Vietnam J Comput Sci (2016) 3:15–34
DOI 10.1007/s40595-015-0051-z

REGULAR PAPER

Language representability of finite place/transition Petri nets

Roberto Gorrieri[1]

Received: 9 May 2015 / Accepted: 8 October 2015 / Published online: 24 October 2015

Abstract Finite-net multi-CCS is a CCS-like calculus which is able to model atomic sequences of actions and, together with parallel composition, also multi-party synchronization. This calculus is equipped with a labeled transition system semantics and also with an unsafe P/T Petri net semantics, which is sound w.r.t. the transition system semantics. For any process p of the calculus, the net associated to p by the semantics has always a finite number of places, but it has a finite number of transitions only for so-called well-formed processes. The main result of the paper is that well-formed finite-net multi-CCS processes are able to represent all finite, statically reduced, P/T Petri nets.

Keywords Operational semantics · Process algebra · Petri nets

1 Introduction

Finite-state labeled transition systems (i.e., LTSs with finitely many states and transitions) can be expressed by the CCS [25] sub-calculus of *finite-state processes*, i.e., the sequential processes generated from the empty process **0**, prefixing $\mu.p$, alternative composition $p_1 + p_2$ and a finite number of process constants C, each one equipped with a defining equation of the form $C \stackrel{\text{def}}{=} p$. More precisely, the semantics of any finite-state CCS process is a finite-state LTS and, conversely, given a reduced, finite-state LTS TS, it is possible to define a finite-state CCS process p_{TS} such that the operational semantics for p_{TS} generates an LTS isomorphic to TS. Hence, this famous result of Milner offers a process calculus to represent, up to isomorphism, all and only finite-state LTSs.

This paper addresses the same language representability problem for finite labeled Place/Transition Petri nets without capacity bounds on places. We single out a fragment (called *finite-net processes*) of an extension of CCS (called multi-CCS, fully described in [19]), such that not only all processes of this fragment generate finite P/T nets, but also for any finite (statically reduced) P/T net we can find a term of the calculus that generates it. This solves the open problem of providing a process calculus representing finite P/T Petri nets, and opens interesting possibilities of cross-fertilization between the areas of Petri nets and process calculi. In particular, it is now possible, on the one hand, (i) to define any (statically reduced) finite P/T net compositionally and (ii) to study algebraic laws for net-based behavioral equivalences (such as net isomorphism) over such a class of systems; on the other hand, it is now possible (iii) to reuse all the techniques and decidability results available for finite P/T nets [12] also for this fragment of multi-CCS, as well as (iv) to continue the study of non-interleaving semantics, typical of Petri nets (e.g., [11,27,29]), also for process algebras (initiated in [8,28]), in particular for finite-net multi-CCS.

Finite-net multi-CCS includes the operator $\underline{\alpha}.s$ of *strong prefixing* (in contrast to normal prefixing $\mu.t$), which states that the visible action α is the initial part of an atomic sequence that continues with the sequential process s. So, by strong prefixing, a transition can be labeled with a sequence of visible actions. This operator, introduced in [16,17] with a slightly different semantics, is also at the base of multi-party synchronization, obtained as an atomic sequence of binary CCS-like synchronizations. In finite-net multi-CCS, parallel composition may occur inside the body of a recursively

✉ Roberto Gorrieri
roberto.gorrieri@unibo.it

[1] Dipartimento di Informatica, Scienza e Ingegneria, Università di Bologna, Mura A. Zamboni, 7, 40127 Bologna, Italy

defined constant C; on the contrary, the restriction operator (νa) is not allowed in the body of C. So, a finite-net process may be represented as $(\nu L)t$, where L is a set of actions (if L is empty, the restriction operator is not present) and t a restriction-free process.

We equip our calculus with a net semantics that, differently from the approach by Degano et al. [8–10,28], uses *unsafe* P/T nets, as done in [13,15] for a CCS sub-calculus without restriction, and in [5] for the π-calculus, where however inhibitor arcs are used to model restriction. The extension of the approach to restriction and strong prefixing is not trivial and passes through the introduction of an auxiliary set of *restricted* actions i.e., actions which are only allowed to synchronize. We prove that the net semantics associates a P/T net $\mathrm{Net}(p)$ to any finite-net multi-CCS process p, such that $\mathrm{Net}(p)$ has finitely many places; if p is *well-formed*, then $\mathrm{Net}(p)$ has also finitely many transitions; intuitively, process p is well-formed if the sequences that p may generate via strong prefixing have never the possibility to synchronize. We also provide a soundness result, i.e., p and $\mathrm{Net}(p)$ are bisimilar [25]. Finally, we also prove the *representability theorem*: for any finite, statically reduced, P/T net N, we can find a well-formed, finite-net multi-CCS process p_N such that $\mathrm{Net}(p_N)$ and N are isomorphic.

The paper is organized as follows. Section 2 contains some basic background on LTSs and Petri nets. Section 3 introduces the process calculus called *finite-net multi-CCS*, its interleaving semantics in terms of LTSs, the well-formedness condition on its processes, and also the concurrent readers/writers example. Section 4 defines the operational net semantics for the calculus and presents the finiteness theorem (for any well-formed process p, $\mathrm{Net}(p)$ is finite) and some examples of net construction. Section 5 provides the soundness theorem (p and $\mathrm{Net}(p)$ are bisimilar). Section 6 proves the language expressibility theorem, i.e., the representability theorem mentioned above. Finally, some conclusions are drawn in Sect. 7, together with a comparison with related literature, in particular with the earlier version [18] of this paper. This paper is the full version of the extended abstract [14].

2 Background

2.1 Labeled transition systems and bisimulation

Definition 1 A *labeled transition system* (or *LTS* for short) is a triple $TS = (Q, A, \rightarrow)$ where

- Q is the countable set of states,
- A is the countable set of labels,
- $\rightarrow \subseteq Q \times A \times Q$ is the transition relation.

In the following $q \xrightarrow{a} q'$ denotes $(q, a, q') \in \rightarrow$. A *rooted* LTS is a pair (TS, q_0) where $\mathrm{TS} = (Q, A, \rightarrow)$ is a transition system and $q_0 \in Q$ is the *initial state*; a rooted LTS is usually represented as $\mathrm{TS} = (Q, A, \rightarrow, q_0)$. A *path* from q_1 to q_{n+1} is a sequence of transitions $q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} \ldots q_n \xrightarrow{a_n} q_{n+1}$. We say that q' is *reachable* from q if there exists a path from q to q'. A rooted LTS $(Q, A, \rightarrow, q_0)$ is *reduced* if all the states in Q are reachable from the initial state q_0.

Definition 2 Given two LTSs $\mathrm{TS}_1 = (Q_1, A, \rightarrow_1)$ and $\mathrm{TS}_2 = (Q_2, A, \rightarrow_2)$ a *bisimulation* between TS_1 and TS_2 is a relation $R \subseteq (Q_1 \times Q_2)$ such that if $(q_1, q_2) \in R$ then for all $a \in A$

- $\forall q_1'$ such that $q_1 \xrightarrow{a}_1 q_1'$, $\exists\, q_2'$ such that $q_2 \xrightarrow{a}_2 q_2'$ and $(q_1', q_2') \in R$
- $\forall q_2'$ such that $q_2 \xrightarrow{a}_2 q_2'$, $\exists\, q_1'$ such that $q_1 \xrightarrow{a}_1 q_1'$ and $(q_1', q_2') \in R$.

If $\mathrm{TS}_1 = \mathrm{TS}_2$ we say that R is a bisimulation on TS_1. Two states q and q' are bisimilar, $q \sim q'$, if there exists a bisimulation R such that $(q, q') \in R$.

2.2 Finite place/transition Petri nets

We recall some basic notions on finite P/T Petri nets (see, e.g., [7,30–32] for an introduction). We use here a non-standard notation that better suits our needs.

Definition 3 (*Multisets*) Let $\mathbb{N}$ be the set of natural numbers. Given a set S, a *finite multiset* over S is a function $m : S \rightarrow \mathbb{N}$ such that $\mathrm{dom}(m) = \{s \in S \mid m(s) \neq 0\}$ is finite. The set of all finite multisets over S, $\mathcal{M}_{\mathrm{fin}}(S)$, is ranged over by m. A multiset m such that $\mathrm{dom}(m) = \emptyset$ is called *empty* and is denoted with $\emptyset$, with abuse of notation. We write $m \subseteq m'$ if $m(s) \leq m'(s)$ for all $s \in S$. The operator $\oplus$ denotes *multiset union*: $(m \oplus m')(s) = m(s) + m'(s)$. The operator $\ominus$ denotes *multiset difference*: if $m' \subseteq m$, then $(m \ominus m')(s) = m(s) - m'(s)$. The *scalar product* of a natural j with m is $(j \cdot m)(s) = j \cdot (m(s))$. A finite multiset m over a finite set $S = \{s_1, \ldots, s_n\}$ can be represented also as $k_1 \cdot s_1 \oplus k_2 \cdot s_2 \oplus \cdots \oplus k_n \cdot s_n$, where $k_j = m(s_j) \geq 0$ for $j = 1, \ldots, n$.

Definition 4 (*Finite P/T Petri nets*) A labeled *finite Place/Transition Petri net* is a tuple $N = (S, A, T)$, where

- S is the finite set of *places*, ranged over by s (possibly indexed),
- A is the finite set of *labels*, ranged over by a (possibly indexed), and
- $T \subseteq (\mathcal{M}_{\mathrm{fin}}(S) \backslash \emptyset) \times A \times \mathcal{M}_{\mathrm{fin}}(S)$ is the finite set of *transitions*, ranged over by t (possibly indexed), such that $\forall a \in A\, \exists t \in T$ with $l(t) = a$.

A finite multiset over the set S of places is called a *marking*. Given a marking m and a place s, we say that the place s contains $m(s)$ *tokens*. Given a transition $t = (m, a, m')$, we use the notation ${}^\bullet t$ to denote its *pre-set* m (which cannot be an empty marking), $t^\bullet$ for its *post-set* m' and $l(t)$ for its label a. Hence, transition t can be also represented as ${}^\bullet t \xrightarrow{l(t)} t^\bullet$. A *P/T system* is a tuple $N(m_0) = (S, A, T, m_0)$, where (S, A, T) is a P/T net and m_0 is a finite multiset over S, called the *initial marking*.

Our definition of T as a set of triples ensures that the net is *transition simple*: for any $t_1, t_2 \in T$, if ${}^\bullet t_1 = {}^\bullet t_2$ and $t_1^\bullet = t_2^\bullet$ and $l(t_1) = l(t_2)$, then $t_1 = t_2$. We are also assuming that a transition has a nonempty pre-set. These are the only constraints we impose over the definition of P/T net. The additional condition that A is covered by T (i.e., $\forall a \in A\, \exists t \in T$ with label a) is just for economy.

Definition 5 (*Net isomorphism*) Two P/T nets $N_1 = (S_1, A, T_1)$ and $N_2 = (S_2, A, T_2)$ are *isomorphic* if there exists a bijection $f : S_1 \to S_2$, homomorphically extended to markings, such that $(m, a, m') \in T_1$ iff $(f(m), a, f(m')) \in T_2$. Two systems $N_1(m_1)$ and $N_2(m_2)$ are *isomorphic* if N_1 and N_2 are isomorphic by f, which, additionally, preserves the initial markings: $f(m_1) = m_2$.

