A Refactoring Discovering Tool based on Graph Transformation

Our tool prototype implements a method, based on graph transformation to discover refactoring sequences between two versions of a software system. In case a refactoring sequence exists, our tool can also help reveal the functional equivalence between the two versions of the system, at least, as far as refactorings can assure behaviour preservation.

Searching Refactorings

Does a refactoring sequence exist between two versions of a system?

A Graph Transformation Approach

Based on the refactoring formalization works by Ron Meis, we have defined a Java metamodel type graph to represent: Java program as Java program graphs; refactorings as graph transformation rules.

Modeling the Problem

we model the sequence search as a state space search problem

source system start state

refactoring operations state transitions

target system goal state

sequence existence? goal reachability?

refactoring sequence start-to-goal-path

and we apply a depth-first search algorithm

Results

- Approach validated with a toy example
  - Search rules implemented so far:
    - pullUpMethod
    - renameMethod
    - useSuperType
    - removeMethod
    - removeClass
  - Approach support renamings and multiple refactorings applied over the same element
  - The current implementation is a proof of concept that offers very promising results

Future Work

- improve visualization of results: to automatically generate the state space graph
- Extend the set of supported refactorings, implementing more refactoring searching rules
- Measure scalability over industrial-size systems
- Test other graph transformation tools as PROGRES

Open Issues

- Our algorithm is only partially correct, thus it could not terminate in certain cases. Would we be able to assure a finite state space?
- It is not possible to represent every refactoring with AGG. Can we find a more suitable graph transformation tool?
- We address documenting software evolution and revealing functional equivalence between two system versions. To which real scenarios can our results be applied?