Universidade Federal de Santa Catarina

Relatório - Trabalho III

INE5416 - Paradigmas de Programação

André William Régis (19200411)
Julien Hervot de Mattos Vaz (19202276)
Paola de Oliveira Abel (17100531)

Florianópolis
2022

Sumário

1 Análise do problema

O puzzle Vergleichssudoku a ser resolvido pelo programa é uma variação do conhecido Sudoku, com as mesmas regras de não repetição de elementos em linhas, colunas e regiões. com o acréscimo de uma nova regra em que os elementos devem respeitar as condições impostas pelos comparadores entre eles, os comparadores sendo “maior que” e “menor que”. Nota-se que elementos de regiões diferentes não são comparados entre si.

2 Programação de restrições

Pelo o que entendemos da programação de restrições, ela é usada para declarar e delimitar o domínio ao qual as variáveis das soluções possíveis pertencem. Usada em conjunto com a propagação de restrições, é possível calcular as soluções baseando-se na consistência dos intervalos possíveis de valores ao invés de um valor unitário. Ela foi usada no trabalho para otimizar o cálculo da solução, utilizando-se a biblioteca `clpfd`.

As principais ferramentas da biblioteca que foram utilizadas foram o operador `ins`, que permite declarar explicitamente os domínios (dentro do conjunto dos números inteiros) de todas as variáveis de uma lista, o predicado `all_distinct`, que declara que cada elemento de uma lista tem um valor único dentro da lista considerando o intervalo de valores possíveis,e os operadores #> e #<, que são equivalentes aos operadores “maior que” e “menor que”, mas que são usados para operar sobre variáveis da biblioteca, que por padrão têm um intervalo de valores possíveis, e portanto os operadores conseguem restringir esse domínio para que ele respeite as operações declaradas.

3 Vantagens e desvantagens

3.1 Vantagens do paradigma lógico em relação ao paradigma funcional

A maior vantagem do paradigma lógico para a implementação de um resolvedor de quebra-cabeças é que não é necessário declarar a sequência de passos para a execução da solução, mas sim declarar as propriedades do quebra-cabeça que irão permitir a sua resolução com base em operações lógicas. Dessa forma, quando se entende a forma correta de declarar o quebra-cabeça dentro do programa, a linguagem consegue resolvê-lo graças à utilização do backtracking nativo de seu funcionamento.

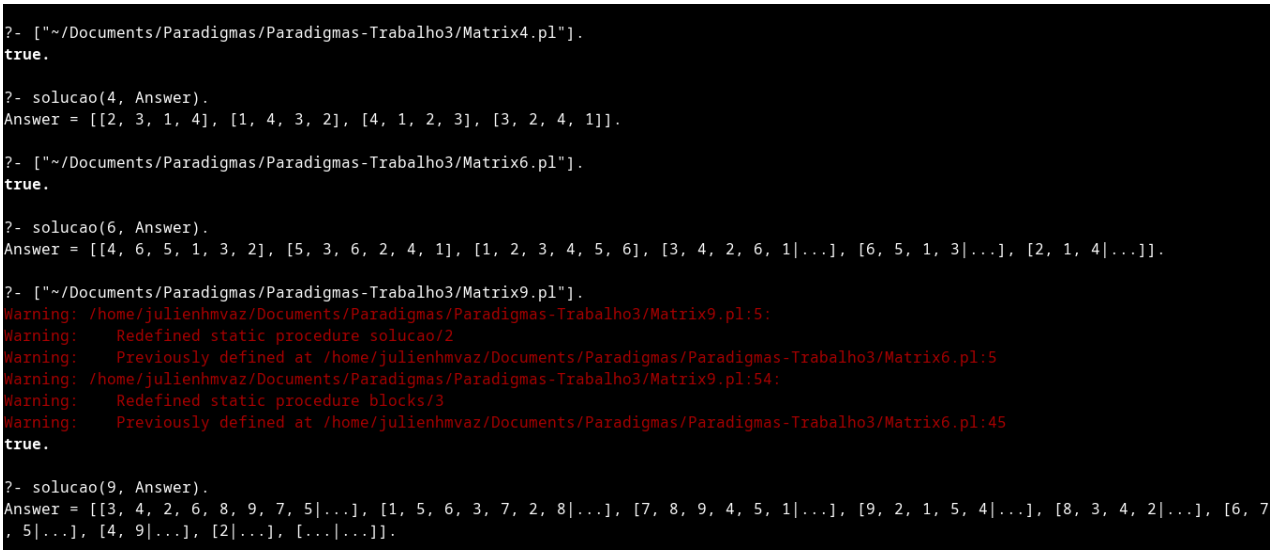
3.2 Desvantagens do paradigma lógico em relação ao paradigma funcional

Uma desvantagem é que precisamos declarar explicitamente cada variável da matriz, para que os comparadores acessem os índices corretos dos elementos a se comparar, fazendo com que essa parte tome muitas linhas de código do programa. No paradigma funcional, achamos formas de operar sobre os elementos do quebra-cabeça de forma puramente aritmética.

4 Entrada do programa

Para modificar a entrada do programa, o usuário deverá alterar os comparadores do quebra-cabeça, um por um, nos respectivos arquivos para cada ordem de matriz.

