Extended Process Rewrite Systems: Expressiveness and Reachability*

Mojmír Křetínský, Vojtěch Řehák**, and Jan Strejček

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

Abstract. We unify a view on three extensions of Process Rewrite Systems (PRS) and compare their expressive power with that of PRS. We show that the class of Petri nets is less expressive up to bisimulation equivalence than the class of PA processes extended with a finite state control unit. Further we show our main result that the reachability problem for PRS extended with a so called weak finite state unit is decidable.

1 Introduction

An automatic verification of current software systems often needs to model them as infinite-state systems, i.e. systems with an evolving structure and/or operating on unbounded data types. Infinite-state systems can be specified in a number of ways with their respective advantages and limitations. Petri nets, pushdown automata, and process algebras like BPA, BPP, or PA all serve to exemplify this. Here we employ the classes of infinite-state systems defined by term rewrite systems and called *Process Rewrite Systems* (PRS) as introduced by Mayr [May00]. PRS subsume a variety of the formalisms studied in the context of formal verification (e.g. all the models mentioned above).

A PRS is a finite set of rules $t \xrightarrow{a} t'$ where a is an action under which a subterm t can be reduced onto a subterm t'. Terms are built up from an empty process ε and a set of process constants using (associative) sequential "." and (associative and commutative) parallel "$\|$" operators. The semantics of PRS can be defined by labelled transition systems (LTS) – labelled directed graphs whose nodes (states of the system) correspond to terms modulo properties of "." and "$\|$" and edges correspond to individual actions (computational steps) which can be performed in a given state. The relevance of various subclasses of PRS for modelling and analysing programs is shown e.g. in [Esp02], for automatic verification see e.g. surveys [BCMS01,Srb02].

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

** The co-author has been supported by Marie Curie Fellowship of the European Community Programme Improving the Human Research Potential and the Socio-economic Knowledge Base under contract number HPMT-CT-2000-00093.

P. Gardner and N. Yoshida (Eds.): CONCUR 2004, LNCS 3170, pp. 355–370, 2004.

Mayr [May00] has also shown that the reachability problem (i.e. given terms t, t': is t reducible to t'?) for PRS is decidable. Most research (with some recent exceptions, e.g. [BT03,Esp02]) has been devoted to the PRS classes from the lower part of the PRS hierarchy, especially to pushdown automata (PDA), Petri nets (PN) and their respective subclasses. We mention the successes of PDA in modeling recursive programs (without process creation), PN in modeling dynamic creation of concurrent processes (without recursive calls), and CPDS (communicating pushdown systems [BET03]) modeling both features. All of these formalisms subsume a notion of a finite state unit (FSU) keeping some kind of global information which is accessible to the redices (the ready to be reduced components) of a PRS term – hence a FSU can regulate rewriting. On the other hand, using a FSU to extend the PRS rewriting mechanism is very powerful since the state-extended version of PA processes (sePA) has a full Turing-power [BEH95] – the decidability of reachability is lost for sePA, including all its superclasses (see Figure 1), and CPDS as well.

This paper presents a hierarchy of PRS classes and their respective extensions of three types: fcPRS classes ([Str02], inspired by concurrent constraint programming [SR90]), wPRS classes ([KŘS03], PRS systems equipped with weak FSU inspired by weak automata [MSS92]), and state-extended PRS classes [JKM01]. The classes in the hierarchy (depicted in Figure 1) are related by their expressive power with respect to (strong) bisimulation equivalence. As the main contribution of the paper we show that the reachability problem remains decidable for the very expressive class of wPRS. This result deserves some additional remarks:

- It determines the decidability borderline of the reachability problem in the mentioned hierarchy; the problem is decidable for all classes except those with Turing power. In other words, it can be seen as a contribution to studies of algorithmic boundaries of reachability for infinite-state systems.
- In the context of verification, one often formulates a property expressing that *nothing bad occurs*. These properties are called *safety properties*. The collection of the most often verified properties [DAC98] contains 41% of such properties. Model checking of safety properties can be reduced to the reachability problem. Moreover, many successful verification tools concentrate on reachability only. Therefore, our decidability result can be seen as a contribution to an automatic verification of infinite-state systems as well.
- Given a labelled transition system $(S, Act, \longrightarrow, \alpha_0)$ with a distinguished action $\tau \in Act$, we define a *weak trace set* of a state $s \in S$ as

  $$wtr(s) = \{w \in (Act \smallsetminus \{\tau\})^* \mid s \stackrel{w}{\Longrightarrow} t \text{ for some } t \in S\},$$

  where $s \stackrel{w}{\Longrightarrow} t$ means that there is some $w' \in Act^*$ such that $s \stackrel{w'}{\longrightarrow} t$ and w is equal to w' without τ actions. Two systems are *weak trace equivalent* if the weak trace sets of their initial states are the same. So far it has been known that weak trace non-equivalence is semi-decidable for Petri nets (see e.g. [Jan95]), pushdown processes (due to [Büc64]), and PA processes (due to [LS98]). Using the decidability result, it is easy to show that the weak

trace set is recursive for every state of any wPRS. Hence, the weak trace non-equivalence is semi-decidable for (all subclasses of) wPRS.
- Our decidability result has been recently applied in the area of cryptographic protocols. Hüttel and Srba [HS04] define a replicative variant of a calculus for Dolev and Yao's ping-pong protocols [DY83]. They show that the reachability problem for these protocols is decidable as it can be reduced to the reachability problem for wPRS.

The outline of the paper is as follows: after some preliminaries we introduce a uniform framework for specifying all extended PRS formalisms in Section 3 and compare their relative expressiveness with respect to bisimulation equivalence in Section 4. Here we also solve (to the best of our knowledge) an open problem on the relationship between the PN and sePA classes by showing that PN is less expressive (up to bisimulation equivalence) than sePA. In Section 5 we show that all classes of our fcPRS and wPRS extensions keep the reachability problem decidable. The last section summarises our results.

