Universidade do Minho

Licenciatura em Engenharia Informática

2011

SRCR

Grupo 8

Trabalho Pratico 1

Ano Lectivo de 2010/2011

54745 **André Pimenta**
54808 **Cedric Pimenta**
54825 **Daniel Santos**
54738 **João Gomes**
54802 **Milton Nunes**

11 de Abril de 2011

Resumo

Neste relatório serão apresentados todos os passos fundamentais da realização do segmento 1 do trabalho práctico da cadeira de Sistemas de Representação de Conhecimento e Raciocínio, juntamente com todos os conceitos fundamentais e importantes da programação em lógica. Serão apresentadas todas as decisões e as devidas jsutificações destas mesmas decisões na elaboração deste segmento

Conteúdo

Conteúdo **i**

Lista de Figuras **ii**

1 Introdução **1**
1.1 Motivação e objectivos 1
1.2 Estrutura 1
1.3 Introdução 1

2 Preliminares **3**
2.1 Estudos Anteriores 3
2.2 Programação em lógica e PROLOG 3

3 Descrição do Trabalho e Análise de Resultados **5**
3.1 Base de conhecimento 5
3.2 Funcionalidades basicas 6
3.2.1 Identificar os serviços existentes num determinado local, região ou país 6
3.2.2 Identificar os locais onde esteja presente um determinado serviço ou conjunto de serviços 7
3.2.3 Identificar os serviços que não se podem encontrar numa determinada região 7
3.2.4 Determinar o caminho entre dois locais 8
3.2.5 Determinar o caminho entre dois serviços 8
3.2.6 Determinar o caminho que permite percorrer uma sequência de serviços 9
3.3 Funcionalidades extra 9

4 Conclusão **10**

Bibliografia **11**
4.1 Referências Bibliográficas 11
4.2 Referências WWW 11

5 Anexos **12**
5.1 Elementos do Grupo 12
5.2 Codigo 13

Lista de Figuras

3.1 Esquema da base de conhecimento 6
3.2 Questão 1 7
3.3 Questão 2 7
3.4 Questão 3 8
3.5 Questão 4 8
3.6 Questão 5 8
3.7 Questão 6 9

5.1 André Pimenta 12
5.2 Cedric Pimenta 12
5.3 Daniel Santos 12
5.4 João Miguel 12
5.5 Milton Nunes 12

Capítulo 1

Introdução

1.1 Motivação e objectivos

Após os conhecimentos adquiridos na linguagem de programação em lógica PROLOG, este projecto surge com o objectivo de colocar em prática esses mesmos conhecimentos.
Assim, o nosso principal objectivo é desenvolver, através dos conhecimentos que possuímos, um sistema de representação de conhecimento e raciocínio com capacidade para caracterizar um universo de discurso. Através deste exemplo prático, tornar-se-á perceptível como o PROLOG pode ser utilizado na resolução de problemas complexos.
As nossas motivações na realização deste projecto resumem-se a provar a nós mesmos que conseguimos aplicar aquilo que estudamos a problemas reais e que somos capazes de criar sistemas utéis, ou seja, trata-se de gratificação pessoal.

1.2 Estrutura

O presente relatório é constituído por 5 capítulos. O primeiro, este, menciona a motivação e os objectivos do projecto, apresentando uma pequena introdução. No **Capítulo II**, apresentaremos os conhecimentos que achamos que o leitor deverá possuir para entender o projecto, enquanto no **Capítulo III** explicaremos tudo o que foi desenvolvido durante o nosso trabalho. Por fim, no **Capítulo IV** apresentaremos a conclusão e interpretação dos resultados obtidos; e o **Capítulo V** conterá toda a bibliografia consultada.

1.3 Introdução

Inserido na componente prática da unidade curricular Sistemas de Representação de Conhecimento e Raciocínio surge este projecto que se trata da primeira de várias partes de um enunciado. Este trabalho tem como principal objectivo motivar-nos para a utilização da linguagem de programação PROLOG, no âmbito da representação de conhecimento e da construção de mecanismos de raciocínio para a resolução de problemas.
Uma vez que uma forte componente desta unidade curricular é sobre o ensino de PROLOG, pretende-se, com este projecto, desenvolver um sistema de representação de conhecimento e raciocínio com capacidade para caracterizar um universo de discurso com o qual se possa abordar a temática da localização de serviços. Assim, este trabalho consiste em construirmos uma base de conhecimento sobre cidades, regiões e países, sendo que a cada cidade teremos associados diversos tipos de serviços. Para além desta informação, teremos, ainda disponível, informação sobre as ligações entre cidades (tipo de ligação, origem e destino, distância e valor

monetário e tempo).
Uma vez que os conhecimentos sobre PROLOG ainda são básicos, para a resolução deste projecto, apenas trabalhamos com listas, com os factos presentes na base de conhecimento e com os conhecimentos adquiridos anteriormente sobre a Teoria de Grafos.

