# Efficient Algorithms for Using Automata in Infinite-State Verification

Ondřej Lengál

Advisor: prof. Ing. Tomáš Vojnar, Ph.D.

Co-supervised by: Mgr. Lukáš Holík, Ph.D.

Faculty of Information Technology
Brno University of Technology

March 27, 2014

# Scope of the Thesis

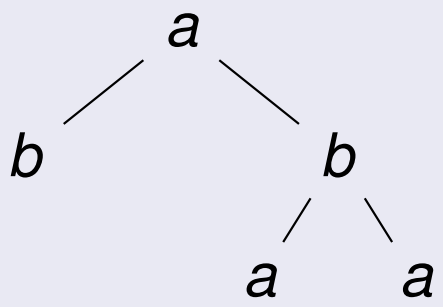
- automata-based FV of parts of real-world programs . . .
  - e.g. the Linux OS kernel, C/C++ standard libraries
- . . . with complex dynamic data structures,
  - e.g. **red-black** trees, skip lists
- efficient automata manipulation techniques,
- the use of automata in other applications,
  - e.g. decision procedures of various logics.

# Outline

1. Forest automata-based verification of programs,
2. a new decision procedure for a fragment of separation logic,
3. ongoing research.

# Tree Automata

- An extension of finite automata from words to trees.

| a word | a tree |
| --- | --- |
| *aaabbaab* |  |

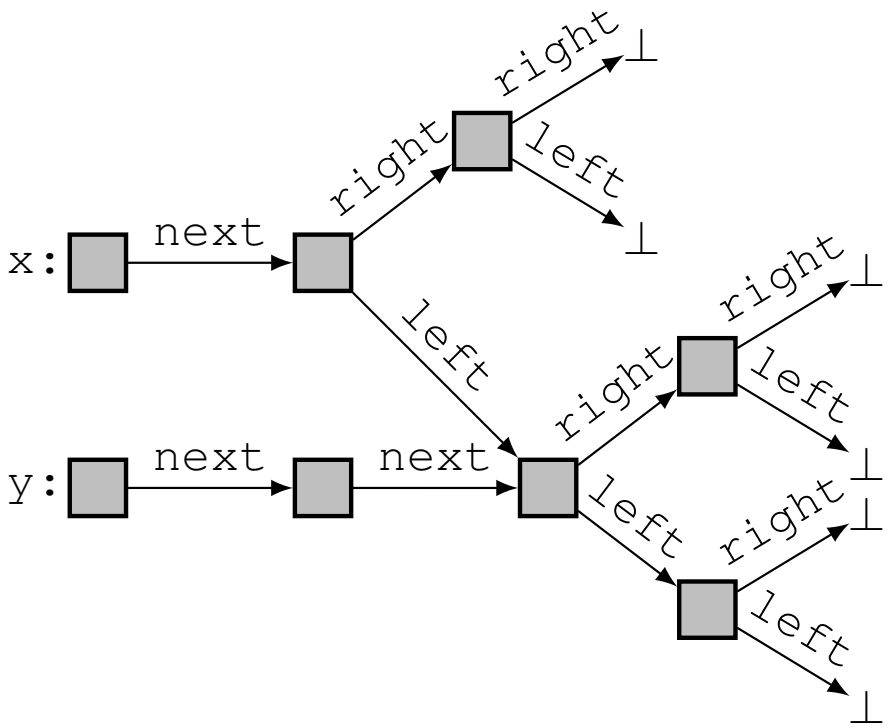


| finite automaton | tree automaton |
| --- | --- |
| $q \to a(p)$ | $q \to a(p_1, \ldots, p_n)$ |