Related Work: In the context of reachability analysis one can see at least two approaches: (i) abstraction (approximate) analysis techniques on stronger 'models' such as sePA and its superclasses with undecidable reachability, e.g. see a recent work [BET03], and (ii) precise techniques for 'weaker' models, e.g. PRS classes with decidable reachability, e.g. [LS98] and another recent work [BT03]. In the latter one, symbolic representations of set of reachable states are built with respect to various term structural equivalences. Among others it is shown that for the PAD class and the same equivalence as in this paper, when properties of sequential and parallel compositions are taken into account, one can construct nonregular representations based on counter tree automata.

2 Preliminaries

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 $\alpha \xrightarrow{a} \beta$ instead of $(\alpha, a, \beta) \in \longrightarrow$), $\alpha_0 \in S$ is a distinguished *initial state*.

We use the natural generalization $\alpha \xrightarrow{\sigma} \beta$ for finite sequences of actions $\sigma \in Act^*$. The state α is *reachable* if there is $\sigma \in Act^*$ such that $\alpha_0 \xrightarrow{\sigma} \alpha$.

A binary relation R on set of states S is a *bisimulation* [Mil89] iff for each $(\alpha, \beta) \in R$ the following conditions hold:

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

Bisimulation equivalence (or *bisimilarity*) on a LTS is the union of all bisimulations (i.e. the largest bisimulation).

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, $X \in \mathit{Const}$ is a process constant (used as an atomic process), '$\|$' and '.' mean parallel and sequential compositions respectively.

The set $\mathit{Const}(t)$ 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* as:

1 – terms consisting of a single process constant only, in particular $\varepsilon \notin 1$,
S – *sequential* terms - without parallel composition, e.g. $X.Y.Z$,
P – *parallel* terms - without sequential composition. e.g. $X \| Y \| Z$,
G – *general* terms with arbitrarily nested sequential and parallel compositions.

Definition 1. *Let $\mathit{Act} = \{a, b, \cdots\}$ 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) *Δ is a pair (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 $a \in \mathit{Act}$ is an atomic action,*
- *$t_0 \in \beta$ is an* initial state.

Given PRS Δ we define $\mathit{Const}(\Delta)$ as the set of all constants occurring in the rewrite rules of Δ or in its initial state, and $\mathit{Act}(\Delta)$ 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 $(S, \mathit{Act}(\Delta), \longrightarrow, t_0)$, where $S = \{t \in \beta \mid \mathit{Const}(t) \subseteq \mathit{Const}(\Delta)\}$ 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 arises, we sometimes speak about a "process rewrite system" meaning a "labelled transition system generated by process rewrite system".

Some classes of (α, β)-PRS correspond to widely known models as finite state systems (FS), basic process algebras (BPA), basic parallel processes (BPP), process algebras (PA), pushdown processes (PDA, see [Cau92] for justification), and Petri nets (PN). The other classes were introduced (and named as PAD, PAN, and PRS) by Mayr [May00]. The correspondence between (α, β)-PRS classes and acronyms just mentioned can be seen in Figure 1.

3 Extended PRS

In this section we recall the definitions of three different extensions of process rewrite systems, namely *state-extended PRS (sePRS)* [JKM01], *PRS with a finite constraint system (fcPRS)* [Str02], and *PRS with a weak finite-state unit* *(wPRS)* [KŘS03]. In all cases, the PRS formalism is extended with a finite state unit of some kind.

sePRS. State-extended PRS corresponds to PRS extended with a finite state unit without any other restrictions. The well-known example of this extension is the state-extended BPA class (also known as pushdown processes).

wPRS. The notion of weakness employed in the wPRS formalism corresponds to that of weak automaton [MSS92] in automata theory. The behaviour of a weak state unit is acyclic, i.e. states of state unit are ordered and non-increasing during every sequence of actions. As the state unit is finite, its state can be changed only finitely many times during every sequence of actions.

fcPRS. The extension of PRS with finite constraint systems is motivated by *concurrent constraint programming (CCP)* (see e.g. [SR90]). In CCP the processes work with a shared *store* (seen as a constraint on values that variables can represent) via two operations, *tell* and *ask*. The *tell* adds a constraint to the store provided the store remains *consistent*. The *ask* is a test on the store – it can be executed only if the current store implies a specified constraint.

Formally, values of a store form a bounded lattice (called a *constraint system*) with the lub operation $\wedge$ (least upper bound), the least element *tt*, and the greatest element *ff*. The execution of *tell(n)* changes the value of the store from o to $o \wedge n$ (provided $o \wedge n \neq ff$ – consistency check). The *ask(m)* can be executed if the current value of the store o is greater than m.

The state unit of fcPRS has the same properties as the store in CCP. We add two constraints (m, n) to each rewrite rule. The application of a rule corresponds to the concurrent execution of *ask(m)*, *tell(n)*, and rewriting:

– a rule can be applied only if the actual store o satisfies $m \leq o$ and $o \wedge n \neq ff$,
– the application of the rule rewrites the process term and changes the store to $o \wedge n$.

We first define the common syntax of the aforementioned extended PRS and then we specify the individual restrictions on state units.

Definition 2. *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* extended (α, β)-PRS Δ *is a tuple* $(M, \leq, R, m_0, t_0)$*, where*

– M *is a finite set of* states *of the* state unit*,*
– $\leq$ *is a binary relation over* M*,*
– R *is a finite set of* rewrite rules *of the form* $(m, t_1) \xrightarrow{a} (n, t_2)$*, where* $t_1 \in \alpha$*,* $t_1 \neq \varepsilon$*,* $t_2 \in \beta$*,* $m, n \in M$*, and* $a \in Act$*,*
– *Pair* $(m_0, t_0) \in M \times \beta$ *forms a distinguished* initial state *of the system.*

The specific type of an extended (α, β)-PRS is given by further requirements on $\leq$. An extended (α, β)-PRS is

- (α, β)-*sePRS* without any requirements on $\leq$.[1]
- (α, β)-*wPRS* iff $(M, \leq)$ is a partially ordered set.
- (α, β)-*fcPRS* iff $(M, \leq)$ is a bounded lattice. The lub operation (least upper bound) is denoted by $\wedge$, the least and the greatest elements are denoted by *tt* and *ff*, respectively. We also assume that $m_0 \neq ff$.