Capítulo 2

Preliminares

Neste capítulo vão ser apresentados alguns conceitos fundamentais para a elaboração deste trabalho e algumas das ferramentas fundamentais para a elaboração do mesmo.

2.1 Estudos Anteriores

Para a realização deste trabalho foram necessários alguns conhecimentos anteriores sobre programação em lógica e, posteriormente, o uso da linguagem de programação PROLOG.
Este conhecimento foi adquirido ao longo das aulas teóricas (programação em lógica) e aulas teorico-praticas (PROLOG) de Sistemas de Representação de Conhecimento e Raciocínio.
Sobre estes conhecimentos, devemos destacar todos os conceitos que foram aprendidos tais como o que são predicados, o que são cláusulas, o que é a base de conhecimento, entre outros conceitos de programação em lógica que serão explicados ao longo deste relatório.
Após termos alguns conhecimentos de programação em lógica falta colocá-los em prática, e, é aqui, que entram os conhecimentos de PROLOG e da ferramenta SICSTus usada para compilar e interpretar o código desenvolvido nesta linguagem.

2.2 Programação em lógica e PROLOG

Apresentamos, então, alguns conceitos fundamentais para a realização deste trabalho dentro da programação em lógica:

- Factos: declaram coisas que são incondicionalmente verdadeiras;
- Regras: declaram coisas que podem ser ou não verdadeiras, dependendo da satisfação das condições dadas;
- Conectivos : a unica estruturação modular do conhecimento possivel é através destes;
- Teoremas: deduzidos através de regras e factos e servem para provar algo.

Estes são alguns exemplos dos conhecimentos base para perceber a programação em lógica. Após se ter estes conhecimentos, é necessário traduzir estes e aplicá-los na linguagem PROLOG. Deixamos, então, alguns exemplos importantes para a usar:

- se : representa-se por ”:- ”;
- e : representa-se por ”,”;
- ou : representa-se por ”;”;
- unificação (//) : sempre que a uma variável for atribuído um valor, esta passará a ser uma constante em que o seu valor não mudará mais ;

Com os conceitos apresentados neste capítulo e mais alguns importantes que serão explicados quando forem mencionados, foi possivel a elaboração deste trabalho prático.

Capítulo 3

Descrição do Trabalho e Análise de Resultados

Neste capítulo vamos apresentar os passos do desenvolvimento deste segmento, acompanhados pelas explicações dos mesmos. Faremos também uma breve análise dos resultados obtidos.

3.1 Base de conhecimento

Base de Conhecimento define bases de dados ou conhecimento acumulados sobre um determinado assunto.
Para a elaboração do nosso trabalho tornou-se importante definir uma base de conhecimento que possa responder aos pedidos do enunciado. Pedidos estes que devem conseguir responder a uma série de questões como a existência de serviços e a localização destes, que poderá ser descrita como o local, região ou pais onde se encontram os serviços.
Falta agora descrever na base de conhecimento, os caminhos entre cidades para que se possa responder ao resto do enunciado.
Os serviços serão descritos por **servico(Servico)**, as cidades por **cidade(Cidade)**, as regiões por **regiao(Região)**, os paises por **pais(Pais)** e os caminhos por **caminho(autoestrada, braga, porto, 50, 30, 17).**
Com todos estes conceitos apresentados até agora conseguimos representar toda a informação necessária para responder aos desafios que nos foram propostos e que serão apresentados mais á frente neste relatório.
Posto isto, apresentamos, de seguida, um esquema a explicar as ligações na base de conhecimento.

