Trabajo Práctico 2
Programación Lógica

Paradigmas de Lenguajes de Programación — 2^o cuat. 2010

Fecha de entrega: 28 de Septiembre

Este trabajo consiste en implementar en Prolog algunas operaciones sobre expresiones regulares.

La solución debe realizarse en un solo archivo Prolog. Pueden utilizarse todos los predicados y metapredicados existentes.

Introducción

Para quienes no las conozcan, una expresión regular es una manera de expresar una cadena o un conjunto de cadenas a través de un patrón de texto. A cada ER E le corresponde un lenguaje $\mathcal{L}(E)$ que es el conjunto de cadenas de caracteres que representa.

La estructura de las ER que nos interesan y sus lenguajes son:

$$\begin{aligned} \mathcal{L}([a_1, a_2, a_3, \ldots, a_n]) &= \{[a_1, a_2, a_3, \ldots, a_n]\} \\ \mathcal{L}(choice(E_1, E_2)) &= \mathcal{L}(E_1) \cup \mathcal{L}(E_2) \\ \mathcal{L}(concat(E_1, E_2)) &= \mathcal{L}(E_1) \times \mathcal{L}(E_2) \\ \mathcal{L}(rep1(E_1)) &= \cup_{i=1}^{\infty} \mathcal{L}(E_1)^i \end{aligned}$$

Nota: $\mathcal{L}(E)^i = \mathcal{L}(E) \times \mathcal{L}(E) \times \ldots \times \mathcal{L}(E)$ (i veces).

Ejercicios

Primera parte

Implementar los siguientes predicados considerando las expresiones regulares definidas arriba.

Ejercicio 1

esLista (+X) que es verdadero sólo si X es una lista. ◇

Ejercicio 2

literales (+E, −Sigma) que es verdadero si Sigma es la lista de literales que aparece en la ER E. Llamamos *literales* a los elementos básicos que aparecen en las (sub)expresiones de la forma $[a_1, a_2, a_3, \ldots, a_n]$. Se espera que los literales sean caracteres. Sigma no deberá tener repetidos.

Sugerencia: implementar literal (+E,−A) que es verdadero si el carácter A aparece en la ER E (sin importar resultados repetidos).

```
?- literales(concat("a",choice("a","b")),Sigma).
Sigma = [97, 98].
```

◇

Nota: el intérprete de Prolog no muestra los caracteres en sí, sino su código ASCII. Es por eso que los caracteres se ven como números en los ejemplos.

Ejercicio 3

match(+E,+Palabra) que es verdadero sólo si Palabra es una cadena perteneciente a $\mathcal{L}$(E).

Sugerencia: implementar match(+E, +S, −Matched, −Tail) que es verdadero si matched pertenece a $\mathcal{L}$(E) y S es el resultado de concatenar Matched con Tail. ◇

Ejercicio 4

palabrasHasta(+K, +Sigma, −P) que enumera todas las palabras de longitud entre 1 y K (inclusive), usando el conjunto de literales Sigma; es decir, instancia P en cada una de esas palabras, sin arrojar resultados repetidos. Asumir Sigma sin repetidos.

Sugerencia: implementar palabrasExacto(+K, +Sigma, −P) que enumera todas las palabras cuya longitud es *exactamente* K. ◇

Ejercicio 5

lenguajeSimpleHasta(+K,+E,−P) que enumera todas las palabras de longitud entre 1 y K que pertenecen a $\mathcal{L}$(E). Realizar una implementación de tipo Generate & Test partiendo de palabrasHasta/3.

```
?- lenguajeSimpleHasta(3,concat(rep1(choice("a","b")),"c"),P).
P = [97, 99] ;
P = [98, 99] ;
P = [97, 97, 99] ;
P = [98, 97, 99] ;
P = [97, 98, 99] ;
P = [98, 98, 99] ;
false.
```

◇

Ejercicio 6

lenguajeInteligenteHasta (+K,+E,−P) que enumera todas las palabras de longitud entre 1 y K que pertenecen a $\mathcal{L}$(E). Realizar una implementación que genere los resultados aprovechando la estructura de la ER.

Sugerencia: implementar matchExacto(+K,+E,−P) que enumera todas las palabras de longitud *exactamente* K que pertenecen a $\mathcal{L}$(E), aprovechando la estructura de la ER. ◇

Segunda parte

Las expresiones regulares se pueden utilizar para capturar parte de un texto e identificarlo. Para esto se extiende la estructura de las expresiones regulares agregando group(X,E), donde X es una variable libre y E una ER. El lenguaje descripto por group(X,E) es el mismo que el de E, pero como efecto adicional se unifica a X con la porción de texto que coincidió con E en un "match".

