CM

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS

DEL

INSTITUTO POLITECNICO NACIONAL

DEPARTAMENTO DE INGENIERIA ELECTRICA

SECCION DE COMPUTACION

" METODO RM PARA LA DEMOSTRACION AUTOMATICA DE TEOREMAS "

Tesis que presenta el Lic. en Computación Guillermo De Ita Luna para obtener el grado de **MAESTRO EN CIENCIAS** en la especialidad de Ingenieria Eléctrica. Trabajo dirigido por el Dr. Guillermo Morales Luna.

Becario del CONACYT

México D.F., Abril de 1989

XM

A MIS PADRES

Sr. José De Ita Pérez

Sra. Judith Luna Soto

Al pensamiento tenaz, lucha constante, ejemplo invaluable de fe y trabajo. Con respeto y amor a quien dirigió mis primeros pensamientos y fortaleció mi espíritu. A ti madre.

A MIS HERMANOS

Compañeros fraternales que siempre me han apoyado. Con cariño a Ustedes que han sido mis grandes amigos.

Expreso mi agradecimiento al asesor: Dr. Guillermo Morales Luna, por su contribución, por dirigir, alentar, apoyar y revisar este trabajo. Sus comentarios fueron una ayuda inapreciable en el desarrollo del mismo. Agradezco la ayuda brindada por los profesores: Dr. Josef Kolar y al Dr. Ernesto López que aparte de los consejos en la elaboración del trabajo hicieron una valiosa revisión de su documentación.
Igualmente agradezco la ayuda brindada por mi compañera: Rosario Varela Hernández cuya motivación y comprensión del trabajo, me motivó y auxilió para la terminación del mismo.
Extiendo mi agradecimiento a los profesores y compañeros de la sección de computación del CINVESTAV, quienes permitieron una atmósfera intelectual agradable y rica en experiencias para su desarrollo. Este trabajo fué auspiciado por la beca número 54414 del Consejo Nacional de Ciencia y Tecnología (CONACYT).

INDICE

INTRODUCCION

Una de las más grandes ambiciones en la ciencia, ha sido la de encontrar un procedimiento general de decisión para la prueba de teoremas. Aunque tal deseo data desde Leibniz (1646-1716), esto no fue formalizado sino hasta principios del siglo XX, en que David Hilbert planteó a la comunidad matemática el construir un sistema formal general, completo y consistente para realizar correctamente (de acuerdo a las reglas del sistema) toda demostración matemática.

Un procedimiento que se convirtió en fundamental fué dado por Jacques Herbrand (1930), como un método mecánico para la prueba de teoremas, aunque tal procedimiento era muy ineficiente en la práctica. Pero no fué hasta que Kurt Gödel (1936) reveló la imposibilidad de construir un sistema formal (axiomático) completo y consistente en el cual puedan demostrarse todas las verdades matemáticas. Aún con tales limitaciones la **Demostración Automática de Teoremas**, una de las ramas en la Inteligencia Artificial, se mantuvo en investigación, hasta que en la década de los 60's un creciente interés es puesto en ella, propiciado por la aplicación de la computación en las técnicas de demostración, que da como resultado la creación de diversos métodos como son: Davis y Putnam (1960), Gilmore (1960), Loveland (1962), Davis y Hinman (1964), Robinson (1965), Prawitz (1969).

En la actualidad, las técnicas utilizadas en la Demostración Automática de Teoremas, se aplican a muchas áreas, tales como:

- El análisis, síntesis y verificación de programas por computadora.
- Los sistemas deductivos de respuestas a preguntas.
- Los sistemas de planeación.
- Los sistemas expertos.
- El modelado para el razonamiento del sentido común.
- Como base para los lenguajes de programación lógica.

Y el número de aplicaciones va en aumento.

El objetivo del método desarrollado en este trabajo de tesis, es el de diseñar y construir un procedimiento general original que realice eficientemente(igual o mejor que los métodos ya conocidos) la prueba de teoremas en el cálculo de predicados.
Como es sabido, el procedimiento inicial dado por Herbrand es ineficiente en la práctica, ya que se basa en la generación de una sucesión de conjuntos $S_1 \ldots S_N \ldots$ con instancias básicas del conjunto cláusular inicial S. Tal sucesión puede ser infinita y en principio no se sabe para cual valor de N, puede probarse la inconsistencia de S_N. Con el fin de agilizar la búsqueda del S_N, Davis y Putnam adicionaron cuatro reglas que simplifican la prueba para la inconsistencia de S_N.

Con el fin de evitar la generación de tal sucesión, dos técnicas principales han sido propuestas: una es el principio de Resolución de Robinson y otra el método propuesto por Prawitz.

Sin embargo, el autor considera que estas dos últimas técnicas no son excluyentes y por el contrario, evitando las incoveniencias inherentes a cada una de ellas y aprovechando las ventajas de ambas, las ideas principales en estas técnicas fueron combinadas en un modelo basado en reglas, con un orden parecido al de las reglas en el método de Davis y Putnam. Esto forma la base del método diseñado por el autor denominado **METODO RM.**

El interés por realizar este trabajo puede remontarse a la formación tenida por el autor en la carrera de 'Licenciado en Computación' realizada en la E.C.F.M. de la Universidad Autónoma de Puebla, en particular a los cursos de Lógica Computacional cursados tanto en esta escuela como en el CINVESTAV - I.P.N., en donde además se efectuaron proyectos en los que se programaron algunos de los métodos ya conocidos en la Demostración Automática de Teoremas.

En el capítulo I de este documento de tesis, se dan los fundamentos teóricos necesarios para la comprensión de la teoría en la que se sustentan los métodos de demostración. También se explican algunos de los elementos a utilizar por el método RM.

En el capítulo II se presenta el procedimiento de Herbrand, se explica tanto el método RM como los métodos en los que se basa.

En el capítulo III se describe a detalle al método RM, se explican los diversos módulos que lo componen; las estrategias que utiliza, el algoritmo lógico y de control y la estructura de datos usada para su implementación en el lenguaje LISP.

Por último, en las conclusiones se mencionan los objetivos alcanzados, los conocimientos y experiencias adquiridas, las limitaciones encontradas al trabajo y se mencionan algunas de las posibles aplicaciones y/o extensiones a éste trabajo.

El apéndice A, guía a los futuros usuarios del programa. Indica tanto como ejecutarlo, así como la sintaxis a seguir al introducir los datos de entrada. Señala el formato y el tipo de resultados que se debe esperar. Y por último se ejemplifica una sesión con el programa.

PRELIMINARES

Los procesos del pensamiento han sido objeto de estudio de varias ciencias, una de ellas que desde sus inicios trató de formalizar los procesos intelectivos del razonamiento es la Lógica. Se dice que la capacidad de razonar es la que distingue al ser humano de otras especies, resulta paradójico entonces, el querer mecanizar 'eso' que es lo más humano que se tiene. Sin embargo, ya los griegos sabían que el razonamiento es un proceso sujeto a esquemas y que en parte está gobernado por leyes perfectamemte formulables. Aristóteles formalizó los silogismos y Euclides la geometría.

En el siglo XIX los lógicos ingleses George Boole y August De Morgan sometieron los esquemas estrictamente deductivos del razonamiento a una formalización que deja muy atrás la formalización Aristotélica. Gottlob Frege y Giuseppe Peano se dieron a la tarea de combinar el razonamiento formal con el estudio de conjuntos y números. En Gotinga, Alemania, David Hilbert elaboró formalizaciones de geometrías más estrictas que las de Euclides. Todos estos esfuerzos iban dirigidos al objetivo de aclarar que es lo que se entiende por Demostración.

El intento de hacer una codificación completa de los modos universalmente aceptados del razonamiento fue realizado por Whitehead y Russell (1913) en su obra 'Principia Mathematica'. Ellos se propusieron derivar toda la matemática de la lógica y además sin contradicciones. Aún cuando en esos tiempos no se sabía si los métodos presentados eran coherentes consigo mismos y si toda la matemática quedaba realmente englobada en los Principia Mathematica.

En este capitulo se presenta el formalismo inicial universalmente aceptado, en el cual se basan los intentos por formalizar los procesos del razonamiento, que son el Cálculo Proposicional y el Cálculo de Predicados.

1.1 CALCULO DE PROPOSICIONES

En el cálculo proposicional se trabaja con enunciados declarativos que pueden ser bien verdaderos o bien falsos, pero no ambos. Tal enunciado declarativo es llamado una proposición. Símbolos tales como: P,Q,R serán usados para denotar ciertas proposiciones que serán llamadas **átomos** o **fórmulas atómicas**.

Para formar **proposiciones compuestas** se usan los conectivos lógicos: $\neg$ (no), $\vee$ (o), $\wedge$ (y), $\rightarrow$ (si...entonces), $\equiv$ (si y solo si) y se auxilia de los paréntesis '()' para delimitar el alcance de estos conectivos.

Def. Una **Fórmula Bien Formada (FBF)** ó **Fórmula** se define recursivamente de la manera siguiente:
1.- Un átomo es una fórmula.

2.- Si G y H son fórmulas, entonces $(\neg G)$, $(\neg H)$, $(G \vee H)$, $(G \wedge H)$, $(G \rightarrow H)$, $(G \cong H)$ son fórmulas.
3.- Todas las fórmulas son aquellas generadas por aplicaciones finitas de las dos reglas anteriores.

Los valores veritativos de las fórmulas compuestas $(\neg G)$, $(\neg H)$, $(G \vee H)$, $(G \wedge H)$, $(G \rightarrow H)$, $(G \cong H)$, están relacionados a los valores veritativos de G y de H de la forma siguiente:
1.- $(\neg G)$ es verdadera si G es falsa. $(\neg G)$ es falsa en otro caso.
2.- $(G \wedge H)$ es verdadera sólo cuando G y H ambas son verdaderas, en otro caso es falsa.
3.- $(G \vee H)$ es verdadera cuando al menos una de G o H es verdadera, en otro caso es falsa.
4.- $(G \rightarrow H)$ es falsa, si G es verdadera y H es falsa, en otro caso es verdadera.
5.- $(G \cong)H$ es verdadera cuando tanto G como H tienen el mismo valor de verdad (ambas son verdaderas o ambas son falsas), en otro caso es falsa.
Se denotará al valor veritativo verdadero con 'V' y al valor veritativo falso con 'F'.

Def. Dada una fórmula G con n átomos $A_1,\ldots,A_n$ una **interpretación** de G es un asignamiento de valores veritativos a $A_1,\ldots,A_n$ en el que a todo A_i se le asigna uno de los valores V ó F, pero no ambos.

Def. Una fórmula se dice que es **verdadera bajo una interpretación** si es evaluada con V en tal interpretación. En otro caso es **falsa**.

Def. Una fórmula se dice **válida** si es verdadera bajo toda interpretación. Una fórmula es **inválida** si no es válida.

Def. Una fórmula se dice **inconsistente (insatisfacible)** si es falsa bajo cualquier interpretación. Una fórmula se dice **consistente** si no es inconsistente.

Las **estructuras deductivas** pueden también ser representadas matemáticamente. Una estructura deductiva es una representación formal de un proceso de razonamiento: se obtiene una conclusión a partir de las premisas. La formalización de las estructuras deductivas en la **teoría de la demostración** tiene como elementos principales a un sistema de fórmulas válidas construidas a partir de:

A) Una serie de formas o esquemas de fórmulas que se suponen válidas por hipótesis, llamadas **axiomas**.

B) Unas **reglas de demostración** o **reglas de inferencia** que permiten obtener nuevas fórmulas válidas a partir de los axiomas.

El sistema de axiomas y reglas de inferencia debe ser consistente. Es decir, no debe ser posible demostrar en tal sistema, a una fórmula y a su negación.

1.2 CALCULO DE PREDICADOS

Una teoría que extiende el poder de expresión del cálculo de proposiciones es el cálculo de predicados. Se define al cálculo de predicados por su sintaxis y su semántica. Para especificar su sintaxis se define un alfabeto de símbolos y la forma en que estos símbolos se combinan para formar expresiones legítimas del lenguaje.

SINTAXIS DEL CALCULO DE PREDICADOS

El alfabeto L consta de la unión de los conjuntos siguientes

a) SE: símbolos especiales , (coma), ()
b) CO: conectivos $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\cong$
c) CU: cuantificadores $\forall$(universal), $\exists$(existencial)
d) Vars: Variables $x_1, x_2, \ldots$
e) Ctes: Constantes $a_1, a_2, \ldots$
f) FUN: funciones f_i^j i,j en $\mathbb{N}$
g) REL: relaciones R_i^j i,j en $\mathbb{N}$

L_{log} = SE $\cup$ CU $\cup$ Vars $\cup$ CO -se dice ser la **parte lógica** de L
L_{sig} = Ctes $\cup$ FUN $\cup$ REL - es la parte **distintiva** o **signatura** de L
L_{log} es la parte común a todo alfabeto L de una teoría de primer orden.

Def. El conjunto de los **términos** (Term) se define recursivamente como sigue
a) Si t $\in$ Vars o t $\in$ Ctes entonces t $\in$ Term
b) Si $f_i^n \in$ FUN, $t_1, \ldots, t_n \in$ Term, entonces $f_i^n(t_1, \ldots, t_n) \in$ Term
c) Un término es lo obtenido de aplicar (a) y (b) un número finito de veces.