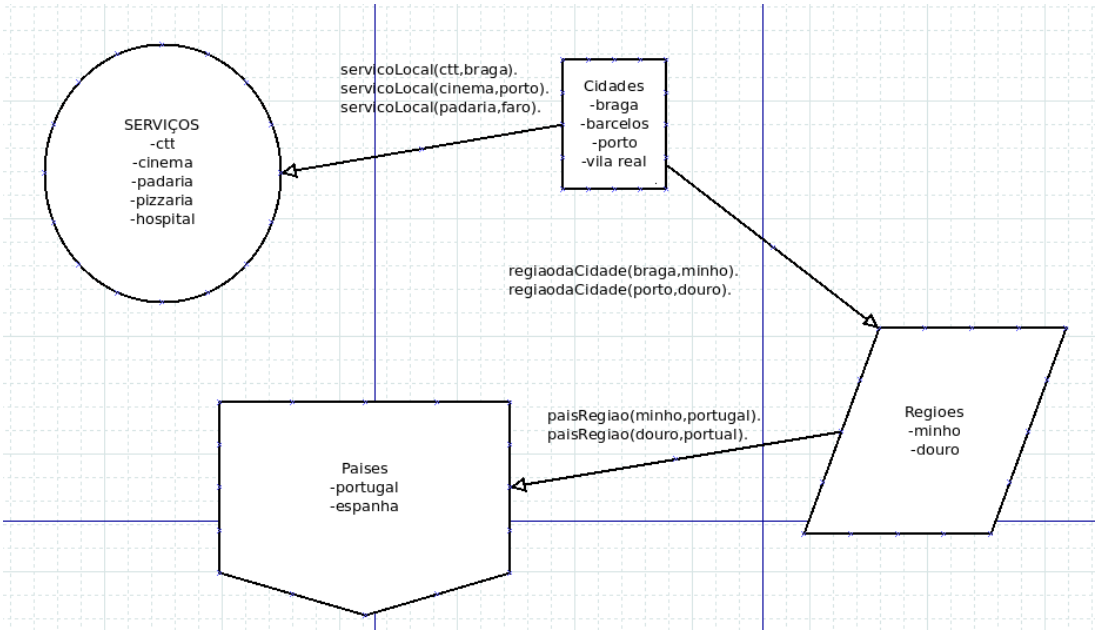

Figura 3.1: Esquema da base de conhecimento

3.2 Funcionalidades basicas

Após a apresentação da base de conhecimento do nosso trabalho torna-se possível desenvolver alguns teoremas e axiomas que consigam responder aos desafios apresentados para este trabalho.
Desta forma, apresentaremos e explicaremos a forma como resolvemos cada um destes desafios e quais os resultados obtidos pelas soluções por nós apresentadas.

3.2.1 Identificar os serviços existentes num determinado local, região ou país

Prentede-se, neste desafio, que seja possível apresentar todos os serviços localizados numa determinada cidade, região e País.
Para identificar todos os serviços de uma cidade basta-nos procurar todos os serviços que estejam ligados à cidade que se pretende pelo axioma *servicoLocal(servico,cidade)*, desta forma, obtemos todos os serviços disponíveis numa determinada cidade.

De modo a identificar todos os serviços numa determinada região precisamos de conhecer e identificar todas as cidades desta mesma região, pois os serviços apresentam-se associados a cidades e não a regiões, nem a países. Desta forma, recorrendo ao axioma *regiaoDaCidade(Cidade,Regiao)* conseguimos conhecer e identificar as cidades da região que pretendemos, e agora já com as cidades podemos obter, facilmente, os serviços associados às cidades que, por sua vez, se associam a uma região.

Falta-nos agora definir a forma de obter os serviços de um determinado país.
Seguindo a lógica utilizada anteriormente, através do axioma *paisRegiao(regiao,pais)* podemos conhecer e identificar as regiões de um determinado país, com as regiões obtemos as cidades e com estas obtemos os serviços que a elas se associam e que, por sua vez, estão associadas ao país do qual pretendemos conhecer os serviços que ele dispõe.

```
servicosCidade(Cidade, L):-
        findall(X,servicoLocal(X,Cidade),L).

servicosRegiaoAux([],[]).
servicosRegiaoAux([H|T],L):-
        servicosCidade(H,L1),
        servicosRegiaoAux(T,L2),
        concat(L1,L2,LR),remRep(LR,L).

servicosRegiao(Regiao,L):-
        findall(C,regiaoDaCidade(C,Regiao),Cidades),servicosRegiaoAux(Cidades,L).

servicosPaisAux([],[]).
servicosPaisAux([H|T],L) :-
        servicosRegiao(H,L1),
        servicosPaisAux(T,L2),
        concat(L1,L2,LR),remRep(LR,L).

servicosPais(Pais,L):-
        findall(P,paisRegiao(R,Pais),Regs),servicosPaisAux(Regs,L).
```

