Refining the Process Rewrite Systems Hierarchy via Ground Tree Rewrite Systems[1]

Stefan Göller, University of Bremen
Anthony Widjaja Lin, Yale-NUS College, Singapore

In his seminal paper, Mayr introduced the well-known Process Rewrite Systems (PRS) hierarchy, which contains many well-studied classes of infinite-state systems including pushdown systems, Petri nets and PA-processes. A seperate development in the term rewriting community introduced the notion of Ground Tree Rewrite Systems (GTRS), which is a model that strictly extends pushdown systems while still enjoying desirable decidable properties. There have been striking similarities between the verification problems that have been shown decidable (and undecidable) over GTRS and over models in the PRS hierarchy such as PA and PAD processes. It is open to what extent PRS and GTRS are connected in terms of their expressive power. In this paper we pinpoint the exact connection between GTRS and models in the PRS hierarchy in terms of their expressive power with respect to strong, weak, and branching bisimulation. Among others, this connection allows us to give new insights into the decidability results for subclasses of PRS, e.g., simpler proofs of known decidability results of verifications problems on PAD.

Categories and Subject Descriptors: F.1.1 [**Models of Computation**]: Automata, Relations between models; F.4.1 [**Mathematical Logic**]: Temporal Logic; F.4.2 [**Grammars and Other Rewriting Systems**]: Parallel rewriting systems; F.4.3 [**Formal Languages**]: Classes defined by grammars or automata

General Terms: Sequentiality, Parallelism, Concurrency, Rewrite Systems, Expressive Power

Additional Key Words and Phrases: Rewrite Systems, ground terms, trees, hierarchy

ACM Reference Format:
Göller, S., Lin, A.W. 2011. Refining the Process Rewrite Systems Hierarchy via Ground Tree Rewrite Systems. ACM Trans. Comput. Logic V, N, Article A (January YYYY), 27 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

The study of infinite-state verification has revealed that *unbounded recursions* and *unbounded parallelism* are two of the most important sources of infinity in computer programs. Infinite-state models with unbounded recursions such as Basic Process Algebra (BPA), and Pushdown Systems (PDS) have been studied for a long time (e.g. [Baeten et al. 1987; Muller and Schupp 1985]). The same can be said about infinite-state models with unbounded parallelism, which include Basic Parallel Processes (BPP) and Petri nets (PN), e.g. [Christensen 1993; Hack 1976; Esparza and Nielsen 1994]. While these aforementioned models are either *purely sequential* or *purely parallel*, there are also models that simultaneously inherit both of these features. A well-known example are PA-processes [Bergstra and Klop 1985], which are a common generalization of BPA and BPP. It is known that all of these models are not Turing-powerful in the sense that decision problems such as reachability are still decidable (e.g. see [Burkart et al. 2001]), which makes them suitable for verification.

In his seminal paper [Mayr 2000], Mayr introduced the Process Rewrite Systems (PRS) hierarchy (see leftmost diagram in Figure 1) containing several models of infinite-state systems that generalize

[1] An extended abstract of this paper has appeared in the proceedings of CONCUR 2011 [Göller and Lin 2011a].

This work was done while Anthony Lin was at Oxford. He thanks EPSRC (EP/H026878/1) for their generous support.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

the aforementioned well-known models with unbounded recursions and/or unbounded parallelism. The idea is to treat models in the hierarchy as a form of term-rewrite systems, and classify them according to which terms are permitted on the left and right hand side of the rewrite rules. In addition to the aforementioned models of infinite-state systems, the PRS hierarchy contains three new models: (1) Process Rewrite Systems (PRS), which generalize PDS, PA-processes, and Petri nets, (2) PAD-processes, which unify PDS and PA-processes, and (3) PAN-processes, which unify both PA-processes and Petri nets. Mayr showed that the hierarchy is strict with respect to strong bisimulation. Despite its expressive power PRS is not Turing-powerful since reachability is still decidable for this class. Before the PRS hierarchy was introduced, another class of infinite-state systems called Ground Tree/Term Rewrite Systems (GTRS) already emerged in the term rewriting community as a class with good decidability properties. While extending the expressive power of PDS, GTRS still enjoys decidability of reachability (e.g. [Brainerd 1969; Coquidé et al. 1994]), recurrent reachability [Löding 2003], model checking first-order logic with reachability [Dauchet and Tison 1990], and model checking the fragments LTL_{det} and $\mathrm{LTL}(\mathbf{F}_{\mathrm{s}}, \mathbf{G}_{\mathrm{s}})$ of LTL [To and Libkin 2010; To 2010]. Due to the tree structures that GTRS use in their rewrite rules, GTRS can be used to model concurrent systems with both unbounded parallelism (a new thread may be spawned at any given time) and unbounded recursions (each thread may behave as a pushdown system).

When comparing the definitions of PRS (and subclasses thereof) and GTRS, one cannot help but notice their similarity. Moreover, there is a striking similarity between the problems that are decidable (and undecidable) over subclasses of PRS like PA/PAD-processes and GTRS. For example, reachability, EF model checking, and $\mathrm{LTL}(\mathbf{F}_{\mathrm{s}}, \mathbf{G}_{\mathrm{s}})$ and LTL_{det} model checking are decidable for both PAD-processes and GTRS [Bozzelli et al. 2009; Löding 2003; Mayr 2000; 2001; To 2010; To and Libkin 2010]. Furthermore, model checking general LTL properties is undecidable for both PA-processes and GTRS [Bozzelli et al. 2009; To and Libkin 2010]. Despite these, the precise connection between PRS hierarchy and GTRS is currently still open.

Contributions: In this paper, we pinpoint the precise connection between the expressive powers of GTRS and models inside the PRS hierarchy with respect to strong, branching, and weak bisimulation. Bisimulations are well-known and important notions of semantic equivalences on transition systems. Among others, most properties of interests in verification (e.g. those expressible in standard modal/temporal logics) cannot distinguish two transition systems that are bisimilar. Strong/weak bisimulations are historically the most important notions of bisimulations on transition systems in verification [Milner 1989]. Weak bisimulations extend strong bisimulations by distinguishing observable and non-observable (i.e. τ) actions, and only requiring the observable behavior of two systems to agree. In this sense, weak bisimulation is a coarser notion than strong bisimulation. Branching bisimulation [van Glabbeek and Weijland 1996] is a notion of semantic equivalence that is strictly coarser than strong bisimulation but is strictly finer than weak bisimulation. It refines weak bisimulation equivalence by preserving the branching structure of two processes even in the presence of unobservable transitions (that are labeled by a silent action τ); it is required that all intermediate states that are passed through during τ-transitions are related.

Our results are summarized in the middle and right diagrams in Figure 1. Our first main result is that the expressive power of GTRS with respect to branching and weak bisimulation is strictly in between PAD and PRS but incomparable with PAN. This result allows us to transfer many decidability/complexity results of model checking problems over GTRS to PA and PAD processes. In particular, it gives a simple proof of the decidability of the problem of model checking the logic EF over PAD [Mayr 2001], and decidability (with good complexity upper bounds) of the problem of model checking the common fragments LTL_{det} and $\mathrm{LTL}(\mathbf{F}_{\mathrm{s}}, \mathbf{G}_{\mathrm{s}})$ of LTL over PAD (this decidability result was initially given in [Bozzelli et al. 2009] without upper bounds). We also show that Regular Ground Tree Rewrite Systems (RGTRS) [Löding 2003] — the extension of GTRS with possibly infinitely many GTRS rules compactly represented as tree automata — have the same expressive power as GTRS up to branching/weak bisimulation. Our proof technique also implies that PDS is equivalent to prefix-recognizable systems (e.g. see [Burkart et al. 2001]), abbreviated as PREF, up

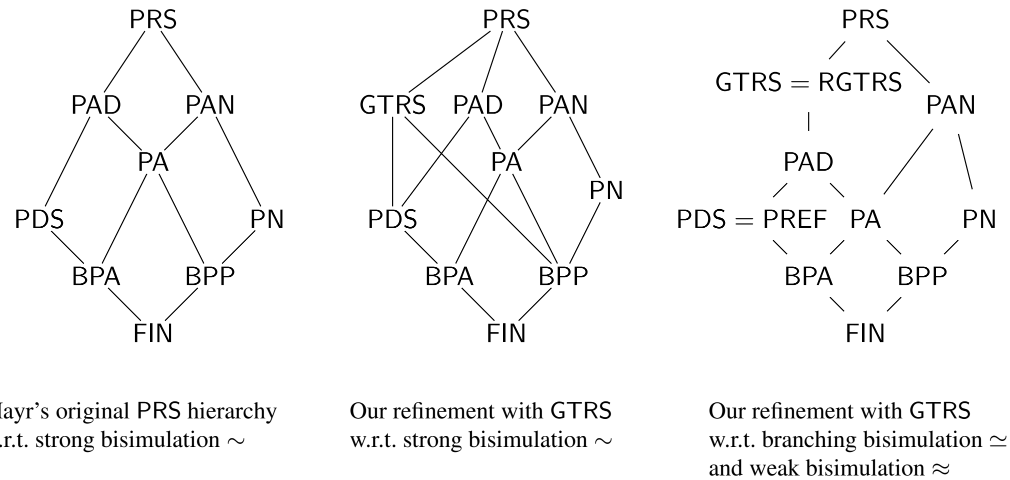

Fig. 1. Depictions of Mayr's PRS hierarchy and their refinements via GTRS as Hasse diagrams (the top being the most expressive). The leftmost diagram is the original (strict) PRS hierarchy where expressiveness is measured with respect to strong bisimulation. The middle (resp. right) diagram is a strict refinement via GTRS with respect to strong (resp. weak/branching) bisimulation.

to branching/weak bisimulation. On the other hand, when we investigate the expressive power of GTRS with respect to strong bisimulation, we found that PAD (even PA) is no longer subsumed in GTRS. Despite this, we can show that up to strong bisimulation GTRS is strictly more expressive than BPP and PDS, and is strictly subsumed in PRS. Finally, we mention that our results imply that Mayr's PRS hierarchy is also strict with respect to weak bisimulation equivalence.

Related work: Our work is inspired by the work of Lugiez and Schnoebelen [Lugiez and Schnoebelen 2002] and Bouajjani and Touili [Bouajjani and Touili 2003], which study PRS (or subclasses thereof) by first distinguishing process terms that are "equivalent" in Mayr's sense [Mayr 2000]. This approach allows them to make use of techniques from classical theory of tree automata for solving interesting problems over PRS (or subclasses thereof). Our translation from PAD to GTRS is similar in spirit.

There are other models of multithreaded programs with unbounded recursions that have been studied in the literature. Specifically, we mention Dynamic Pushdown Networks (DPN) and extensions thereof (e.g. see [Bouajjani et al. 2005]) since an extension of DPN given in [Bouajjani et al. 2005] also extends PAD-processes. We leave it for future work to study the precise connections between these models and GTRS.

Organization: Preliminaries are given in Section 2. We provide the models of infinite-state systems (PRS, GTRS, etc.) in Section 3. Our containment results (e.g. PAD is subsumed in GTRS up to branching bisimulation) can be found in Section 4. Section 5 gives the separation results for the refined PRS hierarchies. Finally, we briefly discuss applications of our results in Section 6.

2. PRELIMINARIES

By $\mathbb{N} = \{0, 1, 2, \ldots\}$ we denote the non-negative integers. For each $i, j \in \mathbb{N}$ we define the interval $[i, j] = \{i, i+1, \ldots, j\}$.

Transition systems and notions of quivalence: Let us fix a countable set of action labels Act. A *transition system* is a tuple $\mathcal{T} = (S, \mathbb{A}, \{\xrightarrow{a} \mid a \in \mathbb{A}\})$, where S is a set of *states*, $\mathbb{A} \subseteq \mathsf{Act}$ is a finite set of action labels, and where $\xrightarrow{a} \subseteq S \times S$ is a set of *transitions* for each $a \in \mathbb{A}$. We write $s \xrightarrow{a} t$ to abbreviate $(s,t) \in \xrightarrow{a}$. We apply similar abbreviations for other binary relations over S. For each $s \in S$ and $R \subseteq S \times S$, we write sR to denote that there is some $t \in S$ with $(s,t) \in R$. For each $\Lambda \subseteq \mathbb{A}$, we define $\xrightarrow{\Lambda} = \bigcup_{a \in \Lambda} \xrightarrow{a}$ and we define $\longrightarrow = \xrightarrow{\mathbb{A}}$. For reasons of better readability, for a binary relation $R \subseteq S \times S$, we also write R_*, R_+ and R_n to denote R^* (the reflexive and transitive closure of R), R^+ (the transitive closure of R), and R^n (the n-fold iteration of R), respectively. Whenever $\mathcal{T}$ is clear from the context and $U \subseteq S$, we define $\mathsf{post}^*_\Lambda(U) = \{t \in S \mid \exists s \in U : s \xrightarrow{\Lambda}{}^* t\}$. In case $U = \{s\}$ is a singleton, we also write $\mathsf{post}^*_\Lambda(s)$ for $\mathsf{post}^*_\Lambda(U)$.

A *pointed transition system* is a pair $(\mathcal{T}, s)$, where $\mathcal{T}$ is a transition system and s is some state of $\mathcal{T}$. Let $\mathcal{T} = (S, \mathbb{A}, \{\xrightarrow{a} \mid a \in \mathbb{A}\})$ be a transition system. A relation $R \subseteq S \times S$ is a *strong bisimulation* if R is symmetric and for each $(s,t) \in R$ and for each $a \in \mathbb{A}$ we have that if $s \xrightarrow{a} s'$, then there is $t \xrightarrow{a} t'$ such that $(s', t') \in R$. We say that s is *strongly bisimilar* to t (abbreviated by $s \sim t$) whenever there is a strong bisimulation R such that $(s,t) \in R$.

Next, we define the notions of branching bisimulation and weak bisimulation. For this, let us fix a *silent action* $\tau \notin \mathbb{A}$ and let $\mathbb{A}_\tau = \mathbb{A} \cup \{\tau\}$. Moreover let $\mathcal{T} = (S, \mathbb{A}_\tau, \{\xrightarrow{a} \mid a \in \mathbb{A}_\tau\})$ be a transition system. We define the binary relations $\xRightarrow{\tau} = (\xrightarrow{\tau})^*$ and $\xRightarrow{a} = (\xrightarrow{\tau})^* \circ \xrightarrow{a} \circ (\xrightarrow{\tau})^*$ for each $a \in \mathbb{A}$.

A binary relation $R \subseteq S \times S$ is a *branching bisimulation* [van Glabbeek and Weijland 1996] if R is symmetric and if, for each $(s,t) \in R$ and $s \xrightarrow{a} s'$, then either of the following two conditions hold: (i) $a = \tau$ and $(s', t) \in R$ or (ii) $a \in \mathbb{A}_\tau$ and we have $t \xRightarrow{\tau} t' \xrightarrow{a} t'' \xRightarrow{\tau} t'''$ satisfying $(s, t') \in R$, $(s', t'') \in R$, and $(s', t''') \in R$. We say that s is *branching bisimilar* to t (abbreviated by $s \simeq t$) whenever there is a branching bisimulation R such that $(s,t) \in R$.

A binary relation $R \subseteq S \times S$ is a *weak bisimulation* if R is symmetric and for each $(s,t) \in R$ and for each $a \in \mathbb{A}_\tau$ we have that if $s \xrightarrow{a} s'$, then there is $t \xRightarrow{a} t'$ such that $(s', t') \in R$. We say that s is *weakly bisimilar* to t (abbreviated by $s \approx t$) whenever there is a weak bisimulation R such that $(s,t) \in R$.

Each of the three introduced bisimulation notions can be generalized between states s_1 and s_2 where s_1 (resp. s_2) is a state of some transition system $\mathcal{T}_1$ (resp. $\mathcal{T}_2$), by simply taking the disjoint union of $\mathcal{T}_1$ and $\mathcal{T}_2$.

Example 2.1. In the following transition system we have $x \approx x'$ but $x \not\simeq x'$:

Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be classes of transition systems and let $\equiv\, \in \{\sim, \simeq, \approx\}$ be some notion of equivalence. We write $\mathcal{C}_1 \leq_\equiv \mathcal{C}_2$ if for every pointed transition system $(\mathcal{T}_1, s_1)$ with $\mathcal{T}_1 \in \mathcal{C}_1$ there exists some pointed transition system $(\mathcal{T}_2, s_2)$ with $\mathcal{T}_2 \in \mathcal{C}_2$ such that $s_1 \equiv s_2$. We write $\mathcal{C}_1 \equiv \mathcal{C}_2$ in case $\mathcal{C}_1 \leq_\equiv \mathcal{C}_2$ and $\mathcal{C}_2 \leq_\equiv \mathcal{C}_1$.

