Termination and Boundedness for Well-structured Pushdown Systems

Suhua Lei, Xiaojuan Cai
BASICS Lab, Shanghai Jiao Tong University

Mizuhito Ogawa
School of Information Science, Japan Advanced Institute of Science and Technology

May 17, 2016

IS-RR-2016-001

Termination and Boundedness for Well-Structured Pushdown Systems

Suhua Lei[1], Xiaojuan Cai[1], and Mizuhito Ogawa[2]

[1] BASICS Lab, Shanghai Jiao Tong University, China
leisuhua,cxj@sjtu.edu.cn
[2] Japan Advanced Institute of Science and Technology, Japan
mizuhito@jaist.ac.jp

Abstract. Well-structured pushdown systems (WSPDSs) extend pushdown systems with well-quasi-ordered (possibly infinitely many) states and stack alphabet. As an expressive model for concurrent recursive computations, WSPDSs are believed to "be close the border of undecidability" [11]. Most of the decidability results are known only on subclasses. In this paper, we investigate the decidability of the termination and boundedness problems for WSPDSs using two algorithms: One is an extension of the reduced reachability tree technique proposed by Leroux *et. al.* in [11]. The other is based on Post*-automata technique which has been successfully applied in the model checking of pushdown systems. The complexity of both are Hyper-Ackermannian for *bounded WSPDSs*. We implement both algorithms and make experiments on a large number of randomly generated WSPDSs. The results illustrate that the Post*-automata based algorithm sometimes behaves an order of magnitude faster. [3]

1 Introduction

Pushdown systems (PDSs) and Vector addition systems (VASs) are both powerful models with decidable reachability. A PDS has finitely many control states equipped with a stack for storing words over finite stack alphabet and the transition rules may push or pop on the stack. It is natural for modeling recursive computations. A VAS consists of integer vector addition rules over vectors of natural numbers. It is usually used for describing concurrent computations.

A well-structured pushdown system (WSPDS) [4] is a pushdown system with both well-quasi-ordered control states and stack alphabet, which can be viewed as a combinations of a pushdown system and a well-structured transition system (WSTS) [9,2].

Like other VAS-extensions, WSPDSs have four crucial problems for verification : *Reachability problem* asks whether a configuration is reachable; *Coverability problem* asks whether a configuration can be covered by a reachable configuration; *Termination problem* asks whether all runs terminate; *Boundedness problem* asks whether the reachability set (the set of reachable configurations) is finite.

[3] A part of the results are orally presented at YR-ICALP 2015 (Kyoto, 5 July 2015).

The decidability of these properties are confirmed on only subclasses of WSPDSs, e.g., the coverability is decidable for WSPDSs with finite control states and well-quasi-ordered stack alphabet [4] and 1-dimension vector control states and finite stack alphabet [12]. Note that the boundedness and the reachability may be undecidable for a WSTS, if its transitions are not strictly monotonic; an example is a VAS with reset operations [7].

In this paper, we show that the termination problem is decidable for general WSPDSs and the boundedness problem is decidable for *strict* WSPDSs (WSPDS with strictly monotonic transitions). Our contributions include:

- The reduced reachability tree algorithm in [11] is generalized to WSPDSs in a straightforward way, and applied to prove the decidability of the termination/boundedness problem (Section 3).
- The Post*-automata algorithm is proposed and provides alternative algorithms for the termination/boundedness problem of WSPDSs (Section 4).
- We induce Hyper-Ackermannian upper and lower bounds of both algorithms for *bounded* WSPDSs where control states and stack alphabet are vectors (Section 5).
- The experimental results show practical advantages of the algorithm based on Post*-automata (Section 6).

The reachability tree and the Post*-automata algorithms can be regarded as an analogy to two ways of the emptiness checking on pushdown automata, i.e., the pumping lemma and the P-automata (Post*) [8], whose (upper bound) complexity are exponential and cubic, respectively. The latter is typically used for the implementation of pushdown model checking[15].

Related Work

This paper is inspired by [11]. They also proposed the notion of "*well-structured pushdown systems*" (Definition III.1 in [11]), which are WSPDSs (in this paper) with finite stack alphabet. Their reduced reachability tree technique is an extension of a Karp-Miller tree to prove the decidability of termination and boundedness problems.

Our work extends their technique to general WSPDSs, i.e., with both states and stack alphabet being well-quasi-orders, and proves the decidability of termination (resp. boundedness) for general (resp. strict) WSPDSs. Our reachability tree and complexity estimation heavily depend on [11]. Our originality is mostly on the Post*-automata algorithm. We (as well as [11]) focus on finitely branching and strongly monotonic transitions. Observation beyond them (e.g., [2]) requires careful attention.

There are several models of concurrent recursive computation, e.g., a *branching VAS* (BVAS) [6], a *recursive VASS* (RVASS) [3], and an *alternating VASS* (AVASS) [5]. Their coverability is known to be decidable, but the reachability is left open except that an AVASS is known to be undecidable [5]. All of them can be encoded into WSPDSs with finite control states, and their decidability of the coverability is obtained from general arguments on WSPDSs [4][4]. A *pushdown* VAS [11] is a WSPDS $\langle(\mathbb{N}^k, \leq), \Gamma, \Delta\rangle$ with finite stack alphabet Γ. If its dimension k is 1, its coverability is decidable [12], while that with larger dimensions is still open.

2 Pushdown systems and P-automata

2.1 Pushdown system

Definition 1. *A* pushdown system (PDS) *is a 4-tuple* $\langle P, \Gamma, \Delta, c_0\rangle$ *where* P *is a set of states,* Γ *is stack alphabet,* c_0 *is the initial configuration and* $\Delta \subseteq P \times \Gamma^{\leq 1} \times P \times \Gamma^{\leq 2}$ *is a set of transitions. A transition* $\hookrightarrow$ *between configurations is defined as follows.*

$$\mathit{inter}\frac{(p,\gamma \to p',\gamma') \in \Delta}{\langle p,\gamma w\rangle \hookrightarrow \langle p',\gamma' w\rangle} \quad \mathit{push}\frac{(p,\gamma \to p',\alpha\beta) \in \Delta}{\langle p,\gamma w\rangle \hookrightarrow \langle p',\alpha\beta w\rangle}$$
$$\mathit{pop}\frac{(p,\gamma \to p',\epsilon) \in \Delta}{\langle p,\gamma w\rangle \hookrightarrow \langle p',w\rangle}$$

where a configuration $\langle p, w\rangle$ *is a pair of a state* p *and a word* $w \in \Gamma^*$*, and we write* $p, w \to q, v$ *if* $(p, w, q, v) \in \Delta$. $\hookrightarrow^*$ *is the reflexive transitive closure of* $\hookrightarrow$.

Given some initial configuration c_0, a *run* starting from c_0 is a finite or infinite sequence: $c_0 \hookrightarrow c_1 \hookrightarrow c_2 \hookrightarrow \cdots$. The *reachability set* of c_0 is defined $\{c \mid c_0 \hookrightarrow^* c\}$.

- *Termination problem* decides whether each run starting from c_0 is finite, and
- *boundedness problem* decides whether the reachability set of c_0 is finite.

For later convenience, we introduce two more forms of rules. *simple push* and *non-standard pop.* They often appear in instances of WSPDSs, such as a branching VAS and a recursive VASS.

$$\text{simple-push}\frac{(p,\epsilon \to p',\alpha) \in \Delta}{\langle p,w\rangle \hookrightarrow \langle p',\alpha w\rangle}$$
$$\text{nonstandard-pop}\frac{(p,\alpha\beta \to p',\gamma) \in \Delta}{\langle p,\alpha\beta w\rangle \hookrightarrow \langle p',\gamma w\rangle}$$

Throughout the paper, we use the following notational convention.

- $\alpha, \beta, \gamma, \cdots$ range over Γ,
- $w, v, \cdots$ range over words in Γ^*. $|w|$ denotes the length of word w and $w[i]$ denotes the i-th symbol in w. The *head* $h(p, w)$ of a configuration $\langle p, w\rangle$ is $(p, w[1])$ if $w \neq \epsilon$; otherwise, $h(p, w) = (p, \perp)$.
- $p, q, \cdots$ range over states, and $c, d, \cdots$ range over configurations.
- We use $\to$ in rules, $\hookrightarrow$ for transitions between configurations, and $\mapsto$ for edges (transitions) in Post*-automata.
- We denote $\mathbb{N}$ (resp. $\mathbb{Z}$) for the set of natural numbers (resp. integers).

[4] The reachability problem of Alternating VASS cannot be interpreted in WSPDS because of conflicts with the monotonicity.

2.2 *Post**-automaton

A *P-automaton* [8] is an automaton that accepts the set of reachable configurations of a PDS.

Definition 2. *Given a PDS $\mathcal{M} = \langle P, \Gamma, \Delta, c_0 \rangle$, a P-automaton $\mathcal{A}$ is a quadruplet (S, Γ, ∇, F) where*

- *S is the set of states and $S \cap P \neq \emptyset$,*
- *F is the set of final states, and $F \subseteq S$,*
- *$\nabla \subseteq S \times (\Gamma \cup \{\epsilon\}) \times S$ is the set of transitions.*

We write $s \overset{\gamma}{\mapsto} s'$ for $(s, \gamma, s') \in \nabla$ (possibly $\gamma = \epsilon$), and $\mapsto^*$ for the reflexive transitive closure of $\mapsto$.

$\mathcal{M}$ accepts *$\langle p, w \rangle$ for $p \in P$ and $w \in \Gamma^*$ if there exists some $f \in F$ s.t. $p \overset{w}{\longmapsto}{}^* f$. We use $L(\mathcal{A})$ to denote the set of configurations that $\mathcal{A}$ accepts.*

Let $\mathcal{A}_0$ be the initial P-automaton that accepts $C_0 = \{c_0\}$. Let *post**(C_0) be the set of all successors of configurations of C_0. The *saturation* procedure to compute *post**(C_0) starts from $\mathcal{A}_0$, and repeatedly adds states and edges according to the rules of a PDS until convergence.