To shorten our notation we prefer mt over (m, t). As in the PRS case, instead of $(mt_1 \xrightarrow{a} nt_2) \in R$ where $\Delta = (M, \leq, R, m_0, t_0)$, we usually write $(mt_1 \xrightarrow{a} nt_2) \in \Delta$. The meaning of $Const(\Delta)$ (process constants used in rewrite rules or in t_0) and $Act(\Delta)$ (actions occurring in rewrite rules) for a given extended PRS Δ is also the same as in the PRS case.

The semantics of an extended (α, β)-PRS system Δ is given by the corresponding labelled transition system $(S, Act(\Delta), \longrightarrow, m_0t_0)$, where[2]

$$S = M \times \{t \in \beta \mid Const(t) \subseteq Const(\Delta)\}$$

and the relation $\longrightarrow$ is defined as the least relation satisfying the inference rules corresponding to the application of rewrite rules (and dependent on the concrete formalism):

$$\text{sePRS} \quad \frac{(mt_1 \xrightarrow{a} nt_2) \in \Delta}{mt_1 \xrightarrow{a} nt_2} \qquad \text{wPRS} \quad \frac{(mt_1 \xrightarrow{a} nt_2) \in \Delta}{mt_1 \xrightarrow{a} nt_2} \text{ if } n \leq m$$

$$\text{fcPRS} \quad \frac{(mt_1 \xrightarrow{a} nt_2) \in \Delta}{ot_1 \xrightarrow{a} (o \wedge n)t_2} \text{ if } m \leq o \text{ and } o \wedge n \neq ff$$

and two common inference rules

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

where $t_1, t_2, t_1' \in \mathcal{T}$ and $m, n, o \in M$.

Instead of $(1, S)$-sePRS, $(1, S)$-wPRS, $(1, S)$-fcPRS, ... we use a more natural notation seBPA, wBPA, fcBPA, etc. The class seBPP is also known as *multiset automata (MSA)* or *parallel pushdown automata (PPDA)*, see [Mol96].

4 Expressiveness

Figure 1 describes the hierarchy of PRS classes and their extended counterparts with respect to bisimulation equivalence. If any process in class X can be also defined (up to bisimilarity) in class Y we write $X \subseteq Y$. If additionally $Y \not\subseteq X$ holds, we write $X \subsetneq Y$ and say X is less expressive than Y. This is depicted by

[1] In this case, the relation $\leq$ can be omitted from the definition.

[2] If Δ is an fcPRS, we eliminate the states with *ff* from S as they are unreachable.

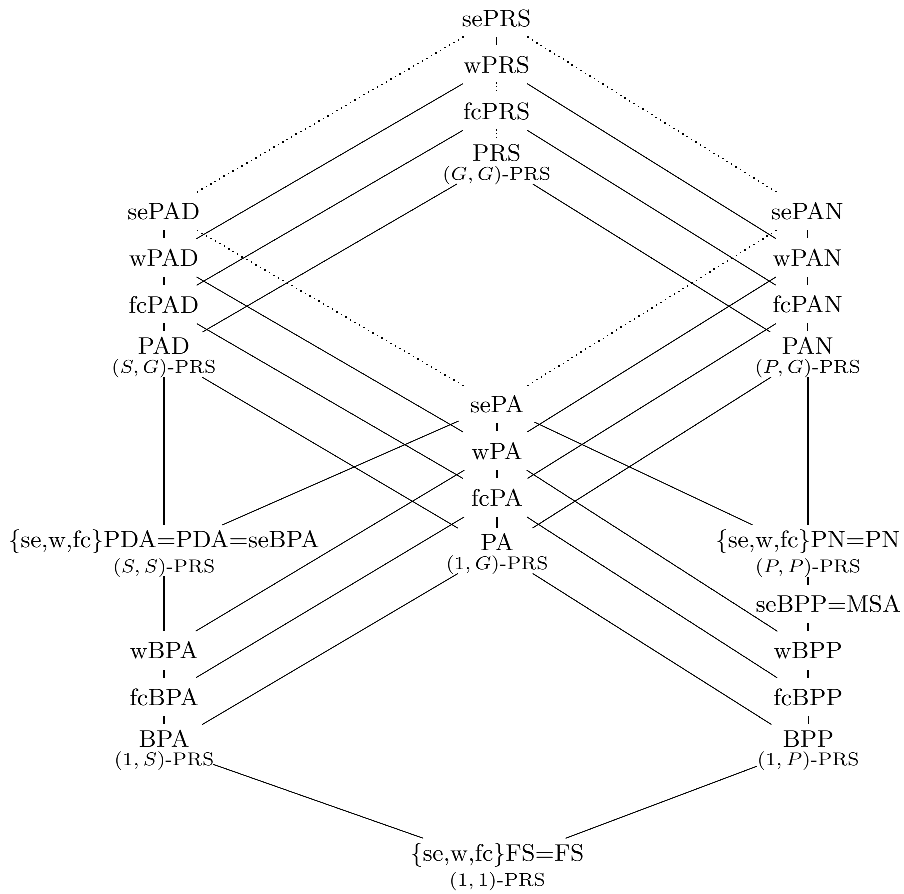

Fig. 1. The hierarchy of classes defined by (extended) rewrite formalisms

the line(s) connecting X and Y with Y placed higher than X in Figure 1. The dotted lines represent the facts $X \subseteq Y$, where we conjecture that $X \subsetneq Y$ hold.

Some observations (even up to isomorphism) are immediate, for example

1. the classes FS, PDA and PN coincide with their extended analogues,
2. if e $\in \{se, w, fc\}$ and X $\subseteq$ Y then eX $\subseteq$ eY, and
3. (α, β)-PRS $\subseteq$ (α, β)-fcPRS $\subseteq$ (α, β)-wPRS $\subseteq$ (α, β)-sePRS for all (α, β)-PRS.

