Comparing the Expressive Power of Well-structured Transition Systems

Parosh Aziz Abdulla[1], Giorgio Delzanno[2], and Laurent Van Begin[3⋆]

[1] Uppsala University, Sweden, parosh@it.uu.se
[2] Università di Genova, Italy, giorgio@disi.unige.it
[3] Université Libre de Bruxelles, Belgium, lvbegin@ulb.ac.be

Abstract. We compare the expressive power of a class of well-structured transition systems that includes relational automata, Petri nets, lossy channel systems, and constrained multiset rewriting systems. For each one of these models we study the class of languages generated by labelled transition systems describing their semantics. We consider here two types of accepting conditions: coverability and reachability of a given configuration. In both cases we obtain a strict hierarchy in which constrained multiset rewriting systems is the the most expressive model.

Keywords. Expressiveness, well-structured systems, language theory.

1 Introduction

The theory of well-structured transition systems [1, 13] is a powerful tool for studying the decidability of verification problems of infinite-state systems. A system is well-structured when its transition relation is monotonic with respect to a well-quasi ordering defined over configurations. A well-known example of well-structured system is that of *Petri nets* [19] equipped with marking inclusion [1, 13]. For a well-structured transition system, the *coverability problem* can be decided by the symbolic backward reachability algorithm scheme proposed in [1]. Since checking safety properties can be translated into instances of the coverability problem, an algorithm for coverability like that proposed in [1] can be used for automatic verification of an infinite-state system. This connection has been exploited in order to develop automatic verification procedures for Petri nets and their extensions [10, 11], for abstract models of imperative programs called *relational automata* [9], for abstract models of unreliable communication systems called *lossy (FIFO) channel systems* [5, 8], and for *constrained multiset rewriting systems* [2]. The latter model is an extension of Petri nets in which tokens are colored with natural numbers and in which transitions have numerical conditions defined over variables representing colors. The resulting model can be applied to model parameterized systems in which individual processes have local data that range over an infinite domain.

⋆ Research fellow supported by the FNRS.

Although several efforts have been spent in studying the expressive power of variations of Petri nets (see e.g. [10, 12, 14]), a comparison of the relative expressiveness of the class of well-structured transition systems is still missing. Such a comparison is a challenging research problem with a possible practical impact. Indeed, it can be useful to extend the applicability of a verification method (e.g. a particular instance of the scheme of [1]) to an entire class of models.

In this paper we apply tools of language theory to formally compare the expressive power of a large class of well-structured infinite-state systems that includes constrained multiset rewriting systems, lossy channel systems, (extensions of) Petri nets, and relational automata. To achieve this goal, for each one of these models we study the class of languages generated by labeled transition systems describing their semantics. We consider here two types of accepting conditions: coverability (with respect to a fixed ordering) and reachability of a given configuration. Two models are considered to be equivalent if they generate the same class of languages.

For coverability accepting conditions, we obtain the following classification. We first prove that lossy channel systems are equivalent to a syntactic fragment of constrained multiset rewriting, we named Γ_0. The fragment Γ_0 is obtained by restricting conditions of a rule in such a way that equalities cannot be used as guards. Furthermore, we prove that lossy channel systems are strictly less expressive than the full model of constrained multiset rewriting systems. We then show that Petri nets are equivalent to a syntactic fragment of constrained multiset rewriting systems, we named Γ_1, obtained by considering nullary predicates only. We also prove that Petri nets are strictly less expressive than lossy channel systems. We then prove that relational automata are equivalent to a syntactic fragment of constrained multiset rewriting, we named Γ_2, obtained by imposing an upper bound on the size (number of predicates) of reachable configurations. Finally, we prove that Γ_2 generates the class of regular languages. This implies that relational automata are strictly less expressive than Petri nets. In the paper we also extend the comparison to extensions of Petri nets like *transfer/reset nets* and *broadcast protocols* [10, 11] and to *lossy vector addition systems* [18]. Specifically, we prove that all these models are strictly less expressive than constrained multiset rewriting systems.

For reachability accepting conditions, we obtain a slightly different classification. First, we prove that Γ_0 is equivalent to constrained multiset rewriting systems and *two counter machines*. Thus, with reachability acceptance, Γ_0 and constrained multiset rewriting systems turn out to be strictly more expressive than lossy channel systems. On the contrary, Γ_1 is still equivalent to Petri nets and strictly less expressive than Γ_0 and Γ_2 is still equivalent to relational automata and to finite automata. Finally, we show that lossy channel systems and Petri nets define incomparable classes of languages.

Concerning related work, the relative expressiveness of well-structured systems has been investigated for a limited number of extensions of Petri nets with reset, transfer, and non-blocking arcs in [12, 14]. Classical results on finite and infinite languages generated by Petri nets can be found, e.g., in [15]. A classification of infinite-state systems in terms of structural properties and decidable verification problems is presented in [16]. The classification is extended to well-structured systems in [7]. A classification of the complexity of the decision procedures for coverability is studied in [17]. In contrast with the aforementioned work, we provide here a strict classification of the expressive power of several well-structured transition systems built with the help of tools of language theory. An extended version of the present paper is available as technical report [3].

Outline In Section 2, we give some preliminary notions on well-structured transition systems. In Section 3, we give some first results on the class of languages accepted by constrained multiset rewriting systems. In Section 4, 5, and 6, we compare the class of languages recognized by constrained multiset rewriting systems and, respectively, lossy channel systems, Petri nets, and relational automata. Finally, in Section 7 we discuss some final remarks.

2 Preliminaries on Well-structured Transition Systems

In this section we recall some definitions taken from [1]. A *transition system* is a tuple $T = (S, R)$ where S is a (possibly infinite) set of configurations, R is a finite set of transitions where each $\xrightarrow{\sigma} \in R$ is a binary relation over S, i.e. $\xrightarrow{\sigma} \subseteq S \times S$. We use $\gamma \xrightarrow{\sigma} \gamma'$ to denote $(\gamma, \gamma') \in \xrightarrow{\sigma}$, and $\gamma \xrightarrow{\rho_1 \cdots \rho_k} \gamma'$ to denote that there exist $\gamma_1, \ldots, \gamma_{k-1}$ such that $\gamma \xrightarrow{\rho_1} \gamma_1 \ldots \xrightarrow{\rho_{k-1}} \gamma_{k-1} \xrightarrow{\rho_k} \gamma'$. A quasi-ordering $(S, \preceq)$ is a *well-quasi ordering* if for any infinite sequence $s_1 s_2 \ldots s_i \ldots$ there exist indexes $i < j$ such that $s_i \preceq s_j$. A transition system $T = (S, R)$ is *well-structured* with respect to a quasi-order $\preceq$ on S iff: (i) $\preceq$ is a well-quasi ordering; (ii) for any $\xrightarrow{\sigma} \in R$ and $\gamma_1, \gamma_1', \gamma_2$ s.t. $\gamma_1 \preceq \gamma_1'$ and $\gamma_1 \xrightarrow{\sigma} \gamma_2$, there exists γ_2' s.t. $\gamma_1' \xrightarrow{\sigma} \gamma_2'$ and $\gamma_2 \preceq \gamma_2'$, i.e., T is *monotonic*. We use $T = (S, R, \preceq)$ to indicate a *well-structured transition system* (wsts for short).