Def. Una **Fórmula Atómica** (FA) se define como:
$R_j^n \in$ REL (también llamados predicados), $t_1, \ldots, t_n \in$ Term entonces $R_j^n(t_1, \ldots, t_n) \in$ FA

Def. Las **Fórmulas Bien Formadas** (FBF) se construyen mediante las reglas siguientes
a) Si $\alpha \in$ FA entonces $\alpha \in$ FBF
b) Si $\alpha, \beta \in$ FBF entonces $\neg\alpha$, $\alpha \wedge \beta$, $\alpha \vee \beta$, $\alpha \rightarrow \beta$, $\alpha \cong \beta \in$ FBF
c) Si $\alpha \in$ FBF , x $\in$ Vars entonces $(\forall x\ \alpha)$, $(\exists x\ \alpha) \in$ FBF
d) Una FBF es una expresión obtenida de la aplicación finita de (a), (b) y (c).

Sea $x_1, \ldots, x_n$ una lista de variables, mediante $\phi(x_1, \ldots, x_n)$ se indicará que la fórmula ϕ tiene como algunas de sus variables libres a $x_1, \ldots, x_n$.

El cálculo de predicados sobre el alfabeto L se construye a partir de un conjunto de axiomas.

AXIOMAS

1.- $(\alpha \rightarrow (\beta \rightarrow \alpha))$

2.- $((\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow (\alpha \rightarrow \gamma))$

3.- $((\neg\alpha \rightarrow \neg\beta) \rightarrow ((\neg\alpha \rightarrow \beta) \rightarrow \alpha))$

4.- $((\forall x\ \alpha(x)) \rightarrow \alpha(t))$ donde $t \in$ Term

5.- $((\forall x\ (\alpha \rightarrow \beta(x))) \rightarrow (\alpha \rightarrow \forall x\ \beta(x)))$ siempre que x no esté en α

Def. Si una variable x en una FBF está cuantificada ($\forall x$, $\exists x$) se dice que x tiene una ocurrencia **ligada** o **cerrada**, en otro caso se dice que x tiene una ocurrencia **libre**. Una FBF en la que todas sus variables tienen ocurrencias cerradas se llama **enunciado** o **fórmula bien formada cerrada**. En otro caso se dice **abierta**.

El cálculo de predicados a tratar aquí se dice de primer orden porque permite la cuantificación sólo sobre las variables. Las reglas de inferencia que se definen para este sistema son

Modus Ponens (MP)

De ϕ

Y $\phi \rightarrow \gamma$

Se infiere γ

Generalización (GE)

De $\phi(x)$

Se infiere $\forall x\ \phi(x)$

Def. Si A es un conjunto de FBF se dice que $\phi \in$ FBF **se deduce de** A, denotandose $A \vdash \phi$, si existe una sucesión $\phi_1, \ldots, \phi_n \in$ FBF tal que

1.- $\forall i=1,\ldots,n$, ϕ_i bien es un axioma ó bien $\phi_i \in A$ ó bien ϕ_i se obtiene de un ϕ_j ó de un ϕ_j y un ϕ_k $(j,k < i)$ mediante Generalización o Modus Ponens respectivamente.

2.- $\phi_n = \phi$

Si $A = \varnothing$ y $A \vdash \phi$, se escribe $\vdash \phi$,y se dice que ϕ es un Teorema.

SEMANTICA DEL CALCULO DE PREDICADOS

Una **interpretación** de un lenguaje de primer orden L, se da a partir de un conjunto no vacío D llamado **dominio** y un asignamiento de valores tal que

1.- A cada $c \in$ Cte, se le asocia un único $c' \in D$

2.- Para todo $f^n_j \in$ FUN, se le asocia un único mapeo f'^n_j de D^na D

f'^n_j: D^n ---> D.

Tal interpretación se propaga a los términos de la manera siguiente

a) Si $t = c$ entonces $t' = c'$

b) Si $t = f^n_j(t_1,\ldots,t_n)$ entonces $t' = f'^n_j(t'_1,\ldots,t'_n)$ donde t'_i es la interpretación del término t_i para $i = 1,\ldots,n$.

Para interpretar a las fórmulas del lenguaje

1.-Se hereda la interpretación dada en el cálculo de proposiciones para los conectivos lógicos

2.-A toda relación $R^n_j \in$ REL se le asocia una función de D^n a {V,F}

3.- ϕ = ($\forall$xG(x)) es evaluada como V, si el valor de G es V para todo d $\in$ D, en otro caso ϕ es falsa.
4.- ϕ = ($\exists$xG(x)) es evaluada como V, si G es V cuando al menos para algún d $\in$ D, G(d) es V en otro caso ϕ es falsa.

Una FBF ϕ sin variables libres bien es verdadera ó bien es falsa, es decir, si ϕ es cerrada, entonces se cumple ϕ ó $\neg\phi$. Si ϕ es una FBF con variables libres, se considera como una relación en el dominio de la interpretación, la cual puede ser verdadera para algunos valores del dominio y falsa para algunos otros.

Definiciones
Una interpretación I se dice ser un **modelo M** para un conjunto A de FBF's si para todo $\phi \in$ A, ϕ es verdadera en I. Y se llama **contramodelo** si existe un $\phi \in$ A tal que ϕ es falsa en I.

$\phi \in$ FBF es **satisfacible** si existe al menos una interpretación que la verifica. Si ϕ tiene variables libres e I es una interpretación se dice que ϕ se **verifica** en I, si no existen valores en el dominio de I, tal que al sustituirse por las variables libres hagan a ϕ falsa.

$\phi \in$ FBF es **lógicamente válida** si ϕ es verdadera para toda interpretación y en cualquier dominio y se denota $\vdash \phi$.
$\phi \in$ FBF es una **contradicción** si $\neg\phi$ es lógicamente válida.

Se dice que ϕ es una **consecuencia lógica** de A si en toda interpretación que es modelo para A satisface a ϕ. Y se escribe A $\models \phi$. De acuerdo a tal definición dado que cada variable con ocurrencia libre se interpreta por un elemento del dominio su valor permanece constante en todas las ocurrencias de cada variable.
ϕ no es una consecuencia lógica de A si existe al menos una interpretación I, tal que I satisface a A pero I no satisface a ϕ. Esto no excluye que puedan existir algunas interpretaciones J que satisfacen tanto a A como a ϕ.

1.3 REPRESENTACION CLAUSULAR

FORMA PRENEX: $\phi(x_1,\ldots,x_n)$ está en forma prenex si $\phi(x_1,\ldots,x_n) = Q_1y_1,\ldots,Q_my_m\beta(x_1,\ldots,x_n,y_1,\ldots,y_m)$ donde β es una FBF libre de cuantificadores (llamada **matriz**) y $Q_1,\ldots,Q_m$ es una sucesión de cuantificadores $\forall$ ó $\exists$ (llamada **prefijo**). Es decir, ϕ está en forma prenex si tiene a sus cuantificadores al principio.

$\phi(\mathbb{X})$ está en **forma clausular** si tiene la forma $\phi(\mathbb{X}) = \mathbb{Q}\forall\beta(\mathbb{X},\mathbb{Y})$ con β en forma normal conjuntiva y en $\mathbb{Q}$ sólo aparecen cuantificadores $\forall$. a $\phi(\mathbb{X})$ se le llama cláusula.

Teorema. Para cualquier FBF $\phi(x_1,\ldots.x_n)$ existe una forma clausular equivalente.

Se da un procedimiento que traduce toda FBF ϕ a su forma clausular y se ilustra la demostración con la FBF
($\forall$x (P(x) $\rightarrow$ (($\forall$y (P(y) $\rightarrow$ P(f(x,y)))) $\wedge$ $\neg$($\forall$y(Q(x,y) $\rightarrow$ P(y))))))

PASO 1. Eli implicación. **Cambiar todas** las
FBF's del
(∀x(¬P(x) ∨)))) ∧ ¬(∀y(¬Q(x,y) ∨ P(y))))))

PASO 2. Red símbolos de **negación.Que afecten**
sólo a fórm
(∀x(¬P(x) ∨)))) ∧ (∃y(Q(x,y) ∧ ¬P(y))))))

PASO 3. Est es. El objeto **de esto es que cada**
cuantificad ariable.
(∀x(¬P(x) ∨ y)))) ∧ (∃z(Q(x,z) ∧ ¬P(z))))))
ya que pued las apariciones **de una variable**
por alguna iar el valor **de verdad de la FBF.**

PASO 4.Elimi xistenciales. **Si el cuantificador**
existencial nce de uno universal, **la variable**
cuantificad uede depender **de la cuantificada**
universalmc es explícitamente definida por
alguna func ción **de Skolem (en honor a Thorolf**
Skolem), si á dentro de uno **universal, enton-**
ces la var or una **constante de Skolem.** La
introducci lem hace que **la fórmula pierda**
generalidad la consideración **de predicados**
indetermin
Entonces su cias de z dentro del alcance del
cuantificad Skolem g(x)
(∀x(¬P(x) ∨))) ∧ (Q(x,g(x)) ∧ ¬P(g(x))))))

PASO 5. Con Puesto que **los únicos cuantifi-**
cadores qu sales y todos tienen diferentes
variables, frente de la FBF.
(∀x (∀y ((x,y))) ∧ (Q(x,g(x)) ∧
¬P(g(x))))))
Prefijo (_____)

lz

PASO 6. Pon rma **normal conjuntiva.** Escribir a
la matriz c n conjunto finito de disyunciones
de fórmulas
(∀x(∀y ((¬))) ∧ (¬P(x) ∨ Q(x,g(x))) ∧
(¬

PASO 7. Eli tificadores **universales.** Suponer
por conven ables de la matriz están univer-
salmente cu
((¬P(x) ∨ (¬P(x) ∨ Q(x,g(x))) ∧ (¬P(x) ∨
¬P(g(x)))

PASO 8. Eli Se sustituyen **a los símbolos ∧** de
conjunción
¬P(x) ∨ ¬P ∨ Q(x,g(x)), ¬P(x) ∨ ¬P(g(x))

PASO 9. Ren símbolos de **variables pueden ser**
cambiados c mbolo no aparezca en más de una
cláusula. S valor de verdad de la FBF.
¬P(x) ∨ ¬P ∨ Q(z,g(z)), ¬P(u) ∨ ¬P(g(u))

1.4 DEMOSTRACION POR REFUTACION

Una técnica para demostrar que $A \models \phi$, es la de **reducción al absurdo** o de refutación. Esta técnica es la más utilizada en los Demostradores Automáticos de Teoremas. Tal técnica trabaja de la manera siguiente: Se supone cierto A y a $\neg\phi$, y si esto lleva a una inconsistencia, es decir, si $A \cup \{\neg\phi\}$ es una contradicción entonces se concluye que ϕ se deduce de A. Y queda demostrado que $A \models \phi$.

1.5 PROCEDIMIENTO DE UNIFICACION

Al probar teoremas que involucran fórmulas cuantificadas, es muchas veces necesario el **empatar** ciertas subexpresiones. Por ejemplo: el aplicar la combinación de Modus Ponens con la Generalización para producir P(A) de las FBF's

1.- $(\forall x)(Q(x) \rightarrow P(x))$

2.- $Q(A)$

es necesario utilizar la sustitución $\langle x/A \rangle$ que hace las expresiones Q(x) y Q(A) idénticas. Encontrar sustituciones de términos por variables para hacer expresiones idénticas es llamado **Unificación**.

Una sustitución es un conjunto de pares del tipo $\langle v_K/t_K \rangle$, en donde v_K es una variable, t_K es un término y en t_K no aparece la variable v_K, esto último es para no restringir la generalidad de la interpretación.

Por ejemplo, para la FBF, P(x,f(y),B) a la cual se le aplican los siguientes conjuntos de sustituciones:

s1 = $\{\langle x/z \rangle\}$ - quedando - P(z,f(y),B)

s2 = $\{\langle y/A \rangle\}$ - quedando - P(x,F(A),B)

s3 = $\{\langle x/g(z) \rangle, \langle y/A \rangle\}$ - quedando - P(g(z),F(A),B)

s4 = $\{\langle x/C \rangle, \langle y/A \rangle\}$ - quedando - P(C,f(A),B)

El resultado del primer ejemplo es llamado una **variante alfabética** de la FBF original, puesto que solo se han renombrado las variables y el valor de la FBF resultante es equivalente a la FBF original. El del último ejemplo es llamado **instancia básica**, puesto que ninguno de los términos en la FBF contiene variables.

Dada una FBF A y una sustitución s, se denomina As a la fórmula resultante de sustituir cada ocurrencia de v_K que aparece en A por el término t_K, donde la sustitución $\langle v_K/t_K \rangle$ está en s.

La composición de dos sutituciones s1 y s2 es denotada por s1s2, la cual es la sustitución obtenida por la aplicación de s2 a los términos (2da. componente de las parejas ordenadas) de s1 y agregando algunos pares ordenados de s2 que no contienen a las variables (1era. componente de las parejas ordenadas) de s1.
Por ejemplo:

s1s2 = $\{\langle z/g(x,y) \rangle\}\ \{\langle x/A \rangle, \langle y/B \rangle, \langle w/C \rangle, \langle z/D \rangle\} =$
$\{\langle z/g(A,B) \rangle, \langle x/A \rangle, \langle y/B \rangle, \langle w/C \rangle\}$

La composición de sustituciones es asociativa, es decir (s1s2)s3 = s1(s2s3). Pero en general no es conmutativa; s1s2 ≠ s2s1

Si una sustitución s es aplicada a todo miembro de un conjunto $\{E_i\}$ de expresiones, el conjunto de instancias de sustituciones es denotado por $\{E_i\}$ s. se dice que el conjunto $\{E_i\}$ es **unificable** si existe una sustitución s tal que $E_1s = E_2s = \ldots = E_ns$ y a s se le llama el unificador de $\{E_i\}$.

