Capítulo 1

Algoritmo de unificación de Robinson

1.0 Introducción

Con objeto de encontrar el unificador de máxima generalidad de dos términos se han propuesto numerosos algoritmos siendo el más conocido el de Robinson. El algoritmo de unificación de Robinson data del año 1965.

La claridad del método utilizado por el algoritmo lo hace muy útil, tanto desde un punto de vista didáctico como desde un punto de vista de implementación.

Vamos a exponer el método en un cuadro general y posteriormente lo aplicaremos a varios ejemplos.

Para la mejor comprensión del algoritmo damos unas nociones previas que serán de utilidad.

1.1 Elementos para el algoritmo de unificación

Entre los elementos que vamos a considerar se encuentran el llamado *primer par de discordancia* entre dos términos, y la aparición de variables dentro de un término *occur check* (comprobación de apariciones).

1.1.0 Primer par de discordancia

Tomemos dos términos que no sean iguales. Eso significa que habrá alguna diferencia entre ellos. Esta diferencia puede hallarse en el nombre del funtor, en el número de argumentos, o en alguno de los argumentos.

Si la diferencia está en el nombre del funtor, el primer par de discordancia es la pareja formada por los dos términos. Si está en el número de argumentos, al igual que antes, el primer par de discordancia resulta ser el formado por los dos términos considerados.

En cambio, si coinciden en los funtores y en el número de argumentos, el primer par de discordancia debe ser buscado entre sus argumentos empezando a considerar éstos de izquierda a derecha. Se busca en el primer argumento, y si aquí no lo hubiera, se busca en el segundo, ... Es evidente que al final se debe encontrar este par de discordancia pues hemos partido de dos términos distintos.

Como se ve, hemos dado una definición recursiva de lo que es un par de discordancia. Veamos algunos ejemplos.

Ejemplo 1.0 Sean los términos

$$p(a, f(b, c), g(X))$$
$$p(a, f(X, Y), g(h(Z))$$

según lo dicho, el primer par de discordancia es b y X encontrados dentro del segundo argumento como primer argumento. Obsérvese que no hemos tomado $f(b, c)$ y $f(X, Y)$ como primer par de discordancia pues aplicando la definición dada, hemos de introducirnos en este argumento ya que coinciden sus funtores y número de argumentos. Ejemplo

Un par de discordancia suele expresarse entre llaves. Así el par de discordancia del ejemplo anterior es $\{b, X\}$.

Ejemplo 1.1 Sean los términos

$$p(g(a, b, c), X)$$
$$p(f(X, a), Z)$$

En este caso, el primer par de discordancia es $\{g(a, b, c), f(X, a)\}$. Ejemplo

Como conclusión podemos decir que siempre es posible encontrar un par de discordancia entre dos términos distintos y que, en el peor de los casos, estará formado por esos dos términos.

Sólo cuando dos términos son iguales, no podemos encontrar el primer par de discordancia. Cualquier algoritmo que realice la búsqueda del primer par de discordancia debe primero asegurarse de que no está ante dos términos iguales.

1.1.1 La comprobación de apariciones

Con la comprobación de apariciones (*occur check*) simplemente queremos detectar cuándo una variable aparece dentro de un término. Si bien desde un punto de vista formal esto no ofrece ninguna complicación, sí la ofrece cuando se intenta sistematizar este procedimiento.

La razón simplemente está en que es tedioso tener que buscar por cada argumento y dentro del mismo la aparición de una variable.

En cualquier caso, sólo se trata de eso. Veamos un ejemplo.

Ejemplo 1.2 La variable X se encuentra dentro de los términos $p(a, f(b, X))$ y $q(X, g(X, Y))$ pero no de los términos $p(a, Z, g(a, b)$ y $q(a, b)$. Ejemplo

1.2 El algoritmo

Sean E y F dos términos que queremos unificar. Consideramos inicialmente $\sigma_0 = \{\}$ una sustitución vacía, es decir, que no cambia ninguna variable. Dado que vamos a realizar un proceso iterativo, consideramos inicialmente $E_0 = \sigma_0(E)$ y $F_0 = \sigma_0(F)$. En cada iteración k del algoritmo se realizan los siguientes pasos:

Paso 1 Si $E_k = F_k$ entonces las cláusulas E y F son unificables y un unificador de máxima generalidad es $\sigma = \sigma_k \circ \cdots \circ \sigma_0$. Además, el término E_k es el término unificado. En este caso el proceso termina aquí.

Paso 2 Si $E_k \neq F_k$ entonces se busca el primer par de discordancia entre E_k y F_k. Sea éste D_k.

Paso 3 Si D_k contiene una variable y un término (pueden ser dos variables y una de ellas hace de término) pasamos al siguiente paso. En otro caso los términos no son unificables y terminamos el proceso.

Paso 4 Si la variable aparece en el término se produce un *occur check* por lo que E y F no unifican y terminamos. Si esto no ocurre pasamos al siguiente paso.

