Lógica Computacional 2016-2, nota de clase 8
Unificación de términos

Favio Ezequiel Miranda Perea Araceli Liliana Reyes Cabello
Lourdes Del Carmen González Huesca

7 de abril de 2016

La regla de resolución binaria de Robinson, que veremos más adelante, proporciona un método más para decidir la correctud de un argumento formalizado en lógica de predicados.

Si bien la idea del método es la misma que en la lógica proposicional, es decir llegar a la cláusula vacía para mostrar que un argumento es incorrecto o que una fórmula es válida, en lógica de predicados el método se complica debido a la presencia de variables, por ejemplo y a diferencia de las literales $p, \neg p$ en lógica proposicional, las dos literales $Pax, \neg Pzb$ no pueden resolverse aunque tengan el mismo símbolo de predicado, pues no son contrarias estrictamente. Sin embargo, como veremos después, en esta clase de cláusulas todas las variables están cuantificadas universalmente de manera implicita, por lo que en realidad tenemos $\forall x Pax$ y $\forall z \neg Pzb$ por lo que podemos aplicar la sustitución $\sigma = [x, z := b, a]$ a la fórmula sin cuantificadores, obteniendo como resultado Pab y $\neg Pab$ que sí son literales contrarias. Por lo tanto la búsqueda y uso de sustituciones que permitan obtener literales contrarias será de primordial importancia para usar resolución en lógica de predicados. Esta clase de sustituciones se conocen como unificadores y en esta nota veremos como es posible hallarlas algoritmicamente.

1. Sustituciones

El concepto de sustitución usado hasta ahora ha sido informal, en adelante necesitamos una definición formal discutida aquí.

Definición 1 *Una sustitución en un lenguaje de predicados es una función $\sigma : Var \to Term$ con la propiedad de que $\sigma(x) = x$ para casi todas las variables. Es decir $\sigma(x_i) \neq x_i$ únicamente para un número finito de variables x_i con $1 \leq i \leq n$. Esto se conoce en matemáticas como la propiedad de que una función tenga soporte finito, siendo el soporte de σ el conjunto de variables x_i con $\sigma(x_i) \neq x_i$.*

Obsérvese que una sustitución σ queda completamente determinada mediante su soporte (finito), si éste es $x_1, \ldots, x_n$ con $\sigma(x_i) = t_i$ entonces la substitución se denotará con

$$\sigma = [x_1 := t_1, \ , x_2 := t_2, \ \ldots, x_n := t_n]$$

o de manera más breve $\sigma = [\vec{x} := \vec{t}]$

La composición de sustituciones se define como una composición de funciones de donde se sigue para motivos de implementación la siguiente caracterización

Proposición 1 *Sean $\sigma = [x_1 := t_1, \ , x_2 := t_2, \ \ldots, x_n := t_n]$ $\tau = [y_1 := s_1, \ \ldots, y_m := s_m]$ sustituciones. La composición, denotada $\sigma \circ \tau$ o simplemente $\sigma\tau$, es una sustitución definida por:*

$$\sigma\tau = [z_1 := t_1\tau, \ldots, z_k := t_k\tau, \, w_1 := r_1, \ldots, w_l := r_l]$$

donde $k \leq n, l \leq m$, las z_i son exactamente aquellas x_p tales que $x_p \neq t_p\tau$ y los pares $w_j := r_j$ son aquellos pares $y_q := s_q$ tales que $y_q \notin \{x_1, \ldots, x_n\}$

Demostración. Ejercicio ⊣

Proposición 2 *La composición de sustituciones es asociativa, es decir si σ, ρ, τ son sustituciones entonces $(\sigma\rho)\tau = \sigma(\rho\tau)$.*

Demostración. Ejercicio ⊣

Ejemplo 1.1 Sean

$$\sigma = [x := fy,\ y := w],\ \rho = [x := gw,\ z := b],\ \tau = [y := b,\ w := fc,\ v := w]$$

Entonces

$$\begin{aligned} \sigma\rho &= [x := fy,\ y := w,\ z := b] \\ \rho\tau &= [x := gfc,\ z := b,\ y := b,\ w := fc,\ v := w] \\ (\sigma\rho)\tau &= [x := fb,\ y := fc,\ z := b,\ w := fc,\ v := w] \\ \sigma(\rho\tau) &= [x := fb,\ y := fc,\ z := b,\ w := fc,\ v := w] \end{aligned}$$

2. Unificación

El proceso de unificación consiste en encontrar, dado un conjunto de literales o términos W, una substitución σ de tal forma que el conjunto imagen $W\sigma$ conste de un solo elemento. En adición a sus aplicaciones en programación lógica, la unificación también es importante para los sistemas de reescritura de términos, el razonamiento automátizado y los sistemas de tipos.

