On the Expressive Power of Restriction and Priorities in CCS with replication

Jesus Aranda, Frank Valencia, Cristian Versari

On the Expressive Power of Restriction and Priorities in CCS with replication

Jesús Aranda[1], Frank D. Valencia[2], and Cristian Versari[3]

[1] LIX École Polytechnique and Universidad del Valle Colombia **
[2] CNRS and LIX École Polytechnique
[3] Università di Bologna

Abstract. We study the expressive power of restriction and its interplay with replication. We do this by considering several syntactic variants of CCS$_!$ (CCS with replication instead of recursion) which differ from each other in the use of restriction with respect to replication. We consider three syntactic variations of CCS$_!$ which do not allow the use of an unbounded number of restrictions: CCS$_!^{-!\nu}$ is the fragment of CCS$_!$ not allowing restrictions under the scope of a replication. CCS$_!^{-\nu}$ is the restriction-free fragment of CCS$_!$. The third variant is CCS$_{!+pr}^{-!\nu}$ which extends CCS$_!^{-!\nu}$ with Phillips' priority guards.
We show that the use of unboundedly many restrictions in CCS$_!$ is necessary for obtaining Turing expressiveness in the sense of Busi et al [8]. We do this by showing that there is no encoding of RAMs into CCS$_!^{-!\nu}$ which preserves and reflects convergence. We also prove that up to failures equivalence, there is no encoding from CCS$_!$ into CCS$_!^{-!\nu}$ nor from CCS$_!^{-!\nu}$ into CCS$_!^{-\nu}$. As lemmata for the above results we prove that convergence is decidable for CCS$_!^{-!\nu}$ and that language equivalence is decidable for CCS$_!^{-\nu}$. As corollary it follows that convergence is decidable for restriction-free CCS. Finally, we show the expressive power of priorities by providing an encoding of RAMs in CCS$_{!+pr}^{-!\nu}$.

1 Introduction

As for other language-based formalisms (e.g., logic, formal grammars, λ-calculus, etc) a fundamental part of the research in *process calculi* involves the study of the expressiveness of *syntactic* fragments or variants of a given process calculus.

Process calculi provide a *language* in which the structure of *terms* represents the structure of processes together with a *transition* relation to represent computational steps. Consider for example CCS [15]. The parallel composition term $P|Q$, which is built from the terms P and Q represents the process that results from the parallel execution of the processes P and Q. The *restriction* $(\nu x)P$ represents a process P with a private/local/restricted/bound resource x–e.g., a location, a link, or a name. Processes with infinite behaviour are often specified with *recursive expressions* of the form $\mu X.P$ which behaves as $P[\mu X.P/X]$, i.e., P with the (free) occurrences of X replaced by $\mu X.P$. A transition semantics dictates that if P may have a transition into P' by performing an action α, written $P \xrightarrow{\alpha} P'$, then $P \mid Q \xrightarrow{\alpha} P' \mid Q$ and if α does not involve x also $(\nu x)P \xrightarrow{\alpha} (\nu x)P'$.

** The work of Jesús Aranda has been supported by COLCIENCIAS (Instituto Colombiano para el Desarrollo de la Ciencia y la Tecnología "Francisco José de Caldas") and INRIA Futurs.

Classifying Criteria. One natural approach to comparing expressiveness of two given process calculus variants is by comparing them wrt some standard process equivalence $\asymp$. If there exists a computable function (*encoding*) $[\![\cdot]\!]$ from the terms of a variant $\mathcal{C}$ into the terms of another variant $\mathcal{C}'$ such that for every P in $\mathcal{C}$ we have $P \asymp [\![P]\!]$, we say that $\mathcal{C}'$ is *at least as expressive as* $\mathcal{C}$ up to $\asymp$.

Another way of classifying the variants of a given calculus is by considering the complexity or decidability of a fundamental property of processes. For example, the decidability of *convergence* (the existence of a terminating computation) or *divergence* (the existence of a non-terminating computation).

The CCS$_!$ Calculus. The CCS$_!$ calculus [7] is a variant of CCS which instead of using recursive expressions to specify infinite behaviour uses processes of form $!P$. The replicated process $!P$ can be thought of as abbreviating the parallel composition $P \mid P \mid \ldots$ of an unbounded number of P processes. In [8] it is shown that CCS$_!$ is less expressive than CCS wrt (weak) bisimilarity. It is also shown that convergence is undecidable for CCS$_!$ while divergence, unlike for CCS, is decidable.

Turing Expressiveness and Convergence in CCS$_!$. A remarkable expressiveness result in [8] states that, in spite of its being less expressive than CCS, CCS$_!$ is in fact Turing powerful. This is done by encoding Random Access Machines (RAMs) [16]. The fundamental property of the encoding is that it *preserves (and reflects) convergence*; i.e., the RAM converges if and only if its encoding converges.

The CCS$_!$ encoding of RAMs in [8] uses an unbounded number of restrictions arising from having restriction operators *under the scope* of a replication operator as for example in $!(\nu x)P$. Similarly, the CCS encoding of RAMs in [7] involves also an unbounded number of restrictions arising from having restrictions under the scope of recursive expressions as for example in $\mu X.(\nu x)(P \mid X)$. One then may wonder if the generation of unboundedly many names is necessary for Turing Expressiveness.

This Work. In this paper we study the expressiveness of restriction and its interplay with replication. We do this by considering two syntactic fragments of CCS$_!$, namely CCS$_!^{-!\nu}$ and CCS$_!^{-\nu}$ which differ from CCS$_!$ in the occurrences of restriction under the scope of replication. These fragments and a variant of CCS$_!$, CCS$_{!+pr}^{-!\nu}$, as well as our classification criteria are described and motivated below.

Although different in nature, our work was inspired by the study of decidable classes (wrt satisfiability) of formulae involving the occurrence of existential quantifiers *under the scope* of universal quantification. E.g., Skolem showed that the class of formulae of the form $\forall y_1 \ldots y_n \exists z_1 \ldots z_m F$, where F is quantifier-free formula, is undecidable while from Gödel we know that its subclass $\forall y_1 y_2 \exists z_1 \ldots z_m F$ is decidable [5].

