Regulated Multiset Rewriting Systems

Matej Trojáš, Samuel Pastva, David Šafránek, and Luboš Brim
Faculty of Informatics, Masaryk University, Brno, Czech Republic

Abstract. Multiset rewriting systems provide a formalism particularly suitable for the description of biological systems. We present an extension of this formalism with additional controls on the derivations as a tool for reducing possible non-deterministic behaviour by providing additional knowledge about the system. We introduce several regulation mechanisms and compare their generative power.

1 Introduction

Multiset rewriting systems (MRSs) [15] provide a modelling formalism suitable in particular for describing biological systems [1,2]. MRS can be considered as a low-level formalism, describing how multisets of elements can be modified by applying rewriting rules representing biological processes. Such a basic mechanism can serve as a base to build higher-level languages (e.g. rule-based modelling [3,6,12], membrane computing [17], Petri nets [4]), providing more convenient approaches to describe interesting biological phenomena. The mechanistic nature of MRS supports extendible and decomposable system construction by separate rule description, which becomes extremely useful when newly discovered biological knowledge needs to be introduced into the system.

Due to the non-deterministic behaviour of MRS, there are types of knowledge that cannot be simply covered by adding new rules or modifying existing ones. For example, consider a situation where a rule is always followed by another one (e.g. complex formation and its phosphorylation in circadian clock mechanism [8]), two rules cannot take place after each other (e.g. cell division after entering apoptosis phase [20]), a rule always has a priority over another one (e.g. cell differentiation [11]), or a rule cannot be used in a particular context (e.g. explicit modelling of inhibition in a signalling pathway).

A possible approach to capture the above-described types of behaviour is to employ some rewriting control in the form of a regulated application of rewriting rules. Regulation of rewriting thus could provide an additional mechanism for controlling the rule application by restricting the conditions when a rule can be enabled. Regulated rewriting is well-established in the area of string grammars [7] and has been extended to several other modelling formalisms such as Petri nets [13] or membrane computing [10]. In [9] authors introduced an approach to regulate MRS using first-order logical formulae over variables, allowing to express constraints to rules application. While this approach significantly increases the expressive power of MRS, it is not particularly suitable for reasoning about the type of behaviour based on relationships among rewriting rules as described in the examples above.

In this paper, we introduce separate classes of regulation mechanisms of multiset rewriting systems. They are based on our experience with modelling in systems biology [18,21,22] and correspond to the form of specification experts outside of computer science expect. We define several regulation mechanisms, namely: regular rewriting restricted by regular language over rules, ordered rewriting restricted by strict partial order on rules, programmed rewriting restricted by successors for each rule, conditional rewriting restricted by prohibited context for each rule, and concurrent-free rewriting restricted by resolving concurrent behaviour of multiple rules. We investigate and compare the generative power of the defined types of regulations and show the effect of regulations on the expressive power (both summarised in Figure 14).

2 Multiset rewriting systems

In this section, we recall some definitions and known results about multisets and rewriting systems over them. Intuitively, a multiset is a set of elements with allowed repetitions. A multiset rewriting rule describes how a particular multiset is transformed into another one. A multiset rewriting system consists of a set of multiset rewriting rules, defining how the system can evolve, and an initial multiset, representing the starting point for the rewriting.

Let $S$ be a finite set of elements. A multiset over $S$ is a total function $M : S \rightarrow \mathbb{N}$ (where $\mathbb{N}$ is the set of natural numbers including 0). For each $a \in S$ the multiplicity (the number of occurrences) of $a$ is the
number \( M(a) \). Operations and relations over multisets are defined in a standard way, taking into account the repetition of elements (for details we refer to [4]).

A multiset rewriting rule over \( S \) is a pair \( \mu = (\ast \mu, \mu^\ast) \) of multisets over \( S \), written as \( \mu : \ast \mu \rightarrow \mu^\ast \) for convenience. The rule rewrites elements specified in the left-hand side \( \ast \mu \) to elements specified in the right-hand side \( \mu^\ast \).

A multiset rewriting system (MRS) over \( S \) is a pair \( \mathcal{M} = (\mathcal{X}, \mathcal{M}_0) \), where \( \mathcal{X} \) is a finite set of multiset rewrite rules and \( \mathcal{M}_0 \) is the initial multiset (state), both over \( S \).

We denote by \( \mathcal{MRS} \) the class of (non-regular) multiset rewriting systems. An example of an MRS is given in Figure 1.

\[
\mathcal{M} = \left\{ \begin{array}{l}
\mathcal{M}_0 = \{ A, A \} \\
\mathcal{X} = \left\{ \mu_1 : \{ A, A \} \rightarrow \{ B, A \}, \mu_2 : \{ A \} \rightarrow \{ X \} \right\}
\end{array} \right\}
\]

Fig. 1. Example of an MRS over elements \( S = \{ A, B, C, X \} \).

Let \( M \) be a multiset and \( \mathcal{M} = (\mathcal{X}, \mathcal{M}_0) \) an MRS, both over \( S \). A rule \( \mu \in \mathcal{X} \) is enabled at \( M \) if \( \ast \mu \subseteq M \).

The application of an enabled rule \( \mu \in \mathcal{X} \) to \( M \), written \( M \rightarrow_{\mu} M' \), creates a multiset \( M' = (M \setminus \ast \mu) \cup \mu^\ast \). A run \( \pi \) of \( \mathcal{M} \) is an infinite sequence of multisets \( \pi = \mathcal{M}_0 \mathcal{M}_1 \mathcal{M}_2 \cdots \) such that for any step \( i > 0 \) holds that \( \mathcal{M}_{i-1} \rightarrow_{\mu_i} \mathcal{M}_i \) for some \( \mu_i \in \mathcal{X} \). We denote by \( \pi[i] \) the multiset created in step \( i \). A run label \( \overrightarrow{\pi} \) of a run \( \pi \) is an infinite sequence of rules \( \overrightarrow{\pi} = \mu_1 \mu_2 \mu_3 \cdots \) such that for any step \( i > 0 \) holds that \( \pi[i-1] \rightarrow_{\mu_i} \pi[i] \). We denote by \( \overrightarrow{\pi}[1] \) the rule applied in step \( i \).

The semantics of the system \( \mathcal{M} \) is a (potentially infinite) set \( \mathcal{L}(\mathcal{M}) \) of all possible runs such that \( \forall \pi \in \mathcal{L}(\mathcal{M}) \), \( \pi[0] = \mathcal{M}_0 \) (runs start in the initial multiset). In order to ensure the infiniteness of runs, we implicitly assume the presence of a special empty rule \( \varepsilon = (\emptyset, \emptyset) \). We require that this rule can be applied only when no other rule of the system is enabled. It also ensures that the set of rules \( \mathcal{X} \) is always non-empty. In Figure 2 we give an example of runs.

\[
\begin{array}{ccccccc}
A, A & \xrightarrow{\mu_1} & A, B & \xrightarrow{\mu_2} & A, C & \xrightarrow{\mu_4} & A, B & \xrightarrow{\mu_5} & X, B & \xrightarrow{\mu_5} & \ldots \in \mathcal{L}(\mathcal{M}) \\
A, A & \xrightarrow{\mu_2} & A, X & \xrightarrow{\varepsilon} & X, X & \xrightarrow{\varepsilon} & X, X & \xrightarrow{\varepsilon} & X, X & \xrightarrow{\varepsilon} & \ldots \in \mathcal{L}(\mathcal{M})
\end{array}
\]

Fig. 2. Example of runs of the MRS from Figure 1, including their run labels.

## 3 Regulated rewriting

The semantics of MRS is a set of all possible runs induced by its non-deterministic nature. In this section, we introduce regulated MRS, which restricts the applicability of rules, leading to a reduced set of possible runs (reduced semantics).

