Programación lógica y bases de datos

Lógica computacional

6 de mayo de 2024

1. Unificadores

En la sección pasada hablamos de sustituciones y sus composiciones. Los unificadores son un tipo específico de sustituciones que hacen que dos términos sean idénticos. Si tenemos $f(a, y, z)$ y $f(x, b, z)$ la sustitución $\{x/a, y/b\}$ es un unificador de ambos términos. No es el único, podría también usarse $\{x/a, y/b, z/a\}$ pero es menos general.

Definición 1 *Sean θ y τ sustituciones. Decimos que θ es* más general que *τ si para alguna sustitución η tenemos $\tau = \theta\eta$.*

Ejemplo: La sustitución $\{x/y\}$ es más general que $\{x/a, y/a\}$, ya que $\{x/y\}\{y/a\} = \{x/y\}$. **Contraejemplo:** $\{x/y\}$ no es más general que $\{x/a\}$.

Lema 1 *θ es más general que η y η es más general que θ si y sólo si por algún renombramiento γ tal que $Var(\gamma) \subseteq Var(\theta) \cup Var(\eta)$, tenemos $\eta = \theta\gamma$*

Definición 2

1. *θ es llamado un* unificador *se s y t si $s\theta = t\theta$. Si un unificador de s y t existe, decimos que s y t son unificables.*
2. *θ es llamado un* unificador de máxima generalidad *(*mgu*, por sus siglas en inglés,* umg *en español) de s y t si es un unificador de ambos y es más general que cualquier otro unificador de s y t.*
3. *Un umg θ se s y t se llama fuerte si para todos los unificadores η de s y t tenemos $\eta = \theta\eta$*

Ejemplo: Tenemos los términos $f(g(x, a), z)$ y $f(y, b)$. Entonces $\{x/c, y/g(c, a), z/b\}$ es uno de sus unificadores, al igual que $\{y/g(x, a), z/b\}$ que es más general que el primero, ya que $\{x/c, y/g(c, a), z/b\} = \{y/g(x, a), z/b\}\{x/c\}$. De hecho es el más general y es fuerte ya que:

$$\{x/c, y/g(c, a), z/b\} = \{y/g(x, a), z/b\}\{x/c, y/g(c, a), z/b\}$$

Contraejemplo: Considera los términos $f(g(x, a), z)$ y $f(g(x, b), b)$. No son unificables.

2. Algoritmos para obtener umg

Ya empezamos a ver que poder determinar si dos términos son unificables puede ser complicado, aún más complicado paraece ser hallar el unificador de máxima generalidad. Existen varios algoritmos para obtenerlo, veremos algunos.

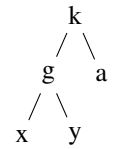

Figura 1: Árbol para el algoritmo no determinista de Robinson para el término $k(g(x, y), a)$

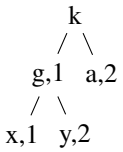

Figura 2: Árbol aumentado para el algoritmo no determinista de Robinson

2.1. Algoritmo no determinista de Robinson

Los términos son vistos como árboles aumentados.

- El árbol asociado a una variable tiene un sólo nodo, el de la variable en sí
- El árbol de una función $f(t_1, t_2, ..., t_n)$, al nodo marcado como f se le asocia el árbol asociado a t_1, t_2,...,t_n en el orden necesario.
- El árbol asociado a una constante tiene un sólo nodo, el de la constante misma.

Se aumenta el árbol indicando el número de hijo que es del nodo padre (en el orden de izquierda a derecha, como escribimos). Una vez con esto obtenemos el trayecto para un término w hasta la raíz. Por ejemplo para el término y tenemos la trayectoria $\langle k \rangle$, $\langle g, 1 \rangle$ y $\langle y, 2 \rangle$.

Supongamos que tenemos el término w con ocurrencia en s y u con ocurrencia en t, el par w, u son un *par en desacuerdo* si sus caminos de acceso difieren sólo por la etiqueta del último nodo.

Un par es simple si uno de ellos **no** es una variable que ocurre en el otro término.

Lema 2 ***(Desacuerdo)*** *Sean u y w un par en desacuerdo de s y t:*

- *Todo unificador de s y t es un unificador también de u y w*
- *Si el par u, w es simple, entonces ambos términos son unificables, de hecho, cualquier sustitución determinada por u, w es un umg fuerte de u y w.*
- *Si el par u, w no es simple, entonces u y w no son unificables.*

Ahora sí el algoritmo de unificación de Robinson:

- Sea θ igual a ϵ
- Mientras $s\theta \neq t\theta$ repite:

- u y w son un par simple $\implies$ sea γ una sustitución dada por u y w, sea $\theta = \theta\gamma$
- u y w no son un pare simple $\implies$ detente como fallo.

