Refining the Process Rewrite Systems Hierarchy via Ground Tree Rewrite Systems

Stefan Göller[1] and Anthony Widjaja Lin[2]

[1] Universität Bremen, Institut für Informatik, Germany
[2] Oxford University Department of Computer Science, UK

Abstract. In his seminal paper, R. Mayr introduced the well-known Process Rewrite Systems (PRS) hierarchy, which contains many well-studied classes of infinite 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.

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 the 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. [2, 21]). The same can be said about infinite-state models with unbounded parallelism, which include Basic Parallel Processes (BPP) and Petri nets (PN), e.g. [10, 14]. 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 [3], 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 is still decidable (e.g. see [9]), which makes them suitable for verification.

In his seminal paper [18], R. Mayr introduced the Process Rewrite Systems (PRS) hierarchy (see leftmost diagram in Figure 1) containing several models of infinite-state systems that generalize 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/right hand sides of the rewrite rules. In addition to the aforementioned models of infinite 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 of 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 nice decidability properties. While extending the expressive power of PDS, GTRS still enjoys decidability of reachability (e.g. [8, 11]), recurrent reachability [15], model checking first-order logic with reachability [12], and model checking the fragments LTL_{det} and $\mathrm{LTL}(\mathbf{F}_\mathrm{s}, \mathbf{G}_\mathrm{s})$ of LTL [24, 23]. 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 [7, 15, 18, 19, 23, 24]. Furthermore, model checking general LTL properties is undecidable for both PA-processes and GTRS [7, 24]. Despite these, the precise connection between the 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 [20]. 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 [25] 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 τ-actions; 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 model checking the logic EF over PAD [19], and decidability (with good complexity upper bounds) 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 [7] without upper bounds). In fact, we also show that Regular Ground Tree Rewrite Systems (RGTRS) [15] — 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 [9]), abbreviated as PREF, up 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 [16] and Bouajjani and Touili [6], which study PRS (or subclasses thereof) by first distinguishing process terms that are "equivalent" in Mayr's sense [18]. 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 [5]) since an extension of DPN given in [5] 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 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.

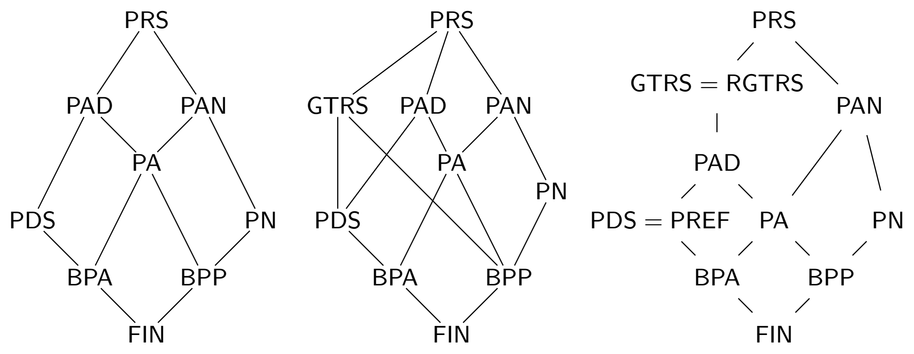

Fig. 1. Depictions of Mayr's PRS hierarchy and their refinements via GTRS as Hasse diagrams (the top being the most expressive). The leftmost diagram is the original (strict) PRS hierarchy where expressiveness is measured with respect to strong bisimulation. The middle (resp. right) diagram is a strict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

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 weak/branching/strong bisimulation equivalence: Let us fix a countable set of action labels Act. A *transition system* is tuple $\mathcal{T} = (S, \mathbb{A}, \{\xrightarrow{a} \mid a \in \mathbb{A}\})$, where S is a set of *states*, $\mathbb{A} \subseteq$ Act is a finite set of action labels, and where $\xrightarrow{a} \subseteq S \times S$ is a set of *transitions*. 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 $R \subseteq S \times S$, 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 $\longrightarrow = \xrightarrow{\mathbb{A}}$. 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 $\overset{\tau}{\Longrightarrow} = (\xrightarrow{\tau})^*$ and $\overset{a}{\Longrightarrow} = (\xrightarrow{\tau})^* \circ \xrightarrow{a} \circ (\xrightarrow{\tau})^*$ for each $a \in \mathbb{A}$.

