayo *“Program Verification: the Very Idea”* aparecido en la misma afamada publicación de ACM.
Astutamente Fetzer primero buscó eliminar a la competencia, según él la crítica de D&L&P aunque mal planteada apunta con tino a la desacralización de la verificación formal, por lo cual en la primera parte de su escrito socava la postura sociológica sobre la aceptación de los teoremas y demostraciones. Un mal similar al astigmatismo padecen aquellos quienes ven como una distorsión cualquier perspectiva social del desarrollo matemático, pero el otro extremo visualizado por D&L&P revela una miopía al carecer de un nítido enfoque del contenido de los mensajes. Fetzer rebana con la navaja filosófica la cornea de los ojos discursivos de Lipton et al, cercenado al creer del conocer:

Después de todo, una “demostración” es una demostración por su validez y no por su aceptación (por nosotros) como válida, así como lo que hace verdadero a un enunciado es lo que afirme que sea el caso sea el caso, no meramente lo que se cree (por nosotros) y por consiguiente llamemos verdadero. Por consiguiente, el proceso social no es necesario ni suficiente para que una demostración sea válida, como DeMillo, Lipton y Perlis implícitamente conceden.[43]

Una creencia para ganarse el estatuto de conocimiento debe satisfacer ciertas condiciones necesarias y suficientes dependiendo de la naturaleza del dominio de la creencia-conocimiento, Fetzer puntualiza. Así, los resultados de la lógica y las matemáticas son establecidos por métodos deductivos y requieren de demostraciones.

Soplan los vientos filosóficos, aunque hasta el momento como un remolino, una vuelta conduce a otra vuelta, espirales del desatino. Para escapar de las agruras circulares, el filósofo (alca)Fetzer impone una caracterización de las demostraciones válidas:

Nada más se le puede exigir apropiadamente: su conclusión debe ser verdadera si todas sus premisas son verdaderas, esta propiedad determina la validez de una demostración.[44]

De este modo, Fetzer nos guía hacia la fuente eólica común, nos remite hacia la lógica. La anterior propiedad es denominada por los miembros de esa disciplina ser consecuencia lógica. Fetzer evita ensuciarse con el fondo fangoso y encerrarse en una forma antiséptica, en ningún momento indica cómo le hacen los matemáticos para cumplir en sus demostraciones con su caracterización ni tampoco tajantemente enaltece a las pruebas formales como las únicas válidas. Para bien o para mal, De Millo y sus secuaces describieron un cómo aunque pasaron por alto el problema de la especificación de los mensajes, es decir, ¿qué tienen las demostraciones (mensajes) para que en una primera instancia un individuo y después un grupo de ellos puedan engancharse por su convenc imiento? Tymoczko en conjunción con Fetzer contestarían: por sus curvas semánticas –epistemológicas- y líneas sintácticas –lógicas- las cuales deben trazar la verdad de las premisas a las conclusiones. Los objetos procesados en el engranaje social descrito por D&L&P pueden ser tanto demostraciones como mitos, tanto rumores como ritos. Fetzer implícitamente acepta la socialización de las matemáticas, aunque en un principio una demostración pueda ser válida por su contenido, en la práctica matemática se necesitan de lectores/editores para depurar, fijar y esclarecer la ruta desde la verdad de las premisas hacia la verdad de las conclusiones. Fetzer remarca la meta, los esfuerzos se enfilan hacia el conocimiento de un teorema y una de las vías para acceder a él es por medio de la comprensión de su(s) demostración(es). Y la comprensión de una demostración ocurre en un nivel individual pero también se construye por medio de una interacción social. La leyenda del matemático solitario ya no galopa en las llanur as de las matemáticas desde hace mucho tiempo, o parafraseando la advertencia de Tymoczko transcrita al final del segundo capítulo, ya no debe ni trotar. Contrario a las opiniones de aquellos quienes repudien todo lo apellidado “social”, la estructuración social de las matemáticas ha sido un marco favorable para su desarrollo saludable.

En teoría Fetzer invalidó la madre de todas las bombas de la crítica de D&L&P contra la verificación formal (la sociabilidad de una demostración/verificación ocasiona su convencimiento), pero en la cotidianidad es un estallido inminente. Pues, aunque una verificación formal fuera válida, a su autor lo pueden acusar de solipsismo o llanamente admirarlo con la total ignorancia de su abultado trabajo y hacer caso omiso de sus resultados. Por lo que la refutación de la condición inherente de ser parias en las verificaciones formales adquiere un mayor peso. Nada inexpugnable impide el tránsito de las verificaciones formales en los circuitos sociales análogos a los recorridos por las demostraciones, nada siempre y cuando se tenga confianza en el instrumento computadora. Si bien las verificaciones formales sufrirán de otros factores inhibidores de la circulación, como lo son el apoyo económico o el prestigio del área, estos factores afectan a todas las

demostraciones hoy en día. Así entonces la confianza en las acciones de la computadora es el elemento crítico, porque como bien señalan D&L&P, solos perecemos en las garras de las horripilantes y tenebrosas –por sus desmedidas proporciones y escasa luz de ingenio- verificaciones formales. Y como ha sido expuesto desde la demostración computarizada exhaustiva de H&A&K&1&0 y las pruebas computarizadas del capítulo tercero, la asistencia de la computadora en varias etapas del trayecto (generación/ratificación/comprensión) es un mal necesario. Es un mal si pretendemos a la perfección pues, ¿acaso no inspiran dudas y recelos las autoproclamaciones de infalibilidad? Los dientes en las fauces lamen a la cola. Desciendan y corten al pomposo proyecto. Afilemos a los colmillos filosóficos y desprendámonos del lastre de los sueños de grandeza. Si bien seguiremos tropezando, será por otras torpezas y no por atorarnos absolutamente con la cola.

Regresando al ejemplo aventado desde la Torre Latinoamericana, ¿verdaderamente los dos cuerpos caen al mismo tiempo en el vacío? Sí, mientras nos situemos en las tierras de la abstracción. Es decir, si aceptamos a la teoría newtoniana –o al espacio de Minkowsi- la afirmación de la caída simultánea se sostiene. Si un físico padece la comezón provocada por la enfermedad de la realidad, a lo mejor intenta aliviar al prurito montando un experimento para reproducir *en la medida de lo posible* las condiciones de la aserción. Para rascarse y atender a su intoxicación el físico consigue una cámara con una bomba turbomolecular y construye un dispositivo (pasamuros) en el cual no se fuga el "vacío" -parte del espacio en la cual no hay materia ni energía que interfiera con las condiciones del experimento- para provocar la caída y además él para sus fines aprovecha el campo gravitacional de la tierra. No obstante, el físico de indias sabe de antemano que su experimento será una aproximación, por medio de dispositivos y objetos materiales recrea con cierto grado de precisión las condiciones del vacío reduciendo más no eliminando la interferencia del mundo material en su experimento. Entre paréntesis, ojalá también sea consciente de la utilidad e inutilidad de su proceder, pues el rascarse origina más comezón. La realidad impone sus condiciones y quienes trabajan con ella las sufren aunque no faltará quienes le saquen provecho. Avra Cohn, quien ha interactuado con el mundo físico al investigar y efectuar la verificación formal de hardware, en el artículo *"Notion of Proof in Hardware Verification"* (contemporáneo al de Fetzer) recopiló algunas de sus reflexiones en torno a las ventajas y alcances de la verificación del hardware. En las primeras líneas del ensayo, puntualiza:

... la verificación del hardware es en muchas maneras más dócil que la del software -es a menudo más fácil de escribir una especificación clara que captura la funcionalidad de un sistema de hardware que de software - y las pruebas del hardware tienden a tener una cierta uniformidad de estructura propicia para el tratamiento mecánico.[45]

Después de las gracias arriban las desgracias (para algunos):

Un dispositivo puede describirse de una manera formal, y la descripción puede ser verificada; pero, hay ninguna manera de asegurar la exactitud de la descripción. De hecho,

cualquier descripción está ligada a la inexactitud en algunos aspectos, ya que no puede esperarse que refleje una situación física entera incluso en un instante, mucho menos a través del tiempo cuando evolucione; el modelo de un dispositivo necesariamente es una abstracción (una simplificación).[46]

Con el mismo tono de las observaciones de Smith, Cohn señala una vicisitud en la verificación formal del hardware provocada por retozar en el mundo material. Utilizando la terminología de Smith, toda verificación de hardware será relativa y su validez estará relacionada con el modelo del dispositivo y no con el dispositivo real. La observación de Hoare permanece vigente, el correcto funcionamiento del programa afirmado por su verificación formal SIEMPRE estará supeditado a los caprichos de la realidad asoladores de los componentes electrónicos de la computadora. Fetzer da la estocada final al trasladar a la incertidumbre física hasta el alma de la computadora, los programas:

La idea misma de la verificación formal de un programa es un equívoco. Si interpretamos programa como la codificación de algoritmos, entonces ninguna dificultad especial se alza cuando "verificamos" que la salida O se sigue de la entrada I como una consecuencia lógica de axiomas de la forma I c O [I causa O, o con la notación de Hoare, I{c}O]. Bajo tal interpretación, sin embargo, nada afirma la verificación de un programa acerca de la ejecución del programa en cualquier máquina física. Interpretado programa como la codificación de algoritmos que puede ser compilada y ejecutada, sin embargo, que la salida O se obtenga de la entrada I como una consecuencia lógica de los axiomas de la forma I c O confirma nada de manera absoluta porque la verdad de estos axiomas depende de propiedades causales de sistemas físicos cuyo presencia o ausencia sólo es comprobable por medio de procedimientos inductivos.[47]

Fetzer dice nada nuevo, en la formulación concisa radica su contundencia (dicho sea de paso en su artículo se abstuvo de emplear un lenguaje más filoso (fico) considerando la formación académica de sus lectores). Los lenguajes de programación caracterizan máquinas abstractas, máquinas en donde toda instrucción puede ser interpretada/obedecida fidedignamente. En esta dimensión puede reinar el paradigma duro de la *matematización* defendido y definido en cuatro incisos por Hoare. Para aquellos –y los hubo- cuyas pretensiones flotaban hacia la materialización de la perfección en los programas ejecutados por computadoras, Fetzer les recuerda la gravidez de la materia. Los cuatro incisos del paradigma duro son rescritos por el filósofo añadiendo en cada uno el adjetivo “aplicadas”, v.gr. la programación es una actividad de las matemáticas aplicadas. El paradigma no se abandona, se ablanda con los martillazos de la realidad.

