24a
20)

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ARAGÓN

SATISFACCIÓN DE RESTRICCIONES POR REFERENCIA SIMBÓLICA EN DIBUJOS GEOMÉTRICOS

TESIS QUE PARA OBTENER EL TÍTULO DE
"INGENIERO EN COMPUTACIÓN"
PRESENTA:

José Antonio Massé Márquez

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ARAGON
UNIDAD ACADEMICA

VNIVERSIDAD NACIONAL
AVFNOMA DE
MEXICO

Ing. SILVIA VEGA MUYTOY
Jefe de la Carrera de Ingeniería
en Computación,
P r e s e n t e .

En atención a la solicitud de fecha 7 de noviembre del año en curso, por la que se comunica que el alumno JOSE ANTONIO MASSE MARQUEZ, de la carrera de Ingeniería en Computación, ha concluido su trabajo de investigación intitulado " SATISFACCION DE RESTRICCIONES POR REFERENCIA SIMBOLICA EN DIBUJOS GEOMETRICOS ", y como el mismo ha sido revisado y aprobado por usted se autoriza su impresión; así como la iniciación de los trámites correspondientes para la celebración del examen profesional.

Sin otro particular, le reitero las seguridades de mi atenta consideración.

ATENTAMENTE
"POR MI RAZA HABLARA EL ESPIRITU"
San Juan de Aragón, Méx., 8 de noviembre de 1994
EL JEFE DE LA UNIDAD

Lic. ALBERTO TRARBA ROSAS

c c p Asesor de Tesis.
c c p Interesado.

AIR'lla.

Dedicado A :

El amor más puro, la tenacidad verdadera, lucha y visión futurista del ser más cercano a mí. Madre este trabajo es uno de los frutos de tus constantes luchas.

Agradecimientos

Agradezco a usted, amable lector, porque un interés muy particular del autor es que este humilde documento sea de utilidad para usted. Esta hoja está dedicada a agradecer también a las personas e instituciones que han contribuído de alguna manera a que un servidor termine un libro de esta naturaleza, por lo que luego de dar gracias a Dios, quisiera que usted, querido lector, sepa quienes son aquellos que me han ayudado y sin los cuales no hubiera sido posible este documeto.

Gracias doy a la institución que me dió formación y escuela: ENEP Aragón, UNAM.

Gracias al Instituto de Investigaciones Eléctricas por haberme brindado la oportunidad de realizar esta tesis y al Departamento de Sistemas de Información por brindarme los medios de trabajo y de subsistencia para la elaboración de la misma.

Sin tener como pargar tanto interés, paciencia y conocimientos, agradezco de corazón al Dr. Luis Alberto Pineda amigo y director de esta investigación.

Aprecio también en todo lo que vale el consejo sabio y oportuno además de la gran ayuda en los aspectos de geometría del buen Sergio Santana.

Agradezco también a las personas que desinteresadamente me orientaron en problemas específicos: Ricardo del Cueto, Elí de la Torre y Tomás Rodriguez.

A mi señora madre le agracezco tanto que le dedico este trabajo.

Gracias a Claudia Estela, Laura Leticia y Carlos Alberto Massé Márquez por haberme brindado siempre su apoyo y reafirmalo en el tiempo que duró la realización de la tesis.

Por ponerle sal a la vida de becario quiero agrader a mis amigos Gabriela Garza, Nirya Islas, Adrián Pellegrini, Mayolo Camacho, Mónica Moreno, Guadalupe Susano, Silvia Aguilar, Victor Cortés, Ismael Daza, J. C. Bolan, Rafael Morales, Juan de Dios Aradillas, Gabriel Meraz, Homobono Meraz, Tomás Aradillas, Gabriel Ramírez, Yasmín Hernández e Italo Coronado.

Por último y por ser como el postre, agradezco a mi sobrino Rodrigo porque con su inocente ternura me muestra las cosas más grandes e importantes de la vida: una nueva vida, un niño y, quien sabe, un futuro ingeniero.

Indice

Introducción.

Antecedentes.

El diseño humano es un proceso de solución de problemas en el que los objetivos del proceso no se encuentran definidos de manera precisa; sin embargo, es necesario conocer algunas características del mismo de antemano para que la realización de este proceso sea posible. Conocemos al conjunto de características que especifican a un proceso de diseño como "restricciones de diseño". El área de diseño a la que se enfoca la investigación de este trabajo es la de modelado geométrico. Una herramienta muy popular para asistir esta tarea por medios computacionales es la familia de programas que conocemos como modeladores geométricos. Estos programas ponen varias ayudas a disposición del diseñador; sin embargo, normalmente no permiten la definición y satisfacción de las restricciones de diseño que el diseñador tenga en mente. La mayoría de los modeladores no proveen un soporte adecuado para satisfacer las restricciones geométricas. Si el diseñador, asistiéndose de un modelador geométrico común, crea un objeto estableciendo restricciones durante su creación, es decir, asociando un conjunto de características a dicho objeto; y en el transcurso de diseño alguna de las características es violada, el modelador común no efectúa acción alguna para asegurarse que dicha condición

sea restablecida. A la acción de actualizar un dibujo de acuerdo a las características que debe contener la conocemos como *"satisfacción de restricciones"*.

Los esfuerzos por resolver este aspecto del diseño han tomado diferentes direcciones. El método tradicional consiste en traducir el dibujo en las ecuaciones que lo describen y resolver dichas ecuaciones con diferentes técnicas. En el presente trabajo seguimos una alternativa diferente a los métodos comunes para resolver problemas de satisfacción de restricciones. Se enfrenta el problema por medio de manipulación simbólica y se enmarcan los horizontes que desde esta perspectiva se vislumbran para el problema de satisfacción de restricciones.

Para validar las ideas que se plasman a lo largo de este trabajo se desarrolló un sistema gráfico interactivo llamado S2RS. El sistema corre actualmente en una DECstation 5000 con sistema operativo ULTRIX y bajo el sistema de ventanas X-Windows. El módulo de satisfacción de restricciones fué programado en su etapa inicial en C-Prolog y actualmente se encuentra en BinProlog 2.20. La interfase gráfica está programada con el sistema para desarrollo de interfases gráficas Tk/Tcl.

Los conceptos mencionados de restricción, satisfacción de restricciones y métodos de satisfacción de restricciones son analizados más a detalle en el contenido de la presente tesis. Se delimitan a continuación objetivos y alcances de la misma.

Objetivos de la Tesis.

1.- Mostrar que la anipulación simbólica es una alternativa en el problema de satisfacción de restricciones con un sistema de software que permita resolver restricciones gráficas mediante referencia simbólica.

2.- Mostrar que el nivel de abstracción en el que se conceptualizan los problemas de diseño en un sistema de dibujo es un factor que influye considerablemente en el proceso de satisfacción de restricciones.

3.- Mostrar que la expresividad de un lenguaje de representación es un factor que influye considerablemente en el proceso de satisfacción de restricciones.

Alcances de la Tesis.

1.- Los objetos gráficos en el sistema son bidimensionales.

2.- El sistema contará con una interfase gráfica para facilitar la interacción con el usuario. La interacción con el usuario será para manipulación de los objetos gráficos desplegados en pantalla.

3.- El sistema resolverá restricciones por el método de referencia simbólica.

Contenido de laTesis.

La tesis esta formada por tres capítulos a saber:

Capitulo 1.- Este capítulo tiene como objetivo presentar los conceptos de *restricción, satisfacción de restricciones, técnicas de satisfacción de restricciones* y una semblanza de cada una de ellas y ejemplos de sistemas de satisfacción de restricciones. También se introduce en este capítulo el método símbólico a seguir para resolver restricciones.

Capitulo 2.- En este capítulo se presenta la definición del lenguaje de satisfacción de restricciones que se usa para hacer referencia a los objetos gráficos. Se expone la especificación formal y las reglas sintácticas y semánticas de interpretación de expresiones del lenguaje.

Capítulo 3.- En éste se presenta al sistema S_2RS, se explica como el nivel de abstracción del lenguaje y la expresividad del mismo contribuyen al proceso de satisfacción de restricciones. Se expone el concepto de solución de problemas de satisfacción de restricciones por métodos de referencia simbólica y se da una introducción a la solución por medio de procesos de inferencia sintéticos. Este capítulo también muestra algunas características del sistema de software.

Apéndice A El apéndice muestra algunas pantallas del sistema S_2RS trabajando.

"Crítica significa apreciación justa, sobre todo apreciación de las posibilidades del hombre como creador y sostenedor de la cultura... Tocante a la razón humana, hace ver sus limitaciones, pero al mismo tiempo garantiza su obra posible y creadora...La crítica hace al hombre".

Kant

Capítulo 1

Técnicas de Satisfacción de Restricciones

1.1 Introducción.

La motivación principal de este trabajo surge del deseo de proporcionar herramientas para apoyar el diseño de sistemas de dibujo y CAD que realizan operaciones complejas de edición gráfica de acuerdo con las espectativas de los usuarios. En el contexto de la modelación gráfica en el que estamos interesados, las intenciones de los dibujantes humanos son especificadas a través de restricciones de carácter geométrico, topológico y conceptual. El objetivo de este capítulo es explicar el concepto de restricción y otros conceptos asociados, así como sus técnicas de satisfacción y los retos que éste presenta actualmente.

Para comenzar nuestra explicación sobre las restricciones partiremos de un escenario hipotético: imagine usted que está en una sesión interactiva con un modelador gráfico arbitrario el cual será utilizado con fines ilustrativos. Las características del mismo van a ser descritas sobre la marcha de la explicación. Digamos que usted, con este modelador, dibuja un cuarto de baño con una tina y sus tuberías asociadas, como se muestra en la Figura 1.1.

El cuarto de baño consiste de dos rectángulos y dos líneas discontinuas. Uno de los rectángulos, el más grande, representa al cuarto de baño. En este rectángulo existe una pequeña abertura en su parte inferior y una línea inclinada que representa la puerta del baño. El rectángulo más pequeño representa la tina y las líneas discontinuas T_1 y T_2 representan a las tuberías. La tubería T_1 representa un segmento del drenaje y la tubería T_2 representa a su vez la conexión entre la tina y el drenaje. Omitimos el mecanismo de interacción hombre-máquina mediante el cual son creados los objetos (rectángulos y líneas) ya que este aspecto de la interacción no es relevante para nuestros actuales propósitos; sin embargo, volveremos sobre dicho punto en las conclusiones de este trabajo.

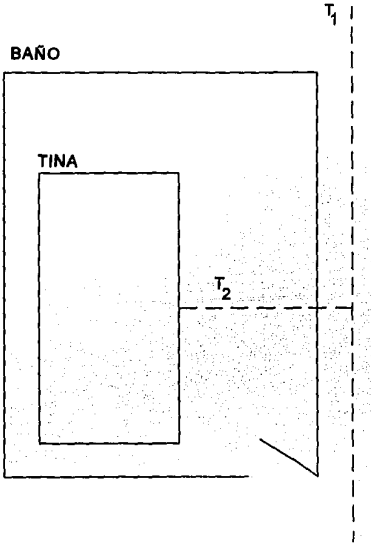


Figura 1.1.- Cuarto de baño. Las líneas T_1 y T_2 representan tuberías.

Supongamos que ahora usted se da cuenta de que la colocación de la tina no es la más adecuada, por ejemplo, porque un tocador podría estar mejor ubicado cerca de

la puerta. Así que ahora usted se dispone a cambiar la tina de lugar. ¿Cómo podemos hacer este cambio manteniendo la tina funcional? Hay que aclarar que por funcional entendemos que tenemos la intención de que la conexión entre la tina y el drenaje se mantenga de manera apropiada. Para realizar esta tarea podemos proceder a seleccionar el rectángulo que representa a la tina con el ratón (mouse) para desplazarlo (mediante una operación de "drag") hasta la posición deseada.

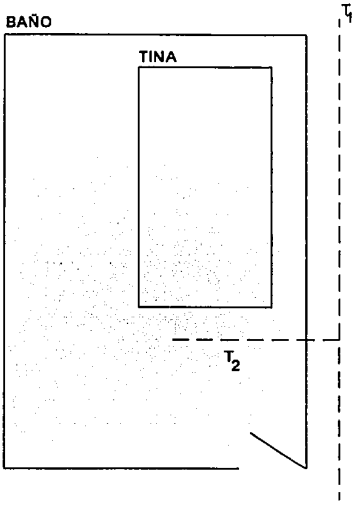


Figura 1.2.- Posible resultado obtenido con un modelador imaginario al desplazar la tina.

Si esta operación se realiza sin nunguna consideración especial la tina quedaría desconectada (véase Figura 1.2) y por lo tanto, fuera de funcionamiento. Existen varias posibilidades para resolver este problema de acuerdo con nuestras intenciones: unas que dependen del usuario y otras que dependen del programa. Entre las primeras, la más sencilla sería que el usuario borrara la línea T_2 e hiciera una nueva línea conectando la tina con la tubería general de manera explícita. Otra solución de este tipo sería que el usuario seleccionara interactivamente el extremo que ha quedado desconectado de la línea T_2 y lo desplazara hasta que este coincidiera nuevamente con

la línea que define el contorno de la tina. Este procedimiento dejaría la tubería T_2 inclinada y no horizontal como ésta se encontraba en la configuración original. Como ya se ha dicho, en este tipo de operaciones, el razonamiento depende única y exclusivamente del usuario ya que es éste quién toma todas las decisiones necesarias para resolver el problema.

En el segundo tipo de operaciones a la que nos hemos referido, es el programa quien toma las decisiones necesarias para modificar el dibujo. Programar este tipo de tareas implica la modelación de un proceso de razonamiento. En este ejemplo, definimos como una *restricción de diseño* nuestra intención de que las líneas que representan al borde de la tina y a la tuberia de desague deben de estar conectadas antes y después de la manipulación interactiva. El proceso por medio del cual este problema es resuelto lo denominamos *satisfacción de restricciones*.

1.2. Noción Intuitiva del Concepto de Restricción.

El concepto de restricción es inherente a la actividad de diseño [Darses 90] y es normalmente mencionado cuando se habla de procesos de solución de problemas. Intuitivamente definimos una restricción como una condición de naturaleza geométrica e inclusive conceptual que el usuario tiene la intención de que sea satisfecha por una representación gráfica [Pineda 92]. Cuantitativamente, un conjunto de condiciones de diseño pueden ser expresadas por medio de un sistema de ecuaciones. En este contexto, una restricción puede ser entendida como una relación entre variables cuyos valores están limitados por la naturaleza de la restricción.

En el ejemplo que hemos estado manejando tenemos la intención de expresar que las tuberías están conectadas para que el agua pueda fluir. Es importe hacer notar que esta restricción es de caracter conceptual, y que restricciones de esta clase se expresan normalmente por medio del lenguaje natural (p. ej. español). Por otro lado, la tina y las tuberías están representadas por líneas y por lo tanto, la restricción se establece también a nivel geométrico y topológico. Es este el nivel de abstracción al que enfocamos nuestra investigación. De hecho, la intención que hemos tratado de satisfacer en nuestro ejemplo se especifica no por una sino por varias restricciones, como son las siguientes:

1) Que a lo largo de la sesión interactiva exista una conexión entre las líneas T_1 y T_2.

2) Que a lo largo de la sesión interactiva exista una conexión entre el rectángulo y la línea T_2.

3) Que a lo largo de la sesión interactiva las líneas T_1 y T_2 permanezcan perpendiculares.

4) Que la línea T_2 permanezca horizontal durante el proceso de diseño.

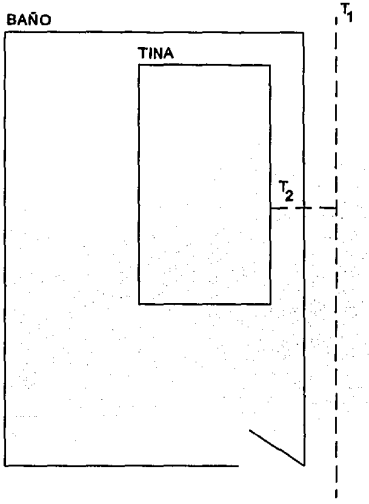


Figura 1.3.- Posible resultado obtenido cuando el modelador maneja restricciones.

Si el sistema de dibujo fuera capaz de interpretar estas restricciones, el modelo del dibujo sería como se muestra en la Figura 1.3. Note sin embargo, que las cuatro restricciones mencionadas no especifican completamente el problema de interacción gráfica a resolver. Dicho conjunto no nos indica por ejemplo, a que altura o proporción del rectángulo se va a hacer la conexión con la tubería T_2. Tampoco se dice nada acerca de la magnitud de la línea T_2. Más aún, si la magnitud de T_2 fuera constante en el cambio aparecería un segmento de T_2 a la derecha de T_1, lo cual evidentemente no deseamos. Como se vé, el conjunto de configuraciones diferentes que satisfacen las restricciones que han sido expresadas en forma explícita es extremadamente grande, y en muchos casos infinito. El punto que se quiere enfatizar es que normal-

mente existe un conjunto indeterminado de posibles soluciones, por lo que nuestro problema no es sólo hallar una configuración cualquiera en el espacio de solución, sino aquella que el usuario tenga en mente. Lo que se quiere es que el sistema pueda interpretar, de manera aproximada, la intención del usuario.

El trabajo desarrollado en ésta tesis esta orientado a esta segunda forma de resolver los problemas que surgen en los sistemas de dibujo interactivo. En el tipo de sistemas que nos proponemos estudiar, no se deja toda la carga del razonamiento al usuario, ya que el sistema debe ser capaz de interpretar cuando menos un conjunto pequeño de las intenciones que el usuario exprese a través de la sesión interactiva. En este contexto, si el grado de inteligencia del sistema es pobre, la carga sobre el usuario será alta; por el contrario, si el grado de inteligencia es alto, la carga sobre el usuario será mínimizada, aunque siempre habrá algún tipo de participación del usuario en el proceso.

1.3 Lenguaje de Satisfacción de Restricciones.

Es importante distinguir entre una restricción y el lenguaje por medio del cual ésta es expresada. Considere que la misma restricción puede ser expresada mediante el español, el inglés, algún lenguaje de tipo lógico, una secuencia específica de comandos interactivos o incluso los mismos símbolos gráficos involucrados introducidos mediante una convención preestablecida. Es importante hacer notar que diferentes lenguajes difieren en sus propiedades expresivas y si algún concepto puede ser expresado en un lenguaje no significa que éste pueda ser expresado en un lenguaje alternativo de menor grado de expresividad. El punto que queremos enfatizar es que una parte importante de resolver el problema de satisfacción de restricciones consiste en contar con un lenguaje que exprese con suficiente claridad los objetos y las restricciones que sobre ellos se apliquen. Un lenguaje que cumple con estas características se conoce como un lenguaje de satisfacción de restricciones [Leller 88]. Un sistema de satisfacción de restricciones es aquel que encuentra soluciones a los programas escritos en lenguajes de restricciones a partir de métodos de solución de problemas llamados Técnicas de Satisfacción de Restricciones. Estas técnicas encuentran los valores de los objetos que hacen que se cumplan las restricciones. Las técnicas de satisfacción de restricciones pueden ser clasificadas en dos familias: las técnicas basadas en métodos numéricos y las técnicas basadas en métodos simbólicos. En este capítulo se presentan un resúmen y discusión de las técnicas numéricas, después se procede a una explicación de las técnicas simbólicas que son aquellas en las cuales se enfoca nuestra investigación.

Veamos la diferencia entre un programa escrito en un lenguaje declarativo y un programa en un lenguaje de restricciones. Considere que se quiere contar con un programa que calcule la conversión de temperatura de grados Fahrenheit a grados Centígrados y viceversa. Por ejemplo, observe la siguiente expresión:

$$C=(F-32) \times 5/9$$

tal expresión es, en sí misma, un programa en un lenguaje de satisfacción de restricciones en el que los valores que pueden tomar las variables "C" y "F" están restingidos por la expresión. El trabajo del sistema de satisfacción de restricciones es encontrar una solución que satisface esta relación, ya sea que se tenga un valor de F para encontrar C o un valor en C para encontrar F. Es decir, en un sistema de satisfacción de restricciones típico, este programa puede encontrar la temperatura en grados centígrados dada la temperatura en grados fahrenheit y viceversa. No se requiere una línea de código que resuelva para C y otra línea que resuelva para F como sucedería en un lenguaje declarativo (C, Pascal y otros) en el que se tendría que añadir una línea de código semejante a:

$$F = 32 + 9/5 \times C$$

y otras líneas que establezcan las condiciones en las que se usa una u otra ecuación.

Así, en un lenguaje de satisfacción de restricciones de un sistema de dibujo geométrico, un programa representa al dibujo mismo, ya que las figuras que componen los dibujos como líneas, puntos, círculos, etc., pueden ser considerados como el valor de las expresiones en el lenguaje y las intersecciones entre ellos, distancias, ángulos, paralelismo entre líneas y otras relaciones que los encuadran y que se tiene la intención de que permanezcan, pueden ser consideradas como las restricciones del programa. Un criterio similar es el que estableció Ivan Sutherland para la identificación de objetos de diseño representados a través de dibujos en su programa Sketchpad :

"La construcción de un dibujo con Sketchpad es en sí misma un modelo del proceso de diseño. Las localizaciones de los puntos y de las líneas del dibujo modelan las variables de diseño y las restricciones geométricas aplicadas a los puntos y líneas del dibujo modelan las restricciones de diseño que limitan los valores de las variables de diseño. La habilidad del Sketchpad para satisfacer las restricciones geométricas aplicadas la parte de un dibujo modelan la habilidad de un buen diseñador para satisfacer todas las condiciones de diseño impuestas por las limitaciones de sus

materiales, costo, etc." [Sutherland 63 pp. 332].

1.4. Técnicas de Satisfacción de Restricciones.

En esta sección veremos cuáles son las técnicas de satisfacción de restricciones que han sido usadas tradicionalmente, apoyandonos principalmente en Leler [Leler 88].

Las técnicas de satisfacción de restricciones se componen de un conjunto de reglas de solución de problemas y de un mecanismo de control. El conjunto de reglas de solución de problemas puede ser de propósito general o para alguna aplicación específica, en nuestro caso para figuras geométricas. El mecanismo de control, como su nombre lo indica, controla cuándo y cómo se aplican las reglas. Los problemas que puede resolver un sistema de satisfacción de restricciones dependerán de las reglas que éste tiene definidas; es decir, si el sistema cuenta con una regla de solución para determinada relación deseada en el dibujo, entonces la restricción puede ser resuelta. Si un sistema puede satisfacer la relación de paralelismo o perpendicularidad, esto indica que dicho sistema cuenta con una regla o un conjunto de ellas para resolver tales relaciones. Además, la flexibilidad de las reglas y el modo de llevar a cabo al sistema dependen del mecanismo de control usado.

El deseo de resolver las restricciones en el modelado geométrico por computadora tiene una larga historia que se remonta al trabajo de disertación doctoral de Ivan Sutherland [Sutherland 63]. Las soluciones tomadas, aunque han variado y mejorado, generalmente se han mantenido sobre la tendencia de los métodos numéricos. A continuación presentamos un breve resúmen de estos métodos.

Un programa en un lenguaje de restricciones puede ser visto como una gráfica. Por ejemplo, la figura 1.4 es una representación gráfica del programa

$$C \times 1.8 + 32 = F$$

en donde los nodos rectangulares representan variables cuyo valor no ha sido definido, y los nodos redondos representan operadores. Los arcos representan tanto a los argumentos como a los resultados producidos por los operadores. Los argumentos se asignan al lado izquierdo de cada operador y su resultado es asignado por el lado derecho. Los operadores que no toman argumentos (operadores nulos) son constantes.

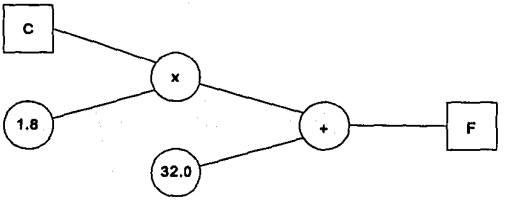


Figura 1.4.- Gráfica de Restricciones para un Programa de Conversión de Temperatura.

1.4.1. Propagación Local.

La Propagación Local de Estados Conocidos o simplemente Propagación Local es el mecanismo de satisfacción de restricciones más simple y el más fácil de programar. Consiste en propagar los valores conocidos por los arcos de una gráfica de restricciones. Un nodo que mantiene un valor constante contiene toda la información que necesita dado que su valor no depende de ningún otro nodo, así que se puede disparar inmediatamente y enviar el valor sobre su arco. Cuando un nodo recibe información suficiente de sus arcos se "dispara", es decir, calcula los valores para los arcos que no los contienen. Los nuevos valores se propagan a través de los arcos provocando que otros nodos se disparen y así sucesivamente. Las reglas de solución de problemas son locales para cada nodo puesto que solo trabajan con la información contenida en los arcos conectados al mismo.

Un nodo "+", por ejemplo, tiene asociadas tres operaciones cada una de las cuales puede ser disparada cuando los dos arcos que corresponden a sus argumentos contengan valores. Por ejemplo, si sus arcos son etiquetados con p, q y r, de modo tal que la restricción es $r=p+q$, entonces el nodo contendrá las reglas

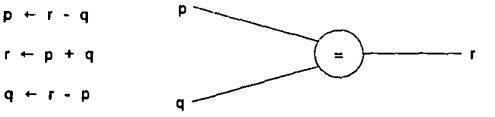


Figura 1.5.- Reglas de un nodo (+) más.

Por ejemplo, si los valores llegan en p y q, tal como sería en un lenguaje de

programación normal, el valor de *r* sería obtenido mediante la primera operación; sin embargo, si los valores son *p* y *r*, entonces su usará la regla $q \leftarrow r - p$.

La principal desventaja de las técnicas de propagación local es que sólo pueden usar información local a un nodo y muchos problemas comunes de satisfacción de restricciones no pueden ser resueltos de este modo ya que la propagación local no puede resolver gráficas que contienen ciclos. Un ejemplo de estos es el siguiente programa para encontrar el promedio de dos números:

$$A + T = B$$
$$B + T = C$$

Este programa restringe que *B* sea el promedio de *A* y *C*; mientras que *T* sea la diferencia entre *A* y *B*. Como se puede apreciar en la figura 1.6, la gráfica de restricciones para este programa contiene un ciclo.

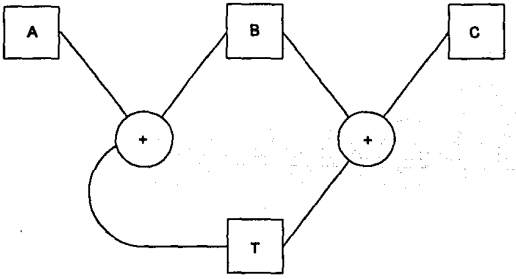


Figura 1.6.- Gráfica de restricciones con un ciclo.

Si se dan valores a las variables *A* y *C*, el método de propagación local no es capaz de resolver este problema, ya que no es posible determinar de forma única en un proceso convergente los valores de *B* y *T*. Por ejemplo si *A* es 1 y *C* es 11, entonces las ecuaciones se convierten en:

$$1 + T = B$$
$$B + T = 11$$

Note que en el ciclo de la figura 1.6 el valor de *B* depende del valor de *T* y viceversa. Hay una dependencia cíclica entre *B* y *T*, por lo que se requiere mayor

información adicional para deducir valores para cualquiers de las dos variables. Si no damos un valor para *B* o *T* entonces ningún nodo se puede disparar, así que la propagación local no puede romper el ciclo. Por supuesto que hay una solución pero para encontrarla se deben usar técnicas más poderosas como la que se muestra a continuación.

1.4.2. Relajación.

Un método que puede resolver el problema anterior es una técnica clásica de aproximación numérica iterativa llamada *Relajación*. La Relajación hace una asignación inicial de los valores de las variables desconocidas y si con los valores supuestos no se cumple alguna restricción entonces calcula el error que causaría la asignación de estos valores a las variables. Este proceso se repite hasta que el error es minimizado, teniendo cuidado de que el proceso converja hacia una solución. Diferentes formas de relajación forman diferentes heurísticas para hacer las suposiciones[1]. Las heurísticas son cualquiera de las múltiples técnicas de solución de problemas que envuelven el uso de conocimiento intuitivo, tales como: prueba y error, "regla de dedo" y otros métodos informales que generalmente dan las soluciones más aproximadas al valor real. En los programas de satisfacción de restricciones se dispone de un procedimiento de búsqueda basado en suposiciones que tienen una alta probabilidad de ser ciertas y que contribuyen a la solución del problema.

Una de las ventajas de las técnicas iterativas es que aún cuando el valor del error no tienda a cero puede ser minimizado; así que estas técnicas son útiles para encontrar soluciones aproximadas a problemas con múltiples restricciones. Entre sus desventajas está el hecho de que las restricciones a ser relajadas deben poderse expresar como ecuaciones lineales; de otra forma, puede ser que el método no converja. También puede ser que haya más de una solución y que el método de relajación encuentre tan sólo una de ellas.

El método de relajación, aún para sistemas de ecuaciones lineales que están garantizados para converger, es muy lento y aunque la relajación puede ser usada directamente como mecanismo de satisfacción de restricciones es comúnmente usada sólo después de que el método de propagación local ha sido probado y ha fallado. Existen varios trucos que pueden ser usados para acelerar la relajación cuando es usada en combinación con las técnicas de propagación local. Una de estas técnicas reduce el número de objetos que necesitan ser relajados tomando uno de ellos, y

[1]El término heurística proviene de la palabra griega que significa encontrar o descubrir.

entonces determina que otros valores podrían ser deducidos usando propagación local si el valor de dicho objeto fuera conocido. Los valores que pueden ser deducidos no necesitan ser supuestos para cada iteración de relajación. Esta técnica puede ser usada en el programa que encuentra el promedio de 1 y 11 del ejemplo anterior. Si fuera usada la relajación sencilla, entonces *B* y *T* tendrían que ser relajados. En lugar de hacer eso tomamos a *B* para relajarse, y para efectuar una suposición inicial ponemos el valor de *B* igual a cero. Si el cero se propaga hasta el segundo nodo "+", ese nodo se dispara y cambia el valor de *T* a 11. Este valor se propaga al primer nodo "+", el cual se dispara y cambia el valor de *B* a 12, Véase la figura 1.7 en la que se indica el flujo de la propagación con las puntas de flechas. Ahora *B* tiene dos valores, la suposición inicial (cero), y el valor que fue propagado de regreso (12). El error de *B* es la diferencia entre estos dos valores, y la nueva suposición para el valor de *B* es el promedio de los dos. El promedio de 12 y 0 es 6, que es el valor correcto. Para verificar este proceso, el nuevo valor es propagado nuevamente por el ciclo para ver si éste regresa sin cambio. Este paso final también determina el valor de *T* que es 5.

