INF-253 Lenguajes de Programación
Tarea 5: Prolog

Profesor: Esteban Daines - Roberto Diaz
Ayudante Cátedras: Sebastian Godinez
Ayudante Tareas: Gabriel Valenzuela - Monserrat Figueroa

29 de Noviembre de 2018

1. Objetivos

Conocer y aplicar correctamente los conceptos y técnicas de programación logica, utilizando el lenguaje Prolog.

2. Reglas

- Se presentarán 4 problemas a implementar en Prolog. Cada problema debe ser resuelto en archivos separados.
- En cada problema se otorga la libertad de escoger el formato de como se realizaran las consultas, de implementar utilizando la estructura que deseen y la manera de retornar los resultados. Siempre y cuando tenga coherencia con el problema indicado.
- A partir del punto anterior, es necesario que en cada archivo donde se implemente cada problema, exista un comentario con las instrucciones de como realizar las consultas.
- La tarea debe estar desarrollada en Prolog utilizando SWI-Prolog, el cual puede ser obtenido en el siguiente link:
  - http://www.swi-prolog.org/download/stable

3. Problemas

1. **Subconjuntos de suma dada**

   Sea W un conjunto de enteros no negativos y M un número entero positivo. El problema consiste en implementar un algoritmo para encontrar todos los posibles subconjuntos de W cuya suma sea exactamente M.

2. **Alcance de ciudades**

   Se tiene una cantidad de combustible limitada con la que se quiere viajar. el problema trata de encontrar todas las ciudades alcanzables desde una ciudad inicial con la cantidad de combustible disponible. Para esto se pide implementar un grafo **dirigido** donde el peso de cada arco es la cantidad de combustible que se consume desde una ciudad a otra.

   El mapa de las ciudades debe ser diseñado por el grupo, cumpliendo las siguientes condiciones:

   - Mínimo 10 ciudades.
   - Mínimo 15 arcos.
   - Todas las ciudades deben ser alcanzables desde cada una.
   - Los pesos de los arcos no pueden ser todos iguales.
   - Debe presentar al menos un ciclo.
   - No se permitirá tener hechos que indiquen explícitamente todos los posibles caminos o peso total desde una ciudad a otra.

3. **Maquina de estados finitos**

   Un autómata finito o maquina de estados finitos, es un modelo matemático de una máquina que acepta cadenas de un lenguaje definido sobre un alfabeto. Consiste en un conjunto finito de estados y un conjunto de transiciones entre esos estados, que dependen de los símbolos de la cadena de entrada. El autómata finito acepta una cadena x si la secuencia de transiciones correspondientes a los símbolos de x conduce desde el estado inicial a un estado final.

   Se pide diseñar un autómata que cumpla las siguientes condiciones:

   - Mínimo 10 estados.
   - Mínimo 2 estados de aceptación.
   - El alfabeto debe contener mínimo 3 símbolos.
   - Mínimo un ciclo (mínimo de tres estados).
   - Todas los símbolos del alfabeto deben ocuparse en al menos una transición .

   Además, se debe poder realizar una consulta, dado una cadena de símbolos, saber si esta pertenece al lenguaje o no, entregando el estado en que se detuvo.

4. **TDA Diccionario**

   Para este problema se pide emular el funcionamiento de métodos de un diccionario de Python a través de consultas.

   - Agregar: Se pide agregar un elemento con su llave correspondiente al diccionario.

- Obtener valor: Dado una llave retornar el valor relacionado.
- Obtener llaves: Retornar todas las llaves que se han agregado hasta el momento.
- Obtener todos los valores: Retornar todos los valores que se han agregado hasta el momento .
- Largo: Retornar la cantidad de relaciones que se tienen en el diccionario.

4. Archivos a Entregar

- Problema 1: subconjunto.pro o subconjunto.pl.
- Problema 2: ciudades.pro o ciudades.pl.
- Problema 3: fsm.pro o fsm.pl.
- Problema 4: set.pro o set.pl.
- grafo.png o grafo.jpg: Imagen del grafo implementado en el problema 2.
- fsm.png o fsm.jpg: Imagen del autómata implementado en el problema 3.

5. Sobre Entrega

- Los problemas que no tengan comentadas las instrucciones de consultas **no serán revisados**.
- Se debe trabajar en grupos de dos personas, no es necesario que sean del mismo curso.
- La entrega debe realizarse en tarball (tar.gz) y debe llevar el nombre: Tarea5LP_RolIntegrante-1_RolIntegrante-2.
- El archivo README.txt debe contener nombre y rol de los integrantes del grupo.
- La entrega será mediante moodle y el plazo máximo de entrega es hasta el domingo 16 de Diciembre a las 23:55.
- Por cada hora o fracción de atraso se descontarán **50 puntos**. En los primeros 20 minutos el descuento será de sólo **5 puntos**.
- Las copias serán evaluadas con nota 0.

6. Evaluación

Puntajes

- Problema 1: 25 puntos.
- Problema 2: 25 puntos totales
  - Implementación de grafo: 10 puntos.
  - Determinar ciudades alcanzables 15 puntos.
- Problema 3: 25 puntos totales.
  - Implementación de autómata: 8 puntos.
  - Determinar si la cadena pertenece al lenguaje: 7 puntos.
  - Entregar el estado en que se detuvo: 10 puntos.
- Problema 4: 25 puntos totales.
  - Agregar: 8 puntos.
  - Obtener valor: 5 puntos.
  - Obtener llaves y valores: 3 puntos cada una.
  - Largo: 6 puntos.

Descuentos

- No respetar condiciones en el diseño del autómata o grafo (4 puntos c/u)
- No respetar reglas de entrega (40 puntos)