Al final de tantas creencias y críticas destructivas, los defectos y cualidades mencionados en la inicial frase lapidaria de Dijikstra se conservan. La verificación formal no demuestra la ausencia total de errores, pero puede ser una herramienta bastante poderosa para depurar a los programas y corregir el diseño de los dispositivos electrónicos. Perdida la esperanza en el mesías de la verificación formal, la confianza en los sistemas computacionales debe instaurarse por varios agentes y sujeta a diversos factores. Por

ejemplo, puede complementarse el probado experimental con la verificación formal, la manera natural de hacerlo es intentar la verificación formal en todas las partícipes de la ejecución del programa viables para este método (programa, compilador, sistema operativo, diseño de algunas de las capas de hardware) y probar experimentalmente a los componentes electrónicos del sistema (microprocesador, memorias, buses, etc.).
Si la verificación formal hubiera sido el mesías esperado, tampoco las soluciones del absoluto habrían salvado a los mortales de sus cuitas binarias. Desde la primera sección se advirtió, las verificaciones formales son una pesada cruz para los interesados. Es un agobio realizarlas, inspeccionarlas por nuestros propios medios es un quebranto. Las pecaminosas aplicaciones comerciales, industriales o académicas cuyos propósitos requieren códigos más extensos e intrincados son ariscas a la evangelización. Independientemente de su carácter divino, las verificaciones formales hacen grandes revelaciones. Son portadoras de buenas nuevas, tanto lógicas como epistemológicas. Por un lado pueden asegurar el correcto funcionamiento del programa sometido a los caprichos de los demás elementos del sistema no verificados y en última instancia del hardware o al grado de exactitud del modelo que incorpore variables físicas. Aunque para recibir esta agradable noticia, probablemente la verificación de la verificación formal sea efectuada otra vez por un objeto digital aclimatado a esas burdas tormentas derivativas y no por el sujeto usuario quien fácilmente es acribillado por esa torrencial lluvia de lo simple; todo bien exige un sacrificio. Por otro lado pueden fomentar un conocimiento anatómico del programa, una descripción meticulosa de lo que realmente hace el programa o incluso de lo que realmente queremos que haga, el bisturí de la verificación formal abre las entrañas del código y nos permite encontrar errores de otra manera indetectables y descubrir objetivos o impedimentos inapreciados en la especificación original. Aunque para sondear en las vísceras epistemólogicas, probablemente se recurra al apoyo del instrumento; la longitud y la aridez retortijada de los intestinos dificulta recorrerlos por entero solos.

La costumbre genera confianza. Si un programa después de su periodo de pruebas y/o tras ganar su diploma de verificado en su uso durante un largo lapso de tiempo acontece ninguna falla, entonces la confianza en el programa se puede extender paulatinamente entre sus usuarios quizás hasta constituirse como una fe en los digitales milagros. Y en efecto, a lo mejor el programa nunca se equivoca. Y en concreto, talvez el programa gracias a la acción en conjunto de la verificación formal y el probado experimental tenga una probabilidad de falla extremadamente baja. Sin embargo, la confianza siempre será una cuestión de fe, por más racional que ésta sea.

Las demostraciones adquiridas por medio de programas y las verificaciones de pruebas formales ejercidas por programas en su nacimiento están envueltas por el humo de la incertidumbre. Someterlas a procesos de desintoxicación parecidos a los acontecidos con las demostraciones normales no es una alternativa, es la opción. Sin embargo, los procesos de validación eventualmente llegan a un callejón cuya única salida es confiar en el instrumento, tal como los matemáticos confían en sus compañeros. La confianza tampoco es fortuita. Así como hay elementos para depositar con mayor medida la confianza en un colega matemático, v.gr. su trayectoria académica, también los sistemas computacionales

poseen ciertos mecanismos -tanto teóricos como sociales- reforzadores de la confianza, v.gr. si han sido verificados formalmente o si han resuelto problemas abiertos. Sin embargo, por más certificados, recomendaciones y garantías que posean, los asistentes digitales deductivos adquirirán el grado de convencimiento equivalente al de sus patrones siempre y cuando estos sistemas computacionales hayan sido incrustados en el núcleo de las mecanismos deductivos aceptados pero también manejados por la comunidad matemática y recurrir a ellos sea algo común, cuando la computadora no sea apreciada únicamente como una bizarra y ocasional fuente de descubrimientos sino como una cotidiana y normal herramienta para el conocimiento.
Según lo exhibido a lo largo y ancho –aunque poco profundo- de los capítulos de la tesis, su adopción se está tramitando.

Cierro el capítulo con un par de opiniones provenientes de investigadores cuyos nombres ya desfilaron. Son un par de comentarios con ideas coincidentes aunque con una intensidad e intencionalidad diferentes. Son dos opiniones cuya temática gira entorno al grado de certeza de los programas y ambas son concluyentes.
Lipton, Perlis y De Millo dispararon las siguientes palabras de gracia en el rostro de los idealistas de la verificación formal:

Es nada más que un chauvinismo del símbolo por lo que algunos computólogos suponen que nuestras estructuras son mucho más importantes que las estructuras materiales porque pueden ser perfectas.
...En la práctica de la programación no debe permitirse que la verificabilidad ensombrezca a la confiabilidad. Los científicos no deben confundir a los modelos matemáticos con la realidad y la verificación formal no es más que un modelo de credibilidad.[48]

La bala fue recibida frontalmente por Shankar Natarajan y le respondió gracias pero no gracias, yo no soy nadie para recibirte en mi cara. Después del extenso recorrido por sus demostraciones apoyadas por Nqthm, Natarajan en los estertores de su libro *Metamathematics, Machines and Gödel's Proof* se cuestiona sobre si son absolutamente correctas sus demostraciones elaboradas con la participación de su ayudante mecánico y sin vacilar responde con un tajante NO. La muerte de las pretensiones libera, en el funeral del absoluto la utilidad canta el siguiente hosanna, eco de las plegarias de Dijikstra elevadas hacia la confiabilidad en los programas:

...la asistencia mecánica es útil para encontrar y corregir errores del razonamiento pero no puede garantizar que se hayan eliminado los errores.[49]

NOTAS

(1) Hoare, C.A., *An axiomatic basis for computer programming,* Communications of the ACM, 12, 1969, 576-580.

Así empieza el artículo de Hoare:
Computer programming is an exact science in that all the properties of a program and all the consequences of executing it in any given environment can, in principle, be found out from the text of the program itself by means of purely deductive reasoning.

(2) Por ejemplo para probar el Lema 2 Hoare demuestre el teorema:

$$y \leqslant r \supset r + y \times q = (r - y) + y \times (1 + q)$$

Cuya prueba es:

$$\begin{aligned}
\text{A5} \quad & (r - y) + y \times (1 + q) \\
& = (r - y) + (y \times 1 + y \times q) \\
\text{A9} \quad & = (r - y) + (y + y \times q) \\
\text{A3} \quad & = ((r - y) + y) + y \times q \\
\text{A6} \quad & = r + y \times q \quad \text{provided } y \leqslant r
\end{aligned}$$

Luego simplemente hace la conjunción de la conclusión demostrada en el paso previo con el teorema. En el artículo original resalta la ausencia de la definición de prueba, Hoare enseña con el ejemplo dejando sin explicitar algunas reglas de inferencia aceptadas (como la conjunción) manchando un poco de informalidad su tratamiento formal de la programación, mácula justificable pues en el fondo no le importa la lógica en sí sino la lógica aplicada a la computación.

(3) Berg, Helmut et al, *Formal methods of program verification and specification,* Prentice Hall, 1982, págs. 20-21.

(4)

> The practice of supplying proofs for nontrivial programs will not become widespread until considerably more powerful proof techniques become available, and even then will not be easy. But the practical advantages of program prov-

pág 579 de(1).

(5) Antes del trabajo de Hoare hubo otros investigadores claves para el desarrollo del paradigma de la matematización de las programación, resaltan McCarthy, Floyd y Naur. El artículo de Robert Floyd *"Assigning Meanings to Programs"* (Mathematical Aspects of

Computer Science, pages 19-32, 1967) fue la influencia principal de (1). En ese artículo, Floyd indicaba como realizar una interpretación semántica lógica-matemática de las instrucciones en un programa a partir de diagramas de flujo. Floyd pregonaba por captar la asistencia de la computadora para llevar a cabo las verificaciones formales. Un alumno de él, King, siguió su consejo en su tesis de doctorado:
King, James , *A Program Verifier*, Ph. D. thesus, Carnegie Mellon University, 1969.

(6)
Hoare, C.A.R, *Mathematics of Programming*, discurso pronunciado por Tony Hoare en el Museo de la Computación en Boston para el décimo aniversario de BYTE. transcrito en *Formal Verification*, editado por Timothy Colburn y James Fetzer, Kluwer Academic Publishers, págs. 135-153

The most powerful method of solving a complicated problem is to split it into simpler subproblems, which can then be solved independently.
pág. 146

(7)
the execution of the program, then the techniques described in this paper may be used to prove the correctness of the program, provided that the implementation of the programming language conforms to the axioms and rules which have been used in the proof. This fact itself might also be established by deductive reasoning, using an axiom set which describes the logical properties of the hardware circuits. When the correctness of a program, its compiler, and the hardware of the computer have all been established with mathematical certainty, it will be possible to place great reliance on the results of the program, and predict their properties with a confidence limited only by the reliability of the electronics.
pág. 579 de (1)

(8) *However, program proving, certainly at present. will be difficult even for programmers of high caliber; and may be applicable only to quite simple program designs. As in other areas, reliability can be purchased only at the price of simplicity.*

pág, 580 de (1)

(9) Hoare, C.A.R. página 135 de (6)

(10) El método de las aserciones inductivas es una mezcla de los trabajos de Hoare(1) y Floyd (5). La tesis doctoral de King empleó este método.

(11) Boyer, Robert S., and Moore, J Strother *Program Verification*, JAR Vol. 1, no. 1 (1985) 17-23.

Boyer-Moore además de aplaudir los triunfos de las aserciones inductivas identificaron un punto de inflexión en la verificación, empero.