The strictness ('$\subsetneq$') of the PRS-hierarchy has been proved by Mayr [May00], that of the corresponding classes of PRS and fcPRS has been proved in [Str02], and the relations among MSA and the classes of fcPRS and wPRS have been studied in [KŘS03]. Note that the strictness relations wX $\subsetneq$ seX hold for all X = PA, PAD, PAN, PRS due to our reachability result for wPRS given in Sec. 5 and due to the full Turing-power of sePA [BEH95].

These proofs together with Moller's result establishing MSA $\subsetneq$ PN [Mol98] complete the justification of Figure 1 – with one exception, namely the relation between the PN and sePA classes. Looking at two lines leaving sePA down to the left and down to the right, we note the "left-part collapse" of (S, S)-PRS and PDA proved by Caucal [Cau92] (up to isomorphism). The right-part counterpart is slightly different due to the previously mentioned result that MSA $\subsetneq$ PN. In the next subsection we prove that PN $\subsetneq$ sePA (in fact it suffices to demonstrate PN $\subseteq$ sePA as the strictness is obvious).

4.1 *PN* $\subsetneq$ *sePA*

We now show that Petri nets are less expressive (with respect to bisimilarity) than sePA processes. In this section, a Petri net Δ is considered in traditional notation (see e.g. [Pet81]). Let $Const(\Delta) = \{X_1, \ldots, X_k\}$, a state $X_1^{p_1} \| \ldots \| X_k^{p_k}$ of a PN Δ is written as $(p_1, \ldots, p_k)$ and called *marking*. Each p_i is the number of *tokens* at the *place* P_i. Any rewrite rule $X_1^{l_1} \| \ldots \| X_k^{l_k} \xrightarrow{a} X_1^{r_1} \| \ldots \| X_k^{r_k}$ (where $l_i, r_i \geq 0$) is written as $(l_1, \ldots, l_k) \xrightarrow{a} (r_1, \ldots, r_k)$ and called *transition*[3]. The heart of our argument is a construction of a sePA Δ' bisimilar to a given PN Δ.

The main difficulty in this construction is to maintain the number of tokens at the places of a PN. To this end, we may use two types of sePA memory: a finite control (FSU), which cannot represent an unbounded counter, and a term of an unbounded length, where just one constant can be rewritten in one step.

Our construction of a sePA Δ' can be reformulated on intuitive level as follows. Let a marking $(p_1, \ldots, p_k)$ mean that we have p_i units of the i-th currency, $i = 1, \ldots, k$. An application of a PN transition $(l_1, \ldots, l_k) \xrightarrow{a} (r_1, \ldots, r_k)$ has the effect of a currency exchange from p_i to $p_i - l_i + r_i$ for all i. A sePA reseller Δ' will have k finite pockets (in its FSU) and k bank accounts (a parallel composition of k sequential terms t_i). The reseller Δ' maintains an invariant $p_i = pocket_i + account_i$ for all i. To mimic a PN transition he must obey sePA rules, i.e. he may use all his pockets, but just one of his accounts in one exchange. A solution is to do $pocket_i \leftrightarrow account_i$ transfers cyclically, $i = 1, \ldots, k$. Hence, rebalancing $pocket_i$ the reseller Δ' must be able to perform the next $k-1$ exchanges without accessing $account_i$ (while visiting the other accounts). Therefore, Δ' needs sufficiently large (but finite) pockets and sufficiently high (and fixed) limits for $pocket_i \leftrightarrow account_i$ transfers. We show these bounds exist.

In one step the amount of the i-th currency cannot be changed by more than $L_i = max\,\{\,|l_i{-}r_i|\,;(l_1, \ldots, l_k) \xrightarrow{a} (r_1, \ldots, r_k)$ is a PN transition$\}$, thus $M_i = k{\cdot}L_i$ is an upper bound for the total effect of k consecutive steps. Any rebalancing of $pocket_i$ sets its value into $\{M_i, \ldots, 2M_i - 1\}$ (or $\{0, \ldots, 2M_i - 1\}$ if $account_i$ is empty). Hence, after k transitions the value of $pocket_i$ is in $\{0, \ldots, 3M_i - 1\}$. In the next rebalancing $account_i$ can be increased or decreased (if it is not empty) by M_i to get $pocket_i$ between M_i (or 0 if $account_i$ is empty) and $2M_i - 1$ again.

[3] Till now, X^n denoted the parallel composition of n copies of X. In the rest of the section, it denotes the sequential composition of n copies of X.

Each state of sePA Δ' consists of a state of a FSU and a term (parallel composition of k stacks representing accounts). A state of a FSU is in the product

$$\underbrace{\{1,\ldots,k\}}_{update\ controller} \times \underbrace{\{0,\ldots,3M_1-1\}}_{pocket_1} \times \ldots \times \underbrace{\{0,\ldots,3M_k-1\}}_{pocket_k}.$$

The *update controller* goes around the range and refers to the account being updated (rebalanced) in the next step. The value of each $pocket_i$ (subsequently denoted by m_i) is equal to the number of tokens at P_i counted modulo M_i.

We define $2k$ process constants $B_i, X_i \in Const(\Delta')$, where B_i represents the bottom of the i-th stack and each X_i represents M_i tokens at place P_i. The i-th stack t_i is of the form $X_i^n.B_i$, where $n \geq 0$.

For a given initial marking $\alpha_0 = (p_1,\ldots,p_k)$ of a PN Δ we construct the initial state $\beta_0 = (1, m_1,\ldots,m_k)\, t_1\|\cdots\|t_k$ of the sePA Δ', where denoting $n_i = max(0, (p_i \text{ div } M_i) - 1)$ we put $m_i = p_i - n_i \cdot M_i$ and $t_i = X_i^{n_i}.B_i$. In other words we have $p_i = m_i + n_i \cdot M_i$ and moreover $m_i \in \{M_i,\ldots,2M_i-1\}$ if p_i is big enough (i.e. $p_i \geq M_i$).