A regulated multiset rewriting system (rMRS) is a triple \( \overrightarrow{\mathcal{M}} = (\mathcal{X}, \mathcal{M}_0, \zeta) \) such that \( \mathcal{M} = (\mathcal{X}, \mathcal{M}_0) \) is an MRS and \( \zeta \) is a regulation, which is used to reduce the set of possible runs \( \mathcal{L}(\mathcal{M}) \) to \( \mathcal{L}(\overrightarrow{\mathcal{M}}) \). The exact definition of \( \zeta \) depends on the particular reduction mechanisms and is specified for individual types of suggested regulation below.

### 3.1 Regular rewriting

Informally, the idea is to define an \( \omega \)-regular language \( \zeta \) over rules, that is, a regular language over infinite words. Then, only runs with the run label from this language are allowed. The set of words \( \zeta \), due to its infiniteness, is almost always (though not necessarily) described via some more convenient mechanism (such as \( \omega \)-regular expression).

In a regular multiset rewriting system, the regulation \( \zeta \) is defined as an \( \omega \)-regular language over set of rules \( \mathcal{X} \). The set of possible runs \( \mathcal{L}(\mathcal{M}) \) is reduced to a set of regular runs \( \mathcal{L}(\overrightarrow{\mathcal{M}}) = \{ \pi \in \mathcal{L}(\mathcal{M}) \mid \overrightarrow{\pi} \in \zeta \} \). Moreover, we require that \( \forall \overrightarrow{\pi} \in \zeta \exists \pi \in \mathcal{L}(\mathcal{M}) \) (there are no extra words in \( \zeta \)).
We denote by RR the class of regular multiset rewriting systems. An example of an RR system is available in Figure 3 with an example of valid and invalid runs in Figure 4.

\[
\mathcal{M} = \left\{ \begin{array}{l}
M_0 = \emptyset, \\
\zeta = (\mu_1, \mu_2)^* \cdot \epsilon^*, \\
X = \{ \mu_1 : \emptyset \rightarrow \{A\}, \mu_2 : \{A\} \rightarrow \{B\}, \mu_3 : \{B\} \rightarrow \emptyset \} \end{array} \right\}
\]

Fig. 3. Example of a regular multiset rewriting system over a set of elements \(S = \{A, B\}\). The regulation \(\zeta\) allows a particular finite sequence of rules followed by infinite application of rule \(\epsilon\).

3.2 Ordered rewriting

The ordered rewriting regulation is based on a strict partial order on rules. Thus, only runs with the run label, which does not break the ordering in the immediate successors, are allowed.

In an ordered multiset rewriting system, the regulation \(\zeta\) is a strict partial order over a set of rules. For any two rules \(\mu, \mu' \in X\) we write \(\mu < \mu'\) iff \((\mu, \mu') \in \zeta\). The set of possible runs \(L(M)\) is reduced to a set of ordered runs \(L_1(M) = \{ \pi \in L(M) \mid \forall i > 0. \pi[i+1] < \pi[i] \} \).

We denote by OR the class of ordered multiset rewriting systems. An example of an OR system is available in Figure 5 with an example of valid and invalid runs in Figure 6.

\[
\mathcal{M} = \left\{ \begin{array}{l}
M_0 = \emptyset, \\
\zeta = \{ (\mu_2, \mu_2) \} \\
X = \{ \mu_1 : \emptyset \rightarrow \{A\}, \mu_2 : \{A\} \rightarrow \emptyset \} \end{array} \right\}
\]

Fig. 5. Example of an ordered multiset rewriting system over a set of elements \(S = \{A\}\). The regulation \(\zeta\) defines order \(\mu_1 < \mu_2\) on rules \(\mu_1, \mu_2\), which does not allow \(\mu_1\) to be immediately used after \(\mu_2\).

3.3 Programmed rewriting

Informally, for every rule, there exists a set of successor rules. Then, only runs with the run label where each rule is followed by its successor are allowed.
In a programmed multiset rewriting system, the regulation $\zeta : \mathcal{X} \rightarrow 2^\mathcal{X}$ is a successor function, assigning to each rule a set of successor rules. The set of possible runs $\mathcal{L}(\mathcal{M})$ is reduced to a set of programmed runs $\mathcal{L}(\mathcal{M}) = \{ \pi \in \mathcal{L}(\mathcal{M}) \mid \forall i > 0. \, \overline{\pi}[i + 1] \in \zeta(\overline{\pi}[i]) \}$. We denote by $\mathcal{PR}$ the class of programmed multiset rewriting systems. An example of a $\mathcal{PR}$ system is available in Figure 7 with an example of valid and invalid runs in Figure 8.

$$\mathcal{M} = \left\{ \begin{array}{l} M_0 = \emptyset, \quad \zeta = \{ \mu_1 \rightarrow \{ \mu_2 \}, \mu_2 \rightarrow \{ \mu_1 \} \}, \\
\mathcal{X} = \{ \mu_1 : \emptyset \rightarrow \{ A \}, \mu_2 : \emptyset \rightarrow \{ B \} \} \end{array} \right\}$$

Fig. 7. Example of a programmed multiset rewriting system over a set of elements $S = \{ A, B \}$. The regulation $\zeta$ allows only alternate application of rules $\mu_1$ and $\mu_2$.

$$\begin{array}{c}
\mu_1 \quad \mu_2 \\
\mu_1 \quad \mu_2
\end{array}$$

Fig. 8. Example of valid and invalid runs of the $\mathcal{PR}$ system from Figure 7. Run $\pi_1 \in \mathcal{L}(\mathcal{M})$ because rules $\mu_1$ and $\mu_2$ are alternating and run $\pi_2 \notin \mathcal{L}(\mathcal{M})$ because the run label $\overline{\pi}_2$ violates the successor function, e.g. $\overline{\pi}_2[2] \notin \zeta(\overline{\pi}[1])$ (i.e. $\mu_1 \notin \zeta(\mu_1)$).

### 3.4 Conditional rewriting

The idea is to define multiset of elements (or multiple of them) for each rule, which represents a prohibited context where the rule cannot be used. Then, a run is valid, if in every step, there prohibited context of the used rule is not subset of the current multiset.

In a conditional multiset rewriting system, the regulation $\zeta$ is defined as a conditional function, assigning a set of multisets over elements $S$ to each rule. Each of these multisets represent the prohibited context in which the rule cannot be applied, i.e. it is not a subset of the current state. The set of possible runs $\mathcal{L}(\mathcal{M})$ is reduced to a set of conditional runs $\mathcal{L}(\mathcal{M}) = \{ \pi \in \mathcal{L}(\mathcal{M}) \mid \forall i > 0. \, \forall A \in \zeta(\overline{\pi}[i]). \, A \not\subseteq \overline{\pi}[1 - 1] \}$. We denote by $\mathcal{CR}$ the class of conditional multiset rewriting systems. An example of a $\mathcal{CR}$ system is available in Figure 9 with an example of valid and invalid runs in Figure 10. Note that in the case when there is only one prohibited context defined for a rule, we simplify the notation by omitting the parent set.

$$\mathcal{M} = \left\{ \begin{array}{l} M_0 = \emptyset, \quad \zeta = \{ \mu_1 \rightarrow \{ B \}, \mu_2 \rightarrow \emptyset \}, \\
\mathcal{X} = \{ \mu_1 : \emptyset \rightarrow \{ A \}, \mu_2 : \emptyset \rightarrow \{ B \} \} \end{array} \right\}$$

Fig. 9. Example of a conditional multiset rewriting system over a set of elements $S = \{ A, B \}$. The regulation $\zeta$ limits the application of rule $\mu_1$ to states where an element $B$ is not present.

### 3.5 Concurrent-free rewriting

The idea is to detect concurrent rules and assign a priority to one of them. Then, only runs where a prioritised rule was used in place of non-prioritised ones are valid. We say two rules $\mu, \mu'$ are concurrent iff $\mu \cap \mu' \neq \emptyset$ (i.e., rules rewrite common elements).