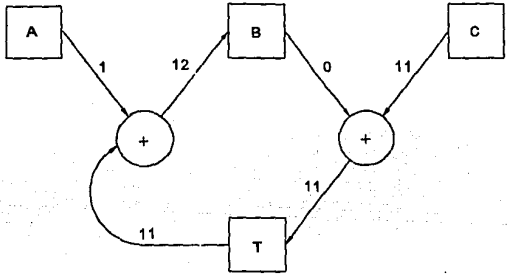


Figura 1.7.- Gráfica del programa de conversión de temperatura indicando el flujo de la propagación usando Relajación.

Note que todas las variables contribuyen con sus valores en el ciclo descrito anteriormente. Sin embargo, considere qué pasaría con este método si el proceso sólo tomara un subconjunto de las variables en la gráfica. El método de relajación aplicado a toda la gráfica afectaría tanto a las variables de las ramas conectadas al ciclo como aquellas externas a él. Esto sugiere una heurística para resolver gráficas complejas la cual se explica a continuación.

1.4.3. Propagación con poda temporal.

La propagación local que se explicó anteriormente puede resolver gráficas con la información que pueda obtener de sus arcos (tambien llamada información local). La segunda tecnica que se explicó, la de Relajación, contribuye a la solución de gráficas cíclicas, pero no hay manera de decirle al sistema cuáles variables están en un ciclo y cuáles son independientes de éste. Sin embargo si consideramos la gráfica en forma global podemos aislar aquellas ramas que no contribuyan directamente para el cálculo de las variables en un ciclo dado, para resolverlo por separado, particionando de esta manera la solución a la gráfica de restricciones en un conjunto de problemas simples. Después de que la relajación ha determinado el valor de las variables dentro de un ciclo, sus valores son propagados hacia las ramas independientes.

A fin de hacer la poda, se usa la técnica conocida como Propagación con poda temporal. En esta técnica identifica un objeto con pocas restricciones de manera que su valor pueda ser podado de la gráfica sin alterar la configuración global de éste. Considere el siguiente ejemplo:

$$5 \times Y = X$$
$$X + X = 40$$

cuya gráfica de restricciones se muestra en la figura 1.8. En esta figura el objeto X está en un ciclo, y el objeto Y tiene sólo una restricción asociada. Una vez que el valor de X sea conocido será posible satisfacer la restricción $5 \times Y = X$. Por consiguiente podemos podar la variable Y de la gráfica de restricciones junto con el nodo del signo de multiplicación y la constante 5, quedando únicamente el ciclo que contiene al objeto X. A continuación será posible determinar que X es igual a 20 por medio de relajación. Finalmente la rama podada podrá ser reacomodada y el valor de y , que es igual a 4, podrá ser calculado por medio de propagación local.

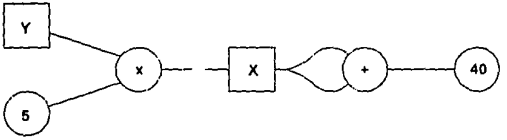


Figura 1 8.- Gráfica que contiene un ciclo.

Una dificultad con esta técnica es determinar cuáles objetos pueden ser podados.

Típicamente la heurística usada es que una variable puede ser removida de la gráfica si sólo tiene una restricción asociada a ella. Este método puede ser refinado con una técnica llamada *Precompilación,* la cual consiste en precompilar el paso final de propagación cuando se poda un objeto de la gráfica. Durante la precompilación se sustituye el nodo a podar por la regla que lo resuelve usando los objetos que se conocen o se conocerán cuando la propagación los haya enontrado. Así, una vez que la relajación ha resuelto el ciclo, se usa directamente la regla sustituida para calcular el valor de la variable del nodo podado, sin tener que hacer el paso final de propagación local.

La procompilación no se limita necesariamente a gráficas con ciclos y puede ser utilizada en cualquier gráfica de restricciones. La ventaja de la técnica de propagación con poda temporal sobre la de propagación local cuando la gráfica no contiene ciclos es que no se tiene que hacer una propagación completa para cada valor de la variable buscada. Consideremos la gráfica del programa de conversión de temperatura de la figura 1.9.

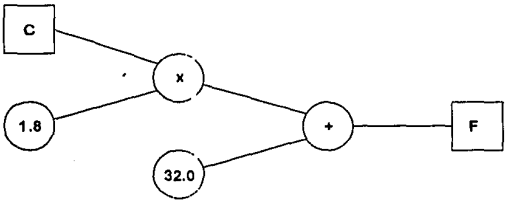


Figura 1.9.- Gráfica inicial de un programa de conversión de temperatura.

Si se desea usar el programa para encontrar los valores en grados Centígrados a partir de grados Fahrenheit, es posible precompilarlo propagando con poda temporal. Si *F* es el dato de entrada el objeto C puede ser podado, compilando la deducción $C \leftarrow T_t / 1.8$, donde T_t es una variable emporal. La gráfica resultante se muestra en la figura 1.10. Como la variable T_t tiene sólo una restricción asociada, podemos quitarla temporalmente y compilar la deducción $T_t \leftarrow F - 32$. Este proceso continua hasta que todos los nodos son removidos de la gráfica. En este ejemplo, hemos compilado las deducciones $T_t \leftarrow F - 32$ y $C \leftarrow T_t / 1.8$ que juntas forman el procedimiento $C \leftarrow (F - 32) / 1.8$. Observe que si se hubiera usado la técnica de propagación local se habría requerido de una propagación completa para cada nuevo valor de *F*.

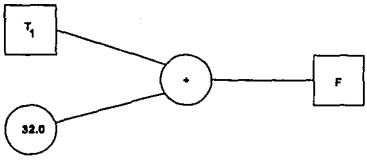


Figura 1.10.- Forma de la gráfica de restricciones después de la primera deducción.

La propagación con poda temporal es ligeramente menos poderosa que la propagación local ya que no siempre es posible aprovechar todas las reglas asociadas a un nodo. Considere, por ejemplo, la regla en un nodo de multiplicación para la multiplicación por cero (figura 1.11). Si llega un valor cero ya sea para *p* o *q*, entonces el valor de cero puede ser disparado hacia *r* sin haber esperado el valor del otro arco. Sin embargo, esta regla no podría aprovecharse si el esquema se encontrara precompilado.

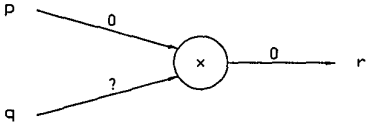


Figura 1.11.- Propagación del cero a través de un nodo por (x).

1.4.4. Retractación.

Pudiera ser que cuando se está trabajando con satisfacción de restricciones, el usuario decida modificar alguna restricción. Por ejemplo, que la restricción fuera $a = 0$ y el usuario decidiera asignar a la variable *a* otro valor diferente de cero. Si solamente cambia el valor de un objeto; es decir, si el cambio no afecta la forma de la gráfica de restricciones, entonces esos cambios pueden ser manipulados por una forma de propagación conocida como *Retractación,* que es un tipo de propagación en el que los cambios son propagados a través de una gráfica de restricciones, retractando el valor anterior (dejándolo sin valor), que a su vez retracta otros valores que fueron obtenidos a partir de él y luego propagando los nuevos valores. Para resolver restricciones que envuelven tiempo, la retractación es usada como si el tiempo fuera invariante y para cada nuevo valor de tiempo, se retracta el valor anterior y se propaga el nuevo.

Aunque la retractación es muy usada en sistemas de satisfacción de restricciones que usan propagación local, tiene algunos problemas. El cambio de un valor en una gráfica puede desencadenar el cambio de una o más de las ecuaciones originales y determinar cuál de ellas cambiar depende de la intención del usuario lo que convierte al cambio en un problema difícil que generalmente es resuelto por medio de alguno de los siguientes métodos.

1.- Tomar las premisas que causaron el menor cambio en el resto de la gráfica. Esta solución requiere determinar cuál premisa resultó en el menor número de propagaciones y puede producir un resultado equivocado.

2.- Tomar las premisas que son más cercanas al cambio, aunque este método es más fácil, comúnmente es más erróneo.

3.- Hacer que el usuario especifique cuál premisa cambiar. Esto puede confundir al usuario, especialmente si no sabe qué premisas quiere cambiar.

4.- El último método y el más usado consiste en tomar algunas premisas aleatoriamente permitiendo que el usuario decida si la elección estuvo equivocada.

La retractación tiene algunos problemas como es el hecho de tratar de adivinar las intenciones del usuario cuando toma algunas premisas a cambiar, también es común que produzca retracciones innecesarias y recalculaciones de variables cuyos valores no son afectados por el cambio.

1.4.5. Vistas Redundantes.

Cuando los problemas de restricciones no pueden ser resueltos usando relajación, se puede usar el método de vistas redundantes. Este método permite al usuario ayudar al sistema a resolver problemas que van más allá de las capacidades del método. Consiste en que el usuario proporcione al sistema una vista adicional de la restricción para que ésta pueda ser resuelta usando propagación local. La vista redundante consiste en escribir la restricción en una forma alternativa más equivalente desde el punto de vista lógico. Por ejemplo, considere las siguientes restricciones de un programa:

$$5 \times X = X$$
$$X + X = 40$$

La restricción $X + X = 40$ no puede ser resuelta usando propagación local debido al ciclo que contiene, pero puede ser resuelta si el usuario le suministra la restricción equivalente:

$$2 \times X = 40$$

Sería preferible que el sistema pudiera usar alguna técnica de propagación local que descubriera las partes que no pueda resolver para transformarlas en gráficas equivalentes accesibles al método. Esto es exactamente lo que hace el método llamado Transformación Gráfica.

1.4.6. Transformación Gráfica.

El método de Transformación Gráfica utiliza reglas de reescritura para transformar subgráficas de un programa de restricciones en otras más simples. Estas reglas sirven para escribir una expresión en otra equivalente. Por ejemplo, una regla de reescritura es:

$$V + V = 2 \times V$$

donde V es una variable que se puede igualar a cualquier expresión. Se pueden usar otras reglas para hacer más reglas de reescritura, tales como las distributivas. Para resolver las ecuaciones:

$$5 \times Y = X$$
$$X + X = 40$$

un sistema con transformación gráfica transforma la gráfica de la figura 1.8 en la gráfica de la figura 1.12 que se muestra a continuación.

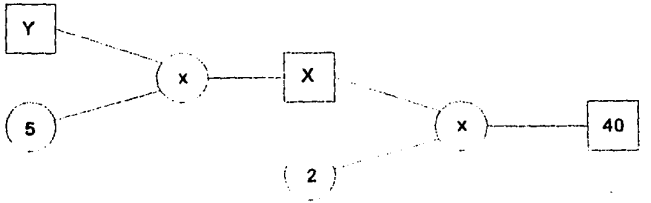


Figura 1.12.- Gráfica obtenida con ayuda de reglas de rescritura.

El método de transformación gráfica es más poderosos que el método de propagación local, ya que ésta sólo trabaja sobre un único nodo y los valores en los

arcos conectados a él. La transformación gráfica opera sobre subgráficas tomadas como entidades individuales; sin embargo, también tiene sus desventajas ya que sólo puede transformar ciclos simples (tales como $X + X$). Los ciclos formados por ecuaciones simultáneas más complejas no pueden ser resueltos usando la transformación gráfica a excepción de unos cuantos casos. Resulta conveniente, por lo tanto, hacer una revisión de los métodos que se han presentado desde la definición del término *restricción*. Podemos observar que conforme incrementamos el nivel de dificultad en una gráfica de restricciones también presentamos un método con la habilidad de resolverlo. Aunado a esto también se observa que conforme el método es más especializado en algún aspecto, es menos práctico para poder ser usado como método de propósito general. Podemos decir que existe un compromiso entre generalidad y eficiencia de tal manera que mientras más eficiente sea un método, más específico será el conjunto de problemas que éste pueda resolver. No podemos esperar que existan métodos especializados para resolver todos los problemas que se puedan plantear en forma arbitraria y por lo tanto nos vemos en la necesidad de abordar el problema general de resolver sistemas de restricciones representados directamente por sistemas de ecuaciones algebraicas.

1.4.7. Solución de Ecuaciones.

En los métodos explicados anteriormente las restricciones estaban escritas en ecuaciones, luego eran convertidas a gráficas de restricciones. Para los métodos que se discuten en esta sección, se resuelven directamente las ecuaciones. Un algoritmo muy usado para resolver restricciones escritas en forma de ecuaciones es el conocido como Resolvedor Simple de Ecuaciones. Para ver de que manera funciona éste, veamos antes que pasos sigue una persona para resolver un sistema de ecuaciones. Suponga usted que un conjunto dado de restricciones puede ser expresado por medio del siguiente sistema:

$$\begin{aligned} &(1) \quad p + 2p + 5q = 14 \\ &(2) \quad p - q = 2 \end{aligned}$$

note que si *(1)* se expresara mediante una gráfica de restricciones, ésta contendría un ciclo debido a la subexpresión $p + 2p$. Considere, sin embargo, que el ciclo puede ser evitado combinando ambos términos, como se muestra en (3).

$$\begin{aligned} &(3) \quad 3p + 5q = 14 \\ &(4) \quad p - q = 2 \end{aligned}$$

El sistema puede ser resuelto de la siguiente forma:

1.- Se despeja p en *(4)*, obteniendo

$$(5) \quad p = 2 + q$$

2.- Se sustituye p en *(3)* por su valor $(2 + q)$, lo que produce la ecuación:

$$(6) \quad 3\,(2 + q) + 5\,q = 14.$$

3.- Se despeja q de *(5)*, de dónde obtenemos que $q = 3$.

4.- Sustituimos el valor de q en *(5)* y obtenemos el valor de p que es 5.

El procedimiento anterior ilustra en forma particular, la solución del sistema de las ecuaciones lineales dadas. El procedimiento que sigue el Resolvedor Simple de Ecuaciones está basado principalmente en el método de eliminación gaussiana[2]. El procedimiento que sigue el Resolvedor Simple de Ecuaciones consiste en convertir cada ecuación en una expresión cuyas variables estén ordenadas. Para este fin se puede usar cualquier método de ordenación total entre distintas variables, como el orden lexicográfico. El orden lexicográfico es una forma de ordenar los términos de una expresión en orden alfabético y si los términos de una ecuación contienen más de una letra, éstos son ordenados como se hace con las palabras en un diccionario, esto es, cada letra de un término también es ordenada alfabéticamente. Por ejemplo si tenemos la ecuación $sr + p + q$, y la ordenamos en forma lexicográfica se convertirá en: $p + q + rs$. Una vez que las ecuaciones estén convertidas en expresiones cuyas variables hayan sido ordenadas, se procede a agrupar todos los términos que tienen las mismas variables (sumando sus coeficientes). En el ejemplo anterior, la subexpresión $p + 2p$ fué agrupada en $3p$. Las ecuaciones ordenadas y agrupadas de esta forma son conocidas como ecuaciones lineales ordenadas.

[2]El Resolvedor Simple de ecuaciones no es el proceso de Eliminación Gaussiana completo porque no abarca, por ejemplo, las estrategias de pivoteo.

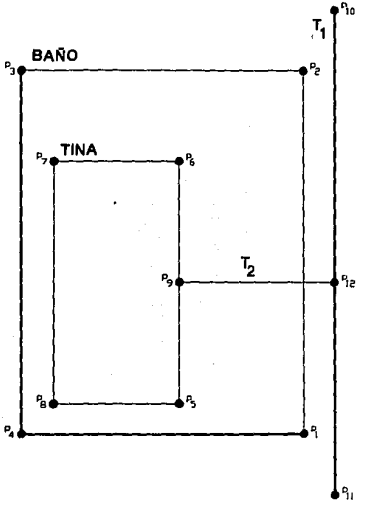


Figura 1.13.- Dibujo de un cuarto de baño en el que los vértices de los objetos básicos se denotan para crear las ecuaciones del dibujo.

Después de que las ecuaciones son convertidas en ecuaciones lineales ordenadas se igualan a cero y son resueltas una a una. Existen cuatro soluciones posibles a saber:

1.- Si la combinación lineal ordenada es la constante cero, tenemos que $0 = 0$, que implica que la ecuación era redundante y es desechada.

2.- Si la combinación lineal ordenada es una constante k diferente de cero, tenemos que $k = 0$, lo que es una contradicción, esto comúnmente se manipula enviando un mensaje de error y dando por terminado el proceso.

3.- Si la combinación lineal ordenada contiene una variable, como $A \times X + B$, donde A y B son constantes y X una variable, entonces tenemos que $X = -B/A$, borramos

esta combinación lineal ordenada y reemplazamos todas las ocurrencias de esta variable en otras ecuaciones por su valor ($-B/A$).

4.- Si la combinación lineal contiene más de una variable, entonces tomamos una de las variables y la resolvemos en términos de las otras variables. Nuevamente borramos la combinación lineal ordenada y reemplazamos todas las ocurrencias de esa variable por su valor. No importa cuál variable es tomada para resolverlo.

Hagamos un breve paréntesis para analizar este método y sus características. El algoritmo descrito en los párrafos anteriores es fácil de programar pero con él no es posible resolver ecuaciones con desigualdades. Las desigualdades son situaciones que también se presentanentre las restricciones gráficas. Un ejemplo de este tipo de relaciones sería, por ejemplo, que el usuario indicara que el ángulo de una línea dada no debiera ser mayor a un valor α, es decir, que la línea pueda tomar cualquier valor menor o igual que α. Las restricciones con desigualdades no pueden ser resueltas por Eliminación Gaussiana, pero pueden ser resueltas por el método Simplex (Schuarz trata ampliamente el algoritmo Simplex y el de Eliminación Gaussiana [Schuarz 89]). Un problema con la solución de ecuaciones es que al no existir gráfica de restricciones, no hay forma de evitar todos los cálculos para cada nuevo valor de las variables en el sistema de ecuaciones. Además el proceso de traducción de las restricciones de dibujo geométrico en ecuaciones no es un problema sencillo y resultaría muy osado pretender que el usuario del sistema escriba las ecuaciones del dibujo y de sus restricciones. Considere nuevamente el ejemplo inicial de este capítulo en el que se dibujó un cuarto de baño (véase la figura 1.13) en el que nuestra intención era poder desplazar la tina manteniendo la evacuación de agua funcional. Para resolver este problema por algún método de solución de ecuaciones se tienen que escribir ecuaciones del dibujo y sus restricciones. Una forma de hacer esto es, de acuerdo a Santana [Santana 87] como se muestra a continuación: Asignamos un nombre P_i (X_i,Y_i) a cada uno de los vértices del dibujo, dónde $i = 1$ a n y n = número de vértices en el dibujo (figura 1.13). Llamemos al estado original del dibujo "estado 1" y al estado después del cambio "estado 2". La nomenclatura de los puntos es :

$$P_1 = (X_1, Y_1)$$

para los puntos en el estado 1, y :

$$\dot{P}_1 = (\dot{X}_1, \dot{Y}_1)$$

para puntos en el estado 2.

Las diferencias de un punto entre el estado 1 y el estado 2 son

(1) $\Delta X_i = \dot{X}_i - X_i$

(2) $\Delta Y_i = \dot{Y}_i - Y_i$

La distancia de un punto entre el estado 1 y el estado 2 esta dada por

(3) $\sqrt{\Delta^2 X_i + \Delta^2 Y_i}$

Sea P_5 el punto que se quiere mover hasta una posición P_0. En el estado 2 P_5 es $\dot{P}_5$, la diferencia entre $\dot{P}_5$ y P_0 (el sitio a donde se quería mover originalmente) es:

(4) $DX_0 = \dot{X}_5 - X_0$

(5) $DY_0 = \dot{Y}_5 - Y_0$

El problema consiste entonces en ecotrar el vector

(6) $\hat{Q} = (\dot{X}_1, \dot{X}_2, \dot{X}_3, \ldots, \dot{X}_{12}, \dot{Y}_1, \dot{Y}_2, \dot{Y}_3, \ldots, \dot{Y}_{12})$

tal que se minimiza la función

(7) $S(\hat{Q}) = \sum_{i=1}^{12} (\Delta^2 X_i + \Delta^2 Y_i) - \Delta^2 X_5 - \Delta^2 Y_3 + 2.3\,(D^2 X_0 + D^2 Y_0)$

Para entender el plantemiento en (7) consdere a la figura 1.14 en la que los puntos P_i conforman un dibujo original y los puntos $\dot{P}_i$ conforman las nuevas posiciones de los puntos P_i obtenidas luego de indicar que se traslade el punto P_1 hasta la posición P_0. Note que la posición P_0 no es donde finalmente quedó trasladado el punto P_1 sino en la posición $\dot{P}_i$; esto se debe a que cuando se resuelven las ecuaciones los valores que se encuentran para las nuevas posiciones de los puntos pueden ser tan distantes que la nueva posición de P_1 no es cercana a P_0 por lo que

es necesario encontrar la mínima distancia entre P_1 y P_0 (proceso de minimización) resultando el punto $\dot{P}_1$.

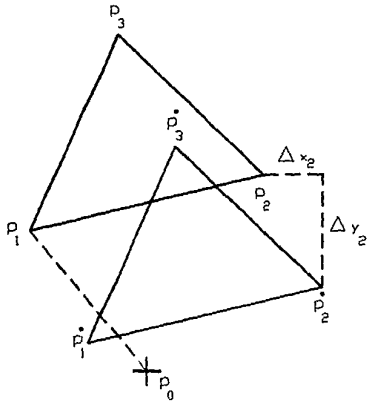


Figura 1.14.- Desplazamiento de una figura tomando como pivote al punto P1 cuya nueva posición se pide en la posición P_0.

Volvamos a la función objetivo (7), examinemosla en dos partes. La parte *A* es:

$$\sum_{i=1}^{12} (\Delta^2 X_i + \Delta^2 Y_i) - \Delta^2 X_5 - \Delta^2 Y_5$$

y la parte *B*:

$$23\,(D^2 X_0 + D^2 Y_0)$$

La parte *A* representa la suma de los cuadrados de las distancias de todos los puntos entre el estado 1 y el estado2 excepto para el punto 5, que es el punto que se quiere mover (el que sirve como referencia para mover todo lo demás).

La parte *B* representa el cuadrado de la distancia entre el punto $\dot{P}_5$ y el punto p_0. Este último es la posición a dónde se quiere mover el punto p_5.

La ecuación (7) tiene dos objetivos, el objetivo principal es que se cumpla el requerimiento del usuario, lo que equivale a que *B* sea cero. El objetivo secundario es que respetando las restricciones impuestas, todos los otros puntos se muevan lo menos posible, lo que equivale a minimizar *A* .

El coeficiente del cuarto término de la ecuación (7) se obtiene del factor 2n-1, en donde n es el numero de variables; dicho factor fué obtenido por [Moreno 93].

Continuando con el establecimiento de las ecuaciones del problema tenemos que éste está sujeto a:

a) Restricciones de incidencia:

Que P_9 incida sobre la recta de va del punto P_6 al punto P_5:

(8) $(X_9-X_5)\,(Y_6-Y_5) = (Y_9-Y_5)\,(X_6-X_5)$

Que P_{12} incida sobre la recta de va del punto P_{11} al punto P_{10}:

(9) $(X_{12}-X_{10})\,(Y_{11}-Y_{10}) = (Y_{12}-Y_{10})\,(X_{11}-X_{10})$

b) Restricciones de permanencia de dimensiones:

dimensiones de la tina:

(10) $X_5-X_6 = A$

(11) $Y_7-Y_6 = B$

dimensiones del cuarto:

(12) $X_1-X_4 = C$

(13) $Y_3-Y_4 = D$

Hasta aquí la definición del problema. Para solucionarlo es necesario linealizar debido a los términos cuadráticos en (7), (8) y (9). De acuerdo con Moreno [Moreno 93], el proceso de solución nos llevará a producir matrices de un número elevado de elementos. Para un problema real, con un mayor número de objetos y con

restricciones de más complejidad, es más difícil el proceso de traducción y por ende su solución. Sobre esta ruta se ha seguido trabajando. Un ejemplo es el trabajo de [Moreno 93] en el que se resuelven restricciones por programación matemática aplicando el método Simplex a los sistemas de ecuaciones y cuyos resultados dejan entrever la necesidad de crear heurísticas que nos permitan aprovechar la experiencia del ser humano. Dado lo cual, la posición que comparte este trabajo con Pineda [Pineda 89] es la de utilizar herramientas de Inteligencia Artificial para la solución del problema. A continuación se ilustra el tipo de métodos que se emplean en Inteligencia Artificial para resolver restricciones.

1.4.8. Satisfacción de Restricciones en Inteligencia Artificial.[3]

En Inteligencia Artificial existe un método para resolver problemas de restricciones llamado simplemente Satisfacción de Restricciones. En general, en Inteligencia Artificial los problemas son atacados por métodos de búsqueda. Todo proceso de búsqueda puede ser visto como un recorrido sobre una gráfica dirigida en la cual un nodo representa un estado del problema. El problema se encuentra en diferentes estados conforme se avanza en su proceso de solución, es decir, va desde su estado inicial (únicamente el problema con sus restricciones iniciales), hasta un estado en que el problema satisface su conjunto de restricciones. En la gráfica dirigida cada arco representa una relación entre los estados del problema (nodos). Generalmente los arcos indican que reglas o que valores se asumieron en las variables del problema para llegar a el siguiente estado. El proceso de usar búsquedas para resolver problemas consiste en aplicar las reglas apropiadas a estados de problemas específicos para generar nuevos estados a los que se puedan aplicar las reglas y así sucesivamente hasta que se encuentra una solución.

A grandes rasgos, el método de satisfacción de restricciones consiste en un proceso de búsqueda sistemática y si dicho proceso conduce a una contradicción entre las restricciones del problema, el proceso retorna a la última decisión que tomó durante su búsqueda y escoge un camino alternativo. La solución del problema es encontrada cuando el proceso de búsqueda llega a un estado en que todas la restricciones del problema son satisfechas. Dada la estructura de los problemas de satisfacción de restricciones, para resolverlos es útil aumentar la descripción del estado del problema con una lista de restricciones que cambia conforme se resuleven las piezas del problema; esta lista de restricciones contiene a las restricciones iniciales

[3] Basado en Artificial Inteligence de Elaine Rich [Rich 83].

y a las nuevas suposiciones hechas durante el proceso. La forma general del procedimiento de satisfacción de restricciones es el siguiente:

1.- Hacer hasta que se encuentre una solución completa o hasta que todas las rutas hayan conducido a contradicciones entre las restricciones nuevas inferidas y el resto del conjunto de restricciones (a estas contradicciones se les conoce como finales muertos):

   1.- Seleccionar un nodo de la gráfica de búsqueda que no haya sido seleccionado antes.

   2.- Aplicar las reglas de inferencias de restricciones al nodo seleccionado para generar posibles restricciones nuevas que contribuyan a la solución del problema.

   3.- Si el conjunto de restricciones contiene una contradicción entonces se reporta que esta ruta es un fin muerto.

   4.- Si el conjunto de restricciones describe una solución completa enton-ces se reporta éxito.

   5.- Si no se encuentra solución ni contradicción entonces se aplican reglas para generar soluciones parciales nuevas que sean consistentes con el conjunto de restricciones que se tengan hasta entonces. Se insertan estas soluciones en la gráfica de búsqueda.

Se ha observado que es este tipo de procesos es mejor hacer suposiciones en variables cuyos valores tienen mayor número de restricciones en vez de hacer suposiciones sobre variables cuyos valores apenas están restringidos, Esto se debe a que si hay un número menor de elecciones posibles durante el proceso, entonces cualquiera de ellas tiene mayor probabilidad de ser correcta que si el número de elecciones posible es muy grande. Las decisiones sobre nuevas suposiciones cuando no existen más datos para la solución en ese estado son basadas en heurísticas que generalmente llevan a la solución.

Veamos de que manera es útil este método para nuestros propósitos. Suponga que tenemos un dibujo en el cual algunos objetos están restringidos, si dicho dibujo sufre alguna modificación, el sistema debe de poder encontrar un estado en el cual las restricciones se cumplen. El método de satisfacción de restricciones sirve para este propósito porque va resolviendo paulatinamente las restricciones de modo que

al resolver unas (al ir encontrando valores para las variables restringidas) es más fácil resolver las restantes. La ventaja de este método sobre el anterior es que si se llega a una contradicción entre los valores de las variables, és decir, si se llega a una inconsistencia en el sistema, es posible regresar de una en una sobre las decisiones tomadas y escoger otro valor de la varible que ocupe a cada estado. Otra ventaja más de este método es que es posible resolver desigualdades (véanse problemas criptoaritméticos [Leller 88].

Hasta este punto hemos mencionado a los métodos de satisfacción de restricciones más conocidos y usados. Antes de explicar la forma en que se resuelven los problemas en esta tesis se muestra en la siguiente sección ejemplos de sistemas que usan los métodos descritos en esta sección.

1.5. Sistemas de Satisfacción de Restricciones Existentes.

Como ya hemos dicho, los problemas de satisfacción de restricciones y las técnicas usadas para resolverlos tienen ya su historia dentro del mundo de la computación. A medida que estas técnicas se vuelven mas sofisticadas, tienden a ser de aplicaciones mas especificas. Los sistemas explicados a continuación usan algunas de las técnicas descritas en la sección 1.3. Estos no han sido los únicos sistemas que han sido diseñados hasta hoy para resolver el problema que nos ocupa, sino que son de los más conocidos debido a sus características. En cada uno de ellos se menciona qué técnica o combinación de técnicas utiliza para resolver las restricciones.