Because the first approach [inductive assertions] has received by far the most attention, it has produced the most impressive results to date. However, the field is now moving away from the inductive invariant approach.
pág. 17

(12)J. Moore y Matt Kaufman, ACL2 , *A Computational Logic for Applicative Common Lisp*. http://www.cs.utexas.edu/users/moore/acl2
(13) Common Lisp is the standard list processing programming language. It is documented in: Guy L. Steele, *Common Lisp The Language,* Digital Press, 12 Crosby Drive, Bedford, MA 01730, 1990

(14) *It is not easy to get ACL2 to prove hard theorems. To do so, the user must* ***understand***
** the model,*
** ACL2 as a mathematical logic, and*
** be able to construct a proof (in interaction with ACL2).*

Kaufman, Moore. http://www.cs.utexas.edu/users/moore/acl2/v2-9/What_is_Required_of_the_User_lparen_Q_rparen_.html

(15) Shankar Natarajan, *Metamathematics, Machines and Gödel's Proof*, Cambridge University Press, 1994

A surprisingly large fraction of discrete mathematics and computer science can still be expressed in this quantifier-free logic.
pág. 28

(16) En lugar de la Aritmética de Peano, Shankar optó por Z2 ,un sistema formal con el mismo poder representabilidad aritmética, para montar el teatro de la incompletud de Gödel. Z2 es la teoría de los conjuntos hereditarios de lo finito, i.e. aquellos de cardinalidad finita donde todos sus elementos, sus elementos de sus elementos, sus elementos de sus elementos de sus elementos,... son finitos (destierra al axioma del conjunto inductivo de ZF. axioma donde se establece la existencia de conjuntos infinitos). La elección se debió entre otras cosas al manejo de los pares ordenados proporcionado por Nqthm (los conjuntos pueden ser representados transparentemente como listas). En (15) se describe la demostración del teorema de incompletud de Shankar-Nqthm, en ese libro además demuestra otro resultado metamatemático, la consistencia del cálculo lambda establecida por el teorema Church-Rosser.

(17) Kunen, K., *A Ramsey Theorem in Boyer-Moore Logic,* J. Automated Reasoning 15 (1995) 217-235.

Ramsey's Theorem states that one can, for each k, n, c, find a number $R(k, n, c)$ large enough so that whenever we partition all the n-tuples from the set $\{0, 1, 2, \ldots R(k, n, c)\}$ into c pieces, there is a set of size at least k which is *homogeneous* for the partition (that is, all of its n-tuples are in the same piece of the partition). The Paris-Harrington extension adds the seemingly harmless extra requirement that the homogeneous set be *large*, meaning that its size is at least as big as its first element. For example, $\{3, 5, 7\}$ is large but $\{4, 5, 7\}$ isn't.

Now, it is easy to see that Ramsey's Theorem can be proved within PRA, and the $n = 2$ case has been verified on Nqthm by Matt Kaufmann (see [1]). However, the Paris-Harrington extension cannot be proved even in full PA [5]. It can be proved in PRA plus ϵ_0 induction, as was demonstrated explicitly by Ketonen and Solovay [4], and hence, potentially, can be proved in Nqthm.

We have indeed verified this theorem on Nqthm, and describe the proof in this paper.

Donde PRA=ARP= Aritmética Recursiva Primitiva, PA=Aritmética de Peano, aritmética de los naturales definida por los axiomas de Peano escritos originalmente en latín:

1. el 0 es un número
2. El sucesor inmediato de un número también es un número
3. 0 no es el sucesor inmediato de ningún número
4. Dos números no tienen el mismo sucesor inmediato
5. Toda propiedad perteneciente a 0 y al sucesor inmediato de todo número que también tenga esa propiedad pertenece a todos los números (*inducción matemática*)

[1] Basin, D. and Kaufmann, M, The Boyer-Moore Prover and Nuprl: An Experimental Comparison Technical Report #58. Computational Logic. Inc.. 1990.

[4] Ketonen, J. and Solovay, R., Rapidly Growing Ramsey Functions, *Annals of Math* 113 (1981) 267 – 314.

[5] Paris, J. and Harrington, L., A Mathematical Incompleteness in Peano Arithmetic, in *Handbook of Mathematical Logic*, J. Barwise, ed., North-Holland, 1978, pp. 1133 – 1142.

La extensión Paris-Harrington del teorema de Ramsey es un indemostrable -para el sistema PA- "natural" al contexto de una teoría matemática, carece del barniz especular artificioso del espectacular indemostrable de Gödel.

La inducciones transfinitas están acotadas por ε_0, por ejemplo en algún momento de la demostración se prueba:

Lemma 14. For each $\alpha < \epsilon_0$ and $k < \omega$: $k \leq \Lambda(\alpha, k)$, and the interval $[k, \Lambda(\alpha, k)]$ is α – large.

Proof. Induct on α. ■

Los ordinales son representados en ACL2 con números naturales y con listas:

```
ordinal            ACL2 representation
  0                  0
  1                  1
  2                  2
  3                  3
  ...                ...
  w                 '((1 . 1) . 0)
  w+1               '((1 . 1) . 1)
  w+2               '((1 . 1) . 2)
  ...                ...
  w*2               '((1 . 2) . 0)
  (w*2)+1           '((1 . 2) . 1)
  ...                ...
  w*3               '((1 . 3) . 0)
  (w*3)+1           '((1 . 3) . 1)
  ...                ...
   2
  w                 '((2 . 1) . 0)
  ...                ...
   2
  w +w*4+3          '((2 . 1) (1 . 4) . 3)
  ...                ...
   w
  w                 '((((1 . 1) . 0) . 1) . 0)
  ...                ...
    2
   w
  w                 '((((2 . 1) . 0) . 1) . 0)
...                ...
    w
   w
  w                 '((((((1 . 1) . 0) . 1) . 0) . 1) . 0)
  ...                ...
```

(18) J Moore, Tom Lynch, and Matt Kaufmann , *A Mechanically Checked Proof of the Correctness of the Kernel of the AMD5K86 Floating-Point Division Algorithm*, *IEEE Transactions on Computers*, **47**(9), pp. 913-926, Sep., 1998.
Correctness of the AMD5k86 Floating-Point Division: *If p and d are double extended precision floating-point numbers (d /= 0) and mode is a rounding mode specifying a rounding style and target format of precision n not exceeding 64, then the result delivered by the K5 microcode is p/d rounded according to mode.*

J Moore, *Proving Theorems about Java-like Byte Code*, in E.-R. Olderog and B. Steffen (eds.) Correct System Design -- Issues, Methods and Perspectives, LNCS
This paper describes a formalization of an abstract machine very similar to the Java Virtual Machine but far simpler. Techniques are developed for specifying the properties of classes and methods for this machine. The paper illustrates two such proofs, that of a static method implementing the factorial function and of an instance method that destructively manipulates objects in a way that takes advantage of inheritance.

(19) William McCune and Olga ShumskyMatlin, *Ivy: A Preprocessor and Proof Checker for First-Order Logic,* Chapter 16 of Computer-Aided Reasoning: ACL2 Case Studies (Kaufmann, Manolios, Moore, eds.) Kluwer, 2000

(20) http://www.cs.utexas.edu/users/boyer/ftp/nqthm/nqthm-1992/examples/kunen/paris-harrington.proofs

(21) http://www.cs.utexas.edu/users/boyer/ftp/nqthm/nqthm-1992/examples/kunen/paris-harrington.pdf

En esta presentación aunque se eliminan las demostraciones individuales Kunen escribe algunos comentarios para impulsar la comprensión de la demostración, por ejemplo el siguiente es un extracto de la parte inicial de la prueba:

```
; (expt m n ) = m^n, as in Lisp
```

DEFINITION:
$\text{expt}(m, n)$
$=$ **if** $n \simeq 0$ **then** 1
else $m * \text{expt}(m, n-1)$ **endif**

THEOREM: distributivity
$((u * y) + (a * y)) = ((u + a) * y)$

THEOREM: associativity-of-times
$(m * u * y) = ((m * u) * y)$

THEOREM: addition-of-exponents
$\text{expt}(m, a + b) = (\text{expt}(m, a) * \text{expt}(m, b))$

```
; consider the exponenial b^e -- it's monotonic as a function
; of b and of e separately

; as a function of b
```

THEOREM: monoton-of-exp-1
$(b1 \leq b2) \rightarrow (\text{expt}(b1, e) \leq \text{expt}(b2, e))$

THEOREM: plus-difference
$((e1 \leq e2) \wedge (e1 \in \mathbf{N}) \wedge (e2 \in \mathbf{N})) \rightarrow ((e1 + (e2 - e1)) = e2)$

... también señala cuando desactiva a una función demostrada como regla de reescritura:

THEOREM: monoton-of-exp-aux1
$((e1 \leq e2) \wedge (e1 \in \mathbf{N}) \wedge (e2 \in \mathbf{N}))$
$\rightarrow \quad (\text{expt}(b, e2) = (\text{expt}(b, e1) * \text{expt}(b, e2 - e1)))$

THEOREM: expt-is-positive
$(b \not\simeq 0) \rightarrow (0 < \text{expt}(b, e))$

THEOREM: monoton-of-exp-aux2
$(b \not\simeq 0) \rightarrow (x \leq (x * \text{expt}(b, e)))$

THEOREM: monoton-of-exp-2
$((e1 \leq e2) \wedge (e1 \in \mathbf{N}) \wedge (e2 \in \mathbf{N}) \wedge (b \not\simeq 0))$
$\rightarrow \quad (\text{expt}(b, e1) \leq \text{expt}(b, e2))$

```
; 0 is an exception because 0^0 = 1 but 0^1 = 0
```

EVENT: Disable monoton-of-exp-aux1.

EVENT: Disable monoton-of-exp-aux2.

(22) *...the proofs here are, in principle, surveyable even without mechanical assistance. It is left to the reader whether the proofs here are convincing, but they clearly are formalizable.*
pág. 185 de (15)

(23) *1. Believe in the soundness of the Boyer Moore theorem prover, and*
2.Be convinced that the incompleteness theorem has been correctly stated.
pág. 21 (15)

(24)*Program testing can be used to show the presencia of bugs, but never to show their absence!*
Edsger Dijkstra, *StructuredProgramming,* Software Engineering Techniques: Report on a Conference Sponsored by the NATI Science Committee, Rome , Italiy 1969, ed. Buxton & Randell, págs 84-88. p.85
(25) *The galling thing about the generally poor quality of much current software is that there is no extrinsic reason for it; perfection is, in principle, possible. Unlike physical devices:*
(i) There are no natural laws limiting the tolerance to which a progam can be manufactured; it can be build exactly as specified.
(ii) There is no Heisenberg uncertainty principle operative; once built, a program will behave exactly as prescribed -
(iii) There is no friction or wear; the correctness and performance of a program will not decay with time.
Wulf, W.A., *Introduction to Part I: Comments on Current Practice*, en Research Directions in Software Technology 1979, Cambridge, MA: MIT Press, págs 39-43

(26) Richard A. De Millo, Richard J. Lipton, and Alan J. Perlis ,*Social processes and proofs of theorems and programs*, Communications of the ACM, Volume 22 , Issue 5 May 1979, Pages: 271 - 280

(27) *...a proof is only a step in the direction of confidence. We believe that, in the end, it is a social process that determines whether mathematicians feel confident about a theorem - and we believe that, because no comparable social process can take place among program verfiers, program verification is bound to fail.*
pág. 271 de (26)

(28) *...the proof of a theorem is a message. A proof is not a beautiful abstract object with an independent existence.*
pág. 273 de (26)

