Lógica de Predicados

Introducción

El Cálculo Proposicional, resulta útil para muchas aplicaciones computacionales, entre las que podemos citar, análisis de circuitos, análisis y confiabilidad de sistemas mediante árboles lógicos, diversas aplicaciones de satisfactibilidad a problemas de planeación, etc.

Sin embargo, la principal debilidad de la lógica proposicional es su limitada habilidad de expresar conocimiento. Existen varias sentencias complejas que pierden mucho de su significado cuando se las representa mediante lógica proposicional.

Por esto se desarrolló una forma lógica más general capaz de representar todos los detalles expresados en las sentencias. Esta es la *lógica de predicados*.

Para explicar esta situación, consideremos el argumento conformado con las siguientes sentencias del Cálculo Proposicional:

```
P = Todos los mamíferos son mortales
Q = Lassie es un mamífero
_____________________________________
R = Lassie es mortal
```

Las expresiones P, Q, R, son proposiciones, puesto que todas ellas son enunciados que pueden ser evaluados como V (verdaderos) o F (falsos). Hay que hacer notar, sin embargo, que mediante la aplicación de reglas de inferencia del Cálculo Proposicional, no es posible deducir R a partir de las premisas P y Q anteriores, ya que el Cálculo Proposicional no tiene acceso a los elementos comunes que conforman estas proposiciones, como son mamífero, mortal y Lassie, e indispensables para llegar a la conclusión R resultante. Sin embargo, esta misma expresión en Cálculo de Predicados, se podría escribir distinguiendo los elementos constitutivos de cada proposición. Es decir, el Cálculo de Predicados se aplica para las mismas proposiciones que pueden ser enunciadas en Cálculo Proposicional, con la diferencia que en el primero se tiene acceso a los elementos constitutivos de cada proposición.

La lógica de predicados está basada en la idea de las sentencias realmente expresan **relaciones entre objetos**, así como también **cualidades y atributos de tales objetos**. Los objetos pueden ser personas, objetos físicos, o conceptos. Tales *cualidades*, *relaciones* o *atributos*, se denominan ***predicados***. Los *objetos* se conocen como ***argumentos*** o ***términos*** del predicado.

Al igual que las proposiciones, los predicados tienen un valor de veracidad, pero a diferencia de las preposiciones, su valor de veracidad, depende de sus términos. Es decir, un predicado puede ser verdadero para un conjunto de términos, pero falso para otro.

Por ejemplo, el siguiente predicado es verdadero:

color (yerba, verde)

el mismo predicado, pero con diferentes argumentos, puede no ser verdadero:

color (yerba, azul) ***o*** ***color (cielo, verde)***

Los predicados también pueden ser utilizados para asignar una cualidad abstracta a sus términos, o para representar acciones o relaciones de acción entre dos objetos. Por ejemplo:

- mortal (juan_carlos)
- clima (martes, lluvioso)
- ave (gaviota)
- ama (roberto, vanesa)
- lee (alex, novela)
- mordio (boby, cartero)

Al construir los predicados se asume que su veracidad está basada en su relación con el mundo real. Naturalmente, siendo prácticos, trataremos que los predicados que definimos estén de acuerdo con el mundo que conocemos, pero no es absolutamente necesario que así lo hagamos. En lógica de predicados el establecer como verdadero un predicado es suficiente para que así sea considerado. Demos el siguiente ejemplo, que indica que Ecuador está en Europa:

parte_de (ecuador, europa)

Obviamente, esto no es verdadero en el mundo real, pero la lógica de predicados no tiene razón de saber geografía y si el predicado es dado como verdadero, entonces es considerado como lógicamente verdadero. Tales predicados, establecidos y asumidos como lógicamente verdaderos se denominan ***axiomas***, y no requieren de justificación para establecer su verdad.

La lógica de predicados, se ocupa únicamente de métodos de argumentación sólidos. Tales argumentaciones se denominan ***Reglas de Inferencia***. Si se da un conjunto de axiomas que son aceptados como verdaderos, las reglas de inferencia garantizan que sólo serán derivadas consecuencias verdaderas.

Tanto los conectivos lógicos, como los operadores dados anteriormente para la lógica proposicional, son igualmente válidos en lógica de predicados. De hecho, la lógica proposicional es un subconjunto de la lógica de predicados.

