Universidad Simón Bolívar
Dpto. de Computación y Tecnología de la Información
CI3661 - Taller de Lenguajes de Programación I
Enero-Marzo 2014

Programación Lógica - Tarea

1. Idiot Sort (4 puntos)

¿Cansado de implantar QuickSort, MergeSort, ShellSort, Insertion Sort, Selection Sort y Bubble Sort? Llame ya al 800-PROLOG y obtenga `idiotsort(Lista,Ordenada)`. Este predicado triunfa si `Ordenada` tiene los mismos elementos de `Lista` pero ordenados según su valor ascendente, y lo hace de una manera muy idiota: genera permutaciones de `Lista` y verifica si alguna de ellas está ordenada. Es una estafa, pero permite saber si Ud. domina la técnica generar-probar y sabe explotar el *backtracking* de Prolog para conseguirlo.

Su solución debe tener *exactamente* tres predicados: el principal, el generador de soluciones y el verificador. Todos los predicados deben ser puros, de manera que se puedan usar para ordenar y para desordenar.

2. Cuadrados mágicos pandiagonales (5 puntos)

Un Cuadrado Mágico Pandiagonal[1], también conocido como Cuadrado Diabólico, es un cuadrado de n filas y n columnas, conteniendo los números enteros desde 1 hasta n^2, dispuestos de manera tal que la suma de los números en cada fila, en cada columna, en cada diagonal mayor y en cada diagonal menor sea $(n^3 + n)/2$. Un cuadrado mágico de tamaño 4 puede ser

1	8	10	15
14	11	5	4
7	2	16	9
12	13	3	6

Note que la suma en cada fila y cada columna es precisamente $(4^3+4)/2 = 34$, resultado que también se cumple en ambas diagonales mayores, pues $1+11+16+6 = 12+2+5+15 = 34$, y en las diagonales menores $7 + 13 + 10 + 4 = 8 + 5 + 9 + 12 = 14 + 2 + 3 + 15 = 34$.

Escriba el predicado `diabolico(Lista)`, que triunfe si su único argumento es una lista de números enteros que corresponda con un cuadrado mágico pandiagonal de lado 4. El ejemplo anterior se consultaría

```
?- diabolico([1,8,10,15,14,11,5,4,7,2,16,9,12,13,3,6]).
yes
```

Su predicado debe ser puro, en el sentido que si es consultado con una variable sin unificar, permita *enumerar* todos los cuadrados mágicos pandiagonales, esto es

```
?- diabolico(Evil).
Evil = [1,8,10,15,14,11,5,4,7,2,16,9,12,13,3,6] ? ;
... otros cuadrados mágicos pandiagonales ...
```

3. Un agente de viajes (6 puntos)

Escribiremos una serie de predicados para asistir al pasajero en preparar un itinerario de viaje.

3.1. Representando vuelos y horarios

Un vuelo particular tiene hora de salida, hora de llegada y días en los cuales está disponible, hecho que representaremos utilizando *estructuras* Prolog de la forma

```
salida / llegada / numero / lista-de-dias
```

donde:

- `salida` y `llegada` son *estructuras* de la forma `12:10`, `7:20`, `21:30` que representan horas del día. Note que esto requerirá definir `":"` como un operador infijo.
- `numero` es un átomo que representa el número de vuelo según la aerolínea.
- `lista-de-dias` puede tener tres interpretaciones:
  1. Puede ser una lista conteniendo átomos `lun`, `mar`, `mie`, `jue`, `vie`, `sab` o `dom` para indicar los días de la semana en que el vuelo ocurre.
  2. Puede ser el átomo `todos` para indicar todos los días de la semana.
  3. Puede ser el átomo `habiles` para indicar lunes a viernes, ambos inclusive.

Así, podremos escribir

```
9:40 / 12:10 / nw600 / [lun, mar, mie, jue, vie]
```

Ahora podremos expresar los horarios de vuelo, usando el predicado `horario/3` para representar todos los vuelos entre dos ciudades particulares. Los dos primeros argumentos del predicado serán las ciudades origen y destino del vuelo, respectivamente, mientras que el tercer argumento será una lista con todos los posibles vuelos. Por ejemplo

```
horario(new_york,
        san_francisco,
        [ 12:10 / 5:00 / nw610 / [ lun, mar, mie, jue, vie ],
          5:30 / 10:15 / united440 / [ lun, mar, vie ],
          9:30 / 14:15 / aa120 / todos ]).
```

El archivo `agente.pro` que Ud. puede descargar del mismo lugar donde descargó este enunciado, contiene una base de datos modelo con la definición de algunos vuelos. Su primer trabajo es agregar vuelos adicionales de manera tal que si existe un vuelo desde la Ciudad A hasta la Ciudad B, también exista un vuelo desde la Ciudad B hasta la Ciudad A. Puede especificar los días, horas y nombres de vuelo que desee.

3.2. Cálculo de Rutas

Se desea que Ud. implante el predicado `ruta(Origen,Destino,Dia,Ruta)` que triunfe si `Ruta` es una lista de vuelos entre `Origen` y `Destino` que pueda llevarse a cabo el `Dia` indicado, sujeto a las siguientes condiciones:

- Si hay un vuelo directo entre `Origen` y `Destino`, la lista que contiene su horario es una `Ruta` válida.
- Si hay un vuelo directo entre `Origen` e `Intermedio`, y una ruta entre `Intermedio` y `Destino`, la Ruta que comienza con el vuelo entre `Origen` e `Intermedio` es válida si hay suficiente tiempo entre la llegada a Intermedio y la salida desde Intermedio.

Supondremos que en los aeropuertos grandes (New York, Chicago y Los Angeles) se requieren 90 minutos para el transbordo; en los aeropuertos medianos (San Francisco, Dallas y Miami) se requieren 60 minutos para el transbordo, y en el resto de los aeropuertos bastan 40 minutos.

3.3. Cálculo de Giras

Se desea que Ud. implante el predicado `gira(Ciudades,Dias,Ruta)` que triunfe si `Ruta` es una lista de vuelos tales que permita visitar las ciudades indicadas en la lista `Ciudades`, en ese orden, pasando *al menos* tantos `Dias` en cada ciudad como se haya indicado.

3.4. Interfaz de Usuario

Utilice las facilidades de lectura y escritura de Prolog para proveer el predicado `ui/0` que ofrezca una interfaz de menú simple, que permita al usuario preparar rutas y giras. La interfaz debería permitir resolver problemas como:

1. ¿Cómo puedo ir desde A hasta B? Debería mostrar todas las maneras posibles de viajar, dando preferencia a los vuelos con menor cantidad de escalas.
2. ¿En cuáles días puedo viajar desde A hasta B? Debería mostrar todas las maneras posibles de viajar, dando preferencias a los vuelos con menor cantidad de escalas, y ordenados según día de la semana.
3. ¿En cuáles días puedo viajar directamente entre A y B?
4. ¿Cómo puedo hacer una gira entre A y B, con paradas de N días?
5. ¿Cómo puedo hacer un viaje de ida y vuelta entre A y B, posiblemente con paradas de N días?

4. Fracciones periódicas (5 puntos)

Escriba el predicado `periodo(Dividendo,Divisor,Periodo)` que triunfe *exactamente una vez* si `Periodo` es la *lista* de dígitos que constituyen el período al calcular la división. El predicado debe fallar si `Dividendo` o `Divisor` no son números enteros positivos, o si `Divisor` es el cero. Por ejemplo,

```
?- periodo(3,4,C).
C = [0]            % 3/4 = 0.750000000...
yes
?- periodo(4,3,C).
C = [3]            % 4/3 = 1.333333333...
yes
?- periodo(2,11,C).
C = [1,8]          % 2/11 = 0.1818181818...
yes
?- periodo(1,7,C).
C = [1,4,2,8,5,7]  % 1/7 = 0.142857142857...
yes
?- periodo(2,0,C).
no.
?- periodo(-4,6,C).
no
```

Entrega de la Implementación

Ud. debe entregar un archivo `.tar.gz` o `.tar.bz2` (**no** puede ser `.rar`, ni `.zip`) que al expandirse genere un directorio con el nombre de su grupo (e.g. `G42`) dentro del cual encontrar **solamente**:

- Los archivos `idiotsort.pro, cuadrados.pro, agente.pro` y `periodo.pro` conteniendo el código fuente Prolog para implantar los predicados correspondientes a cada sección.
- Se evaluará el correcto estilo de programación [2], empleando una indentación adecuada y consistente en *cualquier* editor ("profe, en el mío se ve bien" es inaceptable).
- El archivo **debe** estar completa y correctamente documentado. Es **inaceptable** que la documentación tenga errores ortográficos.
- **Fecha de Entrega.** Domingo 2014-06-01 hasta las 23:59 VET
- **Valor de Evaluación.** Veinte (20) puntos.

Referencias

[1] Página de Cuadrados Mágicos Pandiagonales en Wikipedia

[2] Coding Guidelines for Prolog