The CCS$_!$ Variants. As explained above CCS$_!$ allows processes with restriction under the scope of replication and hence they can generate an unbounded number of restricted names. In order to allow only processes with a number of restricted names bounded by their size, we consider CCS$_!^{-!\nu}$ which represents the CCS$_!$ fragment which does not allow restrictions under the scope of replications. To illustrate the expressiveness of CCS$_!^{-!\nu}$ take for example $P = (\nu k)(\nu u)(\bar{k} \mid !(k.a.(\bar{k} \mid \bar{u})) \mid k.!(u.b))$ which uses only

two restricted names. The reader familiar with CCS can verify that the set of (maximal) finite sequences of visible actions performed by P corresponds to the *context-free* language $a^n b^n$. A similar but slightly more complex example involves a $\text{CCS}_!^{-!\nu}$ process with only five restricted names whose set of (maximal) finite sequences of visible actions corresponds to the *context-sensitive* language $a^n b^n c^n$–see [2].

Now, one may wonder whether a process that uses only a number of restricted names bounded by its size, can be encoded, perhaps by introducing some additional non-restricted names, into one which uses none. For this purpose we shall also consider the *restriction-free* fragment of $\text{CCS}_!$, which shall denote as $\text{CCS}_!^{-\nu}$.

Finally, we may also wonder whether some other natural process construct can replace the use in $\text{CCS}_!$ of unboundedly many restrictions in achieving Turing expressiveness. For this purpose we shall consider $\text{CCS}_{!+pr}^{-!\nu}$ which is $\text{CCS}_!^{-!\nu}$ extended with Phillips' *priority guards* construct [17].

Classifying Criteria. Our main comparison criteria for the above variants are the decidability of *convergence* and their relative expressiveness wrt *failures equivalence* [6,15].

As mentioned before, convergence is a fundamental property of processes and its preservation and reflection are also fundamental properties of the encoding of RAMs in $\text{CCS}_!$. Furthermore, we choose it over divergence because the former is undecidable for $\text{CCS}_!$ while the latter is already known to be decidable for $\text{CCS}_!$.

Failures equivalence is a well-established notion of process equivalence and we choose it over other equivalences because of its sensitivity to convergence. In fact unlike failures equivalence, other standard equivalences for observable behaviour such as weak-bisimilarity, must testing, trace equivalence and language equivalence may actually equate a convergent process with a non-convergent one. This claim about sensitivity to convergence will be shown later on in the paper (Section 3) once we fix our notation.

Contributions. Our main contributions are the following:

- We show that convergence is decidable for $\text{CCS}_!^{-!\nu}$ and thus that there is no (computable) encoding, which preserves and reflects convergence, of RAMs using only a bounded number of restricted names. We do this by encoding $\text{CCS}_!^{-!\nu}$ into Petri Nets. Thus convergence is also decidable for the fragment of CCS with no restrictions within recursive expressions, here refered to as $\text{CCS}^{-\mu\nu}$, because of the convergence preserving and reflecting encoding into $\text{CCS}_!^{-!\nu}$ given in [12].
- We show that, up to failures equivalence, $\text{CCS}_!$ is strictly more expressive than $\text{CCS}_!^{-!\nu}$ and, similarly, that $\text{CCS}_!^{-!\nu}$ is strictly more expressive than $\text{CCS}_!^{-\nu}$. Thus up to failures equivalence, we cannot encode a process with an unbounded number of restrictions into one with a bounded number of restrictions, nor one with a bounded number of restrictions into a restriction-free process.
- We show that adding Phillips' priority guards to $\text{CCS}_!^{-!\nu}$ renders the resulting calculus capable of encoding RAMs. Furthermore, unlike the encoding into $\text{CCS}_!$ and just like the encoding into CCS, the encoding of RAMs into $\text{CCS}_{!+pr}^{-!\nu}$ preserves and reflects both convergence and divergence. This bears witness to the expressive power of Phillips' priority guards.

The classification of the various fragments mentioned above are summarized in Figure 1. The undecidability of convergence and decidability of divergence for $\text{CCS}_!$ as well as the undecidability of both divergence and convergence for CCS were shown in [7,8]. The other results are derived from the work here presented.

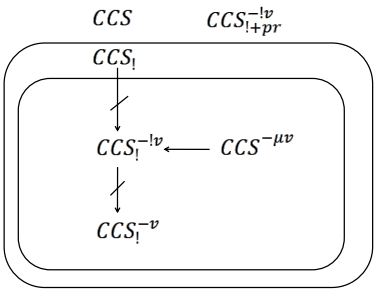

Fig. 1. A (crossed) arrow from $\mathcal{C}$ to $\mathcal{C}'$ represents the (non) existence of an encoding from $\mathcal{C}$ into $\mathcal{C}'$ preserving and reflecting failures equivalence. Convergence is/isn't decidable for each $\mathcal{C}$ in/outside the inner rectangle. Divergence is/isn't decidable for each $\mathcal{C}$ in/outside the outer rectangle.

2 The Calculi

CCS processes can perform actions or synchronize on them. These actions can be either offering port *names* for communication, or the so-called *silent* action τ. We presuppose a countable set $\mathcal{N}$ of port *names*, ranged over by $a, b, x, y \ldots$ and their primed versions. We then introduce a set of *co-names* $\overline{\mathcal{N}} = \{\overline{a} \mid a \in \mathcal{N}\}$ disjoint from $\mathcal{N}$. The set of *labels*, ranged over by l and l', is $\mathcal{L} = \mathcal{N} \cup \overline{\mathcal{N}}$. The set of *actions* Act, ranged over by α and β, extends $\mathcal{L}$ with a new symbol τ. Actions a and $\overline{a}$ are thought of as *complementary*, so we decree that $\overline{\overline{a}} = a$. We also decree that $\overline{\tau} = \tau$.

The $\text{CCS}_!$ processes [7] are defined as in CCS except that recursive expressions are replaced by replication expression of the form $!P$.

Definition 1. *Processes in $\text{CCS}_!$ are built from names by the following syntax:*

$$P, Q \ldots := 0 \mid \alpha.P \;\Big|\; P + Q \;\Big|\; P \mid Q \;\Big|\; (\nu a)P \;\Big|\; !P \tag{1}$$

Convention 1 *We use $\Sigma_{i \in I} P_i$ where $I = \{i_1 \ldots i_n\}$, to denote $P_{i_1} + \ldots + P_{i_n}$., the order of the summands being insignificant. We use $\Pi_{i \in I} P_i$, where $I = \{i_1 \ldots i_n\}$, to denote $P_{i_1} \mid \ldots \mid P_{i_n}$. Both $\Sigma_{i \in I} P_i$ and $\Pi_{i \in I} P_i$ are assumed to be 0 if $I = \emptyset$. The names a and $\bar{a}$ in P are said to be* bound *in $(\nu a)P$. The* bound names *of P, $bn(P)$, are those with a bound occurrence in P, and the* free names *of P, $fn(P)$, are those*

with an not bound occurrence in P. The set of names of P, $n(P)$, is then given by $fn(P) \cup bn(P)$. We use $(\nu a_1 \ldots a_n)P$ as an abbreviation of $(\nu a_1)(\nu a_2)\ldots(\nu a_n)P$.

