A.R. CAVALLI*

RESUMEN

En este trabajo se define un método de decision para la logica temporal lineal. Este método se define como una extension del principio de resolucion a los operadores temporales y posee la forma de un conjunto de reglas recursivas que estan basadas en inferencias tautologicas de la logica temporal.

Se realiza una comparacion con el método de los "tableaux semanticos" dando el grafo correspondiente al "tableau" y el arbol de resolucion que se obtiene aplicando el método de resolucion definido por nosotros, mostrando el caracter semantico del primero y sintactico del segundo. La aplicacion de los tableaux semanticos a la verificacion de propiedades de vivacidad ("liveness properties") de los programas, exige establecer ciertas restriccionnes sobre los modelos, mientras que el método de resolucion que opera por transformaciones de las formulas no plantea esta limitacion.

* Docteur d'Etat es Sciences (Université Paris VII, 1984), logicas y modelos para la especificacion y verificación de sistemas concurrentes, Escuela de Computëcion - Universidad Central de Venezuela (U.C.V.), Caracas, y actualmente LITP (Laboratoire d'Informatique Théorique et Programmation), Université de Paris VII, 2 Place Jussieu, 75251 PARIS Cedex 05.

La logica temporal puede ser considerada como un instrumento adecuado para la especificación de programas paralelos.

En efecto, en los programas paralelos, no se puede obtener el resultado final del calculo a partir del analisis de las valores de entrada-salida. La ejecución en paralelo puede producir interferencias que modifican el resultado final del calculo, o sea, que el analisis de los programas concurrentes debe tener en cuenta las secuencias de ejecución.

La logica temporal que describe situaciones que cambian en el tiempo, permite formalizar toda la ejecución de un programa y no solamente la relación de entrada-salida.

La interrelación de los programas paralelos plantea nuevos problemas a resolver : la ausencia de bloqueo de los procesos, ya sea bajo la forma de ausencia de bloqueo del sistema global o de cada uno de los procesos ; la sincronización de los procesos, la progresión del sistema, etc...

Surge entonces la necesidad de crear lenguajes que permitan la expresion de este tipo de problemas.

Es para dar respuesta a los problemas planteados por la especificación y verificación de programas paralelos que la logica temporal ha sido estudiada bajo tres aspectos :

- como instrumento de expresion y de descripción de propriedades de programas. Se trata de responder a la pregunta : el lenguaje de la lógica temporal permite expresar de manera exhaustiva todo lo que es verdadero en un modelo lineal arbitrario ?

Dar una respuesta a dicha pregunta es de primera importancia ya que podemos representar la ejecución de un programa como una secuencia de estados.

- como instrumento deductivo

Se trata de responder à la pregunta : Tal propriedad invariante o eventual, expresada por la formula ϕ de la logica temporal, es verdadera para un programa dado ?

La respuesta lleva a desarrollar sistemas deductivos para la lógica temporal, como los desarrollados por Owicki-Lamport, Manna-Pnueli, etc... Estos sistemas deductivos son, en general, completos y consistentes.

-como instrumento de décision

En este caso la pregunta es : la formula ϕ que expresa la especificacion de un programa, posee un modelo ?

Los métodos que han sido desarrollados para dar una respuesta a esta pregunta se basan en métodos de decisión que han sido desarrollados para la lógica clasica . Estos métodos se definen ya sea aplicando un punto de vista "sémantico", es decir, tratando de construir un modelo de la formula ϕ. Este enfoque ha dado lugar a los "semantic tableaux" como método de decision. O aplicando un punto de vista "sintáctico", es decir, aplicando operaciones sobre la fórmula que preservan su satisfabilidad. Si obtenemos la expresión vacia, entonces la formula es insatisfacible, dado que la expresión vacia no posee modelo.

Este enfoque, que se basa en el principio de resolución para la logica clásica, nos ha permitido definir el método de decisión que vamos a exponer en este trabajo y que vamos a comparar con el método de los "semantic tableaux".

