FI MU

Faculty of Informatics
Masaryk University

Process Rewrite Systems with Weak Finite-State Unit

by

Mojmír Křetínský
Vojtěch Řehák
Jan Strejček

FI MU Report Series FIMU-RS-2003-05

 September 2003

Process Rewrite Systems with Weak Finite-State Unit*†

Mojmír Křetínský Vojtěch Řehák Jan Strejček

Faculty of Informatics
Masaryk University
Botanická 68a, 602 00 Brno
Czech Republic
{kretinsky,rehak,strejcek}@fi.muni.cz

September 26, 2003

Abstract

Various classes of infinite-state processes are often specified by rewrite systems. We extend Mayr's Process Rewrite Systems (PRS) [12] by finite-state unit whose transition function satisfies some restrictions inspired by weak finite automata. We classify these models by their expressiveness and show how the hierarchy of new classes (w.r.t. bisimilarity) is related to both PRS hierarchy of Mayr and two other hierarchies of PRS extensions introduced in [9, 21].

1 Introduction

As the state-space infiniteness of concurrent systems has a various, real-life sources (e.g. data manipulation, asynchronous communication, etc.), a motivation is to provide adequate representations of concurrent systems as well as to study the possibilities of their formal verification. Concurrent systems can be modelled in a number of ways (e.g. Process Algebras, Petri Nets, etc.), however a unifying view is to interpret them as labelled

*This work has been partially supported by GAČR, grant No. 201/03/1161.

†This is a full version of INFINITY'03 paper.

transition systems (LTS) with possibly infinite number of states. LTS families are often specified via a variety of rewrite systems and form hierarchies (w.r.t. bisimulation equivalence), see for example [5, 3, 14, 12]. Here we employ the classes of infinite-state systems defined by Process Rewrite Systems (PRS, introduced by Mayr in [12]) as they contain a variety of the formalisms studied in the context of formal verification. For surveys of formal verification techniques and results see for example [14, 4, 2, 10, 19].

It is possible to extend rewriting mechanisms by a finite-state unit [14, 9]. However this extension is very powerful as the state-extended version of PA processes (i.e. the state-extended (1,G)-PRS) has a full Turing-power [1], while unrestricted PRS is not Turing-powerful [12]. One of motivations for introducing state-extended PRS classes can be seen as follows: it is required to specify some process classes which are not explicitly present in PRS hierarchy. This can be exemplified by the class of so called Parallel Push-Down Automata (PPDA) introduced by Moller as a state-extended version of BPP in [14]. Please note BPP forms a proper subclass of PPDA which is properly contained in Petri nets [15]. We also note the problem of bisimulation equivalence on BPP is decidable [6] (the recent results [20, 8] show it is PSPACE-complete), while the same problem on their state-extended version is undecidable [14].

In this paper we aim at weakening the strength of state-extension by putting some restrictions on (the transition function of) finite-state unit – we use weak finite automaton as introduced in [16], but used here as a nondeterministic (NFA) rather than alternating one. A NFA $A = (Q, \Sigma, \delta, q_0, F)$ is *weak* if its state space is partitioned into a disjoint union $Q = \bigcup Q_i$, and there is a partial order $\geq$ on the collection of the Q_i. The transition function $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ is such that if $q \in Q_i$ and $q' \in \delta(q, a)$ then $q' \in Q_j$, where $Q_i \geq Q_j$. The set F of final states will not play any role in this paper, however recall it is required that $Q_i \subseteq F$ or $Q_i \cap F = \emptyset$ for each Q_i. Due to their connections to (modal) logics weak (nondeterministic, alternating,...) automata have been used in several contexts and in fact their slightly modified variants have been employed. For example we refer to [11] where the common fragment of a linear time logic LTL [17] and a branching time logic ACTL (which is a fragment of CTL [7]) is given by exactly those LTL formulae the negation of which can be represented by 1-weak Büchi automaton (automaton is 1-weak if each partition block contains exactly one state). We mention the 1-weak variant only and skip the others as this will serve as a suitable abstraction used in our definition of PRS with weak unit.

In state-extended PRS a finite-state unit keeps a sort of global information accessible to all parallel (ready to be reduced) components of a PRS term. For example different (sub)sets of rewriting rules can be applied depending on the current state of unit. Elsewhere [21] one of the authors of this paper enriched (pure) PRS by 'monotonically evolving' unit and showed the introduced fcPRS classes ('fc' standing for 'finitely constrained' – see Section 3) are strictly more expressible than the respective classes of PRS provided the respective PRS class does not subsume some notion of 'control state' as e.g. PDA or Petri Nets do.

Some basic definitions and properties of PRS (taken from[12]) and fcPRS ([21]) are recalled in Sections 2 and 3. We introduce PRS with weak unit (wPRS) and mention state-extended PRS in Section 4. Some relationships between the respective new classes and the already existing ones as well as the (combined) hierarchy of PRS and (relevant classes of) extended PRSs are shown in Section 5.

2 Process rewrite systems (PRS)

A *labelled transition system (LTS)* $\mathcal{L}$ is a tuple $(S, Act, \longrightarrow, \alpha_0)$, where S is a set of *states* or *processes*, *Act* is a set of *atomic actions* or *labels*, $\longrightarrow \subseteq S \times Act \times S$ is a *transition relation* (written $\alpha \stackrel{a}{\longrightarrow} \beta$ instead of $(\alpha, a, \beta) \in \longrightarrow$), $\alpha_0 \in S$ is a distinguished *initial state*. A state $\alpha \in S$ is *terminal* (or *deadlocked*, written $\alpha \not\longrightarrow$) if there is no $a \in Act$ and $\beta \in S$ such that $\alpha \stackrel{a}{\longrightarrow} \beta$. We also use the natural generalization $\alpha \stackrel{\sigma}{\longrightarrow} \beta$ for finite sequences of actions $\sigma \in Act^*$. The state α is *reachable* if there is $\sigma \in Act^*$ such that $\alpha_0 \stackrel{\sigma}{\longrightarrow} \alpha$.

A binary relation R on set of states S is a *bisimulation* [13] iff for each $(\alpha, \beta) \in R$ the following conditions hold:

- $\forall\, \alpha' \in S, a \in Act : \alpha \stackrel{a}{\longrightarrow} \alpha' \Longrightarrow (\exists \beta' \in S : \beta \stackrel{a}{\longrightarrow} \beta' \wedge (\alpha', \beta') \in R)$
- $\forall\, \beta' \in S, a \in Act : \beta \stackrel{a}{\longrightarrow} \beta' \Longrightarrow (\exists \alpha' \in S : \alpha \stackrel{a}{\longrightarrow} \alpha' \wedge (\alpha', \beta') \in R)$

Bisimulation equivalence on an assumed LTS is the maximal bisimulation (i.e. union of all bisimulations).

Let $Const = \{X, \ldots\}$ be a countably infinite set of *process constants*. The set $\mathcal{T}$ of *process terms* (ranged over by $t, \ldots$) is defined by the abstract syntax

$$t = \varepsilon \mid X \mid t_1.t_2 \mid t_1 \| t_2,$$

where ε is the empty term, $X \in Const$ is a process constant (used as an atomic process), '$\|$' and '.' mean parallel and sequential compositions respectively. The set $Const(t)$ is the set of all constants occurring in a process term t. We always work with equivalence classes of terms modulo commutativity and associativity of '$\|$' and modulo associativity of '.' We also define $\varepsilon.t = t = t.\varepsilon$ and $t\|\varepsilon = t$.