Sketchpad.

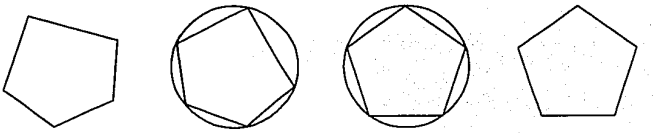


Figura 1.15.- Dibujando un pentágono regular usando Sketchpad.

Este sistema fué creado por Ivan Sutherland, conocido como el padre de la computación gráfica debido principalmente a este trabajo que es una parte de su tesis doctoral. Con el sistema Sketchpad uno podría dibujar un objeto complejo, trazando una figura simple y añadiéndole restricciones. Por ejemplo, los pasos

requeridos en Sketchpad para construir un pentágono regular son: Trazar un polígono regular de cinco lados (fig 1.15). Mientras se traza, la línea actual que está siendo descrita actúa como una banda de hule entre el fin de la última línea y la posición del cursor para darle al usuario retroalimentación inmediata. Cuando el cursor se aproxima a algún punto existente como un fin de línea, entonces se cierra ahí, de manera que figuras como los polígonos pueden ser trazadas fácilmente. Para hacer regular el polígono, se restringen sus vértices a yacer en un círculo y sus lados de igual longitud. Por último, se borra el círculo con que fue restringido el dibujo dejando un pentágono regular.

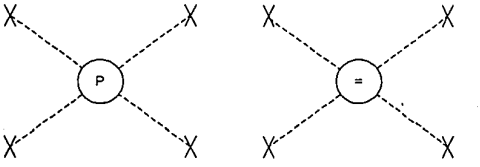


Figura 1.16.- Restricciones abstractas de Sketchpad. Los puntos delimitados por las cruces indican los extremos de las líneas.

En Sketchpad los objetos gráficos primitivos son los puntos, las líneas y los arcos circulares. Cualquier dibujo, como el pentágono dibujado, puede ser usado como un objeto primitivo convirtiéndolo en una *macro*. Una macro tiene un conjunto de puntos unidos que son usados para fundir una instancia de estos dentro de otro dibujo. Las restricciones primitivas incluyen hacer dos líneas paralelas, perpendiculares o de igual longitud. Las líneas también pueden ser hechas horizontales o verticales y un punto puede ser restringido a permanecer en un arco o en una línea. Las restricciones son representadas abstractamente en la pantalla como un nodo circular que contiene un símbolo, Por ejemplo, la figura 1.16 contiene las representaciones gráficas de la restricción para hacer dos líneas paralelas y de la restricción para hacer dos líneas de igual longitud.

Las líneas discontinuas de la figura 1.16 conectan a la restricción con sus puntos asociados. En este caso las restricciones no están asociadas a nada. La facilidad de macro puede ser usada para simular la suma de nuevas restricciones. Por ejemplo, la figura 1.17 muestra dos líneas que están restringidas a ser paralelas y de la misma magnitud.

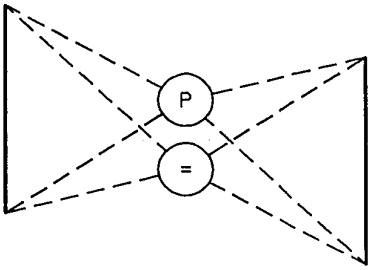


Figura 1.17.- Creando una macro.

Podemos usar el dibujo anterior como una restricción, convirtiéndolo en una macro con las dos líneas como puntos de adherencia. Cuando queremos usar esta restricción se llama a una instancia de la macro y se *fusionan* las líneas adheridas con las líneas que queramos restringir a que sean paralelas y de igual longitud. Una fusión es como una restricción que hace dos objetos iguales, excepto que los dos objetos en realidad son reemplazados para ser el mismo objeto equivalente. Las fusiones son realizadas recursivamente, de manera que todos los objetos son también fusionados. Por ejemplo, cuando dos líneas son fusionadas, sus puntos terminales son fusionados; cuando dos puntos son fusionados, sus coordenadas *x* y *y* son fusionadas. La mezcla de dos valores escalares es una primitiva que hace dichos valores iguales. El programa Sketchpad satisface las restricciones por medio de un método que Sutherland llamó Método de un Paso, que no es otra cosa sino propagación con poda temporal y cuando este método falla se usa el método de relajación. Para los valores iniciales del paso de relajación el Sketchpad usa los valores que en ese momento se encuentren en el dibujo en la pantalla. Esto sumado al hecho de que las restricciones geométricas que usa Sketchpad son bastante bien comportadas, hace de la relajación un método razonablemente rápido e interactivo.

Desafortunadamente, el Sketchpad usaba entrada gráfica y hardware de display muy costosos para su tiempo. Su modo de operación interactivo depende de un ancho de banda alto de comunicación entre el usuario y el procesador, un ancho de banda que no podía ser soportado por sistemas de computación de tiempo compartido hasta que las estaciones de trabajo se hicieron económicamente justificables.

ThingLab.

Existen dos importantes versiones de este sistema: *ThingLab* y *ThingLab II*. Ambos son sistemas interactivos de satsifacción de restricciones orientados a objetos, programados en Smalltalk 80. En particular, ThingLab (la primera versión) "proporciona al usuario un ambiente propicio para construir modelos dinámicos de experimentos en geometría y física tales como simulaciones de objetos geométricos restringidos, circuitos eléctricos simples, acoplamientos mecánicos,..." [Borning 86a pp 271]. Fué escrito por Alan Borning como parte de su tesis doctoral en Stanford. El ThingLab II es un sistema diseñado para la construcción de interfaces de usuario. Cuenta con una librería de objetos primitivos que son usados como bloques de construcción. El comportamiento generalmente es especificado con restricciones. Cada objeto tiene una lista de restricciones que lo caracterizan y cada restricción tiene un conjunto de procedimientos que pueden ser invocados para satisfacerla, por ejemplo: Si uno desea restringir que la variable A sea igual a la suma de las variables B y C, esta restricción tendría tres métodos que la satisfacerían:

B + C = A cuando se conocen B y C

A - C = B cuando se conocen A y C, y

A - B = C cuando se conocen A y B.

El sistema cuenta con un intérprete de restricciones. El intérprete es el encargado de añadir o suprimir restricciones en respuesta a acciones del usuario sobre el dibujo. El mecanismo de satisfacción de restricciones está basado en gráficas de restricciones y en ThingLab II es posible editar dichas gráficas interactivamente. Además de los métodos de satisfacción de restricciones usados por Sketchpad, ThingLab también usa propagación local de estados conocidos. El ThingLab gana velocidad dividiendo la satisfacción de restricciones en dos estados: planeación y tiempo de corrida. Durante la planeación, un procedimiento SmallTalk, llamado un *método* es generado automáticamente para satisfacer un conjunto de restricciones. Este método puede entonces ser invocado repetidamente en tiempo de corrida. Por ejemplo, la esquina de un rectángulo puede ser llevada alrededor de la pantalla en tiempo real aproximadamente y el rectángulo se irá moviendo mientras que las restricciones van siendo satisfechas por el procedimiento compilado. Los problemas que se presentan en ThingLab son los asociados a la propagación local y a la relajación.

Juno

El sistema Juno está constituido por un lenguaje de descripción de figuras y editor de imágenes *wyswyg*[4]. El lenguaje de Juno es relativamente fácil para programar, sus variables representan puntos en los que se aplican cuatro restricciones en las que una línea es referida por sus puntos terminales:

HOR (p,q)	la línea (p,q) es horizontal.
VER (p,q)	la línea (p,q) es vertical.
(p,q) PARA (r,s)	las líneas (p,q) y (r,s) son paralelas.
(p,q) CONG (r,s)	las líneas (p,q) y (r,s) son de igual longitud.

Como las restricciones actúan sobre puntos y no sobre líneas, pueden existir restricciones entre dos puntos sin que realmente exista una línea entre ellos. Por ejemplo, la restricción VER (m,n) solamente especifica que los puntos m y n tienen la misma coordenada *x*.

En Juno, las restricciones son representadas gráficamente como iconos. Por ejemplo, la restricción HOR es representada como un cuadro orientado horizontalmente y la restricción CONG como un compás de trazo. Si el usuario selecciona el icono del compás y escoge cuatro puntos, entonces queda insertada la restricción CONG en el programa. Juno usa una forma de relajación para la satisfacción de restricciones. Como las restricciones PARA y CONG son cuadráticas, la propagación no es de mucha utilidad pero tampoco se requerirfa una relajación general. De hecho Juno utiliza iteraciones de Newton-Raphson y aunque este método es mucho más rápido que la relajación, toma un tiempo proporcional al cubo del número de variables. Para acelerar las iteraciones, Juno le pide al usuario una suposición del valor inicial ya sea textual o gráficamente.

Entre las principales contribuciones de Juno está la forma en que representa las restricciones gráficamente y la habilidad de construir automáticamente un programa de restricciones a partir de una representación gráfica. Para Juno hacer dibujos es como hacer figuras geométricas con un compás y una regla. Pero el dominio pensado para Juno es muy limitado, su único dato-objeto es el punto y sobre él sólo hay cuatro restricciones aplicables.

[4] Palabra formada a partir de las iniciales de la frase en inglés: what you see what you get (lo que usted ve es lo que tiene).

TK!Solver.

El TK!Solver [Konopa 84] es un sistema de solución de problemas de propósito general, muy similar al ThingLab, las restricciones a ser resueltas son declaradas al sistema como ecuaciones que en primera instancia se solucionan por propagación local y si no pueden ser resueltas de esa manera, entonces se les busca solución por relajación.

Magrite.

Magrite [Gosling 83] es un sistema interactivo de tipo gráfico y aunque es muy parecido al sistema Sketchpad y al ThingLab, difiere de ellos en que no usa relajación sino técnicas algebraicas para transformar las gráficas que no pueden ser resueltas usando propagación local en gráficas que si puedan ser resueltas de este modo.

IDEAL.

IDEAL [VanWyk 80] es un lenguaje para hacer gráficos en documentos. En él, las restricciones son expresadas como ecuaciones que son resueltas por algoritmos de solución de ecuaciones lineales. Esto limita a las restricciones que pueden ser resueltas en sólo aquellas que se reducen a relaciones lineales. En este sistema, los objetos primitivos son números complejos por lo que cada restricción es en realidad dos restricciones una de parte real y otra en la parte imaginaria.

Graflog.

El sistema Graflog es un sistema gráfico interactivo que permite definir objetos gráficos y aplicar restricciones sobre ellos, restricciones que son satisfechas en el transcurso de la sesión interactiva. El sistema fué creado por L. A. Pineda como parte de su tesis doctoral en la Universidad de Edimburgo y difiere en la forma de ver al problema de satisfacción de restricciones del resto de los autores mencionados. Según el mismo Pineda: "...conceptualizar el problema de satisfacción de restricciones como un problema simbólico en lugar de numérico contribuye a sentar con mayor claridad la noción de "restricción", a simplificar los métodos de solución y a explicar el proceso inferencial intuitivo subyacente en la mocificación de dibujos en el curso de las sesiones de trazo interactivas" [Pineda 92a, pp 1]. En Graflog los objetos gráficos se manipulan como símbolos gráficos definidos en un lenguaje de representación siguiendo la idea del mismo Pineda que dice que un dibujo puede ser descrito completamente por términos lingüísticos que sirven como nombres o descripciones de los símbolos gráficos que conforman el dibujo.

El graflog cuenta con un lenguaje gráfico, un intérprete de dicho lenguaje, un conjunto de reglas heurísticas para resolver restricciones, una interface gráfica con botones que permiten crear los objetos, restringirlos y hacer consultas al sistema. Los objetos gráficos que el Graflog conoce son el punto, la línea, polilíneas, polígonos y arcos principalmente, pero las manipulaciones y los razonamientos de las heurísticas para resolver restricciones estan en función de los puntos; por lo que la manipulación conocida es la traslación de un punto de un objeto dado. El Graflog cubre también algunos aspectos de solución de problemas de Lenguaje Natural.

Por sus características gráficas, el Graflog sirve como punto de partida para la elaboración de este trabajo. De Graflog se toma la idea principal que soporta este trabajo que es la de conceptualizar al problema de satisfacción de restricciones como simbólico creando un sistema de representación de los objetos para hacer la manipulación simbólica.

1.6 Motivación de la tesis.

A lo largo del capítulo hemos podido observar diversas formas de atacar el problema de satisfacción de restricciones y los sistemas en que estas técnicas han sido implementados. A excepción del método de inteligencia artificial descrito en la sección 1.3.7., los demás están basados en ecuaciones. Unos requieren que el usuario escriba las ecuaciones para que el sistema pueda hacer una gráfica de restricciones y trabajar sobre ella; ejemplos de sistemas de este tipo son TKISolver e IDEAL. Otros pueden por sí mismos hacer las gráficas a partir del dibujo, por ejemplo el sistema Juno y el ThingLab. Es conveniente resaltar que para un usuario familiarizado con computadoras pero inexperto en este tipo de sistemas, será difícil describir las condiciones que quiere que permanezcan en su dibujo en forma de ecuaciones. Ahora, para el caso de los sistemas que traducen al dibujo en ecuaciones, encontramos que el proceso de traducción incrementa el costo en tiempo para obtener una solución (sobre este punto volveremos más adelante). Querer solucionar el problema de satisfacción de restricciones presenta problemas más serios. Si se quiere resolver por eliminación gaussiana, encontramos como principal obstáculo que en restricciones gráficas son comunes las desigualdades (sección 1.3.6.). Si se quiere resolver por ecuaciones lineales, resulta que al describir un dibujo en ecuaciones, comunmente no es lineal el sistema. El proceso de linealización y la solución por el método Simplex incrementan el número ecuaciones y el de variables (sección 1.3.6).

Con lo anterior no queremos decir que los sistemas que usan estas técnicas sean deficientes. Cada uno de ellos resuelve, con el método escogido, restricciones para alguna aplicación específica. Lo que sí se quiere decir es que la manipulación

simbólica de los objetos del dibujo es una opción que es útil, porque al tratar a los objetos de dibujo como entes abstractos, se evita el proceso de traducir un dibujo en ecuaciones y facilita que de una manera interactiva se pueda restringir a los objetos del dibujo aplicando las restricciones directamente en ellos, tomandolas como si fuesen elementos del dibujo. Esto es posible si el sistema cuenta con un lenguaje que permita referirnos a objetos gráficos y a las restricciones que pueden ser aplicadas sobre ellos. La manipulación simbólica permite razonar sobre los objetos en forma análoga al ser humano y no sobre las coordenadas, como lo hacen las técnicas numéricas examinadas. Consideremos el ejemplo inicial del cuarto de baño.

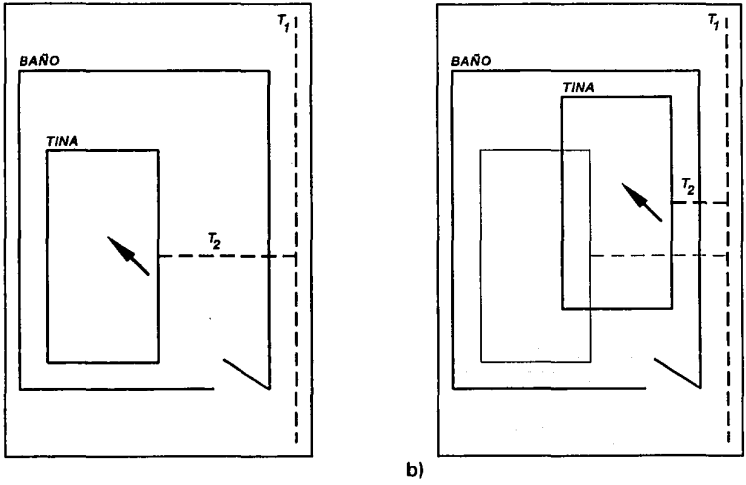


Figura 1.18.- Operación de arrastre sobre el dibujo de la tina. En la figura a) se muestra la figura inicial y la marca del ratón sobre la pantalla. En la figura b) se puede obserbar la operación de drag sobre el polígono y como el sistema de satisfacción de restricciones responde.

En el dibujo superior la línea que representa a la tubería T_1 la restringimos a permanecer vertical, la línea que representa a la tubería T_2 se restringe a permanecer horizontal. Además esta última línea es restringida a que uno de sus puntos extremos corresponda a el punto medio de la línea lateral derecha de la tina, el segundo punto se restringe a yacer sobre la línea que representa a la tubería T_1. De manera que

cuando se desplaza al polígono que representa a la tina, el sistema actualiza el dibujo de acuerdo a sus restricciones. Por lo tanto, una vez expresada nuestra intención de trasladar la tina de posición, son dibujadas en su nueva posición la tina y la tubería T_2 (como se muestra en la figura 1.18 b). Observe que el movimiento hecho por el usuario fué simplemente aplicado sobre la tina, pero como la restricción que asocia un extremo de T_2 con la posición de la tina entonces el cambio de T_2 es automático. Es decir, el foco de atención del usuario es la tina, pero dado que la posición de la tubería depende de la tina, aquella es afectada por la traslación de esta última. En este sistema no se crean ecuaciones del dibujo sino que se guarda en memoria una descripción del dibujo que contiene a todos los objetos del mismo junto con sus restricciones. De este lugar son tomados los objetos para ser manipulados por el sistema. Dicha descripción es hecha a partir de un lenguaje suficientemente rico que contiene a todos los objetos de nuestro interés y a las relaciones gráficas que sobre ellos queremos aplicar. Por medio de estas relaciones entre objetos simples nos podemos referir a objetos complejos y cuando modificamos unos se ven afectados los otros debido a las dependencias en el lenguaje entre ambos. En el siguiente capítulo se expone la definición del lenguaje y la forma en que el sistema interpreta las expresiones del lenguaje.

"Lo que nos rodea es el mundo representado, los estados de la mente son el mundo de representación, nuestras teorías de representación son en realidad representaciones de los estados de la mente..."

David E. Rumelhart

Capitulo 2.

Definición de un Lenguaje de Restricciones

2.1 Introducción.

Del capítulo anterior se puede resumir que el problema de satisfacción de restricciones consiste en contar con un mecanismo que permita hacer referencia a objetos gráficos y relaciones entre ellos y mantener estas relaciones durante una sesión interactiva. Mecanismo que integrado en un sistema de modelación facilite la creación de figuras geométricas complejas a través de la asociación de restricciones a figuras geométricas sencillas. Según Montalvo [Montalvo 86] a fín de poder hacer referencia más fácilmente a los objetos en el dominio gráfico, uno se puede apoyar en las descripciones simbólicas y en una maquinaria de inferencia aplicada ya sea a la interface gráfica del sistema, a la parte de la aplicación o a ambas. La mejor forma de capturar las descripciones y el razonamiento es con sistemas de representación del conocimiento. Según Rumelhart y Norman [Rumelhart 1983] una representación es

"algo que se entiende por otra cosa"[5]. En este terreno es importante distinguir entre el *mundo representado* (lo que queremos representar) y el *mundo de representación* (cómo vamos a representar al mundo representado). El mundo de representación debe reflejar de alguna forma aspectos del mundo representado. En nuestro caso, si tenemos una línea como parte del dibujo actual en la pantalla, su representación en nuestro sistema representacional puede ser la palabra *línea1* como el nombre de dicha línea. En adelante se verá cómo el conocimiento es representado a manera de una colección de símbolos en un lenguaje al que llamaremos *G*.

Un lenguaje es un conjunto de signos. Los signos están compuestos por dos aspectos, uno interno y otro externo [Sassure 74]. Sassure denominó al aspecto externo como "significante" (*signifiant* en palabras del mismo Sassure). El significante es el vehículo a través del cual el signo se hace presente para nosotros, es la materia, forma, sonido, luz u otro medio accesible a nuestros sentidos. El otro aspecto, el interno, al que Sassure llamó "significado" (*signifié*), es lo que se comunica; lo que nosotros entendemos al percibir el significante, es decir, lo que el signo representa. El vínculo entre ambos aspectos es asociado por convención. Quienes usan un determinado lenguaje, cualquiera que ése sea, están de acuerdo o conocen lo que representa cada signo y así se pueden entender.

Según Kolman [Kolman 84] un lenguaje es la especificación completa de los siguientes tres elementos:

- Debe existir un conjunto S que consiste de todas los símbolos que son parte del lenguaje y un subconjunto S^* consistente de todas las cadenas finitas formadas con elementos de S.

- Sintaxis. Se designa a un subconjunto de S^* como el conjunto de enunciados construidos apropiadamente en dicho lenguaje. Las reglas sintácticas establecen cuáles cadenas de S^* son las expresiones del lenguaje.

- Semántica. Conjunto de reglas que determinan el significado de los símbolos básicos y el significado de las expresiones compuestas.

De acuerdo a lo anterior, podemos ver que los objetos gráficos y las relaciones entre ellos pueden ser representados mediante un lenguaje. Dicho lenguaje de representación nos sirve como un lenguaje de satisfacción de restricciones dado que

[5]Del inglés "something that stands for something else".

éstos describen a los objetos que el sistema requiera dependiendo de los problemas que se deseen resolver y las restricciones relativas a dichos objetos. En el transcurso de éste capítulo veremos la especificación de un lenguaje de restricciones representacional y los conceptos relacionados con dicha especificación.

2.2 Especificacion Formal de un Lenguaje Gráfico.

Hasta ahora se ha discutido el uso del lenguaje para representar el conocimiento; sin embargo, esta no es su única utilidad también nos sirve para especificar formalmente las relaciones entre los elementos del lenguaje como las reglas de formación de expresiones válidas en el mismo. El propósito de una especificación formal es el de proveer una caracterización precisa de un sistema de software a un nivel conveniente de abstracción. Veamos dos conceptos importantes que involucra lo anterior: la caracterización del sistema y la palabra abstracción. Por caracterización del sistema entendemos a la determinación de las peculiaridades del sistema, por ejemplo, qué conceptos maneja (tipos de datos), qué relaciones existen, etc. Por abstracción entendemos al proceso de generalizar de tal forma que algunas características del objeto al que nos estemos refiriendo puedan ser ignoradas. Por ejemplo, podemos decir que el nivel de abstracción de nuestro sistema de modelación gráfica va a estar basado en coordenadas en la pantalla, así que cualquier figura que se quiera hacer tendrá que ser en base a las coordenadas de sus vértices; o podemos decir que el nivel de abstracción del sistema va a estar basado en puntos, entonces una línea será hecha en base a dos puntos. En el segundo ejemplo se muestra un nivel mayor de abstracción con respecto al primero ya que en aquel podemos hablar de puntos La característica que estamos ocultando es que estos puntos estan hechos en base de un par ordenado de números, o sea, de coordenadas.

De la especificación del lenguaje dependen el alcance y/o ventajas que presente el sistema con respecto a algún otro. Las especificaciones son útiles al hacer un sistema de software porque nos dicen cómo razonar cuando estamos haciendo un programa en dicho lenguaje y nos puede servir como segunda fuente de información acerca del sistema. A pesar de que no es en sí misma una documentación de usuario, puede servir de referencia cuando surgen preguntas sobre el comportamiento del sistema.

Procedemos ahora a definir un lenguaje de representación particular al que se llamará lenguaje G. Para poder comenzar con la especificación formal del lenguaje G definamos las figuras que queremos manipular. Dado que la idea del modelador es que se puedan realizar figuras complejas en base a figuras geométricas sencillas aplicando restricciones, se considera que es suficiente contar con puntos, líneas (en lo

subsecuente llamaremos líneas a los segmentos de rectas), circulos, polilíneas (secuencias de líneas) y polígonos. En los siguientes párrafos se verá como se llega a una especificación formal de un lenguaje siguiendo las ideas de Goguen et all [Goguen 73].

Creamos un conjunto con todos los símbolos (palabras) que el lenguaje va a contener. A todo el conjunto de símbolos al que llamaremos **S**, lo dividimos en subconjuntos **s** de acuerdo a la idea que transmiten, es decir al mensaje que consigo llevan. Por lo que a estos conjunto se les llama portadores. Entre los conjuntos así formados existen operaciones que producen objetos que corresponden a alguno de los conjuntos. Lo anterior es denominado por Goguen como un álgebra $\sum$ clasificada en **S** o un álgebra multiclasificada. Para especificar un lenguaje, en particular el lenguaje G, se reúnen en un conjunto a todos los elementos del mismo tipo. Si hay varios elementos de tipo **s** los reunimos en un conjunto al que llamamos portador del tipo **s**. Por ejemplo, los elementos que son del tipo línea los reunimos en el portador de tipo línea (línea). Esto significa que todas las líneas que podemos dibujar en la pantalla están contenidas en el conjunto portador línea. Cuando se escribe línea, se hace referencia simplemente al tipo de dato y no a una línea en particular.

Nuestra intención es definir la composición de las figuras básicas para obtener figuras complejas, por lo que es necesario habilitar un conjunto de operaciones sobre las figuras básicas. Por ejemplo, en la sesión interactiva de dibujo puede surgir la necesidad de hacer referencia al punto resultante de la intersección de dos líneas, o verificar si un par de líneas son paralelas. Para hacer estas referencias se requieren funciones que tomen sus argumentos de los portadores apropiados y produzcan valores en el portador correspondiente. Por ejemplo, la función *intersección*:línea,línea→punto es un operador que toma como valores dos objetos de tipo línea y produce como valor un elemento de tipo punto[6].

Se reúnen a los portadores en un conjunto **S** de tipos. Llamamos a los conjuntos portadores de la forma A_s. Por ejemplo, $A_{línea}$, A_{punto}, representan a los portadores de líneas y puntos. Los tipos de datos del lenguaje son conjuntos de esa forma. Un tipo de dato es un conjunto de objetos con operaciones que dan acceso a estos objetos. El valor de los tipos de datos reside en su propiedad de proveer abstracción. Consideramos el término abstraer en el sentido de enfocarse en propiedades que nos parezcan esenciales ignorando otras que no lo sean. Según Lizkov y Zilles [Lizkov 74]:

[6] En lo subsecuente aparecerán escritos los nombres de las clases subrayados y los nombres de las funciones en mayúsculas.

"Un tipo de dato abstracto define una especie de objetos abstractos los cuales están caracterizados completamente por las operaciones posibles en esos objetos... Cuando un operador hace uso de un dato_objeto abstracto, el operador está interesado sólo en el comportamiento que exhibe dicho objeto, pero no en los detalles de cómo ese comportamiento es logrado por medio del desarrollo de un sistema".

Para la definición de nuestro lenguaje gráfico consideramos necesario manejar los siguientes tipos de datos: booleano, real, punto, línea, polilínea, círculo y polígono. Para el tipo booleano, por ejemplo, las operaciones que dan lugar a un objeto de este tipo son verticalidad de una línea, horizontalidad de una línea, perpendicularidad entre dos líneas y paralelismo entre dos líneas. Las operaciones que acabamos de describir serán: *vertical, horizontal, paralelas* y *perpendiculares*.

Siguiendo las líneas de Goguen [Goguen 78] tenemos las siguientes definiciones:

Definicion 1.-"Un algebra multiclasificada es escencialmente una famila de conjuntos A_1 (llamados portadores del álgebra) con una colección de operaciones (es decir funciones) entre ellos. El conjunto índice S para los portadores es llamado el conjunto tipo".

Así, si el conjunto S es {línea, booleano}, un albegra clasificada-S para este conjunto S tendría dos portadores: $A_{línea}$, $A_{booleano}$, que podría tener entre sus operaciones VERTICAL: $A_{línea} \rightarrow A_{booleano}$ y PARALELAS: $A_{línea} \times A_{línea} \rightarrow A_{booleano}$.

Definicion 2.-Una llave clasificada-S o dominio del operador Σ es una famila $\Sigma_{w,s}$ de conjuntos, en donde $s \in S$ y $w \in S^*$ (S^* es el conjunto de todas las cadenas finitas de S incluyendo la cadena vacía λ}. $F \subset \Sigma_{w,s}$ es un símbolo de operación de categoría w,s, de aridad w y de tipo s.

La aridad w determina el tipo de dominio de la operación, y el tipo s determina el tipo de dato del rango de ésta. Por ejemplo, el operador *perpendicular* pertenece al conjunto $\Sigma_{w,s}$, donde w = línea, línea y s = booleano. En el presente documento se definirá como *clase del operador* a las especificaciones tanto de la aridad como del rango del operador. En nuestro ejemplo la clase del operador *perpendicular* es la cadena línea línea,booleano. Para constantes básicas, la aridad es la cadena vacía λ. Por ejemplo, para la constante *punto* tenemos al conjunto $\Sigma_{\lambda,punto}$ que define al conjunto de puntos que podemos dibujar en la pantalla. La especificación formal del lenguaje consiste en definir el conjunto S de las clases que existen en el lenguaje y las funciones que dan lugar a elementos de dichas clases ($\sum_{w,s}$).

De acuerdo a las definiciones anteriores, un lenguaje así definido no reconoce como válidas funciones en las que sus argumentos no son del tipo adecuado. Sin

embargo, existen casos en que aún cuando los argumentos son del tipo correcto, sus valores no son los apropiados. Por ejemplo, considere la expresión que denota al punto de intersección de la fig. 2.1 a): *interseccion(linea1, linea2)* que es una instancia de la función *interseccion*:linea linea,punto. Si se tralada de posición a la línea l_2 como se muestra en 2.1 b) entonces no hay punto de intersección y el dibujo se encuentra en un estado en que la función no puede producir un objeto del tipo adecuado (un punto).

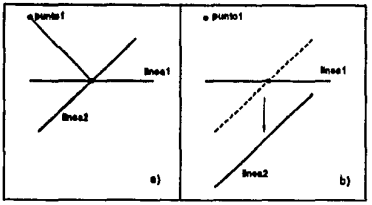


Figura 2.1.- Punto de tipo error. Con el movimiento de la línea l_2 desaparece la intersección.

