Unificación de términos
Lógica Computacional 2022-1, Nota de clase 9

Favio Ezequiel Miranda Perea Araceli Liliana Reyes Cabello
Lourdes Del Carmen González Huesca Pilar Selene Linares Arévalo

18 de enero de 2022
Facultad de Ciencias UNAM

La regla de resolución binaria de Robinson, que veremos más adelante, proporciona un método sintáctico para decidir la correctud de un argumento formalizado en lógica de predicados. En esta nota veremos un concepto básico del método de resolución binaria para lógica de predicados que es la unificación.

Si bien la idea del método de prueba resolución es la misma que en la lógica proposicional, es decir llegar a la cláusula vacía para mostrar que un argumento es correcto o que una fórmula es válida, en lógica de predicados el método se complica debido a la presencia de variables.

Recordemos que en lógica proposicional, la cláusula vacía se obtiene al resolver dos literales contrarias [1], por ejemplo $p, \neg p$. Las literales en lógica de predicados son las fórmulas atómicas que son los predicados. Pero a diferencia de las literales en lógica proposicional, resolver dos literales en predicados requiere de más cuidado. Por ejemplo, el par de literales $P(a,x), \neg P(z,b)$ no puede resolverse aunque tengan el mismo símbolo de predicado pues no son contrarias estrictamente dado que no comparten las mismas variables y/o constantes en el mismo orden.

Sin embargo en ciertos casos, como veremos después, podemos aplicar una sustitución para *unificar* las literales y aplicar resolución. Del ejemplo anterior, se puede utilizar la sustitución $\sigma = [x, z := b, a]$ para $P(a,x)$ y $\neg P(z,b)$ y obtenier como resultado $P(a,b)$ y $\neg P(a,b)$ que ahora 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 cómo es posible hallarlos algorítmicamente.

1. Sobre la sintaxis de los términos y predicados

Con el objetivo de eliminar lo más posible el uso de paréntesis anidados, los términos y predicados se escribiran sin usar paréntesis ni comas que separen sus argumentos. Esto no causa ambigüedad alguna dado que en la signatura cada símbolo tiene definido un índice y número de argumentos. Por ejemplo, si sabemos que $f^{(2)}, P^{(3)}$ es decir, el símbolo de función f espera dos argumentos y el símbolo de predicado P espera tres, entonces la fórmula atómica $P(f(x,a),b,z)$ puede escribirse como $Pfxabz$. Los paréntesis pueden restaurarse sin ambigüedad pues sabemos el índice de f es 2, de donde el único término que empieza con f es fxa siendo la variable x y la constante a sus dos argumentos. De esta forma fxa es el primero de los tres argumentos de P. En la cadena restante bz tienen que aparecer dos términos y la única manera para que esto suceda es con la constante b y la variable z.

[1] Las literales proposicionales son fórmulas atómicas, las constantes lógicas o variables proposicionales) o negaciones de fórmulas atómicas.

Veamos otros ejemplos, dando primero los índices de los símbolos; la expresión sín paréntesis y la correspondiente expresión sin paréntesis.

- $Q^{(2)}, f^2, g^{(1)}\quad Qgfxbfgyz\quad Q\Big(g(f(x,b)), f(g(y),z)\Big)$
- $f^{(1)}, g^{(2)}, h^{(3)}\quad gfwhxgzaffb\quad g\Big(f(w), h(x, g(z,a), f(f(b)))\Big)$

2. Sustituciones

El concepto de sustitución (simultánea) ha sido estudiado en la nota 7, a continuación revisaremos otra definición formal.

Definición 1 *Una sustitución en un lenguaje de predicados es una función recursiva $\sigma : Term \to Term$ 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 sustitución se denotará con

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

o de manera más breve $\sigma = [\vec{x} := \vec{t}\,]$ [2].

