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 $(P \vee \neg Q) \rightarrow R$.

Tenemos entonces

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

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

Ejemplos

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

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

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

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 $\{P \vee Q, \neg P \vee Q, P \vee \neg Q, \neg P \vee \neg Q\}$.

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)

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)

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.