Verification of Pointer Programs Based on Forest Automata

Martin Hruška
ihruska@fit.vutbr.cz

Why?

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

How?

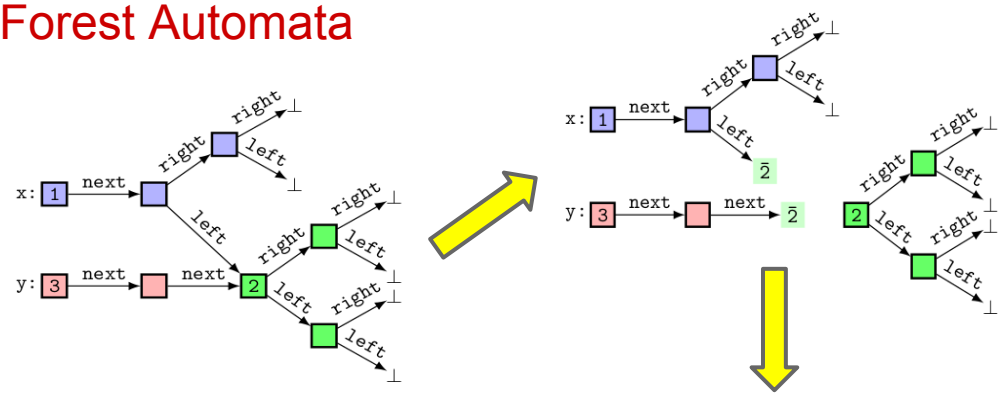
- Formal methods, particularly forest automata
- Automata represent reachable states of program
- Local reasoning (as in separation logic)
- Implementation in Forester tool as GCC plug-in

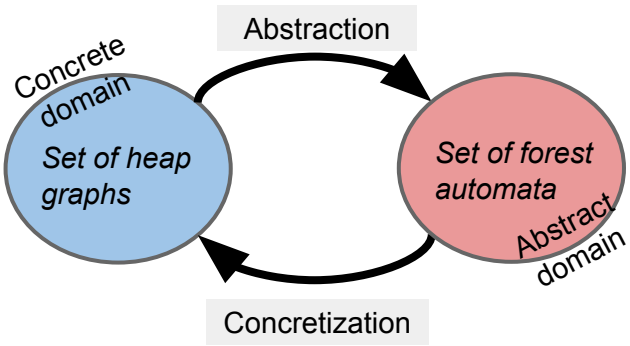
What?

- 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;
};
```





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

- Q is a finite set of states.
- Γ is a ranked alphabet.
- Δ is a set of transition rules set with rules in the form $(q, \{a_1, \ldots, a_m\}, q_1 \cdots q_n)$ 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* $d\langle 1\rangle = q \mapsto (a_1, q_1 \cdots q_{\#a_1}) \cdots d\langle n\rangle = q \mapsto (a_m, q_{n-\#a_m+1} \cdots q_n)$ and we denote the i-th rule term of sequence again by $d\langle i\rangle$ where $i \in \{1, \ldots, m\}$.
- $R \subseteq Q$ is a finite set of root states.

Contribution

SV-COMP'15 - Software verification competition

TACAS'15 - Attendance at prestigious international conference

Backward run

- Abstraction over forest automata enables representation of infinite state space
- Abstraction gives analysis chance to terminate and accelerates computation
- As a trade-off, abstraction overaproximates state-space
- Necessary to refine abstraction ⇨ backward run
- More precise abstraction - predicate abstraction

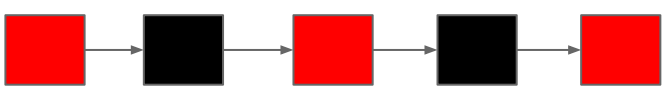
Predicate abstraction

- 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
- Precise enough to analyse data structures never analysed before

Forester & VATA

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

Red-black list - verified for the first time ever



More complex data structures coming soon