Por ejemplo s =$\{\langle x/A\rangle, \langle y/B\rangle\}$ unifica a $\{P(x,f(y),B), P(x,f(B),B)\}$ y da como resultado P(A,f(B),B). Sin embargo este unificador no es el más simple, puede notarse que no es necesario el susituir a x por A para alcanzar la unificación.
El unificador más general (u.m.g.) g de un conjunto $\{E_i\}$, tiene la propiedad de que para cualquier otro unificador s de $\{E_i\}$ .existe una sustitución s' tal que: $\{E_i\}$ s = $\{E_i\}$ gs'.
El u.m.g es el unificador más general ya que contiene el número mínimo de parejas, y por tal, liga al mínimo posible de variables. Aún mas, el valor de la instancia producida por u.m.g es única excepto por variantes alfabéticas.

1.6 INDECIBILIDAD DEL CALCULO DE PREDICADOS

La pregunta de si la Matemática formaba un sistema coherente y completo, fue formulada por David Hilbert ante la comunidad mundial matemática, retando a encontrar una demostración rigurosa de ello. Esta fue la meta de los Principia Mathematica, que intentaba derivar toda la matemática de la lógica, el reto de Hilbert se traducía entonces a demostrar la consistencia y completez de estos Principia Mathematica.
Pero no fue sino hasta el año de 1936 en que Kurt Gödel publicó un artículo que daba respuesta a tal reto. El trabajo de Gödel reveló no solo que había 'hoyos' irreparables en el sistema axiomático propuesto por Russell y Whitehead, sino también más en lo general, que absolutamente ningún sistema axiomático podía producir todas las verdades relativas a la matemática, ni siquiera a un subconjunto de ella, como es la teoría de Números, salvo que se tratara de un sistema no coherente. Esto derrumbaba la esperanza de demostrar la coherencia de un sistema formal completo.

El teorema de Gödel no tardó en tener en la teoría de la computación, un principio paralelo (descubierto por Alan Turing), que marcaba las limitaciones teóricas hasta en la computadora más potente que pudiera imaginarse.

El hecho de que no sea posible el construir un programa computacional (un procedimiento o un formalismo matemático), que pueda decidir correctamente cuando, para cualquier proposición dada, este es ó no un teorema en la teoría en que se está trabajando, es llamado **indecibilidad** de los procedimientos de prueba.

Sin embargo, la idea de emular (mecanizar) los procesos del razonamiento al demostrar ciertas proposiciones no es del todo inútil, ya que pueden construirse programas (procedimientos) semidecidibles, es decir, programas que encuentran la demostración de que una proposición dada es un teorema, aunque pueden quedar en un ciclo infinito cuando tal proposición no es un teorema.
Esto impide, de manera general, el asegurar cuando un programa Demostrador de Teoremas, dada una entrada, deberá parar o no.

El interés por realizar programas de Demostración Automática de Teoremas sigue manteniéndose, dado que existe una subclase de enunciados en las teorías de primer orden, que pueden ser demostrados automaticamente y que resultan ser interesantes demostraciones, aparte de que su aplicación a otras ramas de la Inteligencia Artificial ha sido fundamental.

Cabe señalar que a principios de los ochenta's la lógica constituye una base del desarrollo de las técnicas informáticas en tres lineas:
- Las lógicas de Hoare, como apoyo de las técnicas de verificación de programas.
- Los lenguajes de programación lógica, que constituyen un nuevo enfoque conceptual de la programación, capaz de formular tanto los algoritmos de la programación clásica como las bases de conocimiento para los Sistemas Expertos.
- El modelado para el razonamiento del 'sentido común' para los Sistemas Expertos, tanto en sus vertientes clásicas, lógica difusa o no monótona [CUE].

En el capitulo siguiente se abordan las bases teóricas de la Demostración Automática de Teoremas, el procedimiento de Herbrand y los siguientes Métodos de Demostración Automática de Teoremas: el Método de Davis y Putnam, el Método de Prawitz y el Método de Reducción Matricial. Este último es el construido en este trabajo de tesis.

METODO RM PARA LA DEMOSTRACION AUTOMATICA DE TEOREMAS

Una aproximación importante en la prueba mecánica de teoremas fue dado por Jacques Herbrand (1930). Herbrand desarrolló un algoritmo para hallar una interpretación que pueda probar la inconsistencia de una fórmula, aunque si la fórmula es consistente, tal interpretación puede no existir y el algoritmo deberá parar después de un número finito de pasos. El método de Herbrand es la base para varios procedimientos automáticos de demostración: Gilmore (1960); Davis y Putnam (1960); Loveland (1962); Davis y Hinman (1964); Robinson (1965); Prawitz (1969).

En este capítulo se presenta primero el procedimiento de Herbrand, después algunos de los métodos clásicos como: el de Davis y Putnam, el de Robinson, el de Prawitz y por último se expone el método RM (Reducción Matricial), que unifica y extiende algunas de las ideas de los anteriores métodos clásicos.

2.1 FUNDAMENTOS TEORICOS

Definición.

Un conjunto S de cláusulas es **inconsistente** si y solo si cualquier interpretación para S resulta ser un contramodelo.

Aún cuando es imposible el considerar a toda interpretación sobre todo dominio, existe un dominio llamado **universo de Herbrand** de S, tal que S es inconsistente si S es falso bajo toda interpretación en este dominio.

EL PROCEDIMIENTO DE HERBRAND

Definiciones:

Sea $H_0 = \begin{cases} \{\text{constantes en S}\} & \text{si tal conjunto no es vacio} \\ \{a\} & \text{en otro caso} \end{cases}$

y para todo n, sea $H_{n+1} = H_n \cup \{f_i^k(t_1,\ldots,t_k) \mid t_1,\ldots,t_k \in H_n \ \& \ f_i^k \text{ letra funcional que aparece en S}\}$

el universo de Herbrand es: $H(S) = \bigcup_{i \geq 0} H_i$

Si en S no aparece ningún símbolo de función, entonces el universo de Herbrand es finito y $H(S) = H_0$. Pero si en S aparecen símbolos de función H(S) es infinito.

Toda expresión donde no aparecen variables es llamada **expresión básica**.

El **conjunto atómico de S ó Base de Herbrand** está definido como:

$Atom(S) = \{P_i^k(t_1,\ldots,t_k) \mid P_i^k \in REL \text{ y está en S y } t_1,\ldots,t_k \in H(S)\}$

Una **cláusula básica** de una cláusula C en S se obtiene al reemplazar a las variables de C por elementos de H(S).

Se define una interpretación especial sobre H(S), llamada la **interpretación-H** de S, como la transformación I tal que
a) I transforma toda constante de S en sí mismo.
b) $\forall f_i^k$ en S, $t_1,\ldots,t_k$ de S, $f_i^k(t_1,\ldots,t_k)$ se transforma en $f'^k_i(t'_1,\ldots,t'_k) \in H_0$ donde $t'_1,\ldots,t'_k \in H(S)$ y cada t'_i es el resultado de la transformación de t_i
c) A cada elemento de la Base de Herbrand $\{P_1,\ldots,P_n\}$ se le asocia un valor falso o verdadero. Tal asignación puede representarse por la lista $I = (m_1,\ldots,m_n)$, donde m_j es P_j si se asigna el valor verdadero ó m_j es $\neg P_j$ en otro caso.

Si la base de Herbrand tiene n símbolos predicativos, habrá 2^n distintas interpretaciones-H.

Dada una interpretación J sobre cualquier dominio D, su correspondiente interpretación-H, denotada I, está formada como sigue:
Cada $d_i \in D$ se transforma a un $t_i \in H(S)$. Si P_i^k es V (ó F) en J, entonces $P_i^k(t_1,\ldots,t_k)$ es asignado también V (ó F) en I. Esto facilita la demostración del lema siguiente.

Lema: Si una interpretación J sobre un dominio D satisface un conjunto de cláusulas S, entonces su correspondiente interpretación-H también satisface a S.
La demostración puede analizarse en [CUE, pág. 379].

Teorema: Un conjunto S de cláusulas es inconsistente si y solo si S es falso bajo toda interpretación-H de S.
Demostración.
($\Rightarrow$) Por definición: S es inconsistente si y solo si es falsa bajo toda interpretación en cualquier dominio, en particular si el dominio es el universo de Herbrand.
($\Leftarrow$) Se supone que S es falso para toda interpretación-H de S y supongase que S no es inconsistente. Luego debe existir una interpretación J sobre algún dominio D que satisface a S. Por el lema anterior la correspondiente interpretación-H de J satisface a S, esto contradice que S sea falso para toda interpretación-H de S por tanto S es inconsistente

Un procedimiento de decisión para probar la inconsistencia de S, usando el procedimiento de Herbrand sería:
a) Formar el Universo de Herbrand de S: H(S)
b) Generar de manera ordenada y exhaustiva las interpretaciones-H de S.
c) Si ninguna de estas interpretaciones satisface a S, S es inconsistente.

2.2 TRATAMIENTOS CLASICOS

En esta parte se esbozan las ideas de algunos de los métodos clásicos en el tratamiento de la demostración automática de teoremas.

Tanto el procedimiento de Herbrand como el método de Davis y Putnam, trabajan con conjuntos sucesivos $S_1,\ldots,S_i,\ldots$ de S, que contienen instancias básicas de las cláusulas en S. Y se intenta detectar un S_N de tal sucesión que sea inconsistente. Como puede notarse, la sucesión S_i es en general infinita y en principio no se sabe para cual valor de N, S_N será inconsistente. Esto hace la verificación de la inconsistencia de los S_i un proceso poco efectivo.

METODO DE DAVIS Y PUTNAM

Sea S un conjunto de cláusulas básicas. El método consiste de la aplicación de las cuatro reglas siguientes.

1.- **Omisión de Tautologías.** Borrar todas las cláusulas en S que representan tautologías (Una cláusula es una tautología si contiene a una literal y a su negación).

2.- **Regla de cláusula unitaria.** Si hay una cláusula en S que contiene a solo una literal L, definimos:

$S_1 = \{ C_i \in S \mid L \in C_i \}$

$S_2 = \{ M \mid (M \vee \neg L) \in S\}$

$S_3 = \{ C_i \in S \mid \text{ni } L \notin C_i \text{ ni } \neg L \notin C_i \}$

S puede simplificarse a $S' = S_2 \cup S_3$, ya que la consistencia de S equivale a la de S'.
Si en S se tuvieran dos cláusulas unitarias, una con L y la otra con ¬ L, entonces la cláusula vacía (NIL) se deriva de esta regla. En tal caso S es inconsistente.

3.- **Regla del literal puro.** Una literal L en S, se dice pura si y solo si la literal ¬L no aparece en ninguna cláusula de S.
Si una literal L es pura en S, se puede borrar a todas las cláusulas de S que contienen a L. El conjunto restante S'es inconsistente si lo es S.
Si después de aplicar estas reglas el conjunto S' obtenido, es vacío, entonces el conjunto original S es consistente.

4.- **Regla de división.** Se definen los siguientes conjuntos.

$S_1 = \{M \mid (M \vee L) \in S\}$

$S_2 = \{N \mid (N \vee \neg L) \in S\}$

$S_3 = \{O \mid O \in S \text{ y ni } L \text{ ni } \neg L \text{ aparecen en } O \}$

entonces S es inconsistente si tanto $(S_1 \cup S_3)$ como $(S_2 \cup S_3)$ son inconsistentes.

Todas las reglas de este método son sanas, es decir que la aplicación de ellas preserva la inconsistencia.

PROCEDIMIENTO DE RESOLUCION GENERAL

Con el fin de evitar la generación de la sucesión $S_0,\ldots,S_N,\ldots$ de instancias básicas de S, dos técnicas principales han sido propuestas: Una es el principio de Resolución de Robinson, y otra el método propuesto por Prawitz.

La **Resolución** se basa en dos principios: Silogismo Disyuntivo y Unificación [NOR]. El silogismo disyuntivo es un principio clásico de inferencia. De las premisas:

$P \vee Q$ y $\neg P$ se infiere Q.

ó bien, otra manera de expresar esta regla de inferencia llamada principio de resolución, es el siguiente:

de: $P \vee Q$

y: $\neg P \vee R$

---

se infiere: $Q \vee R$

A nivel proposicional el principio de resolución actúa como una ley de cancelación, la cual, dadas dos cláusulas con literales complementarias (literales que resuelven) forma una nueva cláusula derivada, conocida como **resolvente**.

Para poder aplicar el principio de resolución a cláusulas generales (que contienen términos y variables) se requiere aplicar "unificación sobre las literales complementarias".
Si se representa a las cláusulas como conjuntos de literales (con la disyunción entre las literales sobreentendida). Si $C_1 = \{ L_i \}$, $C_2 = \{ M_j \}$ y entre C_1 y C_2 no se tienen variables comunes (renombrando variables si es necesario), sean l y m instancias de una misma literal, tales que: $l \in C_1$, $\neg m \in C_2$ y además existe el u.m.g. s para $\{l, \neg m\}$. Entonces se dice que las cláusulas C_1 y C_2 **resuelven** y la cláusula: $\{C_1 - l\}s \cup \{C_2 - \neg m\}s$ se llama **resolvente** de C_1 y C_2.