Cada uno de los argumentos en los ejemplos de predicados dados anteriormente, representan a un objeto específico. Tales argumentos se denominan *constantes*. Sin embargo, en la lógica de predicados se pueden tener argumentos que en determinado momento pueden ser desconocidos. Estos son los argumentos tipo *variable*.

En el ejemplo: *color (yerba, X),* la variable *X*, puede tomar el valor de *verde*, haciendo que el predicado sea verdadero; o puede tomar el valor de *azul*, dando lugar a que el predicado sea falso.

Las variables, también pueden ser cuantificadas. Los cuantificadores que típicamente se utilizan en lógica de predicados son:

- El cuantificador universal; $\forall$ indica que la fórmula bien formada, dentro de su alcance, es verdadera para todos los valores posibles de la variable que es cuantificada. Por ejemplo:

$$\forall X \rightarrow \textit{Establece que "para todo X, es verdad que . . . "}$$

- El cuantificador existencial; $\exists$, indica que la fórmula bien formada, dentro de su alcance, es verdadera para algún valor o valores dentro del dominio. Por ejemplo:

$$\exists X \rightarrow \textit{Establece que "existe un X, tal que . . . "}$$

A continuación se dan algunos ejemplos de predicados cuantificados:

$\forall$ X, [niño (X) $\Rightarrow$ le_gusta (X, helados)].

$\forall$ Y, [mamífero (Y) $\Rightarrow$ nace (Y, vivo)].

$\exists$ Z, [cartero(Z) $\wedge$ mordió (boby, Z)].

Desde el punto vista de representación, los cuantificadores son difíciles de usar. Por lo que es deseable reemplazarlos con alguna representación equivalente, más fácil de manipular. El caso del cuantificador universal es más simple ya que se asume a todas las variables como universalmente cuantificadas.

Sintaxis de la lógica de predicados

En el Cálculo de Predicados se usan varios tipos de símbolos:

Un conjunto de elementos llamado átomos.
Caracteres en minúsculas y números: raquel, a, b, 3200.
Un vocabulario V de variables
Se utilizarán letras MAYÚSCULAS: X, Y, Z.
Un vocabulario F de símbolos funcionales
Se utilizarán letras MAYÚSCULAS: EMPLEADO (jaime, 32)
Un vocabulario P de símbolos predicativos.
Se utilizarán letras minúsculas: cartero (juan)

Cada símbolo funcional y predicativo tiene asociado un número entero que es su aridad.

Definición

Un término es:
Un átomo
Una variable
Un símbolo funcional seguido de una sucesión de términos (aridad).

Un Predicado es:
Un símbolo predicativo seguido de una sucesión de términos (aridad).

Una regla o cláusula es alguna de las siguientes alternativas:
A
$\neg B_1 \vee \ldots \vee \neg B_n \vee A$
$\neg B_1 \vee \ldots \vee \neg B_n$

Donde A, B1, …, Bn son predicados.

Un programa lógico es un conjunto de reglas.

Cláusula de Skolem

La sintaxis presentada, proviene de la lógica de primer orden. En el lenguaje de dicha lógica, se utilizan los vocabularios mencionados, conectores lógicos y cuantificadores (universal y existencial) como se observó previamente.

Con miras de unificar el tratamiento de las fórmulas bien formadas de la lógica de primer orden, se han definido diversas formas normales, entre las que resulta de particular interés la llamada forma clausal de Skolem, cuyo formato es el siguiente:

$\forall$ X1, …, Xj $[A_1 \vee \ldots \vee A_n \vee \neg B_1 \vee \ldots \vee \neg B_n]$.

o, igualmente

$\forall$ X1, …, Xj $[A_1 \vee \ldots \vee A_n \Leftarrow B_1 \wedge \ldots \wedge B_n]$.

Dónde $A_1 \vee \ldots \vee A_n \vee \neg B_1 \vee \ldots \vee \neg B_n$ son predicados en el sentido de la definición vista.
En este formato no aparecen los cuantificadores existenciales y los predicados están relacionados únicamente por el conector $\vee$.

Se puede demostrar que para toda fórmula bien formada del cálculo de predicados, existe una fórmula clausal de Skolem tal que ambas tienen las mismas propiedades de satisfactibilidad semántica.