For each transition $(l_1,\ldots,l_k) \xrightarrow{a} (r_1,\ldots,r_k)$ of the PN Δ we construct the set of sePA rules $(s, m_1,\ldots,m_k)\, t \xrightarrow{a} (s', m'_1,\ldots,m'_k)\, t'$ such that they obey the following conditions:

- Update controller conditions: $s, s' \in \{1,\ldots,k\}$ and $s' = (s \mod k) + 1$.
- The general conditions for pockets ($1 \leq i \leq k$):
  - $m_i, m'_i \in \{0,\ldots,3M_i-1\}$,
  - $m_i \geq l_i$ (i.e. the transition can be performed),
  - if $i \neq s$ then $m'_i = m_i - l_i + r_i$.

We now specify m'_s and the terms t, t'. The first two *Bottom rules* are the rules for working with the empty stack. The next three *Top rules* describe the rewriting of process constant X_s. Depending on the value of $\overline{m_s} = m_s - l_s + r_s$, there are *dec*, *inc*, and *basic* variants manipulating the s-th stack.

Rule	t	$\overline{m_s} \in$	m'_s	t'
Bottom-basic rule	B_s	$\{0,\ldots,2M_s-1\}$	$\overline{m_s}$	B_s
Bottom-inc rule	B_s	$\{2M_s,\ldots,3M_s-1\}$	$\overline{m_s} - M_s$	$X_s.B_s$
Top-dec rule	X_s	$\{0,\ldots,M_s-1\}$	$\overline{m_s} + M_s$	ε
Top-basic rule	X_s	$\{M_s,\ldots,2M_s-1\}$	$\overline{m_s}$	X_s
Top-inc rule	X_s	$\{2M_s,\ldots,3M_s-1\}$	$\overline{m_s} - M_s$	$X_s.X_s$

Theorem 1. *PN $\subsetneq$ sePA with respect to bisimulation equivalence.*

Proof. (Sketch) Let Δ is a PN and Δ' is the sePA constructed as described above. In the following β refers to a state $(s, m_1,\ldots,m_k)\, X_1^{n_1}.B_1\|\ldots\|X_k^{n_k}.B_k$ of the sePA Δ' while α refers to a marking $(p_1,\ldots,p_k)$ of the PN Δ. We show

$$R = \{(\alpha,\beta) \mid \beta \text{ is reachable and } p_i = m_i + n_i \cdot M_i \text{ for all } i = 1,\ldots k\}$$

is a bisimulation relation.

Let us assume that $(\alpha, \beta) \in R$ and a transition $(l_1, \ldots, l_k) \stackrel{a}{\longrightarrow} (r_1, \ldots, r_k)$ fired in α leads to α'. Exactly one sePA rule derived from this PN transition (see the table of rules) is applicable on β. (This statement is due to the straightforward observations: if $n_i \neq 0$ then $m_i \geq L_i$, hence $p_i \geq l_i$ iff $m_i \geq l_i$.) The application of this rule on β leads to $\beta' = (s', m'_1, \ldots, m'_k)\, X_1^{n'_1}.B_1 \| \ldots \| X_k^{n'_k}.B_k$ which satisfies $m'_i + n'_i \cdot M_i = m_i + n_i \cdot M_i - l_i + r_i$ and $0 \leq m'_i < 3M_i$ for all i. Hence, $(\alpha', \beta') \in R$. The symmetric case proceeds in a similar way.

Note that the pair of the initial marking α_0 of the PN Δ and the initial state β_0 of the sePA Δ' is in R. Hence, Δ and Δ' are bisimilar. We have demonstrated that PN $\subseteq$ sePA (with respect to bisimulation equivalence).

The strictness of this relation follows from two of the results mentioned in the introduction, namely the full Turing-power of sePA [BEH95] and the decidability of reachability for PN [May81]. $\square$

We note that the sePA system constructed by our algorithm does not need to be isomorphic to the original PN system (e.g. due to the different values of the update controller).

5 Reachability for wPRS Is Decidable

In this section we show that for a given wPRS Δ and its states rt_1, st_2 it is decidable whether st_2 is reachable from rt_1 or not (recall that st_2 is reachable from rt_1 if a sequence of actions σ such that $rt_1 \stackrel{\sigma}{\longrightarrow} st_2$ exists).

Our proof exhibits a similar structure to the proof of decidability of the reachability problem for PRS [May00]; first we reduce the general problem to the reachability problem for wPRS with rules containing at most one occurrence of a sequential or parallel operator, and then we solve this subproblem using the fact that the reachability problems for both PN and PDA are decidable [May81, Büc64]. The latter part of our proof is based on a new idea of *passive steps* presented later.

To get just a sketch of the following proof we suggest to read the definitions and statements (skipping their technical proofs). Some of them are preceded by comments that provide some intuition.

As the labels on rewrite rules are not relevant here, we omit them in this section. To distinguish between rules and rewriting sequences we use $rt_1 \succ^{\Delta} st_2$ to denote that the state st_2 is reachable from rt_1 in wPRS Δ. Further, states of weak state unit are called *weak states*.

Definition 3. *Let Δ be a wPRS. A rewrite rule in Δ is* parallel *or* sequential *if it has one of the following forms:*

$$\begin{array}{rllll} \text{parallel rules:} & pX \longrightarrow qY\|Z & pX\|Y \longrightarrow qZ & pX \longrightarrow qY & pX \longrightarrow q\varepsilon, \\ \text{sequential rules:} & pX \longrightarrow qY.Z & pX.Y \longrightarrow qZ & pX \longrightarrow qY & pX \longrightarrow q\varepsilon, \end{array}$$

where X, Y, Z are process constants and p, q are weak states. A rule is trivial *if it is both parallel and sequential (i.e. it has the form $pX \longrightarrow qY$ or $pX \longrightarrow q\varepsilon$). A wPRS Δ is in* normal form *if every rewrite rule in Δ is parallel or sequential.*

Lemma 1. *For a wPRS Δ, terms t_1, t_2, and weak states r, s, there are terms t'_1, t'_2 of wPRS Δ' in normal form satisfying $rt_1 \succ^{\Delta} st_2 \iff rt'_1 \succ^{\Delta'} st'_2$. Moreover, wPRS Δ' and terms t'_1, t'_2 can be effectively constructed.*

Proof. In this proof we assume that the sequential composition is left-associative. It means that the term $X.Y.Z$ is $(X.Y).Z$ and so its subterms are X, Y, Z, and $X.Y$, but not $Y.Z$. However, the term $Y\|Z$ is a subterm of $X.(Y\|Z)$.