These above-mentioned equivalences can also be characterized by the standard Attacker-Defender game. Strong (resp. weak) bisimilarity can be described by simple pebble games between two players: *Attacker* and *Defender*. Attacker's goal is to prove that two given processes are *not* strongly (resp. *not* weakly) bisimilar, while Defender tries to prove otherwise. We will refer to Attacker as *him* and to Defender as *her*. In every round of the game, there is a pebble placed on a unique state in each transition system. Attacker then chooses one transition system and moves the pebble from the pebbled state to one of its successors by an action $\xrightarrow{a}$, where $a \in \mathbb{A}$ (resp. $a \in \mathbb{A}_\tau$ in the weak bisimulation game). Defender must imitate this by moving the pebbled state from the other system to one of its successors by the same action $\xrightarrow{a}$ (resp. $\xRightarrow{a}$). If one player cannot move, then the other player wins. Defender wins every infinite game. Two states s and t are strongly/weakly bisimilar (resp. not strongly/not weakly bisimilar) if and only if Defender (resp. Attacker) has a winning strategy on the game with initial pebble configuration (s, t).

Ranked trees: Let $\preceq$ denote the prefix order on $\mathbb{N}^*$, i.e. $x \preceq y$ for $x, y \in \mathbb{N}^*$ if there is some $z \in \mathbb{N}^*$ such that $y = xz$, and $x \prec y$ if $x \preceq y$ and $x \neq y$. A *ranked alphabet* is a collection of finite and pairwise disjoint alphabets $A = (A_i)_{i \in [0,k]}$ for some $k \geq 0$. For simplicity we identify A with $\bigcup_{i \in [0,k]} A_i$. A *ranked tree* (over the ranked alphabet A) is a mapping $t : D_t \to A$, where $D_t \subseteq [1,k]^*$ satisfies the following: D_t is non-empty, finite and prefix-closed and for each $x \in D_t$ with $t(x) \in A_i$ we have $x1, \ldots, xi \in D_t$ and $xj \notin D_t$ for each $j > i$. We say that D_t is the *domain* of t — we call these elements *nodes*. A *leaf* is a node x with $t(x) \in A_0$. We also refer to $\varepsilon \in D_t$ as the *root* of t. By Trees_A we denote the set of all ranked trees over the ranked alphabet A. We also use the usual term representation of trees, e.g. if t is a tree with root a and left (resp. right) subtree t_1 (resp. t_2) we have $t = a(t_1, t_2)$.

Let t be a ranked tree and let x be a node of t. We define $xD_t = \{xy \in [1,k]^* \mid y \in D_t\}$ and $x^{-1}D_t = \{y \in [1,k]^* \mid xy \in D_t\}$. By $t^{\downarrow x}$ we denote the *subtree of* t with root x, i.e. the tree with domain $D_{t^{\downarrow x}} = x^{-1}D_t$ defined as $t^{\downarrow x}(y) = t(xy)$. Let $s, t \in \mathsf{Trees}_A$ and let x be a node of t. We define $t[x/s]$ to be the tree that is obtained by replacing $t^{\downarrow x}$ in t by s; more formally $D_{t[x/s]} = (D_t \setminus xD_{t^{\downarrow x}}) \cup xD_s$ with $t[x/s](y) = t(y)$ if $y \in D_t \setminus xD_{t^{\downarrow x}}$ and $t[x/s](y) = s(z)$ if $y = xz$ with $z \in D_s$. Define $|t| = |D_t|$ as the number of nodes in a tree t.

Regular tree languages: A *nondeterministic tree automaton (NTA)* is a tuple $\mathcal{A} = (Q, F, A, \Delta)$, where Q is a finite set of *states*, $F \subseteq Q$ is a set of *final states*, $A = (A_i)_{i\in[0,k]}$ is a ranked alphabet, and $\Delta \subseteq \bigcup_{i\in[0,k]} Q^i \times A_i \times Q$ is the *transition relation*. A *run* of $\mathcal{A}$ on some tree $t \in \mathsf{Trees}_A$ is a mapping $\rho : D_t \to Q$ such that for each $x \in D_t$ with $t(x) \in A_i$ we have $(\rho(x1), \ldots, \rho(xi), t(x), \rho(x)) \in \Delta$. We say ρ is *accepting* if $\rho(\varepsilon) \in F$. By $L(\mathcal{A}) = \{t \in \mathsf{Trees}_A \mid$ there is an accepting run of $\mathcal{A}$ on $t\}$ we denote the *language* of $\mathcal{A}$. A set of trees $U \subseteq \mathsf{Trees}_A$ is *regular* if $U = L(\mathcal{A})$ for some NTA $\mathcal{A}$. The *size* of an NTA $\mathcal{A}$ is defined as $|\mathcal{A}| = |Q| + |A| + |\Delta|$.

3. THE MODELS

3.1. Mayr's PRS hierarchy

In the following, let us fix a countable set of process constants (a.k.a. process variables) $\mathbb{X} = \{A, B, C, D, \ldots\}$. The set of *process terms* is given by the following grammar, where X ranges over $\mathbb{X}$:

$$t, u \quad ::= \quad 0 \quad | \quad X \quad | \quad t.u \quad | \quad t\|u$$

The operator . is said to be *sequential composition*, while the operator $\|$ is referred to as *parallel composition*. In order to minimize clutters, we assume that both operators . and $\|$ are left-associative, e.g., $X_1.X_2.X_3.X_4$ stands for $((X_1.X_2).X_3).X_4$. The *size* $|t|$ of a term is defined as usual. Mayr distinguishes the following classes of process terms:

$\mathbb{1}$ Terms consisting of a single constant $X \in \mathbb{X}$.
$\mathbb{S}$ Process terms without any occurrence of parallel composition.
$\mathbb{P}$ Process terms without any occurrence of sequential composition.
$\mathbb{G}$ Arbitrary process terms possibly with sequential or parallel compositions.

By $\mathbb{1}(\Sigma)$, $\mathbb{S}(\Sigma)$, $\mathbb{P}(\Sigma)$, respectively $\mathbb{G}(\Sigma)$ we denote the set $\mathbb{1}$, $\mathbb{S}$, $\mathbb{P}$, respectively $\mathbb{G}$ restricted to process constants from Σ, for each finite subset $\Sigma \subseteq \mathbb{X}$.

A *process rewrite system (*PRS*)* is a tuple $\mathcal{P} = (\Sigma, \mathbb{A}, \Delta)$, where $\Sigma \subseteq \mathbb{X}$ is a finite set of process constants, $\mathbb{A} \subseteq \mathsf{Act}$ is a finite set of action labels, and Δ is a finite set of rewrite rules of the form $t_1 \mapsto_a t_2$, where $t_1 \in \mathbb{G}(\Sigma) \setminus \{0\}$, $t_2 \in \mathbb{G}(\Sigma)$ and $a \in \mathbb{A}$. Other models in the PRS hierarchy are Finite Systems (FIN), Basic Process Algebras (BPA), Basic Parallel Processes (BPP), Pushdown Systems (PDS), Petri Nets (PN), PA-processes (PA), PAD-processes (PAD), and PAN-processes (PAN). They can be defined by restricting the terms that are allowed on the left/right hand side of the PRS rewrite rules as specified in the following tables.

Model	L.H.S.	R.H.S
FIN	$\mathbb{1}(\Sigma)$	$\mathbb{1}(\Sigma)$
BPA	$\mathbb{1}(\Sigma)$	$\mathbb{S}(\Sigma)$
BPP	$\mathbb{1}(\Sigma)$	$\mathbb{P}(\Sigma)$

Model	L.H.S.	R.H.S
PDS	$\mathbb{S}(\Sigma)$	$\mathbb{S}(\Sigma)$
PA	$\mathbb{1}(\Sigma)$	$\mathbb{G}(\Sigma)$
PN	$\mathbb{P}(\Sigma)$	$\mathbb{P}(\Sigma)$

Model	L.H.S.	R.H.S
PAD	$\mathbb{S}(\Sigma)$	$\mathbb{G}(\Sigma)$
PAN	$\mathbb{P}(\Sigma)$	$\mathbb{G}(\Sigma)$

We follow the approach of [Lugiez and Schnoebelen 2002; Bouajjani and Touili 2003] to define the semantics of PRS. While Mayr [Mayr 2000] directly works on the equivalence classes of terms (induced by some equivalence relation $\equiv$ defined by some axioms including associativity and commutativity of $\|$) to define the dynamics of PRS, we shall initially work on term level. More precisely, given a PRS $\mathcal{P} = (\Sigma, \mathbb{A}, \Delta)$, we write $\mathcal{T}_0(\mathcal{P})$ to denote the transition system $(\mathbb{G}(\Sigma), \mathbb{A}, \{\xrightarrow{a} \mid a \in \mathbb{A}\})$ where $\xrightarrow{a}$ is defined by the following rules:

$$\frac{t_1 \xrightarrow{a} t_1'}{t_1\|t_2 \xrightarrow{a} t_1'\|t_2} \qquad \frac{t_2 \xrightarrow{a} t_2'}{t_1\|t_2 \xrightarrow{a} t_1\|t_2'} \qquad \frac{t_1 \xrightarrow{a} t_1'}{t_1.t_2 \xrightarrow{a} t_1'.t_2} \qquad \frac{}{u \xrightarrow{a} t}\ (u \mapsto_a t) \in \Delta$$

We now define Mayr's semantics of PRS in terms of $\mathcal{T}_0(\mathcal{P})$. First of all, let us define the equivalence relation $\equiv$ on terms using the following proof rules:

$$
\begin{array}{lll}
\dfrac{}{t.0 \equiv t}\ R0. & \dfrac{}{t_1.(t_2.t_3) \equiv (t_1.t_2).t_3}\ A. & \dfrac{t_1 \equiv u_1 \qquad t_2 \equiv u_2}{t_1.t_2 \equiv u_1.u_2}\ \text{Con.} \\[2ex]
\dfrac{}{0.t \equiv t}\ L0. & \dfrac{}{t_1\|(t_2\|t_3) \equiv (t_1\|t_2)\|t_3}\ A\| & \dfrac{t_1 \equiv u_1 \qquad t_2 \equiv u_2}{t_1\|t_2 \equiv u_1\|u_2}\ \text{Con}\| \\[2ex]
\dfrac{}{t\|0 \equiv t}\ R0\| & \dfrac{}{t_1\|t_2 \equiv t_2\|t_1}\ C\| & \dfrac{u \equiv u' \qquad u' \equiv u''}{u \equiv u''}\ \text{Trans} \\[2ex]
\dfrac{}{0\|t \equiv t}\ L0\| & \dfrac{}{u \equiv u}\ \text{Ref} & \dfrac{t \equiv u}{u \equiv t}\ \text{Sym}
\end{array}
$$

Here, u, t, t_i, u_i range over all terms in $\mathbb{G}$. Intuitively, the axioms defining $\equiv$ say that 0 is *identity*, while the operator . (resp. $\|$) is associative (resp. associative and commutative). The rules (Con.) and (Con$\|$) are standard *context rules* in process algebra saying that term equivalence is preserved under substitutions of equivalent subterms. Finally, Trans, Sym, and Ref state that $\equiv$ is an equivalence relation. In the sequel, we also use the symbol $\equiv_1$ to denote the equivalence relation on process terms that allows all the above axioms except for (A$\|$) and (C$\|$). Obviously, $\equiv_1 \subseteq \equiv$. Given a term $t \in \mathbb{G}$, we denote by $[t]_\equiv$ (resp. $[t]_{\equiv_1}$) the $\equiv$-class (resp. $\equiv_1$-class) containing t.

Mayr's semantics on a PRS $\mathcal{P} = (\Sigma, \mathbb{A}, \Delta)$ such that $\mathcal{T}_0(\mathcal{P}) = (\mathbb{G}(\Sigma), \mathbb{A}, \{\xrightarrow{a} \mid a \in \mathbb{A}\})$ is a transition system $\mathcal{T}(\mathcal{P}) = (S, \mathbb{A}, \{E_a \mid a \in \mathbb{A}\})$, where $S = \{[t]_\equiv \mid t \in \mathbb{G}(\Sigma)\}$ and where $(C, C') \in E_a$ iff there exist $t \in C$ and $t' \in C'$ such that $t \xrightarrow{a} t'$. An important result by Mayr [Mayr 2000] is that the PRS hierarchy is strict with respect to strong bisimulation.

3.2. (Regular) ground tree rewrite systems and prefix-recognizable systems

A *regular ground tree rewrite system* (RGTRS) is a tuple $\mathcal{R} = (A, \mathbb{A}, R)$, where A is a ranked alphabet, $\mathbb{A} \subseteq$ Act is a finite set of action labels and where R is finite set of rewrite rules $L \stackrel{a}{\hookrightarrow} L'$, where L and L' are regular tree languages given as NTA. The transition system defined by $\mathcal{R}$ is $\mathcal{T}(\mathcal{R}) = (\mathsf{Trees}_A, \mathbb{A}, \{\xrightarrow{a} \mid a \in \mathbb{A}\})$, where for each $a \in \mathbb{A}$, we have $t \xrightarrow{a} t'$ if and only if there is some $x \in D_t$ and some rule $L \stackrel{a}{\hookrightarrow} L' \in R$ such that $t^{\downarrow x} \in L$ and $t' = t[x/s']$ for some $s' \in L'$ (we say that the rule was applied at node x).

A *ground tree rewrite system* (GTRS) is an RGTRS $\mathcal{R} = (A, \mathbb{A}, R)$, where for each $L \stackrel{a}{\hookrightarrow} L' \in R$ we have that both $L = \{t\}$ and $L' = \{t'\}$ is a singleton; we also write $t \stackrel{a}{\hookrightarrow} t' \in R$ for this.

A *prefix-recognizable system* (PREF) is an RGTRS $\mathcal{R} = (A, \mathbb{A}, R)$, where only A_0 and A_1 may be non-empty. We note that analogously pushdown systems can be defined as GTRS $\mathcal{R} = (A, \mathbb{A}, R)$, where only A_0 and A_1 may be non-empty.

4. CONTAINMENT RESULTS

In this section, we prove the following containment results: PAD $\leq_\simeq$ GTRS (Section 4.1), BPP $\leq_\sim$ GTRS and GTRS $\leq_\sim$ PRS, and finally RGTRS $=_\simeq$ GTRS (Section 4.2).

4.1. PAD $\leq_\simeq$ GTRS

THEOREM 4.1. *Given a* PAD $\mathcal{P} = (\Sigma, \mathbb{A}, \Delta)$ *and a term* $t_0 \in \mathbb{G}(\Sigma)$*, there exists a* GTRS $\mathcal{R} = (A, \mathbb{A}_\tau, R)$ *and a tree* $t_0' \in \mathsf{Trees}_A$ *such that* $(\mathcal{T}(\mathcal{P}), [t_0]_\equiv)$ *is branching bisimilar to* $(\mathcal{T}(\mathcal{R}), t_0')$*. Furthermore,* $\mathcal{R}$ *and* t_0' *may be computed in time polynomial in* $|\mathcal{P}| + |t_0|$.

Before proving this theorem, we shall first present the general proof strategy. The main difficulty of the proof is that the domain of $\mathcal{T}(\mathcal{P})$ consists of $\equiv$-classes of process terms, while the domain of $\mathcal{T}(\mathcal{R})$ consists of ranked trees. On the other hand, observe that the other semantics $\mathcal{T}_0(\mathcal{P})$ is closer to a GTRS since the domain of $\mathcal{T}_0(\mathcal{P})$ consists of process terms (not equivalence classes thereof). Therefore, the first hurdle in the proof is to establish a connection between $\mathcal{T}(\mathcal{P})$ and $\mathcal{T}_0(\mathcal{P})$. To this end, we will require that t_0 *and all process terms in* $\mathcal{P}$ *have a minimum number of zeros and have no right-associative occurrence of the sequential composition operator*. We will then pick a small subset of the axioms of $\equiv$ as τ-transitions, which we will add to $\mathcal{T}_0(\mathcal{P})$. These axioms include those that reduce the occurrences of 0 from terms, and the rule that turns a right-associative occurrence of the sequential composition operator into a left-associative occurrence. The resulting pointed transition system $(\mathcal{T}_0(\mathcal{P}), t_0)$ will become branching bisimilar to $(\mathcal{T}(\mathcal{P}), [t_0]_\equiv)$. In fact, fixing t_0 as the initial configuration, we will see that further restrictions to the axioms for $\equiv$ (e.g. associativity of .) may be made resulting in a pointed transition system that can be easily captured in the framework of GTRS.

Adding the τ-transitions to $\mathcal{T}_0(\mathcal{P})$: We define the relation $\xrightarrow{\tau}$ on arbitrary process terms given by the following proof rules:

$$\frac{}{0.t \xrightarrow{\tau} t} \qquad \frac{}{t\|0 \xrightarrow{\tau} t} \qquad \frac{t_1 \xrightarrow{\tau} t_1'}{t_1.t_2 \xrightarrow{\tau} t_1'.t_2}$$

$$\frac{}{t.0 \xrightarrow{\tau} t} \qquad \frac{}{t_1.(t_2.t_3) \xrightarrow{\tau} (t_1.t_2).t_3} \qquad \frac{t_2 \xrightarrow{\tau} t_2'}{t_1.t_2 \xrightarrow{\tau} t_1.t_2'}$$

$$\frac{}{0\|t \xrightarrow{\tau} t} \qquad \frac{t_1 \xrightarrow{\tau} t_1'}{t_1\|t_2 \xrightarrow{\tau} t_1'\|t_2} \qquad \frac{t_2 \xrightarrow{\tau} t_2'}{t_1\|t_2 \xrightarrow{\tau} t_1\|t_2'}$$

Here, t, t_i, t_i' are allowed to be any process terms. Observe that these τ-transitions remove redundant occurrences of 0 and turn a right-associative occurrence of the sequential composition into a left-associative one. Observe that we do not allow associativity/commutativity axioms for $\|$ in our definition of $\xrightarrow{\tau}$. It is easy to see that $\xrightarrow{\tau} \subseteq \equiv_1 \subseteq \equiv$. We now prove a few technical properties about $\xrightarrow{\tau}$ in the following lemmas.