(29) *After enough internalization, enough transformation, enough generalization, enough use, and enough connection, the mathematical community eventually decides that the central concepts in the original theorem, now perhaps greatly changed, have an ultimate stability. If the various proofs feel right and the results are examined from enough angles, then the truth of the theorem is eventually considered to be established. The theorem is thought to be true in the classical sense -that is, in the sense that it could be demonstrated by formal, deductive logic, although for almost all theorems no such deduction ever took place or evil will.*
pág.274 de (26)

(30) *[como la manifestación de placer de una puta]*
Verifications are not messages; a person who ran out into the hall to communicate his latest verification would rapidly find himself a social pariah. Verifications cannot really be read; a reader can flay himself through one of the shorter ones by dint of heroic effort, but that's not reading. Being unreadable and -literally- unspeakable, verifications cannot be internalized, transformed, generalized, used, connected to other disciplines, and eventually incorporated into a community consciousness. They cannot acquire credibility gradually, as a mathematical theorem does; one either believes them blindly, as a pure act of faith, or not all.
pág. 275 de (26)

(31) *...we argue that the world of production software is itself discontinuous. No programmer would agree that large production systems are composed of nothing more than algorithms and small programs. Patches, ad hoc constructions, bandaids and tourniquets, bells and whistles, glue, spit and polish, signature code, blood-sweat-and-tears. and, of course, the kitchen sink -the colorful jargon of the practicing programmer seems to be saying something about the nature of the structures he works with; maybe theoreticians ought to be listening to him.*
pág. 277 de (26)

(32) *In many applications, algorithm play almost no role, and certainly, presents almost no problem.*
Hoare, C.A.R., *Software Management*, Research Directions in Software Technology, MIT Press, 1978

(33) *The formal demonstration that a program is consistent with its specifications has value only if the specification and the program are independently derived. In the toy program atmosphere of experimental verification, this criterion is easily met. But in real life, if during the design process a program fails, it is changed, and the changes are based on knowledge of its specifications; if the specifications are changed, and those changes are based on knowledge of the program gained through the failure. In either case, the requirement of having independent criterion to check against each other is no longer met. We hope that no one would suggest that programs and specifications should not be repeatedly modified during the design process. That would be a position of incredible poverty - the sort of poverty that does, we fear, result from infatuation with formal logic.*
pág. 275 de (26)

(34) Naur, Peter, *Formalization in Program Development*, BIT 22(4): 437-453 (1982) reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 190-210

Observación contra la visión fragmentaria de la cita trascrita: El computólogo danés no es enemigo de la matematización, ni tampoco tajantemente rechaza la formalización de las especificaciones y las verificaciones; al contrario, ha sido de los principales propulsores de la matematización y a la formalización la considera un mal necesario en algunas circunstancias. Sin embargo, para Naur lo informal no es antónimo de rigor, ni lo formal sinónimo de certeza. Es más, en algún lugar del artículo escribe:

Even in the most formalized mathematical argument the justification of each argument step in terms of a rule of inference must finally rest with the author's intuitive, informal acceptance that the rule applies and has been used correctly.

Para Naur lo informal preside en nuestro intelecto, por lo que:
Instead of regarding the formal mode of expression as an alternative to the informal mode we must view it as a freely introduced part of the basic informal mode, having sometimes great advantages, mostly for the expression of highly specialized assertions.

(35) *Faced with the difficulties in expressing required solutions using established programming languages, the advocates of so-called formal specifications offer what is claimed to be a help to the programmers. According to their ideas the programmer must first express the solution in a so-called formal specification language, second, express how to get that solution in a programming language, and third, prove that the program meets*

with that solution. From discussion above it should be clear that in the view presented here this approach offers no help to the programmer , but only adds to his or her burdens.
pág. 207 de (34)

(36) Brian Cantwell Smith, *Limits of Correctness in Programs*, a paper prepared for the Symposium on Unintentional Nuclear War, Fifth Congress of the International Physicians for the Prevention of Nuclear War, Budapest, Hungary, June 28–July 1 1985. reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 275-293.

(37) Maurer, W.D., *Letter to Editor*, Communications of the ACM 22 , 1979, págs. 625-629

(38) Richard A. De Millo, *Formal Semantics and the Logical Structure of Programming Languages*, Ph.D. thesis, Georgia Tech, 1972

Madcap fue un lenguaje de programación de alto nivel desarrollado en el laboratorio de los Alamos, con la gracia de utilizar una notación con superíndices y subíndices tal como se acostumbra en las matemáticas.

(39) *Verifications are long and involved but shallow; that's what's wrong with them. The verification of even a puny program can run into dozens of pages, and there's not a light moment or a spark of wit on any of those pages.*
pág. 276 de (26)

(40) Scherlis, William, Scott, Danna, *First steps toward inferential programming,* in Mason, R.E.A. (ed.), *Information Processing 83*, IFIP, Elsevier Science Publishers (North-Holland), September 1983, pp. 199-212. reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 99-133

But, even relatively elementary programs tend to be more complicated than elementary theorems. Why? Because in a certain sense more of their structure has to be made explicit. A proof is often more a genteel tourist guide than instructions for a treasure hunt. Programs, on the other hand, not only operate on very highly structured data, but they must do so in unobvious ways in order to be efficient.
págs. 113-114

(41)
-though they mention the early muddy history of the 'solutions' to the Four Colour Conjecture- De Millo, Lipton and Perlis do not discuss the subsequent machine-aided proof of this world famous conjecture. (Their paper was received by the editors in October 1978 and probably written much earlier, the first announcement by Haken, Appel, and Koch of

the solution of the conjecture was made in September 1976, and the paper was published in September 1977)
pág. 116 de (40)

(42) Fetzer, James, *Program Verification: The very Idea,* Communications of the ACM 31, 1988, págs. 1048-1063. reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993): 321-359

Indeed, while social processes are crucial in determining what theorems the mathematical community takes to be true and what proofs it takes to be valid, they do not thereby make them true or valid.
pág. 322

(43) *After all, what makes (what we call) a proof a proof is its validity rather than its acceptance (by us) as valid, just as what makes a sentence true is what it asserts to be the case is the case, nor merely that it is believed (by us) and therefore referred to as true. Social processing, therefore, is neither necessary nor sufficient for a proof to be valid, as DeMillo, Lipton and Perlis implicitly concede.*
pág. 325 de (42)

(44) *No more can appropriately be claimed than that its conclusion must be true if all its premises are true, which is the defining property of a valid demonstration.*
pág. 327 de (42)

(45) Cohn, Avra , *The Notion of Proof in Hardware Verification,* Avra Cohn: The Notion of Proof in Hardware Verification. J. Autom. Reasoning 5(2): 127-139 (1989)
reprinted in Timothy R. Colburn, James H. Fetzer, & Terry L. Rankin (eds.), *Program Verification: Fundamental Issues in Computer Science* (Dordrecht, Holland: Kluwer Academic Publishers, 1993) pp 359-375

First, hardware verification is in many ways more tractable than software (program) verification -it is often easier to write a clear specification that captures the functionality of a system of hardware than software- and hardware proofs tend to have a certain uniformity of structure which is well suited to mechanical treatment.
pág. 359

(46) *A device can be described in a formal way, and the description verified; but as with intentions, there is no way to assure the accuracy of the description. Indeed, any description is bound to be inaccurate in some respects, since it cannot be hoped to mirror an entire physical situation even at an instant, much less as it evolves through time; a model of a device is necessarily an abstraction (a simplification).*
págs. 364-365 de (45)

(47) *The very idea of program verification trades upon an equivocation. Interpreted in senses (i) and (ii)* [*(i)algorithms (ii)encodings of algorithms*] *, there is no special difficulty that arises in "verifying" that output O follows from input I as a logical consequence of axioms of the form I c O. Under such interpretation, however, nothing follows from the verification of a "program" concerning the performance of any physical machine. Interpreted in senses (iii) and (iv)* [*(iii) encodings of algorithms that can be compiled (iv)encodings of algorithms that can be compiled and executed by a machine] , however, that output O follows from input I as a logical consequence from axioms of the form IcO, cannot be subject to absolute verification, precisely because the truth of these axioms depends upon the causal properties of physical systems, whose presence or absence is only ascertainable by means of inductive procedures.*
pág. 346 de (42)

(48) *It is nothing but symbol chauvinism that makes computers scientists think that our structures are so much important than material structures that they should be perfect. ... For the practice of programming, however, verifiability must not be allowed to overshadow reliability. Scientists should not confuse mathematical models with reality -and verification is nothing but a model of believability.*
pág. 279 de (26)

(49) *Should we regard a mechanically verified proof as certifiably correct? Definitely not. As mentioned above, mechanical assistance is useful in finding and correcting reasoning error but it cannot guarantee that errors have been eliminated.*
pág. 184 de (15)

PRELUDIOS Y FUGAS

Es el final. Comienzo con algunos dulces porvenires presagiados por los capítulos previos. Proyecciones hacia donde las naturalezas inquietas talvez encuentren rutilantes preguntas o se pierdan con oscuras respuestas. Algunas rutas ya fueron transitadas, pero la exploración según mi opinión puede y debe proseguir con la ampliación y refinación de los mapas. Sólo la benevolente autocomplacencia –por displicente- o una autocrítica acérrima –por contundente- pueden suspender las figuraciones y parar el trazado, me parece.

FUGAS

LA ERA DIGITAL

La computadora es una herramienta bastante socorrida en las matemáticas aplicadas. Desde el capítulo primero, se mostró cómo la voracidad deductiva de las puras también puede con los unos y ceros ser aplacada. Las utilerías de métodos numéricos, graficación, manipulación algebraica o geométrica o aritmética o estadística, edición de texto matemático y simulación –por mencionar algunas- tienen como nuevos compañeros a los programas para demostrar teoremas y verificar pruebas. Los asistentes mecánicos se han infiltrado hasta la médula de la investigación matemática pero también se han extendido en la superficie pedagógica al impulsar y mejorar la transmisión del conocimiento. Internet ha impelido la masificación de la difusión de artículos, distribución de libros y divulgación de las matemáticas; además la presentación del conocimiento gracias al instrumento ha sido mejorada –v.gr. con inserción de hipervínculos y elementos multimedia-. Independientemente de la calidad del tejido argumentativo, la confección de este escrito requirió de varios hilos de información deshilvanados de la red de redes. En el capítulo tercero se mencionó al ambiciosa proyecto digital de la cimentación de la una nueva biblioteca de Alejandría, refugio de las preciosas piedras matemáticas, empresa cuyo término luce en la lejanía pero con impacto incluso en su fase embrionaria. ¿Qué reductos ha conquistado la revolución digital en las matemáticas? ¿Qué tanto peso tiene la computadora en la actividad matemática diaria? ¿Cuáles son las ventajas y desventajas desprendidas por la incorporación de las computadoras en la vida matemática? En esta tesis se indicaron algunos esbozos de respuesta, pero el panorama es demasiado extenso y son tantas las perspectivas –filosóficas, históricas, sociológicas, económicas- desde donde puede analizarse para conformarse y suspender las indagaciones. Como ejercicio de curiosidad y clarificación estadística, puede uno recopilar en los departamentos de matemáticas de su universidad cuántos matemáticos recurren a los sistemas computacionales, para qué lo hacen (docencia o investigación o comunicación académica o por mero y sano ocio), qué programas utilizan, cuántas horas pasan frente a un monitor, etc. Este ejercicio a gran escala puede revelar el grado de digitalización de la actividad matemática y a pequeña escala el de la universidad donde uno habita. *La humanidad tiene una nueva clase de instrumento que ha incrementado al poder de razonamiento mucho más*

potente que el fortalecimiento de nuestra visión brindado por cualquier instrumento óptico. ¿Habremos llegado a la era cuando la trastocada frase de Leibniz hará eco con el consentimiento de ésta emitido por la comunidad matemática?

