Aritmética

Aritmética en Prolog

PROLOG PROPORCIONA UNA SERIE DE HERRAMIENTAS ARITMÉTICAS BÁSICAS PARA ENTEROS Y NÚMEROS REALES.

- Aritmética
- 2 + 3 = 5
- 3 x 4 = 12
- 5 – 3 = 2
- 3 – 5 = -2
- 4 : 2 = 2
- 1 es el resto cuando 7 se divide por 2

- Prolog
- ?- 5 is 2+3.
- ?- 12 is 3*4.
- ?- 2 is 5-3.
- ?- -2 is 3-5.
- ?- 2 is 4/2.
- ?- 1 is mod(7,2).

Algunas instrucciones para complementar

- X = Y.
  - Unificación: intenta hacer X e Y iguales mediante unificación.
- X \= Y.
  - Su opuesto: no unifican
- X == Y.
  - Identidad.
  - Más restrictivo que X = Y.
  - Se satisface si X = Y, pero no necesariamente a la inversa.

Algunas instrucciones para complementar

- Lectura
  - read(X), % sirve para almacenar el valor a una variable
- Escritura
  - write(X),
  - display(X)
  - put(X)
- Formato:
  - nl %escribe un salto de línea
  - tab(X) %escribe X espacios en blanco

Algunas mas

- Plus sirve para sumar los argumentos recibidos.
- Ejemplo:
  - ?- plus(1,2,3).
  - ?- plus(1,2,5).
    - false.
  - ?- plus(1,2,SUMA).
    - SUMA = 3

- Between: Encuentra un numero en un rango.
- Ejemplo:
  - ?- between(1,7,5).
  - ?- between(1,7,N).
    - N = 1 ;
    - N = 2 ;
    - N = 3 ;
    - N = 4 ;
    - N = 5 ;
    - N = 6 ;
    - N = 7 .
  - ?- between(1,7,9).
    - false

Algunas mas

Operador	Descripción
<	Menor
>	Mayor
=<	Menor que
>=	Mayor que
=\=	Diferente
is	Evalúa si un numero equivale a una expresión
=:=	Igual
Mod	Modulo

Funciones	Descripción
Abs	Valor absoluto
Sign	Signo de un numero
Min	Valor mínimo
Max	Valor máximo
Random	Numero aleatorio.
Round	Redondeo
Floor	Redondeo hacia arriba
Ceiling	Redondeo hacia abajo
Sqrt	Raíz
Pown	Potencia
Pi	Valor de Pi

Definición de predicados con aritmética

- addThreeAndDouble(X,Y):- Y is (X+3) * 2.
- ?- addThreeAndDouble(1,X).
  - X=8
- ?- addThreeAndDouble(2,X).
  - X=10

Una mirada más cercana

- Es importante saber que +, -, / y * no realizan ninguna aritmética.
- Expresiones como 3+2, 4-7, 5/5 son términos ordinarios de Prolog
  - Functor: +, -, /, *
  - Aridad: 2
  - Argumentos: integersX=10
- ?- X = 3 + 2.
  - X = 3+2
- ?- 3 + 2 = X.
  - X = 3+2

Is

- Para forzar a Prolog a evaluar realmente las expresiones aritméticas, tenemos que usar is tal como lo hicimos en los otros ejemplos
- Esta es una instrucción para que Prolog realice cálculo.
- Debido a que este no es un predicado Prolog ordinario, existen algunas restricciones.
- ?- X is 3 + 2.
  - X = 5
- ?- 3 + 2 is X.
  - ERROR: is/2: Arguments are not sufficiently instantiated
- ?- Result is 2+2+2+2+2.
  - Result = 10

Restricciones para el uso de Is

- Somos libres de usar variables en el lado derecho del predicado is
- Pero cuando Prolog realmente lleva a cabo la evaluación, las variables deben ser instanciadas con un término Prolog libre de variables.
- Este término Prolog debe ser una expresión aritmética.

Notación

- Dos observaciones finales sobre las expresiones aritméticas
  - 3+2, 4/2, 4-5 son solo términos ordinarios de Prolog
  - En una notación fácil de usar: **3+2** es realmente **+(3,2)** y así sucesivamente.
  - Además, el predicado is es un predicado Prolog de dos lugares

    is(X,+(3,2)).