Definición 2 *Sea W un conjunto no vacío de términos. Un unificador de W es una substitución σ tal que $|W\sigma| = 1$, es decir tal que el conjunto $W\sigma$ resultante de aplicar σ a todos los elementos de w consta de un solo elemento. Si W tiene un unificador decimos que W es unificable.*

Ejemplo 2.1 Sea $W = \{gxfy, gxfx, guv\}$ entonces la sustitución $\sigma = [x := a,\ y := a,\ u := a,\ v := f(a)\}$ es un unificador de W, ya que $W\sigma = \{gafa\}$

Un conjunto de términos puede tener una infinidad de unificadores o ninguno, dado un conjunto finito de fórmulas W, es decidible mediante un algoritmo si W es unificable o no; si W es unificable el algoritmo proporciona un unificador llamado *unificador más general.*

Definición 3 *Un unificador σ de un conjunto de términos W, se llama unificador más general* **(umg)** *si para cada unificador τ de W, existe una sustitución ϑ, tal que $\sigma\vartheta = \tau$*

La importancia de los unificadores más generales es que nos permiten representar de manera finita un número infinito de sustituciones.

Ejemplo 2.2 Sean $W = \{fgaxgyb, fzguv\}$ con $f^{(2)},\ g^{(2)}$ y

$$\tau = [x := a, z := gaa, y := u, v := b] \qquad \sigma = [z := gax, y := u, v := b].$$

Entonces τ y σ son unificadores de W y es fácil ver que σ resulta ser umg; en particular si $\vartheta = [x := a]$ entonces $\sigma\vartheta = \tau$.

Antes de discutir un algoritmo de unificación es conveniente hacer un análisis intuitivo del problema. Basta analizar un conjunto de dos términos digamos $W = \{t_1, t_2\}$, queremos ver si el conjunto W es unificable.

Hay que analizar varios casos:

- Los términos t_1 y t_2 son constantes. En este caso $t_i\sigma = t_i$ para cualquier sustitución σ, de manera que W será unificable si y sólo si $t_1 = t_2$.
- Alguno de los términos es una variable. Supongamos que $t_1 = x$. Si x figura en t_2 entonces W no es unificable, en caso contrario $\sigma = [x := t_2]$ unifica a W.
- t_1 y t_2 son términos funcionales. En este caso W es unificable si y sólo si se cumplen las siguientes dos condiciones:
  - Los símbolos principales (es decir los primeros), de t_1 y t_2 son el mismo.
  - Cada par correspondiente de subexpresiones de t_1 y t_2 deben ser unificables.

Ejemplo 2.3 El conjunto $\{c, d\}$ no es unificable pues consta de dos constantes diferentes.
El conjunto $\{x, fy\}$ es unificable mediante $\sigma = [x := fy]$.
El conjunto $\{gxw, hya\}$ no es unificable pues g y h son símbolos distintos.
$\{fxgyw, fagbhw\}$ no es unificable pues los subtérminos w y hw no son unificables.

Para el caso general en que $W = \{t_1, \ldots, t_n\}$ el unificador se obtiene aplicando recursivamente el caso para dos términos como sigue:

- Hallar μ u.m.g. de t_1, t_2
- Hallar ν Unificador de $\{t_2\mu, t_3\mu_1, \ldots, t_n\mu\}$
- El unificador de W es la composición $\nu\mu$.

2.1. El algoritmo de unificación de Martelli-Montanari

A continuación se describe el algoritmo de unificación de Martelli-Montanari[1]:

- Entrada: un conjunto de ecuaciones $\{s_1 = r_1, \ldots, s_k = r_k\}$ tales que se desea unificar simultaneamente s_i con r_i, $1 \leq i \leq k$.
- Salida: un unificador más general μ tal que $s_i\mu = r_i\mu$ para toda $1 \leq i \leq n$.

Colocar como ecuaciones las expresiones a unificar, escoger una de ellas de manera no determinística que tenga la forma de alguna de las siguientes opciones y realizar la acción correspondiente:

Nombre de la regla	$t_1 = t_2$	Acción
Descomposición DESC	$fs_1 \ldots s_n = ft_1 \ldots t_n$	sustituir por $\{s_i = t_i\}$
Desc. fallida DFALLA	$fs_1 \ldots s_n = gt_1 \ldots t_n$ donde $g \neq f$	falla
Eliminación ELIM	$x = x$	eliminar
Intercambio SWAP	$t = x$ donde t no es una variable	sustituir por la ecuación $x = t$
Sustitución SUST	$x = t$ donde x no figura en t	eliminar $x = t$ y aplicar la sustitución $[x := t]$ a las ecuaciones restantes
Sustitución fallida SFALLA	$x = t$ donde x figura en t y $x \neq t$	falla

[1] "An efficient unification algorithm", ACM Trans. Prog. Lang. and Systems vol. 4, 1982, p.p. 258-282

Este algoritmo termina cuando no se puede llevar a cabo ninguna acción o cuando falla. En caso de éxito se obtiene el conjunto vacío de ecuaciones y el unificador más general se obtiene al componer todas las sustituciones usadas por la regla de sustitución en el orden en que se generaron.