LA SINTÁCTICA SEMÁNTICA, LA SEMÁNTICA SINTÁCTICA

La Máquina Geométrica mencionada en las postrimerías del primer capítulo expandió el horizonte del razonamiento mecánico al incorporar a la semántica como guía de sus derivaciones sintácticas. Aunque la semántica aludida no superara por mucho el suelo sintáctico, la traducción entre enunciados en una lógica de primer orden y frases de la geometría analítica respalda la atribución de una "semántica" a la Máquina Geométrica. Entonces, ¿las computadoras en las matemáticas pueden deducir también por medio de significados? Antes de poder contestar, se debe definir lo que se entiende por significado en las oraciones matemáticas, tarea para la cual la computadora puede ocasionar una corriente de ideas y disputas que confluyan en los meandros de la inteligencia artificial donde navegan aquellos anegados con los problemas cognoscitivos. $x^2+y^2=r^2$, $\forall x \exists !y(x \in C \Rightarrow d(x,y)=r)$, O, ... una computadora puede transitar entre varias de nuestras caracterizaciones del objeto matemático denominado círculo y puede manipular ecuaciones, fórmulas, figuras, a partir de la interacción de ellas. Queja inmediata, la computadora entiende nada de los significados geométricos-algebraicos-lógicos, para ella todo se reduce a trabajar con cadenas binarias. Contestación trapera, aunque sepamos acerca del cerebro tanto como una oveja sabe de economía (al final de cuentas es cuestión de trasquilar y repartir la lana), talvez todo aquello glorificado y al mismo tiempo denostado con el nombre de pensamiento –en particular el análisis semántico- no sea más que cadenas de impulsos nerviosos. Queja responsiva, aunque el funcionamiento del cerebro sea un misterio, a nivel mental comprendemos a y razonamos con las distintas acepciones de círculo mientras que la computadora nunca superará el nivel del on/off. Contestación trapacera, la computadora puede trabajar con algunas caracterizaciones de círculo y realizar operaciones a partir de ellas dando la apariencia de comprenderlas, al menos en función de los resultados. Más aun, la interrogante se remite en cómo la capacidad de los programas de procesar "semánticamente" enunciados matemáticos puede repercutir en la noción del significado ya sea modificándola para incluir a la "semántica" manejada por las computadoras, conservándola tal cual está para acreditar la inclusión o exclusión de la "semántica" manipulable por algunos asistentes binarios. Queja conclusiva, faltan enunciar muchos puntos y hacerlo con mayor cuidado para poder darle a la discusión un cauce adecuado. Sin embargo, al dirigir las inquisiciones a la semántica matemática la pesca filosófica puede ser más fructífera por la restricción de los significados en sistemas lingüísticos menos rebuscados que el lenguaje natural –el lenguaje matemático tiene una estructura más apegada a las reglas y al orden- o el visual –las significaciones ostensivas no son tan determinantes en las matemáticas-. Además, a partir de los significados puede intentarse reconstruir el contenido epistemológico de los poemas matemáticos, las demostraciones. Porque el conocimiento en ellas probablemente radique en las redes entre los distintos organismos –objetos, funciones, relaciones, operaciones, fórmulas...- matemáticos, porque la sustancia epistemológica talvez sea meramente un bordado semántico montado en un

armazón sintáctico. No es una arremetida reduccionista, en ningún momento propongo explicar a la geometría euclidiana partiendo únicamente de los significados de sus objetos-operaciones-relaciones elementales como punto, línea, círculo, pertenencia, intersección, etc. Aunque fuera una estrategia sólida, es una apuesta inútil pues sus ganancias son limitadas. Las definiciones de los objetos no abarcan la totalidad de sus significados. La estructuración de los objetos en enunciados propaga en ambas direcciones los significados; el significado de la oración a partir del significado de sus componentes puede ser dilucidado, pero también gracias al significado de la oración puede reconstruirse el significado de algunos de sus componentes. El significado de la demostración tampoco se reduce a la suma de los significados de sus versos, ¡oh!, me temo. Por otro lado el problema semántico puede seguir un curso más formal y menos brumoso al apegarse más a la lógica. En el capítulo tercero se mencionó brevemente al programa Mace, programa capaz de construir modelos finitos y buscar en ellos contraejemplos. Si semántica es la interpretación de los enunciados del sistema lingüístico –formal- en una estructura lingüística –matemática-, entonces hay varios programas de razonamiento deductivo con facultades semánticas. Se puede intentar extender esta definición, así significado es un proceso -¿o resultado?- de traducción entre distintos lenguajes –el lenguaje destino y fuente puede ser el mismo-. Se puede virar hacia el más allá semántico, encerrando entre un par de disconexos garrotes retorcidos cual perturbadoras hoces a la traducción en sí, es decir, ¿cómo relacionamos a los objetos en el sistema origen con los elementos de la estructura destino? ¿el significado es un conjunto de reglas relacionales y no tanto los elementos relacionados mediante ellas? Otro enfoque similar y distinto lo proporciona Alan Bundy con su ciencia del razonamiento. Bundy, quien también ha trabajado con programas de razonamiento deductivo, propone en cierto sentido identificar a la semántica con patrones sintácticos. En parte lo espolean motivos pragmáticos, pues sus hipótesis pueden ser implementadas en las computadoras y mediante ellas ser probadas, dando pie a la experimentación y a la retroalimentación. Si el éxito le sonríe, entonces además de la resonancia filosófica sonarían los aportes teórico-tecnológicos a los programas de razonamiento deductivo.

SE VENDEN LÓGICAS AL MEJOR POSTOR

La amplitud y la profundidad del presente ensayo son mínimas. Hay en el mercado otros programas-paradigmas además de los aquí mencionados. El par de sistemas para los cuales fueron destinadas más palabras para su descripción merecen una exégesis más detallada, la información desplegada acerca los sistemas resolutivos –Otter- y el sistema recursivo –ACL2- se quedó al ras de la superficie. El proyecto Mizar merece más atención. La investigación puede tomar un cariz más lógico, aunque también se puede estudiar las repercusiones filosóficas o computacionales –si las hay- de las distintas escuelas del razonamiento deductivo computarizado. Por ejemplo, hay algunos sistemas demostradores de teoremas entarimados sobre lógicas constructivistas ya que entre otras cosas estas lógicas abogan por métodos computables, la existencia debe demostrarse por construcción y no por tercero excluido vía reducción al absurdo. Y dichos métodos tienen media implementación ganada, son métodos cercanos a los linderos de la computabilidad material.

¿Para qué puede servir esto? Robert Constable le encontró un uso sugerente, la automatización de la programación correcta. En lugar de programar para luego verificar, ¿por qué no programar y demostrar casi al mismo tiempo? ¿por qué separar al programa de su prueba, no sería mejor programar con base a la prueba de lo que se está implementando sea correcto? Si la prueba está adscrita a una lógica constructivista, entonces esa prueba potencialmente puede tornar en programa automáticamente. Por otro lado Constable también se fijó como meta la formalización digital de las matemáticas en un sistema constructivista, resucitando viejas disputas filosóficas con un cuerpo moderno lleno de transistores. Para él, la intuición computacional es más sólida que la fundamentación teórico-conjuntista, ¿será? A diseccionar el Nuprl de Constable.

HACIA LA MECANIZACIÓN DE LAS CONJETURAS

Wang lo señaló, las asistentes mecánicas *pueden tener el buen uso de la producción masiva de ejemplos concretos*, conminando a sus colegas a experimentar con ellas, siendo el verbo previo una acción esencial para conjeturar. A mediados de los setenta Douglas Lenat programó a un ciberniño prodigio, proveyéndole de un conjunto conformado por conceptos básicos propios de un infante (sic) y de un conjunto de sofisticadas reglas heurísticas propias de un adulto (sic). Resultado: AM ("Automated Mathematician) + Ph.D de Lenat. ¿Qué tanto descubrió el pequeño genio inveterado? En 1985 Fajtlowicz diseñó un programa conjeturador llamado Graffiti aplicable a varios campos de la matemática y de la química. En la teoría de las gráficas es donde Graffiti ha disfrutado su mayor éxito. Así como las demostraciones computarizadas suscitan polémicas en torno al concepto "demostrar", los bichos digitales de este tipo no desmeritan al binario espíritu sedicioso ¿Conjeturar o conjurar o conjuntar al azar?

PRELUDIOS

En la agonía defiendo al nacimiento y desarrollo del escrito. ¿Por qué la caracterización de Tymoczko de las demostraciones es acertada? ¿Por qué basar en ella la exégesis de las demostraciones computarizadas? ¿A qué demonio se debe invocar o a que santo se debe rezar para afirmar que si x es una demostración entonces x es *convincente*, x es *revisable* y x es *formalizable*? Independientemente de la validez o la falta de ella de las modificaciones propuestas desde el segundo capítulo –los trueques entre *formalizable* por *computable* y *revisable* por *inteligible* más el raspado de los conectivos conjuntivos- o del mayor reparo en los factores sociales enfatizado desde el capítulo tercero, ¿por qué tomarnos la molestia de arropar a las demostraciones con esa caracterización? Porque es lo suficientemente general para ser aplicable, porque especifica tan poco que la hace una prenda lo suficientemente flexible para intentar ponérsela a las demostraciones. La puntada más arriesgada efectuada por Tym es explicar el poder de *convencimiento* de una demostración a través de la conjunción de su *revisabilidad* con su tácita *formalización.* Sin embargo en lugar de despejar las dudas solo las traspasa. Ser *revisable* es una propiedad en algunos

aspectos con una interpretación demasiado maleable para ser concluyente y en otros demasiado rígida para ser aceptable. Ser *formalizable* es una propiedad más concisa pero sumisa a la esperanza y a la fe, las demostraciones raramente vienen vestidas con una presentación formal, por lo que es más un supuesto que un hecho palpable. Gracias a las demostraciones computarizadas la caracterización puede ser refinada y reforzada. En los capítulos segundo y tercero lo hice y espero no haber resbalado. En los esterto res de la tesis recopilo y afilo lo anteriormente tallado y detallado.