    - X = 5

Aritmética y listas

- ¿Qué tan larga es una lista?
  - La lista vacía tiene longitud: cero;
  - Una lista no vacía tiene longitud: uno más la longitud de su cola.

```
len([],0).

len([_|L],N):-  len(L,X), N is X + 1.
```

- ?- len([a,b,c,d,e,[a,x],t],X).
- X=7

Acclen

- El predicado acclen tiene tres argumentos
  - La lista cuya longitud queremos encontrar
  - La longitud de la lista, un entero
  - Un acumulador, haciendo un seguimiento de los valores intermedios para la longitud
- El acumulador de acclen:
  - El valor inicial del acumulador es 0
  - Añadir 1 al acumulador cada vez que podamos tomar recursivamente el encabezado de una lista
  - Cuando llegamos a la lista vacía, el acumulador contiene la longitud de la lista.

Acclen

acclen([],Acc,Acc).

acclen([_|L],OldAcc,Length):- NewAcc is OldAcc + 1, acclen(L,NewAcc,Length).

- ?-acclen([a,b,c],0,Len).
- Len=3
- yes

Acclen (árbol)

?- acclen([a,b,c],0,Len).

/ \

no ?- acclen([b,c],1,Len).

/ \

no ?- acclen([c],2,Len).

/ \

no ?- acclen([],3,Len).

/ \

Len=3 no

Agregamos un predicado acumulador

```
acclen([ ],Acc,Acc).

acclen([ _|L],OldAcc,Length):- NewAcc is OldAcc + 1, acclen(L,NewAcc,Length).

length(List,Length):- acclen(List,0,Length).
```

- ?-length([a,b,c], X).
- X=3

Recursión de cola

- ¿Por qué es acclen mejor que len?
- Acclen es cola recursiva, y len/2 no lo es.
- Diferencia:
  - En predicados recursivos de cola, los resultados se calculan completamente una vez que llegamos a la cláusula base
  - En predicados recursivos que no son recursivos de cola, todavía hay objetivos en la pila cuando llegamos a la cláusula base.

Comparación

SIN COLA RECURSIVA

```
len([],0).

len([_|L],NewLength):-
        len(L,Length),
        NewLength is Length + 1.
```

CON COLA RECURSIVA

```
acclen([],Acc,Acc).

acclen([_|L],OldAcc,Length):-
        NewAcc is OldAcc + 1,
        acclen(L,NewAcc,Length).
```

Árbol de búsqueda para len

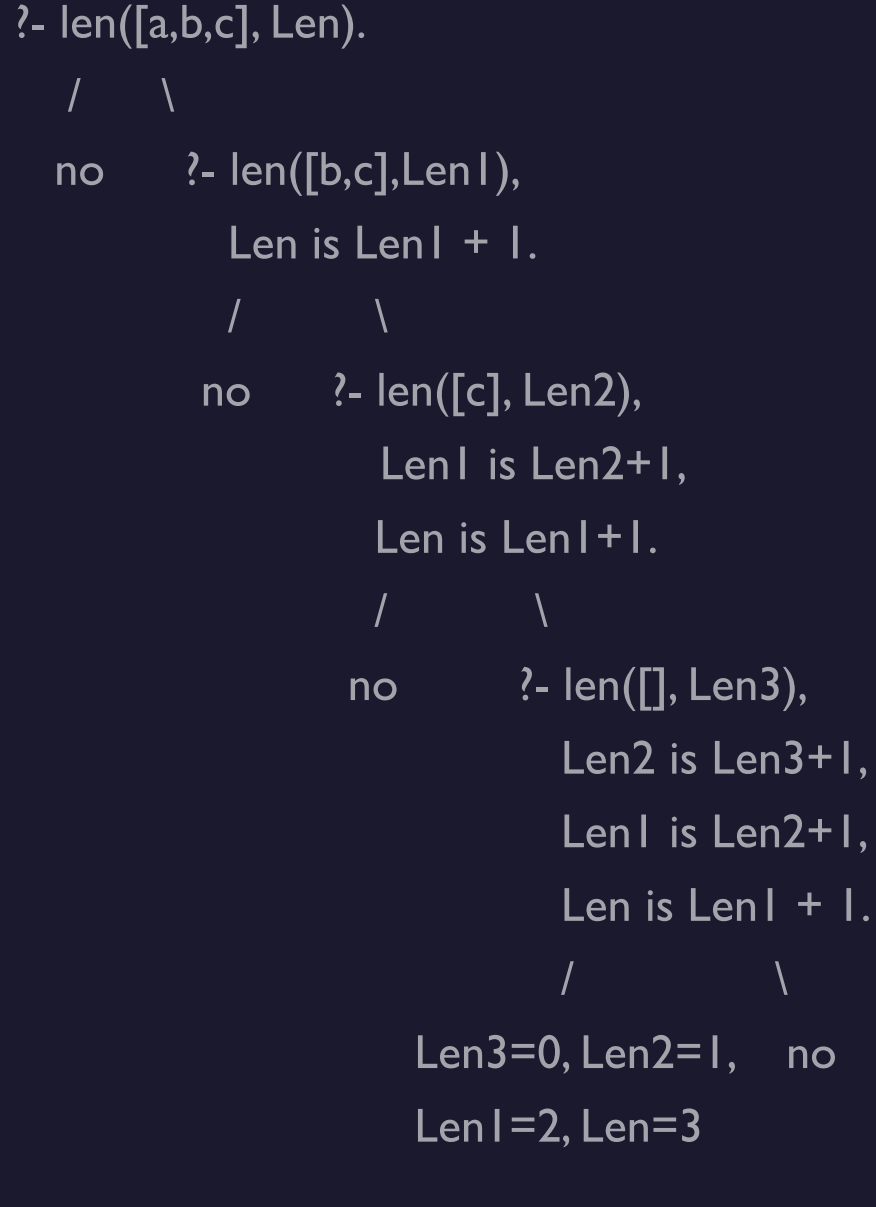
SIN COLA RECURSIVA

```
len([],0).

len([_|L],NewLength):-

        len(L,Length),

        NewLength is Length + 1.
```