O resultado mostrado pelo terminal é incompleto, como mostra a imagem abaixo:



Porém, o mesmo é mostrado na tela corretamente pelo uso de um plugin de render do SWISH (versão de browser do ambiente do SWI-Prolog), utilizável pelo comando “`:- use_rendering(table)`”, o qual organiza de forma gráfica a matriz resposta, respeitando a estrutura de linhas e colunas.

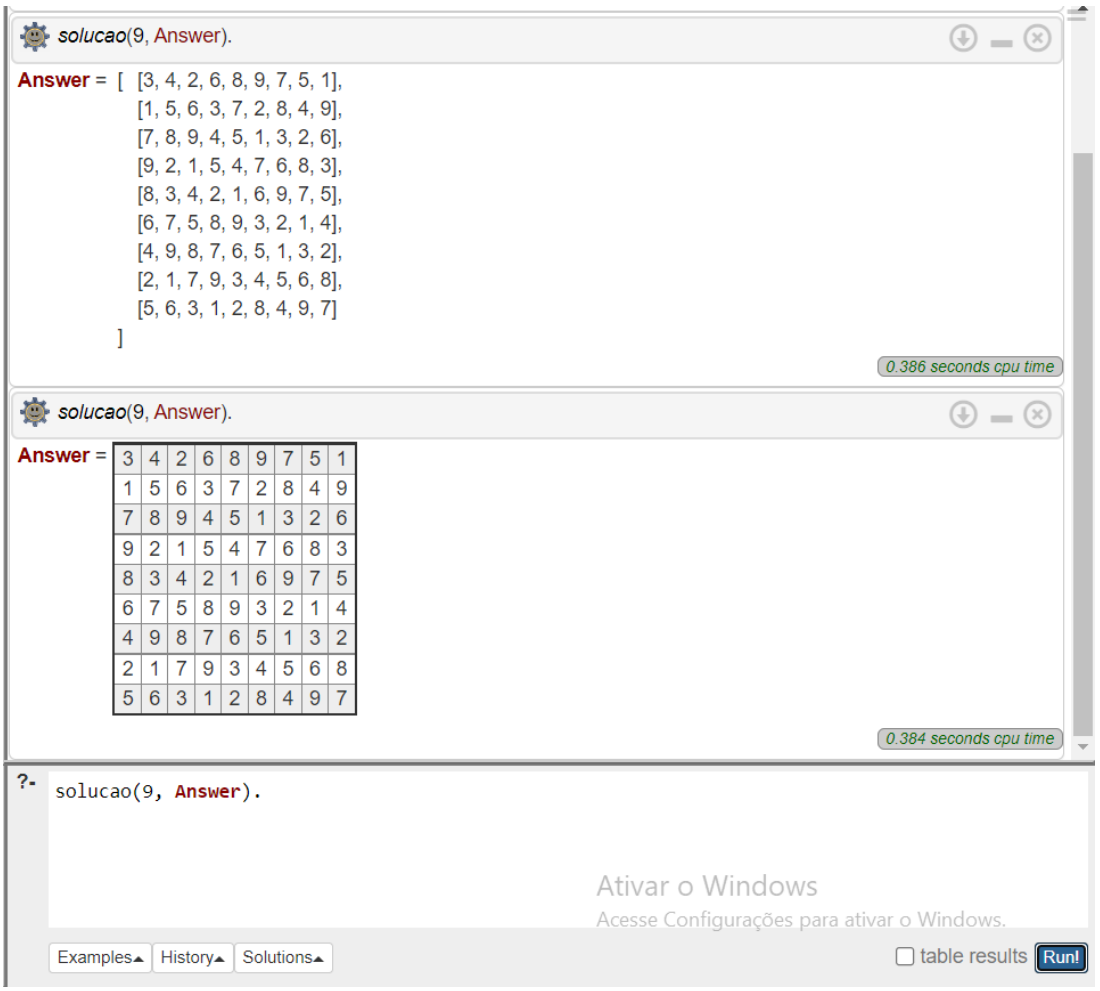


Figura 1: Comparação da formatação do resultado sem e com a utilização do *plugin* de *render* do SWISH.

5 Organização do grupo

Reunimo-nos para revisar as estratégias do trabalho anterior e reestruturar os arquivos e as funções a serem implementadas de antemão. Obtivemos sucesso ao estudar a implementação de um resolvedor de sudoku mostrada em sala de aula e a biblioteca `clpfd`. A princípio as soluções dos três tamanhos de matrizes foram implementadas no mesmo arquivo, mas depois percebemos que era necessário separá-las em um arquivo para cada tamanho. Após a implementação, as soluções para cada tamanho foram testadas separadamente. O trabalho foi concluído na presença de todos por chamada remota.

6 Dificuldades Encontradas

Só conseguimos obter sucesso na implementação da solução após implementações de solução de problemas mostradas em sala de aula, já que tivemos dificuldade em adaptar o pensamento de um paradigma imperativo, usado nos trabalhos anteriores e na maior parte das linguagens de programação populares, para um paradigma declarativo.

7 Código Fonte

A implementação do código utilizado para elaborar este relatório pode ser encontrado em: https://github.com/AndreKuru/Paradigmas-Trabalho3