Ejercicio 7

matchConGrupos(+E,?Palabra) que es verdadero sólo si Palabra pertenece a $\mathcal{L}$(E). Se deben unificar las variables libres de los grupos. La captura debe aplicarse la primera vez únicamente. Vale la misma sugerencia que en match/2.

```
?– matchConGrupos(concat("nombre:",group(X,rep1(choice("a","b")))),"nombre:baba").
X = [98, 97, 98, 97] ;
false.

?– matchConGrupos(concat(group(X,rep1("a")),rep1("a")),"aaaa").
X = [97] ;
X = [97, 97] ;
X = [97, 97, 97] ;
false.

?– matchConGrupos(concat(group(X,choice("m","p")),"esa"),P).
X = [109], P = [109, 101, 115, 97] ;
X = [112], P = [112, 101, 115, 97] ;
false.

?– matchConGrupos(rep1(group(X,choice("a","b"))),"ab").
X = [97] ;
false.
```

Notar que en el último caso hay un solo resultado porque la captura se aplica solamente la primera vez. ◇

Ejercicio 8

incluidoHasta(+K,+E1,+E2) que sea verdadero si todas las palabras de longitud entre 1 y K que pertenecen a $\mathcal{L}$(E1), también pertenecen a $\mathcal{L}$(E2).

Sugerencia: implementar hayDiferenciaHasta(+K,+E1,+E2) que sea verdadero si existe alguna palabra de longitud entre 1 y K que pertenezca a $\mathcal{L}$(E1) y no a $\mathcal{L}$(E2). ◇

Ejercicio 9

igualesHasta(+K,+E1,+E2) que sea verdadero si los lenguajes de E1 y E2 coinciden, teniendo en cuenta sólo las palabras de longitud menor o igual que K. ◇

1. Predicados y metapredicados útiles

- var(?X) y nonvar(?X). Observar que nonvar(X) es equivalente a not(var(X)).

```
?– var(X).
true.

?– var(42).
fail.

?– X = 42, var(X).
fail.
```

- between(+Low, +High, −Value).

  ?– between(100, 102, X).
  X = 100 ;
  X = 101 ;
  X = 102.

- member(−Elem, +List).

  ?– member(X, [cero, uno, dos, tres]).
  X = cero ;
  X = uno ;
  X = dos ;
  X = tres.

- append(?List1, ?List2, ?List3).

  ?– append([1,2,3],[4,5],A).
  A = [1,2,3,4,5]

  ?– append([1,2,3],W,[1,2,3,4,5]).
  W = [4,5]

- setof(+Template, +Goal, −Set).

  ?– setof(X,between(100, 102, X),L).
  L = [100, 101, 102].

Pautas de entrega

Se debe entregar el código impreso con la implementación de los predicados pedidos. Cada predicado debe contar con un comentario donde se explique su funcionamiento. Cada predicado asociado a los ejercicios debe contar con ejemplos que muestren que exhibe la funcionalidad solicitada. Además, se debe enviar un e-mail conteniendo el código fuente en Prolog a la dirección `plp-docentes@dc.uba.ar`. Dicho mail debe cumplir con el siguiente formato:

- El título del mensaje debe ser "[PLP;TP-PL]" seguido inmediatamente del nombre del grupo.
- El código Prolog debe acompañar el e-mail y lo debe hacer en forma de **archivo adjunto** (puede adjuntarse un .zip o .tar.gz).

El código debe poder ser ejecutado en `SWI-Prolog`. No es necesario entregar un informe sobre el trabajo, alcanza con que el código esté adecuadamente comentado.

Los objetivos a evaluar en la implementación de los predicados son:

- Corrección.
- Declaratividad.
- Reuso de predicados previamente definidos.
- Uso de unificación, backtracking y reversibilidad de los predicados que correspondan.
- Salvo donde se indique lo contrario, los predicados no deben instanciar soluciones repetidas. Vale aclarar que no es necesario filtrar las soluciones repetidas si la repetición proviene de las características de la entrada.

  Importante: se admitirá un único envío, sin excepción alguna. Por favor planifiquen el trabajo para llegar a tiempo con la entrega.