CC41A Lenguajes de Programación
Tarea Recuperativa - Semestre Otoño 2006
Profesor: Luis Mateu

(actualizado el 26 de Julio de 2006)

Una expresión booleana contiene variables, que pueden tomar los valores verdadero o falso (1 o 0), y los operadores lógicos de conjunción, disyunción y negación (*vacío*, + y ~). La negación tiene la mayor precedencia, seguida de la conjunción y por último la disyunción. La asociatividad es de izquierda a derecha. También se usa parentización cuando la regla de precedencia o asociatividad no es la adecuada. Por simplicidad, las variables son de *un solo carácter alfabético* y la expresión *no contiene espacios*. Ejemplos:

Expresión	***Expresión desambiguada con paréntesis***
ab+~c	(a b) + (~ c)
a+b+c	(a+b)+c
a~(bcd)+~bc	(a (~((b c) d)))+((~b) c)

En esta tarea se pide programar la relación `resolver(Exp, L)`. Cada elemento de la lista `L` es un mapa de asociaciones entre las variables de la expresión `Exp` y valores de verdad, como por ejemplo `[(a,0),(b,1),(c,0)]`. `L` contiene todos los mapas de asociaciones que hacen que la expresión `Exp` se haga verdadera. Ejemplos:

Consulta	***Respuestas***
`?- resolver("~ab", L).`	`L=[[(a,0),(b,1)]]`
`?- resolver("a+b", L).`	`L=[[(a,0),(b,1)],[(a,1),(b,0)], [(a,1),(b,1)]]`
`?- resolver("ab+~c", L).`	`L=[ [(a,0),(b,0),(c,0)], [(a,0),(b,1),(c,0)], [(a,1),(b,0),(c,0)], [(a,1),(b,1),(c,0)], [(a,1),(b,1),(c,1)]]`
`?- resolver("a+~a", L).`	`L=[[(a,0)],[(a,1)]].`
`?- resolver("~(a+b)", L).`	`L=[[(a,0),(b,0)]]`

Es irrelevante el orden en que aparecen (i) las variables en un mapa de asociación y (ii) los mapas de asociaciones en L. Un mapa de asociacion *no* puede contener variables *duplicadas* y la lista de mapas *no* puede tener asociaciones *duplicadas*.

Indicaciones

- El examen de 2005 le dará una buena idea sobre como hacer el análisis sintáctico de la expresión:

  http://www.dcc.uchile.cl/~lmateu/CC41A/controles/ex-051.pdf
- Construya un árbol de sintaxis abstracta para la expresión de manera similar a como se indica en el enunciado de la tarea recuperativa de 2005:

  http://www.dcc.uchile.cl/~lmateu/CC41A/2005/tareas/t4.html
- Extraiga del árbol abstracto la lista de variables.
- Haga un predicado que genere todas las combinaciones de valores posibles para la lista de variables.
- Evalúe el árbol de sintaxis abstracta de la expresión con cada una de las posibles combinaciones de valores.
- Utilice `findall` para encontrar todas las soluciones de una consulta.
- Utilice `char_code(+, Plus)` para obtener el código ASCII de +.
- Utilice `char_type(K, alpha)` para determinar si K es alfabético.
- Utilice `string_to_atom("a", X)` para obtener "a" en forma de símbolo.
- Utilice `atom(A)` para determinar si A es un átomo.
- Consulte la documentación de SWI-Prolog.

Cuando programe en Prolog escriba reglas simples y pruébelas en forma independiente. Pruebe incrementalmente cada una de las reglas a medida que resuelve problemas más complejos.

Entrega

La tarea se entrega en U-cursos. El plazo vence *impostergablemente* el Lunes 14 de Agosto. *No* se aceptarán tareas que funcionen a medias.