LENGUAJES DE PROPOSICIONES

Febrero de 2023

Descripción básica

El objeto central de la lógica es el estudio de los razonamientos, es decir, el estudio de los procesos de inferencia que a partir de las premisas permiten obtener conclusiones correctas.

Y en la lógica aplicada a la informática, nos interesamos por los procesos de inferencia que se pueden implementar en un computador.

Entre las principales aplicaciones de la lógica en la informática, podemos citar las siguientes:

Descripción básica

(1) Diseño de SAT-solvers (programas basados en lógica que permiten resolver muchos problemas prácticos).
(2) Verificación de circuitos computacionales.
(3) Verificación de programas.
(4) Programación declarativa.

Descripción básica

Hay lenguajes de programación que están basados en el formalismo de la lógica. Para poder entender programas escritos en estos lenguajes de programación es entonces necesario conocer los conceptos básicos de la lógica.

En primer lugar, estudiaremos los lenguajes lógicos más simples, que son los lenguajes proposicionales.

El concepto de proposición

Por una proposición entendemos una frase declarativa, es decir, una frase de la que tiene sentido decir si es verdadera o falsa.

Las siguienes frases son ejemplos de proposiciones:

(1) 6 es divisor de 18.

(2) $7 > \sqrt{50}$.

(3) Si llueve, las calles se mojan.

(4) Hoy es festivo.

Las conectivas lógicas

Antes de definir el concepto de lenguaje de proposiciones, recordemos las definiciones de las conectivas lógicas.

Representamos por V y F a los valores de verdad. Es decir, denotamos por V al valor "verdadero" y por F al valor "falso".

(1) Si φ es una proposición, denotamos por $\neg\varphi$ a su negación. Por tanto, tenemos la siguiente tabla de verdad:

φ	$\neg\varphi$
V	F
F	V

Las conectivas lógicas

(2) Si φ, ψ son proposiciones, su disyunción la denotamos por $\varphi \vee \psi$. Por tanto, tenemos la siguiente tabla de verdad:

φ	ψ	$\varphi \vee \psi$
V	V	V
V	F	V
F	V	V
F	F	F

Las conectivas lógicas

(3) Si φ, ψ son proposiciones, su conjunción la denotamos por $\varphi \wedge \psi$. Tenemos entonces la siguiente tabla de verdad:

φ	ψ	$\varphi \wedge \psi$
V	V	V
V	F	F
F	V	F
F	F	F

Las conectivas lógicas

(4) Si φ, ψ son proposiciones, la proposición $\varphi \to \psi$ se llama proposición condicional, y se lee "si φ entonces ψ" o "φ implica ψ". La tabla de verdad para esta conectiva lógica es la siguiente:

φ	ψ	$\varphi \to \psi$
V	V	V
V	F	F
F	V	V
F	F	V

En una proposición $\varphi \to \psi$, diremos que φ es el antecedente y ψ el consecuente.

Las conectivas lógicas

(5) Si φ, ψ son proposiciones, la proposición $\varphi \leftrightarrow \psi$ representa la equivalencia de φ y ψ. La tabla de verdad para esta conectiva es, por tanto, la siguiente:

φ	ψ	$\varphi \leftrightarrow \psi$
V	V	V
V	F	F
F	V	F
F	F	V

El concepto de lenguaje de proposiciones

Por un átomo entendemos una frase declarativa que no puede descomponerse en frases más simples. Sea σ un conjunto finito de átomos. Definimos entonces el lenguaje de las σ-fórmulas proposicionales como el conjunto de elementos generados por las siguientes reglas:

(1) Todo elemento de σ es una fórmula.

(2) Si φ es una fórmula, $\neg\varphi$ también lo es.

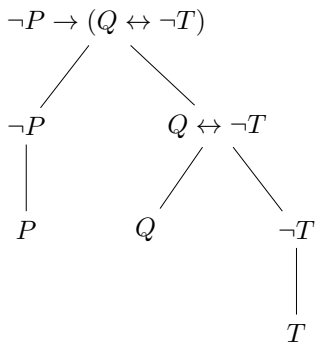
(3) Si φ, ψ son fórmulas, entonces $(\varphi \vee \psi)$, $(\varphi \wedge \psi)$, $(\varphi \to \psi)$ y $(\varphi \leftrightarrow \psi)$ son también fórmulas.

Árboles genealógicos

Dada una fórmula de un lenguaje de proposiciones, podemos representar la manera en que la fórmula se ha construido aplicando las reglas anteriores por medio de un árbol, al que llamaremos árbol de generación (o árbol genealógico) de la fórmula.

Por ejemplo, para la fórmula $\neg P \to (Q \leftrightarrow \neg T)$ tenemos el siguiente árbol de generación:

Ejemplo de árbol genealógico



Formalizaciones

Muchas frases declarativas del lenguaje natural se pueden formalizar en lógica de proposiciones. Para ello, se utilizan las siguientes reglas:

Reglas de formalización.

(1) Una frase de la forma "A o B" se representa por $A \vee B$.
(2) Una frase de la forma "A y B" se representa por $A \wedge B$.
(3) Una frase de la forma "si A entonces B" (es decir, es suficiente que se cumpla A para que se cumpla B) se representa por $A \to B$.
(4) Una frase de la forma "B sólo si A" (es decir, es necesario que se cumpla la condición A para que se cumpla la condición B) se representa por $B \to A$.
(5) Una frase de la forma "A si y sólo si B" se representa por $A \leftrightarrow B$.

Ejemplo

Consideremos la frase:

"LLoverá, si hay nubes".

Átomos:

$L =$ lloverá,
$N =$ hay nubes.

La frase se formaliza por $N \to L$. Se observa que la frase es falsa.

Ejemplo

Consideremos la frase:

"LLoverá, sólo si hay nubes".

Átomos:

$L =$ lloverá,
$N =$ hay nubes.

La frase se formaliza por $L \to N$. Se observa que la frase es verdadera.

Ejemplo

Consideremos la frase:

"Hay que ser hábil y tener suerte para ganar".

Átomos:

$H =$ ser hábil,
$S =$ tener suerte,
$G =$ ganar.

La frase se formaliza por $G \to (H \wedge S)$.

Interpretaciones

Si σ un conjunto finito de átomos, definimos una σ-interpretación como una asignación de los valores de verdad a los elementos de σ.

Para evaluar una fórmula φ en una interpretación I, se sustituye cada átomo A que aparezca en φ por $I(A)$ (es decir, por el valor de verdad asociado a A).

Representamos por $I(\varphi)$ al resultado de evaluar φ en I.

Ejemplo

Consideremos $\sigma = \{P, Q, R\}$. Definimos la interpretación I por $I(P) = F$, $I(Q) = F$ e $I(R) = V$. Consideremos la fórmula $\varphi = (P \leftrightarrow Q) \wedge R$. Entonces,

$$I(\varphi) = (F \leftrightarrow F) \wedge V = V \wedge V = V.$$

Tautologías, contradicciones y fórmulas satisfactibles

(1) Una fórmula φ es una tautología, si φ es cierta en todas las interpretaciones.
(2) Una fórmula φ es satisfactible, si φ es cierta en alguna interpretación.
(3) Una fórmula φ es una contradicción (o una fórmula insatisfactible), si φ es falsa en todas las interpretaciones.

Construyendo la tabla de verdad de una fórmula, se puede saber si la fórmula es tautología, satisfactible o contradicción.