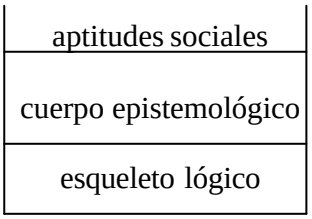
La *revisabilidad* exige mucho aunque tiene buenos motivos para hacerlo. Es la propiedad neurálgica en las demostraciones estándares. Gracias a la *revisabilidad* la mayoría de las nuevas demostraciones se rechazan o aceptan. Aunque el cuerpo de una demostración tenga un sólido esqueleto lógico, sin una musculatura epistemológica fuerte y un tejido epitelial atractivo el movimiento del conocimiento de la demostración puede suspenderse. Debido a los usuales procesos sociales acaecidos en distintas fases de la vida de las demostraciones, normalmente se les pide como requisito ser totalmente legibles. Los autores, editores, jueces y su público en general deben poder extraer los contenidos y cerciorar la correcta ilación de los argumentos en una demostración. Si bien en la formulación original de Tym la *revisabilidad* al principio apunta hacia la experiencia epistemológica personal, es decir , el modelo del individuo solitario apto para redactar/leer/editar/avalar o denostar/modificar a la demostración, la embestida argumentativa del mismo Tym, Perlis, De Millo, Lipton y muchos más aquí no mencionados sugiere intercambiar el peso del individuo aislado por la ponderación sociológica, es decir, es un grupo de personas (modelo que no excluye a los extraordinarios casos solitarios aunque se les preste tanta atención como a las pulgas de mi dedo gordo del pie) quienes redactan/leen/editan/validan a las demostraciones. Y donde hay un esfuerzo colectivo pululan otros factores además de los cognoscitivos. Una demostración puede ser desatendida aunque sea válida por la escasa reputación de su autor o el escaso interés que genere. El *convencimiento* para bien y para mal no se reduce exclusivamente a la *revisabilidad* y a la hipotética *formalización*, por lo que debe darse cabida a ciertos factores sociales revueltos en la construcción de una demostración mientras nos preocupe formular un modelo con una correspondencia con lo representado. Bajo una tónica similar en el capítulo tercero se evidenció con las traducciones de la prueba computarizada EQP/McCune de la Conjetura de Robbins que la *revisabilidad* para posibilitar la ratificación de la demostración y para servir como punto de acceso a su contenido epistemológico necesita mostrar el mensaje con una forma sociable. No basta con tener la correcta apariencia de una prueba formal, prueba de la validez de una demostración, la demostración debe tener una presentación propicia para establecer una comunicación con sus lectores y los cúmulos de ellos para disparar a los procesos encargados de infundirle vida (comprenderla, corregirla, parafrasearla, emplearla con otras conjeturas, generalizarla), mecanismos montados en un organismo social. Las demostraciones computarizadas también pueden acceder a este ajetreado tránsito, aunque algunas pruebas computarizadas completas requieran de nuevo como vehículo a su instrumento progenitor para recorrer a su accidentada comprensión. Por otro lado, la catatónica demostración de H&A&K&1&0 provocó el derrumbamiento de la exigencia por la legibilidad total y de la información

autocontenida, dando origen a la *inteligibilidad.* La extracción del conocimiento en esta demostración no obliga al lector a enajenarse en la inspección de miles y miles de operaciones simples, basta con comprender a los algoritmos y saber lo suficiente de computación para acreditar la confianza en los programas y en el instrumento ejecutor. Demasiadas veces he escrito conocimiento, por lo que es tiempo de hacerse a un lado: la *inteligibilidad* (*revisabilidad*) debe considerarse como un recipiente pulido y acondicionado para contener al líquido epistemológico concerniente a las fuentes matemáticas de donde proceda la demostración, líquido de naturaleza diversa (v.gr. las demostraciones geométricas tienen una viscosidad epistemológica distinta a las demostraciones algebraicas en teoría de grupos, los líquidos pueden mezclarse empero) cuyo estudio rebasa los límites de este ensayo aunque no rebose al recipiente propuesto y admito que su exclusión rebosa una astenia desentendida de las concepciones duras de la filosofía. Sin embargo insisto y recalco, mientras queramos mantener contacto con la realidad a las preconcepciones si se les da cabida, es porque pueden ser aterrizadas y porque son explicativas. Definan como quieran conocimiento pero tengan en consideración lo siguiente: difícilmente su definición cubrirá a la variedad del conocimiento matemático a menos de que (1) su definición admita refinaciones y particularizaciones o (2) su definición diga nada y por vacuidad sea admisible. Quienes caigan en lo primero deben ser capaces de escuchar y filtrar al ruido relativista, quienes se pierdan en lo segundo sigan propagando al éxtasis provocado por la elocuencia del silencio (aunque sea un silencio encerrado en una barahúnda argumentativa).

Ser *formalizable* irradia un aura esotérica. Aunque el cumplimiento de esta propiedad baste para solventar la validez de una demostración cuando la demostración emerja con una presentación formal, es insuficiente para garantizar su convencimiento. Los bloques lógicos deben ser convertidos en fino polvo epistemológico de difusión más fácil y a veces en el triturado puede participar la computadora, tal como fue reseñado en el capítulo tercero. Mientras que para materializar a esta propiedad se puede recurrir de nuevo a los sistemas al estilo Mizar, razón última por la cual decidí renombrar al ser *formalizable* por ser *computable.* La formalización por medio de sistemas computacionales ofrece varias ventajas, en especial el apoyo de un asistente binario en primera instancia capaz de corregir y convalidar a las pruebas con una eficacia mayor que la de sus amos humanos y en una fase más avanzada –con suerte no tan distante- facultado para generar de manera más automatizada a las pruebas formales –habilidad ya exhibida por algunos programas como Otter-. Con respecto a esta última ayuda brindada por los programas algunos objetarán la aspereza de los pasos derivativos en sus pruebas manufacturadas, aspereza contraria a los ideales de simpleza alegados por Wang o Hilbert. La queja previa en su momento ya fue atendida, la complejidad aparente no es más que un espejismo de la sencillez con una aceleración eléctrica, en un sólo paso derivativo algunos programas demostradores hacen varias operaciones simples. La honestidad convierte a la esperanza en esclava de los dictámenes de una objetividad inclemente. La abrumadora cantidad de nuevas demostraciones en las matemáticas cuya magnitud saludablemente continúa en aumento hace de la formalización digital una obra a muy largo plazo, francamente ahora luce como una luz al final de un túnel extremadamente largo. Todavía falta mejorar el estatus de la

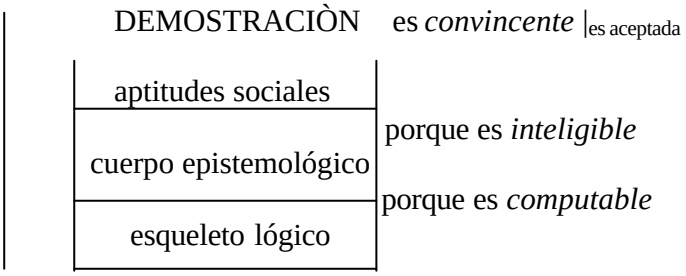
computadora en la comunidad matemática y además esta herramienta debe ganarse su confianza. La computadora no es un mero ábaco sofisticado ni únicamente una burda máquina de Turing, es una herramienta polifacética con una probabilidad de error reducible aunque de supresión imposible. Sin temor a la redundancia aviso de nuevo y sobre aviso no hay engaño: formalizar es una tarea nada sencilla, los náufragos perdidos en estas turbias aguas deben nadar en ríspidas corrientes de signos y detectar diligentemente axiomatizaciones favorables, muy a su pesar muchas de las veces sólo lograrán avistar al fracaso. La navegación gracias a los instrumentos deductivos digitales tal vez sea más veloz y más segura, desembarcar en una prueba sigue siendo una proeza empero. Ahora no hay recursos ni marineros suficientes para la empresa, ni los habrá en mucho tiempo. Sin embargo, las prometedoras ganancias inhiben a la abdicación. Formalizar con todo y las sospechas sobre el instrumento y las escasas expectativas actuales de éxito otorga invaluables premios: solidifica, clarifica, clasifica y ordena al acervo matemático. La formalización debe conservarse viva, aunque sea en un estado de animación suspendida. El manifiesto QED no debe ser olvidado, aunque sea recordado en obtusos trabajos.

Las demostraciones son mensajes pero no son cualquier mensaje, poseen una estructura y contenido propio de los escritos matemáticos. Ahora opero a la inversa, adopto y adapto un modelo socorrido de redes computacionales (OSI-TCP/IP) para representar a la aceptación-comunicación de dichos mensajes. La peculiaridad atractiva del modelo ultrajado radica en su estratificación jerárquica y en las distintas relaciones entre las capas aledañas sin importar mucho lo que representan, por lo cual su explicación es innecesaria. El modelo propuesto tiene tres capas representativas de la clase de elementos identificados a lo largo de la tesis en las demostraciones matemáticas: sociales, epistemológicos y lógicos. Es un modelo vertical y horizontal, cada capa tiene características endémicas pero entre capas vecinas existen puntos para transitar de una a la otra. La capa sobre la cual descansan las otras dos representa a los elementos lógicos, la intermedia a los epistemológicos y la capa superior vecina de la previa a los sociales. De este modo, la caracterización de una demostración bajo este modelo estaría conformada así:

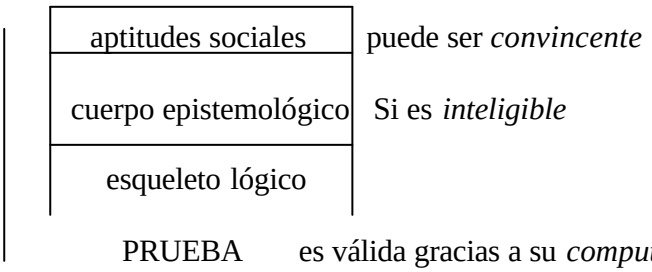


La caracterización de la demostración puede interpretarse en el modelo de la manera siguiente. La estratificación en capas remite al ámbito de las tres componentes, la transición entre ellas las alude directamente. En el sentido de arriba hacia abajo, en la cumbre del modelo se encuentra el ser *convincente*, propiedad con marcados acentos sociales. ¿Por qué es convincente la demostración? Porque los canales de comunicación pueden ser abiertos y subsecuentemente pueden ser disparados los procesos de

edición/ratificación/transformación de la demostración. Si bien algunos factores sociales intervienen o interfieren en la apertura (amplitud horizontal del modelo), la *inteligibilidad* es quien normalmente abre las puertas del contenido epistemológico del mensaje; de aquí la estratificación vertical y el punto de acceso entre las aptitudes sociales y el cuerpo epistemológico ofrecido por esta característica. Normalmente una demostración será atractiva si su contenido epistemológico lo es, contenido en constante desarrollo a varias manos, a varias miradas; muchas cabezas piensan mejor que una. Restrinjamos la duda sobre el convencimiento, si la demostración es *convincente* entonces la demostración debió ser aceptada. Luego, ¿por qué puede ser aceptada una demostración? Porque otra vez su *inteligibilidad* permite el decodificado del mensaje, activando al sistema inmunológico de la disciplina a la que pertenezca la demostración. Sobreviven aquellas que superan a la inspección epistemológica hecha por los especialistas (tal vez asistidos por una computadora) quienes al no lograr hacer blanco en algún error no la fagocitan. De este modo la demostración es aceptada por el cuerpo epistemológico del área matemática en cuestión, de sus hipótesis válidamente se siguen sus conclusiones. ¿Y en donde descansa su validez? En el esqueleto lógico en dos niveles: el s ideal (y a veces materializado) y el práctico (muchas veces idealizado). En el sentido ideal la demostración es válida porque es *computable*, en el práctico lo es porque el armazón lógico no exhibe fisuras (al menos no fueron localizadas) y sirve de sostén al tejido epistemológico de la demostración, tejido inspeccionado por los leucocitos del cuerpo epistemológico donde resida. En ambos niveles, el cuerpo epistemológico precisa de un esqueleto lógico sólido, en el extremo ideal se exigirá su presentación formal, en el otro nos conformaremos con formalizaciones inacabadas, sombra de aquellas otras figuraciones puras aunque no sea tan claro establecer el vínculo entre el objeto formal y su proyección fusca. La ortodoxia filosófica apoyaría al siguiente diagrama:

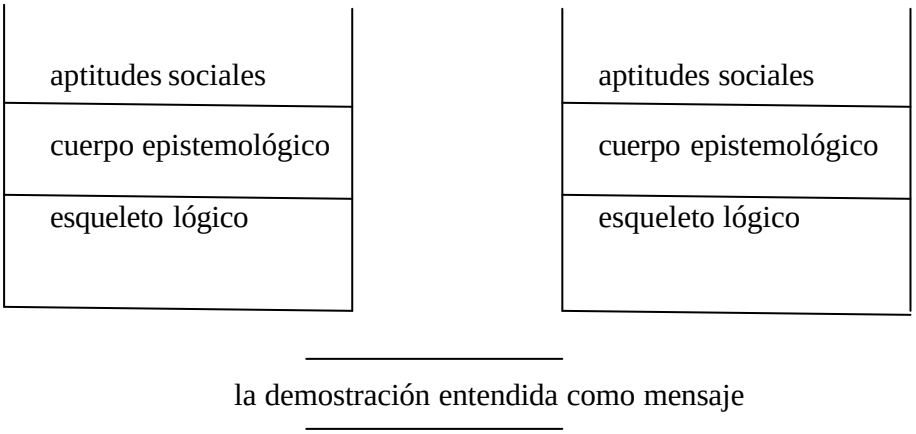


En el sentido inverso, en el sótano mediante la *computabilidad* se puede establecer la validez de una prueba y mediante su *inteligibilidad* puede propulsarse su *convencimiento.*



El crecimiento horizontal permite enriquecer al modelo incorporando cuestiones propias de cada capa, mientras más arriba se localice más frondosos serán los injertos. La interpretación ortodoxa del modelo para conservar la enhiesta verticalidad requiere de los cimientos construidos por medio de la formalización (digital) de las demostraciones, en el nivel superior identifica al impacto social de una demostración con su relevancia epistemológica y más abajo resguarda a los elementos epistemológicos en la solidez de la bóveda de los sistemas formales. Si nuestro interés nos impele a alejarnos de las interpretaciones fundacionales para aproximarnos más a las demostraciones funcionales se debe renunciar al forzoso descenso hacia al rigor fomentando la expansión horizontal. De este modo, el esqueleto lógico se emplea para enderezar la postura eliminando errores pero por si solo no sostiene la veracidad de las demostraciones. Las dos capas superiores –si la actividad matemática goza de buena salud principalmente la de en medio- adquieren el protagonismo en los procesos que culminan con la aceptación e integración de las demostraciones con su correspondiente teorema al bagaje matemático. No obstante aún bajo estos lineamientos la transición descendente debe ser impulsada. La exclusión de una demostración por factores meramente sociales es comprensible aunque no justificable. La volatilidad epistemológica puede explotar en equivocaciones virulentas si se abandona el hábito del rigor lógico. Tampoco es sensato suspender las exploraciones y dedicarse de lleno a la pavimentación formal de todas las rutas descubiertas. La investigación pionera no debe suspenderse para erigir monumentales calzadas de los muertos, caminos gloriosos de un mausoleo.

La interpretación adjudicada al modelo en este párrafo alude a la comunicación de una demostración, comunicación mediante la cual acontecen sus fases de desarrollo hasta alcanzar la etapa madura de pleno convencimiento. La demostración es escrita, editada, corregida, modificada gracias a los canales de comunicación entre los matemáticos, sin embargo, el mensaje está codificado en el par de capas inferiores para validar a la información propalada. La epístola es un poema epistemológico escrito en una métrica lógica. La aceptación por el público depende de la calidad poética de la carta pero también influyen otras variables como la fama del autor. La posterior figura simboliza lo expuesto:



En el sentido previo el modelo propuesto se asemeja más al modelo padre (OSI-TCP/IP) de la comunicación en redes computacionales. Bajo esta interpretación, la investigación sobre las propiedades de las redes de comunicación de las demostraciones resulta tentadora. Las explicaciones sociales meramente descriptivas y escritas en un lenguaje natural son insatisfactorias. ¿Acaso no vale la pena mejorar el lenguaje natural de los delineamientos sociológicos haciéndolo más conciso y menos ambiguo? Más aún, se debe intentar asociar a las aptitudes sociales con los elementos epistemológicas porque la información sociológica aunque sea importante para completar el análisis no deja de ser un tanto circunstancial y accesoria. ¿Por qué no tornar hacia un lenguaje más matemático? ¿Por qué no optar por un estudio menos pasivo capaz de predecir además de explicar? Además, para emprender el estudio sociológico-matemático se cuenta con la ventaja de que el objetivo de las indagaciones es una comunidad relativamente pequeña, uniforme, hermética y sin tantas contingencias externas. En un principio la estadística podría ser la herramienta del análisis, transformando a variables matemáticas algunas variables sociales, v.gr. el nivel de prestigio de un matemático en función del número de sus publicaciones, la importancia para el área matemática de sus investigaciones –medida aproximada y tasable por los expertos en el área-, por el número de alumnos de doctorado, etc. De hecho, algunas perspectivas probabilísticas y deterministas de las redes computacionales pueden trasladarse después de los pertinentes ajustes a la inspección de las redes de comunicación de la demostración. Por ejemplo, en Internet existe el fenómeno de la concentración masiva de la distribución de la información en algunos cuantos nodos. Este fenómeno se traduce a una ley al estilo del conjunto inevitable: existe un conjunto U de nodos talque cualquier comunicación entre un par de sistemas de la red pasa al menos por un elemento de U. Otra ley afín: entre cualquier par de sistemas conectados no existen más de n eslabones entre ellos de los cuales al menos uno pertenece a U. Desde hace algunos años existe un juego matemático relacionado con el trashumante matemático Paul Erdös. El número de Erdös define el grado de cercanía en la colaboración entre un matemático cualquiera y el exótico pero fructífero Erdös. Así, Erdös es el origen y se le pega al número cero. Cualquiera que haya escrito un artículo junto con Erdös posee el número uno. Cualquiera que haya colaborado con una persona con un número de Erdös uno se gana al menos el número de Erdös dos. Quien no tenga un vínculo con una persona con un número de Erdös natural, entonces al infortunado se le asocia su número de Erdös con el infinito.

Sea $C=\{x \text{ tal que } x \text{ colaboró en la elaboración de un artículo con } z\}$ entonces si $z \neq$ Erdös $\text{Erdös}(z)=\text{mín}\{\text{Erdös}(x) \text{ tal que } x \in C\}+1$, además $\text{Erdös}(\text{Erdös})=0$. De este modo, el número de Erdös es una función recursiva –parcial si el registro bibliográfico lo es- . Las conjeturas sobre el número de Erdös aparte de su interés lúdico pueden servir como leyes del comportamiento de la red de comunicación matemática. La siguiente conjetura de pinta ociosa predica con el ejemplo: en teoría de gráficas a partir de la muerte de Erdös todo creador de un nuevo teorema en esa rama tiene un número de Erdös acotado por N+número de años trascurridos desde la muerte de Erdös. De esta manera podemos buscar encontrar los personajes centrales en los distintos campos matemáticos, definiendo a partir de esta investigación entre seria y juguetona la noción sociológica de autoridad. Nunca se debe subestimar ni sobrestimar a los modelos matemáticos, siempre serán aproximaciones aunque su exactitud tienda a refinarse.

El modelo estratificado evita la ceguera de las polarizaciones. La noción de demostración no puede ser rebozada totalmente con el manto de un nivel, es un error típico aunque obedece a fines más ambiciosos. Los abogados de la sociología del conocimiento fuerte decretarán con un poco de respaldo histórico la primacía de la capa social, condenando a la parálisis a su análisis al cortarle de antemano las piernas. Para ellos les recomiendo la lectura de libros acerca del método axiomático y los sistemas formales, les serviría para apuntar sus cañones de la contingencia a objetivos más endebles. Eviten la epistemología y la ontología, los filósofos llevan años lidiando con ellas y tienen más experiencia en ese campo de batalla. El conocimiento como creencia social y la génesis natural-social de los objetos matemáticos los conducirá al paredón si no extreman precauciones, aunque tuvieran razón es una pelea ventajosa para los filósofos. Donde deben inocular su cáncer social es en un órgano donde la metástasis sea inminente e imparable. Deben atacar al sistema axiomático. Deben saber que los matemáticos sitúan a la intuición en los extremos de los sistemas formales, la intuición se transpira en los axiomas y en la metateoría. La veracidad de las pruebas en los sistemas formales es convalidada por los primeros principios o por los resultados metateóricos, y ese par de polos tiene como centro magnético a la intuición matemática, la cual en el mejor de los casos atrae a la verdad o al menos repele a la mentira. La intuición posee una fundamentación filosófica más inestable mientras más real, más sólida mientras más ficticia Traten de caracterizar a la intuición como una construcción social y si triunfan canten vítores por haber aniquilado a los sueños de los filósofos y de sueños, sepan, viven los hombres. Para los filósofos testarudos con aspiraciones a explicar a las demostraciones únicamente a partir de la capa lógica, les sugiero que aprendan a programar y se unan a los grupos de idealistas cuya meta es formalizar digitalmente a las matemáticas. Hace falta mano de obra y su colaboración sería más que bienvenida. No teman a las imposibilidades teóricas, despreocúpense de los resultados de Gödel o de la falibilidad de las computadoras, son pequeñeces en comparación de la faraónica obra en la cual participan. A los de en medio, aunque no me llame Jesús, los regurgito en el siguiente párrafo.