Una forma de salir de este problema es enviar un mensaje de error y truncar la operación. Alternativamente se pueden inhibir las operaciones de este tipo, no permitiendo operaciones en las que los argumentos no sean del tipo correcto, o que no puedan tomar lugar operaciones en que los argumentos son del tipo correcto pero cuyos valores son inapropiados, manteniendo al usuario informado con un mensaje parecido a: "Error, esta operación no se puede realizar". La solución que aquí se adoptó para manipular estas situaciones consiste en manipular los mensajes error como tipos de datos abstractos. Según Pineda [Pineda 92a],

"Para todo tipo, existe un conjunto correspondiente de elementos error abstractos de este tipo. En la interpretación, tales elementos corresponden a la denotación de expresiones gráficas en estados en los cuales las denotaciones normales no están definidas...Una relación de orden es definida como sigue:
a) Para todo par de tipos s y s_e, $s_e \leq s$"

Lo que significa que los conjuntos de tipo error estan contenidos en los conjuntos de las clases pero no en forma inversa, por ejemplo un elemento línea error está contenido en el conjunto de los elementos de clase línea pero un elemento línea que no es erróneo, no está contenido en el conjunto de los elementos línea erróneos.

Una característica importante del sistema es la propagación del elemento error. En la Figura 2.1 b) podemos observar que el punto de intersección no puede ser dibujado. Como es un elemento punto error, la línea dibujada en 2.1 a) (definida en base a dicho punto) tampoco puede ser visualizada ya que es un elemento línea-error. Este mecanismo permite que en el momento en que se restablezcan las condiciones apropiadas normales (que vuelva a existir intersección en este ejemplo), los objetos pueden ser dibujados nuevamente en la pantalla.

Tomando en cuenta estas consideraciones procedamos a definir el lenguaje G.

S = {booleano, booleano_e, real, real_e, punto, punto_e, línea, línea_e, polilínea, polilínea_e, círculo, círculo_e, polígono, polígono_e}.

La signatura multiclasificada-S para el lenguaje consta de los siguientes conjuntos de constantes:

1.- Definiciones de tipo básico.

1.- $\Sigma_{\lambda,\text{punto}}$ = {*punto1, punto2, punto3,...*}

2.- $\Sigma_{\lambda,\text{real}}$ es un conjunto de numerales

3.- $\Sigma_{\lambda,\text{booleano}}$ = {*falso,verdadero*}

4.- $\Sigma_{\lambda,\text{línea}}$ = {*linea1, linea2, linea3,...*} Un conjunto de líneas

5.- $\Sigma_{\lambda,\text{círculo}}$ = {*circulo1, circulo2, circulo3,...*} Un conjunto de círculos

6.- $\Sigma_{\lambda,\text{polilínea}}$ = {*polilinea1, polilinea2, polilinea3,...*} Un conjunto de polilíneas

7.- $\Sigma_{\lambda,\text{polígono}}$ = {*poligono1, poligono2, poligono3,...*} Un conjunto de polígonos

Los conjuntos de constantes contienen a los elementos con que podemos contar en el sistema, por ejemeplo, el conjunto $\Sigma_{\lambda,\text{punto}}$ = {*punto1, punto2, punto3,...*} corresponde a las constantes de tipo punto que podemos dibujar en la pantalla. Cdo que debemos contar con una forma de construir cada uno de los puntos, líneas, etc, entonces procedemos a definir los que llamaremos constructores básicos. En éstos se conjugan algunas clases y dan lugar a otra diferente, por ejemplo, para formar una línea, se conjugan dos operadores de tipo punto.

2.- Constructores del lenguaje.

1.- $\Sigma_{real\ real,\ punto} = \{punto\}$
El conjunto de elementos de tipo punto definido por dos elementos de tipo real

2.- $\Sigma_{dot\ dot,\ linea} \cup \Sigma_{dot\ real,\ linea} = \{linea\}$
El conjunto de elementos de tipo línea definido por dos puntos y definido por un punto y un real.

3.- $\Sigma_{punto\ real,\ circulo} = \{circulo\}$
El conjunto de elementos de tipo círculo definido por un punto (el centro) y un real (el radio).

4.- $\Sigma_{linea\ linea,\ polilinea} \cup \Sigma_{polilinea\ linea,\ polilinea} = \{polilinea\}$
El conjunto de elementos de tipo polilínea definido por una polilínea y una línea.

5.- $\Sigma_{polilinea,\ poligono} = \{poligono\}$
El conjunto de elementos de tipo polígono definido por una polilínea.

3.- Otros constructores del lenguaje a los que llamamos no básicos son los conjuntos:

1.- $\Sigma_{linea\ linea,\ punto} = \{interseccion,\ union\_t,\ union\_e\}$
El conjunto de los elementos de clase punto definido por la intersección de dos líneas.

2.- $\Sigma_{linea\ circulo,\ punto} = \{i\_n\_circle,\ i\_p\_circle\}$
El conjunto de los elementos de tipo punto definidos por la intersección de una línea y un círculo.

3.- $\Sigma_{punto\ punto,\ punto} = \{puntomed,\ vectsum\}$
El conjunto de los elementos de tipo punto definidos por dos puntos.

4.- $\Sigma_{real\ punto,\ punto} = \{escalar\}$
Escala un vector de posición.

4.- Predicados geométricos.

5.- $\Sigma_{\text{línea, booleano}} = \{vertical, horizontal\}$

6.- $\Sigma_{\text{línea línea, booleano}} = \{paralelas, colineales, perpendiculares\}$

7.- $\Sigma_{\text{línea, real}} = \{angulo\_de, longitud\}$

8.- $\Sigma_{\text{línea línea, real}} = \{angulo\_entre\}$

El conjunto de términos de funciones aritméticas del lenguaje son:

1.- $\Sigma_{\text{real real, real}} = \{sum, rest, mult, div\}$

Estos son todos los operadores que existen en el lenguaje G.

Existe una notación gráfica para mostrar los tipos y sus operaciones. Esta ilustra en forma más clara la especificación de fuentes y destinos en las operaciones o funciones. En esta notación los tipos (es decir, los portadores de los tipos) son indicados por óvalos y las operaciones son indicadas por nodos, las líneas incidentes indican los tipos de los argumentos o entradas para la operación y la salida de la flecha señala el tipo del valor regresado por la operación. Por ejemplo, para las operaciones ***perpendiculares*** y *vertical* tenemos la notación gráfica:

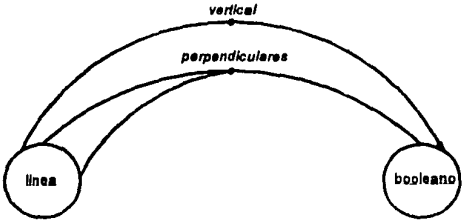


Figura 2.2.- Notación gráfica para las operaciones *perpendiculares* y *vertical*.

Un diagrama más interesante porque muestra la recursividad en la definición del lenguaje es el que envuelve la definición de los objetos de tipo línea y la definición de los objetos de tipo punto (fig. 2.3). Hemos establecido en la definición formal

que las líneas pueden ser constituídas por dos elementos de tipo punto (no olvide que también existen líneas constituídas por un punto y un valor real) y que existen puntos que pueden ser definidos por el cruce de dos líneas. Existen también líneas definidas por puntos y puntos definidos por líneas. la figura 2.3 resalta la característica recursiva en la definición del lenguaje.

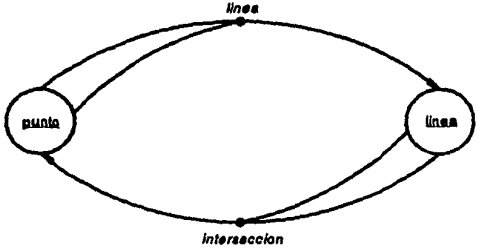


Figura 2.3.- Recursividad en el lenguaje G

2.2.1 Reglas Sintácticas de Formación de Expresiones

Mencinamos cuando introducimos la definición inicial de un lenguaje que un lenguaje es la especificación completa de: a) Un universo de símbolos, b) Reglas de Sintaxis y c) Semántica. El primer punto queda satisfecho al determinar al conjunto **S** con los tipos de datos del lenguaje. El segundo punto, el que toca a la sintaxis, se cumple con las reglas sintácticas de formación, en ellas se especifica cuáles son los términos bien formados en el lenguaje G. Estas reglas son:

1.- Cualquier contante de **S** es un término del lenguaje.

2.- Si $t_1,\ldots,t_n$ son términos de tipo $s_1,\ldots,s_n$, respectivamente, y si F es un símbolo de una función de clase w,s donde $w = s_1,\ldots,s_n$, entonces $F(t_1,\ldots,t_n)$ es una término del lenguaje G.

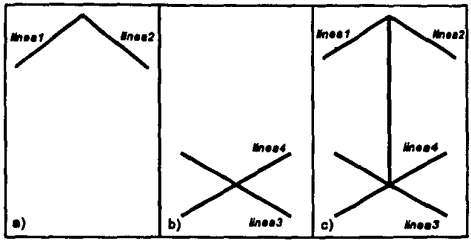


Figura 2.4.- a) Unión de dos líneas. b) Intersección de dos líneas. c) Línea obtenida por dos construtores del lenguaje.

A partir de estas reglas podemos escribir con nuestro lenguaje diversas expresiones que se refieren a figuras geométricas y relaciones entre ellas. Por ejemplo, si en la segunda regla sustituimos $t_1,...,t_n$ por *punto1, punto2* que son de clase punto, entonces F tiene que ser de aridad punto punto, por lo que *linea(punto1, punto2)* es una expresión bien formada en G. En otras palabras, sólo se tiene que comprobar que la aridad y el tipo de una expresión sean del tipo de dato adecuado al símbolo operador.

La figura 2.4.a) muestra el punto en el que se intersectan dos líneas se intersectan en uno de sus extremos. La expresión que se refiere a dicho punto corresponde en expresiones de lenguaje G a:

punto1 = union_e(linea1, linea2)

En la figura 2.4.b) se presenta una intersección franca entre dos líneas. La expresión que se refiere al punto de intersección es:

punto2 = interseccion(linea3, linea4)

En la figura 2.4.c) se muestra la línea representada por

linea1 = linea(unión_e(linea1, linea2), intersección(linea3, linea4))

Note que esta línea está constituida por dos puntos, uno obtenido a partir de la intersección de dos líneas y el otro es obtenido a partir de la unión extrema de dos líneas distintas a las anteriores.

2.2.2 Proceso de Intrpretación de Expresiones.

El proceso de interpretación es un proceso ligado al lenguaje de representación. Este proceso es el encargado de traducir lo que las estructuras represtacionales del *mundo de representación* significan en el *mundo representado.* Con lo especificado hasta ahora tenemos lo necesario para escribir expresiones gramaticalmente correctas y aún cuando se ha dado una idea del significado de las expresiones en los ejemplos expuestos, es necesario especificar el procedimiento que nos permite establecer una relación entre una expresión en G y su represen-tación en la pantalla. Es necesario especificar el procedimiento de interpretación. La interpretación del lenguaje es hecha por un programa cuya entrada es una expresión de G y cuya salida es el resultado de calcular su valor en términos del dibujo en la pantalla en un momento determinado. En la sesión interactiva, el dibujo está desplegado en la pantalla pero en el sistema está en una base de conocimientos En ésta se van guardando las expresiones que corresponden a las figuras y restricciones que el usuario aplica.

Una descripción básica es un operador en el que todos sus argumentos son constantes básicas del lenguaje. De hecho, el resultado del proceso de interpretación es la descripción básica del objeto gráfico a ser dibujado y no puede ser sustituida por una descripción más simple. El objeto gráfico al que corresponde una descripción básica puede ser dibujado directamente en la pantalla. Pero si los parámetros de un operador no constituyen una expresión básica, entonces el proceso de interpretación se encarga de evaluar estos términos y producir su descripción básica de acuerdo al estado en que se encuentre la base de conoci-mientos. Por ejemplo; suponga que la base de conocimientos consiste de:

linea1 = linea(punto1, punto2)
linea2 = linea(punto3,punto4)
circulo = circulo(interseccion(linea1,linea2),real(3))

El lado derecho de las primeras igualdades son descripciones básicas dado que sus argumentos son constantes, no así el lado derecho de la tercera expresión, ya que los argumentos de *"circulo"* corresponden a otra función (intersección) y a un real. Sin embargo, podemos reducir el valor de c_1 en relación a la base de conocimientos de la siguiente forma:

circulo(interseccion(linea1,linea2),real(3))
circulo(interseccion(linea(punto1,punto2),linea(punto3,punto4)),real(3))
circulo(punto(x,y),real(3))

donde las coordenadas de *punto(x,y)* se calculan por geometría.

Siguiendo las líneas de Pineda [Pineda XX] tenemos que la interpretación de una expresión ϕ del lenguaje G se hace de acuerdo con las siguientes reglas:

(1) Si ϕ es una descripción básica, entonces el valor de ϕ es el mismo ϕ.

(2) Si ϕ es una descripción no básica, entonces reducir los parámetros de la descripción y devolver su descripción básica

(3) Si ϕ es culaquier otra expresión, entonces reducir los parámetros de esta expresión y devolver el valor de la expresión.

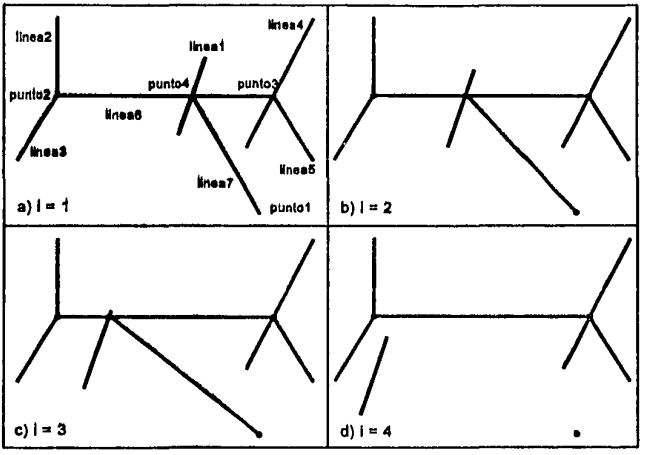


Figura 2.5.- Diferentes estados de una sesión interactiva en G.

El tercer punto se refiere a expresiones como la que describen a la figura 2.5.a), que es:

linea(interseccion(linea1,linea(union_e(linea2,linea3),union_t(linea4,linea5))),punto1)

cuya interpretación es:

linea(interseccion(linea1,linea(punto2,punto3)),punto(X:Y)) por la regla (3),

linea(intersección (linea(punto(X1:Y1),punto(X2:Y2))
linea(punto(X3:Y3), punto(X4:Y4))),punto(X:Y)). por la regla (3).

linea(punto(X5:Y6),punto(X:Y)) por la regla (2).

linea(punto(X5:Y6),punto(X:Y)) por la regla (1).

2.2.2.1 Interpretación Parcial de Expresiones.

Observe que el resultado de la interpretación dado en el ejemplo anterior sólo puede obtenerse cuando la base de datos se encuentra en un estado en que pueden dibujarse los objetos descritos. La posición en la patalla de los objetos descritos por constructores no básicos depende únicamente de su propia descripción. Así, si en la sesión interactiva se desplaza *linea1* veremos que va cambiando de inclinación y tamaño *linea7* porque un extremo de ella no está definido por un punto fijo en la pantalla, sino que depende de la intersección de *linea1* y *linea6*. En la figura 2.5 representamos la secuencia del movimiento de *línea7*. La figura d) se encuentra en un estado en el que *linea7* ya no puede ser dibujada, pero el movimiento de *linea1* es posible gracias a los tipos de datos erróneos abstractos. La interpretación del dibujo en este estado es:

linea(interseccion(linea1,linea(union_e(linea2,linea3),union_t(linea4,linea5))),punto1)

linea(interseccion(linea1,linea(punto2,punto3)),punto(X:Y))

linea(intersección (linea(punto(X1:Y1),punto(X2:Y2))
linea(punto(X3:Y3), punto(X4:Y4))),punto(X:Y)).

linea(punto_e,punto(X:Y))

linea_e

El resultado es un elemento error de las líneas y aunque no puede ser dibujada, es una línea debido a la relación linea_e ≤ línea. A las interpretaciones de este tipo las denominamos interpretaciones parciales. A los elementos tipo error los llamaremos simplemente elementos error, y a los elementos que pueden ser dibujados los llamaremos elementos normales. Las reglas para manipular expresiones parciales de

expresiones de G son:

4.- Si todos los argumentos del operador de una expresión son elementos normales del tipo correspondiente, entonces el valor de la expresión es un elemento normal de su tipo.

5.- Si todos los argumentos del operador de una expresión son elementos normales de sus tipos correspondientes pero el valor de la expresión es un elemento error del tipo correspondiente, se coloca el error en una pila de errores para una inspección posterior y la producción de un mensaje de error.

6.- Si un argumento de la expresión es un elemento error de su clase errónea correspondiente, entonces el valor de la expresión es un elemento error del tipo correspondiente a la expresión.

2.2.3 Proceso de Interpretación Semántica.

El programa interprete funciona aplicando las 6 reglas dadas en la sección anterior para producir una expresión que pueda ser dibujada directamente en la pantalla, o para producir un elemento error. Teóricamente definimos a la interpretación semántica de un lenguaje en relación a un modelo *M*. Según Pineda [Pineda y Lee 19], un modelo *M* para G es una n-ada ordenada (G,I,F), en donde $G = \langle G_s \rangle_{s \in S}$ es una familia *S* ordenada alfabéticamente de conjuntos no vacíos (es decir, los conjuntos de símbolos gráficos de diferentes tipos pertenecen a esta familia), *I* es un conjunto de estados $I_1,\ldots,I_n$ (que corresponden a los estados de la sesión interactiva) y *F* que es una función de interpretación cuyo dominio es un conjunto de constantes de G (es decir, los números y las constantes básicas) y cuyo rango está en *G* según las cuatro reglas de abajo.

Para cada tipo definimos a una función *g* como una función que tiene como dominio al conjunto de todas las variables del tipo y como rango a algún elemento de G_s para toda varible de clase *s*. Se tomará como convención notacional que $[[\alpha]]^{M,i,g}$ es, el valor semántico de una expresión α con respecto a un modelo *M*, en un estado $i \in I$, y un valor de asignación g.

El rango de la función de interpretación F para las expresiones de G es:

1 Si α es una constante o una descripción básica de clase *s*, entonces:

$$[[\alpha]]^{M,i,g} = [[F(\alpha)]](i),$$

en donde $[[F(\alpha)]](i)$ es el valor de $[[F(\alpha)]]$ en el estado *i*.

2 si *f* es el símbolo de una función de categoría w,s , donde $w = s_1,\ldots,s_n$, entonces $[[f\ ]]^{M,g}$ es una función con dominio en $G_{s_1} x \ldots x\ G_{s_n}$ y rango en G_s. Para una operación geométrica, esta función es la función calculada por un algoritmo geométrico asociado a dicha operación. Para los constructores básicos de las clases gráficas, esta función verifica que la geometría de un símbolo está definida apropiadamente.

3 Si ϕ es un término de la fomra $f(t_1,\ldots,t_n)$, donde *f* es un operador de clase w,s en que $w = s_1,\ldots,s_n$ y $t_1,\ldots,t_n$ son expresiones de clases $s_1,\ldots,s_n$, entonces:

$$[[\ f(t_1,\ldots,t_n)\ ]]^{M,i\,g} = [[\ f\ ]]^{M,g}(\ [[t_1\ ]]^{M,i,g},\ldots,[[t_n\ ]]^{M,i,g}\ ]]\ ).$$

4 Si ϕ y ψ son expresiones de clase *s*, entonces $[[\phi=\psi]]^{M,i,g}$ es verdadero si y solo si $[[\phi]]^{M,i,g}$ es igual que $[[\psi]]^{M,i,g}$.

Observe que en la segunda regla la interpretación de la función *f* no depende del estado *i* porque los símbolos de operaciónes geométricas y lógicas son los mismos en cualquier estado.

Con estas cuatro reglas semánticas se expone la interpretación de expresions de G. En ellas está contenida la especificación procedural hecha anteriomente. La regla (1) establece que el referente (el significado) del nombre de un símbolo gráfico o de su descripción, es el mismo símbolo gráfico. La segunda regla concierne a la interpretación de los nombres de los operadores geométricos. La tercera regla establece que la interpretación de los argumentos de una función, es la aplicación de la función denotada por el símbolo operador de la expresión sobre los objetos denotados por los argumentos. La regla cuatro establece el mecanismo por el cual, cuando se requiere la interpretación del nombre de un símbolo, se evalúa la descripción asociada a dicho nombre de símbolo; así, si evaluamos *linea2,* la interpretación entregará *linea(punto1, punto2).*

Las reglas semánticas también abarcan la interpretación de expresiones de G aún cuando los elementos no tienen representación en la pantalla. Debido a la adición de los elementos error en las clases respectivas, podemos pensar que todas las funciones son totales. Entonces con estas reglas, la interpretación de una expresión es un elemento normal o un elemento error de su clase en cualquier

situación.

A manera de ejemplo, veamos cómo se aplican las reglas de interpretación sobre la siguiente expresión.

linea(union_t(linea1,linea2), punto1)

1.- Aplicación de la regla 3 a toda la expresión,

$$[[\textit{linea}\ (\textit{union\_t}\ (\textit{linea1},\ \textit{linea2}\ ),\ \textit{punto1}\ )\ ]]^{M,i,g} = [[\textit{linea1}\ ]]^{M,g}\ (\ [[\textit{union\_t}(\textit{linea1},\ \textit{linea2})\ ]]^{M,i,g},\ \ [[\textit{punto1}]]^{M,i,g}\ )$$

2.- Aplicación de la regla 1 sobre *union_t.*

$$[[\ \textit{union\_t}\ ]]^{M,i,g} = [[\textit{union\_t}\ ]]^{M,g}\ (\ [[\textit{linea1}\ ]]^{M,i,g},\ [[\textit{linea2}]]^{M,i,g}\ )$$

2.1.- Aplicación de la regla 4, sustitución de la constante *linea1* por su descripción básica

$$[[\textit{linea1}\ ]]^{M,i,g} = [[\ \textit{linea(punto2,\ punto3}\ ]]^{M,i,g}$$

2.2.- Aplicación de la regla 3 sobre *linea,*

$$[[\textit{linea(punto2,punto3)}]]^{M,i,g} = [[\ \textit{linea}\ ]]^{M,g}\ (\ [[\textit{punto2}]]^{M,i,g},\ [[\textit{punto3}]]^{M,i,g})$$

como los puntos son constantes básicas[7], *linea(punto2, punto3)* es una expresión bien definida. Se repiten 2.1 y 2.2 para *linea2.*

2.3.- Aplicación de la regla 3. Se resuelve mediante un algoritmo geométrico para encontrar el punto de unión entre ambas líneas,

$$[[\textit{union\_t}\ ]]^{M,g}\ ([[\textit{linea(punto2,punto3)}]]^{M,i,g},\ [[\textit{linea(punto4,punto5)}]]^{M,i,g}) = [[\ p_{union\_t}]]^{M,i,g}$$

en donde $[[\ p_{union\_t}]]^{M,i,g}$ es el punto que resulta del algoritmo geométrico, es decir, es el significado de [[*union_t*]].

[7]Considerados así para mayor claridad en este ejemplo, aunque en la gramática de G los puntos tienen descripciones básicas en términos de dos reales.

3.- Aplicación de la regla 1; evaluación de *punto1*.

[[*punto1*]]^{M,i,g} esta dado por la función de interpretación de el modelo *M*.

4.- Aplicación de la regla 3. Se sustituye el resultado de 2.3) en 1) y se aplica el algoritmo geométrico asociado a línea que verifica que los dos puntos sean expresiones bien definidas.

[[*linea*]]^{M,g} ([[*punto1*]]^{M,i,g}, [[p_{union_t}]]^{M,i,g}),

que es una descripción básica del lenguaje.

Una vez definido teórica y proceduralmente el proceso de interpretación semántica, queda completada la especificación del lenguaje, ahora es necesario explicar en qué consisten los algoritmos geométricos para la reducción de los constructores en el proceso de interpretación.

2.3 Reducción de Expresiones en el Proceso de Interpretación.

Como se ha dicho anteriormente, la interpretación de las expresiones del lenguaje G ha sido implementada en un programa intérprete, éste sigue las reglas descritas en la sección anterior. Durante esta sección se explicará en que consisten los algoritmos de reducción de las expresiones en descripciones básicas. Ejemplos de expresiones que requieren una reducción hecha por un algoritmo geométrico son:

interseccion(linea1, linea2)
y *circulo(union_t(linea1, linea2), real(3))*

que de acuerdo a su categoría, son reducidos a:

punto1
y *circulo(p2, real(3))*,

respectivamente. Como puede observar, los algoritmos geométricos son llamados cuando la expresión no consiste de un constructor básico con descripción básica; es decir, todos los constructores no basicos como *intersección, angulo_de,* etc, requieren de un algoritmo de solución geométrico que entregue el valor de la clase que corresponda a cada constructor (como un elemento del tipo punto para *interseccion,* un elemento de tipo real para *angulo_de,* etc.).

En primer lugar se explican los algoritmos para encontrar el punto de intersección entre dos líneas, el punto de de unión entre extremos de dos líneas *(union_e)* y el punto de unión entre el extremo de una línea y un punto cualquiera a lo largo de otra línea *(union_t)*. Todas estas funciones han sido derivadas del mismo rezonamiento y provienen de la solución del problema que se muestra a continuación.

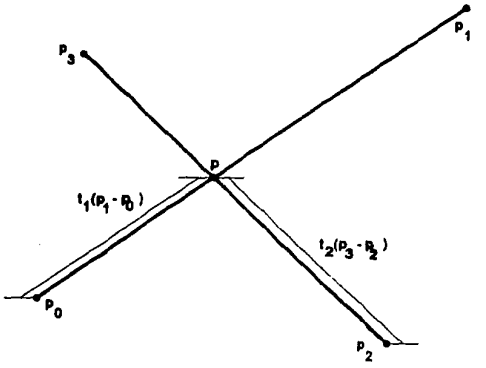


Figura 2.6.- Obtencion del punto de intersección por ecuaciones paramétricas.

Considere la figura 2.6, En ella se han dibujado a los vectores de posición como puntos simplemente. El punto en donde se intersectan las dos rectas dibujadas ha sido nombrado p. La recta dibujada de p_0 a p_1 corresponde a la resta de los vectores de posición p_1-p_0. Por lo tanto, el vector de posición p es la suma del vector de posición p_0 más un vector proporcional al vector p_1-p_0 sobre su misma línea de acción, es decir, p_0 más el vector p_1-p_0 multiplicado por un escalar t. Aplicando el mismo razonamiento para el vector p_3 - p_2 tenemos que el punto de intersección es:

2.1) $p = p_0 + t_1 (p_1 - p_0)$

2.2) $p = p_2 + t_2 (p_3 - p_2)$

en donde si $t_1 = 0$, entonces $p = p_0$, si $t_1 = 1$, entonces $p = p_1$.
Si $0 \leq t_1 \leq 1$ y $0 \leq t_2 \leq 1$, entonces p se encuentra sobre las líneas.

Si $t_1 \geq 1$ o $t_1 \leq 0$, entonces p se encuentra en la proyección del vector $p_1 - p_0$.
Si $t_2 \geq 1$ o $t_2 \leq 0$, entonces p se encuentra en la proyección del vector $p_3 - p_2$.

La solución de las ecuaciones 2.1) y 2.2) es como sigue. Escribiendo 1) y 2) en términos de las coordenadas (x,y) de los puntos, tenemos que

2.3) $(x,y) = (x_0,y_0) + t_1(x_1,y_1 - x_0,y_0)$

2.4) $(x,y) = (x_2,y_2) + t_2(x_3,y_3 - x_2,y_2)$

de donde se desprende que

2.5) $x = x_0 + t_1(x_1 - x_0)$, $y = y_0 + t_1(y_1 - y_0)$

2.6) $x = x_2 + t_2(x_3 - x_2)$, $y = y_2 + t_2(y_3 - y_2)$

haciendo x de 2.5) igual a x de 2.6) y y de 2.5) igual a y de 2.6), obtenemos

2.7) $x_0 + t_1(x_1 - x_0) = x_2 + t_2(x_3 - x_2)$

2.8) $y_0 + t_1(y_1 - y_0) = y_2 + t_2(y_3 - y_2)$

escribiendo los términos independientes de t_1 y t_2 del lado izquierdo de la ecuación:

2.9) $(x_2 - x_0) = t_1(x_1 - x_0) - t_2(x_3 - x_2)$

2.10) $(y_2 - y_0) = t_1(y_1 - y_0) - t_2(y_3 - y_2)$

escribiendo las restas en forma de diferencias (Δ), tenemos que

2.11) $\Delta x_{20} = t_1\Delta x_{10} - t_2\Delta x_{32}$

2.12) $\Delta y_{20} = t_1\Delta y_{10} - t_2\Delta y_{32}$

que reescrito en forma matricial es

$$2.13) \quad \begin{bmatrix} \Delta x_{20} \\ \Delta y_{20} \end{bmatrix} = \begin{bmatrix} \Delta x_{10} & -\Delta x_{32} \\ \Delta y_{10} & -\Delta y_{32} \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$

para fines de notación tenemos que

$$2.14) \quad \Delta = \begin{bmatrix} \Delta x_{10} & -\Delta x_{32} \\ \Delta y_{10} & -\Delta y_{32} \end{bmatrix}$$

resolviendo tenemos que

$$2.15) \quad t_1 = \frac{\begin{vmatrix} \Delta x_{20} & -\Delta x_{32} \\ \Delta y_{20} & -\Delta y_{32} \end{vmatrix}}{|\Delta|} = \frac{|\Delta t_1|}{|\Delta|}$$

$$2.16) \quad t_2 = \frac{\begin{vmatrix} \Delta x_{10} & \Delta x_{20} \\ \Delta y_{10} & \Delta y_{20} \end{vmatrix}}{|\Delta|} = \frac{|\Delta t_2|}{|\Delta|}$$

De esta manera, el algoritmo geométrico encuentra los parámetros t_1 y t_2, después los sustituye en 2.3) y 2.4) para encontrar el punto de intersección para la función *interseccion* verificando que $0 < t_1 < 1$ y $0 < t_2 < 1$, para considerar sólo las intersecciones francas y no las que incluyen a los extremos.

