Predicados Básicos de Prolog

TAII(I) - Curso 2005-2006

Existe un conjunto de predicados predefinidos en Prolog que podrás utilizar en tus programas para realizar operaciones de entrada/salida, trabajar con archivos, etc... A continuación se muestra un breve resumen de los mismos, y que están relacionados con:

1. Predicados de manejo del intérprete y la base de hechos y reglas
2. Control de flujo de ejecución
3. Predicados (operadores) de comparación
4. Predicados de comprobación de tipos
5. Entrada/Salida estándar
6. Entrada/Salida desde archivos
7. Funciones para la Depuración de Programas

1. Predicados de manejo del intérprete y la base de hechos y reglas

arg(N,estructura(arg1,...,argn),Y) Tiene éxito si Y coincide con el N-ésimo argumento de la estructura.

asserta(P) Introduce P en la base de hechos, al comienzo de la definición de P.

assertz(P) Introduce P en la base de hechos, al final de la definición de P.

chdir(+Path) Cambia el directorio de trabajo al directorio Path.

consult(File) Carga el archivo File. El nombre del archivo irá entre comillas simples, por ejemplo, ′/home/alumno/taii1/programa.pl′. Es equivalente a escribir [′/home/alumno/taii1/programa′]. El archivo va sin extensión aunque por defecto buscará el archivo File o File.pl, toma las cláusulas que encuentre en ese archivo y las inserta en la base de hechos.

delete_file(+File) Borra el fichero especificado.

exists_file(+File) Produce éxito cuando el fichero especificado existe (esto no implica que el usuario disponga de permiso de lectura o escritura sobre ese fichero).

functor(estructura(arg1,...,argn),X,Y) Tiene éxito si X coincide con el functor de la estructura, e Y con el número de argumentos.

halt Termina la ejecución del interprete.

help(S) Muestra ayuda sobre un átomo simbólico, e.g. help(assert).

listing Muestra todos los hechos y reglas introducidos en la base de datos del intérprete.

listing(P) Muestra todos los hechos y reglas sobre el predicado P.

notrace Termina el modo de traza o seguimiento.

reconsult(F) Igual que el anterior, pero todos los predicados definidos en F reemplazarán a las definiciones que ya pudieran existir en el intérprete [no está en todos los intérpretes].

rename_file(+File1,+File2) Renombra File1 como File2.

retract(P) Elimina P de la base de hechos.

shell Inicia un shell interactivo con Unix, o un terminal en Windows. El shell finaliza al teclear exit.

shell(+Command) Ejecuta un comando de Unix.

size_file(+File,-Size) Unifica la variable Size con el número de caracteres contenidos en File.

statistics Muestra una tabla con información estadística acerca de la utilización del sistema.

time(+Goal) Ejecuta *Goal* y muestra por pantalla el tiempo utilizado, el número de inferencias lógicas realizadas y la medida de *lips* (logical inferences per second).

trace Comienza el modo de traza o seguimiento de ejecución.

[+Filespec] Lee las cláusulas contenidas en el archivo especificado (*Filespec*) y las inserta en la base de datos (es una acción similar a ejecutar el predicado consult)[no está en todos los intérpretes].

2. Control de flujo de ejecución

call(P) Fuerza la comprobación de P como si se tratara de una consulta realizada al intérprete.

false Predicado sin argumentos, que siempre falla.

true Predicado sin argumentos, que siempre se da por satisfecho.

+Goal1, +Goal2 . Conjunción. Se hace cierta cuando ambas metas pueden probarse.

+Goal1; +Goal2 . Disyunción. Se hace cierta alguna de las metas puede probarse.

! Corte.

\+ **+Goal** Negación. Es ciertar si *Goal* no puede probarse.

3. Predicados (operadores) de comparación

X < Y X e Y han de estar instanciadas a dos valores numéricos; comprueba que la primera sea menor que la segunda.

X > Y X e Y han de estar instanciadas a dos valores numéricos; comprueba que la primera sea mayor que la segunda.

X <= Y X e Y han de estar instanciadas a dos valores numéricos; comprueba que la primera sea menor o igual que la segunda.

X >= Y X e Y han de estar instanciadas a dos valores numéricos; comprueba que la primera sea mayor o igual que la segunda.

X = Y Unifica X a Y.

X \ = Y Relación inversa a la anterior.

X == Y Comprueba si X e Y están actualmente asociadas. Por ejemplo, X=a, Y=a, X == Y devuelve *yes*, pero X=a, X==Y devuelve *no*.

X \ == *Y* Relación inversa a la anterior.

X is E Asocia a la variable X el valor de evaluar la expresión numérica E. Todas las variables E deben estar instanciadas a valores numéricos.