Si dos cláusulas resuelven, ellas pueden tener más de un resolvente, puesto que puede tenerse más de una forma de elegir al l y $\neg$ m. En cualquier caso, se tiene sólo un número finito de resolventes. Por ejemplo con:

$C_1 = P(x,F(A)) \vee P(x,f(y)) \vee Q(y)$ y

$C_2 = \neg P(z,f(A)) \vee \neg Q(z)$

si elegimos a $l = P(x,f(A))$ y a $\neg m = \neg P(z,f(A))$, se obtiene el resolvente: $P(z,f(y)) \vee \neg Q(z) \vee Q(y)$

y con: $l = Q(y)$ y a $\neg m = \neg Q(z)$, se obtiene el resolvente:
$P(x,f(A)) \vee P(x,f(z)) \vee \neg P(z,f(A))$

El **proceso general de resolución** parte de un conjunto inicial de cláusulas S_0 y por la aplicación de todas las posibles resoluciones sobre S_0, se forma el conjunto de resolventes S_1, se construye entonces el conjunto S_2 que es el producto de resolver a S_1

con $S_0 \cup S_1$, y así sucesivamente se forma el conjunto S_n como producto de las resoluciones de S_{n-1} con $S_0 \cup S_1 \cup \ldots \cup S_{n-1}$.

Si el número de cláusulas no es pequeño y coinciden en ellas varias letras predicativas, el número de pasos para generar a la cláusula NIL es importante, así como el número de resolventes a generar. Por esto se tienen distintos criterios selectivos que simplifican el proceso, el conjunto de criterios selectivos articulados sistemáticamente constituye una estrategia de resolución [CUE]. Esta estrategia de resolución, determina la completitud y eficiencia del procedimiento. No se conoce a la mejor estrategia de resolución para todos los casos y cada estrategia conlleva a diferentes problemas.

METODO DE PRAWITZ

La idea de Prawitz es que dado un conjunto de cláusulas S éste es inconsistente si hay un conjunto M (copia o variante de S) y un conjunto de sustituciónes Θ tal que $M\Theta$ es inconsistente. Tal idea es una forma diferente de tratar el teorema de Herbrand.
Por ejemplo:

$$S = \{ P(x), \sim P(a) \vee \sim P(b) \}$$

y M una copia de S

$$M = \{ P(x_1), P(x_2), \sim P(a) \vee \sim P(b) \}$$

con la sustitución: $\Theta = \{ x_1/a, x_2/b \}$ puede probarse que $M\Theta = S'$ es inconsistente.

El procedimiento original de Prawitz puede verse como:
1.- transformar M a su Forma Normal Disyuntiva (FND). ($\vee D_i$)
2.- Para toda Frase D_i en la disyunción, se forma la condición sustitución $\alpha_{i1}, \ldots, \alpha_{ik_1}$ llamada θ_i que hace inconsistente a D_i
3.- Se forma la condición $\theta = \theta_1 \& \theta_2 \& \ldots \& \theta_m$
4.- Si la composición de sustituciones en θ puede realizarse sin contradicciones, se halló la solución de M.
5.- En otro caso buscar otros θ_i y regresar al paso 3.

La idea detrás de este procedimiento es el de evitar la generación en orden arbitrario de subconjuntos $S_1, \ldots, S_i, \ldots$ de S tal que algún S_i sea inconsistente (según el método de Herbrand), en su lugar se buscan las sustituciones α_i que lleven a la inconsistencia de S.

En un trabajo posterior (1969) Prawitz agilizó su procedimiento al reordenar al conjunto de cláusulas S en una matriz en Forma Normal Conjuntiva. Entre cada predicado (A_{ij}) de la matriz está el signo $\vee$ (disyunción) y entre cada línea de la matriz el signo $\wedge$ (conjunción).

$$M = \begin{matrix} A_{11} & \cdots & A_{1n_1} \\ A_{21} & \cdots & A_{2n_2} \\ \vdots & & \vdots \\ A_{m1} & \cdots & A_{mn_k} \end{matrix}$$

Def. Un camino es una sucesión $A_{1j_1}, \ldots, A_{mj_k}$ obtenida por tomar exactamente una literal de cada línea. La FND de S es obtenida al formar la conjunción de las literales de un camino y entonces tomar la disyunción de todos los caminos posibles.

La matriz M es inconsistente para alguna sustitución de las variables que aparecen si existe un conjunto θ de condiciones-sustitución con las siguientes propiedades:
1.- Cada camino de M contiene al menos una posible contradicción formandose el conjunto θ con las condiciones-sustitución que corresponden a las contradicciones de cada camino.
2.- En θ existe un conjunto completo y consistente de sustituciones. Tal conjunto es llamado conjunto refutación de M.

ESTRATEGIA DE REFINAMIENTO

Para la búsqueda de caminos contradictorios en M, si α es una condición-sustitución que corresponde a una contradicción entre las literales (A_{ij}, A_{kp}), i <> k y tal que ni A_{ij} ni A_{kp} se encuentran solos en sus líneas (n_i, n_k > 1). Si α es consistente con el θ que se lleva construido, entonces se añade α a θ. En este caso M es una contradicción si θ traduce tanto a M' y a M'' en contradicciones donde:
M' es M sin la línea i salvo A_{ij} y eliminando A_{kp} de la cláusula k
M'' se obtiene de M por eliminar A_{ij} de la cláusula i y a todas las literales de la línea k excepto A_{kp}.

Ejemplo:

M	M'	M''
Q(x,f(x)), P(x)	Q(x,f(x)), P(x)	Q(x,f(x)), P(x)
P(a), ~Q(a,f(a)), P(x)	P(a)	~Q(a,f(a)), P(x)
~P(x), Q(x,y), Q(a,f(a))	Q(x,y), Q(a,f(a))	~P(x)
~P(z), ~Q(x,f(a))	~P(z), ~Q(x,f(a))	~P(z), ~Q(x,f(a))

con α = <x/a>, θ = { <x/a> }. Se buscarán conjuntos-refutaciones para M' y M'' que contengan a α.
Mientras M contiene 36 caminos M' y M'' contienen juntos solo 16 caminos.

Si A_{ij} ó A_{kp} (pero no ambos) se encuentran solos en sus líneas, entonces θ traduce a M en una contradicción si traduce a M' o a M'' en una contradicción.

Este principio de reducción matricial genera un árbol de matrices y un conjunto θ de condiciones sustitución.

Si en una matriz Mi se escoge una posible contradicción entre dos literales que se encuentran solas en sus respectivas líneas, el nodo de la matriz es un nodo final.
El procedimiento termina con θ como un conjunto refutación para M cuando cada rama del árbol termina con solo nodos finales.

Cada $\theta_i \in T$ debe ser consistente con los otros θ_j $(j<i) \in \theta$, si en algún punto no existe tal θ_i, se regresa a la matriz precedente y se busca otra posible contradicción para tal matriz, si en los regresos sin hallar sustituciones consistentes para las contradicciones retornamos a la matriz original, el proceso termina indicando que el conjunto inicial es consistente.

Algunas de las inconveniencias que se ven a este método son:
- Es un método que requiere uso de una gran cantidad de memoria para el almacenamiento de las matrices y del conjunto-sustitución.
- El regreso a una matriz precedente es costoso, puesto que se elimina tanto a la matriz corriente como a su conjunto sustitución
- Llevar el conjunto sustitución θ consistente, implica un procedimiento adicional para revisar por cada sustitución que se realice, que ésta sea consistente con las que ya se tienen en T.

2.3 METODO DE REDUCCION MATRICIAL (RM)

En general el método intenta unificar, combinar y extender algunas reglas del método de Davis y Putnam con las de Prawitz, la técnica de demostración seguida es por refutación.
Este método en sus primeros pasos transforma al conjunto inicial de cláusulas S en una matriz de cláusulas (como en el método de Prawitz), adicionando propiedades a cada cláusula; como el número de literales, el de literales positivas, y el número de literales negativas. Propiedades similares son colocadas a la matriz recien formada. La idea al adicionar tales propiedades es para facilitar la aplicación de los siguientes pasos del método.

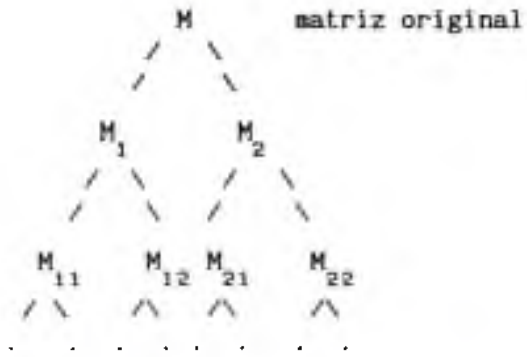
Enseguida se intenta trabajar con el número mínimo de cláusulas, por lo que se aplican estrategias de simplificación, como:
- Eliminación de Tautologías
- Reducción clausular (elimina literales repetidos de la cláusula)
- Eliminación de cláusulas con literales puros

Después de simplificar a cada cláusula y reducir el número de cláusulas, el método intenta probar la inconsistencia de la matriz utilizando para ello a las cláusulas unitarias (cláusulas que constan de sólo un literal), aplica resolución y eliminación por inclusión de las cláusulas unitarias con todas las demás posibles (Regla del literal aislado en el método de Davis y Putnam). Este paso es aplicado hasta que no más resoluciones y eliminaciones puedan aplicarse sobre cláusulas unitarias.

Def. Una cláusula D incluye a otra cláusula C, si existe una sustitución θ tal que $C\theta \subseteq D$. Por ejemplo:
$P(x)$ está incluida en $P(y) \vee Q(z)$ (con $\theta = \langle x/y \rangle$)
Una cláusula D que incluye a alguna otra cláusula, puede eliminarse sin afectar la inconsistencia del resto del conjunto clausular.

Toda cláusula resolvente de una cláusula unitaria con otra, es adicionada a la matriz de cláusulas, en tanto que el proceso de inclusión elimina a toda cláusula de la matriz que sea incluida por la cláusula unitaria. Este es un proceso finito, puesto que se tiene un número finito de cláusulas (entre ellas las unitarias), y la longitud del resolvente E de una cláusula D con una cláusula unitaria, es en cada paso reducido en uno. Es decir, si definimos: Longitud(C) = número de literales en C, y E es un resolvente de D y C (cláusula unitaria), entonces: Longitud(E) = Longitud(D) - 1.

Si en este paso se genera la cláusula NIL, entonces el método termina indicando inconsistencia del conjunto original de cláusulas.
Si después de la aplicación de este paso el conjunto resultante de cláusulas es vacío, entonces el conjunto inicial de cláusulas es satisfacible.
Si después de este paso no se prueba la inconsistencia o satisfactibilidad de la matriz clausular, se aplica entonces el principio de reducción matricial del método de Prawitz, la cual divide a la matriz que se está trabajando M en dos matrices M1 y M2, este paso va formando el árbol matricial.

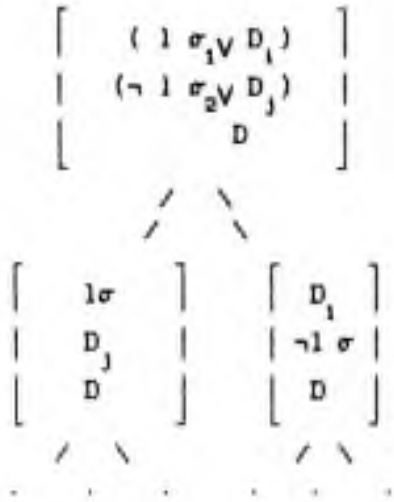


Cada una de las nuevas matrices creadas tienen una cláusula unitaria más que la matriz padre de la que provienen. Y el proceso general vuelve a aplicarse ahora para las nuevas matrices hijos.

Para dividir la matriz M en dos matrices M1 y M2, la estrategia a seguir es la siguiente:
Se buscan dos cláusulas C_i y C_j no unitarias (puesto que las unitarias ya han sido resueltas) que resuelvan sobre una literal l

Entonces $C_i = l\ \sigma_1 \vee D_1$; $C_j = \neg\ l\ \sigma_2 \vee D_j$; estas dos cláusulas son sustituidas en M1 por: $C'_i = l\ \sigma$; $C'_j = D_j\sigma$ y en M2 por: $C''_i = D_i\sigma$; $C''_j = \neg\ l\ \sigma$. Siendo $\sigma = u.m.g(\sigma_1, \sigma_2)$.

Tal regla se basa en la tautología:



Lo que genera un árbol de matrices. Y a cada una de las submatrices se le puede aplicar el algoritmo general.

Para seleccionar la literal 'l', se busca la literal de mayor frecuencia (mayor número de apariciones) en la matriz, ya que esto permite tener en M1 la nueva cláusula unitaria $l\sigma$ y en M2 a la cláusula unitaria $\neg\ l\sigma$. Y como l es la literal de mayor frecuencia, se tiene mas posibilidad para resolver y eliminar por inclusión con estas nuevas cláusulas unitarias.

Si en algún punto del árbol matricial, no es posible encontrar las cláusulas C_i y C_j con las cuales resolver, entonces se realiza un retroceso hacia la matriz padre y se selecciona a la siguiente literal más frecuente 'l_1' distinta a la 'l' anterior sobre la cual se dividió, es decir, se toma otra línea de ataque para la demostración.