# Forest Automata-based Verification

# Forest Automata-based Verification

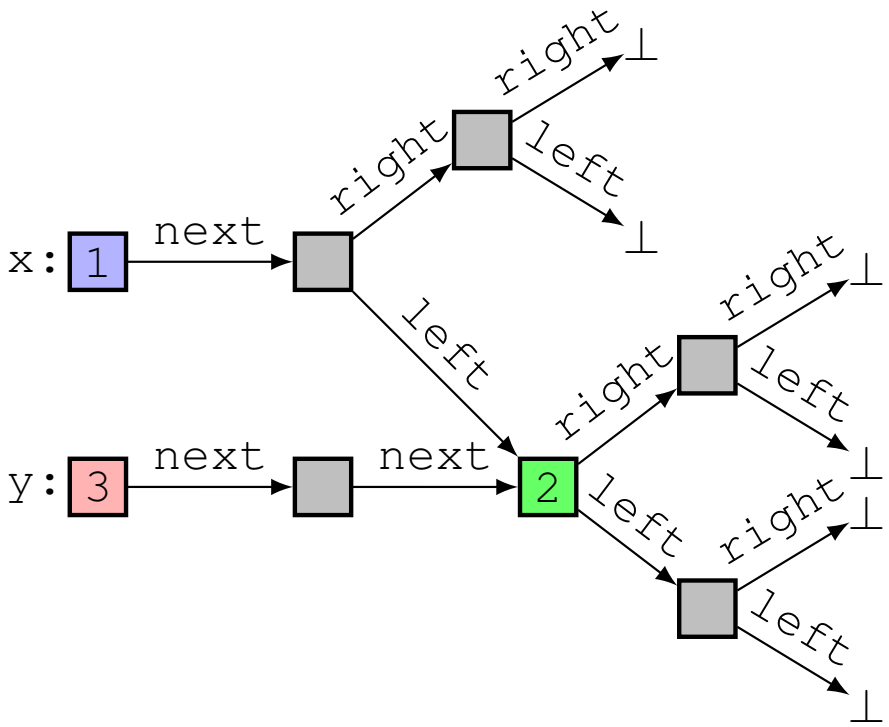
- Forest decomposition of a heap



# Forest Automata-based Verification

- Forest decomposition of a heap
  - Identify cut-points

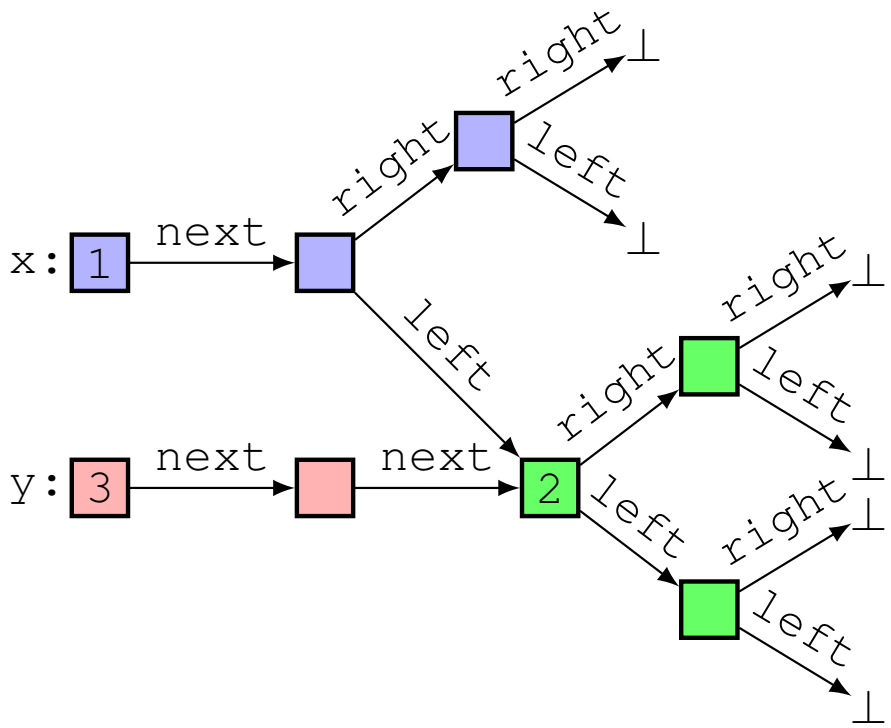
nodes referenced:
- by variables, or
- multiple times