Ejemplos

La fórmula $P \vee \neg P$ es una tautología.

La fórmula $P \wedge \neg P$ es una contradicción.

La fórmula $P \to \neg P$ es satisfactible pero no tautología, ya que si P es verdad la fórmula es falsa, y si P es falso la fórmula es verdadera.

La fórmula $((P \to Q) \wedge P) \to Q$ es una tautología.

La fórmula $(P \vee Q) \wedge (\neg P \wedge \neg Q)$ es una contradicción, ya que ninguna interpretación satisface la fórmula.

Fórmulas equivalentes

Decimos que dos fórmulas φ, ψ son lógicamente equivalentes, si para toda interpretación I, tenemos que $I(\varphi) = I(\psi)$.

Si φ, ψ son lógicamente equivalentes, escribiremos $\varphi \equiv \psi$.

Las siguientes equivalencias lógicas se comprueban mediante tablas de verdad.

Tabla de equivalencias lógicas

(1) $\varphi \leftrightarrow \psi \equiv (\varphi \to \psi) \wedge (\psi \to \varphi)$.
(2) $\varphi \to \psi \equiv \neg\varphi \vee \psi$.
(3) $\varphi \vee \psi \equiv \psi \vee \varphi$, $\varphi \wedge \psi \equiv \psi \wedge \varphi$.
(4) $(\varphi \vee \psi) \vee \chi \equiv \varphi \vee (\psi \vee \chi)$, $(\varphi \wedge \psi) \wedge \chi \equiv \varphi \wedge (\psi \wedge \chi)$.
(5) $\varphi \vee V \equiv V$, $\varphi \vee F \equiv \varphi$, $\varphi \wedge V \equiv \varphi$, $\varphi \wedge F \equiv F$.
(6) $\varphi \vee (\psi \wedge \chi) \equiv (\varphi \vee \psi) \wedge (\varphi \vee \chi)$,
$\varphi \wedge (\psi \vee \chi) \equiv (\varphi \wedge \psi) \vee (\varphi \wedge \chi)$.
(7) $(\varphi \vee \psi) \wedge \varphi \equiv \varphi$, $(\varphi \wedge \psi) \vee \varphi \equiv \varphi$,
(8) $\neg(\neg\varphi) \equiv \varphi$.
(9) $\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$.
(10) $\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$.
(11) $\neg(\varphi \to \psi) \equiv \varphi \wedge \neg\psi$.

Observaciones

A las reglas 1 y 2 de la tabla anterior se las llama regla de la implicación y regla de la equivalencia. La regla 3 es la ley conmutativa. La regla 4 es la ley asociativa. La regla 5 es la ley de identidad. La regla 6 es la ley distributiva. La regla 7 es la ley de absorción. Y las reglas 8-11 son las reglas de la negación.

Demostración de la equivalencia 11

φ	ψ	$\varphi \to \psi$	$\neg(\varphi \to \psi)$	$\neg\psi$	$\varphi \wedge \neg\psi$
V	V	V	F	F	F
V	F	F	V	V	V
F	V	V	F	F	F
F	F	V	F	V	F

Se observa que las fórmulas $\neg(\varphi \to \psi)$ y $\varphi \wedge \neg\psi$ tienen la misma tabla de verdad. Por tanto, $\neg(\varphi \to \psi)$ y $\varphi \wedge \neg\psi$ son equivalentes.

Introducción

En la clase de hoy, introduciremos el concepto de demostrador, y mostraremos un método para poder diseñar demostradores, que es el llamado "método de resolución". Los demostradores se utilizan para poder diseñar verificadores de sistemas informáticos, y son también importantes en la programación declarativa, debido a que algunos intérpretes de lenguajes declarativos, como es el caso del lenguaje Prolog, son demostradores.

El concepto de fórmula en forma conjuntiva

(1) Un literal es un átomo o a la negación de un átomo.
(2) Una cláusula es una disyunción (posiblemente vacía) de literales.
(3) Una fórmula φ está en forma normal conjuntiva (FNC), si φ es de la forma $\phi_1 \wedge \ldots \wedge \phi_n$ donde $\phi_1, \ldots, \phi_n$ son cláusulas.

Representamos por $\square$ a la cláusula vacía. Se tiene entonces que $\square$ es una contradicción, ya que una interpretación I satisface una disyunción de fórmulas $\varphi_1 \vee \ldots \vee \varphi_n$ si y sólo si hay un $i \in \{1, \ldots, n\}$ tal que I satisface φ_i. Por tanto, como $\square$ es una disyunción vacía, no hay ninguna interpretación I que satisface $\square$.

Algoritmo para transformar una fórmula en forma normal conjuntiva

(1) Aplicar las equivalencias
$\varphi \to \psi \equiv \neg\varphi \vee \psi$.
$\varphi \leftrightarrow \psi \equiv (\varphi \to \psi) \wedge (\psi \to \varphi)$.
(2) Aplicar las equivalencias
$\neg(\neg\varphi) \equiv \varphi$.
$\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$.
$\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$.
(3) Aplicar las equivalencias
$\varphi \vee (\psi \wedge \chi) \equiv (\varphi \vee \psi) \wedge (\varphi \vee \chi)$,
$\varphi \wedge (\psi \vee \chi) \equiv (\varphi \wedge \psi) \vee (\varphi \wedge \chi)$.

Ejemplo

Calculamos una FNC de la fórmula $(P \vee \neg Q) \to R$.

Tenemos entonces

$(P \vee \neg Q) \to R \equiv \neg(P \vee \neg Q) \vee R \equiv (\neg P \wedge Q) \vee R \equiv$
$(\neg P \vee R) \wedge (Q \vee R)$.

Ejemplo

Calculamos una FNC de la fórmula $(P \wedge (Q \to R)) \to S$.

Tenemos entonces

$(P \wedge (Q \to R)) \to S \equiv (P \wedge (\neg Q \vee R)) \to S \equiv$
$\neg(P \wedge (\neg Q \vee R)) \vee S \equiv (\neg P \vee \neg(\neg Q \vee R)) \vee S \equiv$
$\neg P \vee (Q \wedge \neg R) \vee S \equiv \neg P \vee S \vee (Q \wedge \neg R) \equiv$
$(\neg P \vee S \vee Q) \wedge (\neg P \vee S \vee \neg R)$.

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 $(\varphi_1 \wedge \ldots \wedge \varphi_n) \to \varphi$ 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 $\{\varphi_1\} \models \varphi_2$.

Ejemplos

(1) $\{(P \wedge Q) \to R, P, Q\} \models R$.

(2) $\{P \to Q, \neg Q\} \models \neg P$.

(3) $(P \to Q) \models (\neg Q \to \neg P)$.

Demostradores

El objetivo de la demostración automática es encontrar métodos para validar razonamientos que se puedan implementar en un computador. A través de dichos métodos se diseñan los llamados demostradores, que son programas que determinan si una fórmula φ es consecuencia lógica de fórmulas $\varphi_1, \ldots, \varphi_n$.

Los demostradores tienen importantes aplicaciones tanto en la Informática como en la Matemática. En el caso de la Matemática, se han diseñado demostradores de teoremas, como son los programas OTTER y PROVER9, que se han utilizado para responder a preguntas en diferentes áreas de las Matemáticas.

