Information and Computation 259 (2018) 41–71

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Analyzing pushdown systems with stack manipulation[☆]

Fu Song

School of Information Science and Technology, ShanghaiTech University, 393 Middle Huaxia Road, Pudong, Shanghai 201210, China

ARTICLE INFO

Article history:
Received 14 July 2015
Received in revised form 10 October 2017
Available online 19 December 2017

Keywords:
Reachability analysis
Pushdown systems
Saturation procedure
Boolean program
Software verification

ABSTRACT

Pushdown systems with transductions (TrPDSs) are an extension of pushdown systems (PDSs) by associating each transition rule with a transduction, which allows to inspect and modify the stack content at each step of a transition rule. In this work, we propose two novel saturation procedures to compute $pre^*(C)$ and $post^*(C)$ for finite TrPDSs. From these two saturation procedures, we present two algorithms to compute $pre^*(C)$ and $post^*(C)$ that are suitable for implementation. We also show that the algorithms for computing $pre^*(C)$ and $post^*(C)$ also work for *weak* finite TrPDSs, where closure is defined with respect to the underlying PDSs. These results are extended to *left contextual* TrPDSs, which is an extension of finite TrPDSs. Finally, we show how the presence of transductions enables the modeling of Boolean programs with call-by-reference parameter passing and low-level assembly programs that manipulate the program stack content via a stack pointer.

1. Introduction

A pushdown system (PDS) consists of a finite set of states and a finite stack alphabet, where stack can store context. PDS is one of the most widely used formalisms for modeling sequential programs with recursion [1–3]. There are many works investigating the model-checking problem of PDSs. Bouajjani et al. proposed a *saturation* method for model-checking reachability, based on which LTL and alternating free μ-calculus also can be verified [4]. This saturation method is also discovered independently by Finkel et al. [5]. Based on these results, many efficient algorithms for model-checking PDSs were proposed [1,2,6,7], which lead to several software model-checking tools such as Moped [8], PDSolver [9], PuMoC [10] for C/C++, Java and/or Boolean programs verification.

Various extensions of PDSs such as multi-stack PDSs [11], pushdown networks [12–15] and well-structured PDSs [16] have been proposed for modeling concurrent (thread-creation) programs with recursion. In order to model timed (resp. probabilistic) behavior, models that combine timed (resp. probabilistic) automata and PDSs were investigated in the literature, e.g., discrete/dense-timed pushdown systems [17,18], nested timed automata [19] (resp. probabilistic PDSs [20,21]). For dataflow analysis purpose, weighted PDSs [22], extended weighted PDSs [23] and weighted PDSs with indexed weight domains [24] were proposed, where transitions are associated with values from semirings. For stack manipulation of PDSs, Esparza et al. introduced PDSs with checkpoint [25] that can check the full stack content against a regular language (recognized by a finite state automaton) over the stack alphabet. This model is called conditional PDSs in [26] and transformable PDSs in [27]. Uezato and Minamide extended PDSs with transductions (TrPDSs) which associate each transition with a transduction. The associated transductions can check the stack content and modify the whole stack content. TrPDSs are a

[☆] A preliminary version has been presented to CONCUR 2015.
E-mail address: songfu@shanghaitech.edu.cn.

https://doi.org/10.1016/j.ic.2017.12.003
0890-5401/© 2017 Elsevier Inc. All rights reserved.

generalization of PDSs with checkpoint and discrete-timed PDSs. In general, TrPDSs are Turing complete. To achieve a decidability result, Uezato and Minamide considered *finite* TrPDSs which restrict the closure of transductions appearing in the transitions of a TrPDS to be finite. They showed that a finite TrPDS can be simulated by a PDS. Therefore, the reachability problem of finite TrPDSs is decidable. Moreover, the saturation procedure that calculates the set $pre^*(C)$ of predecessor configurations for a given regular set of configurations C can be directly extended from PDSs to finite TrPDSs. Finite TrPDSs can be seen as a special case of weighted PDSs with indexed weight domains [24,28].

1.1. Contributions

In this work, we follow the direction of [27] and make a comprehensive study of the reachability problem of TrPDSs. The main contributions of this article are summarized as follows:

- A novel saturation procedure is proposed that computes the set $pre^*(C)$ of predecessor configurations for a given regular set of configurations C of TrPDSs (cf. Section 3.1). This saturation procedure avoids *pseudo formal power series semirings* that was introduced in [27] to compute $pre^*(C)$.
- A saturation procedure is introduced to compute the set $post^*(C)$ of successor configurations for a given regular set of configurations C of TrPDSs (cf. Section 4.1). TrPDSs can be simulated by PDSs as shown in [27]. Therefore, $post^*(C)$ could be computed by applying the saturation procedure of PDSs [1]. Our saturation procedure directly computes a kind of finite state automaton that exactly recognizes $post^*(C)$. We believe that our direct approach is more convenient for studying optimal algorithms or BDD-based symbolic techniques.
- Algorithms of the saturation procedures for computing $pre^*(C)$ and $post^*(C)$ are presented (cf. Section 3.2 and Section 4.2), which are suitable for implementation. We show that the computations of both $pre^*(C)$ and $post^*(C)$ are fixed-parameter tractable with respect to the number of transductions. From these two algorithms, we observe that the definition of finiteness of finite TrPDSs can be refined. For this, we introduce weak finite TrPDSs, where the closure of transductions is defined with respect to the underlying TrPDSs rather than only the set of transductions. We show that these two algorithms still work for weak finite TrPDSs.
- We introduce an extension of finite TrPDSs, called left contextual TrPDSs, which can be viewed as an extension of prefix-recognizable systems with stack manipulation [29,6,30,7]. The approaches for computing $pre^*(C)$ and $post^*(C)$ of finite TrPDSs are not suitable for computing $pre^*(C)$ and $post^*(C)$ of left contextual TrPDSs. To overcome this problem, We present an reduction from the reachability problem of left contextual TrPDSs to the problem of finite TrPDSs (cf. Section 6).
- We show that TrPDSs are powerful enough to model Boolean programs with call-by-reference parameter passing and low-level assembly programs that manipulate the program stack content via a stack pointer. Boolean programs in the literature [31] only consider call-by-value parameter passing which can be modeled by PDSs, and low-level assembly program models cannot directly model the manipulation of program stack via a stack pointer. Using our approach, safety properties of Boolean programs with mixed call-by-reference and call-by-value parameter passing can be directly verified (cf. Section 7.1). Moreover, TrPDSs can model low-level assembly programs in a more precise manner than the approaches of [32,33]. This could be used to characterize assembly programs for which the reachability problem is decidable via decidable TrPDSs (cf. Section 7.2).

This article is an extended version of [34] which presented saturation procedures and their algorithms for computing $pre^*(C)$ and $post^*(C)$ of finite TrPDSs. In addition to the results presented there, we provide detailed examples and proofs, introduce weak finite TrPDSs, and left contextual TrPDSs and thereof $pre^*(C)$ and $post^*(C)$ computing approaches, and present new potential applications of TrPDSs to model low-level assembly programs and Boolean programs.

1.2. Organization

Section 2 introduces basic definitions. In Section 3 (resp. Section 4), we present the saturation procedure and its algorithm for computing $pre^*(C)$ (resp. $post^*(C)$), which is suitable for implementation. Weak finite TrPDSs and their reachability problem are presented in Section 5. Section 6 introduces left contextual TrPDSs and shows how to reduce their reachability problem to the problem of finite TrPDSs. In Section 7, we present two potential applications of TrPDSs for modeling and verifying Boolean programs with call-by-reference parameter passing and low-level assembly programs that manipulate the program stack content via a stack pointer. Section 8 discusses related work. Section 9 concludes and discusses future work.

2. Preliminaries

2.1. Finite-state transducers and transductions

Definition 1. A *finite-state transducer* (FST) $\mathbf{T}$ is a tuple $(Q, \Gamma, \delta, I, F)$, where Q is a finite set of states, Γ is a finite alphabet, $\delta \subseteq Q \times \Gamma^* \times \Gamma^* \times Q$ is a finite set of transition rules, $I \subseteq Q$ (resp. $F \subseteq Q$) is a finite set of initial (resp. final) states. The transducer is *letter-to-letter* if $\delta \subseteq Q \times \Gamma \times \Gamma \times Q$.

We will write $q \xrightarrow{\omega_1/\omega_2} q'$ for every $(q,\omega_1,\omega_2,q') \in \delta$. Let $\longrightarrow^*$ be the smallest relation such that $q \xrightarrow{\epsilon/\epsilon}{}^* q$ for every $q \in Q$; if $q \xrightarrow{\omega_1/\omega_2}{}^* q'$ and $q' \xrightarrow{\omega_3/\omega_4} q''$, then $q \xrightarrow{\omega_1\omega_3/\omega_2\omega_4}{}^* q''$. A FST $\mathbf{T}$ transduces a string $\omega_1 \in \Gamma^*$ into a string $\omega_2 \in \Gamma^*$ if there exist states $q_0 \in I$ and $q_f \in F$ such that $q_0 \xrightarrow{\omega_1/\omega_2}{}^* q_f$. The language $L(\mathbf{T})$ of a FST $\mathbf{T}$ is the set of pairs (ω_1,ω_2) such that $\mathbf{T}$ can transduce ω_1 into ω_2.

A *transduction* $\tau \subseteq \Gamma^* \times \Gamma^*$ is a relation over Γ^*. A transduction τ is *rational* (regular) and *length-preserving* if there is a letter-to-letter transducer $\mathbf{T}$ such that $\tau = L(\mathbf{T})$. Let τ_{id} denote the *identity transduction*, i.e., $\tau_{id} = \{(\omega,\omega) \mid \forall\omega \in \Gamma^*\}$. In the rest of this article, we assume that transductions (resp. transducers) are length-preserving rational (resp. letter-to-letter) unless stated explicitly, and we do not differentiate the terms transduction and transducer. Given a transduction τ, let $\tau(\omega) = \{\omega' \mid (\omega,\omega') \in \tau\}$ for $\omega \in \Gamma^*$.

The *composition* $\circ$ of two transductions τ_1, τ_2 is defined as

$$\tau_1 \circ \tau_2 = \{(\omega_1,\omega_3) \in \Gamma^* \times \Gamma^* \mid \exists\omega_2 \in \Gamma^*, (\omega_1,\omega_2) \in \tau_1 \text{ and } (\omega_2,\omega_3) \in \tau_2\}.$$

It is easy to show the following proposition.

Proposition 1. *For every transduction τ, $\tau \circ \tau_{id} = \tau = \tau_{id} \circ \tau$.*

The *left quotient* $\lceil\cdot,\cdot\rfloor^{-1}$ over a transduction is defined as follows: $\forall\omega_1,\omega_2 \in \Gamma^*$ with $|\omega_1| = |\omega_2|$, $\lceil\omega_1,\omega_2\rfloor^{-1}\tau = \{(\omega,\omega') \in \Gamma^* \times \Gamma^* \mid (\omega_1\omega,\omega_2\omega') \in \tau\}$.

Proposition 2. [27] *For every $\omega_1,\omega_2 \in \Gamma^n$, for every transduction τ_1,τ_2,*

$$\lceil\omega_1,\omega_2\rfloor^{-1}(\tau_1 \circ \tau_2) = \bigcup_{\omega_3 \in \Gamma^{|\omega_1|}} \big((\lceil\omega_1,\omega_3\rfloor^{-1}\tau_1) \circ (\lceil\omega_3,\omega_2\rfloor^{-1}\tau_2)\big).$$

Let $\mathcal{T}$ be a set of transductions, the closure $\langle\mathcal{T}\rangle^{\cup}$ of $\mathcal{T}$ over the composition $\circ$, left quotient $\lceil\cdot,\cdot\rfloor^{-1}$ and union $\cup$ is defined as follows:

- $\mathcal{T} \subseteq \langle\mathcal{T}\rangle^{\cup}$, $\emptyset \in \langle\mathcal{T}\rangle^{\cup}$ and $\tau_{id} \in \langle\mathcal{T}\rangle^{\cup}$;
- if $\tau_1,\tau_2 \in \langle\mathcal{T}\rangle^{\cup}$, then $\tau_1 \circ \tau_2 \in \langle\mathcal{T}\rangle^{\cup}$ and $\tau_1 \cup \tau_2 \in \langle\mathcal{T}\rangle^{\cup}$;
- if $\tau \in \langle\mathcal{T}\rangle^{\cup}$, then $\lceil\gamma,\gamma'\rfloor^{-1}\tau \in \langle\mathcal{T}\rangle^{\cup}$ for all $\gamma,\gamma' \in \Gamma$.

Similarly, let $\langle\mathcal{T}\rangle$ denote the closure of $\mathcal{T}$ over the composition $\circ$ and left quotient $\lceil\cdot,\cdot\rfloor^{-1}$ defined as follows:

- $\mathcal{T} \subseteq \langle\mathcal{T}\rangle$, $\emptyset \in \langle\mathcal{T}\rangle$ and $\tau_{id} \in \langle\mathcal{T}\rangle$;
- if $\tau_1,\tau_2 \in \langle\mathcal{T}\rangle$, then $\tau_1 \circ \tau_2 \in \langle\mathcal{T}\rangle$;
- if $\tau \in \langle\mathcal{T}\rangle$, then $\lceil\gamma,\gamma'\rfloor^{-1}\tau \in \langle\mathcal{T}\rangle$ for all $\gamma,\gamma' \in \Gamma$.

Proposition 3.
(a) The set $\langle\mathcal{T}\rangle$ is finite iff the set $\langle\mathcal{T}\rangle^{\cup}$ is finite.
(b) The set $\langle\mathcal{T}\rangle^{\cup}$ is the semigroup generated by $(\langle\mathcal{T}\rangle,\cup)$, that is, $\forall\tau \in \langle\mathcal{T}\rangle^{\cup}$, $\exists\tau_1,\ldots,\tau_m \in \langle\mathcal{T}\rangle$ for $m \geq 1$ such that $\tau = \bigcup_{i=1}^{m}\tau_i$.

Proof. Proof of (a): If $\langle\mathcal{T}\rangle^{\cup}$ is finite, we can immediately get that $\langle\mathcal{T}\rangle$ is finite, as $\langle\mathcal{T}\rangle \subseteq \langle\mathcal{T}\rangle^{\cup}$. The other direction immediately follows from (b). We only need to prove (b).
Proof of (b): Let $\mathcal{T}'$ be the semigroup generated by $(\langle\mathcal{T}\rangle,\cup)$. If $\langle\mathcal{T}\rangle$ is finite, then $\mathcal{T}'$ is finite. We prove that $\mathcal{T}' = \langle\mathcal{T}\rangle^{\cup}$. It is sufficient to show that the following two conditions hold:

- for all $\tau \in \mathcal{T}'$, $\lceil\gamma,\gamma'\rfloor^{-1}\tau \in \mathcal{T}'$ for all $\gamma,\gamma' \in \Gamma$,
- for all $\tau_1,\tau_2 \in \mathcal{T}'$, $\tau_1 \circ \tau_2 \in \mathcal{T}'$.

Suppose $\tau = \tau_1 \cup \cdots \cup \tau_n \in \mathcal{T}'$ with $\tau_1,\cdots,\tau_n \in \langle\mathcal{T}\rangle$, then $\lceil\gamma,\gamma'\rfloor^{-1}(\tau_1 \cup \cdots \cup \tau_n) = \bigcup_{i=1}^{n}\lceil\gamma,\gamma'\rfloor^{-1}\tau_i$. Since $\tau_1,\cdots,\tau_n \in \langle\mathcal{T}\rangle$, we obtain that $\lceil\gamma,\gamma'\rfloor^{-1}\tau_1,\cdots,\lceil\gamma,\gamma'\rfloor^{-1}\tau_n \in \langle\mathcal{T}\rangle$. Thus, we get that $\lceil\gamma,\gamma'\rfloor^{-1}(\tau_1 \cup \cdots \cup \tau_n) \in \mathcal{T}'$.

Now, we show that for all $\tau_1,\tau_2 \in \mathcal{T}'$, $\tau_1 \circ \tau_2 \in \mathcal{T}'$. Suppose $\tau_i = \tau_1^i \cup \cdots \cup \tau_{m_i}^i$ with $\tau_1^i,\cdots,\tau_{m_i}^i \in \langle\mathcal{T}\rangle$ for $i \in \{1,2\}$, then $\tau_1 \circ \tau_2 = (\tau_1^1 \cup \cdots \cup \tau_{m_1}^1) \circ (\tau_1^2 \cup \cdots \cup \tau_{m_2}^2) = \bigcup_{1\leq i\leq m_1, 1\leq j\leq m_2}\tau_i^1 \circ \tau_j^2$. Since for every $i : 1 \leq i \leq m_1$ and every $j : 1 \leq j \leq m_2$, $\tau_i^1,\tau_j^2 \in \langle\mathcal{T}\rangle$, then $\tau_i^1 \circ \tau_j^2 \in \langle\mathcal{T}\rangle$. Thus, we get that $\tau_1 \circ \tau_2 \in \mathcal{T}'$. □

2.2. Pushdown systems with transductions

Pushdown systems with transductions (TrPDSs) [27] are an extension of pushdown systems by associating each transition with a transduction which checks and/or modifies the stack content by applying the transduction. This extension allows TrPDSs to model sequential programs that manipulate the stack content rather than only the top of the stack.

Definition 2. A *pushdown system with transductions* (TrPDS) $\mathcal{P}$ is a tuple $(P, \Gamma, \mathcal{T}, \Delta)$, where P is a finite set of control states, Γ is a finite alphabet, $\mathcal{T}$ is a finite set of transductions over Γ^*, $\Delta \subseteq (P \times \Gamma) \times \mathcal{T} \times (P \times \Gamma^*)$ is a finite set of transition rules. A TrPDS is a *pushdown system* (PDS) if $\mathcal{T} = \{\tau_{id}\}$.

We will write $\langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle p', \omega\rangle$ for every $(p, \gamma, \tau, p', \omega) \in \Delta$. A *configuration* of a TrPDS $\mathcal{P}$ is a pair $\langle p, \omega\rangle \in P \times \Gamma^*$ where p is the control state and ω is the stack content. Let $\mathcal{C}_\mathcal{P}$ denote the set of all the configurations $P \times \Gamma^*$ of the TrPDS $\mathcal{P}$. The TrPDS $\mathcal{P}$ is called *finite* if the set $\langle\mathcal{T}\rangle$ (i.e., $\langle\mathcal{T}\rangle^\cup$) is finite. If $\langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle p', \omega\rangle$, then for every $\omega' \in \Gamma^*$, the configuration $\langle p, \gamma\omega'\rangle$ is an *immediate predecessor* of the configuration $\langle p', \omega u\rangle$ for every $u \in \tau(\omega')$, and the configuration $\langle p', \omega u\rangle$ for every $u \in \tau(\omega')$ is an *immediate successor* of the configuration $\langle p, \gamma\omega'\rangle$. Let $\Longrightarrow_\mathcal{P} \subseteq \mathcal{C}_\mathcal{P} \times \mathcal{C}_\mathcal{P}$ be the *immediate successor relation*, i.e., for every $\omega', u \in \Gamma^*$, $\langle p, \gamma\omega'\rangle \Longrightarrow_\mathcal{P} \langle p', \omega u\rangle$ if $\langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle p', \omega\rangle$ and $u \in \tau(\omega')$. A *run* of $\mathcal{P}$ is a sequence of configurations $c_1 c_2 \cdots$ such that for every $i \geq 1$, c_{i+1} is an immediate successor of c_i.

Let $\Longrightarrow^n_\mathcal{P} \subseteq \mathcal{C}_\mathcal{P} \times \mathcal{C}_\mathcal{P}$ be the successor relation over configurations of $\mathcal{P}$ defined as follows:

- $c \Longrightarrow^0_\mathcal{P} c$ for every $c \in \mathcal{C}_\mathcal{P}$;
- $c \Longrightarrow^n_\mathcal{P} c''$ if there exists $c' \in \mathcal{C}_\mathcal{P}$ such that $c \Longrightarrow_\mathcal{P} c'$ and $c' \Longrightarrow^{n-1}_\mathcal{P} c''$.

Let $\Longrightarrow^*_\mathcal{P} \subseteq \mathcal{C}_\mathcal{P} \times \mathcal{C}_\mathcal{P}$ denote the reflexive transitive closure of the immediate successor relation $\Longrightarrow_\mathcal{P}$, i.e., $\Longrightarrow^*_\mathcal{P} = \bigcup_{i\geq 0} \Longrightarrow^i_\mathcal{P}$. Let $\Longrightarrow^+_\mathcal{P} \subseteq \mathcal{C}_\mathcal{P} \times \mathcal{C}_\mathcal{P}$ denote the transitive closure of the immediate successor relation $\Longrightarrow_\mathcal{P}$, i.e., $\Longrightarrow^+_\mathcal{P} = \bigcup_{i\geq 1} \Longrightarrow^i_\mathcal{P}$.

The *predecessor function* $pre_\mathcal{P} : 2^{\mathcal{C}_\mathcal{P}} \longrightarrow 2^{\mathcal{C}_\mathcal{P}}$ of $\mathcal{P}$ is defined as follows: $pre_\mathcal{P}(C) = \{c \in \mathcal{C}_\mathcal{P} \mid \exists c' \in C : c \Longrightarrow_\mathcal{P} c'\}$. The reflexive transitive closure of $pre_\mathcal{P}$ is denoted by $pre^*_\mathcal{P}$. Formally, $pre^*_\mathcal{P}(C) = \{c \in \mathcal{C}_\mathcal{P} \mid \exists c' \in C : c \Longrightarrow^*_\mathcal{P} c'\}$. Similarly, the *successor function* $post_\mathcal{P} : 2^{\mathcal{C}_\mathcal{P}} \longrightarrow 2^{\mathcal{C}_\mathcal{P}}$ of $\mathcal{P}$ is defined as follows: $post_\mathcal{P}(C) = \{c \in \mathcal{C}_\mathcal{P} \mid \exists c' \in C : c' \Longrightarrow_\mathcal{P} c\}$. The reflexive transitive closure $post^*_\mathcal{P}$ of $post_\mathcal{P}$ is defined as $post^*_\mathcal{P}(C) = \{c \in \mathcal{C}_\mathcal{P} \mid \exists c' \in C : c' \Longrightarrow^*_\mathcal{P} c\}$. From now on, we will drop the subscript $\mathcal{P}$ if $\mathcal{P}$ is clear from the context.

We now define two kinds of reachability problems:

Definition 3. Given a TrPDS $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$, let $C \subseteq \mathcal{C}_\mathcal{P}$ be a regular set of configurations (defined below): for $c \in \mathcal{C}_\mathcal{P}$,

- the *forward reachability problem* is to determine whether $c \in post^*_\mathcal{P}(C)$;
- the *backward reachability problem* is to determine whether $c \in pre^*_\mathcal{P}(C)$.

We will solve the forward (resp. backward) reachability problem by first computing $post^*_\mathcal{P}(C)$ (resp. $pre^*_\mathcal{P}(C)$) and then check whether $c \in post^*_\mathcal{P}(C)$ (resp. $c \in pre^*_\mathcal{P}(C)$).

Given two TrPDSs $\mathcal{P}$ and $\mathcal{P}'$, $\mathcal{P}'$ *encodes* $\mathcal{P}$ iff $\mathcal{C}_\mathcal{P} \subseteq \mathcal{C}_{\mathcal{P}'}$ and for every two configurations $c, c' \in \mathcal{C}_\mathcal{P}$, $c \Longrightarrow_\mathcal{P} c'$ iff $c \Longrightarrow^+_{\mathcal{P}'} c'$ such that the configurations used in the derivation of $c \Longrightarrow^+_{\mathcal{P}'} c'$ are in $\mathcal{C}_{\mathcal{P}'} \setminus \mathcal{C}_\mathcal{P}$.

Theorem 1. *Given a TrPDS $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$, we can construct a new TrPDS $\mathcal{P}' = (P, \Gamma', \mathcal{T} \cup \{\tau_{id}\}, \Delta')$ such that the following statements hold:*

(a) $|\omega| \leq 2$ *for every transition rule* $\langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle q, \omega\rangle \in \Delta'$*;*
(b) $\mathcal{P}'$ *encodes* $\mathcal{P}$*;*
(c) $|\Gamma'| = \mathbf{O}(|\Gamma| + |\Delta|)$ *and* $|\Delta'| = \mathbf{O}(|\Delta|)$.

Proof. Proof of (a).

Let Γ' and Δ' be the least sets such that

- $\Gamma \subseteq \Gamma'$;
- if $\langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle q, \omega\rangle \in \Delta$ with $|\omega| \leq 2$, then $\langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle q, \omega\rangle \in \Delta'$;
- if $r = \langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle q, \gamma_1\gamma_2 \cdots \gamma_n\rangle \in \Delta$ with $n > 2$, then add the fresh stack symbols $\{\gamma^r_1, \cdots, \gamma^r_{n-2}\}$ into Γ' and Δ' contains the following transition rules:
  - $\langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle q, \gamma^r_{n-2}\gamma_n\rangle$;
  - $\langle q, \gamma^r_i\rangle \overset{\tau_{id}}{\hookrightarrow} \langle q, \gamma^r_{i-1}\gamma_{i+1}\rangle$, for every $i : 2 \leq i \leq n-2$;
  - $\langle q, \gamma^r_1\rangle \overset{\tau_{id}}{\hookrightarrow} \langle q, \gamma_1\gamma_2\rangle$.

Proof of (b).

($\Longrightarrow$) Given two configurations $c, c' \in \mathcal{C}_{\mathcal{P}}$, suppose $c \Longrightarrow_{\mathcal{P}} c'$, we show that $c \Longrightarrow^{+}_{\mathcal{P}'} c'$ such that the configurations used in the derivation of $c \Longrightarrow^{+}_{\mathcal{P}'} c'$ are in $\mathcal{C}_{\mathcal{P}'} \setminus \mathcal{C}_{\mathcal{P}}$. Since $c \Longrightarrow_{\mathcal{P}} c'$, then, there exists a transition rule $r = \langle p, \gamma \rangle \overset{\tau}{\hookrightarrow} \langle q, \omega \rangle$ in Δ such that $c = \langle p, \gamma\omega' \rangle$ and $c' = \langle q, \omega u \rangle$ for some $u \in \tau(\omega')$.

If $|\omega| \leq 2$, then $\langle p, \gamma \rangle \overset{\tau}{\hookrightarrow} \langle q, \omega \rangle \in \Delta'$. Hence, we get that $c \Longrightarrow_{\mathcal{P}'} c'$. The result immediately follows.
Otherwise if $|\omega| > 2$, let $\omega = \gamma_1\gamma_2 \cdots \gamma_n$ with $n > 2$. Then, we get that for some $u \in \tau(\omega')$,

- $c = \langle p, \gamma\omega' \rangle \Longrightarrow_{\mathcal{P}'} \langle q, \gamma^r_{n-2}\gamma_n u \rangle$;
- $\langle q, \gamma^r_i \gamma_{i+2} \cdots \gamma_n u \rangle \Longrightarrow_{\mathcal{P}'} \langle q, \gamma^r_{i-1}\gamma_{i+1} \cdots \gamma_n u \rangle$ for every $i : 2 \leq i \leq n-2$;
- $\langle q, \gamma^r_1 \gamma_3 \cdots \gamma_n u \rangle \Longrightarrow_{\mathcal{P}'} \langle q, \gamma_1\gamma_2\gamma_3 \cdots \gamma_n u \rangle = c'$.

The result immediate follows.

($\Longleftarrow$) Given two configurations $c, c' \in \mathcal{C}_{\mathcal{P}}$, suppose $c \Longrightarrow^{+}_{\mathcal{P}'} c'$ such that the configurations used in the derivation of $c \Longrightarrow^{+}_{\mathcal{P}'} c'$ are in $\mathcal{C}_{\mathcal{P}'} \setminus \mathcal{C}_{\mathcal{P}}$, we show that $c \Longrightarrow_{\mathcal{P}} c'$.

If $c \Longrightarrow_{\mathcal{P}'} c'$, then the result immediately follows. Otherwise $c \Longrightarrow_{\mathcal{P}'} c_{n-2} \Longrightarrow_{\mathcal{P}'} \cdots \Longrightarrow_{\mathcal{P}'} c_1 \Longrightarrow_{\mathcal{P}'} c'$ with $n > 2$ and $c_i \in \mathcal{C}_{\mathcal{P}'} \setminus \mathcal{C}_{\mathcal{P}}$ for every $i : 1 \leq i \leq n-2$. Hence, for every $i : 1 \leq i \leq n-2$, the top of the stack of c_i is in $\Gamma' \setminus \Gamma$. According to the construction of $\mathcal{P}'$, there exists $t = \langle p, \gamma \rangle \overset{\tau}{\hookrightarrow} \langle q, \gamma_1\gamma_2 \cdots \gamma_n \rangle \in \Delta$ such that

- there exists $\langle p, \gamma \rangle \overset{\tau}{\hookrightarrow} \langle q, \gamma^t_{n-2}\gamma_n \rangle \in \Delta'$ such that $c = \langle p, \gamma\omega' \rangle$ and $c_{n-2} = \langle q, \gamma^t_{n-2}\gamma_n u \rangle$ for some $u \in \tau(\omega')$;
- and for every $i : 2 \leq i \leq n-2$, there exists $\langle q, \gamma^t_i \rangle \overset{\tau_{id}}{\hookrightarrow} \langle q, \gamma^t_{i-1}\gamma_{i+1} \rangle \in \Delta'$ such that $c_{i-1} = \langle q, \gamma^t_{i-1}\gamma_{i+1} \cdots \gamma_n u \rangle$;
- and there exists $\langle q, \gamma^t_1 \rangle \overset{\tau_{id}}{\hookrightarrow} \langle q, \gamma_1\gamma_2 \rangle \in \Delta'$ such that $c' = \langle q, \gamma_1\gamma_2 \cdots \gamma_n u \rangle$;

Hence, we get that the result.