Vamos a ampliar el algoritmo.

2.2. Algoritmo de Robinson

Para ampliarlo ahora se buscarán los pares en desacuerdo en cierto orden. Para ello no necesitaremos de la estructura de árbol, ya que el orden que daremos para la búsqueda de los pares será el orden textual, la forma de leer el término, de izquierda a derecha.

- Sea θ igual a ϵ
- iniciamos la lectura en los primeros símbolos a la izquierda de $s\theta$ y $t\theta$
- Mientras $s\theta \neq t\theta$ repite:
  - Avanza de manera simultanea a la derecha hasta que un par de símbolos diferentes en $s\theta$ y $t\theta$ sean encontrados.
  - Determina el par de términos w y u cuyos símbolos iniciales son los identificados y realiza las acciones asociadas:
    - u y w son un par simple $\implies$ sea γ una sustitución dada por u y w, sea $\theta = \theta\gamma$
    - u y w no son un pare simple $\implies$ detente como fallo.

Ejemplo: Sea el par de términos:

$$k(z, f(x, b, z))$$
$$k(h(x), f(g(a), y, z))$$

Primer par en desacuerdo: z, $h(x)$. Este par define la sustitución $\{z/h(x)\}$, aplicándola:

$$k(h(x), f(x, b, h(x)))$$
$$k(h(x), f(g(a), y, h(x)))$$

El siguiente par en desacuerdo: x, $g(a)$. Esto define la sustitución $\{x/g(a)\}$, aplicando:

$$k(h(g(a)), f(g(a), b, h(g(a))))$$
$$k(h(g(a)), f(g(a), y, h(g(a))))$$

El siguiente par en desacuerdo: b, y. Esto define la sustitución $\{y/b\}$, aplicando:

$$k(h(g(a)), f(g(a), b, h(g(a))))$$
$$k(h(g(a)), f(g(a), b, h(g(a))))$$

Ya son términos idénticos, las sustituciones que forma el umg son $\{z/h(x)\{x/g(a)\}\{y/b\} = \{z/h(g(a)), x/g(a), y/b\}$.

3. Clausulas definidas

Definición 3 *Una clausula definida es un enunciado de la forma*

$$\forall x_1, ..., x_m.(\exists y_1, ..., y_k.(\beta_1 \wedge \cdots \wedge \beta_n)) \implies \alpha$$

donde $\beta_1, ..., \beta_n$ y α son fórmulas atómicas y dependen de las variables $y_1, ..., y_k$ y $x_1, ..., x_m$ respectivamente, pero no comparten esa dependencia. Está clausula se escribirá como

$$\alpha \leftarrow \beta_1, ..., \beta_n$$

- *α es el encabezado y $\beta_1, .., \beta_n$ el cuerpo.*
- *Una fórmula puede carecer de cuerpo o encabezado. Si no tiene encabezado es una **meta**.*
- *Un programa lógico es un conjunto de clausulas definidas, ninguna de las cuales es una meta.*

Una meta se escribe como

$$\leftarrow \beta = \neg\exists y_1, ..., y_k.\beta$$

Algunas definiciones extras y una paráfrasis de *metas*, si escribimos la clausula definida o de *Horn* como

$$A_1 \vee ... \vee A_k \leftarrow B_1 \wedge ... \wedge B_m$$

1. $k = 1$ y $n > 0$ es lo que se llama una *regla*
2. $k = 1$ y $n = 0$ es lo que se llama un *hecho*
3. $k = 0$ y $n > 0$ es lo que se llama un *objetivo* (una meta)

Un ejemplo: Como ecribiríamos una relación familiar entre dos hermanas como una clausula definida.

$$hermanas(X, Y) \leftarrow mujer(X) \wedge mujer(Y) \wedge padres(X, P, M) \wedge padres(Y, P, M)$$

4. Resolución Proposicional

Propuesta en 1965 por J.A. Robinson, se trabaja sobre clausulas. No es el caso más general pero lo usaremos para dar una introducción a la resolución SLD. Este tipo de resolución es que convierte una fórmula de lógica proposicional a un conjunto de clausulas.