LEMMA 4.2. *The following statements hold:*

— *(Termination) For all terms t, there exists a unique term $t_\downarrow$ such that $t \xrightarrow{\tau}_* t_\downarrow$ and $t_\downarrow \not\xrightarrow{\tau}$. Furthermore, all paths from t to $t_\downarrow$ are of length at most $O(|t|^2)$, and moreover $t_\downarrow$ is computable from t in polynomial time.*

— *(Confluence) For all terms $t \equiv_1 t'$, there exists t'' such that $t \xrightarrow{\tau}_* t''$ and $t' \xrightarrow{\tau}_* t''$.*

Lemma 4.2 is a basic property of a rewrite system commonly known as *confluence* and *termination* (e.g. see [Baader and Nipkow 1998]). In fact, it does not take long (i.e. polynomially many steps in the worst case) to terminate.

PROOF. Confluence is easy to prove by induction on the height of the proof tree that $t \equiv t'$.

It remains to prove termination. We first prove the existence of a term u such that $t \xrightarrow{\tau}_* u$ and $u \not\xrightarrow{\tau}$. In doing so, we will also prove that all paths from t to u are of length at most $O(|t|^2)$.

We define a function Value : $\mathbb{G} \to \mathbb{N}$ that measures how many zeros a process term has and "how right-associative" it is. More formally, given a process term $t = (D, \tau)$, where $D \subseteq \{1,2\}^*$ is a tree domain and $\tau : D \to \Sigma$ is the labeling function, we let Value(t) be the sum of two values: (1) the number of occurrences of 0 in t and (2) $\sum_{w \in D, \tau(w) = .} |w|_2$, where $|w|_2$ counts the number of occurrences of 2 (i.e. a right turn made when traversing from the root) in w. Note that the value of item (2) counts the number of right turns made when traversing from the root in all node locations labeled by a sequential composition. It is easy to see that Value$(t) \leq O(|t|^2)$. It is also easy to prove that, for all terms t_1, t_2, if $t_1 \xrightarrow{\tau} t_2$ then we have Value$(t_2) <$ Value(t_1). This can be easily proven by induction on the height of the proof trees that witnesses $t_1 \xrightarrow{\tau} t_2$. Therefore, this proves the existence of u such that $t \xrightarrow{\tau}_* u$ and $u \not\xrightarrow{\tau}$. Now, by confluence, we also have uniqueness of such u. □

LEMMA 4.3. *The following statements hold: (1) If $t \equiv_1 t'$, then $t_\downarrow = t'_\downarrow$, (2) If $0 \equiv v$, then $v \xrightarrow{\tau}_* 0$, and (3) If $X_1.X_2 \ldots X_n \equiv v$, then $v \xrightarrow{\tau}_* X_1.X_2 \ldots X_n$.*

Lemma 4.3 gives the form of the unique "minimal" term with respect to $\xrightarrow{\tau}$ given various different initial starting points.

PROOF. Statement (1) is an easy corollary of Lemma 4.2. Statement (2) can be proven by induction on the size of v in the same way as in the proof of Lemma 4.2.

We now prove (3). To this end, define Par to be a function mapping an arbitrary term t to a number n counting the number of subterms $t_1 \| t_2$ in t such that $t_1 \not\equiv 0$ and $t_2 \not\equiv 0$ (i.e. each t_i contains at least one occurrence of a process constant). It is easy to see that $t \equiv t'$ implies that Par$(t) =$ Par(t'). This can be easily proven by induction on the height of the proof tree that witnesses $t \equiv t'$. Therefore, by (2), subterms that are $\equiv$-equivalent to 0 will be replaced by 0 using τ-transitions. Therefore, each $v \equiv X_1.X_2 \ldots X_n$ satisfies $v_\downarrow = Y_1.Y_2 \ldots Y_m$ for some $m \in \mathbb{N}$. It is easy to see that $X_1.X_2 \ldots X_n = Y_1.Y_2 \ldots Y_m$. This can be done by showing that (i) for each $v, v' \in \mathbb{G}$ such that $v \equiv X_1.X_2 \ldots X_n \equiv v'$ it is the case that $f(v) = f(v')$ for the function f defined in the proof of Lemma 4.2, and (ii) use the fact (from Proof of Lemma 4.2) that each element of the image of f has a unique minimal representation with respect to $\xrightarrow{\tau}$. Statement (i) can easily be proved by induction on the proof trees that witness $v \equiv v'$ and by using the fact that (a) $f(t) = 0$ if $t \equiv 0$, and (b) Par$(v) =$ Par$(v') = 0$ (by the above discussions). □

For the rest of the proof of Theorem 4.1, we assume the following conventions:

CONVENTION 4.4. *The term t_0 and all process terms that appear in the rewrite rules Δ of $\mathcal{P}$ are minimal with respect to $\xrightarrow{\tau}$. That is, each of these terms t satisfies $t = t_\downarrow$.*

We now add these τ-transitions into $\mathcal{T}_0(\mathcal{P})$. So, we will write $\mathcal{T}_0(\mathcal{P}) = (\mathbb{G}(\Sigma), \mathbb{A}_\tau, \{\xrightarrow{a}: a \in \mathbb{A}_\tau\})$. Our first technical result is that the equivalence relation $\equiv$ is indeed a branching bisimulation on $\mathcal{T}_0(\mathcal{P})$.

We recall that $\mathbb{G}(\Sigma)$ is state set of $\mathcal{T}_0(\mathcal{P})$ and that $\mathbb{G}(\Sigma)/\equiv$ is the state of $\mathcal{T}(\mathcal{P})$.

LEMMA 4.5. *$\equiv$ is a branching bisimulation on $\mathcal{T}_0(\mathcal{P})$.*

PROOF. Take arbitrary terms $u, t \in \mathbb{G}(\Sigma)$. We assume that $u \equiv t$. Obviously, for all $u' \in \mathbb{G}(\Sigma)$, if $u \xrightarrow{\tau} u'$, then $u' \equiv t$ since $\xrightarrow{\tau} \subseteq \equiv$. Therefore, it suffices to prove the following for each $a \in \mathbb{A}$: **(C1)** if there exists $u' \in \mathbb{G}(\Sigma)$ such that $u \xrightarrow{a} u'$, then there exists $t_1, t' \in \mathbb{G}(\Sigma)$ such that $t \xrightarrow{\tau}_* t_1 \xrightarrow{a} t'$, and that $u \equiv t_1$, $u' \equiv t'$; and **(C2)** if there exists $t' \in \mathbb{G}(\Sigma)$ such that $t \xrightarrow{a} t'$, then there exists $u_1, u' \in \mathbb{G}(\Sigma)$ such that $u \xrightarrow{\tau}_* u_1 \xrightarrow{a} u'$, and that $u_1 \equiv t$, $u' \equiv t'$.

We will prove this by induction on the height of the proof trees that witness $u \equiv t$. For the base cases, we consider proof trees of height 1. We will only look at several of these cases (the rest being similar):

(1) Rule (R0.). In this case, $u = v.0$ and $t = v$. Condition (C2) is obvious since $u \xrightarrow{\tau} t$. For condition (C1), observe that our PAD $\mathcal{P}$ do not admit occurrences of 0 in the term on the left side of a rule. Therefore, by the definition of $\xrightarrow{a}$, it is the case that $u \xrightarrow{a} u'$ implies $u' = v'.0$ for some $v' \in \mathbb{G}(\Sigma)$ satisfying $v \xrightarrow{a} v'$. Therefore, we have $t \xrightarrow{a} v'$ and $u' \equiv v'$.

(2) Rule (L0.). Condition (C1) is vacuous since $0.t \not\xrightarrow{a}$ by the definition of $\xrightarrow{a}$. Condition (C2) is true since $0.t \xrightarrow{\tau} t$.
(3) Rule (A.). Condition (C2) is obvious since $v_1.(v_2.v_3) \xrightarrow{\tau} (v_1.v_2).v_3$. Condition (C1) can be seen from the fact that the left side of a term on the left side of a rule does not have a subterm of the form $v_1.(v_2.v_3)$. Therefore, if $v_1.(v_2.v_3) \xrightarrow{a} u'$, then we have $v_1 \xrightarrow{a} v_1'$. Hence, $(v_1.v_2).v_3 \xrightarrow{a} (v_1'.v_2).v_3 \equiv v_1'.(v_2.v_3)$.

Let us now look at the inductive cases:

(1) Rule (Sym). This case is immediate from the inductive hypothesis due to the symmetry between (C1) and (C2).
(2) Rule (Trans). This case is also immediate from the induction hypothesis and the transitivity of branching bisimulation.
(3) Rule (Con.). We will only prove (C1) since (C2) is symmetric in this case. We have $u_1.u_2 \xrightarrow{a} u'$, $u = u_1.u_2$, $t = t_1.t_2$, and $u \equiv t$. There are two cases to consider. First, the formal proof of $u_1.u_2 \xrightarrow{a} u'$ is of height 1, which implies that $(u_1.u_2, a, u') \in \Delta$. Since the left side of a PAD rule is of the form $X_1.X_2 \dots X_n$, Lemma 4.3 implies that $t \xrightarrow{\tau}_* u$. Therefore, (C1) is immediately satisfied. The second case is when the height of the formal proof of $u_1.u_2 \xrightarrow{a} u'$ is of height > 1. In this case, we have $u_1 \xrightarrow{a} u_1'$. We may then invoke the induction hypothesis on $u_1 \equiv t_1$ and immediately obtain (C1).
(4) Rule (Con$\|$). This is the same as the second case of the previous item. □

As an immediate corollary, we obtain that $(\mathcal{T}_0(\mathcal{P}), t_0)$ is equivalent to $(\mathcal{T}(\mathcal{P}), [t_0]_\equiv)$ up to branching bisimulation.

COROLLARY 4.6. *The relation $R = \{(C, t) \subseteq \mathbb{G}(\Sigma)/\equiv \times \mathbb{G}(\Sigma) : t \in C\}$ is a branching bisimulation between $\mathcal{T}(\mathcal{P})$ and $\mathcal{T}_0(\mathcal{P})$.*

PROOF. We write $\mathcal{T}(\mathcal{P}) = (\mathbb{G}(\Sigma)/\equiv, \mathbb{A}, \{E_a : a \in \mathbb{A}\})$. In the following, we will take arbitrary process terms $t, t' \in \mathbb{G}(\Sigma)$ and $\equiv$-classes $C, C' \in \mathbb{G}(\Sigma)/\equiv$. We assume in this proof that $t \in C$.

If $t \xrightarrow{\tau} t'$, then trivially $t' \in C$. If $t \xrightarrow{a} t'$ and $t \in C$, then $(C, C') \in E_a$ with $t' \in C'$ by definition of E_a.

If $C \xrightarrow{a} C'$, then there exists $u, u' \in S$ such that $u \in C$, $u' \in C'$ and that $u \xrightarrow{a} u'$. But then $u \equiv t$ and so by Lemma 4.5 there exists a sequence

$$t \xrightarrow{\tau}_* t_2 \xrightarrow{a} t_3 \xrightarrow{\tau}_* t'$$

such that $u \equiv t_2$, $u' \equiv t_3$, and $u' \equiv t'$. Therefore, that $t' \in C'$ and so the proof is complete. □

Removing complex τ-transitions: Corollary 4.6 implies that we may restrict ourselves to the transition system $\mathcal{T}_0(\mathcal{P})$. At this stage, our τ-transitions still contain some rules that cannot easily be captured in the framework of GTRS, e.g., left-associativity rule of the sequential composition. We will now show that fixing an initial configuration t_0 allows us to remove these τ-transitions from our systems.

Recall that our initial configuration t_0 satisfies $t_0 = (t_0)_\downarrow$. Denote by W the set of all subterms (either of t_0 or of a left/right side of a rule in $\mathcal{P}$) rooted at a node that is a right child of a .-labeled node. It is easy to see that Convention 4.4 implies that each $t \in W$ satisfies $t = t_\downarrow$. Consequently, each $t \in W$ cannot be of the form $t_1.t_2$ or 0 since t is a right child of the sequential composition. Furthermore, $|W|$ is linear in the size of $\mathcal{P}$.

LEMMA 4.7. *Fix a term $t \in post^*(t_0)$ with respect to $\mathcal{T}_0(\mathcal{P})$. Then, any subterm of t which is a right child of a .-labeled node is in W.*

This lemma can be easily proved by induction on the length of the witnessing path that $t \in post^*(t_0)$ and that this invariant is always satisfied. This lemma implies that some of the rules for defining

$\xrightarrow{\tau}$ may be restricted when only considering $post^*(t_0)$ as the domain of our system, resulting in the following simplified definition:

$$\frac{}{0.t \xrightarrow{\tau} t}\; t \in W \qquad \frac{}{t\|0 \xrightarrow{\tau} t} \qquad \frac{t_1 \xrightarrow{\tau} t_1'}{t_1.t_2 \xrightarrow{\tau} t_1'.t_2}\; t_2 \in W$$
$$\frac{}{0\|t \xrightarrow{\tau} t} \qquad \frac{t_2 \xrightarrow{\tau} t_2'}{t_1\|t_2 \xrightarrow{\tau} t_1\|t_2'} \qquad \frac{t_1 \xrightarrow{\tau} t_1'}{t_1\|t_2 \xrightarrow{\tau} t_1'\|t_2}$$

Observe that the rule $t.0 \xrightarrow{\tau} t$ may be omitted since no subterm of $t \in post^*(t_0)$ of the form $u.0$ exists. Moreover, the rule $t_1.(t_2.t_3) \xrightarrow{\tau} (t_1.t_2).t_3$ is never applicable since no subterm of $t \in post^*(t_0)$ of the form $t_1.(t_2.t_3)$ exists. Other rules are omitted because any subterm of t of the form $t_1.t_2$ must satisfy $t_2 \in W$, and that each $u \in W$ satisfies $u = u_{\downarrow}$ (which implies $u \not\xrightarrow{\tau}$).

Finally, in order to cast the system into the GTRS framework, we will further restrict rules of the form $t\|0 \xrightarrow{\tau} t$ or $0\|t \xrightarrow{\tau} t$. Let l-prefix($\mathcal{P}$) be the set of all prefixes of words w appearing on the left hand side of the rules in $\mathcal{P}$ treated as left-associative terms. More formally, l-prefix($\mathcal{P}$) contains 0 (a term representation of the empty word) and all subterms u of a term appearing on the left hand side of a rule in $\mathcal{P}$ rooted at a node location of the form 1^*. We define $\rightsquigarrow_\tau$ to be the restriction of $\xrightarrow{\tau}$, where rules of the form $0\|t \xrightarrow{\tau} t$ and $t\|0 \xrightarrow{\tau} t$ are restricted to $t \in$ l-prefix($\mathcal{P}$). We let $\mathcal{T}_0'(\mathcal{P})$ to be $\mathcal{T}_0(\mathcal{P})$ with $\xrightarrow{\tau}$ replaced by $\rightsquigarrow_\tau$.

LEMMA 4.8. *$(\mathcal{T}_0'(\mathcal{P}), t)$ is branching bisimilar to $(\mathcal{T}_0(\mathcal{P}), t)$.*

PROOF. Define $\equiv_2$ to be the equivalence relation on process terms that are generated by Axioms (R0$\|$), (L0$\|$), (Ref), (Con.), (Con$\|$), (Trans), and (Sym). Obviously, we have $\equiv_2 \subseteq \equiv_1 \subseteq \equiv$. It is not hard to prove that $\equiv_2$ is the desired branching bisimulation relation. This can be easily proven on the induction on the height of the proof trees that witness $u \equiv_2 t$. □

Constructions of the GTRS: It is now not difficult to cast $\mathcal{T}_0'(\mathcal{P})$ into GTRS framework. To construct the GTRS, we let A be the ranked alphabet containing: (i) a nullary symbol for each process variable occuring in $\mathcal{P}$, (ii) a binary symbol for the binary operator $\|$, and (iii) a unary symbol $\hat{t}$ for each term $t \in W$. Since each subterm u of a term $t \in post^*(t_0)$ of the form $t_1.t_2$ satisfies $t_2 \in W$, we may simply substitute u with the term $\hat{t_2}(t_1)$ and perform this substitution recursively on t_1. Denote by $\lambda(t)$ the resulting term over the new alphabet A after this substitution is performed on a process term t. The desired GTRS is $\mathcal{R} = (A, \mathbb{A}_\tau, R)$, where R is defined as follows. For each rule $t \mapsto_a t'$ in $\mathcal{P}$, where $a \in \mathbb{A}$, we add the rule $\lambda(t) \overset{a}{\hookrightarrow} \lambda(t')$ to R. For each $t \in$ l-prefix($\mathcal{P}$), we add $0\|t \overset{\tau}{\hookrightarrow} t$ and $t\|0 \overset{\tau}{\hookrightarrow} t$ to R. Finally, we add the transition rule $\hat{t}(0) \xrightarrow{\tau} t$ for each $t \in W$. It is now not difficult to show that $(\mathcal{T}_0'(\mathcal{P}), t) \simeq (\mathcal{T}(\mathcal{R}), \lambda(t))$, which immediately implies Theorem 4.1.