In a concurrent-free multiset rewriting system, the regulation $\zeta \subseteq \mathcal{X} \times \mathcal{X}$ is a binary relation over rules, for which holds that (1) the relation is irreflexive, (2) any two rules $\mu, \mu'$ with $(\mu, \mu') \in \zeta$ are concurrent, and (3) if $(\mu, \mu') \in \zeta$, then $(\mu', \mu) \notin \zeta^+$, where $\zeta^+$ is the transitive closure of relation $\zeta$. The third condition
ensures acyclicity, which disables ambiguous priority resolving. The set of possible runs \( \mathcal{L}(\mathcal{M}) \) is reduced to a set of \emph{concurrent-free} runs \( \mathcal{L}(\mathcal{M})^c = \{ \pi \in \mathcal{L}(\mathcal{M}) | \forall i > 0 \forall \mu \in \mathcal{X} \text{ enabled at } \pi[i], (\mathcal{P}[i+1], \mu) \notin \zeta \} \).

We denote by \( \mathcal{CFR} \) the class of concurrent-free multiset rewriting systems. An example of a \( \mathcal{CFR} \) system is available in Figure 11 with an example of valid and invalid runs in Figure 12.

\[\mathcal{M} = \left\{ m_0 = \{A\}, \ z = \{ (\mu_2, \mu_2) \}, \ \{\mathcal{X} = \{ \mu_1 : \{A\} \rightarrow \{A, B\}, \mu_2 : \{A, B\} \rightarrow \{A\}, \mu_3 : \{A\} \rightarrow \emptyset \} \right\}\]

Fig. 11. Example of a concurrent-free multiset rewriting system over a set of elements \( S = \{A, B\} \). The regulation \( \zeta \) gives priority to rule \( \mu_2 \) over the concurrent rule \( \mu_3 \) in the case they are both enabled, which makes sure element \( A \) cannot be removed until any \( B \)s are present. Please note that the priority does not need to be resolved for every concurrent pair of rules (e.g., \( \mu_1 \) and \( \mu_2 \)).

\[\mathcal{M} = \{ \mu_1 : \{A, B\} \rightarrow \{A, B, \}, \mu_2 : \{A, B\} \rightarrow \{A\}, \mu_3 : \{A\} \rightarrow \emptyset \} \]

Fig. 12. Example of valid and invalid runs of the \( \mathcal{CFR} \) system from Figure 11. Run \( \pi_1 \in \mathcal{L}(\mathcal{M}) \) because rule \( \mu_2 \) was used with priority when rule \( \mu_3 \) was also enabled. Run \( \pi_2 \notin \mathcal{L}(\mathcal{M}) \) the rule \( \mu_3 \) was used when the prioritised rule \( \mu_2 \) was also enabled.

| Acronym | Class         | Regulation                                                                 |
|---------|---------------|-----------------------------------------------------------------------------|
| RR      | regular       | run labels belong to \( \omega \)-regular language                        |
| OR      | ordered       | subsequent rules are restricted by partial order                           |
| PR      | programmed    | successors of rules are explicitly specified                               |
| CR      | conditional   | rules cannot be used in context of specified elements                       |
| CFR     | concurrent-free | priority is given to one of the concurrent rules                          |

Fig. 13. Summary of defined regulated multiset rewriting systems.

Fig. 14. Summary of generative power comparison among classes of regulated systems.
4 Properties of regulated multiset rewriting systems

In this section, we state some general properties of the classes of rewriting systems defined in the previous section (summarised in Figure 13). First, we compare the classes on the level of generative power, i.e. runs they can generate, and then we discuss their expressive power, i.e. the functions they can compute (both summarised in Figure 14).

We say two runs \( \pi, \pi' \) are **equal** if \( \forall i \in \mathbb{N}, \pi[i] = \pi'[i] \) (regardless of the run labels). We say two systems \( M, M' \) are equal if their corresponding sets of runs \( \mathcal{L}(M), \mathcal{L}(M') \) are equal.

Let \( C \) be an arbitrary class of regulated systems. We define the **generative power** of a class \( C \) as a set \( g(C) = \{ \mathcal{L}(M) \mid M \in C \} \) of all possible sets of runs. We can compare classes on their level of generative power defining the following operators for any two classes \( C_1, C_2 \):

\[
\begin{align*}
\subset & \quad C_1 \subset C_2 \iff g(C_1) \subseteq g(C_2) \\
\supset & \quad C_1 \supset C_2 \iff g(C_1) \not\subseteq g(C_2) \\
\subseteq & \quad C_1 \subseteq C_2 \iff g(C_1) \subseteq g(C_2) \\
\supset & \quad C_1 \supset C_2 \iff g(C_1) \not\subseteq g(C_2)
\end{align*}
\]

4.1 Generative power comparison

All regulated classes are strictly more generative than non-regulated class.

**Theorem 1.** \( \forall C \in \{\text{RR}, \text{OR}, \text{PR}, \text{CR}, \text{CFR}\}. \text{MRS} \subseteq C \)

**Proof.** To show that \( \text{MRS} \subseteq C \), we show that there exists a neutral regulation \( \zeta_0 \) for any class \( C \) such that \( \forall M \in \text{MRS}, \mathcal{L}(M) \subseteq C \) with \( \zeta = \zeta_0 \) such that \( \mathcal{L}(M) = \mathcal{L}(M) \). In other words, the neutral regulation places no restrictions on rules application. The neutral regulations for individual classes are defined as follows:

\[
\begin{align*}
\text{RR} : \zeta_0 & = \mathcal{X}^\omega \\
\text{OR} : \zeta_0 & = \emptyset \\
\text{PR} : \zeta_0 & = \{ \mu \rightarrow \emptyset \mid \mu \in \mathcal{X} \} \\
\text{CR} : \zeta_0 & = \emptyset \\
\text{CFR} : \zeta_0 & = \emptyset
\end{align*}
\]

To show the strictness of the relations, we find an \( M \in C \) such that \( \forall M \in \text{MRS} \) holds \( \mathcal{L}(M) \neq \mathcal{L}(M) \). We only show that \( \text{MRS} \subseteq \text{OR} \), and we skip other proofs as they are based on the same idea of showing that their respective example (used above in the class definition) does not have a representative in \( \text{MRS} \) with an equal set of runs.

We use \( \mathcal{M} \in \text{OR} \) from Figure 5 and the example of valid run \( \pi_1 \) from Figure 6. The rule \( \mu_1 \) applied in step one and \( \mu_2 \) applied in step four enforce that \( \mu_1, \mu_2 \in \mathcal{X} \) for the \( M \in \text{MRS} \) (because there is no other way these transitions could happen). However, then a run \( \pi' \in \mathcal{L}(M) \):

\[
\begin{array}{cccc}
\circ & \xrightarrow{\mu_1} & \bullet & \xrightarrow{\mu_2} & \circ & \xrightarrow{\mu_1} & \bullet & \xrightarrow{\mu_2} & \mathcal{L}(M) & \ldots
\end{array}
\]

is possible, but clearly run \( \pi' \not\in \mathcal{L}(M) \). From this follows that \( \mathcal{L}(M) \neq \mathcal{L}(M) \) and thus the strictness of inclusion. \( \square \)

From Figure 1 also follows that all regulated classes have a common intersection in \( \text{MRS} \). Now we show some relations among regulated classes. These are mostly incomparable, as we will show below. However, there are also two pairs of classes, namely \( \text{OR} \supset \text{PR} \) and \( \text{CFR} \supset \text{CR} \), which have a subclass relation.