To formalize the comparison between models, a wsts $T = (S, R, \preceq)$ can be viewed as a language acceptor. For this purpose, we assume a finite alphabet Σ and a labelling function $\lambda : R \mapsto \Sigma$ that associates to each transition of R a symbol of $\Sigma \cup \{\epsilon\}$, where ϵ denotes the empty sequence ($w \cdot \epsilon = \epsilon \cdot w = w$ for any $w \in \Sigma^*$). In the following, we use $\gamma_1 \xrightarrow{w} \gamma_2$ with $w \in \Sigma^*$ to denote that $\gamma_1 \xrightarrow{\rho_1 \cdots \rho_k} \gamma_2$ and $\lambda(\xrightarrow{\rho_1}) \cdots \lambda(\xrightarrow{\rho_k}) = w$. Furthermore, we associate to T an *initial* configuration $\gamma_{init} \in S$ and a *final* configuration $\gamma_{acc} \in S$ and assume an accepting relation $\bowtie : S \times S$. For a fixed accepting relation $\bowtie$, we define the language accepted (generated) by $T = (S, R, \preceq, \gamma_{init}, \gamma_{acc})$ as:

$$L(T) = \{w \in \Sigma^* \mid \gamma_{init} \xrightarrow{w} \gamma \text{ and } \gamma_{acc} \bowtie \gamma\}$$

In this paper we consider two types of accepting relations:

- *Coverability*: the accepting relation $\bowtie$ is defined as $\gamma_{acc} \preceq \gamma$.
- *Reachability*: the accepting relation $\bowtie$ is defined as $\gamma_{acc} = \gamma$.

Let $\mathcal{M}$ be a wsts model (e.g. Petri nets) and let T be one of its instances (i.e. a particular net). We define $L_c(T)$, resp $L_r(T)$, as the language accepted by T with accepting relation $\bowtie_c$, resp. $\bowtie_r$. We say that L is a c-language, resp. r-language, of $\mathcal{M}$ if there is an instance T of $\mathcal{M}$ such that $L = L_c(T)$, resp. $L = L_r(T)$. We use $L_c(\mathcal{M})$, resp. $L_r(\mathcal{M})$, to denote the class of c-languages, resp. r-languages, of $\mathcal{M}$. Finally, we use $\mathcal{L}_1 \not\sim \mathcal{L}_2$ to denote that $\mathcal{L}_1$ and $\mathcal{L}_2$ are incomparable classes of languages.

3 Constrained Multiset Rewriting Systems

In this section we recall the main definitions and prove the first results for *constrained multiset rewriting systems* [2]. Let us first give some preliminary definitions. We use $\mathbb{N}$ to denote the set of natural numbers and $\overline{n}$ to denote the interval $[0, \ldots, n]$ for any $n \in \mathbb{N}$. We assume a set $\mathbb{V}$ of variables which range over $\mathbb{N}$, and a set $\mathbb{P}$ of unary predicate symbols. For a set A, we use A^* and $A^{\otimes}$ to denote the sets of (finite) words and (finite) multisets over A respectively. Sometimes, we write multisets as lists, so $[1, 5, 5, 1, 1]$ represents a multiset with three occurrences of 1 and two occurrences of 5; $[\ ]$ represents the empty multiset. We use the usual relations and operations such as $\leq$ (inclusion), $+$ (union), and $-$ (difference) on multisets. For a set $V \subseteq \mathbb{V}$, a *valuation* Val of V is a mapping from V to $\mathbb{N}$. A *condition* is a finite conjunction of *gap order* formulas of the forms: $x <_c y$, $x \leq y$, $x = y$, $x < c$, $x > c$, $x = c$, where $x, y \in \mathbb{V}$ and $c \in \mathbb{N}$. Here $x <_c y$ stands for $x + c < y$. We often use $x < y$ instead of $x <_0 y$. Sometimes, we treat a condition ψ as a set, and write e.g. $(x <_c y) \in \psi$ to indicate that $x <_c y$ is one of the conjuncts in ψ. We use *true* to indicate an empty set of conditions. A *term* is of the form $p(x)$ where $p \in \mathbb{P}$ and $x \in \mathbb{V}$. A *ground term* is of the form $p(c)$ where $p \in \mathbb{P}$ and $c \in \mathbb{N}$. We sometimes say that a predicate symbol is *nullary* to mean that its parameter is not relevant (hence may be omitted).

A *constrained multiset rewriting system (CMRS)* $\mathcal{S}$ consists of a finite set of *rules* each of the form $L \rightsquigarrow R : \psi$, where L and R are multisets of terms, and ψ is a condition. We assume that ψ is consistent (otherwise, the rule is never enabled). For a valuation Val, we use $Val(\psi)$ to denote the result of substituting each variable x in ψ by $Val(x)$. We use $Val \models \psi$ to denote that $Val(\psi)$ evaluates to *true*. For a multiset T of terms we define $Val(T)$ as the multiset of ground terms obtained from T by replacing each variable x by $Val(x)$. A *configuration* is a multiset of ground terms. Each rule $\rho = L \rightsquigarrow R : \psi \in \mathcal{S}$ defines a relation between configurations. More precisely, $\gamma \xrightarrow{\rho} \gamma'$ if and only if there is a valuation Val s.t. the following conditions are satisfied: (i) $Val \models \psi$, (ii) $\gamma \geq Val(L)$, and (iii) $\gamma' = \gamma - Val(L) + Val(R)$. As an example, consider the rule:

$$\rho = [p(x)\ ,\ q(y)] \ \rightsquigarrow\ [q(z)\ ,\ r(x)\ ,\ r(w)] \ \ :\ \ \{x <_2 y\ ,\ x <_4 z\ ,\ z <_0 w\}$$

A valuation which satisfies the condition is $Val(x) = 1$, $Val(y) = 4$, $Val(z) = 8$, $Val(w) = 10$, Therefore, we have that $[p(1), p(3), q(4)] \xrightarrow{\rho} [p(3), q(8), r(1), r(10)]$. A run σ is a sequence of transitions $\gamma_0 \xrightarrow{\rho_1} \gamma_1 \xrightarrow{\rho_2} \ldots \xrightarrow{\rho_n} \gamma_n$; where $\lambda(\rho_1) \cdot \ldots \lambda(\rho_n)$ is the word associated to σ for some labelling function λ.

Let us fix a CMRS $\mathcal{S}$ operating on a set of predicate symbols $\mathbb{P}$. Let $cmax$ be the maximal constant which appears in the rules of $\mathcal{S}$; $cmax$ is equal to 0 if there are no constant in $\mathcal{S}$. We now define an ordering $\preceq_c$ on configurations extracted from the ordering defined in [2] to solve the coverability problem.