Tenemos las clausulas, que se convierte en una *meta* para finalmente dar solución:

$$\frac{\alpha \leftarrow \beta_1, ..., \beta_n \quad \leftarrow, \gamma_1, \gamma_k}{\leftarrow \beta_1, ...\beta_n, \gamma_1, ..., \gamma_k}$$

Aplicamos un algoritmo para tener la solución:

Algorithm 1 Algoritmo de resolución proposicional

Require: Conjunto de clausulas
Ensure: Detecta si C es insatisfacible
  Buscar dos cláusulas $C_1, C_2 \in C$ tales que exista ℓ que cumple $\ell \in C_1$ y $\neg\ell \in C_2$
  if Se encuentran las cláusulas **then**
    Calcular $R_\ell(C_1, C_2)$ (el resultante[1]) y añadirlo a C
    if Si $R_\ell = \emptyset$ **then**
      Salir y declarar que C es insatisfacible
    else if Si no **then**
      Volver al primer estado
    end if
  else if En caso de no encontrarse las cláusulas **then**
    Salir y declarar que C es insatisfacible
  end if

5. Bases de datos relacionales

Ya en la primera parte del curso hablamos de relaciones, ese es el concepto base en una base de datos relacional, pero vamos puntualizando.

Sean $D_1, D_2, ..., D_n$ una colección de símbolos llamados *dominios*, los miembros del dominio son atómicos o indivisibles, no se puede acceder a una parte propia de un miembro.

Definición 4 *Una* relación de base de datos R *sobre los dominios* $D_1, D_2, ..., D_n$ *es un subconjunto de* $D_1 \times D_2 \times ... \times D_n$. *Se dice en este caso que* R *es* $n-$*ario. Una* base de datos relacional *es un número finito de tales relaciones (finitas).*

Ejemplo: Sean las relaciones, siguiendo con los ejemplos de familia[2], *PADRE*, *MADRE* y *MAPADRES* y los dominios *HOMBRE*:= adán, beto, *MUJER* := ana, bere y *PERSONA* := *HOMBRE* $\cup$ *MUJER*.

De los dominios podemos definir las relaciones *HOMBRE* $\times$ *PERSONA*, *MUJER* $\times$ *PERSONA* y *PERSONA* $\times$ *PERSONA*. POr pomer un ejemplo de como se vería una de estas relaciones:

$$HOMBRE \times PERSONA = \begin{Bmatrix} \langle adan, adan\rangle & \langle beto, adan\rangle \\ \langle adan, beto\rangle & \langle beto, beto\rangle \\ \langle adan, ana\rangle & \langle beto, ana\rangle \\ \langle adan, bere\rangle & \langle beto, bere\rangle \end{Bmatrix}$$

Las relaciones *PADRE*, *MADRE* y *MAPADRES* será sobre estas otras relaciones, de la forma:

$$\begin{aligned} PADRE :=& \{\langle adan, beto\rangle, \langle adan, bere\rangle\} \\ MADRE :=& \{\langle ana, beto\rangle, \langle ana, bere\rangle\} \\ MAPADRE :=& \{\langle adan, beto\rangle, \langle adan, bere\rangle, \langle ana, beto\rangle, \langle ana, bere\rangle\} \end{aligned}$$

Una representación sintáctica diferente puede ser con tablas:

[2]Disculpen el binarismo de los ejemplos, así lo pone el libro. Si en un futuro estas notas se amplían buscaré la manera de mejorar esos ejemplos.

PADRE

C_1	C_2
adan	beto
adan	bere

MADRE

C_1	C_2
ana	beto
ana	bere

MAPADRE

C_1	C_2
adan	beto
adan	bere
ana	beto
ana	bere

O como una colección de eneadas etiquetadas:

$$
\begin{aligned}
&padre(adan, beto).\\
&padre(adan, bere).\\
&madre(ana, beto).\\
&madre(ana, bere).\\
&mapadre(adan, beto).\\
&mapadre(adan, bere).\\
&mapadre(ana, beto).\\
&mapadre(ana, bere).
\end{aligned}
$$

Las representaciones son isomórficas siempre y cuando no se ponga mucha importancia a los nombgres de las columnas, los *atributos* que son de ayuda para establecer relaciones pero no son cruciales.