**Theorem 2.** \( \text{OR} \supset \text{PR} \)

**Proof.** To show that \( \text{OR} \supset \text{PR} \), we describe how for any \( M \in \text{OR} \) an equivalent system \( M' \in \text{PR} \) can be constructed. The partial order \( \zeta \) defined on the rules does not allow a rule lower in the order to be used immediately after a rule higher in the order. In other words, rules higher in the order or those incomparable can be applied. Using this fact we construct successor function \( \zeta'(\mu) = \{ \bar{\mu} \in \mathcal{X} \mid (\mu, \mu) \not\in \zeta \} \) for all \( \mu \in \mathcal{X} \) and \( \mathcal{X}' = \mathcal{X}, \mathcal{M}_0' = \mathcal{M}_0 \).

To prove that \( \mathcal{L}(M) = \mathcal{L}(M') \), we must show that for any state \( \mathcal{M} \) and rule \( \mu \in \mathcal{X} \) it holds that \( M \) can apply \( \mu \) to \( \mathcal{M} \Leftrightarrow M' \) can apply \( \mu \) to \( \mathcal{M} \). This is trivial for the initial state \( \mathcal{M}_0 \) since neither of the systems set any restrictions on rule application in the first step (from the definition). For any other general case

\[
\begin{array}{cccc}
\ldots & \xrightarrow{\mu_{\text{res}}} & \mathcal{M} & \xrightarrow{\mu} & \ldots
\end{array}
\]

with an enabled rule \( \mu \):

\Rightarrow: If \( M \) can apply \( \mu \) to \( \mathcal{M} \), that means that \( (\mu, \mu_{\text{res}}) \not\in \zeta \). From that and definition of \( \zeta' \) follows that \( \mu \in \zeta'(\mu_{\text{res}}) \). Therefore, \( M' \) also can apply \( \mu \) to \( \mathcal{M} \).

If \( M \) can not apply \( \mu \) to \( \mathcal{M} \), that means that \( (\mu, \mu_{\text{res}}) \in \zeta \). From that and definition of \( \zeta' \) follows that \( \mu \not\in \zeta'(\mu_{\text{res}}) \). Therefore, \( M' \) also can not apply \( \mu \) to \( \mathcal{M} \).
⇐: If \( \mathcal{M}' \) can apply \( \mu \) to \( \mathcal{M} \), that means that \( \mu \in \zeta'(\mu_{\text{pre}}) \). From the definition of \( \zeta' \) follows that \( (\mu, \mu_{\text{pre}}) \notin \zeta \). Therefore, \( \mathcal{M} \) also can apply \( \mu \) to \( \mathcal{M} \).

If \( \mathcal{M}' \) can not apply \( \mu \) to \( \mathcal{M} \), that means that \( \mu \notin \zeta'(\mu_{\text{pre}}) \). From the definition of \( \zeta' \) follows that \( (\mu, \mu_{\text{pre}}) \notin \zeta \). Therefore, \( \mathcal{M} \) also can not apply \( \mu \) to \( \mathcal{M} \).

To show the strictness of the inclusion, we find an \( \mathcal{M} \in \mathbb{PR} \) such that \( \forall \mathcal{M}' \in \mathbb{OR} \) holds \( \mathcal{L}(\mathcal{M}) \neq \mathcal{L}(\mathcal{M}') \).

We use \( \mathcal{M} \in \mathbb{PR} \) from Figure 7. The example of a valid run \( \pi_1 \) from Figure 8 enforces that \( \mu_1 \in \mathcal{X}' \) due to its first step. The symmetric run \( \pi_1' \):

\[
\begin{array}{ccccccc}
\mu_2 & \rightarrow & \mu_1 & \rightarrow & \mu_2 & \rightarrow & \mu_1 & \rightarrow & \mu_2 \\
\Rightarrow & \Rightarrow & \Rightarrow & \Rightarrow & \Rightarrow & \Rightarrow & \Rightarrow & \Rightarrow & \Rightarrow \\
\mu_2 & \rightarrow & \mu_1 & \rightarrow & \mu_2 & \rightarrow & \mu_1 & \rightarrow & \mu_2
\end{array}
\]

ensures that \( \mu_2 \in \mathcal{X}' \) also due to its first step. Then, for the ordered system \( \mathcal{M}' \), there are three options w.r.t. \( \zeta' \) how these two rules can be related (regardless of other rules in \( \mathcal{X}' \)):

\[
\begin{array}{ccccccc}
\mu_1, \mu_2 \text{ are} & \text{incomparable} & \Rightarrow & \Rightarrow & \Rightarrow & \Rightarrow & \Rightarrow \\
\mu_1 < \mu_2 & \Rightarrow & \Rightarrow & \Rightarrow & \Rightarrow & \Rightarrow & \Rightarrow \\
\mu_2 < \mu_1 & \Rightarrow & \Rightarrow & \Rightarrow & \Rightarrow & \Rightarrow & \Rightarrow
\end{array}
\]

for each case, there is a run valid in \( \mathcal{L}(\mathcal{M}') \) which is invalid in \( \mathcal{L}(\mathcal{M}) \), i.e. \( \mathcal{L}(\mathcal{M}) \neq \mathcal{L}(\mathcal{M}') \). □

**Theorem 3.** \( \text{CFR} \sqsubseteq \text{CR} \)

**Proof.** To show that \( \text{CFR} \subseteq \text{CR} \), we describe how for any \( \mathcal{M} \in \text{CFR} \) an equivalent system \( \mathcal{M}' \in \text{CR} \) can be constructed. First, we set \( \mathcal{M}_0 = \mathcal{M}_1, \mathcal{X}' = \mathcal{X}, \) and \( \zeta'(\mu) = \emptyset \) for all \( \mu \in \mathcal{X}' \). Then, we investigate each pair \( (\mu, \mu) \) in the relation \( \zeta \) (i.e. \( \mu, \mu \) are concurrent) individually:

1. if \( ^*\mu \supseteq ^*\mu, \)
   - rule \( \mu \) can never be used ⇒ remove it from \( \mathcal{X}' \)
2. if \( ^*\mu \subset^* \mu \) or \( ^*\mu \) are incomparable
   - rule \( \mu \) can never be used when \( \mu \) is enabled ⇒ \( \zeta'(\mu) = \{^*\mu\} \cup \zeta'(\mu) \)

To prove that \( \mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{M}') \), we must show that for any state \( \mathcal{M} \) and enabled rule \( \mu \in \mathcal{X} \) it holds that \( \mathcal{M} \) can apply \( \mu \) to \( \mathcal{M} \) ⇔ \( \mathcal{M}' \) can apply \( \mu \) to \( \mathcal{M} \):

\( \Rightarrow \): If \( \mathcal{M} \) can apply \( \mu \) to \( \mathcal{M} \), that means that \( \forall \mu' \in \mathcal{X} \) enabled at \( \mathcal{M} \) \( (\mu, \mu') \notin \zeta \). No forbidden context was introduced for rule \( \mu \), therefore also \( \mathcal{M}' \) can apply \( \mu \) to \( \mathcal{M} \).

If \( \mathcal{M} \) can not apply \( \mu \) to \( \mathcal{M} \), that means that \( \exists \mu' \in \mathcal{X} \) enabled at \( \mathcal{M} \) \( (\mu, \mu') \notin \zeta \). In case (1), rule \( \mu \) is removed from \( \mathcal{X}' \) and therefore also \( \mathcal{M}' \) can not apply \( \mu \) to \( \mathcal{M} \). In case (2), the new forbidden context \( \mu' \) is introduced for rule \( \mu' \); since \( \mu' \) is obviously enabled, it means that also \( ^*\mu' \subset \mathcal{M} \) and therefore \( \mathcal{M}' \) also can not apply \( \mu \) to \( \mathcal{M} \).

\( \Leftarrow \): If \( \mathcal{M}' \) can apply \( \mu \) to \( \mathcal{M} \), that means that \( \forall \mu' \in \mathcal{X}' \) holds that \( \mathcal{A} \subseteq \mathcal{M} \). Since for case (1) obviously \( \mu \in \mathcal{X}' \) and there was no forbidden context introduced in case (2), there is no concurrent rule \( \mu' \in \mathcal{X}' \) with \( (\mu, \mu') \in \zeta \). Therefore also \( \mathcal{M} \) can apply \( \mu \) to \( \mathcal{M} \).

