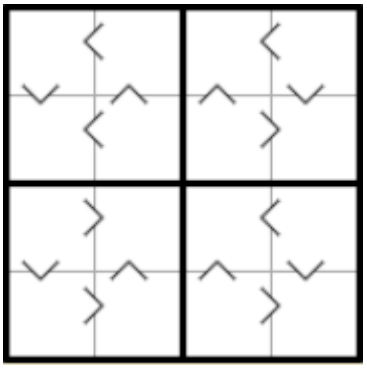
Matheus Paulon Novais - 20203078

Trabalho III: Prolog

1. Descrição

Entre as três opções disponíveis para o trabalho foi escolhida a do *Vergleichssudoku*, também conhecido como *Inequality Sudoku*[Sudoku comparativo) ou *Greater Than Sudoku*(Sudoku maior que, nome dado pelos símbolos presentes no tabuleiro). Esse jogo é uma vertente muito parecida com o sudoku comum pelos espaçamentos do tabuleiro, podendo ser 4x4(com 4 casas de 2x2), 6x6(com quatro casas de 3x2) e 9x9(com quatro casas de 3x3).

O jogo inicia com um tabuleiro de sudoku zerado, e a forma de resolvê-lo é identificar os símbolos de maior-que(“>”) juntamente com o de menor-que(“<”) presentes no tabuleiro, causando uma comparação direta entre as casas. Assim como aprendemos no começo da nossa caminhada no mundo matemático, a boca aberta do símbolo fica para o lado que é maior que o outro, devemos então completar os números que vão de 1 à *N* (*N* = tamanho de células em cada casa) em uma casa do tabuleiro a partir dessas noções.



Exemplo número 1: Vergleichssudoku

2. Programação de restrições

Para facilitar o desenvolvimento desse trabalho, e para ajudar o tempo de execução dado em cima do problema, foi sugerido pelo professor o uso de restrições dado pela biblioteca de CLP(Constraints Logic Programming). Essas restrições agem de forma parecida com os predicados já usados em Prolog, facilitando assim um pouco da compreensão em cima delas.

Mas diferentemente dos predicados, as restrições podem atrasar as validações e ainda testar de forma unificada todas as limitações dadas em cima da constatação dada.

Dentro desses dessa biblioteca a usada dentro do trabalho foi a CLP(FD) que é baseada em cima de *integers*, que é amplamente utilizada dentro do contexto de *problem solving*. No programa, primeiro fazemos todas as restrições e depois deixamos com que o programa procure as opções possíveis e ache a que se encaixe.

A partir disso foram desenvolvidos predicados e restrições para manter primeiramente a noção de um bloco de sudoku e depois para seguir a regra da vertente escolhida do jogo.

A não necessidade de programar o backtracking deste trabalho foi provavelmente a maior vantagem encontrada do Prolog, mas entender como condicionar de forma correta e como os predicados e restrições interagem entre si dentro e a menor liberdade quanto a funções imprimiu uma desvantagem durante o processo de desenvolvimento.

3. Entrada e Saída de dados

A entrada é dada por uma matriz com listas que descrevem os quatros lados de cada casa, com quatro valores para indicar se contém algum sinal, e caso tenha, qual sinal é.

Cada lista segue o exemplo de (esquerda, direita, cima, baixo), e os valores possíveis para cada um deles são:

3 -> Não contém sinal na parede indicada

2 -> Símbolo de maior que “>”

1 -> Símbolo de menor que “<”

Exemplo segundo a imagem do primeiro tópico:

[3, 1, 3, 2]	[2, 3, 3, 1]	[3, 1, 3, 1]	[2, 3, 3, 2]
[3, 1, 1, 3]	[2, 3, 2, 3]	[3, 2, 2, 3]	[1, 3, 1, 3]
[3, 2, 3, 2]	[1, 3, 3, 1]	[3, 1, 3, 1]	[2, 3, 3, 2]
[3, 2 , 1, 3]	[1, 3, 2, 3]	[3, 2, 2, 3]	[1, 3, 1, 3]

O resultado é dado diretamente na saída do console após a execução do programa

4. Paradigma Funcional e Lógico

Dentro do paradigma funcional a maior facilidade e vantagem possivel é a familiaridade com esse tipo de paradigma, saber como trabalhar uma ideia facilita o processo, a popularidade das linguagens em paradigma lógico facilita também para tirar dúvidas, seja pessoalmente ou pela internet, mas tem como severa desvantagem a dificuldade de resolver problemas lógicos, onde você precisa dar mais voltas para chegar em um objetivo.

No paradigma lógico temos a facilidade de após, a compreensão de como ele funciona, ter ferramentas especializadas em resolver problemas lógicos, e criar eles de forma parecidas como são pensados pela nossa cabeça, a maior desvantagem foi com certeza a dificuldade de se familiarizar com as regras criadas e o funcionamento dele em si.

5. Dificuldades

A maior dificuldade encontrada nesse último trabalho relacionado a sudoku, foi provavelmente o tempo resultante do semestre encurtado. A falta de tempo para poder se dedicar às listas e durante as aulas junto com o cansaço do semestre corrido levou a dificuldade de concentração e devoção ao fim da matéria.

Dentro da linguagem Prolog, a maior dificuldade foi compreender como utilizar a biblioteca CLP e como criar regras e predicados que conseguissem influenciar a matriz de entrada e aplicá-los na matriz de saída. Entender como é

a matriz de entrada foi a única dificuldade que teve fácil resolução, mas assim como nos outros trabalhos, tratar com matriz de listas atrapalhou o andamento do projeto em algumas oportunidades.