Introducción

En la primera parte de la clase, estudiaremos el método de resolución para los lenguajes de predicados.

A continuacón, daremos una introducción al lenguaje de programación Prolog, cuyo interpretador está basado en el método de resolución para los lenguajes de predicados.

Empezamos viendo un ejemplo de aplicación del Algoritmo de Unificación.

Ejemplo

Aplicamos el algoritmo de unificación a las expresiones $e_1 = Rzz$, $e_2 = Rf(a)g(x)$.

Inicialmente, tenemos que la pila P está vacía, el vector λ está vacío y $b = false$.

En el primer paso de cómputo del algoritmo, se pone en la pila la igualdad $Rzz = Rf(a)g(x)$, es decir, tendremos $P = [Rzz = Rf(a)g(x)]$. Y entramos entonces en el bucle while del algoritmo.

Sustituimos entonces en la pila la igualdad $Rzz = Rf(a)g(x)$ por las igualdades $z = f(a)$ y $z = g(x)$. Por tanto, tendremos $P = [z = f(a), z = g(x)]$.

Ejemplo

A continuación, sacamos de la pila la ecuación $z = f(a)$, y entonces sustituimos en la igualdad $z = g(x)$ que queda en la pila z por $f(a)$ y ponemos dicha ecuación $z = f(a)$ en el vector λ. Por tanto, tendremos $P = [f(a) = g(x)]$, $\lambda = \{z = f(a)\}$.

Ahora, en la única ecuación que tenemos en la pila $f(a) = g(x)$ tenemos dos términos $f(a)$, $g(x)$, los cuales no son variables. Por tanto, ponemos $b = true$.

Así pues, como $b = true$, salimos del bucle while (etapa (2) del algoritmo). Entonces, en la etapa (3), el algoritmo de unificación da como salida "fallo". Por tanto, las expresiones de la entrada e_1, e_2 no son unificables

Concepto de resolvente

Recordamos a continuación el concepto de resolvente de dos cláusulas.

Cuando se computa el resolvente de dos cláusulas, se supone que las cláusulas no comparten variables. Si no es así, previamente hay que renombrar las variables de alguna de las dos cláusulas.

Entonces, supongamos que φ_1 y φ_2 son cláusulas sin variables en común tales que $\varphi_1 \equiv \psi_1 \vee \varphi_1'$, $\varphi_2 \equiv \neg\psi_2 \vee \varphi_2'$ y ψ_1, ψ_2 son unificables. Consideremos un unificador λ de ψ_1, ψ_2 obtenido mediante el algoritmo de unificación. Diremos entonces que la cláusula $(\varphi_1' \vee \varphi_2')\,\lambda$ es un resolvente de las cláusulas φ_1 y φ_2.

Regla de resolución

entrada : dos cláusulas φ_1 y φ_2 de un lenguaje de predicados.

salida : un resolvente de φ_1 y φ_2.

Utilizando esta regla de resolución, se puede entonces generalizar al contexto de la lógica de predicados el teorema de resolución que vimos para la lógica de proposiciones.

Ejemplo

Demostramos por resolución que la cláusula vacía $\square$ se deduce de las siguientes cláusulas:

$\varphi_1 = \neg Qxy \vee \neg Py \vee Rf(x)$,

$\varphi_2 = \neg Rz$,

$\varphi_3 = Qab$,

$\varphi_4 = Pb$.

Tenemos entonces la siguiente prueba por resolución:

Ejemplo

1. $\neg Qxy \vee \neg Py \vee Rf(x)$ input
2. $\neg Rz$ input
3. Qab input
4. Pb input
5. $\neg Pb \vee Rf(a)$ (1,3) tomando $\{x = a, y = b\}$
6. $Rf(a)$ (4,5)
7. $\square$ (2,6) tomando $\{z = f(a)\}$

Algoritmo de resolución

entradas : fórmulas $\varphi_1, \ldots, \varphi_n, \varphi$ de un lenguaje de predicados.

salida : Si φ es consecuencia lógica de $\varphi_1, \ldots, \varphi_n$, da "éxito". Si no, el algoritmo da "fallo" o no para.

(1) Calcular formas clausales $(\varphi_1)^{cl}$, ..., $(\varphi_n)^{cl}$, $(\neg\varphi)^{cl}$ de las fórmulas $\varphi_1, \ldots, \varphi_n, \neg\varphi$ respectivamente, y tomar las cláusulas $\psi_1, \ldots, \psi_m$ que aparecen en los núcleos de dichas formas clausales.

