Capítulo 4

Estrategias

La administración de la regla de resolución en los sistemas de demostración automática de teoremas ha sido utilizada con dos técnicas esencialmente:

- la técnica de gestión de conjuntos de cláusulas
- la técnica de exploración del árbol de las deducciones

No obstante, cada una de estas técnicas ha dado lugar a numerosas variantes.

En general, dada una técnica nos interesa de ella su *corrección*, su *completitud* y su *eficiencia*. Una técnica es *correcta* cuando no conduce jamás a considerar como insatisfacible un conjunto de cláusulas que es satisfacible. Una técnica es *completa* cuando termina revelando que es insatisfacible el conjunto que trata, si realmente lo es. La eficiencia es un concepto relativo.

4.1. Gestión por Saturación

De entre las estrategias, la más simple y natural es la *estrategia de saturación*:

- Partimos de un conjunto de cláusulas Σ_0 del cual buscamos conocer si es satisfacible o no.
- Dado $0 < i$, efectuamos todas las resoluciones y todas las disminuciones posibles a partir de las cláusulas de Σ_{i-1}, lo que genera unas nuevas cláusulas que añadimos a Σ_{i-1} para obtener el conjunto Σ_i.
- Se detiene el proceso cuando $\square$ es un elemento de Σ_i o cuando $\Sigma_i = \Sigma_{i+1}$.

La estrategia de gestión por saturación es completa y correcta según el Teorema 3.2.1. Sin embargo, no es eficiente; en realidad, desde el punto de vista teórico no supone mejora alguna respecto al algoritmo primario sugerido en la Observación 2.5.3 y es profuso en la generación y uso de cláusulas de poca utilidad, de hecho los conjuntos Σ_i crecen de forma exponencial. Ejemplo de ineficiciencia es el tratamiento que da al conjunto $\Sigma_0 = \{p \vee q, \neg p \vee q, p \vee \neg q, \neg p \vee \neg q\}$ (desarrollado en [4], pag. 92 y 93). Además, es posible que siendo Σ_0 satisfacible sin embargo sea Σ_i distinto de Σ_{i+1} para todo $0 \leq i$; es el caso de $\Sigma_0 = \{p(a), \neg p(x) \vee p(f(x))\}$.

Ejemplo 4.1.1. Queremos saber si es insatisfacible el conjunto $\{p, \neg p \vee q, \neg q \vee p, \neg q\}$. Entonces generamos:

- $\Sigma_0 = \{p, \neg p \vee q, \neg q \vee p, \neg q\}$
- $\Sigma_1 = \Sigma_0 \cup \{q, \neg q \vee q, \neg p \vee p, \neg p\}$
- $\Sigma_2 = \Sigma_1 \cup \{p, \square, \ldots\}$

4.2. Gestión por Saturación con Simplificación

Definición 4.2.1. La cláusula $\lambda_1 \vee \cdots \vee \lambda_k$ (en realidad la sentencia $\forall x_1 \cdots \forall x_n(\lambda_1 \vee \cdots \vee \lambda_k)$, no lo olvidemos) distinta de la vacía es una *cláusula tautológica* cuando existen $1 \leq i \neq j \leq k$ tales que $\lambda_i = \lambda_j^c$.

Ejemplo 4.2.1. La cláusula

$$\forall x\big(p(f(x)) \vee q(x) \vee \neg r(x) \vee \neg p(f(x))\big)$$

es tautológica.

Definición 4.2.2. La cláusula σ *subyace bajo* la cláusula τ (o también τ *generaliza* a σ) si existe una sustitución Φ tal que τ es la cláusula $\xi \vee \Phi\sigma$, para cierta ξ.

Ejemplo 4.2.2.

