Abstract—Well-structured pushdown systems (WSPDSs) extend pushdown systems with both 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”[8]. The decidability of most model checking properties, such as termination, reachability, are remained to be open. In this paper, we prove the decidability of termination and boundedness problems for WSPDS using two methods: One is an extension of the reduced reachability tree technique proposed by Leroux et. al. in [8]; and the other is based on Post1-automata technique which has been successfully applied in the model checking of Pushdown systems. We show that the complexity of both are Hyper-Ackermannian for bounded WSPDSs. We design efficient implementation algorithms for both methods and make experiments on the termination detection on a huge number of randomly generated WSPDSs. The results illustrate that the Post1-automata based algorithm is more stable, i.e., both behave more or less the same for simple instances of WSPDS, but the reduced reachability tree algorithm is sometimes an order of magnitude slower for complex instances of WSPDS.1

I. 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.

Well-structured pushdown systems (WSPDSs) [3] are combinations or extensions of these two models. A WSPDS is a pushdown system with both well-quasi-ordered control states and stack alphabet, e.g., control states of WSPDS can be vectors and the stack can store vectors. The coverability has shown to be decidable for some restricted WSPDSs, e.g., finite control states with well-quasi-ordered stack alphabet [3].

Instead of coverability, this paper focuses on termination and boundedness for general WSPDSs. The former asks whether all runs of a given WSPDS are finite, the latter asks whether its reachability set is finite, and both are shown to be decidable for general WSPDSs. Our contributions include:

- The reduced reachability tree method in [8] is generalized to WSPDSs (Section IV).
- The Post1-automata method for termination/boundedness is proposed (Section V).
- We restrict WSPDSs to bounded WSPDSs where control states and stack alphabet are vectors, and induce Hyper-Ackermannian upper and lower bounds for both methods (Section VI).
- The experiments of the termination detection are performed on randomly generated bounded WSPDSs (Section VII). The results show a practical advantage of the algorithm based on Post1-automata.

The reachability tree and the Post1-automata methods 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 (Post1) [6], whose (upper bound) complexity are exponential and cubic, respectively. The latter is typically used for the implementation of pushdown model checking.

Related Work

This paper is inspired by [8]. They also propose the notion of “well-structured pushdown systems” (Definition III.1 in [8]), which are WSPDSs (in this paper) with finite stack alphabet. They use reduced reachability tree technique, which is an extension of Karp-Miller tree technique 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/boundedness for general WSPDSs.

There are several models of concurrent recursive computation, e.g., a branching VAS (BVAS) [5], a recursive VASS (RVASS) [2], and an alternating VASS (AVASS) [4]. All of them are known that coverability is decidable, but reachability is left open except for AVASS. Reachability of an AVASS is undecidable [4] since it requires zero-tests. All of them can be encoded into WSPDSs with finite control states, and coverability of each is also decidable from general arguments on WSPDSs [3]2. For WSPDSs with states being natural numbers (1-dimension vectors) and finite stack alphabet, coverability is decidable [9], while the reachability and coverability for WSPDSs with larger dimension still remain open.

1A part of the results are orally presented at YR-ICALP 2015 (Kyoto, 5 July 2015).

2The reachability problem of Alternating VASS can not be interpreted in WSPDS because it violates monotonicity.
II. PUSHDOWN SYSTEMS AND P-AUTOMATA

A. Pushdown system

Definition II.1. A pushdown system (PDS) is a triplet \( \langle P, \Gamma, \Delta \rangle \) where \( P \) is a set of states, \( \Gamma \) is stack alphabet, and \( \Delta \subseteq P \times \Gamma^{\leq 1} \times P \times \Gamma^{\leq 1} \) is a set of transitions. A transition \( \rightarrow \) between configurations is defined as follows.

\[
\begin{align*}
\text{inter} & : (p, \gamma \rightarrow p', \gamma') \in \Delta \\
\text{push} & : (p, \gamma \rightarrow p', \alpha \beta) \in \Delta \\
\text{pop} & : (p, \gamma \rightarrow p', \epsilon) \in \Delta
\end{align*}
\]

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

For later convenience, we introduce two more forms of rules:\(^1\)

\[
\begin{align*}
\text{simple-push} & : (p, \epsilon \rightarrow p', \alpha \beta) \in \Delta \\
\text{nonstandard-pop} & : (p, \alpha \beta \rightarrow p', \gamma) \in \Delta
\end{align*}
\]

Throughout the paper, we will use the following notation convention.

- \( \alpha, \beta, \gamma, \cdots \) range over \( \Gamma \),
- \( w, v, \cdots \) range over words in \( \Gamma^* \). \( |w| \) 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, \( h(p, w) = (p, \bot) \).
- \( p, q, \cdots \) range over states, and \( c, d, \cdots \) range over configurations.
- We use \( \rightarrow \) in rules, \( \rightarrow \) for transitions between configurations, and \( \triangleright \) for edges (transitions) in Post\(^*\)-automata.
- We denote \( \mathbb{N} \) (resp. \( \mathbb{Z} \)) for the set of natural numbers (resp. integers).

B. Post\(^*\)-automaton

A P-automaton \( [6] \) is an automaton that accepts the set of reachable configurations of a PDS. We focus on the forward reachability set.

Definition II.2. Given a PDS \( \mathcal{M} = \langle P, \Gamma, \Delta, C_0 \rangle \), a P-automaton \( \mathcal{A} \) is a quadruplet \( \langle S, \Gamma, \nabla, F \rangle \) where

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

We write \( \langle s, \gamma, s' \rangle \in \nabla \) (possibly \( \gamma = \epsilon \), and \( \triangleright^* \) for the reflexive transitive closure of \( \triangleright \). \( \mathcal{M} \) accepts \( \langle p, w \rangle \) for \( p \in P \) and \( w \in \Gamma^* \) if \( p \triangleright^* f \in F \). 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 \( C_0 \), the set of initial configurations. The saturation procedure to compute \( \text{post}^*(C_0) \) starts from \( \mathcal{A}_0 \), and repeatedly adds states and edges according to the rules of a PDS convergence.