El método se define como una extensión de la resolución clásica a los operadores temporales y posee la forma de un conjunto de reglas recursivas. Este articulo esta organizado de la siguiente manera : en la seccion 1 damos el sistema de logica temporal que utilizamos y une breve explicacion del fundamento de nuestras reglas ; en la seccion 2 las reglas que definen el método de resolucion ; la seccion 3 explica de manera suscinta el método de los tableaux semanticos comparandolo con el método de resolucion y, finalmente, en la seccion 4 damos las conclusiones.

1 - LOGICA TEMPORAL

El sistema de lógica temporal que utilizamos es el sistema de lógica temporal lineal definido por Manna y Pnueli [Ma - Pn] . Este sistema supone los siguientes axiomas y reglas de inferencia :

A_1 - $\Vdash$ $\Box$ $(A \supset B) \supset (\Box A \supset \Box B)$

A_2 - $\Vdash$ $O(A \supset B) \supset (OA \supset OB)$

A_3 - $\Vdash$ $O \neg A \equiv \neg O A$

A_4 - $\Vdash$ $\Box A \supset A \,\&\, O A \,\&\, O \Box A$

A_5 - $\Vdash$ $\neg \Box A \equiv \Diamond \neg A$

A_6 - $\Vdash$ $\Box(A \supset OA) \supset (A \supset \Box A)$

A_7 - $\Vdash$ A Until $B \equiv B \vee (A \,\&\, O(A \text{ Until } B))$

A_8 - $\Vdash$ A Until $B \supset \Diamond B$

Règle R_1 : si A es una tautologia entonces $\Vdash A$
Règle R_2 : si $\Vdash A$ y $\Vdash A \supset B$ entonces $\Vdash B$ (Modus Ponens)
Règle R_3 : si $\Vdash A$ alors $\Vdash \Box A$ (generalización)

Previo a la definición de las nuevas reglas, consideremos la formulacion del principio de resolución para la lógica clásica :

"Dadas dos cláusulas C_1 y C_2, si existe un literal L_1 en C_1 que es complementario de un literal L_2 en C_2, entonces eliminamos L_1 y L_2 de C_1 y C_2 respectivamente, y construimos la disyuncion de las clausulas restantes".

Por ejemplo, si $C_1 = \neg p \vee C'_1$ y $C_2 = p \vee C'_2$, aplicando el principio de resolucion obtenemos

$$\frac{\neg p \vee C'_1 \qquad p \vee C'_2}{C'_1 \vee C'_2}$$

Podemos considerar que el principio de resolucion se basa en la inferencia tautologica

$$\frac{\neg C'_2 \supset p \qquad p \supset C'_1}{\neg C'_2 \supset C'_1}$$

y que se formula de la primer manera para aplicarlo a formulas en forma normal.

Ahora bien si consideramos formulas temporales de tipo $\Box p$, $\Diamond q$, $0\ p$, p Until q, donde p, q, son literales y la definicion $\Box A = \neg \Diamond \neg A$, donde A es una formula temporal, podemos definir las siguientes reglas de resolucion:

1) $$\frac{\Box p \vee C'_1 \qquad \Diamond \neg p \vee C'_2}{C'_1 \vee C'_2}$$

que estara basada en la inferencia tautologica

$$\frac{\neg C'_1 \supset \Box p \qquad \Box p \supset C'_2}{\neg C'_1 \supset C'_2}$$

2) $$\frac{\Box p \vee C'_1 \qquad \Box \neg p \vee C'_2}{C'_1 \vee C'_2}$$

basada en la inferencia tautologica

$$\frac{\neg C'_1 \supset \Box p \qquad \Diamond p \supset C'_2}{\neg C'_1 \vee C'_2}$$

Suponemos aqui el axioma $\Box A \supset \Diamond A$.

3) $$\frac{\Box p \vee C'_1 \qquad \neg p \vee C'_2}{C'_1 \vee C'_2}$$

basado en la inferencia tautologica

$$\frac{\neg C'_1 \supset \Box p \qquad p \supset C'_2}{\neg C'_1 \supset C'_2}$$

Suponemos aqui el axioma $\Box A \supset A$.