We distinguish four *classes of process terms*: '1' stands for terms consisting of a single process constant only (i.e. $\varepsilon \notin \mathbf{1}$), 'S' are *sequential* terms – without parallel composition, 'P' are *parallel* terms - without sequential composition, 'G' are *general* terms with arbitrarily nested sequential and parallel compositions.

Definition 2.1. *Let* $Act = \{a, b, \cdots\}$ *be a countably infinite set of* atomic actions, $\alpha, \beta \in \{1, S, P, G\}$ *such that* $\alpha \subseteq \beta$. *An* (α, β)-PRS (process rewrite system) *is a pair* $\Delta = (R, t_0)$*, where*

- R *is a finite set of* rewrite rules *of the form* $t_1 \xrightarrow{a} t_2$*, where* $t_1 \in \alpha$*,* $t_1 \neq \varepsilon$*,* $t_2 \in \beta$ *are process terms and* $a \in Act$ *is an atomic action,*
- $t_0 \in \beta$ *is an* initial state.

Unless stated otherwise we assume a given Δ as in Definition 2.1. We define $Const(\Delta)$ as the set of all constants occurring in the rewrite rules of Δ or in its initial state, and $Act(\Delta)$ as the set of all actions occurring in the rewrite rules of Δ. We sometimes write $(t_1 \xrightarrow{a} t_2) \in \Delta$ instead of $(t_1 \xrightarrow{a} t_2) \in R$.

The semantics of Δ is given by the LTS $(S, Act(\Delta), \longrightarrow, t_0)$, where $S = \{t \in \beta \mid Const(t) \subseteq Const(\Delta)\}$ and $\longrightarrow$ is the least relation satisfying the inference rules:

$$\frac{(t_1 \xrightarrow{a} t_2) \in \Delta}{t_1 \xrightarrow{a} t_2}, \qquad \frac{t_1 \xrightarrow{a} t_1'}{t_1\|t_2 \xrightarrow{a} t_1'\|t_2}, \qquad \frac{t_1 \xrightarrow{a} t_1'}{t_1.t_2 \xrightarrow{a} t_1'.t_2}.$$

If no confusion can arise, we sometimes speak about "process rewrite system" meaning "labelled transition system generated by process rewrite system".

The *PRS-hierarchy* of (α, β)-PRS is depicted as a sub-hierarchy in Figure 1. Some classes included in the hierarchy correspond to widely known models as Finite State systems (FS), Basic Process Algebras (BPA), Basic Parallel Processes (BPP), Process Algebras (PA), Push-Down Processes (PDA, see [5] for justification), and Petri Nets (PN). The other three classes were introduced (and named) by Mayr [12]. The relationship between the class name and its (α, β)-PRS counter-part is given in Figure 1 as well.

PRS-hierarchy is not strict w.r.t. language equivalence (e.g. both BPA and PDA define the class of ε-free context-free languages). The strictness of the hierarchy w.r.t. bisimularity follows from the results presented (or cited) in [3, 14] and from the following two examples [12].

Example 2.2. *A PDA system with the initial state $U.X$ which is not bisimilar to any PAN system (for proof see [12]).*

$$
\begin{array}{lll}
U.X \xrightarrow{a} U.A.X & U.A \xrightarrow{a} U.A.A & U.B \xrightarrow{a} U.A.B \\
U.X \xrightarrow{b} U.B.X & U.A \xrightarrow{b} U.B.A & U.B \xrightarrow{b} U.B.B \\
U.X \xrightarrow{c} V.X & U.A \xrightarrow{c} V.A & U.B \xrightarrow{c} V.B \\
U.X \xrightarrow{d} W.X & U.A \xrightarrow{d} W.A & U.B \xrightarrow{d} W.B \\
V.X \xrightarrow{e} V & V.A \xrightarrow{a} V & V.B \xrightarrow{b} V \\
W.X \xrightarrow{f} W & W.A \xrightarrow{a} W & W.B \xrightarrow{b} W
\end{array}
$$

Example 2.3. *A Petri net given as (P, P)-PRS with the initial state $X\|A\|B$ which is not bisimilar to any PAD process (for proof see [12]).*

$$
\begin{array}{llll}
X \xrightarrow{g} X\|A\|B & Y\|A \xrightarrow{a} Y & X\|A \xrightarrow{d} Z & Y\|A \xrightarrow{d} Z \\
X \xrightarrow{c} Y & Y\|B \xrightarrow{b} Y & X\|B \xrightarrow{d} Z & Y\|B \xrightarrow{d} Z
\end{array}
$$

3 PRS with finite constraint systems (fcPRS)

In this section we recall the extension of process rewrite systems with finite constraint systems. This extension has been directly motivated by constraint systems used in *concurrent constraint programming* (CCP) – see e.g. [18]. A *constraint system* is a bounded lattice $(C, \vdash, \wedge, tt, ff)$, where C is the set of *constraints*, $\vdash$ (called *entailment*) is an ordering on this set, $\wedge$ is the lub operation, and tt (*true*), ff (*false*) are the least and the greatest elements of C respectively ($ff \vdash tt$ and $tt \neq ff$). The constraint system describes a state space and possible evolution of a unit called *store*.

Definition 3.1. *Let $\alpha, \beta \in \{1, S, P, G\}$ such that $\alpha \subseteq \beta$. An (α, β)-*fcPRS (PRS with finite constraint system) *is a tuple $\Delta = (\mathcal{C}, R, t_0)$, where*

- *$\mathcal{C} = (C, \vdash, \wedge, tt, ff)$ is a finite constraint system describing the* store*; the elements of C represent* values of the store,
- *R is a finite set of* rewrite rules *of the form $(t_1 \xrightarrow{a} t_2, m, n)$, where $t_1 \in \alpha$, $t_1 \neq \varepsilon$, $t_2 \in \beta$ are process terms, $a \in Act$ is an atomic action, and $m, n \in C$ are constraints,*
- *$t_0 \in \beta$ is a distinguished* initial process term.

The semantics of an (α,β)-fcPRS system $\Delta = (\mathcal{C}, R, t_0)$ is given by the LTS $(S, Act(\Delta), \longrightarrow, (t_0, tt))$, where $S = \{t \in \beta \mid Const(t) \subseteq Const(\Delta)\} \times (C \setminus \{\mathit{ff}\})$ and $\longrightarrow$ is the least relation satisfying the inference rules:

$$\frac{(t_1 \stackrel{a}{\longrightarrow} t_2, m, n) \in \Delta}{(t_1, o) \stackrel{a}{\longrightarrow} (t_2, o \wedge n)} \quad \text{if } o \vdash m \text{ and } o \wedge n \neq \mathit{ff},$$

$$\frac{(t_1, o) \stackrel{a}{\longrightarrow} (t_1', p)}{(t_1 \| t_2, o) \stackrel{a}{\longrightarrow} (t_1' \| t_2, p)}, \qquad \frac{(t_1, o) \stackrel{a}{\longrightarrow} (t_1', p)}{(t_1.t_2, o) \stackrel{a}{\longrightarrow} (t_1'.t_2, p)}.$$

The two side conditions in the first inference rule are very close to principles used in CCP. The first one ($o \vdash m$) ensures the rule $(t_1 \stackrel{a}{\longrightarrow} t_2, m, n) \in \Delta$ can be used only if the current value of the store o *entails* m (it is similar to *ask(m)* in CCP). The second condition ($o \wedge n \neq \mathit{ff}$) guarantees that the store stays *consistent* after application of the rule (analogous to the consistency requirement when processing *tell(n)* in CCP).

An important observation is that the value of a store can move in a lattice only upwards as $o \wedge n$ always entails o. Intuitively, partial information can only be added to the store, but never retracted (the store is *monotonic*).