A binary relation $R \subseteq S \times S$ is a *branching bisimulation* if R is symmetric and if for each $(s, t) \in R$ the following two conditions hold: (i) if $s \xrightarrow{\tau} s'$, then $(s', t) \in R$ and (ii) if $s \xrightarrow{a} s'$ for some $a \in \mathbb{A}$, then there is $t \overset{\tau}{\Longrightarrow} t' \xrightarrow{a} t'' \overset{\tau}{\Longrightarrow} t'''$ such that $(s, t'), (s', t''), (s', t''') \in R$. 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 $t \overset{a}{\Longrightarrow} t'$ 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 $\mathcal{T}_2$.

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, see e.g. [13] and the references therein.

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 $L(\mathcal{A}) = \{t \in \mathsf{Trees}_A \mid \text{there is an accepting run of } \mathcal{A} \text{ on } t\}$ 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, $\mathbb{A} \subseteq$ 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 PRS hierarchy are Finite Systems (FIN), Basic Process Algebra (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	$\mathbb{S}(\Sigma)$	$\mathbb{S}(\Sigma)$
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 [16, 6] to define the semantics of PRS. While Mayr [18] 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:

$$\frac{}{t.0 \equiv t}\ R0. \qquad \frac{}{t_1.(t_2.t_3) \equiv (t_1.t_2).t_3}\ A. \qquad \frac{t_1 \equiv u_1 \quad t_2 \equiv u_2}{t_1.t_2 \equiv u_1.u_2}\ \text{Con.}$$

$$\frac{}{0.t \equiv t}\ L0. \qquad \frac{}{t_1\|(t_2\|t_3) \equiv (t_1\|t_2)\|t_3}\ A\| \qquad \frac{t_1 \equiv u_1 \quad t_2 \equiv u_2}{t_1\|t_2 \equiv u_1\|u_2}\ \text{Con}\|$$

$$\frac{}{t\|0 \equiv t}\ R0\| \qquad \frac{}{t_1\|t_2 \equiv t_2\|t_1}\ C\| \qquad \frac{u \equiv u' \quad u' \equiv u''}{u \equiv u''}\ \text{Trans}$$

$$\frac{}{0\|t \equiv t}\ L0\| \qquad \frac{}{u \equiv u}\ \text{Ref} \qquad \frac{t \equiv u}{u \equiv t}\ \text{Sym}$$

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 [18] 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, $\mathbb{A} \subseteq \mathsf{Act}$ is a finite set of action labels and where R is finite set of rewrite rules $L \overset{a}{\hookrightarrow} L'$, where L and L' are regular tree languages over A 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 $L \overset{a}{\hookrightarrow} L' \in R$ such that $t^{\downarrow x} = s$ and $t' = t[x/s']$ for some $s \in L$ and some $s' \in L'$.

A *ground tree rewrite system* (GTRS) is an RGTRS $\mathcal{R} = (A, \mathbb{A}, R)$, where for each $L \overset{a}{\hookrightarrow} L' \in R$ we have that both $L = \{t\}$ and $L' = \{t'\}$ is a singleton; we also write $t \overset{a}{\hookrightarrow} t' \in R$ 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 1 **(**PAD $\leq_{\simeq}$ GTRS**).** *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 S' 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 more close to a GTRS since the domain S 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:

$$\boxed{\begin{array}{lll}
\dfrac{}{0.t \xrightarrow{\tau} t} & \dfrac{}{t\|0 \xrightarrow{\tau} t} & \dfrac{t_1 \xrightarrow{\tau} t_1'}{t_1.t_2 \xrightarrow{\tau} t_1'.t_2} \\
\dfrac{}{t.0 \xrightarrow{\tau} t} & \dfrac{}{t_1.(t_2.t_3) \xrightarrow{\tau} (t_1.t_2).t_3} & \dfrac{t_2 \xrightarrow{\tau} t_2'}{t_1\|t_2 \xrightarrow{\tau} t_1\|t_2'} \\
\dfrac{}{0\|t \xrightarrow{\tau} t} & \dfrac{t_1 \xrightarrow{\tau} t_1'}{t_1\|t_2 \xrightarrow{\tau} t_1'\|t_2} & \dfrac{t_2 \xrightarrow{\tau} t_2'}{t_1.t_2 \xrightarrow{\tau} t_1.t_2'}
\end{array}}$$

Here, t is allowed to be any process term. Observe that these τ-transitions remove redundant occurrences of 0 and turns 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 note a few simple facts about $\xrightarrow{\tau}$ in the following lemmas.