La composición de sustituciones se define como una composición de funciones de donde se ha tomado la siguiente caracterización por motivos de implementación:

Proposición 1 *Sean $\sigma = [x_1 := t_1,\ \ldots,\ x_n := t_n]$ y $\tau = [y_1 := s_1,\ \ldots,\ y_m := s_m]$ dos 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 variables 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 $y_q \notin \{x_1, \ldots, x_n\}$ (es decir, son los pares no incluidos en $x_1, \ldots, x_n$).

Demostración. Ejercicio $\dashv$

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

Demostración. Ejercicio $\dashv$

Ejemplo 1 Sean

$$\sigma = [x := f(y),\ y := w] \qquad \rho = [x := g(w),\ z := b] \qquad \tau = [y := b,\ w := f(c),\ v := w]$$

[2] También se puede representar como $[x_1, x_2, \ldots, x_n := t_1, t_2, \ldots, t_n]$.

Entonces

$$\sigma\rho = [x := f(y),\ y := w,\ z := b]$$

$$\rho\tau = [x := g(f(c)),\ z := b,\ y := b,\ w := f(c),\ v := w]$$

$$(\sigma\rho)\tau = [x := f(b),\ y := f(c),\ z := b,\ w := f(c),\ v := w]$$

$$\sigma(\rho\tau) = [x := f(b),\ y := f(c),\ z := b,\ w := f(c),\ v := w]$$

Dado que estamos considerando a las sustituciones por sí solas y no como una operación que requiere de un término o fórmula, resulta natural preguntarse si la operación recursiva de sustitución vista anteriormente, digamos $t[\vec{x} := \vec{r}]$ coincide con la operación $t\sigma$ cuando σ es la sustitución $\sigma = [\vec{x} := \vec{r}]$. Esto es consecuencia directa de la siguiente propiedad de asociatividad, mostrada a continuación con un ejemplo:

Consideremos la fórmula $A =_{def} \forall x(\neg Q(x,z) \to \forall y(R(w,y) \to Q(b,x)))$ y las siguientes sustituciones. Usaremos la siguiente α-equivalencia:

$$\forall x(\neg Q(x,z) \to \forall y(R(w,y) \to Q(b,x))) \sim_\alpha \forall x'(\neg Qx'z \to \forall y'(Rwy' \to Qbx'))$$

Se cumple que $(A\rho)\tau = A(\rho\tau)$

$$\begin{aligned}
(A\rho)\tau &= (\forall x'(\neg(Qx'z)\rho \to \forall y'((Rwy')\rho \to (Qbx')\rho)))\tau \\
&= (\forall x'(\neg(Qx'b) \to \forall y'((Rwy') \to (Qbx'))))\tau \\
&= (\forall x'(\neg(Qx'b)\tau \to \forall y'((Rwy')\tau \to (Qbx')\tau))) \\
&= (\forall x'(\neg(Qx'b) \to \forall y'((Rf(c)y') \to (Qbx'))))
\end{aligned}$$

$$\begin{aligned}
A(\rho\tau) &= (\forall x'(\neg(Qx'z) \to \forall y'((Rwy') \to (Qbx'))))(\rho\tau) \\
&= (\forall x'(\neg(Qx'z) \to \forall y'((Rwy') \to (Qbx'))))[x := f(b),\ y := f(c),\ z := b,\ w := f(c),\ v := w] \\
&= (\forall x'(\neg(Qx'b) \to \forall y'((Rf(c)y') \to (Qbx'))))
\end{aligned}$$

Dejamos como ejercicio demostrar que para cualquier A y cualesquiera sustituciones σ, τ se cumple

$$(A\sigma)\tau = A(\sigma\tau)$$

.

3. Unificación

El proceso de unificación consiste en encontrar, dado un conjunto de literales o términos W, una sustitución σ de tal forma que el conjunto resultante $W\sigma$ 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. A continuación estudiamos el caso general así como un algoritmo de unificación.