4.2. Further containment results

In this section we prove all remaining containment results for completing Figure 1.

THEOREM 4.9. BPP $\leq_\sim$ GTRS.

PROOF. Let $\mathcal{P} = (\Sigma, \mathbb{A}, \Delta)$ be some BPP. Let A be the ranked alphabet with $A_0 = \Sigma \uplus \{\$\}$ and $A_2 = \{\bullet\}$. For each parallel term $\alpha = Y_1\| \cdots \|Y_n \in \mathbb{P}(\Sigma)/\equiv$, with $Y_i \in \Sigma$ for each $i \in \{1, \ldots, n\}$, we define the ranked tree $t(\alpha)$ inductively on n: If $n = 0$ then $t(\alpha) = \$$, if $n \geq 1$ then $t(\alpha) = \bullet(Y_n, t(Y_1\| \cdots \|Y_{n-1}))$.

We define the GTRS $\mathcal{R} = (A, \mathbb{A}, R)$, where $R = \{X \overset{a}{\hookrightarrow} t(\alpha) \mid (X \mapsto_a \alpha) \in \Delta\}$. For any (parallel) process term $\alpha \in \mathbb{P}(\Sigma)$ with respect to $\mathcal{T}(\mathcal{P})$ one can easily check that α is strongly bisimilar to $t(\alpha)$ with respect to $\mathcal{T}(\mathcal{R})$. □

THEOREM 4.10. GTRS $\leq_\sim$ PRS.

PROOF. Let $\mathcal{R} = (A, \mathbb{A}, R)$ be a GTRS. We will construct a PRS $\mathcal{P} = (\Sigma, \mathbb{A}, \Delta)$ and a mapping $\mu : \mathsf{Trees}_A \to \mathbb{G}(\Sigma)$ such that each $t \in \mathsf{Trees}_A$ with respect to $\mathcal{T}(\mathcal{R})$ is strongly bisimilar to $\mu(t)$ with respect to $\mathcal{T}(\mathcal{P})$. Assume the maximal rank of A is k. Then we put $\Sigma = A \uplus \{X_1, \ldots, X_k\}$. Inductively we define $\mu(t)$ as follows: $\mu(a) = a$ for each $a \in A_0$ and $\mu(f(t_1, \ldots, t_\ell)) = \left(\mu(t_1).X_1 \parallel \cdots \parallel \mu(t_\ell).X_\ell\right).f$ for each $f \in A_\ell$ with $\ell \geq 1$. We define the set of rewrite rules as $\Delta = \{\mu(t) \mapsto_a \mu(t') \mid t \overset{a}{\hookrightarrow} t' \in R\}$. □

THEOREM 4.11. RGTRS $\simeq$ GTRS.

PROOF. Since for every GTRS there is an isomorphic RGTRS it remains to prove that RGTRS $\leq_\simeq$ GTRS. Assume an RGTRS $\mathcal{R} = (A, \mathbb{A}, R)$, assume k is the maximal rank of A, and assume $L_1 \overset{a_1}{\hookrightarrow} L'_1, \ldots, L_n \overset{a_n}{\hookrightarrow} L'_n$ to be an enumeration of R. We assume that each L_i is given by the NTA $\mathcal{A}_i = (Q_i, F_i, A, \Delta_i)$ and L'_i is given by the NTA $\mathcal{A}'_i = (Q'_i, F'_i, A, \Delta'_i)$ for each $i \in \{1, \ldots, n\}$. We will construct a GTRS $\mathcal{R}' = (A', \mathbb{A}, R')$ with $A \subseteq A'$ such that every $t \in \mathsf{Trees}_A$ with respect to $\mathcal{T}(\mathcal{R})$ is branching bisimilar to t with respect to $\mathcal{T}(\mathcal{R}')$.

We put $A'_i = A_i$ for each $1 \leq i \leq k$ and $A'_0 = A_0 \uplus \biguplus_{i \in [1,n]} Q_i \uplus Q'_i$. For each rule $(q_1, \ldots, q_j, a, q) \in \bigcup_{i \in [1,n]} \Delta_i \uplus \Delta'_i$ we add the rules $a(q_1, \ldots, q_j) \overset{\tau}{\hookrightarrow} q$ and $q \overset{\tau}{\hookrightarrow} a(q_1, \ldots, q_j)$ to R' plus the rules $\{q \overset{a_i}{\hookrightarrow} q' \mid i \in \{1, \ldots, n\}, q \in F_i, q' \in F'_i\}$. The reader now easily verifies that by construction we have that in $\mathcal{T}(\mathcal{R}')$ the relation $\xrightarrow{\tau}_*$ is an equivalence relation. For every $t \in \mathsf{Trees}_{A'}$, let $[t]$ denote the equivalence class of t with respect to $\xrightarrow{\tau}_*$. Finally one easily verifies that the relation $R = \{(t, t') \mid t \in \mathsf{Trees}_A, t' \in [t]\} \subseteq \mathsf{Trees}_A \times \mathsf{Trees}_{A'}$ is a branching bisimulation between $\mathcal{T}(\mathcal{R})$ and $\mathcal{T}(\mathcal{R}')$ that relates each t with respect to $\mathcal{T}(\mathcal{R})$ with t with respect to $\mathcal{T}(\mathcal{R}')$. □

In analogy to Theorem 4.11 one can prove the following.

COROLLARY 4.12. PDS $\simeq$ PREF.

5. SEPARATION RESULTS

In this section, we provide the separation results in the two refined hierarchies. We first note two known separation results: (1) BPA $\not\leq_\approx$ PN (e.g. see [Christensen 1993]), and (2) BPP $\not\leq_\approx$ PDS since there is a BPP trace language that is not context-free (e.g. see references in [Bouajjani et al. 1995]) and trace equivalence is coarser than weak bisimulation equivalence.

5.1. PA $\not\leq_\sim$ GTRS

Some properties of GTRS: We introduce some notions that were also used in [Löding 2003]. Let $\mathcal{R} = (A, \mathbb{A}, R)$ be an arbitrary GTRS. For each $t \in \mathsf{Trees}_A$, we define $\mathsf{height}(t) = \max\{|x| : x \in D_t\}$. We define the number $h_\mathcal{R} = \max\{\mathsf{height}(t) \mid \exists t' \in \mathsf{Trees}_A\ \exists \sigma \in \mathbb{A} : t \overset{\sigma}{\hookrightarrow} t' \in R \text{ or } t' \overset{\sigma}{\hookrightarrow} t \in R\}$ and $|\mathcal{R}| = |R| \cdot h_\mathcal{R} + |A| + |\mathbb{A}|$. In other words, $h_\mathcal{R}$ is the maximum height of trees on left/right hand sides of any rule in $\mathcal{R}$. We start off by proving a pumping lemma for GTRS:

LEMMA 5.1. *Let $\Lambda \subseteq \mathbb{A}$. For every $t_0 \in \mathsf{Trees}_A$ there is some $N \leq \exp(|\mathcal{R}| + \mathit{height}(t_0))$ such that $t_0 \xrightarrow{\Lambda}_N$ implies $t_0 \xrightarrow{\Lambda}_n$ for infinitely many $n \in \mathbb{N}$.*

PROOF. First, it is easy to see that there is some NTA $\mathcal{A}$ with $|\mathcal{A}| \leq O(|t_0|)$ such that $L(\mathcal{A}) = \{t_0\}$. It is well known that there is some NTA $\mathcal{B}$ with $L(\mathcal{B}) = \mathsf{post}^*_\Lambda(t_0)$ and $|\mathcal{B}| \leq \mathsf{poly}(|t_0| + |\mathcal{R}|)$,

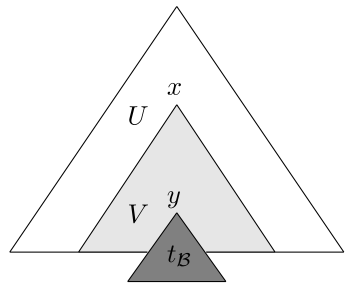

Fig. 2. The tree $T^1 = U[V[t_{\mathcal{B}}]]$.

see also [Löding 2003]. We define

$$N \quad = \quad |\{t \in \mathsf{Trees}_A \mid \text{height}(t) \leq |\mathcal{B}|\}| + 1.$$

Proving that $N \leq \exp(|\mathcal{R}| + \text{height}(t_0))$ is standard. We make a case distinction if $L(\mathcal{B}) = \mathsf{post}^*_\Lambda(t_0)$ contains a tree of height strictly greater than $|\mathcal{B}|$ or not. On the one hand, assume that $L(\mathcal{B})$ does not contain a tree of height strictly greater than $|\mathcal{B}|$. This implies $|L(\mathcal{B})| < N$. Since $t_0 \xrightarrow{\Lambda}_N$, by the pigeonhole principle, there is some $t \in \mathsf{Trees}_A$ with $t_0 \xrightarrow{\Lambda}_* t \xrightarrow{\Lambda}_+ t$. This implies $t_0 \xrightarrow{\Lambda}_n$ for infinitely $n \in \mathbb{N}$. On the other hand assume that $L(\mathcal{B})$ contains at least one tree t with height$(t) > |\mathcal{B}|$. By reasoning in the same ways as for the Pumping Lemma for regular tree languages, we can pump some context of t arbitrarily often and hereby obtain trees of arbitrarily large height. Since the pumped trees can become arbitrarily large and are all reachable from t_0, the path lengths from t_0 to these trees can become arbitrarily long too, thus implying $t_0 \xrightarrow{\Lambda}_n$ for infinitely many $n \in \mathbb{N}$. $\square$

The separating PA: For the following proof we work on terms modulo the equivalence relation $\equiv$. Consider the PA $\mathcal{P} = (\Sigma, \mathbb{A}, \Delta)$ with $\Sigma = \{A, B, C, D\}$, $\mathbb{A} = \{a, b, c, d\}$ and where Δ consists of the following rewrite rules:

$$A \mapsto_a 0 \qquad B \mapsto_b 0 \qquad C \mapsto_c 0 \qquad D \mapsto_d 0 \qquad A \mapsto_a A \| B \| C$$

For the rest of this section, we wish to prove that the process $\alpha = A.D$ is *not* strongly bisimilar to any pointed GTRS. Before we start with the proof, let us explain the behavior of the process $\alpha = A.D$. From α we can reach via the sequence a^i the process $\alpha_i = (A\|B^i\|C^i).D$ by applying the fifth of the above rewrite rules exactly i times. The application of this rule can thus be seen to make the process bigger. However from the resulting process α_i we can switch with via an a-transition to the process $(B^i\|C^i).D$ from which we can only reach the process D by executing some sequence of actions from $\{w \in \{b, c\}^* : |w|_b = |w|_c = i\}$ (and thus applying the third and fourth of the above-mentioned rules). Having reached D, we can only execute $D \mapsto_d 0$ and hence end in a dead-end. We remark that $\alpha_i \not\sim \alpha_j$ in case $i \neq j$.

So for the sake of contradiction, let us assume some GTRS $\mathcal{R} = (A, \mathbb{A}, R)$ and some $t_\alpha \in \mathsf{Trees}_A$ with $t_\alpha \sim \alpha$. We remark that e.g. by [Löding 2003] it is known that the set of traces that are executable from α (the set of words w with $\alpha \xrightarrow{w}$) is recognizable by some GTRS with τ-transitions.

We call $V[x]$ a *context* if $V \in \mathsf{Trees}_A$ and $x \in D_V$ is a leaf of V. Given a tree $t \in \mathsf{Trees}_A$ and a context $V[x]$, we write $V[t]$ for $V[x/t]$. We define $V^n[t]$ inductively as follows: $V^0[t] = t$ and $V^n = V[V^{n-1}[t]]$ for each $n > 0$.

Let us consider $\mathsf{post}^*_{\{a\}}(t_\alpha)$. First, let $\mathcal{A}$ be some NTA with $L(\mathcal{A}) = \{t_\alpha\}$. A folklore result states that there is some NTA $\mathcal{B}$ with $L(\mathcal{B}) = \mathsf{post}^*_{\{a\}}(L(\mathcal{A})) = \mathsf{post}^*_{\{a\}}(t_\alpha)$, e.g. see [Brainerd 1969; Löding 2003]. Note that $L(\mathcal{B})$ is infinite since α can reach the set $\{\alpha_i \mid i \geq 0\}$ of pairwise non-bisimilar states and we have $t_\alpha \sim \alpha$ by assumption. By applying the Pumping Lemma for regular tree languages (cf. [Comon et al. 2007]), there is some tree $t_\mathcal{B} \in \mathsf{Trees}_A$ and there are contexts $U[x], V[y] \in \mathsf{Trees}_A$ such that

(i) $U[V[t_\mathcal{B}]] \in L(\mathcal{B})$,
(ii) height$(U[V[t_\mathcal{B}]]) \leq 2 \cdot |\mathcal{B}|$,
(iii) height$(V[t_\mathcal{B}]) \leq |\mathcal{B}|$,
(iv) $y \neq \varepsilon$ (in particular V is not a singleton tree), and
(v) $U[V^n[t_\mathcal{B}]] \in L(\mathcal{B})$ for each $n \geq 0$.

The tree $U[V[t_\mathcal{B}]]$ is displayed in Figure 2. We define the tree $T^n = U[V^n[t_\mathcal{B}]]$ for each $n \geq 0$.

The following two constants γ and ℓ that only depend on $|\mathcal{R}|$ and will play an important rule in proving that α cannot be bisimilar to t_α. Recall that $h_\mathcal{R}$ denotes the largest height of a tree that appears on the left-hand side or right-hand side of some rule in $\mathcal{R}$.

Definition 5.2. We define $\ell = 2^{|\{t \in \mathsf{Trees}_A | \text{height}(t) \leq h_\mathcal{R}\}|}$ to denote the number of different subsets of trees in Trees_A of height at most $h_\mathcal{R}$ and we define $\gamma = (\ell + 1) \cdot h_\mathcal{R}$.

We can think of ℓ and γ of as large pumping constants that only depend on $|\mathcal{R}|$. But recall that $\mathcal{R}$ is a GTRS whose tree t_α is bisimilir to the process α of the PA $\mathcal{P}$: Hence, $\mathcal{R}$ implicitly depends on our PA $\mathcal{P}$ and its process α. The following lemma states from the subtree $V^\gamma[t_\mathcal{B}]$ one can only execute constantly many b's only or c's only. In fact it is a simple consequence of Lemma 5.1.

LEMMA 5.3. *There is some constant $J = J(\mathcal{R}, t_\alpha)$ such that for each $\sigma \in \{b, c\}$ we have that if $V^\gamma[t_\mathcal{B}] \xrightarrow{\sigma}_n t$, then $n \leq J$.*

PROOF. Follows immediately from Lemma 5.1 by setting $t_0 = V^\gamma[t_\mathcal{B}]$ and by observing that we cannot have $V^\gamma[t_\mathcal{B}] \xrightarrow{\sigma}_n t$ for infinitely many $n \in \mathbb{N}$. □

The following lemma will be central for our separation result. It states that if $N \in \mathbb{N}$ is sufficiently large one can never shrink the subtree $V^\gamma[t_\mathcal{B}]$ of T^N to some tree of height at most $h_\mathcal{R}$ by only executing b's or only executing c's. Although its proof is technical and therefore postponed to the next section, we give some intuition of its proof already here. Assume some extremely large value N such that the tree $V^\gamma[t_\mathcal{B}]$ is a subtree of the tree $T^N = U[V^N[t_\mathcal{B}]]$. Recall that t_α can reach T^N by executing a's only. For the sake of contradiction, let us assume that the tree $V^\gamma[t_\mathcal{B}]$ could reach a small tree t of height at most $h_\mathcal{R}$ by executing only b's, say. By applying some pumping arguments, one can show that in fact $V^\gamma[t_\mathcal{B}]$ can also reach a very special small tree t_b of height at most $h_\mathcal{R}$ by executing b's only: $V^\gamma[t_\mathcal{B}]$ can reach t_b by b's only, but so can $V^i[t_\mathcal{B}]$ for all $i \in I$, where I is some ultimately periodic set. But this means that there exist two different $M, N \in \mathbb{N}$ such that $t_\alpha \xrightarrow{a}_* T^M$, $t_\alpha \xrightarrow{a}_* T^N$ and $T^M \not\sim T^N$ (for instance one requires many more a's to reach T^N from t_α than the a's required to reach T^M from t_α) *but both T^M and T^N* can reach $U[t_b]$ (and thus reach some identical bisimulation equivalence class!) by executing b's only. This is clearly a contradiction to the behavior of α since the number of c's that T^M and T^N can execute eventually must definitely differ.