Definition II.3. For a PDS \( \mathcal{M} = \langle P, \Gamma, \Delta, C_0 \rangle \), let \( \mathcal{A}_0 = (S_0, \Gamma, \nabla_0, F) \) be the initial P-automaton that accepts \( C_0 \). Post\(^*\)(\( \mathcal{A}_0 \)) is the automaton generated by repeated applications of the following Post\(^*\)-saturation rules:

\[
\begin{align*}
&\text{if } p, w \rightarrow p', \gamma \in \Delta: \\
&(S, \Gamma, \nabla, F), p \triangleright^* \gamma \ q \in \nabla \\
&(S \cup \{p', q\}, \Gamma, \nabla \cup \{p' \triangleright q\}, F)
\end{align*}
\]

\[
\begin{align*}
&\text{if } p, \gamma \rightarrow p', \alpha \beta \in \Delta: \\
&(S, \Gamma, \nabla, F), p \triangleright \alpha \beta \ q \in \nabla \\
&(S \cup \{p', q\}, \Gamma, \nabla \cup \{p' \triangleright_\alpha q\}, F)
\end{align*}
\]

For instance, consider a push rule \( (p, \gamma \rightarrow p', \alpha \beta) \). If \( p \triangleright q \) is \( \in \nabla \), then \( p' \triangleright_\alpha q \rightarrow_\beta q \in \nabla \). The intuition is, if, for \( v \in \Gamma^* \), \( \langle p, \gamma v \rangle \) is in \( \text{post}^*(C_0) \), then \( \langle p', \alpha \beta v \rangle \) is also in \( \text{post}^*(C_0) \) by applying rule \( (p, \gamma \rightarrow p', \alpha \beta) \).

Remark. 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. When we consider \( P \) and \( \Gamma \) to be infinite, Post\(^*\)(\( \mathcal{A}_0 \)) may not finally converge. However, it has a limit \( \bigcup_1 \text{Post}(\mathcal{A}_0) \) (by taking set unions of states and transitions, which monotonically increase). Note that Theorem 1 holds under such generalization.

Theorem 1. \([6]\) Post\(^*\)(\( C_0 \)) = \( L(\text{Post}^*(\mathcal{A}_0)) \).

Remark. Post\(^*\)-saturation introduces \( c \)-transitions when applying standard pop rules. To avoid it, we apply the following preprocessing on a WSPDS.

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

III. WELL-STRUCTURED PUSHDOWN SYSTEMS

A. Well-quasi-orders

A quasi-order \( (S, \preceq) \) is a reflexive transitive binary relation on a set \( S \). We denote \( s < t \) 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 \( s_1, s_2, s_3, \ldots \) 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 \( a_1 a_2 \ldots a_n \leq b_1 b_2 \ldots b_n \), if \( m = n \) and, for each \( i, a_i \leq b_i \) holds, and \( w < v \) if \( w < v \) and \( w \neq v \). Note that \( \leq \) may not be a WQO for a WQO \( \leq \).
B. Well-structured pushdown systems

Well-structured pushdown systems (WSPDSs) are pushdown systems such that the sets of states and stack symbols can be infinite but well-quasi-ordered. Let $\mathcal{F}\{X,Y\}$ denote the set of partial functions from set $X$ to set $Y$.

**Definition III.1.** [3] A WSPDS is a triplet $\mathcal{M} = \langle (P, \preceq), (\Gamma, \leq), \Delta \rangle$ where
- $(P, \preceq)$ and $(\Gamma, \leq)$ are WQOs, and
- $\Delta \subseteq \mathcal{F}(P \times \Gamma^{\leq 2}, P \times \Gamma^{\leq 2})$ is the finite set of monotonic partial functions (w.r.t. $\preceq \times \leq$).

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

Note that the set of heads is well-quasi-ordered by $\preceq \times \leq$ if the bottom element $\bot$ satisfies $\bot \leq \alpha$ for each $\alpha \in \Gamma$.

A WSPDS $(P, \preceq), (\Gamma, \leq), \Delta$ is strict if $(P, \preceq)$ and $(\Gamma, \leq)$ are partial order, and $\Delta$ consists of strictly monotonic partial functions.

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

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

$M$ is a WSPDS with both its states and stack symbols being natural numbers. The transitions rules in $\Delta$ 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.

A pushdown VAS [8] is a WSPDS $(\langle \mathbb{N}^k, \preceq \rangle, \Gamma, \Delta)$ with finite stack alphabet $\Gamma$. A branching VAS (BVAS) [5], a recursive VASS (RVASS) [2], and an alternating VASS (AVASS) [4] are WSPDSs $(Q, \langle \mathbb{N}^k, \preceq \rangle, \Delta)$ with the finite set $Q$ of control states such that a BVASS and an RVASS have only non-standard pop, simple push, and internal rules, and an AVASS has only pop, push, and internal rules.

C. Termination and Boundedness problems

A run starting from some initial configuration $c_0$ is a (finite or infinite) sequence $c_0, c_1, \ldots$ of configurations where $c_{i-1} \rightarrow c_i$ for every index $i > 0$. The reachability set of $c_0$ is the set of all configurations that occur on some run from $c_0$.

Given a WSPDS and an initial configuration $c_0$,
- **termination** asks whether each run starting from $c_0$ is finite, and
- **boundedness** asks whether the reachability set of $c_0$ is finite.

Note that since $\Delta$ is finite, if a WSPDS terminates, then it has finite reachability set by König’s lemma.

**Example III.2.** Recall the WSPDS $M$ in Example III.1. Assume $c_0 = (1,1)$, here is an infinite run with an infinite reachability set:

$$(1,1) \xrightarrow{r_1} (2,0) \xrightarrow{r_2} (4,00) \xrightarrow{r_3} (1,300) \xrightarrow{r_4} (1,10) \xrightarrow{r_4} \cdots$$

Note that unboundedness always implies non-termination, but not vice versa. If we change rule $r_1$ to $r'_1 : p, \alpha \rightarrow p + 1, (\alpha - 1)$, this infinite run contains a finite reachability set: $\langle 1,1 \rangle \xrightarrow{r_3} (2,0) \xrightarrow{r_2} (4,00) \xrightarrow{r_3} (1,300) \xrightarrow{r_4} (1,10) \xrightarrow{r_4} \cdots$.

If we further remove the rule $r_2$, all runs starting from $c_0$ terminate.

IV. The reduced reachability tree method

The reachability tree of a WSPDS $M = \langle (P, \preceq), (\Gamma, \leq), \Delta \rangle$ with an initial configuration $c_0$ is a directed unordered tree defined as follows. Each node of the tree is labeled by a configuration of $M$. The root $r$ is labeled by the initial configuration $c_0$, denoted by $r : c_0$. Each node $u : c_m$ has a child $m : c_n$ for each configuration $c_m$ with $c_n \rightarrow c_m$. Note that the reachability tree of $M$ is finitely branching since $\Delta$ is finite.

A. Termination problem

**Definition IV.1.** A node $s : (p,w)$ pumps a node $t : (q,v)$ if
- there is a path from $s$ to $t$, and every node $t' : (p',w')$ on it satisfies $|w'| \geq |w|$.
- $h((p,w)) \preceq h((q,v))$, 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 [8], 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 IV.1.** Recall the WSPDS $M$ in Example III.1. Starting from $(1,1)$, we have

$$(1,1) \xrightarrow{r_1} (2,00) \xrightarrow{r_2} (4,000) \xrightarrow{r_4} (1,300) \xrightarrow{r_4} (1,10) \xrightarrow{r_4} \cdots$$

We observe that $(4,000)$ is pumpable by $(2,00)$, and $(1,300)$ and $(1,10)$ are pumpable by the root $(1,1)$. The reduced run $(1,1) \rightarrow (2,00) \rightarrow (4,000)$ implies non-termination of $M$.

