Clase 2.1: Aritmética en Prolog. Operadores.

Profesora: Dra. Yisel Garí

Introducción

Prolog es un lenguaje relacional, es decir, es un lenguaje basado en el concepto de relación o predicado de la LPO. Hemos visto también que los argumentos de los predicados pueden ser ocupados por términos descriptivos, es decir, son términos construidos a partir de símbolos de funciones que son constructores n-arios que sirven para definir estructuras de datos. Luego los argumentos de un predicado no son términos que en general se evalúen como si fuesen llamados a funciones.

Aunque Prolog es un lenguaje cuyo objetivo es la programación de soluciones de problemas en los que los procesos lógicos son esenciales, el lenguaje necesita, como todo lenguaje de programación, una aritmética y hay una aritmética de base disponible en cualquier realización de un intérprete.

En algunos ejemplos de programas se han empleado números en Prolog, y ello quiere decir que el intérprete de Prolog aprovecha la representación de números y las operaciones aritméticas básicas del hardware sobre el cual se instala para desarrollar una aritmética y definir a partir de ella funciones matemáticas que puedan ser llamadas desde un programa Prolog.

Operadores aritméticos

Un operador n-ario se define como el nombre de un predicado n-ario o de un símbolo de función o constructor n-ario. Ejemplos de operadores son las operaciones aritméticas. En Prolog el repertorio de dichas operaciones es limitado dado que Prolog es un lenguaje orientado a la manipulación simbólica. La siguiente tabla muestra las operaciones básicas disponibles en la mayoría de las implementaciones de Prolog y los símbolos de función asociados.

Operación	Símbolo de función
Suma	+
Resta	-
Multiplicación	*
División entera	div
Módulo (resto de la división entera)	mod
División	/

Cuadro 1: Table 1. Operaciones aritméticas básicas.

Con estos operadores se pueden construir expresiones aritméticas que no son otra cosa que términos.
Ejemplos:
3 + 2 * 5
3 / 2 + 2 * 5
6 div 2
3 mod 2
Y + 10 / 2

Estas expresiones no se evalúan automáticamente y es preciso forzar su evaluación mediante el operador is/2. Dado un objetivo ":-Expresion1 is Expresion2 ", tanto la parte izquierda como la derecha son expresiones aritméticas, que de contener variables deberán ser instanciadas en el momento de la ejecución del objetivo, cuyo efecto es forzar la evaluación de dichas expresiones y después intentar unificarlas. Si Expresión1 es una variable, quedará instanciada al valor que tome Expresion2. La evaluación de cada operación aritmética se realiza mediante procedimientos especiales, predefinidos (built-in) en el intérprete, que acceden directamente a las facilidades de cómputo de la máquina.

Ejemplos: evaluación de las expresiones aritméticas anteriores.

```
:- X is 3 + 2 * 5
X = 13

:- X is 3 / 2 + 2 * 5
X = 11.5

:- X is 6 div 2
X = 3

:- X is 3 mod 2
X = 1

:- X is Y + 10 / 2
ERROR: is/2: Arguments are not sufficiently instantiated
```

En el último caso el mensaje de error se produce porque la variable Y no está instanciada en el momento en que se evalúa la expresión. El operador is/2 no puede utilizarse para instanciar una variable dentro de una operación, por ejemplo: " :- 40 is X + 10/2. ". Tampoco se debe utilizar para actualizar el valor de una variable, por ejemplo: " :- X is X + 1. ", vea que si X está instanciada a un valor dado lo que ocurre es un fracaso en la unificación del resultado del miembro derecho al ser comparado con el valor de X en el miembro izquierdo.

Operadores para comparar expresiones aritméticas Las expresiones aritméticas pueden compararse utilizando los operadores de comparación que se muestran en la Tabla 2.

Operación	Símbolo de función
Mayor que	>
Menor que	<
Mayor o igual que	>=
Menor o igual que	<=
Igual que	=:=
Distinto que	=\=

Cuadro 2: Table 2. Operadores de comparación de operaciones aritméticas.

Los operadores de comparación aritmética fuerzan la evaluación de las expresiones aritméticas y no producen la instanciación de las variables. En el momento de la evaluación todas las variables deben estar instanciadas.

Caracterización de los operadores
Un lenguaje declarativo como Prolog que tiene entre sus fines la representación y el procesamiento de información textual debe incluir la posibilidad de introducir y definir operadores con la notación requerida para representar texto con determinados fines de lectura y procesamiento.