Proof of (c).

The result follows the fact that for every transition rule $\langle p, \gamma \rangle \overset{\tau}{\hookrightarrow} \langle q, \gamma_1\gamma_2 \cdots \gamma_n \rangle \in \Delta$ with $n > 2$, we add at most $n-2$ new symbols into Γ' and at most $n-1$ new transition rules into Δ'. $\square$

2.3. Finite automata with transductions

To finitely represent regular sets of configurations of TrPDSs, we use finite automata with transductions.

Definition 4. Given a TrPDS $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$, a *finite automaton with transduction and ϵ-moves* (ϵ-TrNFA) **A** is a tuple $(S, \Gamma, \Lambda, \mathcal{T}, S_0, S_f)$, where S is a finite set of states, $\Lambda \subseteq S \times (\Gamma \cup \{\epsilon\}) \times \langle \mathcal{T} \rangle^{\cup} \times S$ is a finite set of transition rules, $S_0, S_f \subseteq S$ are initial and final states. An ϵ-TrNFA **A** is TrNFA if $\Lambda \subseteq S \times \Gamma \times \langle \mathcal{T} \rangle^{\cup} \times S$.

We write $s \overset{\gamma|\tau}{\longmapsto} s'$ for every $(s, \gamma, \tau, s') \in \Lambda$ (note that $\gamma \in \Gamma \cup \{\epsilon\}$). Let $\longmapsto^n: S \times \Gamma^* \times \langle \mathcal{T} \rangle^{\cup} \times S$ be the smallest relation over states of **A** defined as follows:

- $s \overset{\epsilon|\tau_{id}}{\longmapsto}{}^0 s$, for every $s \in S$;
- $s \xmapsto{\gamma\gamma_1\cdots\gamma_n|(\lceil\gamma_1\cdots\gamma_n, \gamma'_1\cdots\gamma'_n\rfloor^{-1}\tau)\circ\tau_1}{}^{n+1} s_2$ for all $\gamma_1, ..., \gamma_n \in \Gamma$, if $\exists s_1 \in S$ such that $s \overset{\gamma|\tau}{\longmapsto} s_1$ and $s_1 \xmapsto{\gamma'_1\cdots\gamma'_n|\tau_1}{}^n s_2$.

TrNFAs are the standard finite state automata if $\mathcal{T} = \{\tau_{id}\}$, a.k.a. $\mathcal{P}$-automata [4]. For the sake of simplification, we sometimes use $\gamma_{[m..n]}$ to denote the word $\gamma_m\gamma_{m+1} \cdots \gamma_n$, where $n \geq m$ and $\gamma_m, \gamma_{m+1}, \cdots, \gamma_n \in \Gamma \cup \{\epsilon\}$.

Proposition 4. *Suppose an ϵ-TrNFA **A** has $s_0 \xmapsto{\gamma^1_{[1..n]}|\tau}{}^n s_n$ with $n > 0$, then, there exist transition rules $s_{i-1} \xmapsto{\gamma^i_i|\tau_i} s_i$ in **A** for all $i \in \{1, \cdots, n\}$ such that*

$$\tau = (\lceil\gamma^1_{[2..n]}, \gamma^2_{[2..n]}\rfloor^{-1}\tau_1) \circ \cdots \circ (\lceil\gamma^{n-1}_n, \gamma^n_n\rfloor^{-1}\tau_{n-1}) \circ \tau_n.$$

Proof. Let us apply induction on n.

- **Basis:** $n = 1$. The result immediately follows from the fact that $s_0 \xmapsto{\gamma^1_1|\tau}{}^1 s_1$, i.e., **A** has a transition rule $s_0 \xmapsto{\gamma^1_1|\tau} s_1$.
- **Step:** $n > 1$. Then, there necessarily exist $s_0 \xmapsto{\gamma^1_1|\tau_1} s_1$ and $s_1 \xmapsto{\gamma^2_{[2..n]}|\tau'}{}^{n-1} s_n$ such that $\tau = (\lceil\gamma^1_{[2..n]}, \gamma^2_{[2..n]}\rfloor^{-1}\tau_1) \circ \tau'$.

  By applying the induction hypothesis: we get that there exist transition rules $s_{i-1} \xmapsto{\gamma^i_i|\tau_i} s_i$ in **A** for $i \in \{2, \cdots, n\}$ such that

$$\tau' = (\lceil \gamma^2_{[3..n]}, \gamma^3_{[3..n]} \rfloor^{-1}\tau_2) \circ \cdots \circ (\lceil \gamma^{n-1}_n, \gamma^n_n \rfloor^{-1}\tau_{n-1}) \circ \tau_n.$$

Thus, we get that

$$\tau = (\lceil \gamma^1_{[2..n]}, \gamma^2_{[2..n]} \rfloor^{-1}\tau_1) \circ (\lceil \gamma^2_{[3..n]}, \gamma^3_{[3..n]} \rfloor^{-1}\tau_2) \circ \cdots \circ (\lceil \gamma^{n-1}_n, \gamma^n_n \rfloor^{-1}\tau_{n-1}) \circ \tau_n. \quad \square$$

Proposition 5. *$\mathbf{A}$ has $s_0 \xmapsto{\gamma^1_{[1..n]}|\tau} {}^n s_n$ and $s_n \xmapsto{\gamma^{n+1}_{[n+1..m]}|\tau'} {}^{m-n} s_m$ with $m \geq n$, where*

- $\tau = (\lceil \gamma^1_{[2..n]}, \gamma^2_{[2..n]} \rfloor^{-1}\tau_1) \circ \cdots \circ (\lceil \gamma^{n-1}_n, \gamma^n_n \rfloor^{-1}\tau_{n-1}) \circ \tau_n$,
- $\tau' = (\lceil \gamma^{n+1}_{[n+2..m]}, \gamma^{n+2}_{[n+2..m]} \rfloor^{-1}\tau_{n+1}) \circ \cdots \circ (\lceil \gamma^{m-1}_m, \gamma^m_m \rfloor^{-1}\tau_{m-1}) \circ \tau_m$,

iff for every $\gamma^1_{n+1}, \cdots, \gamma^1_m, \gamma^n_{n+1} \cdots \gamma^n_m \in \Gamma \cup \{\epsilon\}$,

$$s_0 \xmapsto{\gamma^1_{[1..m]}|(\lceil \gamma^1_{[2..m]}, \gamma^2_{[2..m]} \rfloor^{-1}\tau_1) \circ \cdots \circ (\lceil \gamma^{m-1}_m, \gamma^m_m \rfloor^{-1}\tau_{m-1}) \circ \tau_m} {}^m s_m.$$

Proof. ($\Longrightarrow$) Let us apply induction on n.

- **Basis:** $n = 0$. Then, $s_0 \xmapsto{\epsilon|\tau_{id}} {}^0 s_0$. Hence, we get that

$$s_0 \xmapsto{\gamma^1_{[1..m]}|(\lceil \gamma^1_{[2..m]}, \gamma^2_{[2..m]} \rfloor^{-1}\tau_1) \circ \cdots \circ (\lceil \gamma^{m-1}_m, \gamma^m_m \rfloor^{-1}\tau_{m-1}) \circ \tau_m} {}^m s_m.$$

- **Step:** $n \geq 1$. Since $s_0 \xmapsto{\gamma^1_1 \cdots \gamma^1_n|\tau} {}^n s_n$, we get that $s_0 \xmapsto{\gamma^1_1|\tau_1} s_1$ and $s_1 \xmapsto{\gamma^2_2 \cdots \gamma^2_n|\tau''} {}^{n-1} s_n$ such that

$$\tau'' = (\lceil \gamma^2_{[3..n]}, \gamma^3_{[3..n]} \rfloor^{-1}\tau_2) \circ \cdots \circ (\lceil \gamma^{n-1}_n, \gamma^n_n \rfloor^{-1}\tau_{n-1}) \circ \tau_n.$$

  By applying the induction hypothesis: we get that

$$s_1 \xmapsto{\gamma^2_{[2..m]}|(\lceil \gamma^2_{[3..m]}, \gamma^3_{[3..m]} \rfloor^{-1}\tau_2) \circ \cdots \circ (\lceil \gamma^{m-1}_m, \gamma^m_m \rfloor^{-1}\tau_{m-1}) \circ \tau_m} {}^{m-1} s_m,$$

  for every $\gamma^2_{n+1}, \cdots, \gamma^2_m, \cdots, \gamma^n_{n+1}, \cdots, \gamma^n_m \in \Gamma \cup \{\epsilon\}$. Thus, the result immediately follows from the definition of $\longmapsto^n$.

($\Longleftarrow$) Let us apply induction on n.

- **Basis:** $n = 0$. Then, $s_0 \xmapsto{\epsilon|\tau_{id}} {}^0 s_0$ and $s_0 \xmapsto{\gamma^1_{[1..m]}|(\lceil \gamma^1_{[2..m]}, \gamma^2_{[2..m]} \rfloor^{-1}\tau_1) \circ \cdots \circ (\lceil \gamma^{m-1}_m, \gamma^m_m \rfloor^{-1}\tau_{m-1}) \circ \tau_m} {}^m s_m$.
- **Step:** $n \geq 1$. Then, from $s_0 \xmapsto{\gamma^1_{[1..m]}|(\lceil \gamma^1_{[2..m]}, \gamma^2_{[2..m]} \rfloor^{-1}\tau_1) \circ \cdots \circ (\lceil \gamma^{m-1}_m, \gamma^m_m \rfloor^{-1}\tau_{m-1}) \circ \tau_m} {}^m s_m$, we get that $s_1 \xmapsto{\gamma^2_{[2..m]}|(\lceil \gamma^2_{[3..m]}, \gamma^3_{[3..m]} \rfloor^{-1}\tau_2) \circ \cdots \circ (\lceil \gamma^{m-1}_m, \gamma^m_m \rfloor^{-1}\tau_{m-1}) \circ \tau_m} {}^{m-1} s_m$ and $s_0 \xmapsto{\gamma^1_1|\tau_1} s_1$.
  By applying the induction hypothesis:

$$s_n \xmapsto{\gamma^{n+1}_{[n+1..m]}|(\lceil \gamma^{n+1}_{[n+2..m]}, \gamma^{n+2}_{[n+2..m]} \rfloor^{-1}\tau_{n+1}) \circ \cdots \circ (\lceil \gamma^{m-1}_m, \gamma^m_m \rfloor^{-1}\tau_{m-1}) \circ \tau_m} {}^{m-n} s_m$$

  with $m \geq n$, and $s_1 \xmapsto{\gamma^2_{[2..n]}|(\lceil \gamma^2_{[3..m]}, \gamma^3_{[3..m]} \rfloor^{-1}\tau_2) \circ \cdots \circ (\lceil \gamma^{n-1}_n, \gamma^n_n \rfloor^{-1}\tau_{n-1}) \circ \tau_n} {}^{n-1} s_n$.
  Thus, we get that

$$s_0 \xmapsto{\gamma^1_{[1..m]}|(\lceil \gamma^1_{[2..m]}, \gamma^2_{[2..m]} \rfloor^{-1}\tau_1) \circ \cdots \circ (\lceil \gamma^{m-1}_m, \gamma^m_m \rfloor^{-1}\tau_{m-1}) \circ \tau_m} {}^m s_m. \quad \square$$

Let $\longmapsto^* = \bigcup_{i \geq 0} \longmapsto^i$. A configuration $\langle p, \omega \rangle \in P \times \Gamma^*$ of a TrPDS $\mathcal{P}$ is *recognized* (accepted) by an (ϵ-)TrNFA $\mathbf{A}$ iff $s \xmapsto{\omega|\tau} {}^* s'$ such that $s = p \in S_0$, $s' \in S_f$ and $(\epsilon, \epsilon) \in \tau$. Let $L(\mathbf{A})$ denote the set of configurations recognized by $\mathbf{A}$. A set C of configurations is *rational* (regular) if there exists an (ϵ-)TrNFA $\mathbf{A}$ such that $L(\mathbf{A}) = C$. From now on, we omit the paths of the form $s_1 \xmapsto{\omega|\tau} {}^n s_2$ such that $\tau = \emptyset$, as these paths do not allow the ϵ-TrNFA to accept a configuration.

Given a TrNFA $\mathbf{A}$ and a configuration $\langle p, \omega \rangle \in P \times \Gamma^*$, the *membership problem* is to determine whether $\langle p, \omega \rangle \in L(\mathbf{A})$. We will reduce the reachability problem of TrPDSs to the membership problem of TrNFAs. However, we cannot direct apply the decision procedure from finite-state automata to TrNFAs. Algorithm 1 presents a decision procedure for the membership problem of TrNFAs. The correctness proof is easy and omitted here. Left quotient of each transducer can be done in polynomial time of the transducer, and $|V|$ in Algorithm 1 is at most $\mathbf{O}(|S| \cdot |\Gamma|^n \cdot |\mathcal{T}|^n)$. Therefore, the space complexity of

```
Input  : A TrNFA $\mathbf{A} = (S, \Gamma, \Lambda, \mathcal{T}, S_0, S_f)$ and a configuration $\langle p, \gamma_1...\gamma_n\rangle$
Output: Yes if $\langle p, \gamma_1...\gamma_n\rangle \in L(\mathbf{A})$, otherwise No
1 if $p \notin S_0$ then return No;
2 $V := \{\ (p,\ \{(\gamma_1...\gamma_n, \gamma_1...\gamma_n)\})\ \}$;
3 for $(i := 1,\ i \leq n,\ i++)$ do
4     $V := \{(s', (\lceil \gamma_i, \gamma \rfloor^{-1}\tau) \circ \tau') \mid \exists (s, \tau) \in V,\ \exists s \xrightarrow{\gamma|\tau'} s' \in \Lambda\ s.t.\ (\gamma_i...\gamma_n, \gamma u) \in \tau\}$;
5 if $\exists (s, \tau) \in V\ s.t.\ s \in S_f \wedge (\epsilon, \epsilon) \in \tau$ then return Yes;
6 else return No;
```

Algorithm 1: Membership query of TrNFAs.

Algorithm 1 is $\mathbf{O}(|S| \cdot |\Gamma|^n \cdot |\mathcal{T}|^n)$ and the time complexity is $\mathbf{O}(n \cdot |S| \cdot |\Lambda| \cdot |\Gamma|^n \cdot |\mathcal{T}|^n)$, where n is the length of the stack content of the input configuration.

Remark 1. To check membership of a configuration, instead of recording pairs of states and transductions, we could record pairs of states and words from Γ^* at Line 2 and Line 4. Then, the space complexity is $\mathbf{O}(|S| \cdot |\Gamma|^n)$ and the time complexity is $\mathbf{O}(n \cdot |S| \cdot |\Lambda| \cdot |\Gamma|^n)$. In the rest of this article, we will heavily compute $\longmapsto^n$ which can be computed by leveraging Algorithm 1. Therefore, we present Algorithm 1 in the current style.

Algorithm 1 works for TrNFAs. In order to check membership problem of ϵ-TrNFAs and/or eliminate ϵ-transitions, we introduce a relation $\propto$ over two sets of transition rules, which can be used to eliminate ϵ-transition rules.

Definition 5. Given two sets of transition rules Λ and Λ' of ϵ-TrNFA, $\Lambda' \propto \Lambda$ iff the following conditions hold: where $\gamma \in \Gamma \cup \{\epsilon\}$,

(a) for every $q \overset{\gamma|\tau}{\longmapsto}{}^* q'$ in Λ, there exists $q \overset{\gamma|\tau'}{\longmapsto} q' \in \Lambda'$ such that $\tau \subseteq \tau'$;
(b) for every $q \overset{\gamma|\tau'}{\longmapsto} q' \in \Lambda'$, there exist $q \overset{\gamma|\tau_1}{\longmapsto}{}^* q', \cdots, q \overset{\gamma|\tau_n}{\longmapsto}{}^* q'$ in Λ such that $\tau' = \bigcup_{i=1}^{n} \tau_i$.

Intuitively, Item (a) expresses that ϵ-transition rules can be attracted into non-ϵ-transition rules (i.e., ϵ-transition rules can be eliminated) and Item (b) expresses that transition rules with same starting state, target state and input symbol, but different transductions, can be merged into one transition rule by taking the union of transductions. According to the following proposition, these operations preserve the recognized language, which allows us to reduce the number of transition rules.

Proposition 6. *Given two ϵ-TrNFAs $\boldsymbol{A} = (S, \Gamma, \Lambda, \mathcal{T}, S_0, S_f)$ and $\boldsymbol{A}' = (S', \Gamma', \Lambda', \mathcal{T}', S_0, S_f)$, if $\Lambda' \propto \Lambda$, then $L(\boldsymbol{A}) = L(\boldsymbol{A}')$.*

The proof is straightforward by induction on n for each $q \overset{\omega|\tau}{\longmapsto}{}^n q'$.

Theorem 2. *Given an ϵ-TrNFA $\boldsymbol{A}$ such that the closure of its transductions is finite, one can construct a finite state automaton $\boldsymbol{A}'$ such that $\boldsymbol{A}'$ exactly accepts the set of configurations $L(\boldsymbol{A})$.*

Proof. We first perform the ϵ-transition elimination which produces an equivalent TrNFA $\mathbf{A}_1$, then $\mathbf{A}_1$ can be transformed into an equivalent finite state automaton $\mathbf{A}'$ by the construction in [27]. The TrNFA $\mathbf{A}_1$ is constructed by applying the following procedure:

1. For each pair of transition rules: $q \overset{\epsilon|\tau_1}{\longmapsto} q_1$ and $q_1 \overset{\gamma|\tau_2}{\longmapsto} q_2$ with $\gamma \in \Gamma \cup \{\epsilon\}$, adds a new rule $q \overset{\gamma|\tau_1 \circ \tau_2}{\longmapsto} q_2$ into $\mathbf{A}$, until no new rule can be added;
2. For each rule $q \overset{\epsilon|\tau}{\longmapsto} q_1$, if $(\epsilon, \epsilon) \in \tau$ and q_1 is a final state, then sets q as a final state;
3. Removing all the ϵ-transition rules from $\mathbf{A}$ leads to the TrNFA $\mathbf{A}_1$.

It is easy to verify that $L(\mathbf{A}) = L(\mathbf{A}_1)$. Since the closure of transductions is finite, Item 1 always terminates which implies that the above procedure always terminates. The result immediately follows from the construction of [27]. □

By Theorem 2, one can translate an ϵ-TrNFA into an equivalent finite state automaton, while finite state automata can be seen as a special form of TrNFAs. Thus, ϵ-TrNFAs are closed under boolean operations.

Corollary 1. *ϵ-TrNFAs are closed under boolean operations.*

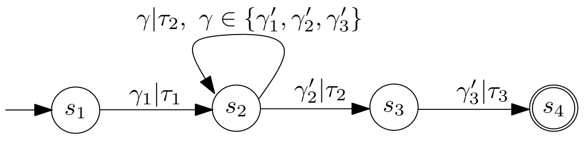


Fig. 1. Transition graph.

Remark 2. Note that it is possible to prove Corollary 1 without transforming ϵ-TrNFAs to finite state automata. Indeed, it is straightforward for the union. For intersection, one could compute a product of two ϵ-TrNFAs, in which the labeled transductions of each pair of transitions are also intersected.

Example 1. Let us consider the ϵ-TrNFA $\mathbf{A} = (S, \Gamma, \Lambda, \mathcal{T}, S_0, S_f)$, where $S = \{s_i \mid i = 1..4\}$, $S_0 = \{s_1\}$, $S_f = \{s_4\}$, $\Gamma = \{\gamma_i, \gamma_i' \mid i = 1..3\}$, $\mathcal{T} = \{\tau_1, \tau_2, \tau_{id}\}$ with $\tau_1 = \{(\gamma_2\omega, \gamma_2'\omega) \mid \omega \in \Gamma^*\}$, $\tau_2 = \{(\gamma_i\omega, \gamma_i'\omega) \mid i = 1..3,\ \omega \in \Gamma^*\}$, and

$$\Lambda = \left\{ \begin{array}{ll} (s_1, \gamma_1, \tau_1, s_2), & (s_2, \gamma_1', \tau_2, s_2), \\ (s_2, \gamma_2', \tau_2, s_2), & (s_2, \gamma_3', \tau_2, s_2), \\ (s_2, \gamma_2', \tau_2, s_3), & (s_3, \gamma_3', \tau_{id}, s_4) \end{array} \right\}.$$

Fig. 1 shows the transition graph of **A**. The language $L(\mathbf{A})$ of **A** is the set $\{\langle s_1, \gamma_1\gamma_2\omega\gamma_2\gamma_3\rangle \mid \omega \in \{\gamma_1, \gamma_2, \gamma_3\}^*\}$. Intuitively, consider a configuration $\langle s_1, \gamma_1\gamma_2\omega\gamma_2\gamma_3\rangle \in L(\mathbf{A})$, from the initial state s_1, there is only one transition $t_1 = (s_1, \gamma_1, \tau_1, s_2)$ which requires that γ_1 is the current input symbol. After t_1 is applied, τ_1 transforms the rest $\gamma_2\omega\gamma_2\gamma_3$ into $\tau_1(\gamma_2\omega\gamma_2\gamma_3) = \gamma_2'\omega\gamma_2\gamma_3$, for which the first symbol of $\gamma_2\omega\gamma_2\gamma_3$ should be γ_2 according to τ_1. Next, **A** iteratively reads $\gamma_2'\omega\gamma_2\gamma_3$ at the state s_2 until there are only two symbols in the rest of the input. To consume $\gamma_2'\omega$ at state s_2, ω should be a word from $\{\gamma_1, \gamma_2, \gamma_3\}^*$, as τ_2 transforms the next input γ_i into γ_i' for $i \in \{1, 2, 3\}$ without changing the rest. Finally, the input is transformed into γ_3' by τ_2 and **A** moves from the state s_2 to the state s_3. From s_3, **A** reads γ_3', transforms ϵ into the word ϵ by τ_{id} and moves to the accepting state s_4.

Theorem 3. [27] *Given a finite TrPDS $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$ and a rational set of configurations C of $\mathcal{P}$, both post*$^*(C)$ *and pre*$^*(C)$ *are rational and effectively computable.*

3. Computing *pre**

In this section, we present a saturation procedure to compute pre^* which is different from the way presented in [27] and a fixed-parameter tractable (FPT for short) algorithm of the saturation procedure that is suitable for implementation.

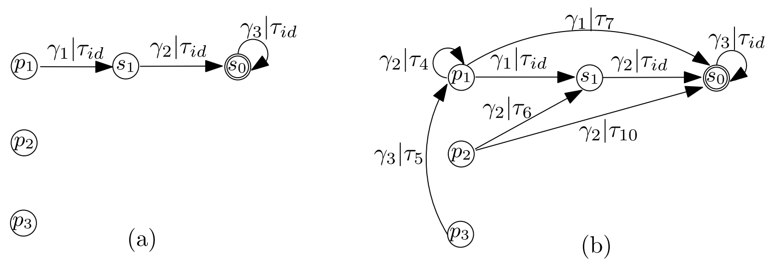
3.1. Saturation procedure for computing pre*

Given a finite TrPDS $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$ and a TrNFA $\mathbf{A} = (S, \Gamma, \Lambda, \mathcal{T}, S_0, S_f)$ that recognizes a rational set of configurations of $\mathcal{P}$, w.l.o.g., we assume $P = S_0$ and there is no transition rule in **A** leading to an initial state and **A** uses only the identity transduction τ_{id} (cf. Section 6.4 of [27]), we construct a new TrNFA $\mathbf{A}^{pre^*} = (S, \Gamma, \Lambda^{pre^*}, \mathcal{T}, S_0, S_f)$ such that $\mathbf{A}^{pre^*}$ recognizes $pre^*(L(\mathbf{A}))$, i.e., $L(\mathbf{A}^{pre^*}) = pre^*(L(\mathbf{A}))$. The construction of $\mathbf{A}^{pre^*}$ is based on a kind of saturation procedure which extends the saturation procedure for computing pre^* of PDSs [4]. Initially, $\mathbf{A}^{pre^*} = \mathbf{A}$, then we iteratively apply the following saturation procedure until no new transition rule can be added into $\mathbf{A}^{pre^*}$.

> If $\langle p, \gamma\rangle \overset{\tau_1}{\hookrightarrow} \langle q, \omega\rangle \in \Delta$ and $q \xmapsto{\omega|\tau_2}{}^* q'$ in the current automaton $\mathbf{A}^{pre^*}$, then, add a transition rule $p \xmapsto{\gamma|\tau_1\circ\tau_2} q'$ into Λ^{pre^*}.

Since the set of states of $\mathbf{A}^{pre^*}$ and the set $\langle\mathcal{T}\rangle$ of transductions are finite, the set of transition rules in $\mathbf{A}^{pre^*}$ is finite. Thus, the above saturation will eventually reach a fixpoint. Intuitively, if there is a transition rule $\langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle q, \gamma^1_{[1..n]}\rangle \in \Delta$, then $\langle p, \gamma\gamma_{[n+1..m]}\rangle \Longrightarrow \langle q, \gamma^1_{[1..m]}\rangle$, for all $\gamma^1_{[n+1..m]} \in \tau(\gamma_{[n+1..m]})$. If the automaton $\mathbf{A}^{pre^*}$ recognizes the configuration $\langle q, \gamma^1_{[1..m]}\rangle$ by a path $q \xmapsto{\gamma^1_{[1..m]}|\tau'}{}^m g$ for some final state g of $\mathbf{A}^{pre^*}$ and $(\epsilon, \epsilon) \in \tau'$, then, we can decompose this path to $q \xmapsto{\gamma^1_{[1..n]}|\tau''}{}^n q'$ and $q' \xmapsto{\gamma^{n+1}_{[n+1..m]}|\tau'''}{}^{m-n} g$ such that if $\tau' = (\lceil\gamma^1_{[2..m]}, \gamma^2_{[2..m]}\rfloor^{-1}\tau_1) \circ \cdots \circ (\lceil\gamma^{m-1}_m, \gamma^m_m\rfloor^{-1}\tau_{m-1}) \circ \tau_m$, then

- $\tau'' = (\lceil\gamma^1_{[2..n]}, \gamma^2_{[2..n]}\rfloor^{-1}\tau_1) \circ \cdots \circ (\lceil\gamma^{n-1}_n, \gamma^n_n\rfloor^{-1}\tau_{n-1}) \circ \tau_n$,
- $\tau''' = (\lceil\gamma^{n+1}_{[n+2..m]}, \gamma^{n+2}_{[n+2..m]}\rfloor^{-1}\tau_{n+1}) \circ \cdots \circ (\lceil\gamma^{m-1}_m, \gamma^m_m\rfloor^{-1}\tau_{m-1}) \circ \tau_m$.



$$\Delta = \left\{ \begin{array}{l} r_1 = \langle p_1, \gamma_1\rangle \overset{\tau_1}{\hookrightarrow} \langle p_2, \gamma_2\gamma_1\rangle, \\ r_2 = \langle p_2, \gamma_2\rangle \overset{\tau_2}{\hookrightarrow} \langle p_3, \gamma_3\gamma_1\rangle, \\ r_3 = \langle p_3, \gamma_3\rangle \overset{\tau_3}{\hookrightarrow} \langle p_1, \gamma_2\rangle, \\ r_4 = \langle p_1, \gamma_2\rangle \overset{\tau_4}{\hookrightarrow} \langle p_1, \epsilon\rangle \end{array} \right\}$$

(c)

$$\begin{array}{ll} \tau_1 = \{(\gamma_2\gamma_3\omega, \gamma_1\gamma_2\omega) \mid \omega \in \Gamma^*\}, & \tau_5 = \{(\gamma\gamma_1\gamma_3\omega, \gamma\gamma_2\gamma_3\omega) \mid \omega \in \Gamma^*, \gamma \in \Gamma\}, \\ \tau_2 = \{(\gamma\gamma_1\gamma_2\omega, \gamma\gamma_3\gamma_3\omega) \mid \omega \in \Gamma^*, \gamma \in \Gamma\}, & \tau_6 = \{(\gamma_1\gamma_1\gamma_2\omega, \gamma_2\gamma_3\gamma_3\omega) \mid \omega \in \Gamma^*\}, \\ \tau_3 = \{(\gamma\gamma_1\gamma_3\omega, \gamma\gamma_1\gamma_2\omega) \mid \omega \in \Gamma^*, \gamma \in \Gamma\}, & \tau_7 = \{(\gamma_2\gamma_3\omega, \gamma_3\gamma_3\omega) \mid \omega \in \Gamma^*\}, \\ \tau_4 = \{(\gamma\gamma_1\gamma_2\omega, \gamma\gamma_2\gamma_3\omega) \mid \omega \in \Gamma^*, \gamma \in \Gamma)\}, & \tau_8 = \{(\gamma_1\gamma_1\gamma_2\omega, \gamma_3\gamma_3\gamma_3\omega) \mid \omega \in \Gamma^*\}. \end{array}$$

(d)

Fig. 2. (a) The TrNFA **A**, (b) the TrNFA $\mathbf{A}^{pre^*}$, (c) the set of transition rules Δ, and (d) consists of related transductions.