The intuition of pumpable nodes is that if the run from $(p,w)$ to $(q,v)$ only changes the top element of $w$, then we can simulate this run from $(q,v)$ to some $(q',v')$ 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 $\langle p_0, w_0 \rangle \xrightarrow{r_1} \langle p_1, w_1 \rangle \cdots$ is an infinite run, we can extract an infinite subsequence, say $\langle p_{i_k}, w_{i_k} \rangle, \langle p_{i_j}, w_{i_j} \rangle, \cdots$, such that each node is chosen if it has the minimal depth of the stack in its suffix run. Note that each pair of $(p_{i_k}, w_{i_k})$ and $(p_{i_j}, w_{i_j})$ 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 $\preceq$, 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.
B. Boundedness problem

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, this infinite run keeps the reachability set finite. Otherwise, it derives infinitely many reachable configurations, if the WSPDS is strict.

Definition IV.2. A node $s : (p,w)$ strictly pumps a node $t : (q,v)$ if $s$ pumps $t$, and either $|w| < |v|$ or $h((p,w)) \neq h((q,v))$.

Example IV.2. In Example IV.1, 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.

V. POST*-AUTOMATA METHOD

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

We first introduce the notion of dependency to the 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.

A. Dependency relation

The dependency is a binary relation $\Rightarrow$ among transitions of a POST*-automaton and is generated during POST*-saturation steps. The dependency relations between transitions of POST*-automaton can reflect the transition of configurations of a WSPDS as shown in Lemma 1.

The dependency relation for a WSPDS is initially set to the empty set $\emptyset$ and we add new dependency relations by the following rules:

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

We denote the transitive closure of $\Rightarrow$ by $\Rightarrow^*$. The dependency relation is generated along with the saturation steps.

Example V.1. Recall the WSPDS $M$ in Example III.1. Let $C_0 = \{c_0 = (1,1)\}$, the POST*-saturation starting from $A_0$ is illustrated in the following graph, where the $\downarrow$ represents the dependency relation between transitions.

The transitions $2 \rightarrow 0 \rightarrow q_{2,0} \rightarrow q'$ are generated from $1 \rightarrow f$ by the push rule $r_1$. Simultaneously, we add both $1 \rightarrow f \Rightarrow 2 \rightarrow q_{2,0}$ and $1 \rightarrow f \Rightarrow q_{2,0} \rightarrow f$.

The transition $4 \rightarrow 2$ is added by the simple push rule $r_2$ and no new dependency pairs added.

The transition $1 \rightarrow q_{2,0}$ is added from transitions $1 \rightarrow 3 \rightarrow 2 \rightarrow q_{2,0}$ by the nonstandard pop rule $r_4$. We add $2 \rightarrow q_{2,0} \Rightarrow 1 \rightarrow q_{2,0}$.

Note that we do not add dependency between $1 \rightarrow q_{2,0}$ and $1 \rightarrow q_{2,0}$ because this nonstandard pop rule decreases the stack and we only add the dependency relation $t_1 \Rightarrow t_2$ 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 A.

Lemma 1. If $p \rightarrow q \equiv^* p' \rightarrow q'$ for $p, p' \in P$, there exists $v \in \Gamma^*$ such that $q' \rightarrow^* q$ and $(p,\gamma) \rightarrow^* (p',\gamma'v)$ in which no simple push transitions appear.

B. 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 V.1. A transition $p \rightarrow q$ for $p \in P$ is pumpable if either

1. there exists a path from $q$ to some $p' \leq p$, i.e., $p \rightarrow q \Rightarrow q \rightarrow^* p'$, or
2) there exist a transition \( p' \xrightarrow{\gamma'} q' \) with \( p' \not\leq p, \gamma' \leq \gamma \), and a path \( p \xrightarrow{\gamma} q \xrightarrow{\gamma^n} q' \) that contains a transition \( p'' \xrightarrow{\gamma''} q'' \) with \( p'' \xrightarrow{\gamma''} q'' \Rightarrow p'' \xrightarrow{\gamma''} q'' \).

Condition 1) in Definition V.1 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 \( \xrightarrow{\gamma} \)) and never pops stack contents. In this case, we do not need to consider the stack.

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 V.1 corresponds to the case that \( \langle p', \gamma' \rangle \) pumps \( \langle p, \gamma w \rangle \) with \( p' \not\leq p, \gamma' \leq \gamma \) and the first rule is not a simple push rule. In the saturation process, the dependency relation between \( p' \xrightarrow{\gamma'} q' \) and \( p \xrightarrow{\gamma} q \) may be interrupted by a simple push transition. However, we can take some transition \( p'' \xrightarrow{\gamma''} q'' \) in the path \( p \xrightarrow{\gamma} q \xrightarrow{\gamma^n} q' \) such that the dependency sequence \( \langle p' \xrightarrow{\gamma'} q' \Rightarrow p'' \xrightarrow{\gamma''} q'' \rangle \) holds before the simple-push operation.

**Example V.2.** Recall the Post*-automaton in Example V.1. The following pumpable transitions imply nontermination.

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

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 \( \xrightarrow{\gamma} \) will be found in finitely many steps. Details and proofs can be found in Appendix A.

**Lemma 2.** Let a WSPDS \( M = \langle (P, \leq), (\Gamma, \leq), (\Delta) \rangle \) and an initial configuration \( c_0 \) such that \( A_0 \) accepts \( c_0 \). There exists a pumpable node in the reachability tree if, and only if, there exists a pumpable transition in Post*\((A_0)\).

**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 \( r : c_0 \).

**C. Boundedness problem**

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

**Definition V.2.** A transition \( p \xrightarrow{\gamma} q \) where \( p \in P \) is strictly pumpable if it satisfies the pumpable conditions in Definition V.1 either by Condition 1), or Condition 2) with an additional condition that \( p' \not\leq p, \gamma' \not\leq \gamma \), or \( p'' \not= p \).

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 V.3.** Recall the Post*-automaton in Example V.1. The two pumpable transitions of Example V.2 are strictly pumpable, which implies unboundedness.

**VI. Complexity issues**

In [8], Hyper-Ackermannian upper and lower bounds for the size of reduced reachability trees are shown for pushdown VASSs. 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 \( \{A_n\} \) each of which computes a fast growing function \( F_{\omega^k}(n) \) and terminates. Since each transition rule increments at most one, the reachability tree has at least \( F_{\omega^k}(n) \) 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 VI.1). By estimating the size of a reduced Post*-automaton by that of the reduced reachability tree, we obtain their Hyper-Ackermannian upper and lower bounds together.

