Reachability for Extended Process Rewrite Systems

Vojtěch Řehák*
Faculty of Informatics, Masaryk University Brno,
Botanická 68a, 602 00 Brno, Czech Republic, rehak@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 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 [12]. 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.

Mayr [12] has also shown that the reachability problem (i.e. given terms t, t': is t reducible to t'?) for PRS is decidable. Most research 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 modelling recursive programs (without process creation) and PN in modelling dynamic creation of concurrent processes (without recursive calls). 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

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

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 [1] – the decidability of reachability and other problems relevant for an automatic verification are lost for sePA, including all its superclasses (see Figure 1).

The paper presents a hierarchy of PRS classes and their respective extensions of three types: PRS with finite constraint system (fcPRS [17], motivated by concurrent constraint programming, see e.g. [16]), state-extended PRS classes [7], and our new formalism of PRS with weak finite-state unit (wPRS, introduced in [9]). The notion of weakness employed in the wPRS formalism corresponds to that of weak automaton [15] in automata theory. In [9] we have shown that all the just mentioned extensions increase the expressive power of those PRS subclasses which do not subsume the notion of finite control. The classes in the hierarchy (depicted in Figure 1) are related by their expressive power with respect to (strong) bisimulation equivalence. Besides of the results on the classification of expressive power of extended PRS classes [9, 8], we have shown that the reachability problem remains decidable for the very expressive class of wPRS [8].

2 Extended PRS

In this section we recall the definitions of three different extensions of process rewrite systems, namely *state-extended PRS (sePRS)* [7], *PRS with a finite constraint system (fcPRS)* [17], and *PRS with a weak finite-state unit (wPRS)* [9]. For detailed description and intuitive explanation we refer to [8].

We distinguish four *classes of process terms*: '1' stands for terms consisting of a single process constant only (e.g. $\varepsilon \notin 1$), '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 Let $Act = \{a, b, \cdots\}$ be a countably infinite set of *atomic actions* and $\alpha, \beta \in \{1, S, P, G\}$ 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.

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

- (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 $m_0 \neq ff$.

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. To shorten our notation we prefer mt over (m, t). Instead of $(mt_1 \stackrel{a}{\longrightarrow} nt_2) \in R$ we usually write $(mt_1 \stackrel{a}{\longrightarrow} nt_2) \in \Delta$.

The semantics of an extended (α, β)-PRS system Δ is given by the corresponding labelled transition system $(S, Act(\Delta), \longrightarrow, m_0 t_0)$, where $S = M \times \{t \in \beta \mid Const(t) \subseteq Const(\Delta)\}$ 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):

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

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

and two common inference rules

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

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)*, see [13]. Let us note that the "non-extended" PRS can be defined as a special case of the extended formalism where M is a singleton.

3 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 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.

The strictness ('$\subsetneq$') between the PRS-hierarchy classes has been proved by Mayr [12], the strictness between the corresponding classes of PRS and fcPRS has been proved in [17], and the strictness relating fcPRSs and wPRSs is shown in [9]. Note the strictness relations wX $\subsetneq$ seX hold for all X = PA, PAD, PAN, PRS due to our reachability result for wPRS given in [8] and due to the full Turing-power of sePA [1]. These proofs together with Moller's result establishing that MSA $\subsetneq$ PN [14] and our proof that PN $\subsetneq$ sePA [8] complete the justification of Figure 1.

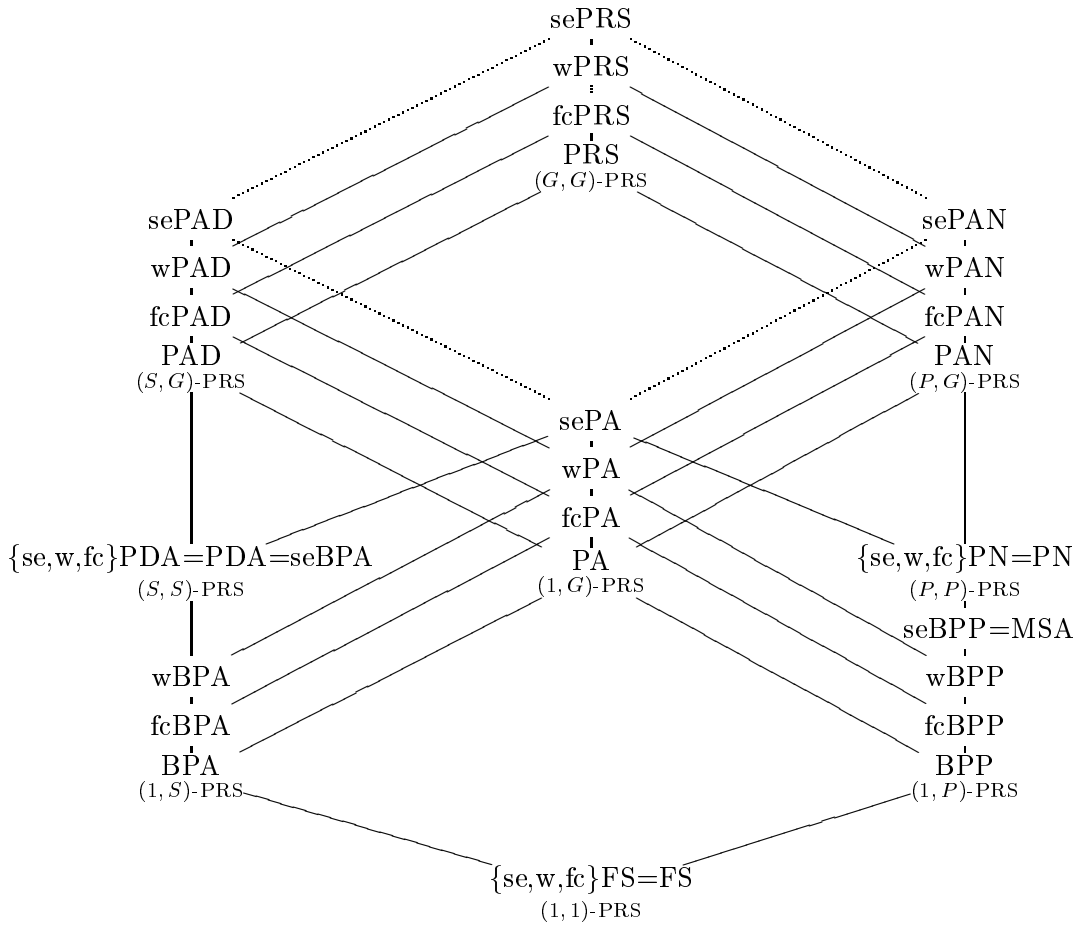

Figure 1: The hierarchy of classes defined by extended process rewrite systems.

4 Reachability Problem is Decidable for wPRS

By the reachability problem we mean to decide for a wPRS Δ with an initial state rt_0 and a given state st, whether the state st is reachable from the initial state rt_0 or not (st is reachable from rt_0 if a sequence of actions σ such that $rt_0 \stackrel{\sigma}{\longrightarrow} st$ exists).

It was proved by Mayr [12] that the reachability problem is decidable for PRS. Our proof exhibits a similar structure; first we reduce the general problem to the reachability problem for wPRS in so-called normal form (i.e. PRS 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 [11, 2]. The latter part of Mayr's proof for PRS 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 introduced the concept of *passive steps* emulating the changes of a weak state produced by the environment. For more details see [8].

To sum up, we proved that the reachability problem for wPRS is decidable. As "stronger" classes (i.e. sePA and its superclasses) of the hierarchy are Turing powerful (thus the problem is undecidable for them), the problem is solved for all the classes in the refined hierarchy.

5 Applications of the Reachability Result

By our opinion (sub)classes of wPRS are suitable for modelling some of the software systems which can be found in real-time control programs as well as in communication and cryptographic protocols. Let us mention that Hüttel and Srba [4] define a replicative variant of a calculus for Dolev and Yao's ping-pong protocols [3]. They show that the reachability problem for these protocols is decidable as it can be reduced to the reachability problem for wPRS, more precisely their replicative ping-pong protocols belong to the wPAD class.

Finally we mention another application of our decidability result exemplifying that the introduction of wPRS was well-motivated and contributes to the results on infinite-state systems. The decidability of the reachability for wPRS opens an easy way how to solve an open problem of a weak trace non-equivalence (for the definition see e.g. [6]) for wPRS and its subclasses.

Using the decidability of reachability for wPRS, it is easy to show that the weak trace set is recursive for every state of any wPRS. So far it has been known that the weak trace non-equivalence is semi-decidable for Petri nets (see e.g. [5]), pushdown processes (due to [2]), and PA processes (due to [10]). It follows from our result that the weak trace non-equivalence is semi-decidable for wPRS. Hence, the border of the semi-decidability was moved up to the class of wPRS in the hierarchy. Let us note that the semi-decidability result is new for some classes of the "non-extended" PRS hierarchy, too; namely PAN, PAD, and PRS. As other classes of our refined hierarchy (i.e. sePA and its superclasses) have a full Turing-power, the problem is undecidable for them. Hence, the problem is solved for all classes of the refined hierarchy.

References

[1] A. Bouajjani, R. Echahed, and P. Habermehl. On the verification problem of nonregular properties for nonregular processes. In *LICS'95*, pages 123–133. IEEE, 1995.

[2] J. R. Büchi. Regular canonical systems. *Archiv fur Mathematische Logik und Grundlagenforschung*, 6:91–111, 1964.

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

[4] H. Hüttel and J. Srba. Recursion vs. replication in simple cryptographic protocols. In *Proceedings of SOFSEM'05*, 2005. To appear.

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

[6] P. Jančar, J. Esparza, and F. Moller. Petri nets and regular behaviours. *Journal of Computer and System Sciences*, 59(3):476–503, 1999.

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

[8] M. Křetínský, V. Řehák, and J. Strejček. Extended Process Rewrite Systems: Expressiveness and Reachability. In *CONCUR'04*, volume 3170 of *LNCS*, pages 355–370. Springer, 2004.

[9] M. Křetínský, V. Řehák, and J. Strejček. On Extensions of Process Rewrite Systems: Rewrite Systems with Weak Finite-State Unit. In *INFINITY'03*, volume 98 of *ENTCS*, pages 75–88. Elsevier, 2004.

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

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

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

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

[14] F. Moller. A Taxonomy of Infinite State Processes, MFCS'98 Workshop on Concurrency. *ENTCS*, 18, 1998.

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

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

[17] J. Strejček. Rewrite systems with constraints, EXPRESS'01. *ENTCS*, 52, 2002.