# Forest Automata-based Verification

- Forest decomposition of a heap
  - Identify cut-points
  - Split the heap into tree components

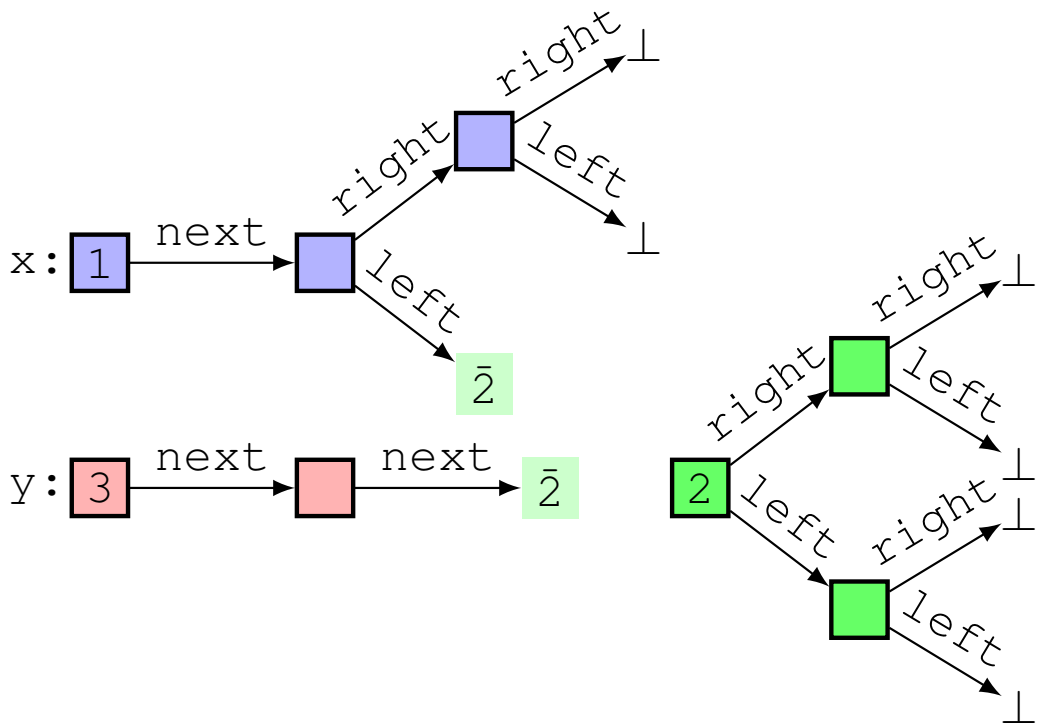
nodes referenced:
- by variables, or
- multiple times



# Forest Automata-based Verification

- Forest decomposition of a heap
  - Identify cut-points
  - Split the heap into tree components
    - references are explicit

