UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Facultad de Ciencias

"DOMINIOS SEMANTICOS PARA LENGUAJES DE PROGRAMACION"

TESIS PROFESIONAL
que para obtener el título de :
M A T E M A T I C O
presenta :
VLADIMIR ESTIVILL CASTRO

México D.F. 1985

INDICE

INTRODUCCION

PRIMERA PARTE

CAPITULO I ELEMENTOS PRELIMINARES

Sección 1 Sintaxis, semántica y pragmática.

Sección 2 Sintaxis Concreta.

Sección 3 Descripción Semántica Formal.

Sección 4 Métodos de Semántica Formal.

Sección 5 Semántica Denotativa.

SEGUNDA PARTE

CAPITULO II LA CATEGORIA DE DOMINIOS

Sección 6 La teoría de Dana Scott.

Sección 7 Dominios mediante Sistemas de Información.

Sección 8 Elementos.

Sección 9 Intuición y presentaciones anteriores.

Sección 10 Morfismos de Dominios.

CAPITULO III CONSTRUCCIONES

Sección 11 Producto y Unión Ajena.

Sección 12 El espacio de funciones.

CAPITULO IV TEORIA DE DOMINIOS

Sección 13 Puntos fijos.

Sección 14 Consideraciones categóricas.

Sección 15 Dominios Reflexivos.

TERCERA PARTE

CAPITULO V APLICACIONES

Sección 16 Dominios sintácticos y semánticos.

Sección 17 Interpretación.

Sección 18 Procedimientos, ambientes y saltos.

Sección 19 Un lenguaje funcional.

Sección 20 Sintaxis sensible al contexto.

Sección 21 Dominios semánticos en PASCAL.

NOTACION

$\emptyset$	El conjunto vacio.
$\bigcap_{T} S$	La intersecciòn de los conjuntos S en la familia T
$\mathbb{N}$	Los nùmeros naturales
$\mathbb{P}(A)$	El conjunto de subconjuntos de A
$\bigcup\{ (u) \mid u \leq x \text{ u finito}\}$	La uniòn de los conjuntos (u) donde $u \leq x$ y u es finito
$\#(A)$	La cardinalidad de A
$\#(\mathbb{N}) = \aleph_0$	
$\forall$	para toda, para todo

INTRODUCION

La presente tesis introduce una Teoría de Computación que enfoca matemáticamente, en vez de operacionalmente, los aspectos de procesamiento en computadoras. La teoría se basa en que tipos de datos pueden definirse matemáticamente bajo una idea de aproximación, sus propiedades y los mapeos resultantes, lo cual conduce al desarrollo de una teoría matemática para la semántica de lenguajes de programación.

Es mediante los lenguajes de programación que el hombre consigue manejar máquinas computadoras , y es en estos lenguajes donde surge una importante y urgente oportunidad de estudio. La formalización sintáctica de los lenguajes de programación está muy avanzada mientras que la teoría semántica es incompleta. A pesar de contar con numerosos ejemplos es necesario proporcionar respuestas matemáticas a interrogantes como ¿Qué es un proceso computable ? ¿Cómo simula una máquina un proceso ? . Los programas entran naturalmente como descripciones de procesos, y es aquí donde aparece la importancia de una definición precisa del significado de un programa. Esta definición requiere explicar cuales son los objetos de cómputo (la estática del problema) y como son transformados (la dinámica).

La Teoría de Autómatas, como parte de una Teoría matemática de Computación , resulta de profundo interés para la parte dinámica y ha logrado formalizar parte de las descripciones, concentrandose en los aspectos algebráicos y considerando estados finitos de una máquina.

Parece ser, que el comprender las características de los llamados lenguajes de alto nivel proporcionará un nuevo paso en el desarrollo de la teoría matemática de Computación.

El trabajar con objetos finitos nos fuerza a pasar por varios niveles de descripción de los conceptos e ideas para llegar a la formalización y simulación de una máquina con las características adecuadas para su aplicación en la realidad. Estos niveles pueden ser exactos matemáticamente si encontramos la correcta abstración para representar las estructuras necesarias. La presente tesis muestra este esfuerzo.

La motivación para formular la teoría matemática presentada aquí es proveer una semántica matemática para los lenguajes de programación de alto nivel. La palabra matemática debe contrastar en el lector con la imagen operacional de la semántica. El significado matemático de un procedimiento debe entenderse como una función de un tipo de datos en las variables de entrada en el tipo de datos de la salida. No consideraremos el enfoque operacional que presenta una historia de los cálculos y operaciones obedeciendo la secuencia señalada por la definición del procedimiento e involucra una elección en representaciones finitas de datos, probablemente en algunos patrones de "bits". El punto es que matemáticamente las funciones son idependientes de sus nombres y maneras de calcularlas, y en este sentido son más simples que las secuencias de operaciones generadas paso a paso, en representaciones particulares.

La semántica matemática evita estas dificultades, y resulta más adecuada para estudiar problemas como la equivalencia de programas. Sin embargo el aspecto operacional no debe ser olvidado por completo, ya que finalmente el programa trabajará en una máquina, así pues la semántica formalizada matemáticamente ,que representa la primera parte de la definición de un lenguaje de programación, debe encaminarse naturalmente a una simulación operacional de los objetos abstractos. Al trabajar con funciones definidas matemáticamente, puede suceder que la función no sea calculable en sentido práctico, a pesar de conocerse que es computable; es importante pues que una teoría de cómputo, además de proveer abstraciones, presente realizaciones físicas para calcular.

El punto esencial que guìa la tesis es que hasta que no exista una definiciòn matemàtica global, no es posible determinar si alguna implantaciòn de un lenguaje es correcta, pues no existe un modelo objetivo, y comunmente un lenguaje de programaciòn està sujeto al compilador particular en que se trabaja.

El propòsito de una semàntica matemàtica es hacer que el conocimiento y entendimiento de un lenguaje sea visible. Esta nueva perspectiva se alcanza mediante la abstraciòn de ideas centrales en objetos matemàticos, los cuales ,pueden ser manipulados en formas matemàticas conocidas.

La Teorìa de Dominios dentro de la Semàntica Denotativa proporciona modelos para los espacios donde se definen funciones computables. Los tipos de espacios necesarios en Semàntica Denotativa no sòlo incluyen espacios complicados, (por ejemplo espacios de funciones) tambièn incluyen espacios definidos recursivamente (los que llamaremos Dominios Reflexivos). Son necesarias, ademàs construcciones especiales de Dominios (o funtores) para crear las estructuras deseadas.

Dentro de esta categorìa se desarrolla la Teorìa de Dominios para semàntica Denotativa, la cual intenta crear un modelo matemàtico para un sistema de tipos y explicar la nociòn de computabilidad con respecto a estos tipos (anteriormente espacios). La construcciòn de Dominios significa la justificaciòn de definiciones recursivas de tipos.

El beneficio de esta teorìa, consiste en la posibilidad de manejar tipos con elementos finitos. La desventaja parece ser que los Dominios deben poseer elementos parciales y elementos totales. En este enfoque fue donde inicialmente se trabajò y las primeras investigaciones fueron proyectadas bajo las definiciones de enrejados (redes o lattices). El oden parcial ($\leq$) para $x \leq y$ fuè utilizado para expresar que x era menos definido (mas parcial) que y. La relación ⊑ debía ser sometida a muchos axiomas para que tuviese las aplicaciones deseadas. El nuevo enfoque realiza la construcción basada en pocos axiomas.

Las ventajas son las siguientes:

.Definiciones simples de los conceptos.

.Propiedades especificas son demostradas en vez de ser asumidas como axiomas.

.Las definiciones son apoyadas fuertemente por la intuición y la construcción resulta natural.

.Los Dominios resultan más útiles.

.Aparecen más aplicaciones, pues es fácil producir Dominios más complejos.

El propósito de la presente tesis es presentar e iniciar el estudio de la Categoría desde el punto de vista de Sistemas de Información, más simple e incorporando muchos éjemplos. Los Dominios son presentados en base a la Teoría de Conjuntos con la ayuda de las ya mencionadas estructuras matemáticas de información .Estos Sistemas de Información son conocidos en la Lógica y su uso parece ir a la par con la intuición.

El objetivo central incluye llevar el trabajo en Teoría de Dominios, anteriormente fundamentado en ideas topológicas (Sistemas de Vecindades), al enfoque lógico basado en Sistemas de Información acompañado de gran cantidad de ejemplos. La tesis inserta este objetivo dentro del desarrollo de la Semántica Denotativa. Con esto en mente, la tesis utiliza la dirección de temas de Stoy[77] pero con un nuevo modelo matemático presentado por D. Scott[82]. Queda, la tesis conformada en tres partes.

La primera parte presenta el contexto en que se desarrolla este campo de investigación y debe considerarse también como una intriducción al tema; esta parte incluye un panorama de los objetivos de la formalización y sus fundamentos, la necesidad del uso de Dominios se plantea y se muestra un trazo de la dirección en que ha surgido y se dirige la teoría.

La segunda parte constituye el nucleo del trabajo, presenta la Categoría de Dominios rodeada de ejemplos. Las construcciones y realizaciones teóricas sobre Dominios quedan detalladas en esta parte bajo el enfoque de Sistemas de Información. Muchos de los ejemplos son tomados del enfoque topológico por D. Scott[81] y esto pretende completar la presentación inicial de Scott[82].

En la tercera parte se muestran algunos ejemplos de aplicaciones concretas y su desarrollo. Estos ejemplos son casos mucho mas globales que los ejemplos anteriores.

PRIMERA PARTE

Capítulo I

ELEMENTOS PRELIMINARES

Sección 1 Sintaxis, semántica y pragmática.

Entendemos que un lenguaje de programación es un sistema de notaciones para describir cálculos y operaciones. Un lenguaje de programación debe entonces ser capaz de describir (para programadores y usuarios de programas) algoritmos e indicar (para una implantación en computadora) las operaciones. Las grandes e inmensas diferencias entre los seres humanos y las computadoras dificultan encontrar una notación adecuada a las capacidades de humanos y máquinas; pensamos entonces en lenguajes en favor de los humanos como "lenguajes de alto nivel", mientras que aquéllos a favor de las computadoras los llamamos "lenguajes de bajo nivel".

En este sentido los casos extremos que se presentan son, en principio los siguientes: El lenguaje más poderoso en una computadora particular es su lenguaje de máquina, el cual proporciona control sobre todos los recursos disponibles en la máquina. Sin embargo, este lenguaje no puede usarse en otras computadoras y para los programadores es muy difícil escribir o leer programas en lenguaje de máquina, simplemente porque los humanos no podemos afrontar la falta de estructura tanto en los programas (sucesiones de instruciones de máquina) como en la representación de los datos (palabras o celdas de memoria).

Por otra parte se encuentran los lenguajes naturales (como el Español o Francés). Pero en la mayoría de los campos de la ciencia y la tecnología ha quedado demostrado que las notaciones simbólicas formales, principalmente de las Matemàticas y la Lògica, son indispensables para la formulaciòn de conceptos e ideas del razonamiento cientìfico. Desafortunadamente, la mayorìa de las notaciones y conceptos matemàticos no son implantables, por diversas razones, en una computadora.

Se espera que en el futuro, el lenguaje de programaciòn ideal combinarà las ventajas de los lenguajes de màquina y la notaciòn matemàtica; a pesar de la fuerte investigaciòn en este sentido la tarea parece aùn extremadamente difìcil. Varios lenguajes existentes han sòlo logrado combinar algunas caracterìsticas y en la mayoria de los casos el resultado final ha sido difìcil de implantar además de que su uso ha resultado incòmodo. Ejemplos conocidos de estos esfuerzos lo constituyen los lenguajes LISP, PROLOG y EUCLID.

Existen tantos lenguajes de programaciòn, tan complejos e irregulares, que resulta casi imposible aprender con detalle cada caracterìstica de cada lenguaje (aunque se trate de los 15 màs importantes). Afortunadamente, muchos conceptos son comunes a los lenguajes de programaciòn; incluso aquellas caracterìsticas que parecen totalmente diferentes son globalmente generales a los mismos principios. Dirigiremos nuestra atenciòn hacia los aspectos generales y conceptos fundamentales.

El estudio de los lenguajes naturales tanto como los artificiales (en particular los de programaciòn) es clasificado tradicionalmente en tres àreas principales:

(a)Sintaxis, la cual concentra sus esfuerzos al estudio de la forma, la estructura y la composiciòn de frases y expresiones en el lenguaje. Es decir la correcta estructuraciòn.

(b)La Semàntica enfoca el significado, las ideas y conceptos de las estructuras o frases del lenguaje.

Finalmente, el campo de

(c) la Pragmàtica estudia los origenes, usos y efectos del lenguaje. Asì, los aspectos pragmàticos de un lenguaje de programaciòn incluyen tècnicas de implantaciòn, metodologìa de la programaciòn o historia del desarrollo del lenguaje.

Las fronteras de esta divisiòn no estan claramente marcadas, algunos aspectos de los lenguajes pueden ser vistos o tratados como sintàcticos o como semànticos, pero para nuestros objetivos la clasificaciòn no darà lugar a confusiones.

El presente trabajo enfoca sus intereses hacia la formalizaciòn de la investigaciòn semàntica. La formalizaciòn sintàctica esta pràcticamente completa, ya que es esencialmente un problema mucho màs sencillo. Presentamos a continuacuòn las principales ideas formales de la Sintaxis.

Secciòn 2 Sintaxis Abstracta y Concreta

En realidad es sencillo especificar informalmente la sintaxis de un lenguaje de programaciòn, pero tambièn es sencillo y mucho màs conveniente una descripciòn formal especìfica. la notaciòn que se utiliza para describir la sintaxis de un lenguaje recibe el nombre de meta-lenguaje. En esta secciòn presentaremos un meta-lenguaje muy usado para describir la sintaxis y posteriormente abandonaremos toda discusiòn de los aspectos sintàcticos.

La sintaxis de un lenguaje de programaciòn puede ser especificada simplemente enumerando las clases sintàcticas y listando todas las formas posibles de las estructuras de dichas clases sintàcticas. De esta manera, la primera parte de una definiciòn sintàctica enlista las clases sintàcticas y los sìmbolos para denotar elementos arbitrarios de cada clase, en la segunda parte aparecen las diferentes opciones de cada clase no elemental a la derecha del sìmbolo ::= separadas por el sìmbolo | (ver ejemplo en: APLICACIONES "sintaxis abstracta de PASCAL").

La sintaxis abstracta dice cuales son las estructuras sintàcticas disponibles pero no especifica si una cadena de caracteres està bien formada, ni tampoco su estructura; por ejemplo, bajo la sintaxis abstracta de PASCAL

"if E then C" e "if E then C else C"

son señaladas como estructuras correctas de enunciados pero no especifica claramente si

"if a then if b then p else q"

es un texto correcto y en este caso ¿a que "then" corresponde el "else"?.

Estos aspectos se tratan en una Sintaxis Concreta o Gramàtica.

El meta-lenguaje Backus-Naur-Form(BNF) es el màs conocido para especificar la sintaxis concreta de un lenguaje de programaciòn . Se especifica una tetrada constituida por un conjunto de elementos No-terminales, un conjunto de sìmbolos terminales, un sìmbolo inicial distinguido y un conjunto de reglas o producciones.

⟨expresiòn⟩ ::= ⟨tèrmino⟩ | ⟨expresiòn⟩⟨operador suma⟩⟨tèrmino⟩

⟨tèrmino⟩ ::= ⟨factor⟩ | ⟨tèrmino⟩⟨operador producto⟩⟨factor⟩

⟨factor⟩ ::= ⟨identificador⟩ | ⟨literal⟩ | (⟨expresiòn⟩)

⟨identificador⟩ ::= A | B | C | ... | Z

⟨operador suma⟩ ::= + | -

⟨operador producto⟩ ::= * | / | mod

⟨literal⟩ ::= 0 | 1 | ... | maxint

Tabla #1

Por ejemplo, considerese la gramática para expresiones aritméticas dada por las producciones de la Tabla #1.

Los objetos no terminales de la gramática son representados entre ⟨ ⟩ y los objetos terminales aparecen tal cuales, el símbolo inicial (no terminal) es el elemento a la izquierda de la primera regla.

En el ejemplo de la tabla #1, la primera regla especifica que una expresión es un término, o una expresión seguida de un operador de suma seguido a su vez de un término. La interpretación de las otras reglas es análoga.

Las producciones señalan cuales son las estructuras del lenguaje y los textos de expresiones que pueden ser reconocidos como válidos o no a partir de las reglas.

Por ejemplo, A+B*C es reconocido como una expresión cuya estructura es:

⟨expresión⟩ ⟨operador suma⟩ ⟨término⟩

donde el término es B*C. Similarmente un análisis indica A*B+C como una estructura resultante de:

⟨expresión⟩ ⟨operador suma⟩ ⟨término⟩

donde ahora el término es C. (ver tabla #2)

```
                                   ⟨expresion⟩
                                        |
   ⟨expresión⟩           ⟨operador suma⟩            ⟨término⟩
        |                      |                         |
    ⟨término⟩                  +         ⟨término⟩ ⟨operador producto⟩ ⟨factor⟩
        |                                    |             |               |
    ⟨factor⟩                             ⟨factor⟩          *       ⟨identificador⟩
        |                                    |                             |
 ⟨identificador⟩                      ⟨identificador⟩                      C
        |                                    |
        A                                    B
```

Tabla #2

Entonces A+B*C es equivalente a A+(B*C) y A*B+C es equivalente a (A*B)+C. Es decir la sintaxis indica que operadores tienen mayor precedencia. (Los operadores de producto tienen mayor prioridad que los de suma).

En resumen, la gramática hace posible recuperar la sucesión de reglas aplicadas a partir del símbolo inicial para obtener el texto o frase particular del lenguaje. A tal sucesión se le denomina una derivación. Una descripción sintáctica formal se llama ambigua si para alguna frase del lenguaje existen dos o mas derivaciones distintas. Por ejemplo si se remplaza la primera producción de la tabla #1 por

<expresión>::=<término>|<expresión><operador suma><expresión>

el texto A-B-C sería reconocido como A-(B-C) y (A-B)-C las cuales no son equivalentes. No es deseable tener una gramática ambigua pero no existe un método general para demostrar si una gramática es ambigua; sin embargo es posible probar que una gramática particular no es ambigua. (Para una mayor discusión sobre las metodologías de representación y reconocimiento de las sintaxis en lenguajes de programación el lector puede consultar Aho y Ullman [79]).

BNF no es la única notación para definir formalmente la sintaxis de un lenguaje de programación, (existen otras descripciones como Metacobol o Sintaxis por Diagramas) pero el objetivo de nuestra discusión es mostrar que los problemas de descripción sintáctica no son de altísima complejidad por lo que la descripción formal se encuentra muy avanzada.

Sección 3 Descripción Semántica Formal

La descripción formal de la semántica de un lenguaje de programación es una especificación precisa del significado de los programas para que sea utilizada por programadores, diseñadores e implantadores de lenguajes, asi como en investigaciones teòricas de lenguajes y computabilidad.

Originalmente el objetivo de utilizar una definiciòn suficientemente precisa del significado de los programas fue la correcta construcciòn de interpretes y compiladores, posteriormente tambièn se intento realizar afirmaciones de los programas rigurosamente fundamentadas, el objetivo tal vez màs importante a futuro parece ser el diseño de lenguajes con descripciones cada vez màs simples. Todo estó lleva a que los programas, al construirse naturalmente, sean correctos, evitando la constante y costosa correcciòn y prueba.

Para alcanzar estos objetivos, las descripciones formales deben proporcionar conceptos independientes a la implantaciòn particular, tècnicas no ambiguas de especificaciòn y una teorìa rigurosa que soporte las afirmaciones sobre programas. Explicaremos a continuaciòn el porque de estos requisitos y como las diferentes descripciones semànticas formales proponen su soluciòn.

Las descripciones semànticas no formales, comunmente necesitan una base de conocimientos y conceptos sobre programaciòn para explicar el significado de una frase del lenguaje; por ejemplo los conceptos de variable(distinto al concepto matemàtico), ambiente, valor actual o valor de una expresiòn son utilizados en la descripciòn de PASCAL(Jensen y Wirth [74]) para explicar el significado de la asignaciòn. En la mayorìa de los casos esto concéptos son explicados en tèrminos de la implantaciòn, lo cual hace que diferntes implantaciones señalen significados distintos a los mismos conceptos. En realidad, la esencia de las ideas no depende de ninguna màquina particular. Una definiciòn semàntica formal, produce un conjunto universal de conceptos en tèrminos de objetos matemàticos abstractos, lo cual permite definir, por ejemplo la asignación sin hacer referencia a ningún mecanismo de implantación.

Las técnicas no ambiguas de especificación son necesarias para que no haya confusión y así, usuarios e implantadores del lenguaje reciban del diseñador una descripción comprensible, sencilla y completa. BNF junto con las técnicas de lenguajes formales ha suministrado una manera adecuada de especificar y pensar acerca de la sintaxis en los últimos 15 años. Desafortunadamente, al enfrentar una problemática más ardua, el desarrollo de la notación y conceptos en semántica ha sido más lento; lo cual ha resultado en que la mayoría de los lenguajes son definidos en Inglés. A pesar de que estas descripciones son aparentemente muy claras, frecuentemente las encontramos inconsistentes e incompletas. La experiencia ha mostrado que no es posible alcanzar la precisión deseada mediante descripciones informales.

Las técnicas de semántica formal posibilitan las demostraciones rigurosas de afirmaciones sobre propiedades de los programas y los lenguajes de programación. Esto se debe a que para probar que un programa es correcto es necesario mostrar que el significado del programa y el significado deseado coinciden, y por lo tanto, ambos significados deben ser claros formalmente. Es gracias a la herramienta de la semántica formal que es posible probar cuales reglas de inferencia sobre programas son válidas en algún lenguaje particular. Los intentos por diseñar lenguajes tales que la intuición sobre reglas de inferencia correspondiese al resultado de las intrucciones y no fuese necesaria la formalización del programa mostraron un gran fracaso cuando los diseñadores del lenguaje EUCLID no pudieron hacer afirmaciones sobre programas en tal lenguaje. (Ver London[78], Gordon[79]).

Otro reciente uso de la semántica formal, se deriva del hecho de que las descripciones formales son legibles, representables y manejables por computadoras. Esto permite la produción de programas para manejar las definiciones semànticas y utilizarlos como generadores de sistemas en implantaciones, anàlogamente a como se producen a partir de las definiciones sintàcticas, los reconocedores sintàcticos o herramientas para compiladores (Ver Moses [76]).

Son claras las ventajas alcanzadas bajo el uso de una descripciòn matemàtica de la semàntica, pero ¿Còmo se realizan estas descripciones?, esta pregunta es respondida, en tèrminos generales a continuaciòn.

Secciòn 4 Mètodos de Semàntica Formal

Existen tres metodologìas principales para presentar una descripciòn semàntica de los lenguajes de programaciòn. Estas son: el enfoque operacional, (representado principalmente por el Vienna Definition Language: VDL), el enfoque axiomàtico y el enfoque denotativo. Cabe subrayar que èstos mètodos no son rivales. Realmente todas son semànticas matemàticas. Sin embargo algunas presentan desventajas con respecto a los objetivos inicialmente señalados y esto ha impulsado los esfuerzos hacia las otras en busca de respuestas a problemas particulares.

El enfoque operacional se esfuerza por explicar las operaciones con las cuales las màquinas ejecutan los programas. Estas operaciones pueden ser descritas mediante notaciones orientadas hacia las màquinas o utilizando tècnicas no denotativas como la Teorìa de Autòmatas. La idea consiste en definir perfectamente una màquina abstracta y el còdigo de operaciones en ella, a pesar de que el modelo matemàtico asì construido es totalmente impràctico, la descripciòn es tan simple que no existe confusiòn alguna sobre el còdigo de operaciones. Finalmente el significado de un programa es dado por una traducciòn a este còdigo.

La primera desventaja de este mètodo se presenta al definir la semàntica de las intrucciones en la màquina abstracta (la ùnica base para esto es su sencillez). Es decir el problema sòlo se traslada a un nivel mas bajo, lo cual complica las aplicaciones pràcticas. Sin embargo, lo que resulta aùn màs serio es que el significado de un programa se describe en tèrminos de simular operaciones y este no es nuestro trabajo, sino el trabajo de la computadora. Una desventaja màs resulta de que programas con significado operacional no corresponden a las estructuras usadas por los programadores, tanto en cuestiòn de instruciones, como de datos, esta falta de relaciòn hace al enfoque operacional muy incòmodo.

La principal ventaja de este mètodo es que sugiere muchas tècnicas de implantaciòn y facilita la construciòn de compiladores.

El enfoque axiomàtico se ocupa de proporcionar axiomas y reglas de inferencia para realizar deduciones sobre programas. A cada enunciado del lenguaje se le asocia uno o màs axiomas, que presentan una posible afirmaciòn con respecto a la ejecuciòn del enunciado o instrucciòn, en tèrminos de lo que serìa cierto despuès y antes de la ejecuciòn.

La principal desventaja del enfoque axiomàtico es que sòlo es aplicable a situacuiones muy simples y en aplicaciones reales resulta muy difìcil señalar los axiomas, incluso para los diseñadores del lenguaje. A pesar de esto, esta tècnica ha sido muy exitosa para la demostraciòn de la correciòn de programas.

En este trabajo nos concentraremos en Semàntica Denotativa, y explicaremos esta àrea en la secciòn siguiente.

Secciòn 5 Semàntica Denotativa.

La idea básica de este enfoque consiste en definir funciones (semánticas), las cuales llevan las estructuras sintácticas en objetos matemáticos abtractos que modelan su significado. Esto es, construcciones sintácticas denotan objetos matemáticos y las funciones que nos traducen las construciones sintácticas en los objetos matemáticos que denotan son las funciones semánticas. Las definiciones de las funciones semánticas son asi, denotativas, y estructuradas de manera que el significado de una frase compuesta se expresa en términos del significado de sus componentes inmediatos.

La construción de estas funciones (usualmente definidas en términos de objetos descritos recursivamente) plantea la problemática de definir sus dominios (también usulamente definidos de manera recursiva). Esta tesis se concentra en la construción de los codominios de las funciones semánticas.

La Semántica Denotativa permite manejar conceptualmente los significados de los programas. Es posible la demostración de la existencia de un lenguaje con los significados específicos (lo cual no ha sido realizado desde el enfoque axiomático). La presente tesis se encuentra dentro del marco de investigación de los fundamentos matemáticos de la computación.

Es conveniente insistir en que los métodos de la semántica formal estan muy relacionados pero parece ser que el enfoque operacional es más adecuado al implantador, el axiomático para el programador y el denotativo para el diseñador. Incluso invitamos al lector interesado a consultar los Capítulos 13 y 14 de Stoy[77] donde las herramientas de los enfoques axiomático y operacional son utilizadas para una definición denotativa.

Presentamos a continuación un ejemplo, y recordamos que la Semántica Denotativa considera a las construciones sintácticas correctas como notaciones de objetos matemàticos. En realidad el manipular objetos abstractos es difìcil sin usar una notaciòn particular. Por ejemplo para hablar de los nùmeros naturales necesitamos algùn conjunto de nombres como: {1,2,3,4,5...} o tal vez {I,II,III,...} o {uno,dos,tres,...}. Como no podemos manipular entidades abstractas, sino hablar de ellas, tal vez las ùnicas cosas que existen son los lenguajes, no entraremos en discusiones filosòficas de esta ìndole, pero consideramos importante señalar que en el presente trabajo concevimos siempre diferentes los objetos matemàticos de la notaciòn para hablar de ellos.

Ejemplo:Notaciòm en base B.

Sea $B \in \mathbb{N}$ un nùmero natural $B \geq 2$ (y por el momento $B \leq 10$), sabemos que todo nùmero entero puede ser escrito en base B si tenemos B sìmbolos diferentes. (En realidad estos sìmbolo ya fueron diseñados y son los dìgitos 0,1,2,3,...,9). La escritura de un nùmero en base B es una notaciòn para los nùmeros enteros, construidos a partir de los naturales y estos fundamentados en los axiomas de Peano (ver Stoll[63]). Es importante distinguir que los enteros son independientes de la notaciòn decimal usual para hablar de ellos.

Presentamos aquì la sintaxis y semàntica de este lenguaje en la Tabla #3. Supondremos la existencia de B sìmbolos distintos, los cuales denotamos por $\underline{0}, \underline{1}, \ldots, \underline{B-1}$. Supondremos tambièn la existencia de un dominio sintàctico Not, que tiene un subconjunto llamado Pos y un Dominio semàntico Ent (que contiene a los enteros) para la funciòn semàntica que definiremos a continuaciòn.

Sintaxis Concreta

No-terminales={N, M} Terminales=$\{\underline{0}, \underline{1}, \ldots, \underline{B-1}\}$

$N \in Not \quad M \in Pos$

$\langle N \rangle ::= \langle M \rangle \mid \langle -M \rangle$

$\langle M \rangle ::= \underline{0} \mid \underline{1} \mid \ldots \mid \underline{B-1} \mid \langle M \rangle \underline{0} \mid \ldots \mid \langle M \rangle \underline{B-1}$

Dominio Semántico

Ent={0, 1, -1, 2, -2, ...} (Dominios de los números enteros)

Función Semántica

$\mathbb{N}: Not \rightarrow Ent$

$\mathbb{N}[[\ \underline{0}\ ]] = 0$

$\mathbb{N}[[\ \underline{1}\ ]] = 1$

$\mathbb{N}[[\ \underline{B-1}\ ]] = B-1$

$\mathbb{N}[[\ M\underline{0}\ ]] = B \times \mathbb{N}[[\ M\ ]]$

$\mathbb{N}[[\ M\underline{1}\ ]] = (B \times \mathbb{N}[[\ M\ ]]) + 1$

$\mathbb{N}[[\ M\underline{B-1}\ ]] = (B \times \mathbb{N}[[\ M\ ]]) + B-1$

$\mathbb{N}[[\ -M\ ]] = - \mathbb{N}[[\ M\ ]]$

Tabla #3

La sintaxis la presentamos en BNF e indica simplemente que Not consiste de todas las sucesiones finitas arbitrariamente largas de símbolos de $\{\underline{0}, \underline{1}, \ldots, \underline{B-1}\}$, opcionalmente antecedidas del signo -. N es un elemento arbitario en Not y M denota un elemento arbitrario en Pos (las notaciones de los positivos). Para definir la función semántica $\mathbb{N}$ en la tabla #3, ésta se define en todos los casos sintácticos posibles, (en algunos de ellos recursivamente). La función asocia a cada notación en base B (a cada número en base B) un único número entero, podemos decir que en cierta forma la función evalua la notación. Es muy importante observar la diferencia entre símbolos subrayados y aquellos que no lo estan; los símblolos subrayados son símbolos (letras) del lenguaje Not, es decir, el lenguaje que estamos definiendo, mientras que los no subrayados representan objetos matemáticos ya conocidos (como el cero, la unidad), las funciones binarias recursivas de los enteros(suma y multiplicación) y la función inverso aditivo. M es una variable sobre Not y los símbolos subrayados son símbolos considerados tal cuales, el escribirlos juntos sólo por ejemplo indica una concatenación del valor de M y el símbolo $\underline{0}$. Para hacer evidente que la función va de estructuras sintácticas utilizamos [[ ]] para indicar el argumento, el cual es alguna expresión del metalenguaje sintáctico.

En muchos casos prácticos no se especifican los conjuntos de terminales y no terminales pues pueden deducirse de las producciones (por la notación ⟨ ⟩). Ahora probaremos que este es el lenguaje que buscamos.

AFIRMACIÓN: Toda notación en Not representa un único entero y todo número entero se representa por al menos una notación en Not ⟨⟩.

Esto es decir que Not es un conjunto de nombres para los números enteros.

DEMOSTRACION.-La demostración es sencilla y se sigue por inducción sobre la longitud de N.

Primera parte. Toda notación representa un único entero.

Caso 1)Supongamos N no tiene signo, (es de la forma M).

Si la longitud de N es 1, entonces $N=\underline{0}$ o $N=\underline{1}$ ...o $N=\underline{B-1}$ en cuyo caso N representa exclusivamente al cero o al uno ,etc.

Supongamos que todas las notaciones de longitud k representan un único entero, y sea N de longitud k+1 y sin signo; luego se tiene que exclusivamente,

$$N=M\underline{0} \text{ o } N=M\underline{1} \text{ o } \ldots \text{ o } N=M\underline{B-1}$$

con M de longitud k, supongamos N=M$\underline{b}$ y por hipótesis de inducción que M representa al entero z, entonces N representa a:

$$(z \times B) + b$$

lo cual completa la inducción.

Caso 2)Si N tiene signo menos, es de la forma -M, entonces M no tiene signo menos y por el caso 1) M representa un único entero z, lo cual implica N representa a -z.

Segunda parte. Para todo entero existe al menos una notación en Not.

Sea z un entero arbitrario, si z es positivo aplicamos inducción. Si z=0 se representa por $\underline{0}$, supongamos todo número positivo menor que z es representable en Not, como

$$z=qB+r, \quad B>r\geq 0 \ , B\geq 2 \text{ tenemos } q<Z$$

y entonces q se representa como algún M en Not, pero entonces z se representa como M$\underline{r}$. Finalmente si z<0 entonces -z se representa como algún M, lo cual implica que z se representa como -M.

[].

Obsérvese que la primera parte implica que la función semántica toma valores enteros y está bien definida, la segunda parte es equivalente a que la función sea sobre(en los enteros).

El lenguaje que hemos definido formalmente es simplemente la notación posicional para los enteros; esta notación no es obvia y es uno de los más grandes inventos matemáticos de la humanidad. (Nótese que otros lenguajes totalmente diferentes han sido utilizados para denotar los mismos objetos matemáticos, como los Números Romanos). Esta notación hizo mecánicas las operaciones aritméticas y permitió concentrar el pensamiento matemático en algo más. La espectativa de alcanzar nuevos niveles de abstración mediante la formalización semántica (es decir una adecuada notación de los lenguajes de programación) aparece en el horizonte.

SEGUNDA PARTE

Capítulo II

LA CATEGORIA DE DOMINIOS

La construcción de funciones semánticas comienza por definir sus dominios. Sorprendentemente en el capítulo anterior utilizamos la palabra Dominio en lugar de Conjunto para nuestro ejemplo. Esto se debe a razones matemáticas técnicas que son el objeto de nuestro estudio. Es decir, un Dominio posee para nosotros una estructura matemática distinta que lo diferencia del concepto que aparece en la definición de función matemática. En realidad los Dominios son conjuntos con una estructura particular. Las ideas intuitivas y las aplicaciones, si se manejan únicamente bajo la Teoría de Conjuntos, es decir dentro de la Categoría de Conjuntos sin más estructura, enfrentan serios problemas.

Un problema difícil de resolver formalmente aparece cuando los conjuntos sobre los cuales deben definirse funciones semánticas deben ser definidos recursivamente por consideraciones prácticas; por ejemplo, el uso de procedimientos con efectos colaterales en un lenguaje de programación (ver ejemplos en las APLICACIONES) lleva a que los Dominios tengan una forma como en la Tabla #4.

Las ecuaciones de la tabla son recursivas; Estado esta definido en términos de Salida, la cual esta en términos de Valor, el cual esta en términos de Procedimiento y éste nuevamente en términos de Estado.

Estado=Memoria x Entrada x Salida

Memoria=Id -> [Valor+{noligado}]

Salida=Valor$^\infty$

Entrada=Valor$^\infty$

Valor=Número + BOOL + Procedimiento

Procedimiento=Estado -> [Estado + {error}]

Tabla #4

En la Categoría de Conjuntos no siempre existen conjuntos que satisfacen las ecuaciones de este tipo, principalmente por problemas de Cardinalidad. Afortunadamente el trabajar en la Categoría de Dominios nos permite interpretar los operadores +, x , -> e ∞ de manera que construyamos soluciones. Es decir, las ecuaciones en general no pueden siempre ser resuletas pero al restringir nuestro dominio de discurso podemos lograr un gran avance.

Algunas de las ideas principales para obtener soluciones se basan en que A->B ⊆ {f|f:A->B} (el espacio de funciones entre Dominios) sea un Dominio donde los mapeos preservan la estructura, y que el = es una representación para isomorfismo y no igualdad.

Finalmente, el hablar de conceptos de programación de alto nivel nos lleva a considerar el espacio de funciones como también un objeto; como una función en general es un objeto infinito, surge la idea de aproximación por objetos finitos (como se aproximan valores en Análisis Numérico). Sin embrago, resulta aún mas difícil de justificar formalmente los conceptos operacionales de programación como procedimientos no restringidos que permiten procedimientos como argumentos. Matemáticamente esto es, funciones que deben estar bien definidas para ser aplicadas a todas las funciones. Hasta ahora, si no restringimos nuestro dominio de discurso no podemos permitir con toda la libertad la aplicaciòn de funciones a funciones sin llegar a ser inconsistentes. La teorìa matemàtica presentada a continuaciòn logra restringir las funciones para que funciones puedan ser aplicadas (incluso a si mismas) consistentemente.

Es importante señalar que el problema de aplicaciòn recursiva (a si misma) surge en la interpretaciòn matemàtica de efectos colaterales y almacenamiento de instrucciones en lenguajes de programaciòn.

Para el problema de almacenar instrucciones, operacionalmente existen muchas soluciones, pero matemàticamente debemos definir una funciòn que enlace los contenidos de las localidades o celdas de memoria. Es decir debemos definir una funciòn matemàtica ,llamèmosla E (estado de la memoria) de las celdas de memoria en los posibles contenidos. Esta funciòn nos dice, dada una localidad de memoria, el valor del contenido ahì almacenado.

$E:L \to V$ donde $k \in L$ (el conjunto de localidades de memoria) y $E(k) \in V$(el conjunto de valores posibles). Si S es el conjunto de Estados ¿ Què es una instruciòn del lenguaje?. Esta es una modificaciòn de los valores almacenados en memoria, matem'ticamente una funci'n $Y:S \to S$
a o

Como operacionalmente las instrucciones del programa son almacenadas, la mas directa formalizaciòn de esto nos lleva a suponer que si E es el estado actual de la màquina y $k \in L$ es donde se guarda una instrucciòn. es decir E(k) es una instrucciòn, tenemos $E(k):S \to S$ pero ¿Què es E(k)(E)?. En realidad el enfoque operacional guarda un còdigo de la instrucciòn pero para realizar la descripciòn formal de tal lenguaje de programaciòn, nos enfrentamos a serias dificultades.

Seccuòn 6 La Teorìa de Dana Scott

Inicialmente la formalización comenzó con la idea de aproximar los elementos infinitos de las funciones semánticas mediante elementos finitos. Se trabajó entonces con ordenes parciales que indicaban una relación entre mayor y menor contenido de información; posteriormente se observó que funciones con sentido práctico entre estos conjuntos (ordenes parciales) debían ser monótonas, es decir a mayor información como datos de entrada, mayor información sería producida como salida. Esta estructura no fue suficiente, pues fue necesario que al aproximar un elemento por elementos finitos el valor de la función en el elemento fuese aproximado por los valores de la función en los elementos finitos. Esto llevó a que las funciones fuesen además continuas y el enfoque se envolvió por conceptos topológicos.

Estos enfoques permitieron uno de los mayores avances en la descripción semántica formal de los lenguajes de programación; que tuvo lugar cuando Dana Scott mostró una Categoría donde podía interpretar consistentemente definiciones recursivas, (tanto de funciones como Dominios).

Muchas de las ideas sobre Semántica Denotativa habían surgido mucho antes, por ejemplo el pionero de las definiciones denotativas, Stratchey, publicó en 1966 "Towards a Formal Semantics" y anteriormente John Mc. Carthy había logrado esbozar la semántica denotativa de un subconjunto de ALGOL-60.

En resumen, lo que D. Scott logró fue fundamentar la semántica denotativa al dar una estructura a los conjuntos para que formacen los objetos de una Categoría y definir x, + y -> en ellos, mostrar como los elementos de los Dominios eran definidos, como aparecía la idea de aproximar y como los Dominios mismos podían ser definidos recursivamente.

D. Scott definió un lenguaje LAMBA, el cual en principio es capaz de expresar la semántica de cualquier lenguaje, todas las funciones o funcionales computables son definibles en LAMBA. Por lo tanto, todo lenguaje definido en LAMDA es implantable en principio. Ademas, LAMBDA es un lenguaje matemático(notación matemática) que no incluye actualizaciones ni saltos, y el ligado de identificadores o variables se realiza en la forma matemática convencional. Esta propiedad permite que LAMBDA sea matemáticamente manipulable en demostraciones. Sin embargo, a pesar de su importancia teórica al garantizar que todo lenguaje tiene almenos una definición semántica (como subconjunto de LAMBA), el lenguaje LAMBA resulta totalmente impráctico.

Es importante indicar que las poco fundamentadas teorías y los primeros e inconsistentes trabajos conducen a versiones modernas. Mucho se avanzó en Semántica Denotativa antes de que existiese una demostración de que los objetos y modelos manejados tenían una fundamentación matemática; pero la mezcla de los conceptos teóricos ha contribuido a enriquecer las técnicas de descripción. Las ideas originales y el trabajo inicial en este sentido son de inclaculable valor.

Los agradables recultados teóricos obtenidos hasta ahora han impulsado la investigación en esta área, y las construciones teóricas iniciales(como el mismo D. Scott[82] ha dicho) que requerían de muchas herramientas, suposiciones y restricciones evolucionan hacia desarrollos más claros intuitivamente y más completos formalmente. Presentamos a continuación un desarrollo más sencillo y elegante de la Teoría de Dominios.

Sección 7 Dominios mediante Sistemas de Información

Intuitivamente, un Sistema de Información es un conjunto de proposiciones o afirmaciones sobre "posibles elementos" del Dominio buscado. El sistema debe contener suficientes proposiciones para distinguir dos elementos distintos, y entonces de manera abstracta podemos identificar el elemento con el conjunto de todas las proposiciones ciertas de él.

Así, dando proposiciones y afirmaciones sobre un elemento, y agrupandolas, construimos la definición del elemento. Pensamos en agrupaciones finitas, pues son las almacenables en una computadora, y pensamos en elementos infinitos como aquellos que requieren de una definición con una infinidad de afirmaciones.

De esta manera intuitiva, elementos parcialmente definidos poseen conjuntos pequeños de propiedades o afirmaciones, mientras que elementos totalmente definidos se identifican con conjuntos grandes de afirmaciones, posiblemente maximales. Para hacer estas ideas precisas presentamos esta:

<u>Definición 7.1</u> Un Sistema de Información es una estructura

$$(D, \Delta, Con, \vdash)$$

donde:

D es un conjunto(el conjunto de datos o proposiciones).

Δ es un elemento distinguido de D(el menor informante).

Con es un conjunto de subconjuntos finitos de D(los conjuntos de datos consistentes).

$\vdash$ es una relación binaria entre elementos de Con y D (relación de implicación).

tales que Con satisface los siguientes axiomas $\forall\ u, v \subseteq D$ finitos:

(i)Si $u \subseteq v$, $v \in Con \Rightarrow u \in Con$

(ii)Si $X \in D \Rightarrow \{X\} \in Con$

(iii)Si $u \vdash X \Rightarrow u \cup \{X\} \in Con$

y $\vdash$ satisface los siguientes axiomas $\forall\ u, v \in Con$ y $\forall\ X \in D$

(iv) $u \vdash \Delta$

(v) Si $X \in u \Rightarrow u \vdash X$

(vi) Si $u \vdash Y$ $\forall$ $Y \in v$ y $v \vdash X \Rightarrow$ $u \vdash X$ ().

Esta definición dice que: (i) Con es cerrado bajo subconjuntos, (ii) contiene todos los singoletes, (iii) la unión de un dato implicado por un conjunto consistente, al conjunto, es consistente, (iv) Δ es implicado por cualquier cosa (v) $\vdash$ es reflexiva y (vi) es transitiva.

Para pensar que quiere decir esta definición, pensamos en los elementos de Con como conjuntos finitos de datos consistentes, algunos de los cuales proveen más informacion que otros. La proposición que no da ninguna información es Δ. Los objetos de datos nos dan información sobre posibles elementos del Dominio que construiremos, de manera que si utilizamos únicamente Δ obtenemos el elemento menos definido denotado por $\perp$. No cualquier colección de datos describe un posible elemento, para esto utilizamos la noción de consistencia, si $u \in$ Con es falso entonces las proposiciones en u no pueden aplicarse a un mismo elemento simultaneamente. El significado intuitivo de $\vdash$ en $u \vdash X$ es que si todas las proposiciones en u son ciertas de un elemento entonces también X lo es. Esto justifica porque pedimos los axiomas (iv), (v) y (vi).

Ejemplo i) Sea $D = \mathbb{N} \cup \{0\}$, los enteros no negativos donde si $n \in D$, n significa la proposición $n \leq x$, donde x es un elemento por determinar e identificamos Δ con 0.

$$\text{Con} = \{S \mid S \subseteq D \text{ y } \#(S) < \aleph_0\}$$

y definimos $\vdash$ como la siguiente relación:

$$\{n_0, \ldots, n_{k-1}\} \vdash m \iff m = 0 \text{ ó } m \leq n_i \text{ para algun } i < k.$$

El sistema anterior verifica los axiomas (i), (ii) (iii) y (iv) de la definición trivialmente, pues Con son todos los subconjuntos finitos de D y $S\vdash 0$ $\forall$ $S\in Con$.

Verificamos:

(v) Sea $u=\{n_0,\dots,n_{k-1}\}\in Con$ y $n_i\in u$ entonces $n_i \leq n_i$ por lo que $u\vdash n_i$ $i=0,1,\dots,k-1$.

(vi) Sea $v=\{n_0,\dots,n_{k-1}\}$ y $u=\{m_0,\dots,m_{l-1}\}$ ambos en Con y supongamos $v\vdash m_i$ $i=0,1,\dots,l-1$ y $u\vdash X$, sea $n=\max\{n_j \mid n_j\in v\}$, entonces $m_i\leq n$ y como $u\vdash X$ $X\leq n$ de donde

$$v\vdash X \qquad ().$$

Ejemplo 2) En el primer ejemplo todos los conjuntos finitos aparecen como consistentes. Modificando ligeramente el ejemplo anterior obtenemos es siguiente Sistema de Información.

Sea $D=\{(n,m)\mid n,m\in \mathbb{N}\cup\{0\}$ y $n\leq m\}$

y una pareja denota la proposución $n\leq x\leq m$ para algún posible elemento x; claramente debemos hacer a las parejas $(0,3)$ y $(4,7)$ inconsistentes. Para Δ incluimos la pareja $(0,\infty)$ con la interpretación natural.

Ahora, definimos

$$u\in Con \iff \exists\ t\in \mathbb{N}\cup\{0\} \text{ 't' } n\leq t\leq m \quad \forall\ (n,m)\in u$$

y para la relación de implicación definimos

$$u\vdash(n_0,m_0) \iff t\in \mathbb{N}\cup\{0\} \text{ y } n\leq t\leq m \ \forall\ (n,m)\in u \text{ implica } n_0\leq t\leq m_0.$$

Nuevamente es necesario verificar que se cumplen los axiomas de la definición.

(i) Es evidente.

(ii) Si $X=(n,m)$ tómese $t=n$ y entonces $\{X\}\in Con$.

(iii) Si $u\in Con$ y $u\vdash X=(n_0,m_0)$ entonces, por la consistencia de u,

$$\exists\ t_0\in \mathbb{N}\cup\{0\} \text{ 't' } n\leq t_0\leq m \quad \forall\ (n,m)\in u,$$

pero entonces como $u\vdash X$, $n_0\leq t_0\leq m_0$

luego $t_0\in \mathbb{N}\cup\{0\}$ y $n\leq t_0\leq m$ $\forall$ $(n,m)\in [u\cup\{X\}]$

$\therefore \quad u \cup \{X\} \in Con.$

(iv) y (v) son inmediatos.

(vi) Si $v \vdash Y \ \forall Y \in u$ y $u \vdash X = (n_0, m_0)$ entonces, sea $t \in N \cup \{0\}$ 't' $n \leq t \leq m \ \forall (n,m) \in v$ debemos probar que $n_0 \leq t \leq m_0$.

Como $v \vdash Y = (k,r) \ \forall Y \in u$, $k \leq t \leq r \ \forall (k,r) \in u$ y como $u \vdash X$, $n_0 \leq t \leq m_0$.

<u>Definición 7.2</u>.- Una relación de implicación $\vdash$ es minimal cuando $u \vdash X$ si y sólo si $X = \Delta$ ó $X \in u$, y $\vdash$ así definida se llama la relación minimal.

().

<u>Lema 7.3</u>.-Toda relación minimal satisface los axiomas (iv), (v) y (vi).

().

DEMOSTRACION.-(iv) y (v) son inmediatos de la definición anterior.