Moreover, since $(\epsilon, \epsilon) \in \tau'$, we get that $(\gamma^1_{[n+1..m]}, \gamma^{n+1}_{[n+1..m]}) \in \tau''$ and $(\epsilon, \epsilon) \in \tau'''$. Applying the saturation procedure, the transition rule $p \xmapsto{\gamma|\tau\circ\tau''} q'$ is added into $\mathbf{A}^{pre^*}$. Therefore, $\mathbf{A}^{pre^*}$ recognizes the configuration $\langle p, \gamma\gamma_{[n+1..m]}\rangle$ by composing $p \xmapsto{\gamma|\tau\circ\tau''} q'$ and $q' \xmapsto{\gamma^{n+1}_{[n+1..m]}|\tau'''}{}^{m-n} g$ into

$$p \xmapsto{\gamma\gamma_{[n+1..m]}|\left(\lceil\gamma_{[n+1..m]}, \gamma^{n+1}_{[n+1..m]}\rfloor^{-1}(\tau\circ\tau'')\right)\circ\tau'''}{}^{m-n+1} g,$$

as $(\gamma_{[n+1..m]}, \gamma^{n+1}_{[n+1..m]}) \in \tau\circ\tau''$ implies that $(\epsilon, \epsilon) \in \left(\lceil\gamma_{[n+1..m]}, \gamma^{n+1}_{[n+1..m]}\rfloor^{-1}(\tau\circ\tau'')\right)\circ\tau'''$.

Example 2. Consider the TrPDS with control states $\{p_1, p_2, p_3\}$ and Δ as shown in Fig. 2(c). Let **A** be the TrNFA as shown in Fig. 2(a). The result automaton after applying the saturation procedure is shown in Fig. 2(b). The result is derived through the following steps:

1. First, since $p_1 \xmapsto{\epsilon|\tau_{id}}{}^* p_1$ and the right-hand side of the transition $r_4 \in \Delta$ is $\langle p_1, \epsilon\rangle$, the saturation procedure adds the transition rule $p_1 \xmapsto{\gamma_2|\tau_4} p_1$ (note that $\tau_4 \circ \tau_{id} = \tau_4$).
2. Then, there is $p_1 \xmapsto{\gamma_2|\tau_4}{}^* p_1$ and the right-hand side of the transition $r_3 \in \Delta$ is $\langle p_1, \gamma_2\rangle$, the saturation procedure adds the transition rule $p_3 \xmapsto{\gamma_3|\tau_5} p_1$, where $\tau_5 = \tau_3 \circ \tau_4 = \{(\gamma\gamma_1\gamma_3\omega, \gamma\gamma_2\gamma_3\omega) \mid \omega \in \Gamma^*, \gamma \in \Gamma\}$.
3. Now, there is $p_3 \xmapsto{\gamma_3\gamma_1|(\lceil\gamma_1,\gamma_1\rfloor^{-1}\tau_5)\circ\tau_{id}}{}^* s_1$ and the right-hand side of the transition $r_2 \in \Delta$ is $\langle p_3, \gamma_3\gamma_1\rangle$, the saturation procedure adds the transition rule $p_2 \xmapsto{\gamma_2|\tau_8} s_1$, where $\tau_6 = \tau_2 \circ (\lceil\gamma_1, \gamma_1\rfloor^{-1}\tau_5) \circ \tau_{id} = \{(\gamma_1\gamma_1\gamma_2\omega, \gamma_2\gamma_3\gamma_3\omega) \mid \omega \in \Gamma^*\}$.
4. This creates the path $p_2 \xmapsto{\gamma_2\gamma_1|(\lceil\gamma_1,\gamma_2\rfloor^{-1}\tau_6)\circ\tau_{id}}{}^* s_0$ which leads to the addition of the transition rule $p_1 \xmapsto{\gamma_1|\tau_7} s_0$ due to r_1, where $\tau_7 = \tau_1 \circ (\lceil\gamma_1, \gamma_2\rfloor^{-1}\tau_6) \circ \tau_{id} = \{(\gamma_2\gamma_3\omega, \gamma_3\gamma_3\omega) \mid \omega \in \Gamma^*\}$.
5. The previous addition creates the path $p_3 \xmapsto{\gamma_3\gamma_1|(\lceil\gamma_1,\gamma_1\rfloor^{-1}\tau_5)\circ\tau_7}{}^* s_0$ which allows the saturation procedure to add the transition rule $p_2 \xmapsto{\gamma_2|\tau_8} s_0$ because of r_2, where $\tau_8 = \tau_2 \circ (\lceil\gamma_1, \gamma_1\rfloor^{-1}\tau_5) \circ \tau_7 = \{(\gamma_1\gamma_1\gamma_2\omega, \gamma_3\gamma_3\gamma_3\omega) \mid \omega \in \Gamma^*\}$.
6. No more new transition rule can be added into $\mathbf{A}^{pre^*}$, so the procedure terminates.

Theorem 4. *Given a finite TrPDS $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$ and a rational set of configurations C of $\mathcal{P}$ recognized a TrNFA $\boldsymbol{A} = (S, \Gamma, \Lambda, \mathcal{T}, S_0, S_f)$, we can construct a TrNFA $\boldsymbol{A}'$ such that $L(\boldsymbol{A}') = pre^*(C)$ in time $\mathbf{O}(|\Delta|^3 \cdot |S|^3 \cdot |\Lambda| \cdot f(|\mathcal{T}|))$ and in space $\mathbf{O}(|\Delta| \cdot |S| \cdot |\langle\mathcal{T}\rangle|)$, where f is some computable function.*

The proof follows from the following two lemmas.

Lemma 1. *For every configuration $\langle q, v\rangle \in L(\mathbf{A})$, if $\langle p, \omega\rangle \Longrightarrow^* \langle q, v\rangle$, then $p \xmapsto{\omega|\tau} {}^* g$ for some final state g of $\mathbf{A}^{pre^*}$ and $(\epsilon, \epsilon) \in \tau$.*

Proof. Suppose $\langle p, \omega\rangle \Longrightarrow^n \langle q, v\rangle$, let us apply induction on n.

- **Basis:** $n = 0$. Then, $p = q$ and $\omega = v$. Since $\langle q, v\rangle \in L(\mathbf{A})$, we get that $q \xmapsto{v|\tau} {}^* g$ for some final state g of $\mathbf{A}^{pre^*}$ and $(\epsilon, \epsilon) \in \tau$. The result immediately follows.
- **Step:** $n > 0$. Then, there exists a configuration $\langle p', \omega'\rangle$ such that

$$\langle p, \omega\rangle \Longrightarrow^1 \langle p', \omega'\rangle \Longrightarrow^{n-1} \langle q, v\rangle.$$

By applying the induction hypothesis to $\langle p', \omega'\rangle \Longrightarrow^{n-1} \langle q, v\rangle$, we get that

$$p' \xmapsto{\omega'|\tau} {}^* g \text{ for some final state } g \text{ of } \mathbf{A}^{pre^*} \text{ and } (\epsilon, \epsilon) \in \tau.$$

Since $\langle p, \omega\rangle \Longrightarrow^1 \langle p', \omega'\rangle$, there exist $\gamma, \gamma_1, \cdots, \gamma_m, \gamma'_{k+1}, \cdots, \gamma'_m \in \Gamma$ such that $\omega = \gamma\gamma'_{[k+1..m]}$, $\omega' = \gamma_{[1..m]}$, $\langle p, \gamma\rangle \overset{\tau'}{\hookrightarrow} \langle p', \gamma_{[1..k]}\rangle \in \Delta$ and $\gamma_{[k+1..m]} \in \tau'(\gamma'_{[k+1..m]})$.

Suppose $\tau = (\lceil\gamma_{[2..m]}, \gamma^2_{[2..m]}\rfloor^{-1}\tau_1) \circ \cdots \circ (\lceil\gamma_m^{m-1}, \gamma_m^m\rfloor^{-1}\tau_{m-1}) \circ \tau_m$, by applying Proposition 5 to $p' \xmapsto{\omega'|\tau} {}^* g$, we obtain that

$$p' \xmapsto{\gamma_{[1..k]}|(\lceil\gamma_{[2..k]},\gamma^2_{[2..k]}\rfloor^{-1}\tau_1)\circ\cdots\circ(\lceil\gamma_k^{k-1},\gamma_k^k\rfloor^{-1}\tau_{k-1})\circ\tau_k} {}^k q'$$

and

$$q' \xmapsto{\gamma^{k+1}_{[k+1..m]}|(\lceil\gamma^{k+1}_{[k+2..m]},\gamma^{k+2}_{[k+2..m]}\rfloor^{-1}\tau_{k+1})\circ\cdots\circ(\lceil\gamma_m^{m-1},\gamma_m^m\rfloor^{-1}\tau_{m-1})\circ\tau_m} {}^{m-k} g.$$

$(\epsilon, \epsilon) \in \tau$, therefore

$$(\gamma_{[k+1..m]}, \gamma^{k+1}_{[k+1..m]}) \in \tau'' = (\lceil\gamma_{[2..k]}, \gamma^2_{[2..k]}\rfloor^{-1}\tau_1) \circ \cdots \circ (\lceil\gamma_k^{k-1}, \gamma_k^k\rfloor^{-1}\tau_{k-1}) \circ \tau_k$$

and

$$(\epsilon, \epsilon) \in \tau''' = (\lceil\gamma^{k+1}_{[k+2..m]}, \gamma^{k+2}_{[k+2..m]}\rfloor^{-1}\tau_{k+1}) \circ \cdots \circ (\lceil\gamma_m^{m-1}, \gamma_m^m\rfloor^{-1}\tau_{m-1}) \circ \tau_m.$$

By applying the saturation rule, $p \xmapsto{\gamma|\tau'\circ\tau''} q'$. Hence, we get that

$$p \xmapsto{\gamma\gamma'_{[k+1..m]}|(\lceil\gamma'_{[k+1..m]},\gamma^{k+1}_{[k+1..m]}\rfloor^{-1}(\tau'\circ\tau''))\circ\tau'''} {}^* g.$$

The result follows from the fact that $(\epsilon, \epsilon) \in \big(\lceil\gamma'_{[k+1..m]}, \gamma^{k+1}_{[k+1..m]}\rfloor^{-1}(\tau' \circ \tau'')\big) \circ \tau'''$. $\square$

Lemma 2. *If $p \xmapsto{\omega|\tau} {}^* q$ in $\mathbf{A}^{pre^*}$, then the following two properties hold:*

(a) *there exists some $p' \xmapsto{\omega'|\tau_{id}} {}^* q$ in $\mathbf{A}$ such that $\langle p, \omega u\rangle \Longrightarrow^* \langle p', \omega' u'\rangle$ for all $u' \in \tau(u)$;*
(b) *moreover, if q is an initial state, then $\omega' = \epsilon$.*

Proof. Let i be the number of transition rules added during the saturation procedure. Let $p \xmapsto{\omega|\tau} {}^*_i q$ denote a path which exists after adding the ith transition rule into $\mathbf{A}^{pre^*}$. Let us apply induction on i.

Basis: $i = 0$. Then, $p \xmapsto{\omega|\tau} {}^* q$ is in $\mathbf{A}$ which implies that $\tau = \tau_{id}$. Let $p = p', \omega = \omega'$ and $u = u'$. Since $\langle p, \omega u\rangle \Longrightarrow^* \langle p, \omega u\rangle$ always holds, then we get that $\langle p, \omega u\rangle \Longrightarrow^* \langle p', \omega' u'\rangle$ and $p' \xmapsto{\omega'|\tau_{id}} {}^*_0 q$ (i.e., $p' \xmapsto{\omega'|\tau_{id}} {}^* q$ in $\mathbf{A}$). Hence, the property (a) holds. The property (b) immediately follows from the fact that there is no transition rule in $\mathbf{A}$ leading to an initial state.

Step: $i \geq 1$. Let $t = q_1 \xmapsto{\gamma|\tau''} q'$ be the ith transition rule added into $\mathbf{A}^{pre^*}$. Let j be the number of times that t is used in the derivation of $p \xmapsto{\omega|\tau} {}^*_i q$. Let us apply a nested induction on j.

- **Basis:** $j = 0$. Then, $p \xmapsto{\omega|\tau} {}^*_{i-1} q$. The result immediately follows from the induction hypothesis on i.
- **Step:** $j \geq 1$. Then, there exist $\gamma_1^1, \cdots, \gamma_m^1 \in \Gamma$ such that $\omega = \gamma^1_{[1..m]}$. Suppose $\tau = (\lceil\gamma^1_{[2..m]}, \gamma^2_{[2..m]}\rfloor^{-1}\tau_1) \circ \cdots \circ (\lceil\gamma_m^{m-1}, \gamma_m^m\rfloor^{-1}\tau_{m-1}) \circ \tau_m$, then, $\gamma = \gamma_k^k$ and $\tau'' = \tau_k$ for some $k \in \{1, \cdots, m\}$. By applying Proposition 5, we get that

- $p \xmapsto{\gamma^1_{[1..k-1]}|(\lceil\gamma^1_{[2..k-1]},\gamma^2_{[2..k-1]}\rfloor^{-1}\tau_1)\circ\cdots\circ(\lceil\gamma^{k-2}_{k-1},\gamma^{k-1}_{k-1}\rfloor^{-1}\tau_{k-2})\circ\tau_{k-1}} {}^{*}_{i-1}\ q_1$,
- $q_1 \xmapsto{\gamma^k_k|\tau_k} q'$,
- $q' \xmapsto{\gamma^{k+1}_{[k+1..m]}|(\lceil\gamma^{k+1}_{[k+2..m]},\gamma^{k+2}_{[k+2..m]}\rfloor^{-1}\tau_{k+1})\circ\cdots\circ(\lceil\gamma^{m-1}_m,\gamma^m_m\rfloor^{-1}\tau_{m-1})\circ\tau_m} {}^{*}_{i}\ q$.

We notice that q_1 is an initial state, by applying the induction hypothesis (induction on i) to $p \xmapsto{\gamma^1_{[1..k-1]}|(\lceil\gamma^1_{[2..k-1]},\gamma^2_{[2..k-1]}\rfloor^{-1}\tau_1)\circ\cdots\circ(\lceil\gamma^{k-2}_{k-1},\gamma^{k-1}_{k-1}\rfloor^{-1}\tau_{k-2})\circ\tau_{k-1}} {}^{*}_{i-1}\ q_1$, we get that

$$\langle p, \gamma^1_{[1..m]}u\rangle \Longrightarrow^* \langle q_1, u'\rangle,$$

for all $u' \in \big(\lceil\gamma^1_{[2..k-1]}, \gamma^2_{[2..k-1]}\rfloor^{-1}\tau_1) \circ \cdots \circ (\lceil\gamma^{k-2}_{k-1}, \gamma^{k-1}_{k-1}\rfloor^{-1}\tau_{k-2}) \circ \tau_{k-1}\big)(\gamma^1_{[k..m]}u)$. Hence, we get that

$$\langle p, \gamma^1_{[1..m]}u\rangle \Longrightarrow^* \langle q_1, \gamma^k_{[k..m]}u'\rangle,$$

for all $u' \in \big(\lceil\gamma^1_{[2..m]}, \gamma^2_{[2..m]}\rfloor^{-1}\tau_1) \circ \cdots \circ \big(\lceil\gamma^k_{[k-1..m]}, \gamma^k_{[k..m]}\rfloor^{-1}\tau_{k-1}\big)(u)$.

Since the transition rule $q_1 \xmapsto{\gamma^k_k|\tau_k} q'$ is added by applying the saturation procedure, then there exists a transition rule $\langle q_1, \gamma^k_k\rangle \overset{\tau'_1}{\hookrightarrow} \langle p_2, \omega_2\rangle \in \Delta$ such that $p_2 \xmapsto{\omega_2|\tau''_1} {}^{*}_{i-1}q'$ and $\tau_k = \tau'_1 \circ \tau''_1$. Then, we get that $\langle q_1, \gamma^k_{[k..m]}u\rangle \Longrightarrow \langle p_2, \omega_2 u'\rangle$, for all $u' \in \tau'_1(\gamma^k_{[k+1..m]}u)$. Hence, we get that for every $\alpha^1_{n+1}, \cdots, \alpha^1_{m-k+n} \in \Gamma \cup \{\epsilon\}$,

$$\langle q_1, \gamma^k_{[k..m]}u\rangle \Longrightarrow^* \langle p_2, \omega_2\alpha^1_{[n+1..m-k+n]}u'\rangle,$$

for all $u' \in \lceil\gamma^k_{[k+1..m]}, \alpha^1_{[n+1..m-k+n]}\rfloor^{-1}\tau'_1(u)$.
Thus, we obtain that for every $\alpha^1_{n+1}, \cdots, \alpha^1_{m-k+n} \in \Gamma\cup\{\epsilon\}$ and $u' \in \big(\lceil\gamma^1_{[2..m]}, \gamma^2_{[2..m]}\rfloor^{-1}\tau_1) \circ\cdots\circ \big(\lceil\gamma^{k-1}_{[k..m]}, \gamma^k_{[k..m]}\rfloor^{-1}\tau_{k-1}\big) \circ \lceil\gamma^k_{[k+1..m]}, \alpha^1_{[n+1..m-k+n]}\rfloor^{-1}\tau'_1(u)$,

$$\langle p, \gamma^1_{[1..m]}u\rangle \Longrightarrow^* \langle p_2, \omega_2\alpha^1_{[n+1..m-k+n]}u'\rangle.$$

Putting $p_2 \xmapsto{\omega_2|\tau''_1} {}^{*}_{i-1}q'$ and $q' \xmapsto{\gamma^{k+1}_{[k+1..m]}|(\lceil\gamma^{k+1}_{[k+2..m]},\gamma^{k+2}_{[k+2..m]}\rfloor^{-1}\tau_{k+1})\circ\cdots\circ(\lceil\gamma^{m-1}_m,\gamma^m_m\rfloor^{-1}\tau_{m-1})\circ\tau_m} {}^{*}_{i}\ q$ together, we get that for every $\alpha^1_{n+1}, \cdots, \alpha^1_{m-k+n} \in \Gamma \cup \{\epsilon\}$,

$$p_2 \xmapsto{\omega_2\alpha^1_{[1..m-k+n]}|\tau'''} {}^{*}_{i}\ q,$$

where $\tau''' = (\lceil\alpha^1_{[n+1..m-k+n]}, \gamma^{k+1}_{[k+1..m]}\rfloor^{-1}\tau''_1) \circ (\lceil\gamma^{k+1}_{[k+2..m]}, \gamma^{k+2}_{[k+2..m]}\rfloor^{-1}\tau_{k+1}) \circ \cdots \circ (\lceil\gamma^{m-1}_m, \gamma^m_m\rfloor^{-1}\tau_{m-1}) \circ \tau_m$. Since the transition rule t is used in $p_2 \xmapsto{\omega_2\alpha^1_{[1..m-k+n]}|\tau'''} {}^{*}_{i}\ q$ less often than in $p \xmapsto{\omega|\tau} {}^{*}_{i}q$, by applying the induction hypothesis (induction on j) to $p_2 \xmapsto{\omega_2\alpha^1_{[1..m-k+n]}|\tau'''} {}^{*}_{i}\ q$, we get that for every $\alpha^1_{n+1}, \cdots, \alpha^1_{m-k+n} \in \Gamma \cup \{\epsilon\}$,

$$\langle p_2, \omega_2\alpha^1_{[n+1..m-k+n]}u\rangle \Longrightarrow^* \langle p', \omega'u'\rangle$$

such that $p' \xmapsto{\omega'|\tau_{id}} {}^{*}_{0}\ q$, for all $u' \in \tau'''(u)$.
Therefore, we get that for every $\alpha^1_{n+1}, \cdots, \alpha^1_{m-k+n} \in \Gamma \cup \{\epsilon\}$,

$$\langle q_1, \gamma^k_{[k..m]}u\rangle \Longrightarrow^* \langle p', \omega'u'\rangle$$

such that $p' \xmapsto{\omega'|\tau_{id}} {}^{*}_{0}\ q$, for all $u' \in \big(\lceil\gamma^1_{[2..m]}, \gamma^2_{[2..m]}\rfloor^{-1}\tau_1) \circ\cdots\circ \big(\lceil\gamma^{k-1}_{[k..m]}, \gamma^k_{[k..m]}\rfloor^{-1}\tau_{k-1}\big) \circ \big(\lceil\gamma^k_{[k+1..m]}, \alpha^1_{[n+1..m-k+n]}\rfloor^{-1}\tau'_1\big) \circ \tau'''(u)$.
We notice that

$$\bigcup_{\alpha^1_{n+1},\cdots,\alpha^1_{m-k+n}\in\Gamma} \big(\lceil\gamma^k_{[k+1..m]}, \alpha^1_{[n+1..m-k+n]}\rfloor^{-1}\tau'_1\big) \circ \tau''' = \left(\begin{array}{c} \big(\lceil\gamma^k_{[k+1..m]}, \gamma^{k+1}_{[k+1..m]}\rfloor^{-1}\tau_k\big)\circ \\ \big(\lceil\gamma^{k+1}_{[k+2..m]}, \gamma^{k+2}_{[k+2..m]}\rfloor^{-1}\tau_{k+1}\big)\circ \\ \cdots \\ (\lceil\gamma^{m-1}_m, \gamma^m_m\rfloor^{-1}\tau_{m-1})\circ \\ \tau_m \end{array}\right)$$

Thus, we get the result. □

```
Input : A finite TrPDS $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$ and a TrNFA $\mathbf{A} = (S, \Gamma, \Lambda, \mathcal{T}, S_0, S_f)$ such that $\mathbf{A}$ uses only $\tau_{id}$ and $\Lambda$ has no transition rule leading to a
        state in $P$
Output: The set of transition rules of $\mathbf{A}^{pre^*}$
$\Lambda' := \Lambda$; $trans := \Lambda$; $\Delta' := \emptyset$;
foreach $\langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle p', \epsilon\rangle \in \Delta$ do Update($p \xrightarrow{\gamma|\tau} p'$)
while $trans \neq \emptyset$ do
    remove $t = q \xrightarrow{\gamma|\tau} q'$ from $trans$;
    foreach $\langle p, \gamma_1\rangle \overset{\tau'}{\hookrightarrow} \langle q, \gamma\rangle \in \Delta \cup \Delta'$ do Update($p \xrightarrow{\gamma_1|\tau'\circ\tau} q'$);
    foreach $\langle p, \gamma_1\rangle \overset{\tau'}{\hookrightarrow} \langle q, \gamma\gamma_2\rangle \in \Delta$ and $\gamma_2' \in \Gamma$ do
        if $\lceil \gamma_2, \gamma_2' \rfloor^{-1}\tau \neq \emptyset$ then
            $\Delta' := \Delta' \cup \{\langle p, \gamma_1\rangle \xhookrightarrow{\tau'\circ(\lceil \gamma_2, \gamma_2' \rfloor^{-1}\tau)} \langle q', \gamma_2'\rangle\}$;
            foreach $q' \xrightarrow{\gamma_2'|\tau_2} q'' \in \Lambda'$ do
                Update($p \xrightarrow{\gamma_1|\tau'\circ(\lceil \gamma_2, \gamma_2' \rfloor^{-1}\tau)\circ\tau_2} q''$);
return $\Lambda'$;
Procedure Update ($q \xrightarrow{\gamma|\tau} q'$)
    if $t = q \xrightarrow{\gamma|\tau'} q' \in \Lambda'$ then
        $t' := q \xrightarrow{\gamma|\tau'\cup\tau} q'$;
        $\Lambda' := \Lambda' \cup \{t'\} \setminus \{t\}$ ;
        if $\tau' \neq \tau' \cup \tau$ then $trans := trans \cup \{t'\} \setminus \{t\}$;
    else if $\tau \neq \emptyset$ then
        $\Lambda' := \Lambda' \cup \{q \xrightarrow{\gamma|\tau} q'\}$;
        $trans := trans \cup \{q \xrightarrow{\gamma|\tau} q''\}$;
```

Algorithm 2: A FPT algorithm for computing pre^*.

We notice that the number $|\Lambda^{pre^*}|$ of transition rules of $\mathbf{A}^{pre^*}$ is at most $\mathbf{O}(|\Lambda| + |\Delta| \cdot |S| \cdot |\langle\mathcal{T}\rangle|)$. For each transition rule $\langle p, \gamma\rangle \overset{\tau_1}{\hookrightarrow} \langle q, \gamma_1\gamma_2\rangle \in \Delta$, paths $q \xrightarrow{\gamma_1\gamma_2|\tau_2}{}^* g$ can be computed in time $\mathbf{O}(f(|\mathcal{T}|) \cdot (|S| + |P|) \cdot |\Lambda^{pre^*}|)$ for some computable function f. Thus, we get that the saturation procedure executes at most in time $\mathbf{O}(|\Delta|^3 \cdot |S|^3 \cdot |\Lambda| \cdot f(|\mathcal{T}|))$. Memory is needed for storing the new transition rules which is bounded by $\mathbf{O}(|\Delta| \cdot |S| \cdot |\langle\mathcal{T}\rangle|)$.

Remark 3. In [27], the authors introduced TrNFAs and present a saturation procedure to compute pre^* without its complexity. They defined the relation $\longmapsto^*$ by introducing a *pseudo formal power series semiring* to solve the associativity problem of the composition of transitions of TrNFAs. Their saturation procedure proceeds based on this semiring. Using the pseudo formal power series semiring defined therein, one obtains only one transition by composing two transitions of TrNFAs. Otherwise, as defined in this article, the authors of [27] argued that finitely many transitions accrue by the composing of two transitions. Since the associativity of the composition of transitions does not hold, then the order of composition of transition rules is sensitive. For instance, consider two transition rules $s \xrightarrow{\gamma_1|\tau_1} s_1$ and $s_1 \xrightarrow{\gamma_2|\tau_2} s_2$, these two rules will yield $s \xrightarrow{\gamma_1\gamma_2'|(\lceil \gamma_2', \gamma_2 \rfloor^{-1}\tau_1)\circ\tau_2}{}^2 s_2$ for all $\gamma_2' \in \Gamma$. However, by looking inside of the saturation procedure of PDSs [4] as well as ours, one neither needs to compose all the pairs of transition rules of TrNFAs nor consider all the possible stack symbol γ_2' as in the above example. Indeed, during the saturation procedure or membership query, the input word ω is given, then the composition is carried out according to ω, similar to Algorithm 1. Therefore, the associativity of the composition is not a problem. Based on this observation, our saturation procedure directly proceeds based on TrNFAs and we show that this problem is fixed-parameter tractable (FPT). We believe our direct approach is more convenient for studying optimal algorithms or BDD-based symbolic techniques.

3.2. A FPT algorithm for computing pre^*

In this section, we present a FPT algorithm of the saturation procedure given in Section 3.1, which is suitable for implementation. Our algorithm is an extension of the efficient algorithm of the saturation procedure for standard pushdown systems [1]. W.l.o.g., we suppose in this section that for every TrPDS $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$, $|\omega| \leq 2$ for every transition rule $\langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle q, \omega\rangle \in \Delta$.