El mismo procedimiento se usa para encontrar la unión en T de dos líneas (*union_t*) pero verificando que se cumpla una de las siguientes condiciones:

$$0 < t_1 < 1, \; t_2 = 0;$$
$$0 < t_1 < 1, \; t_2 = 1;$$
$$t_1 = 0, \; 0 < t_2 < 1;$$
$$t_1 = 1, \; 0 < t_2 < 1.$$

de otra manera no existe intersección en T.

La unión en los extremos de dos líneas es denotada aquí como *union_e.* El algoritmo geométrico que encuentra un puntos así trabaja del mismo modo que los dos anteriores pero tiene que complir un par de las siguientes condiciones:

$$t_1 = 0,\ t_2 = 0;$$
$$t_1 = 0,\ t_2 = 1;$$
$$t_1 = 1,\ t_2 = 0;$$
$$t_1 = 1,\ t_2 = 1.$$

Veamos ahora que pasa si Δ (de la ecuación 2.14) es igual a cero. De 2.14) tenemos que

2.17) $\Delta = -\Delta x_{10}\Delta y_{32} + \Delta x_{32}\Delta y_{10}$

Si consideramos que $\Delta = 0$, entonces

2.18) $\Delta x_{10}\Delta y_{32} = \Delta x_{32}\Delta y_{10}$

es decir,

2.19) $$\frac{\Delta y_{32}}{\Delta x_{32}} = \frac{\Delta y_{10}}{\Delta x_{10}}$$

en donde podemos apreciar que las pendientes de ambas líneas son iguales. Entonces el algoritmo que verifica que dos líneas sean paralelas tiene que comprobar que $\Delta = 0$ y que $\Delta t_1 = 0$ y $\Delta t_2 = 0$.

Si $\Delta t_1 = 0$, entonces

2.20) $-\Delta x_{20}\Delta y_{32} + \Delta x_{32}\Delta y_{20} = 0$

2.21) $$\frac{\Delta y_{32}}{\Delta x_{32}} = \frac{\Delta y_{20}}{\Delta x_{20}}$$

2.22) $$\frac{y_3 - y_2}{x_3 - x_2} = \frac{y_2 - y_0}{x_2 - x_0}$$

y si $\Delta t_2 = 0$ tenemos que

2.23) $$-\Delta x_{10}\Delta y_{20} + \Delta x_{20}\Delta y_{10} = 0$$

2.24) $$\frac{\Delta y_{20}}{\Delta x_{20}} = \frac{\Delta y_{10}}{\Delta x_{10}}$$

2.25) $$\frac{y_2 - y_0}{x_2 - x_0} = \frac{y_1 - y_0}{x_1 - x_0}$$

de 2.22) y 2.25) podemos escribir que

2.26) $$\frac{y_3 - y_2}{x_3 - x_2} = \frac{y_1 - y_0}{x_1 - x_0} = \frac{y_2 - y_0}{x_2 - x_0}$$

lo que indica que además de que las líneas son paralelas, comparten puntos, es decir, que forman una misma línea. Esta es la comprobación que hace el algoritmo geométrico para la función de colinealidad (*colineales*)

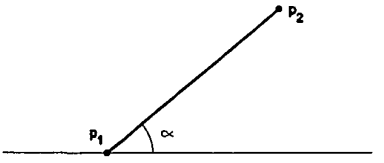


Figura 2.7.- Angulo de una línea.

La función *angulo_de* sirve para determinar el ángulo de una línea. El ángulo ∝ de una línea definida por dos puntos p_1 (x_1,y_1) y $p_2(x_2,y_2)$ (véase fig. 2.6) está determinado por

$$2.27) \quad \alpha = \arctan\frac{y_2 - y_1}{x_2 - x_1}$$

si la línea se encuentra en el primer cuadrante, tomando al punto p_1 como el origen;

$$2.28) \quad \alpha = \pi + \arctan\frac{y_2 - y_1}{x_2 - x_1}$$

si la línea se encuentra en el segundo cuadrante, tomando al punto p_1 como el origen;

$$2.29) \quad \alpha = \pi + \arctan\frac{y_2 - y_1}{x_2 - x_1}$$

si la línea se encuentra en el tercer cuadrante, tomando al punto p_1 como el origen;

$$2.30) \quad \alpha = 2(\pi) + \arctan\frac{y_2 - y_1}{x_2 - x_1}$$

si la línea se encuentra en el cuarto cuadrante, tomando al punto p_1 como el origen.

Para encontrar el ángulo de intersección entre dos líneas α (véase figura inferior), que corresponde a la función *angulo_entre* se determina el ángulo de cada una de las líneas y se calcula la diferencia de ellos.

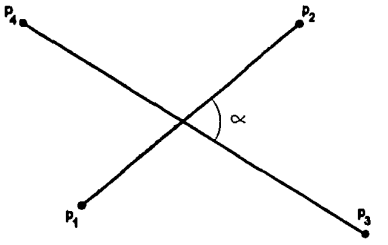


Figura 2.8.- Angulo entre dos líneas

El algritmo geométrico que dice si dos líneas son perpendiculares y que corresponde a la función *perpendiculares* verifica que el ángulo entre las dos líneas sea una ángulo recto, si el ángulo entre las dos líneas es ya sea de 90°, de 270°, de -90° o de -270°, entonces la función entregará el valor booleano de verdadero, de otro modo es falso.

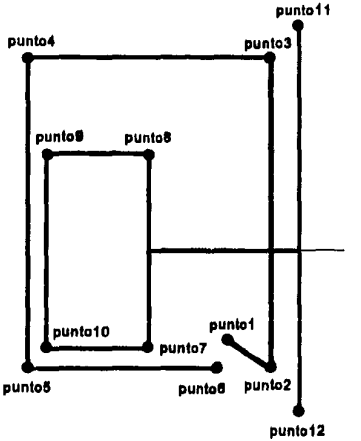
La función *vertical* verifica que una línea sea vertical, El algoritmo geométrico calcula el ángulo de la línea. Si el ángulo es igual a 90°, -90°, 270° o -270°, entonces la función entrega el valor booleano de verdad, de otro modo es falso.

La función *horizontal* comprueba horizontalidad en una línea, Si el ángulo de la línea que se incluye como argumento de la función es diferente a 0° o a 180°, entonces se regresa el valor de falso.

De esta forma es como quedan definidos los algoritmos que calculan los valores de los constructores no básicos y selectores del lenguaje. En general, el resultado del proceso de interpretación son expresiones que nos indican si el valor puede ser dibujado en la pantalla (si la expresión a evaluar tiene un referente que es un elemento normal) o si no puede ser dibujado (es una interpretación parcial y da como resultado un elemento error del tipo al que corresponda la expresión).

Con la definición del lenguaje G que permite la referencia no sólo de objetos gráficos, sino también de relaciones entre estos objetos y con las reglas de interpretación de dicho lenguaje se desarrolló un programa intérprete, éste recibe como entrada una expresión en el lenguaje G y entrega como resultado la interpretación de la misma. la interpretación de los objetos se hace apoyándose en una base de conocimientos. La base de conocimientos consiste de los objetos que se

pueden encontrar en un momento dado en la pantalla de el modelador gráfico. Los objetos se encuentran descritos en la base de conocimientos en un formato sencillo diseñado para este fin. Los objetos de la base de datos van dentro del predicado define(*nombre, descripción*); en donde *nombre* es el nombre de un objeto gráfico válido en G, por ejemplo, *punto1* o *linea3. descripción* es la descripción respectiva del objeto referido por *nombre*, por ejemplo, *punto(3:3) o linea(punto1, punto2)*. La expresión define(*línea1, linea(punto1, punto2)*) se lee: La línea1 define a la línea que va del punto1 al punto2.




```
define(linea1, linea(punto1, punto2)).
define(linea2, linea(punto2, punto3)).
define(linea3, linea(punto3, punto4)).
define(linea4, linea(punto4, punto5)).
define(linea5, linea(punto5, punto6))
define(polilinea1, polilinea(linea1, linea2)).
define(polilinea2, polilinea(polilinea1, linea3)).
define(polilinea3, polilinea(polilinea2, linea4)).
define(polilinea4, polilinea(polilinea3, linea5)).
define(linea6, linea(punto7, punto8)).
define(linea7, linea(punto8, punto9)).
define(linea8, linea(punto9, punto10)).
define(linea9, linea(punto10, punto7)).
define(polilinea5, polilinea(linea6, linea7)).
define(polilinea6, polilinea(polilinea5, linea8)).
define(polilinea7, polilinea(polilinea6, linea9)).
define(poligono1, poligono(polilinea7)).
define(linea10, linea(punto11, punto12)).
define(linea11, linea(puntomed(punto7, punto8),
      angle(0))).
define(linea12, linea(puntomed(punto7, punto8),
      intersect(linea11, linea10))).
```

Figura 2.9.- Descripción de la tina de baño con expresiones del lenguaje G. Debido al espacio en la ilustración solo se etiquetan los objetos de tipo punto, el lector podrá deducir los demás objetos en base a los puntos etiquetados.

Observe que el lenguaje G proporciona los elementos necesarios para referir a los objetos graficos del problema de la tina de baño, con el que iniciamos la exposición de este trabajo. También permite incluir las relaciones entre los objetos. Para describir a las paredes del baño podemos usar un objeto de tipo polilinea, para describir a la tina de baño es conveniente usar un objeto de tipo poligono, las tuberías pueden ser descritas por un objeto de tipo linea, y restingir una de ellas de la forma que se muestra en la figura 2.8. Observe en ésta figura que la tubería que conecta a la tina (*poligono1*), con la tubería general (*linea10*) es hecha con ayuda de una línea de

construcción o de trazo (*linea11*), de manera que la tubería que hace a la tina funcional (*linea12*) va del punto medio de linea6 a el punto donde se intersectan la línea de trazo y la tubería general. En el siguiente capítulo se muestra como expresividad del lenguaje y el nivel de estructura que maneja contribuye al proceso de satisfacción de restricciones.

"...Pero a los bárbaros se les caían de las botas, de las barbas, de los yelmos, de las herraduras, como piedrecitas, las palabras luminosas que se quedaron aquí resplandecientes...el idioma. Salimos perdiendo,... Salimos ganando...Se llevaron el oro y nos dejaron el oro...Se llevaron todo y nos dejaron todo...Nos dejaron las palabras."

Pablo Neruda.

Capítulo 3

Abstracción y Referencia en la Satisfacción de Restricciones.

3.1 Introducción.

El objetivo de este capítulo es mostrar de qué manera el nivel de abstracción del lenguaje y la expresividad del mismo influyen en la solución de problemas de satisfacción de restricciones. A lo largo del capítulo también diferenciamos dos tipos de soluciones para el problema que atacamos y explicamos a cuál de éstos corresponden las que realiza el prototipo implementado.

El lenguaje G y el intérprete del mismo han servido como cimientos para la implemetación del sistema prototipo S_2RS, cuyo nombre se deriva de la función que

el prototipo realiza: Satisfacción de Restricciones por Referencia Simbólica. El S_2RS cuenta con una interface gráfica mediante la cual podemos ver las figuras geométricas correspondientes a las expresiones del lenguaje G que recibe y/o entrega el programa intérprete. La definición del lenguaje (presentada en el capítulo anterior) permite hacer referencia a los objetos gráficos de nuestro dominio de interés. Observe que el lenguaje pudiera haber sido definido para manejar un nivel menor de estructura en los objetos geométricos. Por ejemplo, suponga que definimos un lenguaje F para manejar objetos gráficos cuyos objetos de mayor nivel de estructura son las líneas. Con un lenguaje tal podemos hacer figuras complejas en base a los objetos básicos que maneja. De hecho podemos hacer figuras geométricas de la misma apariencia que las que se pueden hacer con el lenguaje G. Con ambos podemos crear descripciones que se refieren a un rectángulo. La figura 3.1 a) es a simple vista un único objeto, pero para el lenguaje F son varios objetos (líneas) independientes entre sí (fig. 3.1 a). Consideremos que con el lenguaje G es posible definir un polígono y como tal nos podemos referir a él. Consideremos a las representaciones con situaciones de cambio en las que se requiere trasladar figuras similares a una nueva posición. Si nosotros aplicamos una traslación sobre un objeto como el de la figura 3.1 b), que ha sido creado con expresiones del lenguaje G, el sistema responde buscando en la base de conocimientos a la figura (polígono) y cambia de posición los puntos que conforman el polígono. Si esta figura se hiciera con el lenguaje F (véase descripción de la fig. 3.1 a), tendría que ser con figuras independientes (líneas) que dan la impresión de ser un único objeto pero que para desplazarle de lugar tendríamos que recurrir a trasladar línea por línea. Podríamos pensar que para solucionar el desplazamiento de este objeto descrito por el Lenguaje F, sería necesario condicionar cada par de líneas a estar unidas en uno de sus extremos y a mantener la misma longitud de cada una. De modo que cuando hiciéramos una modificación de traslación, sería necesario recurrir a un algoritmo de satisfacción para satisfacer dichas restricciones.

Observe que para hacer dicha traslación, se haría la modificación sobre una de las líneas y el algoritmo de satisfacción de restricciones se encargaría de hacer el resto de la traslación. Sin embargo con el lenguaje G esta figura puede hacerse como un polígono y manipularlo como el objeto a simple vista representa: un polígono (sin ninguna restricción adicional).

De acuerdo a lo anterior, podemos observar que el aumento en el nivel de abstracción del lenguaje contribuye al proceso de satisfacción de restricciones. Según L.A. Pineda[8], el proceso de satisfaccion de restricciones se desplaza en tres dimensiones, una de las cuales es la agragación estructural de los objetos representados en el

[8]Comentario de Luis A Pineda en el Instituto de Investigaciones Eléctricas, Otoño 1994.

lenguaje a los que se hace referencia en el acto deíctico. Otra es la expresividad del lenguaje y la última es el proceso de razonamiento sintético (que explicaremos en breve). Es decir, la solución de una restricción está formada por una de estas componentes o por una combinación de ellas. En lo subsecuente trataremos las características y comportamiento de las componentes del proceso de satisfacción de restricciones que denominaremos de acuerdo con Pineda Referencia y Síntesis [Pineda 92].

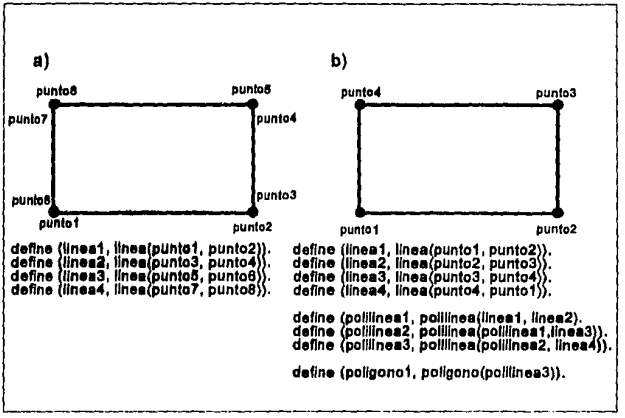


Figura 3.1.- Figura que permite la comparación entre dos descripciones del mismo objeto (a simple vista). La figura a) muestra el objeto y su descripción en un lenguaje de menor nivel de abstracción que el lenguaje G. En la figura b) se muestra la descripción del objeto en lenguaje G.

3.2 La Referencia en el Proceso de Satisfacción de Restricciones.

3.2.1. Traslaciones de objetos en el sistema S_2RS.

Consideremos ahora como se hacen los desplazamientos de los objetos en S_2RS. Comencemos por la estructura geométrica fundamental: el punto. Suponga que tenemos el dibujo de la figura 3.2 en que también podemos ver las expresiones que

lo describen y que están almacenados en la base de conocimientos del sistema.

Si en la sesión interactiva el usuario decide desplazar el punto *punto1*, selecciona el punto con el ratón, hace una operacion de arrastre (drag) y lo libera en una nueva posición. Durante esta operación, el sistema obtiene las coordenadas de la pantalla en donde se seleccionó con el ratón y recorre la base de conocimientos buscando a cual de los puntos corresponden dichas coordenadas. Para este ejemplo corresponden al *punto5*. Cuando el ratón es liberado, el sistema simplemente sustituye las coordenadas de la antigua posición por las coordenadas de la nueva. La figura 3.2 a) se convierte en la figura 3.3 a) como consecuencia de esta operación.

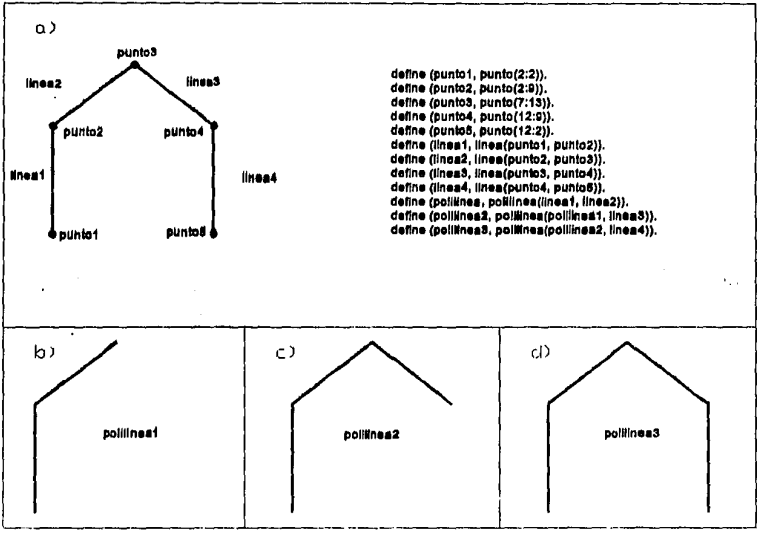


Figura 3.2.-La sección a) muestra una polilínea etiquetada con los nombres de los objetos correspondientes a la descripción. Las secciones b), c) y d) muestran las polilíneas descritas.

Veamos cómo se realiza el movimiento de un objeto de tipo línea. Suponga que queremos trasladar la línea línea4 tomando como figura inicial la figura 3.2 a). Se hace

la selección con ayuda del ratón en algún punto sobre *línea4* para luego ejecutar una operación de arrastre. El sistema toma las coordenadas en donde fué hecha la selección y busca en la base de conocimientos si el punto al que corresponden estas coordenadas se encuentra sobre una de las líneas contenidas en la base de conocimientos. Para este caso encuentra que corresponde a *línea4*. Una vez detectada la línea que se ha escogido, se obtiene el vector diferencia calculado entre la posición donde se hizo la elección y la posición donde terminó la operación de arrastre. Este vector diferencia se suma a los puntos que definen la línea escogida, en este caso sobre los puntos *punto4* y *punto5*. Por ejemplo, si nosotros hacemos la selección de línea apuntando a la coordenada (12:7) y terminamos de hacer el arrastre hasta la posición (16:7), entonces el vector diferencia es (16-12,7-7), esto es (4,0), que es sumado a las coordenadas de los puntos *punto4* y *punto5*. Por lo tanto la definicion del punto *punto4* despues de la traslación queda de la forma *define(punto4,punto(16:9))* y el punto *punto5* queda de la forma *define(punto5, punto(16:2))* con lo cual la figura 3.2 a) se convierte en la figura 3.3 b).

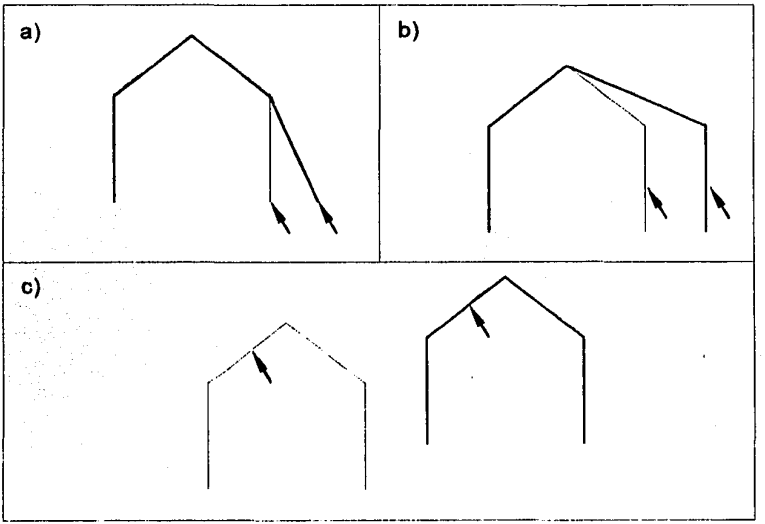


Figura 3.3.-La sección a) muestra la traslación de un punto de la polilínea, la sección b) muestra la traslación de un línea y la sección c) muestra la traslación de la polilínea.

Observe que esta modificación provoca que se afecte también *línea3.* Este resultado es deseable dado que se conserva la propiedad de las polilíneas que consiste en que el último punto de una línea es el primero de la siguiente. Observe que el sistema no tiene que ejecutar alguna otra acción o algoritmo para conservar dicha propiedad, ya que ni la descripción de la polilínea ha sido afectada ni la descripción de las líneas que la conforman. Tan solo la posición de los puntos que las definen ha sido actualizada. Como podemos ver la línea *línea4* sigue siendo descrita por los puntos *punto4* y *punto5* y la línea *línea3* sigue siendo descrita por los puntos *punto3* y *punto4.* Como la posición de *punto4* ha cambiado, entonces las caracteristicas geométricas de las líneas definidas por este punto también cambian (la posición de *línea4* y la longitud de *línea3*).

Ahora procedemos a desglosar el proceso de traslación de un objeto de tipo polilínea. Usted le indica al sistema que va a trasladar una polilínea y con el ratón selecciona en una de la líneas que forman la polilínea. El sistema encuentra la línea que fue seleccionada según el proceso explicado anteriormente y luego busca a cual polilínea pertenece esa línea. A continuación busca si la polilínea encontrada forma parte de alguna otra polilínea; si es así, busca si esta última pertenece a su vez a alguna otra polilínea y así sucesivamente hasta llegar a la polilínea que ya no forma parte de otra. Por ejemplo, si nostros seleccionamos una polilínea escogiendo a *línea2,* el sitema encuentra que esta línea pertenece a *polilínea1,* pero ésta a su vez esta incluida en *polilínea2,* la cual a su vez forma parte de *polilínea3* que no forma parte de otra. De esta forma encuentra que hemos escogido a *polilínea3.* Al hacer la operación de arrastre el proceso de traslación es similar al que se aplica en la traslación de líneas: se calcula el vector diferencia entre la posición donde se escogió el objeto y el lugar donde se desea como nueva posición, dicho vector diferencia se suma a los puntos que forman las líneas que componen la polilínea (Obteniendo algo similar a lo que se muestra en la figura 3.3 c). En conclusion, en S_2RS el movimiento de los objetos se reduce a la aplicación del vector diferencia sobre los puntos que definen el objeto escogido.

En las modificaciones de traslación de objetos que se han mencionado puede surgir la pregunta de a qué tipo de objeto se quiere referir el usuario al escoger un punto (con el ratón) en un objeto dado?. Bien puede ser que se quiere referir a un punto solamente, pero su intención quizá sea referirse a la línea que pasa por el punto escogido, o a la polilínea formada por dicha línea o, por qué no, desea referirse al polígono formado por esa polilínea. En S_2RS se provee de un menú en la interface gráfica mediante la cual el mismo usuario escoge el tipo de cursor que quiere usar, de forma tal que si su intención es trasladar un punto, entonces informa al sistema mediante el menú que su traslación es sobre un objeto de tipo punto. Podemos comparar este tipo de operación con la que se hace en otros editores gráficos en los

que para modifcar esctructuras más complejas se agrupan varios objetos y el editor los reconoce como una sola. Esta operación no es exactamente igual. En S_2RS escogemos el cursor de acuerdo a los tipos de objetos que el sistema reconoce, mientras que en los editores gráficos tradicionales, se agrupa un conjunto de objetos escogiendo el área de la pantalla en donde están los objetos de interés. En S_2RS se tomaron además algunas convenciones para escoger un objeto. Una vez que el sistema es enterado del tipo de objeto que se va a trasladar, la elección del objeto será, de acuerdo con el tipo de objeto, una de las siguientes acciones:

a) Si el cursor es para un objeto de tipo punto, entonces se escoge con el ratón (pick) en el punto de la pantalla donde está el punto gráfico deseado.

b) Si el cursor es para un objeto de tipo línea, entonces se escoge con el ratón en un punto de la pantalla por donde pasa la línea que se desea escoger.

c) Si el cursor es para un objeto de tipo polilínea, entonces se escoge con el ratón en un punto de la pantalla por dónde pasa una línea que forma parte de la polilínea. Note que aún cuando la operación es idéntica que para escoger un objeto de tipo línea, la elección del tipo de cursor elimina la ambigüedad que esta situación pudiera generar.

d) Si el cursor es para un objeto de tipo polígono, entonces se escoge con el ratón en un punto de la pantalla que se encuentre dentro del área fomrada por el polígono.

Ahora que hemos visto como se hacen las traslaciones de los objetos en S_2RS podremos comprender cómo es que se satisfacen algunas restricciones luego de una operación de traslación que afecta a objetos con descripciones restrictivas.

3.2.2. Referencia en la Satisfacción de Restricciones.

Consideremos ahora la figura 3.4 en la que podemos apreciar un rectángulo dividido a la mitad verticalmente y la mitad superior dividida horizontalmente en dos partes iguales. Este ejemplo es nspirado de los ejemplos de Borning [Maloney 89]. A la derecha de la figura se encuentra la descripción de la misma. Observe las descripciones que corresponden a las líneas *línea5* y *línea6* del dibujo, en las que la descripción contiene los elementos necesarios para establecer las restricciones que queremos que conserven las divisiones en el rectángulo. Por ejemplo, el primer punto de *línea5* está restringido a permanecer en el punto medio entre los puntos extremos de *línea4 (punto4* y *punto1).* El segundo punto de línea5 está restringido a ser el punto medio entre los puntos extremos de *línea2 (punto2* y *punto3).* La línea *línea6*

también esta restringida en su descripción, el primer punto es el punto medio de los puntos que definen *linea3* (*punto3* y *punto4*) y el segundo punto es el punto medio de los puntos que forman a *linea5*. Con estas descripciones limitamos a la figura a estar dividida en dos partes iguales y la mitad superior a estar dividida horizontalmente también en dos partes iguales.

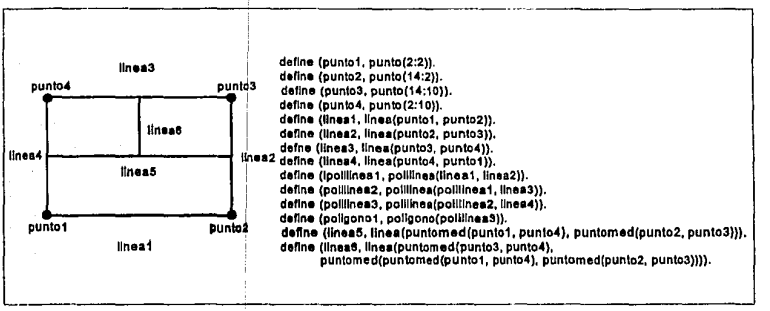


Figura 3.4.-Ejemplo de una ventana dividida en tres paneles

De acuerdo al inciso anterior podemos predecir que pasa si se traslada de lugar una línea del dibujo de la figura 3.4. Por ejemplo, si trasladamos la línea *linea2* hacia la derecha obtendremos el dibujo que se muestra la figura 3.5 en donde también se muestra la descripción correpondiente. En la descripción de esta figura podemos observar que la base de conocimientos no ha cambiado en gran medida, lo que ha cambiado han sido las coordenadas de *punto3* y *punto4,* pero las descripciones de *linea5* y *linea6* permanecen constantes. ¿Cómo es que han sido satisfechas las restricciones?. La misma descripción de los objetos hace la satisfacción de sus relaciones: se modificaron las coordenadas de los elementos básicos que definen al objeto *linea2,* es decir, los objetos *punto3* y *punto4.* El programa intérprete recibe la base de conocimientos modificada de esta manera, interpreta la descripción de cada uno de los objetos y actualiza el dibujo. Por lo que dadas las nuevas coordenadas de *punto3* y *punto4,* se redibujan los objetos en base a sus descripciones ya interpretadas. De esta manera no se tiene que recurrir a un algoritmo de satisfacción de restricciones: las restricciones se resuelven gracias a la descripción de los objetos. Podemos decir que los algoritmos que podrían satisfacer restricciones de este tipo son innecesarios cuando se cuenta con una expresividad adecuada en el lenguaje de

representación que nos permita hacer descripciones que involucren las relaciones gráficas que deseamos que permanezcan invariantes durante el curso de la sesión gráfica interactiva.

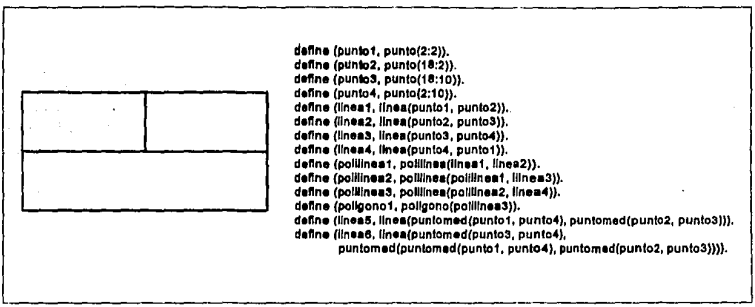


Figura 3.5.-Figura que muestra el resultado obtenido de mover la línea lateral derecha (línea2) de la figura 3.4. El resultado se muestra en la presentación gráfica y en la base de conocimientos.