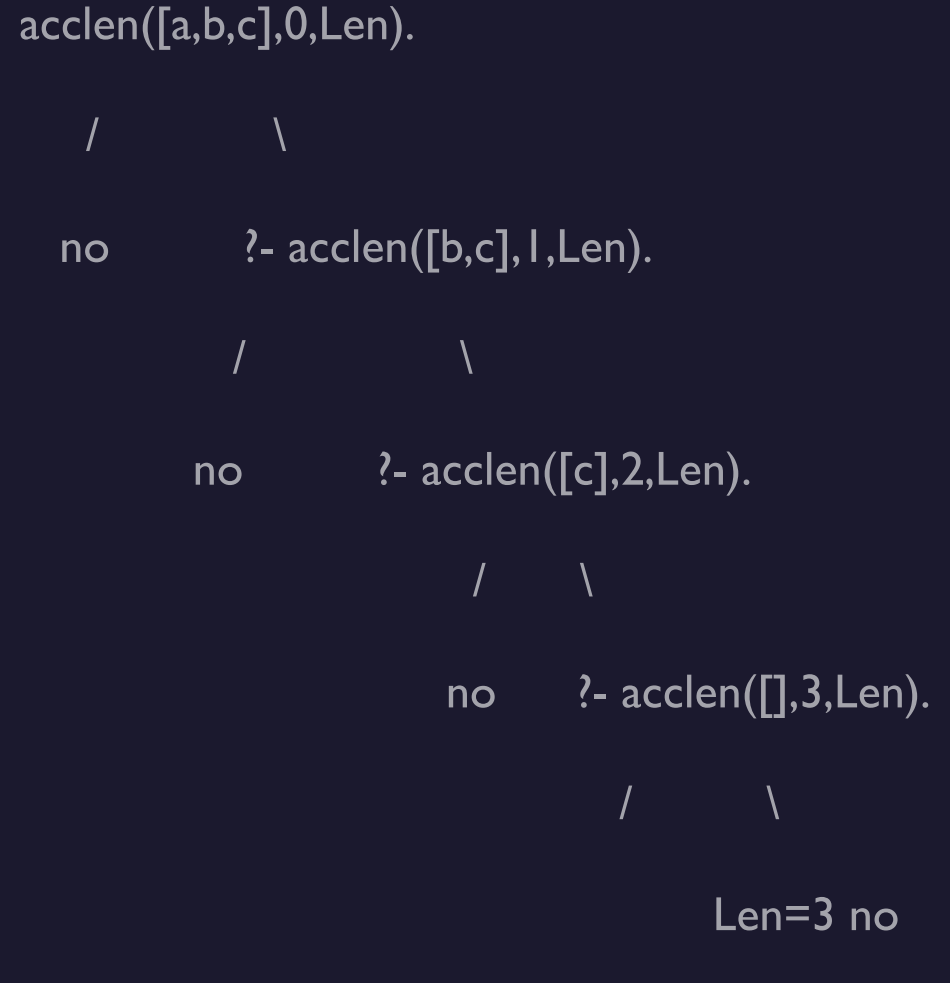


Árbol de búsqueda para acclen

CON COLA RECURSIVA

acclen([],Acc,Acc).

acclen([_|L],OldAcc,Length):- NewAcc is OldAcc + 1, acclen(L,NewAcc,Length).



Comparando enteros

ALGUNOS PREDICADOS ARITMÉTICOS DE PROLOG REALMENTE LLEVAN A CABO LA ARITMÉTICA POR SÍ MISMOS.

