ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

CURSO DE GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

PCS3556 – Lógica Computacional

CAIO VINICIUS SOARES AMARAL - NUSP: 10706083

THALES AUGUSTO SOUTO RODRIGUEZ - NUSP: 10706110

GUSTAVO PRIETO – NUSP 4581945

Prof. Dr. Ricardo Rocha

Exercício de Programação 1

São Paulo

Introdução

O objetivo deste exercício de programação é implementar o algoritmo de fecho transitivo-reflexivo. Ele é um algoritmo bastante relevante na área de relações binárias, que representam relações ente dois elementos na forma de conjuntos de pares ordenados. Nesse caso, a relação binária r é um subconjunto do produto cartesiano de um conjunto A dele com ele mesmo, ou seja $r \subseteq AxA$.

Uma relação binária transitiva é aquela em que se um elemento forma tem relação com o segundo e este tem relação com um terceiro, então o primeiro elemento tem relação com o terceiro. Já uma relação reflexiva é aquela em que todo elemento está relacionado a ele mesmo. O objetivo do algoritmo de fecho transitivo-reflexivo é completar uma relação binária com o mínimo possível de pares ordenados para que ela fique reflexiva e transitiva.

O repositório do projeto pode ser encontrado em: https://github.com/master0022/EP1-Logica-Computacional

Algoritmo

O código desenvolvido apresenta um módulo chamado **ClosureFunctions**, que apresenta duas funções, cada uma representando uma operação diferente. Ambas as funções recebem listas de listas representando uma relação R.

(ex: [[1,2],[3,4],[1,3]]), e retornam uma lista modificada que inclui os elementos do input e novas relações.

Função de fecho reflexivo (**reflexive(relations,index\\0)**)

Esta função faz os seguintes passos:

- analisa o elemento na posição (index).
  - Caso index esteja fora da lista, encerra a execução e retorna relations.
- Analisa a relação [a,b] do elemento extraído anteriormente
  - Caso [a,a] não esteja em relations, executa **reflexive([a,a]+relations,index+1)**. Como [a,a] é adicionado no começo da lista, isso equivale a chamar a função com o mesmo index.
  - O mesmo é feito para o "b".
  - Caso [a,a] e [b,b] já estejam presentes, executar **reflexive(relations,index+1)**. Isto equivale a chamar reflexive para o próximo elemento da lista.

Função de fecho transitivo **transitive(relations,index\\0)**

Esta função faz os seguintes passos:

1. Analisa o elemento na posição index.
   a. Caso index esteja fora de relations, encerrar a execução e retornar relations.
2. Analisa [a,b] do elemento.
3. Procura-se o primeiro elemento [b,c] que exista na lista, tal que [a,c] não esteja já presente em relations.
   a. Caso [b,c] não exista sob esta condição, executa-se **transitive(relations,index+1)**. Isto significa que já foram acrescentados [a,c] para todo c que exista algum [b,c] na lista.
   b. Caso encontrado um [b,c], executamos **transitive(relations++[a,c],index)**. Como já verificamos se [a,c] existia, garantimos não duplicar elementos. Também, como [a,c] é acrescentado ao final da lista, eventualmente a função vai processar [a,c] e acrescentar elementos [a,d], tal que exista [c,d], e assim em diante.

Casos de teste

Para rodar os casos de teste criados, basta entrar na pasta do projeto elixir e executar

mix test

Ou, caso seja desejado usar as funções com casos novos, elas estão disponíveis na pasta /lib, no arquivo ClosureFunctions.

Os casos de teste gerados foram os seguintes, e podem ser encontrados em /test/closure_functions_test.exs:

Fecho transitivo:

Exemplo 1:

Entrada: [[1,2], [2,3], [3,4]]

Saída esperada: [[1,2], [2,3], [3,4], [1,3], [1,4], [2,4]]

Exemplo 2:

Entrada: [[1,2], [2,2], [2,3], [3,4], [3,4], [4,8], [10,24]]

Saída esperada: [[1,2], [2,2], [2,3], [3,4], [3,4], [4,8], [10,24],
[1,3], [1,4], [1,8], [2,4], [2,8], [3,8]]

Fecho reflexivo:

Exemplo 1:

Entrada: [[1,6], [1,2], [2,3], [4,4], [5,7]]

Saída esperada: [[1,6], [1,2], [2,3], [4,4], [5,7], [1,1], [2,2], [3,3], [5,5], [6,6], [7,7]]