LEMMA 5.4. *If $\sigma \in \{b, c\}$ and $V^\gamma[t_\mathcal{B}] \xrightarrow{\sigma}_* t$ for some tree $t \in \mathsf{Trees}_A$, then height$(t) > h_\mathcal{R}$.*

PROOF. The proof is subject of Section 5.2. □

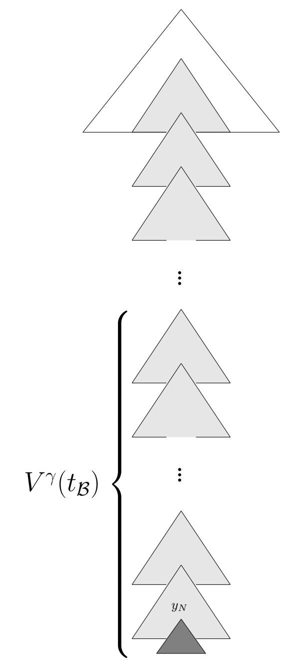

Fig. 3. The tree T^N.

We cleary emphasize that Lemma 5.4 makes a statement about the concrete tree $V^\gamma[t_{\mathcal{B}}]$ (which is a subtree of every tree T^N for every $N \geq \gamma$) and thus crucially rely on our initial assumption that t_α is bisimilar to α.

We can now prove the main result of this section by essentially combining Lemma 5.4 and Lemma 5.3.

THEOREM 5.5. PA $\not\leq_\sim$ GTRS.

PROOF. Before we give a simple winning strategy for Attacker that will contradict $t_\alpha \sim \alpha$ we need a definition. We assume N to be sufficiently large for the following arguments to work and let y_N denote the unique node of T^N where the subtree $t_{\mathcal{B}}$ is rooted at (see also Figure 3). We call a node $z \in D_{T^N}$ of T^N *off-path* if it is not an ancestor of y_N, thus if $z \not\preceq y_N$.

First Attacker plays $t_\alpha \xrightarrow{a}_{N'} T^N$ for some suitable N'. We remark since N is chosen sufficiently large, it follows that N' in turn is also sufficiently large for the following arguments to work. It has to hold for some $s \in \{0, 1\}$:

$$T^N \quad \sim \quad \left(A^{1-s} \| \underbrace{B \| B \cdots \| B}_{N'-s} \| \underbrace{C \| C \cdots \| C}_{N'-s} \right).D \qquad (\bigstar)$$

We only treat the case $s = 1$ (the case $s = 0$ can be proven analogously). Recall that γ is a constant that depends only on $|\mathcal{R}| + |t_\alpha|$. On the one hand we cannot rewrite the subtree $V^\gamma[t_\mathcal{B}]$ of T^N to any tree of height at most $h_\mathcal{R}$ by executing b-labeled transitions only by Lemma 5.4. On the other hand we cannot execute more than J many b-labeled transitions in the subtree $V^\gamma[t_\mathcal{B}]$, where J is the constant of Lemma 5.3. Due to (★) and hence $T^N \xrightarrow{b}_{N'-1}$, Attacker can play at least $N' - 1 - J$ many b-labeled transitions outside the subtree $V^\gamma[t_\mathcal{B}]$. We recall that $N' - J$ can be arbitrarily large since J is a constant that depends only on $|\mathcal{R}| + |t_\alpha|$. By definition of T^N all of these $N' - 1 - J$ many b-labeled transitions can be executed by rewriting subtrees initially rooted at off-path nodes of T^N *outside* the subtree $V^\gamma[t_\mathcal{B}]$.

Recall that the size of the pumping context $V[x]$ and thus of any of its subtree is constantly bounded (i.e. depends only on $|\mathcal{R}| + |t_\alpha|$ but not on N). Attacker now has the following winning strategy. First he continues from T^N by executing $N'-1-J$ many b-labeled transitions by rewriting subtrees rooted at off-path nodes of T^N *outside* $V^\gamma[t_\mathcal{B}]$. Note that after playing these b-labeled transitions the sizes of the subtrees rooted at off-path nodes of T^N outside $V^\gamma[t_\mathcal{B}]$ is still constantly bounded (depending only on $|\mathcal{R}| + |t_\alpha|$ but not on N): this can be proven along the arguments as in the proof of Lemma 5.1 by showing that otherwise infinitely many b-labeled transitions can be executed, clearly contradicting (★).

From the resulting tree Attacker has the possibility to play $N' - 1 - J$ many c-labeled transitions by rewriting trees that are rooted outside the subtree $V^\gamma[t_\mathcal{B}]$ again due to the same reasoning as before. Let us call the resulting tree T'. Again, we have that any subtree rooted at any off-path node outside $V^\gamma[t_\mathcal{B}]$ has constant size. We remark that along his path from T^N to T' Attacker has so far not yet applied any rewrite rule at a node inside the subtree $V^\gamma[t_\mathcal{B}]$. We note that $T' \xrightarrow{b^J c^J d}$, i.e. from T' the sequence $b^J c^J$ is executable thus reaching a tree where a d-labeled rule is executable. But this clearly implies that $T^N \xrightarrow{wd}$ for some $w \in \{b, c\}^*$ where $|w|$ is constantly bounded, clearly contradicting (★). □

5.2. Proof of Lemma 5.4

Recall the context $V[y]$, where $y \neq \varepsilon$ and let $t_\mathcal{B}$ be the tree from the the previous section. The following lemma states that if $V^{i+1}[t_\mathcal{B}]$ can reach a tree of height at most $h_\mathcal{R}$ by executing only σ-labeled transitions (where $\sigma \in \mathbb{A}$), then $V^i[t_\mathcal{B}]$ can also reach a tree of height at most $h_\mathcal{R}$ by executing σ-labeled transitions. In fact the lemma holds even for any non-singleton contexts and arbitary trees to be plugged in, but we just need to state it for the context $V[y]$ and the tree $t_\mathcal{B}$.

LEMMA 5.6. *Fix some symbol $\sigma \in \mathbb{A}$ and some $i \geq 0$. If $V^{i+1}[t_\mathcal{B}] \xrightarrow{\sigma}_* t$ for some $t \in \mathsf{Trees}_A$ with height$(t) \leq h_\mathcal{R}$, then $V^i[t_\mathcal{B}] \xrightarrow{\sigma}_* t'$ for some $t' \in \mathsf{Trees}_A$ with height$(t') \leq h_\mathcal{R}$.*

PROOF. Assume $V^{i+1}[t_\mathcal{B}] \xrightarrow{\sigma}_* t$ for some $t \in \mathsf{Trees}_A$ with height$(t) \leq h_\mathcal{R}$. Let us fix a sequence of σ-labeled rewrite rules $r_1 \cdots r_\ell \in R^\ell$ that witnesses $V^{i+1}[t_\mathcal{B}] \xrightarrow{\sigma}_* t$. To each rule r_i we can assign a position where r_i is applied. One easily verifies that the scattered subsequence of $r_1 \cdots r_\ell$ that one obtains by keeping only those r_i that are applied at positions u_i with $y \preceq u_i$ witnesses that $V^i[t_\mathcal{B}] \xrightarrow{\sigma}_* t'$ for some tree $t' \in \mathsf{Trees}_A$ with height$(t') \leq h_\mathcal{R}$. □

The following lemma states that the there is an arithmetic progression on pumping exponents i for reaching small trees from $V^i[t_\mathcal{B}]$. More precisely, it states that if the tree $V^\gamma[t_\mathcal{B}]$ can reach some tree of height at most $h_\mathcal{R}$ by only executing σ-labeled transitions, then there is already some tree t_σ of height at most $h_\mathcal{R}$ such that $V^{\theta_\sigma + i\cdot\delta_\sigma}[t_\mathcal{B}] \xrightarrow{\sigma}_* t_\sigma$ for each $i \geq 0$, where $\theta_\sigma \geq 1$ is some offset and $\delta_\sigma \geq 1$ is some period for each $\sigma \in \mathbb{A}$.

LEMMA 5.7. *For each $\sigma \in \mathbb{A}$ there exist $\theta_\sigma, \delta_\sigma \geq 1$ such that if $V^\gamma[t_\mathcal{B}] \xrightarrow{\sigma}_* t$ for some $t \in \mathsf{Trees}_A$ with height$(t) \leq h_\mathcal{R}$, then $V^{\theta_\sigma + i\cdot\delta_\sigma}[t_\mathcal{B}] \xrightarrow{\sigma}_* t_\sigma$ for all $i \geq 0$ for some $t_\sigma \in \mathsf{Trees}_A$ with height$(t_\sigma) \leq h_\mathcal{R}$.*

PROOF. Let us define the set

$$\mathsf{SMALL}_j \;=\; \{t \in \mathsf{Trees}_A \mid \text{height}(t) \le h_{\mathcal{R}} \text{ and } V^j[t_{\mathcal{B}}] \xrightarrow{\sigma}_* t\}$$

for each $j \geq 0$. Note that if $\mathsf{SMALL}_{j+1} \neq \emptyset$, then $\mathsf{SMALL}_j \neq \emptyset$ for each $j \geq 0$ by Lemma 5.6. Let us first prove the following claim.

Claim. Assume $d \geq h_{\mathcal{R}}$ and $\mathsf{SMALL}_j = \mathsf{SMALL}_{j+d}$. Then $\mathsf{SMALL}_j = \mathsf{SMALL}_{j+i\cdot d}$ for all $i \geq 0$.

Proof of Claim. Let $d \geq h_{\mathcal{R}}$. We prove $\mathsf{SMALL}_j = \mathsf{SMALL}_{j+d}$ implies that $\mathsf{SMALL}_{j+(i-1)\cdot d} = \mathsf{SMALL}_{j+i\cdot d}$ for all $i \geq 1$ by induction on i. The induction base, i.e. when $i = 1$, holds by assumption.

For the induction step, let $i > 1$. We have to prove $\mathsf{SMALL}_{j+(i-1)\cdot d} = \mathsf{SMALL}_{j+i\cdot d}$. For the inclusion from left to right, assume $t \in \mathsf{SMALL}_{j+(i-1)\cdot d}$. Thus, we have

$$V^{j+(i-1)\cdot d}[t_{\mathcal{B}}] \xrightarrow{\sigma}_* t.$$

Since $y \neq \varepsilon$ and $d \geq h_{\mathcal{R}}$ there is some $t' \in \mathsf{Trees}_A$ with height$(t') \leq h_{\mathcal{R}}$ such that

$$V^{j+(i-1)\cdot d}[t_{\mathcal{B}}] \xrightarrow{\sigma}_* V^d[t'] \xrightarrow{\sigma}_* t \quad \text{and} \quad V^{j+(i-2)\cdot d}[t_{\mathcal{B}}] \xrightarrow{\sigma}_* t'.$$

By induction hypothesis, we have

$$V^{j+(i-1)\cdot d}[t_{\mathcal{B}}] \xrightarrow{\sigma}_* t'.$$

Hence

$$V^{j+i\cdot d}[t_{\mathcal{B}}] = V^d[V^{j+(i-1)\cdot d}[t_{\mathcal{B}}]] \xrightarrow{\sigma}_* V^d[t'] \xrightarrow{\sigma}_* t$$

and therefore $t \in \mathsf{SMALL}_{j+i\cdot d}$.
For the inclusion from right to left, assume $t \in \mathsf{SMALL}_{j+i\cdot d}$. Thus

$$V^{j+i\cdot d}[t_{\mathcal{B}}] \xrightarrow{\sigma}_* t.$$

Since $y \neq \varepsilon$ and $d \geq h_{\mathcal{R}}$ there is some $t' \in \mathsf{Trees}_A$ with height$(t') \leq h_{\mathcal{R}}$ such that

$$V^{j+i\cdot d}[t_{\mathcal{B}}] \xrightarrow{\sigma}_* V^d[t'] \xrightarrow{\sigma}_* t \quad \text{and} \quad V^{j+(i-1)\cdot d}[t_{\mathcal{B}}] \xrightarrow{\sigma}_* t'.$$

By induction hypothesis we have

$$V^{j+(i-2)\cdot d}[t_{\mathcal{B}}] \xrightarrow{\sigma}_* t'.$$

Hence

$$V^{j+(i-1)\cdot d}[t_{\mathcal{B}}] = V^d[V^{j+(i-2)\cdot d}[t_{\mathcal{B}}]] \xrightarrow{\sigma}_* V^d[t'] \xrightarrow{\sigma}_* t$$

and hence $t \in \mathsf{SMALL}_{j+(i-1)\cdot d}$.

This concludes the proof of the claim.

It remains to find, in case $V^{\gamma}[t_{\mathcal{B}}] \xrightarrow{\sigma}_* t$ for some $t \in \mathsf{Trees}_A$ with height$(t) \leq h_{\mathcal{R}}$, some tree $t_\sigma \in \mathsf{Trees}_A$ with height$(t_\sigma) \leq h_{\mathcal{R}}$, some $\theta_\sigma \geq 1$ and some $\delta_\sigma \geq 1$ with $V^{\theta_\sigma + i\cdot\delta_\sigma}[t_{\mathcal{B}}] \xrightarrow{\sigma}_* t_\sigma$ for all $i \geq 0$.

Recall that $\gamma = (\ell + 1) \cdot h_{\mathcal{R}}$, where ℓ denotes the number of all possible sets of trees in Trees_A of height at most $h_{\mathcal{R}}$. Let us assume $V^{\gamma}[t_{\mathcal{B}}] \xrightarrow{\sigma}_* t$ for some $t \in \mathsf{Trees}_A$ with height$(t) \leq h_{\mathcal{R}}$. Since this implies $\mathsf{SMALL}_\gamma \neq \emptyset$, it follows $\mathsf{SMALL}_j \neq \emptyset$ for each $j \in \{0, \ldots, \gamma\}$ by Lemma 5.6. Note that $|\{\mathsf{SMALL}_k \mid 0 \leq k \leq \gamma\}| \leq \ell$. Hence, among the non-empty sets $\mathsf{SMALL}_0, \mathsf{SMALL}_{h_{\mathcal{R}}}, \mathsf{SMALL}_{2\cdot h_{\mathcal{R}}}, \ldots, \mathsf{SMALL}_\gamma$ there are, by the pigeonhole principle, two sets $\mathsf{SMALL}_{\theta_\sigma}$ and $\mathsf{SMALL}_{\theta_\sigma+\delta_\sigma}$ such that $\mathsf{SMALL}_{\theta_\sigma} = \mathsf{SMALL}_{\theta_\sigma+\delta_\sigma}$, where $\delta_\sigma \geq h_{\mathcal{R}}$. The above claim implies that $\mathsf{SMALL}_{\theta_\sigma} = \mathsf{SMALL}_{\theta_\sigma+i\cdot\delta_\sigma}$ for each $i \geq 0$. In particular, there exists some $t_\sigma \in \mathsf{Trees}_A$ with height$(t_\sigma) \leq h_{\mathcal{R}}$ with

$$V^{\theta_\sigma+i\cdot\delta_\sigma}[t_{\mathcal{B}}] \xrightarrow{\sigma}_* t_\sigma \qquad \text{for each } i \geq 0.$$

□

We note that due to $t_\alpha \sim \alpha$ and the definition of PA $\mathcal{P}$ we have that for every $t \in \mathsf{post}^*_{\{a\}}(t_\alpha)$ there is some unique $k \in \mathbb{N}$ with $t_\alpha \xrightarrow{a}_k t$. Thus, for each tree $t \in \mathsf{post}^*_{\{a\}}(t_\alpha)$ we define $k(t)$ to be the *unique* $k \in \mathbb{N}$ with $t_\alpha \xrightarrow{a}_k t$. The following lemma is essentially a consequence of the definition of our PA $\mathcal{P}$.

LEMMA 5.8. *Assume* $\sigma \in \{b, c\}$ *and assume some tree* $t \in \mathsf{Trees}_A$ *such that* $T^n \xrightarrow{\sigma}_* t$ *and* $T^m \xrightarrow{\sigma}_* t$*. Then* $k(T^n) = k(T^m)$*.*

PROOF. Let σ either be the action label b or c. For each $i \geq 1$ let us introduce the following two terms $\alpha_i = (A \| B^i \| C^i).D$ and $\beta_i = (B^{i-1} \| C^{i-1}).D$ of $\mathcal{P}$. Since $T^n, T^m \in \mathsf{post}^*_{\{a\}}(t_\alpha)$, we must have by definition of the two rewrite rules $A \mapsto_a A \| B \| C$ and $A \mapsto_a 0$ in Δ

— $T^n \sim \alpha_{k(T^n)}$ or $T^n \sim \beta_{k(T^n)}$ and
— $T^m \sim \alpha_{k(T^m)}$ or $T^m \sim \beta_{k(T^m)}$.

Let us assume that $k(T^m) \neq k(T^n)$. We will be done once we have shown that there does not exist any tree $t \in \mathsf{Trees}_A$ such that $T^m \xrightarrow{\sigma}_* t$ and $T^n \xrightarrow{\sigma}_* t$. By the assumption $k(T^n) \neq k(T^m)$, by inspecting the only b-labeled or c-labeled rules $B \mapsto_b 0$ and $C \mapsto_c 0$ and a simple case distinction of the four cases

— $T^n \sim \alpha_{k(T^n)}$ and $T^m \sim \alpha_{k(T^m)}$, or
— $T^n \sim \alpha_{k(T^n)}$ and $T^m \sim \beta_{k(T^m)}$, or
— $T^n \sim \beta_{k(T^n)}$ and $T^m \sim \alpha_{k(T^m)}$, or
— $T^n \sim \beta_{k(T^n)}$ and $T^m \sim \beta_{k(T^m)}$

one can easily conclude that there do not exist any trees $t_n, t_m \in \mathsf{Trees}_A$ such that $T^m \xrightarrow{\sigma}_* t_m$ and $T^n \xrightarrow{\sigma}_* t_n$ with $t_n \sim t_m$, simply because both cannot execute the same number of a's, b's and c's: either arbitarily many a's and no a's on the one hand, or a different number of b's/c's on the other hand. In particular, there does not exist any tree $t \in \mathsf{Trees}_A$ such that $T^n \xrightarrow{\sigma}_* t$ and $T^m \xrightarrow{\sigma}_* t$. □

The following straightforwardly follows from the fact that $\mathcal{R}$ is a GTRS and that $\mathsf{post}^*_{\{a\}}(t_\alpha)$ is infinite.