Also note that an execution of a transition which starts in a state with o on the store and which is generated by a rule $(t_1 \stackrel{a}{\longrightarrow} t_2, m, n) \in \Delta$ implies that for every subsequent value of the store p the conditions $p \vdash m$ and $p \wedge n \neq \mathit{ff}$ are satisfied (and thus the use of the rule cannot be forbidden by a value of the store in future). The first condition $p \vdash m$ comes from the monotonic behaviour of the store. The second condition comes from the facts that the constraint n in the rule can change the store only in the first application of the rule and that for each subsequent state p of the store $p \wedge n = p$ holds.

4 PRS with weak finite-state unit (wPRS)

Definition 4.1. *Let $\alpha, \beta \in \{1, S, P, G\}$ such that $\alpha \subseteq \beta$. An (α,β)-*wPRS (PRS with weak finite-state unit) *is a tuple $\Delta = (Q, \geq, R, m_0, t_0)$, where*

- $(Q, \geq)$ *is partially ordered finite set representing states of* weak finite-state unit*; the elements of Q are called* w-states,
- *R is a finite set of* rewrite rules *of the form $mt_1 \stackrel{a}{\longrightarrow} nt_2$ satisfying the condition $m \geq n$, where $m, n \in Q$, $t_1 \in \alpha$, $t_1 \neq \varepsilon$, $t_2 \in \beta$, and $a \in Act$,*
- *$m_0 \in Q$, $t_0 \in \beta$, and m_0t_0 is the* initial state *of the system.*

The semantics of an (α,β)-wPRS $\Delta = (Q, \geq, R, m_0, t_0)$ is given by the LTS $(S, Act(\Delta), \longrightarrow, m_0t_0)$, where $S = \{mt \mid m \in Q, t \in \beta, Const(t) \subseteq Const(\Delta)\}$ and $\longrightarrow$ is defined as the least relation satisfying the inference rules:

$$\frac{(mt_1 \xrightarrow{a} nt_2) \in \Delta}{mt_1 \xrightarrow{a} nt_2}, \qquad \frac{mt_1 \xrightarrow{a} nt_1'}{m(t_1 \| t_2) \xrightarrow{a} n(t_1' \| t_2)}, \qquad \frac{mt_1 \xrightarrow{a} nt_1'}{m(t_1.t_2) \xrightarrow{a} n(t_1'.t_2)}.$$

The presented notion of weakness corresponds to 1-weakness condition in automata theory (mentioned already in Section 1; the general weak unit without considering final states would coincide with standard state-extension as all the states of Q could be included in one partition block). In any transition sequence the w-state components of visited states form a non-increasing sequence w.r.t. $\geq$ (i.e. can only change finitely many times). However, in contrast to fcPRS, the weak unit can forbid the application of any rewrite rule.

State Extended PRS If we relax from the condition $m \geq n$ imposed on rewrite rules in Definition 4.1, we get the definition of state-extended (α,β)-PRS (denoted by prefix 'se'). Instead of $(1,S)$-sePRS, $(1,P)$-sePRS, ... we also use more traditional abbreviations seBPA, seBPP, ... and we also take up this notation for all the classes of both fcPRS and wPRS introduced earlier.

We remind a motivation of introducing sePRS given in Section 1. Of course, seBPA and seBPP coincides with PDA and PPDA respectively. Also recall that (S,S)-PRS and PDA are equivalent w.r.t. bisimilarity as shown by Caucal [5], while seBPP has no bisimulation equivalent counter-part within PRS hierarchy (it is strictly under (P,P)-PRS as shown by Moller in [15]).

5 Relations between classes, refining hierarchy

We start with some very obvious observations. Given process classes X, Y the notation $X \subseteq Y$ means that every LTS definable in class X can be defined (up to bisimulation equivalence) also in class Y. We say Y is at least as expressive as X.

An immediate observation is the classes FS, PDA and PN have the same expressiveness as the corresponding $\{\text{fc}, \text{w}, \text{se}\}$ extended classes. We also note that

$$(\alpha,\beta)\text{-PRS} \subseteq (\alpha,\beta)\text{-fcPRS} \subseteq (\alpha,\beta)\text{-wPRS} \subseteq (\alpha,\beta)\text{-sePRS}$$

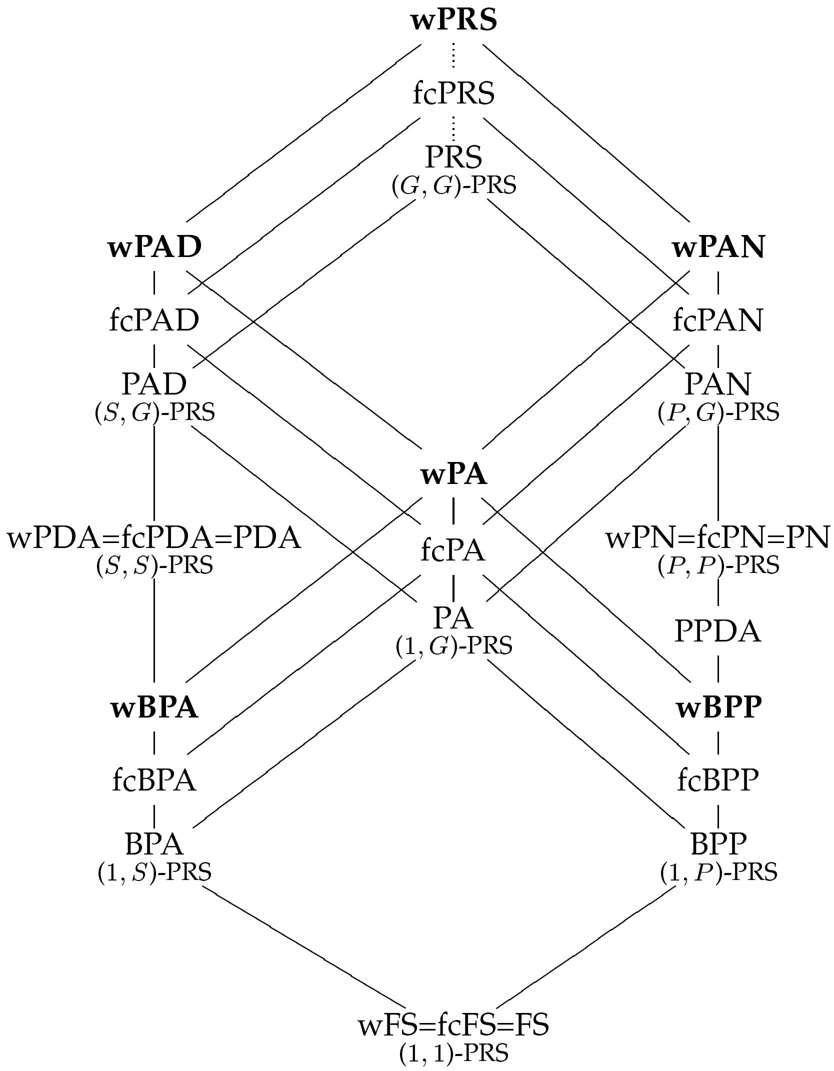

Figure 1: The hierarchy of classes defined by (extended) rewrite formalisms

hold for every (α, β)-PRS class (even up to isomorphism). For the second inclusion take an arbitrary (α, β)-fcPRS Δ with an initial term t_0 and a constraint system $\mathcal{C} = (C, \vdash, \wedge, tt, ff)$. The corresponding (α, β)-wPRS is $\Delta' = (C \smallsetminus \{ff\}, \geq, R', tt, t_0)$, where $\geq = (\vdash)^{-1}$ and $R' = \{o t_1 \xrightarrow{a} (o \wedge n) t_2 \mid (t_1 \xrightarrow{a} t_2, m, n) \in \Delta$ and $o \vdash m$ and $o \wedge n \neq ff\}$. The first and third inclusions are obvious as well.

