Logic-based, Rule-based and Constraint Programming

Summer 2013 (Pro-)Seminar Topics

contact: amira.zaki@uni-ulm.de

Sometimes it is easier to solve a problem in one direction rather the other, like it is easier to disarrange Rubik's cube than to arrange it. Pairs of inverses are quite common in software, e.g., insert/delete operations on data structures, encryption/decryption, compression/decompression, rollback in transactions, etc.. Automatic program inversion could potentially allow programmers to write only one of each inverse pair, halving the time required to write and maintain such code and eliminating bugs due to inconsistencies between the inverses. However the production of the inverse of programs is not a very straight forward task. This seminar aims to present an overview of the different approaches implemented to attempt program inversion.

1. **Program Inversion for LISP** – [1, 4]

   Korf [4] presents a method for generating the inverse function of a program written in a minimal subset of LISP. Each primitive program construct becomes inverted into a separate rule, moreover additional rules are added to handle recursive and auxiliary function calls. Glück [1] refines this method using an operational semantics, to construct the first automatic program inverter described in the literature. The ket features of the method are the simultaneous inversion of an injective system of functions, an equation-oriented transformation, and the use of postcondition inference heuristics. We are interested to present the ideas presented in their works.

2. **Inversion using Value Search Graphs** – [3]

   [3] discusses a new method for generating the inverse of imperative programs using a value search graph and a route graph. These formalisms directly express the amount of state needed to enable the inverse, thereby enabling algorithms that can try to minimize storage space. The paper focuses on handling arbitrary control flows and basic operations of imperative programs. It operates by building a value search graph that represents recoverability relationships between variable values. The problem of recovering previous values is transformed into a graph search one. Forward and reverse code is generated according to the search results. The authors have also worked on imperative programs with loops and a means to inverse them by generating an inverse loop body [2].

3. **Logical Inversion of Imperative Computation** – [5]

   An imperative program can be modeled as a logical relation which describes its computational behavior. Then using logical inference as a tool, it is possible to generate the inverse of such a program. Program relations then denote both forward and backward computations, and the direction of the computation depends upon the instantiation pattern of the relation arguments. This work [5] presents the inversion of a number of nontrivial imperative computations, inverted with minimal logic programming tools.

References

[1] Robert Glück and Masahiko Kawabe. Revisiting an automatic program inverter for lisp. *SIGPLAN Notices*, 40(5):8–17, 2005.

[2] Cong Hou, Daniel Quinlan, David Jefferson, Richard Fujimoto, and Richard Vuduc. Synthesizing loops for program inversion. In *Reversible Computation*, volume 7581 of *Lecture Notes in Computer Science*, pages 72–84. Springer-Verlag, 2013.

[3] Cong Hou, George Vulov, Daniel Quinlan, David Jefferson, Richard Fujimoto, and Richard Vuduc. A new method for program inversion. In *Proceedings of the 21st international conference on Compiler Construction*, volume 7210 of *CC'12*, pages 81–100. Springer-Verlag, 2012.

[4] Richard E. Korf. Inversion of applicative programs. In *Proceedings of the 7th International Joint Conference on Artificial intelligence*, IJCAI'81, pages 1007–1009, 1981.

[5] Brian Ross. Running programs backwards: The logical inversion of imperative computation. *Formal Aspects of Computing*, 9:331–348, 1997.