Let $size(t)$ denote the number of sequential and parallel operators in term t. Given any wPRS Δ, let k_i be the number of rules $(pt \longrightarrow qt') \in \Delta$ that are neither parallel nor sequential and $size(pt \longrightarrow qt') = i$, where $size(pt \longrightarrow qt') = size(t) + size(t')$. Thus, Δ is in normal form iff $k_i = 0$ for every i. In this case, let $n = 0$. Otherwise, let n be the largest i such that $k_i \neq 0$ (n exists as the set of rules is finite). We define $norm(\Delta)$ to be the pair (n, k_n).

We now describe a procedure transforming Δ (if it is not in normal form) into a wPRS Δ' and terms t_1, t_2 into terms t'_1, t'_2 such that $norm(\Delta') < norm(\Delta)$ (with respect to the lexicographical ordering) and $rt_1 \succ^{\Delta} st_2 \iff rt'_1 \succ^{\Delta'} st'_2$.

Let us assume that wPRS Δ is not in normal form. Then there is a rule that is neither sequential nor parallel and has the maximal *size*. Take a non-atomic and proper subterm t of this rule and replace every subterm t in Δ (i.e. in rewrite rules and initial term) and in t_1 and t_2 by a fresh constant X. Then add two rules $pX \longrightarrow pt$ and $pt \longrightarrow pX$ for each weak state p. This yields a new wPRS Δ' and terms t'_1 and t'_2 where the constant X serves as an abbreviation for the term t. By the definition of *norm* we get $norm(\Delta') < norm(\Delta)$. The correctness of our transformation remains to be demonstrated, namely that

$$rt_1 \succ^{\Delta} st_2 \iff rt'_1 \succ^{\Delta'} st'_2.$$

The implication $\Longleftarrow$ is obvious. For the opposite direction we show that every rewriting step in Δ from pl_1 to ql_2 under the rule $(pl \longrightarrow ql') \in \Delta$ corresponds to a sequence of several rewriting steps in Δ' leading from pl'_1 to ql'_2, where l'_1, l'_2 are equal to l_1, l_2 with all occurrences of t replaced by X. Let us assume the rule $pl \longrightarrow ql'$ modifies a subterm t of pl_1, and/or a subterm t appears in ql_2 after the rule application (the other cases are trivial). If the rule modifies a subterm t of l_1 there are two cases. Either l includes the whole t and then the corresponding rule in Δ' (with t replaced by X) can be applied directly on pl'_1, or, due to the left-associativity of a sequential operator, t is not a subterm of the right part of any sequential composition in l_1 and thus the application of the corresponding rule in Δ' on pl'_1 is preceded by an application of the added rule $pX \longrightarrow pt$. The situation when t appears in ql_2 after the application of the considered rule is similar. Either l' includes the whole t and then the application of the corresponding rule in Δ' results directly in ql'_2, or t is not a subterm of the right part of any sequential composition in l_2 and thus the application of the corresponding rule in Δ' is followed by an application of the added rule $qt \longrightarrow qX$ reaching the state ql'_2.

By repeating this procedure we finally get a wPRS Δ'' in normal form and terms t''_1,t''_2 satisfying $rt_1 \succ^{\Delta} st_2 \iff rt''_1 \succ^{\Delta''} st''_2$. $\square$

Mayr's proof for PRS now transforms the PRS Δ in normal form into the PRS Δ' in so-called *transitive normal form* satisfying $(X \longrightarrow Y) \in \Delta'$ whenever $X \succ^{\Delta} Y$. This step employs the local effect of rewriting under sequential rules in a parallel environment and vice versa. Intuitively, whenever there is a rewriting sequence

$$X\|Y \longrightarrow (X_1.X_2)\|Y \longrightarrow (X_1.X_2)\|Z \longrightarrow X_2\|Z$$

in a PRS in normal form, then the rewriting of each parallel component is independent in the sense that there are also rewriting sequences $X \longrightarrow X_1.X_2 \longrightarrow X_2$ and $Y \longrightarrow Z$. This does not hold for wPRS in normal form as the rewriting in one parallel component can influence the rewriting in other parallel components via a weak state unit. To get this independence back we introduce the concept of *passive steps* emulating changes of a weak state produced by the environment.

Definition 4. *A finite sequence of weak state pairs $PS = \{(p_i, q_i)\}_{i=1}^{n}$ satisfying $p_1 > q_1 \geq p_2 > q_2 \geq \cdots \geq p_n > q_n$ is called* passive steps*.*

Let Δ be a wPRS and PS be passive steps. By $\Delta + PS$ we denote a system Δ with an added rule $pX \longrightarrow qX$ for each (p, q) in PS and $X \in Const(\Delta)$. For all terms t_1, t_2 and weak states r, s we write

$$\begin{aligned}
rt_1 \succ^{\Delta+PS}_{triv} st_2 &\quad \textit{iff} \quad rt_1 \succ^{\Delta+PS} st_2 \textit{ via trivial rules,}\\
rt_1 \succ^{\Delta+PS}_{seq} st_2 &\quad \textit{iff} \quad rt_1 \succ^{\Delta+PS} st_2 \textit{ via sequential rules,}\\
rt_1 \succ^{\Delta+PS}_{par} st_2 &\quad \textit{iff} \quad rt_1 \succ^{\Delta+PS} st_2 \textit{ via parallel rules.}
\end{aligned}$$

Informally, $rt_1 \succ^{\Delta+PS} st_2$ means that the state rt_1 can be rewritten into state st_2 provided a weak state can be passively changed from p to q for every passive step (p, q) in PS. Thanks to the finiteness and 'weakness' of a weak state unit, the number of different passive steps is finite.