(2) Calcular resolventes a partir de las entradas $\psi_1, \ldots, \psi_m$. Si en algún momento se obtiene la cláusula vacía $\square$, el algoritmo para y devuelve "éxito".

Ejemplo 1

Consideremos el vocabulario $\sigma = \{Q^1, R^1\}$. Demostramos por el algoritmo de resolución que la fórmula

$$\varphi = (\forall x Qx \wedge \forall x Rx) \rightarrow \forall x(Qx \wedge Rx)$$

es una tautología. En primer lugar, tenemos que computar una forma clausal de $\neg\varphi$. Tenemos que

$\neg\varphi \equiv \forall x Qx \wedge \forall x Rx \wedge \exists x(\neg Qx \vee \neg Rx) \equiv$
$\forall x Qx \wedge \forall y Ry \wedge \exists z(\neg Qz \vee \neg Rz)$. Ahora, sustituimos la variable z por un símbolo de constante c. Tenemos entonces:

$\forall x Qx \wedge \forall y Ry \wedge (\neg Qc \vee \neg Rc) \equiv \forall x \forall y(Qx \wedge Ry \wedge (\neg Qc \vee \neg Rc))$.

Ejemplo 1

Por tanto, la fórmula $\forall x \forall y(Qx \wedge Ry \wedge (\neg Qc \vee \neg Rc))$ es una forma clausal de $\neg\varphi$. Se observa que el núcleo de esta forma clausal consta de las clásulas:

(1) Qx.

(2) Ry.

(3) $\neg Qc \vee \neg Rc$.

Como $\{Ry, Rc\}$ es unificable por $\{y = c\}$, resolvemos (2) y (3), obteniendo:

(4) $\neg Qc$.

Ahora, como $\{Qx, Qc\}$ es unificable por $\{x = c\}$, resolvemos (1) y (4), obteniendo:

(5) $\square$.

Por tanto, φ es una tautología.

Ejemplo 2

Consideremos el vocabulario $\sigma = \{P^2, Q^1, R^1\}$. Consideremos las siguientes σ-fórmulas:

$\varphi_1 = \forall x \forall y((Pxy \wedge Qx) \rightarrow Rx)$,

$\varphi_2 = \forall x \forall y(Pxy \wedge Qx)$,

$\varphi = \forall x Rx$.

Demostramos entonces que φ es consecuencia lógica de φ_1, φ_2, aplicando el algoritmo de resolución. En primer lugar, tenemos que computar formas clausales de las fórmulas φ_1,φ_2, $\neg\varphi$. Obsérvese que $\neg\varphi \equiv \exists x \neg Rx$. Tenemos entonces:

(1) $(\varphi_1)^{cl} = \forall x \forall y(\neg Pxy \vee \neg Qx \vee Rx)$,

(2) $(\varphi_2)^{cl} = \forall x \forall y(Pxy \wedge Qx)$,

(2) $(\neg\varphi)^{cl} = \neg Ra$.

Ejemplo 2

Entonces, tenemos que calcular resolventes a partir de las cláusulas que aparecen en las formas clausales anteriores hasta obtener la cláusula vacía $\square$. Recordemos que cuando se calculan resolventes, hay que renombrar las variables que se repitan en las cláusulas, ya que las variables son locales en las cláusulas en las que aparecen.

Tenemos entonces las siguientes entradas:

Ejemplo 2

(1) $\neg Px_1y_1 \vee \neg Qx_1 \vee Rx_1$.

(2) Px_2y_2.

(3) Qx_3.

(4) $\neg Ra$.

En primer lugar, resolvemos (1) y (4). Observamos que los átomos Rx_1 y Ra son unificables por $\{x_1 = a\}$. Por tanto, obtenemos:

(5) $\neg Pay_1 \vee \neg Qa$.

Ahora, resolvemos (3) y (5). Observamos que los átomos Qa y Qx_3 son unificables por $\{x_3 = a\}$. Por tanto, obtenemos:

(6) $\neg Pay_1$.

Ejemplo 2

Por último, resolvemos (2) y (6). Observamos que los átomos Px_2y_2 y Pay_1 son unificables por $\{x_2 = a, y_2 = y_1\}$. Por tanto, obtenemos:

(7) $\square$.

Ejemplo 3

Consideremos el vocabulario $\sigma = \{a, f^1, g^1, P^1, Q^2\}$.
Consideremos las siguientes σ-fórmulas:

$\varphi_1 = Pa$,

$\varphi_2 = \forall x(Px \to (Pf(x) \vee Pg(x)))$,

$\varphi_3 = \forall x(Qxf(x) \wedge Qxg(x))$,

$\varphi = \exists x \exists y(Px \wedge Qyx)$.