Definición 2 *Sea W un conjunto no vacío de términos. Un unificador de W es una sustitución 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 mismo elemento. Si W tiene un unificador decimos que W es unificable.*

Ejemplo 2 Sea $W = \{g(x, f(y)), g(x, f(x)), g(u, v)\}$ un conjunto de términos, entonces la sustitución $\sigma = [x := a,\ y := a,\ u := a,\ v := f(a)\}$ es un unificador de W ya que $W\sigma = \{g(a, f(a))\}$

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 3 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 el **umg** y 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:

1. 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 $t_1 = t_2$.

2. Alguno de los términos es una variable.
   Supongamos que $t_1 = x$, entonces si x figura en t_2 entonces W no es unificable, en caso contrario $\sigma = [x := t_2]$ unifica a W.

3. 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 izquierda a derecha), de t_1 y t_2 son el mismo.
   - Cada par correspondiente de subexpresiones de t_1 y t_2 deben ser unificables.

Ejemplo 4 Considerando los mismos símbolos de función que el ejemplo anterior, tenemos los siguientes ejemplos:

- 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.
- El conjunto $\{fxgyw, fagbhw\}$ 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 **umg** 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 $\nu\mu$.

3.1. El algoritmo de unificación de Martelli-Montanari

A continuación se describe el algoritmo de unificación de Martelli-Montanari [3]:

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

Para unificar un conjunto de términos:

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

- el algoritmo termina cuando no se puede llevar a cabo alguna 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.

Observaciones: El caso en que se deba unificar a dos constantes iguales a, se genera la ecuación $a = a$ que por la regla de descomposición se sustituye por el conjunto vacío de ecuaciones dado que constantes se consideran funciones sin argumentos, así 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 3.1 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:

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

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

Veamos ahora un ejemplo de un conjunto no unificable:

Ejemplo 3.2 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.

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.*

3.2. 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

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

4. Implementación del algoritmo de unificación

La implementación del algoritmo se sirve de la implementación previa de términos y sustituciones dada en las notas de clase 5 y 7. Esta sección corresponde a la siguiente práctica y se discutirá en su momento.

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) $W = \{t_1, \ldots, t_n\}$.una función
- `unificaLit :: Form -> Form -> [Subst]` que unifique dos literales.

5. Ejercicios

1. Verificar si los siguientes conjuntos son unificables utilizando el algoritmo de Martelli-Montanari, puedes obviar algunos pasos pero debes mostrar los que generan sustituciones.

    a) $W = \{h(f(a), g(x)),\ h(z, z)\}$
    b) $W = \{f(w, f(x, h(z))),\ f(g(x), f(x, y)),\ f(g(x), f(a, b))\}$
    c) $W = \{fxgfayz, fbgfagxcfyx\}$ con $f^{(2)}, g^{(2)}$
    d) $W = \{f\Big(x,\ g(f(a, y), z\Big), f\Big(b,\ g(f(a, g(x, c)), f(y, x))\Big)\}$
    e) $W = \{Q(x, f(x, y)),\ Q(y, f(y, a)),\ Q(b, f(b, a))\}$
    f) $W = \{Pxfy,\ Pgyafb,\ Pgbzw\}$ con $P^{(2)}, f^{(1)}, g^{(2)}$
    g) $W = \{Qazgabc,\ Qafxgaby,\ Qaffwgaxgcba\}$ con $Q^{(3)}, f^{(1)}, g^{(3)}$
    h) $W = \{Pxfxgy,\ Pafgaga,\ Pyfyga\}$ con $P^{(3)}, f^{(1)}, g^{(1)}$
    i) $W = \{Rfayz, Rxyfz, Ryfab\}$ con $R^{(3)}, f^{(1)}$.
    j) $W = \{P(x, f(x), c),\ P(u, b, z)\}$
    k) $W = \{Q(y, z),\ Q(x, f(a)),\ Q(f(z), z)\}$.
    l) $W = \{R(w, f(b), f(g(y))), R(a, x, f(g(y))),\ R(z, f(z), f(u))\}$
    m) $W = \{T(u, v, w, z),\ T(f(z), x, g(h(a, b)), g(c)),\ T(f(g(y)), z, w, g(y))\}$