4) $$\frac{\Box p \vee C'_1 \qquad 0 \neg p \vee C'_2}{C'_1 \vee C'_2}$$

basada en la inferencia tautologica

$$\frac{\neg C'_1 \supset \Box p \qquad 0\ p \supset C'_2}{\neg C'_1 \supset C'_2}$$

Suponemos aqui el axioma $\Box p \supset 0\ p$ y $0 \neg A \equiv \neg 0 A$.

5) $$\frac{\Diamond A \vee C'_1 \qquad \Diamond B \vee C'_2}{\Diamond (A \,\&\, \Diamond B) \vee \Diamond (B \,\&\, \Diamond A) \vee C'_1 \vee C'_2}$$

donde A, B poseen la forma de una conjuncion de clausulas.

Esta regla se basa en la formula valida:

$\Diamond A \,\&\, \Diamond B \supset \Diamond(A \,\&\, \Diamond B) \vee \Diamond(B \,\&\, \Diamond A)$, que podemos llamar "axioma de linearidad".

En el caso del operador Until, tendremos reglas de tipo

$$\frac{p \text{ Until } q \vee C'_1 \qquad \Box \neg q \vee C'_2}{C'_1 \vee C'_2}$$

basada en la inferencia tautologica:

$$\frac{\neg C'_1 \supset p \text{ Until } q \qquad \Diamond q \supset C'_2}{\neg C'_1 \supset C'_2}$$

Suponemos aqui el axioma $p \text{ Until } q \supset \Diamond q$.

$$\frac{p \text{ Until FALSO}}{\text{FALSO}}$$

La definicion de nuestro método exige también la utilizacion de reglas de transformacion de formulas. Estas reglas se basaran en equivalencias entre formulas de la logica temporal.

Las reglas seran

$$\frac{\Box A}{\Box\Box A} \qquad \text{y}$$

$$\frac{\Diamond A}{A \vee \Diamond(\neg A \,\&\, 0\, A)}$$

lo cual nos permitira aplicar las reglas de resolucion a clausulas que expresan la induccion.

Por ejemplo, si tenemos el conjunto de clausulas insatisfacible

p
$\Box(\neg p \vee 0\, p)$
$\Diamond \neg p$

Las clausulas p y $\Box(\neg p \vee 0\, p)$ expresan una induccion, ya que $(p \,\&\, \Box(p \supset 0\, p)) \supset \Box p$. El conjunto de las tres clausulas es inconsistente pero no podemos obtener una refutacion aplicando las reglas mencionadas anteriormente. Para obtener una refutacion, vamos a definir la transformacion del operador $\Diamond$ de manera mas general:

$$\Diamond A = A \vee 0\, A \vee \ldots \vee 0^{n-1} A \vee \Diamond(\neg A \,\&\, 0 \neg A \,\&\, \ldots \,\&\, 0^{n-1} \neg A \,\&\, 0^n A)$$

donde el valor de n esta dado por el numero de operadores 0 (siguiente) que aparecen en la clausula que expresa la induccion.

El método que vamos a definir es sintactico en el sentido de que no existe referencia a la nocion de modelo. Es por esta razon que debemos definir la nocion de forma normal para las formulas de la logica temporal.

Mas aún, los elementos de una forma normal seran cerrados en relacio a las reglas de resolución. En efecto, podemos considerar las reglas de resolución como operaciones sobre cláusulas, es decir, que si consideramos una formula F en forma normal conjunctiva

$F = \&_{i=1}^{m} C^i$, donde cada C^i es una cláusula

y R una regla de n argumentos, entonces

$$R(C_1, \ldots, C_n) = C$$

donde C debe ser una cláusula.