Debido a que en los retrocesos se van probando alternativas de resolución sobre las literales, es necesario ir almacenando en cada matriz padre, el conjunto de literales sobre las cuales se ha intentado la división de matrices, así también las distintas cláusulas C_i y C_j que se usaron en la resolución.

Si en el proceso del retroceso se llega a la matriz raíz y no se encuentran más posibilidades para la división, entonces el método termina indicando que el conjunto original de cláusulas es satisfacible.

El proceso de manera general puede verse como:

1.- **Traducir el conjunto inicial de cláusulas.**
se pasa el conjunto original de cláusulas a una notación matricial y se renombran variables comúnes entre ellas.

2.- **Simplificación y reducción de cláusulas.**
aplicando estrategias de simplificación como:

a) Eliminación de tautologías
b) Eliminación de literales repetidos
c) Eliminación de cláusulas con literales puros

3.- **Regla para cláusula unitaria.**
se buscan cláusulas con una única literal

i) Si existen posibles contradicciones entre dos cláusulas con una única literal entonces el procedimiento termina y S es insatisfacible.
(P(x) ^ ~P(y)) ==> NIL

ii) Para toda cláusula unitaria C = L_1, se elimina ~L_j de las cláusulas que la contengan. Sólo si L_1 unifica con ~L_j
L_i ^ (~L_j v B) ≡ B --> Resolución.

iii) Se eliminan todas las cláusulas que incluyan a la cláusula unitaria.

a) Si después de aplicar el paso 3 el conjunto clausular queda vacío, entonces S es satisfacible

4.- **Regla de División.**
Después de la aplicación reiterada del paso 3, la estructura matricial no ha cambiado. El paso 4 divide a la matriz original M en dos submatrices M_1 y M_2 tales que:
M es inconsistente si M_1 y M_2 son inconsistentes. Y el algoritmo general puede aplicarse reiteradamente a las submatrices creadas.

El método (RM) es el diseñado y construido en este trabajo de tesis, en el capítulo siguiente se detalla con amplitud cada uno de los pasos seguidos en el diseño y construcción de tal método.

EXPLICACION DETALLADA DEL METODO RM

En este capítulo se describe a detalle el método de reducción matricial (RM). Se señalan las diversas estrategias utilizadas con el fin de agilizar el método de demostración. Se pone especial énfasis en describir el algoritmo lógico y de control. Para ésta descripción se utiliza una pseudonotación de un lenguaje artificial de programación, así también se describe la estructura de datos usada para la representación, tanto de las cláusulas, como de la matriz de cláusulas y del árbol matricial. Por último se escriben breves comentarios acerca de la eficiencia del programa que implementa al método RM, programado en el lenguaje LISP.

3.1 ESTRATEGIAS UTILIZADAS EN EL METODO RM

El número de pasos para probar la inconsistencia de un conjunto de cláusulas es importante, por lo que es recomendable el aplicar **estrategias** de manera sistemática que simplifiquen el proceso y al propio tiempo lo hagan programable.
En los métodos de demostración automática de teoremas, diversos tipos de estrategias han sido utilizados para agilizar los pasos en la demostración. Una de las clasificaciones de las estrategias es la realizada por Nilsson [NIL1], las cuales han sido ya expuestas en el capítulo anterior. En resumen, las estrategias son:

*** Estrategias de simplificación**
Eliminación de ciertas cláusulas y de ciertas literales, conservando la completitud del método. El conjunto original de cláusulas es inconsistente si el conjunto simplificado (el conjunto cláusular resultante de la aplicación de las estrategias de simplificación) es inconsistente.

*** Estrategias de refinamiento.**
Son refinamientos basados en resultados de la teoria de la demostración automática de teoremas, que establece que no todas las resoluciones posibles son necesarias de calcular. La idea con estas estrategias es la de generar sólo las cláusulas necesarias que lleven a la prueba del teorema.

*** Estrategias de ordenamiento.**
Estas corresponden más cercanamente a los métodos de búsqueda empleados en las búsquedas en gráficas AND/OR y del espacio de estados. Estas estrategias no prohiben algún tipo particular de resolución, sino que más bien, sirven como guía para saber cuales realizar primero.

Entre las estrategias utilizadas en el método RM están:
A) ESTRATEGIAS DE SIMPLIFICACION.

*** Eliminación de tautologías**
Toda cláusula que contenga a una literal y a su complemento (tal cláusula es una tautología), puede ser eliminada, sin menoscabo de la inconsistencia del conjunto original.

*** Reducción Cláusular**

Si hay repeticiones de literales en una misma cláusula, la eliminación de literales repetidas puede realizarse sin perder información de la cláusula. Por ejemplo en:
$P(A) \vee Q(X) \vee P(A)$ puede eliminarse $P(A)$ quedando $P(A) \vee Q(X)$

*** Eliminación de Literal Puro**

Se dice que un literal P es puro en una fórmula si solo aparece en un estado, es decir, si aparece P en el conjunto entonces no aparece ¬P y reciprocamente. Es posible la eliminación de las cláusulas que contienen a un literal puro, sin afectar la inconsistencia del conjunto original de cláusulas.

*** Eliminación por inclusión**

Una cláusula D que incluye a alguna otra cláusula C, puede eliminarse sin afectar la inconsistencia del resto del conjunto cláusular. En el método RM se trabaja siempre con C (la cláusula que está incluida) siendo una cláusula unitaria. El proceso que prueba si una cláusula D incluye a otra, es el siguiente.

1.- Revisar que la literal P de C (cláusula unitaria) se encuentre en D.
2.- Se aplica una versión restringida del algoritmo de unificación consistente en que las sustituciones de variables se realizan sólo en variables de P, pero no sobre variables que estén en D.
3. Si existe el u.m.g(C,D) con la restricción del paso 2, entonces D incluye a C.

B) ESTRATEGIAS DE REFINAMIENTO.

*** Orden de la Resolución.**

El procedimiento general de la resolución, inicia resolviendo entre las cláusulas del conjunto original S_0, las resolventes resultantes forman el conjunto S_1.

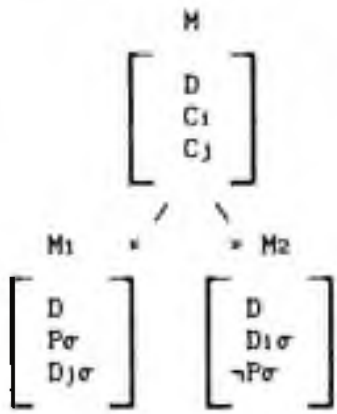
Se resuelven ahora las cláusulas de S_1 con las de $S_0 \cup S_1$, formándose el conjunto S_2. En general, se toma cada cláusula de S_i y se resuelve con las de $S_0 \cup S_1 \cup \ldots \cup S_i$.

En la implementación del método RM, una de las cláusulas resolventes es siempre una cláusula unitaria. Por lo que la búsqueda de cláusulas unitarias C es sobre todo $S_0 \cup S_1 \cup \ldots \cup S_i$, con la alternativa siguiente:
Si $C \in S_i$ entonces la cláusula con la que resuelva debe estar en $S_0 \cup S_1 \cup \ldots \cup S_i$.
Si $C \in S_0 \cup S_1 \cup \ldots \cup S_{i-1}$ entonces debe resolver con una cláusula de S_i.
Esto es con el fin de evitar repetir resoluciones.

*** Principio de División (o Reducción Matricial)**

Para dividir la matriz M en dos matrices (M_1, M_2), se buscan dos clásulas de la forma:
$C_i = P\sigma_1 \vee D_i$
$C_j = \neg P\sigma_2 \vee D_j$

Si aparte de C_i y C_j, M consta de un conjunto D de cláusulas, entonces M1 y M2 se forma de M de la manera siguiente:



Siendo $\sigma = u.m.g(\sigma_1, \sigma_2)$.

Para M1 en lugar de la cláusula C_i, se genera una cláusula unitaria $C'_i = P\sigma$, y en lugar del C_j original se tiene $C'_j = D_j\sigma$.

En M2, se tiene el nuevo $C''_i = D_i\sigma$ y la cláusula unitaria $C''_j = \neg P\sigma$

Puesto que el método aprovecha la existencia de cláusulas unitarias, las nuevas matrices creadas M1 y M2, ofrecen una línea adicional de prueba para la resolución y eliminación de cláusulas.

La elección de las cláusulas sobre las cuales dividir está basado en las ideas siguientes.

i) **Se busca el literal P que aparece con mayor frecuencia en la matriz M.** De tal forma que se obtengan mas posibilidades para las futuras resoluciones y eliminaciones en M1 y M2, haciendo uso de las cláusulas unitarias P y ¬P respectivamente.

ii) **Las cláusulas C_i y C_j deben ser no unitarias (evitándose así reducciones simples).** Puesto que las cláusulas unitarias han sido ya tratadas y una reducción simple no agrega más información de la que ya se tiene en M.

C) ESTRATEGIAS DE ORDENAMIENTO

*** Aplicación sólo sobre cláusulas unitarias de la eliminación por inclusión y de la resolución.**

Una de las dos cláusulas a las que se aplica resolución y eliminación por inclusión, es una cláusula unitaria. Lo que facilita los procedimientos de eliminación y resolución.

Por ejemplo, cuando se busca eliminar las cláusulas que incluyan a la cláusula unitaria C (suponiendo que su único literal es P), se busca sólo sobre la posición del literal P en todas las demás cláusulas de la matriz.

El aplicar resolución entre las cláusulas C y D, siendo C una cláusula unitaria garantiza que la cláusula resolvente es de longitud más corta que D. Puesto que el objetivo del método es generar la cláusula NIL (cláusula de longitud cero), la generación de resolventes con longitudes cada vez más cortas es deseable.

3.2 ESTRUCTURA DE DATOS UTILIZADA EN LA CONSTRUCCION DEL METODO RM

Los objetos con los que trabaja el método RM, a saber las cláusulas, la matriz de cláusulas y el árbol de matrices, son traducidos como datos del programa. Tales datos se forman de la manera siguiente:

Una cláusula es representada mediante su nombre y su valor, por ejemplo: C1$ es el nombre de la cláusula con valor
(((A B)) ((x C)(y z)) ((0))).

LITERAL DE UNA CLAUSULA

El formalismo empleado para denotar a los argumentos de una literal en una cláusula, es mediante una lista de argumentos, utilizando al espacio en blanco para separar los términos del argumento Por ejemplo:

((¬ B z) (A x)) - indica que P tiene dos ocurrencias en la cláusula; de la forma $\neg P(B,z) \vee P(A,x)$

((x y)) - representa al literal P(x,y)

((0)) - indica que no hay ocurrencias del literal P en la cláusula.

CLAUSULA

El valor de la cláusula se ordena lexicográficamente sobre las literales (P_0 P_1 ... P_n), sobreentendiéndose al conector $\vee$ (disyunción) entre ellas, tomando sólo una vez al literal sin importar el número de ocurrencias del mismo en la cláusula.
Dado el orden de las literales (se considera a todas las literales que aparecen en el conjunto inicial de cláusulas), se coloca en su posición a su lista de argumentos. (() () ... ())
P_0 P_1 ... P_n

Por ejemplo, si el conjunto original de cláusulas es

$S = \{ C0 = (Q(B) \vee \neg R(A,x))$
$C1 = (P(x,y,z) \vee Q(B) \vee \neg Q(z))$
$C2 = (R(C,y) \vee \neg P(A,C,x) \vee R(x,A)) \}$

este conjunto cláusular se denotará como:

	P	Q	R
C0$ =	((0))	((B))	((¬ A x)))
C1$ =	((x y z))	((B)(¬ z))	((0)))
C2$ =	((¬ A C x))	((0))	((C y)(x A)))

Puede notarse que una cláusula es representada mediante un nombre Ci$ y su valor. Por ejemplo; C1$ es el nombre de una cláusula con valor (((x y z)) ((B)(¬ z)) ((0))).
El nombre de la cláusula es representado por un átomo de LISP, tales átomos son generados por el programa cada vez que una nueva cláusula es creada, teniendo cuidado de no repetir estos átomos.
Al átomo generado se le asigna un valor, en este valor se coloca la lista de argumentos que tiene la cláusula, en la notación y orden ya descrito. Para facilitar los pasos del método, ciertas propiedades son colgadas al átomo que representa al nombre de la cláusula. Estas propiedades son:

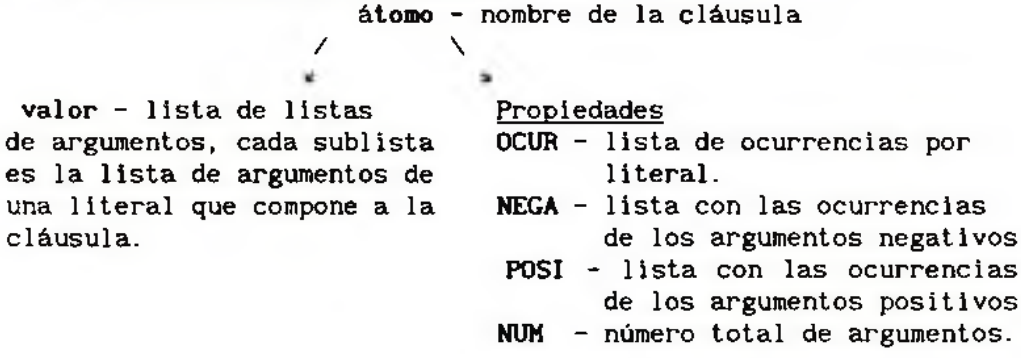
OCUR - **Lista de ocurrencias.** indica el número de argumentos que tiene cada literal que compone a la cláusula.