Demostradores

En el campo de la Informática, los demostradores tienen aplicaciones a la programación y a la verificación de sistemas informáticos. En lo concerniente a la programación, se tiene que hay lenguajes de programación, como el lenguaje Prolog, cuyos interpretadores son demostradores de teoremas. Y en el caso de la verificación de sistemas, los demostradores automáticos se utilizan para demostrar que un sistema informático funciona correctamente y no tiene errores. En el caso, por ejemplo, de los sistemas de hardware, los demostradores automáticos se utilizan para demostrar que los circuitos de la CPU de un ordenador están correctamente diseñados.

El método de resolución

Para validar un razonamiento en lógica de proposiciones hemos de demostrar que la conclusión φ del razonamiento es consecuencia lógica de las premisas $\varphi_1, \ldots, \varphi_n$. Es decir, hemos de demostrar que la fórmula $(\varphi_1 \wedge \ldots \wedge \varphi_n) \rightarrow \varphi$ es una tautología.

Para ello, hay que demostrar que dicha fórmula es cierta en todas las interpretaciones. Sin embargo, el método de generar todas las interpretaciones y ver si alguna de ellas satisface la fórmula es imposible de llevar a la práctica, ya que el número de interpretaciones crece exponencialmente.

Un método más eficiente es el método sintáctico de resolución, que se utiliza en la demostración automática de teoremas y en el cual está basado el diseño de muchos demostradores.

Definición de resolvente

Para dos cláusulas proposicionales φ_1, φ_2, si existe un literal ψ_1 en φ_1 que es complementario de un literal ψ_2 en φ_2, entonces se suprimen ψ_1, ψ_2 de φ_1, φ_2 respectivamente y se construye la disyunción de las cláusulas resultantes. La cláusula así construida se llama resolvente de φ_1 y φ_2.

Por ejemplo, si $\varphi_1 = \neg P \vee Q \vee S$ y $\varphi_2 = \neg Q \vee T$, se tiene que la fórmula $\varphi = \neg P \vee S \vee T$ es un resolvente de φ_1 y φ_2.

Y si $\varphi_1 = P$ y $\varphi_2 = \neg P$, entonces $\square$ es resolvente de φ_1 y φ_2.

Regla de resolución

entradas: dos cláusulas φ_1 y φ_2.

salida: un resolvente de φ_1 y φ_2.

Si $\varphi_1, \ldots, \varphi_n, \varphi$ son cláusulas, escribiremos $\{\varphi_1, \ldots, \varphi_n\} \vdash_R \varphi$ si existe una demostración de φ, tomando a $\varphi_1, \ldots, \varphi_n$ como entradas, utilizando únicamente la regla de resolución.

Ejemplo

Demostramos por resolución que la cláusula vacía $\square$ se deduce del conjunto de cláusulas $\{P \vee Q \vee \neg R, \neg P, P \vee Q \vee R, P \vee \neg Q\}$.

Tenemos la siguiente prueba por resolución:

1	$P \vee Q \vee \neg R$	entrada
2	$\neg P$	entrada
3	$P \vee Q \vee R$	entrada
4	$P \vee \neg Q$	entrada
5	$P \vee Q$	(1,3)
6	P	(4,5)
7	$\square$	(2,6)

Ejemplo

Demostramos por resolución que la cláusula vacía $\square$ se deduce del conjunto de cláusulas $\{P \vee Q, \neg P \vee Q, P \vee \neg Q, \neg P \vee \neg Q\}$.

Tenemos la siguiente prueba por resolución:

1	$P \vee Q$	entrada
2	$\neg P \vee Q$	entrada
3	$P \vee \neg Q$	entrada
4	$\neg P \vee \neg Q$	entrada
5	Q	(1,2)
6	$\neg Q$	(3,4)
7	$\square$	(5,6)

El teorema de resolución

Sean $\varphi_1, \ldots, \varphi_n, \varphi$ fórmulas de un lenguaje de proposiciones. Sea $\psi_1 \wedge \ldots \wedge \psi_k$ una forma normal conjuntiva de $\varphi_1 \wedge \ldots \wedge \varphi_n \wedge \neg\varphi$. Entonces,

$$\{\varphi_1, \ldots, \varphi_n\} \models \varphi \text{ si y sólo si } \{\psi_1, \ldots, \psi_k\} \vdash_R \square.$$

El algoritmo de resolución

El teorema de resolución da lugar al siguiente algoritmo para validar razonamientos en lenguajes de proposiciones.

entradas: fórmulas $\varphi_1, \ldots, \varphi_n, \varphi$.

salida: "éxito" si φ es consecuencia lógica de $\varphi_1, \ldots, \varphi_n$, o "fallo" en caso contrario.

(1) Calcular una FNC $\psi_1 \wedge \ldots \wedge \psi_k$ de la fórmula $\varphi_1 \wedge \ldots \wedge \varphi_n \wedge \neg\varphi$.

(2) Calcular resolventes a partir de las entradas $\psi_1, \ldots \psi_k$ hasta que o bien se obtenga $\square$ o bien no se puedan calcular más resolventes.

Ejemplo

Demostramos por resolución que la fórmula $P \to Q$ es consecuencia lógica del conjunto de fórmulas $\{T \to \neg P, S \to R, \neg Q \to U, R \to T, U \to S\}$. Para ello, tenemos que demostrar por resolución que la fórmula $(T \to \neg P) \wedge (S \to R) \wedge (\neg Q \to U) \wedge (R \to T) \wedge (U \to S) \wedge \neg(P \to Q)$ es una contradicción. Tenemos que $T \to \neg P \leftrightarrow \neg T \vee \neg P$, $S \to R \leftrightarrow \neg S \vee R$, $\neg Q \to U \leftrightarrow Q \vee U$, $R \to T \leftrightarrow \neg R \vee T$, $U \to S \leftrightarrow \neg U \vee S$ y $\neg(P \to Q) \leftrightarrow P \wedge \neg Q$. Tenemos entonces la siguiente prueba por resolución:

Ejemplo

1	$\neg T \vee \neg P$	input
2	$\neg S \vee R$	input
3	$Q \vee U$	input
4	$\neg R \vee T$	input
5	$\neg U \vee S$	input
6	P	input
7	$\neg Q$	input
8	$\neg T$	(1,6)
9	$\neg R$	(4,8)
10	$\neg S$	(2,9)
11	$\neg U$	(5,10)
12	Q	(3,11)
13	$\square$	(7,12)

Ejemplo

Utilizando el método de resolución, vamos a validar el siguiente razonamiento:

"Si ningún banco concede interés, todos los ciudadanos dejan de ahorrar o los precios suben. Si los precios suben, todos los ciudadanos dejan de ahorrar. Hay ciudadanos que no dejan de ahorrar. Por tanto, hay bancos que conceden interés."

Tenemos los siguientes átomos en el razonamiento:

$P =$ hay bancos que conceden interés,

$Q =$ todos los ciudadanos dejan de ahorrar,

$R =$ los precios suben.

Ejemplo

Las premisas del razonamiento son:

$\varphi_1 = \neg P \to (Q \vee R)$,

$\varphi_2 = R \to Q$,

$\varphi_3 = \neg Q$

Y la conclusión del razonamiento es P.

Tenemos entonces la siguiente demostración por resolución:

Ejemplo

1	$P \vee Q \vee R$	input
2	$\neg R \vee Q$	input
3	$\neg Q$	input
4	$\neg P$	input
5	$P \vee Q$	(1,2)
6	P	(3,5)
7	$\square$	(4,6)

Por tanto, el razonamiento es correcto.

