Exercícios de Lógica em Programação

Entrega: 30/05/2006

1) Assuma que lhe foi pedido que fizesse um programa para fazer planejamento de viagens. O programa é muito simples, mas deve ser capaz de responder questões do tipo:

- Em que dias da semana há um vôo direto de Londres para Ljubljana?
- Como posso chegar de Ljubljana a Edimburgo na quinta-feira?
- Tenho que visitar Milão, Ljubljana e Zurique, começando meu vôo a partir de Londres na terça-feira e retornando para Londres na sexta-feira. Em que seqüência devo visitar estas cidades de forma que eu não tenha mais de um vôo por dia?

O programa deve estar baseado no banco de dados com informações de vôos abaixo, onde `timetable(Place1,PLace2,List_of_flights)` representa a informação sobre cada vôo com `Place1` sendo origem, `Place2`, destino, e `List_of_flights`, uma lista contendo informações sobre cada vôo estruturada da seguinte forma: `hora de saída/hora de chegada/número do vôo/lista de dias`.

```
timetable(edinburgh,london,
[ 9:40/10:50/ba4733/alldays,
 13:40/14:50/ba4773/alldays,
   19:40/20:50/ba4833/[mo,tu,we,th,fr,su]]).

timetable(london,edinburgh,
[ 9:40/10:50/ba4732/alldays,
 11:40/12:50/ba4752/alldays,
   18:40/19:50/ba4822/[mo,tu,we,th,fr]]).

timetable(london,ljubljana,
[13:20/16:20/ju201/[fr],
   13:20/16:20/ju213/[su]]).

timetable(london,zurich,
[ 9:10/11:45/ba614/alldays,
 14:45/17:20/sr805/alldays]).

timetable(london,milan,
[ 8:30/11:20/ba510/alldays,
 11:00/13:50/az459/alldays]).

timetable(ljubljana,zurich,
[11:30/12:40/ju322/[tu,th]]).

timetable(ljubljana,london,
```

```
[11:10/12:20/yu200/[fr],
 11:25/12:20/yu212/[su]]).

timetable(milan,london,
[ 9:10/10:00/az458/alldays,
 12:20/13:10/ba511/alldays]).

timetable(milan,zurich,
[ 9:25/10:15/sr621/alldays,
 12:45/13:35/sr623/alldays]).

timetable(zurich,ljubljana,
[13:30/14:40/yu323/[tu,th]]).

timetable(zurich,london,
[ 9:00/9:40/ba613/[mo,tu,we,th,fr,sa],
   16:10/16:55/sr806/[mo,tu,we,th,fr,su]]).

timetable(zurich,milan,
[ 7:55/8:45/sr620/alldays]).
```

Seu problema principal é encontrar rotas exatas entre duas cidades num determinado dia da semana. A rota pode ser programada como: `route(Place1,Place2,Day,Route)`, que satisfaz os seguintes critérios:

- o ponto de partida da rota é Place1;
- o ponto de chegada da rota é Place2;
- todos os vôos devem estar agendados para o mesmo dia da semana (`Day`);
- todos os vôos em `Route` devem estar na relação `timetable`;
- deve haver tempo suficiente para transferência entre vôos.

A rota deve estar estruturada numa lista da seguinte forma: `From-To:número do vôo:Tempo de partida`.
Você vai precisar também dos seguintes predicados auxiliares:

- `flight(Place1,PLace2,Day,Flight_num,Dep_time,Arr_time)` que diz que existe um vôo entre `Place1` e `Place2` no dia da semana `Day` com tempos de chegada e partida.
- `deptime(Route,Time)`, tempo de partida de `Route` é `Time`.
- `transfer(Time1,Time2)`, há uma diferença de pelo menos 40 minutos entre `Time1` e `Time2` suficiente para fazer a transferência de um vôo para outro.

Escreva um programa utilizando esta informação que responda as perguntas apresentadas acima.

2) Dê um exemplo de duas estruturas, `s` e `t` tal que `s` e `t` **não** tenham variáveis comuns, mas que a unificação falha por causa da verificação de ocorrência (Exemplo: X = f(X) falha com verificação de ocorrência).

3) Escreva um programa em Prolog que encontre a interseção de dois conjuntos.

4) Escreva um programa em Prolog que conte o número de regras e fatos de um outro programa Prolog que está armazenado em disco.