LEMMA 5.9. $\{k(T^n) \mid n \in \mathbb{N}\}$ *is an infinite set.*

PROOF. It is easy to see that for every $m \in \mathbb{N}$ there is an $n \in \mathbb{N}$ such that $k(T^n) > m$. □

Let us finally prove Lemma 5.4.

PROOF OF LEMMA 5.4. Let us fix some $\sigma \in \{b, c\}$ and assume by contradiction that $V^\gamma[t_\mathcal{B}] \xrightarrow{\sigma}_* t$ holds for some $t \in \mathsf{Trees}_A$ with $\mathsf{height}(t) \leq h_\mathcal{R}$. Then there is some tree t_σ with $\mathsf{height}(t_\sigma) \leq h_\mathcal{R}$ and $V^{\theta_\sigma + i \cdot \delta_\sigma}[t_\mathcal{B}] \xrightarrow{\sigma}_* t_\sigma$ for each $i \geq 0$ by Lemma 5.7. Let us choose a sufficiently large $N \in \mathbb{N}$ for the following arguments to work. Since N is sufficiently large there exists a sufficiently large $M \in \mathbb{N}$ such that $M < N$, $k(T^M) < k(T^N)$ and $M \equiv N \bmod \delta_\sigma$.

Since we have $T^M = U[V^M[t_\mathcal{B}]] \xrightarrow{\sigma}_* t'$, where $t' = U[V^{M-\theta_\sigma}[t_\sigma]]$. Due to $M \equiv N \bmod \delta_\sigma$ and $M < N$ we also have $T^N = U[V^N[t_\mathcal{B}]] \xrightarrow{\sigma}_* t'$. But then $k(T^M) < k(T^N)$ contradicts Lemma 5.8. □

5.3. GTRS $\not\leq_\approx$ PAD

By Theorem 4.11 it suffices to prove that there is some RGTRS that is not weakly bisimilar to any PAD.

Consider the RGTRS $\mathcal{R} = (A, \mathbb{A}, R)$ with $A_0 = \{X_0, Y_0, Z_0\}$, $A_1 = \{X_1, Y_1\}$, $A_2 = \{\bullet\}$, and $\mathbb{A} = \{a, b, c, d, e, f\}$. First, we add to R the following rewrite rules:

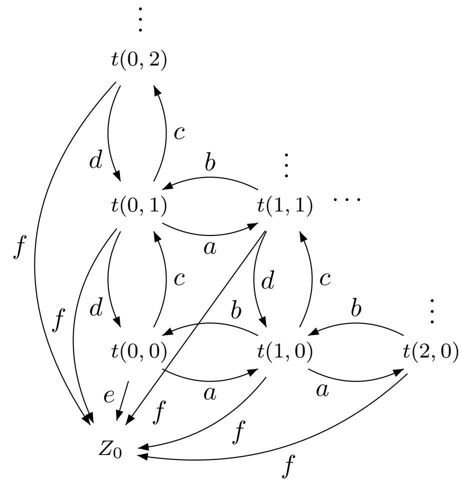

Fig. 4. The transition system $\mathcal{T}(\mathcal{R})$.

— $X_0 \overset{a}{\hookrightarrow} X_1(X_0)$,
— $X_1(X_0) \overset{b}{\hookrightarrow} X_0$,
— $Y_0 \overset{c}{\hookrightarrow} Y_1(Y_0)$,
— $Y_1(Y_0) \overset{d}{\hookrightarrow} Y_0$, and
— $\bullet(X_0, Y_0) \overset{e}{\hookrightarrow} Z_0$.

We note that so far all rewrite rules are standard ground tree rewrite rules. Also note that the singleton tree Z_0 is a dead-end. First, it is easy to see that for every tree in $t \in \mathsf{Trees}_A$ that is reachable from $\bullet(X_0, Y_0)$ we have $t = Z_0$ or t is of the form $t = \bullet(t_X, t_Y)$, where $t_X = X_1^m[X_0]$ and $t_Y = Y_1^n[Y_0]$ for some $m, n \geq 0$. In the latter case we denote t by $t(m, n)$. Finally, we add to R the regular ground tree rewrite rule $\{t(m, n) \mid n \geq 1 \text{ or } m \geq 1\} \overset{f}{\hookrightarrow} Z_0$. The transition system $\mathcal{T}(\mathcal{R})$ is depicted in Figure 4.

It is easy to see that the set of sequences executable from $t(0, 0)$ is not a context-free language.

We call a term $\alpha \in \mathbb{G}(\Sigma)$ of some PAD *inactive* if $\alpha \overset{\sigma}{\not\Longrightarrow}$ for all $\sigma \in \mathbb{A}$. We note that $\alpha \overset{\tau}{\Longrightarrow}$ might be possible even though α is inactive.

LEMMA 5.10. *Assume some* PAD *process α with $\alpha \approx t(m, n)$ for some $m, n \in \mathbb{N}$ and α contains an enabled subterm $\beta_1 \| \beta_2$. Then β_1 or β_2 is inactive.*

PROOF. Assume that α contains an enabled subterm $\beta_1 \| \beta_2$. Thus α can be written as

$$\alpha = (\beta_1 \| \beta_2 \| \cdots \beta_k).\gamma$$

for some $\beta_3, \ldots, \beta_k \in \mathbb{G}(\Sigma)$, where $k \geq 2$. Moreover assume by contradiction that neither β_1 nor β_2 is inactive, hence $\beta_1 \overset{\sigma_1}{\Longrightarrow}$ and $\beta_2 \overset{\sigma_2}{\Longrightarrow}$ for some $\sigma_1, \sigma_2 \in \mathbb{A}$.

First, note that neither $\beta_1 \overset{g}{\Longrightarrow}$ nor $\beta_2 \overset{g}{\Longrightarrow}$ holds (in particular $\sigma_1 \neq g$ and $\sigma_2 \neq g$) for any $g \in \{e,f\}$ since this would imply $\alpha \overset{g\sigma_1}{\Longrightarrow}$ or $\alpha \overset{g\sigma_2}{\Longrightarrow}$, clearly contradicting $\alpha \approx t(m,n)$. Thus $\sigma_1, \sigma_2 \in \{a,b,c,d\}$. But since $\alpha \overset{g}{\Longrightarrow}$ for some $g \in \{e,f\}$ there has to be some $j \in \{1,2\}$ with $\beta_j \overset{\tau}{\Longrightarrow} 0$. We fix this $j \in \{1,2\}$ for the rest of the proof.

Note that whenever $\beta_j \overset{\sigma}{\Longrightarrow} \beta_j'$, where $\sigma \in \mathbb{A}$, then it must follow $\beta_j' \overset{\tau}{\not\Longrightarrow} 0$ since otherwise $\alpha \overset{\tau}{\Longrightarrow} \alpha'$ and $\alpha \overset{\sigma}{\Longrightarrow} \alpha'$ for some α', clearly contradicting $\alpha \approx t(m,n)$. Wrapping up, we have

$$\beta_j \overset{\tau}{\Longrightarrow} 0 \quad \text{and} \quad \beta_j \overset{\sigma_j}{\Longrightarrow} \beta_j' \overset{\tau}{\not\Longrightarrow} 0$$

for some β_j'. Consider the move $\alpha \overset{\sigma_j}{\Longrightarrow} \alpha'$ by Attacker where

$$\alpha' \;=\; (\beta_j' \| \beta_{3-j} \| \beta_3 \cdots \| \beta_k).\gamma.$$

Since $\sigma_j \in \{a,b,c,d\}$ and $\alpha \approx t(m,n)$ we must have $\alpha' \overset{g'}{\Longrightarrow}$ for some $g' \in \{e,f\}$. We have $\beta_{3-j} \overset{g'}{\not\Longrightarrow}$ since otherwise $\alpha \overset{g'\sigma_j}{\Longrightarrow}$. Similarly we must have $\beta_i \overset{g'}{\not\Longrightarrow}$ for all $i \in \{3,\ldots,k\}$. Also, we must have $\beta_j' \overset{g'}{\not\Longrightarrow}$ since otherwise $\alpha \overset{\sigma_j g' \sigma_{3-j}}{\Longrightarrow}$, clearly a contradiction. Due to $\beta_j' \overset{\tau}{\not\Longrightarrow} 0$ we can thus not have $\alpha' \overset{g'}{\Longrightarrow}$ for any $g' \in \{e,f\}$, which is a contradiction. □

The definition of the GTRS $\mathcal{R}$ and Lemma 5.10 allows us to prove that the tree $t(0,0)$ in the transition system $\mathcal{T}(\mathcal{R})$ is not weakly bisimilar to any PAD.

THEOREM 5.11. *GTRS $\not\leq_{\approx}$ PAD.*

PROOF. We claim that there is no PAD that is weakly bisimilar to $t(0,0) = \bullet(X_0, Y_0)$. Let us assume by contradiction that for some PAD $\mathcal{P} = (\Sigma, \mathcal{A}_\tau, \Delta)$ and for some term $\alpha_0 \in \mathbb{G}(\Sigma)$ we have $\alpha_0 \approx t(0,0)$. We can assume without loss of generality that $\alpha_0 \in \mathbb{1}(\Sigma)$ is a process constant. We will use Lemma 5.10 to show that there is some pushdown system $\mathcal{P}' = (\Sigma', \mathbb{A}_\tau, \Delta')$ such that α_0 in $\mathcal{T}(\mathcal{P})$ is weakly bisimilar to α_0 in $\mathcal{T}(\mathcal{P}')$, clearly contradicting that the set of traces executable from t_α is not context-free. Define

$$\Gamma \;=\; \{t \mid \exists u \overset{\sigma}{\hookrightarrow} u' \in \Delta \text{ and } t \text{ is a subterm of } u'\}$$

and put $\Sigma' = \Sigma \uplus \{X_t \mid t \in \Gamma \text{ and } t \text{ is inactive}\}$. For each inactive term t that appears on the right-hand side of any rule in Δ we define

$$\chi(t) = \begin{cases} 0 & \text{if } t \overset{\tau}{\Longrightarrow} 0 \\ X_t & \text{otherwise.} \end{cases}$$

Let us define the mapping $\varphi : \Gamma \to (\Sigma')^*$ inductively as follows:

$$\varphi(t) \quad = \quad \begin{cases} 0 & \text{if } t = 0 \\ U & \text{if } t = U \in \mathbb{1}(\Sigma) \\ \varphi(t_1).\varphi(t_2) & \text{if } t = t_1.t_2 \\ \chi(t_1).\chi(t_2) & \text{if } t = t_1 \| t_2 \text{ and both } t_1 \text{ and } t_2 \text{ are inactive} \\ \varphi(t_1).\chi(t_2) & \text{if } t = t_1 \| t_2 \text{ and } t_1 \text{ is active and } t_2 \text{ is inactive} \\ \varphi(t_2).\chi(t_1) & \text{if } t = t_1 \| t_2 \text{ and } t_1 \text{ is inactive and } t_2 \text{ is active} \end{cases}$$

We define the set of rewrite rules as follows:

$$\Delta' \quad = \quad \{w \overset{\ell}{\hookrightarrow} \varphi(t) \mid w \overset{\ell}{\hookrightarrow} t \in \Delta\} \cup \{X_t \overset{\tau}{\hookrightarrow} \varphi(t) \mid t \in \Gamma\}$$

By Lemma 5.10 we have that α_0 in $\mathcal{T}(\mathcal{P})$ is weakly bisimilar to α_0 in $\mathcal{T}(\mathcal{P}')$. □

5.4. PDS $\not\leq_\approx$ PAN **and** PN $\not\leq_\approx$ GTRS

THEOREM 5.12. PDS $\not\leq_\approx$ PAN.

The proof idea is an adaption of an idea from [Mayr 2000] separating PAN from PDS with respect to strong bisimulation, but is technically more involved. Before we can prove Theorem 5.12, we first need some preparations.

Consider a pushdown process that behaves as follows: First, it executes a sequence of actions $w = \{a, b\}^*$ and then executes either of the following: (1) The action c, then the reverse of w and finally the action e. (2) The action d, then the reverse of w and finally the action f.

Definition 5.13. The separating PDS is $\mathcal{P}_{\text{sep}} = (\Sigma_{\text{sep}}, \mathbb{A}, \Delta_{\text{sep}})$, where

$$\Sigma_{\text{sep}} = \{A, B, U, V, W, X\}, \qquad \mathbb{A} = \{a, b, c, d, e, f\}$$

and where Δ_{sep} is given as follows:

$U.X \mapsto_a U.A.X$	$U.A \mapsto_a U.A.A$	$U.A \mapsto_b U.B.A$
$U.X \mapsto_b U.B.X$	$U.B \mapsto_b U.B.B$	$U.B \mapsto_a U.A.B$
$U.X \mapsto_c V.X$	$U.A \mapsto_c V.A$	$U.B \mapsto_c V.B$
$U.X \mapsto_d W.X$	$U.A \mapsto_d W.A$	$U.B \mapsto_d W.B$
$V.A \mapsto_a V$	$V.B \mapsto_b V$	$V.X \mapsto_e V$
$W.A \mapsto_a W$	$W.B \mapsto_b W$	$W.X \mapsto_f W$

Analogously as in the previous section we work on terms modulo the equivalence relation $\equiv$. Let $\mathcal{P} = (\Sigma, \mathbb{A}_\tau, \Delta)$ be some PAN that allows τ-transitions. Let t be a term of $\mathcal{T}(\mathcal{P})$. A *run* from t is either an infinite sequence $t_1 \stackrel{a_1}{\Longrightarrow} t_2 \stackrel{a_2}{\Longrightarrow} t_3 \cdots$ with $t_1 = t$ and $a_i \in \mathbb{A}$ for each $i \geq 1$ or a finite sequence $t_1 \stackrel{a_1}{\Longrightarrow} t_2 \stackrel{a_2}{\Longrightarrow} \cdots \stackrel{a_n}{\Longrightarrow} t_n$ with $t_1 = t$ and $a_i \in \mathbb{A}$ such that there is no t' and no $a \in \mathbb{A}$ with $t_n \stackrel{a}{\Longrightarrow} t'$.

For each sequence $w \in \mathbb{A}^*$ and each process t, we define the predicate $\mathsf{only}(t, w)$ if and only if both of the following conditions are satisfied:

— all runs from t are finite and
— all these runs execute the sequence w.

Let us recall Dickson's Lemma.

LEMMA 5.14 (DICKSON'S LEMMA). *For every infinite sequence of vectors $M_1, M_2, \ldots$ in $\mathbb{N}^k$ there are $i < j$ such that $M_i \leq M_j$, where $\leq$ on vectors is defined componentwise.*

In the following, we assume that $\Sigma = \{X_1, \ldots, X_{|\Sigma|}\}$. For each parallel process $t \in \mathbb{P}(\Sigma)$, we define $\mathsf{Parikh}(t) \in \mathbb{N}^{|\Sigma|}$ to be the *Parikh image* of t, i.e. the i^{th} entry of $\mathsf{Parikh}(t)$ equals the number of times X_i occurs in t.

LEMMA 5.15. *Assume some* PAN $= (\Sigma, \mathbb{A}_\tau, \{\stackrel{a}{\longrightarrow} \mid a \in \mathbb{A}_\tau\})$ *and two parallel terms $t_1, t_2 \in \mathbb{P}(\Sigma)$, where $\mathsf{Parikh}(t_1) \leq \mathsf{Parikh}(t_2)$. Then $(t_1 \| t) \stackrel{w}{\Longrightarrow}$ implies $(t_2 \| t) \stackrel{w}{\Longrightarrow}$ for all $w \in \mathbb{A}^*$ and for all $t \in \mathbb{G}(\Sigma)$.*

PROOF SKETCH. The lemma can easily be proven by induction on $|w|$. □

The following lemma will be an application of Dickson's Lemma and Lemma 5.15.

LEMMA 5.16. *For every* PAN $\mathcal{P} = (\Sigma, \mathbb{A}_\tau, \Delta)$ *there is a $w \in \{a, b\}^+$ such that all parallel processes $\alpha \in \mathbb{P}(\Sigma)$ do not satisfy any of the following two conditions:*

— *Condition (P1):* $\exists \alpha_c : \alpha \stackrel{c}{\longrightarrow} \alpha_c \ \wedge\ \mathsf{only}(\alpha_c, we)$
— *Condition (P2):* $\exists \alpha_d : \alpha \stackrel{d}{\longrightarrow} \alpha_d \ \wedge\ \mathsf{only}(\alpha_d, wf)$

PROOF. Assume by contradiction that there is some PAN $\mathcal{P} = (\Sigma, \mathbb{A}_\tau, \Delta)$ such that for every $w_i = a^i b$ (where $i \geq 1$) there is some $\alpha^i \in \mathbb{P}(\Sigma)$ such that α^i satisfies condition (P1) or (P2). Recall that $\mathbb{A}_\tau = \{\tau\} \cup \{a, b, c, d, e, f\}$.