Figura 3.2: Questão 1

3.2.2 Identificar os locais onde esteja presente um determinado serviço ou conjunto de serviços

Para este desafio torna-se evidente que temos de chegar ás cidades através dos serviços que estas possuem. Usando, então, o axioma *servicoLocal(servico,cidade)* podemos chegar directamente aos locais onde se encontram os serviços.
Posto isto, trata-se de uma questão de *input*, ou fornecemos apenas um serviço e identificamos os locais associados a este ou recebemos uma lista e identificamos os locais associados a cada um dos serviços.

```
existeServico(Serv,L):-
        findall(C,servicoLocal(Serv,C),L).

existemServicos([],[]).
existemServicos([H|T],L):-
        existeServico(H,L1),
        existemServicos(T,L2),
        concat(L1,L2,LR),remRep(LR,L).
```

Figura 3.3: Questão 2

3.2.3 Identificar os serviços que não se podem encontrar numa determinada região

Para identificar os serviços que não se podem encontrar numa determinada região começamos por selecionar todos os serviços existentes através do facto *servico(Servico*. Após termos todos os serviços disponíveis falta identificar os que não pertencem à região que pretendemos conhecer.
Já referimos a forma como se obtêm os serviços associados a uma região, e é importante para este desafio identificar esses mesmos, uma vez que, tendo todos os serviços disponíveis de uma região, desenvolvemos um teorema que dadas duas listas resulte numa terceira com os elementos da primeira que não pertencem á segunda, ou seja, tendo a lista de todos os serviços e tendo a lista dos serviços de uma região constrói-se uma lista com os serviço disponibilizados que não se encontram na região.

```
todosServicos(L):-
        findall(S,servico(S),L),printLista(L).

servicosForaRegiao(Regiao,L):-
        todosServicos(Todos),
        servicosRegiao(Regiao,ServReg),
        difList(Todos,ServReg,L).

servicosForaCidade(Cidade,L):-
        todosServicos(Todos),
        servicosCidade(Cidade,ServCid),
        difList(Todos,ServCid,L).

servicosForaPais(Pais,L):-
        todosServicos(Todos),
        servicosPais(Pais,ServPais),
        difList(Todos,ServPais,L).
```

Figura 3.4: Questão 3

3.2.4 Determinar o caminho entre dois locais

No desafio de caminho entre locais desenvolvemos o seguinte raciocínio para o resolver: em primeiro lugar ,identificamos o destino final, de seguida encontramos o nodo adjacente e assim sucessivamente até encontrar o nodo inicial indicado. Caso não seja encontrado repete-se o processo para os outros nós adjacentes.
Para evitar que se entre em ciclos, que foi um dos nossos problemas iniciais, a lista final não permite a inserção de nodos repetidos.

```
/* 4-Determinar o caminho entre dois locais;
%caminho(tipo, origem, destino, kms, tempo, custo).*/

caminhoEntreLocais(A, Z, C) :-
    caminhoAux(A, [Z], C).

caminhoAux(A, [A | C1], [A | C1]).
caminhoAux(A, [Y | C1], C) :-
    adjacente(X, Y), not(member(X, C1)),
    caminhoAux(A, [X, Y | C1], C).

adjacente(X, Y) :-
    caminho(_,X,Y,_,_,_).
```

Figura 3.5: Questão 4

3.2.5 Determinar o caminho entre dois serviços

Para determinar o caminho entre dois serviços, é preciso, em primeiro lugar, determinar em que local se encontra cada um dos serviços em questão, através do axioma *servicoLocal(servico,cidade)*. Posto isto, utiliza-se o mesmo método do desafio anterior, que determina se existe algum caminho na base de conhecimento que ligue os dois locais.

```
%5-Determinar o caminho entre dois serviços;
/* caminho mais curto falta fazer */

caminhoEntreServicos(S1,S2,C) :-
    servicoLocal(S1,L1),
    servicoLocal(S2,L2),
    caminhoEntreLocais(L1,L2,C).
```

Figura 3.6: Questão 5

3.2.6 Determinar o caminho que permite percorrer uma sequência de serviços

Para resolver o problema de determinar o caminho entre uma sequência de serviços resolvemos percorrer a lista de serviços utilizando a resolução do desafio anterior. Desta forma, a lista vai sendo percorrida (primeiro, os primeiros dois elementos, de seguida, o segundo e o terceiro elemento e assim sucessivamente), as cidades a que os serviços estão associados vão sendo identificados e o respectivo caminho é determinado.

```
%6-Determinar o caminho que permite percorrer uma sequencia de servicos

caminhoSeqServicos([],C).
caminhoSeqServicos([H],C).
caminhoSeqServicos([H1,H2|T],C) :-
    caminhoEntreServicos(H1,H2,C),
    caminhoSeqServicos([H2|T],C).
```