Paso 5 Construimos una nueva sustitución que vincule la variable con el término de D_k. Sea esta sustitución σ_{k+1}. Construimos ahora dos nuevos términos $E_{k+1} = \sigma_{k+1}(E_k)$ y $F_{k+1} = \sigma_{k+1}(F_k)$ y volvemos al paso 1.

Este algoritmo siempre termina para dos términos cualesquiera. Si los términos no eran unificables terminará indicándolo así y si eran unificables devolverá un unificador de máxima generalidad y el término resultante unificado. Veamos algunos ejemplos.

Ejemplo 1.3 Sean los términos

$$p(a, X)$$
$$p(X, Y)$$

aplicando el algoritmo paso a paso tenemos

Sea $E = p(a, X)$ y $F = p(X, Y)$ y sea $\sigma_0 = \{\}$. Consideremos también $E_0 = p(a, X)$ y $F_0 = p(X, Y)$

Iteración 1 Desarrollamos para $k = 0$

Paso 1 Como $E_0 \neq F_0$ vamos al paso 2.

Paso 2 $D_0 = \{a, X\}$. Vamos al paso 3.

Paso 3 En D_0 uno es un término (a) y el otro una variable (X). Vamos al paso 4

Paso 4 La variable X no aparece en el término a. Vamos al paso 5.

Paso 5 Sea $\sigma_1 = \{X/a\}$. Sean también $E_1 = \sigma_1(E_0) = p(a, a)$ y $F_1 = \sigma_1(F_0) = p(a, Y)$. Volvemos al paso 1.

Iteración 2 Desarrollamos para $k = 1$

Paso 1 Como $E_1 \neq F_1$ vamos al paso 2.

Paso 2 $D_1 = \{a, Y\}$. Vamos al paso 3.

Paso 3 En D_1 uno es un término (a) y el otro una variable (Y). Vamos al paso 4.

Paso 4 La variable Y no aparece en el término a. Vamos al paso 5.

Paso 5 Sea $\sigma_2 = \{Y/a\}$. Sean también $E_2 = \sigma_2(E_1) = p(a, a)$ y $F_2 = \sigma_2(F_1) = p(a, a)$. Vamos al paso 1.

Iteración 3 Desarrollamos para $k = 2$

Paso 1 Como $E_2 = F_2$ el algoritmo ***termina*** con $\sigma = \sigma_2 \circ \sigma_1 \circ \sigma_0 = \{X/a, Y/a\}$ como unificador de máxima generalidad.

Ejemplo

Ejemplo 1.4 Sean $E = E_0 = p(a, f(X, b), Y)$ y $F = F_0 = p(X, f(g(Y), Z), T)$ y sea $\sigma_0 = \{\}$.

Iteración 1 Desarrollamos para $k = 0$.

Paso 1 $E_0 \neq F_0$. Vamos al paso 2.

Paso 2 $D_0 = \{X, a\}$. Vamos al paso 3.

Paso 3 Término y variable. Vamos al paso 4.

Paso 4 No hay *occur check*. Vamos al paso 5.

Paso 5 Sea $\sigma_1 = \{X/a\}$. Sean $E_1 = \sigma_1(E_0) = p(a, f(a, b), Y)$ y $F_1 = \sigma_1(F_0) = p(a, f(g(Y), Z, T))$. Vamos al paso 1.

Iteración 2 Desarrollamos para $k = 1$.

Paso 1 $E_1 \neq F_1$. Vamos al paso 2.

Paso 2 Par de discordancia $D_1 = \{a, g(Y)\}$. Vamos al paso 3.

Paso 3 atomo y funtor. ***No unifican***.

Ejemplo

Ejemplo 1.5 Sean $E = E_0 = p(a, X, h(g(Z)))$ y $F = F_0 = p(Z, h(Y), h(Y))$. Sea $\sigma_0 = \{\}$. Entonces:

Iteración 1 Para $k = 0$.

Paso 1 $E_0 \neq F_0$. Vamos al paso 2.

Paso 2 $D_0 = \{Z, a\}$. Vamos al paso 3.

Paso 3 Variable y termino. Vamos al paso 4.

Paso 4 No hay *occur check*.Vamos al paso 5.

Paso 5 Sea $\sigma_1 = \{Z/a\}$. Sea $E_1 = p(a, X, h(g(a)))$ y sea $F_1 = p(a, h(Y), h(Y))$. Vamos al paso 1.

Iteración 2 Para $K = 1$.

Paso 1 $E_1 \neq F_1$. Vamos al paso 2.

Paso 2 $D_1 = \{X, h(Y)\}$. Vamos al paso 3.

Paso 3 Variable y termino. Vamos al paso 4.

Paso 4 No hay *occur check*.Vamos al paso 5.

Paso 5 Sea $\sigma_2 = \{X/h(Y)\}$. Sea $E_2 = p(a, h(Y), h(g(a)))$ y $F_2 = p(a, h(Y), h(Y))$. Vamos al paso 1.