Following Section V of [8], we define a class of fast growing functions by \( \langle F_\lambda \rangle_\lambda \) for an ordinal \( \lambda \leq \omega^\omega \), defined as

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

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

**Definition VI.1.** Let \( k, d \in \mathbb{N} \). A WSPDS \( A = \langle (\mathbb{N}^d, \leq), (\mathbb{N}^k, \leq), (\Delta) \rangle \) with an initial configuration \( (p_0, w_0) \) is bounded if each rule \( (p, w, q, v) \in \Delta \) with \( v \neq \epsilon \) satisfies

\[
||q - p||_\infty \leq 1 \quad \text{and} \quad ||\Sigma(v) - \Sigma(w)||_\infty \leq 1
\]

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

The size of \( A \) is defined as

\[
|A| = d + k + (d + k) \cdot \max\{|p_0|_\infty, |\Sigma(w_0)|_\infty\} + (d + k) \cdot |\Delta|.
\]
Remark. Note that BVAS [5], RVASS[2], multi-set PDS[7], [12], and Pushdown VASS[8] are all subclasses of bounded WSPDSs.

A. 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 [8]. 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 A.

Theorem 6. (Theorem VI.1 in [8]) For $d \geq 1$ and $n \geq 2$, $L_{\omega d}(n) \leq F_{\omega d}(d \cdot n)$, where $L_{\omega d}(n)$ is the maximal length of $n$-controlled bad nested sequences over $\omega^d$.

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

Each run of a bounded WSPDS $A = ((\mathbb{N}^d, \leq), (\mathbb{N}^k, \leq), \triangle)$ with an initial configuration $(p_0, w_0)$ can be abstracted to a nested sequence over $(\mathbb{N}^{d+k}, \leq)$. We prove that this nested sequence is $n$-controlled for $n = \max\{|p_0|_\infty, |\Sigma(w_0)|_\infty\} + 2$ (Lemma 7 in Appendix A). 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 $|A|$ children, we have an upper bound $|\Delta|F_{\omega d+k}((d+k)\cdot n)+1$ for its size, which is also bounded by $F_{\omega d}(|A|)$.

Theorem 7. The reduced reachability tree of a bounded WSPDS has at most $F_{\omega d}(|A|)$ nodes (thus at most $F_{\omega d}(|A|)$ 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 $2|\Delta|F_{\omega d+k}((d+k)\cdot n)+1 \leq F_{\omega d}(|A|)$.

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

B. Lower bound

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

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

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

Remark. Since a Pushdown VASS $A_n$ is a subclass of bounded WSPDSs, the lower bound for the size of the reduced reachability tree for $A_n$ works for bounded WSPDSs as well.

VII. IMPLEMENTATIONS AND EXPERIMENTS

We designed efficient implementation algorithms for both the reduced reachability tree and Post*-automata methods (Algorithm 2 and 1) for the termination detection of bounded WSPDSs. Correctness of Algorithm 1 is guaranteed by Lemma 3.

Both the algorithms adopt depth-first-search strategy. We use stack nodes to restore nodes in the reachability tree and stack trans to store transitions in the Post*-automaton that are waiting to be checked separately. For the former, each time we pop a node from nodes and check whether it is pumpable. If yes, return NO (means non-terminates) and algorithm stops; otherwise, generate new nodes from this node by the transition rules and put the new generated nodes to the stack. For the latter, each time we pop a transition from trans and check whether it is pumpable. If yes, return NO and algorithm stops; otherwise, generate new transitions by the 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 stacks nodes and trans become empty, it means we have checked all the nodes in the reachability tree and all the transitions in the Post*-automaton, but haven’t found any pumpable node and pumpable transition separately. In this case, both algorithms return YES (means terminates) and stops.

Note that we cannot stop generating nodes from a node who has already in the tree and checking them again, otherwise the algorithm would return wrong answers on some cases, since same nodes in different branches may have different ancestors, which may lead to the possible case that in one branch the node is not a pumpable node, but in another branch it is a pumpable node. Different branches sharing the same nodes may use less memory, but it may lead to the checking process much more complex with the cost of more time.

Experiments are performed on a Windows 7 station with 2.60 GHz Intel(R) Core i5 with 8GB of RAM. 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 $I = p, \gamma$ taken from $[0, 15] \times [0, 15]^\text{dim}$ 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 $(p, v \rightarrow p+q, (v+c) (v+d))$ and a non-standard pop rule $(p, v_1, v_2 \rightarrow p+q, v_1+v_2 + c)$, in which each constant and element of a constant vector are randomly taken from $[10, 10]$.

Source code can be found at: https://github.com/leisuhua/WSPDS
Algorithm 1 Implementation Algorithm based on Post*-automata method for Termination of WSPDS $M = ((P; \leq), (\Gamma; \subseteq), \Delta, C_0)$

Input: $A_0 = (Q, \Gamma, \nabla_0, \emptyset, F)$

Output: If $M$ terminates, return YES; otherwise, return NO.

1: procedure CHECK($\langle p, \gamma, q \rangle$, rel, dep)
2: for $q \rightarrow^* p'$ do
3: if $p' \leq p$ then return NO
4: end if
5: end for
6: for $(p', \gamma', q') \in \nabla$ do
7: if $(p, \gamma) \geq (p', \gamma')$ then return NO
8: end if
9: end for
10: end procedure

11: trans := $(\nabla_0) \cap (P \times \Gamma \times Q); rel := (\nabla_0) \setminus trans; Q' := Q;
12: while trans $\neq \emptyset$ do
13: pop $t = (p, \gamma, q)$ from trans; CHECK($t, rel \cup t$, dep);
14: if $t \notin rel$ then
15: rel := rel $\cup t$;
16: for all $(p, \epsilon \rightarrow p', \gamma') \in \Delta$ do
17: trans := trans $\cup \{(p', \gamma', p)\};$
18: end for
19: end if
20: for all $(p, \gamma \rightarrow p', \gamma)$ \in $\Delta$ do
21: trans := trans $\cup \{(p', \gamma', q)\};$
22: dep := dep $\cup \{t \Rightarrow (p', \gamma', q)\};$
23: end for
24: for all $(p, \gamma \rightarrow p', \epsilon) \in \Delta$ do
25: trans := trans $\cup \{(p', \epsilon, q)\};$
26: for all $(q, \gamma', q') \in \text{rel}$ do
27: dep := dep $\cup \{(q, \gamma', q') \Rightarrow (p', \gamma', q')\};$
28: end for
29: end for
30: for all $(p, \gamma' \rightarrow p', \alpha) \in \Delta$ do
31: for all $(q, \gamma', q') \in \text{rel}$ do
32: trans := trans $\cup \{(p', \alpha, q')\};$
33: dep := dep $\cup \{(q, \gamma', q') \Rightarrow (p', \alpha, q')\};$
34: end for
35: end for
36: for all $(p', \gamma' \rightarrow p'', \alpha) \in \Delta$ do
37: for all $(p', \gamma', \alpha) \in \text{rel}$ do
38: trans := trans $\cup \{(p'', \alpha, q)\};$
39: dep := dep $\cup \{t \Rightarrow (p'', \alpha, q)\};$
40: end for
41: end for
42: for all $(p, \gamma \rightarrow p', \alpha \beta) \in \Delta$ do
43: $Q' := Q' \cup \{(q_p, \alpha)\};$
44: trans := trans $\cup \{(p', \alpha, q_p, \alpha)\};$
45: rel := rel $\cup \{(q_p, \alpha, \beta, q)\};$
46: dep := dep $\cup \{t \Rightarrow (p', \alpha, q_p, \alpha), t \Rightarrow (q_p, \alpha, \beta, q)\};$
47: for all $(p'', \epsilon, q_p, \alpha) \in \text{rel}$ do
48: trans := trans $\cup \{(p'', \beta, q)\};$
49: end for
50: end for
51: else
52: for all $(q, \gamma', q') \in \text{rel}$ do
53: trans := trans $\cup \{(p, \gamma', q')\};$
54: end for
55: end if
56: end if