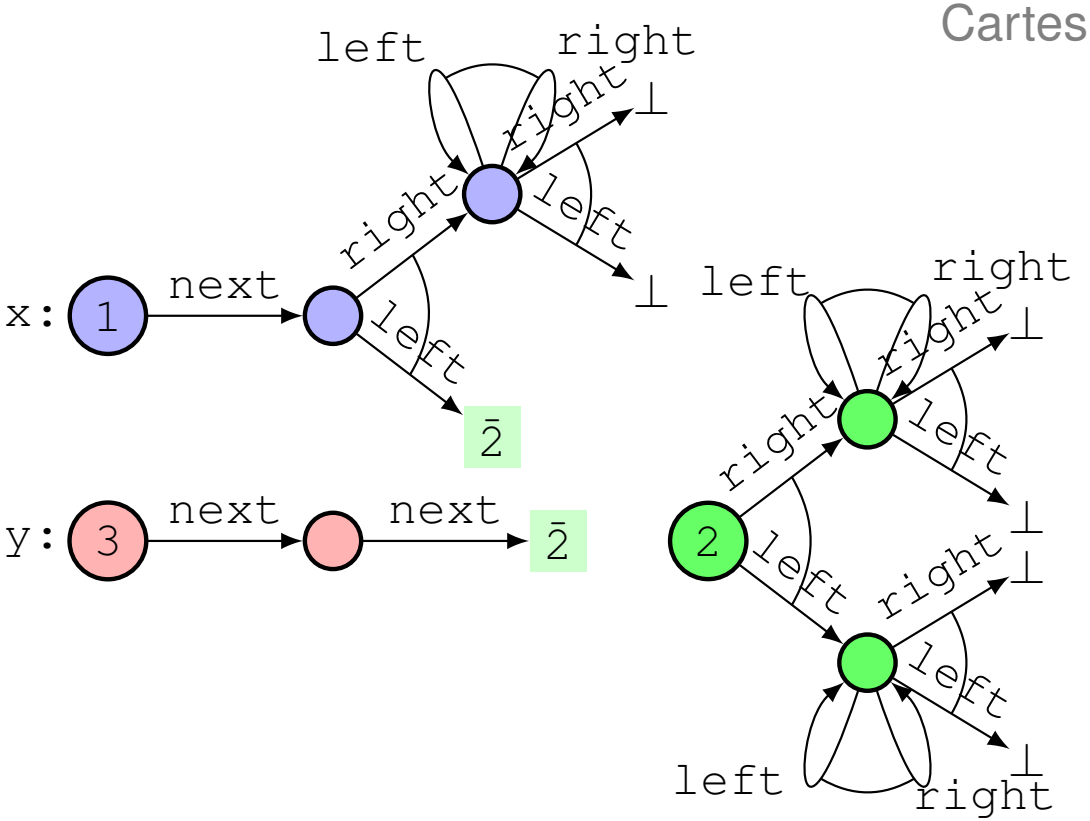
nodes referenced:
- by variables, or
- multiple times



# Forest Automata-based Verification

- Forest decomposition of a heap
  - Identify cut-points ← nodes referenced:
    - by variables, or
    - multiple times
  - Split the heap into tree components
    - references are explicit
  - Sets of heaps:
    - tree automata to represent sets of tree components
    - tuple of tree automata $(TA_1, \ldots, TA_n) \rightsquigarrow$ **forest automaton** (FA)

Cartesian semantics



# Forest Automata-based Verification

- Memory safety of programs manipulating heap.
- Uses a hierarchical encoding of symbols — called boxes,
  - to avoid having infinite tuples of tree components.
- Boxes are discovered automagically,
  - by finding candidate subgraphs to enclose in a box.
- ARTMC-based procedure for obtaining a fixpoint.
- The Forester tool.

# Forest Automata-based Verification (data extension)

- Extending Forester with support for data-dependent programs,
  - binary search trees, sorting algorithms, skip lists, ...
- Adding constraints $\prec_{rr}, \prec_{ra}$ (**r**oot-**r**oot, **r**oot-**a**ll)
  - locally (inside transitions):

$$q^0 \to [\texttt{left}^1(q), \texttt{right}^2(q)] : 0 \succ_{ra} 1 \wedge 0 \prec_{ra} 2$$

  - globally (between a state and a TA):

$$TA_0 \prec_{ar} q \prec_{ra} TA_1$$

# Forest Automata-based Verification (data extension)

- Extending Forester with support for data-dependent programs,
  - binary search trees, sorting algorithms, skip lists, ...
- Adding constraints $\prec_{rr}, \prec_{ra}$ (**r**oot-**r**oot, **r**oot-**a**ll)
  - locally (inside transitions):

$$q^0 \to [\texttt{left}^1(q), \texttt{right}^2(q)] : 0 \succ_{ra} 1 \wedge 0 \prec_{ra} 2$$

  - globally (between a state and a TA):

$$TA_0 \prec_{ar} q \prec_{ra} TA_1$$

- The need to adjust
  - abstract transformers,
  - inclusion checking,
  - box (un-)folding.