La forma normal sera definida como una extensión a los operadores temporales de la forma normal conjunctiva clásica. Asi, una fórmula F estará en forma normal conjunctiva si ella posee la forma de una conjunción de cláusulas donde cada cláusula es una disjunción, ya sea de literales, o de literales precedidos por une secuencia finita de operadores O, o de expresiones de la forma $\Box A$ (donde A posee la forma de una cláusula) ; o de expresiones de la forma $\Diamond A$ (donde A posee la forma general de una conjunción de cláusulas) ; o de expresiones de la forma A Until B, donde A posee la forma general de una cláusula y B la forma general de une conjunción de cláusulas.

Tenemos un teorema de la forma normal que establece que para toda formula F de la logica temporal existe una formula F', en forma normal conjunctiva, equivalente a F. [Ca]

Previamente a la definicion del metodo, damos un procedimiento de reducción, que constituye en el mismo un método de decision para la lógica temporal. [Ca]

Este procedimiento esta basado en el trabajo de Carnap [Car], y reduce el problema de decision para un conjunto de formulas en forma normal, al problema de decision para un conjunto de formulas en forma normal, que se obtienen a partir de las primeras, pero que poseen un número mas pequeno de operadores temporales.

Este procedimiento de reducción es téoricamente correcto pero ineficaz desde el punto de vista de la automatización. Los operadores temporales son olvidados en el procedimiento de reducción y no se puede saber cual era el conjunto de partida. Se necesitan reglas que memoricen las reducciones realizadas.

Estas reglas serán definidas de una manera analoga a la de la resolución clásica en el sentido siguiente :

el objetivo de la resolución clásica es el de eliminar dos literales complementarios de dos cláusulas dadas, por analogia, el objetivo de la resolución temporal sera el de eliminar dos elementos complementarios en cada etapa de una refutación.

En lógica clásica, basta con aplicar una sola regla, la regla Σ_i (p, $\neg$ p), pero en logica temporal sera necesario aplicar más de una regla para poder eliminar literales complementarios y poder obtener la cláusula vacia, en el caso de que el conjunto sea insatisfacible.

La expresión $\Sigma_i(C_1, C_2)$ significa que aplicamos la regla Σ_i a las cláusulas C_1 y C_2. Si eliminamos C_1 y C_2, por aplicación de la regla Σ_i, esto significa que el conjunto $\{C_1, C_2\}$ es inconsistente. En el mismo sentido, la expresión $\Gamma_i(C_1)$ significa que aplicamos las reglas Γ_i a subformulas de C_1. Si, como resultado de la aplicación de las reglas eliminamos C_1, esto significa que C_1 es inconsistente.

Nuestro método de resolución tendra la forma de un conjunto de reglas de tipo $\Sigma_i(A,B)$, $\Gamma_j(C)$, donde A, B y C son cláusulas. Las reglas serán definidas de manera recursiva.

En la siguiente sección, damos las nuevas reglas de resolución que definen el método de resolución para la logica temporal lineal.

2. METODO DE RESOLUCION PARA LA LOGICA TEMPORAL LINEAL

Sean C_1 y C_2 dos cláusulas. Definimos recursivamente reglas de la forma

$\Sigma_i((C_1, C_2)$ $\quad i = 1,\ldots,6$

$\Gamma_j(C_1)$ $\quad j = 1,2$

de la siguiente manera :

2.1 Reglas clásicas

a) $\Sigma_1(p, \neg p)$

$\emptyset$ ($\emptyset$ indica la clausula vacia)

- p será denominado un literal resuelto

b) $\Sigma_1((D_1 \vee D_2, F)$

$\Sigma_i(D_1, F) \vee D_2$

para $i \in \{1,\ldots,6\}$. En general, se chequea para que valor(es) de i se aplica la regla.

En este caso $D_1 \vee D_2$ no es una clausula unitaria, pero esta regla se aplicara a subformulas.