Lemma 2. *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.*

Lemma 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 2 is a basic property of a rewrite system commonly known as *confluence* and *termination* (e.g. see [1]). In fact, it does not take long to terminate. Lemma 3 gives the form of the unique "minimal" term with respect to $\xrightarrow{\tau}$ given various different initial starting points. The proofs of these lemmas are standard. For the rest of the proof of Theorem 1, we assume the following conventions:

Convention 4 *The term t_0 and all process terms in $\mathcal{P}$ are minimal with respect to $\xrightarrow{\tau}$. That is, each of such 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 $\mathcal{T}_0(\mathcal{P})$.

Lemma 5. *$\equiv$ is a branching bisimulation on $\mathcal{T}_0(\mathcal{P})$.*

The proof of this lemma is not difficult but tedious. 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 6. *The relation $R = \{(C, t) \subseteq S' \times S : t \in C\}$ is a branching bisimulation between $\mathcal{T}(\mathcal{P})$ and $\mathcal{T}_0(\mathcal{P})$.*

Removing complex τ-transitions: Corollary 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 subtrees (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 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 7. *Fix a term $t \in post^*(t_0)$ with respect to $\mathcal{T}_0(\mathcal{P})$. Then, any subtree 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:

$$\boxed{\begin{array}{lll}
\dfrac{}{0.t \xrightarrow{\tau} t}\; t \in W & \dfrac{}{t\|0 \xrightarrow{\tau} t} & \dfrac{t_1 \xrightarrow{\tau} t_1'}{t_1.t_2 \xrightarrow{\tau} t_1'.t_2}\; t_2 \in W \\
\dfrac{}{0\|t \xrightarrow{\tau} t} & \dfrac{t_2 \xrightarrow{\tau} t_2'}{t_1\|t_2 \xrightarrow{\tau} t_1\|t_2'} & \dfrac{t_1 \xrightarrow{\tau} t_1'}{t_1\|t_2 \xrightarrow{\tau} t_1'\|t_2}
\end{array}}$$

Observe that the rule $t.0 \xrightarrow{\tau} t$ may be omitted since no subtree 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 subtree of $t \in post^*(t_0)$ of the form $t_1.(t_2.t_3)$ exists. Other rules are omitted because any subtree 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 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 8. *$(\mathcal{T}_0'(\mathcal{P}), t)$ is branching bisimilar to $(\mathcal{T}_0(\mathcal{P}), t)$.*

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 subtree u of a tree $t \in post^*(t_0)$ of the form $t_1.t_2$ satisfies $t_2 \in W$, we may simply substitute u with the tree $\hat{t_2}(t_1)$ and perform this substitution recursively on t_1. Denote by $\lambda(t)$ the resulting tree 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 1.

4.2 Further containment results

Theorem 9. BPP $\leq_\sim$ GTRS.

Proof (sketch). The idea is to construct from some BPP a GTRS, where each leaf corresponds to a process constant. A leaf is either marked or unmarked. An unmarked leaf X can become marked with the fresh symbol \$ via the action a in case the rule $X \mapsto_a 0$ is present in the BPP. Rules of the kind $X \mapsto_a Y_1 \| \ldots \| Y_n$ are realized via $X \overset{a}{\hookrightarrow} \bullet(Y_1, \ldots, Y_n)$ in the GTRS. Moreover the GTRS does not contain any rules, where a marked leaf is on the left-hand side of a rule. □

Theorem 10. GTRS $\leq_\sim$ PRS.

Proof (sketch). Let k be the maximal rank of the alphabet of some GTRS. Although parallel composition is interpreted commutatively we can simulate order by using k additional symbols in a PRS. □

Theorem 11. RGTRS $\simeq$ GTRS.

Proof (sketch). A GTRS can simulate via τ-transitions the bottom-up computation of an NTA. In addition, one provides τ-transitions that allow to undo these transitions. □

In analogy to Theorem 11 one can prove the following.

Corollary 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 [10]), and (2) BPP $\not\leq_\approx$ PDS since there is a BPP trace language that is not context-free (e.g. see references in [4]) 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 [15]. 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 $|\mathcal{R}| = |A| + |\mathbb{A}| + \sum_{t \overset{\sigma}{\hookrightarrow} t' \in R} |t| + |t'|$.