If \( \mathcal{M}' \) can not apply \( \mu \) to \( \mathcal{M} \), that means that \( \exists \mu' \in \mathcal{X}' \) such that \( \mathcal{A} \subseteq \mathcal{M} \). Since for case (1) obviously \( \mu \in \mathcal{X}' \), there was some forbidden context introduced in case (2). From that follows there exists a concurrent rule \( \mu' \in \mathcal{X}' \) enabled at \( \mathcal{M} \) with \( (\mu, \mu') \in \zeta \). Therefore \( \mathcal{M} \) also can not apply \( \mu \) to \( \mathcal{M} \).

To show the strictness of the inclusion, we find an \( \mathcal{M} \in \mathbb{CR} \) such that \( \forall \mathcal{M}' \in \mathbb{CFR} \) holds \( \mathcal{L}(\mathcal{M}) \neq \mathcal{L}(\mathcal{M}') \). We use the following \( \mathcal{M} \in \mathbb{CR} \):

\[
\mathcal{M} = \left\{ \begin{array}{l}
\mathcal{M}_0 = \emptyset, \ \zeta = \{ \mu_1 \rightarrow \{C\}, \mu_2 \rightarrow \{B\} \},\\
\mathcal{X} = \{ \mu_1 : \emptyset \rightarrow \{B\}, \mu_2 : \emptyset \rightarrow \{C\} \}
\end{array} \right\}
\]

with \( \mathcal{S} = \{B,C\} \) where using one of the rules disables the other one. Therefore, the set of runs \( \mathcal{L}(\mathcal{M}) \) contains only two possible runs \( \pi_1 \) and \( \pi_2 \):

\[
\begin{array}{ccccccc}
\mu_1 & \rightarrow & \emptyset & \rightarrow & \mu_1 & \rightarrow & \mu_1 & \rightarrow & \mu_1 & \rightarrow \ldots & \pi_1 \in \mathcal{L}(\mathcal{M}) \\
\mu_2 & \rightarrow & \emptyset & \rightarrow & \mu_2 & \rightarrow & \mu_2 & \rightarrow & \mu_2 & \rightarrow \ldots & \pi_2 \in \mathcal{L}(\mathcal{M})
\end{array}
\]
To ensure at least the first step in both runs, it has to hold that both rules $\mu_1, \mu_2 \in X'$ for $\mathcal{M}' \in \mathcal{CFR}$. Since these two rules are not concurrent, we cannot restrict their application at all using $\zeta'$. From that follows there will be some additional runs present in $\mathcal{L}(\mathcal{M}')$ and therefore $\mathcal{L}(\mathcal{M}) \neq \mathcal{L}(\mathcal{M}')$. \hfill \Box

All the other classes are incomparable. We now investigate individual relationships. The proofs are given by showing counterexamples and using already proven relations.

**Theorem 4. $\mathcal{OR} \nsubseteq \mathcal{CFR}$**

**Proof.** To show that $\mathcal{OR} \nsubseteq \mathcal{CFR}$, we find an $\mathcal{M} \in \mathcal{OR}$ such that $\forall \mathcal{M'} \in \mathcal{CFR}$ holds that $\mathcal{L}(\mathcal{M}) \neq \mathcal{L}(\mathcal{M}')$. We use $\mathcal{M} \in \mathcal{OR}$ from Figure 5. Based on proof of Figure 1, both rules $\mu_1, \mu_2$ belong to $X'$. However, these rules are not concurrent, so they can be used in an arbitrary fashion in any $\mathcal{M}' \in \mathcal{CFR}$. For example, a run:

$$\begin{array}{cccccccc}
\bigcirc & \xrightarrow{\mu_1} & A & \quad & \xrightarrow{\mu_2} & A & \xrightarrow{\mu_1} & \quad & \mathbb{R}
\end{array}$$

is possible in $\mathcal{L}(\mathcal{M}')$ but not in $\mathcal{L}(\mathcal{M})$, from that $\mathcal{L}(\mathcal{M}) \neq \mathcal{L}(\mathcal{M}')$.

To show that $\mathcal{CFR} \nsubseteq \mathcal{OR}$, we find an $\mathcal{M} \in \mathcal{CFR}$ such that $\forall \mathcal{M'} \in \mathcal{OR}$ holds that $\mathcal{L}(\mathcal{M}) \neq \mathcal{L}(\mathcal{M}')$. We use the following $\mathcal{M} \in \mathcal{CFR}$:

$$\mathcal{M} = \left\{ I_0 = \{A, B\}, \quad \zeta = \{ (\mu_2, \mu_1) \}, \quad \mathcal{X} = \{ \mu_1 : \{A, B\} \rightarrow \{A, C\}, \mu_2 : \{A\} \rightarrow \{A, B\} \} \right\}$$

with $S = \{A, B, C\}$. In this system, rules are alternating due to the defined priority on rule $\mu_1$. The set of runs $\mathcal{L}(\mathcal{M})$ contains only one possible run $\pi$:

$$\begin{array}{cccccccc}
\bigcirc & \xrightarrow{\mu_1} & A & \quad & \xrightarrow{\mu_2} & A & \quad & \xrightarrow{\mu_1} & \quad & \mathbb{R}
\end{array}$$

which implies that at least two rules $\mu_1, \mu_2 \in X'$ such that $B \in \mu_1 \land \mathcal{C} \in \mu_2^*$ (from the first step) and $B \in \mu_2^*$ (from the second step). Then, similarly to proof of Figure 2, there are three options (incomparable and ordered in either direction) and for each option, there exists a run possible in $\mathcal{L}(\mathcal{M}')$ but not possible in $\mathcal{L}(\mathcal{M})$, i.e. $\mathcal{L}(\mathcal{M}) \neq \mathcal{L}(\mathcal{M}')$.

Another option would be to create a unique rule $\mu_1 : \pi[i - 1] \rightarrow \pi[i]$ for each step $i$ and placing this rule above each rule $\mu_2$ with $j < i$ in the order $\zeta'$. However, this would require that $X'$ is infinite which the definition does not allow. \hfill \Box

**Theorem 5. $\mathcal{CFR} \nsubseteq \mathcal{PR}$**

**Proof.** The property $\mathcal{PR} \nsubseteq \mathcal{CFR}$ follows directly from $\mathcal{OR} \nsubseteq \mathcal{CFR}$ (Figure 4) and $\mathcal{OR} \subseteq \mathcal{PR}$ (Figure 2).

To show that $\mathcal{CFR} \nsubseteq \mathcal{PR}$, we find an $\mathcal{M} \in \mathcal{CFR}$ such that $\forall \mathcal{M'} \in \mathcal{PR}$ holds that $\mathcal{L}(\mathcal{M}) \neq \mathcal{L}(\mathcal{M}')$. We use $\mathcal{M} \in \mathcal{CFR}$ from Figure 11.

First, we make a general observation about any $\mathcal{M}' \in \mathcal{PR}$: due to the presence of successor function, the rules of a PR system can be seen as a directed successor graph, where vertices are the rules and edges are the possible successors. Such a graph of the $\mathcal{M}'$ system must contain at least one cycle, which (in summation) increases the number of elements $B$ in the systems state. Let $c$ be the maximal number of $B$s produced by any such cycle in this graph.

Next, let $\pi_1 \in \mathcal{L}(\mathcal{M})$ be a run which produces $2c$ of $B$s, then immediately discards them, and removes the element $A$ with rule $\mu_3$. Clearly, $\mathcal{M}'$ must contain rule $\mu_3$ (e.g. fifth step of run $\pi_1$ in Figure 12) and this rule is applied at the end of this run (neglecting the empty rule $\varepsilon$). Additionally, since the number of $B$s produced is $2c$, the run must contain at least one aforementioned cycle from the successor graph.

Finally, let us consider another run $\pi_{11}$ which extends the run $\pi_1$ by adding one repetition of this cycle. The run $\pi_{11}$ is still valid in $\mathcal{M}'$, by definition of the successor graph, and the last transition of this run is still $\mu_3$. However, the number of produced $B$s is higher than the number of $B$s consumed. Such a run is not possible in $\mathcal{M}$, because $\mu_3$ can be fired only when all $B$s have been consumed. Therefore $\pi_{11} \notin \mathcal{L}(\mathcal{M})$ and from that $\mathcal{L}(\mathcal{M}) \neq \mathcal{L}(\mathcal{M}')$. \hfill \Box

**Theorem 6. $\mathcal{OR} \nsubseteq \mathcal{CR}$**

**Proof.** The property $\mathcal{CR} \nsubseteq \mathcal{OR}$ follows directly from $\mathcal{OR} \nsubseteq \mathcal{CFR}$ (Figure 4) and $\mathcal{CFR} \subseteq \mathcal{CR}$ (Figure 3).

To show that $\mathcal{OR} \nsubseteq \mathcal{CR}$, we find an $\mathcal{M} \in \mathcal{OR}$ such that $\forall \mathcal{M'} \in \mathcal{CR}$ holds that $\mathcal{L}(\mathcal{M}) \neq \mathcal{L}(\mathcal{M}')$. We use $\mathcal{M} \in \mathcal{OR}$ from Figure 5. As it was stated in the proof of Figure 1, both rules $\mu_1, \mu_2 \in X'$ of the $\mathcal{M}' \in \mathcal{CR}$. Due to the minimalistic nature of this system, it can be shown for any case of $\zeta'$ definition
(assuming adding prohibited context with increased number of $A$'s has no effect in this case), that $\Sigma(\mathcal{M}')$ contains more possible runs:

\[ \zeta'(\mu_1) = \emptyset \land \zeta'(\mu_2) = \emptyset \]
\[ \emptyset \xrightarrow{\mu_1} A \xrightarrow{\mu_2} \emptyset \xrightarrow{\mu_1} A \xrightarrow{\mu_2} \ldots \notin \mathcal{L}(\mathcal{M}) \]
\[ \zeta'(\mu_1) = \{A\} \land \zeta'(\mu_2) = \emptyset \]
\[ \emptyset \xrightarrow{\mu_1} A \xrightarrow{\mu_1} \emptyset \xrightarrow{\mu_2} \emptyset \xrightarrow{\mu_1} A \xrightarrow{\mu_2} \ldots \notin \mathcal{L}(\mathcal{M}) \]
\[ \zeta'(\mu_1) = \emptyset \land \zeta'(\mu_2) = \{A\} \]
\[ \emptyset \xrightarrow{\mu_1} A \xrightarrow{\epsilon} A \xrightarrow{\epsilon} A \xrightarrow{\epsilon} \ldots \notin \mathcal{L}(\mathcal{M}) \]

from that follows $\mathcal{L}(\mathcal{M}) \neq \mathcal{L}(\mathcal{M}')$.

\[ \square \]

**Theorem 7.** \( \text{PR} \not\subseteq \text{CR} \)

**Proof.** The property \( \text{PR} \not\subseteq \text{CR} \) follows directly from \( \text{OR} \not\subseteq \text{CR} \) (Figure 6) and \( \text{OR} \subseteq \text{PR} \) (Figure 2). The property \( \text{CR} \not\subseteq \text{PR} \) follows directly from \( \text{CFR} \not\subseteq \text{PR} \) (Figure 5) and \( \text{CFR} \subseteq \text{CR} \) (Figure 3).

\[ \square \]

**Theorem 8.** \( \text{RR} \not\subseteq \text{OR} \)

**Proof.** To show that \( \text{RR} \not\subseteq \text{OR} \), we find an \( \mathcal{M} \in \text{RR} \) such that \( \forall \mathcal{M}' \in \text{OR} \) holds that \( \mathcal{L}(\mathcal{M}) \neq \mathcal{L}(\mathcal{M}') \). We use the following \( \mathcal{M} \in \text{RR} \):

\[ \mathcal{M} = \{ \mathcal{M}_0 = \emptyset, \, \zeta = \{ \mu_1, \mu_2, \mu_3, \varepsilon^\omega \}, \, \lambda = \{ \mu_1 : \emptyset \rightarrow \{A\}, \mu_2 : \{A\} \rightarrow \emptyset, \mu_3 : \{A\} \rightarrow \{B\} \} \} \]

with \( \mathcal{S} = \{A, B\} \). In this system, rules can be used only in a particular finite sequence, followed by infinite application of empty rule \( \varepsilon \). Therefore, the set of runs \( \mathcal{L}(\mathcal{M}) \) contains only one possible run \( \pi \):

\[ \emptyset \xrightarrow{\mu_1} A \xrightarrow{\mu_2} \emptyset \xrightarrow{\mu_1} A \xrightarrow{\mu_3} \emptyset \xrightarrow{\epsilon} \ldots \]

Since the run \( \pi \) enforces that rule \( \mu_1 \in \lambda' \), it is inevitable that, for example, run:

\[ \emptyset \xrightarrow{\mu_1} A \xrightarrow{\mu_1} \emptyset \xrightarrow{\mu_1} A \xrightarrow{\mu_3} \emptyset \xrightarrow{\epsilon} \ldots \]

belongs to \( \mathcal{L}(\mathcal{M}') \) because it can be applied in the first step and its subsequent applications cannot be limited in the ordered system (follows from irreflexivity of the order \( \zeta' \)).

To show that \( \text{OR} \not\subseteq \text{RR} \), we find an \( \mathcal{M} \in \text{OR} \) such that \( \forall \mathcal{M}' \in \text{RR} \) holds that \( \mathcal{L}(\mathcal{M}) \neq \mathcal{L}(\mathcal{M}') \). We use \( \mathcal{M} \in \text{OR} \) from Figure 5. Let \( \mathcal{A} = \{ \mu_1^2, \mu_2^2, \varepsilon^\omega \} \) be an \( \omega \)-language over rules \( \mu_1, \mu_2, \varepsilon \). For the system \( \mathcal{M} \), it holds that \( \forall \pi \in \mathcal{L}(\mathcal{M}) \). However, language \( \mathcal{A} \) is obviously not \( \omega \)-regular, therefore such an \( \mathcal{M}' \in \text{RR} \) does not exist.

\[ \square \]

**Theorem 9.** \( \text{RR} \not\subseteq \text{PR} \)

**Proof.** To show that \( \text{RR} \not\subseteq \text{PR} \), we find an \( \mathcal{M} \in \text{RR} \) such that \( \forall \mathcal{M}' \in \text{PR} \) holds that \( \mathcal{L}(\mathcal{M}) \neq \mathcal{L}(\mathcal{M}') \). We use the \( \mathcal{M} \in \text{RR} \) from the proof of Figure 8. The run \( \pi \) actually enforces all the rules \( \mu_1, \mu_2, \mu_3 \) to be present in \( \lambda' \) due to steps one, two, and four, respectively. Let us focus on the successor function \( \zeta' \) for rule \( \mu_1 \) w.r.t. rules \( \mu_2, \mu_3 \). There are four options:

\[ \mu_2 \notin \zeta'(\mu_1) \land \mu_3 \notin \zeta'(\mu_1) \ldots \text{run } \pi \notin \mathcal{L}(\mathcal{M}) \text{ because } \mu_2 \text{ cannot be used after } \mu_1 \]

\[ \mu_2 \notin \zeta'(\mu_1) \land \mu_3 \notin \zeta'(\mu_1) \ldots \text{run } \pi \notin \mathcal{L}(\mathcal{M}) \text{ because } \mu_2 \text{ cannot be used after } \mu_1 \]

\[ \mu_2 \notin \zeta'(\mu_1) \land \mu_3 \notin \zeta'(\mu_1) \ldots \text{run } \pi \notin \mathcal{L}(\mathcal{M}) \text{ because } \mu_3 \text{ cannot be used after } \mu_1 \]

\[ \mu_2 \notin \zeta'(\mu_1) \land \mu_3 \notin \zeta'(\mu_1) \ldots \text{run } \pi \notin \mathcal{L}(\mathcal{M}) \text{ because } \mu_3 \text{ cannot be used after } \mu_1 \]

with \( \pi' \in \mathcal{L}(\mathcal{M}') \) but \( \pi' \notin \mathcal{L}(\mathcal{M}) \). In each case holds \( \mathcal{L}(\mathcal{M}) \neq \mathcal{L}(\mathcal{M}') \).