La regla puede ser aplicada de manera simétrica, es decir, podemos aplicarle a D_2 y F, obteniendo la expresion $\Sigma_i(D_2, F) \vee D_1$.

c) $\Gamma_1(D_1 \,\&\, D_2 \,\&\, F) = \Sigma_i(D_1, D_2) \,\&\, F)$

2.2 Reglas Temporales

Reglas del operador $\Box$

a) Sea $C_1 = \Box E$ y $C_2 = \Delta F$ entonces definimos

$\Sigma_2(\Box E, \Delta F)$

$\Delta\Sigma_i(E, F)$ donde Δ es $\Box$, $\Diamond$ o 0, para $i \in \{1,\ldots,6\}$.

b) Sea $C_1 = \Box E$ y $C_2 = F$, definimos

$$\frac{\Sigma_2(\Box E, F)}{\Sigma_i(E,F)} \quad \text{para } i \in \{1,\ldots,6\}$$

Reglas del operador $\Diamond$

c) Sea $C_1 = \Diamond E$ y $C_2 = \Diamond F$ definimos

$$\frac{\Sigma_3 (\Diamond E, \Diamond F)}{\Sigma_i (E, \Diamond F) \vee \Sigma_j (\Diamond E, F) \quad \text{para } i,j \in \{1,\ldots,6\}}$$

d) Sea $C_1 = \Diamond (D_1 \,\&\, D_2 \,\&\, F)$ definimos

$$\frac{\Diamond (D_1 \,\&\, D_2 \,\&\, F)}{\Diamond (\Sigma_i(D_1,D_2) \,\&\, F) \quad \text{para } i \in \{1,\ldots,6\}}$$

Reglas del operador 0

e) Sea $C_1 = 0 E$ y $C_2 = O F$ definimos

$$\frac{\Sigma_4 (0 E, 0 F)}{0\, \Sigma_i (E, F) \quad \text{para } i \in \{1,\ldots,6\}}$$

Reglas del operador Until

f) Sea $C_1 = E$ Until D y $C_2 = F$, definimos

$$\frac{\Sigma_5 (E \text{ Until } D, F)}{E \text{ Until } \Sigma_i(D, F)}$$

g) Sea $C_1 = E$ Until D y $C_2 = F$, definimos

$$\frac{\Sigma_5 (E \text{ Until } D, F)}{\Sigma_i(E,F) \text{ Until } D}$$

2.3 Reglas de Transformacion

a) $$\frac{\Sigma_6 (\Box E , F)}{\Sigma_i(\Box \Box E, F) \quad \text{para } i \in \{1,\ldots,6\}}$$

b) $$\frac{\Sigma_6 (\langle\rangle E, F)}{\Diamond_n E \vee \Sigma_i (\Diamond_{n-1} (\Box \neg E \wedge O^n E), F) \quad \text{para } i \in \{1,6\}}$$

donde

$$\Diamond_n E = E \vee O E \vee \ldots \vee O^{n-1} E$$

$$\Box_{n-1} \neg E = E \wedge O E \wedge \ldots \wedge O^{n-1} E$$

y para un conjunto de clausulas S, $n \leq k$, donde k es el numero de operadores 0 que aparecen en S.

2.4 Clausulas Resolubles

Definicion. Sean C_1 y C_2 dos clausulas, diremos que C_1 y C_2 son *resolubles* (respectivamente, C_1 es *resoluble*) si podemos aplicar una de las reglas Σ_i, i= 1,...,6 (respectivamente una de las reglas Γ_i, i=1,2).

2.5 Reglas de simplificacion

Definicion. Sean C_1 y C_2 dos clausulas resolubles (respectivamente C_1 una clausula resoluble), entonces una clausula $R(C_1, C_2)$ (respectivamente $R(C_1)$) se denomina una *resolvente* de C_1 y C_2 (respectivamente de C_1) si ella es el resultado de sustituir:

Ø	por	toda	ocurrencia	de (Ø & E)
E	"	"	"	(Ø v E)
Ø	"	"	"	Δ Ø, donde Δ es $\Box$, $\Diamond$ o 0^j, j>=0;
E	"	"	"	Ø Until E
Φ	"	"	"	E Until Ø

en Σ_i (C_1, C_2), (Γ_j (C_1)), $i \in \{1,...,6\}$, $j \in \{1,2\}$, tantas veces como sea posible.