Lemma 13. *Let $\Lambda \subseteq \mathbb{A}$. For every $t_0 \in \mathsf{Trees}_A$ there is some $N = \exp(|\mathcal{R}| + \mathit{height}(t_0))$ such that $t_0 \xrightarrow{\Lambda}^{N}$ implies $t_0 \xrightarrow{\Lambda}^{n}$ for infinitely many $n \in \mathbb{N}$.*

The separating PA: 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 state $\alpha = A.D$ in $\mathcal{T}(\mathcal{P})$ is *not* strongly bisimilar to any pointed GTRS. So for the sake of contradiction, let us assume some GTRS $\mathcal{R} = (A, \mathbb{A}, R)$ and some $t_\alpha \in \mathsf{Trees}_A(\mathcal{R})$ with $t_\alpha \sim \alpha$. We note that e.g. by [15] it is known that the set of maximal sequences executable from α (the language of α when $\mathcal{P}$ is interpreted as a language acceptor) are recognizable by some GTRS [15].

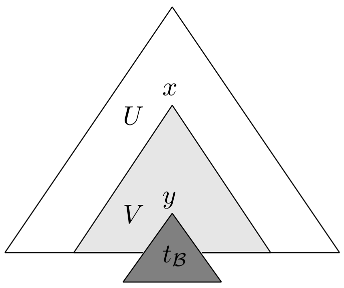

Fig. 2. The tree $T^1 = U[V[t_\mathcal{B}]]$.

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

Let us consider $\mathsf{post}^*_{\{a\}}(t_\alpha)$. First, there is some NTA $\mathcal{A}$ with $L(\mathcal{A}) = \{t_\alpha\}$. A folklore result states that there is some NTA $\mathcal{B}$ with $L(\mathcal{B}) = \mathsf{post}^*_{\{a\}}(L(\mathcal{A})) = \mathsf{post}^*_{\{a\}}(t_\alpha)$, see e.g. [15]. Note that $L(\mathcal{B})$ is infinite since α can reach infinitely many pairwise non-bisimilar states and $t_\alpha \sim \alpha$ by assumption. By applying the Pumping Lemma for regular tree languages, 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) $|y| > 0$, i.e. 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 $n \geq 0$. Moreover we define the consant $\gamma = \ell \cdot (h_\mathcal{R} + 1)$ with $\ell = 2^{|\{t \in \mathsf{Trees}_A | \text{height}(t) \leq h_\mathcal{R}\}|}$, i.e. ℓ denotes the number of different subsets of the set of all trees in Trees_A of height at most $h_\mathcal{R}$.

The following lemma states that if $V^\gamma[t_\mathcal{B}]$ can reach some tree of height at most $h_\mathcal{R}$ by only executing the action σ, then there is already some tree t_σ of height at most h_σ such that for all $i \geq 0$ we have $V^{\theta + i\cdot\delta}[t_\mathcal{B}] \xrightarrow[\mathcal{R}]{\sigma}{}^* t_\sigma$.

Lemma 14. *There exist $\theta, \delta \geq 1$ 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 $V^{\theta+i\cdot\delta}[t_\mathcal{B}] \xrightarrow[\mathcal{R}]{\sigma}{}^* t_\sigma$ for all $i \geq 0$ for some $t_\sigma \in \mathsf{Trees}_A$.*

For the rest of this section, we fix θ and δ from Lemma 14. Note that due to $t_\alpha \sim \alpha$ 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 with $t_\alpha \xrightarrow{a}{}^k t$.

Lemma 15. *$\{k(T^n) \mid n \in \mathbb{N}\}$ is an infinite set.*

Let us immediately apply Lemma 15. Let us fix some residue class r modulo δ such that there are infinitely many n with $n \equiv r \bmod \delta$ all having pairwise distinct $k(T^n)$ values. Among these infinitely many n we will choose a sufficiently large $N \geq \theta$ for the following arguments to work. The tree T_N is depicted in Figure 3. Recall that by definition $T^N \in \mathsf{post}^*_{\{a\}}(t_\alpha)$.

The following lemma states that 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.

