Inteligência Artificial – Trabalho 3
Relatório Sobre Programação Funcional vs Programação Declarativa
Miguel Gomes(201505151), Pedro Antunes(201507254) e Rafael Novais(201508010)

1 – Introdução

Um paradigma de programação é um meio de se classificar as linguagens de programação baseado nas suas funcionalidades. Fornece e determina a visão que o programador possui sobre a estruturação e execução do programa. Vamos abordar o paradigma funcional e o paradigma imperativo.

Uma linguagem de programação é um método padronizado para comunicar instruções para um computador, respeitando regras semânticas e sintáticas. Existem vários tipos de linguagens de programação sendo que vamos apenas abordar linguagens funcionais e linguagens declarativas.

1.1 Java - Programação Imperativa

As linguagens imperativas são orientadas a ações, onde a computação é vista como uma sequência de instruções que manipulam valores de variáveis. O fundamento da programação imperativa é o conceito de Máquina de Turing, que consiste numa abstração matemática que corresponde ao conjunto de funções computáveis.

Apesar da linguagem Java ser uma linguagem orientada a objetos, tem por base princípios Imperativos utilizando vários padrões de desenho que se identificam com os princípios da programação imperativa.

1.2 Prolog - Programação em Lógica (Funcional)

Programação funcional é um paradigma da programação que se baseia no conceito de função. Tem origem no Lambda calculus(λ-calculus) e a computação é executada como a avaliação de funções matemáticas. A programação é feita com expressões ou declarações. Pode-se pensar na programação funcional como simplesmente avaliação de expressões.

2 - Flight Planner

O primeiro exercício tem como objetivo implementar um planeador de viagens. As viagens podem ser feitas com voos diretos ou então com voos indiretos. O tempo de transferência entre vôos

(no caso de ser um voo com escala) é de 40 minutos. O objetivo final é conseguir responder a 3 perguntas:

- Em que dias da semana existe voo direto de Place1 para Place2 ?
- Quais são os voos disponíveis de Place1 para Place2 ?
- Partindo de S, quero visitar C1, C2, C3 e voltar a S. Saindo na Tuesday e voltando Friday e visitando uma cidade por dia, qual é a ordem de vista de cidades?
-

2.1 - Flight Planner - PROLOG

Para implementar este exercício em Prolog, começamos por implementar uma relação flight que verifica na timetable se existe voo. Implementamos também uma relação route que verifica voos diretos e indiretos. Usamos uma restrição para verificar se o tempo de transferência nos voos indirectos era suficiente.

Para a resposta a última pergunta foi implementado uma função permutações com o objetivo testar as várias combinações. Combinado isso com a função flight que verifica se existem vôos e obtemos a resposta.

2.2 - Flight Planner - JAVA

Em java foram utilizadas várias classes para tornar mais fácil seguir o fluxo de dados por métodos (das classes) mais explícito.

A classe Base_dados representa a base de dados de todos os voos registados.

Estes dados ficam guardados em várias listas ligadas, cada uma representando o voo, hora de partida, hora de chegada, número de voo e os dias desse voo.

A classe ‘Route’ só contém a estrutura de um voo de uma cidade para a outra, sob forma de rota, e o método para imprimir essa rota com todos os dados como a origem, o destino, hora de partida, hora de chegada, número e dia do voo. Já os voos diretos entre duas cidades estão todos guardados na classe ‘Flight’.

A classe chamada ‘Timetable’ é responsável por todo o cálculo intermédio assim como as pesquisas necessárias para obter a rota requerida pelo utilizador. Para realização das pesquisas de rotas foi usado a pesquisa em largura, pois é o que encontra em melhor tempo a rota necessária. Também foi usado uma matriz de adjacências para simplificar as ligações entre as cidades e a pesquisa em largura, para tal criou-se um índice para cada cidade e unicamente para essa cidade. Esse índice é guardado em

duas HashMap’s ((Cidade, Num.) e (Num., Cidade)) e para marcação das cidades já visitadas usou-se um array de booleanos.

A classe main ‘Routes’ serve unicamente para a organização pela interação com o utilizador.

3 - Grammar Checker

O segundo exercício tem como objetivo implementar um verificador de gramática que verifique uma frase em português. As principais regras aplicadas a este verificador foram:

- Uma frase normalmente é composta por uma parte nominal e uma parte verbal.
- A parte nominal e a parte verbal têm que ter o mesmo modo de flexão, seja ele singular ou plural.
- Os artigos devem estar associados aos substantivos sendo que ambos têm que ser do mesmo gênero e modo de flexão.
- Uma parte nominal pode ter preposição.
- Uma parte nominal pode ter contrações e as mesmas tem que verificar gênero.
- O programa apenas tem uma pequena parte do dicionário português que consegue verificar as frases dadas para teste no enunciado.

3.1- Grammar Checker - PROLOG

A implementação em Prolog do verificador de gramática usa como base Definite Clause Grammars (DCG). As DCG’s são uma maneira conveniente de representar relações gramaticais para várias aplicações de parsing. Pode ser usadas para trabalho de linguagem natural e para criar comandos formais e linguagens de programação.

Utilizando as regras definidas como base, podemos rapidamente criar cláusulas que verifiquem cada uma das propriedades.

3.2- Grammar Checker - JAVA

A implementação em Java do verificador é baseada em métodos simples que diferenciam uma frase (excluem possibilidades) em função do gênero e do número de algumas palavras. A forma como a frase é decomposta em estados gramaticais foi implementada em função das DCG, utilizadas na programação em lógica.

4 - Conclusão

Ao comparar os dois paradigmas de programação denota-se que a abordagem da programação funcional analisa os problemas em função dos meios mais práticos e suscetíveis para encontrar uma solução desejada, e por isso, também eleva o nível de desempenho.

A abordagem imperativa tem uma boa performance quando são problemas de mudanças de estado ou testes de ordens/algoritmos.

Com as soluções desenvolvidas neste trabalho chegamos à conclusão de que existem certos problemas que são melhor resolvidos usando uma programação funcional.

5 - Referências

https://pt.wikipedia.org/wiki/M%C3%A1quina_de_Turing

https://en.wikipedia.org/wiki/Java_(programming_language)#Practices

https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_declarativa

https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_imperativa

https://pt.linkedin.com/pulse/programa%C3%A7%C3%A3o-imperativa-vs-funcional-celso-junior

https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_funcional

https://en.wikipedia.org/wiki/Lambda_calculus