Lemma 3. If $M$ has an infinite run, algorithm1 will return NO; otherwise, recall that $\nabla$ is the set of all transitions of Post$^*$($A_0$), upon termination of algorithm1, $(q, \gamma', q') \in \text{rel}$ holds for any $q, q' \in Q'$ and $\gamma' \in \Gamma \cup \{\epsilon\}$ if and only if $q \rightarrow^* q' \in \nabla$.

Proof of Lemma 3 can be found in Appendix.

Fig.1. (a), (b), and (c) compare between the execution time of the two algorithms on 1000 bounded WSPDSs with 1-, 2-, and 3-dimensional vectors as stack symbols, respectively. Note that we only measure the execution time for checking termination in both approaches, not including the time for parsing the input files, constructing the initial data structures and writing results to output files. The instances are sorted by the execution time of the Post$^*$-automata algorithm. ReachTime and PostTime means the execution time of both algorithms in a logarithmic scale.

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

- **PostTime (ReachTime)** The execution time of the Post$^*$-automata algorithm (the reachability tree algorithm) in milliseconds.
- **NodeNum (EdgeNum)** The number of checked transitions in the generated Post$^*$-automaton (the number of checked nodes in the reduced reachability tree).
- **SpeedRatio** ReachTime/PostTime
- **NumRatio** NodeNum/EdgeNum

We observe that the Post$^*$-automata algorithm shows more stable behavior:

- Both algorithms stops pretty quickly on simple problems.
- The reachability tree algorithm is sometimes slower in magnitude on complex problems, as shown in Table I.
  The opposite is never observed.

The intuition is that the complexity of checking a node and transition is comparable, but in the reachability tree algorithm, the number of nodes need to be checked before the algorithm stops is much larger than that of transitions need to be checked in Post$^*$-automata algorithm, which leads to the execution time of the reachability tree algorithm much more than that of the Post$^*$-automata algorithm. Preliminary experiments on boundedness also show consistent results.

Fig.VII describes the effect of the change of the speed ratio over 100 initial configurations on an arbitrary selected problem from Fig.2(a) We also 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.

VIII. CONCLUSION

Inspired by [8], we studied termination and boundedness for general WSPDSs. We compared two algorithms, the reachability tree algorithm, which is a generalization of that in [8], and the Post$^*$-automata algorithm we proposed. Although the complexity estimation is the same, experiments show that the latter behaves much better than the former.
Fig. 1. (a-c) Post vs. Reach on WSPDS with 1,2,3-dimension vectors as stack symbols, respectively.

Fig. 2. Speed ratio of two algorithms with different initial configurations.

Table I

<table>
<thead>
<tr>
<th>PostTime</th>
<th>EdgeNum</th>
<th>ReachTime</th>
<th>NodeNum</th>
<th>SpeedRatio</th>
<th>NumRatio</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>52</td>
<td>2039</td>
<td>2397</td>
<td>70</td>
<td>46</td>
</tr>
<tr>
<td>105</td>
<td>69</td>
<td>11086</td>
<td>6727</td>
<td>106</td>
<td>97</td>
</tr>
<tr>
<td>507</td>
<td>271</td>
<td>104291</td>
<td>66138</td>
<td>206</td>
<td>244</td>
</tr>
<tr>
<td>74</td>
<td>95</td>
<td>17032</td>
<td>18834</td>
<td>230</td>
<td>198</td>
</tr>
<tr>
<td>523</td>
<td>103</td>
<td>207327</td>
<td>53698</td>
<td>396</td>
<td>521</td>
</tr>
<tr>
<td>153</td>
<td>58</td>
<td>97231</td>
<td>71464</td>
<td>635</td>
<td>1232</td>
</tr>
<tr>
<td>65</td>
<td>71</td>
<td>44947</td>
<td>44164</td>
<td>691</td>
<td>622</td>
</tr>
<tr>
<td>276</td>
<td>191</td>
<td>801889</td>
<td>667065</td>
<td>2905</td>
<td>3492</td>
</tr>
<tr>
<td>426</td>
<td>299</td>
<td>1570971</td>
<td>158307</td>
<td>3688</td>
<td>5285</td>
</tr>
</tbody>
</table>

References


APPENDIX

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 $p \ll q$ holds. $w[2, |w|] = v[2, |v|]$ implies either $w[1] < v[1]$ or $p < q$. Thus, both cases, $n$ strictly pumps $m$.

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

$$(p_0, w_0) \xrightarrow{op_1} \ldots \xrightarrow{op_k} (p_k, w_k) \xrightarrow{op_{k+1}} \ldots \xrightarrow{op_l} (p_i, w_i)$$

such that $(p_k, w_k)$ strictly pumps $(p_i, w_i)$, which leads to an infinite run by iterating the sequence of operations $op_{k+1}, \ldots, op_l$. As the case analysis, if $p_k < p_i$, 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. ☐

We denote transitions of a WSPDS and Post*$\text{-}automata$ by $\rightarrow$ and $\rightarrow^*$, respectively. Let $w \in \Gamma^*$ and

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

We prove the following invariants on Post*$\text{-}automata$.

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

The proof is by induction on the saturation steps of Post*$\text{-}automata$ (See the proof of Lemma 2 in [3]).

Lemma 1. If $p \xrightarrow{\gamma} q \Rightarrow^* p' \xrightarrow{\gamma'} q'$ for $p, p' \in P$, there exists $v \in \Gamma^*$ such that $q \rightarrow^* q$ and $\langle p, q \rangle \rightarrow^* \langle p', \gamma'v \rangle$ in which no simple push transitions appear (thus the stack is kept nonempty).