Ahora para (vi), sean $u, v \in Con$ y $v \vdash Y \ \forall Y \in u$, $u \vdash X$; como $u \vdash X$,

si $X = \Delta$ entonces $v \vdash X$;

si $X \in u$ como $v \vdash Y \ \forall Y \in u$, en particular $v \vdash X$. []

<u>Definición</u> <u>7.4</u>.- Llamamos maximal a la siguiente relación de implicación:

$u \vdash X \iff X = \Delta$ ó $\forall \ v \supseteq u$, $v \in Con$, se tiene $v \cup \{X\} \in Con$. ()

<u>Proposición</u> <u>7.5</u>.-La relación maximal cumple los axiomas (iv), (v) y (vi) de la definición de Sistemas de Información. ().

DEMOSTRACION.-

(iv) trivialmente $u\vdash\Delta$.

(v) Si $X\in u$, entonces $\forall\ v\supseteq u\quad v\in Con$,

$$v\cup\{X\}=v;\quad \therefore\quad v\cup\{x\}\in Con\quad \therefore\quad u\vdash X.$$

(vi) Si $v\vdash Y\ \forall\ Y\in u$ y $u\vdash X$, debemos probar $v\vdash X$.

Si $X=\Delta$, entonces $v\vdash X$;

consideremos $w\in Con\ \ w\supseteq v$,

como $v\vdash Y\ \ \forall\ Y\in u$ si $u=\{Y_1, Y_2, \ldots, Y_n\}$,

$$w\cup\{Y_1\}\in Con \quad y \quad w\cup\{Y_1\}\supseteq v\text{ , de donde}$$

$(w\cup\{Y_1\})\cup\{Y_2\}\in Con$ y por inducción se obtiene

$$w\cup u\in Con;$$

pero $w\cup u\supseteq u$ y $u\vdash X$, por la definición de la relación maximal tenemos $w\cup u\cup\{X\}\in Con$, pero,

$$w\cup\{X\}\subseteq w\cup u\cup\{X\}$$

$$\therefore\quad w\cup\{X\}\in Con$$

$$\therefore\quad v\vdash X.$$ [].

<u>Proposición</u> <u>7.6</u>.-Sea ahora $\vdash_o$ otra relación sobre los mismos conjuntos Con y D, y que satisface los axiomas para Sistemas de Información, emtonces si $u\vdash_o X$, se tiene que $u\vdash X$ bajo la relación maximal. ().

DEMOSTRACION.-Si $u\vdash_o X$, entonces, $\forall\ v\supseteq u\ \ v\in Con$ obtenemos que $v\vdash_o Y\ \ \forall\ Y\in u$, y como $u\vdash_o X$, $v\vdash_o X$, de donde

$$v\cup\{X\}\in Con$$

y por lo tanto, $u\vdash X$ bajo la relación maximal. [].

Ejemplo 3) Sean A y B conjuntos, y sea $D=A\times B\cup\{\Delta\}$. Interpretamos como la información contenida en (a,b) como la afirmación b=f(a) para alguna función f por determinar. Intuitivamente un conjunto finito de datos (proposiciones) es consistente si tales parejas pueden pertenecer a una función, esto es:

(1) $\{(a_o,b_o),\ldots,(a_k,b_k)\}\in Con \iff \forall\, i,j\leq k$ si $a_i=a_j \Rightarrow b_i=b_j$

y si $u\in Con$ entonces siempre $u\cup\{\Delta\}\in Con$. (Este problema puede evitarse si en lugar de (a,b) se considera $\{(a,b)\}$ y $\Delta=\emptyset$).

Damos como $\vdash$ la relación minimal.

El axioma (i) se satisface pues si $u\in Con$ y $v\subseteq u$, en particular también para v se satisface la relación (1), por lo que $v\in Con$.

(ii)Los singoletes cumplen (1).

(iii)Si $u\vdash X$ entonces $X=\Delta$ ó $X\in u$, en ambos casos es directo que $u\cup\{X\}\in Con$.

Por último, como se trata de la relación minimal la verificación de que se trata de un Sistema de Información es completa.

Ejemplo 4)Sea $D=\{\Delta,0,1,00,01,10,11,\ldots,1^n\}$, es decir todas las cadenas finitas de longitud menor o igual que n, formadas por 0 ó 1, además del símbolo Δ para la cadena vacía.

Como notación auxiliar indicamos L(s)=longitud de s ,$\forall$ $s\in D$ y $L(\Delta)=0$, y s(i) al i-ésimo símbolo de izquierda a derecha en la cadena.

Sea $u\subseteq D$ finito, decimos que $u\in Con$ si y sólo si $\forall$ $s,t\in u$ se tiene $s(i)=t(i)$ $i=1,2,\ldots,\min\{L(s),L(t)\}$ y como $\vdash$ utilizamos la relación minimal.

Probaremos que la descripción anterior es un Sistema de Información.

(i)Sea $u\in Con$ y $v\subseteq u$, tomemos $s,t\in v$, entonces $s,t\in u$ y por lo tanto $s(i)=t(i)$ $i=1,2,\ldots,\min\{L(s),L(t)\}$, de donde $v\in Con$.

(ii)Si $s_0\in D$,es claro que $\{s_0\}\in Con$ pues si $s,t\in\{s_0\}$ $s=t=s_0$

en consecuencia $s(i)=t(i)$ $\forall$ i.

(iii)Si $u \vdash s$ debemos probar que $u \cup \{s\} \in Con$, ahora,

si $s \in u$ tenemos $u \cup \{s\} = u \in Con$;

si $s = \Delta$ entonces para $r,t \in u \cup \{s\}$ tenemos

$r(i)=t(i)$ $i=1,..,\min\{L(r),L(t)\}$ pues si $r=\Delta$ ó $t=\Delta$ $\min\{L(r),L(t)\}=0$,

(iv), (v) y (vi) se deben al hecho de que $\vdash$ es minimal.

En este ejemplo interpretamos una cadena s de D como la información acerca de los $L(s)$ dígitos binarios de un posible número entre 0 y 2^{n-1}, es decir, una cadena nos define parcialmente un número y Con son los conjuntos con cifras consistentes.

Ejemplo 5) Sea $D=\mathbb{P}(\mathbb{N})-\{\emptyset\}=\{ S \mid S\subseteq \mathbb{N} \text{ y } S\neq\emptyset\}$

hacemos $\Delta=\mathbb{N}$

y $F=\{S_1,...,S_n\}\in Con$ si y sólo si $\bigcap_F S_i \neq \emptyset$ ó $F=\emptyset$

donde por $\vdash$ damos la relación minimal.

Claramente se satisfacen los axiomas (i) y (ii) de la definición 7.1, ahora

(iii) Supongamos $F \vdash S$

si $S\in F$ entonces $F\cup\{S\}=F\in Con$

si $S=\Delta=\mathbb{N}$ $\bigcap_F S_i = \bigcap_{F\cup\{S\}} S_i \neq \emptyset$

Por último, como $\vdash$ es minimal tenemos que (iv), (v) y (vi) son verdaderos.

Podemos interpretar que un dato $X\in D$ proporciona como información el hecho de que el elemento por definir pertenece a X, es decir, X representa la afirmación $x\in X$.

Posteriormente extenderemos los últimos dos ejemplos con relaciones $\vdash$ más fuertes.

Los ejemplos anteriores y algunos que exploraremos más adelante muestran que algunas veces la estructura radica principalmente en Con y otras veces en $\vdash$ y en algunos casos en ambas. El objeto Δ puede ser de considerable importancia pràctica y sin duda de gran utilidad teòrica, ya que garantiza que todo Sistema de Informaciòn es no vacio y Con contiene a $\{\Delta\}$ y al $\emptyset$.

Debido al axioma de transitividad podemos extender $\vdash$ a una relaciòn binaria entre conjuntos de Con y abusando de la notaciòn la denotaremos tambièn por $\vdash$, identificando $\{X\}$ con X.

<u>Definiciòn 7.7</u>.-Sean $u, v \in Con$, entonces

$$u \vdash v \iff u \vdash X \quad \forall X \in v . \qquad \Diamond.$$

A continuaciòn presentamos algunas propiedades.

<u>Proposiciòn 7.8</u>.- $\forall\ u, v, w, u', v' \in Con$

(i) $\emptyset \vdash \Delta$

(ii) Si $u \vdash v \Rightarrow u \cup v \in Con$

(iii) $u \vdash u$

(iv) $u \vdash v$ y $v \vdash w$ implican $u \vdash w$.

(v) Si $u' \supseteq u \quad u \vdash v \quad v \supseteq v' \Rightarrow u' \vdash v'$.

(vi) Si $u \vdash v$ y $u \vdash v' \Rightarrow u \vdash v \cup v'$. $\Diamond$.

DEMOSTRACION.-(iii) y (v) Son inmediatos de las definiciones de Sistema de Informaciòn y la definiciòn anterior.

(i) $\emptyset \subseteq u \ \forall\ u \in Con \quad \therefore \ \emptyset \in Con$.

(ii) Se prueba por inducciòn sobre la cardinalidad de v.

Si $\#(v)=1$ por el tercer axioma de Sistemas de Informaciòn queda comprobado.

Ahora, si $\#(v)=n$ y $u \vdash v$, sea $X \in v$, entonces $\#(v-\{X\})=n-1$ y por hipòtesis de inducciòn tenemos

$$u \bigcup (v-\{X\}) \in Con$$

además $u\bigcup(v-\{X\}) \vdash Y \quad \forall\ Y \in u$ y como $u \vdash X$

$$u\bigcup(v-\{X\}) \vdash X$$

de donde $u\bigcup(v-\{X\})\bigcup\{X\}=u\bigcup v \in Con$.

(iv) Sea X w arbitrario, debemos probar $u \vdash X$.

Como $v \vdash w$ en particular $v \vdash X$ y como $u \vdash v$,

$$u \vdash Y \quad \forall\ Y\ v$$

por el sexto axioma de la definición 7.1 $u \vdash X$.

(vi) Si $X \in v\bigcup v'$ entonces $X \in v$ ó $X \in v'$ pero por las hipótesis correspondientes a este inciso , tenemos $u \vdash X$

$$\therefore \quad u \vdash v\bigcup v'. \qquad [].$$

Es importante señalar que tomando (i), (ii), (iv), (v) y (vi) de la proposición anterior y los dos primeros axiomas de nuestra definición se construye una definición de Sistema de Información equivalente a la que presentamos, a partir de una relación binaria $\vdash$ sobre Con y posteriormente definiendo la relación original como:

$u \vdash X$ si y sólo si $u \vdash \{X\}$. (No realizaremos los detalles).

Como notación, un Sistema de Información se designa por

$$A=(D_A , \Delta_A , Con_A , \vdash_A)$$

para decir que A es un Sistema de Información cuyas componentes son las indicadas; la notación sin subíndices nos bastará cuando trabajemos con un solo sistema, pero nos será de gran utilidad cuando varios sistemas estén involucrados.

Sección 8 Elementos

Consideremos un Sistema de Información A, los datos significan afirmaciones o proposiciones sobre elementos, es decir, si $X \in D_A$ y x es un elemento, es entonces factible decir que X es cierto de x. Como lo único a nuestro alcance son los datos, suponemos que siempre podemos distinguir elementos distintos, (de lo contrario debemos cambiar a un mayor Sistema de Información).

Formalmente x=y si y sólo si todas las proposiciones ciertas de x son ciertas de y, e inversamente, todas las proposiciones ciertas de y lo son de x. Bajo este principio lo que haremos es identificar un elemento con el conjunto de proposiciones que son ciertas de él. En nuestro modelo los elementos són conjuntos de datos, pero observese que para que un conjunto de datos pueda representar un elemento:

(i) debe ser consistente; y

(ii) debe ser cerrado bajo implicación.

Los requisitos anteriores son claros, pues se trata de posibles elementos y la implicación de una afirmación por proposiciones ciertas acerca de un elemento nos dice que la afirmación es cierta. Inversamente cualquier conjunto con las propiedades (i) y (ii) constituye una definición para un elemento (tal vez parcialmente definido). Esto nos conduce a la siguiente:

<u>Definición 8.1</u>.-Los elementos de un Sistema de Información A donde $A=(D_A, \Delta_A, Con_A, \vdash_A)$ son precisamente aquellos subconjuntos x de D_A tales que:

(i) Si $u \subseteq x$ y $\#(u) < \aleph_0 \Rightarrow u \in Con$.

(ii) Si $u \in Con_A$, $u \subseteq x$ y $u \vdash X \Rightarrow X \in x$. ().

<u>Definición 8.2</u>.-El conjunto de elementos de A se llama el Dominio $|A|$. Como notación escribimos $x \in |A|$ para decir que x es un elemento del Dominio definido por el Sistema de Información A.

().

En el transcurso de la tesis no distinguiremos entre el Sistema de Información A y el Dominio $|A|$ que define, salvo aclaraciones explícitas.

Definicion 8.3 .-Un elemento x es total si no está contenido propiamente en algún otro elemento y. En caso contrario decimos que x es parcial. El conjunto de elementos totales de A se denota Tot_A.

().

Definición 8.4.- Un conjunto que satisface que todo subconjunto finito está en Con se llama consistente. ().

Proposición 8.5.-Si x es consistente, entonces

$$y= \{X \mid u \vdash X \text{ para algún } u \in Con \quad u \subseteq x\}$$

es un elemento llamado el elemento generado por x, denotado $\langle x \rangle$.

().

DEMOSTRACION.-Verificaremos la dos propiedades de la definición de elemento.

(i)Sea y' un subconjunto finito de $y=\langle x \rangle$, entonces si $y' = \{X_1,\ldots,X_n\}$, sean $u_1,\ldots,u_n$ tales que

$$u_i \vdash X_i \quad y \quad u_i \subseteq x \quad u_i \text{ finito} \quad i=1,\ldots,n$$

entonces

$$\bigcup_{i=1}^{n} u_i \quad x \qquad y \quad \bigcup_{i=1}^{n} u_i \quad \text{es finito}$$

de donde

$$\bigcup_{i=1}^{n} u_i \in Con$$

y trivialmente

$$\bigcup_{i=1}^{n} u_i \vdash X_j \quad j=1,\ldots,n$$

por lo que

aplicando inducivamente el axioma (iii) obtenemos

$$\hat{\bigcup_{i=1}} u_i \cup y' \in Con$$

$$\therefore \quad y' \in Con.$$

b) Sea $u \subseteq y$ u finito y tal que $u \vdash X$, entonces debemos comprobar que $X \in y$, es decir que

si $u = \{Z_1, \ldots, Z_m\}$

donde Z_J es tal que existe $u_J \subseteq x$ u_J finito y $u_J \vdash Z_J$, sea

$$v = \hat{\bigcup_{j=1}} u_J \subseteq x$$

entonces v es finito y por lo tanto $v \in Con$, además $v \vdash u$

$$\therefore \quad v \vdash X. \qquad [].$$

<u>Proposición</u> <u>8.6</u>.-x es un elemento si y sólo si

$$x = \langle x \rangle. \qquad [].$$

DEMOSTRACION.-Inmediata de la proposición anterior.

Algunas otras propiedades sobre elementos también son inmediatas, por ejemplo:

<u>Proposición</u> <u>8.7</u>.-

a) $\Delta_A \in x \quad \forall$ elemento x.

b) Todo Dominio contiene un elemento mínimo bajo contención de conjuntos:

$$\underline{1} = \langle \Delta_A \rangle = \{X \in D_A \mid \{\Delta\} \vdash X\}.$$

c) Todos los subconjuntos finitos de D_A son consistentes si y sólo si D_A es el máximo elemento denotado por T_A. ().

Ejemplos

Ejemplo 1) Recordando el primer ejemplo de Sistemas de Información en la sección 7, donde n significaba la proposición $n\leq x$, los elementos son los conjuntos finitos

$$K_m=\{n \mid n\leq m\} \qquad m\in \mathbb{N}\cup\{0\}$$

$$y \quad K=\mathbb{N}\cup\{0\}. \qquad [].$$

DEMOSTRACION.-Tenemos un màximo elemento $\mathbb{N}\cup\{0\}=K$ por la proposición 8.7 c), probar que Km es un elemento es directo, pues como es finito, $K_m\in Con$ y en consecuencia todos sus subconjuntos también, ahora si

$$\{n_0,\dots,n_{k-1}\}\subseteq K_m \quad y$$
$$\{n_0,\dots,n_{k-1}\}\vdash p$$

entonces $p=0$ ó $p\leq n_i$ para algun $i<k$

$$\therefore \quad p\leq m \text{ y } p\in K_m.$$

Probar que son los ùnicos elementos es sencillo pues si

$$L\in \mathbb{N}\cup\{0\}$$

es elemento,

$$\forall\ k\in L \quad \{n \mid n\leq k\}\subseteq L,$$

esto es claro ya que si $k\in L$, $\{k\}\vdash n \quad \forall\ n\leq k;$

$$y \ \{k,n\}\subseteq L \ \forall n\leq k$$

si L no es acotado , $L=\mathbb{N}\cup\{0\}$,

si no $L=K_m$ com $m=\max(L)$. [].

Ejemplo 2) El segundo ejemplo se Sistemas de Información produce como elementos los conjuntos

$$R(m,p)=\{(n,q) \mid n\leq m\leq p\leq q\}$$

donde $m\leq p$ son dados y q puede ser ∞ . Los elementos totales corresponden con los enteros no negativos (existe una biyección), mientras que los elementos parciales corresponden a los casos donde las cotas m y p no coinciden.

DEMOSTRACION.-Probaremos que los $R(m,p)$ son elementos.

Sean $m,p \in \mathbb{N} \cup \{0\}$ y $m \leq p$, entonces para todo subconjunto finito u de $R(m,p)$, tenemos $u \in Con$ pues en particular $n \leq m \leq q$ para cualquier $(n,q) \in u$.

Ahora verificamos la segunda condición para ser elemento; sea $u \subseteq R(m,p)$ y supongamos $u \vdash (n_o, q_o)$ u finito, debemos probar que

$$(n_o, q_o) \in R(m,p)$$

como $u \subseteq R(m,p)$ en particular

$$n \leq m \leq p \leq q \quad \forall \ (n,q) \in u$$

entonces como $u \vdash (n_o, q_o)$, $n_o \leq m \leq p \leq q_o$

por lo que $(n_o, q_o) \in R(m,p)$.

A continuación mostraremos que estos son los únicos elementos posibles.

Sea Q un elemento, entonces sea $p=\inf\{q \mid (n,q) \in Q\}$ y $m=\sup\{n \mid (n,q) \in Q\}$, afirmamos $Q=R(m,p)$.

Si $(n,q) \in Q$ entonces $n \leq m \leq p \leq q$ por lo que $(n,q) \in R(m,p)$

$$\therefore \ Q \subseteq R(m,p).$$

Si $(n,q) \in R(m,p)$ entonces $n \leq m \leq p \leq q$ y por la definición de m y p existen

$(n_o, q_o) \in Q$ 't' $n \leq n_o$ y
$(n_1, q_1) \in Q$ 't' $q_1 \leq q$.

Como $P=\{(n_o,q_o),(n_1,q_1)\} \subseteq Q$ y Q es elemento, ya que P es finito se debe tener que $P \in Con$, pero por la definición de Con existe un entero k 't' $n_i \leq k \leq q_i$ $i=0,1$;
entonces $n \leq k \leq q$, pero por la definición dada para $\vdash$ esto nos dice que $P \vdash (n,q)$, como Q sea elemento, es cerrado bajo implicación de donde $(n,q) \in Q$

$$\therefore \quad R(m,p)\in Q.$$

Como se cumple la doble contención tenemos comprobada la afirmación. La afirmación sobre Tot es ahora clara. [].

Ejemplo 3) Señalaremos que en el ejemplo del Sistema de Información a base de parejas ordenadas de posibles funciones los elementos parciales son gráficas de funciones parciales de A en B junto con Δ; y los elementos totales corresponden a funciones totales, es decir definidas en todo A.

Ejemplo 4) En el quinto ejemplo de la sección anterior teníamos $D=\mathbb{P}(\mathbb{N})-\{\emptyset\}$, $\mathbb{N}=\Delta$, $\vdash$ la relación minimal y

$$F=\{S_1,\dots,S_n\}\in Con \iff F=\emptyset \text{ ó } \bigcap_F S_i \neq \emptyset .$$

Afirmamos que $R_{So}=\{\, S \mid S\supseteq So\}$ $So\in D$ es un elemento, pues si $F\subseteq R_{So}$ es finito $\bigcap_F S\supseteq So$ ó $F=\emptyset$, por lo tanto $F\in Con$.

Si $F\vdash S$ entonces $S=\mathbb{N}$ ó $S\in F$, en cualquier caso $S\supseteq So$, y se concluye $S\in R_{So}$

Finalmente si T es un elemento, si tomamos $So=\bigcap_T S$ entonces

$$T\subseteq R_{So}$$

pues si $S\in T$ $S\supseteq\bigcap_T S = So$ de donde $S\in R_{So}$.

Sin embargo la conteción puede ser propia, por ejemplo sea $T=\{\{1\},\mathbb{N}\}$, entonces todos los subconjuntos finitos de T son:

$$F_1=\emptyset \quad F_2=T \quad F_3=\{\{1\}\} \text{ y } F_4=\{\mathbb{N}\}$$

que están en Con y T es cerrado bajo $\vdash$ sin embargo $T\neq R_{\{1\}}$.

Los elementos $R_{\{i\}}$ $i\in\mathbb{N}$ son elementos totales, esto resulta del hecho de que si $S\subseteq S'$ entonces $R_{S'}\subseteq R_S$; y concluimos Tot es isomorfo a los naturales.

Observaciones.-

a)Para que los elementos coincidan con los R_S debe considerarse un Sistema de Información similar pero con una relación |- más fuerte, por ejemplo:

$$F\vdash S' \text{ si y sólo si } \bigcap_F S\subseteq S'.$$

La demostración es similar al ejemplo 2.

b)En el ejemplo anterior utilizamos los naturales $\mathbb{N}$ como conjunto base y consideramos todos los subconjuntos no vacios como las proposiciones o datos; el ejemplo se puede generalizar a un conjunto base arbitrario ya que su definición se basa únicamente en conceptos conjuntistas.

Ejemplo 5)Este ejemplo es una extensión del ejemplo de cadenas binarias de la Sección 7. Si ahora $D=\{0,1\}^{\infty}\cup\{\Delta\}$, es decir todas las cadenas formadas de 0 ó 1 de longitud finita y la cadena vacla bajo el conjunto Con definido igualmente como:

$$u\in Con \iff \forall\ s,t\in u\ \ s(i)=t(i)\ \ i=1,...,\min\{L(t),L(s)\}$$

con $L(\Delta)=0$ y si $u\in Con$ y s es una cadena

$$u\vdash s \iff \forall\ t\in u\ \ s(i)=t(i)\ \ i=1,..,\min\{L(s),L(t)\}\ ,$$

$$\text{y existe } r\in u\ \text{'t'}\ \ L(r)\geq L(s).$$

Es fácil verificar que en efecto la descripción anterior conforma un Sistema de Información. Por ejemplo , el axioma (iii) se sigue así:

Supongamos $u\in Con$ y $u\vdash s_0$,debemos probar que $u\cup\{s_0\}\in Con$.

Sean $s,t\in u\cup\{s_0\}$,

si $s,t\in u$ directamente se sigue que $s(i)=t(i)$ $i=1,..,\min\{L(s),L(t)\}$;

si $s=t=s_0$ entonces $s(i)=t(i)\ \forall\ i$;

finalmente si $s\in u$ y $t=s_0$ como $u\vdash s_0$

$s_0(i)=s(i)$ $i=1,..,\min\{L(s_0),L(s)\}$ entonces

$$s(i)=t(i)\qquad i=1,...,\min\{L(s),L(t)\}.\qquad [].$$

Y con la misma facilidad se comprueban los demás axiomas.

Los elementos de este Dominio que llamaremos "Binario" pueden ser identificados con una única cadena de la manera siguiente: Si @ es una cadena formada de 0 y 1 que puede ser vacía ,finita o infinita, denotamos por @n a la cadena o segmento inicial de los primeros n símbolos de @ ($n \leq L(@)$ si $L(@) < \aleph_0$ y $n < L(@)$ si $L(@) = \aleph_0$).

Los elementos de Binario son entonces $L_@ = \{@n \mid n \leq L(@)\}$. Es claro que $L_@$ es un elemento pues si $@n, @m \in L_@$ entonces $@n(i) = @m(i) = @(i)$ $i = 1, \ldots, \min\{m, n\}$ y si $u \in L_@$ es tal que $u \vdash s$ entonces $s = @_{L(s)}$. Es claro también que todos los elementos son de esta forma ya que si se considera T un elemento y T es finito sea \$ 't' $\$ \in T$ y $L(\$) = \max\{L(s) \mid s \in T\}$, entonces $T = L_\$$.

Si T es infinito se construye @ inductivamente , $@(1) = s(1)$ para algún $s \in T$, si conocemos $@(1), \ldots, @(n)$ entonces $@m \in T$, $m = 1, \ldots, n$ ya que T es elemento y como es infinito existe $s \in T$ tal que $s \neq @m$ $m = 1, \ldots, n$ sea $@(n+1) = s(n+1)$.

Obsérvese como en este ejemplo aparece la idea de aproximación a elementos infinitos(cadenas infinitas) mediante elementos finitos (sus segmentos iniciales).

Algunas propiedades de los Dominios son presentadas a continuación.

<u>Teorema</u> <u>8.7</u>.-Si A es un Sistema de Información y $x_n \in |A|$ para $n = 0, 1, \ldots$, y $x_0 \subseteq x_1 \subseteq \ldots x_n \subseteq \ldots$ entonces $\bigcup_{n=0}^{\infty} x_n \in |A|$. ().

DEMOSTRACION.-Sea $u \subseteq \bigcup_{n=0}^{\infty} x_n$ un conjunto finito, entonces existe n_0 't' $u \subseteq \bigcup_{n=0}^{n_0} x_n = x_{n_0}$ entonces como $x_{n_0} \in |A|$ se tiene que $u \in Con$.

Ahora, si $u \subseteq \bigcup_{n=0}^{\infty} x_n$ u es finito y $u \vdash X$ entonces existe n_0 't' $u \subseteq \bigcup_{n=0}^{n_0} x_n = x_{n_0}$ y como x_{n_0} es elemento $X \in x_{n_0}$, entonces $X \in \bigcup_{n=0}^{\infty} x_n$,

$$\therefore \bigcup_{n=0}^{\infty} x_n \in |A|. \qquad [].$$

Teorema 8.8.-Si $x, y \in |A|$ entonces $x \cap y \in |A|$. ().

DEMOSTRACION.-Sea u un subconjunto finito de $x \cap y$, entonces $u \subseteq x$ es finito, $\therefore$ $u \in Con$ pues x es elemento.

Además, si $u \subseteq x \cap y$ es tal que $u \vdash X$ entonces como $u \subseteq x$, $u \vdash X$ y $x \in |A|$, $X \in x$,análogamente $u \subseteq y$, $u \vdash X$ $y \in |A|$ de donde $X \in y$

$\therefore$ $X \in x \cap y$ y la prueba está completa. [].

Teorema 8.9.-Si x_α ,$\alpha \in \mathcal{a}$ es una familia arbitaria de elementos de $|A|$ entonces

$$\bigcap_{\mathcal{a}} x_\alpha \in |A|. \qquad ().$$

DEMOSTRACION.-Si $u \subseteq \bigcap_{\mathcal{a}} x_\alpha$ es finito, $u \subseteq x_\alpha$ $\forall \alpha$ y como x_α es elemento $u \in Con$, además si $u \vdash X$ y $u \subseteq x_\alpha$, entonces $X \in x_\alpha$ $\forall \alpha$

$$\therefore \quad X \in \bigcap_{\mathcal{a}} x_\alpha . \qquad [].$$

Sección 9 Intuición y presentaciones anteriores.

A continuación presentamos un marco de las ideas que han contribuido a la presentación que hemos hecho de los Dominios. Mostraremos que los enfoques por enrejados(lattices) y espacios topológicos permanecen en nuestro modelo. El lector puede saltar esta sección y continuar con la descripción de la Categoría de Dominios en la siguiente.

Cuando tenemos un Sistema de Información A, los elementos del Dominio $|A|$ son conjuntos y podemos definir una relación entre ellos mediante la inclusión de conjuntos. La relación $x \subseteq y$ tiene un importante significado intuitivo, nos dice que x està menos determinado o màs parcialmente definido que y, ya que significa que toda proposiciòn cierta de x lo es de un elemento y (aunque probablemente no inversamente). Esta relaciòn tiene impotancia en el sentido de las aplicaciones pràcticas ya que aproximamos elementos infinitos por informaciòn finita y entonces pensamos en $x \subseteq y$ como x aproxima a y.

La relaciòn posee todas las propiedades de su definiciòn conjuntista, en particular:

(i) $x \subseteq x$ (reflexiva).

(ii) Si $x \subseteq y$ $y \subseteq z \Rightarrow x \subseteq z$ (transitiva).

(iii) Si $x \subseteq y$ $y \supseteq x \Rightarrow x = y$ (simètrica).

<u>Proposiciòn</u> <u>9.1</u>.-Todo Dominio |A| es un orden parcial bajo $\subseteq$ ⟨⟩.

DEMOSTRACION.-Observese que esto es directo de la definiciòn de orden parcial. Ademàs no lo estamos asumiendo como axioma de los Dominios, ¡ lo hemos probado!. (Las primeras construcciones que surgieron asumìan que los Dominios eran conjuntos parcialmente ordenados). [].

<u>Proposiciòn</u> <u>9.2</u>.-Todo Dominio |A| es una inf-semired. ⟨⟩.

DEMOSTRACION.-Como ya vimos $x \cap y$ es elemento para toda pareja x, y de elementos, lo cual prueba |A| es una inf-semired. [].

<u>Teorema</u> <u>9.3</u>.-Todo Dominio |A| es una inf-semired completa. ⟨⟩.

DEMOSTRACION.-La prueba es directa del Teorema 8.9. [].

Sin embargo la uniòn de elementos no funciona para obtener los supremos como puede notarse de que $x \cup y$ puede no ser consistente, y en caso de serlo puede no ser cerrado bajo implicaciòn. El resultado siguiente nos da la caracterizaciòn buscada.

<u>Definiciòn</u> <u>9.4</u>.- Sean x, y elementos, si z es un elemento 't' $x \subseteq z$ $y \subseteq z$, y $\forall\, w \in |A|$ 't' $x \subseteq w$ $y \subseteq w$ se tiene $z \subseteq w$, entonces z es el supremo, y lo denotamos por $x \sqcup y$. ⟨⟩.

<u>Lema</u> <u>9.5</u> .-Sean x, y elementos, $x \sqcup y$ existe si y sòlo si $x \cup y$ es consistente, en cuyo caso $x \sqcup y = \langle x \cup y \rangle$ el generado por $x \cup y$.

⟨⟩.

DEMOSTRACION.-Si $x \cup y$ es consistente, entonces $\langle x \cup y \rangle$ es un elemento que cumple con $x \subseteq \langle x \cup y \rangle$, $y \subseteq \langle x \cup y \rangle$. Si consideramos la familia de elementos z 't' $x \subseteq z$ $y \subseteq z$, èsta es no vacìa, sea z un elemento de dicha familia , entonces $x \cup y \subseteq z$, ahora si $u \subseteq x \cup y$ es tal que $u \vdash X$ entonces $u \subseteq z$ y $u \vdash X$, como z es elemento $X \in z$,

$$\therefore \quad \langle x \cup y \rangle \subseteq z$$

$$\therefore \quad \langle x \cup y \rangle = x \sqcup y.$$

Inversamente, si $x \sqcup y$ existe ,entonces $x \cup y \subseteq x \sqcup y$, de donde directamente $x \cup y$ es consistente. [].

<u>Proposiciòn</u> <u>9.6</u>.-Si z es un elemento y $u \subseteq z$, entonces

$\langle u \rangle \subseteq z$. ⟨⟩.

DEMOSTRACION.-Tòmese $x = u$ $y = \emptyset$ y sìgase la demostraciòn del lema anterior. [].

Teorema 9.7.- Sea x_α una familia arbitraria de elementos, $\bigsqcup x_\alpha$ existe si y sòlo si $\bigcup x_\alpha$ es consistente. ⟨⟩.

DEMOSTRACION.-Anàloga al caso para dos elementos. [].

Teorema 9.8.- Si $\top_A$ existe, entonces $|A|$ es una red completa. ⟨⟩.

DEMOSTRACION.-Si $\top_A$ existe, y $x_\alpha \in |A|$ es una familia arbitraria de elementos, entonces $\bigcup x_\alpha \subseteq \top_A$, por lo que

$\bigcup x_\alpha$ es consistente y $\langle \bigcup x_\alpha \rangle$ es el supremo. [].

Las formalizaciònes de Dominios se han trabajado bajo modelos de enrejados y a pesar de su èxito, su exposiciòn era complicada y poco intuitiva. No profundizaremos màs en las propiedades de los Dominios en este sentido y ùnicamente señalaremos que no toda red completa representa un Dominio. (No incluiremos los detalles porque rebazan los objetivos del trabajo). Aquellas redes que representan Dominios se llaman "Completely Consistent Algebraic cpo's".

Definiciòn 9.9.-Un elemento es finito si es de la forma

$\langle u \rangle$ para $u \in Con$ ⟨⟩.

Esto nos lleva al siguiente resultado:

Proposiciòn 9.10.-Sea $x \in |A|$ entonces

$x = \bigsqcup \{ \langle u \rangle \mid u \subseteq x \ \ u \text{ finito} \}$. ⟨⟩.

DEMOSTRACION.-Sea $X \in x$, entonces $\{X\} \in Con$ y $\langle\{X\}\rangle \subseteq x$ por lo que $X \in \bigcup\{\langle u\rangle \mid u \subseteq x \ \ u \text{ finito}\}$.

$$\therefore \quad x \subseteq \bigcup\{\langle u\rangle \mid u \subseteq x \ \ u \text{ finito}\}.$$

Ahora si $X \in \bigcup\{\langle u\rangle \mid u \subseteq x \ \ u \text{ finito}\}$ entonces $X \in \langle u\rangle$ para algún $u \in Con$ y $u \subseteq x$, es decir $\exists\, u \subseteq x$ 't' $u \vdash X$, pero como x es elemento del Dominio $X \in x$.

$$\therefore \quad x = \bigcup\{\langle u\rangle \mid u \subseteq x \ u \text{ finito}\}. \qquad [].$$

La proposición anterior representa intuitivamente el hecho de que todo elemento en el Dominio es el "límite" de sus aproximaciones finitas. Esta idea de límite fuè la que se formalizò topològicamente y se explorò dando una estructura topològica a los Dominios.En nuestro modelo podemos preservar dicha estructura mediante la siguiente:

<u>Definición 9.11</u>.- Sea $u \in Con$, la vecindad de u en $|A|$ se define por:

$$[u]_A = \{x \in |A| \mid u \subseteq x\}.$$

Las vecindades de un elemento x son todas las $[u]_A$ tales que $u \subseteq x$ y denotamos por:

$$Nx = \{[u]_A \mid u \subseteq x\}$$

al conjunto de todas las vecindades del elemento x. $\langle\rangle$.

<u>Proposición 9.12</u>.- Sea $u \in Con$, $u \subseteq y$ si y sòlo si

$$[u]_A \in Ny. \qquad \langle\rangle.$$

DEMOSTRACION.-Si $u \subseteq y$ entonces $[u]_A \subseteq Ny$ por definición.
Si $[u]_A \subseteq Ny$ $u \subseteq y$ tambièn por definiciòn. [].

<u>Proposición 9.13</u>.- Las vecindades estàn en correspondencia biuni-

voca con los elementos finitos de $|A|$. $\langle\rangle$.

DEMOSTRACION.-Recuerdese que y es finito si y sólo si $u \in Con$ y $y=\langle u\rangle$. Definiremos una biyección de las vecindades en los elementos finitos.

Definimos $\varphi([u]_A)=\langle u\rangle$. Es fácil verificar que φ está bien definida.

Claramente $\varphi([u]_A)$ es un elemento finito, φ es sobre pues si $\langle u\rangle$ es un elemento finito, $u\in Con$ y $[u]_A$ es una vecindad tal que $\varphi([u]_A)=\langle u\rangle$.

Ahora, si $\langle u\rangle=\langle v\rangle$,

entonces sea $x\in[u]_A$ entonces $u\subseteq x$ en consecuencia $\langle u\rangle\subseteq x$, es decir $\langle v\rangle\subseteq x$ pero entonces $x\in[v]_A$ $\therefore$ $[u]_A\subseteq[v]_A$

Simétricamente $[v]_A\subseteq[u]_A$.

$$\therefore\ [u]_A=[v]_A$$

$\therefore$ φ es una biyección. [].

<u>Proposición 9.14</u>.-Si $u,v\in Con$ entonces

$$[u]_A\cap[v]_A=[u\cup v]_A\ . \qquad \langle\rangle.$$

DEMOSTRACION.-Si $u\cup v$ es inconsistente, ambos lados son el ϕ.

Ahora, $x\in[u]_A\cap[v]_A$ si y sólo $u\subseteq x$ y $v\subseteq x$, y esto ocurre si y sólo si $u\cup v\subseteq x$, es decir, $x\in[u\cup v]_A$. [].

<u>Proposición 9.15</u>.-Sean x,y elementos tales que $Nx=Ny$, entonces $x=y$. $\langle\rangle$.

DEMOSTRACION.-Por la proposición 9.10 $x=\bigcup\{\langle u\rangle \mid u\subseteq x, u \text{ finito}\}$

Sea $u\in Con$ y $u\subseteq x$, entonces por la proposición 9.12

$[u]_A\in Nx=Ny$, es decir $\langle u\rangle\subseteq y$, entonces $u\subseteq y$ pero esto es para toda $u \in x$, por lo que $x \leq y$.

Simètricamente $y \geq x$.

$$\therefore \quad x=y. \qquad [].$$

De la proposiciòn anterior se sigue que $|A|$ es un espacio topològico T_0, pues dos puntos con las mismas vecindades son iguales. De estos resultados el enfoque topològico condujo el estudio hacia el anàlisis de las funciones continuas entre Dominios, nuestro objetivo aquì fue mostrar que nuestro modelo preserva las estructuras trabajadas anteriormente, pero aquì, las propiedades antes asumidas son probadas a partir de muy pocos y naturales axiomas.

Por ejemplo:

<u>Proposiciòn 9.16</u>.-Todo elemento puede extenderse a un elemento total t. $\langle\rangle$.

DEMOSTRACION.-Sea $x \in |A|$, entonces la familia de elementos que contienen a x es no vacia pues x està en ella. Ademàs es una familia inductiva por el Teorema 8.7 . Aplicando el Lema de Zorn , sabemos que existe al menos un màximo elemento t de la familia

$$\therefore \quad x \leq t.$$

Si ahora z es tal que $t \leq z$, entonces $x \leq z$, de donde z està en la familia , lo cual implica $z \leq t$, pues t es màximo.

$$\therefore \quad z=t \; ; \qquad \therefore \quad t \text{ es total.} \qquad [].$$

Como $Tot_A \subseteq |A|$, Tot_A es un subespacio topològico pero ademàs

<u>Proposiciòn 9.17</u>.- Tot_A es un subespacio topològico Haudorf.

DEMOSTRACION.-Supongamos $s,t \in Tot_A$ y $s \neq t$.

Si $\forall$ $u \in Con$ 't' $u \subseteq s$ se tiene $u \subseteq t$, entonces se tendía $s \subseteq t$ (ya que si $X \in s$ entonces $\{X\} \in Con$ y se tendía $\{X\} \subseteq t$);
entonces como s es màximo $s=t$, lo cual contradice la suposición.

Concluimos, existe $u \in Con$ 't' $u \subseteq s$ y $u \not\subseteq t$, y como $t \notin s$ existe $X \in t$ 't' $X \notin u$.

Sea $r=\{X \in t \mid X \notin u\}$, entonces $r \subseteq t$ y no es vacío, sea $v \subseteq r$ 't' $v \in Con$ y $u \cup v \notin Con$,
(Si $\forall$ $v \in Con$ $v \subseteq r$ $u \cup v \in Con$ afirmamos que $(u \cup t)$ es elemento

Para comprobar la afirmación basta mostrar que $u \cup t$ es consistente:

Si $k \subseteq u \cup t$ k finito , entonces

$$k=\{U_1,\dots,U_n\} \cup \{R_1,\dots,R_m\}$$

con $U_i \in u$ $i=1,\dots,n$ y $R_j \in r$ $j=1,\dots,m$;
tomando $v=\{R_1,\dots,R_m\}$ se tendía $k \subseteq u \cup v$ y entonces k sería consistente ;

como $(u \cup t) \supseteq t$, $(u \cup t)=t$ pues t es màximo pero entonces $t \supset u \cup t$ es decir $u \subseteq t$ lo cual contradice las propiedades de u)

Ahora $v \in Con$, $v \subseteq t$ y $[u]_A \cap [v]_A = [u \cup v]_A = \emptyset$

$\therefore$ Tot_A es Haudorf. []

Sección 10 Morfismos de Dominios

Una de las principales motivaciones e ideas importantes se desarrollan bajo el concepto de los mapeos de un Dominio a otro. Su importancia radica en hacer notar que el espacio de morfismos de

Dominios será a su vez un Dominio.

Consideremos dos Dominios, un mapeo f tomará información sobre un posible elemento como sus datos de entrada y producirá como resultado un conjunto de afirmaciones que generan un elemento o posible valor de la función. Para un conjunto consistente u, el resultado debe ser un conjunto de afirmaciones consistente v del segundo Dominio, a la relación datos->resultado producida por la f la escribimos ufv y para obtener el efecto completo de f debemos considerar todas las posibles v ya que información finita como datos de entrada puede producir una infinidad de resultados. (un ejemplo práctico de esto se presenta en un programa que entra en "loop").

Formalmente escribimos:

<u>Definición</u> <u>10.1</u>.-Sean A y B Sistemas de Información. Un mapeo aproximable $f:A\to B$ es una relación binaria entre Con_A y Con_B tal que:

(i)$\emptyset f \emptyset$.

(ii)ufv y ufv' implican $v \cup v' \in Con_B$, $uf(v \cup v')$ $\forall\ u \in Con_A$ y $\forall\ v, v' \in Con_B$.

(iii)Si $u' \vdash_A u$, ufv, $v \vdash_B v'$ entonces $u'fv'$ $\forall\ u, u' \in Con_A$ $\forall\ v, v' \in Con_B$. ◇.

Intuitivamente la relación ufv nos dice que si proporcionamos u información sobre un posible elemento obtenemos v afirmaciones sobre el valor de la función en el elemento, y entonces las condiciones de la definición dicen que: si no se proporciona información no obtenemos información, si dos conjuntos consistentes son posibles su unión es consistente y por lo tanto un conjunto de afirmaciones del posible valor de la función, finalmente tenemos que la funcioón preserva la relación $\vdash$, es decir si proporcionamos más información que en una entrada anterior onbtendremos cualquier conjunto de resultados implicados por los resultados de la aplicación anterior.

Una propiedad inmediata es la siguiente:

Proposición 10.2.-Si f es un mapeo aproximable de A en B entonces

$$ufv \text{ si y sólo si } uf\{Y\} \ \forall\ Y\in v. \quad ().$$

DEMOSTRACION.-(=>)Si ufv y $Y\in v$ entonces $v\vdash_B Y$ y por la parte (iii) de la definición de mapeo aproximable $uf\{Y\}$.

(<=)Como v es finito aplicamos inducción sobre su cardinalidad:

Si $\#(v)=1$ $v=\{Y\}$ y $uf\{Y\}$ entonces ufv.

Supongamos que, si $uf\{Y\}\ \forall\ Y\in v$ y $\#(v)=n$ entonces ufv; sea v 't' $\#(v)=n+1$ y $uf\{Y\}\ \forall\ Y\in v$, considerese $v'=v-\{Y_0\}$ $Y_0\in v$ entonces por hipótesis de inducción se tiene ufv', es decir, tenemos ufv' y $uf\{Y_0\}$, como f es aproximable, por (ii) en la definición de mapeo aproximable $uf(v'\cup\{Y_0\})$

$$\therefore\ ufv. \qquad [].$$

Definición 10.3.-Si f:A->B es un mapeo aproximable entre Sistemas de Información y $x\in|A|$ definimos el valor de f en x,(la imagen de la función) como:

$$f(x)=\{Y\in D_B \mid uf\{Y\} \text{ para algún } u\subseteq x \text{ u finito}\}.$$

().

Proposición 10.4.-Bajo la definición anterior

$$f(x)=\bigcup\{v \mid v\in Con_B \text{ y } ufv \text{ para algún } u\subseteq x\}. \quad ().$$

DEMOSTRACION.-$f(x)\subseteq\bigcup\{v \mid v\in Con_B$ y ufv para algún $u\subseteq x\}$ es inmediato pues si $Y\in f(x)$ entonces $v=\{Y\}\in Con_B$ ya que B es Sistema de Información.

$f(x) \supseteq \bigcup\{v \mid v \in Con_B \ \ ufv$ para algún $u \subseteq x\}$ se sigue de la proposición 10.2.

[].

Las propiedades siguientes justifican el nombre de función entre Dominios y la notación de función además de proporcionar algunas cualidades que utilizaremos posteriormente.

<u>Proposición</u> <u>10.5</u>.-Sean $f,g:A\rightarrow B$ dos mapeos aproximables entre dos Sistemas de Información, entonces:

(i) f siempre mapea elementos en elementos.

(ii) $f=g \iff f(x)=g(x) \ \forall \ x \in |A|$.

(iii) $f \subseteq g \iff f(x) \subseteq g(x) \ \forall \ x \in |A|$.

(iv) $x \subseteq y$ en $|A|$ implica $f(x) \subseteq g(x)$ en $|B|$.

(v) $\forall \ u \in Con_A$ y $\forall \ v \in Con_B$, $ufv \iff \langle v\rangle \subseteq f(\langle u\rangle)$. ().

DEMOSTRACION.-

(v) ($\Rightarrow$) Si $u \in Con_A$ y $v \in Con_B$ y ufv, sea $X \in \langle v\rangle$, entonces $v \vdash_B X$, de donde $uf\{X\}$ pues f es aproximable como $u \subseteq \langle u\rangle$, $X \in f(\langle u\rangle)$.

($\Leftarrow$) Sean $u \in Con_A$ y $v \in Con_B$ 't' $\langle v\rangle \subseteq f(\langle u\rangle)$, por la proposición anterior $uf\{Y\} \ \forall \ Y \in v$ implica ufv, sea $Y \in v$, debemos probar $uf\{Y\}$; pero $Y \in v \subseteq \langle v\rangle$ entonces $Y \in f(\langle u\rangle)$ es decir $rf\{Y\}$ para algun $r \subseteq \langle u\rangle$, ahora como $r \subseteq \langle u\rangle$ $u \vdash r$ y entonces $uf\{Y\}$.

(i) Sea x un elemento, $y=f(x)$, comprobaremos y es un elemento directamente de la definición.

Sea $v \subseteq y$ v finito, $v=\{Y_1,\ldots,Y_n\}$ entonces existen $u_1,\ldots,u_n$ finitos 't' $u_i \subseteq x$ y $u_i f\{Y_i\}$ $i=1,2,..,n$

pero entonces $\hat{\bigcup_{i=1}} u_i \subseteq x$, $\hat{\bigcup_{i=1}} u_i$ es finito, por lo que

$\hat{\bigcup_{i=1}} u_i \, f\{Y_i\}$ $i=1,2,..,n$ por lo tanto

$\hat{\bigcup_{i=1}} u_i \; f \hat{\bigcup_{i=1}} \{Y_i\}$, es decir, $\hat{\bigcup_{i=1}} u_i \; fv$, y como f es aproximable $v \in Con_B$.

Ahora si $v \vdash_B X$, $\hat{\bigcup_{i=1}} u_i \, f\{X\}$, es decir $X \in y$, por lo tanto, y es elemento de $|B|$.

(iii)(⇒)Supongamos $f \leq g$ y sea $x \in |A|$, debemos probar que $f(x) \subseteq g(x)$, sea $Y \in f(x)$, entonces $uf\{Y\}$ para algún $u \in Con_A$ $u \subseteq x$, pero como $f \leq g$ $uf\{X\}$ implica $ug\{X\}$ entonces $Y \in g(x)$.

(⇐)Supongamos $f(x) \subseteq g(x)$ $\forall x \in |A|$, por demostrar que $\forall u \in Con_A$ y $\forall v \in Con_B$ si ufv entonces ugv.

Supongamos ufv, y sea $Y \in v$ arbitraria, basta probar que $ug\{Y\}$ se cumple; como ufv y $Y \in v$ tenemos $uf\{Y\}$, por hipótesis

$$f(\langle u \rangle) \subseteq g(\langle u \rangle) \quad \text{y}$$

$f(\langle u \rangle)=\{X \in D_B \mid rf\{X\}$ para algún $r \subseteq \langle u \rangle\}$ por lo que

$$Y \in f(\langle u \rangle) \subseteq g(\langle u \rangle) \quad \text{y}$$

$g(\langle u \rangle)=\{X \in D_B \mid rg\{X\}$ para algún $r \subseteq \langle u \rangle\}$, entonces existe r 't' $rg\{X\}$ y $r \subseteq \langle u \rangle$, concluimos $u \vdash r$, y

$$ug\{Y\}.$$

(ii)Observese que esto es inmediato de (iii).