Las expresiones sintácticas válidas en Prolog se construyen en general con una notación prefija de los operadores, como, por ejemplo:

- padre(luis, X).
- $[2, 3]$ que es "azúcar sintáctico" para la representación prefija interna al intérprete de Prolog de la lista .(2, .(3, nil))).

Es decir el operador precede a sus argumentos, los cuales se escriben a continuación de éste, encerrados entre paréntesis y separados por comas. Esta es la forma estandarizada en que el intérprete recibe términos y cláusulas para su interpretación. Sin embargo, como se vio en los apartados anteriores, Prolog cuenta con operadores con una sintaxis que difiere de la estándar: una notación infija de operadores, con la cual se incrementa la facilidad de lectura de un programa. Si esto no fuera así sería muy complicado escribir las operaciones aritméticas, por ejemplo: 2*3+4*5 tendría que escribirse como +(*(2,3),*(4,5)).

Por otra parte véase que las reglas (cláusulas con cabeza y cuerpo) de un programa, para facilitar su lectura como condicionales, tienen una representación infija de su operador principal (:-).
Ejemplo: `member(X [_|Y]) :- member(X,Y).`

Que para el intérprete tiene la representación prefija:
`:- (member(X [_|Y]), member(X,Y)).`

Además si se realiza la lectura declarativa de un programa existen hechos y reglas que tratadas en notación infija son más fáciles de comprender.
Ejemplo:

```
a es_padre_de b.
X es_abuelo_de Y :- X es_padre_de Z , (Z es_padre_de Y ; Z es_madre_de Y).
```

Por tanto puede resultar conveniente introducir operadores con una notación infija o incluso sufija. El lenguaje Prolog proporciona facilidades para realizar esta tarea, es decir, definir o redefinir operadores de forma diferente a la estándar. Para ello analizaremos las características que rigen a un operador.

Cada operador en Prolog está caracterizado, además de por su nombre/aridad , por su precedencia (o prioridad), por su posición (prefija, infija, sufija), y por su asociatividad con respecto a otros operadores.

Nombre El nombre es un átomo o una combinación de caracteres especiales $(+, -, *, /, <, >, =, :, \&, \_,$). Cuando se elija un nombre concreto para un operador habrá que tener en cuenta que ciertos nombres están reservados (e.g., " is ", " = ", " + ", " :- ", etc.).

Precedencia En un término pueden ocurrir diferentes operadores que dan lugar a diferentes subtérminos del término. Ejemplo: 3 + 2 * 5

Para una correcta interpretación del término es necesario establecer en qué orden deberán ser tomados estos operadores. El orden en que se toma un operador con respecto al resto de los operadores se denomina precedencia, la precedencia se denota en Prolog con un número entre 1 y 1200, que indica cómo debe interpretarse una expresión en la que no hay paréntesis. En una expresión sin parénesis, el operador con mayor número de precedencia se convierte en el operador principal de la expresión, el cual es visible en la notación prefija del término al ser el más externo. El empleo de paréntesis anula las reglas de precedencia. Al establecer la precedencia entre operadores se elimina cualquier ambigüedad en la lectura y por lo tanto se realiza la correcta interpretación de un término. Si analiza la Tabla 7.3 puede remarcar que no hay ambigüedad para interpretar 3 + 2 * 5 como 3 + (2 * 5) que es lo que normalmente se hace en matemática, siendo el operador principal la suma.

Precedencia	Operador
500	+,-
400	*,/

Cuadro 3: Table 3. Precedencia de los operadores predefinidos +, -, * y /.

Posición Como ya se ha indicado un operador puede tener una notación prefija que es la usual cuando definimos predicados en Prolog y la estándar para el intérprete. Pero existen operadores unarios y binarios primitivos en Prolog que han sido definidos con una notación infija, prefija o sufija y además esta facilidad es dada también al programador.

La posición de un operador es una característica que se define empleando especificaciones de modo. Los especificadores de modo son "_f_" para operadores infijos, "_f" para operadores sufijos (también llamados posfijos) y "f_" para operadores prefijos. El símbolo "_" puede sustituirse según sea el caso por x o y como se explica más adelante y f denota el operador al cual se le está definiendo la posición.

Por ejemplo, +, - que son operadores binarios infijos de suma y resta respectivamente, son también definidos como operadores unarios prefijos, para denotar números positivos y negativos respectivamente (ver Tabla 4): +2, -2.5.