Relations between the classes of PRS-hierarchy, the corresponding classes extended with finite constraint systems or weak state unit, and the PPDA class (the only class of Moller's hierarchy not covered by previous formalisms) are depicted in Figure 1. The shape of the hierarchy follows from the definitions of included classes and from the relations between considered extensions. The strictness of the PRS-hierarchy (w.r.t. bisimulation equivalence) has been proved by Mayr [12]. The strictness of the hierarchy covering classes from PRS-hierarchy and corresponding classes with finite constraint systems has been proved in [21] (with one exception – the strictness between PRS and fcPRS is just a conjecture).

In the rest of this section we show that

- there is a PDA system which is not bisimilar to any wPAN system,
- there is a PPDA system which is not bisimilar to any wPAD system,
- there is a wBPP system which is not bisimilar to any fcPAD system (this proof formulates a property that is a sufficient condition for a PAD to be bisimilar to some PDA),
- there is a wBPA system which is not bisimilar to any fcPAN system.

These proofs together with the fact that PPDA are strictly less expressive than PN [15] will finish the proof of the strictness of our hierarchy (with two exceptions – the strictness of the relations between PRS, fcPRS, and wPRS remains unproved, although we conjecture the existence of the separating gaps).

5.1 PDA non-bisimilar to wPAN

Example 5.1. *Let us consider a PDA system of Example 2.2 but having $U.X.Y$ as the initial state and two more rewrite rules: $V.Y \xrightarrow{x} U.X.Y$, $W.Y \xrightarrow{x} U.X.Y$. This system, denoted by Δ_1, behaves like that defined in Example 2.2, but whenever the original system terminates, the enhanced Δ_1 is restarted under the action x.*

Lemma 5.2. *There is no wPAN Δ bisimilar to the PDA Δ_1 of Example 5.1.*

Proof. To derive a contradiction assume a wPAN Δ bisimilar to the PDA Δ_1. As the weak state unit of Δ is finite then there exists a reachable state mt of Δ such that every state reachable from mt has also m as its w-state component (the opposite would imply the infiniteness of the weak state unit). There exists a word $w \in \{a, b, c, d, e, f\}^*$ such that $mt \xrightarrow{w.x} mt'$, where mt' is bisimilar to the state $U.X.Y$ of the PDA process Δ_1. If the rules labelled by the action x are removed from Δ and mt' is taken as the initial state, we obtain the system whose all reachable states have m as w-state component and which is bisimilar to the pushdown process of Example 2.2.

Now let Δ' be a PAN system with the initial state t' and with the set of rewrite rules consisting of the rules $l \xrightarrow{v} r$, where $(ml \xrightarrow{v} mr) \in \Delta$ and

$v \in \{a, b, c, d, e, f\}$. It is obvious that this PAN system Δ' is bisimilar to the PDA system defined in Example 2.2 – a contradiction. $\square$

5.2 PPDA non-bisimilar to wPAD

Example 5.3. *Let Δ be a PPDA process with the initial state $xA\|B\|C$ and the following rewrite rules:*

$$\begin{array}{lll}
xC \xrightarrow{g} xA\|B\|C & xA \xrightarrow{d} z\varepsilon & zA \xrightarrow{q} z\varepsilon \\
xC \xrightarrow{c} yC & xB \xrightarrow{d} z\varepsilon & zB \xrightarrow{q} z\varepsilon \\
yA \xrightarrow{a} y\varepsilon & yA \xrightarrow{d} z\varepsilon & zC \xrightarrow{r} xA\|B\|C \\
yB \xrightarrow{b} y\varepsilon & yB \xrightarrow{d} z\varepsilon &
\end{array}$$

The rules labelled by g, c, a, b, d correspond to the rules of the Petri net given in Example 2.3. Hence the PPDA Δ behaves as the mentioned Petri net, but when the Petri net terminates, the PPDA Δ can remove an arbitrary number of A and B symbols from the parallel stack and then "restart" the system under action r.

Lemma 5.4. *There is no wPAD Δ' bisimilar to the PPDA Δ of Example 5.3.*

The proof is similar to the proof of Lemma 5.2 using the fact the Petri net of Example 2.3 is not bisimilar to any PAD system.

5.3 wBPP non-bisimilar to fcPAD

A rewriting system is *deadlockable* if for each reachable nonterminal state s (of its underlying LTS) there is a transition from s to a terminal state, i.e. $\exists\, a, t : s \xrightarrow{a} t \not\longrightarrow$.

Definition 5.5. *A sequential subterm t (i.e. $t \in S$) of term $g \in G$ is a* ready parallel component *iff t is a maximal subtree in the syntax tree of term g such that t is not in the right-hand side subtree of any sequential node (i.e. node corresponding to sequential operator). A ready parallel component t is* live *in a PAD system Δ if t is not deadlocked (i.e. there is a rule applicable to t).*

Intuitively the ready parallel components are defined as the maximal sequential parts of a PAD process g such that g can perform an action a if and only if some of its ready parallel components can perform the same action a.

Lemma 5.6. *Every reachable state of an arbitrary deadlockable PAD system has at most one live ready parallel component.*

Proof. Observe that the application of a PAD rewrite rule can only modify one ready parallel component. Hence there is no way how to deadlock more than one live ready parallel component by one application of a PAD rewrite rule. □

Lemma 5.7. *Every deadlockable PAD system is bisimilar to a PDA system.*

Proof. An idea is to transform PAD rewrite rules onto corresponding PDA rewrite rules (this is sufficient as for every PAD there is a bisimilar PAD system with a single process constant as the initial process term).

There is only one way to revive a deadlocked parallel component, namely to rewrite adjacent components onto ε. For example if $B.C$ is a deadlocked ready parallel component of $(A.C \| B.C).D$ and $(A.C \| B.C).D \stackrel{w}{\longrightarrow} (\varepsilon \| B.C).D = B.C.D$ then the ready parallel component $B.C.D$ can be live.

Let Δ be a PAD system and $X \notin Const(\Delta)$ be a fresh process constant. Let us consider a rewrite rule of Δ with the right hand side containing a maximal subterm of the form $l.(t_1 \| t_2).r$, where $t_1, t_2 \in S$ and l, r can be ε. In an arbitrary transition sequence the components t_1, t_2 generated by the application of the considered rewrite rule become ready at the same time. Thus at least one of them is deadlocked. Let t_2 be deadlocked. We replace the subterm $l.(t_1 \| t_2).r$ of the rule by $l.X.t_1.X.t_2.r$ (or just $t_1.X.t_2.r$ whenever l is ε). The process constant X eliminates any possible (unwanted) interaction of (the tail of) the term l and (the beginning of) the term t_1 (or the tail of t_1 and the beginning of t_2 respectively). Repeating this procedure eliminates all parallel operators from rewrite rules. The resulting PDA system Δ' enriched by rewrite rules of the form $X.s \stackrel{a}{\longrightarrow} s'$ for every rule $s \stackrel{a}{\longrightarrow} s' \in \Delta'$ is bisimilar to a given Δ. □

Example 5.8. *Let Δ_2 be the wBPP system with the initial state pX and the rules:*

$$pX \stackrel{c}{\longrightarrow} pX \| A \| B \qquad pA \stackrel{a}{\longrightarrow} p\varepsilon \qquad pB \stackrel{b}{\longrightarrow} p\varepsilon \qquad pX \stackrel{d}{\longrightarrow} q\varepsilon$$

Lemma 5.9. *There is no PAD system bisimilar to the wBPP system Δ_2 of Example 5.8.*

Proof. Δ_2 is deadlockable. Due to Lemma 5.7 it suffices to prove there is no PDA system bisimilar to Δ_2. This directly follows from the fact that the language L generated by Δ_2 is not context-free ($L \cap c^*a^*b^*d = \{c^k a^l b^m d \mid 0 \leq l, m \leq k\}$ is not a context-free language). □

Lemma 5.10. *There is no fcPAD system bisimilar to the wBPP system Δ_2 of Example 5.8.*

Proof. For the sake of a contradiction we assume a fcPAD Δ bisimilar to Δ_2.