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 $p \xrightarrow{\gamma} q \Rightarrow^* p' \xrightarrow{\gamma'} q'$ for $p, p' \in P \cup Q$, there exists $v \in \Gamma^*$ such that $q' \rightarrow^* q$ and $\text{con}(p, q) \rightarrow^* \text{con}(p', \gamma'v)$ in which no simple push transitions appear.

Proof. We proceed by the induction on the saturation steps of Post*$\text{-}automata$. For $A_0$, immediate from $\Rightarrow_0 = \emptyset$. Assume it holds for $A_i$, where the dependency relation is denoted by $\Rightarrow_i$. $A_{i+1}$ is obtained from $A_i$ by applying a saturation rule once.

We have $\Rightarrow_{i+1} \supseteq \Rightarrow_i$ 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 $p_1 \xrightarrow{\gamma_1} q_1$, the dependency relation is updated by adding $p_1 \xrightarrow{\gamma_1} q_1 \Rightarrow p_2 \xrightarrow{\gamma_2} q_1$ (if not added yet). Assume that we have $p \xrightarrow{\gamma} q \Rightarrow_i p_1 \xrightarrow{\gamma} q_1 \Rightarrow_{i+1} p_2 \xrightarrow{\gamma_2} q_1 \Rightarrow i_1 \xrightarrow{p'} q'\xrightarrow{\gamma'} q'$

By induction hypothesis, there exist $v_1$ and $v_2$ with $q' \xrightarrow{\gamma'} q_1$, $\text{con}(p, \gamma) \rightarrow^* \langle p_1, \gamma_1v_1 \rangle$, and $\langle p_2, \gamma_2v_1 \rangle \rightarrow^* \text{con}(p', \gamma'v_2v_1)$, among which no simple push transitions appear. Finally, the internal rule connects them as $\text{con}(p, \gamma) \rightarrow^* \langle p_1, \gamma_1v_1 \rangle \rightarrow \langle p_2, \gamma_2v_1 \rangle \rightarrow^* \text{con}(p', \gamma'v_2v_1)$

- If the saturation step is by a push rule $(p_1, \gamma_1, p_2, \gamma_2)$ on $p_1 \xrightarrow{\gamma_1} q_1$, the dependency relation is updated by adding $p_1 \xrightarrow{\gamma_1} q_1 \Rightarrow p_2 \xrightarrow{\gamma_2} q_{p_2, \gamma_2}$ and $p_1 \xrightarrow{\gamma_1} q_1 \Rightarrow q_{p_2, \gamma_2} \xrightarrow{\gamma_1} q_1$ (if not added yet).

First, assume that we have $p \xrightarrow{\gamma} q \Rightarrow_i p_1 \xrightarrow{\gamma_1} q_1 \Rightarrow_{i+1} p_2 \xrightarrow{\gamma_2} q_{p_2, \gamma_2} \Rightarrow_i p' \xrightarrow{\gamma'} q'\xrightarrow{\gamma'} q'$

By induction hypothesis, there exist $v_1, v_2$ with $q' \xrightarrow{\gamma'} q_1, \text{con}(p, \gamma) \rightarrow^* \langle p_1, \gamma_1v_1 \rangle$, and $\langle p_2, \gamma_2v_1 \rangle \rightarrow^* \text{con}(p', \gamma'v_2v_1)$ among which no simple push transitions appear. Finally, the push rule connects them as

$\text{con}(p, \gamma) \rightarrow^* \langle p_1, \gamma_1v_1 \rangle \rightarrow \langle p_2, \gamma_2v_1 \rangle \rightarrow^* \text{con}(p', \gamma'v_2v_1)$

Second, assume that we have $p \xrightarrow{\gamma} q \Rightarrow_i p_1 \xrightarrow{\gamma_1} q_1 \Rightarrow_{i+1} p_2 \xrightarrow{\gamma_2} q_{p_2, \gamma_2} \Rightarrow_i p' \xrightarrow{\gamma'} q'\xrightarrow{\gamma'} q'$

By induction hypothesis, there exist $v_1, v_2$ with $q' \xrightarrow{\gamma'} q_1, \text{con}(p, \gamma) \rightarrow^* \langle p_1, \gamma_1v_1 \rangle$, and $\langle p_2, \gamma_2v_1 \rangle \rightarrow^* \text{con}(p', \gamma'v_2v_1)$ among which no simple push transitions appear. Finally, the push rule connects them as

$\text{con}(p, \gamma) \rightarrow^* \langle p_1, \gamma_1v_1 \rangle \rightarrow \langle p_2, \gamma_2v_1 \rangle \rightarrow^* \text{con}(p', \gamma'v_2v_1)$
con(q_2, \epsilon) \rightarrow^* \langle p_1, \gamma_1 \rangle$, and the nonstandard pop rule connects them as $con(p, \gamma) \rightarrow^* con(q_2, \gamma_2v_1) \rightarrow^* \langle p_1, \gamma_1v_2v_1 \rangle \leftarrow \langle p_2, \gamma_3v_1 \rangle \leftarrow^* con(p', \gamma'v_2v_1)$.

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

**Lemma 5.** Let $p, q \in P$ and $\alpha, \gamma \in \Gamma$.
1. If $\langle p, \epsilon \rangle \rightarrow^* \langle q,v \rangle$, we have $q \rightarrow^* v \rightarrow p$.
2. If $\langle p, \alpha \rangle \rightarrow^* \langle q, \gamma v \rangle$ in which no simple push transitions appear (thus during the run the stack stays nonempty) and $p \stackrel{\alpha}{\rightarrow} q'$, we have $q \rightarrow^* q_2 \rightarrow^* q'$ such that either
   - $\alpha = \gamma p \eta$; or
   - $v = v_1 \gamma' v_2$ and $q_2 \rightarrow^* p'' \gamma'' v_2 q'' \rightarrow^* q'$ for some $p'' \in P$ and $p \rightarrow^* q'' \rightarrow^* q'$ for each transition $\iota$ in $p'' \rightarrow^* q'' \gamma'' v_2 q''$.

**Proof.** By induction on the length of $\rightarrow^*$. If $i = 0$, immediate. Assume that they hold for $i$.
1. If $\langle p, \epsilon \rangle \rightarrow^i \langle q_1, \nu_1 \rangle \rightarrow^i \langle q, v \rangle$, by induction hypothesis, we have $q_1 \rightarrow^i q_2 \rightarrow^* q$ which satisfies either of two cases below.
   - $\alpha = \gamma p \eta$; or
   - $v = v_1 \gamma' v_2$ and $q_2 \rightarrow^* p'' \gamma'' v_2 q'' \rightarrow^* q'$ for some $p'' \in P$ and $p \rightarrow^* q'' \rightarrow^* q'$ for each transition $\iota$ in $p'' \rightarrow^* q'' \gamma'' v_2 q''$.