The property \( \text{PR} \not\subseteq \text{RR} \) follows directly from \( \text{RR} \not\subseteq \text{OR} \) (Figure 8) and \( \text{OR} \subseteq \text{PR} \) (Figure 2).

\[ \square \]
Theorem 10. \( \text{RR} \not\subseteq \text{CFR} \)

Proof. To show that \( \text{RR} \not\subseteq \text{CFR} \), we find an \( M \in \text{RR} \) such that \( \forall M' \in \text{CFR} \) holds that \( \Sigma(M) \neq \Sigma(M') \).

We use the \( M \in \text{RR} \) from the proof of Figure 8. As stated in the proof of Figure 9, all rules \( \mu_1, \mu_2, \mu_3 \) belong to \( \mathcal{X}' \). Rules \( \mu_2, \mu_3 \) are concurrent and there are three options how to resolve the concurrency:

- \( (\mu_2, \mu_3) \in \zeta' \)
- \( (\mu_3, \mu_2) \in \zeta' \)

In each case holds \( \Sigma(M) \neq \Sigma(M') \).

To show that \( \text{CFR} \not\subseteq \text{RR} \), we find an \( M \in \text{CFR} \) such that \( \forall M' \in \text{RR} \) holds that \( \Sigma(M) \neq \Sigma(M') \). We use the following \( M \in \text{CFR} \):

\[
M = \left\{ M_0 = \{C\}, \quad \zeta = \{ (\mu_4, \mu_3) \}, \quad \right.
\]
\[
\left. \mathcal{X} = \left\{ \mu_1 : \{C\} \rightarrow \{A, C\}, \mu_2 : \{C\} \rightarrow \{B\} \right\}, \quad \mu_3 : \{A, B\} \rightarrow \{B\}, \mu_4 : \{B\} \rightarrow \emptyset \right\}
\]

with \( \mathcal{S} = \{ \text{A, B, C} \} \). In this system, a number \( n \in \mathbb{N} \) of elements \( \text{A} \) is generated, followed by the change of \( \text{C} \) to \( \text{A} \) and finally, all elements \( \text{A} \) are discarded, including symbol \( \text{B} \). This is ensured by giving priority to rule \( \mu_3 \) over rule \( \mu_4 \). Please note the concurrency between rules \( \mu_1 \) and \( \mu_2 \) is not resolved (which is allowed by definition).

Since runs:

\[
\left\{ \begin{array}{c}
\text{C} \\
\text{C}
\end{array} \right\} \quad \mu_1 \rightarrow \left\{ \begin{array}{c}
\text{A}, \text{C} \\
\text{C}
\end{array} \right\} \quad \mu_2 \rightarrow \left\{ \begin{array}{c}
\text{A} \\
\emptyset
\end{array} \right\} \quad \mu_4 \rightarrow \left\{ \begin{array}{c}
\emptyset \quad \emptyset \quad \emptyset \quad \emptyset
\end{array} \right\}
\]

are both allowed, both \( \mu_2, \mu_4 \in \mathcal{X}' \) and also \( \mu_1, \mu_{111} \in \mathcal{X}' \) such that \( \text{A} \in \mu_1^* \) (from the first step) and \( \text{A} \in \mu_{111}^* \) (from the third step).

Let \( \mathcal{A} = \{ \mu_1^* \mu_2^* \mu_{111}^* \mu_4 \omega \mid n \in \mathbb{N} \} \) be an \( \omega \)-language over rules \( \mu_1, \mu_2, \mu_{111}, \mu_4, \omega \). For the system \( M \), it holds that \( \forall \pi \in \Sigma(M), \mathcal{A} \in \mathcal{A} \) (such that instead of rules \( \mu_1, \mu_{111} \) their possible instances \( \mu_1, \mu_3 \) are used, respectively). However, language \( \mathcal{A} \) is obviously not \( \omega \)-regular, therefore such an \( M' \in \text{RR} \) does not exist. \( \square \)

Theorem 11. \( \text{RR} \not\subseteq \text{CR} \)

Proof. To show that \( \text{RR} \not\subseteq \text{CR} \), we find an \( M \in \text{RR} \) such that \( \forall M' \in \text{CR} \) holds that \( \Sigma(M) \neq \Sigma(M') \).

We use the \( M \in \text{RR} \) from the proof of Figure 8. From the only possible run \( \pi \) it follows that rules \( \mu_2, \mu_3 \) need to be applicable in the context of element \( \text{A} \) while it is only possible candidate to be in their prohibited context. Since that cannot be done, there will be some additional runs in \( \Sigma(M') \). From that \( \Sigma(M) \neq \Sigma(M') \).

The property \( \text{CR} \not\subseteq \text{RR} \) follows directly from \( \text{RR} \not\subseteq \text{CFR} \) (Figure 10) and \( \text{CFR} \supsetneq \text{CR} \) (Figure 3). \( \square \)

4.2 Expressive power

In this section, we discuss expressive power for some of the defined classes. While the generative power discussed above states about the particular sets of runs that can be generated, the expressive power studies functions which can be computed by the formalism.

A multiset rewriting system (with or without regulation) strongly computes a numerical function \( f : \mathbb{N} \rightarrow \mathbb{N} \) by starting in an initial multiset \( M_0 \) containing an input element \( I \) with \( M_0(I) = n \) and always reaches a final multiset \( M_f \) (all subsequent multisets are equal) containing an output element \( O \) with \( M_f(O) = f(n) \). In order for the system to be correct, there cannot exist a run where for the final multiset \( M_f \) holds that \( M_f(O) \neq f(n) \).

Since this definition does not accommodate nondeterminism nicely (which is natural for multiset rewriting), we say a multiset rewriting system weakly computes a numerical function \( f : \mathbb{N} \rightarrow \mathbb{N} \) if there exists a run with \( M_f(O) = f(n) \) and for any other run holds that \( M_f(O) \leq f(n) \). This definition is an adoption of weak computability for Petri nets [14].
Theorem 12. \( \text{MRS} \) weakly compute all computable numerical functions.

For the proof, we refer to [4,5] where it is shown that expressive power of \( \text{MRS} \) is equivalent to expressive power of Petri Nets, because they are both based on simple operations on positive integer counters – decrements and increments. More precisely, they cannot compute all recursive functions because they lack zero-tests which makes them less expressive than Turing machine – in particular, they weakly compute numerical functions [14].

Theorem 13. \( \text{CR} \) and \( \text{CFR} \) compute all computable numerical functions.

Proof. Assuming the Church-Turing thesis is correct, the simplest way how to prove this theorem is in terms of Minsky register machine [16]. Minsky machine has a number of registers storing arbitrary large numbers. A program is a sequence of instructions manipulating the registers. It was shown that two register machines with the following instruction set can compute all computable numerical functions (shown by equivalence to Turing machine [19]):

\[
\begin{align*}
l_1 & : \text{Com}_1; \\
l_2 & : \text{Com}_2; \\
\vdots \\
l_m & : \text{Com}_m; \\
l_{m+1} & : \text{halt};
\end{align*}
\]

where \( m \leq 0 \) and every \( \text{Com}_i \) is one of two types:

- Type I: \( c_j := c_j + 1; \text{goto } l_p \)
- Type II: if \( c_j = 0 \) then \( \text{goto } l_p \) else \( c_j := c_j - 1; \text{goto } l_q \)

with counter index \( j \in \{1, 2\} \) and \( p, q \in \{1, \ldots, m + 1\} \).