Definition 3. *For a PDS $\mathcal{M} = \langle P, \Gamma, \Delta, c_0 \rangle$, let $\mathcal{A}_0 = (S_0, \Gamma, \triangledown_0, F)$ be the initial P-automaton that accepts $C_0 = \{c_0\}$. Post*$(\mathcal{A}_0)$ is the automaton generated by repeated applications of the following Post*-saturation rules:*

- *if $p, w \to p', \gamma \in \Delta$:*

$$\frac{(S, \Gamma, \triangledown, F), p \overset{w}{\longmapsto}{}^* q \in \nabla}{(S \cup \{p'\}, \Gamma, \triangledown \cup \{p' \overset{\gamma}{\mapsto} q\}, F)}$$

- *if $p, \gamma \to p', \alpha\beta \in \Delta$:*

$$\frac{(S, \Gamma, \triangledown, F), p \overset{\gamma}{\mapsto} q \in \nabla}{(S \cup \{p', q_{p',\alpha}\}, \Gamma, \triangledown \cup \{p' \overset{\alpha}{\mapsto} q_{p',\alpha} \overset{\beta}{\mapsto} q\}, F)}$$

For instance, consider a push rule $(p, \gamma \to p', \alpha\beta)$. If $p \overset{\gamma}{\mapsto} q$ is in ∇, then $p' \overset{\alpha}{\mapsto} q_{p',\alpha} \overset{\beta}{\mapsto} q$ is added to ∇. The intuition is, if, for $v \in \Gamma^*$, $\langle p, \gamma v \rangle$ is in *post**(C_0), then $\langle p', \alpha\beta v \rangle$ is also in *post**(C_0) by applying rule $(p, \gamma, p' \to \alpha\beta)$.

*Remark 1. Post**-saturation introduces ϵ-transitions when applying standard pop rules. To avoid it, we apply the following preprocessing on a WSPDS, which results in an equivalent WSPDS.

1. The stack is initialized with a bottom symbol $\perp$,
2. Each standard pop rule $\psi \in \mathcal{F}(P \times \Gamma, P \times \{\epsilon\})$ is replaced with $\psi' \in \mathcal{F}(P \times \Gamma^2, P \times \Gamma)$ and for any $\beta \in \Gamma \cup \{\perp\}$, $\psi'(p, \alpha\beta) = (q, \beta)$ if $\psi(p, \alpha) = (q, \epsilon)$.

For a PDS, $Post^*(\mathcal{A}_0)$ has bounded numbers of states, since each newly added state $q_{p,\gamma}$ is indexed by a pair of a state and a stack symbol, which are finitely many. Thus, the saturation procedure finitely converges and accepts exactly the reachability set of C_0 (Theorem 1). When we consider P and Γ to be infinite, $Post^*(\mathcal{A}_0)$ may not finitely converge. However, it has a limit $\cup_i Post^i(\mathcal{A}_0)$ (by taking set unions of states and transitions, which monotonically increase). Theorem 1 holds under such generalization.

Theorem 1. *[8] $post^*(C_0) = L(Post^*(\mathcal{A}_0))$.*

2.3 Well-structured pushdown systems

A *quasi-order* $(S, \preceq)$ is a reflexive transitive binary relation on a set S. We denote $s \prec t$ if $s \preceq t$ and $t \not\preceq s$. A *partial order* is an anti-symmetric quasi-order. A quasi-order $(S, \preceq)$ is a *well-quasi-order* (WQO), if, for any infinite sequence $s_1, s_2, s_3, \dots$ in S, there exist indices i, j with $i < j$ and $s_i \preceq s_j$.

For WQOs $(X_1, \leq_1)$ and $(X_2, \leq_2)$, a product of WQOs $(X_1 \times X_2, \leq)$ is a WQO by Dickson's Lemma, where $(x_1, x_2) \leq (x'_1, x'_2)$ if $x_1 \leq_1 x'_1$ and $x_2 \leq_2 x'_2$. $(\mathbb{N}^k, \leq)$ is a WQO for $k \in \mathbb{N}$, where $\leq$ is the product extension on $\mathbb{N}^k$.

We denote $a_1 a_2 \dots a_m \lll b_1 b_2 \dots b_n$, if $m = n$ and, for each i, $a_i \leq b_i$ holds, and $w {\ll} v$ if $w \lll v$ and $w \not\ggg v$. Note that $\lll$ may be not a WQO for a WQO $\leq$. We also assume the least element $\perp$ in Γ, representing the stack bottom.

Let $\mathcal{F}(X, Y)$ denote the set of partial functions from a set X to a set Y.

Definition 4. *[4] A* well-structured pushdown system (WSPDS) *is a 4-tuple $\mathcal{M} = \langle (P, \preceq), (\Gamma, \leq), \Delta, c_0 \rangle$ where*

- *$(P, \preceq)$ and $(\Gamma, \leq)$ are WQOs, and*
- *$\Delta \subseteq \mathcal{F}(P \times \Gamma^{\leq 2}, P \times \Gamma^{\leq 2})$ is a finite set of monotonic partial computable functions (w.r.t. $\preceq \times \lll$).*

We denote $\langle p, w \rangle \hookrightarrow \langle p', w' \rangle$ if there exists $\psi(p, u) = (p', u')$ for $u, u' \in \Gamma^{\leq 2}$ where $w = u.v$ and $w' = u'.v$.

Note that the set of heads of configurations is well-quasi-ordered by $\trianglelefteq = \preceq \times \leq$.

A WSPDS $\langle (P, \preceq), (\Gamma, \leq), \Delta \rangle$ is *strict* if $(P, \preceq)$ and $(\Gamma, \leq)$ are partial order, and Δ consists of strictly monotonic partial functions (i.e., $(p, \gamma) \triangleleft (p', \gamma')$ implies $f(p, \gamma) \triangleleft f(p', \gamma')$).

Example 1. Let $\mathcal{M} = \langle (\mathbb{N}, \leq), (\mathbb{N}, \leq), \Delta, c_0 \rangle$ where

$$\Delta = \begin{cases} r_1 : p, \alpha \to p + 1, (\alpha - 1)(\alpha - 1) \\ r_2 : p, \epsilon \to p + 2, 0 \ \text{ if } p \geq 2 \\ r_3 : p, \alpha \to p - 3, \alpha + 3 \\ r_4 : p, \alpha\beta \to p, \alpha + \beta - 2 \end{cases}$$

M is a WSPDS with both its states and stack symbols being natural numbers. The transitions rules in Δ are defined by four monotonic partial functions, each of which denotes sets of push, simple push, internal, and non-standard pop rules, respectively. M is also a strict WSPDS.

Assume $c_0 = \langle 1, 1\rangle$, here is an infinite run with an infinite reachability set:

$$\langle 1,1\rangle \overset{r_1}{\hookrightarrow} \langle 2,00\rangle \overset{r_2}{\hookrightarrow} \langle 4,000\rangle \overset{r_3}{\hookrightarrow} \langle 1,300\rangle \overset{r_4}{\hookrightarrow} \langle 1,10\rangle \overset{r_1}{\hookrightarrow} \cdots$$

If we change rule r_1 to $r_1' : p, \alpha \to p+1, (\alpha - 1)$, this infinite run contains a finite reachability set: $\langle 1,1\rangle \overset{r_1}{\hookrightarrow} \langle 2,0\rangle \overset{r_2}{\hookrightarrow} \langle 4,00\rangle \overset{r_3}{\hookrightarrow} \langle 1,30\rangle \overset{r_4}{\hookrightarrow} \langle 1,1\rangle \overset{r_1}{\hookrightarrow} \langle 2,0\rangle \cdots$. If we further remove the rule r_2, all runs starting from c_0 terminate. □

3 The reduced reachability tree algorithm

The *reachability tree* of a WSPDS $\mathcal{M} = \langle (P, \preceq), (\varGamma, \leq), \varDelta, c_0\rangle$ with an initial configuration c_0 is a rooted unordered tree defined as follows. Each node of the tree is labeled by a configuration of $\mathcal{M}$. The root r is labeled by c_0, denoted by $r : c_0$. Each node $n : c_n$ has a child $m : c_m$ when $c_n \hookrightarrow c_m$. Note that the reachability tree of $\mathcal{M}$ is finitely branching since $\varDelta$ is finite.

3.1 Termination problem

Definition 5. *A node $s : \langle p, w\rangle$* pumps *a node $t : \langle q, v\rangle$ if*

- *there is a path from s to t, and every node $t' : \langle p', w'\rangle$ on it satisfies $|w'| \geq |w|$.*
- *$h(\langle p, w\rangle) \trianglelefteq h(\langle q, v\rangle)$, i.e., $p \preceq q$ and either $w = \epsilon$ or $w[1] \leq v[1]$.*

We call a node *pumpable* if there exists a node pumping it. The notion of pumpable nodes is similar to *subsumed nodes* in [11], but we consider the increase of heads instead of states. Let *the reduced reachability tree* be the largest prefix of the reachability tree such that every *pumpable node* has no children.

Example 2. Recall the WSPDS $\mathcal{M}$ in Example 1. Starting from $\langle 1, 1\rangle$, we have

$$\langle 1,1\rangle \hookrightarrow \langle 2,00\rangle \hookrightarrow \langle 4,000\rangle \hookrightarrow \langle 1,300\rangle \hookrightarrow \langle 1,10\rangle \hookrightarrow \cdots$$

We observe that $\langle 4,000\rangle$ is pumpable by $\langle 2,00\rangle$, and $\langle 1,300\rangle$ and $\langle 1,10\rangle$ are pumpable by the root $\langle 1,1\rangle$. The reduced run $\langle 1,1\rangle \hookrightarrow \langle 2,00\rangle \hookrightarrow \langle 4,000\rangle$ implies non-termination of $\mathcal{M}$. □

The intuition of pumpable nodes is that if the run from $\langle p, w\rangle$ to $\langle q, v\rangle$ only changes the top element of w, then we can simulate this run from $\langle q, v\rangle$ to some $\langle q', v'\rangle$ by monotonicity, satisfying $p \preceq q \preceq q'$, and $w[1] \leq v[1] \leq v'[1]$. We can construct an infinite run by repeating this process.

Conversely, assume $\langle p_0, w_0\rangle \hookrightarrow \langle p_1, w_1\rangle \cdots$ is an infinite run, we can extract an infinite subsequence, say $\langle p_{i_0}, w_{i_0}\rangle, \langle p_{i_1}, w_{i_1}\rangle, \cdots$, such that each node is chosen if it has the minimal depth of the stack in its suffix run. Note that each pair of $\langle p_{i_k}, w_{i_k}\rangle$ and $\langle p_{i_j}, w_{i_j}\rangle$ with $k < j$ in this subsequence satisfies the first condition of *pumpable nodes*. By the fact that the set of heads is well-quasi-ordered with respect to $\trianglelefteq$, it must contain a *pumpable node*.

Theorem 2. *A WSPDS has an infinite run if, and only if, its reduced reachability tree contains a pumpable node.*

3.2 Boundedness problem

The boundedness asks whether the reachability set is finite. We know that any infinite run has a pumpable node. If a pumpable node is exactly the same as the one that pumps it, still an infinite run keeps the reachability set finite. Otherwise, a strict WSPDS enlarges reachable configurations infinitely.