Process expressions are endowed with meaning by using the labeled transitions of the form $P \xrightarrow{\alpha} Q$ which intuitively says that P may perform α and evolve into Q. Similarly, $P \stackrel{s}{\Longrightarrow} Q$, where $s \in \mathcal{L}^*$, means that P can evolve into Q after zero or more transitions labeled with the elements of s without considering τ moves. Formally,

Definition 2. *The labeled transition relation $\stackrel{\cdot}{\longrightarrow}$ is given by the rules in Table 1. Define $\stackrel{s}{\Longrightarrow}$, with $s = \alpha_1 \ldots . \alpha_n \in \mathcal{L}^*$, as $(\xrightarrow{\tau})^* \xrightarrow{\alpha_1} (\xrightarrow{\tau})^* \ldots (\xrightarrow{\tau})^* \xrightarrow{\alpha_n} (\xrightarrow{\tau})^*$. For the empty sequence $s = \epsilon$, $\stackrel{s}{\Longrightarrow}$ is defined as $(\xrightarrow{\tau})^*$.*

$$\text{PREF}\ \frac{}{\alpha.P \xrightarrow{\alpha} P} \qquad \text{RES}\ \frac{P \xrightarrow{\alpha} P'}{(\nu\, a)P \xrightarrow{\alpha} (\nu\, a)P'}\ \text{if } \alpha \notin \{a, \overline{a}\}$$

$$\text{SUM}_1\ \frac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'} \qquad \text{SUM}_2\ \frac{Q \xrightarrow{\alpha} Q'}{P + Q \xrightarrow{\alpha} Q'}$$

$$\text{PAR}_1\ \frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \qquad \text{PAR}_2\ \frac{Q \xrightarrow{\alpha} Q'}{P \mid Q \xrightarrow{\alpha} P \mid Q'}$$

$$\text{COM}\ \frac{P \xrightarrow{l} P' \quad Q \xrightarrow{\overline{l}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad \text{REP}\ \frac{P \mid !P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P'}$$

Table 1. The labeled semantics of CCS$_!$.

Intuition and Basic Ideas. We shall now give some intuition and state some conventions on process expressions. We shall concentrate on the expressions $!P$ and $(\nu x)R$ and their interplay because they are central to our work.

The process $P \mid Q$ represents the parallel execution of P and Q. Intuitively, $P \mid Q$ may perform either an action performed by P, an action performed by Q, or if P and Q perform complementary actions (and thus synchronize), a τ action. Thus, $a.P \mid \bar{a}.Q \xrightarrow{a} P \mid \bar{a}.Q$, $a.P \mid \bar{a}.Q \xrightarrow{\bar{a}} a.P \mid Q$, and $a.P \mid \bar{a}.Q \xrightarrow{\tau} P \mid Q$.

The restriction process $(\nu a)P$ behaves as P except that it can offer neither a nor $\bar{a}$ to its environment. One may think of $(\nu a)P$ as a *local declaration of name* a in P and thus can be used to restrict the possible synchronization (interactions) of a process. For example, $(\nu a)(a.P \mid \bar{a}.Q)$ may not perfom a or $\bar{a}$ though –crucially– it may perform a τ action resulting from the synchronization over the actions a and $\bar{a}$. Thus $(\nu a)(a.P \mid \bar{a}.Q) \not\xrightarrow{\alpha}$ for $\alpha \in \{a, \bar{a}\}$ but $(\nu a)(a.P \mid \bar{a}.Q) \xrightarrow{\tau} (\nu a)(P \mid Q)$.

Replication is the only means to specify infinite behaviour in CCS$_!$. The replication $!P$ behaves as $P \mid P \mid \ldots \mid !P$; unboundedly many $P's$ in parallel.

We can use restriction under the scope of replication to specify an unbounded number of local names. It should be clear from the intuition above that $(\nu a)P$ should behave exactly as $(\nu b)(P[b/a])$ where b is not free in P and $P[b/a]$ is the process that results

from replacing in P every free occurrence of a with b renaming bound names wherever needed to avoid capture (α-conversion). Thus, for example $!(\nu a)P$ can be viewed as

$$(\nu a_1)P[a_1/a] \mid (\nu a_2)P[a_2/a] \mid \ \ldots \ \mid !(\nu a)P$$

thus allowing the declaration of unboundedly many different restricted names $a_1, a_2, \ldots$. As previously mentioned in the introduction, this sort of unbounded generation of restricted names appears to be crucial in the encodings of Turing-powerful formalisms.

The Sub-calculi. To understand the above-mentioned interplay between restriction and replication we shall consider two calculi which arise from syntactic constraints over $\text{CCS}_!$. Namely, $\text{CCS}_!^{-!\nu}$ and $\text{CCS}_!^{-\nu}$.

Definition 3 ($\text{CCS}_!^{-!\nu}$ and $\text{CCS}_!^{-\nu}$). *The processes of $\text{CCS}_!^{-!\nu}$ are those $\text{CCS}_!$ processes which do not have occurrences of a process of the form $(\nu x)P$ within a process of the form $!R$. The processes of $\text{CCS}_!^{-\nu}$ are those $\text{CCS}_!$ processes with no occurrences of processes of the form $(\nu x)P$.*

3 Convergence, Failures and Related Notions

In this section we shall introduce the notions we shall consider as classifying criteria, namely *convergence* and *failures equivalence*, as well as some other related notions.

Following [4], we say that a process generates a sequence of non-silent actions s if it can perform the actions of s in a finite maximal sequence of transitions. More precisely:

Definition 4 (Sequence and language generation). *The process P* generates *a sequence $s \in \mathcal{L}^*$ if and only if there exists Q such that $P \stackrel{s}{\Longrightarrow} Q$ and $Q \stackrel{\alpha}{\not\longrightarrow}$ for any $\alpha \in Act$. Define the language of (or generated by) a process P, $L(P)$, as the set of all sequences P generates. We say that P and Q are language equivalent, written $P \sim_L Q$, iff $L(P) = L(Q)$.*

We recall the notion of *failure* following [15]. We need the following notion:

Definition 5. *We say that P is stable iff $P \stackrel{\tau}{\not\longrightarrow}$.*

Intuitively we say that a pair $\langle e, L\rangle$, with $e \in \mathcal{L}^*$ and $L \subseteq \mathcal{L}$, is failure of P if P can perform e and thereby reach a state in which no further action (including τ) is possible if the environment will only allow actions in L.