Algorithm 2 computes the transition rules of $\mathbf{A}^{pre^*}$ by implementing the saturation procedure from Section 3.1. The basic idea follows from the efficient algorithm for computing pre^* of PDSs [1] which avoids unnecessary operations. Intuitively, for the transition rules of the form $\langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle p', \epsilon\rangle$ or $\langle p, \gamma_1\rangle \overset{\tau'}{\hookrightarrow} \langle q, \gamma\rangle$ in Δ, the algorithm proceeds exactly the same as the saturation procedure given in Section 3.1. Whenever $\mathcal{P}$ has a transition rule in the form of $\langle p, \gamma_1\rangle \overset{\tau'}{\hookrightarrow} \langle q, \gamma\gamma_2\rangle$, we look out for every $q', q'' \in S$ and $\gamma_2' \in \Gamma$, the pairs of transition rules $q \xmapsto{\gamma|\tau} q'$ and $q' \xmapsto{\gamma_2'|\tau_2} q''$ such that $\lceil \gamma_2, \gamma_2' \rfloor^{-1}\tau \neq \emptyset$, so that we can add the transition rule $p \xmapsto{\gamma_1|\tau' \circ (\lceil \gamma_2, \gamma_2' \rfloor^{-1}\tau)\circ\tau_2} q''$. However, the order of such transitions added into the automaton $\mathbf{A}^{pre^*}$ can be arbitrary. Whenever a transition rule like $q' \xmapsto{\gamma_2'|\tau_2} q''$ is found, we have to check whether $q \xmapsto{\gamma|\tau} q'$ exists or not. Then, this checking may be negative, and wastes time to no avail. However, once a transition rule $q \xmapsto{\gamma|\tau} q'$ is seen, we know that all subsequent transitions like $q' \xmapsto{\gamma_2'|\tau_2} q''$ must lead to the addition of the transition rule $p \xmapsto{\gamma_1|\tau' \circ (\lceil \gamma_2, \gamma_2' \rfloor^{-1}\tau)\circ\tau_2} q'$. That's why we introduce a new transition rule $\langle p, \gamma_1 \rangle \xhookrightarrow{\tau' \circ (\lceil \gamma_2, \gamma_2' \rfloor^{-1}\tau)} \langle q', \gamma_2' \rangle$ into Δ' which allows us to add the transition rule $p \xmapsto{\gamma_1|\tau' \circ (\lceil \gamma_2, \gamma_2' \rfloor^{-1}\tau)\circ\tau_2} q''$ once $q' \xmapsto{\gamma_2'|\tau_2} q''$ occurs. Let us explain Algorithm 2 line by line as follows.

Line 1 initializes the algorithm by assigning Λ to Λ', Λ to *trans*, and $\emptyset$ to Δ'. Line 2 handles normal transition rules of the form $\langle p, \gamma \rangle \xhookrightarrow{\tau} \langle p', \epsilon \rangle$, where new transitions $p \xmapsto{\gamma|\tau} p'$ can be immediately added. Once a new transition rule is created, we call the procedure **Update** which will be explained later. Lines 3–10 iteratively removes a transition $t = q \xmapsto{\gamma|\tau} q'$ from *trans* until it is empty. The loop at Line 5 handles the case when q and γ match the right-hand side of transition rules in $\Delta \cup \Delta'$.

The procedure **Update** listed at Lines 12–19 is called whenever a new transition rule $q \xmapsto{\gamma|\tau} q'$ is created. If Λ' contains a transition rule of the form $t = q \xmapsto{\gamma|\tau'} q'$ for any τ', then, we remove t from Λ' and add a new transition rule $q \xmapsto{\gamma|\tau' \cup \tau} q'$ into Λ' at Line 15. In other words, we update the transduction τ' by $\tau' \cup \tau$. Moreover, if $\tau' \cup \tau$ does not equal to τ', we remove t from *trans* and add $q \xmapsto{\gamma|\tau' \cup \tau} q'$ into *trans* at Line 16 for later processing. Otherwise if Λ' has no transition rule like t, we add $q \xmapsto{\gamma|\tau} q'$ into Λ' and *trans*.

Theorem 5. *Given a finite TrPDS $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$ and a TrNFA $\mathbf{A} = (S, \Gamma, \Lambda, \mathcal{T}, S_0, S_f)$, we can compute a TrNFA $\mathbf{A}^{pre^*}$ in time $\mathbf{O}(|S|^2 \cdot f(|\mathcal{T}|) \cdot |\Delta| \cdot |\Gamma|)$ for some computable function f and in space $\mathbf{O}(|S| \cdot |\Delta| \cdot |\langle \mathcal{T} \rangle| \cdot |\Gamma|)$ such that $L(\mathbf{A}^{pre^*}) = pre^*(L(\mathbf{A}))$.*

The proof follows from the following three lemmas.

Lemma 3 *(Termination). Algorithm 2 always terminates.*

Proof. Λ' initially is Λ and can only grow afterwards. Since S and Γ are finite sets, and $\langle \mathcal{T} \rangle^{\cup}$ is finite, thus, the procedure **Update** cannot infinitely add transition rules into Λ'. This implies that the loops at Lines 3–10 and Lines 9–10 execute only finitely many times. Moreover, because Δ is finite, the loops at Line 2 and Lines 6–10 execute only finitely many times. Therefore, Δ' also is finite and the loop at Line 5 executes only finitely many times. Thus, Algorithm 2 always terminates. □

Lemma 4 *(Correctness). Upon termination of Algorithm 2, $\Lambda' \propto \Lambda^{pre^*}$.*

Proof. Since Λ' and Λ^{pre^*} do not have any ϵ-transition, we only need to show that

(a) for every $q \xmapsto{\gamma|\tau} q'$ in Λ^{pre^*}, there exists $q \xmapsto{\gamma|\tau'} q' \in \Lambda'$ such that $\tau \subseteq \tau'$;
(b) for every $q \xmapsto{\gamma|\tau'} q' \in \Lambda'$, there exist $q \xmapsto{\gamma|\tau_1} q', \cdots, q \xmapsto{\gamma|\tau_n} q'$ in Λ^{pre^*} such that $\tau' = \bigcup_{i=1}^{n} \tau_i$.

Proof of (a): We show that upon termination for every $q \xmapsto{\gamma|\tau} q' \in \Lambda^{pre^*}$, there exists $q \xmapsto{\gamma|\tau'} q' \in \Lambda'$ such that $\tau \subseteq \tau'$. Let us apply the induction on the number n of transition rules added into Λ^{pre^*} by applying the saturation procedure.

Basis: $n = 0$. The result follows from the fact that initially $\Lambda^{pre^*} = \Lambda \subseteq \Lambda'$.

Step: $n \geq 1$. Suppose $\langle p, \gamma \rangle \xhookrightarrow{\tau} \langle p', \omega \rangle \in \Delta$, there is $p' \xmapsto{\omega|\tau'}{}^{*} q'$ in Λ^{pre^*}, $p \xmapsto{\gamma|\tau\circ\tau'} q'$ is the nth transition rule added into Λ^{pre^*}. It is sufficient to show that Algorithm 1 calls the procedure **Update**($p \xmapsto{\gamma|\tau''} q'$) such that $\tau \circ \tau' \subseteq \tau''$.

- If $\omega = \epsilon$, then, $p' = q'$ and $\tau' = \tau_{id}$. The procedure **Update**($p \xmapsto{\gamma|\tau} p'$) is called at Line 2.
- If $\omega = \gamma'$, then, by applying the induction hypothesis to $p' \xmapsto{\omega|\tau'} q'$, we get that $p' \xmapsto{\omega|\tau_1'} q' \in \Lambda'$ such that $\tau' \subseteq \tau_1'$. Thus, the procedure **Update**($p \xmapsto{\gamma|\tau\circ\tau_1'} p'$) is called at Line 5. We notice that $\tau \circ \tau' \subseteq \tau \circ \tau_1'$.
- If $\omega = \gamma_1\gamma_2$, then Λ^{pre^*} contains two transition rules $t_1 = p' \xmapsto{\gamma_1|\tau_1} p''$ and $t_2 = p'' \xmapsto{\gamma_2'|\tau_2} q'$ such that $(\lceil \gamma_2, \gamma_2' \rfloor^{-1}\tau_1) \circ \tau_2 = \tau'$. By applying the induction hypothesis to t_1 and t_2, we get that Λ' contains two transition rules $t_1' = p' \xmapsto{\gamma_1|\tau_1'} p''$

and $t_2' = p'' \overset{\gamma_2'|\tau_2'}{\longmapsto} q'$ such that $\tau_1 \subseteq \tau_1'$ and $\tau_2 \subseteq \tau_2'$. Hence, Algorithm 1 adds a transition rule $\langle p, \gamma\rangle \xhookrightarrow{\tau\circ(\lceil\gamma_2,\gamma_2'\rfloor^{-1}\tau_1')} \langle p'', \gamma_2'\rangle$ into Δ' at Line 8.

If the transition rule t_2' is added before the addition of t_1', then, the procedure **Update**($p \xmapsto{\gamma|\tau\circ(\lceil\gamma_2,\gamma_2'\rfloor^{-1}\tau_1')\circ\tau_2'} p'$) is called at Line 10. Otherwise, if the transition rule t_2' is added after the addition of t_1', then, the procedure **Update**($p \xmapsto{\gamma|\tau\circ(\lceil\gamma_2,\gamma_2'\rfloor^{-1}\tau_1')\circ\tau_2'} p'$) is called at Line 5. We notice that $\tau\circ\tau' = \tau\circ(\lceil\gamma_2,\gamma_2'\rfloor^{-1}\tau_1)\circ\tau_2 \subseteq \tau\circ(\lceil\gamma_2,\gamma_2'\rfloor^{-1}\tau_1')\circ\tau_2'$.

Proof of (b): We show that throughout Algorithm 2, for every $q \overset{\gamma|\tau'}{\longmapsto} q' \in \Lambda'$, there exist $q \overset{\gamma|\tau_1}{\longmapsto} q', \cdots, q \overset{\gamma|\tau_n}{\longmapsto} q' \in \Lambda^{pre^*}$ such that $\tau = \bigcup_{i=1}^{n} \tau_i$. It is sufficient to show that whenever the procedure **Update**($p \overset{\gamma|\tau'}{\longmapsto} q'$) is called at Line 2 or Line 5 or Line 10, there exist $p \overset{\gamma|\tau_1}{\longmapsto} q', \cdots, p \overset{\gamma|\tau_n}{\longmapsto} q' \in \Lambda$ such that $\tau = \bigcup_{i=1}^{n} \tau_i$. Let us apply the induction on the number n of times that the procedure **Update**($q \overset{\gamma|\tau'}{\longmapsto} q'$) is called.

Basis: $n = 0$. The result follows from the fact that we initially have $\Lambda' = \Lambda \subseteq \Lambda^{pre^*}$.

Step: $n \geq 1$.

- If the procedure **Update**($p \overset{\gamma|\tau'}{\longmapsto} q'$) is called at Line 2, then, there is a transition rule $\langle p, \gamma\rangle \overset{\tau'}{\hookrightarrow} \langle q', \epsilon\rangle \in \Delta$. Hence, the saturation procedure adds a transition rule $p \overset{\gamma|\tau'}{\longmapsto} q'$ in Λ^{pre^*}. The result follows.
- If the procedure **Update**($p \overset{\gamma|\tau'}{\longmapsto} q'$) is called at Line 5, there is a transition rule $\langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle p', \gamma'\rangle \in \Delta \cup \Delta'$ and $p' \overset{\gamma'|\tau''}{\longmapsto} q'$ such that $\tau' = \tau\circ\tau''$. By applying the induction hypothesis to $p' \overset{\gamma'|\tau''}{\longmapsto} q'$: there are $p' \overset{\gamma'|\tau_1}{\longmapsto} q', \cdots, p' \overset{\gamma'|\tau_n}{\longmapsto} q' \in \Lambda^{pre^*}$ such that $\tau'' = \bigcup_{i=1}^{n} \tau_i$.
  - If $\langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle p', \gamma'\rangle \in \Delta$, then, the saturation procedure adds the transition rules $p \overset{\gamma|\tau\circ\tau_i}{\longmapsto} q'$ into Λ^{pre^*} for $i : 1 \leq i \leq n$. We notice that $\tau' = \tau\circ\tau'' = \tau\circ\bigcup_{i=1}^{n} \tau_i = \bigcup_{i=1}^{n} \tau\circ\tau_i$.
  - If $\langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle p', \gamma'\rangle \in \Delta'$, then, there is $\langle p, \gamma\rangle \overset{\tau_1'}{\hookrightarrow} \langle p'', \gamma_1\gamma_2\rangle \in \Delta$ and $p'' \overset{\gamma_1|\tau_2'}{\longmapsto} p'$ such that $\tau = \tau_1'\circ(\lceil\gamma_2, \gamma'\rfloor^{-1}\tau_2')$. By applying the induction hypothesis to $p'' \overset{\gamma_1|\tau_2'}{\longmapsto} p'$, there are $p'' \overset{\gamma_1|\tau_1^1}{\longmapsto} p', \cdots, p'' \overset{\gamma_1|\tau_{n_1}^1}{\longmapsto} p' \in \Lambda^{pre^*}$ such that $\tau_2' = \bigcup_{j=1}^{n_1} \tau_j^1$. Thus, for all $i : 1 \leq i \leq n$, $j : 1 \leq j \leq n_1$, the saturation procedure adds the transition rules $p \overset{\gamma|\tau_i^j}{\longmapsto} q'$ into Λ^{pre^*}, where $\tau_i^j = \tau_1'\circ(\lceil\gamma_2, \gamma'\rfloor^{-1}\tau_j^1)\circ\tau_i$. We notice that

$$\begin{aligned}
\tau' &= \tau\circ\tau'' \\
&= \tau_1'\circ(\lceil\gamma_2, \gamma'\rfloor^{-1}\tau_2')\circ\tau'' \\
&= \tau_1'\circ(\lceil\gamma_2, \gamma'\rfloor^{-1}\bigcup_{j=1}^{n_1}\tau_j^1)\circ\bigcup_{i=1}^{n}\tau_i \\
&= \bigcup_{1\leq i\leq n, 1\leq j\leq n_1} \tau_1'\circ(\lceil\gamma_2, \gamma'\rfloor^{-1}\tau_j^1)\circ\tau_i
\end{aligned}$$

- If the procedure **Update**($p \overset{\gamma|\tau'}{\longmapsto} q'$) is called at Line 10, then the result immediately follows in the same way as the procedure **Update**($p \overset{\gamma|\tau'}{\longmapsto} q'$) is called at Line 5 and $\langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle p', \gamma'\rangle \in \Delta'$. $\square$

Lemma 5 *(Complexity). Algorithm 2 takes* $\mathbf{O}(|S|^2 \cdot f(|\mathcal{T}|) \cdot |\Delta| \cdot |\Gamma|)$ *time for some computable function* f *and* $\mathbf{O}(|S| \cdot |\Delta| \cdot |\langle\mathcal{T}\rangle| \cdot |\Gamma|)$ *space.*

Proof. We suppose that Δ is organized in a hash table where the keys are the heads (q, γ') of transition rules $\langle p, \gamma\rangle \hookrightarrow \langle q, \gamma'\omega\rangle$. Similar, all the new transition rules in Δ' are put into this hash table at run-time. Thus, the addition of a new transition rule takes constant time. Suppose the time of all operations (i.e., left quotient, composition, union) on transductions used in Algorithm 2 is $f(|\mathcal{T}|)$ for some computable function f. The transductions of transition rules in Λ and Λ' are implemented as a function l (implemented as a hash table), e.g., $l(q \overset{\gamma}{\longmapsto} q') = \tau$ iff $q \overset{\gamma|\tau}{\longmapsto} q'$. Thus, Λ and Λ' are represented as sets Λ_1 and Λ_1' of transition rules of the form $q \overset{\gamma}{\longmapsto} q'$ equipped with the function l. Moreover, Λ_1 and Λ_1' are organized in a hash table, then addition and membership test take constant time. *trans* is implemented as a stack so that addition and removal of transitions (from the stack) take constant time, too. To avoid adding any transition to *trans* more than once, we store all transitions which are ever added to *trans* in an additional hash table. New transitions are added to the table only if they are not already in this table. According to the procedure **Update**, each transition rule of Λ' is added into *trans* once.

Let Δ_i denote $\{\langle p,\gamma\rangle \overset{\tau}{\hookrightarrow} \langle g,\omega\rangle \in \Delta \mid |\omega| = i\}$, for every $i \in \{0,1,2\}$.

Line 2 is executed at most $\mathbf{O}(|\Delta_0|)$ times. Hence, the number of transition rules added into *trans* by Line 2 is at most $\mathbf{O}(|\Delta_0|)$.

Line 8 is executed at most once for each combination of transition rules $q \xmapsto{\gamma|\tau} q'$, and $\langle p,\gamma_1\rangle \overset{\tau}{\hookrightarrow} \langle q,\gamma\gamma_2\rangle \in \Delta$, and stack symbol γ_2', i.e., $\mathbf{O}(|S| \cdot |\langle\mathcal{T}\rangle| \cdot |\Delta_2| \cdot |\Gamma|)$. Hence, the size of Δ' is at most $\mathbf{O}(|S| \cdot |\langle\mathcal{T}\rangle| \cdot |\Delta_2| \cdot |\Gamma|)$. The loop at Lines 9–10 is executed at most $\mathbf{O}(|S|^2 \cdot |\langle\mathcal{T}\rangle|^2 \cdot |\Delta_2| \cdot |\Gamma|)$.

Line 5 is executed at most once for each combination of transition rules $q \xmapsto{\gamma|\tau} q'$, transition rules $\langle p,\gamma_1\rangle \overset{\tau}{\hookrightarrow} \langle q,\gamma\rangle \in \Delta \cup \Delta'$. Since $|\Delta'|$ is at most $\mathbf{O}(|S| \cdot |\langle\mathcal{T}\rangle| \cdot |\Delta_2| \cdot |\Gamma|)$, Line 5 is executed at most $\mathbf{O}(|S| \cdot |\langle\mathcal{T}\rangle| \cdot (|\Delta_1| + |S| \cdot |\langle\mathcal{T}\rangle| \cdot |\Delta_2| \cdot |\Gamma|))$.

Now, let us examine the iterations of the loop at Lines 3–10. Initially, *trans* contains transition rules from Λ at Line 1 and transition rules added at Line 2 which is at most $|\Delta_0|$. The number of other transition rules added into *trans* is no more than $\mathbf{O}(|S| \cdot |\langle\mathcal{T}\rangle| \cdot (|\Delta_1| + |S| \cdot |\langle\mathcal{T}\rangle| \cdot |\Delta_2| \cdot |\Gamma|))$. Thus, Algorithm 2 is in time $\mathbf{O}(|S|^2 \cdot f(|\mathcal{T}|) \cdot |\Delta| \cdot |\Gamma|)$ for some computable function f.

During the execution of Algorithm 2, memory is needed for storing *trans*, Λ' and Δ'. Since the number of transition rules added into *trans* and Λ' is at most $\mathbf{O}(|S| \cdot |\Delta| \cdot |\langle\mathcal{T}\rangle|)$, and since $|\Delta'|$ is at most $\mathbf{O}(|S| \cdot |\langle\mathcal{T}\rangle| \cdot |\Delta_2| \cdot |\Gamma|)$, Algorithm 2 needs at most $\mathbf{O}(|S| \cdot |\Delta| \cdot |\langle\mathcal{T}\rangle| \cdot |\Gamma|)$ space. □

4. Computing *post**

In this section, we present an approach to compute *post** which is different from the way presented in [27]. In [27], *post** is computed by transforming a *finite* TrPDS into an equivalent PDS and then computing *post** of the resulting PDS. We will present a saturation procedure which directly computes *post** similar to computing *pre** given in Section 3. Finally, we give an FPT algorithm implementing this saturation procedure.

*4.1. Saturation procedure for computing post**

Given a finite TrPDS $\mathcal{P} = (P,\Gamma,\mathcal{T},\Delta)$ and a TrNFA $\mathbf{A} = (S,\Gamma,\Lambda,\mathcal{T},S_0,S_f)$ that recognizes a rational set of configurations of $\mathcal{P}$, we can construct an ϵ-TrNFA $\mathbf{A}^{post^*} = (S^{post^*},\Lambda^{post^*},S_0,S_f)$ such that $\mathbf{A}^{post^*}$ recognizes $post^*(L(\mathbf{A}))$, i.e., $L(\mathbf{A}^{post^*}) = post^*(L(\mathbf{A}))$. W.l.o.g., we assume that $S_0 = P$ and there is no transition rule in $\mathbf{A}$ leading to an initial state, $\mathbf{A}$ uses only the identity transduction τ_{id}, and $|\omega| \le 2$ for every transition rule $\langle p,\gamma\rangle \overset{\tau}{\hookrightarrow} \langle q,\omega\rangle \in \Delta$. The construction of $\mathbf{A}^{post^*}$ is similar to the construction of $\mathbf{A}^{pre^*}$ which is an extension of the saturation procedure for computing *post** of PDSs [1].

Given a transduction τ, let $\overline{\tau}$ denote the inversion $\{(\omega_1,\omega_2) \mid (\omega_2,\omega_1) \in \tau\}$ of τ, let $\overline{T}$ denote $\bigcup_{\tau\in T} \overline{\tau}$ for a given set T of transductions.

Proposition 7. *$\overline{\langle\mathcal{T}\rangle} = \langle\overline{\mathcal{T}}\rangle$ and $\overline{\langle\mathcal{T}\rangle^{\cup}} = \langle\overline{\mathcal{T}}\rangle^{\cup}$.*

Proof. The result immediately follows from the facts that:

- for every pair τ_1,τ_2 of transductions, $\overline{\tau_1 \circ \tau_2} = \overline{\tau_2} \circ \overline{\tau_1}$ and $\overline{\tau_1 \cup \tau_2} = \overline{\tau_2} \cup \overline{\tau_1}$;
- for every transduction τ, for all $\gamma_1,\gamma_2 \in \Gamma$, $\overline{\lceil\gamma_1,\gamma_2\rfloor^{-1}\tau} = \lceil\gamma_2,\gamma_1\rfloor^{-1}\overline{\tau}$. □

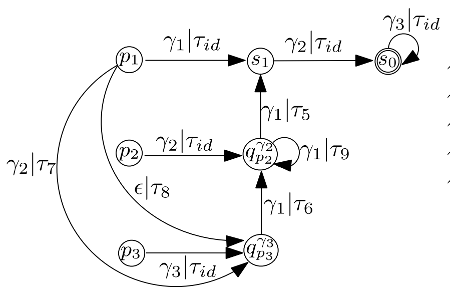
Initially, $\mathbf{A}^{post^*} = \mathbf{A}$, then we iteratively apply the following saturation procedure until the automaton is *saturated* (i.e., no new transition rule can be added):

(i) If $\langle p,\gamma\rangle \overset{\tau}{\hookrightarrow} \langle p',\epsilon\rangle \in \Delta$ and $p \xmapsto{\gamma|\tau'}{}^{*} s$, then add $p' \xmapsto{\epsilon|\ \overline{\tau}\circ\tau'} s$ into Λ^{post^*};

(ii) If $\langle p,\gamma\rangle \overset{\tau}{\hookrightarrow} \langle p',\gamma_1\rangle \in \Delta$ and $p \xmapsto{\gamma|\tau'}{}^{*} s$, then add $p' \xmapsto{\gamma_1|\ \overline{\tau}\circ\tau'} s$ into Λ^{post^*};

(iii) If $\langle p,\gamma\rangle \overset{\tau}{\hookrightarrow} \langle p',\gamma_1\gamma_2\rangle \in \Delta$ and $p \xmapsto{\gamma|\tau'}{}^{*} s$, then add a new state $q_{p'}^{\gamma_1}$ into S^{post^*}, add $p' \xmapsto{\gamma_1|\ \tau_{id}} q_{p'}^{\gamma_1}$ and $q_{p'}^{\gamma_1} \xmapsto{\gamma_2|\ \overline{\tau}\circ\tau'} s$ into Λ^{post^*}.

Intuitively, if there is a transition rule $\langle p,\gamma\rangle \overset{\tau}{\hookrightarrow} \langle p',\epsilon\rangle \in \Delta$, then $\langle p,\gamma\omega\rangle$ is an immediate predecessor of the configuration $\langle p',\omega_1\rangle$ for every $\omega_1 \in \tau(\omega)$. Thus, if the automaton already accepts the configuration $\langle p,\gamma\omega\rangle$ by $p \xmapsto{\gamma|\tau'}{}^{*} s$ and $s \xmapsto{\omega_2|\tau_2}{}^{*} q_f$ for some final state q_f, where $\omega_2 \in \tau'(\omega)$ and $(\epsilon,\epsilon) \in \tau_2$. Then it also ought to accept $\langle p',\omega_1\rangle$, for every $\omega_1 \in \tau(\omega)$. Adding the transition rule $p' \xmapsto{\epsilon|\ \overline{\tau}\circ\tau'} s$ allows the automaton to accept $\langle p',\omega_1\rangle$, for every $\omega_1 \in \tau(\omega)$, as



$$\tau_5 = \{(\gamma_1\gamma_2\omega, \gamma_2\gamma_3\omega) \mid \omega \in \Gamma^*\}$$
$$\tau_6 = \{(\gamma\gamma_3\gamma_3\omega, \gamma\gamma_1\gamma_2\omega) \mid \gamma \in \Gamma, \omega \in \Gamma^*\}$$
$$\tau_7 = \{(\gamma\gamma_1\gamma_2\omega, \gamma\gamma_1\gamma_3\omega) \mid \gamma \in \Gamma, \omega \in \Gamma^*\}$$
$$\tau_8 = \{((\gamma\gamma_2\gamma_3\omega, \gamma\gamma_1\gamma_3\omega) \mid \gamma \in \Gamma, \omega \in \Gamma^*\}$$
$$\tau_9 = \{(\gamma_1\gamma_2\gamma_3\omega, \gamma_1\gamma_1\gamma_2\omega) \mid \omega \in \Gamma^*\}$$

Fig. 3. The resulting TrNFA $\mathbf{A}^{post^*}$.

$\omega_2 \in (\overline{\tau} \circ \tau')(\omega_1)$. Moreover, to simplify the computing of $p \xmapsto{\gamma|\tau'}{}^* s$ which may involve some ϵ-transition rules, whenever a new transition rule is added, we could add a new transition which combines a ϵ-transition and a non-ϵ-transition due to Proposition 6. Then, we only need to check whether $p \xmapsto{\gamma|\tau'} s$ exists or not.

If there is a transition rule $\langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle p', \gamma_1\rangle \in \Delta$, then $\langle p, \gamma\omega\rangle$ is an immediate predecessor of the configuration $\langle p', \gamma_1\omega_1\rangle$ for every $\omega_1 \in \tau(\omega)$. Thus, if the automaton already accepts the configuration $\langle p, \gamma\omega\rangle$ by $p \xmapsto{\gamma|\tau'}{}^* s$ and $s \xmapsto{\omega_2|\tau_2}{}^* q_f$ for some final state q_f, where $\omega_2 \in \tau'(\omega)$ and $(\epsilon, \epsilon) \in \tau_2$. Then it also ought to accept $\langle p', \gamma_1\omega_1\rangle$, for every $\omega_1 \in \tau(\omega)$. Adding the transition rule $p' \xmapsto{\gamma_1|\ \overline{\tau}\circ\tau'} s$ allows the automaton to accept $\langle p', \gamma_1\omega_1\rangle$, for every $\omega_1 \in \tau(\omega)$, as $\omega_2 \in (\overline{\tau} \circ \tau')(\omega_1)$.

If there is a transition rule $\langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle p', \gamma_1\gamma_2\rangle \in \Delta$, then $\langle p, \gamma\omega\rangle$ is an immediate predecessor of the configuration $\langle p', \gamma_1\gamma_2\omega_1\rangle$ for every $\omega_1 \in \tau(\omega)$. Thus, if the automaton already accepts the configuration $\langle p, \gamma\omega\rangle$ by $p \xmapsto{\gamma|\tau'}{}^* s$ and $s \xmapsto{\omega_2|\tau_2}{}^* q_f$ for some final state q_f, where $\omega_2 \in \tau'(\omega)$ and $(\epsilon, \epsilon) \in \tau_2$. Then it also ought to accept $\langle p', \gamma_1\gamma_2\omega_1\rangle$, for every $\omega_1 \in \tau(\omega)$. Adding the transition rules $p' \xmapsto{\gamma_1|\ \tau_{id}} q_{p'}^{\gamma_1}$ and $q_{p'}^{\gamma_1} \xmapsto{\gamma_2|\ \overline{\tau}\circ\tau'} s$ allows the automaton to accept $\langle p', \gamma_1\gamma_2\omega_1\rangle$, for every $\omega_1 \in \tau(\omega)$, as $\omega_2 \in (\overline{\tau} \circ \tau')(\omega_1)$.