Definition 6. *A node $s : \langle p, w \rangle$* strictly pumps *a node $t : \langle q, v \rangle$ if s pumps t, and either $|w| < |v|$ or $h(\langle p, w \rangle) \lhd h(\langle q, v \rangle)$.*

Example 3. In Example 2, all the pumpable nodes are strictly pumpable nodes. We can conclude the unboundedness.

Theorem 3. *A strict WSPDS has an infinite reachability set if, and only if, its reduced reachability tree contains a strictly pumpable node.*

Similar to the termination problem, theorem 3 derives the decidability of the boundedness problem for a strict WSPDS. The proof (Appendix A) is similar to that of Theorem 2. We need the strictness of a WSPDS for the proof because a partial order enables us to conclude $a < b$ from $a \neq b$ and $a \leq b$, and only strictly monotonic transition rules guarantee the strict growth of configurations.

Theorem 2 (Theorem 3) provides an algorithm to decide the termination (boundedness) of WSPDS (strict WSPDS). Algorithm 1 is for termination, in which we generate and check the reachability tree by depth-first searches. The algorithm for boundedness is similar.

The correctness of this algorithm is straightforward. One thing needs to be mentioned is that we cannot stop when meeting nodes with labels already in the tree since the same nodes in different branches have different ancestors, and the pumpability checking depends on all the ancestor nodes.

4 Post*-automata algorithm

The Post*-automata for WSPDSs are the same as those for PDSs, but the former may be infinite. In this section we give alternative algorithms for the termination and boundedness problems based on Post*-automata construction.

We first introduce the notion of the *dependency* [4] to transitions of Post*-automata. We then give the notion of *pumpable transitions* in *Post**-automata, corresponding to the notion of pumpable nodes in reduced reachability trees.

4.1 Dependency relation

The *dependency* is a binary relation $\Rrightarrow$ among transitions of a Post*-automaton and is generated during *Post**-saturation steps. Starting from the empty set $\emptyset$ and we add new dependency relations by the following rules:

1. (inter) If a transition $p' \overset{\gamma'}{\mapsto} q$ is added by a rule $(p, \gamma \to p', \gamma')$ and a transition $p \overset{\gamma}{\mapsto} q$, then add $(p \overset{\gamma}{\mapsto} q) \Rrightarrow (p' \overset{\gamma'}{\mapsto} q)$.

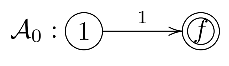
Algorithm 1 Reduced Reachability Tree Algorithm

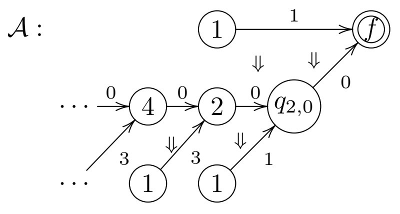
```
Input: $\mathcal{M} = \langle (P, \preceq), (\Gamma, \leq), \Delta, c_0 \rangle$,
Output: If $\mathcal{M}$ terminates, return YES; otherwise, return NO.
 1: $nodes = \{root : c_0\}$
 2: while $nodes != NULL$ do
 3:     currentNode = nodes.pop()
 4:     for all $r \in \Delta$ do
 5:         nodes.push(computeChild(currentNode, r))
 6:     end for
 7:     minLength = currentNode.stack.length
 8:     ancestor = currentNode.father
 9:     while $ancestor != Null$ do
10:         len = ancestor. stack.length
11:         if $minLength \geq len$ then
12:             if currentNode.State≥ ancestor.State                 then
                    &&(currentNode.stack.top ≥ ancestor.stack.top
                            || ancestor.stack = Null)
13:                 return NO.
14:             end if
15:             minLength = len
16:         end if
17:         ancestor = ancestor.father
18:     end while
19: end while
20: return YES
```

2. (nonstandard pop) If a transition $p' \overset{\gamma}{\mapsto} q$ is added by a rule $(p, \alpha\beta \to p', \gamma)$ and transitions $p \overset{\alpha}{\mapsto} p'' \overset{\beta}{\mapsto} q$, then add $(p'' \overset{\beta}{\mapsto} q) \Rrightarrow (p' \overset{\gamma}{\mapsto} q)$.
3. (push) If transitions $p' \overset{\alpha}{\mapsto} q_{p',\alpha} \overset{\beta}{\mapsto} q$ are added by a rule $(p, \gamma \to p', \alpha\beta)$ and a transition $p \overset{\gamma}{\mapsto} q$, then add $(p \overset{\gamma}{\mapsto} q) \Rrightarrow (p' \overset{\alpha}{\mapsto} q_{p',\alpha})$ and $(p \overset{\gamma}{\mapsto} q) \Rrightarrow (q_{p',\alpha} \overset{\beta}{\mapsto} q)$.
4. Otherwise, we do not update $\Rrightarrow$.

We denote the transitive closure of $\Rrightarrow$ by $\Rrightarrow^*$.

Example 4. Recall the WSPDS M in Example 1. Let $C_0 = \{c_0 = \langle 1, 1\rangle\}$, the *Post**-saturation starting from $\mathcal{A}_0$ is illustrated in the following graph, where the $\Downarrow$ represents the dependency relation between transitions.





The transitions $2 \overset{0}{\mapsto} q_{2,0} \overset{0}{\mapsto} f$ are generated from $1 \overset{1}{\mapsto} f$ by the push rule r_1. Simultaneously, we add both $1 \overset{1}{\mapsto} f \Rrightarrow 2 \overset{0}{\mapsto} q_{2,0}$ and $1 \overset{1}{\mapsto} f \Rrightarrow q_{2,0} \overset{0}{\mapsto} f$.

The transition $4 \overset{0}{\mapsto} 2$ is added by the simple push rule r_2 and no new dependency pairs added.

The transition $1 \overset{1}{\mapsto} q_{2,0}$ is added from transitions $1 \overset{3}{\mapsto} 2 \overset{0}{\mapsto} q_{2,0}$ by the nonstandard pop rule r_4. We add $2 \overset{0}{\mapsto} q_{2,0} \Rrightarrow 1 \overset{1}{\mapsto} q_{2,0}$. Note that we do not add dependency between $1 \overset{3}{\mapsto} 2$ and $1 \overset{1}{\mapsto} q_{2,0}$ because this nonstandard pop rule decreases the stack and we only add the dependency relation $t_1 \Rrightarrow t_2$ only when t_2 is generated from t_1, and has equal or longer distance to the final states. □

Lemma 1 shows that dependency relations in Post*-automata reflect the transitions among configurations of the WSPDS. It plays a key role in using Post*-automata techniques to prove the decidability of termination and boundedness for WSPDSs. Proof of Lemma 1 can be found in Appendix B.

Lemma 1. *If $p \overset{\gamma}{\mapsto} q \Rrightarrow^* p' \overset{\gamma'}{\mapsto} q'$ for $p, p' \in P$, there exists $v \in \Gamma^*$ such that $q' \overset{v}{\mapsto}^* q$ and $\langle p, \gamma \rangle \hookrightarrow^* \langle p', \gamma' v \rangle$ in which no simple push transitions appear.*

4.2 Termination problem

To give the notion of a *pumpable transition*, we need to characterize the property that *the content of stack never goes below some depth* in a Post*-automaton. Lemma 1 tells us that if two transitions has dependency relation, then there exists a run that will never let the stack goes below the initial depth. An exception of Lemma 1 is a simple push transition, which interrupts the dependency relation and the depth of stack simply grows. We separately consider these two cases.

Definition 7. *A transition $p \overset{\gamma}{\mapsto} q$ for $p \in P$ is* pumpable *if either*

1. *there exists a path from q to some $p' \preceq p$, i.e., $p \overset{\gamma}{\mapsto} q \overset{w}{\mapsto}^* p'$, or*
2. *there exist a transition $p' \overset{\gamma'}{\mapsto} q'$ with $p' \preceq p, \gamma' \leq \gamma$ and a path $p \overset{\gamma}{\mapsto} q \overset{w}{\mapsto}^* q'$ that contains a transition $p'' \overset{\gamma''}{\mapsto} q''$ with $p' \overset{\gamma'}{\mapsto} q' \Rrightarrow^+ p'' \overset{\gamma''}{\mapsto} q''$.*

Condition 1) in Definition 7 describes the case that $\langle p', \epsilon \rangle$ pumps $\langle p, \gamma w \rangle$ in the reachability tree. The run starts with a simple push transition (which solely can cause p' to be a destination state of $\mapsto$) and never pops stack contents. In this case, we do not need to consider the top element.

Alternatively, the run may start with a push or an internal rule, which relies on the top element of the stack. Condition 2) in Definition 7 corresponds to the case that $\langle p', \gamma' \rangle$ pumps $\langle p, \gamma w \rangle$ with $p' \preceq p, \gamma' \preceq \gamma$ and the first rule is not a simple push rule. In the saturation process, the dependency relation between $p' \overset{\gamma'}{\mapsto} q'$ and $p \overset{\gamma}{\mapsto} q$ may be interrupted by a simple push transition. However, we can take some transition $p'' \overset{\gamma''}{\mapsto} q''$ in the path $p \overset{\gamma}{\mapsto} q \overset{w}{\mapsto}^* q'$ such that the dependency relation ($p' \overset{\gamma'}{\mapsto} q' \Rrightarrow^+ p'' \overset{\gamma''}{\mapsto} q''$) holds before the simple-push operation.

Example 5. Recall the *Post**-automaton in Example 4. The following pumpable transitions imply nontermination.

- The transition $4 \overset{0}{\mapsto} 2$ satisfies 1) in Definition 7.
- The transition $1 \overset{3}{\mapsto} 2$ pumped by transition $1 \overset{1}{\mapsto} f$. It satisfies condition 2) of Definition 7, since during the path $1 \overset{3}{\mapsto} 2 \overset{0}{\mapsto} q_{2,0} \overset{0}{\mapsto} f$, there exists $2 \overset{0}{\mapsto} q_{2,0}$ such that $1 \overset{1}{\mapsto} f \Rrightarrow 2 \overset{0}{\mapsto} q_{2,0}$. After that the dependency sequence is interrupted by a simple push on state 2 (generating $4 \overset{0}{\mapsto} 2$). □

A *reduced Post***-automaton* is obtained by avoiding the application of saturation rules when the saturation procedure reaches a pumpable transition. Lemma 2 reflects the correspondence between a pumpable transition in a Post*-automaton and a pumpable node in the reachability tree. Because of the properties of WQOs, either a pumpable transition or a cycle of $\mapsto$ will be found in finitely many steps. Details and proofs can be found in Appendix C.