(iv)Sean $x, y \in |A|$ $x \subseteq y$, sea $X \in f(x)$, entonces por definición de imagen, $uf\{X\}$ para algún $u \in Con_A$ $u \subseteq x$, como $u \subseteq x \subseteq y$, $X \in f(y)$,

$$\therefore \quad f(x) \subseteq f(y).$$

[].

Obsérvese que todas las funciones aproximables son monótonas, y esto es una consecuencia de la definición y no un supuesto de la teoría.

Es importante distinguir entre la relación ufv y la función f que manda x en $f(x)$, sin embargo no haremos mayor diferenciación pues trabajaremos sólo con funciones aproximables, es decir aquellas para las cuales existe una relación ufv que las define.

Una de las proposiciones más importantes, que nos permite dar mapeos aproximables entre Dominios especificando únicamente una función entre elementos y que caracteriza a las funciones entre Dominios que provienen de mapeos aproximables es:

<u>Proposición 10.6</u>.-Si $f:A\to B$ es una función aproximable entre Dominios entonces la función $f:|A|\to|B|$ satisface:

$$(2)\quad f(x)=\bigcup\{ f(\langle u\rangle) \text{ donde } u\subseteq x \text{ u finito}\} \quad \forall\, x\in|A|;$$

e inversamente, si f es una función $f:|A|\to|B|$ dada en los elementos que satisface (2) entonces proviene de un mapeo aproximable $f':A\to B$. (Decimos también, en este caso, que la función es aproximable).

◊.

DEMOSTRACION.-($\Rightarrow$)Sea $f:A\to B$ un mapeo aproximable, y sea $x\in|A|$ arbitrario.

Si $X\in f(x)$ entonces $\exists\, u\subseteq x$ u finito 't' $uf\{X\}$ (por la definición de imagen de f), entonces $u\subseteq\langle u\rangle$ y entonces $X\in f(\langle u\rangle)$ por lo que $f(x)\subseteq\bigcup\{f(\langle u\rangle) \mid u\subseteq x \text{ u finito}\}$.

Ahora sea $X\in\bigcup\{f(\langle u\rangle) \mid u\subseteq x \text{ u finito}\}$, entonces existe u 't' $u\subseteq x$ y $X\in f(\langle u\rangle)$, es decir existe $v\subseteq\langle u\rangle$ v finito tal que $v\in Con_A$ y $vf\{X\}$ pero $v\subseteq\langle u\rangle\subseteq x$ entonces $X\in f(x)$

$$\therefore \bigcup\{f(\langle u\rangle) \mid u\subseteq x \text{ u finito}\}\subseteq f(x)$$

$$\therefore \quad f(x)=\bigcup\{f(\langle u\rangle) \quad u\subseteq x \text{ u finito}\}.$$

($\Leftarrow$)Definimos la relación f' como el siguiente mapeo aproximable:

$$uf'v \text{ si y sólo si } \langle v\rangle\subseteq f(\langle u\rangle).$$

Probaremos que f proviene de f' y f' es aproximable.

Claramente $\emptyset f'\emptyset$ pues $\emptyset$ está contenido por todo elemento.

Supongamos $uf'v$ y $uf'v'$ entonces

$\langle v\rangle \subseteq f(\langle u\rangle)$ y $\langle v'\rangle \subseteq f(\langle u\rangle)$, entonces $\langle v\rangle \cup \langle v'\rangle \subseteq f(\langle u\rangle)$ y como $v\cup v' \subseteq \langle v\rangle \cup \langle v'\rangle$ tenemos $v\cup v' \subseteq f(\langle u\rangle)$ pero $f(\langle u\rangle)$ es elemento, de donde $\langle v\cup v'\rangle \subseteq f(\langle u\rangle)$, $\therefore$ $uf'(v\cup v')$.

Resta comprobar el tercer inciso de la definición de mapeo aproximable.

Supongamos $u \vdash_A u'$, $u'f'v'$ y $v' \vdash_B v$ entonces $\langle v'\rangle \subseteq f(\langle u'\rangle)$ y $\langle u\rangle \supseteq \langle u'\rangle$ y $\langle v'\rangle \supseteq \langle v\rangle$ de donde $\langle v\rangle \subseteq f(\langle u'\rangle)$, pero f satisface (2), $\therefore$ $f(\langle u'\rangle) \subseteq f(\langle u\rangle)$.

Verificaremos ahora que f proviene de f'.

$f'(x)=\{Y\in D_B \mid uf'\{Y\}$ para algún $u\subseteq x\}$, claramente si $Y\in f'(x)$ existe u 't' $uf'\{Y\}$, entonces por como se definió f'

$$\langle\{Y\}\rangle \subseteq f(\langle u\rangle) \subseteq f(x) \quad \therefore Y\in f(x) \qquad \therefore f'(x) \subseteq f(x).$$

Si $Y\in f(x)=\bigcup\{f(\langle u\rangle) \mid u\subseteq x \text{ u finito}\}$ existe u 't' $Y\in f(\langle u\rangle)$ de donde $\langle Y\rangle \subseteq f(\langle u\rangle)$ pues $f(\langle u\rangle)$ es elemento $\therefore$ $uf'\{Y\}$

$$\therefore \quad Y\in f'(x) \quad \therefore \quad f'(x) \supseteq f(x)$$

$$\therefore \quad f'(x)=f(x)$$

$\therefore$ f proviene de un mapeo aproximable. [].

Es importante indicar que la condición anterior, es decir (2), corresponde a la petición de que nuestras funciones sean monótonas y continuas. Del resultado anterior y el Teorema 8.7 es fácil deducir que si f es aproximable y $x_1 \subseteq x_2 \subseteq ,\ldots, \subseteq x_n ,\ldots$ son elementos, entonces

$$f(\bigsqcup_{i=1}^{\infty} x_i)=f(\bigcup_{i=1}^{\infty} x_i)=\bigcup_{i=1}^{\infty} f(x_i)=\bigsqcup_{i=1}^{\infty} f(x_i).$$

Un resultado interesante que enriquece la teoría consiste en que la composición de mapeos aproximables es aproximable, esto nos conduce al siguiente :

<u>Teorema 10.7</u>.-Los Dominios y los mapeos aproximables forman una

Categoria. ().

Para su demostración utilizaremos los dos siguientes lemas.

Lema 10.8.-Sea A un Sistema de Información, la siguiente fórmula define un mapeo aproximable $I_A: A \to A$

(i) $\forall\ u, v \in Con_A \quad u I_A v \Leftrightarrow u \vdash_A v$

y (ii) este mapeo es tal que $I_A(x) = x \quad \forall\ x \in |A|$.

DEMOSTRACION.-(i)Verificamos directamente de la definición que se trata de un mapeo aproximable:

a)Claramente, por la proposición 7.8 $\emptyset I_A \emptyset$.

b)Si $u I_A v$ y $u I_A v'$ tenemos $u \vdash v$ y $u \vdash v'$ entonces nuevamente por la proposición 7.8 $u \vdash v \cup v'$ entonces $u I_A v \cup v'$.

c)Supongamos $u' \vdash u \quad u I_A v$ y $v \vdash v'$, por la transitividad ya probada $u' \vdash v'$, entonces $u' I_A v'$.

(ii)Sea $x \in |A|$ entonces $I_A(x) = \{Y \in D_A \mid u I_A \{Y\}$ para $u \subseteq x\}$
$= \{Y \in D_A \mid u \vdash Y$ para algun $u \subseteq x\} = \langle x \rangle = x$ [].

La demostración anterior también puede realizarse en términos de la proposiciones 10.6 y 9.10.

El lema anterior nos dice que la relación $\vdash_A$ define un mapeo aproximable en A y que la identidad en |A| es aproximable.

Lema 10.9.-Sean $f: A \to B$ y $g: B \to C$ dos mapeos aproximables, (A, B y C Sistemas de Información), entonces la siguiente fórmula define un mapeo aproximable de A en C denotado por $(g \circ f)$

(i)$\forall\ u \in Con_A \quad \forall\ w \in Con_C \quad u (g \circ f) w \Leftrightarrow u f v$ y $v g w$ para algún $v \in Con_B$,

y es tal que

(ii)$(g \circ f)(x) = g(f(x)) \quad \forall\ x \in |A|$. ().

DEMOSTRACION.-Verificamos la definición.

a)$\emptyset(g \circ f)\emptyset$ pues $\emptyset \in Con_B$ y $\emptyset f \emptyset$ $\emptyset g \emptyset$

b)Supongamos $u(g \circ f)w$ y $u(g \circ f)w'$ entonces existen v y v' tales que ufv vgw ufv' v'g w', como f y g son aproximables $uf(v \cup v')$ y $(v \cup v')g(w\ w')$ se cumplen, entonces

$$u(g \circ f)w \cup w'.$$

c)Si $u' \vdash_A u$ $u(g \circ f)w$ y $w \vdash_C w'$, entonces existe $v \in Con_B$ 't' ufv y vgw , como f y g son aproximables tenemos u'fv y vgw' , pero esto implica $u'(g \circ f)w'$.

Esto comprueba las clàusulas de mapeo aproximable, ahora probaremos que $(g \circ f)(x)=g(f(x))$ $\forall\ x \in |A|$.

(ii)Sea $X \in (g \circ f)(x)$, entonces existe $u \subseteq x$ 't' $u(g \circ f)\{X\}$ entonces $\exists$ $v \in Con_B$ 't' ufv y $vg\{X\}$, entonces $v \subseteq f(x)$, pero esto implica que existe $v \subseteq f(x)$ 't' $vg\{X\}$ es decir $X \in g(f(x))$;

inversamente,

si $X \in g(f(x))$ existe $v \subseteq f(x)$ 't' $vg\{X\}$ por definición de valor de la función, pero como v es finito $v=\{Y_0,..,Y_n\}$, entonces existen $u_1,..,u_n$ $u_i \subseteq x$ i=1,2,..,n 't' $u_i f\{Y_i\}$ i=1,..,n entonces $\hat{\bigcup\limits_{i=1}} u_i$ $f \hat{\bigcup\limits_{i=1}}\{Y_i\}$, es decir $\hat{\bigcup\limits_{i=1}} u_i$ fv pero esto implica

$$X \in (g \circ f)(x).$$

$$\therefore\ (g \circ f)(x)=g(f(x)). \qquad [].$$

DEMOSTRACION.-(Teorema 10.7) Por el lema 10.8 todo Dominio posee identidad, y por el lema 10.9 la composición es asociativa y las identidades se cancelan .

Por ejemplo,

$(I_A \circ f)(x)=I_A(f(x))=f(x)$ y $(f \circ I_A)(x)=f(I_A(x))=f(x)$.

[].

Para decir que dos Dominios son isomorfos utilizamos el concepto categórico de isomorfismo.

Definición 10.10.-|A| es isomorfo a |B| si y sólo si existen morfismos(mapeos aproximables) $f:A\to B$ y $g:B\to A$ tales que

$(g \circ f)=I_A$ y $(f \circ g)=I_B$. En este caso f y g son llamados isomorfismos. <>.

Es importante señalar las siguientes propiedades sobre isomorfismos.

Proposicion 10.11.-Si $f:|A|\to|B|$ es una función en los elementos de los Dominios, entonces son equivalentes:

a) f es un isomorfismo (como morfismo de la categoría de conjuntos) que preserva el orden.

b) f proviene de un mapeo aproximable y es biyectiva en los elementos de |A| y |B|.

c) f es un isomorfismo en la categoría de Dominios, es decir, de acuerdo a la definición 10.10. <>.

DEMOSTRACION.-

a) => b) Supongamos f es un isomorfismo de conjuntos, entonces es biyectiva en los elementos, además si definimos la relación f' como $uf'v \iff \langle v\rangle \in f(\langle u\rangle)$ es directo comprobar que f' es aproximable y f proviene de f'.

c) => a) Es inmediato pues, si f es isomorfismo en la Categoría de Dominios, proviene de un mapeo aproximable , por lo tanto es monótona y preserva el orden, y por la regla de composición (que coincide para elementos con la composición de morfismos entre conjuntos) f es un isomorfismo en conjuntos para elementos.

b) $\Rightarrow$ c) Sea $g(y)=f^{-1}(y)$ $\forall$ $y\in|B|$, g esta bien definida como función pues f es biyectiva en los elementos, además $f(g(y))=y$ $\forall$ $y\in|B|$ y $g(f(x))=x$ $\forall$ $x\in|A|$, lo cual prueba que g es un isomorfismo en los conjuntos de elementos (es decir, un morfismo en la Categoria de Conjuntos) y si $y\leq y'$, entonces $y=f(x)$ $y'=f(x')$ para $x,x'\in|A|$ pues f es sobre en los elementos, entonces $x\leq x'$ (pues si $x\supset x'$ como f es aproximable y por lo tanto monótona $f(x')\supset f(x)$ entonces $y'\supset y$ $\rightarrow\leftarrow$) entonces $g(y)\subseteq g(y')$, como ya probamos a) $\Rightarrow$ b) y g satisface a) tenemos que g proviene de un mapeo aproximable ; entonces f y g son aproximables y

$$(g \circ f)(x)=g(f(x))=x \quad y \quad (f \circ g)(x)=f(g(x)) \quad \forall\ x\in|A|\ \forall\ y\in|B|$$

$\therefore$ f es isomorfismo de Dominios. [].

En lo sucesivo, sólo hablaremos de isomorfismos de Dominios pero es importante tener una caracterización en terminos de funciones.

<u>Proposición</u> <u>10.12</u>.-Si $f:A\rightarrow B$ es un isomorfismo, la imagen de un elemento finito es finita. ⟨⟩.

DEMOSTRACION.-Sea $u\in Con_A$ 't' $y=\langle u\rangle$, como $f(\langle u\rangle)$ es elemento, $f(\langle u\rangle)=\langle f(\langle u\rangle)\rangle$, resta entonces probar que $\exists$ $v\in Con_B$ 't'

$$f(\langle u\rangle)=\langle v\rangle$$

Sea $v=\{X\in D_B \mid rf\{X\}\ r\leq u \quad r \text{ finito}\}$ entonces

Si $X\in f(\langle u\rangle)$ entonces $\exists$ $r\leq\langle u\rangle$ 't' $rf\{X\}$

$\Rightarrow$ $u\vdash r$ y como f es aproximable $uf\{X\}$ pero entonces con $r'=u$ $r'f\{X\}$ y $r'\leq u$ por lo que $X\in v$ y entonces

$\Rightarrow$ $f(\langle u\rangle)\subseteq\langle v\rangle$

Si $X \in v$ $\exists r \subseteq u$ 't' $r f(X)$ pero $u \subseteq \langle u \rangle$ $\therefore$ $X \in f(\langle u \rangle)$

$\therefore$ $v \subseteq f(\langle u \rangle)$ $\therefore$ $\langle v \rangle = f(\langle u \rangle)$.

[].

Proposición 10.13.-Si $f:A \to B$ es aproximable, entonces $\forall\ x \in |A|$

$$f(x)=f(\bigcup \{\langle u \rangle \mid u \subseteq x,\ u \text{ finito}\})= \bigcup f(\langle u \rangle).$$ ().

DEMOSTRACION.-Este resultado es consecuencia directa de proposiciones anteriores. [].

Una observación importante es que un mapeo aproximable queda univocamente determinado por su efecto en los elementos finitos.

Daremos a continuación algunos ejemplo màs de mapeos aproximables.

Proposición 10.14.-Dados Sistemas de Información A, B, C y D, y un elemento $b \in |B|$ fijo, existe un único mapeo aproximable

const $b: A \to B$ tal que:

(i) $(\text{const } b)(x)=b$ $\quad \forall\ x \in |A|$.

Además

(ii) $f \circ (\text{const } b)=\text{const } f(b)$ $\forall$ mapeo aproximable $f:B \to C$.

(iii) $(\text{const } b) \circ g=(\text{const } b)$ $\forall$ mapeo aproximable $g:D \to A$.

().

Obsèrvese que abusamos de la notación ya que en (iii) (const b) tiene dos significados, uno como mapeo de A y otro como mapeo desde D.

DEMOSTRACION.-Sea $b \in |B|$, y sea (const b) definida como sigue:

$$\forall\ u \in Con_A \text{ y } v \in Con_B \quad u(\text{const } b)\, v \Leftrightarrow v \subseteq b.$$

Claramente $\phi(\text{const } b)\phi$;

y si $u(\text{const } b)v$, $u(\text{const } b)v'$ entonces $v\subseteq b$ y $v'\subseteq b$ por lo que $v\cup v'\subseteq b$, y entonces $u(\text{const } b)v\cup v'$;

finalmente , para probar que $(\text{const } b)$ es aproximable :

Si $u'\vdash u$ $u(\text{const } b)v$ $v\vdash v'$ tenemos $v\subseteq b$ y como b es elemento y $v\vdash v'$, entonces $v'\subseteq b$ de donde $u'(\text{const } b)v'$.

$\therefore$ const b es aproximable.

Ahora si x es un elemento arbitrario, es claro que

$$(\text{const } b)(x) \subseteq b,$$

si $X\in b$, tenemos que para cualquier $u\subseteq x$ con u finito

$$u(\text{const } b)\{X\}$$

entonces $X\in(\text{const } b)(x)$, de donde concluimos

$$(\text{const } b)(x)=b \text{ ,}$$

la unicidad se sigue de que si $f(x)=g(x)$ $\forall$ $x\in|A|$ entonces $f=g$.

(ii) $(f \circ (\text{const } b))(x)=f((\text{const } b)(x))=f(b)$ $\forall$ $x\in|A|$, por un argumento de unicidad

$$f \circ (\text{const } b)=\text{const } f(b).$$

(iii) $((\text{const } b) \circ g)(x)=(\text{const } b)(g(x))=b$ $\forall$ $x\in|D|$, nuevamente por unicidad

$$(\text{const } b) \circ g=\text{const } b. \qquad [].$$

Ejemplo

Considerese el siguiente Sistema de Información.

$D=\{\{0\},\{1\},\{0,1\}=\Delta\}$

$F\underset{\vee}{\subseteq}D$ $F\in Con \Leftrightarrow F=\phi$ ó $\bigcap_F S\neq\phi$ y la relación de implicación es:

$$F\vdash X \Leftrightarrow X\supseteq\bigcap_F S.$$

Es fácil comprobar que se trata de un Sistema de Información, mediante la verificación de los axiomas de la definición, también es fácil notar que los elementos totales son:

Verdadero={ {1}, {0,1} }

Falso={ {0} , {0,1} }

y un solo elemento parcial

Indefinido= { {0,1} } .

LLamamos a èste Dominio, el Dominio de los valores de Verdad y lo denotamos por

BOOL={Verdádero, Falso, Indefinido}.

Definimos un mapeo aproximable del Dominio de cadenas binarias llamado Binario (ver ejemplo 4 de elementos Secciòn 7) en el Dominio de valores de verdad, como sigue: Recuerdese que los elementos de Binario son determinados por una ùnica cadena (cuya longitud puede ser finita o infinita), cuando leemos una cadena binaria de izquierda a derecha , contamos el nùmero de sìmbolos 0 antes del primer 1 si esto es impar, damos el valor Verdadero, si es par, Falso, es decir, queremos definir una funciòn f de las cadenas binarias de longitud arbitraria a los valores de verdad tal que, por ejemplo:

f(00000101...)=Verdadero

f(0011011...)=Falso

f(00011011101...)=Verdadero

f(00001101110...)=Fálso

f(0000000000...)=Indefinido

ya que la cadena puede contener su primer sìmbolo 1 en una posiciòn par o impar, o probablemente contiene una infinidad de ceros.

Para ser màs precisos queremos una funciòn tal que:

$$f(0^n 1\bot)=\begin{cases} \text{Verdadero} & \text{si } n \text{ es impar} \\ \text{Falso} & \text{si } n \text{ es par} \\ \text{Indefinido} & \text{si son puros ceros.} \end{cases}$$

Para esto definimos un mapeo aproximable tal que ufv $\Leftrightarrow$

a) $u, v = \phi$,

b b) v=Indefinido

ó c) $\exists$ @$\in$u @(i)=0 i=1,...,n y @(n+1)=1 con n par y v$\leq$Falso

ó d) $\exists$ @$\in$u 't' @(i) i=1,..,n y @(n+1)=1 con n impar y v$\leq$Verdadero.

Probaremos que f así definida es un mapeo aproximable.

(i)Claramente $\phi f \phi$ se cumple.

(ii)Si ufv y ufv' debemos analizar varios casos, analizaremos el caso en que ufv y ufv' cumplen c) de la definición anterior para f y los demás casos se siguen similarmente o más facilmente,(por ejemplo ufv y ufv' no se pueden dar , una por el caso c) y la otra por d) pues entonces u sería inconsistente).

Supongamos $\exists$@$\in$u 't' @(i)=0 i=1,..,n @(n+1)=1 con n par y v$\leq$Falso y $\exists\tau\in$u 't' τ(i)=0 i=1,..,m τ(m+1)=1 y m es par con v'$\leq$Falso, como @,$\tau\in$u @(i)=τ(i) i=1,2,..,min{L(@),L(τ)} .

$\therefore$ m=n y v$\cup$v'$\leq$Falso entonce uf(v$\cup$v').

(iii)Si u'|-u ufv v|-v' (nuevamanete realizaremos un solo caso, y los demás se siguen similarmente).

Caso ufv se cumple por d), entonces $\exists$@$\in$u 't' @(i)=0 i=1,..,n y @(i+1)=1 con n impar y v$\leq$Verdadero, como u'|-u, recuerdese la definición de |- en Binario, debe existir $\tau\in$u 't' τ(i)=0 i=1,2,..,n y τ(n+1)=1 y como v|-v' v'$\leq$Verdadero pues éste es elemento,

$\therefore$ u'fv'.

Ahora veremos como trabaja f con los elementos de Binario.

f($L_@$)={ Y$\in D_V$ | uf{Y} para algun u$\subseteq L_@$ }

si $\exists$ n$\in$N 't' @n$\in L_@$ @n(i)=0 i=1,...,n-1 @n(n)=1 entonces

-si n es par tomando u={@n} vemos que f($L_@$)=Verdadero.

-si n es impar tomando u={@n} f($L_@$)=Falso y -si no existe tal n, $f(L_@)$=Indefinido. ($f(L_@)$ no puede contener ni X={0} ni X={1} ya que @n(i)=0 i=1,..,n $\forall$ n).

Ejemplo

Deseariamos construir una función de Binario en Binario que realizara lo siguiente:

1) $g(0^n 1^k 0c)=0^{n+1}c$

2) $g(1^\omega)=\perp_{Binario}$

3) $g(0^n 1^\omega)=0^n$.

Este ejemplo manda un elemento total en un elemento parcial (tercer inciso) .Intuitivamente g elimina los primeros símbolos 1 consecutivos en una cadena. Realizaremos la definición de mapeo aproximable que logra este efecto en los elementos de Binario.

ugv (=)

a) u, v=$\emptyset$,

ó b) v=$\perp_{Binario}$

ó c) $\exists$ S$\in$u 't' S(i)=0 i=1,...,n (n>0) y si T$\in$v , L(T)<n+1 y $\forall$ t$\in$u T(i)=t(i) i=1,...,min{L(T),L(t)}.

ó d)$\exists$S$\in$u 't' S(i)=0 i=1,..,n (n>0) s(i)=1 i=n+1,...,n+k (k>0) s(n+k+1)=0 L(S)>n+k y si T$\in$v y L(T)<L(S)-k se debe cumplir que $\forall$ t$\in$u

T(i)=t(i) i=1,...,min{n,L(T),L(t)} y

T(i)=t(i+k+1) i=min{n,L(T),L(t)+1},...,min{L(T),L(t)-k-1}.

No probaremos que la definición anterior corresponde a un mapeo aproximable, ya que la prueba es una rutinaria verificación de la definición de mapeo aproximable analizando todos los casos posibles.

Veamos que pasa con los elementos.

Sea L un elemento de Binario (determinado por una única cadena @). Utilizaremos como notación g(@) como $g(L_@)$, y entonces como por definición $g(L_@)=\{Y\in D \mid u g\{Y\}$ para algún $u\subseteq L_@\}$ tenemos

1) Si $@=1^\infty$ observese que $Y=\hat{}\in g(@)$ y no puede ser otro, ya que si $Y\in g(@)$ entonces debe existir $u\in$Con 't' $u\subseteq L_@$ y $ug\{Y\}$ pero obsérvese que para toda $S\in u$ se tendría $S(1)=1$ (no tiene ceros iniciales) así las opciones c) y d) quedan canceladas.

Esto prueba $g(1^\infty)=\underline{1}_{\text{Binario}}$.

2) Si $@=0^m1^\infty$ entonces tomando $u=\{@m\}=v$ por el caso c) tenemos ugv, sin embargo no podemos tener $v=\{@(m+1)\}$ y ugv para ninguna u ya que para cualquier u si $S\in u$ $s(i)=0$ $i=1,..,\min\{L(S),m\}$ y como $L(@(m+1))=m+1$ se anula el caso c) y no existe S con la propiedad pedida en d) . Así $g(@)=L_{@m}$ y por lo tanto

$$g(0^m1^\infty)=0^m.$$

3) Si $@=0^m1^k0c$ entonces si llamamos $T=0^{m+1}c$ es fácil ver que por d) $\{@r\}g\{Tr\}$ $\forall\ r\in\mathbb{N}$ es decir $T=g(@)$ y esto es equivalente a

$$g(0^m1^k0c)=0^{m+1}c.$$

∴ La función tiene el efecto que deseamos. [].

Capitulo III

TEORIA DE DOMINIOS

Secciòn 11 Producto y Uniòn Ajena

En la Categorìa de Conjuntos, el producto cartesiano se construye simplemente considerando el conjunto formado por las parejas ordenadas; esta metodologìa, como veremos, funciona para Dominios, pero su estructura, basada en Sistemas de Informaciòn queda oculta. Por esto daremos la definiciòn en tèrminos de Sistemas de Informaciòn y posteriormente veremos en que forma un elemento del Dominio resultante es una pareja ordenada de elementos de los Dominios constituyentes.

La idea de la definiciòn se refiere a la idea de dar informaciòn sobre posibles elementos. Para dar informaciòn sobre un elemento o pareja, hacemos afirmaciones sobre cada coordenada del elemento.

<u>Definiciòn</u> <u>11.1</u>.-Sean A y B Sistemas de Informaciòn, llamamos AXB, el Sistema Producto, al sistema donde:

(i) $D_{AXB} = \{(X, \Delta_B) \mid X \in D_A\} \cup \{(\Delta_A, Y) \mid Y \in D_B\}$

(ii) $\Delta_{AXB} = (\Delta_A, \Delta_B)$

(iii) $u \in Con_{AXB} \iff prm(u) \in Con_A$ y $sgd(u) \in Con_B$.

(iv') $u \vdash_{AXB} (X, \Delta_B) \iff prm(u) \vdash_A X$

(iv'') $u \vdash_{AXB} (\Delta_A, Y) \iff sgd(u) \vdash_B Y$.

donde u es cualquier subconjunto finito de D_{AXB} y

$$prm(u) = \{X \in D_A \mid (X, \Delta_B) \in u\}$$

$$sgd(u) = \{Y \in D_B \mid (\Delta_A, Y) \in u\} \ \langle\rangle.$$

Obsérvese que los dos conjuntos que unimos en (i) son copias de

D_A y D_B respectivamente.

Para mostrar que nuestra construcción tiene sentido realizamos la siguiente:

<u>Proposición 11.2</u>.-Si A y B son Sistemas de Información entonces:

a) AXB es Sistema de Información.

b) Existen mapeos aproximables

$$\underline{prm}:AXB \to A \quad y \quad \underline{sgd}:AXB \to B$$

tales que, para cualquier Sistema de Información C y mapeos aproximables

$$f:C \to A \qquad y \qquad g:C \to B$$

existe un único mapeo aproximable $\langle f,g\rangle:C \to AXB$ tal que:

$$\underline{prm} \circ \langle f,g\rangle = f \qquad y \qquad \underline{sgd} \circ \langle f,g\rangle = g \quad \langle\rangle.$$

DEMOSTRACION.-Verificamos que AXB satisface los axiomas de la definición 7.1.

(i) Si $u \supseteq v$ y $u \in Con_{AXB}$ entonces

$$prm(v)=\{X\in D_A \mid (X,\Delta_B)\in v\}\subseteq\{X\in D_A \mid (X,\Delta_B)\in u\}=prm(u)\in Con_A$$

y

$$sgd(v)=\{Y\in D_B \mid (\Delta_A,Y)\in v\}\subseteq\{Y\in D_B \mid (\Delta_A,Y)\in u\}=sgd(u)\in Con_B$$

$$\therefore \quad v\in Con_{AXB}.$$

(ii) Si $X\in D_{AXB}$ entonces

$$X=(\Delta_A,Y) \qquad o \qquad X=(Z,\Delta_B)$$

con $\{Y\}\in Con_B$ y $\{Z\}\in Con_A$

entonces $prm(\{X\})=\{\Delta_A\}$ o $\{Z\}$ y

$sgd(\{X\})=\{Y\}$ o $\{\Delta_B\}$

en cualquier caso $\{X\}\in Con_{AXB}$

(iii) Supongamos $u\in Con_{AXB}$ y $u \vdash_{AXB} X$ entonces

$$prm(u\cup\{X\})=\{Z\in D_A \mid (Z,\triangle_B)\in u\cup\{X\}\}$$ y

$$sgd(u\cup\{X\})=\{Y\in D_B \mid (\triangle_A,Y)\in u\cup\{X\}\}$$

como $u\vdash_{AXB} X$

si $X=(Z,\triangle_B)$ $prm(u)\vdash_A Z$, entonces $prm(u)\cup\{Z\}\in Con_A$, de donde $sgd(u\cup\{X\})=sgd(u)\in Con_B$ y $prm(u\cup\{X\})\in Con_A$.

si $X=(\triangle_A,Y)$ simètricamente $sgd(u)\cup\{Y\}\in Con_B$, y observamos que en cualquier caso

$$prm(u\cup\{X\})\in Con_A \quad y \quad sgd(u\cup\{X\})\in Con_B$$

por lo que

$$u\cup\{X\}\in Con_{AXB}.$$

(iv) $\triangle_{AXB}=(\triangle_A,\triangle_B)$, si $u\in Con_{AXB}$ entonces $prm(u)\vdash_A \triangle_A$ y $sgd(u)\vdash_B \triangle_B$, por lo que

$$u\vdash_{AXB} \triangle_{AXB}.$$

(v) Sea $X\in u$, $u\in Con_{AXB}$ entonces si $X=(Z,\triangle_B)$ $Z\in prm(u)$ concluimos $prm(u)\vdash_A Z$, entonces

$$u\vdash_{AXB} X;$$

si $X=(\triangle_A,Y)$ $Y\in sgd(u)$ entonces $sgd(u)\vdash_B Y$ luego entonces

$$u\vdash_{AXB} X.$$

(vi) Supongamos $u\vdash_{AXB} W$ $\forall W\in v$ y $v\vdash_{AXB} X$, debemos probar que $u\vdash_{AXB} X$.

Si $X=(Z,\triangle_B)$ entonces $prm(v)\vdash_A Z$, ahora si $M\in prm(v)$ $W=(M,\triangle_B)\in v$ y entonces $u\vdash W$, es decir

$prm(u)\vdash_A M$ $\forall M\in prm(v)$

como $prm(u)\in Con_A$ $prm(u)\vdash_A Z$ entonces

$$u\vdash_{AXB} X.$$

Igualmente si $X=(\triangle_A,Y)$.

Concluimos que $A \times B$ es Sistema de Información y definiremos ahora los mapeos aproximables

$$\underline{prm}: A \times B \to A \quad y \quad \underline{sgd}: A \times B \to B$$

utilizando la definición de los conjuntos prm() y sgd().

<u>Definición</u> <u>11.3</u>.-Sean $u \in Con_{A \times B}$, $v \in Con_A$ y $w \in Con_B$

(a) $u\underline{prm}v \iff prm(u) \vdash_A v.$

(b) $u\underline{sgd}w \iff sgd(u) \vdash_B w.$ ◊.

Probaremos que estos son mapeos aproximables, verificando la definición para (a) ya que para (b) es análoga.

(i) $\phi\underline{prm}\phi$ se cumple trivialmente por la proposición 7.8 .

(ii) Supongamos $u\underline{prm}v$ y $u\underline{prm}v'$, entonces

$prm(u) \vdash_A v$ y $prm(u) \vdash_A v'$ por lo que $prm(u) \vdash_A (v \cup v')$

$$\therefore\ u\underline{prm}(v \cup v').$$

(iii) Supongamos $u \vdash_{A \times B} u'$ $u'\underline{prm}v'$ y $v' \vdash_A v$ entonces

$u \vdash_{A \times B} X \ \forall X \in u'$ por lo tanto

$prm(u) \vdash_A Z \quad \forall X = (Z, \Delta_B) \in u'$ entonces

$$prm(u) \vdash_A prm(u') \vdash_A v' \vdash_A v$$

$$\therefore\ u\underline{prm}v.$$

<u>Definición</u> <u>11.4</u>.-Sea C un Sistema de Información y $f: C \to A$ y $g: C \to B$ mapeos aproximable, definimos

$$\langle f, g\rangle : C \to A \times B$$

como el siguiente mapeo

$$t\langle f,g\rangle u \iff tf(prm(u)) \quad y \quad tg(sgd(u)).\ ◊.$$

Demostraremos que es aproximable.

(i) $\emptyset\langle f,g\rangle\emptyset$ se cumple pues $prm(\emptyset)=\emptyset$ y $sgd(\emptyset)=\emptyset$ y se tiene $\emptyset f\emptyset$ y $\emptyset g\emptyset$ pues f y g son aproximables.

(ii) Supongamos $t\langle f,g\rangle u$ y $t\langle f,g\rangle u'$ entonces

$$tf(prm(u)),\ tf(prm(u')),\ tg(sgd(u)) \text{ y } tg(sgd(u'))$$

pero f y g son aproximables entonces tenemos

$$tf(prm(u)\cup prm(u')) \quad \text{y } tg(sgd(u)\cup sgd(u')) \text{ pero}$$

$$prm(u)\cup prm(u')=\{Z\in D_A \mid (Z,\Delta_B)\in u\}\cup\{Z\in D_A \mid (Z,\Delta_B)\in u'\}$$

$=\{Z\in D_A \mid (Z,\Delta_B)\in u\cup u'\}= prm(u\cup u')$ entonces

$tf(prm(u\cup u'))$ y análogamente $tg(sgd(u\cup u'))$

$$\therefore\ t\langle f,g\rangle(u\cup u').$$

(iii) Si $t'\vdash_C t$ $t\langle f,g\rangle u$ y $u\vdash_{AXB} u'$, entonces $tf(prm(u))$ y $tg(sgd(u))$ como $u\vdash_{AXB} u'$ tenemos

$$prm(u)\vdash_A prm(u') \text{ y } sgd(u)\vdash_A sgd(u')$$

finalmente f y g son aproximables, asi que

$$t'f(prm(u')) \text{ y } t'g(sgd(u')) \text{ es decir}$$

$$t'\langle f,g\rangle u'.$$

Ahora sea $c\in|C|$ un elemento de $|C|$

$\underline{prm} \circ \langle f,g\rangle\ (c)=\underline{prm}(\langle f,g\rangle(c))$

$=\ \{Z\in D_A \mid u\underline{prm}\{Z\}$ para algún $u\subseteq\langle f,g\rangle(c)\}$

$=\ \{Z\in D_A \mid prm(u)\vdash_A Z$ para algún $u\subseteq\langle f,g\rangle(c)\ \}$

$=(3)\ \{Z\in D_{A'} \mid tf\{Z\}$ para alguna $t\subseteq c\} =f(c)$

La igualdad (3) se sigue al comprobar la doble contención:

Para ($\supseteq$) tomese $u=\{(Z,\Delta_B)\}\subseteq\langle f,g\rangle(c)$ y para ($\subseteq$) obsérvese que si $prm(u)\vdash_A Z$ y $u\subseteq\langle f,g\rangle(c)$ entonces $tf(prm(u))$ entonces $tf\{Z\}$.

De la misma manera se tiene

$$\underline{sgd} \circ \langle f,g\rangle(c)= g(c) \quad \forall\ c\in|C|$$

$$\therefore\ \underline{prm} \circ \langle f,g\rangle=f \qquad \text{y} \quad \underline{sgd} \circ \langle f,g\rangle=g$$

Probaremos ahora el siguiente :

Lema 11.5.-Si r, p son elementos de $A \times B$ tales que

$$\underline{prm}(r)=\underline{prm}(p) \quad y \quad \underline{sgd}(r)=\underline{sgd}(p)$$

entonces $\qquad r=p$. $\qquad$ ().

DEMOSTRACION.-Sea $X \in r$ si $X=(Z, \triangle_B)$, $Z \in \underline{prm}(r)=\underline{prm}(p)$

$\underline{prm}(p)=\{Z \in D_A \mid u\underline{prm}\{Z\}$ para algún $u \leq p\}$

$=\{Z \in D_A \mid prm(u) \vdash_A Z$ para algún $u \leq p\}$ entonces

si $Z \in \underline{prm}(p)$ existe $u \leq p$ 't' $prm(u) \vdash_A Z$, es decir

$$u \vdash_{A \times B} (Z, \triangle_B)$$

y como p es elemento $X \in p$.

Análogamente si $X=(\triangle_A, Y)$ se tiene por $\underline{sgd}(r)=\underline{sgd}(p)$ que $X \in p$;

$$\therefore \quad r \leq p.$$

Simétricamente $\quad p \leq r \quad$ y

$$\therefore \quad r=p. \qquad [].$$

Probaremos ahora la unicidad de $\langle f, g \rangle$.

Si $h: C \to A \times B$ es una función tal que

$$\underline{prm} \circ h=f \quad y \quad \underline{sgd} \circ h=g,$$

sea $c \in |C|$, entonces

$\underline{prm}(h(c))=\underline{prm}(\langle f, g\rangle(c))$

$\underline{sgd}(h(c))=\underline{sgd}(\langle f, g\rangle(c))$ y por el lema anterior

$h(c)=\langle f, g\rangle(c)$, como c fue elegida arbitrariamente en $|C|$

$h=\langle f, g\rangle$

$$\therefore \quad \langle f, g\rangle \text{ es única} \qquad [].$$

Hemos concluido esta larga demostración de la construcción del producto. Mostraremos que podemos dar una biyección entre los elementos del producto y las parejas ordenadas de elementos.

Definición 11.6.-Una pareja de elementos es:

$$(x,y)=\langle \text{const } x, \text{const } y\rangle(\underline{|}_{C}) \qquad ().$$

Es decir (x,y) es el elemento de |AXB| que resulta al considerar $f=\text{const } x:C \to A$ y $g=\text{const } y:C \to B$ y evaluar la función producto $\langle \text{const } x, \text{const } y\rangle$ en $\underline{|}_{C}$.

Obsèrvese que la definición no depende del Sistema de Información C que se tome, pues si

$(x',y')=\langle \text{const } x, \text{const } y\rangle(\underline{|}_{C'})$ se tiene

$\underline{prm}((x,y))=\text{const } x(\underline{|}_{C})=x$.

$\underline{prm}((x',y'))=\text{const } x(\underline{|}_{C'})= x$, anàlogamente

$$\underline{sgd}((x,y))=\underline{sgd}((x',y'))$$

y por el lema 11.5

$$(x,y)=(x',y').$$

Como notación, para eliminar paréntesis inútiles, las funciones evaluadas en eneadas seràn directamente escritas por el nombre de la función seguido de la eneada, asi

$$\underline{prm}((x,y))=\underline{prm}(x,y).$$

La verificación de la consistencia de la definición 11.6 de hecho prueba los incisos (ii) y (iii) de la siguiente:

Proposición 11.7.-Sean A y B Sistemas de Información entonces:

(i) $(x,y)=\{(Z,\Delta_B)\mid Z\in x\}\cup\{(\Delta_A,Y)\mid Y\in y\}\in |AXB|$.

(ii) $\underline{prm}(x,y)=x$

(iii) $\underline{sgd}(x,y)=y$

(iv) $z=(\underline{prm}(z),\underline{sgd}(z)) \quad \forall\ z\in|AXB|$

(v) $\langle f,g\rangle(c)=(f(c),g(c)) \quad \forall\ c\in|C|$

(vi) $(x,y)\subseteq(x',y')$ $(\Leftrightarrow)$ $x\subseteq x'$ $y\subseteq y'$. ().

DEMOSTRACION.- $(x,y)\in|A\times B|$ pues es la imagen de un elemento bajo un mapeo aproximable.

Ahora, (i) si $t=\{(Z,\Delta_B)\mid Z\in x\}\cup\{(\Delta_A,Y)\mid Y\in y\}$, es fàcil ver que t es un elemento, y entonces

$$\underline{prm}(t)=\{Z\in D_A \mid u\ \underline{prm}\ Z\} \text{ para algùn } u\subseteq t\}$$
$$=\{Z\in D_A \mid prm(u)\vdash_A Z \text{ para algùn } u\subseteq t\}$$

Si $Z\in x$, entonces $u=\{(Z,\Delta_B)\}\in t$ y $prm(u)\vdash_A Z$

$$\therefore\ \underline{prm}(x,y)=x\subseteq \underline{prm}(t).$$

Si $Z\in\underline{prm}(t)$ entonces existe u 't' $u\subseteq t$ y $prm(u)\vdash_A Z$ pero $prm(u)\subseteq x$ y como x es elemento $Z\in x$

$$\therefore\ x=\underline{prm}(t)$$

igualmente $y=\underline{sgd}(t)$ y aplicando el lema 11.5

$$(x,y)=t.$$

(iv) Aplicando (ii), (iii) y nuevamente el lema 11.5 a

$$\underline{prm}(z)=\underline{prm}(\underline{prm}(z),\underline{sgd}(z))$$
$$\underline{sgd}(z)=\underline{sgd}(\underline{prm}(z),\underline{sgd}(z))$$

tenemos el resultado.

(v) Aplicando (ii), (iii) y la propiedad universal para $\langle f,g\rangle$ tenemos $\forall\ c\in|C|$

$\underline{prm}(\langle f,g\rangle(c))=f(c)$ y $\underline{prm}(f(c),g(c))=f(c)$

$\underline{sgd}(\langle f,g\rangle(c))=g(c)$ y $\underline{sgd}(f(c),g(c))=g(c)$ y por el lema 7.5

$$\langle f,g\rangle(c)=(f(c),g(c)).$$

(vi) $(\Leftarrow)$ Supongamos $x\subseteq x'$ $y\subseteq y'$ entonces por (i)

$(x,y)=\{(Z,\Delta_B)\mid Z\in x\}\cup\{(\Delta_A,Y)\mid Y\in y\}$

$\subseteq\{(Z,\Delta_B)\mid Z\in x'\}\ \{(\Delta_A,Y)\mid Y\in y'\}=(x',y')$.

$(\Rightarrow)$ Supongamos $(x,y)\subseteq(x',y')$

entonces por ser $\underline{prm}$ y $\underline{sgd}$ aproximables, son monòtonas, de donde $\underline{prm}(x,y)\subseteq\underline{prm}(x',y')$ y $\underline{sgd}(x,y)\subseteq\underline{sgd}(x',y')$ entonces

$x \leq x' \quad y \leq y'.$ [].

De esta manera a una pareja de elementos $x \in |A|$ y $y \in |B|$ le asociamos el elemento (x,y), dado un elemento $z \in |A \times B|$ le asociamos la pareja ordenada $(\underline{prm}(z), \underline{sgd}(z))$. Esta biyección muestra que $|A \times B|$ es isomorfo a $|A| \times |B|$ donde éste último es el producto cartesiano de los conjuntos de elementos.

Mostraremos que en nuestra categoría existe objeto terminal y por lo tanto es una categoría con productos finitos.

Considerese el siguiente Sistema de Información:

$D_1 = \{\Delta\}$, $Con_1 = \{\emptyset, D_1\}$ y $\vdash_1$ la relación minimal, es fácil verificar que el Dominio $|D_1|$ llamado $\mathbb{I}$ tiene como único elemento a $\perp_{\mathbb{I}}$.

De esto se concluye que todos los mapeos aproximables $f:\mathbb{I} \to A$ son constantes y $f:A \to \mathbb{I}$ es siempre un único mapeo aproximable, es más se trata de $f = (\text{const } \perp_{\mathbb{I}})$ y es denotado por $f = 0$.

Lo mencionado antes prueba la:

<u>Proposición 11.8</u>.-$\mathbb{I}$ es objeto terminal de la Categoría de Dominios.

().

Algunos resultados interesantes para el producto son los siguientes

<u>Proposición 11.9</u>.-Sean $u, v \in Con_{A \times B}$ entonces

$u \vdash_{A \times B} v$ si y sólo si $prm(u) \vdash_A prm(v)$ y $sgd(u) \vdash_A sgd(v)$.

().

DEMOSTRACION.-Directamente de la definición 11.1 tenemos

$u \vdash_{A \times B} v$ si y sólo si

$u \vdash_{AXB} X \ \forall X \in v$, y esto se cumple si y sólo si

$prm(u) \vdash_A Z \quad \forall Z$ 't' $(Z, \Delta_B) \in v$ y

$sgd(u) \vdash_B Y \quad \forall Y$ 't' $(\Delta_A, Y) \in v$, y esto si y sólo si

$prm(u) \vdash_A prm(v)$ y $sgd(u) \vdash_B sgd(v)$. [].

<u>Proposición 11.10</u>.-Sea $t \subseteq (x,y) \in |AXB|$ entonces

$$\langle t \rangle = (\langle u \rangle, \langle v \rangle) \text{ donde } u = prm(t) \text{ y } v = sgd(t). \ \langle\rangle.$$

DEMOSTRACION.-Clatamente $t \subseteq (\langle u \rangle, \langle v \rangle)$ y como éste último es elemento $\langle t \rangle \subseteq (\langle u \rangle, \langle v \rangle)$.

Si $X \in (\langle u \rangle, \langle v \rangle)$, supongamos $X = (Z, \Delta_B)$, entonces $Z \in \langle u \rangle$ de manera que $prm(t) \vdash_A Z$ pero entonces $(Z, \Delta_B) \in \langle t \rangle$ y entonces

$\langle t \rangle \supseteq (\langle u \rangle, \langle v \rangle)$. [].

<u>Proposición 11.11</u>.-Una función $f : |AXB| \to |C|$ de dos argumentos, dada en los elementos, proviene de un mapeo aproximable si y sólo si $\forall\, a \in |A|$ y $\forall\, b \in |B|$ las funciones

$$f_b : |A| \to |C| \quad \text{y} \quad f_a : |B| \to |C| \text{ dadas por}$$

$$f_b(x) = f(x,b) \quad \text{y} \quad f_a(y) = f(a,y)$$

respectivamente, provienen de mapeos aproximables. $\langle\rangle$.

DEMOSTRACION.-Supongamos f proviene de un mapeo aproximable, entonces $f_b = f \circ \langle I_A, \text{const } b \rangle$ y

$$f_a = f \circ \langle \text{const } a, I_B \rangle$$

de donde es claro que f_b y f_a provienen de mapeos aproximables.

($\Leftarrow$)Observese que

$f(x,y) = f_y(x) = \bigcup \{ f_y(\langle u \rangle) \mid u \subseteq x \ u \text{ finito} \} =$

$\bigcup \{ f(\langle u \rangle, y) \mid u \subseteq x \ u \text{ finito} \} =$

$\bigcup \{ f_{\langle u \rangle}(y) \mid u \subseteq x \ u \text{ finito} \} =$

$\bigcup \{ \bigcup \{ f_{\langle u \rangle}(\langle v \rangle) \mid v \subseteq y \ v \text{ finito} \} \ u \subseteq x \ u \text{ finito} \}$

$=\bigcup\{f(\langle u\rangle,\langle v\rangle) \mid u\subseteq x \ v\subseteq y$ finitos$\}$

$\therefore$ f es aproximable. []

Definiremos ahora un objeto cercano al coproducto de Dominios. La idea es que los elementos de este Dominio sean, elementos de los Dominios constituyentes, pero de manera exclusiva.

<u>Definición 11.12</u>.-Sean A y B Sistemas de Información. Por A+B, el sistema unión ajena(suma directa), entendemos el sistema donde tenemos un dato $\Delta\notin D_A$ y $\Delta\notin D_B$ y:

(i) $D_{A+B}=\{(X,\Delta)\mid X\in D_A\}\cup\{(\Delta,Y)\mid Y\in D_B\}\cup\{(\Delta,\Delta)\}$

(ii) $\Delta_{A+B}=(\Delta,\Delta)$.