Definition 1. *Given a configuration γ, we define the* index *of γ, $index(\gamma)$, to be a word of the form $D_0 \ \cdots D_{cmax} \ d_0 \ B_0 \ d_1 \ B_1 \ d_2 \ \cdots \ d_n \ B_n$ where*

- *$D_0, \ldots, D_{cmax}, B_0, \ldots, B_n \in \mathbb{P}^{\otimes}$ and $d_0, d_1, \ldots, d_n \in \mathbb{N} \setminus \{0\}$;*
- *B_i must not be empty for $0 \leq i \leq n$;*
- *for each $p \in \mathbb{P}$, D_i contains k occurrences of predicate p iff $p(i)$ occurs k times in γ for $0 \leq i \leq cmax$;*
- *given $v_0 = cmax + d_0$, for each $p \in \mathbb{P}$, B_0 contains k occurrences of predicate p iff $p(v_0)$ occurs k times in γ;*
- *given $v_{i+1} = v_i + d_{i+1}$, for each $p \in \mathbb{P}$, B_{i+1} contains k occurrences of predicate p iff $p(v_{i+1})$ occurs k times in γ for all $0 \leq i < n$;*
- *for all $p(v) \in \gamma$ with $v > cmax$, there exists $i : 0 \leq i \leq n$ such that $v = cmax + d_0 + d_1 + \ldots + d_i$.*

The ordering $\preceq_c$ is defined as follows.

Definition 2. *Let $D_0 \ D_1 \ \cdots \ D_{cmax} \ d_0 \ B_0 \ d_1 \ B_1 \ d_2 \ \cdots \ d_n \ B_n$ be the index of a configuration γ_1 and $D'_0 \ D'_1 \ \cdots \ D'_{cmax} \ d'_0 \ B'_0 \ d'_1 \ B'_1 \ d'_2 \ \cdots \ d'_m \ B'_m$ be the index of a configuration γ_2. Then, $\gamma_1 \preceq_c \gamma_2$ iff $D_i \leq D'_i$ for $0 \leq i \leq cmax$ and there exists a strictly monotone injection $h : \overline{n} \mapsto \overline{m}$ such that $B_0 \leq B'_{h(0)}$, $B_i \leq B'_{h(i)}$, $d_0 \leq \sum_{k=0}^{h(0)} d'_k$, and $d_i \leq \sum_{k=h(i-1)+1}^{h(i)} d'_k$ for $1 \leq i \leq n$.*

In the rest of the paper we assume that the values appearing in the initial configuration γ_{init} and in the accepting configuration γ_{acc} are smaller or equal than $cmax$. The ordering $\preceq_c$ is obtained by composing string embedding and multiset inclusion. From standard properties of orderings, it follows that $\preceq_c$ is a well-quasi ordering. Furthermore, a CMRS is monotonic with respect to corresponding ordering $\preceq_c$. The following property then holds.

Proposition 1. *A CMRS $\mathcal{S}$ equipped with $\preceq_c$ is well-structured.*

We now define a restriction $\ll$ of the relation $\preceq_c$ in which we require that the distribution of predicates in two configurations has the same structure but larger gaps. Formally, under the assumptions of Def. 2, $\gamma_1 \ll \gamma_2$ iff $n = m$, $D_i = D'_i$ for $0 \leq i \leq cmax$, $B_j = B'_j$ for $0 \leq j \leq n$, and $d_k \leq d'_k$ for $0 \leq k \leq n$. A CMRS $\mathcal{S}$ satisfies then the following property (the proof is given in [3]).

Proposition 2. *Let $\gamma_0 \xrightarrow{\rho_0} \gamma_1 \xrightarrow{\rho_1} \ldots \xrightarrow{\rho_{k-1}} \gamma_k \xrightarrow{\rho_k} \gamma$ be a run of $\mathcal{S}$. For any γ' s.t. $\gamma \ll \gamma'$ there exist $\gamma'_1, \ldots, \gamma'_k$ such that $\gamma_i \ll \gamma'_i$ for $i : 1 \leq i \leq k$ and $\gamma_0 \xrightarrow{\rho_1} \gamma'_1 \xrightarrow{\rho_1} \ldots \xrightarrow{\rho_{k-1}} \gamma'_k \xrightarrow{\rho_k} \gamma'$ is still a run of $\mathcal{S}$.*

In other words, with coverability accepting conditions a CMRS $\mathcal{S}$ can recognize a word w passing through configurations where gaps between parameters strictly greater than $cmax$ can be arbitrarily large. As discussed in the rest of the paper this property is very important to compare c-languages accepted by (fragments of) CMRS with those accepted by other wsts.

We are ready now to give a first characterization for the expressive power of CMRS. In [14, Prop. 4], the authors show that there exists a recursively enumerable (RE) language that cannot be recognized by any wsts with coverability acceptance. Hence, the following proposition holds.

Theorem 1. *$L_c(CMRS) \subset RE$.*

With reachability as accepting condition, CMRS recognize instead the class of recursively enumerable languages (RE).

Theorem 2. *$L_r(CMRS) = RE$.*

Proof. We prove that CMRS can weakly simulate 2-counter machines. A *2-counter machine (CM)* operates on two counters and on a finite set Q of control states. A transition updates the control state and executes either an increment, a decrement, or a zero-test of one of the two counters. Operations and tests on counters have their usual semantics, assuming that the values of counters are natural values. In the initial configuration the counters are set to zero. A 2-counter machine accepts an execution if it ends into the control state q_f. Assume a CM $\mathcal{M}$. The CMRS $\mathcal{S}$ that weakly simulates $\mathcal{M}$ operates in a sequence of phases indexed by natural numbers. Counters are represented as a multiset of terms of the form $cnt_1(c)$ and $cnt_2(c)$ where c denotes the current phase. During each phase, $\mathcal{S}$ simulates increment and decrement transitions of $\mathcal{M}$. As soon as $\mathcal{M}$ performs a zero-test of a counter, $\mathcal{S}$ enters an intermediate stage. After conclusion of the intermediate stage, a new phase is started and the index phase is increased. Transitions from q_1 to q_2 that update the current value of a counter are encoded by Γ_0 rules of the following form (they have the same labels as the corresponding CM transitions):

$$\begin{array}{lrl}
(q_1, cnt_i\!:=cnt_i{+}1, q_2) \Rightarrow & [q_1, phase(x)] \rightsquigarrow [q_2, phase(x), cnt_i(x)] & : \ true \\
(q_1, cnt_i\!:=cnt_i{-}1, q_2) \Rightarrow & [q_1, phase(x), cnt_i(x)] \rightsquigarrow [q_2, phase(x)] & : \ true
\end{array}$$

In these rules we update the value of the i-th counter by adding or deleting one occurrence of the term $cnt_i(c)$. Notice that the parameter c must be equal to the current phase index. A transition $(q_1, cnt_1 = 0?, q_2)$ labeled with a is encoded by the following Γ_0 rules (the two first labeled with ϵ, the last one with a):

$$\begin{array}{rcl}
[q_1, phase(x), phase'(x)] \rightsquigarrow [q_2', phase(y), phase'(x)] & : & \{x < y\} \\
[q_2', cnt_2(x), phase(y), phase'(x)] \rightsquigarrow [q_2', cnt_2(y), phase(y), phase'(x)] & : & true \\
[q_2', phase(y), phase'(x)] \rightsquigarrow [q_2, phase(y), phase'(y)] & : & true
\end{array}$$

(The Γ_0 rules encoding the test on cnt_2 are obtained from the previous ones by replacing predicate cnt_2 with cnt_1.) In the first rule we store the current index using $phase'$, and generate a new index which is strictly larger than the current one. This resets counter cnt_1 since all ground terms in its encoding will now have too small arguments for other rules in $\mathcal{S}$ to modify them. With the second rule, we change the arguments of (some of) the ground terms encoding cnt_2 to the new index. The third rule terminates the simulation of the zero-test.