En este punto es importante resaltar que en la sesión interactiva, a descripciones más explícitas, mejores resultados se obtienen. Note que en el dibujo de la figura 3.4 no se ha establecido que el polígono deba permanecer en forma rectangular, por lo que el usuario no debe esperar que las rectas que conforman el polígono permanezcan en forma vertical u horizontal en el transcurso de una sesión interactiva en la que ocurran traslaciones de los objetos. Observe que si selecciona el punto *punto2* para ser trasladado a una nueva posición (abajo a la derecha, por ejemplo), entonces se obtendría algo similar a la figura 3.6. Note sin embargo que nuevamente se resuelven las restricciones impuestas a las líneas línea5 y línea6 por la descripción de estos objetos. Este tipo de soluciones son las que Pineda denominó Soluciones por Referencia. Una forma de asegurar que al trasladar el punto *punto2* permanezcan verticales las líneas *línea2* y *línea4,* y que permanezcan horizontales las líneas *línea1* y *línea3,* es modificando la descripción del dibujo. Una descripción que refleja esta intención se muestra en la figura 3.7. Si se aplica una traslación similar a la que se aplicó a la figura 3.5 (sobre el punto inferior derecho) en la figura 3.7, entonces se obtiene un dibujo similar al que se muestra en la figura 3.8.

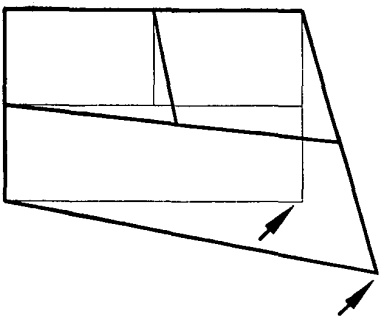


Figura 3.6.-Respuesta obtenida en S_2RS al desplazar el punto inferior derecho de la figura 3.4.

Observe que la traslación aplicada sobre el punto inferior derecho de la fig 3.7 simula una operación de redimensionamiento de pantalla. Esto se debe a la descripción del dibujo. El sistema S_2RS es hecho con la intención de explorar si es factible y en que medida pueden ser resueltos los problemas de satisfacción de restricciones por un nivel de estructura en los objetos a la altura de objetos geométricos básicos y con un lenguaje simbólico capaz de expresar diferentes relaciones en forma de descripción de dichos objetos. Este ejemplo sirve para demostrar que la manipulación simbólica también es una opción para sistemas cuyas intenciones sean resolver problemas de satisfacción de restricciones en el área de interfaces, aunque si se quiere usar S_2RS para explorar las restricciones en el diseño de interfaces sería más cómodo implementar las operaciones de manipu-lación de objetos necesarias para este fin, por ejemplo, el redimensionamiento de objetos.

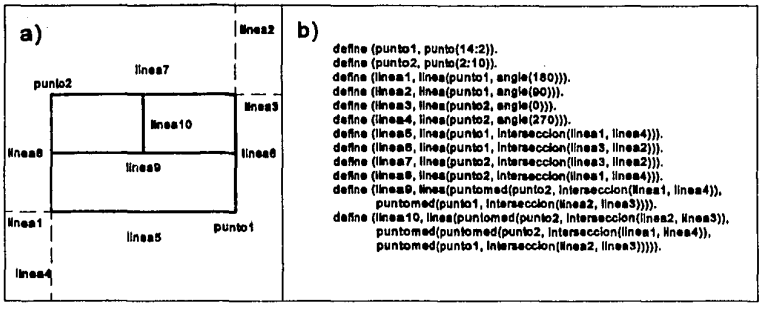


Figura 3.7.-En esta figura se muestra una diferente definición de una ventana dividida en tres paneles

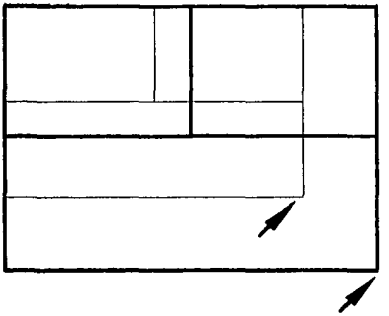


Figura 3.8.-Respuesta obtenida en S_2RS al desplazar el punto inferior derecho de la figura 3.7.

El uso de sistemas de satisfacción de restricciones para resolver problemas que surgen en el diseño de interfaces de usuario ha sido explorado con éxito en el sistema ThingLab II [Maloney et all, 1989]. De hecho, el ejemplo de la figura 3.8 ha sido probado en ThingLab II.Como recordará del capítulo 1 de este trabajo, el enfoque que se da a las restricciones en ThingLab II y la manera en que se resuelven no es por manipulación simbólica. Para este ejemplo, tanto en ThingLab II como en S_2RS, las restricciones que se aplican son, en forma verbal:

1.- Que la ventana este dividida verticalmente en dos partes iguales.
2.- Que la mitad superior esté dividida en mitades iguales a derecha e izquierda.

Aunque las restricciones aplicadas al dibujo son las mismas en ambos sistemas, la manera en que se aplican al dibujo difiere, lo mismo que la forma en que se resuelven. Para ThingLab las restricciones que se aplican a los objetos consisten de un predicado que puede ser usado para probar si la restricción está satisfecha y de uno o más métodos que pueden ser usados para alterar alguna parte o subparte del objeto restringido para satisfacer la restricción [Borning, 1986]. La satisfacción de restricciones en ThingLab es hecha por el *constraints satisfier* (propagación local y sus variantes) que es el mecanismo que decide cuáles restricciones deben ser satisfechas, cuál método debe ser usado para satisfacer cada restricción y en qué orden deben ser invocados los métodos para satisfacer las restricciones.

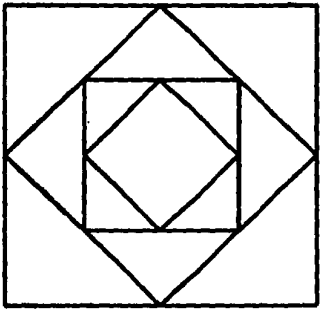


Figura 3.9.-Cada línea de cuadrilátero interno esta acotada por los puntos medios de líneas de su cuadrilátero externo.

Como se ha dicho anteriormente, para S_2RS las restricciones forman parte de la descripción del objeto y son resueltas en la interpretación de dichas expresiones. Un ejemplo que muestra en mayor escala la ventaja de este tipo de soluciones es el de la figura 3.9, ejemplo que también ha sido probado en el sistema ThingLab II. El ejemplo es usado para resolver el siguiente teorema de geometría: dado un cuadrilátero arbitrario, si uno dibuja líneas entre los puntos medios (de las líneas que conforman el cuadrilátero) adyacentes, entonces las nuevas líneas formarán un paralelogramo [Borning, 86]. La misma técnica de construción es aplicada para crear los paralelogramos interiores. En este ejemplo (llamado "de los cuadriláteros

anidados") que corre en S_2RS, uno puede escoger cualquiera de los puntos vértices del cuadrilátero exterior y trasladarlo hasta la posición deseada. La satifacción de restricciones por referencia se encarga de actualizar el dibujo satisfaciendo las restricciones (véase figura 3.10 b)).

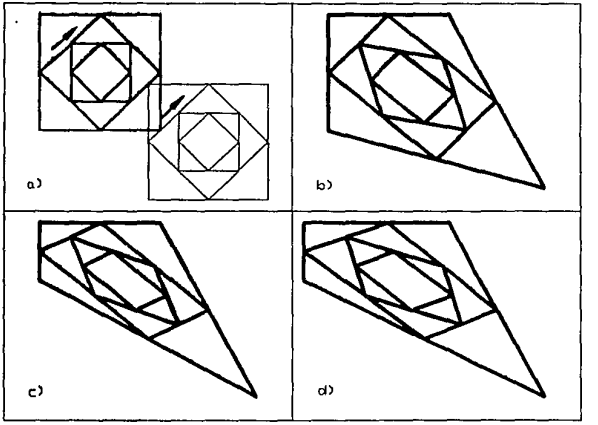


Figura 3.10.-Secuencia de traslaciones y satisfacción de restricciones en la figura de cuadriláteros anidados. a) Traslado del polígono más externo. b) traslado del punto inf. derecho. c) Traslado del punto inf. izquierdo. d) traslado de línea lateral derecha.

En S_2RS podemos obtener otros resultados interesantes. En la figura 3.10 a) se muestra cómo al desplazar el polígono exterior, son actualizados los demás cuadriláteros dadas sus descripciones. Es decir, los cuadriláteros internos son trasladados sin que el usuario tenga que expresar al sistema que se deban desplazar: el usario ya espera que esto suceda porque ha establecido las relaciones entre los objetos del dibujo. En la figura 3.10 b) se muestra una acción que sigue a la de la figura 3.10 a), en esta se traslada el punto inferior derecho del cuadilátero exterior a una nueva posición. La siguiente acción reflejada en la figura 3.10 c) es el desplazamiento del vértice inferior izqierdo hacia arriba y por último en la figura 3.10 d) se traslada la línea lateral izquierda un poco más a la izquierda. Durante toda la interacción mostrada en la figura 3.10 podemos observar la satisfacción de las

restricciones, no importando a qué nivel de abstracción (punto, linea, polígono) afectemos al cuadrilátero más externo. También podemos observar que al modificar objetos simples se modifican objetos compuestos (modificar un punto provoca la modificación de toda la polilínea) y al modificar objetos compuestos se modifican objetos simples (modificar el polígono es la modificación de los puntos que lo conforman.

Dado lo anterior podemos decir que el objetivo del sistema se ha cumplido. El sistema es capaz de interpretar las intenciones del usuario canalizadas a través de descripciones que permiten expresar dichas intenciones. Si el usuario tiene la inteción de que la línea que va a dibujar vaya de la mitad de una línea a la mitad de otra, entonces se restringe a esta línea mediante una descripción que refleja esta intención, el sistema interpreta esta descripción, obtiene las coordenadas en la pantalla a las que corresponde la descripción dada y la presenta. A cada nueva modificación de los objetos del dibujo, el sistema interpreta las descripciones y presenta el nuevo estado dada las nuevas coordenadas.

Nuestro ejemplo de la fig. 3.10 en ThingLab II contiene un total de 50 de sus restricciones. Muchas de ellas se refieren al punto medio entre líneas, consideremos que para satisfacer cada una de ellas, el resolvedor de restricciones debe darse cuenta que algunas restricciones no están satisfechas luego de una modificación de traslación sobre un punto, después debe de trazar un plan para satisfacer las restricciones insatisfechas en el que se incluye cuál de los métodos de ThingLab (propagación local) debe ser usado para satisfacer cada restricción y en que orden las va a ir resolviendo. En situaciones como ésta pueden apreciar las ventajas de un lenguaje de respresentación que nos permita expresar relaciones entre objetos dentro de descripciones de otros.

3.3 La Componente de Síntesis en el Proceso de Satisfacción de Restricciones.

Existen restricciones que no pueden formar parte de la descripción de un objeto geométrico. Las relaciones de paralelismo, perpendicularidad y colinealidad entre dos líneas y las propiedades de verticalidad u horizontalidad de una línea son restricciones con las que no podemos construir objetos geométricos básicos porque el resultado del proceso de interpretación de estas relaciones es un valor booleano. Por ejemplo, para construir un punto se requieren de dos objetos de tipo real, para construir una polilínea necesitamos de dos objetos de tipo línea o un objeto de tipo polilínea y uno de tipo línea, etc. Podemos repasar todos los constructores del lenguaje y encontraremos que

el tipo booleano no es un tipo que forme parte de la aridad de los constructores de los objetos geométricos conocidos en el lenguaje G.

Las restricciones que no pueden formar parte de la descripción de un objeto requieren de un método adicional de solución de problemas. Para satisfacer una restricción de esta naturaleza es necesario recurrir a procesos llamados de síntesis y que no serán tratados a fondo en este trabajo porque requeriría un estudio por separado y sale del alcance de los objetivos de esta tesis. Solo se mencionan algunas características del proceso para tener en cuenta que no todas las restricciones pueden ser resueltas por referencia.

De manera muy general, el proceso de síntesis consiste en primer lugar de verificar si las relaciónes son satisfechas en el estado actual del dibujo. En el proceso de interpretación de la base de datos, si el resultado de la interpretación son valores booleanos de verdadero, significa que las restricciónes están satisfechas. Si el resultado del proceso de interpretación dice que existen condiciones no satisfechas (valores booleanos de falso) entonces es necesario recurrir a un mecanismo que permita encontrar una solución, mecanismo que resolverá una a una las restricciones procurando que la solución no contenga contradicciones entre las restricciones (que se cumplan una pero otras no). Aún cuando las soluciones para este tipo de restricciones no están desarrolladas en S_2RS, se cuenta con los algoritmos que verifican si estas relaciones se cumplen. Esta funcionalidad ha sido integrada a S_2RS con el objetivo de poder extender este sistema para que no solo resuelva las restricciones por referencia sino las de síntesis también en un futuro. El resultado de este tipo de proceso generalmente hace una aportación semántica en el dibujo; es decir, generalmente la solución tiene que hacer un cambio en uno de los objetos involucrados en la restricción. Veamos algunos ejemplos sencillos de esto.

Suponga que contamos con un sistema gráfico interactivo capaz de resolver restricciones del tipo de las que hemos venido hablando. Considere que en el estado actual de las sesión interactiva contamos con el dibujo que se muestra en la figura 3.11 a), la figura muestra simplemente la forma en que fueron dibujados los objetos, sin aplicar restricciones aún sobre ellos. El hecho de que las líneas luzcan verticales y horizontales es contingente. Procedamos a restringir el dibujo. Supongamos que podemos expresar al sistema que deseamos que las líneas línea4 y línea2 son paralelas. El sistema capura esta intención y la almecena en su base de datos guardando la expresión paralelas(línea4,línea2) como objetivo de un proceso de solución de problemas. Ahora procedemos a mover el punto *punto3* como se muestra en la figura 3.11 b). Después de la interacción esperamos que el sistema satisfaga la restricción que hemos impuesto al dibujo.

Supongamos que el sistema interpreta las expresiones que describen el dibujo completo y que se da cuenta que la expresion de paralelismo entre línea2 y línea4 es falsa dado el estado actual del dibujo en la pantalla. El sistema debe de contar con un mecanismo que le permita solucionar restricciones de este tipo. Podemos percatarnos de que la solución a este problema gráfico no es única y que podemos contar con un número considerable de configuraciones que hacen verdadera la relación impuesta en los objetos geométricos. Por ejemplo, una opción sería que el sistema decidiera regresar el punto *punto3* a su posición original, resultado que satisface la restricción pero que en ningún modo es una solución aceptable ya que contradice la ultima acción del usuario que era la de modificar la posición de *punto3.* Dos soluciones diferentes que son más aceptables son las que se muestran en las figuras c) y d). En la figura c) se cambia de posición el punto *punto5;* es decir, se hace paralela a la línea *línea4* con respecto a *línea2* dejando fijo al punto *punto6.* En la figura d) se muestra como se hace paralela la línea *línea4* con respecto a la *línea2* dejando fijo al punto *punto5.* Otra solución que es aceptable es cambiar los dos puntos de posición de manera que la distancia que separa ambas líneas no sea modificada y así no afectar mucho la apariencia del dibujo original.

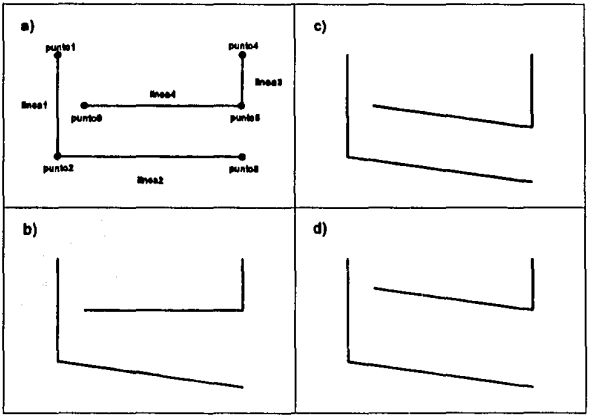


Figura 3.11.-Satisfacción de restricciones por síntesis. a) Figura etiquetada con los objetos que la componen; línea2 y línea4 se restringen a ser paralelas. b) traslación de *punto3.* c) y d) Dos soluciones diferentes obtenidas por síntesis dado el movimiento mostrado en b).

El sistema supuesto puede contar con cualquiera de las soluciones mostradas anteriormente, o con dos o varias de ellas dependiendo de la forma en que este hecho el sistema. Por ejemplo, en Graflog, el usuario puede decir si la solución no le satisface el sistema busca una nueva solución. El problema es saber cual de esas soluciones es la que el usuario tenía en mente al momento de hacer la modificación. Este y otros problemas son discutidos ampliamente en [Pineda 92]. Para el caso que estamos tratando, con las dos primeras soluciones propuestas se afecta la distancia entre las líneas, con la segunda se afectan las posiciones de los dos puntos que definen a *línea4*. A esto es a lo que nos referimos con aportación semántica en la solución de las restricciones que nos ocupan en este inciso.

A los procesos de solución para restricciones en los que el sistema tiene que determinar si la restricción se cumple o no (verdadero o falso respectivamente) y después la soluciona de acuerdo a una o varias reglas según la configuración del dibujo y las restricciones implicadas, son a las soluciones que Pineda denomina por Síntesis.

En la mayoría de los sistemas actuales de satisfacción de restricciones no se distingue entre los problemas de referencia y los problemas de síntesis. De hecho, sus soluciones para toda restricción son por síntesis como el sistema ThingLab y el Graceful. Lo cual no sorprende porque sus métodos de solución no son por manipulación simbolica de objetos gráficos sino soluciones numéricas de ecuaciones que describen a los dibujos y sus restricciones (ver Capítulo 1). Por lo que podemos concluir que la manipulación simbólica por medio de un lenguaje de representación de objetos gráficos sencillos permite dar soluciones fáciles de entender al usuario e incluso permite simplificar el proceso de solución a grado de que se pueden distinguir dos tipos de soluciones: por referencia y por síntesis.

Concluimos esta sección con el ejemplo que hemos venido manejando a lo largo de la tésis. Este ejemplo ha sido probado en S_2RS. El ejemplo del cuarto de baño, es un ejemplo sencillo pero que nos ha resultado útil para casos de ilustración. En la figura 3.12 se muestra el dibujo del cuarto de baño. La representación de este escenario ha ido mostrada en la figura 2.9. Procedemos a explicar el proceso de solución de problemas cuando dicha estructura es modificada. En este dibujo se representa la pared y la puerta del cuarto de baño mediante una trayectoria abierta (polilínea). Se representa también la tina mediante un rectángulo. Finalmente, las tuberías que conectan a la tina para que ésta sea funcional son representadas mediante líneas rectas. Considere que si la tina no está conectada, no se puede drenar el agua.

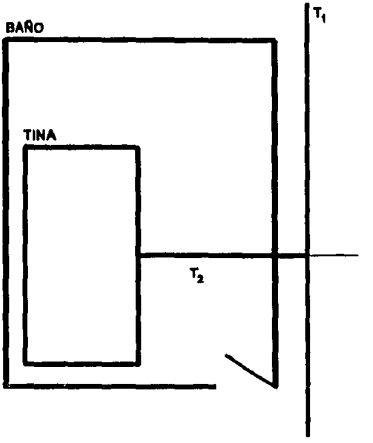


Figura 3.12.-Cuarto de baño obtenido con expresiones del lenguaje G y desplegado en la interface de S_2RS.

Los objetos del dibujo estan representados simbólicamente por expresiones del lenguaje G. El cuarto de baño está formado por una polilínea y la tina de baño está representada por un polígono. Cada una de las dos tuberías esta representada por una línea. Para mantener la funcionalidad de la tina de baño se crea a la línea que representa a la tubería T_1 a partir de una línea de construcción cuyo punto se restringe a ser el punto medio de la línea lateral derecha del polígono. Esto se hace con el fin de que la tubería permanezca a la mitad de la tina lo cual se logra con la restricción *puntomed* que recibe como paramétros dos puntos y entrega como valor un punto que es el punto medio entre los puntos dados. El ángulo de la línea de construcción se establece de 0° para forzar a que la línea que forma a la tubería T_1 permaneza horizontal. Con la línea de construcción se tiene una línea que parte de la mitad de la tina hasta el fin de la pantalla. Como se ha dicho, la línea de construcción no es en sí misma parte de un dibujo sino que sirve como lo hace una línea de trazo en dibujo arquitectónico. En S"RS las líneas de construcción son dibujadas como líneas

delgadas. Por lo tanto, la tubería T_1 se hace con el constructor *línea* cuyo primer punto sera la intersección en T (*unión_t*) entre la línea lateral derecha de la tina y la línea de construcción. El segundo punto será la intersección entre la línea que representa a T_2 y la línea de construcción. Considerando que la línea lateral derecha del polígono es la línea ***línea3***, que está formada por los puntos *punto1* y *punto2*, y que la línea que representa a la tubería T_2 es la línea ***línea4*** Las expresiones en el lenguaje G para la línea de construcción y la tubería son:

línea1 = línea(puntomed(punto1,punto2), angulo(0))

para la línea de construcción y

línea2 = línea(unión_t(línea3, línea1), intersect(línea4,línea1))

para la tubería T_1.

Con las tres restricciones sobre este dibujo es suficiente para lograr la funcionalidad de la tina. Dadas estas restricciones y para cualquier traslación de la tina:

a) la tina y la tubería estan garantizadas a estar conectadas.

b) Ambas tuberias tienen la misma garantía, c) la tubería T_1 no excede los límites entre la línea lateral derecha de la tina y la tubería T_2 y,

d) La tuberia T_1 tendrá la menor longitud entre la línea lateral derecha de la tina y la tuberia T_2.

Como podemos observar en este ejemplo, las restricciones forman parte de la descripción de los objetos por lo que dada una traslación de la tina hacia una posición cualquiera, las restricciones son satisfechas por referencia. Resultados de diferentes traslaciones de este ejemplo obtenidos en S_2RS se muestran en la figura 3.12. Podemos observar también que mientras para programación matemática este ejemplo arrojaba un numero de ecuaciones considerable además de los procesos de linealización que producirían matrices de un número elevado de elementos como se vió en el capítulo 1 de este documento, para sistemas de manipulación simbólica este problema puede ser fácilmente resuelto con sólo tres restricciones en su descripción. Unicamente resta hablar acerca de las detalles del desarrollo del sistema que serán tratados en la siguiente sección.

3.4 Implantación del Sistema S_2RS.

El objetivo de esta sección es el de proveer al lector y al usuario de un documento que contemple cada una de las partes del sistema, esclarecer la forma en que son resueltas las restricciones y mencionar cómo son las manipulaciones en el sistema prototipo. Puede servir al usuario como manual de referencia y también puede servir para dar información a el interesado en extender el sistema para que resulelva restricciones por síntesis.

3.4.1.-Arquitectura funcional del sistema.

El sistema prototipo con el que se validan las ideas plasmadas en este trabajo es el sistema llamado S_2RS cuyo nombre se deriva de las función que éste desempeña: Satisfacción de Restricciones por Referencia Simbólica. Un diagrama de bloques general de la arquitectura funcional del sistema se muestra en la figura siguiente:

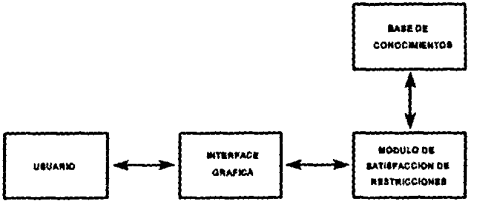


Figura 3.13.-Diagrama de bloques del Sistema S_2RS

El sistema corre actualmente en una DECstation 5000 con sistema operativo ULTRIX y bajo el sistema de ventanas X-Windows. El módulo de satisfacción de restricciones fué programado en su etapa inicial en C-Prolog y actualmente se encuentra en BinProlog. La interface gráfica esta programada con el sistema para desarrollo de interfaces gráficas Tk/Tcl.

3.4.2 La Interface Gráfica.

Para probar que la manipulación simbólica es una alternativa provechosa en la solución de restricciones gráficas bastaba con obtener los resultados dados por el módulo de satisfaccion de restricciones que estaban en expresiones del lenguaje G. Sin embargo, sería una carga muy pesada para algún interesado en el sistema el hecho de que tuviera que aprender el lenguaje G para hacerle una pequeña demostración. Dado lo anterior se provee de una interface gráfica que, entre otras cosas, sirve como traductor entre las expresiones del lenguaje G y las figuras geométricas que representan.

En el prototipo nos hemos enfocado a la parte interactiva que es de interés para el tema que ocupa a esta tesis, aquella que corresponde a la modificación de los objetos gráficos en la pantalla con el objetivo de ver si se mantienen las condiciones asociadas a un dibujo. Por lo tanto, la creación de objetos gráficos y asociación de restricciones de manera interactiva no son contempladas en este sistema. Los problemas de este tipo de interacción son una extensión al presente trabajo.

La interface de usuario está programada con el sistema para desarrollo de interfaces de usuario graficas Tcl/Tk. Tcl (tool command language) es un lenguaje de scripts para hacer y controlar aplicaciones. Su intérprete está hecho con una librería de procedimientos de C. Tk es una de las extensiones más útiles de Tcl. Tk es un toolkit para hacer interfaces gráficas de usuario del sistema de ventanas X11. Tk extiende a Tcl con comandos adicionales para construir interfaces de usuario. Tk también está hecho como una librería de procedimientos en C. Una de las razones por las que es muy rápido desarrollar interfaces con Tcl/Tk es que es un lenguaje interpretado. Cuando se usa la aplicación llamada *wish* uno puede ejecutar y generar al mismo tiempo sin tener que recompilar o reiniciar la aplicación. La aplicación *wish* es un shell par Tcl/Tk su nombre es la contracción de windowing shell. Cuando *wish* es invocado desde la línea de comandos se despliega una ventana vacía en la pantalla y de manera interactiva lee comandos de Tcl/Tk desde la entrada estándar y los pasa a evaluar a el intérprete Tcl/Tk.

Tcl/Tk cuenta con comandos que permiten crear los elementos de interface llamados widgets. Los widgets son los elementos básicos de una interface de usuario; las etiquetas de una ventana, los botones, los menús, las espacios para texto, la pantalla para dibujar (canvas), son ejemplos de widgets. La programación en Tk permite que al crear los widgets uno pueda asociar acciones a ejecutarse cuando ocurre un evento en la pantalla. Los eventos son acciones del usuario expresadas a través del teclado o del ratón, ejemplos de eventos son oprimir algun bontón del ratón, liberar algún boton, oprimir la tecla 'return' del teclado u oprimir cualquier letra del

teclado.

El trabajo que desarrolla la interface en S_2RS es simplemente el de reflejar en forma gráfica el estado de la base de conocimientos que se tenga en el sistema. La manipulación de los objetos gráficos y el razonamiento necesario acerca de dichas manipulaciones es ejecutado en los programas de prolog. La función principal de la interface es la de obtener las coordenadas cuando uno selecciona un objeto en la pantalla (coordenadas que son procesadas en los programas de prolog) y desplegar en la pantalla el estado de la base conocimientos.

Recorriendo con la vista a la interface gráfica de S_2RS de arriba hacia abajo, se tiene en primer lugar a la barra de título, enseguida se puede observar a la barra de menús. Al oprimir el boton 1 del ratón sobre alguna de las opciones se despliega un menú de opciones. Cada menú de opciones contiene una conjunto de botones acomodados en forma de lista los cuales se activan al oprimir y liberar el boton 1 del ratón.

3.4.2.1 El Botón de Menú 'File' de la Barra de Menús

En la hoja 1 del Apendice A se puede observar al menú que correponde a la opción 'File'. La opción 'File' contiene el conjunto de botones que ejecutan alguna acción relacionada con la manipulación de archivos. En orden descendente se pueden distinguir cinco opciones, una para cada botón:

El Botón 'Open'.

Al seleccionar este botón, el sistema esta listo para recibir el nombre de un archivo (un dibujo) a través de la entrada de texto etiquetada como 'File' que se encuentra en la región inferior izquierda de la interface. El nombre del archivo debe corresponder a alguno que esté escrito en Lenguaje G. Una vez especificado el archivo, éste es cargado como la base de conocimientos del sistema y es desplegado en la pantalla de dibujo.

El Botón 'Save'.

Cuando este botón es seleccionado el sistema esta listo para recibir un nombre con el que se guardará el dibujo que se encuetre en ese momento en la pantalla y que podrá ser cargado con la opción 'Open'. La forma de darle el nombre del archivo es a través de la entrada de texto etiquetada 'File'. Cabe hacer notar que durante una interacción el sistema no hace autorespaldos, de manera que si se ha modificado un dibujo y se quiere guardar así, debera ser salvado con esta opción.

El Botón 'Clear'.

Seleccionando este botón se limpian las variables que este usando la interface a su valor inicial, se borra lo que se tenga en la base de conocimientos y se limpia la pantalla. Despues de esta acción se puede volver a abrir otro archivo para comenzar una nueva interacción.

El Botón 'Quit'.

Este botón sirve para dar por terminada una sesión con S_2RS.

3.4.2.2 El Botón de Menú 'Draw' de la Barra de Menús.

Seleccionando este botón aparece un menú con opciones que permiten crear los objetos gráficos que el sistema conoce. Por las razones que se exponen en la explicación del siguiente botón de Menú, solo es posible crear objetos con descripciones simples (sin restricciones). Las opciones del menú 'Draw' son:

El Botón 'Line'.

Al seleccionar este botón, el sistema esta listo para recibir un par de puntos en la pantalla de dibujo para crear una línea. El usuario escoge un punto en la pantalla al oprimir y liberar el botón 1 en la posición deseada.

El Botón 'C_Line'.

Para crear una línea de construcción se selecciona esta opción. Cuando es seleccionado el sistema espera que en la pantalla de dibujo se escoja un punto con el botón 1 del ratón para después recibir un valor real en el widget de entrada de texto etiquetado 'Angle', éste se encuentra en la parte inferior central de la interface. El punto corresponde al punto inicial de la línea y el valor real corresponde al ángulo en grados que va a tener ésta.

El Botón 'Path'.

El botón 'Path' sirve para crear polilíneas en la pantalla de dibujo. Después de escoger esta opción, el sistema espera que se escogan puntos diferentes en la pantalla. A partir del segundo punto escogido,se empiezan a trazar líneas entre ellos.

El Botón 'Polygon'.

Seleccionando este botón, el sistema está preparado para recibir los puntos que conforman los vértices de un polígono. La manera de cerrar el polígono es oprimiendo y liberando el tercer botón del ratón. Con esta acción se crea una línea que va del ultimo punto escogido (botón 1) hasta el primero.

