Verification of Pointer Programs Based on Forest Automata

Martin Hruška
ihruska@fit.vutbr.cz

- Improve software quality
- Find all bugs in program
- Formal proof of correctness of program
- Undecidable problems, or problems with high computational complexity

- Formal methods, particularly forest automata
- Automata represent reachable states of program
- Local reasoning (as in separation logic)
- Implementation in Forester tool as GCC plug-in

- Programs in C
- Complex dynamic data structures (e.g., skip-list of the 2nd and 3rd level)
- Bugs related to pointer manipulation or reachability of an error label

Verification Based on Forest Automata

struct Tree {
    struct Tree* left;
    struct Tree* right;
    int data;
};

Concrete domain
Set of heap graphs
Abstraction
Set of forest automata
Abstract domain
Concretization

A (finite, non-deterministic, top-down) tree automata (over structured labels) is quadruple $A = (Q, 2^T, \delta, R)$ where

- $Q$ is a finite set of states.
- $\Gamma$ is a ranked alphabet.
- $\Delta$ is a set of transition rules set with rules in the form $(q, \{a_1, \ldots, a_m\}, q_1 \ldots q_n)$ where $q, q_1, \ldots, q_n \in Q, \{a_1, \ldots, a_m\} \in \Gamma$. Each rule could be interpreted as a sequence of the rule-terms $d(1) = q \rightarrow (a_1, q_1, \ldots, q_n) \ldots d(n) = q \rightarrow (a_m, q_{m+1}, \ldots, q_n)$ and we denote the $i$-th rule term of sequence again by $d(i)$ where $i \in \{1, \ldots, m\}$.
- $R \subseteq Q$ is a finite set of root states.

Contribution

Backward run
- Abstraction over forest automata enables representation of infinite state space
- Abstraction gives analysis chance to terminate and accelerates computation

Predicate abstraction
- Abstraction over forest automata using predicates represented also by forest automata
- It is more precise and more suitable for refinement than the used height abstraction

As a trade-off, abstraction overapproximates state-space
- Necessary to refine abstraction $\Rightarrow$ backward run
- More precise abstraction - predicate abstraction

Precise enough to analyse data structures never analysed before

SV-COMP’15 - Software verification competition
TACAS’15 - Attendance at prestigious international conference

Forester & VATA
- Forest automata are tuples of tree automata
- VATA is an efficient library for tree automata
- Using VATA in Forester brings:
  ○ Modularity
  ○ Maintainability
  ○ Efficiency

Red-black list - verified for the first time ever

More complex data structures coming soon