Cláusula de Horn

Una fórmula bien formado del cálculo de predicados de primer orden está en forma clausal de Horn sii:

i) está en forma clausal de Skolem
ii) tiene a lo sumo un predicado "positivo".

Por lo tanto son posibles

$\forall$ X1, …, Xj [A].	Tipo I
$\forall$ X1, …, Xj $[A \Leftarrow B_1 \wedge \ldots \wedge B_n]$.	Tipo II
$\forall$ X1, …, Xj $[\neg B_1 \vee \ldots \vee \neg B_m]$.	Tipo III

Cuyos cuantificadores universales pueden ser suprimidos interpretando toda cláusula como clausurada universalmente.

Surgen así las tres posibles cláusulas para la programación lógica. Las de tipo I corresponden a la afirmación del predicado A, las de tipo II representan los hechos condicionales:

si B_1 … B_n son predicados ciertos, entonces A también es cierto. Finalmente las cláusulas de tipo III corresponden a las invocaciones de ejecución del programa, o si se prefiere a las interrogaciones de las relaciones definidas: existen valores de las variables X_1, …, X_j tales que con dichos valores los predicados B_1 y … y B_m son ciertos?.

Unificación

El proceso de unificación soluciona el problema de cómo resolver dos predicados que tengan el mismo símbolo predicativo pero sus argumentos no coinciden.

Sustitución

Una sustitución es un conjunto de asignaciones del tipo X:= t

dónde X es una variable y t es un término. En la sustitución no pueden existir más de una sustitución a la misma variable.

Es necesario aclarar que una sustitución tiene alcance clausular.

Ejemplos:

{ X:= juan, Y:= noel}

{ W:= Z, R := EMPLEADO (T, 3200)}

{ Q:= [], R:= [X, Z]}

Aplicación de una sustitución

Data una sustitución θ y un predicado P, la aplicación de θ a P, produce un nuevo predicado que se denota Pθ, y que corresponde al predicado inicial P donde toda variable asignada en θ se cambia por el término correspondiente, y las otras variables permanecen incambiadas.

Unificador

Dadas dos expresiones del lenguaje definido (por ejemplo dos predicados) E_1 y E_2. Se llama unificador, a una sustitución θ tal que cumple que:

$E_1\theta = E_2\theta$

Es decir que la aplicación de la sustitución a ambas expresiones da la misma expresión.

Ejemplos:

i) Dadas los predicados padre (Z, diego) y padre (jorge, diego).
La sustitución θ = {Z := jorge} es un unificador de las mismas.

ii) Dadas los predicados tio (X, diego) y tio (W, diego).
La sustitución θ = {X := W} es un unificador de las mismas.
La sustitución θ = {W := X} es un unificador de las mismas.
La sustitución θ = {X := guille, W:= guille} es un unificador de las mismas.

iii) Dadas los predicados r (Z, diego) y r (diego, jorge).
No existe unificador para ambos.

Unificador más general

De la definición y de los ejemplos presentados anteriormente, surge que existen expresiones para las cuales no existe un unificador y en otros casos, es posible encontrar más de un unificador.

Para el procedimiento de demostración, y en el caso de existir más de un unificador entre dos predicados que permita aplicar la regla de resolución, va a interesar aquel que sea "más general", en el sentido que necesita asignaciones menos específicas de términos a variables. En el ejemplo 2, los dos primeros unificadores son más generales (a menos de un renombramiento) que el tercero.

Definimos ahora que es un MGU (unificador más general).

Dadas dos sustituciones θ1 y θ2 y ambas son unificadores de las expresiones E1 y E2, se dice que θ1 es más general que θ2 si existe una sustitución θ3 tal que:

$E1\ \theta 1\ \theta 3 = E2\ \theta 2$

Regla de Resolución

Supongamos que $t_1, \ldots, t_n$ y $s_1, \ldots, s_n$ son términos tales que ti y si son unificables con un MGU θ (Most General Unifier) para $1 \leq i \leq n$ y C1 y C2 con cláusulas.

La resolución dice que:

$C_1 \vee R(t_1, \ldots, t_n)$ y $C_2 \vee \neg R(s_1, \ldots, s_n)$ podemos derivar $C_1\theta \vee C_2\theta$

dónde R es un símbolo predicativo.