El Botón 'Circle'.

Mediante la elección de este botón, el sistema esta listo para recibir dos puntos de la pantalla de dibujo, el primero corresponderá a el centro de un círculo y el segundo punto marcará la distancia que tendrá el radio del mismo.

En todos estos casos, la interface envía las coordenadas de la pantalla a rutinas en Prolog que se encargan de ir formando la base de conocimientos del dibujo. Estas rutinas se explican más adelante.

3.4.2.3 El Botón de Menú 'Movements' de la Barra de Menús.

Al presionar el botón 1 del ratón se despliega un menú con cinco opciones. Las opciones de este menú son para poder referir al objeto que se desee modificar. Estas opciones son pensadas para eliminar el problema de la ambiguedad que se presenta cuando queremos seleccionar a un objeto en la pantalla. Suponga que tenemos un polígono en la pantalla y seleccionamos con el ratón sobre algun punto en una línea del polígono, es dificil decir si con esta acción nos queremos referir a ese punto de la línea, a la línea o a el polígono completo. Para evitar este tipo de problemas se han desarrollado diferentes tipos de cursores en este sistema para referir a los objetos deseados.

Cada una de estas opciones se selecciona con el primer botón del ratón. Los objetos gráficos son seleccionados al oprimir el segundo botón y la nueva posición se especifica haciendo un drag con el botón 2 oprimido y liberandolo en la posición deseada. Las opciones del menó 'Movements' son:

El Botón 'Dot'.

Cuando se selecciona este botón, el sistema esta informado de que el objeto que se va a modificar va a ser un objeto de tipo punto. Una vez seleccionado este botón, el usuario se puede desplazar con el ratón hasta el lugar del punto deseado y escogerlo con el botón 2 del ratón oprimiendolo y no liberandolo sino hasta que se haya desplazado el ratón a la posición deseada.

El trabajo de la interface en esta operación es la de obtener las coordenadas de la pantalla en donde se oprime el botón 2. Las coordenadas así obtenidas son enviadas por la interface al programa de prolog que se encarga de buscar el punto al que corresponden dichas coordenadas (pick_dot). La rutina en prolog que hace esto deja un margen de cinco pixeles de error, o sea, busca algun punto de la base de conocimientos cuya coordenada en X sea +/- 5 de la coordenada X obtenida en la

pantalla y cuya coordenada Y sea +/- 5 de la coordenada Y de la pantalla.

La segunda acción de la interface para este botón es la de llamar a la rutina de Prolog que hace que se desplace el punto identificado en la base de conocimientos (move_dot).

El Botón 'Line'.

Al oprimir el primer botón del ratón sobre esta opción, el sistema está enterado de que se va a desplazar a un objeto de tipo línea, de manera que si se oprime el segundo botón del ratón sobre alguna línea en la pantalla, la interface llama a la rutina 'pick_line' de Prolog con las coordenadas obtenidas en la pantalla al hacer la selección del objeto. Despues de hacer el drag con el segundo botón del ratón, la interface detecta este evento y llama a la rutina de prolog 'move_line' enviando como argumentos el nombre de la línea encontrada por 'pick_line' y las coordenadas donde se liberó el ratón que ayudarán a calcular la nueva posición de la línea escogida.

El Botón 'Path'.

Como el procedimiento de selección es el similar al descrito en los botones anteriores, la presente y siguientes explicaciones omitirán estos detalles. El botón 'Path' se selecciona cuando se desea desplazar completamente a una polilínea, Se hace la operación sobre la polilínea deseada oprimiendo el botón 2 del ratón sobre una de las líneas de la polilínea. y el resultado es la misma polilónea (conservando longitudes y ángulos) pero en diferente posición. Las rutinas en prolog que utiliza este botón son 'pick_path' y 'move_path'.

El Botón 'Polygon'.

Cuando se selecciona este botón el sistema espera que el usuario seleccione un objeto de tipo polígono. La selección de un objeto de este tipo debe hacerse oprimiendo el botón 2 del ratón dentro del área del polígono, ya sea en la parte central o cerca de las líneas, siempre y cuando sea en la parte interna del polígono. Cuando se termina el drag, el objeto es trasladado respetando longitudes y ángulos hasta la nueva posición. Las rutinas de prolog que son llamadas en este botón son 'pick_poly' y 'move_poly'.

El Botón 'Circle'.

Al seleccionar este botón el sistema esta enterado de que se va a hacer una operación sobre un objeto de tipo círculo. Un círculo puede ser escogido haciendo la selección del objeto dentro del área que forma. Son 'pick_circle' y 'move_circle' las rutinas en prolog llamadas al ejecutarse la selección y el desplazamiento de un círculo.

3.4.2.4 El Canvas de la Interface Gráfica.

Abajo de la barra de menús se encuentra el canvas de la interface. Un canvas es un widget que despliega una superficie de dibujo y cualquier cantidad de objetos gráficos. Los objetos gráficos que la interface de S_2RS puede desplegar son líneas y círculos. El hecho de que en el sistema se pueda hacer referencia a puntos, polilíneas y polígonos se debe a que son estructuras que se encuentran en la base de conocimientos. Aún cuando en el canvas se pueden observar todas estas estructuras, lo que el sistema dibuja son líneas. Para el usuario este hecho es transparente. Por ejemplo, él ve un polígono y se refiere a éste como un polígono, el polígono existe en la base de conocimientos, pero lo que la interface dibuja son sólo las líneas: Tk no sabe que ese conjunto de líneas forman un polígono. Es por eso que aquí se dice que el razonamiento sobre los objetos gráficos durante las traslaciones es hecho por las rutinas de prolog y que la interface solo sirve como medio de despliege gráfico.

3.4.3 El Módulo de Satisfacción de Restricciones.

Según Moloney el resolvedor de restricciones debe decidir cuándo y cómo resolver las restricciones dando una solución conforme el usuario interactúa con los objetos [Maloney 87, pp 381]. El módulo de satisfacción de restricciones, como ya se ha dicho, ha sido implementado en Prolog. Se ha escogido este lenguaje porque Prolog es un lenguaje de programación para computación simbólica. El módulo ha sido dividido en 4 programas según el tipo de operaciones que realizan las reglas de Prolog.

El programa 'g_intrp'.

Este programa es la codificación en Prolog del lenguaje G y de las reglas de interpretación de dicho lenguaje. Su función es la de recibir una expresión del lenguaje y entregar, de acuerdo a la base de conocimientos, una expresión del lenguaje G en forma básica, así como el valor de un indicador que sirve para detectar si la interpretación ha sido parcial (un elemento error) o total (un elemento 'dibujable'). A grandes rasgos, las acciones del programa *g_intrp* son:

(1) Recibir como argumentos la base de conocimientos en forma de lista y una expresión del lenguaje G a interpretar

(2) Separar la lista de la base de conocimientos en forma de hechos independientes que serán los hechos sobre los que prolog hará las inferencias.

(3) Se llama a la regla *eval* que evaluará a la expresión en turno. Esta regla devuelve el valor de la interpretación de la expresión y el de una bandera llamada *Detect*. La bandera tiene la función de reflejar el tipo de dato del objeto

interpretado; ya sea de un elemento error (punto_e, línea_e, etc) o de un elemento dibujable (punto, línea, etc). Existen al menos dos variantes de la regla para cada símbolo (constructor, selector, etc) del lenguaje G, una de ellas contempla a las expresiones cuya interpretación resulta errónea y otra a las interpretaciones que entregan expresiones básicas del lenguaje. De acuerdo a la bandera detect, se determina cuál es el estado de la interpreta-ción: 'O.K.' si el resultado de la interpretación es una expresión básica del lenguaje, o 'error' si no lo es.

(4) Por último se borran los hechos que sirvieron al proceso de interpretación.

La regla eval es el núcleo del proceso de interpretación. Es en ella donde se puede observar la recursividad del lenguaje y el por qué de las características de la interpretación parcial. Como un ejemplo de las variantes en la regla *eval* veamos las que corresponden al objeto polígono:

1ra. Regla:

Evaluar una expresión del tipo *polígono(P1)* guardando en valor en una variable *Val* y reflejando el tipo de la expresión en la bandera *Detect*.

Evaluar a la expresión P1 guardando su valor en la variable Val1 y reflejando el tipo de la expresión en la bandera Detect1.

Si Detect1 = polilínea y,

Si Val1 es una polilínea que comienza y termina en un mismo punto

entonces el valor de la interpretación Val será una expresión de la forma polígono(Val1).

2da Regla:

Evaluar una expresión del tipo *polígono(P1)* guardando en valor en una variable *Val* y reflejando el tipo de la expresión en la bandera *Detect*.

Evaluar a la expresión P1 guardando su valor en la variable *Val1* y reflejando el tipo de la expresión en la bandera *Detect1*.

Si *Detect1 = polilínea_e*

entonces el valor de la interpretación *Val* será una expresión de la forma *polígono(Val1)* y la bandera *Detect* reflejará un elemento de tipo *polígono_e*.

Tanto en la regla para el elemento error como en la regla para el elemento normal se usa la regla polycondit que sirve para verificar la condición de polígono en una polilínea: la polilínea comienza y termina en el mismo punto.

Las dos variantes de eval descritas anteriormente se encuentran programadas en el orden descrito. Primero la que produce elementos normales (llamados así a los elementos que tienen una descripción básica, es decir, a los elementos dibujables) y después la que es para los elementos de tipo error, La razón es la siguiente: Prolog es secuencial en sus búsquedas: intenta satisfacer la primer regla que se aparea con la expresión que se va a interpretar, si no se cumple una de las condiciones para esta regla, Prolog intenta satisfacer a la expresión con la regla que aparee que se encuentre posterior a la primera.

Según la definición de las dos reglas para polígono, se pueden presentar los siguientes casos:

a) Se cumple la primera regla:

Si	Detect1 = polilínea y Val1 cumple con la condición de polígono,
entonces	Detect = polígono y Val es la descripción básica del polígono.

b) Fallan ambas reglas:

Si	Detect1 = polilínea y Val1 no comple con la condición de polígono
entonces	la expresión no corresponde a un polígono.

c) Falla la primera regla, se cumple la segunda:

Si	Detect1 = polilínea_e y Val1 cumple con la condición de polígono,
entonces	Detect = polígono_e y Val = polígono(polilínea_e).

d) Fallan ambas reglas:

Si	Detect1 = polilínea_e y Val1 no cumple con la condición de polígono
entonces	la expresión no corresponde a un polígono.

Podemos observar que los casos a) y b) son los que se presentarán realmente en una sesión interactiva. Los casos b) y d) no pueden suceder a menos que exista un error semántico en las descripciones: una polilínea cuyo primer y último punto son diferentes no puede ser un polígono (inciso b); y no hay interacción gráfica sobre un polígono que provoque que el primer y último punto llegen a ser diferentes (inciso d).

En este ejemplo podemos observar la propagación del elemento error. En el inciso c) encontramos que si la polilínea que forma al polígono es un elemento error entonces el polígono también es un elemento error. La evaluación de las expresiones de tipo polígono es una de las evaluaciones más fáciles porque depende casi escencialmente de la evaluación de la polilínea. La evaluación (interpretación) de un selector intersección es más interesante porque en ella no solo se puede observar la propagación de elemento error. Si al evaluar las líneas que forman la intersección resulta que al menos una de ellas es un elemento de tipo linea_e, entonces el objeto resultante es un punto de tipo punto_e. También podemos obtener objetos de tipo error si ambas líneas son elementos normales pero no se intersectan. Es decir, existen otras variantes de la regla eval que exhiben propagación de elemento error y detección de situaciones gráficas que producen a los elementos error. Pasemos ahora a describir el contenido y las funciones del resto de los programas del sistema S_2RS.

El programa ***'intrp_aux'***.

intrp_aux es una colección de las rutinas auxiliares que realizan operaciones matemáticas y lógicas requeridas para determinar el valor de la interpretación de las expresiones. Entre estas rutinas encontramos a *polycondit* que es la que verifica la condición de polígono en una polilínea. También están en esta colección las rutinas que encuentran el punto de intersección de una línea y todas las demás requeridas por el programa ***g_intrp***.

El programa ***'geom_rut'***.

Existen rutinas que realizan operaciones matemáticas o lógicas que son usadas por diferentes rutinas del sistema. Estas han sido agrupadas bajo el nombre de ***geom_rut***. Un ejemplo de rutina de este grupo es la que se llama *determinant* cuyos resultado ayudan a determina qué tipo de intersección es la que se forma entre dos líneas. Esta rutina es usada por las rutinas auxiliares de las variantes de *eval* para *intersección, unión_e, angulo_entre, paralelas* y *colineales.*

Hasta aquí quedan definidos los grupos de rutinas que intervienen en el proceso de interpretación de expresiones. La siguiente sección muestra como interactúa el

intérprete con la interface.

3.4.4 Comunicación entre la interface y el intérprete del lenguaje G.

El problema de cómo comunicar al proceso de prolog (modulo de satisfacción de restricciones) con la interface de usuario es resuelto por Tcl/Tk. El sistema Tcl/Tk contiene entre su grupo de comandos algunos que facilitan la comunicación entre su intérprete y el de BinProlog por lo que la comunicación entre BinProlog y Tcl/Tk resultó transparente en el desarrollo de S_2RS. El mecanismo de comunicació entre BinProlog y Tcl/Tk se muestra en diagramas de bloques en la figura 3.14.

Los bloques rectangulares de la figura representan programas. Los del lado derecho son programas en Tcl y los del lado izquierdo son programas en prolog. Los bloques circulares son los intérpretes BinProlog y Tcl/Tk.

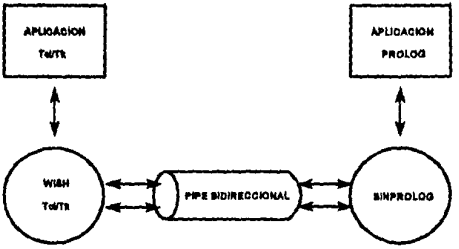


Figura 3.14.-Diagrama de bloques que muestra el proceso de comunicación entre el programa de interface y el sistema de satisfacción de restricciones en prolog.

La comunicación entre el proceso de prolog y el de Tcl/Tk es hecha a través de un ""pipe" de UNIX bidireccional. El pipe puede ser visto como dos archivos temporales uno de los cuales sirve para que el proceso de Tcl/Tk escriba y el proceso de prolog lea lo que Tcl le envía; el otro sirve para que el proceso de prolog escriba y el de Tcl/Tk lea lo que prolog le envía. En realidad un pipe es un tipo especial de redireccionamiento de la entrada y salida que permite conectar dos procesos de Unix. El pipe bidireccional permite redireccionar la salida estándar de un proceso como la

entrada estándar de otro y viceversa.

A grandes rasgos el proceso de comunicación es el siguiente: Comienza la interpretación del programa de interface (el programa en Tcl/Tk). Una de las últimas líneas que interpreta es el comando *start_prolog* cuya función es la de iniciar un proceso de BinProlog. Los comandos de llamadas a prolog son interpretados por wish, enviados a través del pipe y ejecutados por BinProlog. Cuando la aplicación en prolog hace una llamada a Tcl/Tk, BinProlog interpreta el comando que es enviado por el pipe y recibido por wish que lo ejecuta.

Existen varios eventos que ocurren en la interface para S_2RS en los que se requiere la comunicación con Prolog. Cuando una carga un dibujo en la pantalla (botón Open de la opción File de la barra de menús) se llama a una rutina de Prolog que carga archivos como su base de conocimientos. Cada línea de este archivo cargado contiene la expresión que se refiere a un objeto gráfico, el archivo es cargado como una lista que va a ser manipulada durante la sesión interactiva. Una vez cargado el archivo, prolog interpreta cada uno de los elementos de la lista el resultado de la interpretación es una expresión que puede ser dibujada, entonces prolog hace una llamada a una rutina de la interface para que dibuje en la pantalla a dicho objeto. Este proceso se sigue para todos los elementos de la lista.

Otro evento que provoca una llamada a Prolog se presenta cuando se escoge un objeto para ser trasladado. Cuando la interface detecta este evento, hace una llamada a Prolog enviando las coordenadas en la pantalla en donde se hizo la selección y el tipo de objeto que se quiere seleccionar. De acuerdo a esto prolog busca a cual de los objetos del tipo seleccionado corresponde la coordenada escogida. Por ejemplo, si se quería seleccionar un punto, el programa en Prolog busca a cual de los puntos de la base de conocimientos corresponde la coordenada recibida. Prolog llama a Tcl para guardar en una variable global el nombre del objeto seleccionado (p.ej. *punto1*).

Tcl llama nuevamente a prolog cuando en la sesión interactiva un usuario termina de hacer el *drag* de desplazamiento de un objeto. En esta ocasión envía el nombre del objeto escogido y la coordenada donde se liberó el ratón. En Prolog se actualiza la lista de la base de conocimientos de acuerdo a la nueva coordenada, se pide a Tcl que se limpie la pantalla y se llama al procedimiento de prolog que interpreta y envia a dibujar en Tcl. cada elemento de la lista de la base de conocimientos. La rutina en Prolog que detecta a que objeto corresponden las coordenadas obtenidas en la pantalla y la rutina encargada de hacer la operación de traslación a nivel de la base de conocimientos se encuentran en el último conjunto de reglas de Prolog que faltaba mencionar: el programa *g_modif.* En este conjunto de

rutinas tambien se encuentra la rutina que interpreta cada una de las expresiones de la base de conocimientos y las envía a dibujar a Tcl/Tk. Hasta aquí el proceso de comunicación entre la interface programada en Tcl/Tk y el resolvedor de restricciones programado en BinProlog. Con esto se da por terminada la presentación de la investigación y trabajo elaborado con el fín de producir una tesis de licenciatura, resta por presentar las conclusiones del trabajo.

Conclusiones.

Las restricciones gráficas constituyen el medio a través del cual se apoya una tarea de modelado geométrico. El hecho de que el resultado de la modelación sea satisfactorio depende de que las restricciones sean satisfechas de acuerdo a las espectativas del diseñador (usuario del sistema de modelado geométrico). Las técnicas de satisfacción ecuacionales y numéricas expuestas en este trabajo tienen la característa de convertir al problema en ecuaciones que lo describen; pero la solución de una intención humana es resuelta por soluciones numéricas y no infiriendo la intención del usuario, lo cual puede arrojar resultados inesperados. La exposición de las técnicas numéricas en esta tesis sirve para dar al lector una idea clara de la tradición del problema y de la franca diferencia entre éstas y la manipulación simbólica que hace el sistema prototipo S_2RS.

La manipulación simbólica hace posible referir a objetos gráficos y sus propiedades, de manera que es posible hacer modificaciones de traslación sobre los objetos en donde, por un lado, el diseñador piensa en función de un objeto gráfico y por otro, el sistema responde razonando sobre objetos también y no sobre ecuaciones. Este tipo de soluciones se dan gracias al lenguaje subyacente en el sistema S_2RS: el lenguaje G, desarrollado para este fin. El nivel de estructura de los objetos en el lenguaje G es el de los objetos geométricos básicos como el punto, la línea, la plilínea, el círculo, y el polígono. El hecho de que el nivel de estructura de los objetos no sea menor, por ejemplo, que no incluyera objetos como polilínea o polígonos, evita que se tengan que añadir restricciones al dibujo para poder contar con objetos gráficos cuyo comportamiento sea aquel de las polilíneas o los polígonos. Esto indica que

cuando se hace alguna modificación que afecte a los objetos así restringidos, se tiene que recurrir a un algoritmo para satisfacer las restricciones; es decir, existen restricciones que están resueltas intrínsecamente en la estructura de los objetos. Por lo tanto se concluye que el nivel de abstracción es un factor que influye considerablemente en el proceso de satisfacción de restricciones.

Existen expresiones en el lenguaje que describen relaciones pueden formar parte de la descripción de otro objeto gracias a las reglas de interpretación del lenguaje G (capítulo 2). Tales desccripciones representan de manera implícita restricciónes. El resultado del proceso de interpretación de este tipo de expresiones es un objeto geométrico. Lo anterior significa que no es necesario un algoritmo de solución que resuelva este tipo de restricciones. Dado lo cual podemos concluir que la expresividad en el lenguaje es un factor que influye considerablemente en el proceso de satisfacción de restricciones.

Sumarizando, dadas las pruebas efectuadas experimentalmente con el sistema S_2RS podemos concluir que la manipulación simbólica es una alternativa eficiente para resolver restricciones gráficas. Que el nivel de abstracción en el que se conceptualizan los problemas de diseño y la expresividad en el lenguaje de respresentación son factores que influyen considerablemente en el proceso de satisfacción de restricciones.

El sistema S_2RS es un prototipo de sistema para satisfacción de restricciones por referencia simbólica y cumple con los objetivos planteados en su creación. El sistema puede ser extendido para resolver restricciones cuya solución no pueda hacerse por referencia. La extensión puede hacerse añadiendo un módulo cuya entrada sean las restricciones no resueltas después del proceso de interpretación y cuya salida sea una solución a dichas restricciones. Se sugiere que la solución dada a este tipo de restricciones sea también por manipulación simbólica para aprovechar las ventajas que el lenguaje G presenta.

El S_2RS también puede ser adecuado a las necesidades del diseñador. Si el diseñador requiere de nuevas estructuras geométricas, éstas pueden ser añadidas en el lenguaje G introduciendo el tipo de dato, constructor y el valor de la interpretación de un objeto deseado en el programa g_intrp.pl del sistema. En este sentido es muy sencillo agregar tipos de datos abstractos (nuevos objetos) al sistema. Basta que el programador tenga conocimientos básicos del lenguaje de programación Prolog y echar un vistazo al programa g_intrp.pl para agragar el algortimo de interpretación del nuevo tipo de dato. En este aspecto es necesario verificar de que manera pueden interactuar los tipos de datos existentes con el nuevo para integrar todas las posibilidades en la interpretación del nuevo objeto.

El sistema también puede ajustarse para la aplicación a la que se quiera enfocar la satisfacción de restricciones. Por ejemplo, si se quiere usar S_2RS para resolver restricciones en la creación de interfases de usuario se pueden crear nuevas operaciones de manipulación de objetos como redimensionamiento de pantallas o agrupación de varios objetos gráficos en uno nuevo, etc.

Bibliografía

[Alpert 93] Alpert, S.R., *Graceful Interactions with Graphical Constraints*, IEEE Computer Graphics and Applications, Vol 13(2), pp 82-91, 1993.

[Bjorn 90] Bjorn, N.F.B., Maloney, J., Borning, A. *An Incremental Constraint Solver*, Comunications of ACM vol 33(1), pp. 54-63, 1990.

[Borning 81] Borning, A., *The programming Language Aspects or ThingLab, A constraint-oriented Simulation Laboratory*, ACM transactions on Programming Languages and Systems, 3(4), pp. 353-387, 1981.

[Borning 86] Borning, A., Duisberg, R. *Constraint-Based Tools for Building User Interfaces*, ACM Transactions on Graphics, Vol 5(4), pp. 345-374,1986.

[Borning 86a] Borning, *Graphically Defining New Building Blocks in ThingLab*, Human Computer Interaction, Vol2, pp. 269-295, 1986.

[Bratko 90] Bratko, Ivan, *Prolog, Programming for Artificial Inteligence*, Segunda Edición, Addison-Wesley, Gran Bretaña, 1990.

[Clocksin 81] Clocksin, W.F. y Mellish, C.S., *Programming in Prolog*, Springer-Verlag, New York 1981.

[Darses 90] Darses, Françoise, *Constraints in Design: Towards a Methodology of Psycological Analisys Bases on A.I. Formalism*, Human Computer Interaction, INTERACT'90, 1990.

[Fanya 86] Fanya S. Montalvo, *Diagramatic Understanding: Associating Symbolic Descriptions with Images*, IEEE 1986 Workshop on Visual Languages, IEEE Computer Society Press, pp. 4-11, 1986.

[Goguen 78] Goguen, J. A, Thatcher, J. W. y Wagner, E. G., *An Initial Algebra Aproach to the Specification, Correctness and Implementation of Abstract Data Types*, Current Trends in Programming Methodology, Prentice-Hall, pp. 80-149, 1978.

[Gosling 83] Gosling, J. *Algebraic Constraints*, CMU Reporte Técnico CS-83-132, Tesis Doctoral, Universidad Carnegie-Mellon, Pittsburg, 1983.

[Kolman 84] Kolman, B. y Busby, R., Discrete Mathematical Structures for Computer Science, Prentice-Hall, E.U. 1984.

[Konopa 84] Konopasek, M. y Jayaraman, S. *The TKISolver Book*, Osborne-McGraw-Hill, Berkeley, Calif., 1984.

[Leler 88] Leler, W. *Constraint Programming Languages: Their Specification and Generation,* Addison-Wesley, 1988.

[Liskov 74] Liskov, B. y Zilles, S., *Programming with Abstract Data Types*, ACM, SIGPLAN Notices 9, 1974.

[Mallgren 81] Mallgren William Roberts, *Formal Specification of Interactive Graphics Programming Languages*, Universidad de Washington, Disertación Doctoral, 1981.

[Maloney 89] Maloney, J. H., Borning, A., Freeman-Benson, B. N., *Constraint Technology for User-Interface Construction in ThingLab II*, OOPSLA '89 Proceedings, 1989.

[Montalvo 86] Montalvo, F., *Diagram Understanding: Associating Symbolic Descriptions with Images*, 1986 IEEE Computer Society Workshop on Visual Languages, IEEE Computer Society Press, Dallas, 1986.

[Moreno 93] Moreno, M., *Programación Matemática en Redes No Dirigidas en el Espacio Euclideano Aplicada al Diseño Asistido por Computadora*, Tesis de Lic., Escuela de Matemáticas, Universidad Juárez del Estado de Durango, México, 1983.

[Nelson 84] Nelson, G. *How to Use Juno*, Computer Science Laboratory, Xerox PARC, Palo Alto, California, Enero de 1984.

[Ouster 93] Ousterhout, J.K., *Tcl and the Tk Toolkit*, Universidad de California, Addison-Wesley, 1993.

[Pineda 89] Pineda, L.A., *GRAFLOG: A Theory of Semantics for Graphics with Applications to Human-Computer Interaction and CAD Systems*, Ph. D. Thesis, University of Edimburgh, 1989.

[Pineda 92] Pineda, L.A., *Reference, Synthesis and Constraint Satisfaction*, Computer Graphics Forum, Vol 11(3), Eurographics '92 Conference Issue, pp. c333-c334, 1992.

[Pineda 92a] Pineda L.A., y John R. Lee, *Logical Representations for Drafting and CAD Systems*. University Edcaad, University of Edimburgh. Unpublished, 1992.

[Rumelhart 83] Rumelhart E. David y Norman Donald A., *Representation in Memory CHIP*, Technical Report (No. 116), Universidad de California, pp. 15-38, 1983.

[Santana 87] Santana, J., *La Manipulación de un Modelo Geométrico Bidimensional para el Diseño de Torres de Transmisión*. Tesis de Maestría, ITESM, México, 1987

[Schwarz 89] Schwarz, H.R., *Numérical Analysis a Comprenhensive Introduction*, John Wiley & Sons, 1989

[Sassure 74] Sassure, F., *Course in General Linguistics*, Peter Owen Limited, London, 1974.

[Suther 63] Sutherland, I.E., *SKETCHPAD: A Man-Machine Graphical Communication System*, Proc. AFIPS Spring Joint Computer Conference, Vol. 23, pp. 329-346, 1963

[Tarau 94] Tarau Paul, *BinProlog 2.20 User Guide*, Universidad de Moncton, Canadá,1994.

[VanWyk 80] Van Wyk, C. J., *A Language for Typesetting Graphics*, Tesis Doctoral, Universidad de Stanford, E.U., 1980.