The finiteness of the constraint system used in Δ implies that there exists a reachable non-terminal state (t, m) of Δ such that every non-terminal state reachable from (t, m) has also m on the store (the contrary would mean the constraint system is infinite). As (t, m) is non-terminal there exists a word $w \in \{a, b\}^*$ such that $(t, m) \xrightarrow{w} (s, m)$ and (s, m) is bisimilar to the initial state pX of Δ_2. The only transitions starting at states reachable from (s, m) and changing the value of the store can be the transitions leading to terminal states, i.e. the transitions labelled by d. Hence we can directly assume that all rewrite rules of Δ labelled with $x \in \{a, b, c\}$ have the form $(t_1 \xrightarrow{x} t_2, tt, tt)$.

Let Δ' be a PAD system with the set of rewrite rules as

$$\begin{aligned} &\{t_1 \xrightarrow{x} t_2 \mid (t_1 \xrightarrow{x} t_2, tt, tt) \in \Delta, x \neq d\} \cup \\ \cup \quad &\{t_1 \xrightarrow{d} Z \mid (t_1 \xrightarrow{d} t_2, tt, n) \in \Delta, n \neq \mathit{ff}\}, \end{aligned}$$

where $Z \notin Const(\Delta)$ is a fresh process constant. If we restrict the systems Δ and Δ' to actions a, b, c then Δ and Δ' are bisimilar. Furthermore in every state q of Δ' reachable under $w \in \{a, b, c\}^*$ there is atransition labelled by d and starting at q. It suffices to show that this transition is leading to a terminal state.

The state (q, tt) (corresponding to the state q) has a ready parallel component able to perform an action c. This action cannot be disabled by any action performed by another ready parallel component. Hence there is just one ready parallel component able to perform both c and d. For the same reason this component is the only one which is able to perform actions a and b if they are enabled in the state (q, tt). The same holds for the state q of Δ'. Moreover the ready parallel component rewritten by the action d is deadlocked by the process constant Z. Thus the state reached under d is terminal and we get a PAD system Δ' bisimilar to the wBPP Δ_2 of Example 5.8 – a contradiction (see Lemma 5.9). □

5.4 wBPA non-bisimilar to fcPAN

Definition 5.11. *A parallel subterm t ($t \in P \setminus \{\varepsilon\}$) of term $g \in G$ is a* ready sequential component *iff t is a maximal subtree in the syntax tree of term g such that t does not occur in any right sequential component of g.*

A ready sequential component t of term g is called non-trivial *iff $\exists t' \neq \varepsilon$ such that $t.t'$ is a subterm of g (i.e. t is located in subterm $t.t'$ of g, where $t' \neq \varepsilon$).*

Intuitively, ready sequential components are defined as maximal parallel subterms of a fcPAN process g such that g can perform an action a if and only if some of its ready sequential components can perform the action a.

Definition 5.12. *Let $\mathcal{L} = (S, Act, \longrightarrow, \alpha_0)$ be a labelled transition system, $\alpha \in S$, $\Sigma \subseteq Act$, $u \in \Sigma^*$. The LTS $\mathcal{L}|_\Sigma = (S, \Sigma, \longrightarrow \cap (S \times \Sigma \times S), \alpha_0)$ is the restriction of $\mathcal{L}$ to Σ. We use the notation $only^\Sigma(\alpha, u)$ iff u is a prefix of each maximal transition sequence in $\mathcal{L}|_\Sigma$ starting in α.*

Definition 5.13. *A sequential composition of t (i.e. the sequential operator '.' within t) is said to be* accessed *during a rewriting sequence from (t, m) under w iff the left subterm of this sequential composition is rewritten onto ε during the rewriting sequence.*

A sequential composition of t is called accessible *from (t, m) under w iff there is a rewriting sequence from (t, m) under w such that the sequential composition is accessed during this rewriting sequence; otherwise it is called* inaccessible.

Lemma 5.14. *Given arbitrary $i \in \mathbb{N}$, fcPAN Δ with set of constraints C, $a, b \in Act(\Delta)$ ($a \neq b$), and state (t, m) of Δ satisfying $only^{\{a,b\}}((t, m), a^i b)$, it holds that during each rewriting sequence from (t, m) under $a^i b$ there are at most $|C|+1$ rewritten ready sequential components of t such that their sequential compositions are not accessed during the rewriting sequence.*

Proof. Let us suppose that there are more than $|C| + 1$ such components. Then there are at least two of them such that any action performed by these components during the sequence $a^i b$ does not change the store. At least one of them is not performing action b during the sequence. The actions performed by these components can be omitted without any effect on the rest of the considered rewriting sequence as the omitted actions do not change the store and the corresponding sequential compositions are not accessed. Hence, there is a rewriting sequence from (t, m) under $a^j b$ such that $j < i$. This is a contradiction with $only^{\{a,b\}}((t, m), a^i b)$. $\square$

Definition 5.15. *For every fcPAN Δ and every number $n \in \mathbb{N}_0$ we define $K_n(\Delta)$ to be a set of all n-tuples $(k_1, k_2, \ldots, k_n) \in \mathbb{N}^n$ such that for every $a, b \in Act(\Delta)$, $a \neq b$, every state (t, m) of Δ satisfying $only^{\{a,b\}}((t, m), a^{k_1} b a^{k_2} b \ldots a^{k_n} b)$, and every rewriting sequence from (t, m) under $a^{k_1} b a^{k_2} b \ldots a^{k_n} b$, t includes at least n sequential compositions accessed during the rewriting sequence.*

Intuitively, $K_n(\Delta)$ is a set of all n-tuples $(k_1, k_2, \dots, k_n)$ such that each state (t, m) of Δ holding the information that every run under $\{a, b\}^*$ starts with $a^{k_1}ba^{k_2}b \dots a^{k_n}b$ needs at least n sequential compositions.

Lemma 5.16. *For every fcPAN Δ and every number $n \in \mathbb{N}_0$, $K_n(\Delta) \neq \emptyset$.*

Proof. We prove the lemma by induction on n. The base $n = 0$ is easy to prove. As every state (t, m) and every actions a, b satisfy $only^{\{a,b\}}((t, m), \varepsilon)$ and every term includes zero sequential compositions accessed during an empty rewriting sequence, it holds that $() \in K_0(\Delta)$ for every fcPAN Δ.

Induction hypothesis: Let $n \in \mathbb{N}_0$ be such that $K_n(\Delta) \neq \emptyset$ for every fcPAN Δ. We assume the contrary for $n + 1$ and derive a contradiction.