NEGA - **Lista de ocurrencias negativas.** Cada elemento de esta lista da el número de argumentos negativos que tiene el literal situado en similar posición dentro del valor de la cláusula

POSI - **Lista de ocurrencias positivas.** Similar que en NEGA, salvo que ahora indica el número de argumentos positivos.

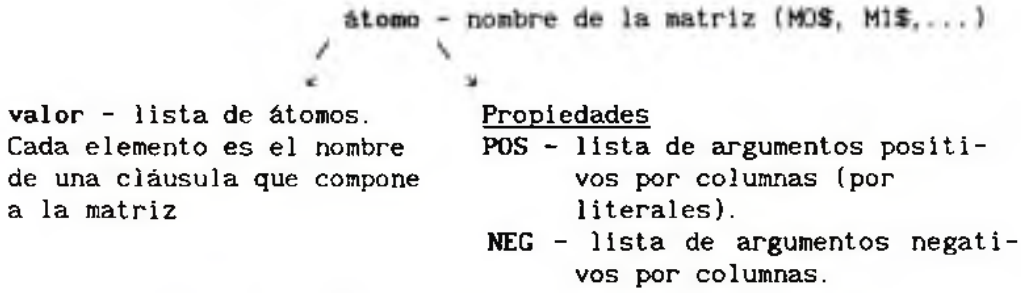
NUM - **Total de argumentos.** Es un átomo numérico que indica el total de argumentos de la cláusula.

Resumiendo, una cláusula se representa en el programa mediante la estructura siguiente.



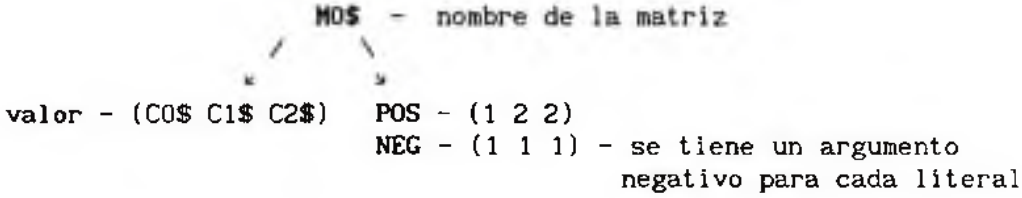
MATRIZ CLAUSULAR

La matriz de cláusulas M se representa por



Al igual que para las cláusulas, una matriz clausular se representa mediante un nombre (un átomo de LISP), y su valor (lista con el nombre de las cláusulas que componen a la matriz). El átomo que representa al nombre de la matriz se genera automáticamente en cada formación de una nueva matriz. El valor de la matriz se representa por una lista, cuyos elementos son los nombres de las cláusulas que la componen.
Las dos propiedades que se cuelgan al átomo que representa al nombre de la matriz, dan información por columnas (por literales). E indican el número de argumentos tanto positivos como negativos para las literales que están en la matriz.

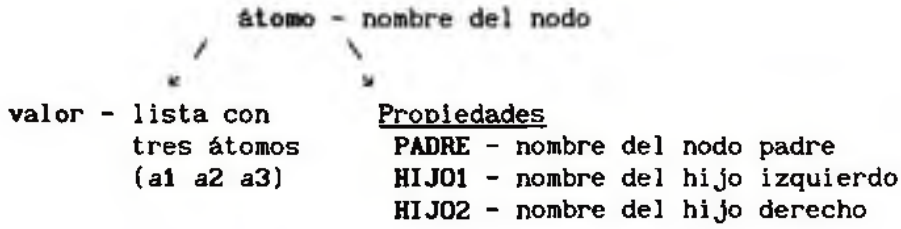
Por ejemplo, si se representa al conjunto anterior S de cláusulas, mediante una matriz cláusular, suponiendo que ya ha sido representada cada cláusula con los nombres; C0$,C1$ y C2$, se obtiene la representación siguiente.



El árbol matricial es un árbol binario compuesto de un conjunto de nodos, donde cada nodo denota a una matriz clausular.

NODO DEL ARBOL MATRICIAL

Un nodo del árbol es implementado de la manera siguiente



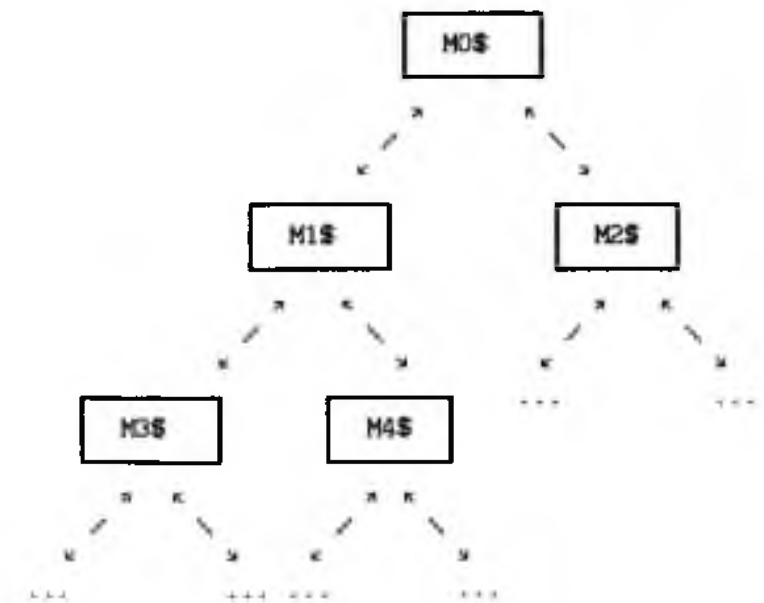
El valor colocado en las propiedades se usa como apuntador, esto permite formar la estructura árbol entre los nodos.
El padre del nodo raíz es NIL, en tanto que para nodos finales tanto HIJO1 como HIJO2 son NIL.
a1 es el nombre de la matriz representada por el nodo.
a2 = átomo - nombre de la lista de letras predicales

valor - lista de las diferentes letras predicales con las que se ha intentado la división de matrices.

a3 = átomo - nombre de la lista para las cláusulas que dividen

valor - una lista que contiene los nombres y los argumentos de las cláusulas C_i y C_j, usadas para la división matricial.

ARBOL MATRICIAL



El recorrido del árbol se hace a lo profundo y para esto se usa una pila llamada **RAMA** donde se colocan a los nodos hijos(nodos a procesar futuramente por el algoritmo).
Para realizar los retrocesos (esto es cuando ya no es posible realizar ni resolución ni división matricial), se usa una pila llamada **BACKUP** donde se coloca a los nodos padres (nodos ya procesados por el algoritmo).

3.3 ALGORITMO LOGICO Y DE CONTROL

PASO 1.
a) **Formar el conjunto de las letras predicativas que aparecen en el conjunto original de cláusulas S.**
b) **Renombrar variables.** De tal forma que para cualesquiera dos cláusulas distintas, no se tiene variables comúnes entre ellas.

PASO 2.
Eliminar tautologias y repeticiones de predicados.
Se reduce al conjunto original de cláusulas, eliminando información repetida e irrelevante.

PASO 3.
Traducir la notación de entrada del conjunto original de cláusulas a la estructura de datos usada para su representación.
Se traduce a las cláusulas de entrada como datos del programa con la estructura anteriormente descrita, adicionando sus propiedades (NUM, POSI, NEGA, OCUR), a cada cláusula.

PASO 4.
Formar la matriz clausular inicial.
Con las cláusulas anteriormente representadas, se forma la matriz inicial MAT0 que será el nodo raíz del árbol matricial.

PASO 5.
Inicialización del árbol matricial.
La matriz construida en el paso 4, se pone como nodo raíz del árbol y se inicializan las estructuras que representan al árbol.

```
NPP <- NIL   ; inicializa lista de literales
CC <- NIL    ; incializa lista de cláusulas que dividen a la matriz
NODO-RAIZ <-  (MAT0 NPP CC) ; nodo raíz
RAMA <- (NODOI)      ;primer nodo del árbol
BACKUP <- (NIL)      ;el padre de la raíz es NIL.
```

PASO 6.
Es inconsistente el conjunto original de cláusulas ?

```
     Si (RAMA == NULL)  Entonces  FIN DEL ALGORITMO
                ;es  inconsistente  el  conjunto  original  de  cláusulas
```

PASO 7.
Recorrer a los hijos buscando inconsistencia.
Tomar el primer nodo aún no procesado.

```
ND <- (POP RAMA)     ;los nodos no procesados están en RAMA
                     ;ND tiene la forma (MAT NPP CC) con:
                     ;MAT - nombre de la matriz a procesar
                     ;NPP - lista de literales
                     ;CC - lista con los pares de cláusulas.
```

PASO 8.
Proceso del literal puro.
Si en las cláusulas que componen a la matriz corriente MAT aparecen literales puros, eliminar las cláusulas que contengan a tales literales. La detección de literales puros, se facilita debido a las propiedades colgadas a la matriz cláusular. Se busca en las propiedades POS y NEG si un literal tiene sólo argumentos positivos y no negativos o viceversa, y entonces tal literal es un literal puro.

PASO 9.

Proceso de cláusula unitaria (literal aislado).

Para saber si una cláusula es unitaria, se revisa en la propiedad NUM de la cláusula, si en ésta propiedad se tiene el valor 1 entonces la cláusula es unitaria.

a) Inicializar conjuntos para el proceso

```
Di = MAT      ;último conjunto de resolventes
CLAT = MAT    ;CLAT = Do ∪ ... ∪ Dn - conjuntos de resolventes
```

```
b)Di+1 <- NIl ; conjunto a formarse con los nuevos resolventes
```

c) Para toda cláusula unitaria Cj en MAT

```
SI Cj ∈ Di Entonces (CLAT = ELIMINA_INCLUSION(Cj en CLAT)
                    RES <- CLAT) ;resolver con Do ∪...∪ Di
    ;para resolventes recien generados, probar eliminación
    ;por inclusión con las cláusulas anteriores.
    En otro caso  RES <- Di   ;Cj ∈ Do ∪...∪ Di-1 ,entonces
                              ;Cj resuelve con cláusulas en Di
```

d) Di+1 <- (RESOLUCION Cj con RES) + Di+1

```
                    ;adicionar a los nuevos resolventes
```

e) Si (NIL ∈ Di+1) **Entonces** Pasar al paso 15

```
    ;si se genera la cláusula NIL, MAT es inconsistente.
```

f) Si (Di+1 ≠ NULL) **Entonces** CLAT <- CLAT ∪ Di+1 y regresar al inciso (b) de este paso.

```
                        ;en CLAT se lleva Do ∪ ... ∪ Di+1
```

En otro caso MAT <- CLAT ;nueva matriz corriente

g) **Si** (MAT == NULL) **Entonces** FINAL DEL ALGORITMO

```
                    ;el conjunto original es satisfacible.
```

PASO 10.

División sobre MAT.

Se generan dos nuevas matrices M1 y M2 creándose sus correspondientes nodos. Estas dos matrices se forman de la manera siguiente **Selección del literal más frecuente.** Se busca la literal 'P' que no esté en NPP y sea la de mayor frecuencia en MAT.

La frecuencia de una literal en la matriz, lo da la suma de sus correspondientes argumentos positivos y negativos, estos valores están colocados en las propiedades POS y NEG de la matriz.

Si no se encuentra tal P **Entonces** pasar al paso 16

```
En otro caso
NPP <- NPP + P  ;adición del literal más frecuente
CC <- NIL       ;inicializar lista de cláusulas para prueba de
                ;la resolución sobre el literal más frecuente.
```

PASO 11.

Selección de las cláusulas C_i y C_j que resuelvan sobre P.

las cláusulas deben tener la forma: $C_i = P\sigma_1 \vee D_i$ y $C_j = \neg P\sigma_2 \vee D_j$ y existe $\sigma = u.m.g.(\sigma_1, \sigma_2)$. Y tal que $\langle C_i, C_j \rangle$ no estén en CC.

tanto la cláusula C_i como C_j deben tener argumentos en la posición de la literal P, lo cual se revisa en la propiedad OCUR de las cláusulas.

Si se encuentran tales C_i y C_j **Entonces** Pasar al paso 13

En otro caso regresar al paso 10

;cambiar de literal sobre el cual resolver

PASO 12.

Creación de dos nuevas matrices M_1 y M_2.

CC <- CC + <D_i,D_j> ;lista de cláusulas que han resuelto sobre P

Las dos nuevas matrices M_1 y M_2 heredan al conjunto cláusular de MAT (se copia la lista de nombres de las cláusulas componentes de la matriz) con excepción de las cláusulas C_i y C_j.

En M_1 las nuevas cláusulas C_i y C_j se llaman C'_i y C'_j formadas como: $C'_i = P\sigma$; $C'_j = D_j\sigma$. En M_2 las nuevas cláusulas C_i y C_j serán C''_i y C''_j formadas por: $C''_i = D_i\sigma$ y $C''_j = \neg P\sigma$.

PASO 13.