Example 3. Consider the TrPDS shown in Fig. 2(a) and the TrNFA **A** shown in Fig. 2(b). The result automaton after applying the saturation procedure is shown in Fig. 3. The automaton is constructed by the following steps:

1. First, $p_1 \xmapsto{\gamma_1|\tau_{id}} s_1$ matches the left-hand side of the transition rule $r_1 = \langle p_1, \gamma_1\rangle \overset{\tau_1}{\hookrightarrow} \langle p_2, \gamma_2\gamma_1\rangle$, we add two transition rules $p_2 \xmapsto{\gamma_2|\tau_{id}} q_{p_2}^{\gamma_2}$ and $q_{p_2}^{\gamma_2} \xmapsto{\gamma_1|\tau_5} s_1$, and add a new state $q_{p_2}^{\gamma_2}$, where $\tau_5 = \overline{\tau_1} \circ \tau_{id} = \{(\gamma_1\gamma_2\omega, \gamma_2\gamma_3\omega) \mid \omega \in \Gamma^*\}$;
2. The added transition rule $p_2 \xmapsto{\gamma_2|\tau_{id}} q_{p_2}^{\gamma_2}$ matches the left-hand side of the transition rule $r_2 = \langle p_2, \gamma_2\rangle \overset{\tau_2}{\hookrightarrow} \langle p_3, \gamma_3\gamma_1\rangle$, thus, we add two transition rules $p_3 \xmapsto{\gamma_3|\tau_{id}} q_{p_3}^{\gamma_3}$ and $q_{p_3}^{\gamma_3} \xmapsto{\gamma_1|\tau_6} q_{p_2}^{\gamma_2}$, and add a new state $q_{p_3}^{\gamma_3}$, where $\tau_6 = \overline{\tau_2} \circ \tau_{id} = \{(\gamma\gamma_3\gamma_3\omega, \gamma\gamma_1\gamma_2\omega) \mid \gamma \in \Gamma, \omega \in \Gamma^*\}$;
3. This in turn, together with $r_3 = \langle p_3, \gamma_3\rangle \overset{\tau_3}{\hookrightarrow} \langle p_1, \gamma_2\rangle$ leads to the addition of the rule $p_1 \xmapsto{\gamma_2|\tau_7} q_{p_3}^{\gamma_3}$, where $\tau_7 = \overline{\tau_3} \circ \tau_{id} = \{(\gamma\gamma_1\gamma_2\omega, \gamma\gamma_1\gamma_3\omega) \mid \gamma \in \Gamma, \omega \in \Gamma^*\}$;
4. We now can apply $p_1 \xmapsto{\gamma_2|\tau_7} q_{p_3}^{\gamma_3}$ to $r_4 = \langle p_1, \gamma_2\rangle \overset{\tau_4}{\hookrightarrow} \langle p_1, \epsilon\rangle$, we add $p_1 \xmapsto{\epsilon|\tau_8} q_{p_3}^{\gamma_3}$, where $\tau_8 = \overline{\tau_4} \circ \tau_7 = \{(\gamma\gamma_2\gamma_3\omega, \gamma\gamma_1\gamma_3\omega) \mid \gamma \in \Gamma, \omega \in \Gamma^*\}$;
5. We now have $p_1 \xmapsto{\gamma_1|(\lceil\gamma_1,\gamma_2\rfloor^{-1}\tau_8)\circ\tau_6}{}^* q_{p_2}^{\gamma_2}$, and can apply r_1 once more. Because, $p_2 \xmapsto{\gamma_2|\tau_{id}} q_{p_2}^{\gamma_2}$ is already added previously, we just add $q_{p_2}^{\gamma_2} \xmapsto{\gamma_1|\tau_9} q_{p_2}^{\gamma_2}$, where $\tau_9 = \overline{\tau_1} \circ (\lceil\gamma_1, \gamma_2\rfloor^{-1}\tau_8) \circ \tau_6 = \{(\gamma_1\gamma_2\gamma_3\omega, \gamma_1\gamma_1\gamma_2\omega) \mid \omega \in \Gamma^*\}$;
6. No more transition can be added, so the procedure terminates.

Theorem 6. *Given a finite TrPDS $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$ and a rational set of configurations C of $\mathcal{P}$ recognized a TrNFA $\mathbf{A} = (S, \Gamma, \Lambda, \mathcal{T}, S_0, S_f)$, we can construct a TrNFA $\mathbf{A}'$ such that $L(\mathbf{A}') = post^*(C)$.*

The proof follows from the following two lemmas.

Lemma 6. *For every configuration $\langle q, v\rangle \in L(\mathbf{A})$, if $\langle q, v\rangle \Longrightarrow^* \langle p, \omega\rangle$, then $p \xmapsto{\omega|\tau}{}^* g$ for some final state g of $\mathbf{A}^{post^*}$ and $(\epsilon, \epsilon) \in \tau$.*

Proof. Suppose $\langle q, v\rangle \Longrightarrow^n \langle p, \omega\rangle$, we proceed by induction on n.

Basis: $n = 0$. Then, $p = q$ and $\omega = v$. Since $\langle q, v\rangle \in L(\mathbf{A})$, we get that $q \xmapsto{v|\tau}{}^* g$ for some final state g of $\mathbf{A}^{post^*}$ and $(\epsilon, \epsilon) \in \tau$. The result immediately follows.

Step: $n > 0$. Then, there exists a configuration $\langle p', \omega' \rangle$ such that

$$\langle q, v \rangle \Longrightarrow^{n-1} \langle p', \omega' \rangle \Longrightarrow^{1} \langle p, \omega \rangle.$$

By applying the induction hypothesis to $\langle q, v \rangle \Longrightarrow^{n-1} \langle p', \omega' \rangle$, we get that

$$p' \overset{\omega'|\tau}{\longmapsto}{}^{*} g \text{ for some final state } g \text{ of } \mathbf{A}^{post^*} \text{ and } (\epsilon, \epsilon) \in \tau.$$

Since $\langle p', \omega' \rangle \Longrightarrow^{1} \langle p, \omega \rangle$, there exist $\gamma \in \Gamma$, $u_1, u'_1, \omega_1 \in \Gamma^*$ such that $\omega' = \gamma u_1$, $\omega = \omega_1 u'_1$, $\langle p', \gamma \rangle \overset{\tau'}{\hookrightarrow} \langle p, \omega_1 \rangle \in \Delta$ and $u'_1 \in \tau'(u_1)$. Hence, we can decompose $p' \overset{\omega'|\tau}{\longmapsto}{}^{*} g$ into $p' \overset{\gamma|\tau_1}{\longmapsto}{}^{*} q_1$ and $q_1 \overset{u''_1|\tau_2}{\longmapsto}{}^{*} g$ such that $\tau = (\lceil u_1, u''_1 \rfloor^{-1} \tau_1) \circ \tau_2$. Since $|\omega_1| \le 2$, we proceed by considering whether $|\omega_1|$ is 0, 1 or 2.

- If $|\omega_1| = 0$, then $\omega_1 = \epsilon$. By applying the saturation procedure, we have $p \overset{\epsilon|\overline{\tau'}\circ\tau_1}{\longmapsto} q_1$. Thus, we get that $p \overset{u'_1|(\lceil u'_1, u''_1 \rfloor^{-1}(\overline{\tau'}\circ\tau_1))\circ\tau_2}{\longmapsto}{}^{*} g$. We notice that $(\epsilon, \epsilon) \in (\lceil u'_1, u''_1 \rfloor^{-1}(\overline{\tau'} \circ \tau_1)) \circ \tau_2$.
- If $|\omega_1| = 1$, then suppose $\omega_1 = \gamma_1$. By applying the saturation procedure, we have $p \overset{\gamma_1|\overline{\tau'}\circ\tau_1}{\longmapsto} q_1$. Thus, we get that $p \overset{\gamma_1 u'_1|(\lceil u'_1, u''_1 \rfloor^{-1}(\overline{\tau'}\circ\tau_1))\circ\tau_2}{\longmapsto}{}^{*} g$. We notice that $(\epsilon, \epsilon) \in (\lceil u'_1, u''_1 \rfloor^{-1}(\overline{\tau'} \circ \tau_1)) \circ \tau_2$.
- If $|\omega_1| = 2$, then suppose $\omega_1 = \gamma_1\gamma_2$. By applying the saturation procedure, we have $p \overset{\gamma_1|\tau_{id}}{\longmapsto} q_p^{\gamma_1}$ and $q_p^{\gamma_1} \overset{\gamma_2|\overline{\tau'}\circ\tau_1}{\longmapsto} q_1$. Then, we have $p \overset{\gamma_1\gamma_2|\overline{\tau'}\circ\tau_1}{\longmapsto} q_1$. Thus, we get that $p \overset{\gamma_1\gamma_2 u'_1|(\lceil u'_1, u''_1 \rfloor^{-1}(\overline{\tau'}\circ\tau_1))\circ\tau_2}{\longmapsto}{}^{*} g$. We notice that $(\epsilon, \epsilon) \in (\lceil u'_1, u''_1 \rfloor^{-1}(\overline{\tau'} \circ \tau_1)) \circ \tau_2$. $\square$

Lemma 7. *If $p \overset{\omega|\tau}{\longmapsto}{}^{*} q$ in $\mathbf{A}^{post^*}$, then the following two properties hold:*

(a) *if q is some state of $\mathbf{A}$, then there exists a pair $(p', \omega') \in P \times \Gamma^*$ such that $p' \overset{\omega'|\tau_{id}}{\longmapsto}{}^{*} q$ in $\mathbf{A}$ and for all $(u, u') \in \tau$, $\langle p', \omega' u' \rangle \Longrightarrow^* \langle p, \omega u \rangle$;*
(b) *if $q = q_{p'}^{\gamma'}$, then for all $(u, u') \in \tau$, $\langle p', \gamma' u' \rangle \Longrightarrow^* \langle p, \omega u \rangle$.*

Proof. Let i be the number of transition rules added during the saturation procedure. Let $p \overset{\omega|\tau}{\longmapsto}{}^{*}_{i} q$ denote a path which exists after adding the ith transition rule into $\mathbf{A}^{post^*}$. Let us apply induction on i.

Basis: $i = 0$. Then, $p \overset{\omega|\tau}{\longmapsto}{}^{*}_{0} q$, which implies that $\tau = \tau_{id}$. Let $p = p'$ and $\omega = \omega'$. Since $\langle p, \omega u \rangle \Longrightarrow^* \langle p, \omega u \rangle$ always holds for all $u \in \Gamma^*$, then we get that

$$\langle p', \omega' u' \rangle \Longrightarrow^* \langle p, \omega u \rangle \text{ for all } (u', u) \in \tau_{id} \text{ and } p' \overset{\omega'|\tau_{id}}{\longmapsto}{}^{*}_{0} q \ (i.e.,\ p' \overset{\omega'|\tau_{id}}{\longmapsto}{}^{*} q \text{in} \mathbf{A}).$$

Hence, the property (a) holds. The property (b) immediately follows from the fact that $q \neq q_{p'}^{\gamma'}$ for any $p' \in P, \gamma' \in \Gamma$.

Step: $i \ge 1$. Let $t = q_1 \overset{\gamma|\tau'}{\longmapsto} q'$ be the ith transition rule added into $\mathbf{A}^{post^*}$. Let j be the number of times that t is used in the derivation of $p \overset{\omega|\tau}{\longmapsto}{}^{*}_{i} q$. Since there is no transition rule in $\mathbf{A}$ leading to an initial state of $\mathbf{A}$, and the saturation procedure does not add any such transitions too, then, if $q_1 \in S_0$, we get that t can be used at most once and only at the start of the path $p \overset{\omega|\tau}{\longmapsto}{}^{*} q$. Let us apply a nested induction on j. The result immediately follows from the induction hypothesis on i when $j = 0$. The rest is devoted to the case $j \ge 1$. We proceed by distinguishing three possible cases.

- If t is added by Item (i) or Item (ii), then, $p = q_1$. Suppose $\omega = \gamma u_1$, then

  $$\text{there exists } q' \overset{u'_1|\tau_1}{\longmapsto}{}^{*}_{i-1} q \text{ such that } \tau = (\lceil u_1, u'_1 \rfloor^{-1} \tau') \circ \tau_1.$$

  Suppose t is added by applying the saturation procedure to $\langle p_2, \gamma_2 \rangle \overset{\tau_2}{\hookrightarrow} \langle q_1, \gamma \rangle$, then we have $p_2 \overset{\gamma_2|\tau_3}{\longmapsto}{}^{*}_{i-1} q'$ such that $\tau' = \overline{\tau_2} \circ \tau_3$. Then,

  $$\tau = (\lceil u_1, u'_1 \rfloor^{-1} (\overline{\tau_2} \circ \tau_3)) \circ \tau_1 = \bigcup_{u''_1 \in \Gamma^{|u_1|}} (\lceil u_1, u''_1 \rfloor^{-1} \overline{\tau_2}) \circ (\lceil u''_1, u'_1 \rfloor^{-1} \tau_3) \circ \tau_1.$$

Hence, for all $u''_1 \in \Gamma^{|u_1|}$, there exists

$$p_2 \xmapsto{\gamma_2 u''_1|(\lceil u''_1, u'_1\rfloor^{-1}\tau_3)\circ\tau_1}{}^{*}_{i-1}\, q.$$

Note that $(\lceil u''_1, u'_1\rfloor^{-1}\tau_3)\circ\tau_1$ may be $\emptyset$ for some u''_1, but there always exists some $u''_1 \in \Gamma^{|u_1|}$ such that $(\lceil u''_1, u'_1\rfloor^{-1}\tau_3)\circ \tau_1 \neq \emptyset$ unless $\tau = \emptyset$.

By applying the induction hypothesis (on j) to $p_2 \xmapsto{\gamma_2 u''_1|(\lceil u''_1, u'_1\rfloor^{-1}\tau_3)\circ\tau_1}{}^{*}_{i-1}\, q$ for all $u''_1 \in \Gamma^{|u_1|}$ such that $(\lceil u''_1, u'_1\rfloor^{-1}\tau_3)\circ \tau_1 \neq \emptyset$, we get that for all $(u_2, u') \in (\lceil u''_1, u'_1\rfloor^{-1}\tau_3)\circ\tau_1$, there exists $(p', \omega') \in P \times \Gamma^*$ such that

$$\langle p', \omega' u'\rangle \Longrightarrow^* \langle p_2, \gamma_2 u''_1 u_2\rangle \text{ and } p' \xmapsto{\omega'|\tau_{id}}{}^{*}_{0}\, q.$$

Since $\langle p_2, \gamma_2\rangle \overset{\tau_2}{\hookrightarrow} \langle q_1, \gamma\rangle \in \Delta$, then, there exists $(p', \omega') \in P \times \Gamma^*$ such that for all $u \in \lceil u''_1, u_1\rfloor^{-1}\tau_2(u_2)$,

$$\langle p', \omega' u'\rangle \Longrightarrow^* \langle p_2, \gamma_2 u''_1 u_2\rangle \Longrightarrow^* \langle q_1, \gamma u_1 u\rangle = \langle p, \omega u\rangle.$$

Since $\tau = \bigcup_{u''_1\in\Gamma^{|u_1|}} (\lceil u_1, u''_1\rfloor^{-1}\overline{\tau_2}) \circ (\lceil u''_1, u'_1\rfloor^{-1}\tau_3)\circ\tau_1$, we get that for all $(u, u') \in \tau$, $u \in \lceil u''_1, u_1\rfloor^{-1}\tau_2(u_2)$ for some u''_1 and u_2. The result immediately follows.

Now, we consider the property (b). Suppose $q = q^{\gamma'}_{p'}$. By applying the induction hypothesis to $p_2 \xmapsto{\gamma_2 u''_1|(\lceil u''_1, u'_1\rfloor^{-1}\tau_3)\circ\tau_1}{}^{*}_{i-1}\, q$ for all $u''_1 \in \Gamma^{|u_1|}$ such that $(\lceil u''_1, u'_1\rfloor^{-1}\tau_3)\circ\tau_1 \neq \emptyset$, we get that for all $(u, u') \in (\lceil u''_1, u'_1\rfloor^{-1}\tau_3)\circ\tau_1$, $\langle p', \gamma' u'\rangle \Longrightarrow^* \langle p_2, \gamma_2 u''_1 u\rangle$. Thus, we get that for all $(u, u') \in \tau$ (note that $u \in \lceil u''_1, u_1\rfloor^{-1}\tau_2(u_2)$),

$$\langle p', \gamma' u'\rangle \Longrightarrow^* \langle p_2, \gamma_2 u''_1 u\rangle \Longrightarrow^* \langle q_1, \gamma u_1 u\rangle = \langle p, \omega u\rangle.$$

- If t is the first transition rule added in Item (*iii*), where $q' = q^{\gamma_1}_{p_1}$ and $\tau' = \tau_{id}$, then we proceed by considering whether t is new or not. If t already exists in $\mathbf{A}^{post^*}$, then the result immediately follows from the induction hypothesis on i. Otherwise if t is a new transition rule, then there is no transition rule starting from or leading to $q^{\gamma_1}_{p_1}$ at this moment. Thus, $p \xmapsto{\omega|\tau}{}^{*}\, q$ must be $p_1 \xmapsto{\gamma|\tau_{id}} q^{\gamma_1}_{p_1}$. This means that we only need to prove the property (b). The result follows from the fact that $\langle p', \gamma' u'\rangle \Longrightarrow^* \langle p', \gamma' u\rangle$, for all $p' \in P,\ \gamma' \in \Gamma$ and $(u, u') \in \tau_{id}$.
- If t is the second transition rule added in Item (*iii*), where $q_1 = q^{\gamma_1}_{p_1}$, then we can decompose $p \xmapsto{\omega|\tau}{}^{*}\, q$ into

$$p \xmapsto{u_1|\tau_1}{}^{*}_{i-1}\, q^{\gamma^1}_{p_1},\ q^{\gamma^1}_{p_1} \xmapsto{\gamma|\tau'} q', \text{ and } q' \xmapsto{v_1|\tau_2}{}^{*}_{i}\, q$$

such that $\tau = (\lceil \gamma_2 v_2, \gamma v'_1\rfloor^{-1}\tau_1)\circ(\lceil v'_1, v_1\rfloor^{-1}\tau')\circ\tau_2$ and $\omega = u_1\gamma_2 v_2$. By applying the induction hypothesis (on i) to $p \xmapsto{u_1|\tau_1}{}^{*}_{i-1}\, q^{\gamma^1}_{p_1}$, we get that

$$\langle p_1, \gamma_1 v\rangle \Longrightarrow^* \langle p, u_1 u\rangle, \text{ for all } (u, v) \in \tau_1.$$

Since t is added by applying the saturation procedure, then there are $\langle p_2, \gamma_2\rangle \overset{\tau_3}{\hookrightarrow} \langle p_1, \gamma_1\gamma\rangle \in \Delta$ and $p_2 \xmapsto{\gamma_2|\tau_4}{}^{*}_{i-1}\, q'$ such that $\tau' = \overline{\tau_3}\circ\tau_4$. Hence, we get that for all $v_3 \in \Gamma^{|v_1|}$, $p_2 \xmapsto{\gamma_2 v_3|(\lceil v_3, v_1\rfloor^{-1}\tau_4)\circ\tau_2}{}^{*}_{i-1}\, q$ (if it exists) uses t less often than in $p \xmapsto{\omega|\tau}{}^{*}\, q$.

By applying the induction hypothesis (on j) to $p_2 \xmapsto{\gamma_2 v_3|(\lceil v_3, v_1\rfloor^{-1}\tau_4)\circ\tau_2}{}^{*}_{i-1}\, q$, we get that if q is a state of $\mathbf{A}$, then there exists $(p', \omega') \in P \times \Gamma^*$ such that

$$\langle p', \omega' u'\rangle \Longrightarrow^* \langle p_2, \gamma_2 v_3 u_2\rangle \text{ and } p' \xmapsto{\omega'|\tau_{id}}{}^{*}_{0}\, q, \text{ for all } (u_2, u') \in (\lceil v_3, v_1\rfloor^{-1}\tau_4)\circ\tau_2.$$

Since

$$\begin{aligned}\tau &= (\lceil \gamma_2 v_2, \gamma v'_1\rfloor^{-1}\tau_1)\circ(\lceil v'_1, v_1\rfloor^{-1}\tau')\circ\tau_2\\ &= (\lceil \gamma_2 v_2, \gamma v'_1\rfloor^{-1}\tau_1)\circ(\lceil v'_1, v_1\rfloor^{-1}(\overline{\tau_3}\circ\tau_4))\circ\tau_2\\ &= (\lceil \gamma_2 v_2, \gamma v'_1\rfloor^{-1}\tau_1)\circ \bigcup_{v_3\in\Gamma^{|v_1|}} (\lceil v'_1, v_3\rfloor^{-1}\overline{\tau_3})\circ(\lceil v_3, v_1\rfloor^{-1}\tau_4)\circ\tau_2,\end{aligned}$$

then for every $(u, u') \in \tau$, there exist $\omega_1,\ u_2,\ v_3$ such that:
- $(u, \omega_1) \in \lceil \gamma_2 v_2, \gamma v'_1\rfloor^{-1}\tau_1$ (i.e., $(\gamma_2 v_2 u, \gamma v'_1\omega_1) \in \tau_1$);
- $(\omega_1, u_2) \in \lceil v'_1, v_3\rfloor^{-1}\overline{\tau_3}$ (i.e., $(v'_1\omega_1, v_3 u_2) \in \overline{\tau_3}$);
- $(u_2, u') \in (\lceil v_3, v_1\rfloor^{-1}\tau_4)\circ\tau_2$.

```
Input : A finite TrPDS $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$ and a TrNFA $\mathbf{A} = (S, \Gamma, \Lambda, \mathcal{T}, S_0, S_f)$ such that $\mathbf{A}$ uses only $\tau_{id}$, $\Lambda$ has no transition leading to a state in $P$
        and no $\epsilon$-transition
Output: The TrNFA $\mathbf{A}^{post^*} = (S^{post^*}, \Lambda', S_0, S_f)$
1  $\Lambda' := \Lambda$; $trans := \Lambda \cap P \times \Gamma \times \mathcal{T} \times S$;
2  $S^{post^*} := S \cup \{q_{p_1}^{\gamma_1} \mid \langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle p_1, \gamma_1\gamma_2\rangle \in \Delta\}$;
3  while $trans \neq \emptyset$ do
4      remove $t = p \xmapsto{\gamma|\tau} q$ from $trans$;
5      if $\gamma \neq \epsilon$ then
6          foreach $\langle p, \gamma\rangle \overset{\tau_1}{\hookrightarrow} \langle p_1, \epsilon\rangle \in \Delta$ do  Update($p_1 \xmapsto{\epsilon \ | \ \overline{\tau_1} \circ \tau} q$);
7          foreach $\langle p, \gamma\rangle \overset{\tau_1}{\hookrightarrow} \langle p_1, \gamma_1\rangle \in \Delta$ do  Update($p_1 \xmapsto{\gamma_1 \ | \ \overline{\tau_1} \circ \tau} q$);
8          foreach $\langle p, \gamma\rangle \overset{\tau_1}{\hookrightarrow} \langle p_1, \gamma_1\gamma_2\rangle \in \Delta$ do
9              Update($p_1 \xmapsto{\gamma_1 \ | \ \tau_{id}} q_{p_1}^{\gamma_1}$);
10             Update($q_{p_1}^{\gamma_1} \xmapsto{\gamma_2 \ | \ \overline{\tau_1} \circ \tau} q$);
11             foreach $p_2 \xmapsto{\epsilon \ | \ \tau_2} q_{p_1}^{\gamma_1} \in \Lambda'$, $\gamma_2' \in \Gamma$ do
12                 Update($p_2 \xmapsto{\gamma_2' \ | \ (\lceil \gamma_2', \gamma_2 \rfloor^{-1} \tau_2) \circ \overline{\tau_1} \circ \tau} q$)
13     else foreach $q \xmapsto{\gamma_1 \ | \ \tau_1} q' \in \Lambda'$, $\gamma_1' \in \Gamma$ do
14         Update($p \xmapsto{\gamma_1' \ | \ (\lceil \gamma_1', \gamma_1 \rfloor^{-1} \tau) \circ \tau_1} q'$)
15 return $\mathbf{A}^{post^*}$;
```

Algorithm 3: A FPT algorithm for computing *post**.

Therefore,

$$\langle p', \omega' u'\rangle \Longrightarrow^* \langle p_2, \gamma_2 v_3 u_2\rangle \Longrightarrow^* \langle p_1, \gamma_1 \gamma v_1' \omega_1\rangle \Longrightarrow^* \langle p, u_1 \gamma_2 v_2 u\rangle = \langle p, \omega u\rangle.$$

We consider now the property (b) and assume that $q = q_{p'}^{\gamma'}$. By applying the induction hypothesis (on j) to $p_2 \xmapsto{\gamma_2 v_3 | (\lceil v_3, v_1 \rfloor^{-1} \tau_4) \circ \tau_2}{}^*_{i-1} q$, for all $(u_2, u') \in (\lceil v_3, v_1 \rfloor^{-1} \tau_4) \circ \tau_2$,

$$\langle p', \gamma' u'\rangle \Longrightarrow^* \langle p_2, \gamma_2 v_3 u_2\rangle \text{ and } p' \xmapsto{\omega' | \tau_{id}}{}^*_0 q.$$

Thus, the property (b) holds. □

4.2. A FPT algorithm for computing post*

In this section, we present a FPT algorithm of the saturation procedure given in Section 4.1 which avoids unnecessary operations. Given a rational set of configurations of C represented by a TrNFA $\mathbf{A}$.

Algorithm 3 computes the transition rules of $\mathbf{A}^{post^*}$ by implementing the saturation procedure given in Section 4.1. The approach is similar to the solution for computing *pre**. We use *trans* to store the transition rules that we still need to examine. Lines 1–2 initialize the algorithm. Initially, Λ' is equal to Λ, while *trans* is equal to $\Lambda \cap P \times \Gamma \times \mathcal{T} \times S$, as transition rules starting from states outside of P do not need to be examined. The set S^{post^*} of states is equal to $S \cup \{q_{p_1}^{\gamma_1} \mid \langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle p_1, \gamma_1\gamma_2\rangle \in \Delta\}$ as described in Section 4.1. The algorithm iteratively removes a transition $t = p \xmapsto{\gamma|\tau} q$ from *trans* until it is empty. The loops at Line 6, Line 7 and Lines 8–10 handle the case when p and γ match the left-hand sides of transition rules in Δ. This is done similar to the saturation rules (i), (ii), and (iii), respectively. The loops at Lines 11–12 and Lines 13–14 handle ϵ-transition rules. In the saturation procedure given in Section 4.1, we have to compute paths $p \xmapsto{\gamma|\tau'}{}^* s$ which may involve several ϵ-transitions. In Algorithm 3, we solve this problem by combining transition pairs of the form $p \xmapsto{\epsilon|\tau_1} q_1$ and $q_1 \xmapsto{\gamma|\tau_2} q$ into transition rules $p \xmapsto{\gamma' | (\lceil \gamma', \gamma \rfloor^{-1} \tau_1) \circ \tau_2} q$ for $\gamma' \in \Gamma$ whenever such a pair is found.

Theorem 7. *Given a finite TrPDS $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$ and a TrNFA $\mathbf{A} = (S, \Gamma, \Lambda, \mathcal{T}, S_0, S_f)$, we can compute a TrNFA $\mathbf{A}^{post^*}$ in $\mathbf{O}(|S| \cdot f(|\mathcal{T}|) \cdot |\Delta|^3 \cdot |\Gamma|)$ time and space for some computable function f such that $L(\mathbf{A}^{post^*}) = post^*(L(\mathbf{A}))$.*

The proof immediate follows from the following results.

Claim. Let $S_1 = S \setminus P$, $S_2 = S^{post^*} \setminus S$, we have the following two properties:

1. $\Lambda' \subseteq \big(S \times \Gamma \times \langle \mathcal{T}\rangle^{\cup} \times (S_1 \cup S_2)\big) \cup \big(P \times \{\epsilon\} \times \langle \mathcal{T}\rangle^{\cup} \times (S_1 \cup S_2)\big)$;
2. $\Lambda' \setminus \Lambda \subseteq \big((P \cup S_2) \times \Gamma \times \langle \mathcal{T}\rangle^{\cup} \times (S_1 \cup S_2)\big) \cup \big(P \times \{\epsilon\} \times \langle \mathcal{T}\rangle^{\cup} \times (S_1 \cup S_2)\big)$;

Proof. Since there is no transition rule in **A** leading to a state in P and **A** has no ϵ-transition rule, we have $\Lambda \subseteq (P \cup S_1) \times \Gamma \times \mathcal{T} \times S_1$. Hence, before the loop at Lines 3–15, $\Lambda' \subseteq (P \cup S_1) \times \Gamma \times \mathcal{T} \times S_1$. The transition rules can be added into Λ' by Line 9 are elements of $(P \times \Gamma \times \mathcal{T} \times S_2)$. Therefore, the transition rules added by Line 6 (resp. Line 7) are elements of $P \times \{\epsilon\} \times \langle\mathcal{T}\rangle^{\cup} \times (S_1 \cup S_2)$ (resp. $P \times \Gamma \times \langle\mathcal{T}\rangle^{\cup} \times (S_1 \cup S_2)$). Similarly, the transition rules added by Line 10 are elements of $S_2 \times \Gamma \times \langle\mathcal{T}\rangle^{\cup} \times (S_1 \cup S_2)$. $\square$

Lemma 8 *(Termination). Algorithm 3 always terminates.*

Proof. Λ' initially is Λ and can only grow afterwards. Since S, $\langle\mathcal{T}\rangle^{\cup}$, Δ and Γ are finite sets, Algorithm 3 introduces new states $q_{p_1}^{\gamma_1}$ only if $\langle p, \gamma\rangle \overset{\tau_1}{\hookrightarrow} \langle p_1, \gamma_1\gamma_2\rangle \in \Delta$, then, the number of transition rules in Λ' is finite. From the definition the procedure **Update**, each transition rule in Λ' is examined at most once. Thus, Algorithm 3 always terminate. $\square$

We have shown that the saturation procedure given in Section 4.1 exactly computes $post*$. Hence, the correctness of Algorithm 3 follows from the following lemma.