1. $p(f(a)) \vee q(a)$ generaliza a $p(x)$
2. $p(a) \vee q(a) \vee r(f(x)) \vee r(b)$ generaliza a $p(x) \vee q(a)$
3. $\neg p(x) \vee q(f(x), a)$ subyace bajo $\neg p(h(y)) \vee q(f(h(y)), a) \vee \neg p(z)$, basta con considerar la sustitución $(x|h(y))$

Teorema 4.2.1. *Sea Σ un conjunto de cláusulas y Σ' el conjunto obtenido de Σ suprimiendo en él todas las cláusula tautológicas. Son equivalentes las siguientes afirmaciones:*

1. *Σ es satisfacible*
2. *Σ' es satisfacible*

Teorema 4.2.2. *Sea Σ un conjunto de cláusulas sin cláusulas tautológicas y Σ' el conjunto obtenido de Σ suprimiendo cualquier cláusula que generalize a (bajo la que subyazca) otra cláusula del conjunto. Son equivalentes las siguientes afirmaciones:*

1. *Σ es satisfacible*
2. *Σ' es satisfacible*

Los teoremas 4.2.1 y 4.2.2 sugieren una modificación interesante en la estrategia de gestión por saturación explicado anteriormente (ver sección 4.1), que justamente atenua el efecto de generar cláusulas inútiles. Se procede como se indicó en la estrategia de saturación, pero antes de cada etapa de saturación se suprimen las tautologías y las generalizaciones. Esta nueva *estrategia de saturación con simplificación* es correcta, porque lo era la estrategia de saturación, y completa por los teoremas 4.2.1 y 4.2.2.

Ejemplo 4.2.3. Estudiamos la satisfacibilidad del conjunto $\{r(a) \vee p(a), p(x) \vee \neg r(b), r(b), \neg p(b), r(a) \vee r(c) \vee \neg r(a)\}$ y empleamos la estrategia de saturación con simplificación.

- $\Sigma_0 = \{r(a) \vee p(a), p(x) \vee \neg r(b), r(b), \neg p(b), r(a) \vee r(c) \vee \neg r(a)\}$
- $\Sigma_0' = \{r(a) \vee p(a), p(x) \vee \neg r(b), r(b), \neg p(b)\}$
- $\Sigma_1 = \{r(a) \vee p(a), p(x) \vee \neg r(b), r(b), \neg p(b), p(x), \neg r(b)\}$
- $\Sigma_1' = \{r(b), \neg p(b), p(x), \neg r(b)\}$
- $\Sigma_2 = \{\square, \ldots\}$

Hay otras simplificaciones posibles utilizando los "predicados evaluables".

4.3. Gestión por Preferencia de Cláusulas Simples

En el método sugerido en la sección 4.1, mejor que construir de una vez todas las cláusulas que pueden ser obtenidas a partir de Σ_i, podemos no construir más que una sola, "bien elegida", añadirla y después volver a comenzar. No obstante, ¿qué significa "bien elegida"? Es una expresión ambigua que da pie a muchos métodos: se puede proceder al azar; se puede elegir la obtenida en primer lugar, cuando hay un orden establecido entre todas las resolventes y disminuciones posibles (por ejemplo, numerando las cláusalas o los predicados), etc.

Pero entre todas las estrategias, la más natural es sin duda la que para pasar de Σ_i a Σ_{i+1} une la cláusula más corta entre las que se pueden obtener con Σ_i. Esta idea se sustenta en que como queremos obtener la cláusula vacía, alcanzamos nuestro objetivo tanto más rápido como cortas sean las cláusulas que se generan.

Ejemplo 4.3.1.

- $\Sigma_0 = \{r(x) \vee \neg p(x), \neg r(b), \neg r(c) \vee q(x) \vee r(f(a)), p(b)\}$
- $\Sigma_1 = \Sigma_0 \cup \{\neg p(b)\}$
- $\Sigma_2 = \Sigma_1 \cup \{\square\}$

Aunque correcta, esta estrategia usada en toda su pureza no es completa. En efecto, consideremos el siguiente ejemplo:

Ejemplo 4.3.2.

- $\Sigma_0 = \{\neg p(x) \vee p(f(x)), p(a), \neg p(a) \vee \neg p(b) \vee \neg p(c), p(b), p(c)\}$
- $\Sigma_1 = \Sigma_0 \cup \{\neg p(f(a))\}$
- $\Sigma_2 = \Sigma_1 \cup \{\neg p(f(f(a)))\}$

Nunca obtenemos la cláusula vacía, salvo que pasemos por la obtención de una cláusula de dos literales, lo cual da la cláusula vacía en tres etapas.

Para hacer esta estratégia completa podemos, por ejemplo, introducir de cuando en cuando una etapa de saturación.

Entre las variantes posibles, destacamos la estrategia de "preferir las cláusulas unitarias" en la que se efectuan prioritariamente todas las resoluciones que hacen intervenir a una cláusula unit. Tal cual esta estrategia no es completa.

4.4. Exploración. Conceptos Generales

Definición 4.4.1. Sea Σ un conjunto de cláusulas de un lenguaje de primer orden. El *árbol de deducciones* de Σ es el árbol etiquetado —en nodos— con raíz (la raíz es un símbolo ajeno al lenguaje, por ejemplo $\bullet$), tal que todo segmento inicial suyo es, en cuanto a la lista ordenada de sus etiquetas excluyendo la raíz, una Σ demostración por resolución. Una *estrategia de exploración* de árboles estará determinada fijando (cfr. [5] para ver ejemplos):

- un subárbol del árbol de las deducciones, normalmente esto se hará imponiendo limitaciones sobre las deducciones que se habrán de tener en cuenta.
- el procedimiento de exploración del subárbol :
  - *primero en profundidad con retroceso* al nodo anterior: ello permite llegar a un nodo profundo más rápidamente, se programa fácilmente y no emplea demasiado espacio de memoria. Si no se fija profundidad, este método de exploración se pierde en la primera rama infinita que alcanza y no permite la exploración entera de un árbol finito. Si en la exploración se fija una profundidad corremos el riesgo de no encontrar un objetivo en una rama por no haber profundizado lo suficiente.
  - *primero en anchura*: ello permite alcanzar cualquier nodo con tal de disponer de tiempo suficiente. La exploración se puede hacer lenta si los nodos tienen en general muchos hijos, con lo cual puede ser muy costoso en tiempo profundizar en el árbol.

Las estrategias generales son análogas a la estrategia de saturación. Consisten simplemente en explorar un árbol en el que han sido incluídas todas las demostraciones por resolución. Si la exploración se hace primero en anchura se obtiene una estrategia correcta y completa. Si se hace primero en profundidad se obtiene una estrategia correcta pero no completa. Estas estrategias son muy ineficaces, más aún que las de saturación, porque consideran diferentes las deducciones que no cambian más que el orden de las fórmulas. Por ello es necesario restringirse en la búsqueda dentro del árbol global a un subárbol suyo en que el aparece la cláusula vacía.

4.5. Estrategias Lineales

Definición 4.5.1. Sea Σ un conjunto de cláusulas, σ_0 un elemento de Σ y γ una cláusula. Una Σ-*demostración lineal* de γ con raíz en σ_0 es una Σ-demostración de γ

$$< \gamma_0, \beta_0, \gamma_1, \beta_1 \ldots, \gamma_{n-1}, \beta_{n-1}, \gamma_n >$$

donde:

- γ_0 es σ_0,
- para todo $0 \leq i \leq n-1$, γ_{i+1} es obtenido por resolución a partir de γ_i (denominada *cláusula central*) y β_i (denominada *cláusula lateral*), o a partir de algún factor de alguna de ellas o de cada una de ellas.
- Cada β_i es un elementos de Σ o existe $j < i$ tal que $\beta_i = \gamma_j$.

Ejemplo 4.5.1. Tomado de [4]. Consideremos el conjunto

$$\begin{aligned}\Sigma = \{&r(a, f(c), f(b)),\\ &p(a),\\ &r(x, x, f(x)),\\ &\neg r(x, y, z) \vee r(y, x, z),\\ &\neg r(x, y, z) \vee q(x, z),\\ &\neg p(w) \vee \neg r(x, y, z) \vee \neg q(w, z) \vee q(w, x) \vee q(w, y),\\ &\neg q(a, b)\}\end{aligned}$$

y sea $\sigma_0 = \neg q(a, b)$. La siguiente es una Σ-demostración lineal con raíz σ_0:

γ_0: $\neg q(a, b)$

β_0: $\neg p(w) \vee \neg r(x, y, z) \vee \neg q(w, z) \vee q(w, x) \vee q(w, y)$

γ_1: $\neg p(a) \vee \neg r(x, b, z) \vee \neg q(a, z) \vee q(a, x)$

β_1: $\neg q(a, b)$

γ_2: $\neg p(a) \vee \neg r(b, b, z) \vee \neg q(a, z)$

β_2: $r(x, x, f(x))$

γ_3: $\neg p(a) \vee \neg q(a, f(b))$

β_3: $p(a)$

γ_4: $\neg q(a, f(b))$

β_4: $\neg r(x, y, z) \vee q(x, z)$

γ_5: $\neg r(a, y, f(b))$

β_5: $r(a.f(c), f(b))$

γ_6: $\square$

Ejemplo 4.5.2. Tomado de [5]. Consideremos el conjunto $\Sigma = \{\neg a \vee \neg b, a \vee \neg c, c, b \vee \neg d, d \vee b\}$ y sea $\sigma_0 = \neg a \vee \neg b$. La siguiente es una Σ-demostración lineal con raíz σ_0:

γ_0: $\neg a \vee \neg b$

β_0: $a \vee \neg c$

γ_1: $\neg b \vee \neg c$

β_1: c

γ_2: $\neg b$

β_2: $b \vee \neg d$

γ_3: $\neg d$

β_3: $d \vee b$

γ_4: b

β_4: $\neg b$

γ_5: $\square$

No obstante, no es ésta la única Σ-demostración lineal con raíz en σ_0, pues también existe la siguiente:

γ_0: $\neg a \vee \neg b$

β_0: $a \vee \neg c$

γ_1: $\neg b \vee \neg c$

β_1: $d \vee b$

γ_2: $\neg c \vee d$

β_2: $b \vee \neg d$

γ_3: $b \vee \neg c$

β_3: c

γ_4: b

β_4: $\neg b \vee \neg c$

γ_5: $\neg c$

β_5: c

γ_5: $\square$

El árbol de todas la Σ-demostraciones lineales con raíz σ_0 es ya demasiado grande, aunque se podría visualizar una parte (cfr. [5], pag. 175)

Se ha establecido (cfr. [4] y [8]) que si existe una demostración por resolución de la cláusula vacía a partir de Σ, entonces existe un Σ-demostración lineal de la cláusula vacía.

Más aún, si $\Sigma = \Sigma' \cup \{\sigma_0\}$ con Σ' satisfacible y Σ insatisfacible, entonces existe una Σ-demostración lineal de la cláusula vacía con raíz σ_0. Esto quiere decir que buscamos saber si el literal τ es consecuencia de Σ', basta con recorrer el árbol de las demostraciones lineales con raíz $\sigma_0 = \tau^c$.

Si se utiliza una estrategia de exploración "primero en anchura" estamos seguros de obtener la cláusula vacía de estar (estrategia completa). Por contra si se utiliza la estrategia de "primero en profundidad con retorno al nodo anterior" podemos hundirnos en una rama infinita y no encontrar la cláusula vacía, aunque esté. El ejemplo más conocido es es siguiente:

Ejemplo 4.5.3. Sea $\Sigma = \{\neg p(x) \vee p(f(x)), \neg p(a), p(x)\}$ y tomemos como ráiz a $\sigma_0 = p(x)$. En el árbol de las Σ-demostraciones lineales con raíz σ_0 se encuentran estas dos:

γ_0: $p(x)$

β_0: $\neg p(a)$

γ_1: $\square$

y "ésta" otra tan larga como se quiera, pero nunca concluyente:

γ_0: $p(x)$

β_0: $\neg p(w) \vee p(f(w))$

γ_1: $p(f(x))$

β_1: $\neg p(w) \vee p(f(w))$

γ_2: $p(f(f(x)))$

β_2: $\cdots$

4.6. Estrategia "Input"

Definición 4.6.1. Sea Σ un conjunto de cláusulas, σ_0 un elemento de Σ y γ una cláusula. Una Σ-*demostración input* de γ con raíz en σ_0 es una Σ-demostración lineal de γ

$$< \gamma_0, \beta_0, \gamma_1, \beta_1 \ldots, \gamma_{n-1}, \beta_{n-1}, \gamma_n >$$

donde para todo $1 \leq i$, γ_i es obtenida por una resolución en la cual una de las cláusulas padre es de Σ.

Ejemplo 4.6.1. Consideremos el conjunto de cláusulas

$$\begin{aligned}\Sigma = \{&\neg p(x,y,u) \vee \neg p(y,z,v) \vee \neg p(x,v,w) \vee p(u,z,w),\\ &p(g(x,y),x,y),\\ &p(x,h(x,y),y),\\ &\neg p(k(x),x,k(x))\}\end{aligned}$$

Tomando como σ_0 la cláusula $\neg p(x,y,u) \vee \neg p(y,z,v) \vee \neg p(x,v,w) \vee p(u,z,w)$ tenemos la siguiente refutación input de Σ.

γ_0: $\neg p(x,y,u) \vee \neg p(y,z,v) \vee \neg p(x,v,w) \vee p(u,z,w)$

β_0: $p(g(x,y),x,y)$

γ_1: $\neg p(y,z,v) \vee \neg p(g(y,u),v,w) \vee p(u,z,w)$

β_1: $p(g(x,y),x,y)$

γ_2: $\neg p(v,z,v) \vee p(w,z,w)$

β_2: $p(x,h(x,y),y)$

γ_3: $p(w,h(v,v),w)$

β_3: $\neg p(k(x),x,k(x))$

γ_4: $\square$

Ejemplo 4.6.2. Sea $\Sigma = \{\neg a, a \vee \neg b, a \vee \neg c \vee \neg d, c, d \vee \neg c\}$ y $\sigma_0 = \neg a$. La Σ-demostración lineal siguiente es input:

γ_0: $\neg a$

β_0: $a \vee \neg c \vee \neg d$

γ_1: $\neg c \vee \neg d$

β_1: c

γ_2: $\neg d$

β_2: $d \vee \neg c$

γ_3: $\neg c$

β_3: c

γ_4: $\square$

De hecho no hay más Σ-demostraciones input con raíz en σ_0 que la anterior y

γ_0: $\neg a$

β_0: $a \vee \neg b$

γ_1: $\neg b$

El Ejemplo 4.6.2 muestra como, aún en un caso sencillo en el que el árbol resulta finito, no es posible prescidir del "retroceso al nodo anterior" en la estrategia de "primero en profundidad" si se quiere encontrar la cláusula vacía.

Contrariamente al caso general de la estrategia lineal, incluso una estrategia de exploración "primero en anchura" no resulta completa. Es el caso de $\Sigma = \{a \vee b, \neg a \vee b, a \vee \neg b, \neg a \vee \neg b\}$ que el árbol de las deducciones input no tiene a la cláusula vacía mientras que el árbol de todas las deducciones sí la tiene.

Definición 4.6.2. Una cláusula distinta de la vacía es una *cláusula de Horn* si posee exactamente un literal positivo. Un *conjunto de Horn* es un conjunto de cláusulas cada de una de las cuales es una cláusula de Horn. Una cláusula no vacía es *negativa* si cada uno de sus literales es negativo.

Teorema 4.6.1. *Sea $\Sigma = \Sigma' \cup \{\sigma_0\}$ un conjunto insatisfacible de cláusulas. Si σ_0 es negativa y Σ' es un conjunto de Horn, entonces existe una Σ-demostración input de $\square$ con raíz σ_0 para la que no se utiliza la regla de disminución.*

Demostración: Consultar [6] ■

Definición 4.6.3. Sea Σ un conjunto de cláusulas, σ_0 un elemento de Σ y γ una cláusula. Una Σ-*demostración unit* de γ con raíz en σ_0 es una Σ-demostración lineal de γ

$$< \gamma_0, \beta_0, \gamma_1, \beta_1 \ldots, \gamma_{n-1}, \beta_{n-1}, \gamma_n >$$

donde para todo $1 \leq i$, γ_i se obtiene por resolución a partir de al menos una cláusula padre unit o un factor unit de alguna cláusula padre.

Teorema 4.6.2. *Existe una refutación unit de un conjunto Σ de cláusulas si, y sólo si, existe una refutación input de Σ.*

Demostración: Se puede ver en [4] ■

Ejemplo 4.6.3. Se pide demostrar que la fórmula $\exists z r(z)$ es consecuencia de las hipótesis: $\forall x(p(x) \to p(f(x)))$, $\forall y((p(f(y)) \wedge p(y)) \to r(y))$ y $p(a)$. En términos de cláusulas, este problema es equivalente a saber si es insatisfacible o no el conjunto $\Sigma = \Sigma' \cup \{\sigma_0\}$, donde:

$$\Sigma' = \{\neg p(x) \vee p(f(x)), \neg p(f(y)) \vee \neg p(y) \vee r(y), p(a)\}$$

y $\sigma_0 = \neg r(z)$. Encontramos la siguiente refutación unit (e input):

γ_0: $\neg r(z)$

β_0: $\neg p(f(y)) \vee \neg p(y) \vee r(y)$

γ_1: $\neg p(f(y)) \vee \neg p(y)$

β_1: $p(a)$

γ_2: $\neg p(f(a))$

β_2: $\neg p(x) \vee p(f(x))$

γ_3: $\neg p(a)$

β_3: $p(a)$

γ_4: $\square$

Observación 4.6.1. La consecuencia esencial del Teorema 4.6.1 es que para la clase de los problemas de la forma $\Sigma = \Sigma' \cup \{\sigma_0\}$, con Σ' un conjunto de cláusulas de Horn y σ_0 una cláusula negativa, la estrategia de exploración "primero en anchura" del árbol de deducciones input de raíz σ_0 es una estrategia correcta y completa.

La estrategia de exploración "primero en profundidad con retorno al nodo anterior" del mismo árbol es correcta y completa para todos los problemas antes citados que engendren un árbol de deducción input finito (y por tanto, para todos los problemas sin variables), pero no es completo en general.

Observación 4.6.2. Observar que cualquier sentencia de la forma

$$\varphi_1 \wedge \cdots \wedge \varphi_n \to \psi \tag{4.1}$$

se escribe de forma equivalente como

$$\neg\varphi_1 \vee \cdots \vee \neg\varphi_n \vee \psi$$

que bien podría originar una cláusula de Horn. Por otra parte, toda sentencia de la forma

$$\psi_1 \wedge \cdots \wedge \psi_m \tag{4.2}$$

al ser negada se convierte en

$$\neg\psi_1 \vee \cdots \vee \neg\psi_m$$

que bien podría originar una cláusula negativa. Casi todos los problemas naturales de implicación tienen por hipótesis sentencias (en cantidad finita) de la forma 4.1 y como tesis una fórmula de la forma 4.2. Además, la naturalidad del problema hace que las hipótesis "tengan modelo". Esto realza la importancia del Teorema 4.6.1.

4.7. Estrategias Ordenadas

Existen numerosas formas de tener en cuenta el orden de los literales en las cláusulas para limitar las demostraciones tomadas en consideración dentro del árbol de las deducciones (cfr. [8] y [4]). Describimos aquí la más simple.

En la *resolución ordenada* las cláusulas son cosideradas como sucesiones finitas de literales (sus literales están ordenados). Para poder llevar a cabo la resolución entre dos cláusulas tienen que concurrir todas las circunstancias que requiere la aplicación de la regla de resolución; pero además, la resolución tiene que ser practicada con los literales que están en cabeza de las cláusulas. Más aún, la resolvente, que debe estar ordenada, debe llevar en las primeras posiciones los literales que resultan de la cláusula resolvente con el literal positivo y después los literales de la otra. Se denomina *demostración ordenada* a toda demostración en la que siempre que se resuelve se utiliza la resolución ordenada.

Ejemplo 4.7.1. Es posible la resolución ordenada entre las cláusulas $p(x) \vee r(x) \vee \neg q(x, y)$ y $\neg p(a) \vee \neg r(f(a))$ y da como resultado $r(a) \vee \neg q(a, y) \vee \neg r(f(a))$. Sin embargo, no es posible la resolución ordenada entre $p(x) \vee r(x) \vee \neg q(x, y)$ y $\neg r(f(a)) \vee \neg p(a)$.

Ejemplo 4.7.2. Sea $\Sigma = \{a \vee b, \neg a \vee c, \neg b \vee d, \neg c, \neg d\}$. La siguiente es una Σ-demostración ordenada de $\square$ con raíz $a \vee b$:

γ_0: $a \vee b$

β_0: $\neg a \vee c$

γ_1: $b \vee c$

β_1: $\neg b \vee d$

γ_2: $c \vee d$

β_2: $\neg c$

γ_3: d

β_3: $\neg d$

γ_4: $\square$

Teorema 4.7.1. *Sea $\Sigma = \Sigma' \cup \{\sigma_0\}$ un conjunto insatisfacible de cláusulas, donde σ_0 es negativa y Σ' es un conjunto de cláusulas de Horn ordenadas colocando el literal positivo en cabeza de las cláusulas. Entonces existe una refutación input ordenada de raíz σ_0.*

Demostración: Ver, por ejemplo, [7] o [1]. ■

Ejemplo 4.7.3. Sea

$$\Sigma' = \{p(x) \vee \neg r(x), p(x) \vee \neg q(f(x)) \vee \neg q(x), q(f(x)) \vee \neg q(x), q(a)\}$$

y $\sigma_0 = \neg p(a)$. El árbol de las demostraciones input ordenadas de raíz σ_0 no tiene más que dos ramas, una no es una refutación:

γ_0: $\neg p(a)$

β_0: $p(x) \vee \neg r(x)$

γ_1: $\neg r(a)$

y la otra sí lo es:

γ_0: $\neg p(a)$

β_0: $p(x) \vee \neg q(f(x)) \vee \neg q(x)$

γ_1: $\neg q(f(a)) \vee \neg q(a)$

β_1: $q(f(x)) \vee \neg q(x)$

γ_2: $\neg q(a) \vee \neg q(a)$

β_2: $q(a)$

γ_3: $\neg q(a)$

β_3: $q(a)$

γ_4: $\square$

Observación 4.7.1. En resumen, para la clase de problemas de la forma $\Sigma = \Sigma' \cup \{\sigma_0\}$, donde Σ' es un conjunto de cláusulas de Horn y σ_0 es una cláusula negativa, la estrategia de exploración "primero en anchura" del árbol de las deducciones input ordenadas de raíz σ_0 es una estrategia correcta y completa. La estrategia de exploración "primero en profundidad con retorno" del mismo árbol es completa y correcta para todos los problemas antes citados que engendren un árbol de deducciones input ordenadas finito.

Observación 4.7.2. El lenguaje *Prolog* está fundado exactamente en el principio de la Observación 4.7.1: no se pueden escribir más que *cláusulas de Horn* y cuando es propuesta una cuestión, se pone en marcha un algoritmo de exploración "primero en profundidad con retorno" del árbol de deducciones input ordenadas de raíz la negación de la cuestión propuesta. Si el árbol es finito, está asegurada por lo dicho la obtención o bien de la cláusula vacía —es decir, un éxito— si la cuestión propuesta admite una respuesta positiva, o bien un fallo si —cuando todo el árbol es recorrido sin ser encontrada la cláusula vacía— si la cuestión propuesta admite una respuesta negativa. Dicho de otra forma, la estrategia de utilización de la resolución de Prolog es incompleta en general y completa cuando el árbol de deducciones particular engendrado es finito.

4.8. Estrategias Input Ordenadas y Extracción de Respuestas

Consideremos el ejemplo siguiente:

$$\Sigma' = \{p(a), p(b), r(f(x)) \vee \neg p(x), q(y) \vee \neg r(y), q(c)\}$$
$$\sigma_0 = \neg q(z)$$

El árbol de deducciones tiene tres deducciones cada una de las cuales conduce a la cláusula vacía (bibujar el árbol como ejercicio). Una de ellas es la siguiente:

γ_0: $\neg q(z)$

β_0: $q(y) \vee \neg r(y)$ (sutitución: $(y|z)$)

γ_1: $\neg r(z)$

β_1: $r(f(x)) \vee \neg p(x)$ (sutitución: $(z|f(x))$)

γ_2: $\neg p(x)$

β_2: $p(a)$ (sutitución: $(x|a)$)

γ_3: $\square$

En el curso de la demostración por resolución la variable z es reemplazada por $f(x)$ y seguidamente x por a; así pues z es reemplazada por $f(a)$. Por tanto, $\Sigma' \cup \{\neg q(f(a))\}$ conduce a la cláusula vacía por resolución:

γ_0: $\neg q(f(a))$

β_0: $q(y) \vee \neg r(y)$

γ_1: $\neg r(f(a))$

β_1: $r(f(x)) \vee \neg p(x)$

γ_2: $\neg p(a)$

β_2: $p(a)$

γ_3: $\square$

lo que significa que $q(f(a))$ resulta de Σ'. Por la misma razón, pero utilizando las otras refutaciones $q(f(b))$ y $q(c)$ resultan de Σ'. Se constata pues que cuando se consideran cláusulas con variables (como $\neg q(z)$), cada demostración de la cláusula vacía da un objeto t (un elemento del universo de Herbrand) tal que $q(t)$ resulta de Σ'. Esto es general y se tiene el resultado siguiente.

Teorema 4.8.1. *Sea $\Sigma' \cup \{\sigma_0(x_1, \ldots, x_n)\}$ un conjunto insatisfacible de cláusulas tal que todas las cláusulas de Σ' son de Horn ordenadas anteponiendo el literal positivo y*

$$\begin{aligned}\sigma_0(x_1, x_2, \ldots, x_n) =& \neg\varphi(x_1, x_2, \ldots, x_n) \\ =& \neg\lambda_1(x_1, x_2, \ldots, x_n) \vee \neg\lambda_2(x_1, x_2, \ldots, x_n) \vee \cdots \neg\lambda_k(x_1, x_2, \ldots, x_n)\end{aligned}$$

es negativa. Entonces cada refutación input ordenada en la que las sustituciones son $(x_1|t_1)$, $(x_2|t_2)$,...,$(x_n|t_n)$ define elementos del universo de Herbrand de Σ' tales que:

$$\varphi(t_1, t_2, \ldots, t_n) = \lambda_1(t_1, t_2, \ldots, t_n) \wedge \lambda_2(t_1, t_2, \ldots, t_n) \wedge \cdots \lambda_k(t_1, t_2, \ldots, t_n)$$

es consecuencia de Σ'. Recíprocamente para toda n-upla $\langle t_1, t_2, \ldots, t_n\rangle$ en el universo de Herbrand de Σ', tal que $\varphi(t_1, t_2, \ldots, t_n)$ sea consecuencia de Σ', existe una refutación input ordenada de raíz $\neg\varphi(x_1, x_2, \ldots, x_n)$ donde las sustituciones $(x_1|t'_1)$, $(x_2|t'_2)$,...,$(x_n|t'_n)$ cumplen que $\varphi(t_1, t_2, \ldots, t_n)$ es una instanciación de $\varphi(t'_1, t'_2, \ldots, t'_n)$.[1]

[1] Es decir es obtenida a partir de $\varphi(t'_1, t'_2, \ldots, t'_n)$ reemplazando ciertas variables por términos.

Esto signfica que para saber cuales son los objetos t para los cuales $q(t)$ es consecuencia de Σ' (o lo que es equivalente $q(t)$ es verdadero en todo modelo de Herbrand de Σ'), basta con explorar el árbol de deducciones input ordenadas por una estrategia "primero en profundidad", (o por una estrategia "primero en profundidad con retorno" si se está seguro que el árbol es finito) anotando por cada cláusula vacía que se encuentre el objeto t asociado.

Éste es el principio de los métodos de extracción de respuestas utilizados en numerosos sistemas de demostración automática (cfr. [9]) y en ciertos lenguajes de interrogación de bases de datos. En particular, el lenguaje Prolog está basado en este resultado. Puede decirse que calcula lo que pasa en el más pequeño modelo de Herbrand de Σ', el conjunto de cláusulas que definen el programa (cfr. [7]).