Construcción de los nodos para las nuevas matrices.

Se generan dos nuevos nodos para el árbol formados por:

```
LIT1 <- NULL     ;inicializar con el conjunto vacío a la lista de
LIT2 <- NULL     ;literales con las que se va a probar la división.
CC1  <- NULL          ;lista de pares de cláusulas para la resolución
CC2  <- NULL          ;del literal más frecuente, también se inicializa
                 ;con el conjunto vacío.
ND1 <- (M1 LIT1 CC1)   ;Construcción de los nuevos nodos
ND2 <- (M2 LIT2 CC2)   ;para las matrices M1 y M2.
```

PASO 14.

Colgar nueva rama binaria al árbol.

Para formar la estructura árbol, se utilizan las propiedades colgadas a cada nodo del árbol.

```
HIJO1(ND)  <- ND1       ;los dos nuevos nodos creados serán los
HIJO2(ND)  <- ND2       ;hijos del nodo que contiene a MAT
PADRE(ND1) <- ND        ;el padre de los nuevos nodos es el nodo
PADRE(ND2) <- ND        ;que contiene a MAT.
PUSH(ND1,RAMA)       ;adición de nuevos nodos al árbol.
PUSH(ND2,RAMA)
PUSH(ND,BACKUP)      ;salvar al padre para posibles retrocesos.
```

Y regresar al paso 6.

PASO 15.

Pasar a la siguiente rama del árbol.

Si no hay en RAMA nodos hermanos de MAT

Entonces Elimina de BACKUP al padre de MAT

;los hijos son nodos fallas.

E ir al paso 6 ;continuar con la otra rama

PASO 16.

Retroceso en la búsqueda.

Si (BACKUP == NULL) **Entonces** Fin del algoritmo

;no hay más nodos padres, S es satisfacible.

En otro caso Retroceder al nodo padre del nodo corriente

Y regresar al paso 11.

;sustituir a la matriz corriente por su matriz
;padre y elegir a otras cláusulas que resuelvan
;sobre el literal que divide a la matriz.

En la construcción del método RM se ha cuidado el uso de memoria. Aún cuando la estructura de datos principal es la matriz de cláusulas, el valor de las cláusulas no es repetido de una matriz a otra, en lugar de ello, al aplicar la división de matrices las cláusulas comunes se heredan de la matriz padre a las matrices hijos. Esto se realiza con solo copiar la lista con los nombres de las cláusulas comunes de la matriz padre y asignarla como la lista de las cláusulas componentes en las matrices hijos.

No se lleva un conjunto de condiciones-sustitución como en el método de Prawitz, en lugar de ello, se aplica el proceso de Unificación. Por lo que las sustituciones se aplican instantaneamente manteniendose la consistencia de sustituciones aplicadas anteriormente.

Los retrocesos a las matrices padres se utilizan como un último recurso; cuando no es posible el resolver y eliminar por inclusión con cláusulas unitarias, ni tampoco es posible aplicar una nueva división matricial.

Todo esto hace que aunque el método RM conste de más reglas que el principio de resolución general, los resultados dados por el método a diversos ejemplos elegidos aleatoriamente, muestran una mayor eficiencia que se refleja en el menor número de pasos necesitados para la demostración automática de los teoremas.

CONCLUSIONES

APORTACIONES

Como resultado del trabajo de esta tesis, se obtuvo un programa demostrador de teoremas en el cálculo de predicados, que solicita como entrada a un conjunto de cláusulas y si tal conjunto representa un razonamiento correcto (un teorema), el programa encuentra y muestra el camino a seguir para su demostración.

El programa se basa en el diseño de un nuevo método para la demostración de teoremas denominado **Método RM**, tal diseño ha sido presentado en el capitulo III de este documento de tesis.

El objetivo inicial planteado, de construir un programa demostrador de teoremas fue cumplido. El comportamiento de los resultados obtenidos mediante el programa, pueden caracterizarse de la forma siguiente:
Si el conjunto de cláusulas de entrada consta de un número 'reducido' de cláusulas (cinco o menos cláusulas) y contiene cláusulas unitarias, entonces el número de pasos para la demostración de la inconsistencia es mayor que con algunas de las estrategias de resolución (por ejemplo usando resolución lineal), debido a que en la demostración el método RM usa exclusivamente el procedimiento de resolución general.
Pero si el conjunto cláusular de entrada expresa a un teorema 'extenso' (en general más de cinco cláusulas) ó la mayoría de las cláusulas son no unitarias, la eficiencia del método RM sobrepasa en mucho a la resolución lineal. Puesto que al aplicar la regla de división, divide al teorema inicial en dos subteoremas y cada uno de estos últimos reduce generalmente en más de la mitad la complejidad de la demostración. Y así sucesivamente, a estos subteoremas se les aplica nuevamente la regla de división, hasta que en algún nodo del árbol de subteoremas, la demostración del subteorema final es realizado con un número mínimo de pasos.
Se espera que el uso y análisis futuro del programa, demuestre su eficacia y facilite el desarrollo de sistemas que utilicen técnicas de deducción lógica.

Las experiencias y conocimientos adquiridos en el desarrollo de este trabajo, pueden clasificarse en dos lineas.
Primero: Mayor conocimiento de las técnicas para la demostración automática de teoremas, así como de las diversas estrategias aplicadas en la optimización de la demostración.
Segundo: Experiencia para futuros desarrollos en ambientes LISP's, usando las facilidades que brinda en el manejo de estructuras dinámicas.

LIMITACIONES

Las limitaciones pueden agruparse en dos clases: las teóricas y las prácticas.

La mayor limitación teórica queda enmarcada en la indecibilidad del problema mismo. Puesto que ya en el teorema de Gödel quedó establecido la imposibilidad de construir un programa demostrador de teoremas en el cálculo de predicados (y en general, en cualquier teoría que incluya a las teorías de primer orden), que de manera completa pueda decidir cuando una fórmula bien construida dentro de la teoría es o no un teorema.
Así pues, se hace necesario caracterizar al subconjunto del Cálculo de Predicados que posea un procedimiento efectivo de demostración. Ya que se han encontrado interesantes teoremas cuya demostración puede realizarse efectivamente. Sin embargo, "efectividad" en este contexto no coincide con "eficiencia", pues si bien el número de pasos para demostrar un teorema es finito, este puede ser muy grande.
Queda entonces el objetivo de incrementar la eficiencia de los programas demostradores y reducir el número de pasos empleados para demostrar fórmulas que efectivamente sean teoremas.

En este trabajo no se trató el análisis de la complejidad del método. Por lo que no puede afirmarse que sea el método más eficiente posible.
Aunque los ejemplos ejecutados en el programa muestran una reducción del número de pasos empleados para su demostración con respecto a métodos clásicos ampliamente trabajados.

Dentro de las limitaciones prácticas, puede mencionarse la omisión de un módulo para la 'comunicación agradable' hombre-máquina, tal que permitiera expresar a las fórmulas de entrada en una notación más simple o más legible para el usuario (posiblemente usando un lenguaje pseudo-natural).

El método implementado no hace uso alguno de heurísticas dependientes del problema. Por lo que puede decirse que el método trabaja a un nivel sintáctico, sin aprovechar ventajas que diera la interpretación (comprensión o semántica) de la fórmula a demostrar.

APLICACIONES FUTURAS

El programa presentado abre una linea de experimentación para nuevas ideas que agilicen las técnicas de demostración automática. Este programa es un ejemplo de que los métodos en la demostración automática de teoremas no pertenecen a un campo cerrado, sino que por el contrario, se espera se construyan mecanismos más eficientes que los hasta hoy practicados.

En particular el método RM proporciona la base para ulteriores desarrollos en áreas fundamentales para la Inteligencia Artificial como:

* El método RM puede formar el núcleo de resolvedores de problemas deductivos.

* El programa puede ser extendido, usando al método RM como base para el diseño de lenguajes de programación lógica. Tipo PROLOG, con la consiguientes ventajas, en constraste al lenguaje PROLOG.
i) Trabaja con cláusulas generales y no sólo de Horn.
ii) La búsqueda para la deducción es guiada por estrategias de ordenamiento y división de teoremas en subteormas, en lugar de una búsqueda lineal (Resolución LUSH).
iii) Irrelevancia del orden dado para las reglas y los hechos.
iv) Uso reducido del retroceso en las búsquedas.
Todo esto, extiende el poder de deducción para los futuros lenguajes de programación lógica, y permite la solución de problemas solubles que hoy en día no pueden resolverse usando PROLOG.

* El método puede aplicarse en la verificación formal de programas Puede describirse la ejecución de un programa mediante una fórmula A, y la condición de que el programa termine por otra fórmula B. Entonces, el verificar la terminación del programa es equivalente a probar que B se deduce de A.

* Puede usarse en los sistemas de planeación, encargandose del mecanismo de deducción.

Y en general, el método RM puede aplicarse a todo tipo de problema expresable como un problema de demostración automática, o que incluya como uno de sus elementos de trabajo a un mecanismo de deducción lógica.

BIBLIOGRAFIA

[CHA] - Chang Chin - Liang and Lee Char - Tung, 'Symbolic Logic and Mechanical Theorem Proving', Academic Press Inc., 1973

[CHAR]- Charniak Eugene and McDermott Drew, 'Introduction to Artificial Intelligence', Addison - Wesley Publishing Co., 1987.

[CUE] - Cuena José, 'Lógica Informática', Alianza-Editorial S.A., Madrid 1985.

[GBA] - Garcés Baés J. Alfonso, 'Introducción a la Programación Lógica', Reporte técnico del Departamento de Ingeniería Eléctrica del CINVESTAV-I.P.N., Serie Amarilla, No. 56, México 1987.

[HOF] - Hofstadter Douglas R. ,'Gödel, Escher y Bach: una eterna trenza dorada', Publicación del CONACYT, 1982.

[KOW1]- Kowalski R. and Hayes P., 'Semantic Trees in Automatic Theorem Proving', Machine Intelligence Vol. 4, B. Meltzer and D. Michie eds., American Elsevier Pub., N.Y. 1969.

[KOW2]- Kowalski R., 'Search Strategies for Theorem Proving', Machine Intelligence Vol. 5, B. Meltzer and D. Michie eds., Edinburgh University Press 1969.

[KOW3]- Kowalski R., 'Logic for Problem Solving', North - Holland Inc., 1979.

[LOV1]- Loveland D. W., 'Theorem-Provers Combining Model Elimination and Resolution', Machine Intelligence Vol. 4, B. Meltzer and D. Michie eds., American Elsevier Pub., N.Y. 1969.

[LOV2]- Loveland D. W., 'Automated Theorem Proving: a logical basis', Amsterdan - New York - North Holland Pub. Co., 1978

[LUC] - Luckham D., 'The Resolution Principle in Theorem Proving', Machine Intelligence Vol. 1+2, Collins Dale and Michie eds., Edinburgh University Press, 1971.

[NIL1]- Nilsson Nils J., 'Problem - Solving Methods in Artificial Intelligence', McGraw - Hill, Inc. 1971.

[NIL2]- Nilsson Nils J., 'Principles of Artificial Intelligence', Tioga Publishing Co., Palo Alto, CA. 1980.

[NOR] - Noriega B.V. Pablo, 'La Lógica de Prolog', Memorias de la II reunión sobre Inteligencia Artificial, Fundación Arturo Rosenblueth, México D.F. 1985.

[PEV] - Pevac I. and Cvetkovic D.,'Man-Machine Theorem Proving in Graph Theory', Artificial Intelligence an International Journal, Vol. 35, No. 1, Mayo 1988.

[PLA] - Plaised David A., 'Non-Horn Clause Logic Programming Without Contrapositives', Journal of Automated Reasoning, Vol. 4, No. 3, Septiembre 1988.

[PLO] - Plotkin G. D.,'A further note on inductive generalization', Machine Intelligence Vol. 6, B. Meltzer and D. Michie eds., Edinburg University Press 1971.

[PRA] - Prawitz D., 'Advances and Problems in Mechanical Proof Procedures', Machine Intelligence Vol. 4, B. Meltzer and D. Michie eds., American Elsevier Pub., N.Y. 1969.

[RIC] - Rich Elaine, 'Artificial Intelligence', McGraw - Hill International Editions, 1983.

[WBU] - W. Buntine, 'Generalized Subsumption and Its Application to Induction and Redundancy', Artificial Intelligence an International Journal, Vol. 36, No. 2, Septiembre 1988.

[WWB] - W. W. Bledsoe, 'Non-resolution Theorem Proving', Artificial Intelligence an International Journal, Vol. 8 and 9, North - Holland Publishing Company, 1977.

MANUAL DE USO Y UNA SESION CON EL PROGRAMA

El programa fue implementado en el lenguaje Lisp, usando al intérprete-compilador MULISP - 87 y corriendo en computadoras personales IBM y/o compatibles. Para dar ejecución al programa, se llama desde el modo ejecución de MULISP al archivo **REDUCMAT**, el cual será ejecutado automaticamente al terminar de ser cargado en la memoria de trabajo.
Debe uno antes asegurarse que el archivo resida en el disco con el que se está trabajando.

Al iniciar la ejecución del programa, como primera parte se solicita la entrada de la lista inicial de cláusulas. Se usa un recuadro en la esquina superior derecha, para señalar la simbología a utilizar en la escritura de las cláusulas. Los símbolos junto con su significado son:

SIMBOLO	SIGNIFICADO
~	Operador lógico negación.
()	Para encerrar el valor de una cláusula.
U,V,W,X,Y,Z	Letras iniciales para denotar a las variables.