(iii) $u\in Con_{A+B} \iff izq(u)\in Con_A$ y $der(u)=\emptyset$

ó (inclusivo)

$izq(u)=\emptyset$ y $der(u)\in Con_B$.

(iv') $u\vdash_{A+B}(X,\Delta) \iff der(u)=\emptyset$ e $izq(u)\vdash_A X$.

(iv'') $u\vdash_{A+B}(\Delta,Y) \iff izq(u)=\emptyset$ y $der(u)\vdash_B Y$.

(iv''') $u\vdash_{A+B}(\Delta,\Delta) \ \forall\ u\in Con_{A+B}$.

donde u es un subconjunto finito de D_{A+B} en (iii) y en (iv'), (iv'') $u\in Con_{A+B}$, además, la notación izq y der significa:

$$izq(u)=\{Z\in D_A \mid (Z,\Delta)\in u\}$$

$$der(u)=\{Y\in D_B \mid (\Delta,Y)\in u\}. \qquad \langle\rangle.$$

Obsèrvese que (Δ,Δ), (Δ,Δ_B) y (Δ_A,Δ) no son equivalentes en este sistema, además $\vdash_{A+B}$ esta definida de manera disyuntiva en vez de conjuntiva como $\vdash_{AXB}$. Esto nos lleva a que A+B contenga copias ajenas de A y B ademas de un elemento extra ($\perp_{A+B}$).

Para mostrar los objetivos de la definición anterior tenemos la siguiente :

Proposición 11.13.-Si A y B son Sistemas de Información entonces:

(i) $A+B$ también lo es.

(ii) Existen mapeos aproximables

$$ini: A \to A+B \quad e \quad ind: B \to A+B$$

tales que para cualquier Sistema de Información C y mapeos aproximables $f: A \to C$ y $g: B \to C$ existe un único mapeo aproximable

$$[f,g]: A+B \to C$$

tal que:

$$[f,g] \circ ini = f \quad , \quad [f,g] \circ ind = g \quad y \quad [f,g](\perp_{A+B}) = \perp_C .$$

().

DEMOSTRACION.-Para ver que se trata de un Sistema de Información veremos que se verifican los axiomas.

(i) Es inmediato del hecho de que si $v \in Con_{A+B}$ y $u \subseteq v$ entonces

$$izq(u) = \{Z \in D_A \mid (Z,\Delta) \in u\} \subseteq \{Z \in D_A \mid (Z,\Delta) \in v\} = izq(v)$$

y de la misma manera

$$der(u) \subseteq der(v)$$

de manera que si $izq(v) \in Con_A$ entonces $izq(u) \in Con_A$

si $der(v) \in Con_B$ entonces $der(u) \in Con_B$

y si $izq(v) = \emptyset$ entonces $izq(u) = \emptyset$

si $der(v) = \emptyset$ entonces $der(u) = \emptyset$.

(ii) Sea $X \in D_{A+B}$, si $X = (Z,\Delta)$ con $Z \in D_A$, entonces

$$izq(\{X\}) = \{Z\} \neq \text{ y } der(\{X\}) = \emptyset \quad \therefore \{X\} \in Con_{A+B} .$$

Si $X = (\Delta, Y)$ con $Y \in D_B$ entonces $izq(\{X\}) = \emptyset$ y $der(\{X\}) = \{Y\}$

$$\therefore \{X\} \in Con_{A+B} .$$

Finalmente si $X = (\Delta,\Delta) \Rightarrow izq(\{X\}) = der(\{X\}) = \emptyset$ y

$$\therefore \{X\} \in Con_{A+B} .$$

(iii) Supongamos $u \vdash_{A+B} X$, si $X = (\Delta,\Delta)$ entonces

$$izq(u) = izq(u \cup \{X\}) \text{ y } der(u) = der(u \cup \{X\}).$$

Si $X=(Z,\triangle)$, entonces $izq(u\cup\{X\})=izq(u)\cup\{Z\}\neq\emptyset$ e $izq(u\cup\{X\})\in Con_{A}$ por la definición de $\vdash_{A+B}$ y además

$$der(u\cup\{X\})=der(u)=\emptyset$$

$$\therefore \quad u\cup\{X\}\in Con_{A+B}.$$

Análogamente si $X=(\triangle,Y)$.

(iv)Como $u\vdash_{A+B}(\triangle,\triangle)$ siempre, el axioma se cumple automáticamente.

(v)Supongamos $X\in u$ $u\in Con_{A+B}$

si $X=(\triangle,Y)$, entonces $Y\in der(u)$ por lo que $der(u)\neq\emptyset$ y $der(u)\vdash_{B} Y$ entonces $u\vdash_{A+B}(\triangle,Y)$.

De la misma manera se prueba si $X=(Z,\triangle)$ y trivilamente si $X=(\triangle,\triangle)$.

(vi)Supongamos $u\vdash_{A+B} W$ $\forall\, W\in v$ y $v\vdash_{A+B} X$ debemos probar que $u\vdash_{A+B} X$.

Supongamos $X=(\triangle,Y)$ entonces $der(v)\neq\emptyset$ y $der(v)\vdash_{B} Y$ afirmamos que $der(u)\vdash_{B} M$ $\forall\, M\in der(v)$; sea $M\in der(v)$ entonces existe $W=(\triangle,M)\in v$ y por hipótesis $u\vdash_{A+B}(\triangle,M)$ de donde $der(u)\neq\emptyset$ y $der(u)\vdash_{B} M$, aplicando la transitividad de $\vdash_{B}$ tenemos $der(u)\vdash_{B} Y$ y por lo tanto $u\vdash_{A+B} X$.

La prueba se sigue igualmente si $X=(Z,\triangle)$ y aún más fácilmente si $X=(\triangle,\triangle)$.

Concluimos que A+B es un Sistema de Información.

Definimos ahora $ini:A\to A+B$ e $ind:B\to A+B$ mediante:

<u>Definición</u> <u>11.14</u>.-Sean A y B Sistemas de Información, definimos los mapeos aproximables $ini:A\to A+B$ e $ind:B\to A+B$ como:

(a)$v\ ini\ u \iff \{(Z,\triangle)\mid Z\in v\}\vdash_{A+B} u.$

(b)$w\ ind\ u \iff \{(\triangle,Y)\mid Y\in w\}\vdash_{A+B} u.$ ◊.

Debemos probar que estas definiciones corresponden a mapeos aproximables. Realizaremos únicamente (a) ya que (b) es análogo.

(i)$\emptyset$ ini $\emptyset$ es directo.

(ii) Supongamos que tenemos v ini u y v ini u', entonces por definición

$\{(Z,\hat{\ })|Z\in v\} \vdash_{A+B} u$ y $\{(Z,\hat{\ })|Z\in v\} \vdash_{A+B} u'$ por la proposición 7.8

$\{(Z,\hat{\ }) \mid Z\in v\} \vdash_{A+B} (u \cup u')$; por lo que v ini $(u \cup u')$.

(iii) Supongamos $v \vdash_{A} v'$ v' ini u' y $u' \vdash_{A+B} u$ entonces

$\{(Z,\hat{\ })|Z\in v\} \vdash_{A+B} \{(Z,\hat{\ })|Z\in v'\}$ pues

izq$(\{(Z,\hat{\ })\mid Z\in v\})=v$ e izq$(\{(Z,\hat{\ }) \mid Z\in v'\})=v'$ y $v \vdash_{A} v'$

por lo tanto $\{(Z,\hat{\ }) \mid Z\in v\} \vdash_{A+B} u$

$\therefore$ v ini u.

Hemos comprobado la definición de mapeo aproximable para ini y ahora comprobaremos que ini e ind cumplen con la propiedad universal para la unión ajena.

Sea C un Sistema de Información arbitrario y $f:A \to C$ y $g:B \to C$ mapeos aproximables,

<u>Definición 11.15</u>.-Definimos entonces

$$[f,g]:A+B \to C \text{ por}$$

$u[f,g]s \iff \emptyset \vdash_{C} s$

ó izq(u)$\neq\emptyset$ e (izq(u))fs

ó der(u)$\neq\emptyset$ y der(u)gs. ().

Probaremos que [f,g] es aproximable verificando la definición.

(i) $\emptyset[f,g]\emptyset$ se cumple por el primer caso de la definición anterior.

(ii) Supongamos u[f,g]s y u[f,g]s', entonces si izq(u)=der(u)=$\emptyset$ entonces $u[f,g](s\cup s')$ debe cumplirse nuevamente por el primer caso.

De lo contrario, uno y sólo uno de los conjuntos der(u) e izq(u) es distinto del vacío, supongamos der(u)$\neq\emptyset$, entonces tenemos

$der(u)gs$ y $der(u)gs'$ pero g es aproximable, de donde $der(u)g(s\cup s')$ con lo que concluimos

$$u[f,g](s\cup s').$$

Simètricamente obtenemos $u[f,g](s\cup s')$ si $izq(u)\neq\emptyset$.

(iii)Supongamos $u\vdash_{A+B} u'$ $u'[f,g]s'$ y $s'\vdash_{C} s$.

Haremos el caso $izq(u')\neq\emptyset$ y los casos $der(u')\neq\emptyset$ y ambos vacios se siguen fàcilmente.

Como $izq(u')fs'$ y $s'\vdash_{C} s$ tenemos, pues f es aproximable, $izq(u')fs$ ahora, como $u\vdash_{A+B} u'$ e $izq(u')\neq\emptyset$ obtenemos que para toda $(Z,\triangle)\in u'$ $u\vdash_{A+B} (Z,\triangle)$ de donde $izq(u)\neq\emptyset$ e $izq(u)\vdash Z$ utilizando nuevamente que f es aproximable y que $izq(u)\vdash_{A} izq(u')$ concluimos $izq(u)fs$, es decir

$$u[f,g]s.$$

Comprobaremos ahora la conmutatividad afirmada por la propiedad universal.

Sea $a\in|A|$ entonces

$[f,g]\circ ini(a)=[f,g](ini(a))$

como $ini(a)=\{X\in D_{A+B}$ $u\ ini\ \{X\}$ para algùn $u\subseteq a\}$

$[f,g](ini(a))=\{S\in D_{C}\ |\ r[f,g]\{S\}$ para algùn $r\subseteq ini(a)\ \}$

(4) $=\{S\in D_{C}\ |\ uf\{S\}$ para algùn $u\subseteq a\} = f(a)$.

La igualdad (4) se sigue de la doble contenciòn de los conjuntos correspondientes:

($\supseteq$)Si $S\in D_{C}$ y $u\subseteq a$ es tal que $uf\{S\}$ entonces si $r=\{(Z,\triangle)|Z\in u\}$ tenemos $r[f,g]\{S\}$ pues $izq(r)=u$ y ademàs

$r\subseteq\{X\in D_{A+B}\ |\ u\ ini\{X\}$ para algùn $u\subseteq a\}=ini(a)$ ya que $u\ ini\{X\}$ para toda $X\in r$ pues

$u\ ini\{X\}$ si y sòlo si $\{(Z,\triangle)|Z\in u\}\vdash_{A+B} X$ si y sòlo si $r\vdash_{A+B} X$

($\subseteq$)Sea $S\in D_{C}$ 't' $r[f,g]\{S\}$ para algùn r ini(a), como r es finito $r=\{X_1,\ldots,X_n\}$ y existen $u_1,\ldots,u_n\subseteq a$ 't' $u_i\ ini\{X_i\}$ $i=1,\ldots,n$ por definición de imagen.

Sea $u=\hat{\bigcup} u_i$ entonces, como ini es aproximable tenemos u ini r y $u\subseteq a$, afirmamos $uf\{S\}$. Sea $r'=\{(Z,\triangle)\mid Z\in u\}$ entonces $r'\vdash_{A+B} r$ y como $[f,g]$ es aproximable, $r'[f,g]\{S\}$ pero $izq(r')=u$ de donde $izq(r')f\{s\}$ y $uf\{S\}$.

De la misma manera se prueba que $[f,g]$ o ind$=g$.

Para probar que $[f,g](\perp_{A+B})=\perp_C$ observese que:

$[f,g](\perp_{A+B})=\{Y\in D_C \mid u[f,g]\{Y\}$ para algún $u\subseteq\perp_{A+B}\}$
$=\{Y\in D_C \mid u[f,g]\{Y\}\ \emptyset\vdash_{A+B} u\}=\{Y\in D_C \mid \emptyset[f,g]\{Y\}\}$
$=\{Y\in D_C \mid \emptyset\vdash_C Y\}=\perp_C$.

Finalmente, para la unicidad tenemos necesidad del:

<u>Lema 11.16</u>.-Sean A,B Sistemas de Información, si $x\in|A+B|$ y $x\neq\perp_{A+B}$ entonces existe $a\in|A|$ ó (exclusivo) $b\in|B|$ tal que:

ini$(a)=x$ ó (exclusivo) ind$(b)=x$. ⟨⟩.

DEMOSTRACION.-Sea $x\in|A+B|$ y $x\neq\perp_{A+B}$, sea $a=\{Z\in D_A \mid (Z,\triangle)\in x\}$ si $a=\emptyset$, tomese $b=\{Y\in D_B \mid (\triangle,Y)\in x\}\neq\emptyset$ y se trata como el caso siguiente.

Si $a\neq\emptyset$, afirmamos $a\in|A|$ e ini$(a)=x$.

Verificamos la definición de elemento para a: sea $v\subseteq a$ y v finito, si $v=\emptyset$ entonces $v\in Con_A$, si $v\neq\emptyset$ tenemos

$$r=\{(Z,\triangle)\mid Z\in v\}\subseteq x$$

como x es elemento $r\in Con_{A+B}$ pero $izq(r)=v$ y por la definición de unión ajena esto implica $v\in Con_A$.

Además, si $v\vdash_A Z$, entonces $r\vdash_{A+B}(Z,\triangle)$ y como x es elemento tenemos $(Z,\triangle)\in x$ de donde $Z\in a$.

$\therefore$ a es elemento.

ini$(a)=\{X\in D_{A+B} \mid u$ ini$\{X\}$ para algún $u\subseteq a\}$
$=\{X\in D_{A+B} \mid \{(Z,\triangle)\ Z\in u\}\vdash_{A+B} X$ para algún $u\subseteq a\}$

Si $X\in x$ entonces $X=(Z,\triangle)$ (X no puede ser de la forma $(\triangle,Y)$ pues

$a \neq \emptyset$ y entonces x no sería elemento pues tendría un subconjunto finito inconsistente). Tomando $u=\{Z\}$ se comprueba que $X \in ini(a)$.

Si $X \in ini(a)$ entonces por definición de ini(a) existe u 't'

$\{(Z,\Delta) \mid Z \subseteq u\} \vdash X$ y $\{(Z,\Delta) \mid Z \subseteq u\} \subseteq x$, como x es elemento, $X \in x$.

Como hemos verificado la doble contención queda probada la igualdad.

El δ es exclusivo, pues de no ser así tomaríamos un dato de b y uno de a y obtendríamos un subconjunto de x finito e inconsistente lo cual contradice que x es elemento.

Probaremos ahora la unicidad de $[f,g]$ y con esto concluimos la demostración de la afirmación para la unión ajena.

Supongamos $h: A+B \to C$ es otra función tal que

$$h \circ ini = f \qquad y \ h \circ ind = g \quad y \ h(\perp_{A+B}) = \perp_{C} .$$

Sea $x \in |A+B|$, si $x = \perp_{A+B}$ entonces directamente $h(x) = [f,g](x)$

si $x \neq \perp_{A+B}$, por el lema anterior existe $a \in |A|$ (si no $b \in |B|$) tal que $ini(a) = x$ (si no $ind(b) = x$) pero esto nos dice

$$h(x) = h(ini(a)) = f(a) = [f,g](ini(a)) = [f,g](x)$$

$$(\quad h(x) = h(ind(b)) = g(b) = [f,g](ind(b)) = [f,g](x) \quad)$$

$$\therefore \ \forall\, x \in |A+B| \quad [f,g](x) = h(x)$$

$$\therefore \ [f,g] = h \qquad\qquad \therefore \ [f,g] \text{ es única. } [].$$

Por las propiedades universales del producto y la unión ajena, así como del objeto terminal, puede verificarse que estos son únicos salvo isomorfismo.

Además es fácil comprobar que _XA y AX_ son extendibles a funtores, lo mismo que _+A y A+_; por ejemplo si consideramos

$$\begin{tikzcd} B \arrow[r, "AX\_"] \arrow[d, "f"'] & AXB \arrow[d, "(IA \circ \underline{prm},\, f \circ \underline{sgd})"] \\ B' \arrow[r] & AXB' \end{tikzcd}$$

Para verificar que es funtor observese que directamente

$I_{AXB} = \langle \underline{prm}, \underline{sgd} \rangle$ y $\forall (x,y) \in |AXB|$

$\langle I_A \circ \underline{prm}, f \circ \underline{sgd} \rangle \circ \langle I_A \circ \underline{prm}, g \circ \underline{sgd} \rangle (x,y)$

$= \langle I_A \circ \underline{prm}, f \circ \underline{sgd} \rangle \; (x, g(y))$

$= (x, f(g(y))) = \langle I_A \circ \underline{prm}, f \circ g \circ \underline{sgd} \rangle (x,y).$

EJEMPLOS

Ejemplo 1)Observese que si $\mathbb{I}$ denota al Dominio con un solo elemento , entonces

$BOOL \cong \mathbb{I} + \mathbb{I}$

ya que $\mathbb{I} + \mathbb{I} = \{ \perp, (\perp_{\mathbb{I}}, \Delta), (\Delta, \perp_{\mathbb{I}}) \}$

y además todo mapeo aproximable de BOOL en algún otro Dominio queda determinado por los valores que tome en

$\perp$=Indefinido $(\perp_{\mathbb{I}}, \Delta)$=Verdadero $(\Delta, \perp_{\mathbb{I}})$=Falso

bajo la restricción que:

(5) La imagen de $\perp$ debe estar contenida en la intersección de las imagenes de Verdadero y Falso.

Inversamente, toda función que satisface (5), cuyo dominio de definición es BOOL, proviene de un mapeo aproximable.

Ejemplo 2) Sea A un Sistema de Información, entonces existe diag:A-> AXA un mapeo aproximable tal que

$\forall x \in |A|$ diag(x)=(x,x). ().

DEMOSTRACION.-Sea $f=I_A$ y $g=I_A$ conforme a la definición del producto de funciones, llamamos $diag = \langle I_A, I_A \rangle$ la cual es garantizada por la propiedad universal del producto, y por esto mismo

$\underline{prm} \circ diag = I_A$ y $\underline{sgd} \circ diag = I_A$

entonces claramente $\forall x \in |A|$ diag(x)=(x,x). [].

Proposición 11.17.-Si A, B y C son Sistemas de Información entonces:

(i) $|A\times B|\cong|B\times A|$.

(ii) $|A\times(B\times C)|\cong|(A\times B)\times C|$.

(iii) Si $|A|\cong|A'|$ y $|B|\cong|B'|$ entonces $|A\times B|\cong|A'\times B'|$.

().

DEMOSTRACION.-(i) Sea $D=A\times B$ y $E=B\times A$ entonces

$\underline{prm}_D : A\times B \to A$ y $\underline{sgd}_D : A\times B \to B$ y tenemos

$$\langle \underline{sgd}_D, \underline{prm}_D\rangle : A\times B \to B\times A$$

análogamente $\langle \underline{sgd}_E, \underline{prm}_E\rangle : B\times A \to A\times B$

ahora $\langle \underline{sgd}_E, \underline{prm}_E\rangle \circ \langle \underline{sgd}_D, \underline{prm}_D\rangle : A\times B \to A\times B$ y $\forall\ (x,y)\ |A\times B|$

$$\langle \underline{sgd}_E, \underline{prm}_E\rangle \circ \langle \underline{sgd}_D, \underline{prm}_D\rangle (x,y)$$
$$=\langle \underline{sgd}_E, \underline{prm}_E\rangle (\langle \underline{sgd}_D, \underline{prm}_D\rangle (x,y))$$
$$=\langle \underline{sgd}_E, \underline{prm}_E\rangle (\underline{sgd}(x,y), \underline{prm}_D(x,y))$$
$$=\langle \underline{sgd}_E, \underline{prm}_E\rangle (y,x) = (x,y)$$

por lo tanto esta composición es $I_{A\times B}$.

Simétricamente

$$\langle \underline{sgd}_D, \underline{prm}_D\rangle \circ \langle \underline{sgd}_E, \underline{prm}_E\rangle = I_{B\times A}$$

de donde $|A\times B|\cong|B\times A|$.

(ii) Definimos $f:|A|\times|B\times C| \to |A\times B|\times|C|$ y $g:|A\times B|\times|C| \to |A|\times|B\times C|$ en los elementos como sigue:

$f(x,(y,z))=((x,y),z)$ y $g((x,y),z)=(x,(y,z))$, por las proposición 10,6 y 11.11 resulta rutinario verificar que provienen de mapeos aproximables y claramente f y g son isomorfismos.

(iii) Sean $f:A \to A'$ y $g:B \to B'$ isomorfismos si llamamos $D=A\times B$ y $E=A'\times B'$ entonces

$$\langle f \circ \underline{prm}_D, g \circ \underline{sgd}_D\rangle : D \to E$$

$$\text{y} \quad \langle f^{-1} \circ \underline{prm}_E, g^{-1} \circ \underline{sgd}_E\rangle : E \to D$$

Es claro que por ser producto y composición de aproximables, los mapeos anteriores son aproximables, y también son inversos, es decir, definen el isomorfismo buscado. [].

Construiremos ahora un ejemplo que simula un producto infinito y cuyos elementos resultan sucesiones.

<u>Proposición</u> <u>II.18</u>.-Sea A un Sistema de Información, entonces el siguiente sistema es un Sistema de Información denotado por A^∞

(i) $D_{A^\infty} = \{\Delta_A\} \cup \{(\Delta_A, X) \mid X \in D_A\} \cup \{(\Delta_A, \Delta_A, X) \mid X \in D_A\} \ldots$

$= \bigcup_{n \geq 1}^{\infty} (\underbrace{\Delta_A, \Delta_A, \ldots, \Delta_A}_{n}, X) \mid X \in D_A\} \cup \{\Delta_A\}.$

(ii) $\Delta_{A^\infty} = \Delta_A$.

(iii) Si $u \subseteq D_A$ y $\#(u) < \aleph_0$ entonces

$u \in Con_{A^\infty} \iff \underline{n}(u) \in Con_A \ \forall\, n \in \mathbb{N}$

(iv) $u \in Con_{A^\infty}$

$u \vdash_{A^\infty} (\overbrace{\Delta_A, \ldots, \Delta_A}^{n}, X) \iff \underline{n}(u) \vdash_A X$

(v) $u \vdash \Delta_{A^\infty} \quad \forall\, u \in Con_{A^\infty}$.

donde $\underline{n}(u) = \{X \in D_A \mid (\underbrace{\Delta_A, \ldots, \Delta_A}_{n}, X) \in u\}$.

().

La verificación de los axiomas de Sistemas de Información para probar la proposición se sigue rutinariamente y la omitiremos. Sin embargo de mucho mayor interés resulta la:

<u>Proposición</u> <u>II.19</u>.-Los elementos de A estan en correspondencia biunivoca con las sucesuiones de elementos de A mediante

$\varphi((x_n)) = \bigcup_{i=1}^{\infty} \{(\underbrace{\Delta_A, \ldots, \Delta_A}_{i}, X) \mid X \in x_i\} \cup \{\Delta_A\}$ ().

DEMOSTRACION.-Sea (x_n) una sucesión de elementos de A primero veremos que $y=\varphi(\langle x_n\rangle)$ es un elemento al verificar la definición.

Sea $v\subseteq y$ v finito , entonces $\exists\, M\in\mathbb{N}$ 't'

$v\subseteq\bigcup_{i=1}^{M}\{(\overbrace{\Delta_A,\dots,\Delta_A}^{i},X)\mid X\in x_i\}\cup\{\Delta_A\}$ entonces $\underline{n}(v)=\emptyset$ $\forall$ $n>M$ y si $n<M+1$ $\underline{n}(v)\subseteq x_n$ y $\#(\underline{n}(v))<\aleph$ por lo que

$$\underline{n}(v)\in Con_A \quad \forall\, n\in\mathbb{N} \quad \therefore \quad v\in Con_A \,.$$

Si además $v\vdash_A R$,entonces si $R=\Delta_A$ no hay nada que probar; si $R=(\overbrace{\Delta_A,\dots,\Delta_A}^{n},X)$ entonces $\underline{n}(v)\vdash_A X$ pero como x_n es elemento de A entoncex $X\in x$ de donde $R\in y$ $\therefore$ y es elemento.

Sea ahora y un elemento de A , construiremos una sucesión de elementos de A $(\langle x_n\rangle)$ tal que $\varphi(\langle x_n\rangle)=y$.

Para cada $n\in\mathbb{N}$ sea

$$x_n=\bigcup\{\underline{n}(u)\mid u\subseteq y \text{ u finito}\}.$$

Debemos probar que x_n así definido es elemento de A.

Sea $u\subseteq x_n$ u finito , deben existir $u_1,\dots,u_k$ 't' $u_i\subseteq y$ $u\subseteq\bigcup_{i=1}^{k}\underline{n}(u_i)$ u_i finitos, llamamos $r=\bigcup_{i=1}^{k}u_i$ entonces $r\in Con_{A^\infty}$ y entonces $\underline{n}(r)\supseteq\bigcup_{i=1}^{k}\underline{n}(u_i)$ de donde $u\subseteq\underline{n}(r)$ y $\therefore$ $u\in Con_A$.

Ahora si $u\vdash_A X$ entonces $\underline{n}(r)\vdash_A X$ y como y es elemento $(\overbrace{\Delta_A,\dots,\Delta_A}^{n},X)\in y$ pero entonces $\{X\}=\underline{n}(\{(\overbrace{\Delta_A,\dots,\Delta_A}^{n},X)\})$) por lo tanto $X\in x_n$; ahora es directo que φ es una biyección.

<u>Corolario 11.20</u>.-Si A es un Sistema de Informción entonces

$$A^\infty\cong A\times A^\infty \qquad\qquad ().$$

DEMOSTRACION.-Identificamos los elementos de A^∞ con las sucesiones de elementos de A y definimos

$f:A^\infty\to A$ como $f(\langle x_n\rangle)=x_1$ y $g:A\to A$ como $g(\langle x_n\rangle)=\langle x_{n+1}\rangle$,es fácil verificar que tanto f como g provienen de mapeos aproximables y entonces $\langle f,g\rangle:A^\infty\to A\times A^\infty$.

Puede verse entonces que si definimos $h:A\times A^\infty\to A^\infty$ por

$h(x,\langle x_n\rangle)=\langle y_n\rangle$ donde $y_1=x$ $y_n=x_{n-1}$ $n>1$,
h resulta aproximable y la inversa de $\langle f,g\rangle$ lo que prueba que son isomorfismos, y concluimos $A^{\infty}\cong A\times A^{\infty}$. []

<u>Proposición 11.21</u>.-En A^{∞}, si $x=\langle x_n\rangle$ $y=\langle y_n\rangle$ son elementos, entonces $x\subseteq y$ si y sòlo si $x_n\subseteq y_n$ $\forall\, n\in\mathbb{N}$ ⟨⟩.

DEMOSTRACION.-(⇒)Supongamos $x\subseteq y$, sea n un natural arbitrario, si $X\in x_n$ entonces $(\overbrace{\Delta_A,\ldots,\Delta_A}^{n},X)\in x$ y por lo que $X\in \underline{n}(\{(\overbrace{\Delta_A,\ldots,\Delta_A}^{n},X\})\in y_n$

$$\therefore\ x_n\subseteq y_n .$$

(⇐)Sea $R=(\overbrace{\Delta_A,\ldots,\Delta_A}^{n},X)\in x$ (Si $R=\Delta_A$ trivialmente $R\in y$) entonces $\underline{n}(\{R\})\in x_n\subseteq y_n$ entonces $X\in y_n$ de donde obtenemos

$$(\overbrace{\Delta_A,\ldots,\Delta_A}^{n},X)\quad \bigcup_{i=1}^{\infty}\{(\overbrace{\Delta_A,\ldots,\Delta_A}^{i},X)\mid X\in y_i\}=y$$

$$\therefore\ x\subseteq y .\qquad []$$

De manera muy similar podemos probar que $A^{\infty}\cong A^{\infty}\times A^{\infty}$.

<u>Proposición 11.23</u>.-$\check{A}\cong A^{\infty}\times A^{\infty}$.

DEMOSTRACION.-Construimos la biyección:

Dada $\langle x_n\rangle\in A$ obtenemos $(\langle a_n\rangle,\langle b_n\rangle)$ $A^{\infty}\times A^{\infty}$ definida por
$\underline{prm}(\langle a_n\rangle,\langle b_n\rangle)=\langle a_n\rangle=\langle x_{2n-1}\rangle$ y $\langle b_n\rangle=\langle x_{2n}\rangle$
(no haremos los detalles). []

EJEMPLO.-Un pequeño lenguaje.

Trabajaremos ahora la definición de un pequeño lenguaje de programación para enfocar las construcciones presentadas hasta el momento dentro del marco de las definiciones semànticas de lenguajes de programación.

En este ejemplo, haremos uso del Dominio BOOL donde recordamos que BOOL={Verdadero, Falso, Indefinido} y $x \sqsubseteq y$ si y sólo si x=Indefinido ó x=y. Como red, BOOL se ve como

Verdadero   Falso

Indefinido

llamaremos $\mathcal{S}$ =BOOL$^{\infty}$, las sucesiones de valores de verdad donde recuérdese que $s, t \in$ BOOL$^{\infty}$, $s \sqsubseteq t$ si y sólo si $s_n \sqsubseteq t_n$ $\forall$ $n \in \mathbb{N}$.

Intuitivamente, la información sobre un valor de verdad puede ser insuficiente para determinarlo, pero con más información podemos definirlo; así mismo, una sucesión de valores de verdad esta mejor definida o la información que la especifica es más completa si alguna de sus coordenadas o entradas esta mejor definida.

Por ejemplo las siguientes sucesiones , cada vez especifican mejor una sucesión de valores de verdad(la aproximan) aunque todas son elementos del Dominio BOOL$^{\infty}$.

⟨Indefinido, Indefinido, Indefinido, Indefinido, ...⟩

⟨Verdadero, Indefinido, Indefinido, Indefinido, ...⟩

⟨Verdadero, Falso, Indefinido, Indefinido, ...⟩

La primera diferencia que podemos notar entre BOOL y $\mathcal{S}$ es que BOOL es un Dominio finito mientras que $\mathcal{S}$ es infinito, es decir, tiene un número infinito de elementos, además esto implica que algunos elementos requieren una infinidad de Información para su definición. Obsérvese como nuestras definiciones de elementos finitos se incorporan a estas ideas.

Podemos usar el orden que tenemos definido en $\mathcal{S}$ para aproximar elementos infinitos por elementos finitos. (Recuerdese que un elemento finito es el generado por un conjunto en Con) Es fàcil verificar que los elementos finitos corresponden a aquellas sucesiones que tienen un número finito de entradas distintas de Indefinido.

Ademàs, si definimos

$$\langle x_n | m\rangle = \begin{cases} x_n & \text{si } n < m+1 \\ \text{Indefinidò} & \text{si } n > m \end{cases}$$

es decir $\langle x_n | m\rangle$ es el segmento inicial de $\langle x_n\rangle$ de longitud m, resulta nuevamente directo verificar que estos elementos satisfacen:

(i) $\langle x_n | m\rangle \subseteq \langle x_n | m+1\rangle$

y por el Teorema 9.10

$$\langle x_n\rangle = \bigsqcup_{m=1}^{\infty} \langle x_n | m\rangle$$

Esto nos permite definir un lenguaje de programaciòn sobre $BOOL^{\infty}$ y discutir que operaciones son computables y como se aproximan los valores por valores finitos.

Primeramente presentamos las producciones de la sintaxis:

b ::= V | F | Cabeza s

s ::= b^* | bs | cola s | si b entonces s' sino s''

par s | impar s | mezcla s' s''

y la definiciòn de las funciones semànticas esta dada por:

$\mathbb{B}$:Clase semàntica de b -> BOOL

$\mathbb{C}$:Clase semàntica de s -> $\mathcal{S}$

(donde b denota una variable en el dominio de la funciòn $\mathbb{B}$ y s, s' y s'' son variables en el Dominio de la funciòn $\mathbb{C}$).

$\mathbb{B}$ [[V]] = Verdadero

$\mathbb{B}$ [[F]] = Falso

$\mathbb{B}$ [[cabeza s]] = $\mathbb{C}$ [[s]]$_1$

$$\mathbb{C}[[\, b^* \,]] = \langle\, \mathbb{B}[[\, b \,]] \,\rangle$$

$$\mathbb{C}[[\, bs \,]] = \langle\, \mathbb{B}[[\, b \,]],\ \mathbb{C}[[\, s \,]]_1,\ \mathbb{C}[[\, s \,]]_2, \ldots \rangle$$

$$\mathbb{C}[[\, \text{cola } s \,]] = \langle\, \mathbb{C}[[\, s \,]]_{n+1} \,\rangle$$

$$\mathbb{C}[[\, \text{Si } b \text{ entonces } s \text{ sino } s' \,]] =$$

$$\begin{cases} \mathbb{C}[[\, s \,]] & \text{si } \ \mathbb{B}[[\, b \,]] = \text{Verdadero} \\ \mathbb{C}[[\, s' \,]] & \text{si } \ \mathbb{B}[[\, b \,]] = \text{Falso} \\ \langle \text{Indefinido} \rangle & \text{si } \ \mathbb{B}[[\, b \,]] = \text{Indefinido} \end{cases}$$

$$\mathbb{C}[[\, \text{par } s \,]] = \langle\, \mathbb{C}[[\, s \,]]_2,\ \mathbb{C}[[\, s \,]]_4, \ldots \rangle$$

$$\mathbb{C}[[\, \text{impar } s \,]] = \langle\, \mathbb{C}[[\, s \,]]_1,\ \mathbb{C}[[\, s \,]]_3, \ldots, \rangle$$

$$\mathbb{C}[[\, \text{mezcla } s'\ s'' \,]] = \langle\, \mathbb{C}[[\, s' \,]],\ \mathbb{C}[[\, s'' \,]])$$

Este es el primer ejemplo donde construimos una definición de un lenguaje y aunque su justificación matemática aún no es completa, esperamos provea al lector con una idea de como se enlazan los conceptos (incluso los de computabilidad y aproximación).

Observese que contiene una copia isomorfa a Binario que corresponde a la imagen de la función (que proviene de un mapeo aproximable)

$$F(L_@) = \langle x_n \rangle \quad \text{donde} \quad \begin{cases} x_n = \text{Verdadero} & \text{si } @(1)=1 \\ x_n = \text{Falso} & \text{si } @(n)=0 \\ x_n = \text{Indefinido} & \text{en otro caso.} \end{cases}$$

Ejemplo)Definimos la siguiente función

cond: $|BOOLXAXA| \to |A|$ en los elementos, como sigue:

(i) cond(Verdadero, x, y) = x

(ii) cond(Falso, x, y) = y

(iii) cond(Indefinido, x, y) = $\perp_A$.

Afirmamos que cond proviene de un mapeo aproximable, notese que esta definición es muy similar a la función en LISP del mismo nombre.

Por la Proposición 11.11 basta probar que f proviene de un mapeo aproximable en cada uno de sus argumentos.

Sección 12 El espacio de funciones.

Como ya hemos mencionado antes, las funciones o mapeos entre Dominios son de gran importancia para las definiciones semánticas, e incluso resulta indispensable poder manejar el conjunto de mapeos entre dos Dominios para las aplicaciones en lenguajes de programación particulares.

Lo interesante de la Categoría de Dominios es que el conjunto de mapeos aproximables entre Dominios posee a su vez, una estructura de Dominio. Esto se alcanza, como ya habíamos explorado en un ejemplo, mediante la:

<u>Definición 12.1</u>.-Sean A y B Sistemas de Información. Por $A\to B$, el espacio de funciones, entendemos el sistema donde:

(i) $D_{A\to B} = \{(u,v) \mid u \in Con_A,\ v \in Con_B\}$

(ii) $\Delta_{A\to B} = (\emptyset,\emptyset)$

y donde si $n \in N$ y $w = \{(u_0,v_0),(u_1,v_1),\ldots,(u_n,v_n)\}$ tenemos

(iii) $w \in Con_{A\to B} \Leftrightarrow$

$\forall I \subseteq \{0,1,2,\ldots,n\}$ 't' $\bigcup_I u_i \in Con_A$ se tiene $\bigcup_I v_i \in Con_B$

(iv) si $w \in Con_{A\to B}$, $w \vdash_{A\to B} (u',v') \Leftrightarrow$

$\bigcup_I v_i \vdash_B v'$ donde $I = \{i \mid u' \vdash_A u_i\}$ $\diamond$.

Las ideas en que se basa la definición anterior son las siguientes:

Para determinar un mapeo aproximable $f:A\to B$ debemos decir que parejas (u,v) $u\in Con_A$ y $v\in Con_B$ estan relacionadas bajo ufv. Cada pareja proporciona una cantidad finita de información acerca de las funciones posibles que contendrían tal pareja. La pareja que menos información nos da es pues $(\emptyset,\emptyset)$ ya que toda función aproximable tiene esta pareja, así que esta pareja no nos da ninguna idea de que función particular estamos definiendo.

Debemos dar una formalización para la consistencia de parejas y la implicación de parejas. Esto es lo que hace la definición, una pareja (u,v) significa la proposición o afirmación siguiente sobre la función: si se proporciona u información a la función obtenemos cuando menos v información sobre el valor de la función.

En realidad toda pareja por si sola es consistente, pero un conjunto de parejas puede no ser consistente, por esto utilizamos (iii) en la definición que nos dice:

Si las afirmaciones en w son ciertas de al menos un mapeo aproximable se debe tener (iii), pues si consideramos un conjunto de indices I para el cual la unión de las u_i es consistente, debe ser información que puede ser suministrada conjuntamente a la función correspondiente a las afirmaciones de w, y por lo tanto el conjunto de valores posibles debe ser consistente, ya que el objeto definido debe ser un mapeo aproximable y la unión de los valores debe ser un posible resultado.

Obsérvese que si (iii) es cierto w podría extenderse al menos a una función aproximable.

(iv) Nos dice que una serie de afirmaciones sobre un mapeo aproximable en w implican una pareja (u',v') si y sólo si el conjunto formado por las segundas componentes de aquellas parejas, cuya primer componente es implicada por u', implica v', lo cual intuitivamente es claro: si un posible dato u' implica un conjunto de datos de los cuales conocemos sus resultados y estos implican un resultado v', esta pareja (u',v') debe estar en la función; ahora si (u',v') es una pareja de un mapeo aproximable, debe cumplirse el tercer axioma de mapeo aproximable y por lo tanto la relación.

La base de nuestra justificación está en la:

Proposición 12.2.-Si A y B son Sistemas de Información, entonces $A\to B$ también lo es y además los mapeos aproximables

$f:A\to B$ son exactamente los elementos de $|A\to B|$. ◇.

DEMOSTRACION.-Primero probaremos que $A\to B$ es Sistema de Información.

(i) Sea $w\in Con_{A\to B}$ y $w'\subseteq w$ entonces si $w'=\{(u_0,v_0),\dots,(u_n,v_n)\}$ y $w=\{(u_0,v_0),\dots,(u_m,v_m)\}$ con $m+1>n$ es claro que si $I\subseteq\{0,1,\dots,n\}$ y $\bigcup_I u_i\in Con_A$ en particular $I\subseteq\{0,1,\dots,m\}$ y por lo tanto $\bigcup_I v_i\in Con_B$ y se satisface el lado izquierdo de (iii) en la definición del espacio de funciones

$$\therefore\ w'\in Con_A .$$

(ii) Sea $(u_0,v_0)\in D_{A\to B}$ con $u_0\in Con_A$ $v_0\in Con_B$ entonces si $w=\{(u_0,v_0)\}$ $I=\emptyset$ ó $I=\{0\}$ y entonces $\bigcup_I u_i\in Con_A$ y $\bigcup_I v_i\in Con_B$

$$\therefore\ w\in Con_{A\to B} .$$

(iii) Supongamos $w\vdash(u',v')$ debemos demostrar que

$$w\cup\{(u',v')\}\in Con_{A\to B} .$$

Sea $w=\{(u_0,v_0),\dots,(u_{n-1},v_{n-1})\}$, si llamamos (u_n,v_n) a (u',v') y tomamos $I\subseteq\{0,1,2,\dots,n\}$

Si $I\subseteq\{0,1,\dots,n-1\}$ no hay nada que probar pues $w\in Con_{A\to B}$.

Sea I 't' $n\in I$ y $\bigcup_I u_i\in Con_A$ por la definición y como $w\vdash(u_n,v_n)$, si $L=\{j\mid u_n\vdash u_j\}$, $\bigcup_L u_j\in Con_A$ e

$$\bigcup_I u_i \vdash_A \bigcup_L u_j$$

pues $n\in I$, entonces $\bigcup_I u_i\cup\bigcup_L u_j\in Con_A$ como $w\in Con_{A\to B}$ tomando $J=I-\{n\}\cup\{j\mid u_n\vdash u_j\ j<n\}$ tenemos $\bigcup_J v_j\in Con_B$

y si $K=\{j \mid u_n \vdash u_j\ j<n\}$

$$\bigcup_J v_J \supseteq \bigcup_K v_J$$

además $\bigcup_K v_J \vdash_B v_n$ por lo que

$$\bigcup_J v_J \vdash_B v_n$$

como $\bigcup_J v_J \vdash_n v$ tenemos $\bigcup_I v_i \in Con_B$

$$\therefore\ w \cup \{(u',v')\} \in Con_{A\to B}.$$

(iv) $w \vdash_{A\to B} (\emptyset,\emptyset)$ se sigue por vacuidad.

(v) Sea $(u_J,v_J)\in w$ debemos probar que $w \vdash_{A\to B} (u_J,v_J)$ pero $u_J \vdash u_J$ de donde si $I=\{i \mid u_J \vdash u_i\}$ $v_J \in \bigcup_I v_i$

(vi) Supongamos $w' \vdash (u_i,v_i)\ \forall\ (u_i,v_i)\in w$ y $w \vdash (u',v')$ comprobaremos que $w' \vdash (u',v')$.

Si $w'=\{(u_0',v_0'),\dots,(u_k',v_k')\}$, como $w \vdash_{A\to B} (u',v')$ sea $I=\{i \mid u' \vdash u_i\}$ entonces $\bigcup_I v_i \vdash v'$
además $\forall\ i\in I$ $w' \vdash (u_i,v_i)\in w$ si llamamos $Ji=\{j \mid u_i \vdash u_J'\}$ tenemos

$$\bigcup_{Ji} v_J' \vdash_B v_i \quad \forall\ i\in I$$

por la transitividad de $\vdash_A$ como $u' \vdash u_i$ $\forall\ i\in I$

$$u' \vdash u_J' \quad \forall\ j \in \bigcup_I Ji \ \text{entonces}$$

$\bigcup_{\cup Ji} u_J' \in Con_A$ y como $w' \in Con_{A\to B}$ $\bigcup_{\cup Ji} v_J' \in Con_B$ entonces
si $J=\{j \mid u' \vdash u_J'\}$ $\bigcup_{\cup Ji} v_J' \subseteq \bigcup_J v_J'$ y por la transitividad de $\vdash_B$ y como $\bigcup_{\cup Ji} v_J' \vdash_B \bigcup_I v_i$

$$\bigcup_J v_J' \vdash_B \bigcup_I v_i \vdash v'$$

$$\therefore\ w' \vdash_{A\to B} (u',v'). \qquad [].$$

Concluimos que $A\to B$ es Sistema de Informaciôn, pues verifica los axiomas.

Ahora sea $f:A\to B$ un mapeo aproximable, probaremos que es un elemento de $|A\to B|$ (pensando f como una realciôn $(u,v) \Leftrightarrow ufv$).

(i) Sea $w \subseteq f$ w finito entonces $w=\{(u_0,v_0),\dots,(u_n,v_n)\}$ y sea

$I \subseteq \{0,1,\dots,n\}$ 't' $\bigcup_I u_i \in Con_A$

como $u_i f v_i$ $i=0,1,2,\ldots,n$ $\quad \bigcup_I u_i \vdash_A u_i \;\; \forall\, i\in I$

$\bigcup_I u_i f v_i \;\; \forall\, i\in I$, entonces como I es finito $\bigcup_I v_i \in Con_B$ y

$$\bigcup_I u_i f \bigcup_I v_i \qquad \therefore\; w\in Con_{A\to B} .$$

(ii) Si además $w \vdash_{A\to B} (u_{n+1}, v_{n+1})$ entonces si $I=\{i \mid u_{n+1} \vdash_A u_i \;\; i<n+1\}$ tenemos $\bigcup_I v_i \vdash_B v_{n+1}$ debemos probar

$$u_{n+1} f v_{n+1} ;$$

ahora, claramente $\bigcup_I u_i \in Con_A$ y como f es aproximable tenemos $u_i f v_i$ $\forall\, i\in I$ por lo que $\bigcup_I u_i f v_i$ pero $u_{n+1} \vdash_A \bigcup_I u_i$ y $\bigcup_I v_i \vdash_B v_{n+1}$ por lo que $u_{n+1} f v_{n+1}$.

$$\therefore\; (u_{n+1}\; v_{n+1})\; f$$

$\therefore$ f es elemento.

Inversamente, si f es un elemento de $|A\to B|$, afirmamos que f es aproximable.

(i) Como f es elemento $\Delta_{A\to B} \in f$, es decir $(\emptyset,\emptyset)$ f y por lo tanto

$$\emptyset f \emptyset.$$

(ii) Supongamos ufv y ufv' se cumplen, debemos probar que $uf(v\cup v')$ se cumple.

Pero si llamamos $w=\{(u,v),(u,v')\}$ f entonces $u\in Con_A$ entonces por el inciso (ii) de la definición del sistema $A\to B$ $v\cup v'\in Con_B$ y además $u \vdash_A u$ y $v\cup v' \vdash_B v\cup v'$ por lo que $w \vdash_{A\to B} (u, v\cup v')$ pero como f es elemento

$$(u, v\cup v')\in f \qquad \therefore\; uf(v\cup v').$$

(iii) Supongamos $u' \vdash_A u$ ufv y $v \vdash_B v'$, sea $w=\{(u,v)\}$ entonces $w \vdash_{A\to B} (u',v')$ pues si $u=u_1$ $v=v_1$ $I=\{\, i \mid u' \vdash u_i \,\}$ $\bigcup_I u_i = u$ $\bigcup_I v_i = v$ y como $v \vdash_B v'$ tenemos $u'fv'$.

Esto prueba que f es un mapeo aproximable por lo que la demostración de la proposición 12.2 queda concluida. [].

El nombre de mapeo aproximable para los morfismos de la Categoría que estamos describiendo se debe al hecho que observamos inicialmente, de que los elementos en los Dominios son los límites de sus aproximaciones finitas, y los mapeos aproximables forman los elementos de un Dominio particular,(el Dominio del operador ->). Dado que podemos construir explícitamente los mapeos aproximables finitos (w), al examinar las definiciones presentadas nos damos cuenta de como, de manera constructiva los mapeos aproximables se pueden aproximar por funciones simples (aquellas que toman sólo un número finito de valores). Este es el sentido preciso de aproximación que da el nombre.

El operador -> puede combinarse con los operadores + y X e incluso repetirse iterativamente; esto se realiza para construir los tipos de datos de alto nivel de las funciones semánticas. En muchas categorías el operador -> no puede utilizarse de manera constructiva, ya que la categoría no resulta cerrada bajo esta operación. Muchos autores, entre los que se incluye D. Scott consideran esta propiedad muy valiosa y por ello llaman a la Categoría Cartesianamente Cerrada; sin embargo, en nuestro concepto, basado en la Teoría de Categorías presentada en Mac Lane[72] la categoría no resulta Cartesianamente Cerrada (no existen coproductos ni objeto inicial).

Comprobaremos estas afirmaciones un poco más adelante, donde contaremos con todos los elementos necesarios, por ahora continuaremos nuestra presentación con algunos ejemplos de importancia sobre mapeos aproximables entre construcciones de Dominios.

<u>Proposición</u> <u>12.3</u>.-Existe un mapeo aproximable llamado $eval:(B\to C)XB \to C$ tal que si $g:B\to C$ y $y\in|B|$ entonces

$$eval(g,y)=g(y). \qquad \langle\rangle.$$

DEMOSTRACION.-Definiremos la relación para eval como sigue: sea $r\in Con_{(B\to C)XB}$ entonces si $w=prm(r)$ y $u'=sgd(r)$ sea $v'\in Con_C$

definimos r eval v' si y sólo si $w\vdash_{B\to C}(u',v')$.

Afirmamos que eval así definido es un mapeo aproximable:

(i)Claramente $\emptyset$ eval $\emptyset$ se cumple.