Lemma 2. *Given a WSPDS $M = \langle (P, \preceq), (\Gamma, \leq), \Delta, c_0 \rangle$ with an initial configuration c_0 such that $\mathcal{A}_0$ accepts c_0. There exists a pumpable node in the reachability tree if, and only if, there exists a pumpable transition in Post$^*(\mathcal{A}_0)$.*

Theorem 4. *For a WSPDS and an initial configuration c_0, a reduced Post*-automaton finitely converges. Moreover, it converges within k steps of the saturation, where k is the size of the reduced reachability tree rooted at $r : c_0$.*

4.3 Boundedness problem

Similar to the reduced reachability tree, we focus on strict WSPDSs for boundedness.

Definition 8. *A transition $p \overset{\gamma}{\mapsto} q$ where $p \in P$ is* strictly pumpable *if it satisfies the* pumpable *conditions in Definition 7 either by Condition 1), or Condition 2) with an additional condition that $p' \prec p$, $\gamma' \prec \gamma$, or $p'' \neq p$.*

Similar to termination, a *strictly reduced Post***-automaton* avoids applying saturation rules on strictly pumpable transitions. Theorem 5 shows the decidability of boundedness for strict WSPDSs.

Theorem 5. *The strictly reduced Post*-automaton of a strict WSPDS finitely converges.*

Example 6. Recall the *Post**-automaton in Example 4. The two pumpable transitions of Example 5 are strictly pumpable transitions, which implies unboundedness. □

Theorem 4 and 5 provide alternative algorithms for the termination and boundedness problems. Algorithm 2 is for termination checking and the one for boundedness checking is similar.

In Algorithm 2 we store transitions that has been saturated and to be saturated in *rel* and *trans* respectively. Every transition in *trans* is checked the pumpability and then applied saturation rules. Note that we assume preprocessing on WSPDS by rewriting all pop rules to non-standard pop rules in order to removing ϵ transitions. Hence there are four types of rules to be considered (from Line 18 to Line 38), and Line 29 is especially for the case that transition (p', γ', p) has been put into *rel*, but the newly added transition (p, γ, q) makes the saturation on $p' \xmapsto{\gamma'\gamma} q$ possible. The correctness is guaranteed by Lemma 3 and the proof is in Appendix D.

Lemma 3. *If $\mathcal{M}$ has an infinite run, Algorithm 2 will return NO; otherwise, it will return YES, and $(p, \gamma, q) \in rel$ if and only if $p \xmapsto{\gamma} q$ is an edge in Post$^*(\mathcal{A}_0)$.*

5 Complexity issues

In [11], Hyper-Ackermannian upper and lower bounds for the size of reduced reachability trees are shown for pushdown VASS. They estimate the maximum length of bad nested sequences, which is the height of reachability trees in the worst case. Since the reduced reachability tree for a pushdown VASS is finitely branching, the upper bound for the size of the reduced reachability tree can be obtained from the upper bound of its height. For a lower bound, they construct a family of pushdown VASS $\{\mathcal{A}_n\}$ each of which computes a fast growing function $F_{\omega^\omega}(n)$ and terminates. Since each transition rule increments at most one, the reachability tree has at least $F_{\omega^\omega}(n)$ nodes.

In this section, we generalize their methodology to bounded WSPDSs, in which both states and stack symbols are vectors and transition rules can change them by a constant (see Definition 9). BVAS [6], RVASS[3], multi-set PDS[10,16], and Pushdown VASS[11] are all subclasses of bounded WSPDSs. By estimating the size of a reduced Post*-automaton and that of the reduced reachability tree, we obtain their Hyper-Ackermannian upper and lower bounds together.

Following Section V of [11], we define a class of fast growing functions by $(F_\lambda)_\lambda$ for an ordinal $\lambda \leq \omega^\omega$, defined as

$$F_\lambda(n) = \begin{cases} n+1 & \text{if } \lambda = 0 \\ F_{\lambda'}^{n+1}(n) & \text{if } \lambda = \lambda' + 1 \\ F_{\lambda_n}(n) & \text{if } \lambda < \omega^\omega \text{ is a limit ordinal} \\ F_{\omega^{n+1}}(n) & \text{if } \lambda = \omega^\omega \end{cases}$$

where $\lambda_n = \omega^d a_d + \cdots + \omega^r(a_r - 1) + \omega^{r-1}(n+1)$ for a limit ordinal $\lambda = \omega^d a_d + \cdots + \omega^r a_r$ in its Cantor normal form with $d \geq r$ and $a_r > 0$. Here, F_ω is Ackermann function, and we use a Hyper-Ackermann function F_{ω^ω} to estimate the complexity.

Definition 9. *Let $k, d \in \mathbb{N}$. A WSPDS $\mathcal{A} = \langle (\mathbb{N}^d, \leq), (\mathbb{N}^k, \leq), \triangle, c_0 \rangle$ with the initial configuration $c_0 = \langle p_0, w_0 \rangle$ is* bounded *if each rule $(p, w, q, v) \in \Delta$ with $v \neq \epsilon$ satisfies*

$$\|q - p\|_\infty \leq 1 \quad \textit{and} \quad \|\Sigma(v) - \Sigma(w)\|_\infty \leq 1$$

Algorithm 2 The Post*-automata Algorithm

```
Input: $\mathcal{A}_0 = (Q, \Gamma, \nabla_0, \emptyset, F)$
Output: If $\mathcal{M}$ terminates, return YES; otherwise, return NO.
 1: procedure CHECK($(p, \gamma, q)$, rel, dep)
 2:     for $q \mapsto^* p'$ do
 3:         if $p' \preceq p$ then return NO
 4:         end if
 5:     end for
 6:     for $(p', \gamma', q') \Rrightarrow^+ (p, \gamma, q)$ do
 7:         if $(p, \gamma) \geq (p', \gamma')$ then return NO
 8:         end if
 9:     end for
10: end procedure
11: $trans, rel, Q' := (\nabla_0) \cap (P \times \Gamma \times Q), (\nabla_0) \setminus trans, Q$;
12: while $trans \neq \emptyset$ do
13:     pop $t = (p, \gamma, q)$ from trans;
14:     CHECK(t, $rel \cup t$, dep);
15:     if $t \in rel$ then continue
16:     end if
17:     $rel := rel \cup t$;
18:     for all $(p, \epsilon \to p', \gamma') \in \Delta$ do
19:         $trans := trans \cup (p', \gamma', p)$;
20:     end for
21:     for all $(p, \gamma \to p', \gamma') \in \Delta$ do
22:         $trans := trans \cup \{(p', \gamma', q)\}$;
23:         $dep := dep \cup \{t \Rrightarrow (p', \gamma', q)\}$;
24:     end for
25:     for all $(p, \gamma\gamma' \to p', \alpha) \in \Delta$, $(q, \gamma', q') \in rel$ do
26:         $trans := trans \cup \{(p', \alpha, q')\}$;
27:         $dep := dep \cup \{(q, \gamma', q') \Rrightarrow (p', \alpha, q')\}$;
28:     end for
29:     for all $(p', \gamma'\gamma \to p'', \alpha) \in \Delta$, $(p', \gamma', p) \in rel$ do
30:         $trans := trans \cup \{(p'', \alpha, q)\}$;
31:         $dep := dep \cup \{t \Rrightarrow (p'', \alpha, q)\}$;
32:     end for
33:     for all $(p, \gamma \to p', \alpha\beta) \in \Delta$ do
34:         $Q' := Q' \cup \{q_{p',\alpha}\}$;
35:         $trans := trans \cup \{(p', \alpha, q_{p',\alpha})\}$;
36:         $rel := rel \cup \{(q_{p',\alpha}, \beta, q\}$;
37:         $dep := dep \cup$
                $\{t \Rrightarrow (p', \alpha, q_{p',\alpha}), t \Rrightarrow (q_{p',\alpha}, \beta, q)\}$;
38:     end for
39: end while
40: return YES.
```

where $\|\boldsymbol{m}\|_\infty$ denotes the largest component for a vector $\boldsymbol{m}$, and $\Sigma(w)$ denotes $\Sigma_{i\in\{1..n\}}\alpha_i$ if $w = \alpha_1 \cdots \alpha_n$ and $\boldsymbol{0}$ if $w = \epsilon$.

The size of $\mathcal{A}$ is defined as

$$|\ \mathcal{A}\ | = d + k + (d+k)\cdot max\{\|p_0\|_\infty, \|\Sigma(w_0)\|_\infty\} + (d+k)\cdot|\triangle|.$$

5.1 Upper bound

Each node (configuration) of a reachability tree for a WSPDS can be abstracted to a pair consisting of the head and the depth of the stack. With this abstraction, a path in a reachability tree of a WSPDS is a nested sequence [11]. Upper bound for the height of the reduced reachability tree equals the maximal length of bad nested sequences. We give only main results here and leave the formal definitions in Appendix E.

Theorem 6. *(Theorem VI.1 in [11]) For $d \geq 1$ and $n \geq 2$, $L_{\mathbb{N}^d}(n) \leq F_{\omega^d}(d\cdot n)$, where $L_{\mathbb{N}^d}(n)$ is the maximal length of n-controlled bad nested sequences over $\mathbb{N}^d$.*

Theorem 6 generalizes to bounded WSPDSs, since runs of bounded WSPDSs are n-controlled nested sequences.

Each run of a bounded WSPDS $\mathcal{A} = \langle(\mathbb{N}^d, \leq), (\mathbb{N}^k, \leq), \triangle\rangle$ with an initial configuration $\langle p_0, w_0\rangle$ can be abstracted to a nested sequence over $(N^{d+k}, \trianglelefteq)$. We observe that this nested sequence is n-controlled for $n = max\{\|p_0\|_\infty, \|\Sigma(w_0)\|_\infty\}+2$ (Lemma 7 in Appendix E). Thus, Theorem 6 implies that the height of the reduced reachability tree is at most $F_{\omega^{d+k}}((d+k)\cdot n)$. Since each node of the reduced reachability tree can have at most $|\Delta|$ children, we have an upper bound $|\Delta|^{F_{\omega^{d+k}}((d+k)\cdot n)+1}$ for its size, which is also bounded by $F_{\omega^\omega}(|\ \mathcal{A}\ |))$.