Podemos usar los mismos métodos que el la lógica proposicional para mostrar que para cláusulas de Horn:

$$\mathbf{(Q_1 \wedge \ldots \wedge Q_n) \Rightarrow R(t_1, \ldots, t_n),\quad (R(s_1, \ldots, s_n) \wedge \ldots \wedge P_1 \wedge P_m) \Rightarrow S}$$

$$\vdash_{\mathbf{Res}} \mathbf{(Q_1\theta \wedge \ldots \wedge Q_n\theta \wedge P_1\theta \wedge \ldots \wedge P_m\theta) \Rightarrow S\theta}$$

dónde θ es un MGU para t_i y s_i $(1 \leq i \leq n)$

Notación Prolog

:- significa "implicado por"

, significa "y"

mayúsculas para las variables y minúsculas para los predicados, por lo tanto

$(Q_1 \wedge \ldots \wedge Q_n) \Rightarrow R$ se escribe r :- $q_1, \ldots, q_n$

<u>Ejemplo 1</u>: Dado el siguiente programa (utilizaremos la notación Prolog)

a) s(X) :- q(Y), r(X, Y)
b) q(X) :- p(X)
c) p(b)
d) r(a, b)

dada la consulta s(a). La resolución formal es la siguiente:

1) $s(X) \vee \neg q(Y) \vee \neg r(X, Y)$	Hipótesis
2) $q(X) \vee \neg p(X)$	
3) $p(b)$	
4) $r(a, b)$	
5) $\neg s(a)$	Hipótesis agregada
6) $\neg q(Y) \vee \neg r(a, Y)$	Res por 1 y 5, {X:=a en 1}
7) $\neg p(Y) \vee \neg r(a, Y)$	Res por 2 y 6, {X:=Y en 6}
8) $\neg r(a, b)$	Res por 3 y 7, {Y:= b en 7}
9) □	Res por 4 y 8.

Se determina entonces que s(a) es derivable del programa.

<u>Ejemplo 2</u>: Dado el siguiente programa:

padre (jorge, diego)	**i**
hermano (ricardo, jorge)	**ii**
tio (X, Y) :- padre (Z, Y), hermano (X, Z)	**iii**

y la consulta: **tio (W, diego)** **iv**

La cláusula 3 está cuantificada existencialmente por lo tanto es válida para el caso que la variable Y tenga como valor “diego”

tio (X, diego) :- padre (Z, diego), hermano (X, Z) **v**

Como las variables tienen un alcance de cláusula donde aparecen, se puede reemplazar W por X en la consulta. Por lo tanto se puede resolver la consulta con la cláusula anterior.

padre (Z, diego), hermano (X, Z) **vi**

Al resolver i con vi, por lo tanto se reemplaza Z por jorge.

padre (jorge, diego), hermano (X, jorge) **vii**

Resolviendo i con vii quedaría

hermano (X, jorge) **viii**

Finalmente instanciando X con “ricardo”, se obtiene la cláusula vacía.

