Inteligência Artificial
Planeador de viagens e Analisador de frases

Bruno Silva(up201503818) e Diogo Lourenço(up201508214)

M.I.E.R.S.I. - Faculdade de Ciências da Universidade do Porto

1. Introdução

Neste trabalho foi proposto a implementação de dois programas em que cada um devia ser construido em duas linguagens diferentes. Em que uma das linguagens tinha de ser imperativa e a outra declarativa. Mas a pergunta que se coloca é, qual a diferença entre as duas? Para responder a esta pergunta é necessário explicar o que é cada uma.

Uma **linguagem imperativa** pode ser vista como um conjunto de ações, enunciados ou comandos que mudam o estado de um programa. Programas imperativos são uma sequência de comandos para o computador executar, onde é necessário que o programador detalhe ao promenor a maneira como quer que o programa se comporte. Já uma **linguagem declarativa** é escrita de forma mais simples, pois o programador necessita apenas de explicar o que quer fazer e não como deve ser feito.

A linguagem declarativa utilizada neste projeto foi *Prolog* e a linguagem imperativa foi *Java.*

Linguagem Imperativa:

Vantagens:

- Tendem a ser mais legíveis;
- São bastante eficientes, pois o programador tem que especificar o que quer fazer;
- Mais fácil para implementar funções;

Desvantagens:

- Como é necessário escrever mais código, o programador fica mais vulnerável a erros.

Exemplos de linguagens: *Java*, *C* e *C++*

Linguagens Declarativas

Vantagens:

- O tamanho do código é curto, isto porque não é necessário explicar todos os passos ao computador;
- O código é reciclável porque a maioria das funções vêm implementadas na própria linguagem;
- Devido à sua simplicidade, é uma linguagem de fácil aprendizagem (pelo menos para pessoas novas à programação). No entanto, programadores habituados a linguagens imperativas podem sentir algumas dificuldades na aprendizagem.

Desvantagens:

- Como o código é simplificado ao máximo, estes tipos de linguagens podem tornar o código ilegivel para programadores menos experientes;
- Como a maior parte das funções estão embutidas na própria linguagem torna-se complicado para o progamador criar novas funções.

Exemplos de linguagens: *Prolog* e *SQL*

2. Descrição dos Problemas

2.1- Problema 1: Planejador de Viagens

O primeiro problema é a criação de um plano de viagens que dado uma base de dados com a descrição dos voos permite resolver as seguintes tarefas:

1. Saber em que dias existe voo direto entre Place1 e Place2;

2. Quais são os voos disponíveis entre Place1 e Place2 num determinado dia da semana. Podendo existir escalas senão houver voo direto entre os dois locais;

3. Em que sequência deve o utilizador visitar as cidades C1, C2 e C3, partindo numa terça-feira da cidade S e voltando numa sexta-feira para a cidade de origem, de modo que o utilizador não precise de realizar mais do que um voo por dia.

2.2- Problema 2: Analisador de Frases

O segundo problema consistia na criação de uma plataforma capaz de analisar frases a nivel gramatical na lingua portuguesa, para isso é necessário:

1. A criação de uma base de dados com todas as palvras no dicionário português e a sua classificação, isto é, se é um verbo, nome próprio, etc. (Na base de dados deste projeto encontram-se apenas algumas palavras);

2. Divisão da frase em palavras para posterior análise individual;

3. Descrição das implementações em *Prolog*

3.1- Problema 1: Planejador de Viagens

Na implementação em *Prolog*, foram implementadas três funções principais.

A função flight(Place, Place2, Day, Flight_num, Dep_time, Arr_time), que permite verificar e dizer se existe um voo direto com o número Flight_num entre Place1 e Place2 no dia Day com as respetivas horas de saída e chegada.

A função route(Place1, Place2, Day, Route), que permite procurar e dizer exatamente a rota entre as duas cidades dadas no dia Day. Esta função funciona tanto para o caso de voos diretos como para o caso de voos que é preciso realizar transferência.

E por último a função plan(C1, C2, C3, S), que dadas três cidades C1, C2, C3 e a cidade de origem ou de partida S, calcula e imprime a ordem em que devemos visitar estas três cidades começando a viagem numa terça-feira e voltando para a cidade de origem numa sexta-feira, não realizando mais do que um voo por dia. Para além disso a função também imprime o número do voo, hora de partida e hora de chegada de cada voo.