Lemma 9 *(Correctness). Upon termination of Algorithm 3, $\Lambda' \propto \Lambda^{post^*}$.*

Proof. Since in the saturation procedure for $post*$ and Algorithm 3, all the added ϵ-transition rules are starting from states in P and leading to states in S or states of the form $q_{p_1}^{\gamma_1}$, then $p \xmapsto{\gamma|\tau}{}^{*} q'$ iff either $p \xmapsto{\gamma|\tau} q'$, or $p \xmapsto{\epsilon|\tau_1'} q_1 \wedge q_1 \xmapsto{\gamma'|\tau_1''} q' \in \wedge \tau = \lceil\gamma, \gamma'\rfloor^{-1}\tau_1' \circ \tau_1''$. Note that if $\gamma = \epsilon$, then $p \xmapsto{\gamma|\tau}{}^{*} q'$ iff $p \xmapsto{\gamma|\tau} q'$. Thus, due to the combination of transition pairs of the form $p \xmapsto{\epsilon|\tau_1} q_1$ and $q_1 \xmapsto{\gamma|\tau_2} q$ into transition rules $p \xmapsto{\gamma'|(\lceil\gamma',\gamma\rfloor^{-1}\tau_1)\circ\tau_2} q$, it is sufficient to show that for every $\gamma \in \Gamma \cup \{\epsilon\}$:

(a') if $p \xmapsto{\gamma|\tau} q'$ in Λ^{post^*}, then there exists $p \xmapsto{\gamma|\tau'} q' \in \Lambda'$ such that $\tau \subseteq \tau'$;

(b') if the addition of $p \xmapsto{\gamma|\tau'} q'$ into Λ' is made by calling the procedure **Update** at Line 6 or Line 7 or Line 9 or Line 10, then there exist $p \xmapsto{\gamma|\tau_1} q', \cdots, p \xmapsto{\gamma|\tau_n} q'$ in Λ^{post^*} such that $\tau' = \bigcup_{i=1}^{n} \tau_i$;

(c') the addition of $p \xmapsto{\gamma|\tau'} q'$ into Λ' is made by calling the procedure **Update** at Line 12 or Line 15, then there exist $p \xmapsto{\gamma|\tau_1}{}^{*} q', \cdots, p \xmapsto{\gamma|\tau_n}{}^{*} q'$ in Λ^{post^*} such that $\tau' = \bigcup_{i=1}^{n} \tau_i$.

Proof of (a'): Suppose $p \xmapsto{\gamma|\tau} q'$ in Λ^{post^*}, we show that there exists $p \xmapsto{\gamma|\tau'} q' \in \Lambda'$ such that $\tau \subseteq \tau'$ by induction on the number i of transition rules added by applying the saturation procedure. The result immediately follows from the fact that $\Lambda^{post^*} = \Lambda$ when $i = 0$. We only need to consider the case in which $i \geq 1$. We proceed by distinguishing three possible cases.

- If $t = p \xmapsto{\gamma|\tau} q' \in \Lambda^{post^*}$ is added by Item (i) or Item (ii), then there exists a transition rule $\langle p', \gamma'\rangle \overset{\tau_2'}{\hookrightarrow} \langle p, \gamma\rangle \in \Delta$ such that $p' \xmapsto{\gamma'|\tau_2''}{}^{*}_{n-1} q'$ in Λ^{post^*} and $\tau = \overline{\tau_2'} \circ \tau_2''$. By applying the induction hypothesis to $p' \xmapsto{\gamma'|\tau_2''}{}^{*}_{n-1} q'$ in Λ^{post^*}, there is $p' \xmapsto{\gamma'|\tau_3''} q' \in \Lambda'$ such that $\tau_2'' \subseteq \tau_3''$. Hence, Algorithm 3 will call the procedure **Update**$(p \xmapsto{\gamma|\overline{\tau_2'}\circ\tau_3''} q')$ at Line 6 or Line 7. Then, there exists $p \xmapsto{\gamma|\tau'} q' \in \Lambda'$ such that $\tau' \supseteq \overline{\tau_2'} \circ \tau_3'' \supseteq \overline{\tau_2'} \circ \tau_2'' = \tau$.
- If $t = p \xmapsto{\gamma|\tau} q' \in \Lambda^{post^*}$ is the first transition rule added in Item (iii), where $q' = q_p^{\gamma}$, then $\tau = \tau_{id}$ and there are $\langle p', \gamma'\rangle \overset{\tau_1}{\hookrightarrow} \langle p, \gamma\gamma_2\rangle \in \Delta$ such that $p' \xmapsto{\gamma'|\tau_1'}{}^{*}_{n-1} s$ in Λ^{post^*}. We proceed by distinguishing two possible cases.
  - Assume $p' \xmapsto{\gamma'|\tau_1'}{}^{*}_{n-1} s$ in Λ^{post^*} is derived from $p' \xmapsto{\gamma'|\tau_1'}_{n-1} s \in \Lambda^{post^*}$, by applying the induction hypothesis, we get that there exists $p' \xmapsto{\gamma'|\tau_2'} s$ in Λ' such that $\tau_1' \subseteq \tau_2'$. Therefore, Line 9 of Algorithm 3 will call **Update**$(p \xmapsto{\gamma|\tau_{id}} q')$. The result follows.
  - Assume $p' \xmapsto{\gamma'|\tau_1'}{}^{*}_{n-1} s$ in Λ^{post^*} is derived from $p' \xmapsto{\epsilon|\tau_2}_{n-1} q_1 \in \Lambda^{post^*}$ and $q_1 \xmapsto{\gamma_1|\tau_2'}_{n-1} s \in \Lambda^{post^*}$ such that $\tau_1' = (\lceil\gamma', \gamma_1\rfloor^{-1}\tau_2) \circ \tau_2'$, then by applying the induction hypothesis, there exists $p' \xmapsto{\epsilon|\tau_3} q_1 \in \Lambda'$ and $q_1 \xmapsto{\gamma_1|\tau_3'} s \in \Lambda'$ such that $\tau_2 \subseteq \tau_3$ and $\tau_2' \subseteq \tau_3'$. Hence, $p' \xmapsto{\gamma'|(\lceil\gamma',\gamma_1\rfloor^{-1}\tau_3)\circ\tau_3'}{}^{*} s$ in Λ'. Therefore, Line 9 of Algorithm 3 will call **Update**$(p \xmapsto{\gamma|} q')$. The result follows.
- If $t = p \xmapsto{\gamma|\tau} q' \in \Lambda^{post^*}$ is the second transition rule added in Item (iii), where $p = q_{p_1}^{\gamma_1}$, then there are $\langle p', \gamma'\rangle \overset{\tau_1}{\hookrightarrow} \langle p_1, \gamma_1\gamma\rangle \in \Delta$ and $p' \xmapsto{\gamma'|\tau_1'}{}^{*}_{n-1} s$ in Λ^{post^*} such that $\tau = \overline{\tau_1} \circ \tau_1'$. We proceed by distinguishing two possible cases.

– Assume $p' \xmapsto{\gamma'|\tau_1'}{}^*_{n-1} s$ in Λ^{post*} is derived from $p' \xmapsto{\gamma'|\tau_1'}{}_{n-1} s \in \Lambda^{post*}$, by applying the induction hypothesis, we get that there exists $p' \xmapsto{\gamma'|\tau_2'} s$ in Λ' such that $\tau_1' \subseteq \tau_2'$. Therefore, Line 9 of Algorithm 3 will call **Update**($p \xmapsto{\gamma|\overline{\tau_1}\circ\tau_2'} q'$). The result follows.

– Assume $p' \xmapsto{\gamma'|\tau_1'}{}^*_{n-1} s$ in Λ^{post*} is derived from $p' \xmapsto{\epsilon|\tau_2}{}_{n-1} q_1 \in \Lambda^{post*}$ and $q_1 \xmapsto{\gamma_1|\tau_2'}{}_{n-1} s \in \Lambda^{post*}$ such that $\tau_1' = (\lceil\gamma', \gamma_1\rfloor^{-1}\tau_2) \circ \tau_2'$, then by applying the induction hypothesis, there exists $p' \xmapsto{\epsilon|\tau_3} q_1 \in \Lambda'$ and $q_1 \xmapsto{\gamma_1|\tau_3'} s \in \Lambda'$ such that $\tau_2 \subseteq \tau_3$ and $\tau_2' \subseteq \tau_3'$. Hence, $p' \xmapsto{\gamma'|(\lceil\gamma', \gamma_1\rfloor^{-1}\tau_3)\circ\tau_3'} {}^* s$ in Λ'. Therefore, Line 10 of Algorithm 3 will call **Update**($q \xmapsto{\gamma|\overline{\tau_1}\circ(\lceil\gamma', \gamma_1\rfloor^{-1}\tau_3)\circ\tau_3'} q'$). The result follows.

Proof of (b') and (c'): We proceed by induction on the number i of transition rules added by Algorithm 3. The result immediately follows from the fact that $\Lambda' = \Lambda \subseteq \Lambda^{post*}$ when $i = 0$. We only need to consider the case in which $i \geq 1$.

Suppose the addition of $p \xmapsto{\gamma|\tau'} q'$ into Λ' is made by calling the procedure **Update** at Line 6 or Line 7 or Line 9 or Line 10, we show that there exist $p \xmapsto{\gamma|\tau_1} q', \cdots, p \xmapsto{\gamma|\tau_n} q'$ in Λ^{post*} such that $\tau' = \bigcup_{i=1}^n \tau_i$ by induction on the number i of transition rules added by Algorithm 3. The property (c') immediately follows from the property (b'). We proceed by distinguishing three possible cases.

- If $t = p \xmapsto{\gamma|\tau} q'$ is added by Line 6 or Line 7, then there exists a transition rule $\langle p', \gamma'\rangle \overset{\tau'}{\hookrightarrow} \langle p, \gamma\rangle \in \Delta$ such that $p' \xmapsto{\gamma'|\tau''}{}_{i-1} q' \in \Lambda'$ and $\tau = \overline{\tau'} \circ \tau''$. By applying the induction hypothesis, there are $p' \xmapsto{\gamma'|\tau_1} {}^* q', \cdots, p' \xmapsto{\gamma'|\tau_n} {}^* q'$ in Λ^{post*} such that $\tau'' = \bigcup_{j=1}^n \tau_j$. Then, the saturation rule adds $p \xmapsto{\gamma|\overline{\tau'}\circ\tau_j} q'$ into Λ^{post*} for every $j : 1 \leq j \leq n$. Hence, the property (b') holds.
- If $t = p \xmapsto{\gamma|\tau} q'$ is added by Line 9, then $q' = q_p^\gamma$ and there exists a transition rule $\langle p', \gamma'\rangle \overset{\tau'}{\hookrightarrow} \langle p, \gamma\gamma_1\rangle \in \Delta$ such that $p' \xmapsto{\gamma'|\tau''}{}_{i-1} s \in \Lambda'$. By applying the induction hypothesis, there are $p' \xmapsto{\gamma'|\tau_1} {}^* s, \cdots, p' \xmapsto{\gamma'|\tau_n} {}^* s$ in Λ^{post*} such that $\tau'' = \bigcup_{j=1}^n \tau_j$. Then, there is $p \xmapsto{\gamma|\tau_{id}} q_p^\gamma$ in Λ^{post*}. Hence, the property (b') holds.
- If $t = p \xmapsto{\gamma|\tau} q'$ is added by Line 10, then $p = q_{p_1}^{\gamma_1}$ and there are $\langle p', \gamma'\rangle \overset{\tau'}{\hookrightarrow} \langle p_1, \gamma_1\gamma\rangle \in \Delta$ and $p' \xmapsto{\gamma'|\tau''}{}_{i-1} q' \in \Lambda'$ such that $\tau = \overline{\tau'} \circ \tau''$. By applying the induction hypothesis, there are $p' \xmapsto{\gamma'|\tau_1} {}^* q', \cdots, p' \xmapsto{\gamma'|\tau_n} {}^* q'$ in Λ^{post*} such that $\tau'' = \bigcup_{j=1}^n \tau_j$. Then, there are $q_{p_1}^{\gamma_1} \xmapsto{\gamma'|\overline{\tau'}\circ\tau_1} {}^* q', \cdots, q_{p_1}^{\gamma_1} \xmapsto{\gamma'|\overline{\tau'}\circ\tau_n} {}^* q'$ in Λ^{post*}. Hence, the property (b') holds. □

Lemma 10 *(Complexity). Algorithm 3 takes* $\mathbf{O}(|S| \cdot f(|\mathcal{T}|) \cdot |\Delta|^3 \cdot |\Gamma|)$ *time and space for some computable function* f.

Proof. We suppose that Δ is organized in a hash table where the keys are the heads (p, γ) of transition rules $\langle p, \gamma\rangle \hookrightarrow \langle q, \gamma'\omega\rangle$. Λ' and Λ are also implemented as hash tables where the keys are the source states of transition rules in Λ' and Λ, so that all the needed addition, membership test and removal operations take constant time. The transductions of transition rules in Λ and Λ' are implemented as a function l (implemented as a hash table), e.g., $l(q \xmapsto{\gamma} q') = \tau$ iff $q \xmapsto{\gamma|\tau} q'$. Thus, Λ and Λ' are represented as sets Λ_1 and Λ_1' of transition rules of the form $q \xmapsto{\gamma} q'$ equipped with the function l. To avoid adding any transition to *trans* more than once, we store all transitions which are ever added to *trans* in an additional hash table. New transitions are added to the table only if they are not already in this table. According to the procedure **Update**, each transition rule of Λ' is added into *trans* once. Thus, for each transition rule in Λ', the main loop between Lines 4–15 is executed only once.

Line 6 is executed once for every combination of $\langle p, \gamma\rangle \overset{\tau_1}{\hookrightarrow} \langle p_1, \epsilon\rangle \in \Delta$ and $p \xmapsto{\gamma|\tau} q \in \Lambda'$, thus at most $\mathbf{O}(|\Delta_0| \cdot (|S \setminus P| + |S' \setminus S|))$ times, i.e., $\mathbf{O}(|\Delta_0| \cdot (|S \setminus P| + |\Delta_2|) \cdot |\langle\mathcal{T}\rangle|)$ times. Similarly, Line 7, Line 9 and Line 10 are executed at most $\mathbf{O}(|\Delta_1| \cdot (|S \setminus P| + |\Delta_2|) \cdot |\langle\mathcal{T}\rangle|)$ times, $\mathbf{O}(|\Delta_2| \cdot (|S \setminus P| + |\Delta_2|) \cdot |\langle\mathcal{T}\rangle|)$ and $\mathbf{O}(|\Delta_2| \cdot (|S \setminus P| + |\Delta_2|) \cdot |\langle\mathcal{T}\rangle|)$ times, respectively.

Since the number of ϵ-transition rules is at most $\mathbf{O}(|\Delta_0| \cdot (|S \setminus P| + |\Delta_2|) \cdot |\langle\mathcal{T}\rangle|)$, then, Line 12 is executed at most $\mathbf{O}(|\Delta_2| \cdot |\Delta_0| \cdot (|S \setminus P| + |\Delta_2|) \cdot |\langle\mathcal{T}\rangle| \cdot |\Gamma|)$ times.

Now, let us consider the loop at Lines 14–15. Since all the ϵ-transition rules are starting from states in P and leading to states in $S \setminus P$ or $S' \setminus S$. If $q \in S \setminus P$, then this loop executes at most $\mathbf{O}(|\Delta_0| \cdot |\langle\mathcal{T}\rangle| \cdot |\Lambda| \cdot |\Gamma|)$ times. If $q \in S' \setminus S$, then this loop executes at most $\mathbf{O}(|\Delta_0| \cdot |\Delta_2| \cdot |\langle\mathcal{T}\rangle| \cdot |\Lambda| \cdot (|S \setminus P| + |\Delta_2|) \cdot |\Gamma|)$ times.

Line 4 is executed at most once for every transition rule in *trans*, i.e., $\mathbf{O}((|S \setminus P| + |\Delta_2|) \cdot |\langle\mathcal{T}\rangle| \cdot (|\Lambda| + |\Delta_0| \cdot |\Delta_2| \cdot |\Gamma|))$.

Thus, Algorithm 3 needs at most $\mathbf{O}(|S| \cdot f(|\mathcal{T}|) \cdot |\Delta|^3 \cdot |\Gamma|)$ time and space for some computable function f. □

Remark 4. Solving the forward (resp. backward) reachability problem consists of two parts: first computing $post^*_{\mathcal{P}}(C)$ (resp. $pre^*_{\mathcal{P}}(C)$) and then checking whether $c \in post^*_{\mathcal{P}}(C)$ (resp. $c \in pre^*_{\mathcal{P}}(C)$). The latter problem is solved by Algorithm 1 which is in exponential time of the length of the stack content of the input configuration c. Consequently, it is efficient to decide whether $c \in post^*_{\mathcal{P}}(C)$ (resp. $c \in pre^*_{\mathcal{P}}(C)$) if the length of the stack content of the input configuration c is small.

5. Computing *pre** and *post** for weak finite TrPDSs

In Section 3 and Section 4, for finite TrPDSs, we presented saturation procedures and their FPT algorithms to compute $pre^*(C)$ and $post^*(C)$ of a given rational set C of configurations. These results are consistent with the ones of [27]. By carefully examining the algorithms, we found that for each left quotient operation $\lceil \gamma, \gamma' \rfloor^{-1}\tau$ used in Algorithm 2 (resp. Algorithm 3), there must exist a transition rule $\langle p, \gamma_1 \rangle \overset{\tau'}{\hookrightarrow} \langle q, \gamma_2\gamma_3 \rangle \in \Delta$ such that $\gamma_3 = \gamma$ (resp. $\gamma_3 = \gamma'$). In other words, there is no left quotient operation $\lceil \gamma, \gamma' \rfloor^{-1}\tau$ in Algorithm 2 (resp. Algorithm 3) if Δ does not contain a transition rule of the form $\langle p, \gamma_1 \rangle \overset{\tau'}{\hookrightarrow} \langle q, \gamma_2\gamma \rangle$ (resp. $\langle p, \gamma_1 \rangle \overset{\tau'}{\hookrightarrow} \langle q, \gamma_2\gamma' \rangle$). However, in the definition of finiteness of TrPDSs, the full stack alphabet is considered in the left quotient. This means that when the infinity of TrPDSs comes from computing left quotient *irrespective* of how the stack symbols are actually used, $pre^*(C)$ and $post^*(C)$ can still be computed via Algorithm 2 and Algorithm 3, respectively. In this section, we first define the closure $\langle \mathcal{T} \rangle^{\cup}$ of transductions $\mathcal{T}$ with respect to TrPDSs that precisely characterizes the "wanted" sets of transductions. Based on this new definition of closure, we introduce *weak finite* TrPDSs which is more general than finite TrPDSs. Moreover, $pre^*(C)$ and $post^*(C)$ of a given rational set C of configurations for a weak finite TrPDS are also rational and effectively computable. Indeed, the computing of $pre^*(C)$ and $post^*(C)$ for a weak finite TrPDS can be done using Algorithm 2 and Algorithm 3 presented in Section 3 and Section 4, respectively.

Given a TrPDS $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$, let Γ_2 denote the set $\{\gamma_2 \mid \langle p, \gamma \rangle \overset{\tau}{\hookrightarrow} \langle p_1, \gamma_1\gamma_2 \rangle \in \Delta\}$, the closure $\langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$ of $\mathcal{T}$ over the composition $\circ$, left quotient $\lceil \cdot, \cdot \rfloor^{-1}$ and union $\cup$ with respect to $\mathcal{P}$ is defined as follows:

- $\mathcal{T} \subseteq \langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$, $\emptyset \in \langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$ and $\tau_{id} \in \langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$;
- if $\tau_1, \tau_2 \in \langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$, then $\tau_1 \circ \tau_2 \in \langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$ and $\tau_1 \cup \tau_2 \in \langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$;
- if $\tau \in \langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$, then $\lceil \gamma, \gamma' \rfloor^{-1}\tau \in \langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$ for all $\gamma \in \Gamma_2, \gamma' \in \Gamma$.

The closure $\langle \mathcal{T} \rangle_{\mathcal{P}}$ of $\mathcal{T}$ over the composition $\circ$ and left quotient $\lceil \cdot, \cdot \rfloor^{-1}$ with respect to $\mathcal{P}$ is defined as follows:

- $\mathcal{T} \subseteq \langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$, $\emptyset \in \langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$ and $\tau_{id} \in \langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$;
- if $\tau_1, \tau_2 \in \langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$, then $\tau_1 \circ \tau_2 \in \langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$;
- if $\tau \in \langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$, then $\lceil \gamma, \gamma' \rfloor^{-1}\tau \in \langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$ for all $\gamma \in \Gamma_2, \gamma' \in \Gamma$.

Proposition 8. *The following three are equivalent:*

1. $\langle \mathcal{T} \rangle_{\mathcal{P}}$ *is finite;*
2. $\langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$ *is finite;*
3. $\overline{\langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}}$ *is finite.*

Proof. If $\langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$ is finite, we can immediately get that $\langle \mathcal{T} \rangle_{\mathcal{P}}$ and $\overline{\langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}}$ are finite. On the other hand, if $\overline{\langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}}$ is finite, we get that $\langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$ is finite. We only need to show that if $\langle \mathcal{T} \rangle_{\mathcal{P}}$ is finite, then $\langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$ is finite. To show this, we show that $\langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$ is the semigroup generated by $(\langle \mathcal{T} \rangle_{\mathcal{P}}, \cup)$.

Let $\mathcal{T}'$ be the semigroup generated by $(\langle \mathcal{T} \rangle_{\mathcal{P}}, \cup)$. We show that the following conditions hold:

- for all $\tau \in \mathcal{T}'$, we have $\lceil \gamma, \gamma' \rfloor^{-1}\tau \in \mathcal{T}'$ for all $\gamma \in \Gamma_2, \gamma' \in \Gamma$,
- for all $\tau_1, \tau_2 \in \mathcal{T}'$, we have $\tau_1 \circ \tau_2 \in \mathcal{T}'$.

Suppose $\tau = \tau_1 \cup \cdots \cup \tau_n \in \mathcal{T}'$ with $\tau_1, \cdots, \tau_n \in \langle \mathcal{T} \rangle_{\mathcal{P}}$, then $\lceil \gamma, \gamma' \rfloor^{-1}(\tau_1 \cup \cdots \cup \tau_n) = \bigcup_{i=1}^{n} \lceil \gamma, \gamma' \rfloor^{-1}\tau_i$. Since $\tau_1, \cdots, \tau_n \in \langle \mathcal{T} \rangle_{\mathcal{P}}$, we obtain that $\lceil \gamma, \gamma' \rfloor^{-1}\tau_1, \cdots, \lceil \gamma, \gamma' \rfloor^{-1}\tau_n \in \langle \mathcal{T} \rangle_{\mathcal{P}}$. Thus, we get that $\lceil \gamma, \gamma' \rfloor^{-1}(\tau_1 \cup \cdots \cup \tau_n) \in \mathcal{T}'$.

Now, we show that for all $\tau_1, \tau_2 \in \mathcal{T}'$, $\tau_1 \circ \tau_2 \in \mathcal{T}'$. Suppose $\tau_i = \tau_1^i \cup \cdots \cup \tau_{m_i}^i$ with $\tau_1^i, \cdots, \tau_{m_i}^i \in \langle \mathcal{T} \rangle_{\mathcal{P}}$ for $i \in \{1, 2\}$, then $\tau_1 \circ \tau_2 = (\tau_1^1 \cup \cdots \cup \tau_{m_1}^1) \circ (\tau_1^2 \cup \cdots \cup \tau_{m_2}^2) = \bigcup_{1 \le i \le m_1, 1 \le j \le m_2} \tau_i^1 \circ \tau_j^2$. Since for every $i : 1 \le i \le m_1$ and every $j : 1 \le j \le m_2$, $\tau_i^1, \tau_j^2 \in \langle \mathcal{T} \rangle_{\mathcal{P}}$, then $\tau_i^1 \circ \tau_j^2 \in \langle \mathcal{T} \rangle_{\mathcal{P}}$. Thus, we get that $\tau_1 \circ \tau_2 \in \mathcal{T}'$. □

A TrPDS $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$ is called *weak finite* if $\langle \mathcal{T} \rangle^{\cup}_{\mathcal{P}}$ is finite. Obviously, a finite TrPDS must be weak finite, while a weak finite TrPDS is not necessarily finite.

We observe that Algorithm 2 and Algorithm 3 can be used to compute $pre^*(C)$ and $post^*(C)$ for a rational set of configurations of weak finite TrPDSs. Indeed, the "finite" condition of TrPDSs is only used to guarantee the termination of Algorithm 2 and Algorithm 3. Algorithm 2 and Algorithm 3 also always terminate for weak finite TrPDSs based on the following facts: the set of transductions can be labeled to transition rules of TrNFA is $\langle\mathcal{T}\rangle^{\cup}_{\mathcal{P}}$ (resp. $\overline{\langle\mathcal{T}\rangle^{\cup}_{\mathcal{P}}}$) in Algorithm 2 (resp. Algorithm 3).

Theorem 8. *Let TrPDS $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$ be a weak finite and $\boldsymbol{A} = (S, \Gamma, \Lambda, \mathcal{T}, S_0, S_f)$ a TrNFA.*

- *We can compute a TrNFA $\boldsymbol{A}^{pre^*}$ in time $\mathbf{O}(|S|^2 \cdot f(|\mathcal{T}|) \cdot |\Delta| \cdot |\Gamma|)$ for some computable function f and in space $\mathbf{O}(|S| \cdot |\Delta| \cdot |\langle\mathcal{T}\rangle| \cdot |\Gamma|)$ such that $L(\boldsymbol{A}^{pre^*}) = pre^*(L(\boldsymbol{A}))$.*
- *We can compute a TrNFA $\boldsymbol{A}^{post^*}$ in $\mathbf{O}(|S| \cdot f(|\mathcal{T}|) \cdot |\Delta|^3 \cdot |\Gamma|)$ time and space for some computable function f such that $L(\boldsymbol{A}^{post^*}) = post^*(L(\boldsymbol{A}))$.*

Example 4. Let us consider the TrPDS $\mathcal{P} = (\{p\}, \{0, 1\}, \{\tau_0, \emptyset, \tau_{id}\}, \Delta\})$, where $\tau_0 = \{(0\omega_1 0\omega_2, 1\omega_1 1\omega_2) \mid \omega_1 \in \{1\}^*, \omega_2 \in \{0, 1\}^*\}$ which replaces the first two occurrences of 0 in the stack by 1, $\Delta = \{\langle p, 0\rangle \overset{\tau_0}{\hookrightarrow} \langle p, 01\rangle\}$. Let τ_1^1 denote $\lceil 0, 1\rfloor^{-1}\tau_0 = \{(\omega_1 0\omega_2, \omega_1 1\omega_2) \mid \omega_1 \in \{1\}^*, \omega_2 \in \{0, 1\}^*\}$, let $\tau_1^i = \tau_1^1 \circ \tau_1^{i-1}$ for every $i \geq 2$, and $\tau_2^i = \tau_0 \circ \tau_1^i$ for every $i \geq 1$. Then, $\langle\mathcal{T}\rangle = \{\emptyset, \tau_{id}, \tau_0, \tau_1^i, \tau_2^i \mid \forall i \geq 1\}$. Since $\tau_0 \neq \tau_1^i$ and $\tau_0 \neq \tau_2^i$ for every $i \geq 1$, $\tau_1^i \neq \tau_1^j$ and $\tau_2^i \neq \tau_2^j$ for every $i > j \geq 1$, $\tau_1^i \neq \tau_2^j$ for every $i, j \geq 1$, $\langle\mathcal{T}\rangle$ is an infinite set. Thus, the TrPDS $\mathcal{P}$ is not finite. However, $\langle\mathcal{T}\rangle_{\mathcal{P}} = \{\tau_0, \emptyset, \tau_{id}\}$. Thus, the TrPDS $\mathcal{P}$ is weak finite.

6. Pushdown systems with left contextual transductions

In Section 5, we showed that $pre^*(C)$ and $post^*(C)$ are still computable for a given rational set of configurations C of weak finite TrPDSs. However, this extension do not break the length-preserving restriction. In this section, we present automata-theoretic approaches to compute $pre^*(C)$ and $post^*(C)$ for a given rational set of configurations C of TrPDSs whose transductions are not a-prior limited length-preserving. Precisely speaking, we consider *left contextual transductions* that are in the form of $\tau = \{(\omega_1\omega_1', \omega_2\omega_2') \mid \omega_1 \in L(\mathbf{A}_1), \omega_2 \in L(\mathbf{A}_2), (\omega_1', \omega_2') \in \tau'\}$, where $\mathbf{A}_1$ and $\mathbf{A}_2$ are two finite state automata called *left input context* and *left output context* respectively, τ' is a rational length-preserving transduction. Left contextual transductions are an extension of prefix-recognizable systems with stack manipulation [29,6,30,7]. Left contextual transductions are also useful for modeling programs that rewrite the topmost subword of the stack content. For instance, to model a stack-based assembly program, the control points of the program are usually stored at the top of the TrPDS's stack, the content of program's stack is stored after the top of the TrPDS's stack. Contextual transductions allow us easily and succinctly model the statements such as *pushad* that pushes the values of several registers onto the program's stack, and *popad* that pops the values of several registers from the program's stack.