5.1. Bases de datos deductivas y álgebra relacional

Podemos representar las bases de datos anteriores como programas[3], de la forma:

$$
\begin{aligned}
mapadre(X, Y) &\leftarrow padre(X, Y).\\
mapadre(X, Y) &\leftarrow madre(X, Y).\\
padre(adan, beto).&\\
padre(adan, bere).&\\
madre(ana, beto).&\\
madre(ana, bere).&
\end{aligned}
$$

Que es lo que llamamos una *base de datos intencional*, que sólo está formada por reglas y hechos no básicos. Es deductiva pues a partir de las fórmulas atómicas dadas se pueden deducir nuevas.

Las operaciones primitivas del álgebra relacional son

- Unión: Dadas dos relacione n-arias R_1 y R_2 sobre el mismo dominio, su unión es:
$$\{\langle x_1, ..., x_n\rangle | \langle x_1, ..., x_n\rangle \in R_1 \vee \langle x_1, ..., x_n\rangle \in R_2\}$$

[3]Esta es la relevancia de estudiar bases de datos relacionales en este tema, un programa funcional realmente es una base de datos.

- La diferencia $R_1 \setminus R_2$ de dos relaciones sobre el mismo dominio da:

$$\{\langle x_1, ..., x_n\rangle \in R_1 | \langle x_1, ..., x_n\rangle \notin R_2\}$$

- El producto cartesiano de dos relaciones, $R_1 \times R_2$, da la nueva relación:

$$\{\langle x_1, ..., x_m, y_1, ..., y_n\rangle | \langle x_1, ..., x_m\rangle \in R_1 \wedge \langle y_1, ..., y_n\rangle \in R_2\}$$

- La proyección es el reacomodo de atributos, por ejemplo, sea la relación $F(X, Y)$, la proyección del primer atributo está dada como:

$$\pi_1 R_1(X, Y) := \{\langle x_1\rangle | \langle x_1, x_2\rangle \in R_1\}$$

- La selección son las eneadas de la relación R que cumplen la fórmula F, es decir

$$\sigma_R(R) := \{\langle x_1, ..., x_n\rangle \in R | \text{ F es cierta para } \langle x_1, ..., x_n\rangle\}$$

- La conjunción natural (no es primitiva) de dos relaciones, $R_1 \bowtie R_2$, si $T_1, ..., T_k$ son los atributos que aparecen tanto en R_1 como en R_2, entonces la *conjunción natural* es:

$$R_1 \bowtie R_2 :=_{\pi_A \sigma_{R.T_1 = S.T_1 \wedge ... \wedge R.T_k = S.T_k}} (R_1 \times R_2)$$

  donde A es la lista de todos los atributo del producto cartesiano $R_1 \times R_2$ con excepción de $S.T_1, ..., S.T_k$, es decir, se toma el producto cartesiano de las dos relaciones y se seleccionan sólo las eneadas que tienen valores idénticos en la columna con el mismo atributo, deshaciéndose de las columnas superfluas.

Ejemplo: Para ver como es que se hace la conjunción natural tomemos las relaciones R dada en la primera tabla y S en la segunda:

R

C_1	C_2
adan	beto
beto	cata

S

C_2	C_3
adan	beto
beto	cata
cata	david

El producto cartesiano de ambas relaciones será:

$R \times S$

C_1	$R.C_2$	$S.C_2$	C_3
adan	beto	adan	beto
adan	beto	beto	cata
adan	beto	cata	david
beto	cata	adan	beto
beto	cata	beto	cata
beto	cata	cata	david

Sólo nos quedamos con las eneadas que tengan el mismo valor de $R.C_2$ y $S.C_2$, entonces

$\pi_{C_1, R.C_2, C_3} \sigma_{R.C_2 = S.C_2} R \times S$

C_1	$R.C_2$	$S.C_2$	C_3
adan	beto	beto	cata
beto	cata	cata	david

Y eliminamos las columna repetida:

$R \bowtie S$

C_1	$R.C_2$	C_3
adan	beto	cata
beto	cata	david

Una operación más, que no es primitiva pues puede expresarse a partir de las demás, al igual que la conjunción natural, es el *cambio de nombre*, que se escribe como $\rho_{B|A}(r)$, cambia de nombre el atributo A de la relación r por el nuevo nombre, que antes no se usaba en esa relación, B.

Ejemplo: Se tiene la relación R_1

R_1

Nombre	Lugar
pulque	hidalgo
tejuino	jalisco
tepache	michoacan

Se aplica $\rho_{fermentado|nombre}(R_1)$ y la nueva tabla queda:

$\rho_{fermentado|nombre}(R_1)$

Fermentado	Lugar
pulque	hidalgo
tejuino	jalisco
tepache	michoacan