Introducción

En la primera parte de la clase de hoy, introduciremos algunos conceptos fundamentales para los lenguajes de predicados.

A continuación, empezaremos a estudiar el método de resolución para los lenguajes de predicados, el cual es una generalización del método de resolución que vimos para los lenguajes de proposiciones.

Tautologías, contradicciones y fórmulas satisfactibles

Decimos que una fórmula φ es una tautología, si φ es cierta en todas las interpretaciones.

Decimos que una fórmula φ es una satisfactible, si φ es cierta en alguna interpretación.

Y decimos que φ es contradicción , si φ es falsa en toda interpretación.

Procediendo como hicimos en lógica de proposiciones, podemos comprobar que una fórmula φ es una tautología si y sólo si $\neg\varphi$ es una contradicción.

El concepto de consecuencia lógica

Definimos a continuación el concepto de consecuencia lógica, el cual nos permite validar razonamientos.

Una fórmula φ es consecuencia lógica de fórmulas $\varphi_1, \ldots, \varphi_n$, si la fórmula $(\varphi_1 \wedge \ldots \wedge \varphi_n) \rightarrow \varphi$ es una tautología.

Si φ es consecuencia lógica de $\varphi_1, \ldots, \varphi_n$, escribiremos $\{\varphi_1, \ldots, \varphi_n\} \models \varphi$.

Y escribiremos $\varphi_1 \models \varphi_2$ en lugar de $\{\varphi_1\} \models \varphi_2$.

Demostradores

Un demostrador es un programa que determina si una fórmula φ es consecuencia lógica de fórmulas $\varphi_1, \ldots, \varphi_n$.

Los lenguajes de predicados se utilizan entonces para diseñar demostradores.

En el caso de las Matemáticas, los lenguajes de predicados se han utilizado para diseñar los demostradores OTTER y PROVER9, que son capaces de responder a preguntas en diferentes áreas de las Matemáticas.

Y en el caso de la Informática, hay lenguajes de programación, como es el caso del lenguaje Prolog, que están basados en la formalismo de la lógica de predicados, y cuyos interpretadores son demostradores de teoremas.

El método de resolución

Nuestro objetivo ahora es generalizar el método de resolución que vimos en la lógica de proposiciones a la lógica de predicados.

El método de resolución para la lógica de predicados se utiliza para poder diseñar demostradores.

Para poder definir el algoritmo de resolución para la lógica de predicados, necesitamos definir previamente el concepto de forma clausal de una fórmula.

Formas clausales

Al igual que en lógica de proposiciones, definimos un literal como un átomo o la negación de un átomo. Y definimos una cláusula como una disyunción (posiblemente vacía) de literales. Denotamos por $\square$ a la cláusula vacía.

Decimos entonces que una fórmula φ está en forma clausal, si φ es de la forma $\forall x_1 \ldots \forall x_n(\psi_1 \wedge \ldots \wedge \psi_m)$ donde $\psi_1, \ldots, \psi_m$ son cláusulas. A la fórmula $\psi_1 \wedge \ldots \wedge \psi_m$ la llamaremos núcleo de la forma clausal.

Vemos a continuación un algoritmo para calcular una forma clausal φ' de una fórmula de predicados φ.

Algoritmo para obtener una forma clausal de una fórmula

(1) Aplicar las siguientes equivalencias:

$\varphi \to \psi \equiv \neg\varphi \vee \psi$.

$\varphi \leftrightarrow \psi \equiv (\varphi \to \psi) \wedge (\psi \to \varphi)$.

(2) Aplicar las siguientes equivalencias:

$\neg(\neg\varphi) \equiv \varphi$.

$\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$.

$\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$.

$\neg(\varphi \to \psi) \equiv \varphi \wedge \neg\psi$.

$\neg\exists x\varphi \equiv \forall x\neg\varphi$.

$\neg\forall x\varphi \equiv \exists x\neg\varphi$.

Algoritmo para obtener una forma clausal de una fórmula

(3) Aplicar las siguientes equivalencias:

$Qx(\varphi \vee \psi) \equiv Qx\varphi \vee \psi$, si $Q \in \{\exists, \forall\}$ y x no aparece en ψ.

$Qx(\varphi \wedge \psi) \equiv Qx\varphi \wedge \psi$, si $Q \in \{\exists, \forall\}$ y x no aparece en ψ.

(4) Eliminar los cuantificadores existenciales.