Lemma 16. *If $V^\gamma[t_\mathcal{B}] \xrightarrow{\sigma}^* t$, then we have height$(t) > h_\mathcal{R}$ for each $t \in \mathsf{Trees}_A$ and each $\sigma \in \{b, c\}$.*

Let y_N denote the unique node of T^N where the subtree $t_\mathcal{B}$ is rooted at. We call a node $z \in D_{T^N}$ of T^N *off-path* if $z \not\preceq y_N$. For each tree $t \in \mathsf{Trees}_A$ and each $\sigma \in \mathbb{A}$, we define $\sup_\sigma(t) = \sup\left\{j \in \mathbb{N} \mid t \xrightarrow{\sigma}^j\right\}$.

Intuitively speaking, the following lemma states that from the subtree $V^\gamma[t_\mathcal{B}]$ of T^N and subtrees of T^N that are rooted at off-path nodes one only execute a constantly long sequences from b^*c^* or from c^*b^* (unless $t_\alpha \sim \alpha$ is violated). Let us define $\overline{b} = c$ and $\overline{c} = b$. We note that γ and $\mathcal{B}$ only depend on $\mathcal{R}$ and on t_α but not on N.

Lemma 17. *Let $\sigma \in \{b, c\}$. Then there is some constant $J = J(\mathcal{R}, t_\alpha)$ such that $\sup_\sigma(t) \leq J$ whenever either $t = V^\gamma[t_\mathcal{B}]$ or $T^{N\downarrow z} \xrightarrow{\overline{\sigma}}^* t$ for some off-path z.*

We can now prove the main result of this section.

Theorem 18. $\mathsf{PA} \not\leq_\sim \mathsf{GTRS}$.

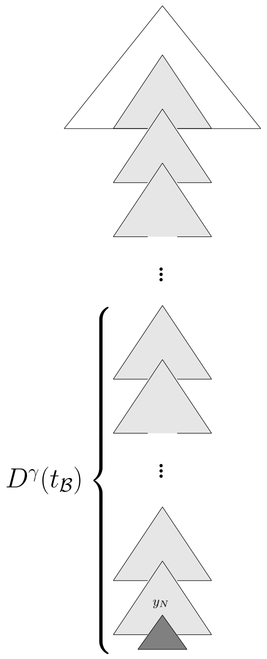

Fig. 3. The tree T^N.

Proof. We give a simple winning strategy for Attacker that contradicts $t_\alpha \sim \alpha$. First Attacker plays $t_\alpha \xrightarrow{a}^{k(T^N)} T^N$. We remark since N is chosen sufficiently large, it follows that $k(T_N)$ is sufficiently large for the following arguments to work. It has to hold for some $s \in \{0, 1\}$

$$T^N \quad \sim \quad \left(A^{1-s} \| \underbrace{B\|B \cdots \|B}_{k(T^N)-s} \| \underbrace{C\|C \cdots \|C}_{k(T^N)-s} \right) .D \qquad (\bigstar)$$

We only treat the case $s = 1$ (the case $s = 0$ can be proven analogously). Recall that γ is a constant that only depends on $\mathcal{R}$ and t_α. On the one hand we cannot modify the subtree $V^\gamma[t_\mathcal{B}]$ of T^N to any tree of height at most $h_\mathcal{R}$ by executing b's only by Lemma 16. On the other hand we cannot execute more than J many b's from the subtree $V^\gamma[t_\mathcal{B}]$, where J is the constant of Lemma 17. Thus, since $T^N \xrightarrow{b}^{k(T^N)-1}$ holds, Attacker can play $k(T^N) - 1 - J$ many b's outside the subtree $V^\gamma[t_\mathcal{B}]$. We recall that $k(T^N) - J$ can be arbitrarily large since J is a constant that only depends on $\mathcal{R}$ and t_α. By definition of T^N all of these $k(T^N) - 1 - J$ many b's can be played on subtrees initially rooted at off-path nodes of T^N *outside* the subtree $V^\gamma[t_\mathcal{B}]$. However from each of these subtrees that are initially rooted at off-path nodes outside the subtree $V^\gamma[t_\mathcal{B}]$, we can execute at most J many b's.

Analogously Attacker can execute $k(T^N) - 1 - J$ many c's from T^N all on subtrees initially rooted at off-path nodes of T_N *outside* the subtree $V^\gamma[t_\mathcal{B}]$.

