# Terceiro Trabalho de Sistemas Inteligentes

# Entrega: 17/05/2013

18 de Abril de 2013

1. Assuma que lhe foi pedido para fazer um programa para planeamento 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 Edinburgh 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 que 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 em Prolog, utilizando a informação dada, que responda às perguntas apresentadas acima.

Escreva o mesmo programa utilizando uma linguagem (imperativa/procedural) da sua escolha.

2. Escreva um programa em Prolog, utilizando DCG (Definite Clause Grammar), que faça análise de frases em português. Uma parte de uma gramática para o português é dada na Figura 2, onde a primeira sentença fala sobre a estrutura de frases no singular e a segunda fala sobre a estrutura de frases no plural. Esta sintaxe é aceita por Prolog! (consultar "Definite Clause Grammars" - DCGs em qualquer livro sobre Prolog). Uma sentença é composta por uma frase nominal seguida por uma frase verbal. Parte das regras que definem uma frase nominal estão representadas pela cláusula `frase_nom`.

   Defina o restante da gramática (frases verbais, complementos) e um dicionário de palavras (verbos, nomes, adjetivos, etc) e utilize o conjunto de frases da Figura 2 para testar seu programa. **Atenção aos apóstrofes que devem ser utilizados quando as palavras nas frases começam com letra maiúscula!**

---

```
sentenca --> frase_nom, frase_verb.
sentenca --> frase_nom_p, frase_verb_p.

frase_nom --> artigo_f, subst_f.
frase_nom --> artigo_m, subst_m.
frase_nom --> subst_f.
frase_nom --> subst_m.

frase_nom_p --> artigo_p_f, subst_p_f.
frase_nom_p --> artigo_p_m, subst_p_m.
frase_nom_p --> subst_p_f.
frase_nom_p --> subst_p_m.
```

Figure 1: Subconjunto de estrutura da língua portuguesa

---

Faça uma pequena modificação no seu programa para que este retorne a estrutura gramatical das frases analisadas. Por exemplo, a consulta `sentenca(X,['A',vida,corre],[])` deveria retornar na variável X o valor:

`sent(frase_nom(artigo('A'),substantivo(vida)),frase_verbal(verbo(corre))).`

Observe que a frase do conjunto de exemplos corretos "O sino bateu na menina" é sintaticamente correta, mas é semanticamente incorreta. Há técnicas para codificação de gramáticas com semântica associada, porém **este tema está fora do escopo deste curso e deste trabalho.**

Escreva o mesmo programa utilizando uma linguagem (imperativa/procedural) da sua escolha.

FRASES CORRETAS

```
sentenca(['A',menina,corre,para,a,floresta],[]).
sentenca(['A',menina,corre,para,a,mae],[]).
sentenca(['A',vida,corre],[]).
sentenca(['O',tempo,corre],[]).
sentenca(['O',cacador,correu,com,os,lobos],[]).
sentenca(['A',noticia,correu,pela,cidade],[]).
sentenca(['As',lagrimas,corriam,pelo,rosto],[]).
sentenca(['O',rio,corre,para,o,mar],[]).
sentenca(['A',menina,bateu,a,porta],[]).
sentenca(['A',porta,bateu],[]).
sentenca(['O',vento,bateu,a,porta],[]).
sentenca(['A',menina,bateu,na,porta],[]).
sentenca(['O',martelo,bateu,na,porta],[]).
sentenca(['A',menina,bateu,no,cachorro],[]).
sentenca(['A',menina,bateu,no,tambor],[]).
sentenca(['Os',tambores,bateram],[]).
sentenca(['O',sino,bateu],[]).
sentenca(['A',menina,corre],[]).
sentenca(['A',vida,correu],[]).
sentenca(['A',noticia,correu,para,a,floresta],[]).
sentenca(['A',vida,correu,com,os,lobos],[]).
sentenca(['A',menina,bateu,a,mae],[]).
```

FRASES INCORRETAS

```
sentenca(['A',tempo,corre],[]).
sentenca(['O',tempo,correram],[]).
sentenca(['A',cacador,corriam,pela,rosto],[]).
sentenca(['A',tambores,correu,pela,floresta],[]).
sentenca(['Os',tambores,bateu,na,porta],[]).
sentenca(['O',sino,bateu,na,meninas],[]).
```

Figure 2: Conjunto de frases para teste (ao alto, frases corretas; embaixo, frases incorretas)

Juntem-se em grupos de 2 elementos, onde um elemento do grupo vai fazer a implementação em Prolog e o outro vai fazer a implementação em outra linguagem. No relatório vão relatar as facilidades/dificuldades de implementação que cada um teve.

Entregar:

1. trabalho escrito com a descrição da implementação.

   Organização do trabalho escrito:

   Introdução

   Descrição/Representação do problema em Prolog

   Descrição/Representação do problema em outra linguagem

   Comentários sobre representação do conhecimento e facilidades/dificuldades de implementação nas duas linguagens

   Comentários Finais e Conclusões

2. enviar o código fonte dos programas, como compilar e formato da entrada para cada problema, isto é, um pequeno manual de como rodar os programas (pode ser um 'help' ou 'readme'). Além disso, em que ambiente foi compilado (tipos e versões do SO e da linguagem). Seu programa deve correr na minha máquina (com fedora instalado). Não assuma que eu tenho uma IDE (Integrated Development Environment) de qualquer tipo. O programa deve correr na linha de comando.

   A submissão será através do Moodle UP. Por favor, não inclua caracteres acentuados no nome do ficheiro de submissão.

Todos os trabalhos serão apresentados em data a combinar. **Os dois componentes do grupo deverão estar presentes durante a apresentação**. Quem nao estiver presente vai ter nota zero!