Para que as principais funções do programa funcionem corretamente, foram criadas funções auxiliares que resolvem problemas específicos, como por exemplo, a função transfer, que verifica se o tempo de transferência entre dois voos é maior ou igual a 40 minutos.

3.2- Problema 2: Analisador de Frases

Para analisar uma frase usando o programa é necessário chamar a função frase que em seguida chama duas outras funções. Uma para frases no plural e outra para frases no singular. A partir daqui o programa vai dividir a frase em palavras e vai analisá-las segundo as regras gramaticais portuguesas. Irá verificar se o primeiro elemento da frase é um artigo e se sim qual o sexo dele, depois verifica se a palavra que vem depois é um nome e do mesmo sexo, se o resultado der negativo retorna negativo caso contrário continua a dividir a frase até encontrar um defeito, caso não encontre retorna verdadeiro e o caminho até chegar à solução.

Para isto ser possível é necessário criar uma base de dados com todas as palavras utilizadas neste projeto e a sua descrição, isto é, qual o sexo e a sua caracteristica (se é um verbo, nome, artigo, singular/plurar, etc.).

4. Descrição das implementações em *Java*

4.1- Problema 2: Analisador de Frases

A implementação em *Java* necessita de mais funções e de mais linhas de código comparado com o problema resolvido em *Prolog*. Outra grande diferença é o facto de ser obrigatório criar recursividade para a correta análise das frases, ao contrário daquilo que acontecia em *Prolog*, pois essa tarefa é feita automaticamente pela própria linguagem sem ser necessária a intervenção do programador.

Existem três classes para a resolução deste exercício, sendo elas a classe main, que tem como nome Phrases e serve apenas para arrancar o programar e receber a frase a ser analizada e chamar as outras classes que vão resolver o problema.

Outra das classes, e a mais importante é a classe Phrase. É nesta classe que irá ser guardado o "sexo" da frase e se é plurar ou singular. Também vai verificar se as palavras seguintes seguem as regras e divide a frase em palavras, guarda o tipo de cada palavra, ou seja, se é um artigo, verbo, etc num array de strings para depois apresentar o resultado caso a frase esteja correta.

Já a classe BaseDados é a base de dados do programa. Mas para além disso vai analisar o tipo de cada palavra e decidir qual é o tipo de palavra que deve surgir depois. Para isto foi necessário criar uma função recursiva que vai analisar palavra a palavra. Toda esta função retorna valores de verdade ou falso.

5. Comentários finais

Na implementação dos problemas acima descritos na linguagem *Prolog*, como se tratava de uma linguagem diferente e nova para nós, no principio foi difícil conseguir perceber como funcionava.

Na implementação na linguagem *Java* foi um pouco difícil pensar numa solução para a implementação dos problemas mesmo estando já familiarizado com a linguagem. Isto porque foi preciso a utilização de funções auxiliares para conseguir resolver o problema principal. Sendo assim preciso muitas mais linhas de código comparado com a linguagem *Prolog*.

6. Conclusão

Na realização deste projeto ficou bem claro as diferenças entre as linguagens imperativas e declarativas, cada uma delas com as suas vantagens e desvantagens.

Mas neste projeto a linguagem ideal a ser utilizada é *Prolog*, pois é uma linguagem que requer maior atenção no raciocínio, mas que acaba por ser menos trabalhosa do que *Java*.

7. Bibliografia

Brakto, I. (1986). Prolog Programming for Artificial Intelligence. Wokingham: Addison-Wesley Publishing Company, Inc.

CIS, C. (28 de Setembro de 1994). Cornell CIS Computer Science. Obtido de http://www.cs.cornell.edu/info/Projects/Nuprl/cs611/fall94notes/cn2/subsection3_1_2.html

Peter, R. S. (2010). Artificiall Intelligence, A Modern Approach Third Edition. New Jersey: Pearson Education Inc.

Wikipédia. (Abril de 2011). Wikipédia. Obtido de Programação Declarativa: https://pt.wikipedia.org/wiki/Programação_declarativa

Wikipédia. (Janeiro de 2011). Wikipédia. Obtido de Programação Imperativa: https://pt.wikipedia.org/wiki/Programação_imperativa

Obtido de Programação Imperativa vs Programação Declarativa: http://informacaocomdiversao.blogspot.pt/2006/01/programao-imperativa-versus-programao.html