Demostramos entonces que φ es consecuencia lógica de $\varphi_1, \varphi_2, \varphi_3$, aplicando el algoritmo de resolución.
En primer lugar, tenemos que computar formas clausales de las fórmulas φ_1,φ_2, φ_3 y $\neg\varphi$. Tenemos entonces:

(1) $(\varphi_1)^{cl} = Pa$.

(2) $(\varphi_2)^{cl} = \forall x(\neg Px \vee Pf(x) \vee Pg(x))$.

(3) $(\varphi_3)^{cl} = \forall x(Qxf(x) \wedge Qxg(x)$.

(4) $(\neg\varphi)^{cl} = \forall x \forall y(\neg Px \vee \neg Qyx)$.

Ejemplo 3

Entonces, tenemos que calcular resolventes a partir de las cláusulas que aparecen en las formas clausales anteriores hasta obtener la cláusula vacía $\square$. Recordemos que cuando se calcula un resolvente de dos cláusulas, hay que renombrar las variables que se repitan en las cláusulas, ya que las variables son locales en las cláusulas en las que aparecen.

En este ejemplo, tenemos que la variable x que aparece en $(\varphi_2)^{cl}$, la x que aparece en $(\varphi_3)^{cl}$ y la x que aparece en $(\neg\varphi)^{cl}$ se han de renombrar. Tenemos entonces las siguientes entradas:

Ejemplo 3

(1) Pa.

(2) $\neg Px_1 \vee Pf(x_1) \vee Pg(x_1)$.

(3) $Qx_2 f(x_2)$.

(4) $Qx_3 g(x_3)$.

(5) $\neg Px_4 \vee \neg Qy_4 x_4$.

En primer lugar, resolvemos (1) y (2). Observamos que los átomos Pa y Px_1 son unificables por $\{x_1 = a\}$. Por tanto, obtenemos:

(6) $Pf(a) \vee Pg(a)$.

Ejemplo 3

Ahora, resolvemos (5) y (6). Observamos que los átomos $Pf(a)$ y Px_4 son unificables por $\{x_4 = f(a)\}$. Por tanto, obtenemos:

(7) $Pg(a) \vee \neg Qy_4 f(a)$.

A continuación, resolvemos (3) y (7). Observamos que los átomos $Qx_2 f(x_2)$ y $Qy_4 f(a)$ son unificables por $\{x_2 = a, y_4 = a\}$. Por tanto, obtenemos:

(8) $Pg(a)$.

Ahora, resolvemos (5) y (8). Observamos que los átomos Px_4 y $Pg(a)$ son unificables por $\{x_4 = g(a)\}$. Por tanto, obtenemos:

(9) $\neg Qy_4 g(a)$.

Ejemplo 3

Por último, resolvemos (4) y (9). Observamos que los átomos $Qx_3g(x_3)$ y $Qy_4g(a)$ son unificables por $\{x_3 = a, y_4 = a\}$. Por tanto, obtenemos:

(10) $\square$.

El lenguaje de programación PROLOG

Es un lenguaje de programación declarativo, basado en la lógica de predicados, que se utiliza para interaccionar con bases de datos relacionales. Se utiliza fundamentalmente en la programación para la inteligencia artificial, siendo el diseño de los sistemas expertos una de sus principales aplicaciones.

En la página web www.swi-prolog.org está disponible una versión gratuita de Prolog.

Empezamos definiendo los conceptos más básicos de Prolog. Un identificador de Prolog es una secuencia de letras, dígitos y caracteres de subrayado en donde el primer carácter no es un dígito. Una variable en Prolog es entonces un identificador que comienza por una letra mayúscula o por el carácter de subrayado.

El lenguaje de programación PROLOG

En Prolog, tenemos los siguientes tipos de constantes:

- números enteros.
- números decimales (el tipo float).
- identificadores que empiezan por una letra minúscula.
- tiras de caracteres encerradas por comillas simples.

Los símbolos básicos de operador y de relación en Prolog son análogos a los símbolos de los lenguajes de programación usuales.

Fórmulas de Prolog

Una fórmula de Prolog es o bien una fórmula atómica φ de un lenguaje de predicados o bien una fórmula del tipo $(\varphi_1 \wedge \ldots \varphi_n) \rightarrow \varphi$ donde $\varphi_1, \ldots, \varphi_n, \varphi$ son fórmulas atómicas de un lenguaje de predicados.