Definition 6 Given a labeled P/T net $N = (S, A, T)$, we say that a transition t is *enabled* at marking m, written as $m[t\rangle$, if ${}^\bullet t \subseteq m$. The execution of t enabled at m produces the marking $m' = (m \ominus {}^\bullet t) \oplus t^\bullet$, denoted by $m[t\rangle m'$. The set of markings *reachable* from m, denoted by $[m\rangle$, is defined as the least set such that

- $m \in [m\rangle$ and
- if $m_1 \in [m\rangle$ and, for some transition $t \in T$, $m_1[t\rangle m_2$, then $m_2 \in [m\rangle$.

Given a P/T system $N(m_0) = (S, A, T, m_0)$, we say that m is *reachable* if m is reachable from the initial marking m_0. A P/T system $N(m_0) = (S, A, T, m_0)$ is said *safe* if for all $m \in [m_0\rangle$ and for all $s \in S$ we have that $m(s) \leq 1$.

Definition 7 Given a P/T system $N(m_0) = (S, A, T, m_0)$, the *interleaving marking graph* of $N(m_0)$ is the rooted LTS $\mathrm{IMG(N(m_0))} = ([m_0\rangle, A, \to, m_0)$, where m_0 is the initial state and the transition relation $\to \subseteq \mathcal{M}_{\mathrm{fin}}(S) \times A \times \mathcal{M}_{\mathrm{fin}}(S)$ is defined by $m \xrightarrow{a} m'$ if and only if there exists a transition $t \in T$ such that $m[t\rangle m'$ and $l(t) = a$. The P/T systems $N_1(m_1)$ and $N_2(m_2)$ are *interleaving bisimilar*—denoted by $N_1(m_1) \sim N_2(m_2)$—if and only if there exists a bisimulation $R \subseteq [m_1\rangle \times [m_2\rangle$ such that $(m_1, m_2) \in R$.

Definition 8 (*Dynamically reduced*) A P/T system $N(m_0) = (S, A, T, m_0)$ is dynamically reduced if

- $\forall s \in S\, \exists m \in [m_0\rangle$ such that $m(s) \geq 1$, and
- $\forall t \in T\, \exists m, m' \in [m_0\rangle$ such that $m[t\rangle m'$.

Definition 9 (*Statically reduced*) Given a finite P/T net $N = (S, A, T)$, we say that a transition t is *statically enabled* by a set of places $S' \subseteq S$, denoted by $S'[\![t\rangle$, if $\mathrm{dom}({}^\bullet t) \subseteq S'$. Given two sets of places $S_1, S_2 \subseteq S$, we say that S_2 is *statically reachable in one step* from S_1 if there exists $t \in T$, such that $S_1[\![t\rangle$, $\mathrm{dom}(t^\bullet) \not\subseteq S_1$ and $S_2 = S_1 \cup \mathrm{dom}(t^\bullet)$; this is denoted by $S_1 \stackrel{t}{\Longrightarrow} S_2$. The *static reachability relation* $\stackrel{*}{\Longrightarrow} \subseteq \wp(S)_{\mathrm{fin}} \times \wp(S)_{\mathrm{fin}}$ is the least relation such that

- $S_1 \stackrel{*}{\Longrightarrow} S_1$ and
- if $S_1 \stackrel{*}{\Longrightarrow} S_2$ and $S_2 \stackrel{t}{\Longrightarrow} S_3$, then $S_1 \stackrel{*}{\Longrightarrow} S_3$.

A set of places $S_k \subseteq S$ is the *largest* set statically reachable from S_1 if $S_1 \stackrel{*}{\Longrightarrow} S_k$ and for all $t \in T$ such that $S_k[\![t\rangle$, we have that $\mathrm{dom}(t^\bullet) \subseteq S_k$.

Given a finite P/T system $N(m_0) = (S, A, T, m_0)$, we denote by $[\![\mathrm{dom}(m_0)\rangle$ the largest set of places statically reachable from $\mathrm{dom}(m_0)$, i.e., the largest S_k such that $\mathrm{dom}(m_0) \stackrel{*}{\Longrightarrow} S_k$. A finite P/T net system $N(m_0) = (S, A, T, m_0)$ is *statically reduced* if all the places are statically reachable from the places in the initial marking, i.e., if $[\![\mathrm{dom}(m_0)\rangle = S$.

Note that if $N(m_0) = (S, A, T, m_0)$ is statically reduced, then all the transitions in T are statically enabled by S. Note also that if $N(m_0) = (S, A, T, m_0)$ is dynamically reduced, then it is also statically reduced. However, there are statically reduced P/T systems that are not dynamically reduced. For instance, the statically reduced P/T system $N(s_1) = (\{s_1, s_2, s_3\}, \{a, b\}, \{(s_1, a, s_2), (2 \cdot s_1, b, s_3)\}, s_1)$ cannot reach dynamically place s_3.

3 Finite-net multi-CCS

Now we present finite-net multi-CCS: first its syntax, then the LTS operational semantics, followed by the definition of well-formedness; finally, an example.

3.1 Syntax

Let $\mathcal{L}$ be a denumerable set of names (inputs), ranged over by $a, b, \ldots$. Let $\overline{\mathcal{L}}$ be the set of co-names (outputs), ranged over by $\overline{a}, \overline{b}, \ldots$. The set $\mathcal{L} \cup \overline{\mathcal{L}}$, ranged over by $\alpha, \beta, \ldots$, is the set of visible actions. With $\overline{\alpha}$ we mean the complement of α, assuming that $\overline{\overline{\alpha}} = \alpha$. Let $\mathrm{Act} = \mathcal{L} \cup \overline{\mathcal{L}} \cup \{\tau\}$, such that $\tau \notin \mathcal{L} \cup \overline{\mathcal{L}}$, be the set of actions, ranged over by μ. Action τ denotes an invisible, internal activity. Let $\mathcal{C}$ be a denumerable set of process constants, disjoint from Act, ranged over by $A, B, C, \ldots$. The process terms are generated by the following abstract syntax

$$s ::= \mathbf{0} \mid \mu.t \mid \underline{\alpha}.s \mid s + s$$
$$t ::= s \mid t \mid t \mid C$$
$$p ::= t \mid (va)p,$$

where we are using three syntactic categories: s, to range over sequential processes (i.e., processes that start sequentially), t, to range over restriction-free processes, and, finally p, to range over-restricted processes.

As for CCS [25], term $\mathbf{0}$ is the terminated process, $\mu.t$ is a normally prefixed process where action μ is first performed and then t is ready. Note that $s + s'$ is the sequential process obtained by the alternative composition of *sequential* processes s and s'; hence we are restricting the use of $+$ to so-called *guarded sum*. Term $t \mid t'$ is the parallel composition of t and t'. $(va)p$ is process p where the name a is made private by applying the restriction operator over a. Finally, C is a process constant, equipped with a defining equation $C \stackrel{\text{def}}{=} t$, i.e., the body of a constant must be a restriction-free process. The only new operator of the calculus is *strong prefixing*: $\underline{\alpha}.s$ is a strongly prefixed process, where α is the first action of a transaction that continues with the *sequential* process s (provided that s can complete the transaction).

We sometimes use the syntactic convention of writing $(va)((vb)p))$ as $(va, b)p$. Generalizing this convention, a finite-net multi-CCS process may be represented as $(\nu L)t$, where L is a set of actions (if L is empty, the restriction operator is not present) and t is a restriction-free process.

The set $\mathcal{P}$ of *processes* contains those terms which use *finitely many* constants only and are, w.r.t. the constants they use, *closed* (all possess a defining equation) and *guarded* (for any defining equation $C \stackrel{\text{def}}{=} t$, any occurrence of a constant in t is within a *normally prefixed* subprocess $\mu.t'$ of t). $\mathcal{P}_{\text{seq}}$ is the set of *sequential processes*, i.e., those of syntactic category s. Note that the restriction operator cannot occur syntactically in any term of $\mathcal{P}_{\text{seq}}$. With abuse of notation, $\mathcal{P}$ will be ranged over by $p, q, r, \ldots$ (hence p may denote any kind of process terms, also sequential ones), possibly indexed.

Definition 10 For any finite-net multi-CCS process p, the set of its sequential subterms $\text{sub}(p)$ is defined by means of the auxiliary function (with the same name, with abuse of notation) $\text{sub}(p, \emptyset)$, whose second parameter is a set of already known constants, initially empty, described in Table 1.

Proposition 1 *For any finite-net multi-CCS process p, the set of its sequential subterms $\text{sub}(p)$ is finite.*

Proof By induction on the definition of $\text{sub}(p, \emptyset)$. The base cases are $\text{sub}(\mathbf{0}, I)$ and $\text{sub}(A, I)$ when $A \in I$. Note that induction will end eventually because the constants that a finite-net multi-CCS process may use are finitely many. □

Table 1 Sequential subterms of a process

$$\text{sub}(\mathbf{0}, I) = \{\mathbf{0}\} \qquad \text{sub}(\mu.p, I) = \{\mu.p\} \cup \text{sub}(p, I)$$
$$\text{sub}((va)p, I) = \text{sub}(p, I) \qquad \text{sub}(\underline{\alpha}.p, I) = \{\underline{\alpha}.p\} \cup \text{sub}(p, I)$$
$$\text{sub}(p_1 + p_2, I) = \{p_1 + p_2\} \cup \text{sub}(p_1, I) \cup \text{sub}(p_2, I)$$
$$\text{sub}(p_1 \mid p_2, I) = \text{sub}(p_1, I) \cup \text{sub}(p_2, I)$$
$$\text{sub}(A, I) = \begin{cases} \emptyset & A \in I, \\ \text{sub}(p, I \cup \{A\}) & A \notin I \wedge A \stackrel{\text{def}}{=} p \end{cases}$$

Table 2 Operational rules [symmetric rule (Sum_2) omitted]

$$\text{(Pref)}\ \frac{}{\mu.p \xrightarrow{\mu} p} \qquad \text{(Cong)}\ \frac{p \equiv p' \xrightarrow{\sigma} q' \equiv q}{p \xrightarrow{\sigma} q} \qquad \text{(Sum}_1\text{)}\ \frac{p \xrightarrow{\sigma} p'}{p + q \xrightarrow{\sigma} p'}$$

$$\text{(Par)}\ \frac{p \xrightarrow{\sigma} p'}{p \mid q \xrightarrow{\sigma} p' \mid q} \qquad \text{(S-Pref)}\ \frac{p \xrightarrow{\sigma} p'}{\underline{\alpha}.p \xrightarrow{\alpha \diamond \sigma} p'} \quad \alpha \diamond \sigma = \begin{cases} \alpha & \text{if } \sigma = \tau, \\ \alpha\sigma & \text{otherwise} \end{cases}$$

$$\text{(S-Res)}\ \frac{p \xrightarrow{\sigma} p'}{(va)p \xrightarrow{\sigma} (va)p'}\ a, \overline{a} \notin n(\sigma)$$

$$\text{(S-Com)}\ \frac{p \xrightarrow{\sigma_1} p' \quad q \xrightarrow{\sigma_2} q'}{p \mid q \xrightarrow{\sigma} p' \mid q'}\ \text{Sync}(\sigma_1, \sigma_2, \sigma)$$

Remark 1 Note that for any processes p and q, if the sequential process p is such that $p \in \text{sub}(q)$, then $\text{sub}(p) \subseteq \text{sub}(q)$. This is because the definition of $\text{sub}(q)$ recursively calls itself on all of its sequential subterms.