Finally, we add to $\mathcal{S}$ the following rules (all labeled by ϵ) for $i \in \{1, 2\}$:

$$\begin{array}{rcll} [q_{\mathit{fin}}] & \rightsquigarrow & [q''_{\mathit{fin}}] & : \ \mathit{true} \\ [q''_{\mathit{fin}}\ ,\ \mathit{phase}(x)\ ,\ \mathit{cnt}_i(x)] & \rightsquigarrow & [q''_{\mathit{fin}}\ ,\ \mathit{phase}(x)] & : \ \mathit{true} \\ [q''_{\mathit{fin}}\ ,\ \mathit{phase}(x)\ ,\ \mathit{phase}'(y)] & \rightsquigarrow & [q''_{\mathit{fin}}] & : \ \mathit{true} \end{array}$$

By means of these additional rules, when we reach state q_{fin} we can move to q''_{fin} and erase the ground terms corresponding to the counters. The key observation here is that ground terms with parameters strictly less than the current phase are not removed during the simulation procedure described above. This implies that there exists an execution where $\mathcal{S}$ recognizes a word w that reaches $[q_f]$ iff there exists an execution where CM recognizes the word w that reaches q_f. Finally, the class of languages accepted by 2-counter machines with reachability accepting condition is RE. □

4 Lossy FIFO Channel Systems

In this section we study the relationship between a fragment of CMRS, we named $\mathit{\Gamma}_0$, and *lossy (FIFO) channel systems* (LCS) [5].

In the fragment $\mathit{\Gamma}_0$ of CMRS every rule $L \rightsquigarrow R : \psi$ satisfies the following conditions: every variable x occurs at most once in L and at most once in R, and ψ does not contain equality constraints. As an example, $[p(x), r(y)] \rightsquigarrow [q(x), r(z)] : x < y, y < z$ is a rule in $\mathit{\Gamma}_0$, whereas $[p(x), q(x)] \rightsquigarrow [q(y)] : \mathit{true}$ and $[p(x)] \rightsquigarrow [q(y), r(y)] : \mathit{true}$ are not in $\mathit{\Gamma}_0$.

A *Lossy FIFO Channel System* (LCS) consists of an asynchronous parallel composition of finite-state machines that communicate through sending and receiving messages via a finite set of unbounded lossy FIFO channels (in the sense that they can non-deterministically lose messages). Formally, an LCS $\mathcal{F}$ is a tuple (Q, C, N, δ) where Q is a finite set of control states (the Cartesian product of those of each finite-state machine), C is a finite set of channels, M is a finite set of messages, δ is a finite set of transitions, each of which is of the form (q_1, Op, q_2) where $q_1, q_2 \in Q$, and Op is a mapping from channels to channel operations. For any $c \in C$ and $a \in M$, an operation $Op(c)$ is either a *send* operation $!a$, a *receive* operation $?a$, the *empty* test $\epsilon?$, or the *null* operation *nop*. A configuration γ is a pair (q, w) where $q \in Q$, and w is a mapping from C to M^* giving the content of each channel. The initial configuration γ_{init} of $\mathcal{F}$ is the pair (q_0, ε) where $q_0 \in Q$, and ε denotes the mapping that assigns the empty sequence ϵ to each channel. The (strong) transition relation (that defines the semantics of machines with *perfect* FIFO channels) is defined as follows: $(q_1, w_1) \xrightarrow{\sigma} (q_2, w_2)$ if and only if $\sigma = (q_1, Op, q_2) \in \delta$ such that if $Op(c) =!a$, then $w_2(c) = w_1(c) \cdot a$; if $Op(c) =?a$, then $w_1(c) = a \cdot w_2(c)$; if $Op(c) = \epsilon?$ then $w_1(c) = \epsilon$ and $w_2(c) = \epsilon$; if $Op(c) = nop$, then $w_2(c) = w_1(c)$. Now let $\preceq_l$ be the quasi ordering on LCS configurations such that $(q_1, w_1) \preceq_l (q_2, w_2)$ iff

$q_1 = q_2$ and $\forall c \in C : w_1(c) \preceq_w w_2(c)$ where $\preceq_w$ indicates the subword relation. By Higman's theorem, we know that $\preceq_l$ is a well-quasi ordering. We introduce then the weak transition relation $\stackrel{\sigma}{\Longrightarrow}$ that defines the semantics of LCS: we have $\gamma_1 \stackrel{\sigma}{\Longrightarrow} \gamma_2$ iff there exists γ_1' and γ_2' s.t. $\gamma_1' \preceq_l \gamma_1$, $\gamma_1' \stackrel{\sigma}{\longrightarrow} \gamma_2'$, and $\gamma_2 \preceq_l \gamma_2'$. Thus, $\gamma_1 \stackrel{\sigma}{\Longrightarrow} \gamma_2$ means that γ_2 is reachable from γ_1 by first losing messages from the channels and reaching γ_1', then performing a transition, and, thereafter losing again messages from channels. As shown in [5], an LCS is well-structured with respect to $\preceq_l$. Furthermore, as shown in [6], in presence of transitions labeled with ϵ, we can restrict our attention to systems with only one channel. As a last remark, notice that for any model with lossy semantics like LCS, e.g. lossy vector addition systems [18], the class of c-languages coincide with the class of r-languages, i.e., $L_r(LCS) = L_c(LCS)$.

Our first result is that Γ_0 and LCS define the same class of c-languages.

Theorem 3. $L_c(\Gamma_0) = L_c(LCS)$.

