Efficient Algorithms for Tree Automata

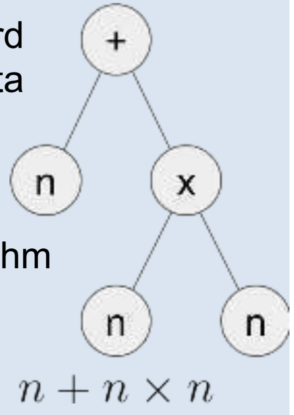
Author: Ondřej Valeš
Supervisor: Ing. Ondřej Lengál Ph.D.

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Tree Automata

- Extension of standard finite (word) automata
- Operates on trees (branching words) instead of words

Goal is to develop algorithm that efficiently checks language inclusion and equivalence on tree automata.



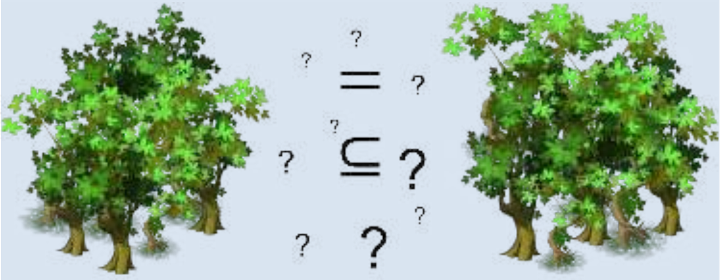
$n + n \times n$

Picture 1. Example tree

Applications

Tree automata are used in verification of programs with tree shaped dynamic data structures, for example red-black trees or threaded trees.

Language inclusion and equivalence check is an important step in verification process.

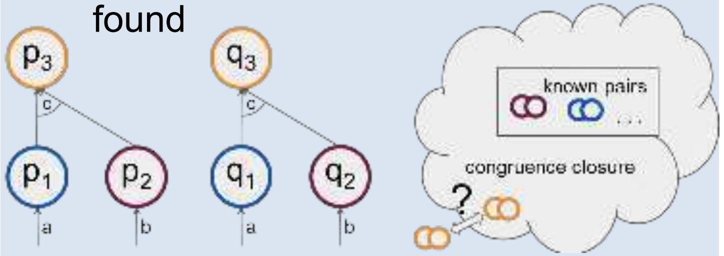


Bisimulation up-to congruence

- Extension of *Hopcroft and Karp* algorithm for use with tree automata
- Simultaneous executions of single steps in both automata
- Determinization *on the fly*
- Premature termination if counterexample is found
- Parts of the search space that are covered (i. e. cannot contain unique counterexample) ale discarded.

Algorithm operation

1. Create initial state pairs
2. Step over the same symbol in both automata, reached states form a new pair
3. Discard new pair if it is in closure
4. Repeat until no new pairs can be created or counterexample was found

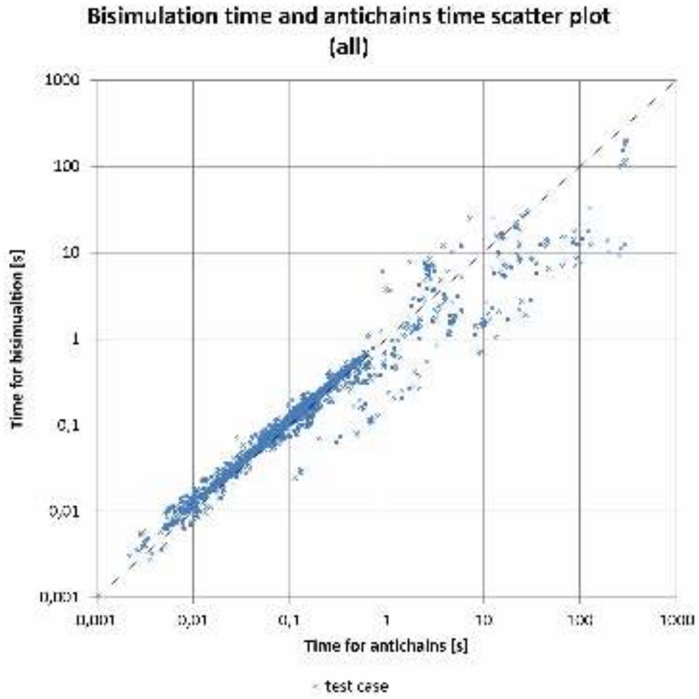


Picture 2. Illustration of *bisimulation up-to congruence* operation.

Experimental results

Bisimulation up-to congruence was compared with state-of-the-art *antichain* algorithm. Algorithms were tested on a set of 9025 automata pairs.

It was shown that *bisimulation* performs similarly on easy cases and often outperforms *antichain* algorithm on harder cases, in extreme cases by one order of magnitude.



Picture 3. Runtime comparison of *bisimulation* and *antichain* algorithms.