We give a reduction from the reachability problem of a TrPDS with left contextual transductions to the one of a TrPDS with rational length-preserving transductions by performing a kind of "product" of the TrPDS with finite state automata, that is all the left input/output context. Then, computing $pre^*(C)$ and $post^*(C)$ of the TrPDS with left contextual transductions is reduced to computing $pre^*(C)$ and $post^*(C)$ of another TrPDS with length-preserving transductions. $pre^*(C)$ and $post^*(C)$ of the TrPDS with left contextual transductions will be computable if the closure of the set of underlying length-preserving transductions is finite.

6.1. Left contextual TrPDSs

A transduction $\tau \subseteq \Gamma^* \times \Gamma^*$ is a *left contextual transduction* if there exists a rational length-preserving transduction τ' and left contexts $\mathbf{A}_1$ and $\mathbf{A}_2$ such that $\tau = \{(\omega_1\omega_1', \omega_2\omega_2') \mid \omega_1 \in L(\mathbf{A}_1), \omega_2 \in L(\mathbf{A}_2), (\omega_1', \omega_2') \in \tau'\}$, where $\mathbf{A}_1$ and $\mathbf{A}_2$ are called *left input context* and *left output context*, respectively. For every left contextual transduction τ, let $\tau_{\mathbf{lp}}$ denote the rational length-preserving transduction τ', $\mathbf{A}_\tau^i$ and $\mathbf{A}_\tau^o$ respectively be the left contexts $\mathbf{A}_1$ and $\mathbf{A}_2$ such that the above condition holds. Intuitively, the left input context $\mathbf{A}_\tau^i$ appends words at the left-side of input part of the rational length-preserving transduction $\tau_{\mathbf{lp}}$. Similarly, the left output context $\mathbf{A}_\tau^o$ appends words at the left-side of output part of the rational length-preserving transduction $\tau_{\mathbf{lp}}$.

Given a finite set $\mathcal{T}$ of left contextual transductions, let $\mathcal{T}_{\mathbf{lp}}$ denote the set of the rational length-preserving transductions $\{\tau_{\mathbf{lp}} \mid \tau \in \mathcal{T}\}$. A TrPDS $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$ is called *left contextual* TrPDS if $\mathcal{T}$ is a finite set of left contextual transductions.

6.2. Prefix-recognizable systems

Prefix-recognizable systems were proposed by Caucal in [29], which generalize pushdown systems. The model checking problem of LTL and μ-calculus on prefix-recognizable systems were well-studied in [6,30,7]. In this section, we show that prefix-recognizable systems can be viewed as a special case of left contextual TrPDSs.

Definition 6. [29,6] A *prefix-recognizable system* (PRS)[1] $\mathcal{P}$ is a tuple (P, Γ, Δ), where P is a finite set of control states, Γ is a finite alphabet, and Δ is a finite set of transition rules of the form (p, e_1, e_2, e_3, p') such that $p, \ p' \in P$ and $e_1, \ e_2, \ e_3$ are regular expressions over Γ.

A *configuration* of PRS $\mathcal{P}$ is a pair $\langle p, \omega \rangle$ such that $p \in P$ and $\omega \in \Gamma^*$. Given a regular expression e over Γ, let $L(e) \subseteq \Gamma^*$ denote the language of the regular expression e. If $(p, e_1, e_2, e_3, p') \in \Delta$, then for every $\omega_1, \omega_2, \omega \in \Gamma^*$ such that $\omega_1 \in L(e_1)$, $\omega \in L(e_2)$ and $\omega_2 \in L(e_3)$, the configuration $\langle p, \omega_1\omega \rangle$ is an *immediate predecessor* of the configuration $\langle p', \omega_2\omega \rangle$, denoted by $\langle p, \omega_1\omega \rangle \Longrightarrow_{\mathcal{P}} \langle p', \omega_2\omega \rangle$.

We observe that for each transition rule (p, e_1, e_2, e_3, p') of the PRS $\mathcal{P}$, e_1 can be seen as a left input context, e_2 can be represented by a transduction that checks the input word and produces the input without any change, and e_3 can be seen as a left output context. Then, PRSs can be seen as a special case of left contextual TrPDSs. Formally, given a PRS $\mathcal{P} = (P, \Gamma, \Delta)$, let $\mathcal{P}' = (P, \Gamma \cup \sharp, \mathcal{T}, \Delta')$ be a TrPDS such that $\sharp \notin \Gamma$ is a new stack symbol, $\mathcal{T} = \{\tau_t \mid t \in \Delta\}$, and

$$t = (p, e_1, e_2, e_3, p') \in \Delta \text{ iff } \langle p, \sharp \rangle \overset{\tau_t}{\hookrightarrow} \langle p', \sharp \rangle \in \Delta',$$

where $\tau_t = \{(\omega_1\omega, \omega_2\omega) \mid \omega_1 \in L(\mathbf{A}_1), \omega_2 \in L(\mathbf{A}_3), (\omega, \omega) \in \tau'_t\}$, $\mathbf{A}_1$ and $\mathbf{A}_3$ are two finite state automata such that $L(\mathbf{A}_1) = L(e_1)$ and $L(\mathbf{A}_3) = L(e_3)$, τ_2 is the transduction such that $L(\tau'_t) = \{(\omega, \omega) \in \Gamma^* \times \Gamma^* \mid \omega \in L(e_2)\}$. It is easy to verify that the closure of $\{\tau'_t \mid t \in \Delta\}$ is finite. Indeed, due to pumping lemma of finite state automata, left quotient operations on these transductions only yield a finite number of new transductions τ' satisfying that for each pair of word $(\omega_1, \omega_2) \in \tau'$, $\omega_1 = \omega_2$. On the other hand, the composition does not introduce any new transductions. Therefore, $\mathcal{P}'$ is a left contextual TrPDS.

6.3. An reduction for left contextual TrPDSs

Given a left contextual TrPDS $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$ and a rational set of configurations $C \subseteq \mathcal{C}_{\mathcal{P}}$, we construct a TrPDS $\mathcal{P}' = (P', \Gamma, \mathcal{T}_{\mathbf{lp}}, \Delta')$ in polynomial time of $\mathcal{P}$ such that $pre^*_{\mathcal{P}}(C) = pre^*_{\mathcal{P}'}(C) \cap P \times \Gamma^*$ and $post^*_{\mathcal{P}}(C) = post^*_{\mathcal{P}'}(C) \cap P \times \Gamma^*$.

W.l.o.g., for technical reasons, we assume that for every left contextual transduction $\tau \in \mathcal{T}$, the sets of states of the left input context $\mathbf{A}^i_\tau$ and the left output context $\mathbf{A}^o_\tau$ are disjointed. Δ' and P' are computed as follows: for every transition rule $t = \langle p, \gamma \rangle \overset{\tau}{\hookrightarrow} \langle p', \omega \rangle \in \Delta$, let $\mathbf{A}^i_\tau = (S^i, \Gamma, \Lambda^i, \{\tau_{id}\}, S^i_0, S^i_f)$ and $\mathbf{A}^o_\tau = (S^o, \Gamma, \Lambda^o, \{\tau_{id}\}, S^o_0, S^o_f)$,

1. for every $s \in S^i_0$, $\langle p, \gamma \rangle \overset{\tau_{id}}{\hookrightarrow} \langle p^t_s, \epsilon \rangle \in \Delta'$;
2. for every $s \overset{\gamma'|\tau_{id}}{\longmapsto} s' \in \Lambda^i$,
   (a) $\langle p^t_s, \gamma' \rangle \overset{\tau_{id}}{\hookrightarrow} \langle p^t_{s'}, \epsilon \rangle \in \Delta'$,
   (b) if $s' \in S^i_f$, then $\langle p^t_s, \gamma' \rangle \overset{\tau_{\mathbf{lp}}}{\hookrightarrow} \langle p^t_{s''}, \epsilon \rangle \in \Delta'$ for every $s'' \in S^o_f$;
3. for every $s \overset{\gamma''|\tau_{id}}{\longmapsto} s' \in \Lambda^o$, $\gamma' \in \Gamma$, $\langle p^t_{s'}, \gamma' \rangle \overset{\tau_{id}}{\hookrightarrow} \langle p^t_s, \gamma''\gamma' \rangle \in \Delta'$;
4. for every $s \in S^o_0$, $\gamma' \in \Gamma$, $\langle p^t_s, \gamma' \rangle \overset{\tau_{id}}{\hookrightarrow} \langle p', \omega\gamma' \rangle \in \Delta'$.

Intuitively, suppose $\mathcal{P}$ is at the configuration $\langle p, \gamma\omega' \rangle$ with $\gamma \in \Gamma, \omega' \in \Gamma^*$ and there is a transition rule $t = \langle p, \gamma \rangle \overset{\tau}{\hookrightarrow} \langle p', \omega \rangle \in \Delta$, then $\langle p, \gamma\omega' \rangle \Longrightarrow_{\mathcal{P}} \langle p', \omega\omega'' \rangle$ for every $\omega'' \in \tau(\omega')$. There necessarily exists $\omega_1, \omega'_1, \omega_2, \omega'_2 \in \Gamma^*$ such that $\omega' = \omega_1\omega'_1, \omega'' = \omega_2\omega'_2$, $\omega_1 \in L(\mathbf{A}^i_\tau)$, $\omega_2 \in L(\mathbf{A}^o_\tau)$ and $(\omega'_1, \omega'_2) \in \tau_{\mathbf{lp}}$. To have $\langle p, \gamma\omega_1\omega'_1 \rangle \Longrightarrow^*_{\mathcal{P}'} \langle p', \omega\omega_2\omega'_2 \rangle$, we let the run of $\mathcal{P}'$ moves from $\langle p, \gamma\omega_1\omega'_1 \rangle$ to $\langle p^t_s, \omega_1\omega'_1 \rangle$, for every initial state $s \in S^i_0$ (Item 1). After that, the run of $\mathcal{P}'$ mimics the run of $\mathbf{A}^i_\tau$ on the word ω_1 and pops ω_1 from the stack until $\mathcal{P}'$ reaches the configuration $\langle p^t_{s'}, \gamma'\omega'_1 \rangle$ (Item 2(a)), when γ' is the bottom of ω_1. Since $\mathbf{A}^i_\tau$ accepts the word ω_1 iff the run of $\mathcal{P}'$ can reach the configuration $\langle p^t_{s'}, \gamma'\omega'_1 \rangle$ from $\langle p, \gamma\omega_1\omega'_1 \rangle$ such that there is a transition $s' \overset{\gamma'}{\longmapsto} s_1 \in \Lambda^i$ with $s_1 \in S^i_f$. The later happens iff the run of $\mathcal{P}'$ reaches a configuration $\langle p^t_{s_2}, \omega'_2 \rangle$ from $\langle p, \gamma\omega_1\omega'_1 \rangle$ for some $s_2 \in S^o_f$ ensured by adding the transition $\langle p^t_{s'}, \gamma' \rangle \overset{\tau_{\mathbf{lp}}}{\hookrightarrow} \langle p^t_{s_2}, \epsilon \rangle$ (Item 2(b)). Moving to the configuration $\langle p^t_{s_2}, \omega'_2 \rangle$ for $s_2 \in S^o_f$ allows $\mathcal{P}'$ to mimic the run of $\mathbf{A}^o_\tau$ in a *reverse* way over the word ω_2 and pushes ω_2 onto the stack until reaching an initial state s_3 of $\mathbf{A}^o_\tau$, i.e., the run of $\mathcal{P}'$ reaching the configuration $\langle p^t_{s_3}, \omega_2\omega'_2 \rangle$. Item 3 guarantees that $\mathbf{A}^o_\tau$ accepts the word ω_2 iff the run of $\mathcal{P}'$ moves from $\langle p^t_{s_2}, \omega'_2 \rangle$ to the configuration $\langle p^t_{s_3}, \omega_2\omega'_2 \rangle$ such that $s_3 \in S^o_0$. Finally, the run of $\mathcal{P}'$ moves from $\langle p^t_{s_3}, \omega_2\omega'_2 \rangle$ to $\langle p, \omega\omega_2\omega'_2 \rangle$ by the transitions added Item 4. Therefore, $\langle p, \gamma\omega_1\omega'_1 \rangle \Longrightarrow_{\mathcal{P}} \langle p', \omega\omega_2\omega'_2 \rangle$ iff $\langle p, \gamma\omega_1\omega'_1 \rangle \Longrightarrow^*_{\mathcal{P}'} \langle p', \omega\omega_2\omega'_2 \rangle$.

[1] Note that the standard definition of prefix-recognizable systems does not include control states [29]. Indeed, a prefix-recognizable system without states can simulate a prefix-recognizable system with states by having the state as the first letter of the unbounded store. We use the definition of prefix-recognizable systems with control states for the sake of uniform notation, similar to [6,30].

Theorem 9. *The reachability problem of left contextual TrPDSs $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$ is decidable if $\langle \mathcal{T}_{\mathbf{lp}} \rangle$ is finite. Indeed, one can translate the reachability problem of left contextual TrPDSs into the problem of finite TrPDSs in polynomial time.*

Proof. We show that for every $\langle p, \omega\rangle, \langle p', \omega'\rangle \in \mathcal{C}_{\mathcal{P}}$, $\langle p, \omega\rangle \Longrightarrow_{\mathcal{P}} \langle p', \omega'\rangle$ iff $\langle p, \omega\rangle \Longrightarrow^*_{\mathcal{P}'} \langle p', \omega'\rangle$ and the derivation of $\langle p, \omega\rangle \Longrightarrow^*_{\mathcal{P}'} \langle p', \omega'\rangle$ only uses configurations in $(P' \setminus P) \times \Gamma^*$.

($\Longrightarrow$) Suppose $\langle p, \omega\rangle \Longrightarrow_{\mathcal{P}} \langle p', \omega'\rangle$, then there exists a transition rule $t = \langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle p', \omega''\rangle \in \Delta$ such that $\omega = \gamma u$, $\omega' = \omega'' u'$ for some $u' \in \tau(u)$. There necessarily exist $\omega_1, \omega_1', \omega_2, \omega_2' \in \Gamma^*$ such that $u = \omega_1\omega_1'$, $u' = \omega_2\omega_2'$, $\omega_1 \in L(\mathbf{A}^i_\tau)$, $\omega_2 \in L(\mathbf{A}^o_\tau)$ and $(\omega_1', \omega_2') \in \tau_{\mathbf{lp}}$.

Let $\omega_1 = \gamma_1 ... \gamma_n$, $\omega_2 = \gamma_1' ... \gamma_{n'}'$ such that $\gamma_1, ..., \gamma_n, \gamma_1', ..., \gamma_{n'}' \in \Gamma$. Then, the following two hold:

- for every $k : 1 \le k \le n$, there exists $s_k \xmapsto{\gamma_k} s_{k+1} \in \Lambda^i$ such that $s_1 \in S^i_0$ and $s_{n+1} \in S^i_f$;
- for every $k : 1 \le k \le n'$, there exists $s_k' \xmapsto{\gamma_k'} s_{k+1}' \in \Lambda^o$ such that $s_1' \in S^o_0$ and $s_{n'+1}' \in S^o_f$.

By Item 1, we get that $\langle p, \gamma\omega_1\omega_1'\rangle \Longrightarrow_{\mathcal{P}'} \langle p^t_{s_1}, \omega_1\omega_1'\rangle$. By Item 2(a), we get that $\langle p^t_{s_1}, \omega_1\omega_1'\rangle = \langle p^t_{s_1}, \gamma_1 ... \gamma_n\omega_1'\rangle \Longrightarrow_{\mathcal{P}'} \langle p^t_{s_2}, \gamma_2 ... \gamma_n\omega_1'\rangle \Longrightarrow_{\mathcal{P}'} ... \Longrightarrow_{\mathcal{P}'} \langle p^t_{s_n}, \gamma_n\omega_1'\rangle$. By Item 2(b), $\langle p^t_{s_n}, \gamma_n\omega_1'\rangle \Longrightarrow_{\mathcal{P}'} \langle p^t_{s'_{n'+1}}, \omega_2'\rangle$. By Item 3, we get that $\langle p^t_{s'_{n'+1}}, \omega_2'\rangle \Longrightarrow_{\mathcal{P}'} \langle p^t_{s'_{n'}}, \gamma_{n'}'\omega_2'\rangle \Longrightarrow_{\mathcal{P}'} ... \Longrightarrow_{\mathcal{P}'} \langle p^t_{s_1'}, \gamma_1' ... \gamma_{n'}'\omega_2'\rangle = \langle p^t_{s_1'}, \omega_2\omega_2'\rangle$. By Item 4, $\langle p^t_{s_1'}, \omega_2\omega_2'\rangle \Longrightarrow_{\mathcal{P}'} \langle p', \omega''\omega_2\omega_2'\rangle$. Therefore, the result immediately follows.

($\Longleftarrow$) Suppose $\langle p, \omega\rangle \Longrightarrow^*_{\mathcal{P}'} \langle p', \omega'\rangle$ and the derivation of $\langle p, \omega\rangle \Longrightarrow^*_{\mathcal{P}'} \langle p', \omega'\rangle$ only uses configurations in $(P' \setminus P) \times \Gamma^*$, then there exist $\gamma, \gamma_1, ..., \gamma_n, \gamma_1', ..., \gamma_{n'}' \in \Gamma$, $\omega_1, \omega_1', \omega_2 \in \Gamma^*$ such that $\omega = \gamma\gamma_1 ... \gamma_n\omega_1$, $\omega' = \omega_2\gamma_1' ... \gamma_{n'}'\omega_1'$ and the following conditions hold:

- $\langle p, \gamma\gamma_1 ... \gamma_n\omega_1\rangle \Longrightarrow_{\mathcal{P}'} \langle p^t_{s_1}, \gamma_1 ... \gamma_n\omega_1\rangle$, i.e., $\langle p, \gamma\rangle \overset{\tau_{id}}{\hookrightarrow} \langle p^t_{s_1}, \epsilon\rangle \in \Delta'$;
- for every $k : 1 \le k \le n-1$, $\langle p^t_{s_k}, \gamma_k ... \gamma_n\omega_1\rangle \Longrightarrow_{\mathcal{P}'} \langle p^t_{s_{k+1}}, \gamma_{k+1} ... \gamma_n\omega_1\rangle$, i.e., $\langle p^t_{s_k}, \gamma_k\rangle \overset{\tau_{id}}{\hookrightarrow} \langle p^t_{s_{k+1}}, \epsilon\rangle \in \Delta'$;
- $\langle p^t_{s_n}, \gamma_n\omega_1\rangle \Longrightarrow_{\mathcal{P}'} \langle p^t_{s'_{n'+1}}, \omega_1'\rangle$, i.e. $\langle p^t_{s_n}, \gamma_k\rangle \overset{\tau}{\hookrightarrow} \langle p^t_{s'_{n'+1}}, \epsilon\rangle \in \Delta'$ with $(\omega_1, \omega_1') \in \tau$;
- $\langle p^t_{s'_{n'+1}}, \omega_1'\rangle \Longrightarrow_{\mathcal{P}'} \langle p^t_{s'_{n'}}, \gamma_{n'}'\omega_1'\rangle$, i.e., $\langle p^t_{s'_{n'+1}}, \gamma''\rangle \overset{\tau_{id}}{\hookrightarrow} \langle p^t_{s'_{n'}}, \gamma_{n'}'\gamma''\rangle \in \Delta'$ for every $\gamma'' \in \Gamma$;
- for every $k : 1 < k \le n'$, $\langle p^t_{s_k'}, \gamma_k' ... \gamma_{n'}'\omega_1'\rangle \Longrightarrow_{\mathcal{P}'} \langle p^t_{s'_{k-1}}, \gamma'_{k-1}\gamma_k' ... \gamma_{n'}'\omega_1'\rangle$, i.e., $\langle p^t_{s'_{k'}}, \gamma''\rangle \overset{\tau_{id}}{\hookrightarrow} \langle p^t_{s'_{k'-1}}, \gamma'_{k-1}\gamma''\rangle \in \Delta'$ for every $\gamma'' \in \Gamma$;
- $\langle p^t_{s_1'}, \gamma_1' ... \gamma_{n'}'\omega_1'\rangle \Longrightarrow_{\mathcal{P}'} \langle p', \omega_2\gamma_{1'}' ... \gamma_{n'}'\omega_1'\rangle$, i.e., $\langle p^t_{s_1'}, \gamma''\rangle \overset{\tau_{id}}{\hookrightarrow} \langle p', \omega_2\gamma''\rangle \in \Delta'$ for every $\gamma'' \in \Gamma$.

Then, there exists a transition such that the transduction of the transition has those automata $\mathbf{A}^i = (S^i, \Gamma, \Lambda^i, \{\tau_{id}\}, S^i_0, S^i_f)$ and $\mathbf{A}^o = (S^o, \Gamma, \Lambda^o, \{\tau_{id}\}, S^o_0, S^o_f)$, and the following conditions hold:

- for every $k : 1 \le k \le n$, there exists $s_k \xmapsto{\gamma_k} s_{k+1} \in \Lambda^i$ such that $s_1 \in S^i_0$ and $s_{n+1} \in S^i_f$;
- for every $k : 1 \le k \le n'$, there exists $s_k' \xmapsto{\gamma_k'} s_{k+1}' \in \Lambda^o$ such that $s_1' \in S^o_0$ and $s_{n'+1}' \in S^o_f$;
- $(\omega_1, \omega_1') \in \tau$;
- $t = \langle p, \gamma\rangle \overset{\tau'}{\hookrightarrow} \langle p', \omega_2\rangle \in \Delta$, where $\tau' = \{(\omega_3\omega_3', \omega_4\omega_4') \mid \omega_3 \in L(\mathbf{A}^i), \omega_4 \in L(\mathbf{A}^o), (\omega_3', \omega_4') \in \tau\}$.

Therefore, we get that $\langle p, \omega\rangle \Longrightarrow_{\mathcal{P}} \langle p', \omega'\rangle$. $\square$

Remark 5. Note that left context excludes the empty word ϵ, that is $S_f \cap S_0 = \emptyset$. Suppose a transition rule $t = \langle p, \gamma\rangle \overset{\tau}{\hookrightarrow} \langle p', \omega\rangle \in \Delta$ needs to have $\epsilon \in L(\mathbf{A}^0_\tau)$. We can replace t by two transition rules $t_\epsilon = \langle p, \gamma\rangle \overset{\tau_\epsilon}{\hookrightarrow} \langle p', \omega\rangle \in \Delta$ and $t_{\overline{\epsilon}} = \langle p, \gamma\rangle \overset{\tau_{\overline{\epsilon}}}{\hookrightarrow} \langle p', \omega\rangle \in \Delta$, where $\tau_\epsilon = \{(\omega_1\omega_1', \omega_2') \mid \omega_1 \in L(\mathbf{A}^i_\tau), (\omega_1', \omega_2') \in \tau_{\mathbf{lp}}\}$ and $\tau_{\overline{\epsilon}} = \{(\omega_1\omega_1', \omega_2\omega_2') \mid \omega_1 \in L(\mathbf{A}^i_\tau), \omega_2 \in L(\mathbf{A}^0_\tau), \omega_2 \neq \epsilon, (\omega_1', \omega_2') \in \tau_{\mathbf{lp}}\}$. While transition rules of the form τ_ϵ can be handled by readapting the reduction of left contextual TrPDSs to finite TrPDSs. The case when $\epsilon \in L(\mathbf{A}^i_\tau)$ can be processed in similar way.

However, in general, the reachability problem of left contextual TrPDSs is undecidable if $\langle \mathcal{T}_{\mathbf{lp}} \rangle$ is infinite. This is an immediate consequence of the undecidability result of TrPDSs $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$ with infinite set $\langle \mathcal{T} \rangle$ [27].

Theorem 10. *The reachability problem of left contextual TrPDSs $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$ is undecidable if $\langle \mathcal{T}_{\mathbf{lp}} \rangle$ is infinite.*

7. Applications

In [27], Uezato and Minamide presented two potential applications of TrPDSs: checking reachability of conditional PDSs [35,25] and discrete-timed PDSs [17] via pre^* or $post^*$ computing of TrPDSs. In this section, we will present another two potential applications of TrPDSs on software verification. We show how the presence of transductions enables the modeling of Boolean programs with call-by-reference parameter passing and low-level assembly programs that manipulate program stack content via a stack pointer.

7.1. Application to Boolean programs with call-by-reference parameter passing

Boolean programs in which all variables and parameters (call-by-value) have Boolean type are thought of as an abstract representation of C/C++ programs with recursion [31]. In their definition, Boolean programs contain procedures with call-by-value parameter passing rather than call-by-reference parameter passing. While call-by-reference parameter passing is a widely used programming paradigm in C/C++, Java, etc. Using TrPDSs, we can verify safety properties of Boolean programs with call-by-reference parameter passing.

7.1.1. Boolean programs with call-by-reference parameter passing

A *Boolean program* BP is a tuple $(Proc, main, G)$, where $Proc$ is a finite set of procedures, $main \in Proc$ is the *initial* procedure, G is a finite set of global Boolean variables. Every procedure $r \in Proc$ is a tuple (N_r, E_r, L_r), where N_r is a finite set of control points with r_{entry} as the unique entry node, E_r is a finite set of edges, L_r is the finite set of local Boolean variables in r. W.l.o.g., we assume that $L_r \cap G = \emptyset$ for all $r \in Proc$. $L'_r = L_r \cup G$ is a set of *visible variables* in r. Let L_r^{ref} be the set of all the call-by-reference formal parameters of the procedure r, L_n^{ref} (resp. L_n) be the set L_r^{ref} (resp. L_r) such that $n \in N_r$. Given a procedure call $stmt = r(v_1, ..., v_m)$ at the control point n whose return address is n', for every formal parameter v' of r, let $\mathsf{fRef}(n')(v') \in \{v_1, ..., v_m\}$ be the actual parameter of v' at the caller site n'.

A *valuation* $\xi \subseteq L'_r$ is a subset of L'_r meaning that the Boolean value of $x \in L'_r$ is 1 if $x \in \xi$, otherwise 0. Let $\xi(x) = 1$ if $x \in \xi$, otherwise 0. Let $\xi[d/x]$ be the valuation such that $\xi[d/x](y) = d$ if $x = y$, otherwise $\xi(y)$. The edges in E_r are of the form $(n, \xi, stmt, n')$ meaning that n' is the next control point of n when the valuation at n is ξ, where $stmt$ is the statement at n. Let $[\![stmt]\!]_\xi$ be the valuation after executing the statement $stmt$ that is neither a procedure call nor return. The details of the execution model and semantics of other statements refer to [31].

7.1.2. Modeling approach

W.l.o.g., we assume that call-by-reference actual parameters are local variables. Indeed, global variables are always visible for all procedures and do not need to be passed by parameters. Different from call-by-value parameter passing which keeps its own copy at callee site, a parameter passed by call-by-reference will not keep its own copy at callee site. Precisely speaking, the values of call-by-reference actual parameters at a caller procedure are always same as the values of the corresponding formal parameters at the callee procedure. We will use transductions to encode the changing of call-by-reference formal parameters, as transductions in TrPDSs allow us to manipulate the stack content rather than the top of stack in PDSs.

The construction of TrPDSs from Boolean programs BP with call-by-reference parameter passing follows the standard modeling approach of PDSs from Boolean programs [1] except the assignments to reference variables should also effect on the value of the corresponding actual parameters at the corresponding caller site. The valuations of global variables G are put in the control locations of the TrPDSs, the pairs of the valuations of local variables and control points (i.e., nodes) of the program are stored in the stack of the TrPDSs. The TrPDS model will be $\mathcal{P} = (2^G, \bigcup_{r \in Proc}(N_r \times 2^L_r), \mathcal{T}, \Delta)$. A configuration of the TrPDS model is in the form of $c = \langle \xi, (n_0, \xi_0) \cdots (n_k, \xi_k) \rangle$ meaning that the execution of BP is at the control point n_0 with ξ as the valuation of global variables, ξ_0 as the valuation of local variables of the procedure containing n_0. Moreover, $(n_1, \xi_1) \cdots (n_k, \xi_k)$ is the calling history of the execution such that for every $i : 1 \leq i \leq k$, n_i is the return address of the procedure call that jumped into the procedure containing n_{i-1} and ξ_i is the stored valuation of local variables when the procedure call is made. Different from Boolean programs only with call-by-value parameter passing, a local variable of a procedure may be a call-by-reference parameter of the procedure. In this case, the value of a local variable and its referenced variable are identical. Therefore, the potential possible configurations of the TrPDS model should only have admitted valuations with respect to the local variables and their referenced variables. Formally, a word $(n_0, \xi_0) \cdots (n_k, \xi_k)$ or a configuration $\langle \xi, (n_0, \xi_0) \cdots (n_k, \xi_k) \rangle$ is *admissible* if for every $i : 0 \leq i \leq k-1$ and every $v \in L_{n_i}^{ref}$, $v \in \xi_i$ iff $\mathsf{fRef}(n_{i+1})(v) \in \xi_{i+1}$.