Observamos que toda fórmula de Prolog es una cláusula de un lenguaje de predicados, ya que tenemos que $(\phi_1 \wedge \ldots \phi_n) \rightarrow \phi \equiv \neg\phi_1 \vee \ldots \neg\phi_n \vee \phi$. En Prolog, se utiliza la siguiente notación:

- El símbolo de conjunción se denota por una coma.
- El símbolo de implicación se denota por $:-$.
- Una fórmula de Prolog $(\phi_1 \wedge \ldots \phi_n) \rightarrow \phi$ se escribe como $\phi :- \phi_1, \ldots, \phi_n$.

Un programa en Prolog es entonces una secuencia finita de fórmulas de Prolog.

Programas y objetivos

Si φ es una fórmula de un programa en Prolog y $X_1, \ldots X_n$ son las variables que aparecen en φ, el significado de φ es la fórmula $\forall X_1 \ldots \forall X_n \varphi$. Es decir, suponemos que las variables que aparecen en los programas de Prolog están cuantificadas universalmente.

Cuando se tiene un programa en Prolog, se desea obtener información del programa. Para elo se escribe un objetivo, que se define como una fórmula del tipo $\varphi_1, \ldots, \varphi_n$ donde $n \geq 1$ y $\varphi_1, \ldots, \varphi_n$ son fórmulas atómicas de un lenguaje de predicados.

Si φ es un objetivo y $Z_1, \ldots, Z_k$ son las variables que aparecen en φ, el significado de φ es la fórmula $\exists Z_1 \ldots \exists Z_k \varphi$. Por tanto, suponemos que las variables que aparecen en un objetivo de Prolog están cuantificadas existencialmente.

Mecanismo de ejecución del Prolog

Si P es un programa de Prolog y ψ es un objetivo, el interpretador de Prolog intenta demostrar que ψ es consecuencia lógica del programa P, utilizando el algoritmo de resolución para la lógica de predicados.

Es decir, el interpretador de Prolog es un demostrador de teoremas, que valida razonamientos utilizando el método de resolución para la lógica de predicados.

Ejemplo de programa en Prolog

Vemos un programa en Prolog que calcula los antepasados de una persona en un árbol genealógico.

En primer lugar, representamos el árbol genealógico mediante una base de datos, utilizando los predicados:

padre(X,Y): "X es el padre de Y".

madre(X,Y): "X es la madre de Y".

antepasado(X,Y): "X es antepasado de Y".

Supongamos entonces que la siguiente base de datos representa el árbol genealógico:

padre(juan,pedro).

padre(pedro, vicente).

padre(vicente, alberto).

padre(alberto, luis).

.....

Ejemplo de programa en Prolog

madre(carmen, alberto).

madre(maria, pedro).

madre(ana, vicente).

madre(rosa, luis).

.....

Ahora, para completar el programa, incluimos las siguientes cláusulas para poder calcular los antepasados:

antepasado(X,Y) :- padre(X,Y).

antepasado(X,Y) :- madre(X,Y).

antepasado(X,Y) :- antepasado(X,Z), padre(Z,Y).

antepasado(X,Y) :- antepasado(X,Z), madre(Z,Y).

Obligatoriamente, al final de cada cláusula de un programa en Prolog se ha de poner un punto.

Ejemplo de programa en Prolog

Entonces, por lo explicado anteriormente, se tiene que el significado de la primera cláusula del programa anterior es la fórmula

$\forall x \forall y$ (padre(X,Y) $\rightarrow$ antepasado(X,Y)).

El significado de la segunda cláusula es la fórmula

$\forall x \forall y$ (madre(X,Y) $\rightarrow$ antepasado(X,Y)).

El significado de la tercera cláusula es la fórmula

$\forall x \forall y \forall z$ ((antepasado(X,Z) $\wedge$ padre(Z,Y)) $\rightarrow$ antepasado(X,Y)).

Y el significado de la cuarta cláusula es la fórmula

$\forall x \forall y \forall z$ ((antepasado(X,Z) $\wedge$ madre(Z,Y)) $\rightarrow$ antepasado(X,Y)).

Ejemplo de programa en Prolog

Ahora, para recabar información del programa, hemos de escribir un objetivo seguido de un punto. Por ejemplo, supongamos que ponemos como objetivo:

antepasado(X,alberto).

Entonces, el interpretador demostrará por resolución que el objetivo antepasado(X,alberto) es consecuencia lógica del programa, y nos dará como soluciones los antepasados de alberto.

Bibliografía para el tema 2

- Lógica para informáticos (R. Farré)

  Capítulos 4 y 5.

- Lógica simbólica para informáticos (P.J. Iranzo)

  Capítulo 5.