2.6 Etapas de la resolucion

2.6.1 Sean C_1 v C y C_2 v C' dos clausulas. La regla de inferencia

i) $\dfrac{C_1 \text{ v } C \qquad C_2 \text{ v } C'}{R(C_1, C_2) \text{ v } C \text{ v } C'}$

sera aplicada si C_1 y C_2 son resolubles.

ii) $\dfrac{C_1 \text{ v } C}{R(C_1) \text{ v } C}$

sera aplicada si C_1 es resoluble.

2.6.2 Sea E(DvDvF) una clausula. La siguiente regla sera aplicada

iii) $\dfrac{E(D \text{ v } D \text{ v } F)}{E(D \text{ v } F)}$

donde, en general , E(E') indica que E' es una subformula de E.

2.7 Deduccion y Refutacion

Definicion. Sea S un conjunto de clausulas. Una *deduccion* de C a partir de S es una secuencia finita de clausulas $C_1,...,C_n$, tales que:
-C_n es C
-C_i ($1 \leq i \leq n$) es :
- una clausula de S o,
- una clausula obtenida a partir de C_j, $j < i$ utilizando las reglas de inferencia ii) o iii) o
- una clausula obtenida a partir de C_j y C_k $j,k < i$, utilizando la regla de inferencia i).

Una deduccion de la clausula vacia se llama una *refutacion*.

El método de resolucion para la logica temporal lineal es completo para el sistema axiomatico definido por los axiomas y reglas de los operadores □, ◇, O y es "sound" para el sistema que incluye los axiomas del operador Until.

El teorema de completitud posee la formulacion siguiente : un cojunto S de clausulas es insatisfacible si y solamente si S es refutable.

3. LOS TABLEAUX SEMANTICOS

El método de los tableaux semanticos para la logica temporal se basa en el método de los tableaux para la logica clasica [Be]. En este trabajo no daremos los detalles de dicho método pero en sintesis podemos decir que este método procede de la manera siguiente : descompone una formula en sus componentes elementales, es decir, en formulas atomicas (una variable proposicional o su negacion) o en aquéllos que poseen O como conectivo principal.

Esta descomposicion permite separar las exigencias formuladas por la formula en exigencias sobre el estado inicial (formulas atomicas) y en exigencias sobre el resto de la secuencia (las formulas que poseen O como conectivo principal).

A partir del procedimiento de descomposicion, se aplica un procedimiento de construccion de un grafo modelo. Este grafo representa todos los modelos potenciales de la formula y del mismo se podra extraer un modelo de la formula.

Una vez construido el grafo se realizara un procedimiento de chequeo y eliminacion para determinar si todos los modos del grafo son satisfacibles y si todas las eventualidades pueden ser realizadas.

Una formula sera satisfacible si el procedimiento de chequeo y eliminacion, suponiendo cada vez que una sola proposicion atomica es verdadera, no produce el grafo vacio.

3.1 En lo que sigue, vamos a comparar el método de los tableaux semanticos y el método de résolucion que hemos definido en 2.

Los ejemplos han sido elegidos tomando como punto de partida los dados por Wolper [Wo].

Ejemplo : Daremos como ejemplo un conjunto de clausulas satisfacible, porque si partimos de un conjunto insatisfacible, el procedimiento de descomposicion nos daria el conjunto vacio. Partimos del conjunto $S = \{\Box (q \supset \neg (p \text{ Until } r)), \Box q\}$, aplicamos el procedimiento de descomposicion suponiendo que la formula atomica q es verdadera.

Los pasos en el procedimiento de descomposicion seran los siguientes

1 - $\Sigma q = \{\{\Box (q \supset (\neg p \text{ Until } r)), \Box q\}$
2 - $\Sigma q = \{\{q \supset (\neg p \text{ Until } r), O \Box (q \supset (\neg p \text{ Until } r)), \Box (q \supset (\neg p \text{ Until } r))^*$
q, $O \Box q, \Box q ^*\}$

Simplificacion por q.