Let Δ be a fcPAN such that for every $(k_1, \dots, k_n, k_{n+1}) \in \mathbb{N}^{n+1}$ there exist distinct actions $a, b \in Act(\Delta)$, a state (t, m) of Δ satisfying the condition $only^{\{a,b\}}((t, m), a^{k_1}b \dots a^{k_n}ba^{k_{n+1}}b)$, and a rewriting sequence from (t, m) under $a^{k_1}ba^{k_2}b \dots a^{k_n}ba^{k_{n+1}}b$ such that t includes at most n sequential compositions accessed during this rewriting sequence. Due to induction hypothesis, we can choose $k_1, \dots, k_n$ such that $(k_1, \dots, k_n) \in K_n(\Delta)$. Hence, for every $l \in \mathbb{N}$, there exist distinct actions $a_l, b_l \in Act(\Delta)$, a state (t_l, m_l) of the system Δ satisfying $only^{\{a_l,b_l\}}((t_l, m_l), a_l^{k_1}b_l \dots a_l^{k_n}b_la_l^lb_l)$, and a rewriting sequence

$$(t_l, m_l) \xrightarrow{a_l^{k_1}b_l \dots a_l^{k_n}b_l} (t'_l, m'_l) \xrightarrow{a_l^l b_l} (t''_l, m''_l)$$

such that t_l includes exactly n sequential compositions accessed during the rewriting under $a_l^{k_1}b_l \dots a_l^{k_n}b_l$ and no other sequential composition of t_l is accessed during the rewriting sequence from a state (t'_l, m'_l) under $a_l^lb_l$.

Let α be an infinite subsequence of sequence $\{(t'_l, m'_l)\}$ such that all states in α have the same value of the store (say m') and the same corresponding pair of letters a_l, b_l (say a, b); the existence of such a subsequence follows from the finiteness of the constraint system and $Act(\Delta)$.

Besides the non-rewritten subterms of t_l (called *blue subterms*), in t'_l of α there are new subterms (called *green subterms*) created during the rewriting sequence form (t_l, m) under $a^{k_1}b \dots a^{k_n}b$. Further, there are two types of sequential compositions in t'_l, the non-accessed sequential compositions of t_l (called *blue sequential compositions*) and sequential compositions created during the rewriting sequence form (t_l, m) under $a^{k_1}b \dots a^{k_n}b$ (called *green sequential compositions*). The green subterms of t'_l are created during at most $k_1 + k_2 + \dots + k_n + n$ actions. Hence their number and syntactical lengths are bounded independently on l.

Let *interesting ready sequential components (irs-components)* be subterms of t'_l corresponding to ready sequential component of term t'_l with all green sequential compositions replaced by parallel compositions. As the blue sequential compositions are not accessed during the rewriting sequence under $a^l b$, the irs-components are the only subterms of t'_l which can be possibly rewritten under $a^l b$. As there can be only green sequential compositions in every irs-component, we consider every irs-component with a green subterm as a parallel composition of one blue subterm and one green subterm.

Let α' be an infinite subsequence of α such that between every two states t'_i, t'_j of α', $i < j$, there is a bijection of irs-components with a green subterm such that corresponding green subterms are identical and for each corresponding blue subterms $s_i, s_j \in P$ there is a term $s \in P$ such that $s_i \| s = s_j$; the existence of such subsequence follows from the bound of the number and the syntactical lengths of green subterms and due to Dickson's lemma.

In terms t_l of α' there can be also irs-components without a green subterm. As we are interested in the rewriting sequence under $a^l b$ only, we can narrow down the meaning of the marking *irs-components without a green subterm*; in the following irs-components without a green subterm are the only irs-components without a green subterm rewritten during the rewriting sequence under $a^l b$. As every irs-component without a green subterm is a ready sequential component of a blue sequential composition, from Lemma 5.14 it follows that there are at most $|C| + 1$ irs-components without a green subterm, where C is a set of constraints of Δ. Due to Dickson's lemma, there are two states $(t'_i, m'), (t'_j, m')$ in α', such that $i < j$ and there is a bijection of the irs-components without a green subterm such that for each corresponding components $s_i, s_j \in P$ there is a term $s \in P$ such that $s_i \| s = s_j$.

To sum up, there is a bijection between irs-components of t'_i and t'_j (we do not consider irs-components without green subterm that are not rewritten under $a^i b$ and $a^j b$ respectively) such that green subterms of corresponding components are identical and blue subterms of irs-components of t'_i are included in blue subterms of corresponding components of t'_j. Besides considered irs-components there are no other subterms of t'_i rewritten during the rewriting sequence under $a^i b$. Hence the sequence of actions $a^i b$ performed by (t'_i, m') can be performed also by (t'_j, m'). The contradiction follows from $only^{\{a,b\}}((t_i, m), a^{k_1} b \dots a^{k_n} b a^i b)$, $only^{\{a,b\}}((t_j, m), a^{k_1} b \dots a^{k_n} b a^j b)$, and $i < j$. $\square$

Example 5.17. *Let us consider the following wBPA system with initial state pX.*

$$\begin{array}{llll} pX \xrightarrow{a} pAX & pX \xrightarrow{b} pBX & pA \xrightarrow{c} p\varepsilon & pB \xrightarrow{d} p\varepsilon \\ pA \xrightarrow{a} pAA & pA \xrightarrow{b} pBA & pA \xrightarrow{e} q\varepsilon & pB \xrightarrow{f} q\varepsilon \\ pB \xrightarrow{a} pAB & pB \xrightarrow{b} pBB & qA \xrightarrow{e} q\varepsilon & qB \xrightarrow{f} q\varepsilon \end{array}$$

In the following, actions a, b, c, d are called *I-actions* and actions e, f are called *II-actions*). Rules labelled by I-actions or II-actions are called *I-rules* or *II-rules*, respectively. States reachable from the initial state through I-actions are called *I-states*.

In the rest of this subsection we prove that there is no fcPAN system bisimilar wBPA given above.

Let bis-fcPAN denote an assumed fcPAN system Δ bisimilar to wBPA of Example 5.17 such that I-rules of Δ are of the form $(t_1 \xrightarrow{x} t_2, tt, tt)$. Please note these rules cannot be forbidden by any value of the store.

Lemma 5.18. *If there is a fcPAN Δ bisimilar to the wBPA of Example 5.17, then bis-fcPAN Δ' exists.*

Proof. As the constraint system of Δ is finite it follows there exists a I-state (t, m) of Δ such that each I-state reachable from (t, m) has also m on the store (the contrary implies the infiniteness of the constraint system). As (t, m) is a I-state, there exists a word $w \in \{c, d\}^*$ such that $(t, m) \xrightarrow{w} (s, m)$ and (s, m) is bisimilar to the initial state pX of the wBPA. The system Δ' is derived from Δ as follows: the constraint system is restricted to the part above m (including m renamed to *tt*), s is the initial term, and I-rules are of the form $(t_1 \xrightarrow{x} t_2, tt, tt)$, where $(t_1 \xrightarrow{x} t_2, c, m)$ are I-rules of Δ such that $m \vdash c$. □

Lemma 5.19. *Each I-state of a bis-fcPAN Δ has exactly one ready sequential component that is not deadlocked.*

Proof. As no I-state is deadlocked, each I-state contains at least one non-deadlocked ready sequential component. We prove that there is exactly one such a component. We assume contrary and derive a contradiction. Let t, s be two distinct non-deadlocked ready sequential components of a I-state. The are two cases.

At first we discuss I-states non-bisimilar to the initial state. We may assume that I-state can perform e action. Let t be the ready sequential component that can perform e action.

- s cannot perform any I-action as the e action performed by t is not able to forbid the I-rules (i.e., to disable these actions),
- s cannot perform e. Otherwise, neither s can perform enabled I-actions, nor other ready sequential component can perform them (according to the previous item with exchanged t for s),
- s cannot perform f as this action is disabled in the considered state.

We have a contradiction as non-deadlocked component s cannot do any action.