Figura 3.7: Questão 6

3.3 Funcionalidades extra

Após a conclusão das funcionalidades mais básicas exigidas, e de forma a enriquecer o nosso trabalho, decidimos implementar uma funcionalidades adicionais.
Assim sendo, passaremos a explicar, por partes, estas novas funcionalidades. Em primeiro lugar, criámos os teoremas necessários para que pudessemos remover e adicionar serviços, localizações e caminhos. Após isto, decidimo-nos focar sobre os serviços e, assim sendo, temos funcionalidades que permitem mostrar todos os serviços de uma cidade (e a quantidade de serviços), o número total de cidades de uma região e o número total de regiões de um país; além disto, temos um teorema que nos mostra um local e o número de serviços associados a este, outro que nos indica o local com mais serviços e outro que nos dá o total de cidades e regiões que possui um determinado serviço.
Por fim, criámos um conjunto de invariantes que permite a não existência de informação replicada e impede que sejam eliminadas informações que estejam relacionadas com outro factos; algo que após alguma discussão entre o grupo, concordámos que seria extremamente essencial.

Capítulo 4

Conclusão

Antes de mais, gostaríamos de destacar a importância deste trabalho para a compreensão e aperfeiçoamento dos conhecimentos obtidos, até este momento, na disciplina de Sistemas de Representação de Conhecimento e Raciocinio. Para a resolução dos obstáculos que foram aparecendo no desenvolvimento do projecto, tivemos de procurar informação em variadas fontes, que não as aulas, o que nos deu conhecimentos mais gerais e vastos sobre o próprio PROLOG.
Uma vez finalizado o projecto, podemos concluir que os objectivos propostos foram alcançados com sucesso. Construímos uma base de conhecimento de acordo com o que era pretendido no enunciado e implementámos as funcionalidades necessárias. Para além disto, decidimos ainda implementar algumas características ou funcionalidades que achamos que seriam interessantes existir neste sistema.
Analisando os resultados obtidos, temos que todas as funcionalidades funcionam de acordo com as nossas expectativas e estão de acordo com a nossa base de conhecimento, isto é, as informações obtidas são muito úteis neste caso em particular.
Para além dos pequenos problemas que surgiram ao longo da realização do projecto e que foram ultrapassados com relativamente facilidade, apenas dois problemas são dignos de se mencionar: o primeiro foi se haveríamos de considerar um caminho como sendo de ida-e-volta ou apenas ida; acabamos por considerar somente ida, isto é, para representar um caminho entre Lisboa e Braga nas duas direcções, teriamos de representar o caminho entre Braga e Lisboa e entre Lisboa e Braga. O segundo problema não surgiu como problema de PROLOG mas sim como problema criativo uma vez que tivemos de imaginar novas funcionalidades úteis para implementar, além das exigidas, algo que foi resolvido após várias trocas de ideias.
Em suma, estamos muito satisfeitos com o trabalho que desenvolvemose sentimos que todo o projecto foi muito gratificante.

Bibliografia

4.1 Referências Bibliográficas

[Analide, 2011]ANALIDE, Cesar, NOVAIS, Paulo, NEVES, José,
"Sugestões para a Elaboração de Relatórios",
Relatório Técnico, Departamento de Informática, Universidade do Minho, Portugal, 2011.

[Bratko, xxxx]BRATKO, Ivan,
"PROLOG: Programming for Artificial Intelligence",
E.Kardelj University, J.Stefan Institute Yugoslavia, 1986.

4.2 Referências WWW

"http://pt.wikibooks.org/wiki/Prolog/Listas",
10 de Abril de 2011.

"http://pt.wikibooks.org/wiki/Prolog/Comando_Fail_ e _ a_ combinação_ Cut/Fail",
09 de Abril de 2011.

Capítulo 5

Anexos

5.1 Elementos do Grupo

Figura 5.1: André Pimenta

Figura 5.2: Cedric Pimenta

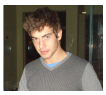

Figura 5.3: Daniel Santos

Figura 5.4: João Miguel

Figura 5.5: Milton Nunes

5.2 Codigo