Definition 6 (Failures). *A pair $\langle e, L\rangle$, where $e \in \mathcal{L}^*$ and $L \subseteq \mathcal{L}$, is a failure of P iff there is P' such that: (1) $P \stackrel{e}{\Longrightarrow} P'$, (2) $P' \stackrel{l}{\not\longrightarrow}$ for all $l \in L$, and (3) P' is stable.*

Define $Failures(P)$ as the set of failures of a process P. We say that P and Q are failures equivalent*, written $P \sim_F Q$ iff $Failures(P) = Failures(Q)$.*

We recall the notions of convergence and divergence following [7,8]. Intuitively, a process converges if it can reach a stable process after a sequence of τ moves. A process is deemed divergent iff it can perform an infinite sequence of τ moves.

Definition 7 (Convergence and Divergence). *We say that P is convergent iff there is a stable process Q such that $P(\xrightarrow{\tau})^*Q$. We say that P is divergent iff $P(\xrightarrow{\tau})^\omega$, i.e., there exists an infinite sequence $P = P_0 \xrightarrow{\tau} P_1 \xrightarrow{\tau} \ldots$.*

We conclude this section by stating relations between the above notions which we shall use in the following sections.

3.1 Some Basic Properties of Failures

We claimed in the introduction that unlike other standard notions such as weak bisimilarity, must testing and trace equivalence, failures equivalence never equates a convergent process with a non-convergent one. In fact,

Proposition 1. *Suppose that $P \sim_F Q$. Then P is convergent iff Q is convergent.*

To justify the rest of the above claim, take $P = \tau.!a.0$ and $P' = !\tau.0$. Clearly P converges but P' does not, however they are both language equivalent. Now take $Q = \tau.!\tau.0 + \tau.0$ and $Q' = !\tau.0$. Thus Q converges but Q' does not. It can be verified that Q and Q' are equated by these standard equivalences.

We shall use the fact that failures equivalence implies language equivalence.

Proposition 2. $\sim_F \subseteq \sim_L$.

4 Decidability of Convergence in $\mathbf{CCS}_!^{-!\nu}$

In this section we show the decidability of convergence for $\mathrm{CCS}_!^{-!\nu}$ by a reduction to the same problem for a fragment of Petri Nets.

4.1 Convergence-invariant properties in fragments of $\mathbf{CCS}_!^{-!\nu}$

Notice that decidability of convergence in $\mathrm{CCS}_!^{-!\nu}$ can be reduced to the decidability of convergence in $\mathrm{CCS}_!^{-\nu}$.

Proposition 3. *For every P in $CCS_!^{-!\nu}$ one can effectively construct a $CCS_!^{-\nu}$ process P', such that P converges if only if P' converges.*

Proof. (Outline) First α-convert P so that each bound name in P is replaced with a unique bound name. Then remove from the resulting process each occurrence of a "(νx)". Let P' be the resulting restriction-free process. One can verify that P' converges iff P converges. □

Consequently, in what follows we reduce the convergence problem for $\mathrm{CCS}_!^{-\nu}$ to convergence problem in Petri nets.

In order to simplify the reduction to Petri Nets, we shall consider the fragment $\mathrm{CCS}_{s!}^{-\nu}$ of those $\mathrm{CCS}_!^{-\nu}$ processes in which replication can only be applied to prefix or summation processses.

Proposition 4. *For every $CCS_!^{-\nu}$ process P, one can effectively construct a $CCS_{s!}^{-\nu}$ process Q such that P converges iff Q converges.*

Proof. (Outline) Systematically replace in P every occurrence of the form $!!R$ with $!R$, $!(Q|Q')$ with $!Q \mid !Q'$, and $!0$ with 0. The resulting process converges iff P converges.

Finitely Branching Transition System. In order to prove the decidability of convergence, we shall make use of an alternative but equivalent definition (up to failures equivalence) of the transition relation for $\text{CCS}_!$. The equivalent definition can be obtained by replacing Rule Rep in Table 1 with the rules in Table 2.

$$\text{REPL}_1 \ \frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P' \mid !P} \qquad \text{REPL}_2 \ \frac{P \xrightarrow{\alpha} P' \quad P \xrightarrow{\overline{\alpha}} P''}{!P \xrightarrow{\tau} P' \mid P'' \mid !P}$$

Table 2.

One can verify that the resulting transition relation is finitely-branching. This is essential for being able to provide an effective Petri net construction for any given $\text{CCS}_{s!}^{-\nu}$ process.

4.2 The Reduction to Petri Nets

Here we shall provide a (Unlabelled Place/Transition) Petri Net semantics for $\text{CCS}_{s!}^{-\nu}$ which considers only the τ moves. For these Petri Nets convergence is decidable [11].

Definition 8 (Petri Nets). *A Petri net is a 3-tuple (S, T, m_0), where S is a set of places, T is a set of transitions $\mathcal{M}_{fin}(S) \times \mathcal{M}_{fin}(S)$ with $\mathcal{M}_{fin}(S)$ being a finite multiset of S called a marking. The (non-empty) multiset m_0 is the initial marking; for each place $s \in S$, there are $m_0(s)$ tokens.*

A transition (c, p) is written in the form $c \Longrightarrow p$. A transition is enabled at a marking m if $c \subseteq m$. The execution of the transition produces the marking $m' = (\ m \setminus c\ ) \oplus p$ (where $\setminus$ and $\oplus$ are the difference and the union operators on multisets). This is written as $m \rhd m'$. If no transition is enable at m we say that m is a dead marking.

We say that the Petri net (S, T, m_0) converges *iff there exists a dead marking m' such that $m_0(\rhd)^* m'$.*

Intuitively, we will associate to each $\text{CCS}_{s!}^{-\nu}$ a Petri net so that:

- Places are identified as syntactic components reachable from P,
- Markings are descriptions of processes reachable from P through τ-actions. The places and tokens in the marking represent different syntactic components and their number of occurrences in the process described.
- Transitions represent the τ-actions enabled to be performed at certain process. Input places correspond to the components in the process involved in the τ-action and Output places are the components to be enabled once the τ-action has been executed.