4. Predicados de comprobación de tipos

atom(X) Comprueba si X está asociada a un átomo simbólico (no numérico).

compound(X) Comprueba si X es un término compuesto.

float(X) Comprueba si X está asociada a un número real.

integer(X) Comprueba si X está asociada a un número entero.

nonvar(X) Comprueba si X es una variable libre.

rational(X) Comprueba si X está asociada a un número racional (los racionales incluyen a los enteros).

real(X) Comprueba si X está asociada a un número real [no está en todos los intérpretes].

string(X) Comprueba si X está asociada a una cadena de caracteres (escrita entre dobles comillas) [no está en todos los intérpretes].

var(X) Comprueba si X está asociada a una variable.

5. Entrada/Salida estándar

A continuación, se muestran algunos predicados básicos para poder realizar operaciones de entrada/salida desde/sobre el terminal de salida de la computadora.

current_op(?Precedence,?Type,?Name) Devuelve éxito cuando Name está definido como un operador de tipo Type cuya precedencia es Precedence. (Otros predicados relacionados con este, op/3).

display(+Term) Escribe el término Term sobre la salida estándar del dispositivo. Este predicado suele emplearse normalmente para examinar la representación interna de un término.

flush Vuelca la salida de un programa sobre la salida estándar actual (ver flush_output/1).

get(-Char) Lee de la entrada actual caracteres y unifica Char con el próximo carácter introducido (distinto al carácter blanco). Char se unifica con -1 si se trata del final de un fichero.

get0(-Char) Lee de la entrada actual caracteres y unifica Char con el próximo carácter introducido. Char se unifica con -1 si se trata del final de un fichero.

nl Escribe una línea en blando (carácter newline) sobre la salida estándar actual.

op(+Precedence,+Type,+Name) Declara a Name como un operador de tipo Type con una precedencia Precedence.

put(+Char) Escribe el carácter Char sobre la salida estándar del dispositivo.

read(-Term) Lee un término desde la entrada estándar del dispositivo.

skip(+Char) Lee y salta caracteres desde la entrada estándar hasta que encuentra el carácter Char.

tab(+Amount) : Escribe un número dado (Amount) de espacios en blanco en la salida estándar del dispositivo (Amount debe ser una expresión que pueda evaluarse como un entero positivo).

write(+Term) Escribe el término Term sobre la salida estándar.

writeq(+Term) Escribe el término Term sobre la salida estándar (sitúa el término entre comillas). En este predicado, los términos pueden ser vueltos a leer con el predicado read/1.

6. Entrada/Salida desde archivos

close(+Stream) Cierra el fichero especificado por Stream.

display(+Stream,+Term) Muestra un término Term que se encuentra en el fichero especificado por Stream.

get(+Stream, -Char) Lee el siguiente carácter imprimible de un fichero y unifica su correspondiente valor ASCII con Char.

get0(+Stream, -Char) Lee el siguiente carácter de un fichero y unifica su correspondiente valor ASCII con Char.

nl(+Stream) Escribe una línea en blanco en el fichero especificado.

open(+SrcDest,+Mode,?Stream) Apertura del fichero especificado por SrcDest (especifica un fichero Unix), el Mode puede ser de lectura (read), escritura (write) o para realizar una ampliación del mismo (append). El término Stream puede ser una variable (se instanciará a un entero que identifica mi fichero), o un átomo (en este caso se trata de un identificador de fichero). En caso de no existir el fichero lo crea.

put(+Stream,+Char) Escribe el carácter Char, en el fichero Stream.

read(+Stream,-Term) Lee un término desde un fichero.

see(+SrcDest) Abre un fichero para lectura y se sitúa al comienzo del mismo.

seeing(?SrcDest) Unifica el nombre del fichero abierto actualmente con SrcDest .

seen Cierra el fichero actualmente abierto, y devuelve la entrada estándar del dispositivo al teclado del terminal.

skip(+Stream,+Char) Lee y salta caracteres desde un fichero (Stream) hasta que encuentra el carácter Char.

tab(+Stream,+Amount) Escribe un número dado (Amount) de espacios en blanco un fichero (Stream).

tell(+SrcDest) Abre un fichero para escritura como si se tratase de la salida estándar.

telling(?SrcDest) Devuelve el nombre de el fichero abierto por tell (unifica el nombre del actual fichero de salida con SrcDest).

told Cierra el fichero que se encuentre actualmente abierto, y devuelve la salida estándar del dispositivo a la pantalla del terminal.

write(+Stream,+Term) Escribe el término Term sobre el fichero Stream.

writeq(+Term) Escribe el término Term sobre el fichero Stream (los inserta entrecomillados).

7. Funciones para la Depuración de Programas

debug Arranca el depurador (detendrá la ejecución de los programas en los puntos espías).

debugging Imprime el estado del depurador y los puntos espías sobre la salida actual.

nodebug Detiene el depurador (quita el modo traza y no se detiene en ningún punto espía).

nospy(+Pred) Borra el punto espía situado en el predicado especificado.

nospyall Borra todos los puntos espía del programa.

notrace Detiene el proceso de traza del programa.

spy(+Pred) Sitúa un punto espía en el todos los predicados especificados por Pred.

trace Arranca el proceso de traza del programa.

tracing Devuelve éxito cuando el proceso de traza está activado.