3.2 Operational semantics with LTSs

The operational semantics for finite-net multi-CCS is given by the labeled transition system $(\mathcal{P}, \mathcal{A}, \longrightarrow)$, where the states are the processes in $\mathcal{P}$, $\mathcal{A} = \{\tau\} \cup (\mathcal{L} \cup \overline{\mathcal{L}})^+$ is the set of labels (ranged over by σ and composed of the invisible action τ and by sequences of visible actions), and $\longrightarrow \subseteq \mathcal{P} \times \mathcal{A} \times \mathcal{P}$ is the minimal transition relation generated by the rules listed in Table 2.

We briefly comment on the rules that are less standard. Rule (S-pref) allows for the creation of transitions labeled by nonempty sequences of actions. In order for $\underline{\alpha}.p$ to make a move, it is necessary that p be able to perform a transition, i.e., the rest of the transaction. Hence, if $p \xrightarrow{\sigma} p'$ then $\underline{\alpha}.p \xrightarrow{\alpha \diamond \sigma} p'$, where the label $\alpha \diamond \sigma = \alpha$ if $\sigma = \tau$, $\alpha \diamond \sigma = \alpha\sigma$ otherwise. Note that $\underline{\alpha}.\mathbf{0}$ cannot execute any action, as $\mathbf{0}$ is terminated. If a transition is labeled with $\sigma = \alpha_1 \ldots \alpha_{n-1}\alpha_n$, then all the actions $\alpha_1 \ldots \alpha_{n-1}$ are due to strong prefixes, while α_n is due to a normal prefix (or α_n is a strong prefix followed by a normal prefix τ). Rule (S-Com) has a side-condition on the possible synchronizability of σ_1 and σ_2. Relation $\text{Sync}(\sigma_1, \sigma_2, \sigma)$, defined by the axioms of Table 3, holds if at least one of the two sequences is a single action, say $\sigma_1 = \overline{\alpha}$, and the other starts with the complementary action

Table 3 Synchronization relation Sync

$$\frac{}{\mathrm{Sync}(\alpha, \overline{\alpha}, \tau)} \qquad \frac{\sigma \neq \epsilon}{\mathrm{Sync}(\alpha\sigma, \overline{\alpha}, \sigma)} \qquad \frac{\sigma \neq \epsilon}{\mathrm{Sync}(\overline{\alpha}, \alpha\sigma, \sigma)}$$

α. Note that it is not possible to synchronize two sequences. This means that, usually, a multi-party synchronization can take place only among one *leader*, i.e., the process performing the atomic sequence, and as many other components (the *servants*), as is the length of the atomic sequence, where each servant executes one visible action. This is strictly the case for so-called *well-formed processes*, i.e., processes that do not allow for the synchronization of two sequences, not even indirectly. However, more elaborate forms of synchronization are possible, as illustrated in Sect. 3.3, for non-well-formed processes. Rule (S-Res) is slightly more general than the corresponding one for CCS, as it requires that no action in σ can be a or $\overline{a}$. With $n(\sigma)$ we denote the set of all actions occurring in σ. Formally: $n(\mu) = \{\mu\}$, $n(\alpha\sigma) = \{\alpha\} \cup n(\sigma)$.

There is one further rule, called (Cong), which makes use of the structural congruence $\equiv$, induced by the three axioms in Table 4. Axioms **E1** and **E2** are for associativity and commutativity, respectively, of the parallel operator. Axiom **E3** is for unfolding and explains why we have no explicit operational rule for handling constants in Table 2: the transitions derivable from C are those transitions derivable from the structurally congruent term p if $C \stackrel{\text{def}}{=} p$. Rule (Cong) enlarges the set of transitions derivable from a given process p, as the following example shows. The intuition is that, given a process p, a transition is derivable from p if it is derivable from any p' obtained as a rearrangement in any order (or association) of all of its sequential subprocesses.

Example 1 (*Associativity and commutativity*) Consider $(\underline{a}.b.p \mid \overline{a}.q) \mid \overline{b}.r$. The ternary synchronization among them, $(\underline{a}.b.p \mid \overline{a}.q) \mid \overline{b}.r \xrightarrow{\tau} (p \mid q) \mid r$, can take place, as proved in Table 5, without using rule (Cong). However, if we consider

Table 4 Axioms generating the structural congruence $\equiv$

E1	$(p \mid q) \mid r = p \mid (q \mid r)$	
E2	$p \mid q = q \mid p$	
E3	$A = q$	if $A \stackrel{\text{def}}{=} q$

Table 5 Multi-party synchronization among three processes

$$\dfrac{\dfrac{\dfrac{}{b.p \xrightarrow{b} p}}{\underline{a}.b.p \xrightarrow{ab} p} \qquad \dfrac{}{\overline{a}.q \xrightarrow{\overline{a}} q}}{\underline{a}.b.p \mid \overline{a}.q \xrightarrow{b} p \mid q} \qquad \dfrac{}{\overline{b}.r \xrightarrow{\overline{b}} r}$$
$$(\underline{a}.b.p \mid \overline{a}.q) \mid \overline{b}.r \xrightarrow{\tau} (p \mid q) \mid r$$

the very similar process $\underline{a}.b.p \mid (\overline{a}.q \mid \overline{b}.r)$, then we can see that $\underline{a}.b.p$ is able to synchronize with both $\overline{a}.q$ and $\overline{b}.r$ only by using rule (Cong), as follows:

$$\frac{\underline{a}.b.p \mid (\overline{a}.q \mid \overline{b}.r) \equiv (\underline{a}.b.p \mid \overline{a}.q) \mid \overline{b}.r \xrightarrow{\tau} (p \mid q) \mid r \equiv p \mid (q \mid r)}{\underline{a}.b.p \mid (\overline{a}.q \mid \overline{b}.r) \xrightarrow{\tau} p \mid (q \mid r)}$$

If we consider the slightly different variant process $(\underline{a}.b.p \mid \overline{b}.r) \mid \overline{a}.q$, we see easily that, without rule (Cong), no ternary synchronization is possible, because Sync(ab, $\overline{\text{b}}$, a) does not hold. This example shows that, by using the axioms **E1** and **E2**, it is possible to reorder the servant subcomponents (in this example, subprocesses $\overline{a}.q$ and $\overline{b}.r$) in such a way that the actions they offer are in the expected order by the leader process (in this example, $\underline{a}.b.p$).

Two processes p and q are *bisimilar*, $p \sim q$, if there exists a bisimulation $R \subseteq \mathcal{P} \times \mathcal{P}$ such that $(p, q) \in R$. The following obvious result holds.

Proposition 2 *If $p \equiv q$ then $p \sim q$.*

3.3 Well-formed processes

We propose a syntactic condition on a process p, ensuring that, during its execution, p is unable to synchronize two atomic sequences, not even indirectly; a process satisfying such a syntactic condition will be called *well-formed*. The restriction to well-formed processes will be crucial in the following sections.

The definition of relation $\mathrm{Sync}(\sigma_1, \sigma_2, \sigma)$ requires that at least one of σ_1 or σ_2 be a single action; this is not enough to prevent that two sequences may synchronize, even if indirectly. For instance, assume we have three processes $p_1 = \underline{a}.b.\mathbf{0}$, $p_2 = \overline{a}.\mathbf{0}$ and $p_3 = \underline{\overline{b}}.c.\mathbf{0}$, which may perform the sequences ab, $\overline{a}$, $\overline{b}c$, respectively; then a ternary synchronization is possible, because first we synchronize p_1 and p_2, by Sync(ab, $\overline{\text{a}}$, b), getting a single action b, which can be then used for a synchronization with p_3, by Sync(b, $\overline{\text{bc}}$, c); in such a way, the two atomic sequences ab and $\overline{b}c$ have been synchronized, by means of the single action $\overline{a}$. So, we would like to mark $(p_1 \mid p_2) \mid p_3$ as not well-formed. In order to define well-formed multi-CCS processes, some auxiliary definitions are needed.

Definition 11 (*Initials for sequential processes*) For any sequential process p, $In(p) \subseteq \mathcal{A}$ is the set of *initials* of p, defined inductively as

$$\mathrm{In}(\mathbf{0}) = \emptyset \qquad \mathrm{In}(\mu.p) = \{\mu\}$$
$$\mathrm{In}(\underline{\alpha}.p) = \alpha \diamond \mathrm{In}(p) \qquad \mathrm{In}(p_1 + p_2) = \mathrm{In}(p_1) \cup \mathrm{In}(p_2)$$

where $\alpha \diamond \mathrm{In}(p) = \{\alpha \diamond \sigma \mid \sigma \in \mathrm{In}(p)\}$.

Table 6 Names in sequences of a process

$$\mathrm{ns}(\mathbf{0}, I) = \emptyset \qquad \mathrm{ns}(p_1 + p_2, I) = \mathrm{ns}(p_1, I) \cup \mathrm{ns}(p_2, I)$$
$$\mathrm{ns}(\mu.p, I) = \mathrm{ns}(p, I) \qquad \mathrm{ns}(p_1 \mid p_2, I) = \mathrm{ns}(p_1, I) \cup \mathrm{ns}(p_2, I)$$
$$\mathrm{ns}(\underline{\alpha}.p, I) = \mathrm{ns}(p, I) \cup \{\alpha\} \cup \bigcup_{\sigma \in \mathrm{In}(p) \wedge \sigma \neq \tau} n(\sigma)$$
$$\mathrm{ns}((\nu a)p, I) = \mathrm{ns}(p, I) \backslash \{a, \overline{a}\}$$
$$\mathrm{ns}(A, I) = \begin{cases} \emptyset & A \in I, \\ \mathrm{ns}(p, I \cup \{A\}) & A \notin I \wedge A \stackrel{\mathrm{def}}{=} p \end{cases}$$

Table 7 Well-formedness predicate

$$\frac{}{\mathrm{wf}(\mathbf{0}, I)} \qquad \frac{\mathrm{wf}(p, I)}{\mathrm{wf}(\mu.p, I)} \qquad \frac{\mathrm{wf}(p, I) \quad \nexists \beta.\beta \in \mathrm{ns}(\underline{\alpha}.p, I) \wedge \overline{\beta} \in \mathrm{ns}(\underline{\alpha}.p, I)}{\mathrm{wf}(\underline{\alpha}.p, I)}$$

$$\frac{\mathrm{wf}(p, I)}{\mathrm{wf}((\nu a)p, I)} \qquad \frac{A \in I}{\mathrm{wf}(A, I)} \qquad \frac{\mathrm{wf}(p, I \cup \{A\}) \quad A \stackrel{\mathrm{def}}{=} p \quad A \notin I}{\mathrm{wf}(A, I)}$$

$$\frac{\mathrm{wf}(p_1, I) \quad \mathrm{wf}(p_2, I) \quad \nexists \beta.\beta \in \mathrm{ns}(p_1, I) \wedge \overline{\beta} \in \mathrm{ns}(p_2, I)}{\mathrm{wf}(p_1 \mid p_2, I) \quad \mathrm{wf}(p_1 + p_2, I)}$$

Definition 12 (*Names in sequences of a process*) Let $\mathrm{ns}(p) \subseteq \mathcal{L} \cup \overline{\mathcal{L}}$ be the set of (free) names occurring in sequences *of length two or more* of p. Set $\mathrm{ns}(p)$ is defined by means of the auxiliary function (with the same name, with abuse of notation) $\mathrm{ns}(p, \emptyset)$, whose second parameter is a set of constants, where $\mathrm{ns}(p, I)$ is defined in Table 6.