Now we focus on the states bisimilar to the initial state. Let us assume that t can perform a action and s can perform a b action (t and s can possibly perform the other action as well). Each possible next state has exactly one non-deadlocked ready sequential component. Thus, a action (performed by t) deadlocks or rewrites onto ε term t (the action cannot deadlock or remove s) and the same effect has action b on s. Further, these actions add the ability to perform action e or f to the next state. Hence, a action performed by t changes the ready sequential component s at the same time. This is possible only if t and s are contained in the subterm of the form $(t.r)\|s$, where $r \in P \smallsetminus \{\varepsilon\}$ and a action rewrites t onto ε:

$$(t.r)\|s \xrightarrow{a} r\|s$$

For the same reason there is a term $r' \in P \smallsetminus \{\varepsilon\}$ such that t and s are contained in the subterm of the form $t\|(s.r')$. This is a contradiction. $\square$

Definition 5.20. *A ready sequential component is called* dead *iff it is non-trivial and deadlocked whenever the value of the store is tt. A left subterm of a sequential composition is called* dead *iff it contains (or is) dead ready sequential component. A sequential composition is called* dead *iff its left subterm is dead.*

We distinguish between a *type* and an *instance* of a ready sequential component. The type is given by (syntax of) the corresponding parallel subterm while the instance is given by the subterm together with its position within the term. If it is clear from the context, we do not specify the meaning explicitly.

In what follows any dead ready sequential component occuring in some I-state of Δ is referred to as *dead ready sequential component of bis-fcPAN* Δ.

Lemma 5.21. *Given bis-fcPAN* Δ*, the set of types of dead ready sequential components occuring in I-states of* Δ *is finite.*

Proof. As a dead ready sequential component is non-trivial, it remains ready and unchanged during an arbitrary sequence of I-actions. In a bis-fcPAN , there are only two possibilities of creating a dead ready sequential component. Either it is included in the initial term, or it is on the right-hand side of an applied rule (it cannot be created by deadlocking some non-deadlocked ready sequential component as each I-state has exactly one non-deadlocked ready sequential component). Hence the lemma follows from the fact that the length of an initial term and the set of rules are both finite. $\square$

Definition 5.22. *Let Δ be a bis-fcPAN and r be a dead ready sequential component of Δ. Then r is called*

- restricted *for a I-state (t, tt) of Δ iff there is no I-state with an added instance of dead ready sequential component r reachable from (t, tt),*
- multiplicative *for a I-state (t, tt) of Δ iff for each I-state (t', tt) reachable from (t, tt), there is a I-state (t'', tt) reachable from (t', tt) such that there are more instances of r in (t'', m) than in (t', m) (i.e. arbitrary many instances of r can be added).*

We call a I-state (t_{dr}, tt) of Δ dead-restricted *iff every dead ready sequential component of Δ is either restricted, or multiplicative for (t_{dr}, tt).*

Lemma 5.23. *Let Δ be a bis-fcPAN . There is a dead-restricted state (t_{dr}, tt) of Δ reachable from the initial state.*

Proof. If every dead ready sequential component of Δ is either restricted, or multiplicative in a I-state (t, tt) then the (t_{dr}, tt) is found. Otherwise, there is a dead ready sequential component r such that it is neither restricted, nor multiplicative. As r is not multiplicative for (t, tt), there is a I-state (t', tt) reachable from (t, tt) such that every I-state reachable from (t', tt) has the same number of instances of r as (t', tt). Hence, r is restricted for the I-state (t', tt). Compared to (t, tt), at least one more type of dead ready sequntial component is restricted. Due to Lemma 5.21, we can find (t_{dr}, tt) by applying this procedure finitely many times. $\square$

Lemma 5.24. *Given $i \in \mathbb{N}$, bis-fcPAN Δ, and dead-restricted state (t_{dr}, tt), there is a dead-restricted state $t_{dr}^{(i)}$ of Δ reachable from the state (t_{dr}, tt) such that $t_{dr}^{(i)}$ includes at least i instances of each multiplicative dead ready sequential component.*

Proof. This lemma is a straightforward consequence of the definition of a multiplicative dead ready sequential component. $\square$

In the following, by *a sequential composition is behind a subterm s in term r* we mean that s is included in the left subterm of the sequential composition in term r. By *a subterm t is behind a subterm s in term r* we mean that there is a sequential composition in the term r such that t is included in the left subterm and s is included in the right subterm of this sequential composition.

Lemma 5.25. *If there are more than i instances of dead ready sequential component r in a state (t, tt) of a bis-fcPAN , then all the sequential compositions behind these instances are inaccessible from (t, tt) under any sequence of the form $e^{k_1} f e^{k_2} f \ldots e^{k_n} f$ such that $n \in \mathbb{N}_0$ and $k_j \leq i$ for every $1 \leq j \leq n$.*

Proof. We assume the contrary and derive a contradiction. Let (t, tt) be a state of a bis-fcPAN with more than i instances of some dead ready sequential component such that a sequential composition behind some of these instances is accessible under $e^{k_1} f e^{k_2} f \ldots e^{k_n} f$. The definition of bis-fcPAN implies that $only^{\{e,f\}}((t, tt), e^{k_1} f e^{k_2} f \ldots e^{k_n} f)$ holds. From the definition of fcPAN, it follows that a rule which has been already used cannot be forbidden in future. Thus the rewrite rule generating the first action performed by one of the instances can be immediately applied on the other instances. Hence, coherent sequence of more that i actions with the same label can be performed. This is a contradiction with $only^{\{e,f\}}((t, tt), e^{k_1} f e^{k_2} f \ldots e^{k_n} f)$. □

Lemma 5.26. *There is no fcPAN system bisimilar to the wBPA of Example 5.17.*

Proof. Lemma 5.18 implies that it is sufficient to show that there is no bis-fcPAN . For the sake of contradiction, let us assume that there is a bis-fcPAN system Δ with the initial term t_0 and set of constraints C. Let us consider the following transition sequence:

$$(t_0, tt) \longrightarrow^* (t_{dr}, tt) \longrightarrow^* (t_{dr}^{(k)}, tt) \xrightarrow{w_n} (r, tt)$$

The state (t_{dr}, tt) is a dead-restricted state (its reachability follows from Lemma 5.23). Let l be the number of sequential compositions in t_{dr} and $n = l + |C| + 1$. Lemma 5.16 implies that $K_n(\Delta)$ is non-empty. Let $(k_1, k_2, \ldots, k_n) \in K_n(\Delta)$ and k be the maximum of $k_2, k_2, \ldots, k_n$. Then $(t_{dr}^{(k)}, tt)$ denotes the dead-restricted state with more than k instances of each multiplicative dead ready sequential component (reachable is due to Lemma 5.24). Further, $w_n = ba^{k_n} \ldots ba^{k_2} ba^{k_1}$.

The term r is a non-deadlocked term hence it can be written in the form

$$(\cdots((((((\cdots((((t.t_1)\|s_1).t_2)\|s_2)\cdots .t_n)\|s_n).\gamma)\|\delta).\gamma')\|\delta')\cdots .\gamma^{(m)})\|\delta^{(m)},$$

where $t, t_1, \ldots, t_{n-1} \in P$, $t \neq \varepsilon$ is the only one non-deadlocked ready sequential component, and $t_n, s_1, \ldots, s_n, \gamma, \ldots, \gamma^{(m)}, \delta, \ldots, \delta^{(m)} \in G$.

As $(k_1, \ldots, k_n) \in K_n(\Delta)$ and $only^{\{c,d\}}((r, tt), c^{k_1}d \ldots c^{k_n}d)$ then r includes at least n sequential compositions accessed (see Definition 5.13) during the rewriting sequence from (r, tt) under $c^{k_1}d \ldots c^{k_n}d$.