La notación utilizada para expresar a los predicados es:
(Signo Letra_Predical Argumento$_1$ Argumento$_2$ ... Argumento$_n$)
Si la letra predical está negada, en la posición de signo se pone el símbolo ~, o no se pone nada en otro caso. Se hereda la misma notación para los argumentos, si estos contienen funciones.
Por ejemplo:

Q(A,B)	es denotado como:	(Q A B)
~P(X,A,F(Y))	se denota como:	(~ P X A (F Y))
R(C,F(C))	se denota por:	(R C (F C))

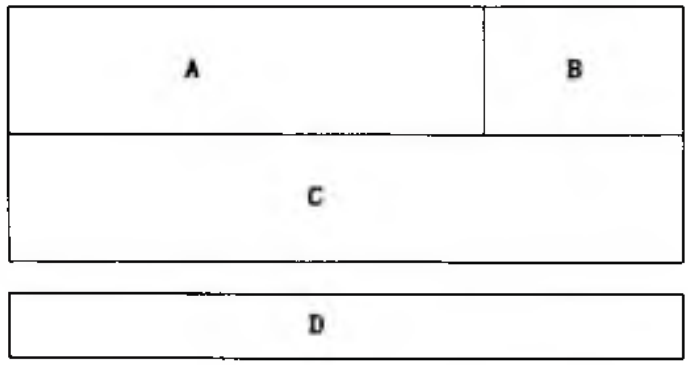
Para indicar el final de la lista de cláusulas, en lugar de escribir el valor de una de ellas se escribe el caracter '.' usado para indicar el final de la lista de entrada.

No es necesario colocar explicitamente al operador OR '$\vee$' entre los predicados, tal operador es implicitamente entendido que existe entre cada par de predicados. Por ejemplo, si la lista inicial se tratara de las cláusulas siguientes.
P(A) $\vee$ ~Q(X,C)
~P(X) $\vee$ ~Q(Z,B) $\vee$ Q(C,D)
~P(Y) $\vee$ ~P(A) $\vee$ Q(Y,F(A))
Tal lista deberá introducirse de la manera siguiente:

```
( (P A) (~ Q X C) )
( (~ P X)(~ Q Z B)(Q C D) )
( (~ P Y)(~ P A)(Q Y (F A)) )
```

Los resultados del programa son mostrados en pantalla (monitor de la computadora). La pantalla se divide en tres partes denotadas como: Ventanas; A, B y C. Y un área en la parte inferior de la pantalla, señalada en el dibujo por la letra D en donde se imprime el mensaje: **"Press any Key to continue"**.
Usado como mensaje temporal para esperar a que el usuario termine de leer la información de las ventanas y oprima cualquier tecla para mostrar otra pantalla de información.

ORGANIZACION DE LAS PANTALLAS DE SALIDA



La ventana A es usada para indicar la fase (paso del método) corriente. En tanto en la ventana C se despliega la información de los objetos con los que se opera en la misma fase.
Las diferentes fases que pueden ser mostradas por el programa, son:

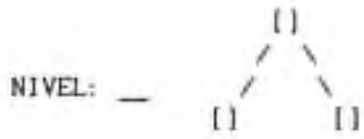
*** Eliminación de literales puros**
Las cláusulas que son eliminadas por contener literales puros son mostradas en la ventana C.

*** Eliminación y Resolución usando literal aislado**
Mostrando en la ventana A, la cláusula unitaria con la que se está trabajando en ese instante. Las cláusulas que son eliminadas o adicionadas a la matriz clausular son desplegadas en la ventana C.

*** División Matricial**
Tanto la literal elegida para resolver, así como las cláusulas usadas para la división matricial, son mostradas en la ventana C.

*** Retroceso al nodo padre**
Cuando ocurre un retroceso, se busca o a una nueva literal o nuevas cláusulas con las cuales intentar la división matricial. Esta nueva información es impresa en la ventana C.

La ventana B tiene el uso exclusivo de indicar al usuario en que nivel del árbol matricial se encuentra trabajando. En el nivel cero dibuja a un nodo (representado por: []), y si el nivel es mayor que cero el dibujo cambia a:



Cuando uno de los dos nodos es falla (la matriz que representa es inconsistente), el dibujo del nodo izquierdo es remarcado. Y si ambos nodos son falla, ambos nodos del dibujo son remarcados, y después es decrementado en uno el nivel del árbol. Si el conjunto original de cláusulas es inconsistente, todo el dibujo del árbol se remarca y un letrero parpadeante conteniendo el mensaje de la inconsistencia es mostrado en la ventana C.

En lo que continua se presenta una sesión con el programa, probandose el método en un ejemplo que aparece en el libro de José Cuena [CUE], en donde es tratado usando el procedimiento de Resolución y Unificación.

```
FASE                                          SIMBOLOGIA

CAPTURA DE LA LISTA INICIAL DE CLAUSULAS      ~  NEGACION
                                              () LIMITA CLAUSULA
AL TERMINAR DE ESCRTBIR CLAUSULA OPRIMIR      U,V,W,X,Y,Z
<ENTER>.FINAL DE LISTA OPRIMIR <.>             LETRA INICIAL PARA
                                               DENOTAR VARS.

D1$= ( (P X Y)(Q W Z) )
D2$= ( (~ P A B)(~ Q B C)(R (F X) C) )
D3$= ( (P A Y)(Q (F U) B)(R (F X) Y) )
D4$= ( (~ P X1 (F U))(Q U X2) )
D5$= ( (~ P B (F A))(Q A (F X))(S A C) )
D6$= ( (P W (F Z))(~ Q A B) )
D7$= ( (~ P A Z)(~ Q B W) )
D8$= .
```

Esta pantalla corresponde a la captura de la lista inicial de cláusulas. Las pantallas restantes indicarán el paso del método en el que se esta trabajando es ese instante, cada una de las pantallas es autoexplicativa, por lo que nos ahorraremos comentarios sobre ellas.

```
FASE                                          OPERANDO NODO
                                                DEL  ARBOL
ELIMINACION DE CLAUSULAS CON LITERAL PURO

                                              NIVEL 0  []
   NOMBRE     VALOR

D2$ VALOR: ( ((~ A B)) ((~ B C)) (((F X) C)) ((0)) )
D3$ VALOR: ( ((A X1))) (((F U) B)) (((F X) X$1)) ((0)) )
D5$ VALOR: ( ((~ B (F A))) ((A (F X))) ((0)) ((A C)) )
```

FASE	OPERANDO NODO DEL ARBOL
DIVISION MATRICIAL	[] NIVEL 0

LITERAL + FRECUENTE: P

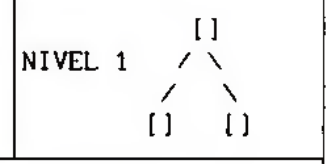
CLAUSULAS ELEGIDAS PARA DIVIDIR LA MATRIZ

D1$= (((X Y)) ((W Z)) ((0)) ((0)))

D2$= (((~ X1 (F U))) ((U X2)) ((0)) ((0)))
EN LOS ARGUMENTOS ((X Y) (~ X1 (F U)))

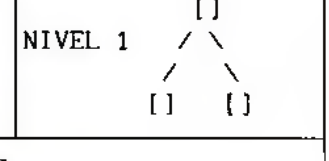
FASE	OPERANDO NODO DEL ARBOL
DIVISION MATRICIAL	[] NIVEL 1 / \ [] []

AUMENTA EN UNO LA PROFUNDIDAD
DEL ARBOL MATRICIAL

FASE	OPERANDO NODO DEL ARBOL
ELIMINACION Y RESOLUCION USANDO LITERAL AISLADO TRABAJANDO CON LA CLAUSULA: D12$ (((0)) ((X$10 X$9)) ((0)) ((0)))	

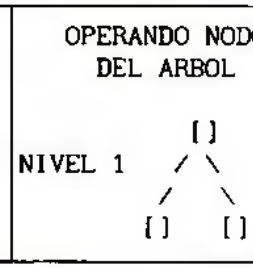
ADICIONANDO LAS CLAUSULAS RESOLVENTES

NOMBRE	VALOR	RESUELTA CON
D13$	(((X$11 (F Z))) ((0)) ((0)) ((0)))	D6$
D14$	(((~ A X$12)) ((0)) ((0)) ((0)))	D7$

FASE	OPERANDO NODO DEL ARBOL
ELIMINACION Y RESOLUCION USANDO LITERAL AISLADO TRABAJANDO CON LA CLAUSULA: D9$ (((~ X$5 (F U))) ((0)) ((0)) ((0)))	

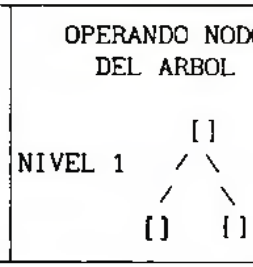
ADICIONANDO LAS CLAUSULAS RESOLVENTES

NOMBRE	VALOR	RESUELTA CON
D15$	(((0)) ((~ A B)) ((0)) ((0)))	D6$

FASE	OPERANDO NODO DEL ARBOL
ELIMINACION Y RESOLUCION USANDO LITERAL AISLADO TRABAJANDO CON LA CLAUSULA: D15$ (((0)) ((~ A B)) ((0)) ((0)))	

ELIMINACION POR INCLUSION A LAS CLAUSULAS

D6$ VALOR: ((X$2 (F Z))) ((~ A B)) ((0)) ((0)))

FASE	OPERANDO NODO DEL ARBOL
ELIMINACION Y RESOLUCION USANDO LITERAL AISLADO TRABAJANDO CON LA CLAUSULA: D15$ (((0)) ((~ A B)) ((0)) ((0)))	

ADICIONANDO LAS CLAUSULAS RESOLVENTES

NOMBRE	VALOR	RESUELTA CON
D16$	(((0)) ((0)) ((0)) ((0))	D12$

EL NODO CORRIENTE ES NODO FALLA

FASE	OPERANDO NODO DEL ARBOL
ELIMINACION Y RESOLUCION USANDO LITERAL AISLADO TRABAJANDO CON LA CLAUSULA: D10$ (((0)) ((X$7 X$6)) ((0)) ((0)))	[] NIVEL 1 / \ / \ [] []

ADICIONANDO LAS CLAUSULAS RESOLVENTES

NOMBRE	VALOR	RESUELTA CON
D17$	(((X$13 (F Z))) ((0)) ((0)) ((0)))	D6$
D18$	(((~ A X$14)) ((0)) ((0)) ((0)))	D7$

FASE	OPERANDO NODO DEL ARBOL
ELIMINACION Y RESOLUCION USANDO LITERAL AISLADO TRABAJANDO CON LA CLAUSULA: D11$ (((X$8 (F U))) ((0)) ((0)) ((0)))	[] NIVEL 1 / \ / \ [] []

ELIMINACION POR INCLUSION A LAS CLAUSULAS

D6$ VALOR: (((X$2 (F Z))) ((~ A B)) ((0)) ((0))

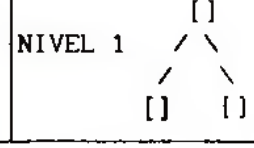
FASE

ELIMINACION Y RESOLUCION USANDO LITERAL AISLADO

TRABAJANDO CON LA CLAUSULA: D11$

(((X$8 (F U))) ((0)) ((0)) ((0)))

OPERANDO NODO DEL ARBOL



ADICIONANDO LAS CLAUSULAS RESOLVENTES

NOMBRE	VALOR	RESUELTA CON
D19$	(((0)) ((~ B X$15)) ((0)) ((0)))	D7$

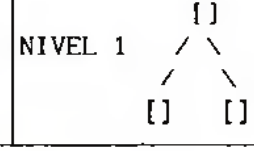
FASE

ELIMINACION Y RESOLUCION USANDO LITERAL AISLADO

TRABAJANDO CON LA CLAUSULA: D19$

(((0)) ((~ B X$15)) ((0)) ((0)))

OPERANDO NODO DEL ARBOL



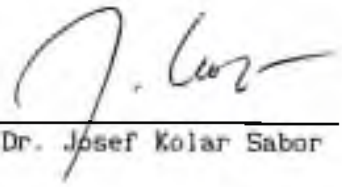
ADICIONANDO LAS CLAUSULAS RESOLVENTES

NOMBRE	VALOR	RESUELTA CON
D20$	(((0)) ((0)) ((0)) ((0)))	D10$

EL NODO CORRIENTE ES NODO FALLA

El jurado designado por la Sección de Computación del Departamento de Ingeniería Eléctrica del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, aprobó esta tesis el seis de Abril de 1989.

Dr. Guillermo Morales Luna



Dr. Josef Kolar Sabor

Dr. Ernesto López Mellado

AUTOR ITA LUNA, G. de

TITULO METODO RM PARA LA DEMOSTRACION AUTOMATICA DE. .

CLASIF. XM 89.4 RGTRO. BI 11160

NOMBRE DEL LECTOR	FECHA PREST.	FECHA DEVOL.
Raúl Hdez. S+P	4/00/[illegible]	13 NOV [illegible]
[illegible]	9/[illegible]	30 NOV 90
[illegible]	30-[illegible]	[illegible]
Mario Velázquez G.	7/I/94	18.II/94
Dr. Guillermo Morales L.	30-11-95	[illegible]