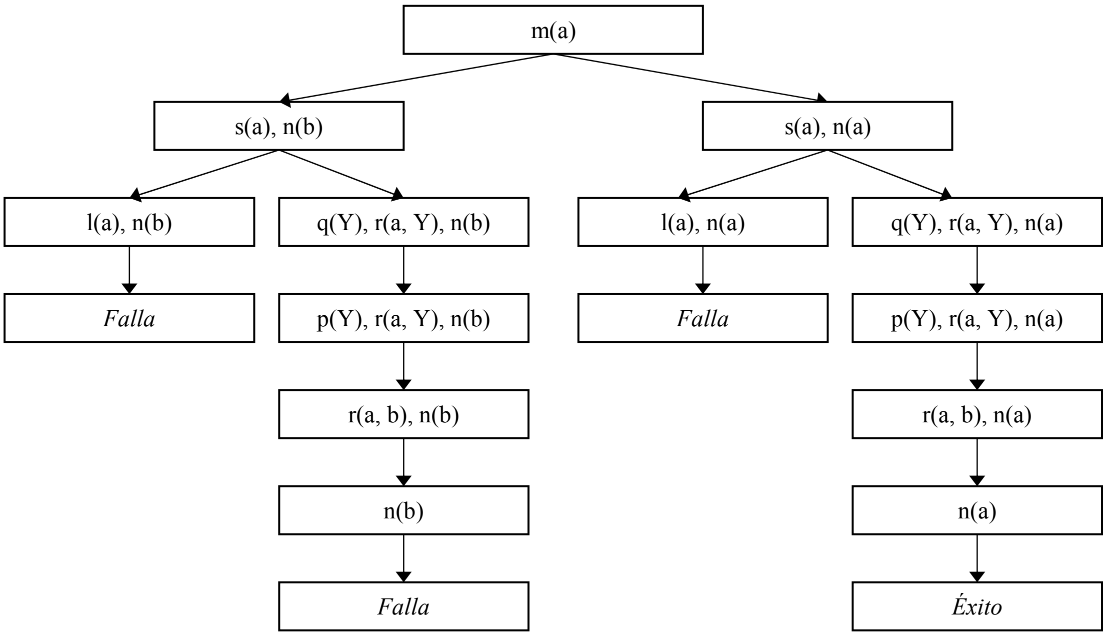
□

Los pasos del ejemplo, constituyen la demostración de la formula **iv**. Resulta claro que al escribir dicha consulta, el usuario no estaba directamente interesado en una prueba, sino más bien en conocer quien es el **tio** de "diego". La variable W original fue cambiada por X en el paso **i**, y esa variable X fue sustituida o instanciada en el paso **v** para completar la prueba. El valor que le fue asignado es "ricardo" y es la respuesta que da el sistema y que corresponde al interés inicial del usuario.

Ejemplo 3: Dado el siguiente programa

m(X) :- s(X), n(b)
m(X) :- s(X), n(X)
s(X) :- l(X)
s(X) :- q(Y), r(X, Y)
q(X) :- p(X)
p(b)
r(a, b)
l(c)
n(a)

Dada la consulta m(a) podemos construir el siguiente árbol de resolución:



Ejemplo 4: Familia Adams.

La Familia Adams esta compuesta por homero, morticia, pericles, merlina, tio_lucas, tio_cosa y la_abuela. Homero es hermano de tio_lucas y de tio_cosa; todos ellos son hijos de la_abuela. Morticia es esposa de homero y con ella ha tenido dos hijos: pericles y merlina.

```
predicates
    varon(nombre)
    mujer(nombre)
    progenitor(nombre,nombre)
    padre(nombre,nombre)
    madre(nombre,nombre)
    hijo(nombre,nombre)
    hija(nombre,nombre)
    hermana(nombre,nombre)
    tio(nombre,nombre)
    abuela(nombre,nombre)

clauses
    varon(homero).
    varon(tio_lucas).
    varon(tio_cosa).
    varon(pericles).

    mujer(morticia).
    mujer(merlina).
    mujer(la_abuela).

    progenitor(la_abuela,homero).
    progenitor(la_abuela,tio_lucas).
    progenitor(la_abuela,tio_cosa).
    progenitor(homero,pericles).
    progenitor(homero,merlina).
    progenitor(morticia,pericles).
    progenitor(morticia,merlina).

    padre(X,Y):-progenitor(X,Y),varon(X).
    madre(X,Y):-progenitor(X,Y),mujer(X).

    hijo(X,Y):-progenitor(Y,X),varon(X).
    hija(X,Y):-progenitor(Y,X),mujer(X).

    hermano(X,Y):-progenitor(Z,X),progenitor(Z,Y),X<>Y,varon(X).

    hermana(X,Y):-progenitor(Z,X),progenitor(Z,Y),X<>Y,mujer(X).

    tio(X,Y):-hermano(X,Z),progenitor(Z,Y).

    abuela(X,Y):-madre(X,Z),progenitor(Z,Y).
```

Consultas:

```
hijo(pericles,homero)
hija(Quien,homero)
padre(X,pericles)
tio(X,Y)
abuela(Quien,pericles)
hermana(X,Y)
```

Respuestas:

```
Verdadero.

Quien=merlina.

X=homero.

X=tio_lucas, Y=pericles.
X=tio_lucas, Y=merlina.
X=tio_cosa, Y=pericles.
X=tio_cosa, Y=merlina.

Quien=la_abuela.

X=merlina, Y=pericles.
X=merlina, Y=pericles.
```