Attacker now has the following winning strategy. First he plays $k(T^N) - 1 - J$ many b's on subtrees rooted at off-path nodes of T^N *outside* $V^\gamma[t_\mathcal{B}]$. After playing these b's the height each of these subtrees is bounded by a constant that only depends on $\mathcal{R}$ and t_α by Lemma 17. Next, Attacker plays $k(T^N) - 1 - J$ many c's at positions outside the subtree $V^\gamma[t_\mathcal{B}]$ and still, by Lemma 17, the height of all subtrees rooted at off-path nodes outside $V^\gamma[t_\mathcal{B}]$ have a height bounded by a constant that only depends on $\mathcal{R}$ and t_α. Let us call the resulting tree T'. 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 implies that $T^N \xrightarrow{wd}$ for some $w \in \{b,c\}^*$ where $|w|$ is bounded by a constant that only depends on $\mathcal{R}$ and t_α, clearly contradicting (★). □

5.2 GTRS $\not\leq_\approx$ PAD

By Theorem 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 singleton rewrite rules: (i) $X_0 \overset{a}{\hookrightarrow} X_1(X_0)$, (ii) $X_1(X_0) \overset{b}{\hookrightarrow} X_0$, (iii) $Y_1 \overset{c}{\hookrightarrow} Y_1(Y_0)$, (iv) $Y_1(Y_0) \overset{d}{\hookrightarrow} Y_0$, and (v) $\bullet(X_0, Y_0) \overset{e}{\hookrightarrow} Z_0$.

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. 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 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.

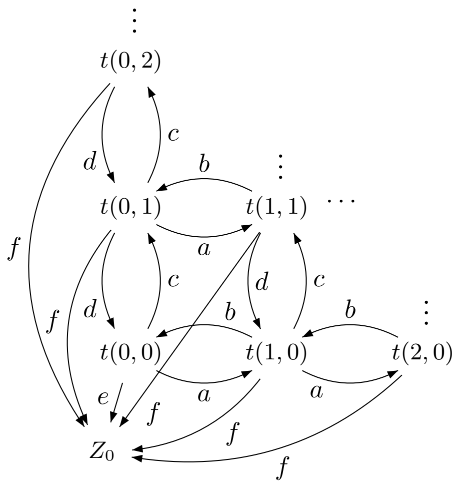

Fig. 4. The transition system $\mathcal{T}(\mathcal{R})$.

It is easy to see that the set of maximal sequences executable from $t(0,0)$ is not a context-free language. 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 call a term $\alpha \in \mathbb{G}(\Sigma)$ *inactive* if $\alpha \not\overset{\sigma}{\Longrightarrow}$ for all $\sigma \in \mathbb{A}$. We note that $\alpha \overset{\tau}{\Longrightarrow}$ might be possible even though α is inactive.

Lemma 19. *Assume some term 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.*

Theorem 20. GTRS $\not\leq_\approx$ PAD.

Proof (sketch). The proof idea is to show that any PAD that satisfies the property of Lemma 19 is already weakly bisimilar to a pushdown process.

5.3 PDS $\not\leq_{\approx}$ PAN and PN $\not\leq_{\approx}$ GTRS

Theorem 21. PDS $\not\leq_{\approx}$ PAN.

Proof (sketch). The proof idea is an adaption of an idea from [18] separating PAN from PDS with respect to strong bisimulation, but is technically more involved. The separating pushdown process 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 an e. (2) The action d, then the reverse of w and finally an f. $\square$

Theorem 22. PN $\not\leq_{\approx}$ 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 [22]), while this language is not a trace language of GTRS (e.g. see [15]).

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 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 (e.g. see [13, 23]) is the extension of Hennessy-Milner logic with reachability operators (possibly parameterized over subsets of all possible actions). The decidability of EF model checking over GTRS has been known for a long time, e.g., it follows from the results of [8, 12]. Together with Theorem 1, this easily gives another proof of the following result of Mayr.

Theorem 23 ([19]). *Model checking EF-logic over* PAD *is decidable.*

The second application is the decidability/complexity of model checking the common fragments LTL$_{det}$ (called deterministic LTL) and LTL$(\mathbf{F}_s, \mathbf{G}_s)$ [7, 17] 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 [23, 24]; decidability with no upper bounds was initially proven in [7].