Posición	Operador
yfx	+,-
fy	+,-

Cuadro 4: Table 4. Posiciones infijas y prefijas de los operadores + y -.

Asociatividad Además de su colocación prefija, infija o sufija con respecto a sus argumentos, es necesario especificar de un operador cuál es su asociatividad con respecto a los operadores principales de los términos que recibe como argumentos. La asociatividad se emplea para eliminar la ambigüedad de las expresiones en las que aparecen operadores con la misma precedencia. Por ejemplo, determinar cómo interpretar una expresión de la forma: " 16 / 4 / 2 ". Si se interpreta como "(16 / 4) / 2 " el resultado es 2, pero si se interpreta como " 16

/ (4 / 2) ” el resultado es 8. La asociatividad de un operador se expresa en Prolog al mismo tiempo que la propiedad de posición, empleando los especificadores de modo, que son átomos (constantes no numéricas) especiales:

Prefijo	Infijo	Sufijo
fx, fy	xfx, xfy, yfx	xf, yf

Cuadro 5: Table 5. Significado de los átomos especiales especificadores de modo.

Como ya se dijo, f denota un operador y x, y denotan sus argumentos. Una “x” significa que ese argumento debe tener un número de precedencia estrictamente menor que el operador que se está especificando. Por otra parte una “y” indica que el argumento correspondiente puede tener una precedencia igual o menor. Se define la precedencia de un argumento como sigue: si el argumento es un término su precedencia es cero, a menos que esté encabezado por un operador (al que se ha asignado una precedencia), en cuyo caso su precedencia es la de ese operador. Si el argumento está encerrado entre paréntesis su precedencia es cero.

Especificador de modo	Asociatividad
fy	A la derecha
yf	A la izquierda
xfy	A la derecha
yfx	A la izquierda

Cuadro 6: Table 6. Asociatividad de los operadores.

Note que el caso de fx, xfx y xf el operador no es asociativo, mientras que si lo es en los restantes casos. Note además que yfy no se define pues esto conlleva a ambigüedades respecto a cómo asociar.

Ejemplos:

1. Como “ / ” tiene posición y asociatividad “ yfx ”, “ 16/ 4/ 2 ” se interpreta como “ (16 / 4) / 2 ”, esto es, el operador “ / ” es asociativo a la izquierda.

2. El operador que define las reglas “ :- ” tiene el especificador de modo es “ xfx ” pues es un operador infijo que no es asociativo.

3. El operador aritmético de cambio de signo “ - ” y la negación “\+ ” suelen especificarse con asociatividad “ fy ” con lo que se hace posible la doble negación (de un número o de un objetivo), por ejemplo: X is - -5, X = 5 .

4. La “ , ” que denota la conjunción en el cuerpo de una cláusula, tiene asociatividad “ xfy ”, es decir es un operador infijo asociativo a la derecha.

5. El “ ; ” que denota la disyunción en el cuerpo de una cláusula, tiene asociativita “ xfy ”, es decir es un operador infijo asociativo a la derecha.

Con respecto a estos dos últimos operadores considere la siguiente cláusula:
abuelo(X,Y) :- padre(X,Z), madre(Z, Y); padre(Z,Y).

Vea que ambos operandos tienen el mismo tipo de asociatividad pero que el operador “ , ” tiene precedencia 1000 y el operador “ ; ” tiene precedencia 1100. Al reformular la cláusula en notación prefija el intérprete hará una interpretación incorrecta de la definición del predicado abuelo/2:
:-(abuelo(X, Y), ;(,(padre(X, Z), madre(Z, Y)), padre(Z, Y))).
Luego resulta necesario agregar paréntesis a la disyunción:
abuelo(X,Y) :- padre(X,Z), (madre(Z, Y); padre(Z,Y)).

La siguiente tabla resume las características de algunos operadores predefinidos en SWI Prolog.

Precedencia	Modo	Nombre
1200	xfx	:-, is
1200	fx	:-
1100	xfy	;
1000	xfy	,
700	xfx	<, =, =:=, =<, >=, \=
500	yfx	+, -
400	yfx	*, /
200	fy	+, -

Cuadro 7: Table 7. Características de algunos operadores en SWI Prolog.

Conclusión Se introdujeron los operadores aritméticos básicos y diversos operadores de comparación de expresiones aritméticas. En Prolog todos los argumentos de los operadores aritméticos deben estar instanciados en la llamada. Un operador se caracteriza por su nombre/aridad, su precedencia, su posición y su asociatividad.