Theorem 7. *The reduced reachability tree of a bounded WSPDS has at most $F_{\omega^\omega}(|\ \mathcal{A}\ |)$ nodes (thus at most $F_{\omega^\omega}(|\ \mathcal{A}\ |)$ edges).*

From Theorem 4, the number of saturation steps of a reduced Post*-automaton is bounded by the size of the reduced reachability tree. Since each saturation step adds at most two transitions, the number of transitions is bounded by $2|\Delta|^{F_{\omega^{d+k}}((d+k)\cdot n)+1} \leq F_{\omega^\omega}(|\ \mathcal{A}\ |)$.

Corollary 1. *The reduced Post*-automaton of a bounded WSPDS A has at most $F_{\omega^\omega}(|\ \mathcal{A}\ |)$ transitions.*

5.2 Lower bound

Theorem 8. *(Theorem VII.7 [11]) For each $n \in \mathbb{N}$, there exists a Pushdown VASS $\mathcal{A}_n$ of the size quadratic to n, such that the reduced reachability tree of $\mathcal{A}_n$ has at least $F_{\omega^\omega}(n)$ nodes.*

The result of $F_{\omega^\omega}(n)$ is stored in the first coordinate of its states of $\mathcal{A}_n$, which is initially 0. Since each transition of $\mathcal{A}_n$ changes one coordinate by at most 1, the Post*-automaton of $\mathcal{A}_n$ has at least $F_{\omega^\omega}(n)$ states, and shares a Hyper-Ackermannian lower bound.

Corollary 2. *For $n \in \mathbb{N}$, there exists a Pushdown VASS $\mathcal{A}_n$ of the size quadratic to n, such that the reduced Post*-automaton of $\mathcal{A}_n$ has at least $F_{\omega^\omega}(n)$ states.*

Remark 2. Since a Pushdown VASS $\mathcal{A}_n$ is a subclass of bounded WSPDSs, the lower bound for the size of the reduced reachability tree for $\mathcal{A}_n$ works for bounded WSPDSs as well.

6 Implementations and Experiments

Experiments are performed on a Windows 7 station with 2.60 GHz Intel(R) Core i5 with 8GB of RAM [5]. Experimental data are randomly generated bounded WSPDSs by setting

– control states to be natural numbers,
– stack symbols to be natural number vectors of dimension 1 to 3,
– an initial configuration $\langle p, \gamma \rangle$ taken from $[0, 15] \times [0, 15]^{dim}$ where $dim \in \{1, 2, 3\}$, and
– 1 to 10 transition rules (the number is also randomly decided), which are randomly chosen from internal, non-standard pop, pop, push, and simple push rules (in a VAS like style, e.g., a push rule $(p, \mathbf{v} \to p+q, (\mathbf{v}+\mathbf{c})\ (\mathbf{v}+\mathbf{d}))$ and a non-standard pop rule $(p, \mathbf{v}_1\ \mathbf{v}_2 \to p+q, \mathbf{v}_1+\mathbf{v}_2+\mathbf{c})$, in which each constant and element of a constant vector are randomly taken from $[-10, 10]$.

Fig.6. (a), (b), and (c) compare between the execution time of the two algorithms on randomly generated 1000 bounded WSPDSs with 1-, 2-, and 3-dimensional vectors as stack alphabet respectively. The lines of **Log(ReachTime)** and **Log(PostTime)** plot the execution time (omitting time for IO) of the reachability tree and Post*-automata algorithms in the logarithmic scale respectively. The zigzag of line **Log(PostTime)** is due to the rounding. The instances are sorted by the execution time of the Post*-automata algorithm.

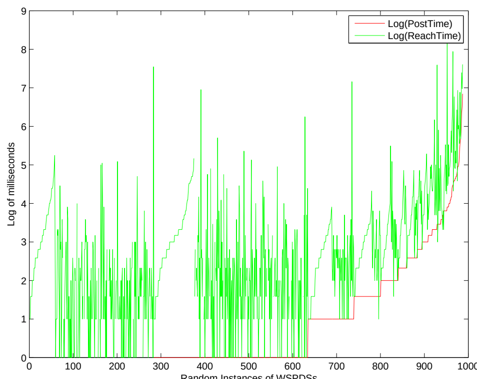
Some cases that significantly differ on complex instances of WSPDSs are extracted to Table 1, in which the columns mean

– **PostTime** (resp. **ReachTime**) is the execution time of the Post*-automata algorithm (resp. the reachability tree algorithm) in milliseconds.
– **EdgeNum** (resp. **NodeNum**) is the number of checked transitions in the generated *Post**-automaton (resp. the reduced reachability tree).

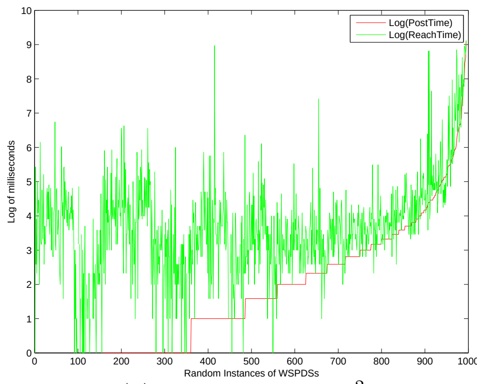
We observe that the *Post**-automata algorithm shows more stable behavior.

– Both algorithms stops quickly on most of problems.

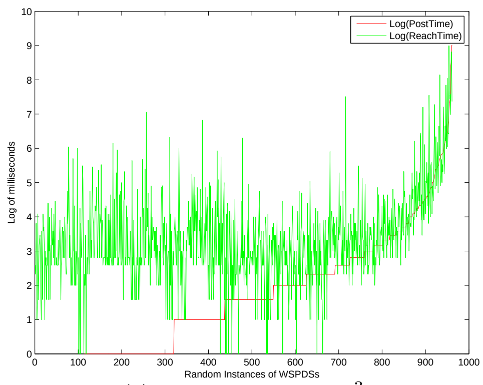
[5] source code can be found at: `https://github.com/leisuhua/WSPDS`



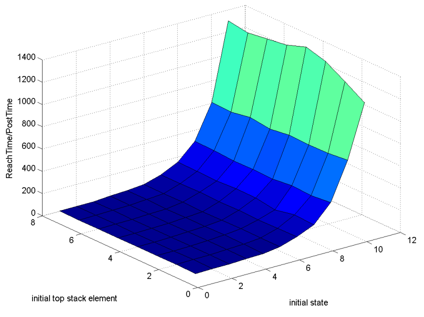
(a) $P = \mathbb{N}$ and $\Gamma = \mathbb{N}$



(b) $P = \mathbb{N}$ and $\Gamma = \mathbb{N}^2$



(c) $P = \mathbb{N}$ and $\Gamma = \mathbb{N}^3$



(d)

Fig. 1. (a-c)Comparison between two algorithms in the logarithmic scale. (d)Speed ratio of two algorithms with various initial configurations.

– The reachability tree algorithm is sometimes slower in magnitude on complex problems, as in Table 1. The opposite is never observed.

Preliminary experiments on boundedness also show consistent results.

Fig.1(d) describes the effect of the change of the speed ratio when initial configurations are modified. This instance is arbitrary selected from Fig.1(a) (i.e., $\Gamma = N$), varying initial configurations in $[1, 12] \times [1, 8]$. Rules of this instance are:

$$\Delta = \begin{cases} r_1 : p, \epsilon \to p - 10, 2 \\ r_2 : p, v_1 v_2 \to p + 7, v_1 + v_2 + 1 \\ r_3 : p, v \to p - 3, v + 5 \\ r_4 : p, v \to p - 2, v + 9 \\ r_5 : p, v \to p - 6, v + 3 \end{cases}$$

We observe that the increase of the size of initial configurations brings significant slow down of the reachability tree algorithm, compared to the Post*-automata algorithm.

Table 1. Problems showing significant differences

Post*-automata		Reachability tree	
PostTime	EdgeNum	ReachTime	NodeNum
29 (ms)	52	2039 (ms)	2397
105	69	11086	6727
507	271	104291	66138
74	95	17032	18834
523	103	207327	53698
153	58	97231	71464
65	71	44947	44164
276	191	801889	667065
426	299	1570971	158307

7 Conclusion

We studied termination and boundedness on WSPDSs and strict WSPDSs, respectively. We compared two algorithms, the reachability tree algorithm, which is a generalization of that in [11], and the Post*-automata algorithm. Although the complexity estimation is hyper-ackermannian for both, experiments show that the latter behaves better than the former.

Acknowledgment

This research is partially supported by NSFC-JSPS joint project 61511140100, NSFC project 61472238 and JSPS KAKENHI Grant-in-Aid for Scientific Research(B) (15H02684).

References

1. M. Atig, A. Bouajjani, and S. Qadeer. Context-bounded analysis for concurrent programs with dynamic creation of threads. *TACAS 2009*. LNCS 5505 107–123.
2. M. Blondin, A. Finkel, P. McKenzie. Handling Infinitely Branching WSTS. *ICALP (2) 2014*. LNCS 8573 13-25.
3. A. Bouajjani and M. Emmi. Analysis of recursively parallel programs. *POPL 2012*. ACM 203–214.
4. X. Cai and M. Ogawa. Well-structured pushdown systems. *CONCUR 2013*. LNCS 8052 121–136.
5. J. Courtois and S. Schmitz. Alternating vector addition systems with states. *MFCS 2014 (1)*. LNCS 8634 220–231.
6. S. Demri, M. Jurdzinski, O. Lachish, and R. Lazic. The covering and boundedness problems for branching vector addition systems. *J. Comput. Syst. Sci.* (2013), 79(1):23–38.
7. C. Dufourd, A. Finkel, P. Schnoebelen. Reset Nets Between Decidability and Undecidability. *ICALP 1998*. LNCS 1443 103–115.
8. A. Finkel, B. Willems, and P. Wolper. A direct symbolic approach to model checking pushdown systems. *Electr. Notes Theor. Comput. Sci.* (1997), 9:27–37.
9. A. Finkel. A Generalization of the Procedure of Karp and Miller to Well Structured Transition Systems. *ICALP 1987*. LNCS 267 499–508.
10. R. Jhala and R. Majumdar. Interprocedural analysis of asynchronous programs. *POPL 2007*. ACM 339–350.
11. J. Leroux, M. Praveen, and G. Sutre. Hyper-ackermannian bounds for pushdown vector addition systems. *CSL-LICS 2014*. ACM/IEEE 63:1–63:10.
12. J. Leroux, G. Sutre, and P. Totzke. On the coverability problem for pushdown vector addition systems in one dimension. *ICALP (2) 2015*. LNCS 9135 324–336.
13. S. Qadeer and J. Rehof. Context-bounded model checking of concurrent software. *TACAS 2005*. LNCS 3440 93–107.
14. G. Ramalingam. Context-sensitive synchronization-sensitive analysis is undecidable. *Trans. ACM on Programming Languages and Systems* (2000), 22(2):416–430.
15. T.W. Reps and S. Schwoon and S. Jha and D. Melski Weighted pushdown systems and their application to interprocedural dataflow analysis *Science of Computer Programming*(2005), 58:206-263
16. K. Sen and M. Viswanathan. Model checking multithreaded programs with asynchronous atomic methods. *CAV 2006*. LNCS 4144 300–314.