Given a Petri net for P the elements of $Sub(P)$ below will be the syntactic components represented by places in the Petri net.

Definition 9. *Define $Sub(P)$, where $P \in \text{CCS}_{s!}^{-\nu}$, as $Sub(0) = \{0\}$, $Sub(\Sigma_{i\in I} P_i) = \{\Sigma_{i\in I} P_i\} \cup (\bigcup_{i\in I} Sub(P_i))$, $Sub(\alpha.P) = \{\alpha.P\} \ \cup \ Sub(P)$, $Sub(!P) = \{!P\} \ \cup \ Sub(P)$, $Sub(P \mid Q) = Sub(P) \ \cup \ Sub(Q)$.*

$Sub(P)$ denotes the set all null, replicated, summation, prefix processes occurring in P. Since a process P may have several parallel occurrences of an element in $Sub(P)$ we use a multi-set $Occur(P)$ take into account its number of occurrences.

Definition 10 (Occurrence). *Let $P \in CCS_{s!}^{-\nu}$. The multiset of processes which occur in P, $Occur(P)$, is given by the following rule: $Occur(P) = Occur(Q) \oplus Occur(R)$ if $P = Q \mid R$ else $Occur(P) = \{P\}$. Furthermore, we say that Q is an occurrence of a process P if and only if $Q \in Occur(P)$.*

$Occur(P)$ associates to a $\text{CCS}_{s!}^{-\nu}$ process P the multiset of its immediate parallel components (occurrences) and will be identified as the marking of P in the Petri net.

We are now ready to define our Petri net encoding of $\text{CCS}_{s!}^{-\nu}$ processes.

Definition 11 (Nets for $\text{CCS}_{s!}^{-\nu}$). *Given a $CCS_{s!}^{-\nu}$ process P, we define its Petri net $N_P = (S, T, m_0)$ where $S = \{Q \mid Q \in Sub(P)\}$, $m_0 = Occur(P)$ and $T = T_1 \cup T_2$ where: $T_1 = \{\{P\} \Longrightarrow Occur(P') \mid P \xrightarrow{\tau} P'\}$ and $T_2 = \{\{P, Q\} \Longrightarrow Occur(P') \oplus Occur(Q') | P \xrightarrow{\alpha} P' \text{ and } Q \xrightarrow{\overline{\alpha}} Q'\}$.*

Clearly, given P, N_P can be effectively constructed—here we use the finite-branching nature of the alternative transition semantics in Section 4.1.

Roughly speaking, the set of transitions T represents the possible τ moves to be performed and the initial marking $Occur(P)$ is the one which identifies the process P. In particular:

- T_1 : this type of transition reflects a τ move coming from one of the components, it is referred as P, going to the process P'. Notice as a token representing P is consumed and the tokens representing P', there might be more than one component, are added, in this way the transition reflects the evolution from the component P into the process P' .
- T_2 : this type of transition reflects the τ-actions resulting from the synchronisation of two components P and Q, as a result of the synchronisation the processes P' and Q' are reached, in this case, a token associated to both P and another one associated to Q are consumed, the tokens representing P' and Q' are added.

Wee can now state the correctness of the encoding of $\text{CCS}_{!}^{-\nu}$ into Petri nets.

Lemma 1 (Convergence-invariance property between $\text{CCS}_{!}^{-\nu}$ and Petri nets). *For any $CCS_{s!}^{-\nu}$ process P, P converges if and only if the Petri net N_P converges.*

Since convergence is decidable for Petri nets [11], we conclude from the above lemma and our effective construction of Petri Nets that convergence is also decidable for $\text{CCS}_{s!}^{-\nu}$. Thus, from Propositions 3 and 4, we obtain the following corollary.

Theorem 2. *Convergence is a decidable property for $CCS_{!}^{-!\nu}$ processes.*

5 Decidability of Language Equivalence in $\mathbf{CCS}_!^{-\nu}$

We now prove that decidability of language equivalence for $\text{CCS}_!^{-\nu}$. The crucial observation is that up to language equivalence every occurrence of a replicated process $!R$ in a $\text{CCS}_!^{-\nu}$ process can be replaced with $!\tau.0$ if R can perform at least an action, otherwise it can be replaced with 0. More precisely, let $P[Q/R]$ the process that results from replacing in P every occurrence of R with Q.

Proposition 5. *Let P be a $\text{CCS}_!^{-\nu}$ process and suppose that $!R$ occurs in P. Then $L(P) = L(P[Q/!R])$ where $Q = !\tau.0$ if there exists α s.t., $R \xrightarrow{\alpha}$ else $Q = 0$.*

Given any R in $\text{CCS}_!$ one can effectively decide whether there exists α such that $R \xrightarrow{\alpha}$ (This can be proven using the alternative finitely-branching presentation of the transition relation in Section 4.1). We can then use the above proposition for proving the following statement.

Lemma 2. *Let P be a $\text{CCS}_!^{-\nu}$ process. One can effectively construct a process P' such that $L(P) = L(P')$ and P' is either $!\tau.0$ or a replication-free $\text{CCS}_!^{-\nu}$ process.*

Proof. (Sketch.) Notice that we can use systematically Proposition 5 to transform any $\text{CCS}_!^{-\nu}$ process P into an language equivalent process Q whose replicated occurrences are all of the form $!\tau.0$. Now a $!\tau.0$ can occur either in a parallel composition, a summation or prefix process. Observe that (1) $P \mid !\tau.0 \sim_L !\tau.0$, (2) $!\tau.0 \mid P \sim_L !\tau.0$, (3) $\alpha.!\tau.0 \sim_L !\tau.0$, (4) $P + !\tau.0 \sim_L P$, (5) $!\tau.0 + P \sim_L P$. One can apply (1-5) from left to right to systematically transform Q into the process P' as required in the lemma. □

From the above lemma, we conclude that every $\text{CCS}_!^{-\nu}$ process can be effectively transformed into a language equivalent finite-state process. Hence,

Theorem 3. *Given P and Q in $\text{CCS}_!^{-\nu}$, the question of whether $L(P) = L(Q)$ is decidable.*

6 Impossibility results for failure-preserving encodings in $\mathbf{CCS}_!$, $\mathbf{CCS}_!^{-!\nu}$ and $\mathbf{CCS}_!^{-\nu}$

In this section, we shall state the impossibility results about the existence of computable encodings from $\text{CCS}_!$ into $\text{CCS}_!^{-!\nu}$ and from $\text{CCS}_!^{-!\nu}$ into $\text{CCS}_!^{-\nu}$ which preserve and reflect failures equivalence. The separation results follow from our previous decidability results and the undecidability results in the literature.