Given two sets of variables $\xi'_0, \xi_0 \subseteq \bigcup_{r \in Proc} L_r^{ref}$, let $\tau_{(\xi'_0,\ \xi_0)} \subseteq \Gamma^* \times \Gamma^*$ be the transduction such that for every $((n_1, \xi_1)...(n_k, \xi_k), (n_1, \xi'_1)...(n_k, \xi'_k)) \in \Gamma^* \times \Gamma^*$, the following two conditions hold:

- $((n_1, \xi_1)...(n_k, \xi_k), (n_1, \xi'_1)...(n_k, \xi'_k)) \in \tau_{(\xi'_0,\ \xi_0)}$ iff $(n_1, \xi_1)...(n_k, \xi_k)$ is admissible,
- and $\forall i : 1 \leq i \leq k$, $\xi'_i = \xi_i \cup \{\mathsf{fRef}(n_i)(v) \mid v \in \xi'_{i-1} \setminus \xi_{i-1}\} \setminus \{\mathsf{fRef}(n_i)(v) \mid v \in \xi_{i-1} \setminus \xi'_{i-1}\}$.

Intuitively, given an admissible word $(n_1, \xi_1)...(n_k, \xi_k)$ and two sets ξ_0 and ξ_0' denoting respectively the valuations of local variables before and after an assignment, $(n_1, \xi_1')...(n_k, \xi_k')$ is the admissible word obtained from $(n_1, \xi_1)...(n_k, \xi_k)$ with the updating of all the actual parameters of the corresponding formal call-by-reference parameters with respect to ξ_0' and ξ_0.

The set Δ of transition rules that mimic the control flow of BP is defined as follows: for every edge $e = (n, \xi, stmt, n')$ in a procedure r,

- $\langle \xi \cap G, (n, \xi \cap L_r)\rangle \overset{\tau_{id}}{\hookrightarrow} \langle \xi \cap G, (r'_{entry}, \xi')(n', \xi \cap L_r)\rangle \in \Delta$ if $stmt$ is a procedure call $r'(v_1, ..., v_m)$, where $\xi' = \{v \in L_{r'} \mid \mathsf{fRef}(n')(v) \in \xi\}$;
- $\langle \xi \cap G, (n, \xi \cap L_r)\rangle \overset{\tau_{id}}{\hookrightarrow} \langle \xi \cap G, \epsilon\rangle \in \Delta$ if $stmt$ is a return;
- $\langle \xi \cap G, (n, \xi \cap L_r)\rangle \overset{\tau_e}{\hookrightarrow} \langle [\![stmt]\!]_\xi \cap G, (n', [\![stmt]\!]_\xi \cap L_r)\rangle \in \Delta$ otherwise, where $\tau_e = \tau_{([\![stmt]\!]_\xi,\ \xi)}$.

The set $\mathcal{T}$ of transductions is $\{\tau_{id}, \tau_e \mid e = (n, \xi, stmt, n') \in E_r, r \in Proc, stmt$ is neither a procedure call nor return$\}$. Intuitively, the TrPDS model mimics the execution of BP. The intuition behind function calls and returns is similar to translating from Boolean programs into PDSs. Suppose the execution of BP is at the control point n with the valuation ξ, and the calling history is $(n_1, \xi_1)...(n_k, \xi_k)$.

If the edge $e = (n, \xi, r'(v_1, ..., v_m), n')$ is in a procedure r, then, the execution of BP will move from n to the entry point r'_{entry} of the procedure r', the valuation $\xi \cap G$ of the global variables stays the same, the valuation of the local variables of r' at r'_{entry} takes the value of the actual parameters at n, the return address n' and the valuation $\xi \cap L_r$ of local variables of r are appended to the calling history $(n_1, \xi_1)...(n_k, \xi_k)$. Therefore, we add the transition rule $\langle \xi \cap G, (n, \xi \cap L_r)\rangle \overset{\tau_{id}}{\hookrightarrow} \langle \xi \cap G, (r_{entry}, \xi')(n', \xi \cap L_r)\rangle$ into Δ which allows the TrPDS model to move from the configuration $\langle \xi \cap G, (n, \xi \cap L_r)(n_1, \xi_1)...(n_k, \xi_k)\rangle$ to $\langle \xi \cap G, (r_{entry}, \xi')(n', \xi \cap L_r)(n_1, \xi_1)...(n_k, \xi_k)\rangle$.

If the edge $e = (n, \xi, return, n')$ is in a procedure r, then, the execution of BP will move from n to the return address n_1, the valuation $\xi \cap G$ of the global variables stays the same, the valuation of the local variables at n_1 recovers the valuation ξ_1 of local variables when the procedure call is made at the previous node of n_1 where the local variables passed by call-by-references parameters passing take the latest values at the end of the procedure r. So, we add the transition rule $\langle \xi \cap G, (n, \xi \cap L_r)\rangle \overset{\tau_{id}}{\hookrightarrow} \langle \xi \cap G, \epsilon\rangle$ into Δ which allows the TrPDS model to move from the configuration $\langle \xi \cap G, (n, \xi \cap L_r)(n_1, \xi_1)...(n_k, \xi_k)\rangle$ to $\langle \xi \cap G, (n_1, \xi_1)...(n_k, \xi_k)\rangle$.

If the edge $e = (n, \xi, stmt, n')$ is in a procedure r such that $stmt$ is neither a procedure call nor return, then, the execution of BP will move from n to the next point n' with the valuation $[\![stmt]\!]_\xi$. Moreover, the local variables at n_1 that corresponds to the formal call-by-reference parameters in r should take the values of the call-by-reference parameters at n'. Similarly, for every $i : 2 \le i \le k$, the local variables at n_i that corresponds to the formal call-by-reference parameters in the procedure containing n_{i-1} should take the values of the call-by-reference parameters at n_i. Therefore, we add the transition rule $\langle \xi \cap G, (n, \xi \cap L_r)\rangle \overset{\tau_e}{\hookrightarrow} \langle [\![stmt]\!]_\xi \cap G, (n', [\![stmt]\!]_\xi \cap L_r)\rangle$ into Δ which allows the TrPDS model to move from the configuration $\langle \xi \cap G, (n, \xi \cap L_r)(n_1, \xi_1)...(n_k, \xi_k)\rangle$ to $\langle [\![stmt]\!]_\xi \cap G, (n', [\![stmt]\!]_\xi \cap L_r)(n_1, \xi_1')...(n_k, \xi_k')\rangle$ for every $((n_1, \xi_1)...(n_k, \xi_k), (n_1, \xi_1')...(n_k, \xi_k')) \in \tau_e$. The transduction $\tau_e = \tau_{([\![stmt]\!]_\xi,\ \xi)}$ correctly specifies the updating of all the actual parameters of the corresponding call-by-reference parameters in the calling history.

From the definitions of transductions $\mathcal{T}$ and admissible, we can see that all the reachable configurations in the TrPDS model from an admissible configuration are also admissible.

Given two transductions $\tau_1 = \tau_{(\xi_1',\ \xi_1)}$, $\tau_2 = \tau_{(\xi_2',\ \xi_2)} \in \mathcal{T}$, then

$$\tau_1 \circ \tau_2 = \begin{cases} \emptyset, & \text{if } \xi_1' \neq \xi_2; \\ \tau_{(\xi_2',\ \xi_1)}, & \text{otherwise.} \end{cases}$$

Given two symbols $(n_1, \xi_1), (n_1', \xi_1') \in \Gamma^*$ and a transduction $\tau = \tau_{(\xi',\ \xi)} \in \mathcal{T}$, we have:

$$\lceil (n_1, \xi_1), (n_1', \xi_1') \rfloor^{-1} \tau = \begin{cases} \tau_{(\xi_1',\ \xi_1)}, & \text{if } n_1 = n_1', \text{ and} \\ & \xi_1' = \left\{ \begin{array}{c} (\xi_1 \cup \{\mathsf{fRef}(n_1)(v) \mid v \in \xi' \setminus \xi\}) \setminus \\ \{\mathsf{fRef}(n_1)(v) \mid v \in \xi \setminus \xi'\} \end{array} \right\}; \\ \emptyset, & \text{otherwise.} \end{cases}$$

Then, for each transduction $\tau \in \langle \mathcal{T} \rangle$, τ must be in the form of $\tau_{(\xi',\ \xi)}$ for some $r \in Proc$ and $\xi, \xi' \subseteq L_r$. Therefore, we can get that $\langle \mathcal{T} \rangle \subseteq \{\tau_{(\xi',\ \xi)} \mid \exists r \in Proc : \xi, \xi' \subseteq L_r\}$ which is finite.

Theorem 11. *The Boolean program BP can reach a control point n of the procedure r with the valuation ξ and the calling history ω from a control point n' of r' with the valuation ξ' and the calling history ω' iff $\langle \xi' \cap G, (n', \xi' \cap L_{r'})\omega'\rangle \Longrightarrow^* \langle \xi \cap G, (n, \xi \cap L_r)\omega\rangle$.*

Using Theorem 11, we can verify safety properties of Boolean programs with mixed call-by-reference and call-by-value parameter passing via solving the reachability problem of TrPDSs. The efficiency heavily relies upon the number of transductions from the modeling of the Boolean program and the saturation. From the modeling approach, the number of control

```
l1: push 1
l2: push 2
l3: call f
fr: ...
    ...
f0: ...
```

(a)

```
l'1: push 0
l'2: push 0
l'3: mov [esp+8] 1
l'4: mov [esp+4] 2
l'5: call f
fr: ...
    ...
f0: ...
```

(b)

$\forall \gamma \in \{1, 2, f_r\}$

$\langle l_1, \gamma\rangle \hookrightarrow \langle l_2, 1\gamma\rangle$
$\langle l_2, \gamma\rangle \hookrightarrow \langle l_3, 2\gamma\rangle$
$\langle l_3, \gamma\rangle \hookrightarrow \langle f_0, f_r\gamma\rangle$

(c)

$\forall \gamma \in \{0, f_r\}$

$\langle l'_1, \gamma\rangle \hookrightarrow \langle l'_2, 0\gamma\rangle$
$\langle l'_2, \gamma\rangle \hookrightarrow \langle l'_3, 0\gamma\rangle$
$\langle l'_3, \gamma\rangle \hookrightarrow \langle l'_4, \gamma\rangle$
$\langle l'_4, \gamma\rangle \hookrightarrow \langle l'_5, \gamma\rangle$
$\langle l'_5, \gamma\rangle \hookrightarrow \langle f_0, f_r\gamma\rangle$

(d)

Fig. 4. (a) Assembly program; (b) Obfuscated assembly program, where f_0 denotes the entry point of the function f, f_r is the return point of the function call to f. (c) and (d) are the PDS models of (a) and (b), respectively.

states, the size of the stack alphabet and transition rules are exponential in the size of variables, as we have to consider all the possible valuations of variables. During the saturation procedure, new transductions may be created via left quotient and composition operators. In the worst case, $\langle \mathcal{T}\rangle = \{\tau_{(\xi',\ \xi)} \mid \exists r \in Proc : \xi, \xi' \subseteq L_r\}$. Therefore, the number of created transductions by the saturation procedure is exponential in the number of variables.

Remark 6. One may argue that Boolean program with mixed call-by-reference and call-by-value parameter passing can be translated into a Boolean program with only call-by-value parameter passing by using global variables which can be verified by existing techniques such as [31]. However, this will lead to larger state space and may degrade performance.

7.2. Application to low-level programs

Off-the-shelf programs and malwares are available in binary code rather than source code. In order to static analyze binary code, binary code is commonly disassembled into an assembly-like low-level program either in an intermediate-representation language or assembly language. Next, the low-level program is translated into a program model for static analysis. In the literature, finite state machines (e.g., [36,37]) and pushdown systems (e.g., [32,33,3]) were proposed to model low-level programs. However, programs in stack-based assembly languages such as X86 assembly language or Java bytecode heavily manipulate local variables stored in stack frame which cannot be modeled precisely using finite state machines or pushdown systems. In this section, we show that TrPDSs are a natural model of programs in stack-based assembly languages. Therefore, reachability analysis algorithms of TrPDSs can be used to verify these low-level programs. We illustrate our idea using a low-level program in Intel X86 32-bit assembly language shown in Fig. 4.

Fig. 4(a) shows an assembly program which calls function f with parameters 2 and 1 (i.e., $f(2,1)$), as parameters are passed by pushing onto the stack, later the function f uses these parameters via accessing the stack. Fig. 4(b) shows an equivalent assembly program of Fig. 4(a) by obfuscating the parameters, in which two 0 are pushed onto the stack and later modified via the stack pointer *esp*. Suppose we are interested to check the property "Whether an assembly program calls the function f with parameters 2 and 1", denoted by $EF\ f(2,1)$.

In [32,33,3], the control points of programs are modeled as control states of PDSs, the stack of programs is modeled as the stack of PDSs. Statements are modeled as PDS transition rules. For instance, the transition rules in Fig. 4(c) models the program in Fig. 4(a). Then $EF\ f(2,1)$ can be correctly verified by reachability checking of the program model shown in Fig. 4(c). However, as stated in [3], this modeling approach cannot precisely model assembly programs that change the stack content by manipulating the stack pointer, i.e., the data in the stack cannot be changed via direct memory access.

Consider the program in Fig. 4(b), the transition rules are given in Fig. 4(d) using this model approach. It is easy to see that this program model does not satisfy $EF\ f(2,1)$. This limitation can be avoided by using TrPDSs, i.e., TrPDSs can model assembly programs that change the stack content by manipulating the stack pointer, as the manipulation of the stack via the stack pointer can be modeled as transductions. The TrPDS model of the program in Fig. 4(b) is defined as $\mathcal{P} = (P, \Gamma, \mathcal{T}, \Delta)$, where $P = \{l'_1, ..., l'_5, f_0, f_r\}$, where f_0 denotes the entry point of the function f and f_r is the return address of the function call, $\Gamma = \{0, 1, 2, f_r\}$, $\mathcal{T} = \{\tau_{id}, \tau_1\}$ with $\tau_1 = \{(\gamma\omega, 1\omega) \mid \gamma \in \Gamma, \omega \in \Gamma^*\}$, Δ is given as follows:

$$\Delta = \left\{ \begin{array}{ll|l} \langle l'_1, \gamma\rangle \overset{\tau_{id}}{\hookrightarrow} \langle l'_2, 0\gamma\rangle, & \langle l'_2, \gamma\rangle \overset{\tau_{id}}{\hookrightarrow} \langle l'_3, 0\gamma\rangle, & \\ \langle l'_3, \gamma\rangle \overset{\tau_1}{\hookrightarrow} \langle l'_4, \gamma\rangle, & \langle l'_4, \gamma\rangle \overset{\tau_{id}}{\hookrightarrow} \langle l'_5, 2\rangle, & \gamma \in \Gamma \\ \langle l'_5, \gamma\rangle \overset{\tau_{id}}{\hookrightarrow} \langle f_0, f_r\gamma\rangle & & \end{array} \right\}.$$

Since $\langle \mathcal{T}\rangle$ is a finite set, $EF\ f(2,1)$ can be verified via reachability checking of the TrPDS model $\mathcal{P}$.

Remark 7. The TrPDS model of this program is a finite TrPDS. There exist some assembly programs containing local variables that can only be modeled as left contextual TrPDSs.

Remark 8. The instruction set of Intel X86 32-bit is rich, we do not present the general modeling approach in this work. Generally speaking, general assembly programs can be modeled by TrPDSs, as TrPDSs are Turing complete. However, what kind of assembly programs that can be modeled by TrPDSs having decidable reachability is unknown. We believe that decidable TrPDSs are useful to characterize the type of assembly programs that have decidable reachability. We leave this to future work.

8. Related work

Model-checking techniques for PDSs were widely studied and applied to program analysis in the literature [4,1,6–10]. PDSs with checkpoint were introduced in [25] as an extension of PDSs. PDSs with checkpoint can inspect the stack content and are applied to analyze programs with runtime inspection. The reachability problem and LTL model-checking for PDSs with checkpoint were studied in [25] and were applied to the analysis of the HTML5 parser specification in [35]. CTL model-checking for PDSs with checkpoint was studied in [2,3]. A similar extension of PDSs was used to formulate abstract garbage collection in the control flow analysis of higher-order programs [38].

Weighted PDSs and extended weighted PDSs were introduced in [22,23] for data-flow analysis purpose. These two extensions associate transitions with elements from semiring domains. The reachability problem is decidable for bounded idempotent semirings. (Extended) weighted PDSs and TrPDSs are quite different two computation models. At least, the elements from a semiring can neither inspect nor modify the stack content except the top most symbol on the stack. To overcome this problem, weighted pushdown systems with indexed weight domains were proposed in [24,28], which generalize weighted PDSs and TrPDSs.

Recently, well-structured PDSs (WSPDSs) that combine well-structured transition systems and PDSs was introduced by [16] in which the infinite set of control states and the infinite stack alphabet are well-quasi-order. WSPDS is a powerful model in which recursive vector addition system with states [39,40], multi-set PDSs [41] and dense-timed PDSs are subsumed [42]. However, the reachability problem is undecidable for WSPDSs. But coverability becomes decidable when the set of control states is finite. In TrPDSs, the set of control states and the stack alphabet are both finite, but the transductions can inspect and modify the stack content.

We should clarify the relation between our work and the works [27,24,28]. TrPDSs were first introduced in [27] which generalize PDSs with checkpoint and discrete-timed PDSs. The authors showed that TrPDSs can be simulated by PDSs and proposed a saturation procedure to compute pre^* which is different from ours. The main goal of [24,28] is to generalize weighted PDSs and TrPDSs. This article aimed to propose algorithms that are suitable for implementation, relax the restriction of finite TrPDSs and investigate practical applications of TrPDSs. We also proposed a saturation procedure to compute $post^*$ and its algorithm that is suitable for implementation. These two algorithms necessarily improve the complexity due to the fact that the algorithms have better complexity than the saturation procedures for pushdown systems. Moreover, we presented a practical application of TrPDSs to modeling and verifying Boolean programs with call-by-reference parameter passing.

9. Conclusion and future work

We introduced two saturation procedures to compute pre^* and $post^*$ for finite TrPDSs. We also presented two algorithms for the saturation procedures and measured their complexity, based on which we introduced weak finite TrPDSs whose pre^* and $post^*$ can also be computed via the two algorithms. We proposed an extension of finite TrPDSs, called left contextual TrPDSs, and presented an reduction from the reachability problem of left contextual TrPDSs to the problem of finite TrPDSs. We showed that TrPDSs are powerful enough to model Boolean programs with call-by-reference parameter passing and low-level assembly programs that manipulate stack content via a stack pointer. This allows us to verify safety properties of Boolean programs with mixed call-by-reference and call-by-value parameter passing and could be used to characterize the type of assembly programs such that model-checking safety properties are decidable.

In future, we plan to implement our techniques in a tool and investigate BDD-based symbolic algorithms by representing transductions and valuations of global and local variables in BDDs. It is also interesting to investigate algorithms for weighted pushdown systems with indexed weight domains, which subsume finite TrPDSs and could be served as a general framework for program analysis [24,28].

Acknowledgments

We gratefully acknowledge the editor and anonymous reviewers for their valuable comments and suggestions to improve the quality of the article. This work was partially supported by NSFC Projects (Nos. 61402179 and 61532019).

References

[1] J. Esparza, D. Hansel, P. Rossmanith, S. Schwoon, Efficient algorithms for model checking pushdown systems, in: Proceedings of the 12th International Conference on Computer Aided Verification, CAV, Chicago, IL, USA, 2000, pp. 232–247.

[2] F. Song, T. Touili, Efficient CTL model-checking for pushdown systems, in: Proceedings of the 22nd International Conference on Concurrency Theory, CONCUR, Aachen, Germany, 2011, pp. 434–449.
[3] F. Song, T. Touili, Efficient CTL model-checking for pushdown systems, Theor. Comput. Sci. 549 (2014) 127–145, https://doi.org/10.1016/j.tcs.2014.07.001.
[4] A. Bouajjani, J. Esparza, O. Maler, Reachability analysis of pushdown automata: application to model-checking, in: Proceedings of the 8th International Conference on Concurrency Theory, CONCUR, Warsaw, Poland, 1997, pp. 135–150.
[5] A. Finkel, B. Willems, P. Wolper, A direct symbolic approach to model checking pushdown systems, Electron. Notes Theor. Comput. Sci. 9 (1997) 27–37, https://doi.org/10.1016/S1571-0661(05)80426-8.
[6] O. Kupferman, M.Y. Vardi, An automata-theoretic approach to reasoning about infinite-state systems, in: Proceedings of the 12th International Conference on Computer Aided Verification, CAV, Chicago, IL, USA, 2000, pp. 36–52.
[7] O. Kupferman, N. Piterman, M.Y. Vardi, An automata-theoretic approach to infinite-state systems, in: Time for Verification: Essays in Memory of Amir Pnueli, 2010, pp. 202–259.
[8] J. Esparza, S. Schwoon, A BDD-based model checker for recursive programs, in: Proceedings of the 13th International Conference on Computer Aided Verification, CAV, Paris, France, 2001, pp. 324–336.
[9] M. Hague, C.L. Ong, Analysing mu-calculus properties of pushdown systems, in: Proceedings of the 17th International Workshop on Model Checking Software Verification, SPIN, Enschede, The Netherlands, 2010, pp. 187–192.
[10] F. Song, T. Touili, PuMoC: a CTL model-checker for sequential programs, in: Proceedings of IEEE/ACM International Conference on Automated Software Engineering, ASE, Essen, Germany, 2012, pp. 346–349.
[11] L. Breveglieri, A. Cherubini, C. Citrini, S. Crespi-Reghizzi, Multi-push-down languages and grammars, Int. J. Found. Comput. Sci. 7 (3) (1996) 253–292, https://doi.org/10.1142/S0129054196000191.
[12] A. Bouajjani, M. Müller-Olm, T. Touili, Regular symbolic analysis of dynamic networks of pushdown systems, in: Proceedings of the 16th International Conference on Concurrency Theory, CONCUR, San Francisco, CA, USA, 2005, pp. 473–487.
[13] V. Kahlon, F. Ivancic, A. Gupta, Reasoning about threads communicating via locks, in: Proceedings of the 17th International Conference on Computer Aided Verification, CAV, Edinburgh, Scotland, UK, 2005, pp. 505–518.
[14] F. Song, T. Touili, Model checking dynamic pushdown networks, in: Proceedings of the 11th Asian Symposium on Programming Languages and Systems, APLAS, Melbourne, VIC, Australia, 2013, pp. 33–49.
[15] F. Song, T. Touili, Model checking dynamic pushdown networks, Form. Asp. Comput. 27 (2) (2015) 397–421, https://doi.org/10.1007/s00165-014-0330-y.
[16] X. Cai, M. Ogawa, Well-structured pushdown systems, in: Proceedings of the 24th International Conference on Concurrency Theory, CONCUR, Buenos Aires, Argentina, 2013, pp. 121–136.
[17] P.A. Abdulla, M.F. Atig, J. Stenman, Dense-timed pushdown automata, in: Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS, Dubrovnik, Croatia, 2012, pp. 35–44.
[18] P.A. Abdulla, M.F. Atig, J. Stenman, The minimal cost reachability problem in priced timed pushdown systems, in: Proceedings of the 6th International Conference on Language and Automata Theory and Applications, LATA, A Coruña, Spain, 2012, pp. 58–69.
[19] G. Li, X. Cai, M. Ogawa, S. Yuen, Nested timed automata, in: Proceedings of the 11th International Conference on Formal Modeling and Analysis of Timed Systems, FORMATS, Buenos Aires, Argentina, 2013, pp. 168–182.
[20] J. Esparza, A. Kucera, R. Mayr, Model checking probabilistic pushdown automata, in: Proceedings of the 19th IEEE Symposium on Logic in Computer Science, LICS, Turku, Finland, 2004, pp. 12–21.
[21] T. Brázdil, A. Kucera, O. Strazovský, On the decidability of temporal properties of probabilistic pushdown automata, in: Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science, STACS, Stuttgart, Germany, 2005, pp. 145–157.
[22] T.W. Reps, S. Schwoon, S. Jha, Weighted pushdown systems and their application to interprocedural dataflow analysis, in: Proceedings of the 10th International Symposium on Static Analysis, SAS, San Diego, CA, USA, 2003, pp. 189–213.
[23] A. Lal, T.W. Reps, G. Balakrishnan, Extended weighted pushdown systems, in: Proceedings of the 17th International Conference on Computer Aided Verification, CAV, Edinburgh, Scotland, UK, 2005, pp. 434–448.
[24] Y. Minamide, Weighted pushdown systems with indexed weight domains, in: Proceedings of the 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS, Rome, Italy, 2013, pp. 230–244, held as part of the European Joint Conferences on Theory and Practice of Software (ETAPS).
[25] J. Esparza, A. Kucera, S. Schwoon, Model-checking LTL with regular valuations for pushdown systems, in: Proceedings of the 4th International Symposium on Theoretical Aspects of Computer Software, TACS, Sendai, Japan, 2001, pp. 316–339.
[26] X. Li, M. Ogawa, Conditional weighted pushdown systems and applications, in: Proceedings of the 2010 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, PEPM, Madrid, Spain, 2010, pp. 141–150.
[27] Y. Uezato, Y. Minamide, Pushdown systems with stack manipulation, in: Proceedings of the 11th International Symposium on Automated Technology for Verification and Analysis, ATVA, Hanoi, Vietnam, 2013, pp. 412–426.
[28] Y. Minamide, Weighted pushdown systems with indexed weight domains, Log. Methods Comput. Sci. 12 (2) (2016), https://doi.org/10.2168/LMCS-12(2:9)2016.
[29] D. Caucal, On infinite transition graphs having a decidable monadic theory, in: Proceedings of the 23rd International Colloquium on Automata, Languages and Programming, ICALP, Paderborn, Germany, 1996, pp. 194–205.
[30] O. Kupferman, N. Piterman, M.Y. Vardi, Model checking linear properties of prefix-recognizable systems, in: Proceedings of the 14th International Conference on Computer Aided Verification, CAV, Copenhagen, Denmark, 2002, pp. 371–385.
[31] T. Ball, S.K. Rajamani, Bebop: a symbolic model checker for Boolean programs, in: Proceedings of the 7th International Workshop on SPIN Model Checking and Software Verification, SPIN, Stanford, CA, USA, 2000, pp. 113–130.
[32] F. Song, T. Touili, Pushdown model checking for malware detection, in: Proceedings of the 18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS, Tallinn, Estonia, 2012, pp. 110–125.
[33] F. Song, T. Touili, Efficient malware detection using model-checking, in: Proceedings of the 18th International Symposium on Formal Methods, FM, Paris, France, 2012, pp. 418–433.
[34] F. Song, W.K. Miao, G.G. Pu, M. Zhang, On reachability analysis of pushdown systems with transductions: application to boolean programs with call-by-reference, in: Proceedings of the 26th International Conference on Concurrency Theory, CONCUR, Madrid, Spain, 2015, pp. 383–397.
[35] Y. Minamide, S. Mori, Reachability analysis of the HTML5 parser specification and its application to compatibility testing, in: Proceedings of the 18th International Symposium on Formal Methods, FM, Paris, France, 2012, pp. 293–307.
[36] J. Kinder, S. Katzenbeisser, C. Schallhart, H. Veith, Detecting malicious code by model checking, in: Proceedings of the 2nd International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, DIMVA, Vienna, Austria, 2005, pp. 174–187.
[37] M.D. Preda, M. Christodorescu, S. Jha, S.K. Debray, A semantics-based approach to malware detection, in: Proceedings of the 34th ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages, POPL, Nice, France, 2007, pp. 377–388.
[38] C. Earl, I. Sergey, M. Might, D.V. Horn, Introspective pushdown analysis of higher-order programs, in: Proceedings of the ACM SIGPLAN International Conference on Functional Programming, ICFP, Copenhagen, Denmark, 2012, pp. 177–188.
[39] A. Bouajjani, M. Emmi, Analysis of recursively parallel programs, in: Proceedings of the 39th ACM SIGPLAN–SIGACT Symposium on Principles of Programming Languages, POPL, Philadelphia, Pennsylvania, USA, 2012, pp. 203–214.

[40] A. Bouajjani, M. Emmi, Analysis of recursively parallel programs, ACM Trans. Program. Lang. Syst. 35 (3) (2013) 10, https://doi.org/10.1145/2518188.
[41] K. Sen, M. Viswanathan, Model checking multithreaded programs with asynchronous atomic methods, in: Proceedings of the 18th International Conference on Computer Aided Verification, CAV, Seattle, WA, USA, 2006, pp. 300–314.
[42] X. Cai, M. Ogawa, Well-structured pushdown system: case of dense timed pushdown automata, in: Proceedings of the 12th International Symposium on Functional and Logic Programming, FLOPS, Kanazawa, Japan, 2014, pp. 336–352.