Theorem 24. *Model checking LTL$_{det}$ over* PAD *is decidable in exponential time 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.*

Acknowledgment: We thank Taolue Chen, Matthew Hague, Jean Christoph Jung, Richard Mayr, Colin Stirling, and anonymous reviewers for their comments. Anthony W. Lin is supported by EPSRC (EP/H026878/1).

References

1. F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
2. J. C. M. Baeten, J. A. Bergstra, and J. W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. In J. W. de Bakker, A. J. Nijman, and P. C. Treleaven, editors, *PARLE (2)*, volume 259 of *Lecture Notes in Computer Science*, pages 94–111. Springer, 1987.
3. J. A. Bergstra and J. W. Klop. Algebra of communicating processes with abstraction. *Theor. Comput. Sci.*, 37:77–121, 1985.
4. A. Bouajjani, R. Echahed, and P. Habermehl. On the verification problem of nonregular properties for nonregular processes. In *LICS*, pages 123–133. IEEE Computer Society, 1995.
5. A. Bouajjani, M. Müller-Olm, and T. Touili. Regular symbolic analysis of dynamic networks of pushdown systems. In M. Abadi and L. de Alfaro, editors, *CONCUR*, volume 3653 of *Lecture Notes in Computer Science*, pages 473–487. Springer, 2005.
6. A. Bouajjani and T. Touili. Reachability analysis of process rewrite systems. In P. K. Pandya and J. Radhakrishnan, editors, *FSTTCS*, volume 2914 of *Lecture Notes in Computer Science*, pages 74–87. Springer, 2003.
7. L. Bozzelli, M. Kretínský, V. Rehák, and J. Strejcek. On decidability of LTL model checking for process rewrite systems. *Acta Inf.*, 46(1):1–28, 2009.
8. W. S. Brainerd. Tree generating regular systems. *Information and Control*, 14(2):217–231, 1969.
9. O. Burkart, D. Caucal, F. Moller, and B. Steffen. Verification on infinite structures. In *Handbook of process algebra*, pages 545–623. Elsevier, North-Holland, 2001. Chapter 9.
10. S. Christensen. *Decidability and Decomposition in Process Algebras*. PhD thesis, Department of Computer Science, The University of Edinburgh, 1993.
11. J.-L. Coquidé, M. Dauchet, R. Gilleron, and S. Vágvölgyi. Bottom-up tree pushdown automata: Classification and connection with rewrite systems. *Theor. Comput. Sci.*, 127(1):69–98, 1994.
12. M. Dauchet and S. Tison. The theory of ground rewrite systems is decidable. In *LICS*, pages 242–248. IEEE Computer Society, 1990.
13. S. Göller and A. W. Lin. The Complexity of Verifying Ground Tree Rewrite Systems. In *LICS*. IEEE Computer Society, 2011. to appear.
14. M. H. T. Hack. *Decidability Questions for Petri Nets*. PhD thesis, MIT, 1976.
15. C. Löding. *Infinite Graphs Generated by Tree Rewriting*. PhD thesis, RWTH Aachen, 2003.
16. D. Lugiez and P. Schnoebelen. The regular viewpoint on pa-processes. *Theor. Comput. Sci.*, 274(1-2):89–115, 2002.
17. M. Maidl. The common fragment of CTL and LTL. In *FOCS*, pages 643–652, 2000.
18. R. Mayr. Process rewrite systems. *Inf. Comput.*, 156(1-2):264–286, 2000.
19. R. Mayr. Decidability of model checking with the temporal logic ef. *Theor. Comput. Sci.*, 256(1-2):31–62, 2001.
20. R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
21. D. E. Muller and P. E. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theor. Comput. Sci.*, 37:51–75, 1985.
22. W. Thomas. Applied automata theory. Course Notes (RWTH Aachen), 2005.
23. A. W. To. *Model Checking Infinite-State Systems: Generic and Specific Approaches*. PhD thesis, LFCS, School of Informatics, University of Edinburgh, 2010.
24. A. W. To and L. Libkin. Algorithmic metatheorems for decidable LTL model checking over infinite systems. In C.-H. L. Ong, editor, *FOSSACS*, volume 6014 of *Lecture Notes in Computer Science*, pages 221–236. Springer, 2010.
25. R. J. van Glabbeek and W. P. Weijland. Branching time and abstraction in bisimulation semantics. *J. ACM*, 43(3):555–600, 1996.