A process p is well-formed if $\mathrm{wf}(p)$ holds, and $\mathrm{wf}(p)$ holds if the auxiliary relation (with the same name, with abuse of notation) $\mathrm{wf}(p, \emptyset)$ holds; the auxiliary relation $\mathrm{wf}(p, I)$, where the second parameter is a set of constants, is defined as the least relation induced by the axioms and rules of Table 7. The assumption that any process uses finitely many constants ensures that the well-formedness predicate is well-defined.

Example 2 Let us consider processes $p_1 = \underline{a}.b.\mathbf{0}$, $p_2 = \overline{a}.\mathbf{0}$ and $p_3 = \overline{b}.c.\mathbf{0}$. Note that $\mathrm{wf}(p_2)$, because $\mathrm{wf}(\mathbf{0})$ holds; similarly, $\mathrm{wf}(b.\mathbf{0})$; as a consequence, $\mathrm{wf}(p_1)$ holds, because $\mathrm{ns}(p_1) = \{a, b\}$ does not contain a pair of complementary actions. In the same way, we can prove that $\mathrm{wf}(p_3)$ holds, with $\mathrm{ns}(p_3) = \{\overline{b}, c\}$. We also have that $\mathrm{wf}(p_1 \mid p_2)$, as no action of $\mathrm{ns}(p_1)$ occurs complemented in $\mathrm{ns}(p_2) = \emptyset$. However, it is not the case that $\mathrm{wf}((p_1 \mid p_2) \mid p_3)$, because there exists an action, namely b, such that $b \in \mathrm{ns}(p_1 \mid p_2)$ and $\overline{b} \in \mathrm{ns}(p_3)$.

To conclude this section, we comment on the well-formedness relation: if a process is well-formed, then it is not possible to synchronize two sequences, not even indirectly. Theorem 2 will prove this fact on the net semantics. A direct proof on the LTS semantics is reported in [19].

3.4 An example: concurrent readers and writers

In this problem, originally introduced in [6], there are two types of processes: reader processes and writer processes. All processes share a common file; so, each writer process must exclude all the other writers and all the readers while writing on the file, while multiple reader processes can access the shared file simultaneously. The problem is to define a control structure that does not deadlock or allow violations of the mutual exclusion criteria.

Assume we have n readers and m writers and that at most $k \leq n$ readers can read simultaneously. We can assume we have k lock resources such that a reader can read if at least one lock is available, while a writer can write if all the k locks are available, so that it prevents all the k possible concurrent reading operations. In a naïve CCS solution to this problem, a deadlock may occur when two writers are competing for the acquisition of the k locks, so that one has acquired i locks and the other one $k - i$, for some $0 < i < k$; in such a situation, both writers are stuck, waiting for the missing locks, and all the readers are not allowed to read as no lock is available. A simple multi-ccS solution to this coordination problem is forcing atomicity on the writer's acquisition of the k locks, so that either all or none are taken. To make the presentation simple, assume that $n = 4$, $k = 3$, $m = 2$. Each reader process R, each lock process L, each writer W can be represented as follows, where action l stands for lock and u for unlock:

$$R \stackrel{\mathrm{def}}{=} l.\mathrm{read}.u.R \quad L \stackrel{\mathrm{def}}{=} \overline{l}.\overline{u}.L \quad W \stackrel{\mathrm{def}}{=} \underline{l}.\underline{l}.l.\mathrm{write}.\underline{u}.\underline{u}.u.W$$

The whole system CRW is defined as

$$\mathrm{CRW} \stackrel{\mathrm{def}}{=} (\nu l, u)(R \mid R \mid R \mid W \mid W \mid L \mid L \mid L),$$

where parentheses are omitted as $\mid$ is associative. Note that a writer W executes a four-way synchronization with the three instances of the lock process L in order to get permission to write:

$$\mathrm{CRW} \xrightarrow{\tau} (\nu l, u)(R \mid R \mid R \mid W' \mid W \mid L' \mid L' \mid L'),$$

where $W' = \mathrm{write}.\underline{u}.\underline{u}.u.W$ and $L' = \overline{u}.L$. The LTS for CRW is finite-state. Note that, to ensure correctness, it is not necessary to require atomicity on the release of the locks: This choice is only done in order to have a smaller model.

4 Operational net semantics

4.1 Places and markings

The finite-net multi-CCS processes are built upon the set $\mathcal{L} \cup \overline{\mathcal{L}}$, ranged over by α, of visible actions. We assume we have also sets $\mathcal{L}' = \{a' \mid a \in \mathcal{L}\}$ and $\overline{\mathcal{L}'} = \{\overline{a'} \mid \overline{a} \in \overline{\mathcal{L}}\}$, where $\mathcal{L}' \cup \overline{\mathcal{L}'}$, ranged over by α', is the set of auxiliary *restricted* actions; by definition, each restricted action α' corresponds

Table 8 Decomposition function

$\mathrm{dec}(\mathbf{0}, I) = \emptyset$	$\mathrm{dec}(\mu.p, I) = \{\mu.p\}$
$\mathrm{dec}(\underline{\gamma}.p, I) = \{\underline{\gamma}.p\}$	$\mathrm{dec}(p + p', I) = \{p + p'\}$
$\mathrm{dec}(p \mid p', I) = \mathrm{dec}(p, I) \oplus \mathrm{dec}(p', I)$	$\mathrm{dec}(A, I) = \begin{cases} \emptyset & A \in I, \\ \mathrm{dec}(p, I \cup \{A\}) & A \notin I \wedge A \stackrel{\mathrm{def}}{=} p \end{cases}$
$\mathrm{dec}((\nu a)p, I) = \mathrm{dec}(p, I)\{a'/a\}$	$a' \in \mathcal{L}'$ is the restricted action corresponding to a

exactly to one visible action α. Set $\mathcal{G} = \mathcal{L} \cup \overline{\mathcal{L}} \cup \mathcal{L}' \cup \overline{\mathcal{L}'}$ is ranged over by γ. The set of all actions $\mathrm{Act}_\gamma = \mathcal{G} \cup \{\tau\}$, ranged over by μ (with abuse of notation), is used to build the set of *extended*, finite-net multi-CCS processes $\mathcal{P}^\gamma$. The infinite set of places, ranged over by s, is $S_{\mathrm{MCCS}} = \mathcal{P}^\gamma_{\mathrm{seq}} \backslash \{\mathbf{0}\}$, i.e., the set of all sequential processes (except $\mathbf{0}$) whose prefixes are in Act_γ and whose strong prefixes are in $\mathcal{G}$.

Function $\mathrm{dec} : \mathcal{P}^\gamma \times \wp(\mathcal{C}) \to \mathcal{M}_{\mathrm{fin}}(S_{\mathrm{MCCS}})$ defines the decomposition of extended processes into markings (see Table 8), where the second argument is the set of already known constants, initially empty. For simplicity sake, we often omit the second argument when it is empty or inessential. Process $\mathbf{0}$ generates no places. The decomposition of a sequential process p produces one place with name p. This is the case of $\mu.p$ (where μ can be any action in Act_γ), $\underline{\gamma}.p$ and $p + p'$. Parallel composition is interpreted as multiset union; e.g., the decomposition of $a.\mathbf{0} \mid a.\mathbf{0}$ produces the marking $a.\mathbf{0} \oplus a.\mathbf{0} = 2 \cdot a.\mathbf{0}$. The decomposition of a restricted process $(\nu a)p$—where $a \in \mathcal{L}$—generates the multiset obtained from the decomposition of p, to which the substitution $\{a'/a\}$ is applied; the application of the substitution $\{a'/a\}$ to a multiset is performed elementwise, as shown in the example below. Finally, a process constant A is first unwound once (according to its defining equation) and then decomposed, if A is not known yet.

We assume that, in decomposing $(\nu a)p$, the choice of the restricted name is fixed by the rule that associates to a visible action a its *unique* corresponding restricted action a'. As a process is of the form $(\nu L)t$ with $L = \{a_1, a_2, \ldots, a_n\}$, it can be first translated to the restriction-free process $t\{a'_1/a_1\} \ldots \{a'_n/a_n\}$ (shortened as $t\{L'/L\}$, for $L' = \{a'_1, \ldots, a'_n\} \subseteq \mathcal{L}'$), and then decomposed to obtain a multiset. Function dec essentially performs this decomposition, by removing the restriction (which can occur only externally, by syntactic definition) and by replacing the bound names in L with the corresponding restricted names in L'.

Proposition 3 *For any restriction-free $t \in \mathcal{P}$,* $\mathrm{dec}((\nu L)t) = \mathrm{dec}(t\{L'/L\})$.

This means that we can restrict our attention to restriction-free processes built over Act_γ, as a restricted process $(\nu L)t$ in $\mathcal{P}$ is mapped via dec to the same marking of the restriction-free process $t\{L'/L\}$ in $\mathcal{P}^\gamma$.

Example 3 Consider the (non-well-formed) finite-net multi-CCS process $p = (\nu a)p'$, where $p' = (a.\mathbf{0} \mid (\underline{a}.a.\mathbf{0} \mid \overline{a}.\mathbf{0}))$. Then,

$$\begin{aligned}
\mathrm{dec}(p) &= \mathrm{dec}(p')\{a'/a\} = \mathrm{dec}(a.\mathbf{0} \mid (\underline{a}.a.\mathbf{0} \mid \overline{a}.\mathbf{0}))\{a'/a\} \\
&= (\mathrm{dec}(a.\mathbf{0}) \oplus \mathrm{dec}(\underline{a}.a.\mathbf{0} \mid \overline{a}.\mathbf{0}))\{a'/a\} \\
&= (\mathrm{dec}(a.\mathbf{0}) \oplus \mathrm{dec}(\underline{a}.a.\mathbf{0}) \oplus \mathrm{dec}(\overline{a}.\mathbf{0}))\{a'/a\} \\
&= (a.\mathbf{0} \oplus \underline{a}.a.\mathbf{0} \oplus \overline{a}.\mathbf{0})\{a'/a\} \\
&= a'.\mathbf{0} \oplus \underline{a'}.a'.\mathbf{0} \oplus \overline{a'}.\mathbf{0} \\
&= \mathrm{dec}(a'.\mathbf{0} \mid \underline{a'}.a'.\mathbf{0} \mid \overline{a'}.\mathbf{0}),
\end{aligned}$$

where a' is the corresponding restricted name in $\mathcal{L}'$.

Function dec is well-defined because the constants a process may use are finitely many. This also ensures the following fact.