A Proof of Theorem 3

Theorem 3. A strict WSPDS has an infinite reachability set if, and only if, its reduced reachability tree contains a strictly pumpable node.

Proof. **(Only-if)** Assume a strict WSPDS M has an infinite reachability set. Let T be the largest prefix of its reachability tree such that, on each branch, all nodes have distinct labels. The tree T is infinite since every configuration in the reachability set is a node in T.

By König's lemma, it follows that T contains finitely many branches in which all nodes are distinct. Since the reduced reachability tree of M is finite, among finitely many branches, there are two nodes $n : (p, w)$ and $m : (q, v)$ such that they are in the reduced reachability tree and n pumps m.

Thus, $(p, w) \neq (q, v)$ and (p, w) pumps (q, v). By definition of pumpable nodes, we have two cases: (1) $|w| < |v|$, and (2) $|w| = |v|$. In case (2), either $w \ll v$ or $p \prec q$ holds. $w[2, |w|] = v[2, |v|]$ implies either $w[1] < v[1]$ or $p \prec q$. Thus, both cases, n strictly pumps m.

(If) Similar to that of Theorem 2. The path from the root to a strictly pumpable node yields a run

$$(p_0, w_0) \xrightarrow{op_1} \dots \xrightarrow{op_k} (p_k, w_k) \xrightarrow{op_{k+1}} \dots \xrightarrow{op_l} (p_l, w_l)$$

such that (p_k, w_k) strictly pumps (p_l, w_l), which leads an infinite run by iterating the sequence of operations $op_{k+1}, ..., op_l$. As the case analysis, if $p_k \prec p_l$, the resulting infinite run visits infinitely many different states; if $|w| < |v|$, the resulting infinite run enlarges the length of the stack infinitely; if $w[1] < v[1]$, the resulting infinite run enlarges the top element of the stack infinitely.

B Proof of Lemma 1

We denote transitions of a WSPDS and Post*-automata by $\hookrightarrow$ and $\mapsto$, respectively. Let $w \in \Gamma^*$ and

$$con(p, w) = \begin{cases} \langle p, w\rangle & \text{if } p \in P \\ \langle p', \alpha w\rangle & \text{if } p = q_{p',\alpha} \in Q \end{cases}$$

First, we need the following invariants on Post*-automata. The proof is by induction on the saturation steps of Post*-automata (See the proof of Lemma 2 in [4]).

Lemma 4. *If $p \overset{w}{\mapsto}{}^* q$ and $p, q \in P \cup Q$, we have $con(q, \epsilon) \hookrightarrow^* con(p, w)$.*

Instead of Lemma 1, we prove Lemma 1', which generalizes $p, p' \in P$ to $p, p' \in P \cup Q$ for $Q = \{q_{p,\alpha} \mid p \in P, \alpha \in \Gamma\}$.

Lemma 1'. If $p \overset{\gamma}{\mapsto} q \Rrightarrow^* p' \overset{\gamma'}{\mapsto} q'$ for $p, p' \in P \cup Q$, there exists $v \in \Gamma^*$ such that $q' \overset{v}{\mapsto}{}^* q$ and $con(p, \gamma) \hookrightarrow^* con(p', \gamma' v)$ in which no simple push transitions appear.

Proof. We proceed by the induction on the saturation steps of Post*-automata. For $\mathcal{A}_0$, immediate from $\Rrightarrow_0 = \emptyset$. Assume it holds for $\mathcal{A}_i$ where the dependency relation is denoted by $\Rrightarrow_i$. $\mathcal{A}_{i+1}$ is obtained from $\mathcal{A}_i$ by applying a saturation rule once. We have $\Rrightarrow_{i+1} \supseteq \Rrightarrow_i$ and perform the case analysis.

– If the saturation step is by a *simple push* rule, the dependency relation is not updated. The lemma holds immediately.
– If the saturation step is by an *internal* rule $(p_1, \gamma_1, p_2, \gamma_2)$ on $p_1 \overset{\gamma_1}{\mapsto} q_1$, the dependency relation is updated by adding $p_1 \overset{\gamma_1}{\mapsto} q_1 \;\Rrightarrow\; p_2 \overset{\gamma_2}{\mapsto} q_1$ (if not added yet). Assume that we have

$$p \overset{\gamma}{\mapsto} q \Rrightarrow_i^* p_1 \overset{\gamma_1}{\mapsto} q_1 \;\Rrightarrow_{i+1}\; p_2 \overset{\gamma_2}{\mapsto} q_1 \Rrightarrow_i^* p' \overset{\gamma'}{\mapsto} q'$$

By induction hypothesis, there exist v_1 and v_2 with $q' \overset{v_2}{\mapsto}{}^* q_1 \overset{v_1}{\mapsto}{}^* q$, $con(p, \gamma) \hookrightarrow^* \langle p_1, \gamma_1 v_1 \rangle$, and $\langle p_2, \gamma_2 v_1 \rangle \hookrightarrow^* con(p', \gamma' v_2 v_1)$, among which no simple push transitions appear. Finally, the internal rule connects them as

$$con(p, \gamma) \hookrightarrow^* \langle p_1, \gamma_1 v_1 \rangle \hookrightarrow \langle p_2, \gamma_2 v_1 \rangle \hookrightarrow^* con(p', \gamma' v_2 v_1)$$

– If the saturation step is by a *push* rule $(p_1, \gamma_1, p_2, \gamma_2\gamma_3)$ on $p_1 \overset{\gamma_1}{\mapsto} q_1$, the dependency relation is updated by adding $p_1 \overset{\gamma_1}{\mapsto} q_1 \;\Rrightarrow\; p_2 \overset{\gamma_2}{\mapsto} q_{p_2,\gamma_2}$ and $p_1 \overset{\gamma_1}{\mapsto} q_1 \;\Rrightarrow\; q_{p_2,\gamma_2} \overset{\gamma_3}{\mapsto} q_1$ (if not added yet).
First, assume that we have

$$p \overset{\gamma}{\mapsto} q \Rrightarrow_i^* p_1 \overset{\gamma_1}{\mapsto} q_1 \;\Rrightarrow_{i+1}\; p_2 \overset{\gamma_2}{\mapsto} q_{p_2,\gamma_2} \Rrightarrow_i^* p' \overset{\gamma'}{\mapsto} q'$$

By induction hypothesis, there exist v_1, v_2 with $q' \overset{v_2}{\mapsto}{}^* q_{p_2,\gamma_2} \overset{\gamma_3}{\mapsto} q_1 \overset{v_1}{\mapsto}{}^* q$, $con(p, \gamma) \hookrightarrow^* \langle p_1, \gamma_1 v_1 \rangle$, and $\langle p_2, \gamma_2\gamma_3 v_1 \rangle \hookrightarrow^* con(p', \gamma' v_2 \gamma_3 v_1)$ among which no simple push transitions appear. Finally, the push rule connects them as

$$con(p, \gamma) \hookrightarrow^* \langle p_1, \gamma_1 v_1 \rangle \hookrightarrow \langle p_2, \gamma_2\gamma_3 v_1 \rangle \hookrightarrow^* con(p', \gamma' v_2 \gamma_3 v_1)$$

Second, assume that we have

$$p \overset{\gamma}{\mapsto} q \Rrightarrow_i^* p_1 \overset{\gamma_1}{\mapsto} q_1 \;\Rrightarrow_{i+1}\; q_{p_2,\gamma_2} \overset{\gamma_3}{\mapsto} q_1 \Rrightarrow_i^* p' \overset{\gamma'}{\mapsto} q'$$

By induction hypothesis, there exist v_1, v_2 with $q' \overset{v_2}{\mapsto}{}^* q_1 \overset{v_1}{\mapsto}{}^* q$, $con(p, \gamma) \hookrightarrow^* \langle p_1, \gamma_1 v_1 \rangle$, and $\langle p_2, \gamma_2\gamma_3 v_1 \rangle \hookrightarrow^* con(p', \gamma' v_2 v_1)$ among which no simple push transitions appear. Finally, the push rule connects them as

$$con(p, \gamma) \hookrightarrow^* \langle p_1, \gamma_1 v_1 \rangle \hookrightarrow \langle p_2, \gamma_2\gamma_3 v_1 \rangle \hookrightarrow^* con(p', \gamma' v_2 v_1)$$

– If the saturation step is by a *nonstandard pop* rule $(p_1, \gamma_1\gamma_2, p_2, \gamma_3)$ on $p_1 \overset{\gamma_1}{\mapsto} q_2 \overset{\gamma_2}{\mapsto} q_1$, the dependency relation is updated by adding $q_2 \overset{\gamma_2}{\mapsto} q_1 \;\Rrightarrow\; p_2 \overset{\gamma_3}{\mapsto} q_1$ (if not added yet). Assume that we have

$$p \overset{\gamma}{\mapsto} q \Rrightarrow_i^* q_2 \overset{\gamma_2}{\mapsto} q_1 \;\Rrightarrow_{i+1}\; p_2 \overset{\gamma_3}{\mapsto} q_1 \Rrightarrow_i^* p' \overset{\gamma'}{\mapsto} q'$$

By induction hypothesis, there exist v_1, v_2 with $q' \overset{v_2}{\mapsto}{}^* q_1 \overset{v_1}{\mapsto}{}^* q$, $con(p,\gamma) \hookrightarrow^* con(q_2, \gamma_2 v_1)$, and $\langle p_2, \gamma_3 v_1\rangle \hookrightarrow^* con(p', \gamma' v_2 v_1)$ among which no simple push transitions appear. By Lemma 4, we have $con(q_2, \epsilon) \hookrightarrow^* \langle p_1, \gamma_1\rangle$, and the nonstandard pop rule connects them as $con(p,\gamma) \hookrightarrow^* con(q_2, \gamma_2 v_1) \hookrightarrow^* \langle p_1, \gamma_1\gamma_2 v_1\rangle \hookrightarrow \langle p_2, \gamma_3 v_1\rangle \hookrightarrow^* con(p', \gamma' v_2 v_1)$

C Proofs of Lemma 2 and Theorem 4

Lemma 4 shows the correspondence from transitions of Post*-automata to those of a WSPDS. In lemma 5, we give the opposite direction.