Iteración 3 Para $k = 2$.

Paso 1 $E_2 \neq F_2$. Vamos al paso 2.

Paso 2 $D_2 = \{Y, g(a)\}$. Vamos al paso 3.

Paso 3 Variable y termino. Vamos al paso 4.

Paso 4 No hay *occur check*.Vamos al paso 5.

Paso 5 Sea $\sigma_3 = \{Y/g(a)\}$. Sea $E_3 = p(a, h(g(a)), h(g(a)))$ y sea $F_3 = p(a, h(g(a)), h(g(a)))$. Vamos al paso 1.

Iteración 4 Para $k = 3$.

Paso 1 $E_3 = F_3$. ***Unifican***. El algoritmo para aquí y la sustitución completa es $\sigma = \{Z/a, X/h(g(a)), Y/g(a)\}$.

Ejemplo

Ejercicio 1.6 *Unificar, si es posible, siguiendo el algoritmo de Robinson los siguientes términos:*

$p(a, W, X, f(f(X)))$	$p(Z, g(Y), g(Z), f(Y))$
$p(a, X)$	$p(Y, b)$
$p(X, X)$	$p(Y, Z)$
$p(X, Y)$	$p(Y, X)$
$p(t(X, t(X, b)))$	$p(t(a, Z))$
$p(t(X, t(X, b)))$	$p(t(a, t(Z, Z)))$
$p(X, f(X))$	$p(f(Z), f(Z))$
$p(f(a), g(Z))$	$p(Y, Y)$
$p(f(a), g(X))$	$p(Y, X)$
$p(f(a), g(X))$	$p(Y, Z)$

1.3 Algoritmo de unificación en Prolog

El algoritmo de unificación que utiliza Prolog es básicamente igual al aquí descrito excepto que no dispone del *Paso 4*. Esto quiere decir que no verifica nunca el *occur check*. La razón de prescindir de este paso es por cuestión de eficiencia ya que la comprobación de *occur check* consume demasiado tiempo, y hay que realizarla muchas veces.

Por otro lado, no es habitual encontrar situaciones que deriven en programas donde se produzcan *occur checks*.

En cualquier caso, si se presenta, dependiendo de la implementación de Prolog, se comporta de una manera u otra. Hay sistemas Prolog que lo detectan y avisan del hecho, y otros que simplemente dejan de funcionar.

1.4 El programa unifica.pl

Este es un programa escrito en Prolog, y especialmente para *SWI-Prolog*, que permite visualizar los pasos del algoritmo de unificación de Robinson (aur) para dos términos dados.

Para su ejecución hay que escribir simplemente `aur.` en el intérprete y nos muestra un símbolo de entrada de datos `aur>` además de unos comentarios que aclaran su uso:

$?-$ *aur*.

ALGORITMO DE UNIFICACION PASO A PASO
==========

*Term*1 = *Term*2. $->$ *Unificación paso a paso*
fin. $->$ *Fin del programa*
ayuda. $->$ *Esta ayuda*

aur $>$

Para ver los pasos de una unificación simplemente hay que escribir los dos términos a unificar separados por un símbolo de igualdad y terminado en punto. En la figura 1.0 se muestra un ejemplo de ejecución.

$aur >\ p(f(X), a) = p(Z, X).$

$E0 = p(f(X),\ a)$
$F0 = p(Z,\ X)$

Iteración 1
  Paso 1 : *Son distintos*
    $E0 = p(f(X),\ a)$
    $F0 = p(Z,\ X)$
  Paso 2 : *Par de discordancia* $D0 = \{Z, f(X)\}$
  Paso 3 : *Variable y termino. OK*
  Paso 4 : *No hay occur check. OK*
  Paso 5 : $Sigma1 = \{Z/f(X)\}$
    $E1 = p(f(X),\ a)$
    $F1 = p(f(X),\ X)$

Iteración 2
  Paso 1 : *Son distintos*
    $E1 = p(f(X),\ a)$
    $F1 = p(f(X),\ X)$
  Paso 2 : *Par de discordancia* $D1 = \{X, a\}$
  Paso 3 : *Variable y termino. OK*
  Paso 4 : *No hay occur check. OK*
  Paso 5 : $Sigma2 = \{X/a\}$
    $E2 = p(f(a),\ a)$
    $F2 = p(f(a),\ a)$

Iteración 3
  Paso 1 : *Son iguales*
    $E2 = p(f(a),\ a)$
    $F2 = p(f(a),\ a)$
    $Sigma\ = \{Z/f(a),\ X/a\}$
  $- - - - - - - - - - >$ *Unifican*

$Term1\ =\ Term2. - >$ *Unificación paso a paso*
fin. $- >$ *Fin del programa*
ayuda. $- >$ *Esta ayuda*

$aur >$

Figura 1.0: Ejemplo uso del programa