¿Por qué nos interesan tanto las demostraciones en las matemáticas? Porque son garantía y son partícipes del conocimiento de los teoremas. Conocer un teorema implica frecuentemente desglosar el contenido epistemológico de su demostración, denominémoslo conocimiento por comprensión. Los procesos de simplificación o la paráfrasis de una demostración pertenecen a esta fuente de conocimiento. Por otro lado, la vinculación del teorema con el resto de la teoría o teorías extranjeras puede otorgarnos un conocimiento por extensión. Modificar al teorema mediante su generalización, gestación de corolarios o reformulación en otra teoría nos brinda un conocimiento por extensión. ¿Qué tan importante es el conocimiento por comprensión? La respuesta depende de la demostración aludida. Algunas demostraciones son tan técnicas o exhiben una parvedad epistemológica tal que poco se puede conocer del teorema por este medio. Las pruebas computacionales completas por su árida presentación pueden ser territorio poco fértil para el conocimiento por comprensión, al menos en su forma bruta. En este caso al menos persiste la otra alternativa para el conocimiento. ¿Y si tampoco es viable? El pánico no se apoderará de nuestros pensamientos, sino la indiferencia. El yermo teorema se mantendría aislado y en peligro de

una total desatención antecesora de su extirpación del cuerpo matemático, pero su condición de teorema prevalecería. Porque una demostración es una garantía de que de las hipótesis se siga la conclusión y no necesariamente un certificado de fecundidad epistemológica. Por cierto, ¿qué tan segura es la garantía? En el capítulo cuarto se señaló la falibilidad de las computadoras y por consecuencia la de sus productos deductivos. Si las computadoras no son perfectas, ¿por qué habríamos de serlo nosotros? Si la historia evidencia varias fisuras en el desarrollo matemático, ¿por qué situarnos más allá de la historia en la histeria al insistir sobre la infalibilidad humana en las matemáticas? Acoto y remarco, falibilidad en las matemáticas humanas. La existencia de un universo ideal donde las verdades son eternas, bellas, etc.$^{\pi}$ la ignoro porque la ignorancia sobre su existencia será lo único que sabré de ella. Si las diatribas materiales destrozaron los sueños de la perfección wulfiana, ¿acaso las fallas compañeras usuales de los sistemas sociales no ocasionarían semejantes estragos en las demostraciones matemáticas? Porque es inobjetable, la gestación de las demostraciones para bien y para mal resiente y se nutre de una estructura social. Un taimado infante luego de haber decorado las paredes de su casa con sendos murales pintados con sus crayolas, supone el perdón del castigo si esconde sus manos manchadas en sus bolsillos y confía en la falta de perspicacia de su madre. Podemos escondernos detrás de las circunstancias, si las matemáticas están mancilladas por factores sociales es un mal necesario para los párvulos mortales pero si estuviésemos bajo el cobijo de la mirada de una abnegada madre entonces ella vería en el muro una obra de arte, una manifestación de la verdad y de la belleza, una representación pueril pero fidedigna de algo inmarcesible. Haciendo grandes concesiones, trasportemos la infalibilidad al nivel individual. Sabemos lo suficiente del cerebro para concluir ¿ ? En el nivel mental, los matemáticos se equivocan a cada rato, es más, los errores pueden servir de abono para futuros aciertos. No es factible argumentar en pro de la infalibilidad a partir de que un matemático siempre haga las cosas bien, esta tentativa carece de pies y cabeza. Además, ¿cómo saber cuando el individuo hierra o acierta? El individuo puede jurar por Gödel que su demostración es correcta, y de hecho puede serlo, pero es por la intervención de otros como la demostración se sustenta. La intromisión de los factores sociales en la validación de las demostraciones no puede quedarse afuera y con ella entran la vacilación quieta y la certidumbre inquieta. En cambio, la detección de errores desde el nivel personal puede llevarse a cabo sin tanto problema. Conforme el número o la calidad de inspectores se incrementan, la eficacia aumenta. En el peor de los casos, ¿la falibilidad en los procesos de detección de errores que tanto afecta? Aclaro, estos procesos fomentan pero no sostienen la validez de las demostraciones, su meta es encontrar errores y no garantizar su ausencia. Por lo cual, si la equivocación en ellos se presenta, una posible demostración válida por un fallo erróneo del perito al limbo se relega. Sin embargo, ¿qué tan comunes son las equivocaciones de esta naturaleza? Habría que escarbar en la historia, pero según me parece, repito, según me parece, son escasas y dispersas. Una situación típica en donde puedan temblar parecidas fallas, es cuando una nueva técnica deductiva una demostración tenga. Sin embargo no es el caso, si hay rechazo es por el recelo sobre esta técnica, no porque la demostración por estar plagada de errores sea incorrecta. Aunque las decisiones equivocadas sean comunes y diversas, la fiabilidad de las garantías nunca en amenaza se tienta sino la expansión del conocimiento al ralentí medra. Si los mecanismos de

detección de errores funcionan, a las engañosas demostraciones a la congeladora avientan, en caso contrario, algunos justos pagarán por los pecadores, no obstante el grado de confianza con dependencia en estos mecanismos constante se queda. En pos de la certeza, conviene hacer más finos a estos filtros, aumentar el número de inspectores y la calidad de los jueces es una atinada estrategia, por lo cual la interacción social en la validación de las demostraciones al mejorar el tamizado sopesa sus inherentes flaquezas. Si se quiere mantener el ideal (aunque siempre será un ideal) de la infalibilidad en las antropomatemáticas, el sentido hacia la rígida consecución de la verdad debe girar a la elástica expulsión del error. La verdad en duda estará, pero nunca la refutación. Los procesos de detección de errores pueden y deben tender a ser correctos, la completez es una sombra empero. Como lo señaló y mostró Shankar, la computadora puede unirse a esta empresa. La formalización asegure o no la veracidad de una demostración, sirve para hacer más precisa a la depuración. Este es (era) uno de los puntos del manifiesto QED, la formalización digital tiene la enorme utilidad de fortalecer a los mecanismos de detección de errores, la torre de Mizar tal vez no nos conduzca hasta el cielo de las verdades pero al menos sí nos aleja de la tierra de las equivocaciones. Olvida la certeza, enfrenta con entereza a la incertidumbre. Enclaustrados en esta cárcel donde la verdad es una felonía porque la ambigüedad rige con alevosía y desdén, confía, todo va a salir bien.

Este es el último párrafo. Para quienes como gatos hayan brincado desde el principio hasta aquí o para aquellos quienes después de leer la tesis su paciencia esté casi agotada, les informo que si x es una demostración entonces x es *convincente* o x es *inteligible* o x es *computable.* En la tesis desde el capítulo segundo esta caracterización fue glosada y desglosada. Su nacimiento y respaldo histórico/filosófico fue propiciado y proporcionado por algunas demostraciones computarizadas. Espero haberlos convencido de que ésta es una herramienta de análisis capaz de enriquecer a nuestra disciplina, pues presenta la más ligera posibilidad de obtener resultados y exhibe la menos liviana posibilidad de generar más preguntas. Con respecto a la demostración computarizada la tranquilidad me subyuga, confío en haber mostrado a la computadora como un instrumento capaz de obtener resultados deductivos y como candidato a ser matemático me vi obligado a asegurarlo.

BIBLIOGRAFÍA

En lugar de abrumar al lector con la impresión de las fichas bibliográficos de todos los artículos y libros utilizados para la elaboración de este ensayo (acción ya realizada en cada uno de los respectivos capítulos) señalaré cinco libros de diversa índole esenciales para la génesis y sustentación de este escrito, más aun, dichos libros son excelentes preámbulos para internarse en algunos de los tópicos discutidos y abusados en esta tesis.

- MacKenzie, Donald, *Mechanizing Proof: Computing, Risk, and Trust*, he MIT Press, 2001.
  Recopilación histórica exhaustiva, diligentemente seleccionada y documentada por el sociólogo Donald MacKenzie. En general aborda el problema de la confiabilidad de los sistemas computacionales, dedicando varios capítulos a la automatización de la generación/verificación de demostraciones. Ofrece un panorama muy amplio, incluso del subtema de las demostraciones computarizadas. Para aquellos interesados en la historia de las ciencias de la computaciòn y/o de las matemáticas, es un libro extremadamente recomendable. Advertencia, el libro tiene un barniz de sociología del conocimiento, aunque sólo charolará para los ojos extremadamente dogmáticos y sensibles.

- Saaty, Thomas y Kainen, Paul , *The four-color problem.* Dover, 1977
  Este libro es la base de la exposición de la demostración computarizada de Haken et al del 4CT, no obstante ofrece otros datos apetecibles para los gustosos de la teoría de gráficas y en particular del 4CT, v,gr. plantea otras formulaciones del famoso teorema.

- Tymoczko Thomas (editor), *New directions in the philosophy of mathematics*, Princeton University Press, New Jersey , 1988
  Libro dedicado para todos los espíritus que ardan por la filosofía de las matemáticas y que estén cansados de los rescoldos de las posturas más ortodoxas. Incluye artículos de Putnam, Wang, Lakatos, Chaitin, Hersh, entre otros. De aquí procede *The Four Color-Problem and Its Philosophical Significance* y por consiguiente muchas de las ideas de Tymoczko discutidas en esta tesis.

- Wos, L. y Pieper, G., *A Fascinating Country in the World of Computing,* World Scientific Press, Singapore ,1999
  Libro de texto para quienes estén interesados en el razonamiento automatizado a la sazón del paradigma clausular.

- Colburn, Timothy; Fetzer, James H. y Rankin, Terry L. (eds.), *Program Verification: Fundamental Issues in Computer Science*, Kluwer Academic Publishers, 1993
  Excelente opción para los interesados en la filosofía de las ciencias de la computación, fuente de varios de los artículos manoseados y citados en el capítulo cuarto.