Definition 5. *Let wPRS Δ be in normal form. If for every $X, Y \in Const(\Delta)$, weak states r, s, and passive steps PS it holds that*

$$\begin{aligned}
rX \succ^{\Delta+PS} sY &\implies rX \succ^{\Delta+PS}_{triv} sY \textit{ then } \Delta \textit{ is in } \text{flatted normal form},\\
rX \succ^{\Delta+PS}_{seq} sY &\implies rX \succ^{\Delta+PS}_{triv} sY \textit{ then } \Delta \textit{ is in } \text{sequential flatted normal form},\\
rX \succ^{\Delta+PS}_{par} sY &\implies rX \succ^{\Delta+PS}_{triv} sY \textit{ then } \Delta \textit{ is in } \text{parallel flatted normal form}.
\end{aligned}$$

The following lemma says that it is sufficient to check reachability via sequential rules and via parallel rules in order to construct a wPRS in flatted normal form. This allows us to reduce the reachability problem for wPRS to the reachability problems for wPN and wPDA (i.e. to the reachability problems for PN and PDA).

Lemma 2. *If a wPRS is in both sequential and parallel flatted normal form then it is in flatted normal form as well.*

Proof. We assume the contrary and derive a contradiction. Let Δ be a wPRS in sequential and parallel flatted normal form. Let us choose passive steps PS and a rewriting sequence in $\Delta + PS$ leading from rX to sY such that $rX \not\succ^{\Delta+PS}_{triv} sY$, the number of applications of non-trivial rewrite rules applied in the sequence is minimal, and all steps of PS are used during the sequence. As the wPRS Δ is in both sequential and parallel flatted normal form, $rX \not\succ^{\Delta+PS}_{seq} sY$ and $rX \not\succ^{\Delta+PS}_{par} sY$. Hence, both sequential and parallel operators occur in the rewriting sequence. There are two cases.

1. Assume that a sequential operator appears first. The parallel operator is then introduced by the rule in the form $pU \longrightarrow qT\|S$ applied to a state $pU.t$, where t is a sequential term. From $q(T\|S).t \succ^{\Delta+PS} sY$ and the fact that at most one process constant can be removed in one rewriting step, it follows that in the rest of the sequence considered, the term $T\|S$ is rewritten onto a process constant (say V) and a weak state (say o) first. Let PS' be passive steps of PS between weak states p and o.
2. Assume that a parallel operator appears first. The sequential operator is then introduced by the rule in the form $pU \longrightarrow qT.S$ applied on a state $pU\|t$, where t is a parallel term. The rest of the sequence subsumes steps rewriting the term $T.S$ onto a process constant (say V) and a weak state (say o). Contrary to the previous case, these steps can be interleaved with steps rewriting the parallel component t and possibly changing weak state. Let PS' be passive steps of PS (between weak states p and o) merged with the changes of weak states caused by rewriting of t.

Consequently, we have a rewriting sequence in $\Delta + PS'$ from pU to oV with fewer applications of non-trivial rewrite rules. As the number of applications of non-trivial rewrite rules used in the original sequence is minimal we get $pU \not\succ^{\Delta+PS'}_{triv} oV$. This contradicts our choice of rX, sY, and PS. $\square$

Example 1. Here, we illustrate a possible change of passive steps (PS to PS') described in the second case of the proof above. Let us consider a wPRS Δ with weak states $r > p > q > t > v > o > s$ and the following rewrite rules

$$rX \longrightarrow pU\|Z \qquad pU \longrightarrow qT.S \qquad qZ \longrightarrow tY \qquad vT.S \longrightarrow oV \qquad oV\|Y \longrightarrow sY$$

as well as the following sequence $rX \succ^{\Delta+\{(t,v)\}} sY$, i.e.

$$r\underline{X} \longrightarrow p\underline{U}\|Z \longrightarrow q(T.S)\|\underline{Z} \longrightarrow \underline{t}(T.S)\|Y \stackrel{passive}{\longrightarrow} v(\underline{T.S})\|Y \longrightarrow o\underline{V\|Y} \longrightarrow sY,$$

where redices are underlined. The sequence constructed due to the case 2 is as:

$$pU \succ^{\Delta+\{(q,t),(t,v)\}} oV, \text{ i.e. } p\underline{U} \longrightarrow \underline{q}(T.S) \stackrel{passive}{\longrightarrow} \underline{t}(T.S) \stackrel{passive}{\longrightarrow} v(\underline{T.S}) \longrightarrow oV.$$

The following lemma employs the algorithms deciding the reachability problem for PDA and PN. Recall that the classes PDA and PN coincide with the classes of wPDA and wPN, respectively.

Lemma 3. *For every wPRS Δ in normal form, terms t_1, t_2 over $Const(\Delta)$, and weak states r, s of Δ a wPRS Δ' can be constructed such that Δ' is in flatted normal form and satisfies $rt_1 \succ^{\Delta} st_2 \iff rt_1 \succ^{\Delta'} st_2$.*

Proof. To obtain Δ' we enrich Δ by trivial rewrite rules transforming the system into sequential and parallel flatted normal forms, which suffices thanks to Lemma 2. Using algorithms deciding reachability for PDA and PN, our algorithm checks if there are some weak states r, s, constants $X, Y \in Const(\Delta)$, and passive steps $PS = \{(p_i, q_i)\}_{i=1}^{n}$ (satisfying $r \geq p_1$ and $q_n \geq s$ as weak states pairs beyond this range are of no use here) such that $rX \succ_{seq}^{\Delta+PS} sY \vee rX \succ_{par}^{\Delta+PS} sY$ and $rX \not\succ_{triv}^{\Delta+PS} sY$. We finish if the answer is negative. Otherwise we add to Δ rules $rX \longrightarrow p_1 Z_1$, $q_i Z_i \longrightarrow p_{i+1} Z_{i+1}$ for $i = 1, \ldots, n-1$, and $q_n Z_n \longrightarrow sY$, where $Z_1, \ldots, Z_n$ are fresh process constants (if $n = 0$ then we add just the rule $rX \longrightarrow sY$). The algorithm then repeats this procedure on the system with the added rules with one difference; the X, Y range over the constants of the original system Δ. This is sufficient as new constants occur only in trivial rules[4]. The algorithm terminates as the number of iterations is bounded by the number of pairs of states rX, sY of Δ, times the number of passive steps PS. The correctness follows from the fact that the added rules have no influence on reachability. □

Theorem 2. *The reachability problem for wPRS is decidable.*

Proof. (Sketch) Let Δ be a wPRS with states rt_1, st_2. We want to decide whether $rt_1 \succ^{\Delta} st_2$ or not. Clearly $rt_1 \succ^{\Delta} st_2 \iff rX \succ^{\Delta'} sY$, where X, Y are fresh constants and Δ' arises from Δ by the addition of the rules $rX \longrightarrow rt_1$ and $st_2 \longrightarrow sY$[5]. Hence we can directly assume that t_1, t_2 are process constants, say X, Y. Lemma 1 and Lemma 3 successively reduce the question whether $rX \succ^{\Delta} sY$ to the question whether $rX \succ^{\Delta'} sY$, where Δ' is in flatted normal form – note that Lemma 1 does not change terms t_1, t_2 if they are process constants. The definition of flatted normal form implies $rX \succ^{\Delta'} sY \iff rX \succ_{triv}^{\Delta'} sY$. Finally the relation $rX \succ_{triv}^{\Delta'} sY$ is easy to check. □

6 Conclusions

We have unified a view on some (non-conservative) extensions of Process Rewrite Systems. Comparing (up to bisimulation equivalence) the mutual expressiveness of the respective subclasses, we have added some new strict relations, including the class of Petri nets being less expressive than the class of PA processes extended with a finite state control unit. Finally, we have shown that a weak state unit extension (and thus a finite constraint system extension as well) of process rewrite systems keep the reachability problem decidable.

Acknowledgements. We would like to thank Hans Hüttel and Jiří Srba for discussions, comments, and pointers; and Luca Aceto for invaluable suggestions.

[4] If the system with added rules is not in sequential or parallel flatted normal form, then there is a counterexample with the constants X, Y of the original system Δ.

[5] If $t_2 = \varepsilon$ then this is not a correct rule. In this case we need to add to Δ' a rule $pt \longrightarrow qY$ for each rule $(pt \longrightarrow q\varepsilon) \in \Delta$.

References

[BCMS01] O. Burkart, D. Caucal, F. Moller, and B. Steffen. Verification on infinite structures. In *Handbook of Process Algebra*, pages 545–623. Elsevier, 2001.

[BEH95] A. Bouajjani, R. Echahed, and P. Habermehl. On the verification problem of nonregular properties for nonregular processes. In *LICS'95*. IEEE, 1995.

[BET03] A. Bouajjani, J. Esparza, and T. Touili. A generic approach to the static analysis of concurrent programs with procedures. *International Journal on Foundations of Computer Science*, 14(4):551–582, 2003.

[BT03] A. Bouajjani and T. Touili. Reachability Analysis of Process Rewrite Systems. In *Proc. of FST&TCS-2003*, volume 2914 of *LNCS*, pages 74–87. Springer, 2003.

[Büc64] J. R. Büchi. Regular canonical systems. *Arch. Math. Logik u. Grundlagenforschung*, 6:91–111, 1964.

[Cau92] D. Caucal. On the regular structure of prefix rewriting. *Theoretical Computer Science*, 106:61–86, 1992.

[DAC98] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett. Property specification patterns for finite-state verification. In Mark Ardis, editor, *Proc. 2nd Workshop on Formal Methods in Software Practice (FMSP-98)*, pages 7–15, New York, 1998. ACM Press.

[DY83] D. Dolev and A. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[Esp02] J. Esparza. Grammars as processes. In *Formal and Natural Computing*, volume 2300 of *LNCS*. Springer, 2002.

[HS04] H. Hüttel and J. Srba. Recursion vs. replication in simple cryptographic protocols. Submitted for publication, 2004.

[Jan95] P. Jančar. High Undecidability of Weak Bisimilarity for Petri Nets. In *Proc. of TAPSOFT*, volume 915 of *LNCS*, pages 349–363. Springer, 1995.

[JKM01] P. Jančar, A. Kučera, and R. Mayr. Deciding bisimulation-like equivalences with finite-state processes. *Theoretical Computer Science*, 258:409–433, 2001.

[KŘS03] M. Křetínský, V. Řehák, and J. Strejček. Process Rewrite Systems with Weak Finite-State Unit. Technical Report FIMU-RS-2003-05, Masaryk University Brno, 2003. to appear in ENTCS as Proc. of INFINITY 03.

[LS98] D. Lugiez and Ph. Schnoebelen. The regular viewpoint on PA-processes. In *Proc. of CONCUR'98*, volume 1466 of *LNCS*, pages 50–66, 1998.

[May81] E. W. Mayr. An algorithm for the general Petri net reachability problem. In *Proc. of 13th Symp. on Theory of Computing*, pages 238–246. ACM, 1981.

[May00] R. Mayr. Process rewrite systems. *Information and Computation*, 156(1):264–286, 2000.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[Mol96] F. Moller. Infinite results. In *Proc. of CONCUR'96*, volume 1119 of *LNCS*, pages 195–216. Springer, 1996.

[Mol98] F. Moller. Pushdown Automata, Multiset Automata and Petri Nets, MFCS Workshop on concurrency. *Electronic Notes in Theoretical Computer Science*, 18, 1998.

[MSS92] 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.

[Pet81] J. L. Peterson. *Petri Net Theory and the Modelling of Systems*. Prentice-Hall, 1981.

[SR90] V. A. Saraswat and M. Rinard. Concurrent constraint programming. In *Proc. of 17th POPL*, pages 232–245. ACM Press, 1990.

[Srb02] J. Srba. Roadmap of infinite results. *EATCS Bulletin*, (78):163–175, 2002. `http://www.brics.dk/~srba/roadmap/`.

[Str02] J. Strejček. Rewrite systems with constraints, EXPRESS'01. *Electronic Notes in Theoretical Computer Science*, 52, 2002.