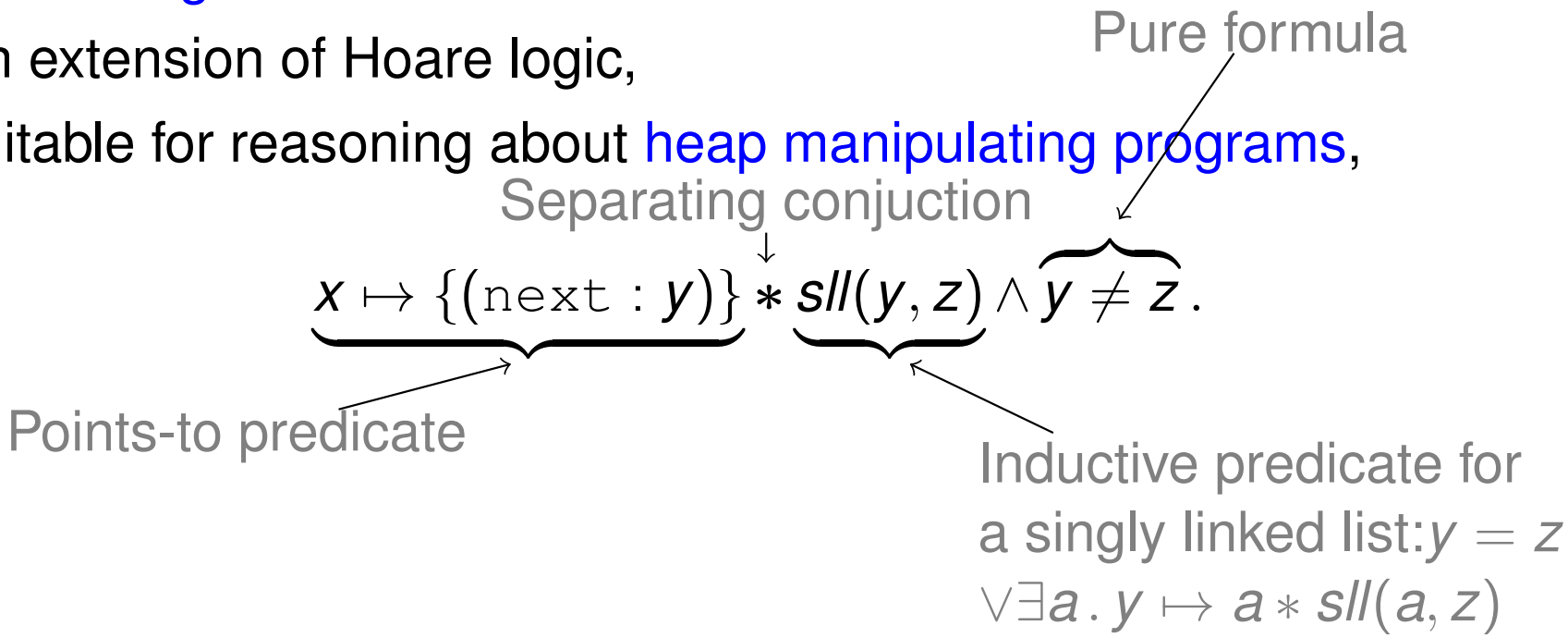
- P. A. Abdulla, L. Holík, B. Jonsson, O. Lengál, C.-Q. Trinh, and T. Vojnar.
  Verification of Heap Manipulating Programs with Ordered Data by Extended Forest Automata.
  In *Proc. of ATVA'13*, LNCS 8172. (regular paper)