Proposition 4 *For any $p \in \mathcal{P}^\gamma$,* $\mathrm{dec}(p)$ *is a finite multiset of places.*

Of course, function dec is not injective, because it considers the parallel operator as commutative, associative, with $\mathbf{0}$ as neutral element. In fact, $\mathrm{dec}((p \mid q) \mid r) = \mathrm{dec}(p \mid (q \mid r))$, $\mathrm{dec}(p \mid q) = \mathrm{dec}(q \mid p)$, $\mathrm{dec}(p \mid \mathbf{0}) = \mathrm{dec}(p)$, etc. However, contrary to the decomposition functions in [8,28], one can prove that our dec is surjective. Take any finite multiset of places $m = k_1 \cdot s_1 \oplus \cdots \oplus k_n \cdot s_n$, for $n \geq 0$ ($m = \emptyset$ if $n = 0$), where each $s_i \in S_{\mathrm{MCCS}}$, for $i = 1, \ldots, n$. Then, process $p = s_1^{k_1} \mid \cdots \mid s_n^{k_n}$, where $s^1 = s$ and $s^{n+1} = s \mid s^n$, is such that $\mathrm{dec}(p) = m$. We can be even more demanding and prove that function dec is surjective even if we restrict ourselves to processes in $\mathcal{P}$.

Proposition 5 *Function* $\mathrm{dec} : \mathcal{P} \to \mathcal{M}_{\mathrm{fin}}(S_{\mathrm{MCCS}})$ *is surjective.*

Proof Take any finite multiset of places $m = k_1 \cdot s_1 \oplus \cdots \oplus k_n \cdot s_n$, for $n \geq 0$ ($m = \emptyset$ if $n = 0$), where each $s_i \in S_{\mathrm{MCCS}}$, for $i = 1, \ldots, n$. It is possible to find a substitution $\rho = \{a_1, \ldots, a_k/a'_1, \ldots, a'_k\}$ (with $a'_i \in \mathcal{L}'$ and $a_i \in \mathcal{L}$) such that, for all $i = 1, \ldots, n$, $p_i = s_i\rho$ and $p_i \in \mathcal{P}_{\mathrm{seq}}$. Take process $p = (\nu a_1 a_2 \ldots a_k)(p_1^{k_1} \mid \cdots \mid p_n^{k_n})$, assuming that no restriction is present if $k = 0$ and that, if $n = 0$, $(p_1^{k_1} \mid \cdots \mid p_n^{k_n}) = \mathbf{0}$. It is easy to observe that $\mathrm{dec}(p) = m$. □

4.2 Properties of places and markings

We now list some useful properties of places and markings. First, we extend the definition of sequential subterm of a process p to a set of places S. The goal is to prove that the sequential subterms of $\mathrm{dom}(\mathrm{dec}(p))$ are essentially the same sequential subterms of p. This property will be useful in proving (Theorem 3) that each place statically reachable from $\mathrm{dom}(\mathrm{dec}(p))$ is a sequential subterm of p (up to a possible renaming of bound names to the corresponding restricted names), so that, since $\mathrm{sub}(p)$ is finite for any p (Proposition 1), the set of all the places statically reachable from $\mathrm{dom}(\mathrm{dec}(p))$ is finite as well.

Definition 13 Function $\mathrm{sub}(-)$, defined over finite-net multi-CCS processes in Definition 10, can be extended to a finite set S of places (i.e., of sequential processes) as follows: $\mathrm{sub}(\emptyset) = \emptyset$ and $\mathrm{sub}(S) = \bigcup_{s \in S} \mathrm{sub}(s)$.

Proposition 6 *For any finite set of places S_1 and S_2, if $S_1 \subseteq \mathrm{sub}(S_2)$, then sub$(S_1) \subseteq \mathrm{sub}(S_2)$.*

Proof By induction on the cardinality of S_1, using Remark 1. □

Proposition 7 *For any set of places S, $S \subseteq \mathrm{sub}(S)$.*

Proof For any sequential process s, Definition 10 ensures that $s \in \mathrm{sub}(s)$; hence, the thesis follows trivially. □

Proposition 8 *If p is restriction-free, then sub(dom*$(\mathrm{dec}(p))$ $\subseteq \mathrm{sub}(p)$ *, while if $p = (\nu L)t$, then sub(dom*$(\mathrm{dec}(p))) \subseteq \mathrm{sub}(p)\{L'/L\}$. *Hence, for any p, $|\mathrm{sub}(p)| \geq |\mathrm{sub}(\mathrm{dom}(\mathrm{dec}(p))|$.*

Proof By induction on the definitions of $\mathrm{sub}(p, I)$ and $\mathrm{dec}(p, I)$.

The first base case is when $p = \mathbf{0}$; in such a case $\mathrm{sub}(\mathbf{0}, I) = \{\mathbf{0}\}$; as $\mathrm{dec}(\mathbf{0}, I) = \emptyset$, the thesis follows trivially. The second base case is when $p = A$ and $A \in I$; in such a case, $\mathrm{sub}(A, I) = \emptyset$; as $\mathrm{dec}(A, I) = \emptyset$, the thesis follows trivially. The other simple case is when p is sequential (and not $\mathbf{0}$); in such a case, $\mathrm{dec}(p, I) = \{p\}$ and the thesis follows trivially.

Now the inductive cases. If $p = p_1 \mid p_2$, then $\mathrm{sub}(p, I) = \mathrm{sub}(p_1, I) \cup \mathrm{sub}(p_2, I)$ and $\mathrm{sub}(\mathrm{dom}(\mathrm{dec}(p, I))) = \mathrm{sub}(\mathrm{dom}(\mathrm{dec}(p_1, I)) \cup \mathrm{sub}(\mathrm{dom}(\mathrm{dec}(p_2, I))$; by induction, we have that $\mathrm{sub}(\mathrm{dom}(\mathrm{dec}(p_i, I)) \subseteq \mathrm{sub}(p_i, I)$, for $i = 1, 2$; hence, the thesis follows trivially. If $p = A$, with $A \notin I$ and $A \stackrel{\mathrm{def}}{=} t$, then $\mathrm{sub}(A, I) = \mathrm{sub}(t, I \cup \{A\})$ and $\mathrm{dec}(A, I) = \mathrm{dec}(t, I \cup \{A\})$; by induction, we have $\mathrm{sub}(\mathrm{dom}(\mathrm{dec}(t, I \cup \{A\})) \subseteq \mathrm{sub}(t, I \cup \{A\}))$, and so $\mathrm{sub}(\mathrm{dom}(\mathrm{dec}(A, I)) \subseteq \mathrm{sub}(A, I)$, as required. Finally, if $p = (\nu L)t$, then $\mathrm{sub}(p, I) = \mathrm{sub}(t, I)$, while $\mathrm{sub}(\mathrm{dom}(\mathrm{dec}((\nu L)t, I))) = \mathrm{sub}(\mathrm{dom}(\mathrm{dec}(t, I)\{L'/L\})) = \mathrm{sub}(\mathrm{dom}(\mathrm{dec}(t, I)))\{L'/L\}$; by induction, we have that $\mathrm{sub}(\mathrm{dom}(\mathrm{dec}(t, I)) \subseteq \mathrm{sub}(t, I)$, and therefore also $\mathrm{sub}(\mathrm{dom}(\mathrm{dec}(t, I)))\{L'/L\} \subseteq \mathrm{sub}(t, I)\{L'/L\}$, from which the thesis follows. □

Now we want to extend the definition of well-formed processes to sets of places. To this aim, we define a notion of *well-behaved* set of places, which will be useful in the next section in proving that any transition statically enabled by a well-behaved set S is such that no synchronization of sequences is possible (Theorem 2); first, we prove that a well-formed process p generates a marking $\mathrm{dec}(p)$ such that $\mathrm{dom}(\mathrm{dec}(p))$ is well-behaved (Theorem 1).

Definition 14 Function $\mathrm{ns}(-)$ of Definition 12 is defined over a set S of places as $\mathrm{ns}(S) = \mathrm{ns}(S, \emptyset)$, where $\mathrm{ns}(S, I) = \bigcup_{s \in S} \mathrm{ns}(s, I)$ and $\mathrm{ns}(\emptyset, I) = \emptyset$.

Lemma 1 *If p is restriction-free, then* $\mathrm{ns}(p, I) = \mathrm{ns}(\mathrm{dom}(\mathrm{dec}(p, I)), I)$.

Proof By induction on the definitions of $\mathrm{ns}(p, I)$ and $\mathrm{dec}(p, I)$. The proof is very similar to that of Proposition 8, hence omitted. □

Definition 15 (*Well-behaved*) A set of places S is *well-behaved* if there exist no $\beta \in \mathcal{G}$ such that $\beta \in \mathrm{ns}(S)$ and $\overline{\beta} \in \mathrm{ns}(S)$.

Theorem 1 *If p is well-formed, then dom*$(\mathrm{dec}(p))$ *is well-behaved.*

Proof By induction on the proof of $\mathrm{wf}(p, I)$. The first base case is $p = \mathbf{0}$, and the thesis trivially holds. The second base case is when $p = A$ and $A \in I$; in such a case, $\mathrm{wf}(A, I)$ holds and $\mathrm{dom}(\mathrm{dec}(A, I)) = \emptyset$, hence the thesis trivially holds.

Now the inductive cases. If $p = \mu.p'$, then $\mathrm{wf}(p, I)$ holds only if $\mathrm{wf}(p', I)$ holds. By induction, we have that $\mathrm{dom}(\mathrm{dec}(p', I))$ is well-behaved. Hence, also $\mathrm{dom}(\mathrm{dec}(\mu.p', I))$ is well-behaved, as $\mathrm{ns}(\mathrm{dom}(\mathrm{dec}(\mu.p', I)), I) = \mathrm{ns}(\{\mu.p'\}, I) = \mathrm{ns}(\mu.p', I) = \mathrm{ns}(p', I)$ and $\mathrm{ns}(p', I) = \mathrm{ns}(\mathrm{dom}(\mathrm{dec}(p', I)), I)$ by Lemma 1.

If $p = \underline{\alpha}.p'$, then $\mathrm{wf}(p, I)$ holds if there exists no $\beta \in \mathcal{G}$ such that $\beta \in \mathrm{ns}(p, I)$ and $\overline{\beta} \in \mathrm{ns}(p, I)$; by Lemma 1, $\mathrm{ns}(p, I) = \mathrm{ns}(\mathrm{dom}(\mathrm{dec}(p, I)), I)$, and so the thesis follows trivially.