2. If $\langle p, \alpha \rangle \rightarrow^i \langle q, \gamma v \rangle$ and $p \rightarrow^* q'$, by induction hypothesis, we have $q_1 \rightarrow^i \gamma_1 v_2 \rightarrow^* q$.

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

**Lemma 2’** Given a WSPDS $M = \langle (P, \leq), (\Gamma, \leq), \Delta \rangle$ and an initial configuration $c_0$. Assume that $A_0$ accepts $c_0$. Then, in Post*(A_0),
1. if $p \rightarrow q$ is pumpable, there exists $w$ with $q \rightarrow^* f$ and $c_0 \rightarrow c_1 \rightarrow^* c_n \rightarrow (p, \gamma w)$ in the reachability tree such that $(p, \gamma w)$ is pumpable;
2. if $p \rightarrow q$ is not pumpable, for every $w$ with $q \rightarrow^* f$, there exists $c_0 \rightarrow c_1 \rightarrow^* c_n \rightarrow (p, \gamma w)$ in the reachability tree such that $(p, \gamma w)$ is not pumpable;

**Proof.** (1) The proof follows to Definition V.1 of pumpable transitions.
- If $p \rightarrow q$ satisfies the first condition in Definition V.1, i.e., there exists some $p' \in P$ such that $p \rightarrow q \rightarrow^* (p', \gamma w)$.
- If $p \rightarrow q$ satisfies the second condition in Definition V.1, i.e., there exists $p'' \rightarrow^* q'' \rightarrow^* f$ with $p'' \leq p$, $\gamma'' \leq \gamma$ and there exists a path $p \rightarrow q \rightarrow^* q''$ containing a transition $p'' \rightarrow^* q''$ with $p'' \rightarrow^* q'' \rightarrow^* q'$. Let $\gamma = w_1 \gamma'' w_2$. By Lemma 1 and 4, there exists $c_0 \rightarrow^* \langle p', \gamma'' w_2 \rangle \rightarrow^* \langle p'', \gamma'' w_1 w_2 \rangle$.

(2) We prove by contradiction. Assume that $p \rightarrow q$ is not pumpable and there exists $w$ with $q \rightarrow^* f$ and $c_0 \rightarrow c_1 \rightarrow^* c_n \rightarrow (p, \gamma w)$ in the reachability tree such that $(p, \gamma w)$ is pumped by $c_i = (p', w')$. From Definition IV.1, we have two cases.
- $\langle p', \epsilon \rangle \rightarrow^* \langle p, \gamma w' \rangle$ with $w = w'' w'$. By Lemma 5, there exists $p \rightarrow q \rightarrow^* p'$. Hence $p \rightarrow q$ is a pumpable transition, which contradicts the assumption.
- $\langle p', \gamma' \rangle \rightarrow^* \langle p, \gamma w'' \rangle$ with $w' = \gamma'' w_1$ and $w = w'' w_1$, during the run the stack stays nonempty. Also by Lemma 5, $p \rightarrow q$ 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. □

Definition A.1. A nested sequence over a set \( S \) is a (finite or infinite) sequence \( (s_0, h_0), (s_1, h_1), \ldots \) 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 \( M = ((P, \preceq), (\Gamma, \preceq), \Delta) \), 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 A.2. A nested sequence \((s_0, h_0), (s_1, h_1), \ldots \) over a quasi-ordered set \((S, \preceq)\) is good if there exists \( i < j \) such that \( s_i \preceq s_j \) and \( h_{i+1}, \ldots, h_j \) are non-decreasing. 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 A.3. A norm for a WQO set \((P, \preceq)\) is a function \( \|\cdot\| : P \rightarrow \mathbb{N} \) such that \( \{ p \in P | \|p\| \leq n \} \) is finite for each \( n \). The structure \((P, \preceq, \|\cdot\|)\) is called a normed WQO.

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

Definition A.4. A nested sequence \((s_0, h_0), (s_1, h_1), \ldots \) over a normed WQO set \((S, \preceq)\) is \( n \)-controlled for \( n \in \mathbb{N} \) if \( \max\{\|s_j\|\} \leq n + j \) for each index \( j \).

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

Remark. Every run of a bounded WSPDS \( A = \langle (\mathbb{N}^d, \preceq), (\mathbb{N}^k, \preceq), \Delta \rangle \) 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 \( 0 \). \((h(p_0, w_0), |w_0|), (h(p_1, w_1), |w_1|), \ldots \) is a nested sequence over \( \mathbb{N}^d \times \mathbb{N}^k \).

Lemma 6. Given a bounded WSPDS \( A = \langle (\mathbb{N}^d, \preceq), (\mathbb{N}^k, \preceq), \Delta \rangle \) with an initial configuration \((p_0, w_0)\). The length of a bad nested sequence \((h(p_0, w_0), |w_0|), \ldots, (h(p_1, w_1), |w_1|), \ldots \) associated with the path \((p_0, w_0), \ldots, (p_1, w_1), \ldots \) will not change by mapping \( \perp \) to \( 0 \).

Proof. Consider one step transition \((p_{i-1}, w_{i-1}) \rightarrow (p_i, w_i)\). 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 \((p_{i-1}, \perp, 0)\). In this case, the nested sequence is bad after mapping \( \perp \) to \( 0 \).
- 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 \( 0 \) 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 \( (\mathbb{N}^d \times \mathbb{N}^k, \preceq, \perceq) \) is \( n \)-controlled for \( n = \max\{\|p_0\|_\infty, \|\Sigma(w_0)\|_\infty\} + 2 \).\( \Box \)

Proof. We define the norm of a WQO \( (\mathbb{N}^d \times \mathbb{N}^k, \preceq, \perceq) \) as \( \|\|p_0, w_1[1]\|\| = \max\{\|p_1\|_\infty, \|w_1[1]\|_\infty\} \). 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.) \( \|\|p_0, w_1[1]\|\| \leq n \) 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 \((p_i, w_i) \rightarrow (p_{i+1}, w_{i+1})\). 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 \( \Sigma(\alpha \beta) - \Sigma(\gamma) \) by \( \|\| \leq 1 \). Hence,

\[ \|\|p_{i+1}, w_{i+1}[1]\|\| \leq \max\{\|p_{i+1}\|_\infty, \|w_{i+1}[1]\|_\infty\} \]

\[ \leq \max\{\|p_i\|_\infty, \|\Sigma(w_i)\|_\infty\} \]

\[ \leq \max\{\|p_i\|_\infty, \|w_i\|_\infty\} + 1 \]

\[ \leq n + i + 1 \]

□

Lemma 3. If \( M \) has an infinite run, algorithm1 will return NO; otherwise, recall that \( \nabla \) is the set of all transitions of \( P^\ast(A_0) \), upon termination of algorithm1, \((q, \gamma', q') \) \( \in re\) holds for any \( q, q' \in Q' \) and \( \gamma' \in \Gamma \cup \{ \epsilon \} \) and only if \( q \rightarrow^\gamma q' \in \nabla \).