Proof. The proof is based on encodings of LCS into Γ_0 and of Γ_0 into LCS. We next sketch the main ideas behind the two encodings (the complete proof is in [3]). In the encoding of an LCS in Γ_0, we represent the content $a_1 \ldots a_n$ of a channel c as a multiset $M_c = [h_c(x), a_1(x_1), \ldots, a_n(x_n), t_c(y)]$ where $x < x_1 < \ldots < x_n < y$. The predicates h_c (head) and t_c (tail) are used as sentinels to mark the two ends of the queue. The operation $!a$ on channel c is implemented by adding a new ground term with predicate a to M_c and by moving the tail to the right. The operation $?a$ on channel c is implemented by consuming an element with predicate a chosen non-deterministically from the multiset M_c and moving the head to the right. This operation simulates a lossy channel in the sense that when we update h_c we forget all elements to the left of the deleted element. Finally, the empty test is simulated by a reset of the channel. Formally, we encode LCS transitions operating on channel c into the following Γ_0 rules with the same labels:

$$\begin{array}{lcrl}
(q_1, !a, q_2) & \Rightarrow & [q_1, t_c(x)] \rightsquigarrow [q_2, a(x), t_c(y)] & : \ \{x < y\} \\
(q_1, ?a, q_2) & \Rightarrow & [q_1, h_c(x), a(y)] \rightsquigarrow [q_2, h_c(y)] & : \ \{x < y\} \\
(q_1, \epsilon?, q_2) & \Rightarrow & [q_1, h_c(x), tail_c(y)] \rightsquigarrow [q_2, h_c(x'), tail(y')] & : \ \{y < x', x' < y'\}
\end{array}$$

It is easy to verify that the resulting Γ_0 model accepts the same language as the original LCS.

The encoding of Γ_0 into LCS is more complicate and exploits special properties of Γ_0. We first exploits Prop. 2 to observe that for any CMRS $\mathcal{S}$ with initial and accepting configuration γ_{init} and γ_{acc} if we replace each gap order formula $x <_c y$ in $\mathcal{S}$ by $x < y$ we obtain a CMRS $\mathcal{S}'$ such that $L_c(\mathcal{S}) = L_c(\mathcal{S}')$. Hence, we assume w.l.o.g. that there is no gap order formula $x <_c y$ with $c > 0$ in $\mathcal{S}$. Secondly, when considering c-languages of a Γ_0 model, we can always restrict our attention to configurations in which ground terms (with parameter greater than $cmax$) are totally ordered with respect to $<$, i.e. in which there cannot be two ground terms (with parameter greater than $cmax$) with the same parameter. The proof of this property requires some attention. A Γ_0-rule may produce indeed a configuration containing two or more ground terms with the same parameter (e.g. when the right-hand side contains unconstrained variables as in $[\,] \rightsquigarrow [p(x), p(y)] : true$). We notice however that Γ_0-rules cannot use equality as guard in a condition. Thus, we can always choose a different evaluation for the variables in a condition such that ground terms assume distinct values and such that the word accepted by the corresponding execution remains the same. As a consequence, a Γ_0 configuration can be represented as a word of predicate symbols (We recall that CMRS configurations are words of multisets of predicate symbols as shown by the definition of $index(\cdot)$). Thus, a Γ_0-rule operates on configurations as a transformation of words. With these properties in mind, it comes natural to build an LCS that uses a lossy channel to encode a configuration and operations on (auxiliary) channels to simulate the transformations on words defined by a Γ_0-rule with lossy semantics. The thesis follows by noticing that, as for any other wsts, a version of Γ_0 with lossy semantics recognizes the same c-languages as those accepted by Γ_0. $\square$

We show next that CMRS are strictly more expressive than LCS and Γ_0.

Theorem 4. *$L_c(LCS) \subset L_c(CMRS)$.*

Proof. We define a language L_{ent} which is accepted by a CMRS and that cannot be accepted by any LCS. Assume a finite alphabet Σ such that $\{\$, \#\} \not\subseteq \Sigma$. For each $w = a_1 \cdots a_k \in \Sigma^*$, we interpret w in the following as the multiset $[a_1, \ldots, a_k]$. Hence, we do not distinguish words in Σ^* from the multiset they represent, and vice versa. In particular, we will use the notation $a_1 \cdots a_k \leq a'_1 \cdots a'_l$ to denote that $[a_1, \ldots, a_k] \leq [a'_1, \ldots, a'_l]$. Define V to be the set of words of the form $w_1\#w_2\# \cdots \#w_n$ where $w_i \in \Sigma^*$ for each $i : 1 \leq i \leq n$. Consider $v = w_1\#w_2\# \cdots \#w_m \in V$ and $v' = w'_1\#w'_2\# \cdots \#w'_n \in V$. We write $v \sqsubseteq v'$ to denote that there is an injection $h : \{1, \ldots, m\} \mapsto \{1, \ldots, n\}$ such that

1. $1 \leq i < j \leq m$ implies $h(i) < h(j)$ (h is monotonic) and
2. $w_i \leq w'_{h(i)}$ ($\leq$ is multiset inclusion) for each $i : 1 \leq i \leq m$.

We now define the language $L_{ent} = \{v\$v' \mid v' \sqsubseteq v\} \subseteq (\Sigma \cup \{\#, \$\})^*$. As an example, given $\Sigma = \{a, b\}$, we have that $[a, b, b]\#[a, b, b]\#[a, a]\$[b, a]\#[a, a]$ is in L_{ent}, whereas $[a, b, b]\#[b, a, b]\#[a, a]\$[a, a]\#[a, b]$ is not in L_{ent}.

We now exhibit a CMRS $\mathcal{S}$ with $L_c(\mathcal{S}) = L_{ent}$. The set of predicate symbols which appear in $\mathcal{S}$ consists of (i) a predicate symbol a for each $a \in \Sigma$, and (ii) the symbols *guess*, *check*, $sep_\#$ and *ok*. The initial configuration γ_{init} is defined as $[guess(0)]$. Furthermore, we have the following rules:
(1) For each $a \in \Sigma$, we have a rule labelled with a and which is of the form

$$[guess(x)] \rightsquigarrow [guess(x)\ ,\ a(x)] \ :\ true$$

Rules of this form are used to guess the letters in w_i in the first part of a word in L_{ent}. We keep track of the symbols inside w_i through their argument. These arguments are all the same by definition of the rule.
(2) A rule labelled with $\#$ of the form:

$$[guess(x)] \rightsquigarrow [sep_\#(x)\ ,\ guess(y)] \ :\ \{x < y\}$$

This rule is used to switch from the guessing of the part w_i to the guessing of the next part w_{i+1}. $sep_{\#}(x)$ remembers the parameter on which the switch has been executed.
(3) A rule labelled with \$ of the form:

$$[guess(x)\ ] \rightsquigarrow [check(y)\ ,\ sep_{\#}(x)] \quad : \quad \{y = 0\}$$

This rule is used to switch from the guessing of the part $w_1\#\ldots\#w_n$ to the selection of the second part of the word. The parameter of *check* is equal to the initial value of *guess*, i.e. to 0. This way, we can scan the word stored in the first phase from left-to-right, i.e., working on the argument order we define a monotonic injective mapping h.
(4) For each $a \in \Sigma$, we have a rule labelled with a which is of the form

$$[check(y)\ ,\ a(y)] \rightsquigarrow [check(y)] \quad : \quad true$$

This rule is used to read a word (multiset) u_i contained in $w_{h(i)}$.
(5) A rule labelled with $\#$ of the form:

$$[check(x)\ ,\ sep_{\#}(x)\ ,\ sep_{\#}(y)\ ] \rightsquigarrow [check(y)\ ,\ sep_{\#}(y)] \quad : \quad \{x < y\}$$

This rule is used to pass from u_i to u_{i+1} for $i \geq 1$.
(6) A rule labelled with ϵ of the form:

$$[check(x)\ ] \rightsquigarrow [ok(y)] \quad : \quad \{y = 0\}$$

This rule is used to non-deterministically terminate the checking phase. The accepting configuration γ_{acc} is defined as $[ok(0)]$.