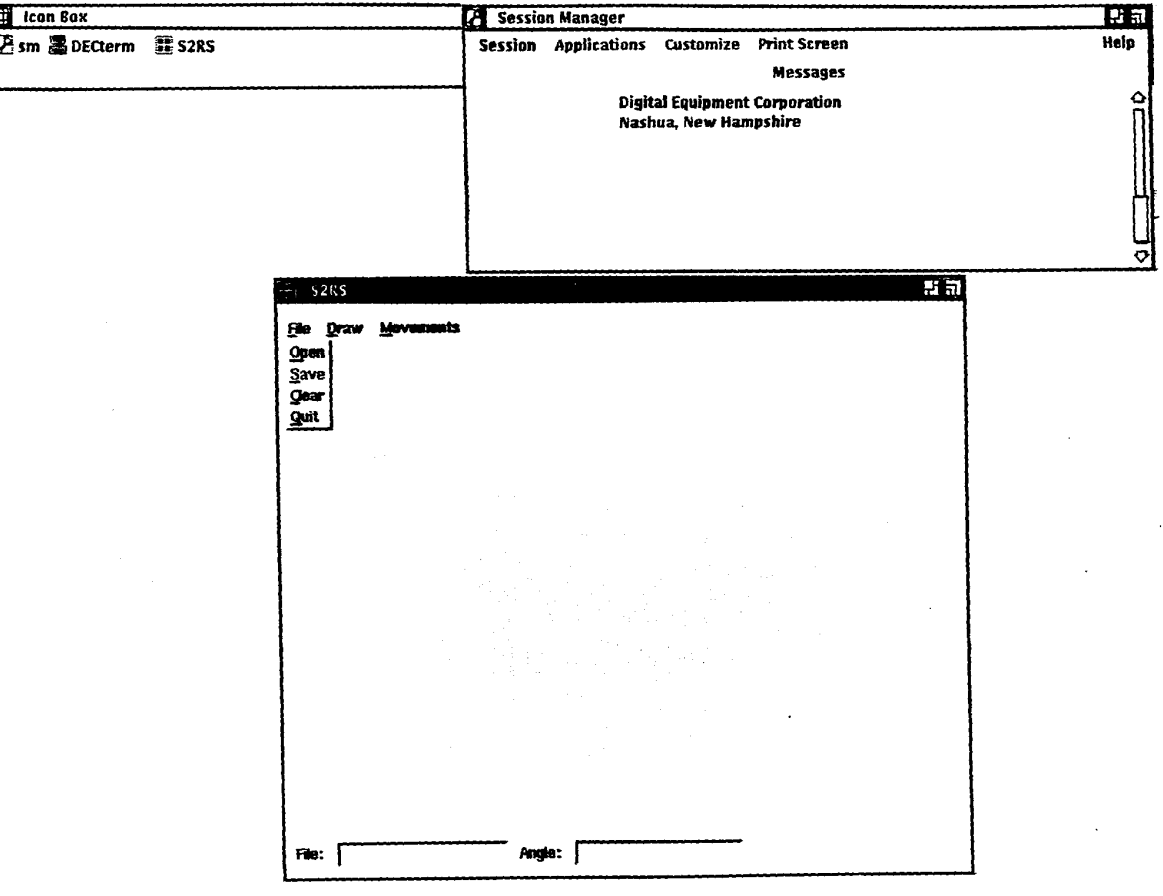
Apéndice

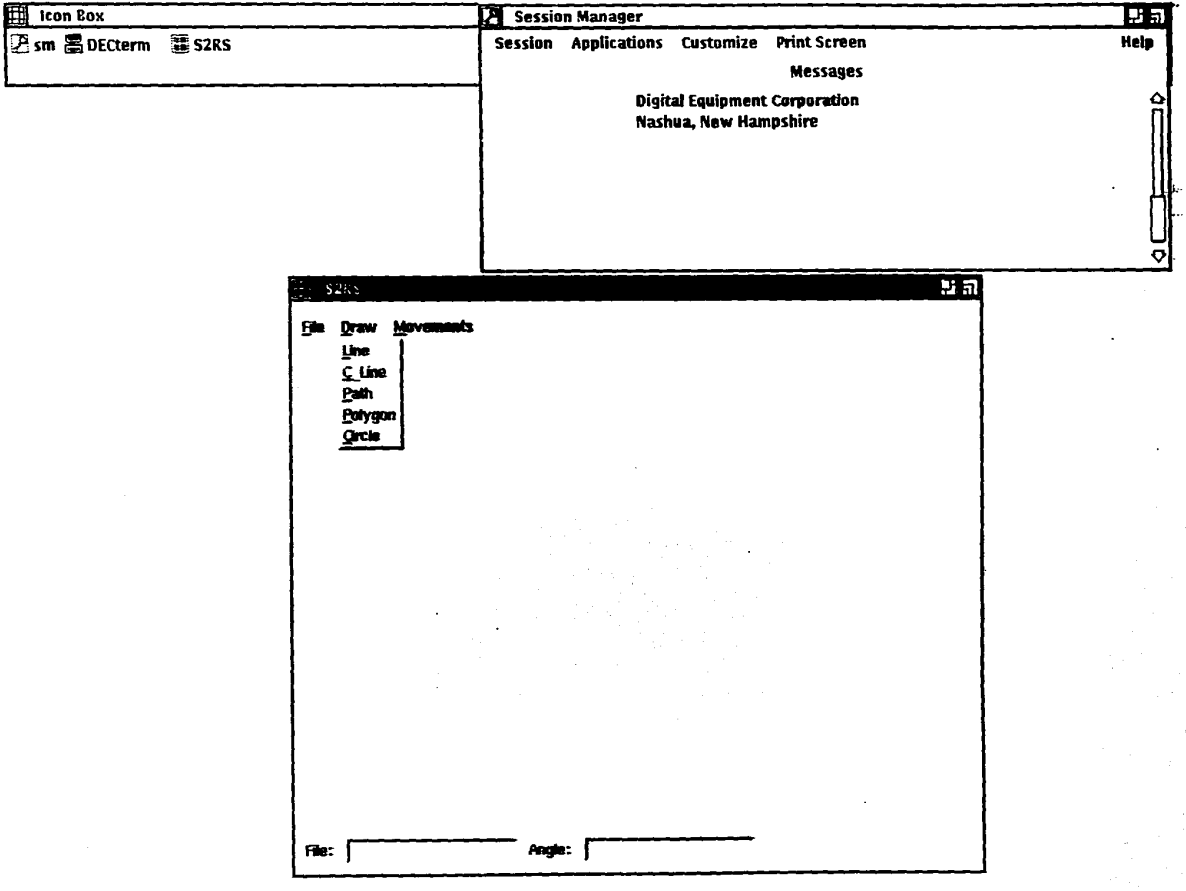
Las hojas que se muestran a continuación presentan pantallas impresas del total de la pantalla de la estación de trabajo en la que se desarrolló el sistema S_2RS. En estas pantallas podemos observar al sistema S_2RS corriendo.

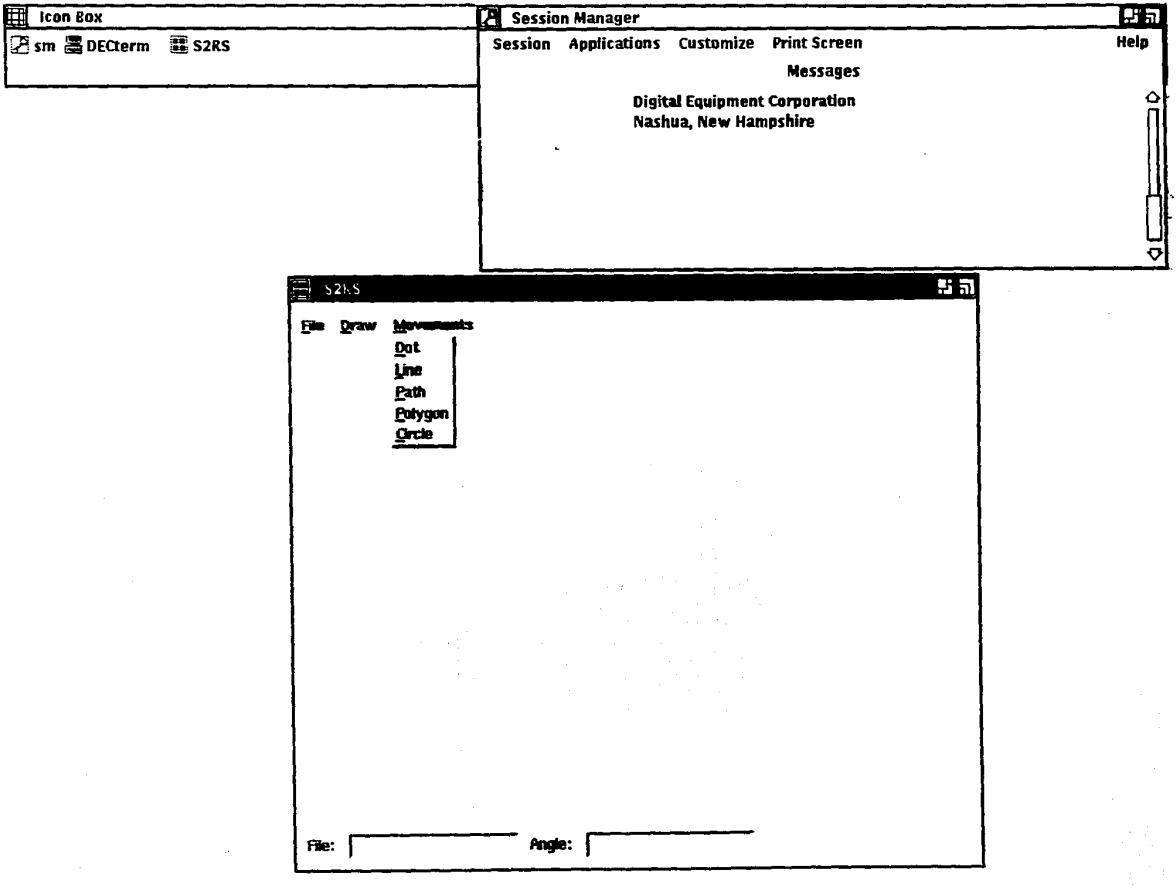
S_2RS

Pantallas con las opciones de los menús del sistema S_2RS.

Las siguientes tres páginas muestran la impresión de pantallas completas de la estación de trabajo en la que corre el sistema S_2RS. En éstas se puede apreciar a la interfaz gráfica del sistema desplegando las opciones de los menús de la barra de menús. En orden de presentación se tienen a las opciones del menú 'File', las opciones del menú 'Draw' y las opciones del menú 'Movements'.


Icon Box
sm
DECterm
S2RS
Session Manager
Session
Applications
Customize
Print Screen
Help
Messages
Digital Equipment Corporation
Nashua, New Hampshire
S2RS
File
Draw
Movements
Open
Save
Clear
Quit
File:
Angle:


Icon Box
sm
DECterm
S2RS
Session Manager
Session
Applications
Customize
Print Screen
Help
Messages
Digital Equipment Corporation
Nashua, New Hampshire
S2RS
File
Draw
Movements
Line
C_Line
Path
Polygon
Circle
File:
Angle:

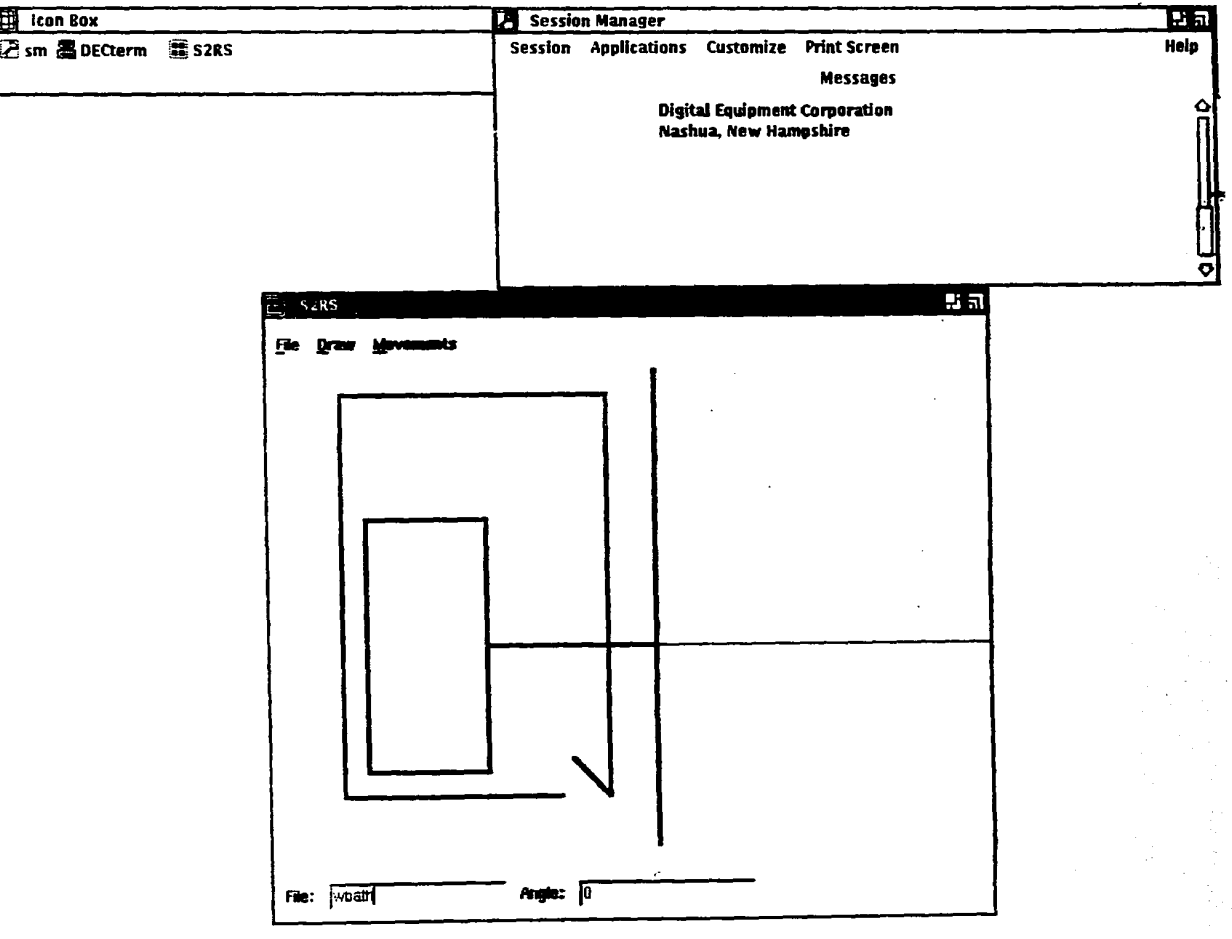

Icon Box
sm
DECterm
S2RS
Session Manager
Session
Applications
Customize
Print Screen
Help
Messages
Digital Equipment Corporation
Nashua, New Hampshire
S2RS
File
Draw
Movements
Dot
Line
Path
Polygon
Circle
File:
Angle:

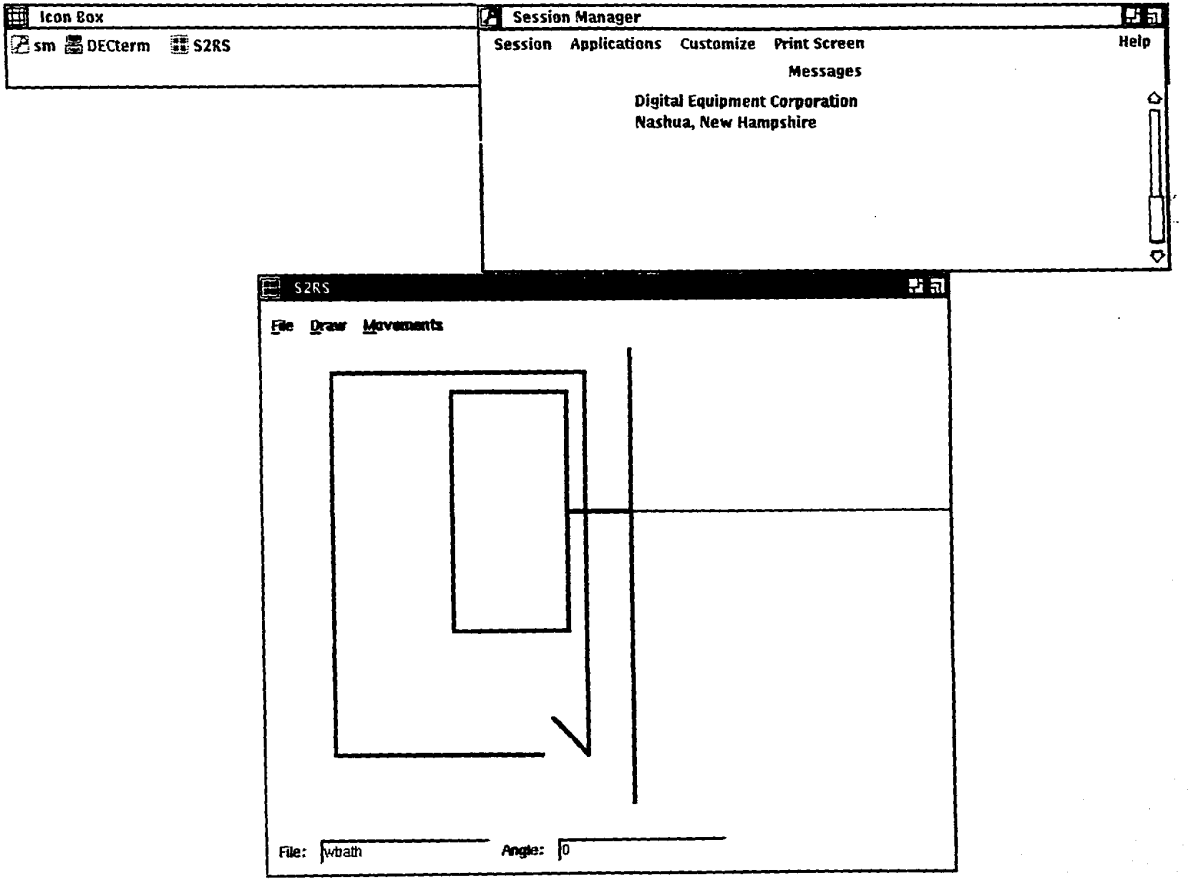
S_2RS

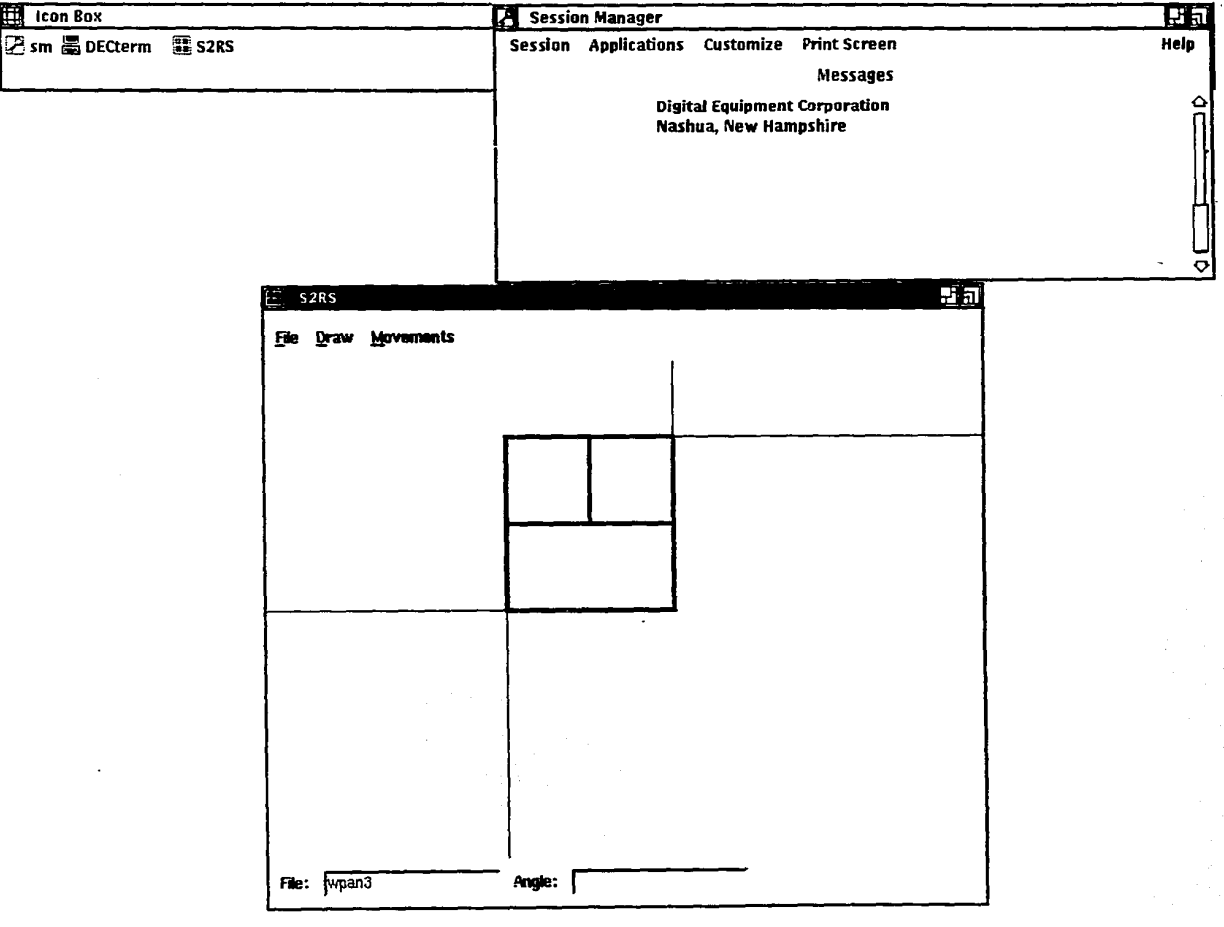
Pantallas con ejemplos corridos en S_2RS.

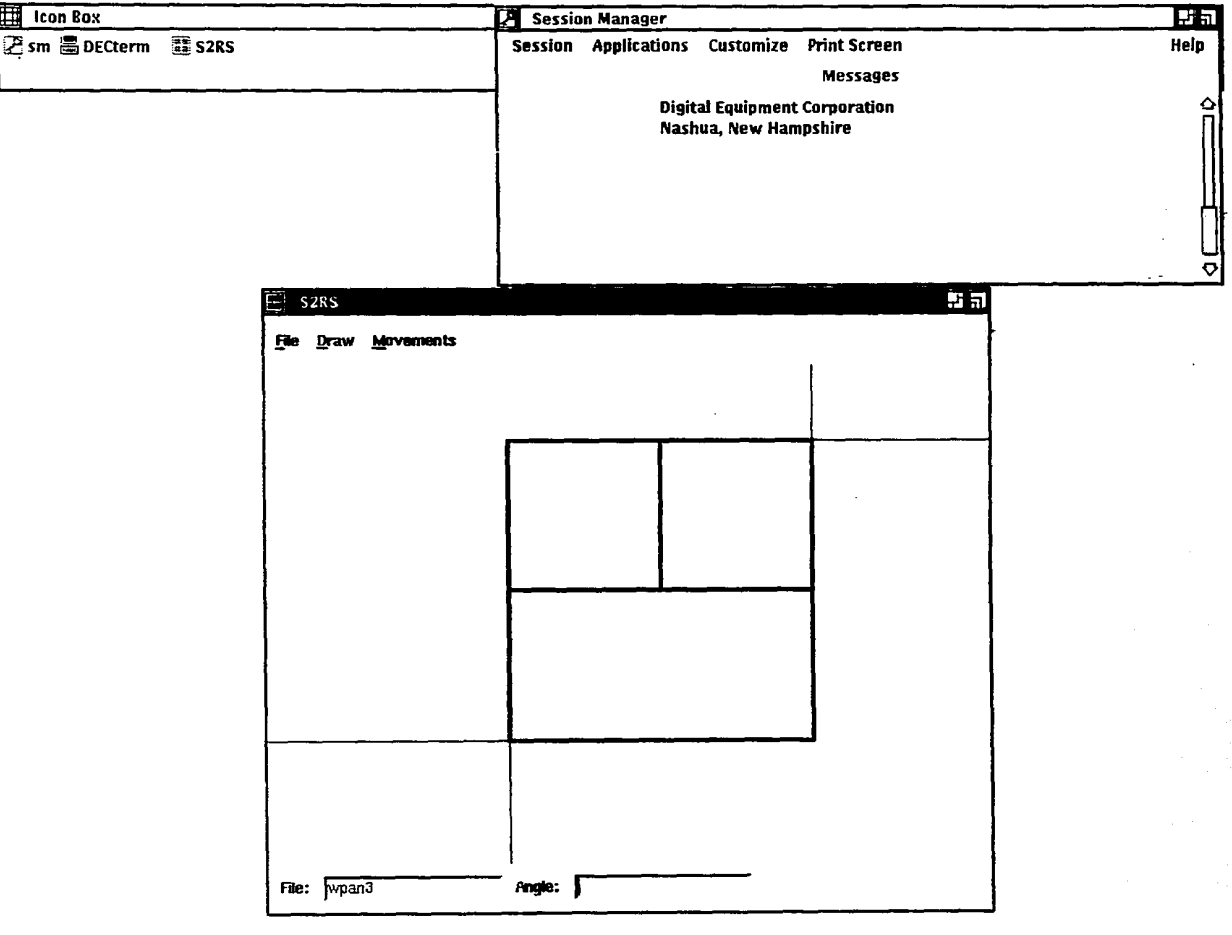
En las siguientes hojas se presentan pantallas del sistema S_2RS corriendo con los ejemplos expuestos en el contenido de este documento. En orden de presentación se tienen:

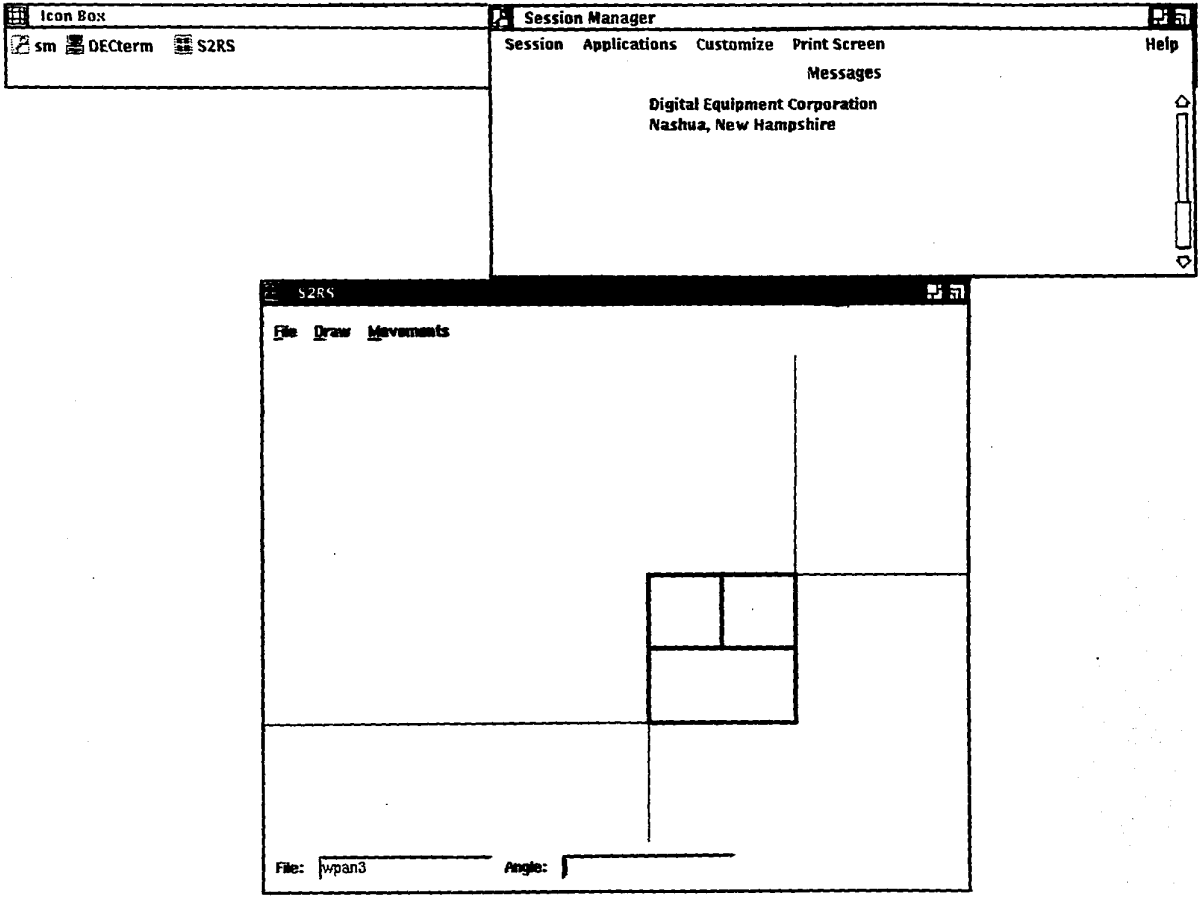
Estado original del ejemplo del cuarto de baño.
Después del la traslación de la tina del cuarto de baño.
Estado original de la ventana de tres páneles.
Redimensionamiento de la ventana de tres páneles (esquina inferior derecha).
Redimensionamiento de la ventana de tres páneles (esquina superior izquierda).
Estado original de ejemplo de los cuadriláteros internos.
Traslación del punto inferior derecho del ejemplo de los cuadriláteros internos.


Icon Box
sm
DECterm
S2RS
Session Manager
Session
Applications
Customize
Print Screen
Help
Messages
Digital Equipment Corporation
Nashua, New Hampshire
S2RS
File
Draw
Movements
File:
Angle:
0


Icon Box
sm
DECterm
S2RS
Session Manager
Session
Applications
Customize
Print Screen
Help
Messages
Digital Equipment Corporation
Nashua, New Hampshire
S2RS
File
Draw
Movements
File:
wbath
Angle:
0


Icon Box
sm
DECterm
S2RS
Session Manager
Session
Applications
Customize
Print Screen
Help
Messages
Digital Equipment Corporation
Nashua, New Hampshire
S2RS
File
Draw
Movements
File:
wpan3
Angle:


Icon Box
sm
DECterm
S2RS
Session Manager
Session
Applications
Customize
Print Screen
Help
Messages
Digital Equipment Corporation
Nashua, New Hampshire
S2RS
File
Draw
Movements
File:
wpan3
Angle:


Icon Box
sm
DECterm
S2RS
Session Manager
Session
Applications
Customize
Print Screen
Help
Messages
Digital Equipment Corporation
Nashua, New Hampshire
S2RS
File
Draw
Movements
File:
wpan3
Angle:

Icon Box

sm DECterm S2RS

Session Manager

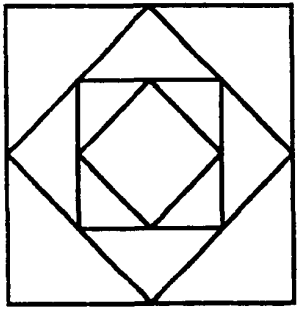
Session Applications Customize Print Screen Help

Messages

Digital Equipment Corporation
Nashua, New Hampshire

S2RS

File Draw Movements



File: wcuad3 Angle:

Icon Box

sm DECterm S2RS

Session Manager

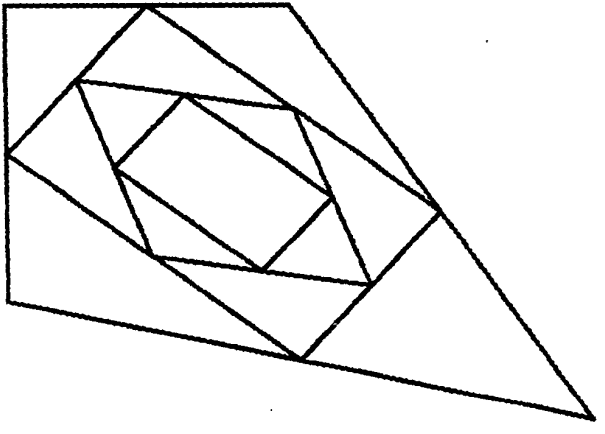
Session Applications Customize Print Screen Help

Messages

Digital Equipment Corporation
Nashua, New Hampshire

S2RS

File Draw Movements



File: wcuad3 Angle: