Equipo Dinamita
Luis Mario Escobar Rosales 420003818
Liera Montaño Miguel Ángel 317257421

Practica 06

20 de enero 2021

Ejercicios

1.-Implementa una relación "traducir(original, traduccion)", de modo que "traduccion.es la traducción en español de .oriqinal"siendo una lista representando una cadena en algún idioma. El idiomaqueda a su elección mientras se pueda mantener con el alfabeto usual. Deberán crear una base de datosque les sea de ayuda como su diccionario.

Primero se genero una base de datos con palabras suficientes para traducir cortas frases, en este caso dic(ingles,espanol), luego planteamos la relacion traducir que de forma que la traduccion fuera simetrica, es decir sin importar el orden de la entrada, la salida iba dar con su respectiva relacion, tanto para una palabra como para una lista de ellas.

2.-A partir del mundo de cubos proporcionado, define las reglas y hechos 'sobre(X, Y)', 'hastaArriba(X)' y 'bloqueado(X)' necesarias para después implementar:

a) 'hastaAbajo(X)' que devuelve el cubo que se encuentre hasta el fondo de una pila de cubos.

b) 'mover(X, Y)' que permita mover X sobre Y si este último esta encima.

Sin duda este fue uno de los ejercicios que más nos ha costado implementar en la practica, ya que sabíamos de antemano que la relación sobre() estaba conformada por hechos, además que llegamos muy rapidamente a la codificación de hastaArriba() hastaAbajo() y bloqueado(), ya que cada una de estas dependia de sobre(). Lo complicado fue cuando no sabíamos como modificar la base de datos que prolog estaba formando con sobre(). Investigamos en muchas fuentes hasta que dimos con los predicados dynamic, assert y retract que nos permiten hacer cambios a nuestra base de datos. Finalmente la implementación de mover dependio de verificar y eliminar las referencias que sobre() para cada cubo que moviamos, esto gracias a la funcion auxiliar sobre()

3.-Tomando el siguiente autómata finito no determinista, define la función de transición, los estados inicial y final para que se pueda resolver la pregunta 'aceptar(S).' con S una lista de símbolos quepuede o no ser aceptada por el AFN.

Se llego al resultado ya que pense en una especie de camino que deberia de recorrerse para llegar de un punto a otro. SI nos daban una lista, teniamos que descomponerla, y verificar que partiendo de el estado inicial y con el primer elemento de la lista de simbolos si el siguiente estado correspondia a dicho simbolo, y asi recursivamente hasta llegar al estado final. Como aceptar solo recibe unalista, pensamos en aceptarAux que recibe como referencia los estados inicial

4.-Implementa una relación 'mezclar(L1, L2, LMerge)' donde L1 y L2 son listas previamente ordenadas por una relación ordenar(L, LOrd) de tu creación. LMerge es la lista ordenada de la mezcla de ambas listas. Deberás utilizar el operador de corte ('!').

Se implemento la relacion ordenar(L,Lord) que dada una lista la ordenada Luego implementamos mezclar(L1,L2,Lmerge), donde las dos listas deben estar ordenadas, en este caso es mas claro ver el uso del operador de (¡"), donde lo utilizamos para acortar posibles caminos cuadno evaluamos A<B,A=B,A>B, claramente solo uno de estos puede ser verdad por lo que al usr operador de corte el prorgama corta las ramas o lo caminos que se pudieron haber generado, haciendo que el programa sea mas eficiente.