(ii)Si r eval v' y r eval v entonces

$w\vdash_{B\to C}(u',v')$ y $w\vdash_{B\to C}(u',v)$ de donde $w\vdash_{B\to C}\{(u',v'),(u',v)\}$ y ya vimos en la demostración anterior que de esto tenemos:

$$\{(u',v),(u',v')\}\vdash_{B\to C}(u',v\cup v')$$

por lo tanto $w\vdash_{B\to C}(u',v\cup v')$ por lo que (r eval $v\cup v'$) se cumple.

(iii)Supongamos $r\vdash r'$ (r' eval v') y $v'\vdash_C v$,entonces tenemos que $w'\vdash_{B\to C}(u',v')$ y $v'\vdash_C v$, también vimos en la prueba anterior que estos dos hechos nos dan:

$$w'\vdash_{B\to C}(u,v)$$

por lo tanto, como $r\vdash_{(B\to C)XB} r'$ tenemos $prm(r)\vdash_{B\to C} prm(r')$ y entonces $w\vdash w'$

$$\therefore\ w\vdash(u,v) \qquad \therefore\ r \text{ eval } v$$

y entonces se satisface la definición de mapeo aproximable.

Para terminar la demostración sea $g:B\to C$ aproximable y $y\in|B|$.

Sea $X\in g(y)=\{X\in D_C \mid ug\{X\}$ para algún $u\subseteq y$ u finito$\}$, entonces existe $u\in Con_B$ 't' u y $ug\{X\}$ y como $\{X\}\in Con_C$

sea $w=\{(u,\{X\})\}\in Con_{B\to C}$, y llamemos

$$r=\{(W,\Delta_{B\to C}) \mid W\in w\}\cup\{(\Delta_B,U) \mid U\in u\}$$

entonces por definición $r\in Con_{(B\to C)XB}$ pues

$$prm(r)=w\ \{\Delta_{B\to C}\}\in Con_{B\to C} \quad \text{y}$$

$$sgd(r)=\{\Delta_B\}\cup u\in Con_B$$

además, claramente r (g,y) pues $prm(r)\subseteq g$ y $sgd(r)\subseteq y$, por la construcción es claro que $prm(r)\vdash(u,\{X\})$

$$\therefore\ r \text{ eval } \{X\} \text{ se cumple}$$

como $eval(g,y)=\{X\in D_C \mid r \text{ eval } \{X\}$ para algún $r\subseteq(g,y)\}$

$$X\in eval(g,y)$$

$$\therefore\ g(y)\subseteq eval(g,y).$$

Ahora si $X\in eval(g,y)$ $\exists r\in Con_{(B\to C)XB}$ 't' $(r\ eval\ \{X\})$ y $r\leq(g,y)$ sea $w=prm(r)\in Con_{B\to C}$ y $u=sgd(r)\in Con_{B}$ entonces $w\vdash_{B\to C}(u,\{X\})$ pero $w=prm(r)\leq g$ y como g es elemento $(u,\{X\})\in g$ de donde $ug\{X\}$ se cumple pero esto es $X\in g(y)$ pues $u\leq y$ y concluimos que

$$eval(g,y)\subseteq g(y)$$

$$\therefore\ g(y)=eval(g,y) \qquad [].$$

El resultado anterior es de profundo interès para nuestra teorìa pues presenta las bases para considerar el operador $\to$ como una exponenciaciòn categòrica y ademàs nos dice que la operaciòn evaluaciòn de una funciòn es aproximable, lo que en otras palabras puede expresarse como que $f(x)$ puede considerarse como una expresiòn en funciòn de x y tambièn de f.

Proposiciòn 12.4.-Sean A,B,C Sistemas de Informaciòn, entonces para todo mapeo aproximable $h:AXB\to C$ existe un ùnico mapeo aproximable $curryh:A\to(B\to C)$ tal que:

$$h = eval\ o\ \langle(curryh)\ o\ \underline{prm}\ ,\ \underline{sgd}\ \rangle. \qquad \langle\rangle.$$

DEMOSTRACION.-Sea $u\in Con_A$ y $r=\{(b_0,c_0),\ldots,(b_{n-1},c_{n-1})\}$ definimos u curryh r si y sòlo si

$\forall\ I\subseteq\{0,1,2,\ldots,n-1\}$ 't' $\bigcup_I b_i\in Con_B$ se cumple

$\{(X,\Delta_B)\mid X\in u\}\cup\bigcup_I\{(\Delta_A,p)\mid p\in b_i\}\ h\ \bigcup_I c_i$.

Es rutinario probar que curryh asì definido es un mapeo aproximable, lo interesante es probar que:

$\forall\ x\in|A|$ y $\forall\ y\in|B|$ $h(x,y)=(curryh)(x)(y)$.

Lo cual termina la demostraciòn de las afirmaciones de la proposiciòn pues:

eval o $\langle curryh\ o\ \underline{prm},\underline{sgd}\rangle(x,y)$

$=eval(curryh \circ \underline{prm}(x,y), \underline{sgd}(x,y))$

$=eval(curryh(x), y)$

$=(curryh)(x)(y)$.

Sea $X \in curryh(x)(y)=\{X \in D_C \mid v(curryh)(x)\{X\}$ para algún $v \leq y\}$ entonces $\exists\, v \in Con_B$ 't' $v \leq y$ $v(curryh)(x)\{X\}$, si $v=\{Y_1,...,Y_n\}$ entonces

$\{(v,\{X\})\} \subseteq (curryh)(x)$ pero entonces $\exists\, u \leq x$ 't' $u \in Con_A$ y $u(curryh)\{(v,\{X\})\}$ pero por la definición de curryh si llamamos

$z=\{(X,\Delta_B) \mid X \in u\} \cup \{(\Delta_A, Y_i)\ i=1,..,n\}$ tenemos

$v\ h\ \{X\}$ y además $z \leq (x,y)$.

$\therefore X \in h(x,y)=\{X \in D_C \mid z\ h\ \{X\}$ para algún $z \leq (x,y)\}$

Inversamente, sea $X \in h(x,y)$ entonces sea z 't' $(z\ h\{X\})$ y $z \leq (x,y)$ $z \in Con_{AXB}$, llamamos

$u=prm(z) \leq prm((x,y)) \leq \underline{prm}(x,y)=x$

y $v=sgd(z) \leq sgd((x,y)) \leq \underline{sgd}(x,y)=y$ entonces $u \in Con_A$ y $v \in Con_B$ y trivialmente $\{X\} \in Con_C$, sea $r=\{(v,\{X\})\}$ afirmamos u curryh r, esto es directo de la definición de la relación para curryh y que

$$\{(X,\Delta_B) \mid X \in u\} \cup \bigcup_I \{(\Delta_A, p_i) \mid p_i \in v\} = z$$

y como $z\ h\ \{X\}$ resulta por $u \leq x$ y la definición de imagen que r curryh(x) pero por la construcción $v \leq y$ tenemos

$$X \in curry(x)(y).$$

La igualdad se sigue de las dos contenciones. [].

El resultado siguiente contribuye a la caracterización de nuestra categoría.

__Proposición 12.5__.-Sean A, B y C Sistemas de Información, entonces

$$|(AXB)\to C| \cong |A\to(B\to C)| \qquad ().$$

DEMOSTRACION.-Si $h:AXB \to C$ entonces $curryh:A\to(B\to C)$ y si $g:A\to(B\to C)$ entonces

$$cong=eval \circ \langle g \circ \underline{prm},\ \underline{sgd}\rangle:(A\times B)\to C$$

y es aproximable pues es la construcción y composición de mapeos aproximables. Por el resultado anterior

$$h=eval \circ \langle curryh \circ \underline{prm},\underline{gsd}\rangle \quad \therefore \ con(curryh)=h$$

ahora, tenemos que

$$eval \circ \langle g \circ \underline{prm},\underline{sgd}\rangle=cong$$

pero por la proposición para curry , también tenemos

$$eval \circ \langle curry(cong) \circ \underline{prm},\underline{sgd}\rangle=cong ,$$

por la unicidad de la función que satisface la relación anterior

$$curry(cong)=g$$

lo cual muestra la biyección que da el isomorfismo de conjuntos, pero como consecuencia de la proposición 10.11 puede verse que "con" y "curry" son isomorfismos de Dominios y por lo tanto aproximables.

[].

Este hecho, como ya mencionamos, ha motivado a muchos autores a clasificar a la Categoría de Dominios como Cartesianamente Cerrada, sin embargo, como veremos , la Categoría de Dominios no lo es en el sentido de la definición de Mac Lane [72].

Presentamos algunos ejemplo más, que corresponden a funciones (operadores) que ya hemos empleado pero no hemos descrito el marco correspondioente. Por ejemplo el mapeo

$$const: B \to (A\to B)$$

que manda cada elemento $b\in|B|$ en la función

$$const\ b:A \to B.$$

<u>Proposición 12.6</u>.-const es un mapeo aproximable.

<>.

DEMOSTRACION.-Comprobaremos la proposición 10.6, es decir que

$$\text{const } b = \bigcup\{\text{const } \langle u \rangle \mid u \subseteq b \; u \text{ finito}\}.$$

Sea $X \in \text{const } b$, entonces $X=(w,v)$ $w \in Con_A$ y $v \in Con_B$, ahora por tratarse de la función const b $v \subseteq b$, como b es elemento $v \subseteq \langle v \rangle \subseteq b$ y $v \in Con_B$

$$\therefore \; (w,v) \in \text{const } \langle v \rangle \quad \therefore X \in \bigcup\{\text{const}\langle u \rangle \mid u \subseteq b \; u \text{ finito}\}$$

Para la contención inversa, tomese

$$X=(w,v) \in \bigcup\{\text{const}\langle u \rangle \mid u \subseteq b \; u \text{ finito}\}$$

entonces existe u 't' $X \in \text{const}\langle u \rangle$ $u \in Con_B$ $u \subseteq b$, entonces $v \in Con_B$ y $X \in u$ y $\langle u \rangle \subseteq b$, entonces por definición $X \in \text{const } b$. [].

Un ejemplo más es la función que resulta de la propiedad universal del producto, podemos pensar en el operador asi:

<u>Proposición</u> <u>12.7</u>.-Sean A, B y C Sistemas de Información entonces la función

$$\langle , \rangle : (C \to A) X (C \to B) \to (C \to (AXB))$$

dada por

$$\langle , \rangle (f,g) = \langle f,g \rangle$$

es un mapeo aproximable. ().

Igualmente la función que resulta de la propiedad universal de la unión ajena nos presenta:

<u>Proposición</u> <u>12.7</u>.-Sean A, B y C Sistemas de Información, entonces la función

$$[,] : (A \to C) X (B \to C) \to (A+B) \to C$$

dada por

$$[,](f,g) = [f,g]$$

es un mapeo aproximable. ().

Y finalmente señalaremos la composición como otro mapeo aproximable. Esta Proposición dice que si llamamos $\mathbb{D}$ a la Categoría de Dominios entonces la Categoría de Dominios es una $\mathbb{D}$-Categoría (sobre si misma) en el sentido de Mac Lane[72], para algunos autores $\mathbb{D}$ es una Categoría Enriquecida.

<u>Proposición</u> <u>12.8</u>.-Sean A, B y C Sistemas de Información entonces la función

$$o:(B\to C)\times(A\to B)\to(A\to C)$$

dada por

$$o(g,f)=g\ o\ f$$

es un mapeo aproximable. ⟨⟩.

DEMOSTRACION.-Realizaremos únicamente la prueba para la composición, ya que para el producto y la unión ajena la demostración es muy similar.

Verificamos la proposición 10.6, es decir debemos comprobar que:

$$o(g,f)=\bigcup\{\ o(\langle p\rangle)\ \text{donde}\ p\subseteq(g,f)\ p\ \text{finito}\}$$

Veremos que se cumple la doble contención,

sea $X\in(g\ o\ f)$ entonces $\{X\}\in Con_{A\to C}$ $X=(u,w)$ $u\in Con_A$ $w\in Con_C$ y además como $\{X\}\subseteq g\ o\ f$ existe v 't' $v\in Con_B$ ufv vgw entonces

$$(u,v)\in f\ \text{y}\ (v,w)\in g\ ,$$

sea $p=\{((u,v),\Delta_{B\to C}),(\Delta_{A\to B},(v,w))\}$ entonces $p\subseteq(g,f)$ y claramente $X\in o(\langle p\rangle)$

$$\therefore\ X\in\bigcup\{\ o(\langle p\rangle)\ p\subseteq(g,f)\ p\ \text{finito}\}.$$

Inversamente, si $X\in\bigcup\{o(\langle p\rangle)$ donde $p\subseteq(g,f)$ p finito $\}$ entonces existe p 't' $X\in o(\langle p\rangle)$, supongamos $X=(u,w)$ $u\in Con_A$ $w\in Con_B$ como $\underline{prm}(\langle p\rangle)\subseteq g$ y $\underline{sgd}(\langle p\rangle)\subseteq f$ entonces

$\underline{prm}(\langle p\rangle):B\to C$ y $\underline{sgd}(\langle p\rangle):A\to B$ y como

$X\in o(\underline{prm}(\langle p\rangle),\underline{sgd}(\langle p\rangle))$ existe $v\in Con_B$ 't'

$v\ \underline{prm}(\langle p\rangle)\ w$ y $u\ \underline{sgd}(\langle p\rangle)\ v$

pero entonces tenemos ugw y ufv $\therefore$ $u(g \circ f)w$ es decir

$(u,w)\in g \circ f$ entonces $X\in(g \circ f)=o(g,f)$. [].

Señalaremos algunos ejemplos más de isomorfismos y funciones de los cuales omitiremos su demostración.

Proposición 12.9.-Sean A, B y C Sistemas de Información, entonces

(i) $A\to(BXC) \cong (A\to B)X(A\to C)$

bajo la biyyección dada por

si $f\ A\to(BXC)$ $\langle \underline{prm} \circ f, \underline{sgd} \circ f\rangle$ $(A\to B)X(A\to C)$

y si (h,g) $(A\to B)X(A\to C)$ entonces $\langle h,g\rangle:A\to(BXC)$.

(ii) $A\to B^{\infty} \cong (A\to B)^{\infty}$ $(A\to(N\to B) \cong N\to(A\to B)$

bajo la biyección:

Si $f:A\to B^{\infty}$ entonces $f_n:A\to B$ dada por $f_n(a)=f(a)_n$

y si $f_n:A\to B$ entonces $f:A\to B^{\infty}$ esta dada por

$$f(a)_n=f_n(a).$$

(iii) $(A+B)\to C \cong (A\to C)X(B\to C)$

donde si $[f,g]$ $(A+B)\to C$ entonces

$[f,g] \circ izq=f:A\to C$ y $[f,g] \circ der=g:B\to C$

si $C_1([f,g]) = [f,g] \circ izq$ y $C_2([f,g]) = [f,g] \circ der$

entonces

$$C_1:(A+B)\to C \to (A\to C) \text{ y } C_2:(A+B)\to C \to (B\to C)$$

por la propiedad universal del producto

$$\langle C_1,C_2\rangle:(A+B)\to C \to (A\to C)X(B\to C)$$

inversamente si (f,g) $(A\to C)X(B\to C)$ entonces

$$\underline{prm}(f,g):A\to C \text{ y } \underline{sgd}(f,g):B\to C$$

por la propiedad universal de la unión ajena

$[\underline{prm}(f,g),\underline{sgd}(f,g)]:(A+B)\to C$ entonces

$$K=[\underline{prm}(\ ),\underline{sgd}(\ )]:(A\to C)\times(B\to C)\to(A+B)\to C$$

y obsérvese que define una biyección. $\langle\rangle$.

Presentamos ahora dos morfismos de interés para las aplicaciones, especialmente en la definición de funciones semánticas. Sin embargo no son isomorfismos.

<u>Proposición 12.10</u>.-Sean A, B y C Sistemas de Información entonces

$$ind \circ \langle \underline{prm}\ ,\ const\ \bot_C \rangle : A\times(B+C) \to (A\times B)+(A\times C)$$

$$ini \circ \langle \underline{prm}\ ,\ constt\ \bot_C \rangle : A\times(B+C) \to (A\times B)+(A\times C)$$

$\langle\rangle$.

Limites

Existe otra manera de construir Dominios, la cual discutiremos muy brevemente y mostraremos algunos ejemplos. Esta forma de construcción parte de un Dominio conocido y utilizando los operadores o construcciones como $+$, $\times$ y $\to$ repite iterativamente la construcción y considera el límite. De esta metodología resultan contrucciones intuitivamente claras de Dominios complicados.

Presentamos un primer ejemplo para ilustrar este método:

Consideramos una sucesión de Dominios A_n $(n>0)$, donde A_0 es un Dominio que contiene por lo menos dos elementos (de no ser así la construcción resulta trivial), y llamamos A_{n+1} al Dominio $A_n \to A_n$, es decir al espacio de funciones de A_n.

Así, llamamos A al límite de la sucesión de Dominios y A es la construcción que se obtiene por este método. Los elementos del Dominio A pueden hacerse corresponder formalmente a sucesiones de elementos de cada uno de los Dominios A_n.

El método produce soluciones satisfactorias pero deben verificarse muchas cosas para afirmar propiedades de la construcción.

Una característica particular del ejemplo anterior y que se utiliza frecuentemente en este tipo de operadores es que pueden definirse mapeos aproximables uno a uno(inclusiones)

$$i_0 : A_0 \to A_1 , \ldots , i_{n-1} : A_{n-1} \to A_n , \ldots$$

de manera que cada Dominio A_n puede considerarse incluido en A_{n+1}, además intuitivamente de que

$$A_{n+1} = A_n \to A_n$$

podemos sospechar (y en efecto puede comprobarse) que

$$A \cong A \to A$$

Esta última expresión nos dice que A es un modelo para el Cálculo Lambda de Church, ya que un elemento de A puede considerarse como un argumento de un Dominio o como una función del Dominio en el Dominio.

Otro ejemplo similar consiste en considerar el Dominio de sucesiones de valores en BOOL como nuestro Dominio de partida y llamar

$$F_0 = \int , F_1 = F_0 \to \int , \ldots , F_n = F_{n-1} \to \int , \ldots$$

de esta manera puede probarse (ver Scott [70].) que F_n puede incluirse en F_{n+1} y además

$$F = F \to \int$$

A partir de esta relación y utilizando los isomorfismos

$(A \to B) X (A \to B) \cong A \to (BXB)$ y $A \to (B \to C) \cong (AXB) \to C$

obtenemos resultados sorprendentes como:

$F X F \cong (F \to \int) X (F \to \int) \cong F \to (\int X \int)$ y como $\int X \int \cong \int$

$F X F \cong F \to \int \cong F$

y $F \to F \cong F \to (F \to \int) \cong F X F \to \int \cong F \to \int \cong F$

Finalmente presentamos un último ejemplo de esta tipo de construcción.

Sea A un Dominio y $A^n = \overbrace{AXAXA...XA}^{n}$

llamamos $A_0 = A$, $A_1 = A_0 + A$, $A_2 = A_1 + A^2$, ...,

$A_n = A_{n-1} + A^n$, si A^+ denota el límite puede deducirse que

$$A^+ = A^+ + A^n \quad \forall\, n \in \mathbb{N}$$

y esta última expresión nos hace observar que A^+ tiene como elementos todas las listas finitas de elementos de A.

Capítulo IV

TEORIA DE DOMINIOS

Sección 13 Puntos fijos

En la descripción semántica de un lenguaje de programación, los significados de las estructuras sintácticas están dados por funciones, sin embargo los valores de estas funciones, en la mayoría de los casos prácticos, no pueden darse explícitamente. La herramienta que nos permite describir, por ejemplo, iteraciones y llamadas recursivas (estructuras de control frecuentes en lenguajes de programación) es la descripción recursiva de las funciones semánticas y sus valores, lo cual ya hemos observado en algunos ejemplos.

Las construcciones de funciones entre Dominios y las construcciones de Dominios que hemos desarrollado proporcionan una gran variedad de ejemplos de nuevos Dominios y funciones aproximables, (objetos y morfismos de nuestra Categoría). Sin embargo, una de las técnicas más productivas y más útiles en la práctica para construir funciones consiste en iterar infinitamente los operadores que hemos definido, y esto lo hacemos mediante definiciones recursivas.

El describir y utilizar definiciones recursivas con un sentido formal es una de las ideas centrales de la teoría matemárica de Scott. La definición recursiva de una función semántica o un valor que describe el significado sintáctico de una estructura de un lenguaje de programación determina una ecuación funcional, o funcional cuyos puntos fijos son las funciones que satisfacen la ecuación recursiva.

Los puntos fijos son de profundo interés para resolver ecuaciones recursivas. En nuestro caso particular elegimos el mínimo punto fijo (recuerdese que nuestros Dominios son inf--lattices y el espacio de funciones es un Dominio).

Existen muchas razones de peso para elegir este punto fijo como la definición o significado de nuestras expresiones recursivas. Las principales, como veremos, son que:

(i)Siempre existe y es único.

(ii)Las implantaciones basadas en un "stack" calculan esta solución.

(iii)El mínimo punto fijo es consistente con cualquier otro punto fijo.

La importancia de los resultados que presentaremos a continuación radica en que muestran que todo mapeo aproximable posee un único mínimo punto fijo y además la operación de obtener el mínimo punto fijo es aproximable. De esta manera podemos garantizar la solución de ecuaciones recursivas e identificar la definición con el mínimo punto fijo.

<u>Teorema 13.1</u>.-Sea A un Sistema de Información, entonces existe un único operador, "el operador mínimo punto fijo", fij, tal que:

(i) $fij:(A\to A)\to A$ es aproximable,

y $\forall$ $f:A\to A$ mapeo aproximable

(ii) $f(fij(f)) \subseteq fij(f)$

(iii) $\forall$ $x\in|A|$ si $f(x)\subseteq x$ entonces $fij(f)\subseteq x$. ◊.

DEMOSTRACION.-A pesar de ser un poco extraño, por simpleza probaremos primeramente la unicidad y después la existencia.

Supongamos existe fij' otro operador que también satisface (i), (ii) y (iii) y sea f un elemento arbitrario de |A->A|, por (i) fij y fij' serían aproximables y tendríamos

$$f(fij(f)) \subseteq fij(f) \text{ por (ii)}$$

pero aplicando (iii) para x=fij(f) tendríamos

$$fij'(f) \subseteq fij(f)$$

simétricamente $fij(f) \subseteq fij'(f)$ pero entonces

$$fij(f)=fij'(f) \quad \forall\ f:A\to A$$

por lo que son iguales, de donde fij es único.

Probaremos ahora la existencia, definimos fij:(A->A) -> A en los elementos cono sigue: Sea f:A->A entonces

$fij(f)= \bigcup \{ v \mid v \in Con_A$ y existe una lista $u_0, u_1, \ldots, u_n \in Con_A$

't'

a) $u_0 = \emptyset$

b) $u_i\, f\, u_{i+1}$, $\forall\ i < n+1$

y c) $u_n = v \}$

Debemos probar que fij(f) es elemento de |A|.

Comprobaremos la definición de elemento, para ello primeramente debemos mostrar que si $v \subseteq fij(f)$ y v es finito entonces $v \in Con_A$.

Supongamos $v=\{Y_1, \ldots, Y_n\}$ entonces existen $v_1, \ldots, v_n \in Con_A$ tales que $Y_i \in v_i$ i=1,..,n y n listas $u_{i1}, \ldots, u_{im}$ tales que satisfacen los requisitos a) , b) y c) para cada i=1,..,n.

Probaremos que $\forall\ v_i, v_j$ $i,j \in \{1,..,n\}$ se tiene $v_i \cup v_j \in Con_A$ y satisface las condiciones a), b) y c) para ser un uniendo de fij(f), después , aplicando inducción se tiene que

$$v \subseteq \bigcup_{i=1}^{n} v_i \in Con_A \,.$$

Sean v_i y v_j como hemos descrito y consideremos sus respectivas listas $u_{i1},\dots,u_{im}$ y $u_{j1},\dots,u_{jm}$ con todas las ues en Con_A; sin pérdida de generalidad podemos asumir $m_i=m_j$ ya que si alguna de las listas es menor incluimos como ues el $\emptyset$ al inicio de la lista hasta igualar m_i con m_j y las listas siguen satisfaciendo a), b) y c).

Afirmamos que la lista

$u_{i1}\cup u_{j1},\ u_{i2}\cup u_{j2},\dots,u_{im}\cup u_{jm}$ satisface a), b) y c) para $v_i\cup v_j$.

Claramente $u_{i1}\cup u_{j1}=\emptyset$ pues $u_{i1}=\emptyset=u_{j1}$

y $u_{im}\cup u_{jm}=v_i\cup v_j$ pues $u_{im}=v_i$ y $u_{jm}=v_j$

resta comprobar b) para la nueva lista pero $\forall\ k<n$ tenemos

$$u_{ik}\ f u_{i(k+1)} \quad \text{y}\ u_{jk}\ f u_{j(k+1)}$$

como $u_{i1}\cup u_{j1}=\emptyset$ para $k=1$ $u_{ik}\cup u_{jk}\in Con_A$

y si para k tenemos que $u_{ik}\cup u_{jk}\in Con_A$ entonces

$$u(ik)\cup u(jk) f u(i(k+1))\cup u(j(k+1))$$ por lo que

$$u_{i(k+1)}\cup u_{j(k+1)}\in Con_A$$ y se satisface b).

Ahora para concluir que $fij(f)\in|A|$ debemos comprobar que si además $v\vdash_A Z$ entonces $Z\in fij(f)$; pero si $v\vdash Z$ de lo realizado antes existe una lista $u_1,\dots,u_n$ que satisface a), b) y c) para que v sea uniendo, ($u_k=\bigcup_{i=1}^{n} u_{ik}$). Entonces, por (ii) y (iii) tenemos $u_{n-1}\ fv$ y como f es aproximable $u_{n-1}\ f\{Z\}$ pero entonces $\{Z\}$ es uniendo en la definición de $fij(f)$ $\therefore$ $Z\in fij(f)$.

Probaremos ahora que fij es aproximable. Por la proposición 10.11 debemos probar que:

$$fij(f)=\bigcup\{\ fij(\langle w\rangle)\ |\ w\subseteq f\ w \text{ finito}\}\ .$$

Verificamos la doble contención:

Sea $X\in fij(f)$, por la definición la función fij existe $v\in Con_A$ y $u_1,\dots,u_n\in Con_A$ tales que $X\in v$ y $u_1=\emptyset$, $u_i f u_{i+1}$ $i<n$, y $u_n=v$

Considérese $w=\{(u_1,u_2),(u_2,u_3),\dots,(u_{n-1},u_n)\}$ entonces $w\subseteq f$ y w es finito, como $X\in v$ $X\in fij(\langle w\rangle)$.

$$\therefore\ fij(f) \subseteq \bigcup \{fij(\langle w\rangle) \mid w \subseteq f\ w \text{ finito}\}$$

Si $X \in \bigcup\{\ fij(\langle w\rangle) \mid w \subseteq f\ w$ finito$\}$ entonces existe w 't' $X \in fij(\langle w\rangle)$ y por la definición de la relación para fij tenemos que existe $v \in Con_A$ y $u_1, u_2, \ldots, u_n \in Con_A$ 't' $X \in v$ $u_1 = \emptyset$, $\{(u_1, u_2), \ldots, (u_{n-1}, u_n)\} \subseteq w$ y $u_n = v$, pero f es elemento entonces $\langle w\rangle \subseteq f$ y por lo tanto $\{(u_1, u_2), \ldots, (u_{n-1}, u_n)\} \subseteq f$

$$\therefore\ u_1 f u_2, \ldots, u_{n-1} f u_n \quad \therefore\ X \in fij(f)$$

$$fij(f) \supseteq \bigcup\{\ fij(\langle w\rangle) \mid w \subseteq f\ w \text{ finito}\}$$

$$\therefore\ fij(f) = \bigcup \{fij(\langle w\rangle) \mid w \subseteq f\ \ w \text{ finito}\}$$

$$\therefore\ \text{fij es aproximable.}$$

Ahora probamos (ii) del Teorema 13.1, sea $f{:}A \rightarrow A$ un mapeo aproximable sea $X \in f(fij(f))$, entonces, por definición de imagen existe $u \in Con_A$ $u \subseteq fij(f)$ 't' $uf\{X\}$, pero ya probamos que si $u \in Con_A$ y $u \subseteq fij(f)$ existe una lista $u_1, \ldots, u_n \in Con_A$ 't' $u_1 = \emptyset$, $u_i f u_{i+1}$ $i < n$, y $u_n = u$, entonces tomando $u_{n+1} = \{X\}$ tenemos una lista para $\{X\}$ $\therefore$ $X \in fij(f)$

$$\therefore\ f(fij(f)) \subseteq fij(f)$$

(iii) Sea $x \in |A|$ 't' $f(x) \subseteq x$ y debemos comprobar que $fij(f) \subseteq x$.

Sea $X \in fij(f)$ entonces por la relación fij existe $v \in Con_A$ y $u_1, \ldots, u_n \in Con_A$ 't' $X \in v$ $u_1 = \emptyset$, $u_i f u_{i+1}$ $i < n$, y $u_n = v$ ahora, $\emptyset \subseteq x$ entonces como x es elemento $\langle \emptyset \rangle \subseteq x$, como f es monótona $f(\langle \emptyset \rangle) \subseteq f(x) \subseteq x$ entonces $u_2 \subseteq x$, procediendo por inducción , suponemos $u_i \subseteq x$,tenemos entonces $\langle u_i \rangle \subseteq x$ pues x es elemento y aplicando f

$$f(\langle u_i \rangle) \subseteq f(x) \subseteq x, \text{ y como } u_i f u_{i+1},\ u_{i+1} \subseteq f(\langle u_i \rangle) \subseteq x$$

$$\therefore\ u_{i+1} \subseteq x$$

de esta manera $u_n \subseteq x$ $\therefore$ $v \subseteq x$ $\therefore$ $X \in x$.

[].

<u>Proposición</u> <u>13.2</u>.-Sea A un Sistema de Información y $f{:}A \rightarrow A$ aproximable entonces

$f(\mathrm{fij}(f))=\mathrm{fij}(f)$. ⟨⟩.

DEMOSTRACION.-Por el inciso (iii) del Teorema anterior

$$f(\mathrm{fij}(f)) \subseteq \mathrm{fij}(f)$$

y tomando $X \in \mathrm{fij}(f)$ vemos que existe $v \in \mathrm{Con}_A$ 't' $X \in v$ y $u_1,\ldots,u_n \in \mathrm{Con}_A$ 't' $u_1=\emptyset$, $u_i\, f u_{i+1}$ $i<n$, y $u_n=v$ entonces (basta tomar la lista haxta $n-1$) y como $X \in v$ $v \vdash_A X$ y de $u_{n-1}\, f v$ concluimos $u_{n-1}\, f\{X\}$

$$\therefore \exists\, u=u_{n-1} \subseteq \mathrm{fij}(f) \text{ 't' } u \in \mathrm{Con}_A \text{ y } u f\{X\}$$

$$\therefore \quad X \in f(\mathrm{fij}(f))$$

$\therefore$ $f(\mathrm{fij}(f)) \supseteq \mathrm{fij}(f)$ y la proposición queda probada. [].

Presentamos ahora una serie de resultados generales sobre el mínimo punto fijo antes de mostrar algunos ejemplos particulares.

<u>Proposición 13.3</u>.-Sea A un Sistema de Información, entonces

$\bigsqcup_{n=1}^{\infty} f^n(\perp_A)=\mathrm{fij}(f)$ $\forall$ $f:A\to A$ aproximable. ⟨⟩.

DEMOSTRACION.-Sea f aproximable, en consecuencia es monótona y por lo tanto

$\perp_A \subseteq f(\perp_A)$ implica $f(\perp_A)\subseteq f^2(\perp_A)$ y por inducción $\forall$ $n\in\mathbb{N}$ se tiene $f^n(\perp_A)\subseteq f^{n+1}(\perp_A)$,entonces por el Teorema 8.7

$\bigsqcup_{n=1}^{\infty} f^n(\perp_A)$ está bien definido, pero f es aproximable entonces

$$f\Big(\bigsqcup_{n=1}^{\infty} f^n(\perp_A)\Big)=\bigsqcup_{n=1}^{\infty} f^{n+1}(\perp_A)=\bigsqcup_{n=1}^{\infty} f^n(\perp_A)$$

por lo que $\bigsqcup_{n=1}^{\infty} f^n(\perp A)$ es un punto fijo.

Si c es un punto fijo entonces $\perp_A \subseteq c$ y entonces $f(\perp_A)\subseteq f(c)=c$ y por inducción $f^n(\perp_A)\subseteq c$ $\forall$ $n\in\mathbb{N}$, como c es elemento , c es cota superior de $\{f^n(\perp_A)\ n\in\mathbb{N}\}$

$$\therefore \quad \bigcup_{n=1}^{\infty} f^n(\perp_A)\subseteq c$$

$$\therefore \quad \bigsqcup_{n=1}^{\infty} f^n(\perp_A) = \mathrm{fij}(f). \ [].$$

Teorema 13.4.-(Principo de Inducción para puntos fijos). Sea A un Sistema de Información y supongamos $f:A\to A$ es aproximable, si $S\subseteq|A|$ es un conjunto tal que:

(i) $\perp_A \in S$

(ii) $x\in S$ siempre implica $f(x)\in S$

(iii) si $\langle x_n\rangle \in S$ y $x_n \subseteq x_{n+1}$ $\forall$ n implica $\bigcup_{n=1}^{\infty} x_n \in S$

entonces:

$$fij(f)\in S. \qquad \langle\rangle.$$

DEMOSTRACION.- $\perp_A \in S$, entonces es fácil comprobar por inducción que (ii) implica $f^n(\perp_A)\in S$ $\forall$ $n\in\mathbb{N}$ y además

$f^n(\perp_A) \subseteq f^{n+1}(\perp_A)$ pues f es aproximable y por lo tanto monótona, tomando

$$x_n = f^n(\perp_A) \text{ tenemos}$$

$$fij(f)=\bigcup_{n=1}^{\infty} f^n(\perp_A) = \bigcup_{n=1}^{\infty} x_n \in S. \quad [].$$

Corolario 13.5.-Si $f,g,h:A\to A$ son aproximables y $h(\perp_A)=g(\perp_A)$ con $h\circ f=f\circ h$, y $f\circ g=g\circ f$, entonces

$$fij(f)\in \{x\in|A| \mid h(x)=g(x)\}. \qquad \langle\rangle.$$

DEMOSTRACION.-Por la proposición anterior, tomando

$S=\{x\in|A| \mid h(x)=g(x)\}$ observamos que:

(i) $\perp_A \in S$

(ii) si $h(x)=g(x)$

$(f\circ h)(x)=f(h(x))=f(g(x))$, entonces $h(f(x))=g(f(x))$

(iii) Sea $\langle x_n\rangle$ 't' $g(x_n)=h(x_n)$ $\forall$ n y $x_n\subseteq x_{n+1}$ entonces por ser g y h aproximables

$$g(\bigcup_{n=1}^{\infty} x_n)= g(\bigcup_{n=1}^{\infty} x_n)=\bigcup_{n=1}^{\infty} g(x_n)=\bigcup_{n=1}^{\infty} h(x_n)=h(\bigcup_{n=1}^{\infty} x_n)=h(\bigcup_{n=1}^{\infty} x_n)$$

$\therefore \bigcup_{n=1}^{\infty} x_n \in S.$ [].

<u>Teorema</u> <u>13.6</u>.—Para todo Sistema de Información A existe una función aproximable

$\varphi: ((A\to A)\to A) \to ((A\to A)\to A)$ dada por

si $\theta:(A\to A)\to A$ y $f:A\to A$ entonces

$\varphi(\theta):(A\to A)\to A$ y $\varphi(\theta)(f)=f(\theta(f))$,

además $fij:(A\to A)\to A$ es el mínimo punto fijo de φ. ◇.

DEMOSTRACION.—Como φ está dada en los elementos debemos probar que: $\varphi(\theta)=\bigcup\{\varphi(\langle w\rangle)$ donde $w\subseteq\theta$ y w finito$\}$.

Verificamos la doble contención de los conjuntos,

Sea $X\in\varphi(\theta)(f)$,entonces $X\in f(\theta(f))$ entonces existe $u\subseteq\theta(f)$ 't' $uf\{X\}$ y $u\in Con_A$ pero entonces existe $r\subseteq f$ $r\in Con_{A\to A}$ 't' $r\theta u$ llamaremos $w=\{(r,u)\}\subseteq\theta$ y afirmamos $X\in\varphi(\langle w\rangle)(f)=f(\langle w\rangle(f))$ pues $uf\{X\}$ y $u\subseteq\langle w\rangle(f)$ ya que $(r,u)\in w\subseteq\langle w\rangle$ y $r\subseteq f$.

Inversamente si $X\in\bigcup\{\varphi(\langle w\rangle)$ $w\subseteq\theta$ w finito$\}(f)$ entonces existe w 't' $X\in\varphi(\langle w\rangle)(f)$ y $w\subseteq\theta$ $w\in Con_{(A\to A)\to A}$ entonces $X\in f(\langle w\rangle(f))$ y entonces existe $u\subseteq\langle w\rangle(f)$ $u\in Con_A$ 't' $uf\{X\}$ pero $\langle w\rangle\subseteq\theta$ pues θ es elemento entonces

$$\langle w\rangle(f)\subseteq\theta(f) \quad \therefore\ u\subseteq\theta(f)$$

de donde $X\in f(\theta(f))$ es decir $X\in\varphi(\theta)(f)$.

Veremos ahora que fij es punto fijo de φ , es decir

$$\varphi(fij)=fij;$$

pero $\forall\ f:A\to A$

$$\varphi(fij)(f)=f(fij(f))=fij(f)$$

y es el mínimo pues si fjj es también otro punto fijo de φ entonces

$$\varphi(fjj)=fjj$$

es decir $\forall\ f:A\to A$

$$\varphi(fjj)(f)=fjj(f)$$

entonces $f(fjj(f))=fjj(f)$ es decir $fjj(f)$ es punto fijo de f pero esto implica $fij(f) \subseteq fjj(f)$ por la definición de mínimo punto fijo, pero esto es $\forall\ f:A\to A$ entonces $fij \subseteq fjj$. [].

Teorema 13.7.-Sea A un Sistema de Información y $a\in|A|$ entonces:

(i)Existe un Dominio $|A_a| = \{x\in|A| \mid x\subseteq a\}$.

(ii)Si a es un punto fijo, todo mapeo aproximable $f:A\to A$ puede ser restringido a un mapeo aproximable $f':A_a \to A_a$ donde

$$f'(x)=f(x)\ \forall\ x\in|A_a|$$

(iii)f' tiene un solo punto fijo en $|A_{fij(f)}|$ y es $fij(f)$.

(iv)Si a es un punto fijo, entonces todo punto fijo de f' en $|A_a|$ esta contenido en a. ().

DEMOSTRACION.-(i)El Dominio buscado es fácil de construir bajo:

$D_{Aa} = \{X\in D_A \mid X\in a\}$;

$u\in Con_{Aa} \iff u\in Con_A$ y $(u)\subseteq a$

$\Delta_{Aa} = \Delta_A$

$\vdash_{Aa} = \vdash_A$.

Es rutinario verificar los axiomas de Sistema de Información, sin embargo supongamos $x\in|A_a|$, debemos probar que $x\subseteq a$,pero si $X\in x$, entonces $X\in D_{Aa}$ de donde $X\in a$.

(ii)Definimos $f'(x)=f(x)\ \forall\ x\in|A_a|$ entonces como $x\subseteq a$, $f(x)\subseteq f(a)=a$ y como está definida en términos de un mapeo aproximable , es directo que f' es aproximable.

(iii)Si $x,y\in|A_a|$ son puntos fijos de f' entonces $x,y\subseteq fij(f)$ pero $x=f'(x)=f(x)$ y $y=f'(y)=f(y)$ entonces como $fij(f)$ es el mínimo punto fijo de f $x,y\supseteq fij(f)$ de donde

$$x=y=fij(f).$$

(iv)Esto es directo de la definición de $|A_a|$. [].

Teorema 13.8.-El operador mínimo punto fijo está unívocamente determinado por las siguientes condiciones:

(i) $\mathrm{Fij}_A ; (A \to A) \to A$ $\forall$ Sistema de Información A.

(ii) $\mathrm{Fij}_A(f)=f(\mathrm{Fij}_A(f))$ $\forall f:A \to A$.

(iii) $f:A \to A$, $g:B \to B$, $h:A \to B$ y $h \circ f=g \circ h$ y $h(\perp_A)=\perp_B$ implican $h(\mathrm{Fij}_A(f))=\mathrm{Fij}_B(g)$. ().

DEMOSTRACION.-Si Fij_A es el operador mínimo punto fijo para el Sistema de Información A, es decir, $\mathrm{Fij}_A=\mathrm{fij}$ considerado en A, entonces ya vimos que satisface las condiciones (i) y (ii) debemos comprobar satisface (iii).

Si $h(\perp_A)=\perp_B$ entonces $g(h(\perp_A))=g(\perp_B)$ pero $g \circ h=h \circ f$ de donde $h(f(\perp_A))=g(\perp_B)$ aplicando g a esta última relación tenemos

$$g(h(f(\perp_A)))=g^2(\perp_B)$$

por la relación $g \circ h=h \circ f$ tenemos

$$h(f^2(\perp_A))=g^2(\perp_B)$$

y por un argumento de inducción tenemos

$$h(f^n(\perp_A))=g^n(\perp_B)$$

com h es aproximable

$$\mathrm{Fij}_B(g)=\bigsqcup_{n=1}^{\infty} g^n(\perp_B)=\bigsqcup_{n=1}^{\infty} h(f^n(\perp_A))=h(\mathrm{Fij}_A(f)).$$

. Inversamente, si Fij_A satisface las condiciones señaladas en el Teorema 13.8 debemos comprobar que corresponde al mínimo punto fijo, pero nuevamente lo único que resta comprobar es que si $x \in |A|$ y $f(x)=x$ entonces $\mathrm{Fij}_A(f) \sqsubseteq x$.

Sea $h:A_x \to A$. dado por $h(x)=x$ (h es la inclusión) entonces si llamamos $f'=f\big|_{A_x}$ observamos que

$$h(\perp_{A_x})=\perp_{A_x}=\perp_A \text{ y}$$

$h \circ f'=f \circ h$

pues $\forall\, y \in |A_x|$ $h(f'(y))=h(f(y))=f(y)$ y $f(h(y))=f(y)$

aplicando el resultado anterior tenemos que como

$h(Fij_{Ax}(f'))=Fij_A(f)$ y $Fij_{Ax}(f')$ es punto fijo de f' $Fij_{Ax}(f')\leq x$ entonces $Fij_A(f)\leq x$. [].

<u>Proposición</u> <u>13.9</u>.-Sean $f,g:A\to A$ aproximables, entonces si $g(\perp)=\perp$ ó $f(\perp)=\perp$ se tiene

$$fij(f \circ g)=f(fij(g \circ f)). \qquad ().$$

DEMOSTRACION.-Si $f(\perp)=\perp$ tomando $h=f$ en el Teorema anterior y

$$f_1=f \circ g,\ f_2=g \circ f \text{ entonces}$$

$$f_1 \circ h=f \circ g \circ f \quad y\ h \circ f_2=f \circ g \circ f$$

entonces $fij(f \circ g)=f(fij(g \circ f))$

Si $f(\perp)\neq\perp$ entonces $g(\perp)=\perp$ y

$$fij(f \circ g)=\bigsqcup_{n=1}^{\infty}(f \circ g)^n(\perp)=\bigsqcup_{n=1}^{\infty}f(g \circ f)^{n-1}(g(\perp))$$

y como f es aproximable esto es

$$f(\bigsqcup_{n=1}^{\infty}(g \circ f)^{n-1}(\perp)=f(fij(g \circ f)). \qquad [].$$

Presentamos ahora algunos ejemplos.

El ejemplo más común de una definición recursiva o ecuación funcional, es la función factorial, cuya definición está dada por

$$f(n)=\begin{cases} 1 & \text{si } n=0 \\ n(f(n-1)) & \text{en otro caso.} \end{cases}$$

Este ejemplo ilustra la forma general de una definición recursiva dada por:

$$f(x) \Leftarrow \tau(f)(x),$$

y eso significa que $f=\tau(f)$, es decir que f es un punto fijo de τ.Como ya mencionamos, el problema es a cual punto fijo se refiere la definición.

Por las razones ya mencionadas, elegimos como el significado de una definición recursiva al mínimo punto fijo de la funcional.

Sin embargo esta elección del mínimo punto fijo como significado de las definiciones recursivas puede parecer arbitraria. Para algunos autores, es preferible la elección de óptimo punto fijo (aquel punto fijo que es el máximo punto fijo con la propiedad de ser consistente con todos los puntos fijos). La idea central para elegir el óptimo punto fijo como significado consiste en proporcionar la máxima solución posible a partir de la definición recursiva sin caer en posibles inconsistecias o ambiguedades.

Debido a serios inconvenientes, como que no siempre el óptimo punto fijo es computable nos concentraremos en considerar únicamente el mínimo punto fijo y remarcamos sus ventajas, comprobadas en los resultados anteriores, entre las más importantes, que siempre existe, es único y es aproximable. Pára una mayor discusión sobre el óptimo punto fijo recomendamos ver Bracho[79].

Ejemplo 1) Definimos $f:A\to B$ recursivamente por

$$f(x) \Leftarrow f(x)$$

entonces $\tau:(A\to B)\to(A\to B)$, $\tau=I_{A\to B}$

de donde concluimos $f=\mathrm{fij}(\tau)=\perp_{A\to B}=\mathrm{const}\ \perp_B$.

Ejemplo 2)En los ejemplos de la sección 7 vimos Dominios N cuyos elementos totales corresponden a los enteros no negativos y aplicando el operador ∞ tenemos el Dominio N^∞ cuyos elementos son sucesiones en N.

Es importante señalar que N^∞ no es isomorfo a $N\to N$; como veremos a todo elemento de N^∞ o sucesión de elementos de N le corresponde al menos un mapeo aproximable en $N\to N$ pero inversamente, todo mapeo aproximable en $N\to N$ define una sucesión en N^∞, sin embargo diferentes mapeos aproximables en $N\to N$ pueden definir la misma sucesión ya que pueden coincidir en los elementos totales y diferir en los parciales. Es importante indicar que si F es el Dominio de funciones (parciales o totales) de los naturales en lo naturales, donde funciòn quiere decir morfismo de la Carегorìa de Conjuntos, ver ejemplo 4 secciòn 7 entonces F es isomorfo a Tot_N, no haremos los detalles.

Definimos las siguiente funciòn aproximable (el lector puede verificar que proviene de un mapeo aproximable) .

val:FXN -> N dada en los elementos por

$$val(\pi,n)=\begin{cases}\pi(n) & \text{si } \pi \text{ està definida en } n\\ \bot_N & \text{en otro caso.}\end{cases}$$

Puede igualmente trabajarse con N^∞

Es fàcila verificar que

$$curry(val):F \to (N\to N)$$

es una funciòn uno a uno en los elementos.

Definimos τ :F->F por

$$\tau(\pi)(n)=\begin{cases}0 & \text{si } n=0\\ \pi(n-1)+n-1 & \text{si } n>0\end{cases}$$

Algunas observaciones que el lector puede verificar son:

(i)si π es total en F entonces $\tau(\pi)$ es total en F

(ii)Si π es parcial, entonces $\tau(\pi)$ es parcial (ya que si π no esta definido en k, $\tau(\pi)$ no està definido k+1)

(iii)$\tau(\pi)$ siempre està definida en 0.

(iv)Ademàs es aproximable pues

$\tau(\pi)=\bigcup\{\tau(\langle\ell\rangle)$ dodne $\ell\leq\pi$ y ℓ finita $\}$

(esto ùltimo se verifica mostrando que $\forall$ elemento $n\in N$

$\tau(\pi)(n)=(\bigcup\{\tau(\langle\ell\rangle)\mid \ell\leq\pi$ y π finito $\})(n)$

Por el Teorema 13.1 tenemos que fij(τ) es el mìnimo punto fijo y nos preguntamos ¿ quièn es f si f=fij(π)? .