Introducción

En la primera parte de la clase de hoy trataremos sobre los llamados SAT-solvers, que son programas basados en la lógica de proposiciones que nos permiten resolver muchos problemas prácticos. Veremos entonces algunos ejemplos de problemas que pueden ser resueltos por los SAT-solvers. Y a continuación, mostraremos el método en el que está basado el diseño de los SAT-solvers, que es el llamado método de Davis-Putnam.

El Problema SAT

El problema SAT. Es el problema de decidir si una fórmula proposicional en FNC es satisfactible.

El interés del problema SAT radica en que hay muchos problemas prácticos que se pueden representar mediante una fórmula de un lenguaje de proposiciones. Entonces, el que el problema tenga o no solución depende de si la fórmula asociada al problema es satisfactible.

Un SAT-solver es un programa que resuelve el problema SAT.

Los SAT-solvers están muy estudiados y son capaces de tratar fórmulas grandes.

Ejemplos de problemas que pueden ser resueltos por SAT-solvers

(1) El problema de distribuir ordenadores portátiles a los profesores de una escuela, de manera que profesores que tienen alguna hora de clase en común no compartan portátil.

(2) El problema de colorear un mapa con cuatro colores de manera que no haya dos países vecinos que tengan el mismo color.

(3) El problema de instalar paquetes de actualización en un ordenador satisfaciendo una serie de restricciones.

(4) El problema de formar un equipo de personas lo más pequeño posible para realizar una serie de tareas.

(5) El problema de trazar rutas de transporte o comunicaciones.

(6) El problema de asignar recursos en procesos industriales.

(7) Problemas para confeccionar horarios de hospitales, escuelas o líneas aéreas.

(8) La resolución de sudokus.

Ejemplo 1

Mostramos el siguiente problema, que puede ser resuelto mediante un SAT-solver.
Queremos colorear el mapa de un continente con cuatro colores de manera que no haya dos países vecinos que tengan el mismo color. Sea P el conjunto de países del continente. Representamos a los cuatro colores por los números 1,2,3,4. Representamos entonces el problema mediante una fórmula proposicional en forma normal conjuntiva de manera que pueda ser resuelto por un SAT-solver. Para ello, para $i \in P$ y $j \in \{1,2,3,4\}$, consideramos la proposición Rij que significa que al país i se le asigna el color j.

Tenemos que formalizar lo siguiente:

Ejemplo 1

(1) A cada país se le asigna un color.

Para cada $i \in P$ ponemos la cláusula

$Ri1 \vee Ri2 \vee Ri3 \vee Ri4$.

(2) Ningún país tiene asignado más de un color.

Para todo $i \in P$ y para todo $j, j' \leq 4$ con $j \neq j'$, ponemos la cláusula

$\neg Rij \vee \neg Rij'$.

(3) Los países vecinos no comparten color.

Para todo $i, i' \in P$ tales que i, i' son países vecinos y para todo $j \leq 4$, ponemos la cláusula

$\neg Rij \vee \neg Ri'j$.

La fórmula buscada es entonces la conjunción de las cláusulas de (1), (2) y (3).

Ejemplo 2

Mostramos el siguiente problema, que puede ser resuelto mediante un SAT-solver.
Tenemos un país con n aeropuertos y queremos que en cada vuelo haya un control antidrogas en el aeropuerto de salida o en el aeropuerto de llegada. Tenemos la lista L de los vuelos existentes formada por pares (i, j) donde i es el aeropuerto de salida del vuelo y j es el aeropuerto de llegada. Además disponemos de k equipos de policía. El problema consiste en determinar los aeropuertos donde se han de situar los equipos de policía.
Se pide entonces representar el problema mediante una fórmula proposicional en forma normal conjuntiva de manera que pueda ser resuelto por un SAT-solver. Para ello, para $i \in \{1, \ldots, k\}$ y $j \in \{1, \ldots, n\}$, considerar la proposición Pij que significa que el "i-ésimo equipo de policía ha de ir al aeropuerto j".

Tenemos que formalizar lo siguiente:

Ejemplo 2

(1) Cada vuelo tiene un equipo de policía en el aeropuerto de origen o en el aeropuerto de destino.

Para cada vuelo (i, j) de la lista L ponemos la cláusula

$P1i \vee P1j \vee P2i \vee P2j \vee \ldots \vee Pki \vee Pkj$.

(2) Ningún equipo de policía puede estar en dos aeropuertos.

Para todo $i \leq k$ y para todo $j, j' \leq n$ con $j \neq j'$, ponemos la cláusula

$\neg Pij \vee \neg Pij'$.

(3) En ningún aeropuerto puede haber dos equipos de policía.

Para todo $i, i' \leq k$ con $i \neq i'$ y para todo $j \leq n$, ponemos la cláusula

$\neg Pij \vee \neg Pi'j$.

La fórmula buscada es entonces la conjunción de las cláusulas de (1), (2) y (3).

Resolución de un sudoku mediante un SAT-solver

Para cada fila i $(1 \leq i \leq 9)$, para cada columna j $(1 \leq j \leq 9)$ y para cada valor k $(1 \leq k \leq 9)$, consideramos el átomo $Rijk$ con el significado "en la fila i columna j del sudoku está el valor k". Representamos entonces el problema mediante una fórmula en FNC con las siguientes cláusulas:

(1) En toda casilla del sudoku hay un valor.

Para cada $i, j \in \{1, \ldots, 9\}$, ponemos la cláusula $Rij1 \vee Rij2 \ldots \vee Rij9$.

(2) En ninguna casilla del sudoku hay más de un valor.

Para cada $i, j \in \{1, \ldots, 9\}$ y para cada $k, k' \in \{1, \ldots, 9\}$ con $k \neq k'$, ponemos la cláusula $\neg(Rijk \wedge Rijk') \equiv \neg Rijk \vee \neg Rijk'$.

Resolución de un sudoku mediante un SAT-solver

(3) En cada fila (o columna o cuadrado de 3×3) ningún valor se repite.

Consideremos el caso de las filas. Para cada $i \in \{1, \ldots, 9\}, j, j' \in \{1, \ldots 9\}$ con $j \neq j'$ y $k \in \{1, \ldots, 9\}$, ponemos la cláusula $\neg(Rijk \wedge Rij'k) \equiv \neg Rijk \vee \neg Rij'k$. Análogamente, se procede para las columnas.

Consideremos ahora el cuadrado $[1, 3] \times [1, 3]$. Para cada $i, i', j, j' \in [1, 3]$ con $(i, j) \neq (i', j')$ y para cada $k \in \{1, \ldots, 9\}$, ponemos la cláusula $\neg(Rijk \wedge Ri'j'k) \equiv \neg Rijk \vee \neg Ri'j'k$. Análogamente, procedemos para el resto de los cuadrados de 3×3.

(4) Ponemos los átomos correspondientes a los números puestos de antemano en el sudoku.

Si en la fila i columna j del sudoku tenemos el valor k puesto de antemano, ponemos el átomo $Rijk$.

Resolución de un sudoku mediante un SAT-solver

Ahora, tomamos la conjunción φ de las cláusulas consideradas en (1), (2), (3) y (4). Se tiene entonces que el sudoku tiene solución si y sólo si la fórmula φ es satisfactible. Además, si el sudoku tiene solución, la interpretación que hace cierta la fórmula φ es la solución del sudoku.

