TAO-WSTS : Theory, algorithms and tools for WSTS

Alain FINKEL
BRAVAS 26 octobre 2017
1. Theory for WSTS

2. Algorithms for WSTS

3. Tools for WSTS
Theory, algorithms and tools for WSTS

- WSTS are monotone TS + a well-founded ordering but without infinite antichains.

- Increasingly used to verify web protocols, cache management protocols, multithreaded programs, ...

- But most of the algorithms used to check templates are those designed 15 - 20 years.

- However, for the last ten years or so, the theory of WSTS has undergone a major revival, notably with the theory of ideals and the possibility of finite, and effective, representation of downward closed sets.
New forward coverability algorithm with ideals enumeration

**Procedure 1:** searches for a coverability certificate of $y$ from $x$

1. $D \leftarrow \downarrow x$
2. **while** $y \notin D$ **do**
3. $D \leftarrow D \cup \downarrow \text{Post}(D)$
4. **return** *true*

**Procedure 2:** enumerates inductive invariants to find non coverability certificate of $y$ from $x$.

1. $i \leftarrow 0$
2. **while** $\neg(\downarrow \text{Post}(D_i) \subseteq D_i$ and $x \in D_i$ and $y \notin D_i)$ **do**
3. $i \leftarrow i + 1$
4. **return** *false*
LTL model checking for very WSTS

• Definition. A **very-WSTS** is a labeled WSTS
  \[ S = (X,\to,\leq) \] such that:
  – S has strong monotonicity
  – C(S) is a deterministic WSTS with strong-strict monotonicity
  – Idl(X) has finitely many levels
Acceleration levels

We say that an infinite sequence of ideals $I_0, I_1, \ldots \in \text{Idl}(X)$ is an acceleration candidate if $I_0 \subset I_1 \subset \cdots$.

**Definition 4.** For every $n \in \mathbb{N}$, the $n^{th}$ level of $\text{Idl}(X)$ is defined as

$$\text{Acc}_n(X) = \begin{cases} \text{Idl}(X) & \text{if } n = 0, \\ \bigcup_{i \in \mathbb{N}} I_i : I_0, I_1, \ldots \in \text{Acc}_{n-1}(X) \text{ is an acceleration candidate} & \text{if } n > 0. \end{cases}$$

We observe that $\text{Acc}_{n+1}(X) \subseteq \text{Acc}_n(X)$ for every $n \in \mathbb{N}$. Moreover, as expected:

$$\text{Acc}_n(\mathbb{N}^d) = \{ I \in \text{Idl}(\mathbb{N}^d) : \omega\text{-rep}(I) \text{ has at least } n \text{ occurrences of } \omega \}.$$

We say that $\text{Idl}(X)$ has finitely many levels if there exists $n \in \mathbb{N}$ such that $\text{Acc}_n(X) = \emptyset$. For example, $\text{Acc}_{d+1}(\mathbb{N}^d) = \emptyset$. 

Ideal Karp-Miller algorithm

Algorithm 4.1: Ideal Karp-Miller algorithm.

1. initialize a tree $\mathcal{T}$ with root $r: \langle I_0, 0 \rangle$
2. while $\mathcal{T}$ contains an unmarked node $c: \langle I, n \rangle$ do
   3. if $c$ has an ancestor $c': \langle I', n' \rangle$ s.t. $I' = I$ then mark $c$
   4. else
      5. if $c$ has an ancestor $c': \langle I', n' \rangle$ s.t. $I' \subset I$
         and $n' = n$ /* no acceleration occurred between $c'$ and $c$ */ then
      6. $w \leftarrow$ sequence of labels from $c'$ to $c$
      7. replace $c: \langle I, n \rangle$ by $c: \langle w^{\infty}(I), n + 1 \rangle$
      8. for $a \in \Sigma$ do
         9. if $a(I)$ is defined then
            10. add arc labeled by $a$ from $c$ to a new child $d: \langle a(I), n \rangle$
            11. mark $c$
   12. return $\mathcal{T}$
Theory, **algorithms** and tools for WSTS

- WSTS algorithmics need to be better understood:

- there are several algorithms for particular classes of models, but it remains to understand the principles of the most efficient algorithms for WSTS.

- Downward-closed sets...
Theory, algorithms and tools for WSTS

• We will test the efficiency of algorithms with prototypes.

• We will continue the effort started in 2000 with the tool FAST and recently with the tool QCOVER by aiming to make the first prototype for solving reachability for Petri nets.
Reachability algorithm

• Reachability algorithm (S : Petri net ; x,y : configurations)

• BEGIN
• IF y is not reachable from x in the associated continuous Petri net OR in the integer Petri net (without guards)
• THEN
• STOP (write « y is not reachable from x »)
• ELSE
• IF y is not coverable from x in S
• THEN
• STOP (write « y is not reachable from x ») ;
• ELSE
• Use over-approximations for solving non-reachability : compute the coverability graph,…
• Use exact accelerations for solving reachability : compute semilinear subsets of the reachability set
• Use machine learning techniques (prospectives)
• Use finally, a complex algorithm like the Leroux algorithm with Presburger invariants, the Mayr-Kosaraju algorithm.
• END