Assuming that $\Sigma = \{a, b\}$, we now show that L_{ent} is not an LCS language. Suppose that $L_c(\mathcal{F}) = L_{ent}$ for some LCS $\mathcal{F} = (Q, \{c\}, M, \delta)$. We show that this leads to a contradiction. Let γ_{init} be the initial global state in $\mathcal{F}$ and γ_{acc} be the accepting global state. We use a binary encoding $enc : Q \cup M \mapsto \Sigma^*$ such that $enc(m) \not\leq enc(m')$ if $m \neq m'$. We will also use a special word $v_{init} \in \Sigma^*$ such that $v_{init} \not\leq enc(m)$ for each $m \in Q \cup M$. It is clear that such *enc* function and v_{init} exist. As an example, if $|Q \cup M| = n$ then we define *enc* as an injective map from $Q \cup M$ to multisets of $n+1$ elements with $i+1$ occurrences of a and $n-i$ occurrences of b for $0 \leq i \leq n$, and we use the multiset with $n+1$ occurrences of b for v_{init}. For instance, for $n = 2$ we use $[a, a, a], [a, a, b], [a, b, b]$ for control states and messages and $[b, b, b]$ for v_{init}. We extend *enc* to global states such that if $\gamma = (q, m_1 m_2 \cdots m_n)$ then

$$enc(\gamma) = enc(q)\#enc(m_1)\#enc(m_2)\#\cdots\#enc(m_n)$$

Observe that (i) $enc(\gamma) \in V$; (ii) for global states γ_1 and γ_2, it is the case that $\gamma_1 \preceq_l \gamma_2$ iff $enc(\gamma_1) \sqsubseteq enc(\gamma_2)$; and (iii) $v_{init} \not\sqsubseteq enc(\gamma)$ for each global state γ.

Since $L_{ent} = L_c(\mathcal{F})$ and $v\$v \in L_{ent}$ for each $v \in V$, it follows that for each $v \in V$, there is a global state γ such that $\gamma_{init} \xrightarrow{v} \gamma \xrightarrow{\$v} \gamma'$ with $\gamma_{acc} \preceq_l \gamma'$. We use $reach(v)$ to denote γ. We define two sequences $\gamma_0, \gamma_1, \gamma_2, \ldots$ of global states, and $v_0, v_1, v_2, \ldots$ of words in V such that $v_0 = v_{init}$, $\gamma_i = \mathit{reach}(v_i)$, and $v_{i+1} = \mathit{enc}(\gamma_i)$ for each $i \geq 0$. By Higman's theorem we know that there is a j such that $\gamma_i \preceq_l \gamma_j$ for some $i < j$. Let j be the smallest natural number satisfying this property. First, we show that $v_i \not\sqsubseteq v_j$. There are two cases: if $i = 0$ then $v_i \not\sqsubseteq v_j$ by (iii); if $i > 0$ then we know that $\gamma_{i-1} \not\preceq_l \gamma_{j-1}$ and hence, following (ii), $v_i = \mathit{enc}(\gamma_{i-1}) \not\sqsubseteq \mathit{enc}(\gamma_{j-1}) = v_j$. Since $\gamma_j = \mathit{reach}(v_j)$, we know that $\gamma_{init} \xrightarrow{v_j} \gamma_j$. By monotonicity, $\gamma_i \xrightarrow{\$v_i} \gamma'_i, \gamma_{acc} \preceq_l \gamma'_i, \gamma_i \preceq_l \gamma_j$ implies $\gamma_j \xrightarrow{\$v_i} \gamma'_j$ with $\gamma_{acc} \preceq_l \gamma'_i \preceq_l \gamma'_j$. We conclude that $\gamma_{init} \xrightarrow{v_j} \gamma_j \xrightarrow{\$v_i} \gamma'_j$ with $\gamma_{acc} \preceq_l \gamma'_j$. Hence, $v_j\$v_i \in L_c(\mathcal{F}) = L_{ent}$ which is a contradiction since $v_i \not\sqsubseteq v_j$. ☐

Let us now consider r-languages. As mentioned at the beginning of the section, the expressive power of LCS remains the same as for coverability accepting conditions, However, this property does not hold anymore for Γ_0.

Proposition 3. $L_c(\Gamma_0) \subset L_r(\Gamma_0) = L_r(CMRS) = RE$.

Proof. It is well known that *perfect* FIFO channel systems with reachability accepting condition recognize the class RE. We prove that perfect channel systems accept the same languages as Γ_0 with reachability accepting condition. Given an LCS $\mathcal{F}$, let $\mathcal{S}$ be the Γ_0 used to encode an LCS in the proof of Theorem 3. In each step of a run σ in $\mathcal{S}$ the head and tail delimiters are moved to the right of their current positions. Thus, a "lost" ground term to left of the head delimiter, i.e. with parameter smaller than that of h_c, can never be removed in successive steps of σ. This implies that an accepting configuration in which all ground terms have parameters strictly greater than the parameter of the head delimiter characterize reachable configurations of a perfect FIFO channel system. ☐

Hence, we have the following property.

Corollary 1. $L_r(LCS) \subset L_r(CMRS)$.

5 Petri Nets and their Extensions

Petri nets (PN), a well-known model of concurrent computation [19], can naturally be reformulated in a multiset rewriting system operating on nullary predicates only (i.e. predicates with no parameters). Let us call Γ_1 this fragment of CMRS. It is easy to see that, if we associate a predicate symbol to each place of a net, configurations and rules of a Γ_1 model are just alternative representations of markings and transitions of a Petri net. As an immediate consequence of this connection, we have that $L_c(\Gamma_1) = L_c(PN)$ and $L_r(\Gamma_1) = L_r(PN)$. To formally compare Γ_1 with the other models, we use some known results on languages accepted by extensions of Petri nets. A *lossy Petri net with inhibitor arcs* (LN) is a Petri net in which it is possible to test if a place has no tokens and in which tokens may get lost before and after executing a transition. A *transfer net* [10] (TN) is a Petri net extended with transfer arcs. A transfer arc specifies an atomic transfer of all tokens in a given set of source places to a given target place. Finally, a *reset net* [10] is a Petri net in which it is possible to atomically remove all tokens from a given place. LN, TN, and RN are well-structured with respect to the inclusion ordering of markings (see, e.g., [10, 11]). For these models, it is simple to verify that $L_c(LN) = L_c(RN) = L_c(TN)$, $L_c(LN) \subseteq L_c(LCS)$, and, as for LCS, $L_r(LN) = L_c(LN)$ (see for [3] for formal proofs). Furthermore, in [14] the authors proved that $L_c(PN) \subset L_c(TN)$. From all these properties, we obtain the following result.