The non-existence of failure-preserving encoding from $\text{CCS}_!$ into $\text{CCS}_!^{-!\nu}$ follows from Proposition 1, Theorem 2 and the undecidability of convergence for $\text{CCS}_!$ [8].

Theorem 4. *There is no computable function $[\![\cdot]\!] : \text{CCS}_! \to \text{CCS}_!^{-!\nu}$ s.t $[\![P]\!] \sim_F P$.*

To state the non-existence of failure-preserving encoding from $\text{CCS}_!^{-!\nu}$ into $\text{CCS}_!^{-\nu}$ we appeal to the undecidability of language equivalence for BPP processes [9,14]. BPP processes form a subset of restriction-free CCS processes. Now we can use the encoding of [12] to transform a restriction-free CCS processes into $\text{CCS}_!^{-!\nu}$—the encoding is correct up to failures equivalence (see [3]). We can therefore conclude, with the help of Proposition 2, that language-equivalence for $\text{CCS}_!^{-!\nu}$ processes is undecidable.

Proposition 6. *Given P and Q in $CCS_!^{-!\nu}$, the problem of whether $P \sim_L Q$ is undecidable.*

From the above proposition, the decidability of language equivalence for $\text{CCS}_!^{-\nu}$ (Theorem 3) and Propositions 1 and 2 we can conclude the following.

Theorem 5. *There is no computable function $[\![\cdot]\!] : CCS_!^{-!\nu} \to CCS_!^{-\nu}$ s.t. $[\![P]\!] \sim_F P$.*

Remark 1. We can use the encoding of [12] to transform any CCS process which uses no restriction within recursive expression into a failures equivalent $\text{CCS}_!^{-!\nu}$ process [3]. Thus, from Proposition 1 and Theorem 2 we can conclude that convergence is also decidable for CCS with no restriction within recursive expressions.

7 Expressiveness of Priorities

In this section we add Phillips' priority guards [17] to $\text{CCS}_!^{-!\nu}$. We shall refer to the resulting calculus as $\text{CCS}_{!+pr}^{-!\nu}$. This calculus corresponds to Phillips' Calculus of Priority Guards (CPG) with replication rather than recursion and no restrictions within the scope of replication—hence it cannot use an unbounded number of restrictions.

We show that $\text{CCS}_{!+pr}^{-!\nu}$ turns out to be Turing powerful in the sense of Busi et al [8] (i.e., preserving and reflecting convergence), thus bearing witness to computational expressiveness of priority guards. Recall that from the previous sections $\text{CCS}_!$, and even CCS, cannot encode Turing machines, in the sense above, without using an unbounded number of restrictions (Theorem 2 and Remark 1).

7.1 $\text{CCS}_{!+pr}^{-!\nu}$

In CPG there are two sets of names: N which corresponds to the set of names used to represent the visible actions in $\text{CCS}_!^{-!\nu}$ and a set of priority names U. Each set has a set of complementary actions : $\bar{N}$ and $\bar{U}$, where $Std = N \cup \bar{N}$ (the standard visible actions), $Pri = U \cup \bar{U}$ (the priority actions), $Vis = Std \cup Pri$ (the visible actions), and $Act = Vis \cup \tau$ (all actions). We let $a, b, \ldots$ range over $N \cup U$; $u, v, \ldots$ over Pri; $\lambda, \ldots$ over Vis; and $\alpha, \beta, \ldots$ over Act. Also $S, T, \ldots$ range over finite subsets of Vis, and $U, V, \ldots$ over finite subsets of Pri.

The syntax of processes in $\text{CCS}_{!+pr}^{-!\nu}$ is like that of $\text{CCS}_!^{-\nu}$, except for the summations which now take the form of *priority-guarded* summations: $\Sigma_{i\in I} S_i : \alpha_i.P_i$ where I and each S_i are finite. The meaning of the priority guard $S : \alpha$ is that α can only be performed if the environment does not offer any action in $\bar{S} \bigcap Pri$ (see [17] for details).

Labelled Transition and Offers

We recall the set $off(P)$ of "higher priority" actions "offered" by P.

Definition 12 (Offers). *Let P be a $CCS_{!+pr}^{-!\nu}$ process and $u \in Pri$. The relation P off u (P offers u) is given by the rules in Table 3. We define $off(P) = \{u \in Pri : P \ off \ u\}$. Finally, we say that P* eschews U *iff $off(P) \cap \bar{U} = \emptyset$.*

$$M + S : u.P + N \ \mathit{off}\ u \quad \mathit{if} \quad u \notin S$$

$$\frac{P \ \mathit{off}\ u}{P \mid Q \ \mathit{off}\ u} \qquad \frac{Q \ \mathit{off}\ u}{P \mid Q \ \mathit{off}\ u}$$

$$\frac{P \ \mathit{off}\ u}{(\nu a) \ P \ \mathit{off}\ u} \ \mathit{if} \ \ a \neq \mathit{name}(u) \qquad \frac{P \ \mathit{off}\ u}{!P \ \mathit{off}\ u}$$

Table 3.

The transitions are conditional on offers from the environment. Intuitively, a transition of the form $P \xrightarrow{\alpha}_U P'$ means that P may perform α as long as the environment does not offer $\overline{u}$ for any $u \in U$ (i.e., the environment "eschews" U). E.g. $a : b.P \xrightarrow{b}_{\{a\}} P$ means that $a : b.P$ may perform b as long as the environment does not offer $\overline{a}$. Thus, $a : b.P \mid \overline{b}.Q$ could evolve into $P \mid Q$ however the system $a : b.P \mid \overline{b}.Q \mid \overline{a}$ could not evolve into $P \mid Q \mid \overline{a}$ as the presence of $\overline{a}$ prevents the execution of b and thus the τ-action resulting from $(b, \overline{b})$ communication. This capability of processes to test the presence or the absence of a channel ready to be performed will be fundamental to represent the test for $zero$ in the encoding of RAMs in $\mathrm{CCS}^{-!\nu}_{!+pr}$ presented in the next subsection. Transitions are determined by the rules in Table 4.