Un programa en JAVA para un SAT-solver

No es difícil escribir en Java un programa para determinar si una fórmula de un lenguaje de proposiciones es satisfactible. El programa genera mediante un bucle "for" anidado todas las posibles interpretaciones, y determina entonces si alguna de ellas satisface la fórmula. En concreto, para fórmulas de tres variables, podemos escribir el siguiente programa en JAVA, en el que ponemos como entrada la fórmula $(P_1 \vee P_2) \wedge (\neg P_1 \vee P_3)$.

Un programa en JAVA para un SAT-solver

```
class Satisfactible
{ public static void main(String [ ] args)
{ boolean b = false;
  boolean [ ] booleanos = {true, false};
  for (boolean P1 : booleanos)
  { for (boolean P2 : booleanos)
  { for (boolean P3 : booleanos)
  { boolean formula = ((P1 || P2) && ( ! P1 || P3));
    if (formula) { b = true;
                   System.out.println("la formula es satisfactible para:");
                   System.out.println("P1=" + P1);
                   System.out.println("P2=" + P2);
                   System.out.println("P3=" + P3);
                   break; }}
    if (b) break; }
    if (b) break; }
    if (!b) System.out.println("la formula es insatisfactible"); }}
```

Un programa en JAVA para un SAT-solver

El programa anterior sirve únicamente para fórmulas que tengan pocas cláusulas, ya que el número de interpretaciones crece exponencialmente con respecto al número de símbolos de proposición. Para poder escribir programas que resuelvan el problema SAT y que puedan tratar con fórmulas que tengan muchas cláusulas, hemos de utilizar el algoritmo de Davis y Putnam, que es el algoritmo en el que están basados los diseños de los SAT-solvers. Este método de Davis y Putnam para determinar si una fórmula proposicional es satisfactible es mucho más eficiente que el algoritmo consistente en generar todas las interpretaciones y ver si alguna de ellas satisface la fórmula.

El método de Davis-Putnam

Para introducir el método de Davis y Putnam, necesitamos algunas definiciones previas.

Si ψ es un literal, escribimos $\sim \psi = \neg\psi$ si ψ es un átomo, y escribimos $\sim \psi = \chi$ si $\psi = \neg\chi$ donde χ es un átomo.

Denotamos por $\square$ a la cláusula vacía. Como $\square$ es una disyunción vacía, $\square$ es una contradicción, ya que una interpretación I satisface una disyunción de fórmulas $\varphi_1 \vee \ldots \vee \varphi_n$ si y sólo si hay un $i \in \{1, \ldots, n\}$ tal que I satisface φ_i. Por tanto, como $\square$ es una disyunción vacía, no hay ninguna interpretación I que satisface $\square$.

Y denotamos por $\blacksquare$ a la conjunción vacía. Por ser una conjunción vacía, $\blacksquare$ es una tautología, ya que una interpretación I satisface una conjunción de fórmulas $\varphi_1 \wedge \ldots \wedge \varphi_n$ si y sólo si para todo $i \in \{1, \ldots, n\}$, I satisface φ_i. Por tanto, como $\blacksquare$ es una conjunción vacía, toda interpretación I satisface $\blacksquare$.

El método de Davis-Putnam

El método de Davis y Putnam consiste en añadir a la regla de resolución otras tres reglas, que preservan la satisfactibilidad de la fórmula de la entrada. Las cuatro reglas reciben entonces como entrada una fórmula en FNC y dan como salida otra fórmula en FNC. Se trata entonces de aplicar dichas reglas a la fórmula de entrada φ hasta llegar o bien a $\square$ o bien a $\blacksquare$. Si llegamos a $\blacksquare$ se tiene que φ es satisfactible, y si llegamos a $\square$ se tiene que φ es una contradicción.

Las reglas de Davis y Putnam son entonces las siguientes:

Regla de la tautología

La denotaremos por (TAU). La entrada es una fórmula φ en FNC. Y la salida es la fórmula φ^* que resulta de eliminar en φ las cláusulas que contengan un par complementario.

Se tiene que φ es satisfactible si y sólo si φ^* es satisfactible.

Por ejemplo, si $\varphi = (P \vee Q \vee \neg S) \wedge (P \vee \neg P \vee R) \wedge \neg Q$, al aplicar la regla de la tautología a φ , obtenemos la fórmula $\varphi^* = (P \vee Q \vee \neg S) \wedge \neg Q$.

Regla de la cláusula elemental

La denotaremos por $(\text{CE})_\psi$ donde ψ es un literal. La entrada es una fórmula φ en FNC tal que ψ es una cláusula de φ. Y la salida es la fórmula φ^* que se calcula de la siguiente forma. En primer lugar, se construye la fórmula χ^* que resulta de eliminar las cláusulas de φ en las que aparece ψ. Si $\chi^* = \blacksquare$, entonces $\varphi^* = \blacksquare$. Si no, φ^* es la fórmula que resulta de eliminar $\sim \psi$ de las cláusulas de χ^*.

Se tiene que φ es satisfactible si y sólo si φ^* es satisfactible.

Por ejemplo, si $\varphi = P \wedge (P \vee \neg Q) \wedge (\neg P \vee \neg Q \vee R)$ y aplicamos la regla $(\text{CE})_P$ a φ, obtenemos la fórmula $\varphi^* = \neg Q \vee R$.

Regla del literal puro

La denotaremos por $(\text{PU})_\psi$ donde ψ es un literal. La entrada es una fórmula φ en FNC tal que ψ es miembro de alguna cláusula de φ, pero $\sim \psi$ no es miembro de ninguna cláusula de φ. La salida es la fórmula φ^* que resulta de eliminar en φ las cláusulas en las que ψ aparece.

Se tiene que φ es satisfactible si y sólo si φ^* es satisfactible.

Por ejemplo, si $\varphi = (\neg P \vee Q) \wedge (\neg P \vee R \vee S) \wedge (\neg Q \vee T) \wedge S$ y aplicamos la regla $(\text{PU})_{\neg P}$ a φ, obtenemos la fórmula $\varphi^* = (\neg Q \vee T) \wedge S$.

Ejemplo 1

Aplicamos el método de Davis-Putnam para determinar si la fórmula $\varphi = (P \vee Q) \wedge (P \vee \neg Q) \wedge (R \vee Q) \wedge (R \vee \neg Q)$ es satisfactible.

Aplicando, en primer lugar, $(\text{PU})_P$ a φ, obtenemos la fórmula $\varphi_1 = (R \vee Q) \wedge (R \vee \neg Q)$.

Aplicando ahora $(\text{PU})_R$ a φ_1, obtenemos $\blacksquare$.

Por tanto, φ es satisfactible.

Ejemplo 2

Aplicamos el método de Davis-Putnam para determinar si la fórmula
$\varphi = (\neg P \vee \neg Q \vee \neg R) \wedge (Q \vee \neg Q \vee R) \wedge (P \vee \neg Q \vee R) \wedge (\neg P \vee Q) \wedge P \wedge R$
es satisfactible.

