Reflexive and Transitive closure Algorithm - an implementation in Elixir

Luciana da Costa Marques
Polytechnic School of the University of So Paulo
e-mail: luciana.marques@usp.br

1. Introduction

This paper presents an implementation of the Reflexive and Transitive Algorithm in the Elixir Language.

2. Formal Definitions

This section contains the formal definitions of Logic and Mathematics used to develop this work.

2.1. Binary Relation

A binary relation of type R Ł A A can be represented by a graph, in which the elements of A are the nodes of this graph. In this case, A is defined as A = 1,2,3,4,5.

2.2. Reflexive and Transitive closure

A reflexive and transitive closure Rr is defined as follows: Rr = R ł a,a:aA, with R being a binary relation.

2.3. Graphs and Maps

As seen in books on Algorithms, a graph can be represented in a list of tuples and adjacency matrices (and variations on both). It is simpler to manipulate Maps with Elixir, so this implementation used the language's Map module to manipulate this.

3. Algorithm Description

The Algorithm developed consists in taking an input list of tuples and output another list of tuples accordingly with the definition in the previous section.

1) Create adjacent matrix based on the input
2) Create "Result" list of tuples, initially empty
3) Add main diagonal in adjacent matrix and add correspondent tuples
4) Search for new adjacent elements using DFS algorithm, complete the matrix with them and add them to "Result"
5) Return "Result"

4. Implementation

The algorithm implementation consisted mainly in the steps from the previous section. The original code generated for this project can be seen at https://github.com/LucianaMarques/PCS3556-ComputationalLogic.git in the "transitivereflexive" folder.

The following functions are defined:

- "closure algorithm()": something linke a main function. It calls
- "creates graph()": API user can add the tuples representing the edges here, limited to 5x5 maximum default size (in this case, set A is 1,2,3,4,5).
- "find nodes()": responsible for looking for all the possible nodes. This function calls all the other functions not listed to get the final result.

5. Tests

Unfortunately, the tests didn't work as expected.

6. Conclusion

The conclusion goes here.

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[2] Cormen *Introduction to Algorithms*

[3] StackExchange "Reflexive Transitive Closure", Visited in February 14, 2019, available in: link