Para ello, reemplazamos toda subfórmula de la forma $\exists z\psi$ por otra fórmula ψ' sin la variable z. Concretamente, ψ' se obtiene a partir de la fórmula ψ reemplazando todas las apariciones de z por un término t de manera que:

Algoritmo para obtener una forma clausal de una fórmula

(a) Si z no se encuentra en el ámbito de ninguna variable cuantificada universalmente, t es un nuevo símbolo de constante c.

(a) En caso contrario, $t = f(x_1, \ldots, x_n)$, donde f es un nuevo símbolo de función y $x_1, \ldots x_n$ son las variables cuantificadas universalmente, en cuyo ámbito se encuentra z.

(5) Renombrar las variables ligadas que sea necesario (para evitar conflictos de nombre).

(6) Mover los cuantificadores hacia fuera.

(7) Aplicar las reglas distributivas en el núcleo de la fórmula obtenida.

Ejemplo 1

Consideremos la fórmula $\varphi = \neg\exists x Px \vee \forall x Qx$. Tenemos entonces: $\varphi = \neg\exists x Px \vee \forall x Qx \equiv \forall x \neg Px \vee \forall x Qx \equiv \forall x \neg Px \vee \forall y Qy \equiv \forall x \forall y (\neg Px \vee Qy)$.

Por tanto, $\forall x \forall y (\neg Px \vee Qy)$ es una forma clausal de φ.

Ejemplo 2

Consideremos la fórmula $\varphi = \neg\exists x Px \vee \forall x \exists y Qxy$. Tenemos entonces:
$\varphi = \neg\exists x Px \vee \forall x \exists y Qxy \equiv \forall x \neg Px \vee \forall x \exists y Qxy$.

Ahora, aplicando la etapa (4) del algoritmo, sustituimos y por $f(x)$. Obtenemos entonces :

$\forall x \neg Px \vee \forall x Qxf(x) \equiv \forall x \neg Px \vee \forall y Qyf(y) \equiv$
$\forall x \forall y(\neg Px \vee Qyf(y))$.

Por tanto, $\forall x \forall y(\neg Px \vee Qyf(y))$ es una forma clausal de φ.

El teorema de la forma clausal

Este teorema afirma que si φ es una fórmula de un lenguaje de predicados y φ' es una forma clausal de φ, se tiene entonces que φ es una contradicción si y sólo si φ' es una contradicción.

Sin embargo, si φ es una fórmula de un lenguaje de predicados y φ' es una forma clausal de φ, en general no es cierto que φ y φ' son lógicamente equivalentes. Para ello, obsérvese que la fórmula Pc es una forma clausal de la fórmula $\exists x Px$, y hemos visto anteriormente que estas dos fórmulas no son lógicamente equivalentes.

El método de resolución

Para poder definir la noción de resolvente de dos cláusulas, necesitamos un criterio para poder emparejar fórmulas atómicas. Dicho criterio queda establecido por el llamado "algoritmo de unificación". Para mostrar dicho algoritmo, necesitamos previamente definir algunos conceptos.

Una sustitución es un conjunto finito $\{x_1 = t_1, \ldots, x_n = t_n\}$ donde $t_1, \ldots, t_n$ son términos de un lenguaje de predicados y $x_1, \ldots x_n$ son variables distintas.

Llamaremos expresión a un término o a una fórmula atómica.

Entonces, si e es una expresión y $\lambda = \{x_1 = t_1, \ldots, x_n = t_n\}$ es una sustitución, denotamos por $e\lambda$ a la expresión que resulta de sustituir en e toda aparición de x_i por t_i para $i = 1 \ldots n$.

Por ejemplo, si $e = Rcxf(y)$ y $\lambda = \{x = g(y), y = b\}$, tenemos que $e\lambda = Rcg(y)f(b)$.

El método de resolución

Si e, e' son dos expresiones y λ es una sustitución, decimos que λ es un unificador de e, e', si $e\lambda = e'\lambda$.

Por ejemplo, consideremos las expresiones $e_1 = Rcxf(g(y))$ y $e_2 = Rzf(z)f(u)$. Consideremos la sustitución $\lambda = \{z = c, x = f(c), u = g(y)\}$. Tenemos entonces:

$$e_1\lambda = e_2\lambda = Rcf(c)f(g(y)).$$

Por tanto, λ es un unificador de e_1 y e_2.

El algoritmo de unificación

El algoritmo de unificación determina si es posible unificar dos expresiones.

entrada : dos expresiones e_1, e_2.

salida : un unificador de e_1 y e_2 o "fallo".

variables : una pila P inicializada vacía, un vector λ inicializado vacío y una variable booleana b inicializada en false.