Tenemos f(0)=0 y f(n+1)=(f)(n+1)=f(n)+n y por un argumento de inducciòn se prueba que

$$f(n)=\sum_{i<n} i$$

Ejemplo 4) Sea T el Dominio del ejemplo 2) de la secciòn 8 cuyos elementos totales son los naturales y sean:

$$\text{pred}, \text{succ}: N \to N$$

dados por las relaciones

$u \text{ succ } v \iff n \in N$ 't' $n \in (a,b) \ \forall \ (a,b) \in u$

$\qquad\qquad\qquad\qquad$ y $n+1 \in (c,d) \ \forall \ (c,d) \in v.$

$u \text{ pred } v \iff n \in N$ 't' $n+1 \in (a,b) \ \forall (a,b) \in u$

$\qquad\qquad\qquad\qquad$ y $n \in (c,d) \ \forall \ (c,d) \in v.$

Es nuevamente directo verificar que pred y succ son aproximables y ademàs satisfacen

$\text{succ}(n)=n+1$

y $\quad \text{pred}(n)=\begin{cases} n-1 & \text{si } n>0 \\ \perp & \text{en otro caso.} \end{cases}$

Ademàs, la funciòn

$$\text{cero}(n) = \begin{cases} \text{Verdadero} & \text{si } n=0 \\ \text{Falso} & \text{si } n\neq 0 \text{ y } n\neq\perp \\ \perp & \text{si } n=\perp \end{cases}$$

proviene de un mapeo aproximable $\text{cero}: N \to \text{BOOL}$ y podemos entonces definir:

$+: N \times N \to N$ como

sea $\rho : (N \times N) \to N \to (N \times N) \to N$ dada por,

dada $f: N \times N \to N$

$\rho(f)(n,m)=\text{cond}(\text{cero}(m), n, f(\text{succ}(n), \text{pred}(m)))$

entonces

$$+=\text{fij}(\rho)$$

y si definimos $\tau: (N \to N) \to (N \to N)$ por, dada $f: N \to N$

$\tau(f)(n)=\text{cond}(\text{cero}(n), 0, f(\text{pred}(n)) + \text{pred}(n))$

entonces por lo visto en el ejemplo anterior

$$\sum_{i<n} i=\text{fij}(\tau)(n).$$

En los ejemplos anteriores, es fàcil ver que τ y ζ son aproximables, pues son la composiciòn de mapeos aproximables. El lector puede verificar que + corresponde a la suma de naturales.

Secciòn 14 Consideraciones Categòricas.

Mostramos una subcategorìa de los Dominios de gran interès y utilidad asì como resultados que fundamentan las definiciones recursivas de Dominios.

<u>Definiciòn 14.1</u>.-Sea $f:A\to B$ un mapeo aproximable, entonces f es estricto, notaciòn $f:A\xrightarrow[s]{}B$, si y sòlo si

$$f(\bot_A)=\bot_B \qquad ().$$

Los Dominios como objetos, y los mapeos aproximables estrictos forman una Categorìa ya que $I_A(\bot_A)=\bot_A$ $\forall$ Dominio A y claramente la composiciòn de mapeos aproximables estrictos es estricta. Esta subcategorìa de la Categorìa de Dominios recibe el nombre de subcategorìa estricta, o la Categorìa $\mathbb{D}$-estricta, o Categorìa de Dominios Estricta.

Como puede comprobarse directamente el objeto terminal en la categorìa $\mathbb{D}$-estricta es $\mathbb{I}$ ya que todo mapeo que llega a $\mathbb{I}$ es estricto, ademàs el producto de dos mapeos aproximables estrictos es estricto pues

$$\langle f,g\rangle(\bot_C)=(f(\bot_C),g(\bot_C))=(\bot_A,\bot_B)=\bot_{AXB}$$

por lo que la Categorìa estricta tiene productos finitos en el sentido categòrico.

La uniòn ajena, es efectivamente el coproducto categòrico en la Categorìa estricta(recuerdese que la uniòn ajena no es coproducto en la Categoría de Dominios pero esto nos muestra que está muy cerca de serlo).

. En la Categoría $\mathbb{D}$-estricta $\mathbb{I}$ es el objeto inicial, pues $\forall$ Dominio $|A|$, $const\perp_A : \mathbb{I} \to A$ es estricto y si $f:\mathbb{I} \to A$ es otro mapeo estricto entonces $f(\perp_{\mathbb{I}})=\perp_A$ y esto nos dice que

$$f(x)=const\perp_A \quad \forall \text{ x elemento de } \mathbb{I}$$

entonces $f=const\perp_A$.

Por este último hecho la Categoría $\mathbb{D}$-estricta tiene coproductos finitos, y el objeto inicial coincide con el terminal, es decir existe el objeto nulo, una conclusión de esto es que la Categoría $\mathbb{D}$-estricta no es Cartesianamente Cerrada en el sentido de MacLane[72].

Sin embargo, algunos autores la llaman Cartesianamente Cerrada pues la colección de mapeos aproximables estrictos forman también un Dominio bajo la

<u>Definición</u> <u>14.2</u>.-Sean A y B Sistemas de Información, el espacio de funciones estrictas, denotado $A_{\overline{s}}\rangle B$, es el sistema donde:

(i) $D_{A_{\overline{s}}\rangle B} = \{(u,v) \mid u \in Con_A \text{ y } v \in Con_B\}$

(ii) $\Delta_{A_{\overline{s}}\rangle B} = (\emptyset, \emptyset)$

y si $n \in \mathbb{N}$ y $w=\{(u_o,v_o),\ldots,(u_{n-1},v_{n-1})\}$ tenemos

(iii) $w \in Con_{A_{\overline{s}}\rangle B} \Leftrightarrow$ si $I \subseteq \{0,1,2,\ldots,n-1\}$ y $\bigcup_I u_i \in Con_A$ entonces $\bigcup_I v_i \in Con_B$ y si $\emptyset \vdash_A \bigcup_I u_i$ entonces $\emptyset \vdash_B \bigcup_I v_i$

(iv) $w \vdash_{A_{\overline{s}}\rangle B} (u',v') \Leftrightarrow I=\{i \mid u' \vdash_A u_i\}$ entonces $v_{\langle\rangle} \vdash_B v'$.

No probaremos el siguiente:

<u>Teorema</u> <u>14.4</u>.-Si A y B son Sistemas de Información entonces $A_{\overline{s}}\rangle B$

es Sistema de Información y los mapeos aproximables estricos son exactamente los elementos de $|A_{\bar{\jmath}} \rangle B|$. ().

Un resultado interesante, que muestra un funtor en sentido opuesto a la inclusión de la Categoría $\mathbb{D}$-estricta en la Categoría de Dominios es:

<u>Proposición</u> <u>14.4</u>.-Sean A y B dos Sistemas de Información, entonces Estricto:(A->B) -> $(A_{\bar{\jmath}} \rangle B)$ definido para todo f:A->B por

$$u \text{ Estricto}(f)\ v \iff \emptyset \vdash_B v \quad ó \quad \emptyset \nvdash_B v, \emptyset \nvdash_A u \text{ y } ufv \quad u \in Con_A \text{ y } v \in Con_B$$

es tal que Estricto(f) es el máyor mapeo aproximable estricto contenido en f. ().

DEMOSTRACION.-Probaremos primeramente que la relación que aparece en la proposición es un mapeo aproximable:

(i)$\emptyset$ Estricto(f) $\emptyset$ se cumple pues $\emptyset \vdash_B \emptyset$.

(ii)Supongamos uEstricto(f)v y uEstrifto(f)v'

si $\emptyset \vdash_B v$ entonces si $\emptyset \vdash_B v'$ inmediatamente $\emptyset \vdash_B v \cup v'$ entonces uEstricto(f)$v \cup v'$;

si $\emptyset \nvdash_B v'$ tenemos ufv' y como

$v' \vdash_B \emptyset$, $v' \vdash_B v$, por ser f aproximable tenemos

$uf(v \cup v')$ entonces uEstricto$(v \cup v')$.

Los otros casos se siguen similarmente.

(iii)Supongamos $u \vdash_A u'$ u'Estricto(f)v' y $v' \vdash_B v$,

si $\emptyset \vdash_B v$ entonces directamente uEstricto(f)v;

si $\emptyset \nvdash_B v$ entonces $\emptyset \nvdash v'$ y por definición u'fv' y $\emptyset \nvdash_A u'$ entonces $\emptyset \nvdash u$ (si $\emptyset \vdash u$ como $u \vdash u'$ $\emptyset \vdash u'$!!!) y como f es aproximable tenemos ufv , pero esto es uEstricto(f)v.

Lo anterior prueba que la relación uEstricto(f)v define un mapeo aproximable.

Para ver que es estricto observamos que:

$\text{Estricto}(f)(\perp_A)=\{X\in D_B \mid u\text{Estricto}(f)\{X\}$ para $u\subseteq\perp_A$ u finito$\}$

sea $X\in\text{Estricto}(f)(\perp_A)$ entonces $\exists\, u\subseteq\perp_A$ 't' $u\text{Estricto}(f)\{X\}$, pero si $u\subseteq\perp_A$ entonces $\emptyset\vdash_A u$ entonces por la definición de Estricto debe cumplirse que $\emptyset\vdash_B \{X\}$ entonces $X\in\perp_B$.

$$\therefore\ \text{Estricto}(f)(\perp_A)\subseteq\perp_B.$$

Inversamente, sea $X\in\perp_B$ entonces $\emptyset\vdash X$ de donde $u\text{Estricto}(f)\{X\}\ \forall\ u\in Con_A$, en particular

$\emptyset\text{Estricto}(f)\{X\}$ entonces $X\in\text{Estricto}(f)(\perp_A)$

$$\therefore\ \perp_B=\text{Estricto}(f)(\perp_A).$$

(En realidad puede concluirse directamente de

$$\text{Estricto}(f)(\perp_A)\subseteq\perp_B$$

la igualdad pues se trata la imagen de un elemento bajo un mapeo aproximable, entonces $\text{Estricto}(f)(\perp_A)$ es elemento).

Sea $(u,v)\in\text{Estrico}(f)$ debemos probar que $(u,v)\in f$

si $\emptyset\vdash_B v$ entonces $uf\emptyset$ se cumple pues f es aproximable y en consecuencia, también ufv, $\therefore\ (u,v)\in f$.

Si $\emptyset\nvdash_B v$ entonces $\emptyset\nvdash_A u$ (si $\emptyset\vdash_A u$ entonces $u\subseteq\perp_A$ entonces $\text{Estricto}(f)(\langle u\rangle)\subseteq\perp_B$ entonces $v\subseteq\perp_B$, $\emptyset\vdash_B v$!)
entonces ufv y $(u,v)\in f$.

Sea ahora $f'\subseteq f$ y f' aproximable estrico, debemos probar que $f'\subseteq\text{Estricto}(f)$.

Sea $(u,v)\in f'$ $u\in Con_A$ y $v\in Con_B$

si $\emptyset\vdash_B v$ entonces $(u,v)\in\text{Estricto}(f)$

si $\emptyset\nvdash_B v$ entonces $\emptyset\nvdash_A u$ (ya que si $\emptyset\vdash u$ $\langle u\rangle\subseteq\perp_A$ y como f' es aproximable $f'(\langle u\rangle)\subseteq f'(\perp_A)=\perp_B$ entonces $v\subseteq\perp_B$, $\emptyset\vdash v$!) .

Como $f'\subseteq f$ entonces $(u,v)\in f$, pero entonces ufv se cumple y esto nos dice que uEstricto$(f)v$, es decir $(u,v)\in$Estricto(f)

$\therefore$ $f' \subseteq$ Estricto(f).

Para probar que Estricto es aproximable probaremos que:

Estrifto$(f)=\bigcup\{$Estricto$(\langle w\rangle) \mid w\subseteq f$ w finito$\}$.

Para toda $w\subseteq f$, $\langle w\rangle\subseteq f$ pues f es elemento, entonces

Estricto$(\langle w\rangle)\subseteq\langle w\rangle\subseteq f$

$\therefore$ Estricto$(\langle w\rangle)\subseteq$Estricto$(f)$ $\forall$ $w\subseteq f$

ya que Estricto(f) es el mayor mapeo aproximable estricto contenido en f; entonces

Estricto$(f)\supseteq\bigcup\{$Estricto$(\langle w\rangle)$ $w\subseteq f$ w finito$\}$

Sea $(u,v)\in$ Estricto(f) entonces $(u,v)\in f$ sea $w=\{(u,v)\}$ afirmamos $(u,v)\in$ Estricto$(\langle w\rangle)$

Si $\emptyset\not\vdash_B v$ como $(u,v)\in$ Estricto(f) ya se vió que entonces $\emptyset\not\vdash_A u$ pero $(u,v)\in w$ entonces $u\langle w\rangle v$ de donde $(u,v)\in$ Estricto$(\langle w\rangle)$.

Si $\emptyset\vdash v$ entonces $(u,v)\in$ Estricto$(\langle w\rangle)$.

$\therefore$ Estricto$(f)\subseteq\bigcup\{$Estricto$(\langle w\rangle)$ $w\subseteq f$ w finito$\}$

$\therefore$ Estricto$(f)=\bigcup\{$Estricto$(\langle w\rangle)$ $w\subseteq f$ w finito$\}$

$\therefore$ Estricto es aproximable ◊.

Una pequeña aplicación de los mapeos aproximables estrictos aparece en el siguiente resultado:

<u>Proposición 14.5</u>.- $A\times A\cong(\text{BOOL}\to A)$ $\forall$ Sistema de Información A.

DEMOSTRACION.-El isomorfismo esta dado por el mapeo aproximable estrico , cond , dado por

$x,y\in|A|$ y $t\in|\text{BOOL}|$

$$\text{cond}(x,y)(t)=\begin{cases} x & \text{si } t=\text{Verdadero} \\ y & \text{si } t=\text{Falso} \\ \perp_A & \text{si } t=\text{Indefinido}. \end{cases}$$ [].

Para finalizar esta sección mostraremos que la Categoría de Dominios NO tiene objeto inicial, esto se debe a que, de todo Dominio |A| en todo Dominio |B|, existe al menos la función aproximable $const\bot_B : A \to B$.

Si $\emptyset$ fuese el objeto inicial, entonces $const\bot_A : \emptyset \to A$ debe ser el único morfismo que existe de $\emptyset$ en A, pero esto indicaría que $\emptyset$ es objeto inicial de la Categoría $\mathbb{D}$-estricta, es decir isomorfo a $\mathbb{I}$ pero ya sabemos que $\mathbb{I}$ no es inicial en la Categoría de Dominios.

Sección 15 Dominios Reflexivos

Una de las razones principales que motivan la Teoría de Dominios es que ésta proporciona una noción de computabilidad incorporando elementos finitos e infinitos. En los ejemplos que hemos presentado hemos visto como funciones (en particular operadores o funcionales) pueden ser definidos en Dominios y debido a sus facultades para ser continuas y aproximables hemos mostrado como pueden ser calculadas por aproximaciones finitas. (Recuerdese que todo elemento es el límite de elementos finitos y los morfismos en la Categoría de Dominios son a su vez elementos de Dominios particulares).

Sin embargo, una razón extremadamente importante para el desarrollo de la teoría radica en la justificación de definiciones recursivas de Dominios, es decir, utilizando las construcciones +, X y -> escribimos una ecuación recursiva y pensamos en el Dominio que la satisface como el Dominio definido por la ecuación.

Los Dominios definidos recursivamente reciben el nombre de Dominios Reflexivos porque, en la mayoría de los casos, estos Dominios contienen copias de si mismos como parte de su estructura. La manera en que esta contención se presenta es mediante el isomorfismo planteado en la ecuación recursiva o Ecuación de Dominios, este isomorfismo relaciona el Dominio definido con una expresión de Dominios donde aparece el Dominio buscado.

Un ejemplo de esto lo constituye algún Dominio D que modele el Cálculo Lambda, entonces se debe tener que

$$D \cong D \to D$$

Consideramos importante que se revicen cuidadosamente los ejemplos que aparecen en el Capítulo APLICACIONES, donde se refleja el hecho de que desde los ejemplos prácticos más simples, los Dominios utilizados son reflexivos.

A continuación introduciremos un panorama de la Teoría que fundamenta la resolución de ecuaciones recursivas para Dominios. (Esta teoría también se aplica para fundamentar muchas de las afirmaciones realizadas en el parrafo de Límites del Capítulo anterior pero con un enfoque ligeramente distinto).

Definición 15.1.-Llamamos a T un endofuntor en una categoría, a un funtor de la Categoría en si misma; en particular, en la Categoría de Dominios, T es una doble función, que a cada Dominio A le asocia otro Dominio T(A) y a cada mapeo aproximable $f:A \to B$ le asocia otro mapeo aproximable $T(f):T(A) \to T(B)$ de manera que las identidades y composiciones se preservan, es decir,

$$T(I_A)=I_{T(A)} \quad y \quad T(f \circ g)=T(f) \circ T(g) \qquad ().$$

Por ejemplo, sea A un Dominio y definimos T en los Dominios por

$$T(B)=A+(BXB)$$

y si $f:B \to C$ definimos

$$T(f)=I_A+(fXf) \quad \text{donde}$$

$fXf:(BXB) \to (CXC)$

está dada en los elementos por

$$(fXf)(b,b')=(f(b),f(b'))$$

$I_A+(fXf):A+(BXB) \to A+(CXC)$ dada en los elementos por

$$(I_A+(fXf))(\perp)=\perp$$

y si $x\in|A+(BXB)|$ $x\neq\perp$ por la proposición 11.16 existe exclusivamente

$a\in|A|$ (ó $(b,b')\in|BXB|$) tal que

$$izq(a)=x \quad \text{ó } der(b,b')=x$$

entonces definimos

$$(I_A+(fXf))(x)=izq(a) \quad (\text{ó } (I_A+(fXf))(x)=der \circ fXf\ (b,b')\ )$$

Es directo verificar que $T(f)$ es aproximable y

$T(IB)=IA+(IBXIB)$ entonces

$T(I_B)(x)= (I_A+(I_B X I_B))(x)$ y según sea el caso

si $x=\perp$ $\quad T(I_B)(x)=\perp$;

si $izq(a)=x$ entonces $T(I_B)(x)=izq(a)=x$

y si $x=der(b,b')$

$$T(I_B)(x)=(der \circ I_B X I_B)(b,b')=der(b,b')=x.$$

Similarmente T preserva composiciones, de donde deducimos que T es un endofuntor.

Obsérvese que en este ejemplo T(f) es siempre estricto de manera que T también es un endofuntor en la Categoría D-estricta.

Presentamos ahora una definición que hace referencia a un concepto categórico, pero para no extendernos en nuestra discusión lo presentaremos de una manera más simple a como aparece en Teoría de Categorías.

Definición 15.2.-Sea T un endofuntor en la categoría de Dominios, una T-álgebra es una pareja formada por un Dominio E y un mapeo aproximable $k:T(E) \to E$. Si $m:T(F) \to F$ es otra T-álgebra entonces un morfismo entre T-àlgebras es un mapeo aproximable $h:E \to F$ tal que el diagrama

$$\begin{tikzcd} T(E) \arrow[r, "k"] \arrow[d, "T(h)"'] & E \arrow[d, "h"] \\ T(F) \arrow[r, "m"'] & F \end{tikzcd} \qquad (6)$$

conmuta,

es decir $h \circ k = m \circ T(h)$. ⟨⟩.

En nuestro ejemplo particular, una T-àlgebra es un mapeo aproximable

$$k:A+(E\times E) \to E.$$

Es importatnte señalar que si se aplica la definiciòn categòrica de T-àlgebra entonces k es necesariamente estricto, esto se debe al hecho de que, en tal definiciòn se pide la conmutatividad de ciertos diagramas, formados por dos transformaciones naturales (cuya existencia tambièn es un requisito), y una de las tales transformaciones naturales $\eta:I \to T$ debe satisfacer entre otras cosas que:

$$\begin{tikzcd} E \arrow[r, "\eta_E"] \arrow[dr, "I_E"'] & T(E) \arrow[d] \\ & E \end{tikzcd} \qquad (7)$$

conmute,

en este ejemplo particular η_E=der o $diag_E$ (donde $diag_E:E \to E\times E$ y $diag(b)=(b,b)$).

Para un estudio màs profundo de T-àlgebras sugerimos ver Mac Lane[72].

Un hecho bàsico para nuestro estudio es:

<u>Proposiciòn 15.3</u>.-Sea T un endofuntor en la Categorìa de Dominios entonces las T-àlgebras forman una Categorìa. ⟨⟩.

La demostraciòn de èste resultado se sigue por verificaciòn directa, y entonces podemos hacer la:

Definición 15.4.-Una T-àlgebra es inicial si es objeto inicial de la Categoría de T-àlgebras. ◊.

Y a partir de esta nomenclatura y con el objetivo de determinar un Dominio D de manera que $T(D)\cong D$ formulamos la:

Proposición 15.5.-Si $i:T(B)\to B$ es una T-àlgebra inicial, entonces tambièn lo es $T(i):T^2(B)\to T(B)$ e i es un isomorfismo de $T(B)$ en B.

◊.

DEMOSTRACION.-Como T es funtor, claramente $T(i):T^2(B)\to T(B)$, es decir, es una T-àlgebra; para ver que es inicial tomamos $m:T(F)\to F$ una T-àlgebra y $g,h:T(B)\to F$ dos mapeos aproximables tales que:

$$\begin{tikzcd} T^2(B) \arrow[r, "T(i)"] \arrow[d, "T(h)"'] & T(B) \arrow[d, "h"] \\ T(F) \arrow[r, "m"'] & F \end{tikzcd} \quad (8) \qquad \begin{tikzcd} T^2(B) \arrow[r, "T(i)"] \arrow[d, "T(g)"] & T(B) \arrow[d, "g"] \\ T(F) \arrow[r, "m"'] & F \end{tikzcd} \quad (9)$$

conmutan,

es decir, $h \circ T(i)= m \circ T(h)$ y $g \circ T(i)=m \circ T(g)$ como $i:T(B)\to B$ es inicial existe $j:B\to T(B)$ tal que

$$T(i) \circ T(j)= j \circ i$$

es decir,

$$\begin{tikzcd} T(B) \arrow[r, "i"] \arrow[d, "T(j)"'] & B \arrow[d, "j"] \\ T^2(B) \arrow[r, "T(i)"'] & T(B) \end{tikzcd} \qquad (10)$$

conmuta

de donde se tiene el diagrama (11),

pero entonces, $i \circ j$ es un morfismo del objeto inicial $i:T(B)\to B$ en si mismo, de donde $i \circ j=I_B$, ademàs tenemos que $h \circ j$ y $g \circ j$ son dos morfismos de la T-àlgebra $i:T(B)\to B$ en $m:T(F)\to F$, por unicidad $h \circ j$

$$\begin{tikzcd}
T(B) \arrow[r, "i"] \arrow[d, "T(j)"'] & B \arrow[d, "j"] \\
T2(B) \arrow[r, "T(i)"] \arrow[d, "T(i)"'] & T(B) \arrow[d, "i"] \\
T(B) \arrow[r, "i"] & B
\end{tikzcd} \qquad (11)$$

conmuta,

$=g \circ j$, pero j es sobre pues $i \circ j=I_B$ tenemos $h=g$ lo que prueba que los dos morfismos que salen de la T-àlgebra $T(i):T^2(B)\to T(B)$ son uno solo, la existencia esta garantizada por el hecho de que si $i:T(B)\to B$ es inicial entonces solo es necesario componer con j para obtener un morfismo de $T(i):T^2(B)\to B$ en cualquier otra T-àlgebra.

Probaremos ahora que i es isomorfismo, ya vimos que $i \circ j=I_B$ y ademàs ver diagrama (10) $j \circ i=T(i) \circ T(j)$ de donde

$$j \circ i=T(i \circ j)=T(I_B)=I_{T(B)} .$$

$\therefore$ i es un isomorfismo y j su inverso. [].

Del resultado anterior, vemos que si tenemos una T-àlgebra inicial, tenemos un Dominio tal que $B\cong T(B)$; desafortunadamente, el hecho de que $T(B)\cong B$ no garantiza que la T-àlgebra $i:T(B)\to B$, donde i es es isomorfismo, sea inicial.

<u>Definiciòn 15.6</u>.-En la Categorìa D-estricta, un endofuntor T es continuo en funciones si para cualesquiera dos Dominios B y C la funciòn $T'=T$ en los mapeos aproximables, dada por

$$T':(B\to C) \to (T(B)\to T(C))$$

$$f \longmapsto T(f)$$

es aproximable. ◇.

<u>Teorema 15.7</u>.-Si T es un endofuntor continuo en funciones (en la

Categoría $\mathbb{D}$-estricta) y $B \cong T(B)$ entonces la T-álgebra $i:T(B)\to B$ dada por el isomorfismo, es tal que para toda T-álgebra $k:T(E)\to E$ existe un morfismo de T-álgebras de $i:T(B)\to B$ en $k:T(E)\to E$.

⟨⟩.

DEMOSTRACION.-Sea $i:T(B)\to B$ la T-álgebra dada por el isomorfismo y sea $j:B\to T(B)$ el isomorfismo inverso, supongamos $k:T(E)\to E$ es una T-álgebra, buscamos un mapeo aproximable $h:B\to E$ que satisfaga

$$h \circ i = k \circ T(h)$$

ó aplicando j a la izquierda

$$h = k \circ T(h) \circ j. \tag{12}$$

En el Dominio $B\to E$, (12) es una ecuación recursiva y la función que la define es aproximable por nuestra hipótesis sobre T, entonces aplicando el operador mínimo punto fijo obtenemos $h=\mathrm{fij}(k \circ T() \circ j)$ existe, sin embargo no siempre es la única función con la propiedad (12) ⟨⟩.

La interrogante final es si existe en la Categoría de Dominios un mínimo Dominio tal que $T(B)\cong B$, en tal caso ¿por qué?. La razón se debe a que los endofuntores más usuales poseen propiedades de continuidad en los Dominios. Estas ideas son mejor expresadas en términos del concepto de subdominio. Este concepto puede definirse cartegóricamente a partir del concepro de subobjeto, a pesar de esto, preferimos presentarlo aquí desde el punto de vista de su definición en términos de Sistemas de Información, ya que de esta forma se conservan muchas de las ideas intuitivas.

<u>Definición 15.8</u>.-Sean A y B Sistemas de Información, decimos que B es menor que A, denotado por $B \trianglelefteq A$ si y sólo si

$$D_B \subseteq D_A \quad \Delta_A = \Delta_B \quad Con_B \subseteq Con_A \quad (\text{si } u \in Con_B \Rightarrow u \in Con_A)$$

y $\vdash_B \subseteq \vdash_A$ (es decir, si $u\vdash_B X$ entonces $u\vdash_A X$).

En este caso decimos que $|B|$ es un subdominio de $|A|$ y también lo denotamos por $|B|\triangleleft|A|$. $\Diamond$.

<u>Definición</u> <u>15.9</u>.-Sea A un Sistema de Información y $D\subseteq D_A$ 't' $\Delta_A\in D$ llamamos $|\langle D\rangle|$ al subdomio generado por el Sistema de Información:

$$\langle D\rangle=(D,\Delta_A,Con_A|_D,\vdash_A|_D). \qquad \Diamond.$$

Observese que si $|A|\triangleleft|B|$ entonces $|A|\subseteq|B|$ y si $B\subseteq|A|$ y $\perp_A\in B$ es fácil construir el Sistema de Información a partir de A que tiene como Dominio B.

Una proposición que contribuye a la caracterización de subdominios en términos de funciones aproximables es:

<u>Proposición</u> <u>15.10</u>.-Si $A\triangleleft B$ entonces existe mapeos aproximables $i:A\to B$ y $j:B\to A$ tales que:

(i) $j\circ i=I_A$ e $i\circ j\subseteq I_B$

(ii) dados por

$$i(x)=\{Y\in D_B\ |Y\in x\}=\{Y\in D_A\ |\ Y\in x\}=x \text{ pues } x\in D_A$$

y

$$j(y)=\{Y\in D_A\ |\ Y\in y\}.$$

A una pareja que satisfaga (i) se le llama la pareja de proyección, y en el caso anterior i, j son únicas. $\Diamond$.

Omitiremos la demostración pues es muy sencilla y el lector la puede imaginar sin problema.

<u>Proposición</u> <u>15.11</u>.-Si $A\triangleleft B$ e (i,j) es la pareja de proyección entonces i, j son estrictos. $\Diamond$.

DEMOSTRACION.-Es claro que $i(\perp)=\perp$ por la definición de i ahora $j(\perp)=j(i(\perp))=I_A(\perp)=\perp$. []

Obsérvese que si $B \triangleleft A$ y $C \triangleleft A$ entonces $C \triangleleft B$ si y sólo si $D_C \subseteq D_B$ y $Con_C \subseteq Con_B$ y $\vdash_C \subseteq \vdash_B$, además la unión de una familia dirigida de subdominios de un Dominio es un subdominio.

Continuando con nuestro desarrollo teórico presentamos la:

<u>Definición</u> <u>15.12</u>.-Un endofuntor en la Categoría de Dominios es monótono en Dominios si y sólo si, siempre que $A \triangleleft B$ con pareja de proyección i,j ,se tiene $T(A) \triangleleft T(B)$ y $T(i),T(j)$ es la pareja de proyección de $T(A)$ y $T(B)$.

<u>Definición</u> <u>15.13</u>.-Decimos que un endofuntor monótono en Dominios es continuo en Dominios si y sólo si , para cada cadena

$$A_1 \triangleleft A_2 \triangleleft , \dots , A_n \triangleleft \dots \quad y \ A_n \triangleleft A \quad \forall n \in \mathbb{N}$$

se tiene $$T\left(\bigcup_{n=1}^{\infty} A_n\right) = \bigcup_{n=1}^{\infty} T(A_n). \qquad \langle\rangle.$$

Nos encontramos ahora en condiciones de enunciar el siguiente Teorema que cubre una amplia gama de ejemplos.

<u>Teorema</u> <u>15.14</u>.-Si T es un endofuntor continuo en funciones, monótono y continuo en Dominios y existe un Dominio A tal que $A \triangleleft T(A)$ entonces existe una T-álgebra inicial. $\langle\rangle$.

DEMOSTRACION.-Las hipótesis sobre el Dominio A nos dicen que T(A) proviene de un Sistema de Información sobre el mismo conjunto de afirmaciones, por la monotonía de T y aplicando inducción tenemos:

$$T^n(A) \triangleleft T^{n+1}(A). \qquad (\text{denotamor } T^0(A)=\{\perp_A\})$$

definimos $B=\bigcup_{n=0}^{\infty} T^n(A)$ entonces por ser T continuo en Dominios tenemos

$$T(B)=T\left(\bigcup_{n=0}^{\infty} T^{n}(A)\right)=\bigcup_{n=0}^{\infty} T^{n+1}(A)=B\ ,$$

entonces B es una T-Álgebra bajo el morfismo identidad, $I_B:B\to B$ es una T-Álgebra y como T es continuo en funciones por el Teorema 15.7 para toda T-Álgebra $k:T(E)\to E$ existe un morfismo de T-Álgebras , de $I_B:T(B)\to B$ en $k:T(E)\to E$ definido por un mapeo aproximable $h:B\to E$ que satisface $k \circ T(h)=h$.

Debemos comprobar que h es único para comprobar que $I_B:B\to B$ es inicial.

Como $T^n(A)\triangleleft B$ para cada n, existe una pareja de proyección (i_n, j_n) tal que $i_n:T^n(A)\to B$ y $j_n:B\to T^n(A)$, definimos $p_n:B\to B$ como $p_n=i_n \circ j_n$, observese que p_n es estricto pues j_n e i_n lo son, y ya que T es monótono en Dominios

$$T(p_n)=T(i_n) \circ T(j_n)=i_{n+1} \circ j_{n+1}=p_{n+1}\ .$$

Obsérvese que p_n esta dado por,

$$p_n(b)=b\ \{Y \mid Y\subseteq a \text{ para algún elemento } a\in|T^n(A)|\}$$

de donde es claro que $p_n(b)\subseteq p_{n+1}(b)\quad \forall\, b\in|B|$ por lo que

$$p_n \subseteq p_{n+1} \qquad \forall\, n\in\mathbb{N}$$

afirmamos que $\bigcup_{n=1}^{\infty} p_n = I_B$.

Sea $b\in|B|$ arbitrario, es claro que $p_n(b)\subseteq b\ \forall\, n\in\mathbb{N}$ de donde $p_n\subseteq I_B\ \forall\, n$ y entonces $\bigcup_{n=1}^{\infty} p_n$ tiene sentido y

$$\bigcup_{n=1}^{\infty} p_n \subseteq I_B\ .$$

Si $b\in|B|=\bigcup_{n=0}^{\infty} T^n(A)$ entonces existe M 't' $b\in|T^M(A)|$, entonces $p_M(b)=b$ de donde $p_n(b)=b\ \forall\, n>M$

$$\therefore\quad \bigcup_{n=1}^{\infty} p_n(b)=b.$$

Definimos $h_n=h \circ p_n:B\to E$, recuérdese que del Teorema 15.7 h es estricto y $p_0(x)=\perp_B\ \forall x\in|B|$, entonces $h_0(x)=\perp_E\ \forall x\in|E|$ es decir

$$h_0=\perp_{B\to E}\ ;$$

además como:

$$k \circ T(h_n)=k \circ T(h \circ p_n)=k \circ T(h) \circ T(p_n)=h \circ T(p_n)$$

$=h \circ p_{n+1} = h_{n+1}$

si llamamos $R:(B\to E)\to(B\to E)$ dado por $R(f)=k \circ T(f)$ observamos que R es aproximable y $R(h_n)=h_{n+1}$ por lo que:

$$fij(R)=\bigcup_{n=0}^{\infty} R^{n}(\perp_{(B\to E)})=\bigcup_{n=0}^{\infty} h_n=\bigcup_{n=0}^{\infty}(h \circ p_n)$$

ahora, "o" es un mapeo aproximable en dos varibles por lo que es aproximable en la segunda, y entonces obtenemos:

$$\bigcup_{n=0}^{\infty}(h \circ p_n)=h \circ \bigcup_{n=0}^{\infty} p_n=h \circ I_B = h$$

Concluimos que $h=fij(R)$, es decir, si h y h' son dos morfismos de T-álgebras de $I_B:T(B)\to B$ en $k:T(E)\to E$ entonces $h=fij(R)$ y $h'=fij(R)$ de donde $h=h'$, es decir h es único y por lo tanto $I_B:T(B)\to B$ es una T-álgebra inicial. [].

Definición 15.15.-Decimos que B es un retracto de A, y lo escribimos $B\trianglelefteq A$ si y sólo si

$$B\cong B' \quad B'\triangleleft A. \qquad \diamond.$$

Proposición 15.16.- $B\trianglelefteq A$ si y sólo si existe una pareja (también llamada pareja de proyección, no necesariamente única)

$i:B\to A$ y $j:A\to B$ con

$$j \circ i=I_B \qquad e \quad i \circ j\leq I_A \qquad \diamond.$$

DEMOSTRACION.-(=>)Si $B\trianglelefteq A$ sea $B'\cong B$ tal que $B'\triangleleft A$, sea $k:B'\to B$ el isomorfismo e (i',j') la pareja de proyección de $B'\triangleleft A$, entonces llamamos

$$i=i' \circ k^{-1}:B\to A \quad y \quad j=k \circ j':A\to B$$

y $\quad j \circ i=k \circ j' \circ i' \circ k^{-1}=k \circ I_A \circ k^{-1}=I_B$

e $\quad i \circ j=i' \circ k^{-1} \circ k \circ j'=i' \circ j' \; I_A$.

(<=)Basta construir B' como la imgen de i donde $j \circ i=I_B$ y si $x\in|B'|$ existe y 't' $i(y)=x$ pero entonces $i \circ j(x)=i(j(i(y)))=i(y)=x$ y para construir B' sólo se requiere definir

$D_{B'} = D_A$ $\Delta_{B'} = \Delta_A$

$u \in Con_{B'}$ $\Leftrightarrow$ u es finito y $u \subseteq i(x)$ para algún $x \in |B'|$

y $u \vdash_{B'} X \Leftrightarrow u \vdash_A X$

dejamos las verificaciones correspondientes al lector interesado.

[].

Es importante mencionar que el nombre de retracto se deriva de consideraciones topológicas iniciales en esta área, donde retracto se refiere a la imagen de una función "a" con la propiedad de que $a \circ a = a$, en nuestro caso $a = i \circ j$ pues

$$(i \circ j) \circ (i \circ j) = i \circ (j \circ i) \circ j = i \circ I_B \circ j = i \circ j.$$

El siguiente resultado, nos dice, en términos de la relación $\trianglelefteq$, en que sentido una T-álgebra inicial es mínima.

<u>Teorema</u> <u>15.17</u>.-Si T es un endofuntor en la Categoría $\mathbb{D}$-estricta continuo en funciones, si $i:T(B) \to B$ es una T-álgebra inicial, entonces para cualquier Dominio E tal que $E \cong T(E)$ tenemos $B \trianglelefteq E$.

<>.

DEMOSTRACION.-Como $i:T(B) \to B$ es inicial, existe un mapeo aproximable $h:B \to E$ que define un morfismo de T-álgebras de $i:T(B) \to B$ en $u:T(E) \to E$ (donde i, u son isomorfismos), ahora como $T(E) \cong E$ por el Teorema 15.7 existe un mapeo aproximable $g:E \to B$ que define un morfismo de T-álgebras en sentido opuesto a h. Pero entonces, $g \circ h$ define un morfismo $i:T(B) \to B$ en si mismo y por ser inicial se debe tener $g \circ h = I_B$, este hecho, y el resultado anterior nos hacen ver que sólo es necesario probar que $h \circ g \subseteq I_E$

Como $i:T(B) \to B$ es inicial (Teorema 15.5) i es isomorfismo, sea

$j=i^{-1}:B\to T(B)$ y sean $u:T(E)\to E$ y $v:E\to T(E)$ los isomorfismos debidos a la hipótesis $E\cong T(E)$. Del hecho de que g y h definen morfismos en la Categoría de T-álgebras tenemos

$$g=i\circ T(g)\circ v \quad \text{y} \quad h=u\circ T(h)\circ j$$

además por ser h único con esta propiedad y el Teorema 15.7 g y h son los mínimos puntos fijos de sus respectivas ecuaciones recursivas, de manera que si definimos inductivamente:

$$g_0=\perp_{E\to B} \quad \text{y} \quad h_0=\perp_{B\to E}$$

y

$$g_{n+1}=i\circ T(g_n)\circ v, \quad h_{n+1}=u\circ T(h_n)\circ j$$

entonces por el Teorema 13.3

$$g=\bigsqcup_{n=0}^{\infty} g_n \quad \text{y} \quad h=\bigsqcup_{n=0}^{\infty} h_n .$$

Ahora observamos que $h_0\circ g_0:E\to E$ y

$$h_0\circ g_0=\text{const}\perp_E\circ\text{const}\perp_B=\text{const}\perp_E$$

concluimos $h_0\circ g_0=\perp_{E\to E}$ y además

$$\begin{aligned}
h_{n+1}\circ g_{n+1} &= u\circ T(h_n)\circ j\circ i\circ T(g_n)\circ v\\
&= u\circ T(h_n)\circ I_B\circ T(g_n)\circ v\\
&= u\circ T(h_n)\circ T(g_n)\circ v=u\circ T(h_n\circ g_n)\circ v
\end{aligned}$$

y como la composición "o" es aproximable en sus dos variables

$$h\circ g=\bigsqcup_{n=0}^{\infty} g_n\circ\bigsqcup_{n=0}^{\infty} h_n=\bigsqcup_{n=0}^{\infty}(h_n\circ g_n)$$

concluimos que $h\circ g$ es el mínimo punto fijo para la ecuación

$$k=u\circ T(k)\circ v$$

pero I_E es un punto fijo de ésta ecuación pues

$$u\circ T(I_E)\circ v=u\circ I_{T(E)}\circ v=u\circ v=I_E$$

$$\therefore\ h\circ g\subseteq I_E . \qquad [].$$

Revisaremos algunos ejemplos para ilustrar como se pueden hacer construcciones explícitas.

Ejemplo 1)Un Dominio de expresiones S.

Sea A un Dominio (Sistema de Información), nuestro objetivo es construir un Dominio S de árboles binarios construidos por elementos de A como átomos, la ecuación de Dominios que queremos resolver es

$$S = A + (S \times S)$$

Si tal Dominio existe, podemos decir, salvo por isomorfismo que los elementos de S son $\perp$ o elementos del Dominio A o parejas de elementos de S.

La existemcia de tal Dominio puede realizarse de muchas maneras, como ya hemos visto, pero ahora realizaremos una construcción. Para realizar una construcción explícita debemos considerar que información debe proporcionarse de un posible elemento. Al proporcionar información podemos entrar en un ciclo hacia atrás que para los elementos finitos no será interminable. Como veremos los elementos infinitos de S pueden contener replicas de si mismos, pero para definir un Dominio por completo lo único indispensable consiste en construir los elementos finitos a partir de los datos en forma sistemática. Afortunadamente las condiciones para satisfacer la ecuación que hemos presentado no son complicadas.

En primer lugar, necesitamos copias de todos los objetos de datos del sistema A en la unión ajena; la manera de conseguir esto consiste en tomar un dato $\triangle \notin D_A$ y definir $\triangle_S = (\triangle, \triangle)$. Esto nos provee un miembro de D_S, aquel que debemos siempre tener. La copia de un dasto $X \in D_A$ será $(X, \triangle)$. Los otros miembros de D_S serán de la forma $(\triangle, R)$ donde R nos da información sobre otra clase de elementos de S. Como S debe ser la unión ajena utilizamos el esquema de la definición 11.12 .

A continuación pensamos que clase de información debe contener una estructura R, como nuestro objetivo es que sea información sobre un producto, nos imaginamos que ya hemos definido D_S y observamos la definición del producto, con esta idea en mente realizamos la siguiena definición inductiva para D_S.

(1) $\Delta_S \in D_S$

(2) $(X,\Delta) \in D_S$ siempre que $X \in D_A$

(3) Si $Y, Z \in D_S$ entonces $(\Delta,(Y,\Delta_S))$ y $(\Delta,(\Delta_S,Z)) \in D_S$

Definir los objetos de datos no define el Dominio, ya que los mismos datos pueden definir Sistemas de Información muy distintos. Los datos son sólo partículas y su significado aparece cuando Con_S y $\vdash_S$ son definidos. Como ya conmocemos las nociones que se aplican para definir el producto y la unión ajena, únicamente copiamos estos patrones adecuadamente para realizar la definición de Con_S,

(5) $u \cup \{\Delta_S\} \in Con_S$ siempre que $u \in Con_S$

(6) Si $w \in Con_A$ entonces $\{(X,\Delta) \mid X \in w\} \in Con_S$

(7) $\forall\ u, v \in Con_S$

$\{(\Delta,(Y,\Delta_S)) \mid Y \in u\} \cup \{(\Delta,(\Delta_S,Z)) \mid Z \in v\} \in Con_S$

Es importante señalar que en nuestras definiciones nos referimos al conjunto definido como el menor conjunto bajo contención que satisface las condiciones de la definición.

Obsérvese que hemos conseguido que un conjunto consistente (sin considerar a Δ_S) sea una copia de un conjunto consistente para el sistema A o una copia de un conjunto consistente para SXS.

Resta definir la relación $\vdash_S$, presentamos a continuación las cláusulas que son prácticamente establecidas por nuestro objetivo.

(8) $u \vdash_S \Delta_S \quad \forall\ u \in Con_S$

(9) Si $u \vdash_S Y$ entonces $u \cup \{\Delta_S\} \vdash_S Y$

(10) Si $w \vdash_A W$ entonces

$\{(X,\Delta) \mid X \in w\} \vdash_S (W,\Delta)$

(11) Si $u \vdash_S X$ y $v \in Con_S$ entonces

$\{(\Delta,(Y,\Delta_S)) \mid Y \in u\} \cup \{(\Delta,(\Delta_S,Z)) \mid Z \in v\} \vdash_S (\Delta,(X,\Delta_S))$

y (12) Si $v \vdash_S X$ y $u \in Con_S$ entonces

$\{(\Delta,(Y,\Delta_S)) \mid Y \in u\} \cup \{(\Delta,(\Delta_S,Z)) \mid Z \in v\} \vdash_S (\Delta,(\Delta_S,X))$

Debe verificarse a travès de una demostraciòn inductiva que se satisfacen los axiomas para que S sea un Sistema de Informaciòn. Los pasos de la demostraciòn son mecànicos y se concentran en analizar los casos especificados de (4) a (7) y de (8) a (12).

Una vez verificado que S es un Sistema de Informaciòn puede construirse el sistema A+(SXS) y observarse, sin mayor dificultad que es idèntico a S. Este hecho, aparentemente sorprendente no es una coincidencia pues se debe a que nuestra notaciòn y construcciòn de S fueron realizadas a la par con las definiciones de uniòn ajena y producto.

Puede ahora verificarse que S es la mìnima soluciòn a la ecuaciòn en el sentido del Teorema 15.17 pues es una T-àlgebra inicial. Intuitivamente es claro que la soluciòn construida debe ser mìnima en algùn sentido pues ùnicamnete incluimos los requisitos indispensables a la ecuaciòn y nada màs.

Señalaremos sin embargo que existen muchas soluciones a esta ecuaciòn de Dominios.

Para finalizar con este ejemplo, indicaremos porque lo hemos llamado el Dominio de expresiones S. La razòn se debe a que existen mapeos aproximables

Atomo:A->S que corresponde a la inclusiòn de A en S

Esàtomo:S->BOOL que decide si un elemento es un àtomo, es decir una copia de un elemento de A en S.

Ademàs como SXS es parte de S podemos definir mapeos aproximables

par:SXS->S (la inclusiòn de SXS en S)

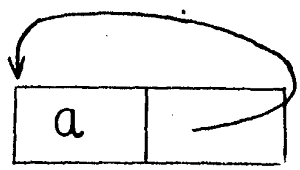
y prm:S->S y sgd:S->S

que en la terminologìa de LISP corresponden a las funciones CONS, CAR y CDR respectivamente.

Nuestro Dominio tiene, sin embargo, algunas diferencias sobre las expresiones S que estamos acostumbrados a trabajar en las implantaciones de LISP, pues aparecen elementos infinitos muchos de los cuales no son directamente representables por la implantación, por ejemplo si $a \in |S|$ entonces utilizando el operador mínimo punto fijo resolvemos la ecuación

$$x=par(Atomo(a),x)$$

cuya solución corresponde a una lista ligada infinita de aes y que en este caso particular puede intentarse simular operacionalmente por una estructura como:



Esta es sólo una muestra de la riqueza del Dominio construido ya que si representamos las expresiones S como árboles binarios observamos que el Dominio S, además de tener los árboles finitos, (corespondientes a los elementos finitos), continene todos los infinitos.

Similarmente, presentamos el siguiente ejemplo como una construcción explícita para la ecuación

$$D = A + (D \to D)$$

La construcción se realiza de manera análoga a la que se hizo para S, siguiendo los patrones de la unión ajena y el espacio de funciones. Presentamos las cláusulas de la construcción a continuación donde $\triangle_A \notin D$ es un dato y definimos $\triangle_D = (\triangle, \triangle)$.

(1) $\triangle_D \in D_D$

(2) $(X, \triangle_D) \in D$ siempre que $X \in D_A$

(3) Si $u, v \in Con_D$ entonces $(\triangle, (u,v)) \in D_D$

(4) $\emptyset \in Con_D$

(5) $u \cup \{\triangle_D\} \in Con_D \quad \forall\ u \in Con_D$

(6) Si $w \in Con_A$ entonces $\{(X, \triangle) \mid X \in w\} \in Con_D$

(7) $\{(\triangle, (u_0, v_0)), \ldots, (\triangle, (u_n, v_n))\} \in Con_D \quad u_i, v_i \in Con_D \quad i < n$

si sólo si $\forall\ I \subseteq \{0,1,\ldots,n\}$ 't' $\bigcup_I u_i \in Con_D$ se tiene $\bigcup_I v_i \in Con_D$

(8) $u \vdash_D \Delta \quad \forall\ u \in Con_D$

(9) Si $w \vdash_A W$ entonces $\{(X,\Delta) \mid X \in w\} \vdash_D (W,\Delta)$

(10) Si $u \vdash_D Y$ entonces $u \cup \{\Delta_D\} \vdash_D Y$

(11) $r=\{(\Delta,(u_0,v_0)),\ldots,(\Delta,(u_n,v_n))\} \vdash_D (\Delta,(u',v'))$

si y sólo si el conjunto $r \in Con_D$ y si $I=\{i \mid u' \vdash_D u_i\}$ se tiene $v_i \vdash_D v'$.

No haremos los detalles que comprueban que se trata de una solución, ya que preferimos mostrar otro ejemplo para una construcción explícita.

El caso que presentamos ahora corresponde a la construcción de un Dominio Universal U, el nombre de Dominio Universal se debe a que puede probarse que todo Dominio que posee una base D_A contable (ver Scott[82]) es isomorfo a un subdominio de U, es decir, que si A es un Sistema de Información y D_A es contable ($\#(D_A) < \aleph_0$) existe una pareja de proyección $a{:}A \to U$ y $b{:}U \to A$ tal que $b \circ a = I_A$ y $a \circ b \sqsubseteq I_U$ o aplicando la notación $A \trianglelefteq U$, A es un retracto de U. Para un estudio más completo del Dominio Universal y sus retractos sugerimos consultar Scott[76], Scott[81], Stot[82] y Bracho[79].