If s_i includes a sequential composition then s_i includes a non-trivial ready sequential component. According to Lemma 5.19, this non-trivial ready sequential component is deadlocked. Hence s_i includes a dead ready sequential component and all sequential compositions behind this component are dead. This means that the compositions are inaccessible under $c^{k_1}d \ldots c^{k_n}d$. Thus at most $i-1$ sequential compositions are accessed under I-actions. This is the contradiction with the property given in the previous paragraph. Hence $s_1, \ldots, s_{n-1} \in P$, $t, t_1, \ldots, t_{n-1} \in P \setminus \{\varepsilon\}$, and all dead ready sequential components of r are included in subterms $\delta, \ldots, \delta^{(m)}$. Further, similarly to the sequential compositions of s_i discussed above, all the sequential compositions of $\delta, \ldots, \delta^{(m)}$ are dead.

From Lemma 5.19 it follows that $t_i \| s_i$ (where $1 \leq i \leq n$) is a new ready sequential component appeared by accessing the i-th sequential compositions. (Precisely, in case of $t_n \notin P$, the new ready sequential component appeared by accessing the n-th sequential composition is a proper subterm of $t_n \| s_n$.)

As $(k_1, \ldots, k_n) \in K_n(\Delta)$ and $only^{\{e,f\}}((r, tt), e^{k_1}f \ldots e^{k_n}f)$, r includes at least n sequential compositions accessed during the rewriting sequence from (r, tt) under $e^{k_1}f \ldots e^{k_n}f$.

We recall that all dead sequential compositions are in $\delta, \ldots, \delta^{(m)}$. From Lemma 5.25 it follows that all the sequential compositions behind a multiplicative dead ready sequential components are inaccessible. Hence, the only accessible sequencial compositions of $\delta, \ldots, \delta^{(m)}$ are behind restricted dead ready sequencial components. According to the definitions, all restricted dead ready sequencial components and the terms behind them have already been created in t_{dr}. Hence, there are at most l such accessible compositions in r.

Thus at least $|C| + 1$ $(= n - l)$ sequential compositions of the subterm $((\cdots((t.t_1)\|s_1)\cdots .t_n)\|s_n)$ are accessed during a rewriting sequence from (r, tt) under $e^{k_1}f \ldots e^{k_n}f$. In other words, all the sequential compositions of $((\cdots((t.t_1)\|s_1)\cdots .t_{|C|+1})\|s_{|C|+1})$ are accessed during the sequence.

For every $1 \leq i \leq |C| + 1$, accessing the i-th composition have to be preceded by changing s_i; otherwise we get the same ready sequential component $t_i \| s_i$ as in the rewriting sequence under $c^{k_1}d \ldots c^{k_n}d$ and I-actions can be performed. The only possibility of forcing the preceding is to change the store during at least one action performed by s_i. Otherwise all the actions performed by s_i can be omitted without any effect on the rewriting sequence accessing the i-th composition. Hence we have to change a constraint at least $|C| + 1$ times. It is the contradiction and Lemma 5.26 is proved. □

6 Conclusion and future work

We have extended Process Rewrite Systems (PRS) [12] by 'weak' finite-state unit and have classified new classes by their expressiveness. We have shown the refined hierarchy (w.r.t. bisimilarity) containing new classes as well as those generated by both PRS and of two other PRS extensions introduced in [9, 21].

We emphasize the results showing that BPP class and its three extensions form a strict (sub)hierarchy w.r.t. bisimulation,

$$\text{BPP} \subsetneq \text{fcBPP} \subsetneq \text{wBPP} \subsetneq \text{seBPP} \subsetneq \text{PN}$$

which is decidable (even PSPACE-complete) on the BPP class and undecidable on the class of state-extended BPP (i.e. PPDA). It remains open for other two classes (i.e. fcBPP and wBPP) and is a subject of our further research. We are motivated by the fact the strictness of two left-most inclusions can be proved (but is not shown here) even for language equivalence. The strictness of inclusion between wBPP and seBPP on the language equivalence level is just our conjecture.

References

[1] A. Bouajjani, R. Echahed, and P. Habermehl. On the verification problem of nonregular properties for nonregular processes. In *Proc. of LICS'95*. IEEE, 1995.

[2] O. Burkart, D. Caucal, F. Moller, and B. Steffen. Verification on infinite structures. In *Handbook of Process Algebra*, pages 545–623. Elsevier, 2001.

[3] O. Burkart, D. Caucal, and B. Steffen. Bisimulation collapse and the process taxonomy. In *Proc. of CONCUR'96*, volume 1119 of *LNCS*, pages 247–262. Springer, 1996.

[4] O. Burkart and J. Esparza. More infinite results. *Electronic Notes in Theoretical Computer Science*, 5, 1997.

[5] D. Caucal. On the regular structure of prefix rewriting. *Theoretical Computer Science*, 106:61–86, 1992.

[6] S. Christensen, Y. Hirshfeld, and F. Moller. Bisimulation is decidable for all basic parallel processes. In *Proceedings of CONCUR'93*, volume 715 of *LNCS*, pages 143–157. Springer, 1993.

[7] E. M. Clarke and E. A. Emerson. Design and synthesis of synconization skeletons using branching time temporal logic. In *Proc. IBM Workshop on Logic of Programs*, volume 131 of *LNCS*, pages 52–71. Springer, 1981.

[8] P. Jančar. Strong bisimilarity on basic parallel processes is PSPACE-complete. In *Proc. of 18th IEEE Symposium on Logic in Computer Science (LICS'03)*, pages 218–227. IEEE Computer Society, 2003.

[9] P. Jančar, A. Kučera, and R. Mayr. Deciding bisimulation-like equivalences with finite-state processes. *Theoretical Computer Science*, 258:409–433, 2001.

[10] A. Kučera and P. Jančar. Equivalence-checking with infinite-state systems: Techniques and results. In *Proc. SOFSEM'2002*, volume 2540 of *LNCS*. Springer, 2002.

[11] M. Maidl. The common fragment of CTL and LTL. In *Proc. 41th Annual Symposium on Foundations of Computer Science*, pages 643–652, 2000.

[12] R. Mayr. Process rewrite systems. *Information and Computation*, 156(1):264–286, 2000.

[13] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[14] F. Moller. Infinite results. In *Proc. of CONCUR'96*, volume 1119 of *LNCS*, pages 195–216. Springer, 1996.

[15] F. Moller. Pushdown Automata, Multiset Automata and Petri Nets, MFCS Workshop on concurrency. *Electronic Notes in Theoretical Computer Science*, 18, 1998.

[16] D. Muller, A. Saoudi, and P. Schupp. Alternating automata, the weak monadic theory of trees and its complexity. *Theoret. Computer Science*, 97(1–2):233–244, 1992.

[17] A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symposium on the Foundations of Computer Science*, pages 46–57, 1977.

[18] V. A. Saraswat and M. Rinard. Concurrent constraint programming. In *Proc. of 17th POPL*, pages 232–245. ACM Press, 1990.

[19] J. Srba. Roadmap of infinite results. *EATCS Bulletin*, (78):163–175, 2002.

[20] J. Srba. Strong bisimilarity and regularity of basic parallel processes is PSPACE-hard. In *Proc. STACS 2002*, volume 2285 of *LNCS*, pages 535–546. Springer, 2002.

[21] J. Strejček. Rewrite systems with constraints, EXPRESS'01. *Electronic Notes in Theoretical Computer Science*, 52, 2002.

Publications in the FI MU Report Series are in general accessible via WWW and anonymous FTP:

`http://www.fi.muni.cz/informatics/reports/`
`ftp ftp.fi.muni.cz (cd pub/reports)`

Copies may be also obtained by contacting:

Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic