Petri Nets Are Less Expressive Than State-Extended PA ⋆

Mojmír Křetínský [a,1] Vojtěch Řehák [a,1] Jan Strejček [a,*,2]

[a] *Faculty of Informatics, Masaryk University*
Botanická 68a, 60200 Brno, Czech Republic

Abstract

We show that the class of Petri nets is less expressive with respect to bisimulation equivalence than the class of PA processes extended with a finite state control unit.

Key words: Petri nets, process algebras, strong bisimulation, expressiveness

1 Introduction

Process rewrite systems (PRS) [7] are widely accepted as a formalism for finite specifications of infinite state systems. One reason for its popularity is that a variety of infinite state systems including *basic process algebras (BPA)*, *basic parallel processes (BPP)*, *pushdown processes (PDA)*, *process algebras (PA)* and *Petri nets (PN)* as well as the class of *finite state systems (FS)* can be uniformly defined as subclasses of PRS given by syntactic restrictions on the form of rewrite rules. Weaker syntactic restrictions give rise to another two classes, namely *PAD* (a common generalization of PDA and PA) and *PAN* (a common generalization of PDA and PN), while the class of general (unrestricted) PRS is a common generalization of PDA and PN. The relevance of PRS (and their subclasses) for modelling and analysing programs is shown, for example, in [2].

⋆ This result has been already mentioned in [6] with a proof just sketched.

* Corresponding author.
Email address: strejcek@fi.muni.cz (Jan Strejček).

[1] Supported by the research centre "Institute for Theoretical Computer Science (ITI)", project No. 1M0545.

[2] Supported by the Academy of Sciences of the Czech Republic, grant No. 1ET408050503.

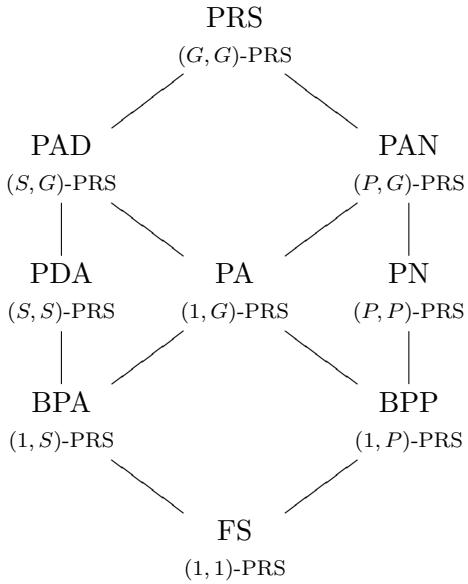

Fig. 1. The PRS-hierarchy

According to their expressive power, the PRS classes can be arranged into a *PRS-hierarchy* [7] depicted in Figure 1. A line between two classes means that the upper class is strictly more expressive than the lower class, in the sense that for every labelled transitions system (LTS) generated by the lower class there is a bisimilar system generated by the upper class, but not vice versa. Two classes of the hierarchy are expressively incomparable if no such relation can be derived from the picture. We recall that LTSs generated by (S, S)-PRS, i.e. sequential/prefix rewrite systems, are exactly those generated by PDA up to the relabelling of states, see [1].

A similar expressiveness hierarchy of classes of infinite state systems is presented in [9,4]. In contrast to the PRS framework, the hierarchy of [9] does not cover the classes combining both sequential and parallel compositions like PA and its superclasses. On the other hand, it contains the class of *multiset automata (MSA)* [4], also known as *parallel pushdowns (PPDA)* [9], which is not covered by the PRS-hierarchy. The MSA class can be defined as BPP systems extended with a finite state control unit. A systematic *state extension* of all PRS classes, denoted by the prefix *se-*, is considered in [5]. Clearly, the classes of FS, PDA and PN coincide with their respective state-extended counterparts, and seBPA coincides with the PDA class. The state-extended versions of the other classes do not coincide with any of the PRS classes.

The expressiveness hierarchy of all PRS classes and their state-extended versions is depicted in Figure 2. The dotted lines represent the relation where the strictness is just conjectured (see [11, Section 3.2.2] for further details). The shape of the hierarchy follows from the definition of state extension, the shape of the PRS hierarchy, Lemma 4.13 of [7] which says that a particular MSA is not bisimilar to any PAD, Theorem 20 of [4] showing a PN which

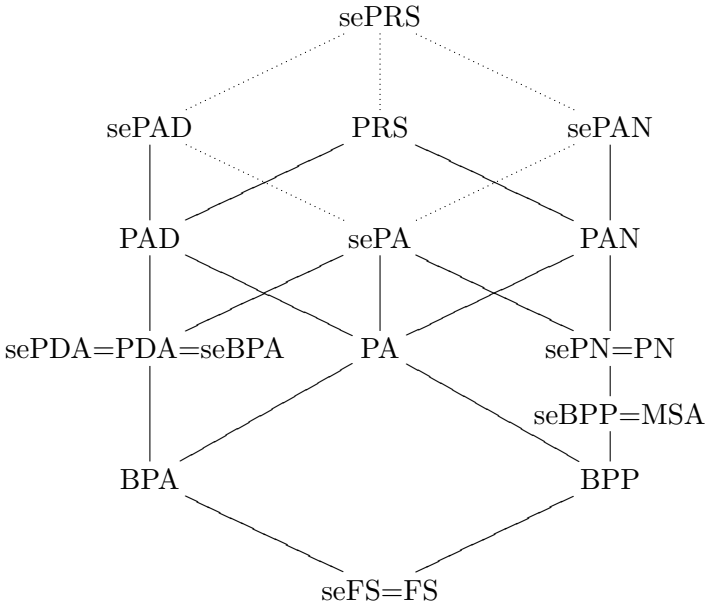

Fig. 2. The state-extended PRS-hierarchy

is not bisimilar to any MSA, and finally the result presented in this paper: every Petri net can be translated into a bisimilar sePA system. Further, the strictness of the relations between PN and sePA follows immediately from incomparability of the classes PDA and PN.[3] (Note that Lemma 4.13 of [7] says that a particular PN is not bisimilar to any PAD, but it is easy to see that the PN is actually an MSA system.)

The rest of the paper is divided into two sections: the next one recalls all the necessary definitions, and the other presents the main result.

2 Preliminaries

Let $Act = \{a, b, \ldots\}$ be a set of *actions*. A *labelled transition system* (LTS) is a triple $(S, \longrightarrow, s_0)$, where S is a set of *states*, $\longrightarrow \subseteq S \times Act \times S$ is a *transition relation*, and s_0 is the *initial state*. We write $s_1 \stackrel{a}{\longrightarrow} s_2$ instead of $(s_1, a, s_2) \in \longrightarrow$.

A binary relation R on S (of an LTS) is a *(strong) bisimulation* [8] iff whenever $(r_1, r_2) \in R$ then, for any $a \in Act$,

- if $r_1 \stackrel{a}{\longrightarrow} r_1'$ then, for some r_2', $r_2 \stackrel{a}{\longrightarrow} r_2'$ and $(r_1', r_2') \in R$, and
- if $r_2 \stackrel{a}{\longrightarrow} r_2'$ then, for some r_1', $r_1 \stackrel{a}{\longrightarrow} r_1'$ and $(r_1', r_2') \in R$.

[3] Loosely speaking in terms of Figure 2, our result means that an evident line connecting seBPP to sePA is replaced by that going from PN to sePA.

States r_1 and r_2 are *bisimilar*, written $r_1 \sim r_2$, iff $(r_1, r_2) \in R$ for some bisimulation R. Two LTSs are *bisimilar* if their initial states are bisimilar.

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

$$t ::= \varepsilon \mid X \mid t.t \mid t \| t$$

where ε is the *empty term*, $X \in Const$ is a process constant, and the operators '.' and '$\|$' stand for *sequential* and *parallel composition* respectively. We always work with equivalence classes of terms modulo commutativity and associativity of '$\|$', associativity of '.', and neutrality of ε, i.e. $\varepsilon.t = t.\varepsilon = t \| \varepsilon = t$. 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 - terms without parallel composition, e.g. $X.Y.Z$,
P – *parallel* terms - terms without sequential composition, e.g. $X \| Y \| Z$,
G – *general* terms - terms with arbitrarily nested sequential and parallel compositions, e.g. $(X.(Y \| Z)) \| W$.

Definition 1 *Let α, β be classes of process terms $\alpha, \beta \in \{1, S, P, G\}$ 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 $t_1 \overset{a}{\hookrightarrow} t_2$, where $t_1 \in \alpha \setminus \{\varepsilon\}$, $t_2 \in \beta$ are process terms and $a \in Act$ is an action,*
- *$t_0 \in \beta$ is the initial term.*

Given a PRS Δ, let $Const(\Delta)$ and $Act(\Delta)$ be the respective (finite) sets of all constants and all actions which occur in the rewrite rules of Δ. We often write $t_1 \overset{a}{\hookrightarrow} t_2 \in \Delta$ instead of $t_1 \overset{a}{\hookrightarrow} t_2 \in R$ where $\Delta = (R, t_0)$.

An (α, β)-PRS $\Delta = (R, t_0)$ determines an LTS whose states are process terms $t \in \beta$ over $Const(\Delta)$ and t_0 is the initial state. The transition relation $\longrightarrow$ is the least relation satisfying the following inference rules (recall that '$\|$' is commutative):

$$\frac{(t_1 \overset{a}{\hookrightarrow} t_2) \in \Delta}{t_1 \overset{a}{\longrightarrow} t_2} \qquad \frac{t_1 \overset{a}{\longrightarrow} t_2}{t_1 \| t \overset{a}{\longrightarrow} t_2 \| t} \qquad \frac{t_1 \overset{a}{\longrightarrow} t_2}{t_1.t \overset{a}{\longrightarrow} t_2.t}$$

The formalism of process rewrite systems can be extended to include a finite-state control unit in the following way.

Definition 2 *Let $M = \{m, n, \ldots\}$ be a set of* control states. *Let α, β be classes of process terms $\alpha, \beta \in \{1, S, P, G\}$ such that $\alpha \subseteq \beta$. An* (α, β)-sePRS

(state-extended process rewrite system) Δ *is a triple* (R, m_0, t_0)*, where*

- R *is a finite set of* rewrite rules *of the form* $(m, t_1) \stackrel{a}{\hookrightarrow} (n, t_2)$*, where* $t_1 \in \alpha \setminus \{\varepsilon\}$*,* $t_2 \in \beta$*,* $m, n \in M$*, and* $a \in Act$*,*
- $m_0 \in M$ *is the initial control state,*
- $t_0 \in \beta$ *is the initial term.*

$M(\Delta)$ *denotes the finite set of control states which occur in* Δ*.*

An (α, β)-sePRS $\Delta = (R, m_0, t_0)$ determines an LTS whose states are the pairs of the form (m, t) such that $m \in M(\Delta)$ and $t \in \beta$ is a process term over $Const(\Delta)$. The initial state of the transition system is the pair (m_0, t_0). The transition relation $\longrightarrow$ is the least relation satisfying the following inference rules:

$$\frac{((m, t_1) \stackrel{a}{\hookrightarrow} (n, t_2)) \in \Delta}{(m, t_1) \stackrel{a}{\longrightarrow} (n, t_2)} \qquad \frac{(m, t_1) \stackrel{a}{\longrightarrow} (n, t_2)}{(m, t_1 \| t) \stackrel{a}{\longrightarrow} (n, t_2 \| t)} \qquad \frac{(m, t_1) \stackrel{a}{\longrightarrow} (n, t_2)}{(m, t_1.t) \stackrel{a}{\longrightarrow} (n, t_2.t)}$$

To shorten our notation we write mt in lieu of (m, t).

Figure 2 shows the correspondence between some of the (α, β)-PRS classes and the classes of infinite state systems mentioned in the Introduction. Instead of (α, β)-sePRS we use the prefix 'se-' together with the acronym for the corresponding (α, β)-PRS class. For example, we use sePA rather than $(1, G)$-sePRS.

To see that the LTSs generated by labelled (place/transition) Petri nets (see e.g. [10]) are exactly those generated by (P, P)-PRS, we use arguments of [7,2]. Let X^n denote a parallel composition of n copies of X (in particular, X^0 denotes ε) for the moment. Let Δ be a (P, P)-PRS and $\{X_1, \ldots, X_k\} = Const(\Delta)$ be the set of its process constants. Each X_i corresponds to a *place* P_i in the net and the number of occurrences of X_i in a process term corresponds to the number of *tokens* in this place P_i. Hence, a process term $X_1^{p_1} \| \ldots \| X_k^{p_k}$ corresponds to the *marking* $(p_1, \ldots, p_k)$, where p_i is the number of tokens in the place P_i. Finally, each rewrite rule in Δ

$$X_1^{l_1} \| \ldots \| X_k^{l_k} \stackrel{a}{\hookrightarrow} X_1^{r_1} \| \ldots \| X_k^{r_k},$$

where $l_i, r_i \geq 0,\ i = 1, \ldots, k$, corresponds to an a-labelled *transition* having places $l_i, l_i > 0$, in its preset and places $r_i, r_i > 0$, in its postset. We shall write this PN transition as $(l_1, \ldots, l_k) \stackrel{a}{\hookrightarrow} (r_1, \ldots, r_k)$.

As we employ the place/transition description of PN, X^n stands only for a sequential composition of n copies of X in the rest of this paper.

3 Main Result

The basic idea of our proof is similar to the idea of Hirshfeld's proof of the fact that PN and MSA are language equivalent. This proof was presented in his talk [3], but was never published.

The heart of our argument is the construction of an sePA Δ' that is bisimilar to a given PN Δ. The main difficulty in this construction is to mimic the number of tokens at the places of a PN. To this end, we may use two types of sePA memory:

- a finite control unit, which cannot represent an unbounded counter,
- a term of an unbounded length, where just one constant can be rewritten in one step.

Intuition

Our construction of an sePA Δ' can be reformulated on an 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

$$(l_1, \ldots, l_k) \stackrel{a}{\hookrightarrow} (r_1, \ldots, r_k)$$

has an effect of a currency exchange from p_i to $p_i - l_i + r_i$ for all i.

An sePA reseller Δ' will have k finite pockets (in its control states) 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. He must obey sePA rules to mimic a PN transition, i.e. he may use all his pockets, but just one of his accounts in one exchange–transition.

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 $account_i$ as he is visiting the other accounts. Therefore, Δ' needs sufficiently large (but finite) pockets and sufficiently high (and still fixed) limits for $pocket_i \leftrightarrow account_i$ transfers. In what follows, we show that these bounds exist.

Bounds

In one step, the amount of the i-th currency can be changed at most by

$$L_i = max\,\{\, l_i, r_i \mid (l_1, \ldots, l_k) \stackrel{a}{\hookrightarrow} (r_1, \ldots, r_k) \text{ is a transition of the PN } \Delta\},$$

thus the upper bound for the total effect of k consecutive steps can be set up to $M_i = k \cdot L_i$. Any rebalancing of the i-th pocket sets its value into

$$\{M_i, \ldots, 2M_i - 1\}$$

(or into $\{0, \ldots, 2M_i - 1\}$ if $account_i$ is empty). Hence, after k transitions the value of $pocket_i$ is in

$$\{0, \ldots, 3M_i - 1\}.$$

Then the next rebalancing takes place and $account_i$ is increased or decreased (if it is not empty) by M_i to achieve the (rebalanced) value of $pocket_i$ in $\{M_i, \ldots, 2M_i - 1\}$.

Construction

Each state of sePA Δ' consists of a control state and a term (parallel composition of k stacks, in fact just counters, representing accounts). Each control state is a member of the following product.

$$\underset{\textit{update controller}}{\{1, \ldots, k\}} \quad \times \quad \underset{pocket_1}{\{0, \ldots, 3M_1 - 1\}} \quad \times \quad \ldots \quad \times \quad \underset{pocket_k}{\{0, \ldots, 3M_k - 1\}}$$

The *update controller* goes in round-robin fashion on its 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 $i = 1, \ldots, k$. The i-th stack t_i is of the form $X_i^n.B_i$ where $n \geq 0$. B_i represents the bottom of the i-th stack, and each X_i represents M_i tokens at place P_i.

Given an initial marking $\alpha_0 = (p_1, \ldots, p_k)$ of 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 $m_i = p_i - n_i M_i$ and $t_i = X_i^{n_i}.B_i$. In other words, we have $p_i = m_i + n_i M_i$ and moreover $m_i \in \{M_i, \ldots, 2M_i - 1\}$ if $p_i \geq M_i$ (i.e. p_i is big enough).

To each transition $(l_1, \ldots, l_k) \overset{a}{\hookrightarrow} (r_1, \ldots, r_k)$ of PN Δ we construct the set of sePA rules

$$(s, m_1, \ldots, m_k)\, t \overset{a}{\hookrightarrow} (s', m_1', \ldots, m_k')\, t'$$

such that they obey the following conditions:

- Update controller conditions:
  - $s, s' \in \{1, \ldots, k\}$ and

  - $s' = (s \bmod k) + 1$.
- 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 "is enabled"), and
  - if $i \neq s$ then $m_i' = m_i - l_i + r_i$.
- The case $i = s$ means to specify m_s' and the terms t, t' that effect the $pocket_s \leftrightarrow account_s$ rebalancing transfer. These are given by the rules of the following table. 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	$\{2M_s, \ldots, 3M_s - 1\}$	$\overline{m_s} - M_s$	$X_s.X_s$

Now we are ready to formulate and prove our main result.

Theorem 3 *Every PN can be translated into a strongly bisimilar sePA.*

PROOF. Let Δ be an arbitrary PN with an initial marking α_0. According to the construction given above, we build the sePA Δ' with the initial state β_0. In the rest of the proof, we show that the binary relation

$$
\begin{aligned}
R = \{(\alpha, \beta) \mid\ & \alpha = (p_1, \ldots, p_k) \text{ is a marking of } \Delta\ , \\
& \beta = (s, m_1, \ldots, m_k)\, X_1^{n_1}.B_1 \| \ldots \| X_k^{n_k}.B_k \text{ is a state of } \Delta'\ , \\
& s \in \{1, \ldots, k\}, \text{ and,} \\
& \text{for all } i = 1, \ldots, k, \text{ it holds that} \\
& \qquad p_i = m_i + n_i M_i \ \wedge \\
& \qquad m_i < 2M_i + (s - i \bmod k) L_i \ \wedge \\
& \qquad \text{if } n_i > 0 \text{ then } M_i - (s - i \bmod k) L_i \leq m_i \ \}
\end{aligned}
$$

is a strong bisimulation.

It follows directly from the construction that the pair (α_0, β_0) of the initial states of Δ and Δ' is in R.

We follow the definition of bisimulation to prove that the relation R is a bisimulation. Let $\alpha = (p_1, \ldots, p_k)$ be a marking of PN Δ and

$$\beta = (s, m_1, \ldots, m_k)\, X_1^{n_1}.B_1 \| \ldots \| X_k^{n_k}.B_k$$

be a state of sePA Δ' such that $(\alpha, \beta) \in R$.

Let us assume that a transition $(l_1, \ldots, l_k) \stackrel{a}{\hookrightarrow} (r_1, \ldots, r_k)$ is fired in α and leads to $\alpha' = (p'_1, \ldots, p'_k)$, i.e. $p_i \geq l_i$ and $p'_i = p_i - l_i + r_i$, for all $i = 1, \ldots, k$. According to the definition of bisimulation, we will show that there is also a state β' of Δ' and a transition with a label a leading from β to β' such that $(\alpha', \beta') \in R$. Looking into the construction, it is easy to see that such a transition exists if $m_i \geq l_i$ for all $i = 1, \ldots, k$. These inequalities can be easily proved as follows. For each $i = 1, \ldots, k$, we discuss two cases:

- If $n_i = 0$ then $(\alpha, \beta) \in R$ implies $p_i = m_i + n_i M_i = m_i$. This, together with $p_i \geq l_i$, leads directly to the desired $m_i \geq l_i$.
- If $n_i > 0$ then $(\alpha, \beta) \in R$ implies $M_i - (s - i \bmod k)L_i \leq m_i$. As M_i is defined to be equal to $k \cdot L_i$, we get that $m_i \geq L_i$. Now, the definition of L_i implies $L_i \geq l_i$ that directly results in $m_i \geq l_i$.

It remains to show that $(\alpha', \beta') \in R$. This can be obtained by a straightforward inspection of the definitions of all the rule types.

The symmetric case, starting with a transition from β, proceeds in a similar way. Hence, Δ and Δ' are bisimilar. $\square$

4 Conclusion

We have presented an algorithm transforming any given Petri net to a bisimilar sePA process, i.e. a PA process extended with a finite state control unit. We note that the sePA system constructed does not need to be isomorphic to the original PN system; it can be exemplified by the states which differ by the values of the update controller only. To our best knowledge, it is not known whether the result can be refined to isomorphism rather than bisimilarity.

References

[1] D. Caucal, On the regular structure of prefix rewriting, Theoretical Computer Science 106 (1992) 61–86.

[2] J. Esparza, Grammars as processes, in: Formal and Natural Computing, vol. 2300 of Lecture Notes in Computer Science, Springer-Verlag, 2002.

[3] Y. Hirshfeld, Comparing the expressive power of simple process algebras, a talk at Grenoble-Alpes d'Huez European School of Computer Science, Methods and Tools for the Verification of Infinite State Systems (1997).

[4] Y. Hirshfeld, F. Moller, Pushdown automata, multiset automata, and Petri nets, Theoretical Computer Science 256 (1-2) (2001) 3–21.

[5] P. Jančar, A. Kučera, R. Mayr, Deciding bisimulation-like equivalences with finite-state processes, Theoretical Computer Science 258 (2001) 409–433.

[6] M. Křetínský, V. Řehák, J. Strejček, Extended Process Rewrite Systems: Expressiveness and Reachability, in: Proceedings of 15th International Conference on Concurrency Theory (CONCUR'04), vol. 3170 of Lecture Notes in Computer Science, Springer-Verlag, 2004.

[7] R. Mayr, Process Rewrite Systems, Information and Computation 156 (1) (2000) 264–286.

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

[9] F. Moller, Infinite results, in: Proceedings of the 7th International Conference on Concurrency Theory (CONCUR'96), vol. 1119 of Lecture Notes in Computer Science, Springer-Verlag, 1996.

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

[11] V. Řehák, On extensions of process rewrite systems, Ph.D. thesis, Faculty of Informatics, Masaryk University, Brno (2007).