Lemma 5. *Let $p, q \in P$ and $\alpha, \gamma \in \Gamma$.*

1. *If $\langle p, \epsilon\rangle \hookrightarrow^* \langle q, v\rangle$, we have $q \overset{v}{\mapsto}{}^* p$.*
2. *If $\langle p, \alpha\rangle \hookrightarrow^* \langle q, \gamma v\rangle$ in which no simple push transitions appear (thus during the run the stack stays nonempty) and $p \overset{\alpha}{\mapsto} q'$, we have $q \overset{\gamma}{\mapsto} q_2 \overset{v}{\mapsto}{}^* q'$ such that either*
   - *$p \overset{\alpha}{\mapsto} q' \ \Rrightarrow^* \ \iota$ for each transition ι in $q \overset{\gamma}{\mapsto} q_2 \overset{v}{\mapsto}{}^* q'$, or*
   - *$v = v_1 \gamma'' v_2$ and $q_2 \overset{v_1}{\mapsto}{}^* p'' \overset{\gamma''}{\mapsto} q'' \overset{v_2}{\mapsto}{}^* q'$ for some $p'' \in P$ and $p \overset{\alpha}{\mapsto} q' \ \Rrightarrow^* \ \iota$ for each transition ι in $p'' \overset{\gamma''}{\mapsto} q'' \overset{v_2}{\mapsto}{}^* q'$.*

Proof. By induction on the length i of $\hookrightarrow^*$. If $i = 0$, immediate. Assume that they hold for i.

1. If $\langle p, \epsilon\rangle \underbrace{\hookrightarrow \cdots \hookrightarrow}_{i \text{ times}} \langle q_i, v_i\rangle \hookrightarrow \langle q, v\rangle$, by induction hypothesis, we have $q_i \overset{v_i}{\mapsto}{}^* p$.
   No matter which rule is applied in $\langle q_i, v_i\rangle \hookrightarrow \langle q, v\rangle$, $q \overset{v}{\mapsto}{}^* p$ holds from the definition of saturation rules.
2. If $\langle p, \alpha\rangle \underbrace{\hookrightarrow \cdots \hookrightarrow}_{i \text{ times}} \langle q_i, \gamma_i v_i\rangle \hookrightarrow \langle q, \gamma v\rangle$ and $p \overset{\alpha}{\mapsto} q'$, by induction hypothesis, we have $q_i \overset{\gamma_i}{\mapsto} q_{i2} \overset{v_i}{\mapsto}{}^* q'$ which satisfies either of two cases below.
   - $p \overset{\alpha}{\mapsto} q' \ \Rrightarrow^* \ \iota$ for each transition ι in $q_i \overset{\gamma_i}{\mapsto}\overset{v_i}{\mapsto}{}^* q'$.
     If $\langle q_i, \gamma_i v_i\rangle \hookrightarrow \langle q, \gamma v\rangle$ is a simple push transition, we have $q \overset{\gamma}{\mapsto} q_i \overset{v}{\mapsto}{}^* q'$ for $v = \gamma_i v_i$. This matches to the second case with $p'' = q_i$, $\gamma'' = \gamma_i$ and $v_1 = \epsilon$.
     Otherwise, by the definition of saturation rules, we have $q \overset{\gamma}{\mapsto}\overset{v}{\mapsto}{}^* q'$ and $p \overset{\alpha}{\mapsto} q' \ \Rrightarrow^* \ \iota$ for each transition ι in $q \overset{\gamma}{\mapsto}\overset{v}{\mapsto}{}^* q'$.
   - $v_i = v_{i1}\gamma'' v_{i2}$ and $q_{i2} \overset{v_{i1}}{\mapsto}{}^* p'' \overset{\gamma''}{\mapsto} q'' \overset{v_{i2}}{\mapsto}{}^* q'$ for some $p'' \in P$ and $p \overset{\alpha}{\mapsto} q' \ \Rrightarrow^* \ \iota$ for each transition ι in $p'' \overset{\gamma''}{\mapsto} q'' \overset{v_{i2}}{\mapsto}{}^* q'$.
     If $\langle q_i, \gamma_i v_i\rangle \hookrightarrow \langle q, \gamma v\rangle$ is a non-standard pop transition with $v_{i1} = \epsilon$, the transition rule is $q_i, \gamma_i \gamma'' \to q, \gamma$. Hence, we have $q \overset{\gamma}{\mapsto} q'' \overset{v_{i2}}{\mapsto}{}^* q'$ and $p \overset{\alpha}{\mapsto} q' \ \Rrightarrow^* \ \iota$ for each transition ι in $q \overset{\gamma}{\mapsto} q'' \overset{v_{i2}}{\mapsto}{}^* q'$.
     Otherwise, we have $v = v_1 \gamma'' v_{i2}$, $q \overset{\gamma v_1}{\mapsto}{}^* p'' \overset{\gamma''}{\mapsto} q'' \overset{v_{i2}}{\mapsto}{}^* q'$, and $p \overset{\alpha}{\mapsto} q' \ \Rrightarrow^* \ \iota$ for each transition ι in $p'' \overset{\gamma''}{\mapsto} q'' \overset{v_{i2}}{\mapsto}{}^* q'$.

Instead of prove Lemma 2 directly, we prove strengthened Lemma 2'.

Lemma 2' Given a WSPDS $M = \langle (P, \preceq), (\Gamma, \leq), \Delta \rangle$ and an initial configuration c_0. Assume that $\mathcal{A}_0$ accepts c_0. Then, in Post$^*(\mathcal{A}_0)$,

1. if $p \overset{\gamma}{\mapsto} q$ is pumpable, there exists w with $q \overset{w}{\mapsto}^* f$ and $c_0 \hookrightarrow c_1 \hookrightarrow^* c_n \hookrightarrow \langle p, \gamma w \rangle$ in the reachability tree such that $\langle p, \gamma w \rangle$ is pumpable;
2. if $p \overset{\gamma}{\mapsto} q$ is not pumpable, for every w with $q \overset{w}{\mapsto}^* f$, there exists $c_0 \hookrightarrow c_1 \hookrightarrow^* c_n \hookrightarrow \langle p, \gamma w \rangle$ in the reachability tree such that $\langle p, \gamma w \rangle$ is not pumpable;

Proof. **(1.)** The proof follows to Definition 7 of pumpable transitions.

– If $p \overset{\gamma}{\mapsto} q$ satisfies the first condition in Definition 7, i.e., there exists some p' s.t. $p \overset{\gamma}{\mapsto} q \overset{w_1}{\mapsto}^* p' \overset{w_2}{\mapsto}^* f$ with $p' \preceq p$. Since c_0 is the only initial configuration, by Lemma 4, we have

$$c_0 \hookrightarrow^* \langle p', w_2 \rangle \hookrightarrow^* \langle p, \gamma w_1 w_2 \rangle,$$

which implies $\langle p', w_2 \rangle$ pumps $\langle p, \gamma w_1 w_2 \rangle$.

– If $p \overset{\gamma}{\mapsto} q$ satisfies the second condition in Definition 7, i.e., there exists $p' \overset{\gamma'}{\mapsto} q' \overset{w_2}{\mapsto}^* f$ with $p' \preceq p, \gamma' \leq \gamma$ and there exists a path $p \overset{\gamma}{\mapsto} q \overset{w}{\mapsto}^* q'$ containing a transition $p'' \overset{\gamma''}{\mapsto} q''$ with $p' \overset{\gamma'}{\mapsto} q' \Rrightarrow^+ p'' \overset{\gamma''}{\mapsto} q''$.
Let $\gamma w = w_1 \gamma'' w_1'$. By Lemma 1 and 4, there exists

$$c_0 \hookrightarrow^* \langle p', \gamma' w_2 \rangle \hookrightarrow^* \langle p'', \gamma'' w_1' w_2 \rangle$$

If $p'' = p$ and $\gamma'' = \gamma$, then $\gamma w = \gamma'' w_1'$ and $\langle p', \gamma' w_2 \rangle$ pumps $\langle p, \gamma w w_2 \rangle$. Otherwise, by Lemma 4, we have $c_0 \hookrightarrow^* \langle p', \gamma' w_2 \rangle \hookrightarrow^* \langle p'', \gamma'' w_1' w_2 \rangle \hookrightarrow^* \langle p, \gamma w_1 \gamma'' w_1' w_2 \rangle = \langle p, \gamma w w_2 \rangle$ Thus, $\langle p', \gamma' w_2 \rangle$ pumps $\langle p, \gamma w w_2 \rangle$.

(2.) We prove by contradiction. Assume that $p \overset{\gamma}{\mapsto} q$ is not pumpable and there exists w with $q \overset{w}{\mapsto}^* f$ and $c_0 \hookrightarrow c_1 \hookrightarrow^* c_n \hookrightarrow \langle p, \gamma w \rangle$ in the reachability tree such that $\langle p, \gamma w \rangle$ is pumped by $c_i = \langle p', w' \rangle$. From Definition 5, we have two cases.

– $\langle p', \epsilon \rangle \hookrightarrow^* \langle p, \gamma w'' \rangle$ with $w = w'' w'$. By Lemma 5, there exists $p \overset{\gamma}{\mapsto} q \overset{w''}{\mapsto}^* p'$. Hence $p \overset{\gamma}{\mapsto} q$ is a pumpable transition, which contradicts to the assumption.
– $\langle p', \gamma' \rangle \hookrightarrow^* \langle p, \gamma w'' \rangle$ with $w' = \gamma' w_1$ and $w = w'' w_1$, and during the run the stack stays nonempty. Also by Lemma 5, $p \overset{\gamma}{\mapsto} q$ is a pumpable transition, which contradicts to the assumption.

Theorem. 4 For a WSPDS and an initial configuration c_0, a reduced *Post**-automaton finitely converges. Moreover, it converges within k steps of the saturation, where k is the size of the reduced reachability tree rooted at $r : c_0$.

Proof. In every step of saturation, we add one or two new transitions. If these new transitions introduce pumpable transitions, the construction of reduced Post*-automaton finishes. Otherwise, they will cover at least one non-pumpable configuration in the reachability tree (Lemma 2') Hence, the number of non-pumpable nodes in the reachability tree bounds the number of saturation steps.

D Correctness of Algorithm 2

Lemma. 3 If $\mathcal{M}$ has an infinite run, Algorithm 2 will return NO; otherwise, it will return YES, and $(p,\gamma,q) \in rel$ if and only if $p \overset{\gamma}{\mapsto} q$ is an edge in Post$^*(\mathcal{A}_0)$.