3 - Σq = {{r, ({¬ p Until r)*, O □ (q ⊃ (¬ p Until r)), □ (q ⊃ (¬ p Until r))*, O □ q, □ q *}, {¬ p ∧ O (¬ p Until r))*, (¬ p Until r)*, O □ (q ⊃ (¬ p Until r)), □ (q ⊃ (¬ p Until r))* O □ q, □ q *}}

por simplificacion por q.

4 - {¬ p ∧ O(¬ p Until r)*, (¬ p Until r)*, O □ (q ⊃ (¬ p Until r), □ (q ⊃ (¬ p Until r))*, O □ q, □ q *}}

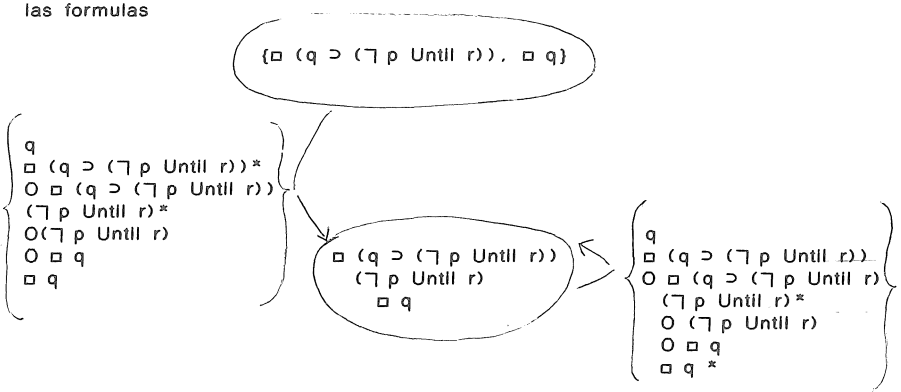
5 - {O (¬ p Until r), (¬ p Until r)*, O □ (q ⊃ (¬ p Until r)), □ (q ⊃ (¬ p Until r))*, O □ q, □ q *}}

El procedimiento de descomposicion suponiendo la proposicion atomica p verdadera, se detiene en el segundo paso, dado que en el conjunto que se obtiene la proposicion q sera falsa, lo que hara que el conjunto sea insatisfacible. Lo mismo sucede cuando suponemos r verdadera :

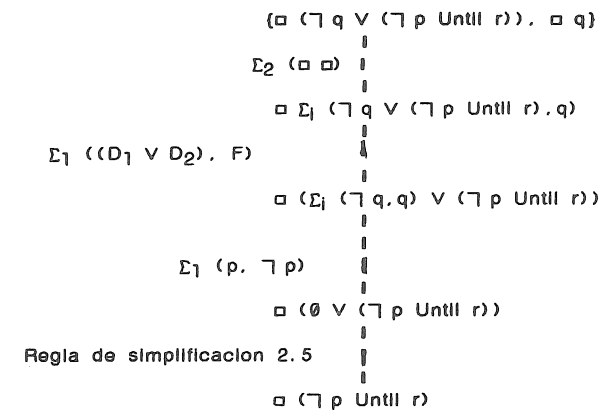
1 - Σp = {{□(q ⊃ (¬ p Until r)), □ q}}

2 - Σp = {q ⊃ (¬ p Until r)), O □ (q ⊃ (¬ p Until r)), □ (q ⊃ (¬ p Until r))* q, o □ q}}

Una vez realizado el procedimiento de descomposicion, se construye el grafo modelo, a partir del cual se obtendran todos los modelos posibles de las formulas



Si aplicamos el método de resolucion definido en la seccion 2, el arbol de resolucion sera el siguiente, previa transformacion de la formula □ (q ⊃ (¬ p Until r)) en forma de clausula :



4. CONCLUSION

La aplicacion del método de resolucion aparece como mas simple, sobre todo cuando se trata de decidir la satisfabilidad de formulas donde aparecen eventualidades, es decir, formulas gobernadas por ◇ o por Until, dado que p Until q ⊃ ◇ q.