ESTOS SON LOS OPERADORES QUE COMPARAN ENTEROS

Aritmética	Prolog
$x < y$	X < Y
$x \leq y$	X =< Y
$x = y$	X =:= Y
$x \neq y$	X =\= Y
$x \geq y$	X >= Y
$x > y$	X > Y

Comparación de operadores

- Tienen el significado obvio
- Fuerza que se evalúe el argumento de la mano izquierda y derecha.
- ?- 4 = 4.
  - true
- ?- 2+2 = 4.
  - false
- ?- 2+2 =:= 4.
  - True

Comparación de números

- Vamos a definir un predicado que toma dos argumentos, y es verdadero cuando:
  - El primer argumento es una lista de enteros
  - El segundo argumento es el entero más alto de la lista
- Idea básica
  - Utilizaremos un acumulador
  - El acumulador realiza un seguimiento del valor más alto encontrado hasta ahora
  - Si encontramos un valor superior, el acumulador se actualizará

Max

accMax([H|T],A,Max):- H > A, accMax(T,H,Max).

accMax([H|T],A,Max):- H =< A, accMax(T,A,Max).

accMax([],A,A).

- ?- accMax([1,0,5,4],0,Max).
- Max=5

Agregamos un acumulador a max

```
accMax([H|T],A,Max):- H > A, accMax(T,H,Max).

accMax([H|T],A,Max):- H =< A, accMax(T,A,Max).

accMax([],A,A).

max([H|T],Max):- accMax(T,H,Max).
```

- ?- max([1,0,5,4], Max).
- Max=5
- ?- max([-3, -1, -5, -4], Max).
- Max= -1

Ejercicios

- ¿Que es lo que responde Prolog a la siguiente practica?

1. X = 3*4.
2. X is 3*4.
3. 4 is X.
4. X = Y.
5. 3 is 1+2.
6. 3 is +(1,2).
7. 3 is X+2.
8. X is 1+2.
9. 1+2 is 1+2.
10. is(X,+(1,2)).
11. 3+2 = +(3,2).
12. *(7,5) = 7*5.
13. *(7,+(3,2)) = 7*(3+2).
14. *(7,(3+2)) = 7*(3+2).
15. 7*3+2 = *(7,+(3,2)).
16. *(7,(3+2)) = 7*(+(3,2)).

Ejercicios

- Defina un procedimiento max(X,Y,R) que asigne a R el mayor de 2 números X e Y, si son iguales X es 0:
- Defina un incremento de predicado de 2 lugares que se mantenga solo cuando su segundo argumento sea un entero mayor que su primer argumento. Por ejemplo, incrementa(4,5) debe mantenerse, pero incrementa(4,6) no.
- Defina una suma de predicado de 3 lugares que se mantenga solo cuando su tercer argumento sea la suma de los dos primeros argumentos. Por ejemplo, suma(4,5,9) debe mantenerse, pero suma(4,6,12) no.
- Escriba un predicado sumeuno cuyo primer argumento sea una lista de enteros, y cuyo segundo argumento sea la lista de enteros obtenidos sumando 1 a cada entero de la primera lista. Por ejemplo, la consulta: ?- sumeuno([1,2,7,2],X), debe dar: X = [2,3,8,3].

Sigamos

- Escribir un programa que calcule el precio de venta al por menor, a partir del precio de venta al por mayor. (precio al por menor = precio al por mayor + ganancia) usando la siguientes convenciones para calcular la ganancia:
  - precio ≤ $100 : 20% ganancia
  - $100 < precio ≤ $1000 : 15% ganancia
  - precio > $1000 : 10% ganancia

- Definir la base de conocimiento que contenga los siguientes hechos:
  - poblacion(provincia, poblacion).
  - superficie(provincia, superficie).
  - Definir el predicado densidad(p, d) que calcule la densidad d que tiene la provincia p.
    - (densidad=poblacion/superficie).
- Definir la base de conocimiento que contenga los siguientes hechos:
  - horoscopo(signo, diaInicio, mesInicio, diaFin, mesFin).
  - Ej: horoscopo(aries,21,3,21,4).
  - Definir el predicado signo(d, m, s) que represente que los nacidos el día d del mes m pertenecen al signo s.

Algunos mas

- Escribir un procedimiento diferencia(X,Y,Z), que ligue en Z la suma de X e Y sí X es mayor a Y , o que ligue en Z la sustracción de X e Y sí X es menor. Caso contrario Z es ligado al valor de X.
- .
- Escribir un programa que permita ingresar al usuario un valor representando cierta cantidad de horas y que imprima la conversión a minutos de esas horas