A Logical Representation of Dataflow Simple Pointer Analysis Algorithm

Michal Sekletar
xsekle00@stud.fit.vutbr.cz

Miroslav Soltes
xsolte00@stud.fit.vutbr.cz

November 1, 2011

In our article we would like to introduce more general notation for representation of data flow. The representation is based on predicate logic rather than on set-theory operations. First we will present the brief comparison of both approaches. Afterwards, we will focus on the logical representation of data flow. In order to present an approach based on the logical representation of data flow we will introduce Datalog language. We will briefly discuss syntax and semantics of Datalog language. We assume readers are familiar with Prolog language because of the common nature of both languages. We will present the following aspects of Datalog language : Datalog rules, intensional and extensional predicates, execution of Datalog programs, incremental evaluation of Datalog programs and problematic Datalog rules. In the final chapter of our presentation on the logical representation of data flow we will introduce some examples and possible use cases of this approach to inter-procedural program analysis.

The second part of our presentation deals with a pointer analysis. We will introduce simple approach to the pointer analysis. Note that introduced method of pointer analysis will be flow-insensitive, therefore we assume there are no procedure calls. The fundamental question we would like to answer is whatever the given pair of pointers may be aliased. The pointer aliasing denotes situation when two pointers may point to the same object in memory. We will briefly mention pitfalls of pointer analysis and explain why is it difficult to implement. As a conclusion we will describe intended flow-insensitive pointer analysis method and its formal representation as a Datalog program.