Aplicando, en primer lugar, la regla de la tautología a φ obtenemos la fórmula
$\varphi_1 = (\neg P \vee \neg Q \vee \neg R) \wedge (P \vee \neg Q \vee R) \wedge (\neg P \vee Q) \wedge P \wedge R.$

Aplicando ahora $(\text{CE})_P$ a φ_1 obtenemos la fórmula
$\varphi_2 = (\neg Q \vee \neg R) \wedge Q \wedge R.$

Aplicando entonces $(\text{CE})_Q$ a φ_2 obtenemos la fórmula
$\varphi_3 = \neg R \wedge R.$

Por último, aplicando $(\text{CE})_R$ a φ_3 obtenemos $\square$.

Por tanto, φ es una contradicción.

Ejemplo 3

Aplicamos el étodo de Davis-Putnam para determinar si la fórmula $\varphi = (P \vee \neg Q) \wedge (\neg P \vee Q) \wedge (R \vee Q) \wedge (\neg R \vee \neg Q)$ es satisfactible.

Se observa que únicamente se puede aplicar la regla de resolución a φ. Se obtiene entonces la fórmula
$\varphi_1 = (\neg Q \vee Q) \wedge (R \vee Q) \wedge (\neg R \vee \neg Q)$.

Aplicando de nuevo la regla de resolución a φ_1, obtenemos la fórmula $\varphi_2 = (\neg Q \vee Q) \wedge (R \vee \neg R)$.

Aplicando ahora la regla de la tautología a φ_2, obtenemos la fórmula $\varphi_3 = R \vee \neg R$.

Y aplicando de nuevo la regla de la tautología a φ_3, obtenemos $\blacksquare$.

Por tanto, φ es satisfactible.

Bibliografía para el tema

- Lógica para informáticos (R. Farré)

  Capítulos 2 y 3.
- Lógica computacional (E. Paniagua)

  Capítulo 5.

LENGUAJES DE PREDICADOS

Marzo de 2023

Introducción

Empezaremos a estudiar los lenguajes de predicados, que son lenguajes lógicos que tienen una capacidad de expresión mayor que los lenguajes de proposiciones.

Hay lenguajes de programación que están basados en el formalismo de la lógica. Y el más importante de ellos, el lenguaje Prolog, está basado en la lógica de predicados. Para poder entender programas en Prolog y poder iniciarse en la programación en dicho lenguaje, es entonces necesario conocer previamente los lenguajes de predicados.

Concepto de predicado

Intuitivamente, un predicado es una expresión formal cuyo tipo de datos es booleano.
Las siguientes expresiones son ejemplos de predicados:

(1) $x + y = 2$.
(2) $x + y \leq z$.
(3) Existe un entero $i \in \{1, \ldots, k\}$ tal que $A[i] > 100$, donde A es un vector de longitud k que tenemos declarado en un programa.

Antes de definir el concepto de lenguaje de predicados, vamos a describir los tipos de símbolos que aparecen en dichos lenguajes.

Símbolos de los lenguajes de predicados

(1) Variables.

$$u, v, x, y, z, u_0, v_0, x_0, y_0, z_0, u_1, v_1, x_1, y_1, z_1, \ldots, u_n, v_n, x_n, y_n, z_n, \ldots$$

(2) Símbolos de constante.

$$a, b, c, d, e, a_0, b_0, c_0, d_0, e_0, a_1, b_1, c_1, d_1, e_1 \ldots, a_n, b_n, c_n, d_n, e_n \ldots$$

(3) Símbolos de operador (o de función).

$$f, g, h, f_0, g_0, h_0, f_1, g_1, h_1 \ldots, f_n, g_n, h_n, \ldots$$

(4) Símbolos de predicado (o de relación).

$$A, B, \ldots, Z, A_0, B_0 \ldots, Z_0, A_1, B_1, \ldots, Z_1, \ldots, A_n, B_n, \ldots, Z_n, \ldots.$$

Símbolos de los lenguajes de predicados

(5) Las conectivas lógicas.

$$\neg, \vee, \wedge, \rightarrow, \leftrightarrow .$$

(6) Los cuantificadores.

$$\exists$$
$$\forall$$

(7) Símbolos auxiliares.

$$($$
$$)$$
$$,$$

Símbolos de los lenguajes de predicados

Los símbolos de función y de predicado tienen asociado un número entero positivo que indica su número de argumentos (aridad). Si $f(R)$ es un símbolo de función (respectivamente de predicado) de n argumentos, escribiremos f^n(respectivamente R^n).

Llamaremos vocabulario a un conjunto finito de símbolos de constante, de operador y de predicado.

El concepto de lenguaje de predicados se define con respecto a un vocabulario. Previamente, necesitamos definir el concepto de término. Intuitivamente, los términos nos permitirán describir elementos del dominio que estemos considerando.

Definición de término

Si σ es un vocabulario, definimos el conjunto de los σ-términos como el conjunto de elementos generado por las siguientes reglas:

(T1) Toda variable es un término.

(T2) Todo símbolo de constante de σ es un término.

(T3) Si f es un símbolo de función de n argumentos de σ y $t_1, \ldots, t_n$ son términos, entonces $f(t_1, \ldots, t_n)$ es un término.

Por ejemplo, si $\sigma = \{c, f^2, R^2\}$, los elementos $x_3, c, f(c, x_3), f(x_2, f(c, x_3))$ son términos. Sin embargo, $f(c)$ no es término, ya que f es un símbolo de función de dos argumentos. Y la expresión Rcv_2 tampoco es un término, ya que por las reglas (T1), (T2), (T3) de construcción de los términos, no es posible que un símbolo de predicado aparezca en un término.

Definición de átomo

Si σ es un vocabulario, llamaremos átomo (o fórmula atómica) a una expresión de la forma $Rt_1 \ldots t_n$ donde R es un símbolo de relación de n argumentos de σ y $t_1, \ldots, t_n$ son términos.

Por ejemplo, si $\sigma = \{c, f^2, R^2\}$, los elementos $Rcx_1, Rx_5f(c, x_3), Rf(c, v_3)c$ son átomos.

Definición de lenguaje de predicados

Si σ es un vocabulario, definimos el conjunto de las σ-fórmulas como el conjunto de elementos generado por las siguientes reglas:

(F1) Todo átomo es una fórmula.

(F2) Si ϕ es una fórmula, $\neg\phi$ también lo es.

(F3) Si ϕ, ψ son fórmulas, entonces $(\phi \vee \psi)$, $(\phi \wedge \psi)$, $(\phi \rightarrow \psi)$ y $(\phi \leftrightarrow \psi)$ son también fórmulas.

(F4) Si ϕ es una fórmula y x es una variable, entonces $\exists x\phi$, $\forall x\phi$ son fórmulas.

Al conjunto de fórmulas generadas por las reglas (F1)-(F4) se le llama lenguaje de predicados.

Reglas de formalización

Para poder representar propiedades formales o situaciones del lenguaje natural en lógica de predicados, deberemos utilizar las siguientes reglas:

(1) Todo A es B se representa por $\forall x(Ax \rightarrow Bx)$.

(2) Ningún A es B se representa por $\forall x(Ax \rightarrow \neg Bx)$.

(3) Algún A es B se representa por $\exists x(Ax \wedge Bx)$.

(4) Algún A no es B se representa por $\exists x(Ax \wedge \neg Bx)$.

Ejemplo

Consideremos la frase:
"A todos los esquiadores les gusta la nieve"

Átomos:

E$x = x$ es esquiador,

$Nx =$ a x le gusta la nieve.