Theorem 5. *$L_c(\Gamma_1) \subset L_c(\Gamma_0)$.*

For r-languages, the classification changes as follows.

Theorem 6. *$L_r(\Gamma_1) \not\sim L_r(LCS)$, $L_r(\Gamma_1) \not\sim L_r(LN)$, and $L_r(\Gamma_1) \subset L_r(\Gamma_0)$.*

Proof. We first prove that $L_r(\Gamma_1) = L_r(PN) \not\subseteq L_c(LCS) = L_r(LCS)$, hence $L_r(\Gamma_1) \not\subseteq L_c(LN) = L_r(LN)$ since $L_c(LN) \subseteq L_c(LCS) = L_r(LCS)$. Consider the language $L = \{a^n b^n \mid n \geq 0\}$. It is easy to verify that there exists a Petri net $\mathcal{N}$ such that $L_r(\mathcal{N}) = L$. We now prove that $L \notin L_r(LCS)$. Per absurdum, suppose there exists an LCS $\mathcal{F}$ such that $L_c(\mathcal{F}) = L$. For any $k \geq 1$, let γ_k and γ'_k be two global states s.t. γ_{init} leads to γ_k by accepting the word a^k, γ_k leads to γ'_k by accepting the word b^k, and $\gamma_{acc} \preceq_l \gamma'_k$. Since $\preceq_l$ is a well-quasi ordering, there exists $i < j$ such that $\gamma_i \preceq_l \gamma_j$. By monotonicity of $\mathcal{F}$, we have γ_j leads to γ'' by accepting the word b^i and $\gamma_{acc} \preceq_l \gamma'_i \preceq_l \gamma''$. We conclude that $a^j b^i \in L_c(\mathcal{F})$ with $i < j$, which gives us a contradiction.

We now prove that $L_c(LN) \not\subseteq L_r(\Gamma_1)$, hence $L_c(LCS) \not\subseteq L_r(\Gamma_1)$. Let $\Sigma = \{a, b\}$ and let L_{par} be the language over the alphabet $\Sigma \cup \{\#\}$ that contains all the words $w_1\#\ldots\#w_n$ with $n \geq 0$ such that $w_i \in \Sigma^*$ and there is no prefix of w_i that contains more occurrences of symbol b than those of symbol a, for $i : 1 \leq i \leq n$. Notice that the number of occurrences of symbols a and b in w_i may be different. The language can be accepted by a LN defined as follows. When we accept the symbol a we add one token in a special place p_a. To accept the symbol b, we remove one token from p_a. To pass from w_i to w_{i+1}, we accept symbol # whenever p_a is empty (in LN the empty test is just a reset).
We now show that L_{par} cannot be recognized by a Petri net. Suppose that there exists a Petri net $\mathcal{N}$ such that $L_r(\mathcal{N}) = L_{par}$. Starting from $\mathcal{N}$, we build a net $\mathcal{N}_1$ by adding a new place d that keeps track of the difference between the number of occurrences of symbols a and b in the prefix of the word that is being processed in $\mathcal{N}$. Furthermore, we add the condition that d is empty to the accepting marking of $\mathcal{N}$. It is easy to verify that $\mathcal{N}_1$ accepts the language L_{bal} consisting of words of the form $w = w_1\#\cdots\#w_n$ where w_i belongs the the language of *balanced parentheses* on the alphabet Σ for $i : 1 \leq i \leq n$. We exploit now [15, Lemma 9.8] that states that L_{bal} cannot be recognized by a Petri net with reachability accepting condition, which gives us a contradiction.

Finally, the property $L_r(\Gamma_1) = L_r(PN) \subset L_r(\Gamma_0)$ follows from [15, Lemma 9.8] and Prop. 3, Indeed, we have that $L_{bal} \in L_r(\Gamma_0) = RE$ and $L_{bal} \notin L_r(\Gamma_1)$. ☐

Finally, we observe that we can use an argument similar to that used in the proof of Theorem 6 to show that $L_r(PN) \not\sim L_c(CMRS)$.

6 (Integral) Relational Automata

In this section we compare the class of languages accepted by a fragment of CMRS, called Γ_2, with those accepted by *relational automata* [9].

The fragment Γ_2 is defined as follows. Let us first use $|B|$ to denote the cardinality of a multiset B. Γ_2 is the fragment of CMRS in which a rule $L \rightsquigarrow R : \psi$ satisfies the condition $|R| \leq |L|$. In other words, in Γ_2 the cardinality of a reachable configuration is always bounded by the cardinality of the initial configuration.

An *(integral) relational automaton* (RA) operates on a finite set X of positive integer variables, and is of the form (Q, δ) where Q and δ are finite sets of control states and transitions respectively. A transition is a triple (q_1, op, q_2) where $q_1, q_2 \in Q$ and op is of one of the following three operations: (i) *reading*: $read(x)$ reads a new value of variable x (i.e. assigns a non-deterministically chosen value to x), (ii) *assignment*: $x := y$ assigns the value of variable y to x; (ii) *testing*: $x < y$, $x = y$, $x < c$, $x = c$, and $x > c$ are guards which compare the values of variables x, y and the natural constant c. Assume a RA $\mathcal{A} = (Q, \delta)$. A *valuation* v is a mapping form X to $\mathbb{N}$. A *configuration* is of the form (q, v), where $q \in Q$ and v is a valuation. We define γ_{init} to be (q_{init}, v_{init}) where $q_{init} \in Q$ and $v_{init}(x) = 0$ for all $x \in X$. For a transition $\rho \in \delta$ of the form (q_1, op, q_2), we let $\gamma_1 \xrightarrow{\rho} \gamma_2$ if and only if $\gamma_1 = (q_1, v_1)$, $\gamma_2 = (q_2, v_2)$, and one of the following holds: $op = read(x)$ and $v_2(y) = v_1(y)$ for each $y \in X - \{x\}$; $op = (y := x)$, $v_2(z) = v_1(z)$ for each $z \in X - \{y\}$, and $v_2(y) = v_1(x)$; $op = (x < y)$, $v_2 = v_1$, and $v_1(x) < v_1(y)$. Other testing operations are defined in a similar manner. In [9] Cerans has shown that RA equipped with the *sparser-than* order of tuples of natural numbers are well-structured. In the case of RA with that order, the coverability accepting condition is equivalent to the control state acceptance, i.e., a word is accepted if it is recognized by an execution ending in a particular control state $q_{acc} \in Q$.

As stated in the following propositions, RA and Γ_2 define the same class of c- and r-languages.

Proposition 4. $L_c(\Gamma_2) = L_c(RA)$.