# Deciding a Fragment of Separation Logic

# Decision Procedure for a Fragment of Separ. Logic

## Separation Logic

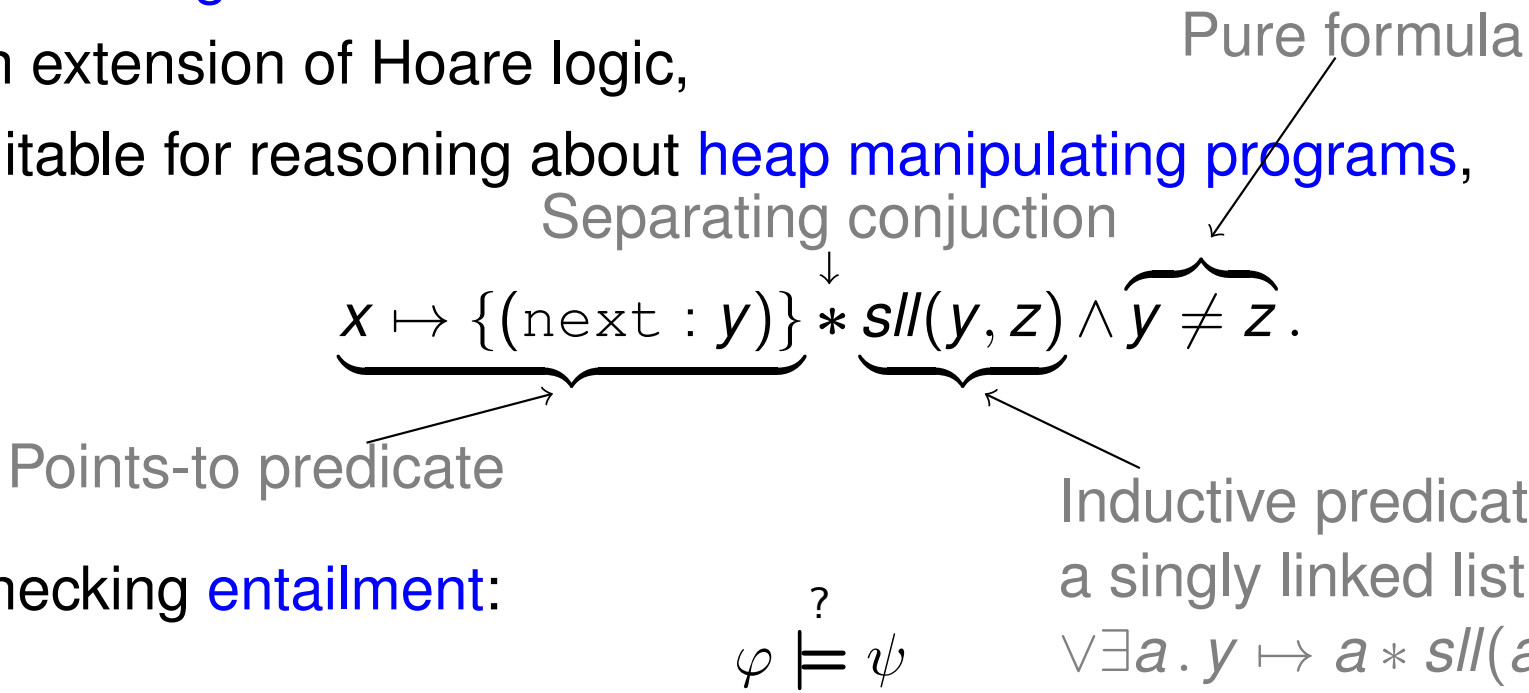
- An extension of Hoare logic,
- suitable for reasoning about heap manipulating programs,



# Decision Procedure for a Fragment of Separ. Logic

## Separation Logic

- An extension of Hoare logic,
- suitable for reasoning about heap manipulating programs,




- Checking entailment:

$$\varphi \overset{?}{\models} \psi$$

e.g.

$$x \mapsto \{(\texttt{next} : y)\} * sll(y, z) * z \mapsto \{(\texttt{next} : w)\} \overset{?}{\models} sll(x, w).$$

- Used e.g. in program analysis for fixpoint detection.
- Undecidable in general, but there exist decidable fragments.