La frase se puede formalizar entonces por $\forall x(Ex \rightarrow Nx)$.

Ejemplo

Consideremos la frase:
"A ningún montañero le gusta la lluvia."

Átomos:

$Mx = x$ es montañero,

$Lx =$ a x le gusta la lluvia.

La frase se puede formalizar entonces por $\forall x(Mx \rightarrow \neg Lx)$.

Ejemplo

Consideremos la frase:
"Todos los bancos tiene clientes descontentos."

Átomos:

$Bx = x$ es un banco,

$Cxy = x$ es cliente de y,

$Dx = x$ está descontento

La frase se puede formalizar entonces por
$\forall x(Bx \to \exists z(Czx \wedge Dz))$.

Introducción

En la última clase definimos el concepto de lenguaje de predicados. En la clase de hoy, estudiaremos algunos de los conceptos más fundamentales sobre estos lenguajes.

Empezamos mostrando cómo se pueden formalizar situaciones del lenguaje natural en los lenguajes de predicados.

Reglas de formalización

Para poder representar propiedades formales o situaciones del lenguaje natural en lógica de predicados, deberemos utilizar las siguientes reglas:

(1) Todo A es B se representa por $\forall x(Ax \to Bx)$.

(2) Ningún A es B se representa por $\forall x(Ax \to \neg Bx)$.

(3) Algún A es B se representa por $\exists x(Ax \wedge Bx)$.

(4) Algún A no es B se representa por $\exists x(Ax \wedge \neg Bx)$.

Ejemplo

Consideremos la frase:
"A todos los esquiadores les gusta la nieve"

Átomos:

E$x = x$ es esquiador,

$Nx =$ a x le gusta la nieve.

La frase se puede formalizar entonces por $\forall x(Ex \to Nx)$.

Ejemplo

Consideremos la frase:
"A ningún montañero le gusta la lluvia."

Átomos:

$Mx = x$ es montañero,

$Lx =$ a x le gusta la lluvia.

La frase se puede formalizar entonces por $\forall x(Mx \to \neg Lx)$.

Ejemplo

Consideremos la frase:
"Todos los bancos tiene clientes descontentos."

Átomos:

$Bx = x$ es un banco,

$Cxy = x$ es cliente de y,

$Dx = x$ está descontento

La frase se puede formalizar entonces por
$\forall x(Bx \to \exists z(Czx \wedge Dz))$.

Ejemplo

Consideremos la frase:
"Los amigos de Joan están alegres"

Átomos:

$Axy = x$ es amigo de y,

$Lx = x$ está alegre.

Además, hemos de representar a Joan por una constante.
Representamos entonces a Joan por c.

La frase se puede formalizar entonces por $\forall x(Axc \to Lx)$.

Variables libres y ligadas

Una aparición de una variable x en una fórmula φ es libre, si dicha aparición no está afectada por ningún cuantificador. En caso contrario, diremos que la aparición es ligada.

Una variable x es libre en una fórmula φ, si hay alguna aparición libre de x en φ. En caso contrario, diremos que la variable es ligada en φ.

Por ejemplo, en la fórmula $\forall x(Rxy \wedge \exists z(Pz \vee Rxz))$, las variables x, z son ligadas y la variable y es libre.

Fórmulas cerradas

Una fórmula ϕ de un lenguaje de predicados es cerrada, si ϕ no tiene variables libres.

Por ejemplo, la fórmula $\phi = \forall x(Rxy \wedge \exists z(Pz \vee Rxz))$ no es cerrada, ya que la variable y es libre en ϕ.

Sin embargo, la fórmula $\phi' = \forall x \forall y(Rxy \rightarrow \exists z(Rxz \wedge Rzy))$ es cerrada, ya que las tres variables x, y, z son ligadas en ϕ'.

Interpretaciones

Nuestro objetivo ahora es dar significado a los términos y a las fórmulas de los lenguajes de predicados. Para ello, definimos el concepto de interpretación en lógica de predicados.

Recordemos previamente que si D es un conjunto no vacío y $n \geq 1$, una función (o un operador) de n argumentos sobre D es una función de D^n en D, donde D^n denota el producto cartesiano de D consigo mismo n veces.

Y un predicado de n argumentos sobre D es un subconjunto de D^n.

Si R es una relación de n argumentos sobre D y $a_1, \ldots, a_n \in D$, escribiremos $Ra_1 \ldots a_n$ en lugar de $(a_1, \ldots, a_n) \in R$.

Interpretaciones

Si σ es un vocabulario, una σ-interpretación es una estructura que consta de lo siguiente:

- Un conjunto no vacío D al que llamaremos dominio de la interpretación.
- Una aplicación tal que:
  - A cada variable x le asigna un elemento $I(x)$ de D.
  - A cada símbolo de constante c de σ le asigna un elemento $I(c)$ de D.
  - A cada símbolo de función f de n argumentos de σ le asocia una función $I(f)$ de n argumentos sobre D.
  - A cada símbolo de predicado R de n argumentos de σ le asocia un predicado $I(R)$ de n argumentos sobre D.

Evaluación de términos

Para evaluar un término t en una interpretación I se sustituyen los símbolos de constante, los símbolos de función y las variables que aparezcan en t por sus correspondientes interpretaciones.

El resultado de la evaluación es un elemento del dominio de I, que será denotado por $I(t)$.

Ejemplo

Supongamos que $\sigma = \{a, b, f^2, g^2\}$ e I es una σ-interpretación cuyo dominio es el conjunto de los números enteros tal que $I(a) = 2$, $I(b) = 3$, $I(f)$ es la suma, $I(g)$ es la multiplicación e $I(v_i) = 3i$ para toda variable v_i. Tenemos entonces:

(a) $I(a) = 2$,

(b) $I(v_2) = 6$,

(c) $I(f(a, v_2)) = 2 + 6 = 8$,

(d) $I(g(b, v_3)) = 3 \times 9 = 27$,

(e) $I(f(a, g(b, v_3))) = 2 + 27 = 29$.

Evaluación de fórmulas

Para evaluar una fórmula φ en una interpretación I se sustituyen los símbolos de predicado, los símbolos de función y los símbolos de constante que aparezcan en φ por sus correspondientes interpretaciones. Una aparición libre de una variable x en φ se sustituye por su interpretación $I(x)$. Y las apariciones ligadas de las variables de φ se interpretan mediante los cuantificadores que las afectan, tomando como dominio de los cuantificadores el dominio de la interpretación I.

Si la evaluación de φ es una propiedad cierta, escribiremos $I(\varphi) = V$; en caso contrario, escribiremos $I(\varphi) = F$.

Ejemplo

Consideremos el vocabulario $\sigma = \{c, f^2, g^2, P^2, Q^1\}$ y la σ-interpretación I definida de la siguiente forma:

- dominio de $I =$ los enteros,
- $I(c) = 3$,
- $I(f) = +$,
- $I(g) = \times$,
- $I(P) = \{(m, n) : m, n$ son enteros tales que $m \leq n\}$,
- $I(Q) =$ conjunto de los números primos,
- $I(v_i) = 2i$ para cada variable v_i.

Evaluamos entonces las siguientes fórmulas en I.
$\varphi_1 = Qv_3$,
$\varphi_2 = Pv_2g(c, v_2)$,
$\varphi_3 = \exists v_0 Pf(v_1, v_2)v_0$,
$\varphi_4 = \forall v_0 \exists v_1 Pv_0v_1$.
$\varphi_5 = \forall v_0 \forall v_1 \exists v_2 (Pv_0v_2 \wedge Pv_2v_0)$.