Las etapas del algoritmo de unificación son entonces las siguientes.

El algoritmo de unificación

(1) Poner $e_1 = e_2$ en la pila P.

(2) while (!P.empty && !b)

1. Sacar la igualdad $e = e'$ que esté en la cima de P.
2. Si e es una variable que no aparece en e', sustituir e por e' en los elementos de P y de λ y añadir la ecuación $e = e'$ a λ. Se procede análogamente, si e' es una variable que no aparece en e
3. Si e, e' son constantes o variables idénticas, continuar.
4. Si $e = f(s_1, \ldots, s_n)$ y $e' = f(t_1, \ldots, t_n)$ para algún símbolo de función f, poner en la pila P las ecuaciones $s_n = t_n$, $\ldots$,$s_1 = t_1$.
5. Si $e = Rs_1 \ldots s_n$ y $e' = Rt_1 \ldots t_n$ para algún símbolo de predicado R, poner en la pila P las ecuaciones $s_n = t_n$, $\ldots$,$s_1 = t_1$.
6. En otro caso, poner $b = true$.

(3) Si $(!b)$ salida $= \lambda$. Si no, salida $=$ "fallo".

Ejemplo 1

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

Inicialmente, tenemos que 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 1

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

Ejemplo 2

Aplicamos el algoritmo de unificación a las expresiones $e_1 = Rzf(z)f(u)$, $e_2 = Rcxf(g(y))$.

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 $Rzf(z)f(u) = Rcxf(g(y))$, es decir, tendremos $P = [Rzf(z)f(u) = Rcxf(g(y))]$. Y entramos entonces en el bucle while.

Sustituimos entonces en la pila la igualdad $Rzf(z)f(u) = Rcxf(g(y))$ por las igualdades $z = c$, $x = f(z)$ y $f(u) = f(g(y))$. Por tanto, tendremos $P = [z = c, x = f(z), f(u) = f(g(y))]$.

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

Ejemplo 2

Ahora, sacamos de la pila la ecuación $x = f(c)$, y metemos dicha ecuación en el vector λ. Por tanto, tendremos $P = [f(u) = f(g(y))]$, $\lambda = \{z = c, x = f(c)\}$.

A continuación, reemplazamos en la pila la ecuación $f(u) = f(g(y))$ por la ecuación $u = g(y)$ (aplicando de nuevo la etapa (2) del algoritmo). Por tanto, tendremos $P = [u = g(y)]$, $\lambda = \{z = c, x = f(c)\}$.

Ahora, sacamos de la pila la ecuación $u = g(y)$, y metemos dicha ecuación en el vector λ. Por tanto, tendremos $P = [\,]$, $\lambda = \{z = c, x = f(c), u = g(y)\}$.

Así pues, como la pila está vacía, salimos del bucle while (etapa (2) del algoritmo). Entonces, en la etapa (3), el algoritmo de unificación da como salida la substitución λ, que es un unificador de las entradas e_1 y e_2.. Por tanto, las expresiones de la entrada e_1, e_2 son unificables.

Concepto de resolvente

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.

Ejemplo 1

Consideremos las cláusulas $\varphi_1 = Px \vee Qf(x)$ y $\varphi_2 = Rxy \vee \neg Qx$. Para calcular un resolvente de φ_1 y φ_2, en primer lugar hemos de renombrar una de las dos variables x, por ejemplo cambiamos la variable x de φ_1 por u. Claramente, los átomos $Qf(u)$ y Qx son unificables por $\lambda = \{x = f(u)\}$. Por tanto, obtenemos como resolvente de φ_1 y φ_2 la cláusula

$$(Pu \vee Rxy)\lambda = Pu \vee Rf(u)y.$$

Ejemplo 2

Consideremos las cláusulas

$$\varphi_1 = Px \vee \neg Qf(x)y \vee Rbx, \varphi_2 = \neg Pa \vee Qf(c)z.$$

Hay dos posibles resolventes para φ_1 y φ_2. En primer lugar, observamos que los átomos Px, Pa son unificables por $\{x = a\}$. Por tanto, obtenemos el resolvente

$$\neg Qf(a)y \vee Rba \vee Qf(c)z.$$

Por otra parte, observamos que los átomos $Qf(x)y$ y $Qf(c)z$ son unificables por $\{x = c, y = z\}$. Por tanto, obtenemos el resolvente

$$Pc \vee Rbc \vee \neg Pa.$$

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 $\{z = f(a)\}$