Esta propiedad para U hacen posible que $U \to U$ este isomorfamente contenido en U y por lo tanto $U \to U$ es un modelo de Cálculo es decir una solución para la ecuación

$$D = D \to D$$

Sin embargo, a pesar de la riqueza y poderío de este Dominio existen todavía muchas interrogantes sobre él que son motivo de investigación, entre éstas se encuentran la investigación sobre la unicidad de la pareja de proyección para un retracto, e incluso determinar si una pareja determina univocamente un Dominio.

U también ha sido usado para construir un Dominio de Dominios, es decir, un Dominio cuyos elementos sean Dominios pero el desarrollo de la teoría que determine que endofuntores y funtores corresponden a mapeos aproximables es un nuevo campo de estudio e investigación, para mayores detalles en estos aspectos consultese Scott[82].

La construcción de U, después de todo es muy simple, primeramente se contruye V un Dominio con un elemento máximo tal que satisface mínimamente la ecuación

$$V = V \times V$$

La definición inductiva para V esta dada por:

(1) $\nabla \neq \triangle$ y $\nabla, \triangle \in D_V$

(2) $(X, \triangle) \in D_V$ ó $(\triangle, Y) \in D_V$ siempre que $X \in D_V$ ó $Y \in D_V$ respectivamente.

En realidad partimos de dos objetos de datos distintos y construimos dos copias simulando el producto.

(3) $u \in Con_V$ $\forall\, u \in D_V$ u finito

y para la relación de implicación tenemos

(4) $u \vdash_V \triangle_V$ $\forall\, u \in D_V$

(5) $u \vdash_V \nabla$ si y sólo si $\nabla \in u$ ó $\{(X \mid (X, \triangle) \in u\} \vdash_V \nabla$ ó $\{Y \mid (\triangle, Y) \in u\} \vdash_V \nabla$

(6) $u \vdash_V (X', \triangle)$ si y sólo si $\nabla \in u$ ó $\{X \mid (X, \triangle) \in u\} \vdash_V X'$

(7) $u \vdash_V (\triangle, Y')$ si y sólo si $\nabla \in u$ ó $\{Y \mid (\triangle, Y) \in u\} \vdash_V Y'$

Es importante señalar que en ésta construcción ∇ implica todo, esto normalmente es incómodo y es usual la realización de la eliminación de para construir un nuevo Dominio, en nuestro caso esta eliminación produce finalmente U.

(8) $D_U = D_V - \{\nabla\}$

(9) $\triangle_U = \triangle_V$

(10) $Con_U = \{u \in D_U \mid$ u es finito y $u \nvdash_V \nabla \}$

(11) $u \vdash_U Y$ si y sólo si $u \in Con_U$ $Y \in D_U$ y $u \vdash_V Y$.

El mismo tipo de eliminación funciona cuando tenemos un Sistema de Información con un dato que implica todo, es decir:

Proposición 15.18.-Sea A un Sistema de Información tal que existe $\nabla_A \neq \Delta_A$ y $\nabla_A, \Delta_A \in D_A$ 't' $\{\nabla\} \vdash_A X \quad \forall X \in D_A$, entonces el sistema B dado por

i) $D_B = D_A - \{\nabla\}$

(ii) $\Delta_B = \Delta_A$

(iii) $Con_B = \{u \in Con_A \mid u \not\vdash_A \nabla\}$

(iv) $u \vdash_B X$ si y sólo si $u \in Con_B$, $X \in D_B$ y $u \vdash_A X$

es un Sistema de Información.

Esperamos que los ejemplos anteriores logren mostrar las ideas que estan presentes rodenado una construcción explícita de Dominios, en particular, la consatrucción explícita de una solución a una ecuación de Dominios.

Las construcciones realizadas se derivan de su necesidad práctica, los productos de Dominios son necesarios para el uso de eneadas y para funciones con varios argumentos, la unión ajena de Dominios se utiliza para hacer copias de Dominios y unirlas formando Dominios que contienen directamente a sus componentes, los espacios de funciones como Dominios proveen operadores que pueden aplicarse a operadores y producir operadores como valores.

Con la teoría presentada tenemos los fundamnetos matemáticos que respaldan las aplicaciones que presentamos a continuación.

TERCERA PARTE

Capítulo V

APLICACIONES

Durante este capítulo presentamos ejemplos de descripciones semánticas denotativas; los ejemplos que mostraremos aparecen dentro del marco formal que hemos construido en los capítulos anteriores, por lo que haremos algunas referencias a la terminología y resultados de dichos capítulos. Sin embargo mantendremos nuestra atención en la problemática de la semántica de lenguajes de programación y nos concentraremos en ilustrar los mecanismos de una descripción semántica formal.

Sección 16 Dominios Sintácticos y Semánticos

Una definición semántica formal en forma denotativa está dada primordialmente por la especificación de funciones semánticas, estas funciones mapean estructuras sintácticas en objetos matemáticos, de manera que el significado de una frase sintácticamente correcta es el objeto matemático asociado.

Las funciones semánticas tienen pues como dominios de definición lo que llamamos Dominios Sintácticos y como codominios de definición Dominios Semánticos. Los Dominios Semánticos son, precisamente, Dominios en la Categoría de Dominios que hemos definido. Los Dominios

Sintàcticos no necesariamente son Dominios en la Categorìa descrita en la tesis.

Presentamos brevemente algunos Dominios que necesitaremos para las aplicaciones pràcticas. Frecuentemenmte mencionaremos Dominios que por su gran utilidad, son comunes a muchas definiciones semànticas, a estos Dominios se acostumbra llamarlos Dominios Estandar y los presentamos sin mayor explicaciòn, ya que si no fueron introducidos anteriormente en ejemplos de capìtulos anteriores, su estructura es sencilla o muy similar a algùn ejemplo ya presentado y el lector no encontrarà problema para construir su definiciòn axiomàtica.

Ejemplos de estos Dominios son:

BOOL={Verdadero, Falso, Indefinido} que corresponde a los valores de verdad.

N , cuyos elementos estan en correspondencia biunivoca con los naturales (y $\perp_N$) y lo denotamos por {0, 1, 2, 3...}

Ent , que corresponde al Dominio de los enteros.

Indicaremos los Dominios Sintàcticos como conjuntos y asumiremos que estàn dados inductivamente por las producciones sintàcticas. Por ejemplo, si tenemos la producciòn:

E ::= +E | -E | E+E | E-E

para las expresiones de algùn lenguaje de programaciòn, entonces consideramos el Dominio Sintàctico cuyos elementos son todas las expresiones de la forma indicada.

Las construcciones de Dominios Semànticos a partir de otros Dominios corresponderàn a las construciones presentadas y seràn denotadas por los operadores +, X -> y ∞ .

Los Dominios Semànticos son objetos de la Categorìa de Dominios y seràn descritos de acuerdo a los ejemplos particualares. En algunos casos, haremos referencia a las tècnicas para la fundamentaciòn de

Dominios Reflexivos ya que, las necesidades de la definición semántica formal nos forzarán a utilizar Dominios de este tipo.

Consideramos importante advertir que frecuentemente unimos a nuestros Dominios un Dominio como {error}, es decir un Dominio cuya red es:

$$\begin{array}{c} \text{error} \\ | \\ \perp_{\{\text{error}\}} \end{array}$$

para equipar al Dominio original con un elemento extra que es incomparable con los otros elementos del Dominio. El incluir estos elementos extra permite, por ejemplo, definir funciones semánticas en situaciones de error ó identificadores no iniciados.

Asumiremos que las proyecciones e inclusiones entre estos Dominios extendidos y sus componentes mapean "error" en si mismo. No especificaremos los detalles técnicos de esta suposición pues nos desviaría de nuestros objetivos y consideramos que el lector puede, a partir de la teoría presentada especificarlos fácilmente. Los elementos extra proveen, como ya mencionamos una forma natural de manejar cálculos u operaciones que conducen a resultados equivocados. La estructura es tal que la verificación de que algún valor es "error" es aproximable, lo cual permite que nuestras definiciones semánticas detecten errores y dirijan apropiadamente situaciones de error. Tampoco detallaremos las afirmaciones anteriores.

Sección 17 Interpretación.

Un pequeño lenguaje, TINY.

Describiremos la sintaxis y principalmente la semántica de un pequeño lenguaje de programación llamado TINY. Nuestro propósito es proveer un primer vehículo para ilustrar los conceptos formales en su aplicación. En nuestros próximos ejemplos describiremos más conceptos sistemáticamente, ahora realizaremos un esquema de las ideas centrales y técnicas asociadas.

TINY esta constituido por dos estructuras principales, "Expresiones" y "Comandos", ambos pueden contener "Identificadores", los cuales son cadenas de letras o dígitos comenzando por una letra. Si I, denota identificadores arbitrarios, E, E_1, E_2 expresiones arbitrarias y C, C_1, C_2 denota comandos arbitrarios, las estructuras de TINY pueden especificarse por:

E ::= 0 | 1 | true | false | read | I | not E | $E_1=E_2$ | E_1+E_2

C ::= output E | I:=E | if E then C_1 else C_2 | while E do C

| $C_1;C_2$

La notación es una variante de BNF, (recordamos que ::= representa algo como "puede ser" y | denota "ó").

Obsérvese que en particualar, si utilizamos las producciones anteriores para definir una gramática, ésta resultaría ambigua pues no podemos especificar si

while E do $C_1;C_2$

corresponde a

(while E do C_1);C_2 ó

while E do ($C_1;C_2$).

No nos preocuparemos por este detalle, el cual corregiremos en ejemplos siguientes.

Para describir intuitivamente la semántica del lenguaje TINY, señalaremos que cada comando en TINY, cuando se ejecuta, altera el estado de la máquina, el estado lo consideramos formado por tres partes:

(i)La memoria, la cual es una correspondencia entre identificadores y valores. En realidad a cada identificador està ligado un valor ò el identificador no està ligado (iniciado).

(ii)La entrada es suministrada por el usuario para la ejecuciòn del programa y consiste de una sucesiòn (posiblemente vacia) de valores que pueden ser leidos utilizando la expresiòn "read".

(iii)La salida, que corresponde inicialmente a una sucesiòn vacia de valores, registra los resultados del comando "output E".

Cada expresiòn en TINY especifica un valor, como la expresiòn puede contener identificadores (por ejemplo a+b) su valor depende del estado. Todos los valores, ya sean de expresiones, ligados a identificadores o valores de entrada y salida son valores de verdad (true, false) ò nùmeros naturales $n \in N$ y el cero.

Explicaremos informalmente que significa cada estructura.

El valor de cada expresiòn es como sigue:

(1) 0 ò 1

El valor de 0 es el nùmero cero y el valor de 1 es el nùmero natural uno.

(2) true ò false

El valor de "true" es el valor de verdad Verdadero, mientas que el valor de "false" es el valor Falso.

(3) read

El valor de "read" es la siguiente partìcula de la entrada (un error ocurre si la entrada es vacìa). Como un efecto colateral, "read" retira la primer partìcula de la entrada de manera que despuès de su ejecuciòn, la cadena de entrada es una partìcula màs corta.

(4) I

El valor de la expresiòn anterior es el valor ligado al identificador en la memeoria (si I no está iniciado ocurre un error).

(5) not E

Si el valor de E es Verdadero, entonces "not E" es Falso, si E es Falso "not E" es Verdadero. En cualquier otro caso tenemos un error.

(6) $E_1=E_2$

El valor de $E_1=E_2$ es Verdadero si el valor de E_1 es el mismo que el valor de E_2 y Falso en otro caso.

(7) E_1+E_2

El valor de esta expresiòn es la suma numèrica de los valores de E_1 y E_2, si alguno de èstos valores no es un nùmero aparece un error.

Para los comandos tenemos:

(8) output E

El valor de E es colocado al inicio de la sucesiòn de salida.

(9) I:=E

I es ligado al valor de E en la memoria encimando cualquier otra liga anterior.

(10) if E then C_1 else C_2

Si el valor de E es Verdadero se ejecuta C_1, si E es Falso se ejecuta C_2 y en cualquier otro caso aparece un error.

(11) while E do C

Si el valor de E es Verdadero se ejecuta C y a continuaciòn "while E do C" es repetido con el estado que resulta de la ejecuciòn de C. Si el valor de E es Falso, no se realiza nada. Si el valor de E no es ni Verdadero ni Falso tenemos un error.

(12) $C_1;C_2$

C_1 se ejecuta y luego C_2

Como ejemplo de un programa en TINY tenemos el siguiente enunciado que calcula la suma de los nùmeros en la entrada.

Obsèrvese que abusamos de algunos detalles sintàcticos como el uso de "(", ")" y la justificaciòn de los margenes para la agrupaciòn de expresiones y comandos respectivamente.

```
sum:=0;
x:=read;
while not(x=true) do
    sum:=sum+x;
    x:=read;
output sum
```

Trataremos ahora de formalizar la descripciòn anterior de TINY. Esperamos que el lector comprenda las ideas principales a pesar de no dominar todos los detalles.

Una parte esencial del formalismo consiste en definir los Dominios, para trabajar la sintaxis definimos los siguientes Dominios Sintàcticos:

Ide={ I | I es un identificador}

Exp={ E | E es una expresiòn}

Com={ C | C es un comando}

Los nombres de los Dominios comenzaràn siempre con mayùscula, obsérvese que los Dominios Sintàcticos tambièn pueden cambiar de lenguaje en lenguaje.

Ahora formalizaremos el concepto de estado. Para esto llamamos Estado, al Dominio cuyos elementos seràn los posibles estados, Memoria al Dominio cuyos elementos seràn nuetro modelo de memoria, Salida al Dominio correspodiente a las posibles salidas o sucesiones de resultados y Valor para el Dominio de valores. Las definiciones de estos Dominios estan dadas por las siguientes ecuaciones de Dominios.

(i) Estado =Memoria X Valor X Salida

(ii) Memoria = Ide –›(Valor + {noligado})

(iii) Entrada = Valor

(iv) Salida = Valor

(v) Valor = N + BOOL

La ecuación (i) significa que Estado es el Dominio de tercias (m, e, sa) donde $m \in$ Memoria, $e \in$ Entrada y $sa \in$ Salida.

(ii) Significa que memoria es el Dominio de las funciones aproximables del Dominio Ide en el Dominio extendido Valor+{noligado}. Si $m \in$ Memoria e $I \in$ Ide entonces el resultado de aplicar m al argumento I está en Valor o es no ligado, en el primer caso el valor m(I) es el valor asociado a I en m, en otro caso I no tiene valor asociado.

Tanto Entrada como Salida son sucesiones de valores en Valor mientras que (v) nos dice que los valores son o un número o un valor de verdad.

Debemos ahora discutir las funciones semánticas para TINY. Las funciones semánticas definirán que es lo que denotan las estructuras sintácticas, para TINY necesitamos:

$\mathbb{E}$: Exp -> {significados de expresiones}

$\mathbb{C}$: Com -> {significados de comandos}

Es decir que si E es una expresión y C un comando $\mathbb{E}[[E]]$ y $\mathbb{C}[[C]]$ son los resultados de aplicar las funciones $\mathbb{E}$ y $\mathbb{C}$ a E y C respectivamente y son los significados definidos por la semántica de estas estructuras.

En general si X es una variable que corre sonbre algún Dominio Sintáctico, utilizaremos $\mathbb{X}$ para nombrar a la función semántica, de manera que por ejemplo

$$\mathbb{C}[[I:=E]] \text{ es el significado de } I:=E.$$

Los parentesis [[y]] son usados indicando objetos o elementos sintácticos al aplicarseles una función semántica.

Para definir el significado de una expresión, ya que la expresión produce un valor, pensaríamos inicialmente que el significado fuese un mienbro de Valor. Para representar esta idea daríamos a la función semántica $\mathbb{E}$ el tipo $\mathbb{E}$:-)Valor y entonces $\mathbb{E}[[E]]$ sería el valor de E, por ejemplo

$$\mathbb{E}[[\text{true}]] = \text{Verdadero}$$

Esto funciona para expresiones que son literales o constantes pero:

(i)La posibilidad de expresiones que causen errores como 1+true;

(ii)La dependencia del valor de expresiones del estadao, como x+1 que depende del valor ligado a x en memoria, o también "read" que depende del valor de entrada;

(iii)La posibilidad de que la evaluación de una expresión modifique el estado, por ejemplo,"read" retira el primer elemento de la cadena de entrada;

no permiten una descripción tan simple para $\mathbb{E}$.

Para manejar(i) debemos definir

$\mathbb{E}$: Exp -) Valor+{error} de manera que

$$\mathbb{E}[[E]] = \begin{cases} v & \text{si } v \text{ es el valor de } E \\ \text{error} & \text{si } E \text{ produce error} \end{cases}$$

por ejemplo $\mathbb{E}[[1+1]]=2$ pero $\mathbb{E}[[1+\text{true}]]=\text{error}$

Para resolver la problemática planteada en (ii) debemos hacer el resultado de una evaluación una función de estado, es decir:

$\mathbb{E}$: Exp -) (Estado -) [Valor+{error}]) de manera que

$$\mathbb{E}[[E]](s) = \begin{cases} v & \text{si } v \text{ es el valor de } E \text{ en } s \\ \text{error} & \text{si la evaluación produce error} \end{cases}$$

y por ejemplo

$$\mathbb{E}[[1+x]](s) = \begin{cases} 1+m(x) & \text{si } s=(m,e,sa) \text{ y } m(x) \text{ es número} \\ \text{error} & \text{en otro caso} \end{cases}$$

Finalmente para enfrentar (iii) debemos complicar aún más el valor de $\mathbb{E}$ pues debe también producirnos nuevos estados.

$$\mathbb{E} : \text{Exp} \to (\text{Estado} \to [\text{Valor} \times \text{Estado} + \{\text{error}\}\ ]\ )$$ y

entonces

$$\mathbb{E}[\![\ E\ ]\!](s) = \begin{cases} (v, s') & \text{donde } v \text{ es el valor de } E \text{ en el estado } s \\ & \text{y } s' \text{ es el estado posterior a la} \\ & \text{evaluación} \\ \text{error} & \text{si la evaluación produce error.} \end{cases}$$

por ejemplo,

$$\mathbb{E}[\![\ \text{read}\ ]\!] = \begin{cases} (\ cb(e), (m, cl(e), sa)\ ) & \text{si } s=(m,e,sa), \ e \text{ es no} \\ & \text{vacio y tiene como primer} \\ & \text{elemento a } cb(e) \text{ y como} \\ & \text{resto a } cl(e). \\ \text{error} & \text{si } e \text{ es vacio.} \end{cases}$$

Formalmente se define $\mathbb{E}$ en los diferentes casos de clases de expresiones, es decir en las clases de elementos de Exp. Por ejemplo para una expresión de la forma I definimos la cláusula semántica:

$$\mathbb{E}[\![\ I\ ]\!](m,e,sa) = (m(I)=\text{noligado}) \to \text{error}, (m(I), (m,e,sa))$$

donde utilizamos la notación

$$b \to v, v'$$

para decir que cuando se cumple b el resultado de la funcioón es v y de lo contrario es v', de manera que la cláusula anterior dice que si m(I) es no ligado (es decir I no está ligado en m) entonces se produce un error, de lo contrario, obtenemos el valor de I corresponde al valor ligado en la memoria, y no se modifica el estado.

De la misma manera, la cláusula para el "read" se rescribe como:

$$\mathbb{E}[\![\ \text{read}\ ]\!](m,e,sa) = \text{vacia}(e) \to \text{error}, (\ cb(e), (m, cl(e), sa)\ )$$

donde vacia(e) se cumple si e es vacia, cb(e) es el primer elemento de la cadena e y cl(e) es la cadena e sin el primer elemento.

Para los comandos, observamos que el efecto de ejecutar un comando es producir un nuevo estado o generar un error, entonces

$$\mathbb{C}:\text{Com} \rightarrow (\text{Estado} \rightarrow [\text{Estado} + \{\text{error}\}\ ]\ )$$

y una clàusula semàntica tìpica es:

$$\mathbb{C}[\![\ \text{output } E\ ]\!](s) = (\mathbb{E}[\![\ E\ ]\!](s) = (v, (m, e, sa)) \rightarrow (m, e, v.sa), \text{error}$$

donde las notaciòn v.sa corresponde a la sucesiòn formada por v como primer tèrmino seguido de la sucesiòn sa. De esta manera la clàusula semàntica $\mathbb{C}[\![$ output E $]\!]$ dice que cuando se aplica a un estado s, primeramente se evalua E con respecto al estado s y si esta evaluaciòn produce un valor v y un estado (m,e,sa) entonces el resultado es el estado (m,e,v.sa). De otra manera el resultado es un error.

Asì, las otras clàusulas semànticas son:

(1) $\mathbb{E}[\![\ 0\ ]\!](s) = (0, s) \qquad \mathbb{E}[\![\ 1\ ]\!](s) = (1, s)$

El valor de los sìmbolos 0 y 1 es el nùmero correspondiente y la evaluaciòn no modifica el estado.

(2) $\mathbb{E}[\![\ \text{true}\ ]\!](s) = (\text{Verdadero}, s) \quad \mathbb{E}[\![\ \text{false}\ ]\!](s) = (\text{Falso}, s)$

El valor de una constante "booleana" es el correspondiente valor de verdad y no se modifica el estado.

(5) $\mathbb{E}[\![\ \text{not } E\ ]\!](s) = (\ \mathbb{E}[\![\ E\ ]\!](s) = (v, s')\ ) \rightarrow$

$$(\text{esBOOL}(v) \rightarrow (\text{No}(v), s'), \text{error})\ ,\ \text{error}$$

donde esBOOL(v) es Verdadero si v se incluye en BOOL pues

$$v \in \text{Valor} = N + \text{BOOL}$$

y Falso en otro caso, ademàs No:BOOL->BOOL es el mapeo aproximable estricto dado por:

$$\text{No}(\text{Verdadero}) = \text{Falso} \quad \text{y} \quad \text{No}(\text{Falso}) = \text{Verdadero}$$

de manera que el valor de "not E" es la negaciòn del valor de E (pero un error se produce si E no tiene valor de verdad o E crea un error al evaluarse). El estado cambia a s' que resulta de la evaluaciòn de E.

(6) $\mathbb{E}[\![\ E_1 = E_2\ ]\!](s) = (\ \mathbb{E}[\![\ E_1\ ]\!](s) = (v', s')\ ) \rightarrow$

$$(\ \mathbb{E}[\![\ E_2\ ]\!](s') = (v'', s'')) \rightarrow (v' = v'', s''), \text{error}), \text{error}$$

donde v'=v'' es Verdadero si v' es igual a v'' y Falso en otro caso.

Concluimos que el resultado de $E_1=E_2$ en el estado s se obtine evaluando primeramente E_1 en s para obtener (v',s') (o error en cuyo caso $E_1=E_2$ produce error), y entonces se evalua E_2 en s' para obtener (v'',s'') (o error, en cuyo caso $E_1=E_2$ causa también error) y finalmente (v'=v'',s'') es regresado como resultado de la estructura sintáctica.

Explicaremos algunas de las cláusulas para enunciados.

$$(7)\ \mathbb{C}[[\ I:=E\ ]](s)=(\mathbb{E}[[\ E\ ]](s)=(v,(m,e,sa)))\rightarrow$$

$$(m[I/v],e,sa),error$$

donde la notación [/] significa:

$$(m[I/v])(I') = \begin{cases} v & \text{si } I=I' \\ m(I') & \text{en otro caso.} \end{cases}$$

es decir, m[I/v] es la misma función que m salvo en I donde ahora toma el valor v.

Entonces $\mathbb{C}$[[I:=E]](s) es el estado idéntico al estado resultande de la evaluación de E en s excepto que el valor v de E es ligado a I en la memoria. (Si E produce error entonces también lo produce I:=E).

$$(8)\ \mathbb{C}\ [[\ \text{if } E \text{ then } C_1 \text{ else } C_2\ ]](s)=$$

$$(\mathbb{E}[[\ E\ ]](s)=(v',s'))\rightarrow$$

$$(esBOOL(v')\rightarrow$$

$$(v'\rightarrow \mathbb{C}[[\ C_1\ ]](s'),\mathbb{C}[[\ C_2\ ]](s')),error),error$$

Cuyo significado es, si E produce como resultado (v',s') cuando es evaluado en s, entonces C_1 o C_2 son ejecutados en el estado s' resultate de la evaluación de E dependiendo de si el valor v' es Verdadero o Falso, cualquier error se propaga para producir un error.

$$(9)\ \mathbb{C}[[\ \text{while } E \text{ do } C\ ]](s)=$$

$$(\mathbb{E}[[\ E\ ]](s)=(v',s'))\rightarrow$$

$$(esBOOL(v')\rightarrow(v\rightarrow(\mathbb{C}[[\ C\ ]](s')=s'')\rightarrow$$

$$\mathbb{C}[[\text{while } E \text{ do } C]](s''),error),s'),error),error$$

Si E produce un error al evaluarse, también "while E do C", si E produce un valor v' y el estado s' entonce si v es Verdadero se ejecuta C para obtener el estado s'' y entonces "while E do C" se ejecuta nuevamente comenzando con s'', si v es Falso el resultado de "while E do C" es s', es decir el estado producido por E. Si v no es un valor de verdad o $\mathbb{C}[[C]](s')$=error entonces "while E do C" produce un error. Obsérvese que la cláusula es recursiva, mientras que $\mathbb{E}$ no lo es.

$$(10)\ \mathbb{C}[[C_1;C_2]](s)=(\mathbb{C}[[C_1]](s)=\text{error})\rightarrow \text{error}, \mathbb{C}[[C_2]](\mathbb{C}[[C_1]](s))$$

es decir, si C_1 produce un error, también lo produce "$C_1;C_2$" de otra manera C_2 es ejecutado en el estado que resulte de la ejecución de C_1.

El ejemplo muestra como son necesarias funciones de alto nivel, uniones ajenas y definiciones recursivas.

Es importante señalar que existen en uso muchas abre iaturas y notaciones especiales para hacer las definiciones de las cláusulas semánticas mucho más cortas y abstractas, introduciremos parte de la notación en ejemplos siguientes, pero sugerimos consultar Gordon[79] para familiarizarse con más simbología.

Como parte de ésta simplificación observese que $\mathbb{E}[[E]]$ es una función de los Estados en ValorXEstado+{error} y $\mathbb{E}[[E]](s)$ es un error o una pareja formada por el valor v relativo al estado s y el nuevo estado. Para recortar el número de paréntesis asumimos que la aplicación de funciones se asocia a la izquierda y omitimos los paréntesis en los argumentos de funciones semánticas; entonces $\mathbb{E}[[E]]s$ es equivalente a $\mathbb{E}[[E]](s)$.

El presente ejemplo muestra que la definición formal de TINY consiste de tres partes principales:

(i)Especificación de los Dominios Sintácticos Exp y Com.

(ii)Especificación de los Dominios Semánticos :Valor, Estado, etc.

(iii)Especificación de las funciones semánticas $\mathcal{E}$ y $\mathcal{C}$ que mapean estructuras sintácticas en sus significados.

En resumen la descripción formal de TINY está dada por:

Sintaxis Abstracta

$I \in$ Ide	Identificadores
$E \in$ Exp	Expresiones
$C \in$ Com	Comandos
$M \in$ Pro	Programas

$E ::= 0 \mid 1 \mid \text{true} \mid \text{false} \mid \text{read} \mid I \mid \text{not } E \mid E_1 = E_2 \mid E_1 + E_2$

$C ::= \text{output } E \mid I := E \mid \text{if } E \text{ then } C_1 \text{ else } C_2 \mid C_1 ; C_2 \mid$
$\quad \text{while } E \text{ do } C$

$M ::= C$

Dominios Semánticos

BOOL={Verdadero, Falso}	valores de verdad
N={0, 1, 2, . . }	enteros no negativos

$v \in$ Valor = N + BOOL

$e \in$ Entrada = Valor

$sa \in$ Salida = Valor

$s \in$ Estado = Memoria X Entrada X Salida

$m \in$ Memoria = Ide -> [Valor + {noligado}]

Funciones Semánticas

$\mathcal{E}$: Exp -> (Estado -> [Valor X Estado + {error}])

$\mathcal{C}$: Com -> (Estado -> Estado+{error})

$\mathcal{M}$: Pro -> (Entrada -> Salida+{error})

$$\mathbb{M}[\![C]\!]e=(\mathbb{C}[\![C]\!](\langle\rangle,e,\langle\rangle)=(m',e',sa))\rightarrow sa,error$$

$$\mathbb{E}[\![E]\!]s=(0,s)$$

$$\mathbb{E}[\![E]\!]s=(1,s)$$

$$\mathbb{E}[\![\text{true}]\!]s=(\text{Verdadero},s)$$

$$\mathbb{E}[\![\text{false}]\!]s=(\text{Falso},s)$$

$$\mathbb{E}[\![\text{read}]\!]s=vacio(e)\rightarrow error,(cb(e),(m,cl(e),sa))$$

$$\mathbb{E}[\![I]\!](m,e,sa)=(m(I)=noligado)\rightarrow error,(m(I),(m,e,sa))$$

$$\mathbb{E}[\![\text{not } E]\!]s=(\ \mathbb{E}[\![E]\!]s=(v',s'))\rightarrow(esBOOL(v')\rightarrow$$
$$(No(v'),s'),error),error$$

$$\mathbb{E}[\![E_1=E_2]\!]s=(\mathbb{E}[\![E_1]\!]s=(v',s'))\rightarrow$$
$$(\ \mathbb{E}[\![E_2]\!]s'=(v'',s''))\rightarrow(v'=v'',s''),error),error$$

$$\mathbb{E}[\![E_1+E_2]\!]s=(\ \mathbb{E}[\![E_1]\!]s=(v',s'))\rightarrow$$
$$(\ \mathbb{E}[\![E_2]\!]s'=(v'',s''))\rightarrow$$
$$(esNum(v')\ y\ esNum(v''))\ \rightarrow(v'+v'',s''),error),error),error$$

$$\mathbb{C}[\![\text{output } E]\!]s=(\ \mathbb{E}[\![E]\!]s=(v,(m,e,sa)))\rightarrow(m,e,v.sa),error$$

$$\mathbb{C}[\![I:=E]\!]s=(\ \mathbb{E}[\![E]\!]s=(v,(m,e,sa)))\rightarrow(m[I/v],e,sa),error$$

$$\mathbb{C}[\![\text{if } E \text{ then } C_1 \text{ else } C_2]\!](s)=\mathbb{E}[\![E]\!]s=(v',s')\rightarrow$$
$$(esBOOL(v')\rightarrow(v'\rightarrow\mathbb{C}[\![C_1]\!]s',\mathbb{C}[\![C_2]\!])s'),error),error$$

$$\mathbb{C}[\![\text{while } E \text{ do } C]\!]s=(\ \mathbb{E}[\![E]\!]s=(v',s')\ )\rightarrow$$
$$(esBOOL(v')\rightarrow(v\rightarrow((\ \mathbb{C}[\![C]\!]s'=s'')\rightarrow$$
$$\mathbb{C}[\![\text{while } E \text{ do } C]\!]s'',error),s'),error),error$$

$$\mathbb{C}[\![C1;C2]\!]s=(\mathbb{C}[\![C1]\!]=error)\rightarrow error,\mathbb{C}[\![C2]\!](\mathbb{C}[\![C1]\!]s)$$

---

donde $\langle\rangle$ denota a la sucesión nula o a la memoria nula.

vacia:Entrada->BOOL no es aproximable, pero esto no tiene consecuencias pues es una función bien definida que únicamente se utiliza para definir una función, mientras que las siguientes son funciones aproximables:

cb:Entrada -> Valor

cl:Entrada -> Entrada

EsBOOL: Valor -> BOOL

No:BOOL -> BOOL

= :Valor X Valor -> BOOL

+ :NXN -> N

(además de otras como esNUM e "y")

Anteriormente discutimos cómo construir una definición semántica formal y ahora indicaremos los aspectos que deben tomarse en cuenta para interpretar un definición semántica dada. Para esto presentamos una ejemplo sencillo y trataremos de comprender la descripción semántica.

Dominios Sintácticos y Sintaxis Abstracta

$n \in Nml$	Notación decimal de enteros
$I \in Ide$	Identificadores
$E \in Exp$	Expresiones
$T, T_1, T_2 \in Com$	Comandos

E ::= true | false | n | I | -E

T ::= skip | abort | I:=E | T_1;T_2 | if E then T | while E do T

Dominios Semánticos

BOOL	Valores de verdad
Ent	Enteros

$e \in E = Ent + BOOL + \{error\}$ Valores expresables

$s \in S = (Ide \to E) + \{error\}$ Estados de memoria

Omitiremos las funciones semànticas para los enteros (para la función semàntica $\mathbb{N}$ ver ejemplo Capìtulo I).

$\mathbb{E} : Exp \to (S \to E)$

$\mathbb{C} : Com \to (S \to S)$

$\mathbb{E}[[\ n\ ]]s = \mathbb{N}[[\ n\ ]]$

$\mathbb{E}[[\ true\ ]]s = Verdadero$

$\mathbb{E}[[\ false\ ]]s = Falso$

$\mathbb{E}[[\ -E\ ]]s = esNum(\mathbb{E}[[\ E\ ]]s) \to -\mathbb{E}[[\ E\ ]]s, error$

$\mathbb{E}[[\ I\ ]]s = (s=error) \to error, s(I)$

$\mathbb{C}[[\ skip\ ]]s = s$

$\mathbb{C}[[\ abort\ ]]s = (s=\perp_s) \to \perp_s, error$

$\mathbb{C}[[\ I:=E\ ]]s = (si\ s \neq error\ y\ \mathbb{E}[[\ E\ ]]s \neq error)$

$s[I/\mathbb{E}[[\ E\ ]]s], error$

$\mathbb{C}[[\ T_1 ; T_2\ ]]s = \mathbb{C}[[\ T_2\ ]](\ \mathbb{C}[[\ T_1\ ]]s\ )$

$\mathbb{C}[[\ if\ E\ then\ T\ ]]s = (s \neq error\ y\ esBOOL(\ \mathbb{E}[[\ E\ ]]s)\ ) \to$

$(\mathbb{E}[[\ E\ ]]s \to \mathbb{C}[[\ T\ ]]s, s), error$

$\mathbb{C}[[\ while\ E\ do\ T\ ]]s = fij(\theta e)s$

donde $e = \mathbb{E}[[\ E\ ]]$ y

$\theta_e(s') = (s' \neq error\ y\ esBOOL(e(s'))\ ) \to$

$e(s') \to \theta_e(\mathbb{C}[[\ T\ ]]s',\ s'), error$

El problema consiste en interpretar la definición semàntica anterior. Primeramente observamos que se trata de un lenguaje que posee un enunciado de asignación, el cual, claramente es utilizado para modificar el valor asociado a un identificador, para representar esto, se define el estado de la memoria como una función de los identificadores en los valores o un estado de error. Lo que en el ejemplo anterior resultaba la memoria.

Así, los significados de comandos son simplemente funciones que transforman estados. Observemos que si se alcanza un estado de error entonces estas funciones lo preservan produciendo como resultado final un estado de error.

Ahora, ¿ que pasa con las expresiones?, de acuerdo a la función semántica $\mathbb{E}$, las expresiones dependen del estado de la máquina y esto concuerda con el hecho de que la expresiones pueden contener identificadores, ya que los significados de las expresiones están dados por una función de los estados en los valores.

Notamos también el uso del operador mínimo punto fijo para definir la cláusula de "while E do T". En este ejemplo los Dominios no son reflexivos y las funciones semánticas se contruyen de manera inductiva, por lo que la fundamentación matemática del ejemplo es más simple.

De la cláusula semántica de "skip", descubrimos que es un comando nulo, mientras que "abort" envía todos los estados a error salvo el estado nulo. Las definiciones de las otras cláusulas son sencillas y su interpretación es muy similar a la instrucciones análogas de TINY.

El lenguaje que presentamos ilustra aspectos de las definiciones semánticas formales, sin embargo es totalmente impráctico, pues no tiene comandos de entrada y salida.

Sección 18 Procedimientos, ambientes y saltos.

Continuando con nuestro desfile de ejemplos presentamos ahora un lenguaje con procedimientos (es decir un mecanismo para abstraer instrucciones o fragmentos de código).

El lenguaje no tiene tipos, efectos colaterales en expresiones, ni variables locales, ni saltos para alterar la secuencia de control. En próximos ejemplos incorporaremos estas ideas. La descripción del lenguaje es presentada a continuación:

Sintaxis Abstracta

$I \in Ide$	Identificadores
$E \in Exp$	Expresiones
$C \in Com$	Comandos
$M \in Pro$	Programas

E ::= 0 | 1 | -E | not E | E+E | E=E | I | procedure C | (E)

C ::= null | I:=E | call E | C;C | if E then C_1 else C_2 |
while E do C | begin C end

M ::= program(I); C .

Dominios Semánticos

T=BOOL	Valores de verdad
Ent={...-1,0,1,2,..}	Enteros
$b \in B = T + Ent$	Valores Básicos
$r \in R = B + P$	Valores almacenables
$s \in S = Ide \to (R+\{noligado\})$	Estados
$p \in P = S \to G$	Procedimientos
$e \in L = R + \{error\}$	Resultados de expresiones
$g \in G = S + \{error\}$	Resultados de Comandos
$A = B + \{error\}$	Respuestas (Resultados)

Funciones Semánticas

$\mathbb{E} : Exp \to (S \to E)$

$\mathbb{C} : Com \to (S \to G)$

$\mathbb{M} : Pro \to (B \to A)$

$\mathbb{E}[[\ 0\ ]]s=0 \qquad \mathbb{E}[[\ 1\ ]]s=1$

$\mathbb{E}[[\ -E\ ]]s=e?Z\to -e, error$ donde $e=\mathbb{E}[[\ E\ ]]s$

$\mathbb{E}[[\ not\ E\ ]]s=e?T\to No(e), error$ donde $e=\mathbb{E}[[\ E\ ]]s$

$\mathbb{E}[[\ E_1+E_2\ ]]s=e_1?Z$ y $e_2?Z \to e_1+e_2, error$

donde $e_i=\mathbb{E}[[E_i\ ]]s$ $i=1,2$

$\mathbb{E}[[\ I\ ]]s=s(I)?R\to s(I), error$

$\mathbb{E}[[\ procedure\ C\ ]]s=\mathbb{C}[[\ C\ ]]$

$\mathbb{E}[[\ (E)\ ]]s=\mathbb{E}[[\ E\ ]]s$

$\mathbb{C}[[\ null\ ]]s=s$

$\mathbb{C}[[\ I:=E\ ]]s=e?R\to s[I/e], error$ donde $e=\mathbb{E}[[E]]s$

$\mathbb{C}[[\ call\ e\ ]]s=e?P\to e(s), error$ donde $e=\mathbb{E}[[E]]s$

$\mathbb{C}[[\ C_1;C_2\ ]]s=g?S\to \mathbb{C}[[C_2\ ]]g, error$ donde $g=\mathbb{C}[[C_1\ ]]s$

$\mathbb{C}[[\ if\ E\ then\ C_1\ else\ C_2\ ]]s=$

$e?T\to(e\to \mathbb{C}[[C_1\ ]]s, \mathbb{C}[[\ C_2\ ]]s), error$

$\mathbb{C}[[\ while\ E\ do\ C\ ]]s=p(s)$

donde recursivamente $\forall\ s'$

$p(s')=e?T\to(e\to(g?S\to p(g), error), s'), error$

donde $e=\mathbb{E}[[\ E\ ]]s'$ y $g=\mathbb{C}[[\ C\ ]]s'$

$\mathbb{C}[[\ begin\ C\ end\ ]]s=\mathbb{C}[[\ C\ ]]s$

$\mathbb{M}[[\ program(I),C.\ ]]b=g?S$ y $g(I)?B\to g(I), error$

donde $g=\mathbb{C}[[\ C\ ]]s[I/b]$ y $\forall\ I'$, $s(I')=noligado$

Procedermos ahora a explicar la definición y la nueva notación introducida en ella.

De la definición sintáctica podemos observar que las formas de expresiones son: dos literales, "0" y "1", dos operaciones unarias ("-" y "not"), dos operaciones binarias ("+" e "="), identificadores globales, un mecanismo para señalar procedimientos sin parámetros, y encapsular con paréntesis. Los comandos son: el enunciado nulo, la asignación, la llamada a un procedimiento (call), ";" para control secuencial, "if" para composición selectiva de comandos y "while" para composición iterativa de comandos, mientras que "begin ... end" se utiliza para encapsular comandos.

A pesar de que la definición sintáctica presentada producirá una gramática ambigua consideramos resposibilidad del programador el adecuado uso de paréntesis y "begin...end" para agrupar y formar las instrucciones del programa.

Un programa consiste de un comando con un identificador para entrada y salida, la entrada del programa se utiliza como valor inicial del identificador y el valor de este identificador después de la ejecución del programa es la salida; observese la cláusula semántica para $\mathbb{M}$.

El siguiente es un ejemplo de un programa en este lenguaje:

```
program(x);
     begin
          y:=x;
          p:=(procedure x:=x+1);
          if x=y then x:=x+(-1) else x:=0;
          call p;
          call p
     end.
```

Para cualquier valor numérico como dato de entrada, el resultado del programa es el número más uno; sin embargo durante la ejecución del programa el valor de x es decrementado y luego incrementado dos veces. Un valor de verdad como dato de entrada produce un error asi como cualquier otro dato cuyo tipo no sea "número entero".

La afirmación correspondiente al parrafo anterior puede probarse formalmente a partir de la definición semántica presentada, en el próximo ejemplo veremos una prueba.

Como es constumbre en las definiciones semánticas, después de la sintaxis abstracta que define los Dominios Sintácticos, aparecen los Dominios Semánticos, definidos por una ecuación de Dominios; junto al Dominio Semántico, usualmente aparece un símbolo distintivo para denotar elementos de dicho Dominio, por ejemplo:

$$b \in B = T + Z \; . \quad \text{valores básicos}$$

especifica que "b" es un símbolo que siempre denota un valor básico, en este caso, puede ser un valor de verdad o un entero. Los estados de memoria son principalmente usados para definir las asignaciones. En este lenguaje los identificadores pueden ser asociados directamentes con los valores almacenados (cuando trabajemos ambientes y variables locales y globales necesitaremos manejar celdas o localidades de memoria).

De esta manera los estados de memoria son simulados por funciones de los identificadores en los valores almacenados, es decir

$$s \in S = Ide \rightarrow (R + \{noligado\})$$

por lo que para cada identificador I, s(I) es el valor de I en el estado s. Un identificador no iniciado o no usado es asociado al valor especial "noligado", de manera que el estado inicial tiene todos los identificadores (salvo el señalado como entrada y salida) asociados a "noligado".

Los procedimientos se simulan como funciones aproximables de los estados en los estados, (o error),

$$p \in P = S \rightarrow G \quad \text{procedimientos}$$

Si p es un procedimiento y s un estado, entonces p(s) es el resultado de invocar el procedimiento p en el estado s, es decir, s es el estado inicial para los cómputos del procedimiento. Obsérvense las definiciones para los Dominios Semànticos, y se notarà que son mutuamente recursivas pues R està en tèrminos del Dominio P, P està en tèrminos de S y S en tèrminos de R.

Existen tres funciones semànticas $\mathbb{E}$ para expresiones, $\mathbb{C}$ para comandos y $\mathbb{M}$ para programas. Como una expresiòn nos da un valor (o error) al evaluarse con respecto a un estado y no produce efectos colaterales, $\mathbb{E}$:Exp->(S->L) entonces si E es una expresiòn $\mathbb{E}$[[E]]s es un error o el valor relativo a s. En la definiciòn anterior asumimos que -> se asocia a la derecha para escribir

$$\mathbb{E} : Exp \rightarrow S \rightarrow E$$

Ejecutar un comando con respecto a un estado s produce un nuevo estado, o error, por esto definimos

$$\mathbb{C} : Com \rightarrow S \rightarrow G$$

y $\mathbb{C}$[[C]]s es el resultado de ejecutar C relativo al estado s.

Finalmente, ejecutar un programa dado un valor bàsico (o en un valor bàsico) nos da una respuesta, es decir un error (que es consecuencia de la apariciòn de algùn error durante la ejecuciòn) o un valor bàsico, por eso

$$\mathbb{M}[[\ M\ ]]b$$

es la respuesta o salida de ejecutar el programa M con el dato b.

Las ecuaciones que definen las clàusulas semànticas hacen uso de funciones entre construcciones de T y Z como NO, +,- ,= etc. las cuales hemos ya explicado con suficiente detalle.

Como puede notarse, frecuentemente es necesario preguntar a que componente de una uniòn ajena pertenece un elemento, para abreviar estas funciones utilizamos un prediacdo especial .?D donde D es la componente de la uniòn ajena, por ejemplo en:

$$B = T + Z$$

tenemos

que se define para $b \in B$ como

$$b?T = \begin{cases} \text{Verdadero} & \text{si } b \text{ esta en la imagen de la inclusiòn de T en B} \\ \text{Falso} & \text{en otro caso} \end{cases}$$

es decir .?T corresponde a la funciòn que anteriormente llamàbamos esBOOL.

Como otra observaciòn a la definiciòn, apuntamos que si somos estrictos debemos utilizar proyecciones e inclusiones para manejar los valores de uniones ajenas, es decir, el esquema

$$B = T + Z$$

deberìa especificarse en la definiciòn. A pesar de esto omitimos su aclaraciòn pues las incluciones y proyecciones correspondientes son fàcilmente inferibles del contexto y el incluirlas nos forzarìa a utilizar suficiente notaciòn para oscurecer considerablemente la descripciòn semàntica.

Tambièn hemos incluido en la definiciòn anterior la notaciòn auxiliar

... donde ...

la cual puede utilizarse para simplificar las clàusulas semànticas, ademàs, obsèrvese que como las expresiones no tienen efectos colaterales es irrelevante el orden de evaluaciòn en E_1+E_2 de E_1 y E_2 (no asì en TINY) .

Asì mismo, incluimos en la clàusula de "while E do C" la palabra "recursivamente" para indicar una definiciòn recursiva cuyo significado està especificado por el operador mìnimo punto fijo.

Nòtese que, la apariciòn de un error se propaga a travès de ejecuciones sucesivas de comandos, esto hace que el resultado de un programa sea error si en algùn momento se produce un error.

Es importante abundar en el manejo de los procedimientos. La declaraciòn de un procedimiento se evalua, simplemente aplicando Π al cuerpo del procedimiento y sin suministrar ningún estado, produciéndose, como resultado de la evaluación, una función de los estados en los resultados de estados; la evaluación correspondiente puede también escribrise como:

$$\mathbb{E}[\![\text{ procedure } C\ ]\!]s=p \text{ donde } \forall\ s'\ p(s')=\mathbb{C}[\![C]\!]s'$$

lo cual aclara que el estado s' de invocación del procedimiento no tiene por que ser el mismo que el estado s de su declaración. Notese que la cláusula para una llamada a un procedimiento define el significado de la estructura en términos del valor de sus constituyentes inmediatos sin hacer referencia a la declaración del procedimiento.

Ambientes

Extenderemos el lenguaje presentado en los parrafos anteriores para incluir "declaraciones". Supongamos que la sintaxis es aumentada para incluir dos formas de declaraciones y una nueva forma de bloque de comandos asi:

$D \in Dcl$ Declaraciones

D ::= new I=E | var I=E

C ::= ... | with D do C

La declaración "new" es una declaración con valor inicial, es decir, la ecuación "new I=E" especifica que si no ha ocurrido un estado de error, el ambiente y estados nuevos difieren de los anteriores en que I está ligado a alguna localidad de memoria no usada antes y la localidad contiene el valor de E; el efecto que produce la declaración "val" es únicamente ligar el identificador permanentemente con el valor de la expresión, modificando el ambiente, como si se tratase de una declaración CONST en PASCAL. El lenguaje que describiremos seguirá la liga estática y las convenciones de ambiente como en PASCAL.

Por ejemplo, el resultado del siguinete programa es 0 y no 1.

```
Program(x);
     with val a=0 do
          with val p=procedure x:=a do
               with val a=1 do
                    call p.
```

Sugerimos ver Stoy[80] donde aparece un ejemplo donde la semántica definida sigue la liga dinámica.

La definición formal tiene ahora las siguientes diferencias:

Aparece un Dominio U de Ambientes, un ambiente, es una función que mapea identificadores, en los valores que denotan.

Los estados de memoria, mapean ahora elementos de un Dominio L de localidades o celdas de memoriá en los valores que contienen.

Obsèrvese que el Dominio de los valores denotables D, es decir, aquellos valores denotables por los identificadores es distinto de los valores almacenables R en las localidades de memoria. Este tipo de diferencia existe en casi todos los lenguajes de programación usados comercialmente.

Las funciones semànticas para expresiones, declaraciones y comandos son ahora definidas de manera que las construcciones sintàcticas son interpretadas con respecto a un ambiente así como a un estado, por ejemplo

$$\mathbb{C}[[C]]u$$

es la transformación de estados denotada por el enunciado C en el ambiente u, y por lo tanro

$$\mathbb{C}[[C]]\ u\ s$$

es el resultado (estado) de aplicar la transformación de estados al estado s, es decir, el estado resultante o error.