Then there must be an infinite subsequence of $\alpha^1, \alpha^2, \cdots$, where (P1) is always satisfied or an infinite subsequence of $\alpha^1, \alpha^2, \cdots$, where (P2) is always satisfied. We assume without loss of generality that there is an infinite subsequence where (P1) is always satisfied. In the following we will further inspect this infinite subsequence. Since Δ is finite, there are only finitely many different rules in Δ that are labeled with the action c. Let $(t_1 \mapsto_c t'_1), \ldots, (t_n \mapsto_c t'_n)$ be an enumeration of these rules. Recall that $t_i \in \mathbb{P}(\Sigma)$ for each $i \in \{1, \ldots, n\}$ since $\mathcal{P}$ is a PAN. Among these rules there has to be some rule $t_\ell \mapsto_c t'_\ell$ that can be applied to infinitely many α^i by which we reach the term α^i_c, respectively. Hence we obtain an infinite subsequence where only this rule is applied. We consider only this subsequence in the following. By Dickson's Lemma, there are indices $j < j'$ such that $\mathsf{Parikh}(\alpha^j) \leq \mathsf{Parikh}(\alpha^{j'})$. We can write $\alpha^j = \beta \| t_\ell$ for some $\beta, t_\ell \in \mathbb{P}(\Sigma)$ and hence $\alpha^{j'} = \beta \| \gamma \| t_\ell$ for some $\gamma \in \mathbb{P}(\Sigma)$. Moreover we have $\alpha^j_c = \beta \| t'_\ell$ and $\alpha^{j'}_c = \beta \| \gamma \| t'_\ell$ for some $t'_\ell \in \mathbb{G}(\Sigma)$. By assumption, we have $\mathsf{only}(\alpha^{j'}_c, w_{j'} e)$ and $\mathsf{only}(\alpha^j_c, w_j e)$. But since $\beta \| \gamma \in \mathbb{P}(\Sigma)$ it follows $\alpha^{j'}_c \overset{w_j e}{\Longrightarrow}$ by Lemma 5.15 , hence contradicting $\mathsf{only}(\alpha^{j'}_c, w_{j'} e)$ since $j < j'$. □

For each $\Omega \in \{A, B\}^*$, we define $w(\Omega) \in \{a, b\}^*$ to be the sequence we obtain from Ω by changing upper case letters to lower case letters.

LEMMA 5.17. *For every* PAN $\mathcal{P} = (\Sigma, \mathbb{A}_\tau, \Delta)$ *there is a sequence* $\Omega \in \{A, B\}^+$ *such that no process term* t *with respect to* $\mathcal{P}$ *is weakly bisimilar to* $U.\Omega.X$ *of* $\mathcal{P}_{sep}$.

PROOF. Assume by contradiction that there is some PAN $\mathcal{P} = (\Sigma, \mathbb{A}_\tau, \Delta)$ such that for every sequence $\Omega \in \{A, B\}^+$ there is a term $t(\Omega)$ such that $t(\Omega) \approx U.\Omega.X$.

Note that for every $\Omega \in \{A, B\}^*$ and every term $t(\Omega)$ the following two properties have to hold:

— (1) There is a term $t_c(\Omega)$ such that $t(\Omega) \overset{c}{\Longrightarrow} t_c(\Omega)$ and $t_c(\Omega) \approx V.\Omega.X$ and thus $\mathsf{only}(t_c(\Omega), w(\Omega)e)$.

— (2) There is a term $t_d(\Omega)$ such that $t(\Omega) \overset{d}{\Longrightarrow} t_d(\Omega)$ and $t_d(\Omega) \approx W.\Omega.X$ and thus $\mathsf{only}(t_d(\Omega), w(\Omega)f)$.

Let $\Omega_0 \in \{A, B\}^+$ be some sequence such that $w(\Omega_0)$ satisfies Lemma 5.16, i.e. *for all* parallel processes $\alpha \in \mathbb{P}(\Sigma)$ with respect to $\mathcal{P}$ we have

— (P1) $\nexists \alpha_c : \alpha \overset{c}{\longrightarrow} \alpha_c \wedge \mathsf{only}(\alpha_c, w(\Omega_0)e)$ and

— (P2) $\nexists \alpha_d : \alpha \overset{d}{\longrightarrow} \alpha_d \wedge \mathsf{only}(\alpha_d, w(\Omega_0)f)$.

We put $\Omega = \Omega_0$, hence we will prove $t(\Omega_0) \not\approx U.\Omega_0.X$. So assume by contradiction that $t(\Omega_0) \approx U.\Omega_0.X$ holds. In the following, if α is (an occurence of) a subterm of some term t and α' is another term, we denote by $t[\alpha/\alpha']$ the term that one obtains from t by replacing (this occurence) of α by α'. We have the following claim.

Claim: There is some term $t'(\Omega_0)$ such that the following conditions hold:

(i) $t(\Omega_0) \overset{\tau}{\Longrightarrow} t'(\Omega_0)$.

(ii) There is some maximal (with respect to the subterm order) enabled parallel subterm $\alpha \in \mathbb{P}(\Sigma)$ of $t'(\Omega_0)$ with $\alpha \overset{c}{\longrightarrow} \alpha_c$ for some $\alpha_c \in \mathbb{G}(\Sigma)$.

(iii) $\alpha \overset{d}{\Longrightarrow} \alpha_d$ for some $\alpha_d \in \mathbb{G}(\Sigma)$.

Proof of the claim. Clearly (i) and (ii) have to hold for some term $t'(\Omega_0)$ by condition (1) from above and since $\mathcal{P}$ is a PAN.

Among all possible choices for $t'(\Omega_0)$ and α we choose $t'(\Omega_0)$ in such a way that for some maximal (with respect to subterm order) parallel subterm $\alpha \in \mathbb{P}(\Sigma)$ of $t'(\Omega_0)$ we have $\alpha \overset{c}{\longrightarrow} \alpha_c$.

Observe that $t'(\Omega_0) \overset{d}{\Longrightarrow}$ and $t'(\Omega_0)[\alpha/\alpha_c] \overset{d}{\not\Longrightarrow}$ holds because $t(\Omega_0) \approx t'(\Omega_0) \approx U.\Omega_0.X$ by assumption. Furthermore, note that no non-empty action is executable from any subterm of $t'(\Omega_0)$ outside α (and thus also not from $t'(\Omega_0)[\alpha/\alpha_c]$ outside α_c) since this would clearly contradict $t'(\Omega_0) \approx U.\Omega_0.X$ due to $\alpha \overset{c}{\longrightarrow} \alpha_c$. Let us assume $\alpha \overset{d}{\not\Longrightarrow}$ for the sake of contradiction. Due to $t'(\Omega_0) \overset{d}{\Longrightarrow}$ and the fact that from $t'(\Omega_0)$ no non-empty action can be executed outside α we must have $\alpha \overset{\tau}{\Longrightarrow} 0$. We surely cannot have $\alpha_c \overset{w}{\Longrightarrow} 0$ for any $w \in \mathbb{A}_\tau^*$ since $t'(\Omega_0)[\alpha/\alpha_c]$ should not be able to reach any process that is weakly bisimilar to $t(\Omega)[\alpha/0] \approx U.\Omega_0.X$ (for instance c may not be executable from $t'(\Omega_0)[\alpha/\alpha_c]$ but from $U.\Omega_0.X$). Also recall that $t'(\Omega_0)[\alpha/\alpha_c]$ cannot execute any non-empty action outside α_c. Hence, due to $\mathsf{only}(t'(\Omega_0)[\alpha \to \alpha_c], w(\Omega_0)e)$ we must have $\mathsf{only}(\alpha_c, w(\Omega_0)e)$, which along with $\alpha \overset{c}{\longrightarrow} \alpha_c$ contradicts (P1). **End of Proof of Claim.**

Let us finish the proof of the lemma and show that $t(\Omega_0) \not\approx U.\Omega_0.X$. Recall our assumption that $t(\Omega_0) \approx U.\Omega_0.X$ for the sake of contradiction. We apply the previous claim and fix some term $t'(\Omega_0)$ such that

(i) $t(\Omega_0) \overset{\tau}{\Longrightarrow} t'(\Omega_0)$,
(ii) there is some maximal (with respect to subterm order) enabled parallel subterm $\alpha \in \mathbb{P}(\Sigma)$ of $t'(\Omega_0)$ with $\alpha \overset{c}{\longrightarrow} \alpha_c$ for some $\alpha_c \in \mathbb{G}(\Sigma)$, and
(iii) $\alpha \overset{d}{\Longrightarrow} \alpha_d$ for some $\alpha_d \in \mathbb{G}(\Sigma)$.

Note that $t'(\Omega_0) \approx U.\Omega_0.X$ has to hold too. Moreover, we cannot have $\mathsf{only}(\alpha_c, w(\Omega_0)e)$ nor $\mathsf{only}(\alpha_d, w(\Omega_0)f)$ by (P1) and (P2). We define $t_c(\Omega_0) = t'(\Omega_0)[\alpha/\alpha_c]$ and $t_d(\Omega_0) = t'(\Omega)[\alpha/\alpha_d]$. Recall that we have $\Omega_0 \in \{A, B\}^+$ by assumption. Without loss of generality we assume that Ω_0 begins with the letter A. The other case is symmetric. Note that we must have $t_c(\Omega_0) \overset{a}{\Longrightarrow}$ and $t_d(\Omega_0) \overset{a}{\Longrightarrow}$ but also $t_c(\Omega_0) \overset{b}{\not\Longrightarrow}$ and $t_d(\Omega_0) \overset{b}{\not\Longrightarrow}$.

We claim that the action a has to be executable by a subterm of $t_c(\Omega_0)$ that is outside α_c *or* by a subterm of $t_d(\Omega_0)$ that is outside α_d. Assume by contradiction that *both* the part of $t_c(\Omega_0)$ outside α_c *and* the part of $t_d(\Omega_0)$ outside α_d cannot execute a. Since α was chosen to be maximal, it follows that both the rest of $t_c(\Omega_0)$ outside α_c and the rest of $t_d(\Omega_0)$ outside α_d cannot execute the action a nor b before α_c (resp. α_d) terminates, i.e. before $\alpha_c \overset{w}{\Longrightarrow} 0$ (resp. $\alpha_d \overset{w}{\Longrightarrow} 0$) happens for some $w \in \mathbb{A}_\tau^*$. Surely, both α_c and α_d have to terminate, for otherwise this would imply $\mathsf{only}(\alpha_c, w(\Omega_0)e)$ or $\mathsf{only}(\alpha_d, w(\Omega_0)f)$, thus contradicting (P1) or (P2). Hence there have to exist (not necessarily different) suffixes v and v' of $w(\Omega_0)$ for which we have $\mathsf{only}(t'(\Omega_0)[\alpha \to 0], ve)$ and $\mathsf{only}(t'(\Omega_0)[\alpha \to 0], v'f)$, clearly a contradiction.

Let us finally assume without loss of generality that a is executable in a subterm of $t_c(\Omega_0)$ that is outside of α_c. But in particular, this implies that the action a is executable in a subterm of $t'(\Omega_0)$ that is outside of α, let t_a be the term that results from this execution, i.e. $t'(\Omega_0) \overset{a}{\Longrightarrow} t_a$. Also, we must have $t_a \overset{c}{\Longrightarrow} t_a[\alpha/\alpha_c]$, thus $t'(\Omega_0) \overset{a}{\Longrightarrow} t_a \overset{c}{\Longrightarrow} t_a[\alpha/\alpha_c]$ in total. But clearly we must also have $t'(\Omega_0) \overset{c}{\Longrightarrow} t'(\Omega_0)[\alpha/\alpha_c] \overset{a}{\Longrightarrow} t_a[\alpha/\alpha_c]$. The latter confluence clearly contradicts $t'(\Omega_0) \approx U.\Omega_0.X$. □

Finally, we can give the proof of Theorem 5.12.

PROOF. Assume by contradiction that there is some PAN $\mathcal{P} = (\Sigma, \mathbb{A}_\tau, \Delta)$ and some term t_0 with respect to $\mathcal{P}$ such that t_0 is weakly bisimilar to the process $U.X$ of the PDS from Definition 5.13. Let Ω be the sequence of Lemma 5.17 for $\mathcal{P}$. The process $U.X$ can reach via the sequence $w(\Omega^R)$, where Ω^R denotes the reverse of Ω, the state $U.\Omega.X$. Hence $t_0 \overset{w(\Omega^R)}{\Longrightarrow} t$ for some process t satisfying $t \approx U.\Omega.X$. However the latter contradicts Lemma 5.17. □

THEOREM 5.18. $\mathsf{PN} \not\leq_\approx \mathsf{GTRS}$

The proof can be done by observing that $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ is a PN language (e.g. see [Thomas 2005]), while this language is not a trace language of GTRS (e.g. see [Löding 2003]).

6. APPLICATIONS

In this section, we provide applications of the connections that we establish between GTRS and the PRS hierarchy. Instead of attempting to exhaust all possible applications, we shall only highlight a few of the key applications. In particular, Theorem 4.1 allows us to transfer decidability/complexity upper bounds on model checking over GTRS to model checking over PA/PAD-processes.

The first application is the decidability of EF-logic over PAD. The logic EF is the extension of Hennessy-Milner logic with reachability operators, possibly parameterized over subsets of all possible actions (e.g. see [Göller and Lin 2011b; Stirling 1998; To 2010]). We briefly recall the syntax of EF-logic (see [Göller and Lin 2011b; Stirling 1998; To 2010] for a more thorough definition). EF-formulas over $\mathbb{A}$ is defined by the following grammar:

$$\varphi \quad ::= \quad \mathtt{true} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \langle\Gamma\rangle\varphi \mid \langle\Gamma^*\rangle\varphi$$

where Γ is ranges over any subset of $\mathbb{A}$. The semantics is standard: $\langle\Gamma\rangle$ means that an action $a \in \Gamma$ can be executed after which φ is satisfied, while $\langle\Gamma^*\rangle$ means that a sequence of actions with labels from Γ can be executed after which φ is satisfied.

The decidability of EF model checking over GTRS has been known for a long time, e.g., it follows from the results of [Brainerd 1969; Dauchet and Tison 1990]. Together with Theorem 4.1, this easily gives another proof of the following result of Mayr.

THEOREM 6.1 ([MAYR 2001]). *Model checking EF-logic over* PAD *is decidable.*

PROOF. In order to model check an EF formula φ with respect to a PAD $\mathcal{P} = (\Sigma, \mathbb{A}, \Delta)$ and initial configuration $[t_0]_\equiv$, we compute in polynomial-time a GTRS $\mathcal{R}$ and a tree t_0' such that $(\mathcal{T}(\mathcal{P}), [t_0]_\equiv) \simeq (\mathcal{T}(\mathcal{R}), t_0')$. Since $\mathcal{R}$ allows τ-transitions, we need to modify φ a little bit. This can be done by replacing each occurrence of $\langle\Gamma\rangle\varphi$ in φ by $\langle\{\tau\}^*\rangle\langle\Gamma\rangle\langle\{\tau\}^*\rangle\varphi$, and replacing each occurrence of $\langle\Gamma^*\rangle\varphi$ in φ by $\langle\Gamma_\tau^*\rangle\varphi$. Let φ' be the modified EF formula. It is easy to check that $(\mathcal{T}(\mathcal{P}), [t_0]_\equiv) \models \varphi$ iff $(\mathcal{T}(\mathcal{R}), t_0') \models \varphi'$. For completeness, we provide a proof for this by induction on φ.

The base case when $\varphi = \mathtt{true}$ is vacuous. Boolean combinations are also obvious. So we proceed to the other inductive cases. We shall only provide the proof for the case when $\varphi = \langle\Gamma\rangle\psi$; the case when $\varphi = \langle\Gamma^*\rangle\psi$ is similar. In this case, we have $\varphi' = \langle\{\tau\}^*\rangle\langle\Gamma\rangle\langle\{\tau\}^*\rangle\psi'$. Suppose that $(\mathcal{T}(\mathcal{P}), [t_0]_\equiv) \models \varphi$. Then, $[t_0]_\equiv \xrightarrow{a} [t_1]_\equiv$ such that $(\mathcal{T}(\mathcal{P}), [t_1]_\equiv) \models \psi$. By branching bisimilarity, we have $t_0' \overset{\tau}{\Longrightarrow} s_0 \xrightarrow{a} s_1 \overset{\tau}{\Longrightarrow} s_2$ (for some trees s_0, s_1, and s_2) such that $[t_0]_\equiv \simeq s_0$, $[t_1]_\equiv \simeq s_1$, and $[t_1]_\equiv \simeq s_2$. By induction, we have $(\mathcal{T}(\mathcal{R}), s_2) \models \psi'$ and so $(\mathcal{T}(\mathcal{R}), t_0') \models \varphi'$. Conversely, suppose that $(\mathcal{T}(\mathcal{R}), t_0') \models \varphi'$. Then, we have $t_0' \overset{\tau}{\Longrightarrow} s_0 \xrightarrow{a} s_1 \overset{\tau}{\Longrightarrow} s_2$ (for some trees s_0, s_1, and s_2) such that $(\mathcal{T}(\mathcal{R}), s_2) \models \psi'$. By branching bisimilarity and that there is no τ-transition in $\mathcal{T}(\mathcal{P})$, there exists $[t_1]_\equiv$ such that $[t_0]_\equiv \xrightarrow{a} [t_1]_\equiv$ and $[t_0]_\equiv \simeq s_0$, $[t_1]_\equiv \simeq s_1$, and $[t_1]_\equiv \simeq s_2$. By induction, we have $(\mathcal{T}(\mathcal{P}), [t_1]_\equiv) \models \psi$ and so $(\mathcal{T}(\mathcal{P}), [t_0]_\equiv) \models \varphi$. This completes our proof. □

The second application is the decidability/complexity of model checking the common fragments LTL$_{det}$ (called deterministic LTL) and LTL$(\mathbf{F}_s, \mathbf{G}_s)$ [Bozzelli et al. 2009; Maidl 2000] of LTL over PAD. These fragments are suffciently powerful for expressing interesting properties like safety, fairness, liveness, and also some simple stuttering-invariant LTL properties. The following two theorems follow from the results for GTRS [To 2010; To and Libkin 2010]; decidability with no upper bounds was initially proven in [Bozzelli et al. 2009].