Proof. Firstly, if \( M \) has an infinite run, then at some time of the execution of the algorithm, a pumpable transition is found by CHECK procedure, Algorithm 1 returns NO.

Secondly, we prove that if \( M \) doesn’t have infinite run, upon termination of algorithm1, \((q, \gamma', q') \in re\) holds for any \( q, q' \in Q' \) and \( \gamma' \in \Gamma \cup \{ \epsilon \} \) if and only if \( q \rightarrow^\gamma q' \in \nabla \).

By Definition II.3, set \( \nabla \) satisfies the following:

- If \( (p, \epsilon \rightarrow p', \gamma') \in \Delta \) and \( p \rightarrow^\gamma q \in \nabla \), then \( p' \rightarrow^\gamma p \in \nabla \)
- If \( (p, \gamma \rightarrow p', \gamma') \in \Delta \) and \( p \rightarrow^\gamma q \in \nabla \), then \( p' \rightarrow^\gamma q \in \nabla \)
- If \( (p, \gamma \rightarrow p', \epsilon) \in \Delta \) and \( p \rightarrow^\gamma q \in \nabla \), then \( p' \rightarrow^\gamma q \in \nabla \)
- If \( (p, \alpha \beta \rightarrow p', \gamma) \in \Delta \) and \( p \rightarrow^\gamma q \in \nabla \), then \( p' \rightarrow^\gamma q \in \nabla \)
- If \( (p, \alpha \beta \rightarrow p', \epsilon) \in \Delta \) and \( p \rightarrow^\gamma q \in \nabla \), then \( p' \rightarrow^\gamma q \in \nabla \)
Algorithm 2 Reduced Reachability Tree based Implementation Algorithm for Termination of WSPDS $\mathcal{M} = \langle (P, \preceq), (\Gamma, \subseteq), \Delta, C_0 \rangle$

Input:
1. $\mathcal{M} = \langle (P, \preceq), (\Gamma, \subseteq), \Delta, C_0 \rangle$
2. $\text{nodes} = \{\text{root} : C_0\}$

Output:
3. $\mathcal{M}$ terminates, return YES; otherwise, return NO.
4. while Nodest! = NULL do
5. currentNode = nodes.pop()
6. for all $r \in \Delta$ do
7. nodes.push(computeChild(currentNode, r))
8. end for
9. minLength = currentNode.stack.length
10. ancestor = currentNode.father
11. while ancestor! = NULL do
12. len = ancestor.stack.length
13. if minLength $\geq$ len then
14. if currentNode.State $\geq$ ancestor.State then
15. if currentNode.stack.top $\geq$ ancestor.stack.top || ancestor.stack = NULL then return NO.
16. end if
17. end if
18. minLength = len
19. end if
20. ancestor = ancestor.father
21. end while
22. endwhile return YES

Firstly, we show that if $(q, q', q'') \in \text{rel}$, then $q \rightarrow^* q' \in \nabla$. Since elements from $\text{trans}$ flow into $\text{rel}$, we inspect all the lines that change $\text{trans or rel}$:

- Lines 11 add elements from $\nabla_0$, which is subset of $\nabla$.
- Line 17 is a case of rule $(p, \epsilon \rightarrow p', q') \in \Delta$.
- Line 22 is a case of rule $(p, \gamma \rightarrow p', q') \in \Delta$.
- Line 25 is a case of rule $(p, \gamma \rightarrow p', \epsilon) \in \Delta$.
- Line 32 and 38 are cases of rule $(p, \alpha \beta \rightarrow p', \gamma) \in \Delta$.
- Line 44 and 45 is a case of rule $(p, \gamma \rightarrow p', \alpha \beta) \in \Delta$.

- In line 48, we have $(p', \beta \rightarrow q', \alpha \beta) \in \text{rel}$ and $(q', \beta, \alpha \beta, \beta, q) \in \text{trans}$. Since both transitions must have resulted from some application of the saturation rule, we conclude that $p' \rightarrow^* q \in \nabla$ holds, so the addition of $(p', \beta, q) \in \text{rel}$ is justified.

Likewise, in line 53, we combine $(p, \epsilon, \gamma, q) \in \text{rel}$ and $(q', \gamma, q') \in \text{rel}$ to $(p, q', q') \in \text{rel}$.

We observe the following: Since there are no transitions leading into $P$, the $\epsilon$-transitions can only go from states in $P$ to states in $Q' \setminus P$ and no two $\epsilon$-transitions can be adjacent. The relation $p \rightarrow^* q \in \nabla$ can thus be written as follows:

$p \rightarrow^* q \in \nabla \iff p \rightarrow q \in \nabla \land \exists q' : p \rightarrow q' \in \nabla \land q' \rightarrow q \in \nabla$

The desired property follows from the following facts:

- Because of lines 11, $\nabla_0 \subseteq \text{rel}$ holds.
- If $(p, \epsilon \rightarrow p', q) \in \Delta$ and $(p, \gamma, q) \in \text{rel}$, then $(p', \gamma, p) \in \text{rel}$ is added by line 17.
- If $(p, \gamma \rightarrow p', q) \in \Delta$ and $(p, \gamma, q) \in \text{rel}$, then $(p', \gamma, q) \in \text{rel}$ is added by line 22.
- If $(p, \gamma \rightarrow p', \epsilon) \in \Delta$ and $(p, \gamma, q) \in \text{rel}$, then $(p', \epsilon, q) \in \text{rel}$ is added by line 25.
- If $(p, \gamma \rightarrow p', \gamma) \in \Delta$ and there is a pair $(p, \gamma, q), t_1 = (q, \gamma, q') \in \text{rel}$ for some $p, q, p' \in Q'$, then we need to add $(p', \gamma, q')$.

1) If $t_1$ was examined before $t_2$, then $(p', \gamma, q')$ is added by line 38.
2) If $t_2$ was examined before $t_1$, then $(p', \gamma, q')$ is added by line 32.
3) If $(p, \gamma \rightarrow p', \alpha \beta) \in \Delta$ and $(p, \gamma, q) \in \text{rel}$, then $(p', \alpha \beta, p', \alpha \beta, \beta, q) \in \text{rel}$ is added by line 44 and 45.
4) Whenever there is a pair $(p, \epsilon, q')$ and $(q', \gamma, q) \in \text{rel}$ for some $p, q, q' \in Q'$ and $\gamma \in \Gamma$, we need to add $(p, \gamma)$. In this case:
1) Assume $(q', \gamma, q)$ is known before $(p, \epsilon, q')$, then $(p, \gamma, q)$ is added by line 53.
2) Otherwise $(p, \epsilon, q')$ is known before $(q', \gamma, q)$, then $(p, \gamma, q)$ is added by line 48.