$$\text{SUM} \ M + S : \alpha.P + N \xrightarrow{\alpha}_{S \cap Pri} P \ \ \mathit{if} \ \ \alpha \ \in S \cap Pri$$

$$\text{PAR}_1 \ \frac{P \xrightarrow{\alpha}_U P' \ \ Q \ \ \mathit{eschews}\ U}{P \mid Q \xrightarrow{\alpha}_U P' \mid Q} \qquad \text{PAR}_2 \ \frac{Q \xrightarrow{\alpha}_U Q' \ \ P \ \ \mathit{eschews}\ U}{P \mid Q \xrightarrow{\alpha}_U P \mid Q'}$$

$$\text{REACT} \ \frac{P \xrightarrow{\lambda}_{U_1} P' \quad Q \xrightarrow{\overline{\lambda}}_{U_2} Q' \ \ P \ \ \mathit{eschews}\ U_2 \ \ Q \ \ \mathit{eschews}\ U_1}{P \mid Q \xrightarrow{\tau}_{U_1 \cup U_2} P' \mid Q'}$$

$$\text{REP} \ \frac{P \mid !P \xrightarrow{\alpha}_U P'}{!P \xrightarrow{\alpha}_U P'}$$

$$\text{RES} \ \frac{P \xrightarrow{\alpha}_U P' \ \ \mathit{if} \ \ \alpha \ \notin \{a, \overline{a}\}}{(\nu\, a)P \xrightarrow{\alpha}_{U - \{a, \overline{a}\}} (\nu\, a)P'}$$

$$\text{SUM} \ \frac{}{\Sigma_{i \in I} \alpha_i.P_i \xrightarrow{a_j} P_j} \mathit{if}\ j \ \in \ I$$

Table 4. An operational semantics for $\mathrm{CCS}^{-!\nu}_{!+pr}$.

Convention 6 *We write $P \xrightarrow{\alpha}_\emptyset P'$ as $P \xrightarrow{\alpha} P'$ (i.e., α is not constrained on offers from the environment thus corresponding to a standard CCS$_!$ transition). Thus, the notions of divergence and convergence for $\mathrm{CCS}^{-!\nu}_{!+pr}$ are obtained as in Definition 7 by replacing $\xrightarrow{\tau}$ with $\xrightarrow{\tau}_\emptyset$.*

7.2 Encoding RAMs in $\mathrm{CCS}^{-!\nu}_{!+pr}$

A RAM can be seen as a program consisting of a finite sequence of instructions labeled with numbers $(1 : I_1), (2 : I_2) \ldots, (m : I_m)$ which modify the values of a finite set of

non-negative registers $r_1, \ldots, r_n$. The instructions are either $Incr(r_j)$ which adds 1 to the contents of register r_j and goes to the next instruction, or $DecJump(r_j, l)$ which tests the register r_j value, if it is not zero then decreases it by 1 and goes to the next instruction, otherwise jumps to instruction l.

A state of a RAM is given by $(i, c_1, \ldots, c_n)$ where i is the program counter indicating the next instruction to be executed, and $c_1, \ldots, c_n$ are the current values of the registers $r_1, \ldots, r_n$ (resp.). Given a program its computation proceeds by executing the instructions as indicated by the program counter. The execution stops when the program counter reaches the value $m+1$ where m is the label of the last instruction; in this case we say that the program terminates.

The Encoding. A register r_j with value c_j (written $r_j : c_j$) is modeled by a corresponding number of processes of the form $\overline{u}_j$.

$$[\![(r_j : c_j)]\!] \;=\; \prod_{1}^{c_j} \overline{u}_j$$

The program counter is modeled with the *absence* of $\overline{p}_i$ (i.e., the action p_i is eschwed by the encoding) indicating that the i-th instruction is the next to be executed. The initial value of the program counter is 1 so by using $\prod_{i=2}^{m+1} \overline{p}_i$ we indicate the absence of $\overline{p}_1$.

The increasing instruction is modelled with a process $[\![(i : Incr(r_j))]\!]$ which is guarded by a τ-action which is only performed when there is an absence of $\overline{p}_i$.

$$[\![(i : Incr(r_j))]\!] \;= !(\{p_i\} : \tau.(\overline{p_i} \mid p_{i+1} \mid \overline{u}_j))$$

Once activated, the instruction increases the register r_j by offering $\overline{u}_j$, and goes to the next instruction by both disallowing the current one by offering $\overline{p}_i$ and allowing the next one by performing p_{i+1} so that $\overline{p}_{i+1}$ can be consumed.

The decreasing instruction is defined similarly. In addition we consider the absence of $\overline{u}_j$ to test for zero.

$$[\![(i : DecJump(r_j, l))]\!] = !(\{p_i\} : u_j.(\overline{p_i} \mid p_{i+1})) | !(\{p_i, u_j\} : \tau.(\overline{p_i} \mid p_l))$$

The encoding of a RAM is given below. Without loss of generality we assume that initially the RAM has all its registers set to zero and its program counter is 1.

Definition 13. *Let R be a RAM with program instructions* $(1 : I_1), \ldots, (m : I_m)$ *and registers* $r_1, \ldots, r_n$*. We define its encoding into* $CCS^{-!\nu}_{!+pr}$ *as:*

$[\![R]\!] = (\nu p_1, \ldots, p_{m+1}, u_1, \ldots, u_n)(\prod_{i=1}^{m} [\![(i : I_i)]\!] \mid \prod_{i=1}^{n} [\![(r_i : 0)]\!] \mid \prod_{i=2}^{m} \overline{p}_i)$

The correctness of the encoding is stated as follows (see the extended version [3]).

Theorem 7. *Let* R *be a RAM with program instructions* $(1 : I_1), \ldots, (m : I_m)$ *and registers* $r_1, \ldots, r_n$*. Then,* R *terminates if and only if* $[\![R]\!]$ *converges. Furthermore,* R *does not terminate if and only if* $[\![R]\!]$ *diverges.*

As corollary we obtain that convergence and divergence are *undecidable* for $CCS^{-!\nu}_{!+pr}$.