Ejemplo

Para evaluar las fórmulas anteriores, utilizamos el algoritmo de evaluación de fórmulas visto anteriormente.

Como en φ_1 no hay cuantificadores, sustituimos los símbolos de φ_1 por sus interpretaciones. Por tanto, el significado de φ_1 en I es la condición "6 es un número primo". Por tanto, $I(\varphi_1) = F$.

Como en φ_2 no hay cuantificadores, sustituimos los símbolos de φ_2 por sus interpretaciones. Por tanto, el significado de φ_2 en I es la condición "$4 \leq 3 \times 4$". Por tanto, $I(\varphi_2) = V$.

Ejemplo

En φ_3, las variables v_1, v_2 aparecen libres y la variable v_o aparece ligada. Por tanto, v_1, v_2 se han de sustituir por $I(v_1), I(v_2)$ respectivamente, y la variable v_0 se ha de interpretar mediante el cuantificador existencial que la afecta. Así pues, el significado de φ_3 en I es la condición " existe un entero n_0 tal que "$2 + 4 \leq n_0$". Por tanto, $I(\varphi_3) = V$.

En φ_4, las variables v_0, v_1 aparecen ligadas. Por tanto, se han de interpretar mediante los cuantificadores que las afectan. Así pues, el significado de φ_4 en I es la condición " para todo entero n_0 existe un entero n_1 tal que $n_0 \leq n_1$". Por tanto, $I(\varphi_4) = V$.

Ejemplo

En φ_5, las variables v_0, v_1, v_2 aparecen ligadas. Por tanto, se han de interpretar mediante los cuantificadores que las afectan. Así pues, el significado de φ_5 en I es la condición " para todo entero n_0 para todo entero n_1 existe un entero n_2 tal que $n_0 \leq n_2$ y $n_2 \leq n_1$". Esta condición es falsa si $n_0 > n_1$ (por ejemplo, si $n_0 = 3$ y $n_1 = 2$). Por tanto, $I(\varphi_5) = F$.

Notación

Si σ es un vocabulario e I es una σ-interpretación, escribiremos:

(a) $\overline{x} = I(x)$ para toda variable x.

(b) $\overline{s} = I(s)$ para todo símbolo $s \in \sigma$.

Utilizaremos esta notación en el siguiente ejemplo.

Ejemplo

Consideremos el vocabulario $\sigma = \{a, f^1, P^1, Q^2\}$ y la σ-interpretación I definida de la siguiente forma:

(1) dominio de $I = \{0, 1\}$,

(2) $\overline{a} = 0$,

(3) $\overline{f}(0) = 1$, $\overline{f}(1) = 0$,

(4) $\overline{P}0 = F$, $\overline{P}1 = V$,

(5) $\overline{Q}00 = V$, $\overline{Q}01 = V$, $\overline{Q}10 = F$, $\overline{Q}11 = V$.

Evaluamos entonces las siguientes fórmulas en I.

$\varphi_1 = \exists x(Px \wedge Qxa)$,

$\varphi_2 = \exists x(Pf(x) \wedge Qxf(a))$,

$\varphi_3 = \forall x \exists y(Px \wedge Qxy)$,

$\varphi_4 = \forall x \forall y(Px \rightarrow Qxy)$.

Ejemplo

La fórmula φ_1 se interpreta mediante la expresión " existe $n \in \{0,1\}$ tal que $\overline{P}n = V$ y $\overline{Q}n0 = V$". Se tiene que la expresión es falsa, ya que si $n = 0$ se tiene que $\overline{P}0 = F$, y si $n = 1$ se tiene que $\overline{Q}10 = F$. Por tanto, $I(\varphi_1) = F$.

La fórmula φ_2 se interpreta mediante la expresión " existe $n \in \{0,1\}$ tal que $\overline{Pf}(n) = V$ y $\overline{Q}n1 = V$". Se tiene que la expresión es verdadera, ya que para $n = 0$ tenemos que $\overline{Pf}(0) = \overline{P}1 = V$, y asimismo tenemos que $\overline{Q}0\overline{f}(0) = \overline{Q}01 = V$. Por tanto, $I(\varphi_2) = V$.

Ejemplo

La fórmula φ_3 se interpreta mediante la expresión "para todo $n \in \{0,1\}$ existe $m \in \{0,1\}$ tal que $\overline{P}n = V$ y $\overline{Q}nm = V$". Se tiene que la expresión falla para $n = 0$, ya que tenemos que $\overline{P}0 = F$. Por tanto, $I(\varphi_3) = F$.

La fórmula φ_4 se interpreta mediante la expresión "para todo $n, m \in \{0,1\}$, $\overline{P}n \to \overline{Q}nm = V$". Se tiene que la expresión falla para $n = 1$ y $m = 0$, ya que $\overline{P}1 \to \overline{Q}10 = V \to F = F$. Por tanto, $I(\varphi_4) = F$.

Fórmulas equivalentes

Decimos que dos fórmulas φ, ψ son lógicamente equivalentes, si para toda interpretación I, tenemos que $I(\varphi) = I(\psi)$.

Si φ, ψ son lógicamente equivalentes, escribiremos $\varphi \equiv \psi$.

Añadimos a las equivalencias lógicas vistas en lógica de proposiciones las siguientes equivalencias:

Fórmulas equivalentes

(1) $\neg\forall x\varphi \equiv \exists x\neg\varphi$.

(2) $\neg\exists x\varphi \equiv \forall x\neg\varphi$.

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

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

(5) $\forall x(\varphi \wedge \psi) \equiv \forall x\varphi \wedge \forall x\psi$.

(6) $\exists x(\varphi \vee \psi) \equiv \exists x\varphi \vee \exists x\psi$.

Ejemplo 1

Consideremos las fórmulas $\varphi_1 = \neg\exists x\forall y(Px \wedge \neg Rxy)$ y $\varphi_2 = \forall x\exists y(Px \to Rxy)$. Demostramos que estas dos fórmulas son lógicamente equivalentes. Tenemos:

$\varphi_1 = \neg\exists x\forall y(Px \wedge \neg Rxy) \equiv \forall x\neg\forall y(Px \wedge \neg Rxy) \equiv \forall x\exists y\neg(Px \wedge \neg Rxy) \equiv \forall x\exists y(\neg Px \vee Rxy) \equiv \forall x\exists y(Px \to Rxy) = \varphi_2$.

Ejemplo 2

Consideremos las fórmulas $\varphi_1 = Pc$ y $\varphi_2 = \exists xPx$. En este caso, las fórmulas no son equivalentes. Para ello, damos una interpretación que separa las fórmulas. Defininimos entonces la interpretación I de la siguiente manera. El dominio de I es $\{0, 1\}$, definimos $I(P) = \{1\}$ y definimos $I(c) = 0$. Se tiene entonces que $I(\varphi_1) = F$, pero $I(\varphi_2) = V$.

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 $\varphi = \neg\exists xPx \vee \forall xQx$. Tenemos entonces: $\varphi = \neg\exists xPx \vee \forall xQx \equiv \forall x\neg Px \vee \forall xQx \equiv \forall x\neg Px \vee \forall yQy \equiv \forall x\forall y(\neg Px \vee Qy)$.

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