El caso en que se deba unificar a dos constantes iguales a genera la ecuación $a = a$ que por la regla de descomposición se sustituye por el conjunto vacío de ecuaciones, es decir, cualquier ecuación de la forma $a = a$ se elimina. En el caso de una ecuación $a = b$ con dos constantes distintas falla por la regla de descomposición fallida.

Ejemplo 2.1 Sean $f^{(2)}, g^{(1)}, h^{(2)}$ y $W = \{fgxhxu,\ fzhfyyz\}$. Mostramos el proceso de ejecución del algoritmo de manera similar al razonamiento de verificación de correctud de argumentos por interpretaciones.

1. $\{fgxhxu = fzhfyyz\}$ Entrada
2. $\{gx = z,\ hxu = hfyyz\}$ DESC,1
3. $\{z = gx,\ hxu = hfyyz\}$ SWAP,2
4. $\{hxu = hfyygx\}$ SUST,3, $[z := gx]$
5. $\{\ x = fyy,\ u = gx\}$ DESC,4
6. $\{u = gfyy\}$ DESC,5, $[x := fyy]$
7. $\varnothing$ DESC,6, $[u := gfyy]$

El unificador se obtiene al componer las sustituciones utilizadas desde el inicio.

$$\begin{aligned} \mu &= [z := gx][x := fyy][u := gfyy] \\ &= [z := gfyy, x := fyy][u := gfyy] \\ &= [z := gfyy, x := fyy, u := gfyy] \end{aligned}$$

Veamos ahora un ejemplo de conjunto no unificable

Ejemplo 2.2 Sean $f^{(3)}, g^{(1)}$ y $W = \{fxyx,\ fygxx\}$.

1. $\{fxyx = fygxx\}$ Entrada
2. $\{x = y,\ y = gx,\ x = x\}$ DESC,1
3. $\{x = y,\ y = gx\}$ ELIM,2
4. $\{y = gy\}$ SUST,3, $[x := y]$
5. X SFALLA,4

Para terminar enunciamos el teorema de correctud total del algoritmo AU.

Teorema 1 (Correctud total del algoritmo de Martelli-Montanari) *Dado un conjunto finito de expresiones W, el algoritmo MM termina, dando como resultado el mensaje "W no es unificable", en el caso en que W no sea unificable, y un unificador más general μ de W en el caso en que W sea unificable.*

2.2. Unificación de literales

El algoritmo de unificación puede generalizarse para unificar literales de manera sencilla simplemente adaptando las reglas de descomposición y descomposición fallida considerando los casos para símbolos de predicado de manera análoga a los símbolos de función.

Como ejemplo se deja al lector unificar el siguiente conjunto

$$W = \{Qxaz, Qyahy, Qxahgb\}$$

3. Implementación del algoritmo de unificación

La implementación del algoritmo se sirve de la implementación previa de términos y sustituciones dada en la nota de clase 4. Implementamos primero el caso particular en que la entrada del algoritmo es una única ecuación $t_1 = t_2$ mediante la función `unifica` descrita abajo.

Las funciones a implementar son:

- `simpSus:: Subst ->Subst` que dada una sustitución elimina de ella los pares con componentes iguales correspondientes a sustituciones de la forma $x := x$
- `compSus:: Subst ->Subst ->Subst` que dadas dos sustituciones devuelve su composición
- `unifica :: Term ->Term ->[Subst]` que dados dos términos devuelva una lista de sustituciones de tal forma que
  - Si t_1, t_2 no son unificables la lista es vacía
  - Si sí lo son la lista contiene como único elemento al unificador correspondiente.

  Para el caso en que t_1, t_2 sean términos funcionales se necesitará una función `unificaListas :: [Term] ->[Term] ->[Subst]` que unifique dos listas de términos de la misma longitud componente a componente. Es decir, `unificaListas [s1,..,sk] [r1,..,rk]` devuelve la lista $[\mu]$ con un unificador μ tal que $s_j\mu = r_j\mu$ para toda $1 \leq j \leq k$. En cualquier otro caso esta función devuelve la lista vacía.
- `unificaConj :: [Term] ->[Subst]` que implementa el caso general para unificar un conjunto (lista) $W = \{t_1, \ldots, t_n\}$.
- `unificaLit :: Form ->Form ->[Subst]` que unifique dos literales.