Proof. Given an RA $\mathcal{A} = (Q, \delta)$ over the set of variables X, we can build the Γ_2 $\mathcal{S}$ defined below. The set of predicate symbols in $\mathcal{S}$ consists of the following: (i) for each $q \in Q$, there is a predicate symbol q in $\mathcal{S}$; and (ii) for each variable x in X, there is a predicate symbol q_x in $\mathcal{S}$. Transitions in δ are encoded via the following CMRS rules (with the same labels)

$$\begin{array}{lcl}
(q_1, read(x), q_2) & \Rightarrow & [q_1, p_x(z)] \rightsquigarrow [q_2, p_x(w)] : true \\
(q_1, x := y, q_2) & \Rightarrow & [q_1, p_x(z), p_y(w)] \rightsquigarrow [q_2, p_x(w), p_y(w)] : true \\
(q_1, x < y, q_2) & \Rightarrow & [q_1, p_x(z), p_y(w)] \rightsquigarrow [q_2, p_x(z), p_y(w)] : \{z < w\}
\end{array}$$

For $X = \{x_1, \ldots, x_n\}$, the initial configuration is $\gamma_{init} = [q_0, p_{x_1}(0), \ldots, p_{x_n}(0)]$. The accepting configuration γ_{acc} is the multiset $[q_{acc}]$.

For the other inclusion, by using Prop. 2, we assume w.l.o.g. that there is no gap order formula $x <_c y$ with $c > 0$ in $\mathcal{S}$ and that $\gamma_{acc} = [ok]$. To justify the second assumption, notice that we can always introduce new predicate symbols ko and ok and a new rule that can be executed only on a configuration γ with $\gamma_{acc} \preceq_c \gamma$ and that replace ko with ok. All other rules are modified to be enabled only at configurations containing ko. Finally, we also observe that we can assume that all configurations of $\mathcal{S}$ have the same size (the size of the initial configuration of the Γ_2 model). Thus, we associate a variable of X to each ground term of the initial CMRS configuration and compose the predicate symbols in a CMRS configuration to form a single control state. CRMS rules can then be simulated in several steps by operations on variables and updates of control states. To each control state containing ok, we add a transition labeled with ϵ to the accepting control state q_{acc}. □

Theorem 7. *$L_c(\Gamma_2)$ = Regular Languages.*

To prove this claim, we define a finite state automaton where states are abstractions of configurations in which we only keep the order on parameters and not their exact values (when parameters are greater than $cmax$). The relation transition of the symbolic graph mimics the transition relation of $\mathcal{S}$. Then, we show (by using Prop. 2) that the symbolic graph contains exactly the information we need to characterize the language recognized by $\mathcal{S}$. Finiteness of the graph allows us to conclude that Γ_2 corresponds to the class of regular languages. The complete construction is given in [3].

We are ready now to compare Γ_2 (hence RA) with the other models studied in this paper. For this purpose, we first observe that Petri nets can accept regular languages (finite automata can be encoded as Petri nets). Furthermore, it is straightforward to build a Petri net that accepts a non-regular language like $L = \{a^n \# b^m \mid n \geq m\}$. As a consequence of this observation and of Theorem 7, we have the following result.

Corollary 2. *$L_c(\Gamma_2) \subset L_c(\Gamma_1)$.*

Let us now consider the reachability accepting condition. We first notice that $L_c(\Gamma_2) = L_r(\Gamma_2) = L_c(RA) = L_r(RA)$. Indeed, in both cases of Γ_2 and RA we can encode the reachability acceptance into the coverability acceptance by adding transitions (labelled with ϵ) that can be fired only from the accepting configuration and leads to a configuration with control state q_{acc} in the case of RA and a configuration containing a special accepting predicate symbol in the case of Γ_2. Thus, we have the following property.

Theorem 8. *$L_r(\Gamma_2) \subset L_r(\Gamma_1)$.*

7 Conclusions

In this paper we have compared the class of languages recognized with coverability and reachability as accepting conditions by *relational automata* (RA), *Petri nets* (PN), *lossy channel systems* (LCS), and *constrained multiset rewriting systems* (CMRS). With both accepting conditions, CMRS turns out to be the most expressive model among the different well-structured systems considered in the paper. Indeed, with coverability as accepting condition we have that $FA = RA < PN < LCS < CMRS < CM$, whereas with reachability we have that $FA = RA < PN, LCS < CMRS = CM$ and PN and LCS are incomparable models. Here FA and CM denote resp. *finite automata* (they recognize regular languages) and *counter automata* (they recognize recursively enumerable languages), and $<$ means "strictly less expressive than". We also prove that *transfer nets*, *reset nets*, *broadcast protocols* and *lossy vector addition systems* are strictly less expressive than CMRS. CMRS can thus be viewed as a unified model for analysis and verification of a large class of infinite-state models.

References

1. P. A. Abdulla, K. Čerāns, B. Jonsson, and T. Yih-Kuen. General decidability theorems for infinite-state systems. LICS 1996: 313–321.
2. P. A. Abdulla and G. Delzanno. On the coverability problem for constrained multiset rewriting. AVIS 2006, an ETAPS 2006 Workshop.
3. P. A. Abdulla, G. Delzanno, and L. Van Begin. Comparing the expressive power of well-structured transition systems. Technical Report, DISI, June 2007.
4. P. A. Abdulla and B. Jonsson. Model checking of systems with many identical timed processes. TCS 290(1): 241-264. 2003.
5. P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. Inf. Comput. 127(2): 91-101 (1996)
6. P. A. Abdulla and B. Jonsson. Undecidable verification problems for programs with unreliable channels. Inf. Comput. 130(1): 71-90 (1996)
7. N. Bertrand and Ph. Schnoebelen. A short visit to the STS hierarchy. ENTCS 154(3): 59-69 (2006)
8. G. Cécé, A. Finkel, and S. P. Iyer. Unreliable channels are easier to verify than perfect channels. Inf. Comput. 124(1): 20-31 (1996)
9. K. Čerāns. Deciding properties of integral relational automata. ICALP 1994: 35–46.
10. C. Dufourd, A. Finkel, P. Schnoebelen. Reset nets between decidability and undecidability. ICALP 1998: 103-115
11. J. Esparza, A. Finkel, R. Mayr. On the verification of broadcast protocols. LICS 1999: 352–359.
12. A. Finkel, G. Geeraerts, J.-F. Raskin and L. Van Begin. On the ω-language expressive power of extended petri nets. TCS 356(3):374-386, 2006.
13. A. Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! TCS 256(1-2): 63-92 (2001).
14. G. Geeraerts, J.-F. Raskin and L. Van Begin. Well-structured languages. Technical report 542, ULB, 2005.
15. M. Hack. Petri net languages. Technical report 159, MIT, 1976.
16. T. A. Henzinger, R. Majumdar, and J.-F. Raskin. A classification of symbolic transition systems. ACM Trans. Comput. Log. 6(1): 1-32 (2005)
17. R. Lazić, T. Newcomb, J. Ouaknine, A.W. Roscoe, J. Worell. Nets with tokens which carry data. To appear in ATPN 2007.
18. R. Mayr. Undecidable problems in unreliable computations. LATIN 2000: 377–386.
19. C. A. Petri. *Kommunikation mit Automaten.* PhD Thesis. Univ. of Bonn. 1962.