# Decision Procedure for a Fragment of Separ. Logic

- The considered fragment is limited to lists
  - that are singly/doubly linked,
  - that may be cyclic and/or nested, and
  - that may contain additional head/tail/skip pointers.

# Decision Procedure for a Fragment of Separ. Logic

- The decision procedure works as follows:

$$\varphi = \Pi \ \wedge \ \varphi_1 * \ldots * \varphi_m \overset{?}{\models} \psi_1 * \ldots * \psi_n \ \wedge \ \Theta = \psi$$

1. normalize $\varphi$ and $\psi$ (adds (dis)equalities, removes emp. segments),
2. check that $\Pi \Rightarrow \Theta$,
3. for each points-to $(x \mapsto \{\ldots\})$ in $\psi$, find a points-to in $\varphi$,
4. partition the rest of $\varphi$, map each class to inductive predicate in $\psi$,
5. for each inductive predicate $P$ in $\psi$ check that $\varphi|_P \models P$ by
   1. encoding $\varphi|_P$ as a tree $T(\varphi|_P)$,
   2. encoding $P$ as a tree automaton $TA(P)$, and
   3. checking that $T(\varphi|_P) \in L(TA(P))$.

- C. Enea, O. Lengál, M. Sighireanu, T. Vojnar.
  Compositional Entailment Checking for a Fragment of Sep. Logic.
  Submitted to SAS'14. (regular paper)

# Ongoing Research

# Ongoing Research

1. A decision procedure for WS*k*S,
   - found a bug in the proposed algorithm, attempting to fix it.
2. Algorithms for handling symbolically represented automata,
   - computation of simulation, language inclusion.
3. Verification of concurrent programs with complex data structures
   - based on forest automata.
4. Adjustable abstraction in Forester,
   - counterexample-guided abstraction refinement.
5. Efficient language inclusion testing for tree automata,
   - based on bisimulation up-to congruence.

# Recent Publications

- Published:
  - L. Holík, O. Lengál, A. Rogalewicz, J. Šimáček, and T. Vojnar. Fully Automated Shape Analysis Based on Forest Automata. In *Proc. of CAV'13*, LNCS 8044. (regular paper)
  - P. A. Abdulla, L. Holík, B. Jonsson, O. L., C.-Q. Trinh, and T. Vojnar. Verification of Heap Manipulating Programs with Ordered Data by Extended Forest Automata. In *Proc. of ATVA'13*, LNCS 8172. (regular paper)
- Submitted:
  - P. A. Abdulla, L. Holík, B. Jonsson, O. L., C.-Q. Trinh, and T. Vojnar. Verification of Heap Manipulating Programs with Ordered Data by Extended Forest Automata. Submitted to Acta Informatica (IF'12 = 0.474). (regular paper)
  - C. Enea, O. Lengál, M. Sighireanu, T. Vojnar. Compositional Entailment Checking for a Fragment of Sep. Logic. Submitted to SAS'14. (regular paper)

# Previous Publications

- Published:
  - L. Holík, O. Lengál, J. Šimáček, and T. Vojnar.
    Efficient Inclusion Checking on Explicit and Semi-Symbolic TA.
    In *Proc. of ATVA'11*, LNCS 6996. (regular paper)
  - O. Lengál, J. Šimáček, and T. Vojnar.
    VATA: A Library for Efficient Manipulation of Non-Deterministic TA.
    In *Proc. of TACAS'12*, LNCS 7214. (regular tool paper: 16 pp)
  - O. Lengál.
    An Efficient Finite Tree Automata Library.
    Lambert Academic Publishing. 2012. (monograph)
- Products:
  - O. Lengál, J. Šimáček, and T. Vojnar.
    VATA: A Library for Efficient Manipulation of Non-Deterministic TA.
    FIT BUT, 2012. (software)