THEOREM 6.2. *Model checking LTL$_{det}$ over* PAD *is* coNP*-complete and is decidable in time exponential in the size of the formula and polynomial in the size of the system. Model checking LTL$(\boldsymbol{F}_s, \boldsymbol{G}_s)$ over* PAD *is decidable in time double exponential in the size of the formula and polynomial in the size of the system.*

We briefly recall the syntax of LTL over a finite set $\Gamma \subseteq \mathbb{A}$:

$$\varphi, \varphi' := a \ (a \in \mathbb{A}) \mid \neg\varphi \ \mid \varphi \vee \varphi' \mid \varphi \wedge \varphi' \mid \mathbf{X}\varphi \mid \varphi\mathbf{U}\varphi'.$$

We shall use the standard abbreviations: $\mathbf{F}\varphi$ for *true*$\mathsf{U}\varphi$, $\mathbb{G}\varphi$ for $\neg\mathbf{F}\neg\varphi$, and $\mathbf{F}_s$ and $\mathbf{G}_s$ for their strict versions: $\mathbf{F}_s\varphi = \mathbf{XF}\varphi$ and $\mathbf{G}_s\varphi = \neg\mathbf{F}_s\neg\varphi$. The semantics $[\![\varphi]\!]$ of an LTL formula φ is standard (e.g. see [To 2010] and references therein): it is the set of ω-words over Γ that satisfy the formula φ. Note: the results in this paper hold even if we include finite (as well as infinite) words for the semantics of LTL.

We now recall the definitions of the fragments LTL$_{det}$ and LTL$(\mathbf{F}_s, \mathbf{G}_s)$ of LTL. The fragment LTL$(\mathbf{F}_s, \mathbf{G}_s)$ contains precisely all LTL formulas that use only two temporal operators $\mathbf{F}_s$ and $\mathbf{G}_s$. Observe that these operators can express future/global operators: $[\![\mathbf{F}\varphi]\!] = [\![\varphi \vee \mathbf{F}_s\varphi]\!]$ and $[\![\mathbf{G}\varphi]\!] = [\![\varphi \wedge \mathbf{G}_s\varphi]\!]$.

The logic LTL$_{det}$, called deterministic LTL, was introduced by Maidl [Maidl 2000]. Its syntax is given as follows:

$$\begin{array}{lcl} \phi, \phi' & := & p \mid \mathbf{X}\phi \mid \phi \wedge \phi' \mid (p \wedge \phi) \vee (\neg p \wedge \phi') \mid \\ & & (p \wedge \phi)\mathbf{U}(\neg p \wedge \phi') \mid (p \wedge \phi)\mathbf{W}(\neg p \wedge \phi'). \end{array}$$

Here p is a boolean combination of actions in Γ. The semantics can be defined in the same way as for LTL. For example, $\phi\mathbf{W}\phi'$ is interpreted as the formula $\mathbf{G}\phi \vee (\phi\mathsf{U}\phi')$, i.e., the *weak until* operator.

We now recall the translations from LTL$_{det}$ and LTL$(\mathbf{F}_s, \mathbf{G}_s)$ formulas into a special subclass of Nondeterministic Büchi Word Automata (NBWA) called *almost linear NBWA* as introduced in [Babiak et al. 2012] To define almost linear NBWAs, we shall first define the notion of linear NBWAs. An NBWA $\mathcal{A} = (\Gamma, Q, \delta, q_0, F)$ is called *linear* (a.k.a. *1-weak*) if there exists a partial order $\preceq \subseteq Q \times Q$ such that $q' \in \delta(q, a)$ implies $q \preceq q'$. Intuitively, the partial order ensures that once $\mathcal{A}$ leaves a state q, it will never be able to come back to q. In other words, graph-theoretically $\mathcal{A}$ looks like a dag possibly with self-loops, i.e., each strongly connected component (SCC) in $\mathcal{A}$ contains only a single state. Observe that every accepting run of $\mathcal{A}$ must eventually self-loop in one final state $q \in F$, i.e., *sink* at q. The *depth* of $\mathcal{A}$ refers to the length of the longest simple path in $\mathcal{A}$.

Definition 6.3. An *almost linear NBWA* $\mathcal{A}$ over the alphabet Γ is a pair of a linear NBWA $\mathcal{B} = (\Gamma, Q, \delta, q_0, F)$ and a function χ mapping each final state $q \in F$ to an LTL formula over Γ of the form

$$\bigwedge_{i \in I} \mathbf{GF}p_i$$

where each p_i is a disjunction of positive atomic formulas. The language $L(\mathcal{A})$ of $\mathcal{A}$ contains all words $w \in \Gamma^\omega$ for which there is an accepting run of $\mathcal{B}$ on w sinking at some $q \in F$ which satisfies $w \models \chi(q)$. The size $\|\mathcal{A}\|$ of $\mathcal{A}$ is simply the sum of $\|\mathcal{B}\|$ and $\sum_{q \in F} \|\chi(q)\|$.

Almost linear NBWAs are not more powerful than NBWAs in terms of expressive power: there is a simple polynomial-time translation from almost linear NBWAs to NBWAs by a technique that is similar to the reduction from generalized Büchi automata to standard Büchi automata. The following propositions are results from [Maidl 2000] (for LTL$_{det}$) and [Babiak et al. 2012] (for LTL$(\mathbf{F}_s, \mathbf{G}_s)$).

PROPOSITION 6.4. *Given an LTL$_{det}$ formula φ, we may compute in polynomial-time a 1-weak NBWA $\mathcal{A}_{\neg\varphi}$ such that $L(\mathcal{A}_{\neg\varphi}) = [\![\neg\varphi]\!]$. Given an LTL$(\boldsymbol{F}_s, \boldsymbol{G}_s)$ formula φ, we may compute in double exponential time*[2] *an almost linear NBWA $\mathcal{A}_{\neg\varphi}$ with at most exponentially large depth such that $L(\mathcal{A}_{\neg\varphi}) = [\![\neg\varphi]\!]$.*

[2]In the previous version of [Babiak et al. 2012], they had a triple-exponential time algorithm, which they improved to double-exponential time.

Using Theorem 4.1, we may compute a GTRS $\mathcal{R} = (A, \mathbb{A}, R)$ and a tree $t'_0 \in \mathsf{Trees}_A$ that is branching bisimilar to the given system $(\mathcal{T}(\mathcal{P}), [t_0]_\equiv)$ generated by an input PAD $\mathcal{P}$. We may now make use of the following two results for GTRS:

PROPOSITION 6.5 ([LIN 2012]). *Given a 1-weak NBWA $\mathcal{A}$ over Γ, a GTRS $\mathcal{R} = (A, \Gamma, R)$ with $\Gamma \subseteq \mathbb{A}$, and a tree $t_0 \in \mathsf{Trees}_A$, the problem of deciding whether there exists an infinite trace $w \in [\![\mathcal{A}]\!]$ from t_0 in the system $\mathcal{T}(\mathcal{R})$ is NP-complete.*

PROPOSITION 6.6 ([TO 2010]). *Given an almost linear NBWA $\mathcal{A}$ over Γ, a GTRS $\mathcal{R} = (A, \Gamma, R)$ with $\Gamma \subseteq \mathbb{A}$, and a tree $t_0 \in \mathsf{Trees}_A$, we may effectively decide whether there exists an infinite trace $w \in [\![\mathcal{A}]\!]$ from t_0 in the system $\mathcal{T}(\mathcal{R})$. Furthermore, this can be done in time exponential in the depth of $\mathcal{A}$ and polynomial in the size of $\mathcal{R}$.*

Therefore, we compute an almost linear NBWA $\mathcal{A}_{\neg\varphi}$ from the negation $\neg\varphi$ of the input LTL$_{det}$ or LTL$(\mathbf{F}_s, \mathbf{G}_s)$ formula. Since $\mathcal{R}$ now has τ-transitions, we explicitly introduce the label τ for the automaton $\mathcal{A}_{\neg\varphi}$ and allow each state in the automaton to loop with a τ-transition. Call the resulting automaton $\mathcal{A}'_{\neg\varphi}$. It is easy to see that $(\mathcal{T}(\mathcal{P}), [t_0]_\equiv) \not\models \varphi$ iff there exists an infinite trace $w \in [\![\mathcal{A}'_{\neg\varphi}]\!]$ from t'_0 in the system $\mathcal{T}(\mathcal{R})$. The complexity upper bound now immediately follows from the propositions above.

We now prove NP-hardness, which turns out to hold already for BPP. The reduction is from satisfiability of boolean formulas in conjunctive normal form with each clause having at most three literals. Given a formula

$$\varphi = C_1 \wedge \cdots \wedge C_m$$

over the boolean variables $\{x_1, \ldots, x_k\}$, where $C_i = l_1^i \vee l_2^i \vee l_3^i$, our process constants are $X_1, \ldots, X_k, C_1, \ldots, C_m$. For each $i = 1, \ldots, k$, we let S_i (resp. $\bar{S}_i$) denote the set of indices $j \in \{1, \ldots, m\}$ where the variable x_i (resp. $\neg x_i$) occurs in the clause C_j; in other words, assigning 1 (resp. 0) to the variable x_i makes C_j true. Now, we add the following rewrite rules:

— $X_i \xrightarrow{x_i} \|_{j \in S_i} C_j$, for each $i = 1, \ldots, k$.
— $X_i \xrightarrow{x_i} \|_{j \in \bar{S}_i} C_j$, for each $i = 1, \ldots, k$.
— $C_i \xrightarrow{c_i} 0$, for each $i = 1, \ldots, m$.

We now define the output LTL$_{det}$ formula. Firstly, define ψ'_i $(i = 1, \ldots, m)$ inductively as follows: $\psi'_1 = c_1$ and $\psi'_i = c_i \wedge \mathbf{X}\psi'_{i-1}$ for each $i \in \{2, \ldots, m\}$. Now, define another sequence of formulas ψ_i $(i = 1, \ldots, k)$ inductively as follows: $\psi_1 = x_1 \wedge \mathbf{X}\psi'_m$ and $\psi_i = x_i \wedge \mathbf{X}\psi_{i-1}$ for each $i \in \{2, \ldots, k\}$. The output formula is $\psi = \psi_k$. It is easy to see that φ is satisfiable iff $X_1 \| \cdots \| X_k \not\models \neg\psi$, which completes our reduction.

Acknowledgment: We thank Taolue Chen, Robert Fink, Matthew Hague, Jean Christoph Jung, Richard Mayr, Colin Stirling, and anonymous reviewers for their comments. This work was done when Anthony W. Lin was at Oxford University, supported by EPSRC (EP/H026878/1).

REFERENCES

BAADER, F. AND NIPKOW, T. 1998. *Term Rewriting and All That*. Cambridge University Press.

BABIAK, T., REHÁK, V., AND STREJCEK, J. 2012. Almost linear büchi automata. *Mathematical Structures in Computer Science 22,* 2, 203–235.

BAETEN, J. C. M., BERGSTRA, J. A., AND KLOP, J. W. 1987. Decidability of bisimulation equivalence for processes generating context-free languages. In *PARLE (2)*, J. W. de Bakker, A. J. Nijman, and P. C. Treleaven, Eds. Lecture Notes in Computer Science Series, vol. 259. Springer, 94–111.

BERGSTRA, J. A. AND KLOP, J. W. 1985. Algebra of communicating processes with abstraction. *Theor. Comput. Sci. 37*, 77–121.

BOUAJJANI, A., ECHAHED, R., AND HABERMEHL, P. 1995. On the verification problem of nonregular properties for nonregular processes. In *LICS*. IEEE Computer Society, 123–133.

Bouajjani, A., Müller-Olm, M., and Touili, T. 2005. Regular symbolic analysis of dynamic networks of pushdown systems. In *CONCUR*, M. Abadi and L. de Alfaro, Eds. Lecture Notes in Computer Science Series, vol. 3653. Springer, 473–487.

Bouajjani, A. and Touili, T. 2003. Reachability analysis of process rewrite systems. In *FSTTCS*, P. K. Pandya and J. Radhakrishnan, Eds. Lecture Notes in Computer Science Series, vol. 2914. Springer, 74–87.

Bozzelli, L., Kretínský, M., Rehák, V., and Strejcek, J. 2009. On decidability of LTL model checking for process rewrite systems. *Acta Inf. 46,* 1, 1–28.

Brainerd, W. S. 1969. Tree generating regular systems. *Information and Control 14,* 2, 217–231.

Burkart, O., Caucal, D., Moller, F., and Steffen, B. 2001. Verification on infinite structures. In *Handbook of process algebra*. Elsevier, North-Holland, 545–623. Chapter 9.

Christensen, S. 1993. Decidability and decomposition in process algebras. Ph.D. thesis, Department of Computer Science, The University of Edinburgh.

Comon, H., Dauchet, M., Gilleron, R., Löding, C., Jacquemard, F., Lugiez, D., Tison, S., and Tommasi, M. 2007. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`. release October, 12th 2007.

Coquidé, J.-L., Dauchet, M., Gilleron, R., and Vágvölgyi, S. 1994. Bottom-up tree pushdown automata: Classification and connection with rewrite systems. *Theor. Comput. Sci. 127,* 1, 69–98.

Dauchet, M. and Tison, S. 1990. The theory of ground rewrite systems is decidable. In *LICS*. IEEE Computer Society, 242–248.

Esparza, J. and Nielsen, M. 1994. Decidability issues for petri nets - a survey. *Bulletin of the EATCS 52*, 244–262.

Göller, S. and Lin, A. W. 2011a. Refining the Process Rewrite Systems Hierarchy via Ground Tree Rewrite Systems. In *CONCUR*. Lecture Notes in Computer Science Series, vol. 6901. Springer, 543–558.

Göller, S. and Lin, A. W. 2011b. The Complexity of Verifying Ground Tree Rewrite Systems. In *LICS*. IEEE Computer Society, 279–288.

Hack, M. H. T. 1976. Decidability questions for petri nets. Ph.D. thesis, MIT.

Lin, A. W. 2012. Weakly-synchronized ground tree rewriting - (with applications to verifying multithreaded programs). In *MFCS*. 630–642.

Löding, C. 2003. Infinite graphs generated by tree rewriting. Ph.D. thesis, RWTH Aachen.

Lugiez, D. and Schnoebelen, P. 2002. The regular viewpoint on pa-processes. *Theor. Comput. Sci. 274,* 1-2, 89–115.

Maidl, M. 2000. The common fragment of CTL and LTL. In *FOCS*. 643–652.

Mayr, R. 2000. Process rewrite systems. *Inf. Comput. 156,* 1-2, 264–286.

Mayr, R. 2001. Decidability of model checking with the temporal logic ef. *Theor. Comput. Sci. 256,* 1-2, 31–62.

Milner, R. 1989. *Communication and Concurrency*. Prentice Hall.

Muller, D. E. and Schupp, P. E. 1985. The theory of ends, pushdown automata, and second-order logic. *Theor. Comput. Sci. 37*, 51–75.

Stirling, C. 1998. The joys of bisimulation. In *Mathematical Foundations of Computer Science 1998, 23rd International Symposium, MFCS'98, Brno, Czech Republic, August 24-28, 1998, Proceedings*. Lecture Notes in Computer Science Series, vol. 1450. Springer, 142–151.

Thomas, W. 2005. Applied automata theory. Course Notes (RWTH Aachen).

To, A. W. 2010. Model checking infinite-state systems: Generic and specific approaches. Ph.D. thesis, LFCS, School of Informatics, University of Edinburgh.

To, A. W. and Libkin, L. 2010. Algorithmic metatheorems for decidable LTL model checking over infinite systems. In *FOSSACS*, C.-H. L. Ong, Ed. Lecture Notes in Computer Science Series, vol. 6014. Springer, 221–236.

van Glabbeek, R. J. and Weijland, W. P. 1996. Branching time and abstraction in bisimulation semantics. *J. ACM 43,* 3, 555–600.