If $p = p_1 + p_2$, then $\mathrm{wf}(p, I)$ holds only if $\mathrm{wf}(p_1, I)$ and $\mathrm{wf}(p_2, I)$ hold and, additionally, there exists no $\beta \in \mathcal{G}$ such that $\beta \in \mathrm{ns}(p_1, I)$ and $\overline{\beta} \in \mathrm{ns}(p_2, I)$. By induction, we have that $\mathrm{dom}(\mathrm{dec}(p_i, I))$ is well-behaved, for $i = 1, 2$; so, there exists no $\beta \in \mathcal{G}$ such that $\beta \in \mathrm{dom}(\mathrm{dec}(p_i, I))$ and $\overline{\beta} \in \mathrm{dom}(\mathrm{dec}(p_i, I))$, for $i = 1, 2$. By Lemma 1, $\mathrm{ns}(\mathrm{dom}(\mathrm{dec}(p_i, I)), I) = \mathrm{ns}(p_i, I)$ for $i = 1, 2$, and so there exists no $\beta \in \mathcal{G}$ such that $\beta \in \mathrm{ns}(p_i, I)$ and $\overline{\beta} \in$

Table 9 Rules for net transitions [symmetric rule (sum_2) omitted]

$$\text{(pref)}\ \frac{}{\{\mu.p\} \xrightarrow{\mu} \text{dec}(p)} \qquad \text{(sum}_1\text{)}\ \frac{\{p\} \xrightarrow{\sigma} m}{\{p + p'\} \xrightarrow{\sigma} m}$$

$$\text{(s-pref)}\ \frac{\{p\} \xrightarrow{\sigma} m}{\{\underline{\gamma}.p\} \xrightarrow{\gamma \diamond \sigma} m} \qquad \text{(s-com)}\ \frac{m_1 \xrightarrow{\sigma_1} m_1'\ m_2 \xrightarrow{\sigma_2} m_2'}{m_1 \oplus m_2 \xrightarrow{\sigma} m_1' \oplus m_2'}\ \text{Sync}(\sigma_1, \sigma_2, \sigma)$$

Table 10 The proof of a net transition

$$\dfrac{\dfrac{\dfrac{\text{(pref)}\ \ }{\{b'.p\} \xrightarrow{b'} dec(p)}\ \text{(s-pref)} \qquad \dfrac{\text{(pref)}\ \ }{\{\overline{a}.q\} \xrightarrow{\overline{a}} dec(q)}}{\{\underline{a}.b'.p\} \xrightarrow{ab'} dec(p)}\ \text{(s-com)} \qquad \dfrac{\text{(pref)}\ \ }{\{\overline{b'}.r\} \xrightarrow{\overline{b'}} dec(r)}}{\{\underline{a}.b'.p, \overline{a}.q\} \xrightarrow{b'} dec(p) \oplus dec(q)}\ \text{(s-com)}$$

$$\{\underline{a}.b'.p, \overline{a}.q, \overline{b'}.r\} \xrightarrow{\tau} dec(p) \oplus dec(q) \oplus dec(r)$$

$\text{ns}(p_i, I)$, for $i = 1, 2$. Therefore, $\text{dom}(\text{dec}(p_1 + p_2, I)) = \{p_1 + p_2\}$ is well-behaved, too, because $\text{ns}(p_1 + p_2, I) = \text{ns}(p_1, I) \cup \text{ns}(p_2, I)$, and then there exists no $\beta \in \mathcal{G}$ such that $\beta \in \text{ns}(p_1 + p_2, I)$ and $\overline{\beta} \in \text{ns}(p_1 + p_2, I)$, as required. The case when $p = p_1 \mid p_2$ is similar to the above, hence omitted.

If $p = (\nu a)p'$, then $\text{wf}(p, I)$ holds only if $\text{wf}(p', I)$ holds. By induction, we have that $\text{dom}(\text{dec}(p', I))$ is well-behaved. It is easy to see that $\text{dom}(\text{dec}(p', I))\{a'/a\}$ is well-behaved too, as the substitution replaces action a with a new name a' not in use. Hence, as $\text{dom}(\text{dec}(p', I))\{a'/a\} = \text{dom}(\text{dec}(p', I)\{a'/a\}) = \text{dom}(\text{dec}(p, I))$, also $\text{dom}(\text{dec}(p, I))$ is well-behaved.

If $p = A$, with $A \notin I$ and $A \stackrel{\text{def}}{=} q$, then $\text{wf}(A, I)$ holds only if $\text{wf}(q, I \cup \{A\})$ holds. By induction, $\text{dom}(\text{dec}(q, I \cup \{A\}))$ is well-behaved. Since $\text{dec}(A, I) = \text{dec}(q, I \cup \{A\})$, also $\text{dom}(\text{dec}(A, I))$ is well-behaved. □

4.3 Net transitions

Let $\mathcal{A}^\gamma = \{\tau\} \cup \mathcal{G}^+$, ranged over by σ with abuse of notation, be the set of labels, and let $\rightarrow \subseteq \mathcal{M}_{\text{fin}}(S_{\text{MCCS}}) \times \mathcal{A}^\gamma \times \mathcal{M}_{\text{fin}}(S_{\text{MCCS}})$, be the least set of transitions generated by the axiom and rules in Table 9, where in a transition $m_1 \xrightarrow{\sigma} m_2$, m_1 is the pre-set, σ is the label and m_2 is the post-set.

Axiom (pref) states that if one token is present in the place $\mu.p$ then a μ-labeled transition is derivable from marking $\{\mu.p\}$, producing the marking $\text{dec}(p)$. This holds for any μ, i.e., for the invisible action τ, for any visible action α as well as for any restricted action α'. In rule (s-pref), γ ranges over visible actions α and *restricted* ones α'. This rule requires that the premise transition $\{p\} \xrightarrow{\sigma} m$ be derivable by the rules, starting form the sequential process p. Rule (sum_1) and its symmetric (sum_2) are as expected: the transition from place $p + p'$ are those from places p and p', as both p and p' are sequential. Finally, rule (s-com) explains how synchronization takes place: it is needed that m_1 and m_2 perform synchronizable sequences σ_1 and σ_2, producing σ; here we assume that Sync has been extended also to restricted actions in the obvious way, i.e., a restricted action $\overline{\alpha'}$ can be synchronized only with its complementary restricted action α' or with a sequence beginning with α'. As an example, net transition $\{\underline{a}.b'.p, \overline{a}.q, \overline{b'}.r\} \xrightarrow{\tau} \text{dec}(p) \oplus \text{dec}(q) \oplus \text{dec}(r)$ is derivable (see Table 10).

Transitions with labels containing restricted actions should not be taken in the resulting net, as we accept only transitions labeled on $\mathcal{A} = \{\tau\} \cup (\mathcal{L} \cup \overline{\mathcal{L}})^+$. However, they are useful in producing acceptable transitions, as two complementary restricted actions can synchronize, producing a τ-labeled transition or shortening the synchronized sequence: e.g., in the example above, the derivable transition $\{b'.p\} \xrightarrow{b'} \text{dec}(p)$ is not an acceptable transition because its label is not in $\mathcal{A}$, while $\{\underline{a}.b'.p, \overline{a}.q, \overline{b'}.r\} \xrightarrow{\tau} \text{dec}(p) \oplus \text{dec}(q) \oplus \text{dec}(r)$ is so. Hence, the P/T net for finite-net multi-CCS is the triple $N_{\text{MCCS}} = (S_{\text{MCCS}}, \mathcal{A}, T_{\text{MCCS}})$, where the set $T_{\text{MCCS}} = \{(m_1, \sigma, m_2) \mid m_1 \xrightarrow{\sigma} m_2$ is derivable by the rules and $\sigma \in \mathcal{A}\}$ is obtained by filtering out those transitions derivable by the rules such that no restricted name α' occurs in σ.

4.4 Properties of net transitions

Some useful properties of net transitions are listed here. First, given a transition $t = (m_1, \sigma, m_2)$, derivable by the rules in Table 9, we show that the marking m_2 generates subterms that are already present in the marking m_1.

Proposition 9 *Let $t = m_1 \xrightarrow{\sigma} m_2$ be a transition derivable by the rules in Table 9. Then, sub(dom(m_2)) $\subseteq$ sub(dom(m_1)).*

Proof By induction on the proof of t. □

Lemma 2 *Let $t = m_1 \xrightarrow{\sigma} m_2$ be a transition derivable by the rules in Table 9. Then,* $\text{ns}(\text{dom}(m_2)) \subseteq \text{ns}(\text{dom}(m_1))$.

Proof By induction on the proof of t. □

Proposition 10 *If* $t = m_1 \xrightarrow{\sigma} m_2$ *is derivable by the rules and* $\mathrm{dom}(m_1)$ *is well-behaved, then* $\mathrm{dom}(m_2)$ *is well-behaved.*

Proof By Lemma 2, we know that $\mathrm{ns}(\mathrm{dom}(m_2)) \subseteq \mathrm{ns}(\mathrm{dom}(m_1))$. Therefore, if $\mathrm{dom}(m_1)$ is well-behaved, then $\mathrm{dom}(m_2)$ is well-behaved, too. □

Corollary 1 *If* S_1 *is well-behaved and* $S_1 \Longrightarrow^* S_k$, *then* S_k *is well-behaved.*

Proof By induction on the static reachability relation $\Longrightarrow^*$. The base case is $S_1 \Longrightarrow^* S_1$ and it is trivial. The inductive case is $S_1 \Longrightarrow^* S_{k-1} \xrightarrow{t} S_k$. By induction we can assume that S_{k-1} is well-behaved. Let $t = m_1 \xrightarrow{\sigma} m_2$ be a transition in T_{MCCS}. The set $\mathrm{ns}(S_{k-1})$ is $\mathrm{ns}(S_{k-1} \backslash \mathrm{dom}(m_1)) \cup \mathrm{ns}(\mathrm{dom}(m_1))$. The set $\mathrm{ns}(S_k)$ is $\mathrm{ns}(S_{k-1}) \cup \mathrm{ns}(\mathrm{dom}(m_2))$. By Lemma 2, we have $\mathrm{ns}(\mathrm{dom}(m_2)) \subseteq \mathrm{ns}(\mathrm{dom}(m_1))$. Therefore, $\mathrm{ns}(S_k) = \mathrm{ns}(S_{k-1}) \cup \mathrm{ns}(\mathrm{dom}(m_2)) \subseteq \mathrm{ns}(S_{k-1}) \cup \mathrm{ns}(\mathrm{dom}(m_1)) = \mathrm{ns}(S_{k-1})$; hence, also S_k is well-behaved. □

Now we want to prove that when a transition $m \xrightarrow{\sigma} m'$, whose label $\sigma \neq \tau$, involves in its proof some sequence of length greater than one, then the names of σ are all contained in $\mathrm{ns}(\mathrm{dom}(m))$.

Lemma 3 *If* $t = (m, \sigma, m')$ *is derivable by the rules of Table 9, and either* $|\sigma| \geq 2$ *or* $\sigma \neq \tau$ *and there exists a transition label* σ' *in its proof tree with* $|\sigma'| \geq 2$, *then* $n(\sigma) \subseteq \mathrm{ns}(\mathrm{dom}(m))$.

Proof By induction on the proof of transition t. If $m = \{\mu.p\}$, then, by axiom (pref), $t = (m, \mu, \mathrm{dec}(p))$. This case is vacuous as the only transition label in the proof tree is μ. If $m = \{\underline{\gamma}.p\}$, then $t = (m, \sigma, m')$ is derivable only if $(\{p\}, \sigma', m')$ is derivable, with $\sigma = \gamma \diamond \sigma'$. If $|\sigma'| \geq 2$, then induction can be applied to conclude that $n(\sigma') \subseteq \mathrm{ns}(\mathrm{dom}(\{p\})) = \mathrm{ns}(\{p\}) = \mathrm{ns}(p)$; hence, $n(\sigma) = \{\gamma\} \cup n(\sigma') \subseteq \{\gamma\} \cup \mathrm{ns}(p)$. Since $\mathrm{ns}(m) = \mathrm{ns}(\{\underline{\gamma}.p\}) = \mathrm{ns}(\underline{\gamma}.p)$ and $\{\gamma\} \cup \mathrm{ns}(p) \subseteq \mathrm{ns}(\underline{\gamma}.p)$, the thesis $n(\sigma) \subseteq n(m)$ follows trivially. If $|\sigma'| = 1$, then two further subcases are possible: either $\sigma' = \tau$ or $\sigma' = \gamma'$; in the former subcase, this is possible only if p, being sequential, has performed a prefix τ via (pref), so that no transition label in the proof tree is longer than one, hence this subcase is vacuous; in the latter subcase, $\sigma = \gamma\gamma'$ and $n(\sigma) \subseteq \mathrm{ns}(\underline{\gamma}.p)$, because $\gamma' \in \mathrm{In}(p)$.

If $m = \{p_1 + p_2\}$, then $t = (m, \sigma, m')$ is derivable only if $(\{p_1\}, \sigma, m')$ or $(\{p_2\}, \sigma, m')$ are derivable. W.l.o.g., assume that $(\{p_1\}, \sigma, m')$; then, if the hypothesis holds for this premise, by induction, we have $n(\sigma) \subseteq \mathrm{ns}(\{p_1\}) = \mathrm{ns}(p_1)$. Since $\mathrm{ns}(p_1) \subseteq \mathrm{ns}(p) = \mathrm{ns}(m)$, the thesis follows by transitivity.

If t is derived by rule (s-com), then $m = m_1 \oplus m_2$, $m' = m'_1 \oplus m'_2$ and transitions $t_1 = (m_1, \sigma_1, m'_1)$ and $t_2 = (m_2, \sigma_2, m'_2)$ are derivable, with $\mathrm{Sync}(\sigma_1, \sigma_2, \sigma)$. As by hypothesis $\sigma \neq \tau$, then σ_1 or σ_2 must be of length greater than one. W.l.o.g., assume $|\sigma_1| \geq 2$. Since $\mathrm{Sync}(\sigma_1, \sigma_2, \sigma)$, necessarily $n(\sigma) \subseteq n(\sigma_1)$. By induction, $n(\sigma_1) \subseteq \mathrm{ns}(\mathrm{dom}(m_1))$; as $\mathrm{ns}(\mathrm{dom}(m_1)) \subseteq \mathrm{ns}(\mathrm{dom}(m_1 \oplus m_2))$, the thesis follows by transitivity. □

Proposition 11 *If* $t = (m, \gamma, m')$ *is derivable by the rules of Table 9 by using rule (s-com), then* $\gamma \in \mathrm{ns}(\mathrm{dom}(m))$.

Proof By induction on the proof of t. If rule (s-com) occurs in the proof of t, then m cannot be a sequential process. Therefore, the first rule must be (s-com), and so $m = m_1 \oplus m_2$, $m' = m'_1 \oplus m'_2$, $t_1 = (m_1, \sigma_1, m'_1)$ and $t_2 = (m_2, \sigma_2, m'_2)$ are derivable, with $\mathrm{Sync}(\sigma_1, \sigma_2, \gamma)$. So, σ_1 or σ_2 must be a sequence of length greater than one, and so by Lemma 3, it follows that $\gamma \in \mathrm{ns}(\mathrm{dom}(m))$. □

Theorem 2 *If* $t = (m, \sigma, m')$ *is derivable by the rules and* $\mathrm{dom}(m)$ *is well-behaved, then the proof of* t *never synchronizes two sequences, not even indirectly.*

Proof By induction on the proof of t. If m is a singleton, then rule (s-com) is never used, and so no synchronization of sequences is possible. Otherwise, the first rule must be (s-com), and so $m = m_1 \oplus m_2$, $m' = m'_1 \oplus m'_2$, $t_1 = (m_1, \sigma_1, m'_1)$ and $t_2 = (m_2, \sigma_2, m'_2)$ are derivable, with $\mathrm{Sync}(\sigma_1, \sigma_2, \sigma)$. As $\mathrm{dom}(m)$ is well-behaved, so are also $\mathrm{dom}(m_1)$ and $\mathrm{dom}(m_2)$; therefore, by induction, we know that in the proofs of transitions t_1 and t_2 two sequences are never synchronized. So, it remains to prove that the thesis holds for the resulting σ. By definition of Sync, if $\sigma = \tau$, then both σ_1 and σ_2 are complementary actions, say $\sigma_1 = \gamma$ and $\sigma_2 = \overline{\gamma}$. If both t_1 and t_2 are derived by using rule (s-com), then t synchronizes two sequences, even if indirectly; however, this is not possible, because Proposition 11 would ensure that $\gamma \in \mathrm{ns}(\mathrm{dom}(m_1))$ and $\overline{\gamma} \in \mathrm{ns}(\mathrm{dom}(m_2))$, contradicting that $\mathrm{dom}(m_1 \oplus m_2)$ be well-behaved. Therefore, t_1 or t_2 is derived without using rule (s-com) and so no synchronization of sequences is produced. By definition of Sync, if $\sigma \neq \tau$, then either σ_1 or σ_2 is a sequence of length greater than one; w.l.o.g. assume that $|\sigma_1| \geq 2$ and $\sigma_2 = \overline{\gamma}$. By Lemma 3, $n(\sigma_1) \subseteq \mathrm{ns}(\mathrm{dom}(m_1))$, in particular, $\gamma \in \mathrm{ns}(\mathrm{dom}(m_1))$. If t_2 is derived by using rule (s-com), then Proposition 11 would ensure that $\overline{\gamma} \in \mathrm{ns}(\mathrm{dom}(m_2))$, contradicting that $\mathrm{dom}(m_1 \oplus m_2)$ be well-behaved. Therefore, t_2 is derived without using rule (s-com) and so no synchronization of sequences is produced. □

Remark 2 By the proof of the prop above, it is clear that any transition $t = m_1 \xrightarrow{\sigma} m_2$ derivable from a *well-behaved*

set of places $\mathrm{dom}(m_1)$ is such that in the proof tree for t, whenever rule (s-com) is used with premise transitions t_1 and t_2, at least one of the two, say t_1 w.l.o.g., is such that ${}^\bullet t_1$ is a singleton and $l(t_1)$ is a single action in Act^γ. That is, any derivable transition t from a well-behaved set of places $\mathrm{dom}(m_1)$ is such that one sequential process $s \in \mathrm{dom}(m_1)$ acts as the *leader* of the multi-party synchronization, while the other sequential components contribute each with a single action, acting as *servants*.

4.5 The reachable subnet Net(p)

The P/T system associated to $p \in \mathcal{P}^\gamma$ is the subnet of N_{MCCS} statically reachable from the initial marking $\mathrm{dec}(p)$. We indicate with $\mathrm{Net}(p)$ such a subnet.

Definition 16 Let p be a process in $\mathcal{P}^\gamma$. The P/T net system statically associated to p is $\mathrm{Net}(p) = (S_p, A_p, T_p, m_0)$, where $m_0 = \mathrm{dec}(p)$ and

$$S_p = [\![\mathrm{dom}(m_0)\rangle \text{ computed in } N_{\mathrm{MCCS}},$$
$$T_p = \{t \in T_{\mathrm{MCCS}} \mid S_p[\![t\rangle\}$$
$$A_p = \{\sigma \in \mathcal{A} \mid \exists t \in T_p \text{ such that } l(t) = \sigma\}.$$

The following three propositions present facts that are obviously true by construction of the net $\mathrm{Net}(p)$ associated to a finite-net multi-CCS process p.

Proposition 12 *For any $p \in \mathcal{P}$, $Net(p)$ is a statically reduced P/T net.*

Proposition 13 *If $\mathrm{dec}(p) = \mathrm{dec}(q)$, then $Net(p) = \mathrm{Net}(q)$.*

Proposition 14 *For any $t \in \mathcal{P}$, let $Net(t) = (S, A, T, m_0)$. Then, for any $n \geq 1$, $Net(t^n) = (S, A, T, n \cdot m_0)$, where $t^1 = t$ and $t^{n+1} = t \mid t^n$.*

For any $(\nu L)t \in \mathcal{P}$, let $Net((\nu L)t) = (S, A, T, m_0)$. Then, for any $n \geq 1$, $Net((\nu L)(t^n)) = (S, A, T, n \cdot m_0)$.

Definition 16 suggests a way of generating $\mathrm{Net}(p)$ with an algorithm based on the inductive definition of the static reachability relation (see Definition 9): start by the initial set of places $\mathrm{dom}(\mathrm{dec}(p))$, and then apply the rules in Table 9 in order to produce the set of transitions (labeled on $\mathcal{A}$) statically enabled at $\mathrm{dom}(\mathrm{dec}(p))$, as well as the additional places statically reachable by means of such transitions. Then repeat this procedure from the set of places statically reached so far. The problems with this algorithm are two:

- the obvious *halting condition* is "until no new places are statically reachable"; of course, the algorithm terminates if we know that the set S_p of places statically reachable from $\mathrm{dom}(\mathrm{dec}(p))$ is finite; additionally,
- at each step of the algorithm, we have to be sure that the set of transitions derivable from the current set of statically reachable places is finite.

We are going to prove these two facts: (i) S_p is finite for any $p \in \mathcal{P}$, and (ii) for any *well-formed* process p, and for any set of places S, statically reachable from $\mathrm{dom}(\mathrm{dec}(p))$, the set of transitions statically enabled at S is finite.

Theorem 3 *For any $p \in \mathcal{P}$, let $Net(p) = (S_p, A_p, T_p, m_0)$ be defined as in Definition 16. Then, set S_p is finite.*

Proof We prove, by induction on the static reachability relation $\Longrightarrow^*$, that any set S_i of places, statically reachable from $\mathrm{dom}(m_0)$, is a subset of $\mathrm{sub}(\mathrm{dom}(m_0))$. This is enough as, by Proposition 8, we know that $|\mathrm{sub}(p)| \geq |\mathrm{sub}(\mathrm{dom}(m_0))|$; moreover, by Proposition 1, $\mathrm{sub}(p)$ is finite and so the thesis follows trivially.

The base case is $\mathrm{dom}(m_0) \Longrightarrow^* \mathrm{dom}(m_0)$. By Proposition 7, we have the required $\mathrm{dom}(m_0) \subseteq \mathrm{sub}(\mathrm{dom}(m_0))$.