Proof. Firstly, an overall description of Algorithm 2 is: we use stack *trans* to store transitions in the Post*-automaton that are waiting to be checked separately. Each time we pop a transition from *trans* and check whether it is pumpable. If yes, return NO(means non-terminates) and algorithm stops; otherwise, generate new transitions by the saturation rules from this transition if it is not in *rel*, a set to restore all the transitions that have been checked, and put the new transitions to the stack *trans*. Meanwhile, we update the dependency relation of transitions maintained in a map *dep*. When the stack *trans* become empty, it means we have checked all the transitions in the Post*-automaton, but haven't found any pumpable transition separately. In this case, the algorithm returns YES(means terminates) and stops.

Secondly, we prove that if $\mathcal{M}$ doesn't have infinite run, upon termination of algorithm 2, $(p,\gamma,q) \in rel$ holds for any $p,q \in Q'$ and $\gamma \in \Gamma \cup \{\epsilon\}$ if and only if $p \overset{\gamma}{\mapsto} q \in \nabla$.

By Definition 3, set ∇ satisfies the following:

- If $(p, \epsilon \to p', \gamma') \in \Delta$ and $p \overset{\gamma}{\mapsto} q \in \nabla$, then $p' \overset{\gamma'}{\mapsto} p \in \nabla$.
- If $(p, \gamma \to p', \gamma') \in \Delta$ and $p \overset{\gamma}{\mapsto} q \in \nabla$, then $p' \overset{\gamma'}{\mapsto} q \in \nabla$.
- If $(p, \alpha\beta \to p', \gamma) \in \Delta$ and $p \overset{\alpha}{\mapsto} p'', p'' \overset{\beta}{\mapsto} q \in \nabla$, then $p' \overset{\gamma}{\mapsto} q \in \nabla$.
- If $(p, \gamma \to p', \alpha\beta) \in \Delta$ and $p \overset{\gamma}{\mapsto} q \in \nabla$, then $p' \overset{\alpha}{\mapsto} q_{p',\alpha}, q_{p',\alpha} \overset{\beta}{\mapsto} q \in \nabla$.

(Only-if) we show that if $(p,\gamma,q) \in rel$, then $p \overset{\gamma}{\mapsto} q \in \nabla$. Since transitions from *trans* flow into *rel*, we examine all the lines that change *trans* or *rel*:

- Lines 11 add elements from ∇_0, which is subset of ∇.
- Line 19 is a case of rule $(p, \epsilon \to p', \gamma') \in \Delta$.
- Line 22 is a case of rule $(p, \gamma \to p', \gamma') \in \Delta$.
- Line 26 and 30 are cases of rule $(p, \alpha\beta \to p', \gamma) \in \Delta$.
- Line 35 and 36 are cases of rule $(p, \gamma \to p', \alpha\beta) \in \Delta$.

(If) We show that if $p \overset{\gamma}{\mapsto} q \in \nabla$, then after termination $(p,\gamma,q) \in rel$. Observe that all the transitions in *trans* eventually end up in *rel*. Therefore we only need to prove that all transitions of ∇ added either to *rel* or *trans* during execution of the algorithm.

The desired property is derived from the following facts:

- Because of lines 11, $\nabla_0 \subseteq rel$ holds.
- If $(p, \epsilon \to p', \gamma') \in \Delta$ and $(p,\gamma,q) \in rel$, then (p',γ',p) is added by line 19.
- If $(p, \gamma \to p', \gamma') \in \Delta$ and $(p,\gamma,q) \in rel$, then (p',γ',q) is added by line 22.

– If $(p, \gamma\gamma_1 \rightarrow p', \gamma_2) \in \Delta$ and whenever there is a pair $t_1 = (p, \gamma, q)$, $t_2 = (q, \gamma_1, q') \in rel$ for some $p, q, p' \in Q'$, then we need to add (p', γ_2, q').
  1. If t_1 was examined before t_2, then (p', γ_2, q') is added by line 30.
  2. If t_2 was examined before t_1, t_2 is in rel when t_1 is examined, then (p', γ_2, q') is added by line 26.

– If $(p, \gamma \rightarrow p', \alpha\beta) \in \Delta$ and $(p, \gamma, q) \in rel$, then $(p', \alpha, q_{p',\alpha})$ and $(q_{p',\alpha}, \beta, q)$ are added by line 35 and 36.

E Nested sequences

Definition 10 ([11]). *A nested sequence over a set S is a (finite or infinite) sequence $(s_0, h_0), (s_1, h_1), ...$ of elements in $S \times \mathbb{N}$ satisfying $h_0 = 0$ and $h_j = h_{j-1} + \{-1, 0, 1\}$ for every index $j > 0$ of the sequence.*

For a WSPDS $M = \langle (P, \preceq), (\Gamma, \leq), \Delta \rangle$, every run, i.e., every path in its reachability tree, can be abstracted as a nested sequence over $P \times \{\perp\} \cup \Gamma$, by mapping each configuration (p, w) to the pair $(h(p, w), |w|)$.

Definition 11 ([11]). *A nested sequence $(s_0, h_0), (s_1, h_1), ...$ over a quasi-ordered set $(S, \trianglelefteq)$ is* good *if there exists $i < j$ such that $s_i \trianglelefteq s_j$ and $h_{i+1}, ..., h_j \geq h_i$. A nested sequence is* bad *if it is not good.*

Consider a path in the reachability tree of a *WSPDS*. The nested sequence associated with this path is good, if, and only if it contains a pumpable node.

Definition 12 ([11]). *A* norm *for a WQO set $(P, \preceq)$ is a function $\|\cdot\| : P \mapsto \mathbb{N}$ such that $\{p \in P | \|p\| \leq n\}$ is finite for each n. The structure $(P, \leq, \|\cdot\|)$ is called a* normed WQO*.*

A WQO set $(\mathbb{N}^k, \leq)$ is normed by the function $\|\cdot\|_\infty$ that maps a vector to its largest component.

Definition 13 ([11]). *A nested sequence $(s_0, h_0), (s_1, h_1), ...$ over a normed WQO set $(S, \trianglelefteq)$ is* n-controlled *for $n \in \mathbb{N}$ if $max\{\|s_j\|\} \leq n + j$ for each index j.*

Let $BAD_P(n)$ be the set of n-controlled bad nested sequences over a WQO set $(P, \preceq)$. The *maximal length* function L_P is $L_P(n) = max\{|w| | w \in BAD_P(n)\}$.

Remark 3. Every run of a bounded WSPDS $A = \langle (\mathbb{N}^d, \leq), (\mathbb{N}^k, \leq), \triangle \rangle$ can be seen as a nested sequence, by mapping each configuration (p, w) to the pair $(h(p, w), |w|)$. Recall that $h(p, w) = (p, w[1])$ if $w \neq \epsilon$, and $h(p, w) = (p, \perp)$ otherwise. We will prove in Lemma 6 that the length of a bad nested sequence will not change after mapping $\perp$ to $\mathbf{0}$. $(h(p_0, w_0), |w_0|), (h(p_1, w_1), |w_1|), ...$ is a nested sequence over $\mathbb{N}^d \times \mathbb{N}^k$.

Lemma 6. *Given a bounded WSPDS $A = \langle(\mathbb{N}^d, \leq), (\mathbb{N}^k, \leq), \triangle\rangle$ with an initial configuration (p_0, w_0). The length of a bad nested sequence $(h(p_0, w_0), |w_0|), \ldots,$ $(h(p_i, w_i), |w_i|), \ldots$ associated with the path $(p_0, w_0), \ldots, (p_i, w_i), \ldots$ will not change by mapping $\perp$ to $\mathbf{0}$.*

Proof. Consider one step transition $(p_{i-1}, w_{i-1}) \hookrightarrow (p_i, w_i)$. If $w_i = \epsilon$ (i.e., $w_i[1] = \perp$), the triplet corresponding to (p_i, w_i) is $(p_i, \perp, 0)$. Since $(p_0, w_0[1], |w_0|)$, $\ldots, (p_i, w_i[1], |w_i|), \ldots$ is a bad nested sequence, there are two cases.

- The triplet corresponding to (p_{i-1}, w_{i-1}) is $(p_{i-1}, \perp, 0)$, where $p_{i-1} > p_i$. In this case, the nested sequence is bad after mapping $\perp$ to $\mathbf{0}$.
- The triplet corresponding to (p_{i-1}, w_{i-1}) is $(p_{i-1}, \gamma, 1)$, where $\gamma \in \mathbb{N}^k$ and $\gamma \neq \perp$. Since $|w_{i-1}| > |w_i|$, the mapping $\perp$ to $\mathbf{0}$ keeps sequences bad.

Lemma 7. *Given a bounded WSPDS with an initial configuration (p_0, w_0), the nested sequence $(h(p_0, w_0), |w_0|), (h(p_1, w_1), |w_1|), \ldots$ over $(\mathbb{N}^d \times \mathbb{N}^k, \leq \times \leq)$ is n-controlled for $n = max\{\|p_0\|_\infty, \|\Sigma(w_0)\|_\infty\} + 2$.*

Proof. We define the norm of a WQO $(\mathbb{N}^d \times \mathbb{N}^k, \leq \times \leq)$ as $\|(p_j, w_j[1])\| = max\{\|p_j\|_\infty, \|w_j[1]\|_\infty\}$. We will prove by induction on an index j that

$$\|(p_j, w_j[1])\| \leq n + j$$

for each index j and $n = max\{\|p_0\|_\infty, \|w_0\|_\infty\} + 2$.

- (**Base step.**) $\|(p_0, w_0[1])\| \leq n$ is immediate.
- (**Induction step.**) Assume $\|(p_i, w_i[1])\| \leq n + i$. We show

$$\|(p_{i+1}, w_{i+1}[1])\| \leq n + (i + 1)$$

  by a case analysis on the rule applied in the transition $(p_i, w_i) \hookrightarrow (p_{i+1}, w_{i+1})$. We consider *push rules* $(p_i, \gamma, p_{i+1}, \alpha\beta)$ in detail. The other cases are similar. By the definition of bounded WSPDS, we have $\|p_{i+1} - p_i\|_\infty \leq 1$ and $\Sigma(\alpha\beta) - \Sigma(\gamma)\|_\infty \leq 1$. Hence,

$$\begin{aligned} \|(p_{i+1}, w_{i+1}[1])\| &= max\{\|p_{i+1}\|_\infty, \|w_{i+1}[1]\|_\infty\} \\ &\leq max\{\|p_{i+1}\|_\infty, \|\Sigma(w_{i+1})\|_\infty\} \\ &\leq max\{\|p_i\|_\infty, \|\Sigma(w_i)\|_\infty\} + 1 \\ &\leq n + i + 1 \end{aligned}$$