8 Concluding remarks and related Work

The most closely related works are [7,8] and they were already discussed in the introduction. In [12] the authors study replication and recursion in CCS focusing on the role of restriction and name scoping. In particular they show that $\text{CCS}_!$ is equivalent to CCS with recursion with *static scoping*. The standard CCS is shown to have *dynamic* scoping precisely because the use of restriction within recursive definitions. However, if no restriction appears within recursive expressions then there is no distinction between static and dynamic scoping. Hence, if no restriction is allowed within recursive expressions then we know from [12] that CCS can be encoded in $\text{CCS}_!$, without restriction under replication, while preserving and reflecting convergence. As for the other direction, clearly $\nu X.(P|X)$ behaves as $!P$. Nevertheless, if recursion is required to be *prefix guarded*, it is not clear how to produce an encoding which preserves and reflects convergence—without appealing to the decidability results for $\text{CCS}_!$ here presented. Consider e.g., $E = \nu X.(P|\alpha.X)$ and $!P$. If $\alpha = \tau$ then E does not converge and $!P$ may—take $P = a.0$. If $\alpha \neq \tau$ then E may converge and $!P$ may not—take $P = \tau.0$.

The authors in [10] also pointed out the role of restriction in the expressiveness of CCS. They showed that strong bisimilarity is decidable for restriction-free CCS, in contrast with the undecidability result for CCS [18]. It is not clear to us how to relate strong bisimilarity with convergence or failures equivalence.

The authors of [1] studied a fragment of the asynchronous π-calculus with restricted forms of bound name generation. A closely related result in of [1] is the decidability of the control reachability problem for restriction-free asynchronous π-calculus. This implies the decidability of the same problem for the restriction-free fragment of asynchronous $\text{CCS}_!$ (i.e., only 0 can be prefixed with an output action). It is not obvious how to relate control reachability to failures equivalence or convergence. Also it is not clear how to encode our $\text{CCS}_!$ fragment into restriction-free asynchronous $\text{CCS}_!$.

In [13] a Petri net semantics is proposed for a subset of CCS without restriction and with guarded choice. Also in [18] it was shown that the subset studied in [13] can not be extended significantly. These works also presuppose guarded recursion in their fragments which seem crucial for their Petri net constructions. We do not restrict our Petri net construction to guarded sums. Furthermore, as explained above, it is not clear how to translate $\text{CCS}_!$ into CCS with guarded recursion while preserving convergence.

In [20] the authors show the decidability of convergence for a restriction-free calculus for the compositional description of chemical systems, called *CFG* which seems closely related to CCS. The calculus, however, presupposes guarded summation and guarded recursion and thus, as argued before, it is not clear how to encode $\text{CCS}_!$ into such a calculus while preserving convergence.

In [17] it was shown that priorities add expressive power to CCS by modelling electoral systems that cannot be modelled in CCS. Also [19] studies two process algebras enriched with different priority mechanisms. The work reveals the gap between the two prioritised calculi and the two non prioritised ones by modeling electoral systems. Both works state the impossibility of the existence of an encoding subject to certain structural requirements such as homomorphism wrt parallel composition and name invariance. Our derived impossibility result about the non-existence of convergent preserving encodings makes no structural assumptions on the encodings. Finally, we claim that our

expressivity results involving priorities are also held by using other priority approaches as they provide the capability of processes to know if another process is ready to perform a synchronisation on some channel or not.

Acknowledgments. We would like to thank Ian Phillips and Uwe Nestmann for insightful discussions on the topics here studied. We are also grateful to Jorge A. Pérez and the anonymous reviewers for their remarks and suggestions.

References

1. R. Amadio and C. Meyssonnier. On decidability of the control reachability problem in the asynchronous $\pi-$calculus. *Nordic Journal of Computing*, 9(2), 2002.
2. J. Aranda, C. D. Giusto, M. Nielsen, and F. D. Valencia. Ccs with replication in the chomsky hierarchy: The expressive power of divergence. In *APLAS 2007*, volume 4807 of *Lecture Notes in Computer Science*, pages 383–398. Springer, 2007.
3. J. Aranda, F. Valencia, and C. Versari. On the expressive power of restriction and priorities in ccs with replication. Technical report, l'École Polytechnique. http://www.lix.polytechnique.fr/Labo/Jesus.Aranda/publications/trccs.pdf, 2008.
4. J. C. M. Baeten, J. A. Bergstra, and J. W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. *J.ACM.*, 40(3):653–682, 1993.
5. E. Borger, E. Gradel, and Y. Gurevich. *The Classical Decision Problem*. Springer Verlag, 1994.
6. S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31(3):560–599, July 1984.
7. N. Busi, M. Gabbrielli, and G. Zavattaro. Replication vs. recursive definitions in channel based calculi. In *ICALP'03*, volume 2719 of *LNCS*, pages 133–144. Springer-Verlag, 2003.
8. N. Busi, M. Gabbrielli, and G. Zavattaro. Comparing recursion, replication, and iteration in process calculi. In *ICALP'04*, volume 3142 of *LNCS*, pages 307–319. Springer-Verlag, 2004.
9. S. Christensen. *Decidability and Decomposition in Process Algebras*. PhD thesis, Edinburgh University, 1993.
10. S. Christensen, Y. Hirshfeld, and F. Moller:. Decidable subsets of ccs. *Comput. J*, 37(4):233–242, 1994.
11. J. Esparza and M. Nielsen. Decidability issues for petri nets. Technical report, BRICS RS-94-8, 1994.
12. P. Giambiagi, G. Schneider, and F. D. Valencia. On the expressiveness of infinite behavior and name scoping in process calculi. In *FoSSaCS*, volume 2987 of *LNCS*. Springer, 2004.
13. U. Goltz. Ccs and petri nets. In I. Guessarian, editor, *Semantics of Systems of Concurrent Processes*, volume 469 of *LNCS*, pages 334–357. Springer, 1990.
14. Y. Hirshfeld. Petri nets and the equivalence problem. In *Proceedings of CSL'93*, volume 832 of *LNCS*, pages 165–174. Springer Verlag, 1993.
15. R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
16. M. Minsky. *Computation: finite and infinite machines*. Prentice Hall, 1967.
17. I. Phillips. Ccs with priority guards. *J. Log. Algebr. Program.*, 75(1):139–165, 2008.
18. D. Taubner. Finite representation of CCS and TCSP programs by automata and Petri nets. In *LNCS 369*. Springer Verlag, 1989.
19. C. Versari, N. Busi, and R. Gorrieri. On the expressive power of global and local priority in process calculi. In *CONCUR*, volume 4703 of *LNCS*, pages 241–255. Springer, 2007.
20. G. Zavattaro and L. Cardelli. Termination problems in chemical kinetics. In *CONCUR*, volume 5201 of *LNCS*, pages 477–491. Springer, 2008.