El interpretar una declaración produce un nuevo ambiente y un nuevo estado.

La descripciòn formal es la siguinete:

Sintaxis Abstracta

$I \in Ide$	Identificadores
$E \in EXp$	Expresiones .
$D \in Dcl$	Declaraciones
$C \in Com$	Comandos
$M \in Pro$	Programas

$E ::= 0 \mid 1 \mid -E \mid \text{not } E \mid E_1 + E_2 \mid E_1 = E_2 \mid I \mid \text{procedure } C \mid (E)$

$C ::= \text{null} \mid I := E \mid \text{call } E \mid C_1 ; C_2 \mid \text{if } E \text{ then } C_1 \text{ else } C_2$

$\quad \text{while } E \text{ do } C \mid \text{begin } C \text{ end} \mid \text{with } D \text{ do } C$

$D ::= \text{new } I = E \mid \text{val } I = E$

$M ::= \text{program}(I);\ C\ .$

Dominios Semànticos

$k \in L$	Localidades
$b \in B = BOOL + N$	
$r \in R = B + P$	Valores almacenables
$s \in S = L \rightarrow (R + \{noligado\})$	Estados
$p \in P = S + G$	Procedimientos
$d \in D = L + R + \{nodefinido\}$	Valores denotables
$u \in U = Ide \rightarrow D$	Ambientes
$e \in E = R + \{error\}$	Resultados de expresiones
$g \in G = S + \{error\}$	Resultados de comandos
$A = B + \{error\}$	Resultados de programas

Funciones Semàntices

$\mathbb{E}: Exp \rightarrow U \rightarrow S \rightarrow E$

$\mathbb{D}: Dcl \rightarrow U \rightarrow S \rightarrow (U \times G)$

$\mathbb{C}: Com \rightarrow U \rightarrow S \rightarrow G$

$\mathbb{M}: Pro \rightarrow B \rightarrow A$

IE[[0]]u s =0   IE[[1]]u s=1

IE[[not E]]u s=e?T->No(E), error   donde e=IE[[E]]u s

IE[[-E]]u s = e?T-> -e, error   donde e=IE[[E]]u s

IE[[E$_1$ +E$_2$]]u s=e$_1$?Z y e$_2$?Z->e$_1$ +e$_2$,error donde e$_i$ =IE[[E$_i$]]u s

IE[[E$_1$ =E$_2$]]u s=e$_1$?B y e$_2$?B->e$_1$ =e$_2$,error donde e$_i$ =IE[[E$_i$]]u s

IE[[I]]u s=d?L->s(d),d?R->d,error   donde d=u(I)

IE[[procedure C]]u s=IC[[C]]u

IE[[(E)]]u s=IE[[E]]u s.

ID[[new I=E]]u s=e?R y existe k L 't' s(k)=noligado->

(u[I/k],s[k/e]),(u,error)   donde e=IE[[E]]u s

ID[[val I=E]]u s=e?R->(u[I/e],s),(u,error)   donde e=IE[[E]]u s

IC[[null]]u s=s

IC[[I:=E]]u s=d?L y e?R->s[d/e],error donde d=u(I) y e=IE[[E]]u

IC[[call E]]u s=e?P->e(s),error   donde e=IE[[E]]u s

IC[[C$_1$;C$_2$]]u s=g?S->IC[[C$_2$]]u g,error   donde g=IC[[C$_1$]]u s

IC[[if E then C$_1$ else C$_2$]]u s=e?T->

(e->IC[[C$_1$]]u s,IC[[C$_2$]]u s),error donde e=IE[[E]]u s

IC[[while E do C]]u s=p(s) donde recursivamente ∀s'

p(s')=e?T->(e->(g?S->p(g),error),s'),error

donde e=IE[[E]]u s y g=IC[[C]]s'

IC[[begin C end]]u s=IC[[C]]u s

IC[[with D do C]]u s=g?S->IC[[C]]u' g,error

donde (u',g)=ID[[D]]u s

IM[[program(I);C.]]b=g?S y g(k)?B->g(k), error

donde g=IC[[C]](u[I/k])(s[k/b])

y donde ∀ I' u(I')=nodefinido

y donde ∀ k' s(k')=noligado.

Es importante mencionar que:

1)Al especificar que cualquier localidad de memoria no usada puede ser tomada para un nuevo ambiente, estamos evitando las consideraciones correspondientes al manejo de memoria.

2)Para un bloque declarado con "with D do C" un nuevo ambiente es creado para su cuerpo, pero otro tipo de estructuras y expresiones heredan el ambiente dado para su evaluación. En particular esto es cierto para la declaración de procedimientos de manera que los identificadores del procedimiento son ligados en el contexto de su definición y no en el de su invocación, por lo que la ejecución sigue la liga estática.

3)La evaluación de un identificador I bajo IE incluye una prueba para verificar si denota un valor o una localidad. Si I denota una localidad, el valor en la localidad es regresado. De esta forma, IE siempre produce el valor básico almacenable asociado a I (o error).

Por ejemplo si ejecutamos el programa antes mencionado tenemos:

Paso 1)

```
IM[[ Program(x);
     with val a=0 do
          with val p=procedure x:=a do
               with val a=1 do
                    call p. ]]b
```

=g?S y g(k)?B->g(k),error

```
donde g=IC[[ with val a=0
               with val p=procedure x:=a do
                    with val a=1 do
                         call p ]](u[x/k])(s[k/b])
```

donde ∀ I'∈Ide u(I')=indefinido y ∀ k'∈L s(k')=noligado.

Paso 2)

Analizamos

```
IC[[ with val a=0 do
        with val p=procedure x:=a do
           with val a=1 do
                call p ]](u[x/k])(s[k/b])
=g'?S->IC[[ with cal p=procedure x:=a do
                with val a=1 do
                     call p . ]]u' g',error
donde(u',g')=ID[[val a=0]](u[x/k])(s[k/b])
ahora
ID[[val a=0]](u[x/k])(s[k/b])
=e?R->(u[x/k][a/e],s[k/b]),(u[x/k],error)
donde e=IE[[0]](u[x/k])(s[k/b])
pero IE[[0]](u[x/k])(s[k/b])=0
∴ ID[[val a=0]](u[x/k])(s[k/b])=(u[x/k][a/0],s[k/b])
Paso 3)
IC[[ with val p=procedure x:=a do
           with val a=1 do
                callp]]u[x/k][a/0] s[k/b]
=g'?S->IC[[with val a=1 do
                call p ]]u' g',error
donde
(u',g')=ID[[ val p=procedure x:=a ]]u[x/k][a/0] s[k/b]
pero
ID[[val p=procedure x:=a]]u[x/k][a/0] s[k/b]
=e?R->(u[x/k][a/0][p/e],s[k/b]),(u[x/k][a/0],error)
donde e=IE[[procedure x:=a]]u[x/k][a/0] s[k/b]
ahora
IE[[procedure x:=a]]u[x/k][a/0] s[k/b]=IC[[x:=a]]u[x/k][a/0]
∴ ID[[val p=procedure x:=a]]u[x/k][a/0] s[k/b]
       =(u[x/k][a/0][p/]C[[x:=a]]u[x/k][a/0] ],s[k/b])
```

Paso 4)

IC[[with val a=1 do

call p]]u[x/k][a/0][p/IC[[x:=a]]u[x/k][a/0]] s[k/b]

=g'?S->IC[[call p]]u' g',error

donde

(u',g')=ID[[val a=1]]u[x/k][a/0][p/IC[[x:=a]]u[x/k][a/0]] s[k/b]

ahora

ID[[val a=1]]u[x/k][a/0][p/IC[[x:=a]]u[x/k][a/0]] s[k/b]

=e?R=>(u[x/k][a/0][p/IC[[x:=a]]u[x/k][a/0]][a/e],s[k/b])

,(u[x/k][a/0][p/IC[[x:=a]]u[x/k][a/0]],error)

donde

e=IE[[1]]u[x/k][a/0][p/]C[[x:=a]]u[x/k][a/0]] s[k/b]

por lo que e=1 entonces

paso 5)

IC[[call p]]u[x/k][a/0][p/IC[[x:=a]]u[x/k][a/0]][a/1] s[k/b]

=IC[[call p]]u[x/k][a/1][p/]C[[x:=a]]u[x/k][a/0]] s[k/b]

=e?P->e(s),error

donde

e=IE[[p]]u[x/k][a/1][p/IC[[x:=a]]u[x/k][a/0]] s[k/b]

ahora

]E[[p]]u[x/k][a/1][p/]C[[x:=a]]u[x/k][a/0]] s[k/b]

=d?L->s(d),d?R->d,error donde d=u(p)

entonces d=IC[[x:=a]]u[x/k][a/0]

por lo que

IE[[p]]u[x/k][a/1][p/IC[[x:=a]]u[x/k][a/0]] s[k/b]

=IC[[x:=a]]u[x/k][a/0]

entonces

IC[[call p]]u[x/k][a/1][p/IC[[x:=a]]u[x/k][a/0]] s[k/b]

=IC[[x:=a]]u[x/k][a/0] s[k/b]

=d?L y e?R-) s[k/b][d/e],error donde

d=u(x) y e=IE[[a]]u[x/k][a/0] s[k/b]

pero entonces d=k y]E[[a]]u[x/k][a/0] s[k/b]=0 pues u(a)=0

∴ IC[[call p]]u[x/k][a/1][p/IC[[x:=a]]u[x/k][a/0]] s[k/b]

=s[k/b][k/0]=s[k/0]

∴ IC[[with val a=1 do

call p]]u[x/k][a/0][p/IC[[x:=a]]u[x/k][a/0]] s[k/b]

=s[k/0]

∴ IC[[with val p=procedure x:=a do

with val a=1 do

call p]]u[x/k][a/0] s[k/b] = s[k/0]

∴ IC[[with a=0 do

with val p=procedure x:=a do

with val a=1 do

call p]]u[x/k] s[k/b]=s[k/0]

∴ IM[[program(x);

with val a=0 do

with val p=procedure x:=a do

with val a=1 do

call p]]b=s[k/0](k)=0

[].

Lo hecho anteriormente puede no ser la manera más directa de seguir la ejecución pero sí la más pausada y acorde a la definición. Esta ejecución prueba que no importa que dato se proporcione al programa, éste da como resultado el valor cero. Esperamos que el lector pueda comprobar la demostración comparando con la tabla de la descripción semántica formal.

Brincos o Saltos incondicionales

La semántica de los saltos es descrita en términos del concepto de "continuación"; la "continuación" de un cómputo, cálculo o operación en la máquina es cualquier sucesión de nuevos cómputos que puede seguir al primer cálculo y formalmente se expresa como una función de los resultados esperados del cálculo en el conjunto de posibles cómputos subsecuentes. Por ejemplo, el resultado esperado de un comando es un nuevo estado, entonces la "continuación" de un comando puede representarse por una función de los posibles estados producidos por el comando en las sucesiónes de comandos que podrían seguir a la ejecución del primer comando. Es decir, la continuación de la ejecución de un comando es la sucesión de operaciones que le sigue expresada como función del estado resultante de la ejecución.

Como un ejemplo más, considerese la ejecución de un comando de la forma:

$$C_1 ; C_2$$

la continuación a la ejecución de C_1 comienza por la ejecución de C_2 relativa al estado resultante de la ejecución de C_1. La ejecución de C_2 tiene como continuación, la continuación del comando $C_1 ; C_2$ es decir lo que puede seguir a C_2 es exactamente lo que puede seguir a $C_1 ; C_2$, mientras que lo que puede seguir a la ejecución de C_1 es únicamente la ejecución de C_2 con respecto al estado resultante de la ejecución de C_1.

La continuación a la evaluación de una expresión depende del valor y estados resultantes de la evaluación. Por ejemplo, en la ejecución de un comando de la forma:

$$\text{if } E \text{ then } C_1 \text{ else } C_2$$

la continuación (lo que puede seguir) a la evaluación de la expresión E es la ejecución de C_1 o C_2, y la discriminación entre estas posibles continuaciones depende del valor de verdad producido por la evaluación de E. Es claro que C_1 o C_2 (el seleccionado) hereda como continuación la continuación de "if E then C_1 else C_2", es decir que todo lo que sigue de C_1 es únicamente lo que sigue de "if E then C_1 else C_2".

En realidad pensamos como la continuación de alguna operación como todas las operaciones que siguen a la ejecución de la operación hasta la terminación de la ejecución del programa, obsérvese que pueden existir continuaciones infinitas que nunca terminan, las cuales corresponderán a elementos infinitos del Dominio de Continuaciones.

Similarmente a la evaluación de las expresiones, la continuación a interpretar una declaración es una operación que depende del ambiente y estado que la declaración produce. Por ejemplo, en la ejecución de un bloque como:

with D do C

la interpretación de D tiene como continuación la ejecución de C con respecto al ambiente y estado resultante de la interpretación.

Para analizar los saltos, consideramos por ejemplo el salto

goto N

en PASCAL. Si queremos describir el efecto de la instrucción en términos del significado de N, que es su único constituyente inmediato, podemos asumir que la etiqueta N denota una continuación de comando que corresponde a proseguir la ejecución en el punto etiquetado.

Por ejemplo, si tenemos

```
begin
  :
13 : x:=x+1;
  :
end.
```

la etiqueta "13" denota la continuación que comienza por ejecutar "x:=x+1" y prosigue la continuación de este comando.

Obsèrvese que el concepto de continuación es una abstración del concepto sintàctico de punto en el programa, ya que dos puntos distintos en el programa pueden representar la misma continuación, es decir, dos ejecuciones a partir de dos puntos son indinstinguibles si se obtiene siempre el mismo resultado para cualquier estado inicial. Por ejemplo si tenemos:

12: null ; 13: x:=x+1; ...

tanto "12" como "13" especifican la misma continuación pues para cualquier estado s, la ejecución a partir de "12" producirà el mismo resultado que realizar la ejecución desde "13", pero las etiquetas se encuentran en puntos distintos del programa.

De la terminología anterior, el efecto del comando

goto N

puede describirse como seguir la continuación denotada por N utilizando el estado actual. De esta forma, la continuación normal (la determinada por el contexto) es ignorada, por ejemplo en:

goto N; C

la continuación normal de goto N que comienza por la ejecución de C es ignorada. Esta es una característica de los saltos porque siempre ignoran la continuación determinada por el contexto y prosiguen con otra continuación.

El "goto N" tiene además dos propiedades importantes:

a) La continuación del goto se obtiene evaluando el destino explícito señalado por su conponente inmediato N, y

b) La continuación del goto es la continuación de un comando.

Introduciremos ahora continuaciones en nuestras descripciones formales.

En los ejemplos anteriores, las funciones semànticas regresaban resultados locales intermedios a la ejecución como valores o estados temporales. Definiremos las funciones semànticas de manera que siempre regresen un resultado global a todo el programa. Cada función semántica se define relativa a una continuación como argumento además de un ambiente y estado. La continuación especifica que debe hacerse con el resultado inmediato para que el programa produzca una respuesta (siempre que no haya error o salto).

Por ejemplo, la continuación de un comando especifica que acciones pueden seguir a la ejecución del comando, como una función de los resultados de ejecución. De esta forma, la continuación de un comando es una función cuyo argumento es un estado y que regresa una respuesta (i.e. resultado de programa).

$$c \in C = S \rightarrow A \quad \text{continuaciones de comandos}$$

donde S es el Dominio de estados y A es el Dominio de respuestas de programa. Entonces la función semántica para los comandos será

$$\mathbb{C}: Com \rightarrow U \rightarrow C \rightarrow S \rightarrow A$$

donde U es el Dominio de Ambientes. $\mathbb{C}[[C]]u\ c\ s$ es la respuesta calculada por el programa donde C es un comando. Si la ejecución de C provee un nuevo estado s', entonces el argumento c será aplicado a s' para producir una respuesta final al programa.

Por ejemplo, la cláusula semántica para el comando nulo es:

$$\mathbb{C}[[null]]u\ c\ s = c(s)$$

esto dice que la respuesta a ejecutar un comando nulo con continuación c relativo al estado s se obtiene aplicando (siguiendo) la continuación c al estado s. Es decir, la ejecución del programa debe continuar con la continuación dada sin ningún cambio en el estado.

Obsérvese que los objetos que resultan de la cláusula semántica anterior son respuestas, resultados de programas, y no resultados intermedios.

Un ejemplo de una cláusula semántica en la cual una continuación es definida y usada es:

$\mathcal{C}[[C_1 ; C_2]]u\ c\ s = \mathcal{C}[[C_1]]u\ c'\ s$

donde $\forall s'$, $c'(s') = \mathcal{C}[[C_2]]u\ c\ s'$

esto especifica que la composición secuencial de los comandos C_1 y C_2 debe ejecutarse como sigue: La respuesta de ejecutar $C_1;C_2$ con continuación c en el estado s es la misma respuesta que ejecutar C_1 con la continuación c' en el estado s, donde c' es ejecutar C_2 y luego la continuación c dada; la especificación corresponde a que C_1 es primeramente ejecutado relativo a la continuación c' que si se aplica a un estado s' produce la respuesta obtenida por C_2 ejecutado relativo a la continuación c. Esto dide que C_2 junto con la continuación c de la estructura especifican la continuación de C_1.

Nótese que no es necesario preguntar si un estado es error y entonces propagarlo como fue hecho en ejemplos precedentes. Cuando se produce un error, la respuesta de programa "error" es automaticamente seleccionada y todo otra respuesta es ignorada. Un ejemplo de este tipo de manejo de errores puede recalcarse en la cláusula semántica de la asignación:

$$\mathcal{C}[[\ I:=E\ ]]u\ c\ s = d?L \rightarrow c(s[d/r]), \text{error donde } d=u(I)$$

cuya interpretación nos dice que si el identificador denota una localidad, la respuesta del programa se obtiene siguiendo la continuaión c con el estado actualizado, de lo contrario la respuesta del programa es error.

Los mismos principios se aplican a otras clases sintácticas. En nuestro lenguaje de programación particular el Dominio de continuaciones de expresiones será

$$k \in K = R \rightarrow A$$

Similarmente, el Dominio de continuaciones de declaraciones estará dado por

$$q \in Q = U \rightarrow S \rightarrow A$$

Por ejemplo, la cláusula semántica

$\mathbb{D}[[val\ I=E]]u\ q\ s = \mathbb{E}[[\ E\ ]]u\ k\ s$

donde $\forall$ r, $k(r)=q(u[I/r])(s)$

especifica que una declaración "val" es interpretada evaluando la expresión y suministrando el nuevo ambiente y estado a la continuación de la declaración q.

En la definición que presentamos a continuación, algunas cláusulas han sido simplificadas cancelando los argumentos de las funciones. Por ejemplo, para la composición secuencial de comandos tenemos

$\mathbb{C}[[C_1;C_2]]u\ c = \mathbb{C}[[C_1]]u\ c'$ donde $c'=\mathbb{C}[[C_2]]u\ c$

que expresa la igualdad de funciones aproximables (elementos del Dominio C).

Similarmente la ejecución para el bloque aparece como:

$\mathbb{C}[[with\ D\ do\ C]]u\ c=\mathbb{D}[[D]]u\ q$

donde $\forall$ u', $q(u')=\mathbb{C}[[C]]u'\ c$

lo cual explica porque es conveniente considerar los Dominios como U->S->A y no (UXS)->A aunque sean isomorfos.

Con continuaciones en el modelo semántico es posible incluir saltos en el lenguaje del ejemplo anterior. La sintaxis puede aumentarse agregando un comando etiquetado y un salto incondicional así:

S€Slt

S ::= goto I

C ::= ... | I:C | S

En el lenguaje que presentaremos, el alcance de la etiqueta en el comando

I:C

será restringida al comando mismo.

Como un ejemplo de un programa en este lenguaje tenemos el siguiente que responderá con el valor 1 a cualquier entrada después de hacer un ciclo interno.

```
program(x);
     begin
               x:=0;
         loop: if not(x=1) then
                    begin
                         x:=x+1;
                         goto loop
                    end
               else
                    null
     end.
```

La semántica es descrita formalmente ampliando el Dominio D para incluir continuaciones de comandos:

$d \in D = L+R+C+\{nodefinido\}$ valores denotables

y

$\mathbb{C}[[I:C]]u\ c=c'$

donde recursivamente

$c'=\mathbb{C}[[\ C\ ]](u[I/c'])\ c$

lo cual especifica que la ejecución de C, relativa a un ambiente en el cual la etiqueta está ligada a una continuación, comienza por otra ejecución de C, es decir, C esta incluido entre lo que puede seguir a I:C . Esta definición recursiva, como ya hemos señalado corresponde al mínimo punto fijo.

Aparece ahora una nueva función semántica

$\mathbb{S}$: Slt -> U -> S -> A

y entonces

$\mathbb{C}[[\ S\ ]]u\ c= \mathbb{S}[[\ S\ ]]u$

donde la cláusula para el salto es:

$\mathbb{S}$[[goto I]] u s=d?C->d(s),error donde d=u(I)

de manera que la continuación c del comando S es ignorada y el efecto del "goto" es seguir la continuación denotada por la etiqueta. Obsérvese que por esto, la función semántica $\mathbb{S}$ para los saltos no requiere de un argumento con valor en las continuaciones.

Sintaxis Abstracta

I∈Ide	Identificadores
E∈Exp	Expresiones
C∈Com	Comandos
D∈Dcl	Declaraciones
S∈Slt	Saltos
M∈Pro	Programas

E ::= 0 | 1 | -E | not E | E_1 +E_2 | E_1 =E_2 | I | procedure C | (E)

C ::= null | I:=E | call E | C_1 ;C_2 | if E then C_1 else C_2 |
while E do C | begin C end | with D do C | I:C | S

S ::= goto I

M ::= program(x); C.

Dominios Semánticos

T=BOOL	Valores de Verdad
Z={0,1,-1,2,-1,..}	Enteros
b ∈ B=T + Z	Valores básicos
r ∈ R=B + P	Valores almacenables
d ∈ D=L + R + C + {nodefinido}	Valores denotables
s ∈ S=L->(R + {noligado})	Memorias
u ∈ U=Ide -> D	Ambientes
c ∈ C=S -> A	Continuaciones de Comandos

$k \in K = R \to A$ Continuaciones de expresiones

$q \in Q = U \to S \to A$ Continuaciones de Definiciones

$p \in P = C \to S \to A$ Procedimientos

$A = B + \{error\}$ Respuestas

Funciones semánticas

$\mathbb{E}: Exp \to U \to K \to S \to A$

$\mathbb{D}: Dcl \to U \to Q \to S \to A$

$\mathbb{C}: Com \to U \to C \to S \to A.$

$\mathbb{S}: Slt \to U \to S \to A$

$\mathbb{M}: Pro \to B \to A$

$\mathbb{E}[[\ 0\ ]]u\ k\ s = k(0)$ $\quad \mathbb{E}[[\ 1\ ]]u\ k\ s = k(1)$

$\mathbb{E}[[\ not\ E\ ]]u\ k\ s = \mathbb{E}[[\ E\ ]]u\ k'\ s$

donde $\forall\ r,\ k'(r) = r?T \to k(No(r)), error$

$\mathbb{E}[[\ -E\ ]]u\ k\ s = \mathbb{E}[[\ E\ ]]u\ k'\ s$

donde $\forall\ r,\ k'(r) = r?Z \to k(-r), error$

$\mathbb{E}[[\ E_1 + E_2\ ]]u\ k\ s = \mathbb{E}[[\ E_1\ ]]u\ k_1\ s$

donde $\forall\ r_1,\ k_1(r_1) = r_1\ ?Z \to \mathbb{E}[[\ E_2\ ]]u\ k_2\ s, error$

y $\forall\ r_2,\ k_2(r_2) = r_2\ ?Z \to k(r_1 + r_2), error$

$\mathbb{E}[[\ E_1 = E_2\ ]]u\ k\ s = \mathbb{E}[[\ E_1\ ]]u\ k_1\ s$

donde $\forall\ r_1,\ k_1(r_1) = r_1\ ?B \to \mathbb{E}[[\ E_2\ ]]u\ k_2\ s$

y $\forall\ r_2,\ k_2(r_2) = r_2\ ?B \to k(r_1 = r_2), error$

$\mathbb{E}[[\ I\ ]]u\ k\ s = d?L \to k(s(d)), d?R \to k(d), error$ donde $d = u(I)$

$\mathbb{E}[[\ procedure\ C\ ]]u\ k\ s = k(\mathbb{C}[[\ C\ ]]u)$

$\mathbb{E}[[\ (E)\ ]]u\ k\ s = \mathbb{E}[[\ E\ ]]u\ k\ s$

$\mathbb{D}[[\ new\ I = E\ ]]u\ q\ s = \mathbb{E}[[\ E\ ]]u\ k\ s$

donde $\forall\ r,\ k(r) =$ si existe $t \in L$ `t` $s(t) = noligado \to$

$q(u[i/t])(s[t/r]), error$

$\mathbb{D}[[\ val\ I = E\ ]]u\ q\ s = \mathbb{E}[[\ E\ ]]u\ k\ s$

```
    donde ∀ r, k(r)=q(u[I/r]) s

IS[[ goto I ]]u s=d?C->d(s),error          donde d=u(I)

IC[[ null ]]u c s=c(s)

IC[[ I:=E ]]u c s=IE[[ E ]]u k s

    donde ∀ r, k(r)=d?L->c(s[d/r]),error     y d=u(I)

IC[[ call E ]]u c s=IE[[ E ]]u k s

    donde ∀ r, k(r)=r?P->r(c)(s),error

IC[[ C ;C  ]]u c s=]C[[ C  ]]u c' donde c'=IC[[ C  ]]u c
      1  2                1                        2
IC[[ if E then C  else C  ]]u c s=IE[[ E ]]u k  s
                1       2                      s
  donde ∀ r, k (r)=r?T->(r->IC[[ C  ]]u c s,IC[[ C  ]]u c s),error
              s                  1               2
IC[[ while E do C ]]u c s]]=c'(s)

    donde recursivamente c'(s')=]E[[ E ]]u k   s'
                                           s'
    y ∀ r, k  (r)=r?T->(r->IC[[ C ]]u c' s',c(s')),error
            s'
IC[[ with D do C ]]u c s=ID[[ D ]]u g s

    donde q(u')(s')=IC[[ C ]]u' c s'

IC[[ begin C end ]]u c s=IC[[ C ]]u c s

IC[[ S ]]u c=IS[[ S ]]u

IC[[ I:C ]]u c= c'      donde recursivamente c'=IC[[ C ]](u[I/c'])c

IM[[ program(x); C. ]]b=IC[[C]](u[I/t]) c s[t/b]

    donde ∀ I' u(I')=nodefinido

    ∀ t' s(t')=noligado

 y   ∀ s', c(s')=s'(t)?B->s'(t),error
```

Recomendamos al lector seguir la ejecución de los tres últimos programas presentados con respecto a la última definición y observar el manejo de los distintos elementos para comprender mejor la definición.

Sección 19 Un lenguaje funcional

Presentamos ahora la definición formal de un lenguaje de programación de tipo funcional y no imperativo como los que hemos presentado hasta el momento.

Sintaxis Abstracta

$B \in Bas$	Notación valores básicos
$I \in Ide$	Identificadores
$E \in Exp$	Expresiones
$D \in Dcl$	Declaraciones

$$E ::= B \mid I \mid \text{fn } I{:}E \mid E_1 (E_2) \mid \text{if } E_1 \text{ then } E_2 \text{ else } E_3 \mid E \text{ where } D$$
$$\mid E \text{ whererec } D \mid (E)$$

$$D ::= I{=}E \mid D_1 \text{ and } D_2$$

Dominios semánticos

$b \in B$	Valores Básicos
$e \in E = B + F$	Valores expresables
$f \in F = D \to E$	Funciones
$d \in D = E$	Valores denotables
$u \in U = Ide \to (D + \{error\})$	Ambientes

Funciones semánticas

$\mathbb{B} : Bas \to B$

$\mathbb{E} : Exp \to U \to E$

$\mathbb{D} : Dcl \to U \to U$

$\mathbb{B}[[\ B\ ]]=$ valor que denota B

$\mathbb{E}[[\ B\ ]]u=\mathbb{B}[[\ B\ ]]$

$\mathbb{E}[[\ I\ ]]u=u(I)$

$\mathbb{E}[[\ \text{fn } I{:}E\ ]]u=f$ donde $f(d)=\mathbb{E}[[\ E\ ]]u[I/d]\ \forall\ d$

$\mathbb{E}[[\ E_1\ (E_2\ ]]u = e?F\to e(\mathbb{E}[[\ E_2\ ]]u), error$ donde $e=\mathbb{E}[[\ E_1\ ]]u$

$\mathbb{E}[[$ if E_1 then E_2 else $E\ ]]u=$

$e?BOOL\to(e\to \mathbb{E}[[\ E_1\ ]]u, \mathbb{E}[[\ E_2\ ]]u), error$ donde $e=\mathbb{E}[[\ E\ ]]u$

$\mathbb{E}[[\ E$ where $D\ ]]u=\mathbb{E}[[\ E\ ]]\ u[\ \mathbb{D}[[D]]u\ ]$

$\mathbb{E}[[\ E$ whererec $D\ ]]u=\mathbb{E}[[\ E\ ]]\ fij(k_u)$

donde $k_u:U\to U$ y $k_u(u')= u[\ \mathbb{D}[[\ D\ ]]u'\ ]$

$\mathbb{E}[[\ (E)\ ]]u=\mathbb{E}[[\ E\ ]]u$

$\mathbb{D}[[\ I=E\ ]]u= u[I/\mathbb{E}[[\ E\ ]]u\ ]$

$\mathbb{D}[[\ D_1$ and $D_2\ ]]u= \mathbb{D}[[\ D_1\ ]]\ u[\ \mathbb{D}[[\ D_2\ ]]u\ ]$

Donde, como la extensiòn a la notaciòn

$$u[I/e](I')=\begin{cases} e & \text{si } I=I' \\ u(I') & \text{en otro caso}\end{cases}$$

utilizamos, para el dominio de u' contenido en el dominio u

$$u[u'](I) = \begin{cases} u'(I) & \text{si } u'(I) \text{ està definido} \\ u(I) & \text{en otro caso}\end{cases}$$

El lector no debe encontrar dificultad en comprender la notaciòn del ejemplo anterior por lo que presentamos un programa.

Consideÿrese

((fn x: (fn y : x)) (3)) (5)

Este programa define una funciòn

$$f:B\to(B\to B)$$

donde f(x)=(fn y : x) , es decir el valor de f(x) es la funciòn

$$f(x):B\to B$$

que es constante x, entonces $\forall\ y\in B$

$$f(x)(y)=x$$

de manera que el resultado del programa anterior debe ser 3.

Confirmèmoslo !.

Paso 1)

$\mathbb{E}[[((fn\ x:(fn\ y:x))(3))(5)]]u$

$= e_1\ ?F \rightarrow e_1\ (\mathbb{E}[[5]]u), error$

donde $e_1 = \mathbb{E}[[(fn\ x:(fn\ y:x))(3)]]u$

Paso 2)

$\mathbb{E}[[(fn\ x:(fn\ y:x))(3)]]u = e_2\ ?F \rightarrow e_2\ (\mathbb{E}[[3]]u), error$

donde $e_2 = \mathbb{E}[[fn\ x:(fn\ y:x)]]u$

Paso 3)

$\mathbb{E}[[fn\ x:(fn\ y:x)]]u = f_2$ donde

$f_2(d) = \mathbb{E}[[fn\ y:x]]u[x/d]\ \forall\ d$

Paso 4)

Entonces,

$e_2(\mathbb{E}[[3]]u) = e_2(3) = f_2(3) = \mathbb{E}[[fn\ y:x]]\ u[x/3]$

Paso 5)

Ahora

$\mathbb{E}[[fn\ y : x]]\ u[x/3] = f_1$

donde $f_1(d) = \mathbb{E}[[x]]u[x/3][y/d]$

pero $\mathbb{E}[[x]]u[x/3][y/d] = u[x/3][y/d](x) = 3$

$\therefore\ f_1(d) = 3\ \forall\ d$

$\therefore\ e_2(\mathbb{E}[[3]]u])(d) = f_1(d) = 3$

Paso 6)

$e_1(\mathbb{E}[[5]]u) = e_1(5) = f_1(5) = 3$

y tenemos el resultado

$\mathbb{E}[[((fn\ x:(fn\ y:x))(3))(5)]]u = 3\ \forall\ u.$

Invitamos al lector a realizar programas y ejecutarlos similarmente.

Sección 20 Sintaxis sensible al contexto

Otra de las aplicaciones de la Teoría de Dominios está presente en el anàlisis de casos sintàcticos determinados por contexto. Como señalamos en el primer capìtulo este tema forma parte de la frontera entre sintaxis y semàntica. Sin embargo, aspectos como la verificaciòn de tipos pueden ser manejados en tèrminos de la Teorìa de Dominios y la especificaciòn formal de la semàntica.

Presentamos un ejemplo de esto a continuaciòn.

Realizaremos una verificaciòn de las restricciones debidas al contexto en el lenguaje de programaciòn que presentamos para saltos. Las restricciones que verificaremos para su correcta sintaxis son:

a) El identificador de una asignaciòn debe ser el identificador de entrada y salida del programa o debe haber sido declarado por un "new".

b)El identificador que indica el destino de un goto debe ser una etiqueta.

c) El uso de un identificador debe realizarzarse dentro del ambiente especificado por su declaraciòn.

Las restricciones son indicadas formalmente definiendo funciones que verifican las frases con respecto a un contexto. El contexto puede pensarse como un ambiente estàtico o tabla de sìmbolos que se modela como un mapeo de identificadores en sus tipos.

$$x \in X = Ide \rightarrow Tp \qquad \text{contextos}$$

Para el lenguaje que estamos presentando los tipos son:

a) lv, que es el tipo asociado al Dominio L de localidades de Memoria.

b) rv, que està asociado con el Dominio R de valores almacenables.

c) cv, que representa al tipo de las continuaciones de comando,

d) indefinido, que es el tipo de los identificadores no iniciados.

Claramente para un lenguaje de programaciòn màs realista los tipos resultarían mucho más complicados.

Las funciones semánticas para expresiones, saltos y comandos simplemente verifican que, respecto a un contexto, el argumento(una frase sintáctica) satisfaga las restricciones antes mencionadas. La función para definiciones produce entonces un nuevo contexto, o propaga una bandera de error. Si M es una frase correspondiente a un programa completo entonces éste es corresto sintácticamente si y sólo si $\underline{m}[[\ M\ ]]$=Verdadero.

Sintaxis Abstracta

$E \in Exp$	Expresiones
$D \in Dcl$	Declaraciones
$S \in Slt$	Saltos
$C \in Com$	Comandos
$T \in Tp$	Tipos
$M \in Pro$	Programas

$E ::= 0 \mid 1 \mid -E \mid \text{not } E \mid E_1 + E_2 \mid E_1 = E_2 \mid I \mid \text{procedure } C \mid (E)$

$D ::= \text{new } I=E \mid \text{val } I=E$

$S ::= \text{goto } I$

$C ::= \text{null} \mid I:=E \mid \text{call } E \mid C_1 ; C_2 \mid \text{if } E \text{ then } C_1 \text{ else } C_2$
$\quad \text{while } E \text{ do } C \mid \text{with } D \text{ do } C \mid \text{begin } C \text{ end} \mid I:C \mid S$

$T ::= lv \mid rv \mid cv \mid \text{indefinido}$

$M ::= \text{program}(I);\ C.$

Dominios semánticos

T=BOOL

$\underline{e}$: Exp -> X -> T

$\underline{d}$: Dcl -> X -> (X + {error})

$\underline{s}$: Slt -> X -> T

$\underline{c}$: Com -> X -> T

$\underline{m}:Pro \rightarrow T$

$\underline{e}[[\ 0\ ]]x=\text{Verdadero} \quad \underline{e}[[1]]x=\text{Verdadero}$

$\underline{e}[[\ \text{not}\ E\ ]]x=\underline{e}[[\ E\ ]]x$

$\underline{e}[[\ -E\ ]]x=\underline{e}[[\ E\ ]]x$

$\underline{e}[[\ E_1+E_2\ ]]x=\ (\underline{e}[[\ E_1\ ]]x)\ \text{y}\ (\underline{e}[[\ E_2\ ]]x)$

$\underline{e}[[\ E_1=E_2\ ]]x=\ (\underline{e}[[\ E_1\ ]]x)\ \text{y}\ (\underline{e}[[\ E_2\ ]]x)$

$\underline{e}[[\ I\ ]]x=\ (x(I)=lv)\ \text{ó}\ (x(I)=rv)$

$\underline{e}[[\ \text{prodecure}\ C\ ]]x=\underline{c}[[\ C\ ]]x$

$\underline{e}[[\ (E)\ ]]=\underline{e}[[\ E\ ]]x$

$\underline{d}[[\ \text{new}\ I=E\ ]]x=\underline{e}[[\ E\ ]]x\rightarrow x[I/lv],\text{error}$

$\underline{d}[[\ \text{val}\ I=E\ ]]x=\underline{e}[[\ E\ ]]x\rightarrow x[I/rv],\text{error}$

$\underline{s}[[\ \text{goto}\ I\ ]]x=(x(I)=cv)$

$\underline{c}[[\ \text{null}\ ]]x=\text{Verdadero}$

$\underline{c}[[\ I:=E\ ]]x=(x(I)=lv)\ \text{y}\ (\underline{e}[[\ E\ ]]x)$

$\underline{c}[[\ \text{call}\ E\ ]]x=\underline{e}[[\ E\ ]]x$

$\underline{c}[[\ C_1;C_2\ ]]x=(\underline{c}[[\ C_1\ ]]x)\ \text{y}\ (\underline{c}[[\ C_2\ ]]x)$

$\underline{c}[[\ \text{if}\ E\ \text{then}\ C_1\ \text{else}\ C_2\ ]]x=(\underline{e}[[E]]x)\ \text{y}\ (\underline{c}[[\ C_1\ ]]x)\ \text{y}\ (\underline{c}[[\ C_2\ ]]x)$

$\underline{c}[[\ \text{while}\ E\ \text{do}\ C\ ]]x=(\underline{e}[[\ E\ ]]x)\ \text{y}\ (\underline{c}[[\ C\ ]]x)$

$\underline{c}[[\ \text{with}\ D\ \text{do}\ C\ ]]x=)(\underline{d}[[\ D\ ]]x\neq\text{error})\ \text{y}\ (\underline{c}[[\ C\ ]](\underline{d}[[\ D\ ]]x))$

$\underline{c}[[\ \text{begin}\ C\ \text{end}\ ]]x=\underline{c}[[\ C\ ]]x$

$\underline{c}[[\ I:C\ ]]x=\underline{c}[[\ C\ ]](x[I/cv])$

$\underline{c}[[\ S\ ]]x=\underline{s}[[\ S\ ]]x$

$\underline{m}[[\ \text{program}(I);\ C.]]=\underline{c}[[\ C\ ]](x[I/lv])\ \text{donde}\ \forall\ I'\ x(I')=\text{indefinido}$

Sección 21 Dominios semánticos para PASCAL

En esta sección describiremos los Dominios Semánticos para el lenguaje de programación PASCAL. Esperamos que este ejemplo ilustre que la semántica formal puede aplicarse a un lenguaje de programación real.

Sintaxis Abstracta de PASCAL

N números

B literales

O operadores

I identificadores

L expresiones a la izquierda

E expresiones

K expresiones estáticas

T expresiones de tipos

Q especificaciones de parámetros

P parámetros formales

D declaraciones

S saltos

C comandos

M programas

```
L ::= I | L.I | L[E] | E↑
E ::= B | I | OE | EOE | I(...,E,...) | L.I | L[E] | E↑ |
          [...,E,...,E..E,...] | (E)
K ::= B | I | OK
T ::= I | (...,I,...) | K..K | ↑I | set of T | file of T |
     array[T]of T | record ...;I:T;...end |
     record..I:T;...case I:Iof...;K:(...;I:T;...);...end
Q ::= I:I | var I:I | procedure I(...;Q;...) |
     function I(...;Q;...):I
P ::= I:I | var I:I | procedure I(...;Q;...) |
```

```
        function I(...;Q;...):I
D ::= const I=K | type I=T | var I:T; | procedure I(...;P;...);C;
        | function I(...;P;...):I;C;
S ::= goto N
C ::= L:=E | I | I(...,E,...) | C;C | if E then C |
        if E then C else C | case E of ...;K:C;...end |
        while E do C | repeat C until E | for I:=E to E do C |
        for I:=E downto E do C | N:C | S | with L do C |
        ...D...begin C end | begin C end
M ::= program I(...,I,...);C.
```

Nota:Las abreviaturas para arreglos multidimensionales han sido omitidas, lo mismo que "label","forward" y "packed".

Valores bàsicos

Los valores bàsicos o escalares en PASCAL son los valores de verdad, caracteres, enteros, àtomos de enumeraciòn, que corresponden a tipos enumerados definidos por el programador, subrangos de èstos y finalmente los nùmeros reales. Asumimos pues la siguiente notaciòn:

T={Falso,Veradero}	Valores de Verdad
H={\`a\`,\`b\`,...,\`z\`,\`0\`,\`1\`,..,\`9\`}	Caracteres
Z={-maxint,...,-2,-1,0,1,2,...,maxint}	Enteros
At	Atomos enumerativos
Re	Reales

El Dominio de àtomos enumerativos puede ser cualquier Dominio isomotfo a N, mientras que Re puede representarse como el Dominio de cadenas de dìgitos y \`.\` donde el \`.\` puede sòlo aparecer cuando màs una vez, (puede definirse como inconcistentes màs apariciones del \`.\`)

El Dominio de indices, es decir, valores que pueden utilizarse como indices de arreglos es

$$I = T + H + Z + At \qquad \text{indices}$$

Para definir el Dominio R de valores almacenables, S de estados, Rv de valores a la derecha de una asignación y Lv el Dominio de valores a la izquierda de una asignación realizamos el siguiente análisis.

Los valores tèoricos que son almacenables en una sola localidad en PASCAL incluyen a todos loa valores bàsicos ademàs de los conjuntos, los archivos, y los apuntadores. El valor de un conjunto es conveniente representarlo por una función de los valores de indices en los valores de verdad, de manera que, si el valor de la función es Verdadero el elemento pertenece al conjunto y si es Falso, el elemento no pertenece al conjunto, así definimos

$$Cj = I \rightarrow T \qquad \text{valores de conjuntos}$$

Un archivo puede ser representado por sucesiones de valores en Rv màs una componente de lectura o esritura para identificar si el archivo es de entrada o salida, entonces:

$$Ar = Rv \times \{lectura, escritura\} \qquad \text{valores de archivos}$$

Un apuntador es un valor izquierdo o el valor especial "nil"

$$Ap = Lv + \{nil\} \qquad \text{apuntadores}$$

Entonces el Dominio de valores almacenables es:

$$R = I + Re + Cj + Ar + Ap + \{nodefinido\} \qquad \text{valores almacenables}$$

donde el valor "no definido" se asocia a un identificador no iniciado. Un estado lo representamos por una función de las localidades de memoria en los valores almacenables o el valor "no ligado" para localidades no ligadas. Consecuentemente:

$$S = L \rightarrow (R + \{noligado\}) \qquad \text{estados}$$

El Dóminio de valores a la derecha de asignaciones puede ser definido como un Dominio reflexivo consistente de la unión ajena de valores almacenables, valores de registros y valores de arreglos, donde los registros se representan por una función de los identificadores en elementos de Rv mientras que los arreglos son funciones de los valores de indices en elementos de Rv:

$$Rv = Rv + (Ide\text{->}Rv) + (I\text{->}Rv)$$

El Dominio de valores a la izquierda se construye similarmente pero su constituyente primitivo son las localidades de memoria.

$$Lv = L + (Ide\text{->}Lv) + (I\text{->}Lv) + (Lv \times L)$$

donde el tèrmino Lv X L representa archivos a la izquierda donde la componente Lv representa un "buffer" para el archivo y L continene un elemento en L.

Un ambiente en PASCAL mapea identificadores en valores de Lv o procedimientos. Los identificadores pueden tambièn ser ligados a tipos o valores de Rv mediante las declaraciones "type" y "const" respectivamente, pero las asociaciones son màs convenientemente representadas en tèrminos de un anàlisis de sintaxis sensible al contexto.

Los procedimientos pueden ser simulados por funciones matemàticas que producen respuestas de programas y que reciben argumentos uno por uno (mediante aplicaciones de "curry"). Estos argumentos corresponden a:

1)Una lista de valores expresados por los paràmetros actuales.

2)Una continuaciòn que serà seguida despuès de la ejecuciòn del cuerpo del procedimiento, y

3)Un estado.

Por lo anterior, los Dominios para procedimientos son:

$P = E \text{->} C \text{->} S \text{->} A$ comandos de procedimientos

$F = E \text{->} K \text{->} S \text{->} A$ expresiones de procedimientos

donde los siguientes Dominios se señalan como:

E valores expresables

C continuaciones de comandos

K continuaciones de expresiones

A respuestas de programas

De esta forma, el Dominio de valores denotables se especifica como:

D = Lv + P + F + (F X L)

donde el nombre de una función denota un elemento de la componente FXL en el ambiente de su definición, la localidad de esta pareja se utiliza para regresar un valor de la ejecución del procedimiento (funciones en PASCAL).

El Dominio de ambientes queda definido por la ecuación

U = (Ide->D) X (N->C) ambientes

donde la componente N->C es para etiquetas de comandos y el Dominio de valores expresables se define por:

E = Lv + Rv + P + F

Como es necesario el uso de continuaciones para comandos, expresiones y declaraciones, las cuales toman un estado como argumento y producen una respuesta final, los Dominios se definen por:

A respuestas

C = S -> A continuaciones de comandos

K = E -> S -> A continuaciones de expresiones

Q = U -> S -> A continuaciones de declaraciones

Invitamos al lector a comparar este análisis con los ejemplos anteriores y esbozar una definición semántica denotativa para PASCAL.

<u>BIBLIOGRAFIA</u>

Aho,A y Ullman,J.

1979 <u>Principles of Compiler Design</u>.Adison Wesley.604pp.

Bjoner,D y Jones, C.

1978 <u>The Vienna Developement Method:The Meta-Language.</u> Lecture Notes in Computer Science.Springer-Verlag.

Bool,G y Jeffrey,R .

1080 <u>Computability and Logic.</u>Cambidge Univerity Press.

Bracho F.

1979 <u>Fixed Points in Pw</u>.Comunicaciones Tècnicas, Serie Naranja IIMAS, UNAM. 42pp.

Gordon, M.

1979 <u>The Denotational Description of Programming Languages. An Introduction</u>. Springer-Verlag 159pp.

Hoare,C.A.y Wirth.

1973 <u>An axiomatic Definition of the Programming Languge PASCAL</u>. Acta Informatica vol 2,Num 4 . 335-356pp.

1969 <u>An axiomatic Basis of Computer Programming</u>. Communications ACM, 12. 576-680,583pp.

Jensen,K y Wirth,N.

1974 PASCAL User Manual and Report. Springer-Verlag.

London,R.L et al.

1978 Proof Rules for the Programming Language Euclid. Acta Informatica Vol 10, Num 1.

Mac Lane,S

1972 Categories for the Working Mathematician. Springer-Verlag.

Mc. Carthy, J.

1966 A Formal Description of a Subset of ALGOL. Fornal Language Description Languages for Computer Programming. 1-7pp.

Milne,R.E. y Stratchey,C.

1975 A Theory of Programming Languge Semantics. John Wiley.

Moses,P.D.

1976 Compiler Generation Using Denotational Semantics. Mathematical Fundantions of Computer Science. Springer-Verlag. 436-411pp.

Scott,D.S.

1970 Outline of a mathematical theory of Computation. Proc 4th Annual Princeton Conf On Information Sciences and Systems 169-176pp.

1976 Data Types as Lattices. SIAM Journal of Computing, vol 5. 522-587pp.

1977 Logic and Programming Languges. 1976 Turing Award Lecture . Communications ACM. Vol 20,Num 20 .634-641 pp.

1981 Lectures on a mathematical theory of Computation. Oxford University Computing Laboratory Technical monograph. 148pp.

1982 Domains for Denotational Semantics. Depertment of Computer Science. Carnegie-Mellon University. 46pp.

Stoy, J. E.

1977 Denotational Semanticas: The Scott-Strchey Approach to Programming Language Theory. MIT. Press ,414pp.

1980 Some Mathematical Aspects of Functional Programming. Oxford University Computing Laboratory. 217-252, pp.

Stoll

1963 Set Theory and Logic. Dover.

Stratchey, C.

1966 Towards a Formal Semantics. Formal Language Description Languages for Computer Programming. North Holland, 192-220pp.

Tennet, R. D.

1981 Principles of Programming Languages. Prentice-Hall.