El método de los tableaux semanticos permite obtener un modelo de la formula : cada secuencia que es un modelo sera un camino en el grafo y cada camino finito que se obtiene a partir del grafo es el prefijo de algun modelo.

Pero, como lo senalan los mismos autores [Ma - Wo], no todos los caminos infinitos satisfacen a las formulas. Sobre todo cuando se trata de formulas con eventualidades, estas pueden no ser verificadas. Para resolver este problema, se podrian elegir los caminos del grafo que satisfacen las eventualidades, pero esto llevaria a obtener un programa que poseera como ejecuciones posibles solamente algunas de las secuencias que satisfacen las especificaciones. A su vez, esto lleva a los procesos involucrados a interactuar siguiendo un esquema prefijado.

Por ejemplo, si las especificaciones son

$$\Box \Diamond a \wedge \Box \Diamond b$$

el camino elegido podria producir la secuencia a, b, a, b, a, b... como modelo. Lo que es correcto pero inaceptable en situaciones donde a, por ej., puede repetirse mucho mas rapido que b.

Obtener un modelo en el cual se pueda decidir de manera dinamica como resolver las eventualidades solo se puede lograr con ciertas hipotesis de equidad y restricciones sobre las formulas.

Se define un criterio que chequea que en todos los caminos infinitos, o bien la eventualidadad se realiza, o bien, infinitamente a menudo, es "posible", lo que bajo la hipotesis de equidad, garantiza que la eventualidad sera realizada.

El método de resolucion aplicando transformaciones sintacticas sobre las formulas, demuestra que las especificaciones son consistentes con respecto a ciertas propriedades, particularmente, a propiedades donde estan involucradas eventualidades.

La ventaja de nuestro método con respecto al método de los tableaux sémanticos, es que no necesitamos construir un modelo para probar que las especificaciones verifican las propriedades deseadas.

Esta caracteristica sera de primera importancia cuando se trate de demostrar propiedades de vivacidad de un sistema, incluyendo la evolucion, no bloqueo, individual y global, etc.

La parte del método que corresponde a las reglas de los operadores $\Box$, $\Diamond$, $\bigcirc$ ha sido implementada bajo la forma de un demonstrador PROTEMP, en el laboratorio Lenguajes y Sistemas, de la Univ. Paul Sabatier, Toulouse.

Bibliografia

[Be] Beth E.W., The foundations of Mathematics, North Holland 1959.

[Ca] Cavalli A.R., Une méthode de décision pour la Logique Temporelle Linéaire. Application à la spécification et vérification des protocoles, Thèse de Doctorat d'Etat, Université de PARIS VII, Diciembre 1984.

[Ca-Fa] Cavalli A.R. y Farinas del Cerro L., A decision mezthod for linear temporal logic, Proceedings of the 7th International Conference on Automated Deduction, Napa, California, USA, Springer Verlag No 170, May 1984.

[Car] Carnap R., Modalities and quantification, JSL Vol 11, 1946, pp33-64.

[La-Owl] Lamport L. and Owicki S., Proving liveness properties of concurrent programs, ACM Transactions on programming languages and systems, Vol. 4, No 3, July 1982.

[Ma-Pn] Manna Z. y Pnueli A., Verification of concurrent programs: the temporal framework. In the Correcteness Problem in Computer Science, Academic Press New York, 1981, pp 215-273.

[Ma-Wo] Manna Z. y Wolper P., Synthesis of Communicating Processes from Temporal Logic Specifications, ACM Transactions on Programming, Languages and Systems, Vol 6, No 1, January 1984, pp 68-93.

[Ro] Robinson J., A machine oriented logic based on the resolution principle, J. ACM, 12, 1965, 23-41.

[Wo] Wolper P., Specification and synthesis of communicating processes using an extended temporal logic, Proceedings of the 9th ACM Symposium on Principles of Programming Languages (Albuquerque, N.M. Jan 25-27), ACM, New York, 1982, pp 20-33.