Computation of the machine with the counters set to initial values (with \( c_1 = n \)) starts at \( l_1 \) and proceeds consistently with the intuitive semantics of the instructions. We say that the machine halts if the computation eventually reaches the halt instruction. Finally, the result of computation is the value of counter \( c_2 \).

Thus, if a register machine can be converted into an equivalent conditional (resp. concurrent-free) regulated MRS, we see that \( \text{CR} \) (resp. \( \text{CFR} \)) can compute all computable numerical functions. First, we show this for \( \text{CR} \). The proof for \( \text{CFR} \) is identical except for a minor modification specific to its regulation approach.

To represent a register machine as an \( \mathcal{M} \in \text{CR} \), we represent the two registers by two elements \( c_1, c_2 \) (such that initially \( c_1 \) has \( n \) repetitions and \( c_2 \) represents the output counter). We use elements \( l_1, \ldots, l_m, l_{m+1} \) to represent the position of the program. The instructions are represented as follows:

- Type I: introduce rule \( \mu_1 : \{l_i\} \rightarrow \{l_p, c_j\} \)
- Type II: introduce rules \( \mu_1 : \{l_i\} \rightarrow \{l_p\} \) and \( \overline{\mu}_1 : \{l_i, c_j\} \rightarrow \{l_q\} \) and set \( \zeta(\mu_1) = (c_j) \)

Defining prohibited context to the rule \( \mu_1 \) in Type II instruction effectively prioritises rule \( \overline{\mu}_1 \) until the counter \( c_j \) is not zero. Finally, the instruction halt is represented by the application of rule \( \varepsilon \).

This shows that a register machine can be converted into \( \text{CR} \). In the case of \( \text{CFR} \), the only modification is in the definition of regulation \( \zeta \) in the Type II instruction – we require that \( (\mu_1, \overline{\mu}_1) \in \zeta \), which ensures that \( \overline{\mu}_1 \) has priority until the counter \( c_j \) is not zero. \( \square \)

5 Conclusions

In this paper, we introduced regulated multiset rewriting systems and discussed relationships among them, showing the usefulness of individual classes as most of them are incomparable. The considered regulation mechanisms were mostly inspired by the needs encountered in modelling of biological phenomenons. As our future work, we would like to explore other mechanisms, for example those based on other subclasses of \( \omega \) languages. We also want to investigate the expressive power of regulated MRS in more detail and focus on the complexity of common problems in system biology, such as reachability. Finally, we will apply multiset rewriting regulations to a rule-based Biochemical Space language [22], which can be seen as an instantiation of MRS, and present biological case studies of the regulations using this language.
References

1. Roberto Barbuti, Giulio Caravagna, Andrea Maggiolo-Schettini, and Paolo Milazzo. An Intermediate Language for the Simulation of Biological Systems. *Electronic Notes in Theoretical Computer Science*, 194(3):19–34, 2008.

2. Stefano Bistarelli, Iliano Cervesato, Gabriele Lenzini, Roberto Marangoni, and Fabio Martinelli. On Representing Biological Systems Through Multiset Rewriting. In *International Conference on Computer Aided Systems Theory*, pages 415–426. Springer, 2003.

3. Pierre Boutilier, Mutamba Maasha, Xing Li, Héctor F Medina-Abarca, Jean Krivine, Jérôme Feret, Ioana Cristescu, Angus G Forbes, and Walter Fontana. The Kappa Platform for Rule-Based Modeling. *Bioinformatics*, 34(13):i583–i592, 2018.

4. Iliano Cervesato. Petri Nets as Multiset Rewriting Systems in a Linear Framework. *Unpublished manuscript*, 1994.

5. Iliano Cervesato. Petri Nets and Linear Logic: a Case Study for Logic Programming. In *Proceedings of the 1995 Joint Conference on Declarative Programming*, pages 313–318. Palladio Press, 1995.

6. Vincent Danos, Russ Harmer, and Glynn Winskel. Constraining Rule-Based Dynamics with Types. *MSCS*, 23(2):272–289, 2013.

7. Jürgen Dassow. Grammars with Regulated Rewriting. In *Formal Languages and Applications*, pages 249–273. Springer, 2004.

8. Tadeáš Děd, David Šafářek, Matej Troják, Matej Klement, Jakub Šalagovič, and Luboš Brim. Formal Biochemical Space with Semantics in Kappa and BNGL. *Electronic Notes in Theoretical Computer Science*, 326:27–49, 2016.

9. Giorgio Delzanno. An overview of MSR (C): A CLP-based framework for the symbolic verification of parameterized concurrent systems. *Electronic Notes in Theoretical Computer Science*, 76:65–82, 2002.

10. Rudolf Freund, Carlos Martín-Vide, and Gheorghe Păun. From Regulated Rewriting to Computing with Membranes: Collapsing Hierarchies. *Theoretical Computer Science*, 312(2-3):143–188, 2004.

11. David Harel, Yaki Setty, Sol Efroni, Naamah Swerdlin, and Irun R Cohen. Concurrency in Biological Modeling: Behavior, Execution and Visualization. *Electronic Notes in Theoretical Computer Science*, 194(3):119–131, 2008.

12. Leonard A Harris, Justin S Hogg, José-Juan Tapia, John AP Sekar, Sanjana Gupta, Ilya Korsunsky, Arshi Arora, Dipak Barua, Robert P Sheehan, and James R Faeder. Bio NetGen 2.2: Advances in Rule-Based Modeling. *Bioinformatics*, 32(21):3366–3368, 2016.

13. Atsunobu Ichikawa and Kunihiko Hiraishi. Analysis and Control of Discrete Event Systems Represented by Petri Nets. In *Discrete Event Systems: Models and Applications*, pages 115–134. Springer, 1988.

14. Jérôme Leroux and Ph Schnoebelen. On Functions Weakly Computable by Petri Nets and Vector Addition Systems. In *International Workshop on Reachability Problems*, pages 190–202. Springer, 2014.

15. José Meseguer. Conditional Rewriting Logic as a Unified Model of Concurrency. *Theoretical computer science*, 96(1):73–155, 1992.

16. Marvin Lee Minsky. *Computation*. Prentice-Hall Englewood Cliffs, 1967.

17. Gheorghe Păun. Membrane Computing and Brane Calculi (some personal notes). *Electronic Notes in Theoretical Computer Science*, 171(2):3–10, 2007.

18. David Šafářek, Jan Červený, Matej Klement, Jana Pospíšilová, Luboš Brim, Dušan Lazár, and Ladislav Nedbal. E-photosynthesis: Web-based platform for modeling of complex photosynthetic processes. *BioSystems*, 103(2):115–124, 2011.

19. John C. Shepherdson and Howard E. Sturgis. Computability of Recursive Functions. *Journal of the ACM (JACM)*, 10(2):217–255, 1963.

20. Maciej Swat, Alexander Kel, and Hanspeter Herzel. Bifurcation Analysis of the Regulatory Modules of the Mammalian G1/S Transition. *Bioinformatics*, 20(10):1506–1511, 2004.

21. Matej Troják, David Šafářek, Jakub Hrabec, Jakub Šalagovič, Františka Romanovská, and Jan Červený. E-cyanobacterium. org: A web-based platform for systems biology of cyanobacteria. In *International Conference on Computational Methods in Systems Biology*, pages 316–322. Springer, 2016.

22. Matej Troják, David Šafářek, Lukrécia Mertová, and Luboš Brim. Executable Biochemical Space for Specification and Analysis of Biochemical Systems. *PLOS ONE*, 15(9):e0238838, 2020.