Now, let us assume that S_i is a set of places statically reachable from $\mathrm{dom}(m_0)$ and let $t = m_1 \xrightarrow{\sigma} m_2$ be such that $S_i \overset{t}{\Longrightarrow} S_{i+1}$. By induction, we know that $S_i \subseteq \mathrm{sub}(\mathrm{dom}(m_0))$. So, we have to prove that the new places reached via t are in $\mathrm{sub}(\mathrm{dom}(m_0))$. Note that since $\mathrm{dom}(m_1) \subseteq S_i$, it follows that $\mathrm{dom}(m_1) \subseteq \mathrm{sub}(\mathrm{dom}(m_0))$ and also that $\mathrm{sub}(\mathrm{dom}(m_1))) \subseteq \mathrm{sub}(\mathrm{dom}(m_0))$, by Proposition 6. By Proposition 7, we have that $\mathrm{dom}(m_2) \subseteq \mathrm{sub}(\mathrm{dom}(m_2)$; by Proposition 9, we have that $\mathrm{sub}(\mathrm{dom}(m_2)) \subseteq \mathrm{sub}(\mathrm{dom}(m_1))$; by transitivity, $\mathrm{dom}(m_2) \subseteq \mathrm{sub}(\mathrm{dom}(m_0))$, and so $S_{i+1} = S_i \cup \mathrm{dom}(m_2) \subseteq \mathrm{sub}(\mathrm{dom}(m_0))$, as required.

Summing up, any place statically reachable from $\mathrm{dom}(m_0)$ is a (possibly, one-time renamed) sequential subterm of p. As by Proposition 1, $\mathrm{sub}(p)$ is finite, then also S_p (the largest set of places statically reachable from $\mathrm{dom}(m_0)$) is finite. □

We now want to prove that for any *well-formed* finite-net multi-CCS process p, and for any set of places $S \subseteq S_{\mathrm{MCCS}}$, statically reachable from $\mathrm{dom}(\mathrm{dec}(p))$, the set of transitions statically enabled at S is finite. Some auxiliary definitions and results are necessary. Given a single place $s \in S$, by $s \vdash t$ we mean that transition $t = (\{s\}, \sigma, m)$ is derivable by the rules in Table 9, hence with $\sigma \in \mathcal{A}^\gamma$.

Lemma 4 *Set $T_s = \{t \mid s \vdash t\}$ is finite, for any $s \in S_{\mathrm{MCCS}}$.*

Proof By induction on the structure of the sequential process s and then by induction on the rules in Table 9.

Given a finite set of places $S \subseteq S_{\mathrm{MCCS}}$, let T_1 be $\bigcup_{s \in S} T_s$, i.e., the set of all transitions, with a singleton preset in S, derivable by the rules with labeling in $\mathcal{A}^\gamma$. Set T_1 is finite,

being the finite union (as S is finite) of finite sets (as T_s is finite for any s).

If p is well-formed, then $\mathrm{dom}(\mathrm{dec}(p))$ is well-behaved by Theorem 1. If S is statically reachable from $\mathrm{dom}(\mathrm{dec}(p))$, then S is well-behaved by Corollary 1. Let $k \in \mathbb{N}$ be the length of the longest label of any transition in T_1. Remark 2 explains that if a multi-party transition t is derivable by the rules from the well-behaved set $\mathrm{dom}({}^\bullet t) \subseteq S$, then its proof contains $k+1$ synchronizations at most, each one between a transition (labeled with a sequence) and a *singleton-preset* transition (labeled with a *single* action). Therefore, the set of all the transitions statically enabled at a well-behaved set S can be defined by means of a sequence of sets T_i of transitions, for $2 \leq i \leq k+1$, where each transition $t \in T_i$ has a preset ${}^\bullet t$ composed of i tokens, as follows:

$$\begin{aligned} T_i = \{&(m_1 \oplus m_2, \sigma, m_1' \oplus m_2') \mid \\ &\exists \sigma_1 \exists \gamma . (m_1, \sigma_1, m_1') \in T_{i-1}, \\ &(m_2, \gamma, m_2') \in T_1, \mathrm{Sync}(\sigma_1, \gamma, \sigma)\} \end{aligned}$$

Note that T_2 is finite, because T_1 is finite; inductively, T_{i+1}, for $2 \leq i \leq k$ is finite, because T_i and T_1 are finite. The set T_S of all the transitions statically enabled at S is $\{t \mid t \in \bigcup_{i=1}^{k+1} T_i \wedge l(t) \in \mathcal{A}\}$, where only transitions labeled on $\mathcal{A}$ are considered. T_S is finite, being a finite union of finite sets; therefore, we have the following result.

Proposition 15 *If $S \subseteq S_{\mathrm{MCCS}}$ is a well-behaved, finite set of places, then set T_S of all the transitions enabled at S is finite.*

Example 4 Consider the non-well-formed process $p = (\nu a)(a.\mathbf{0} \mid (\underline{a}.a.\mathbf{0} \mid \overline{a}.\mathbf{0}))$, discussed in Example 3. We have that $\mathrm{dec}(p) = a'.\mathbf{0} \oplus \underline{a'}.a'.\mathbf{0} \oplus \overline{a'}.\mathbf{0}$, which is not well-behaved because $a' \in \mathrm{ns}(\mathrm{dom}(\mathrm{dec}(p)))$ and $\overline{a'} \in \mathrm{ns}(\mathrm{dom}(\mathrm{dec}(p)))$. It is easy to observe that transition $t_1 = a'.\mathbf{0} \oplus \underline{a'}.a'.\mathbf{0} \oplus \overline{a'}.\mathbf{0} \xrightarrow{\tau} \emptyset$ is derivable, because first we synchronize $\overline{a'}a'$ with a', yielding a', which is then synchronized with $\overline{a'}$, yielding τ. However, the occurrence of action a' produced by the first synchronization may be used to synchronize an additional sequence $\overline{a'}a'$, yielding a' again. Therefore, it is not difficult to see that also $t_n = a'.\mathbf{0} \oplus n \cdot \underline{a'}.a'.\mathbf{0} \oplus \overline{a'}.\mathbf{0} \xrightarrow{\tau} \emptyset$, is statically enabled at $\mathrm{dom}(\mathrm{dec}(p))$, for any $n \geq 1$. Hence, the set of transitions statically enabled at $\mathrm{dom}(\mathrm{dec}(p))$ is infinite.

Theorem 4 *For any well-formed, finite-net multi-CCS process p, $\mathit{Net}(p) = (S_p, A_p, T_p, \mathrm{dec}(p))$ is a finite P/T net.*

Proof If p is well-formed, then $\mathrm{dom}(\mathrm{dec}(p))$ is well-behaved, by Theorem 1, and finite, by Proposition 4. By Proposition 15, set $T_{\mathrm{dom}(\mathrm{dec}(p))}$ is finite. Let S_1 be the set of places $\mathrm{dom}(\mathrm{dec}(p)) \cup \bigcup_{t \in T_{\mathrm{dom}(\mathrm{dec}(p))}} \mathrm{dom}(t^\bullet)$. If

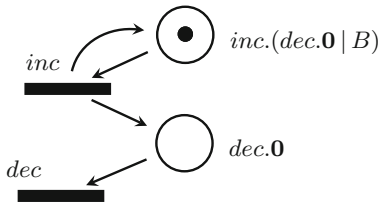

Fig. 1 The finite P/T system for a semi-counter

$S_1 = \mathrm{dom}(\mathrm{dec}(p))$, then $S_p = \mathrm{dom}(\mathrm{dec}(p))$ and $T_p = T_{\mathrm{dom}(\mathrm{dec}(p))}$. Otherwise, repeat the step above for S_1; in fact, S_1 is a finite set of places, because $\mathrm{dom}(\mathrm{dec}(p))$ is finite, set $T_{\mathrm{dom}(\mathrm{dec}(p))}$ is finite and each transition has a finite post-set; moreover, S_1 is well-behaved by Corollary 1. By repeating the step above for S_1, we compute a new finite set T_{S_1} of transitions statically enabled at S_1, and a new finite set S_2 of places statically reachable from S_1 via the transitions in T_{S_1}; if $S_2 = S_1$, then $S_p = S_1$ and $T_p = T_{S_1}$. Otherwise, repeat the step above for S_2. This procedure will end eventually because, by Theorem 3, we are sure that S_p is a finite set. □

Example 5 (*Semi-counter*) A semi-counter, i.e., a counter that cannot test for zero, can be described by the finite-net multi-CCS process $B \stackrel{\mathrm{def}}{=} \mathrm{inc}.(\mathrm{dec}.\mathbf{0} \mid B)$. $\mathrm{Net}(B)$ is the net (S_B, A_B, T_B, m_0) we are going to construct, where the initial marking m_0 is $\mathrm{dec}(B) = \{\mathrm{inc}.(\mathrm{dec}.\mathbf{0} \mid B)\}$. Then, the only enabled transition is

$$t_1 = \{\mathrm{inc}.(\mathrm{dec}.\mathbf{0} \mid B)\} \xrightarrow{\mathrm{inc}} \{\mathrm{dec}.\mathbf{0}, \mathrm{inc}.(\mathrm{dec}.\mathbf{0} \mid B)\}$$

and the set S_1 of places statically reachable in one step from $\mathrm{dom}(m_0)$ is $S_1 = \{\mathrm{dec}.\mathbf{0}, \mathrm{inc}.(\mathrm{dec}.\mathbf{0} \mid B)\}$. From S_1, besides transition t_1 above, also transition $t_2 = \{\mathrm{dec}.\mathbf{0}\} \xrightarrow{\mathrm{dec}} \emptyset$ is derivable, which however does not add any new reachable place. So, S_1 is the set S_B, $\{t_1, t_2\}$ is the set T_B and $\{\mathrm{inc}, \mathrm{dec}\}$ is the set A_B. The resulting net $\mathrm{Net}(B)$ is outlined in Fig. 1. □

Example 6 (*1/3 Semi-counter*) For the well-formed process $p = (\nu c)A$, where $A \stackrel{\mathrm{def}}{=} \mathrm{inc}.(A \mid (\underline{c}.\underline{c}.\mathrm{dec}.\mathbf{0} + \overline{c}.\mathbf{0}))$, three occurrences of inc are needed to enable one dec. $\mathrm{Net}(p)$ is the net (S_p, A_p, T_p, m_0) we are going to construct, where the initial marking m_0 is $\mathrm{dec}(p) = \mathrm{dec}((\nu c)A) = \mathrm{dec}(A)\{c'/c\} = \{\mathrm{inc}.(A \mid (\underline{c}.\underline{c}.\mathrm{dec}.\mathbf{0} + \overline{c}.\mathbf{0}))\}\{c'/c\} = \{s_1\}$; place s_1 is $\mathrm{inc}.(A_{\{c'/c\}} \mid (\underline{c'}.\underline{c'}.\mathrm{dec}.\mathbf{0} + \overline{c'}.\mathbf{0}))$, where the new constant $A_{\{c'/c\}}$ is obtained by applying the substitution $\{c'/c\}$ to the body of A: $A_{\{c'/c\}} \stackrel{\mathrm{def}}{=} \mathrm{inc}.(A_{\{c'/c\}} \mid (\underline{c'}.\underline{c'}.\mathrm{dec}.\mathbf{0} + \overline{c'}.\mathbf{0}))$. Now, only transition $t_1 = \{s_1\} \xrightarrow{\mathrm{inc}} \{s_1, s_2\}$ is derivable from $\mathrm{dom}(m_0) = \{s_1\}$, where $s_2 = \underline{c'}.\underline{c'}.\mathrm{dec}.\mathbf{0} + \overline{c'}.\mathbf{0}$ is a new statically reachable place. Note that s_2 can produce two transitions in T_{s_2}, namely $t' = \{s_2\} \xrightarrow{c'c'\mathrm{dec}} \emptyset$ and

Table 11 The proof of a net transition, where $s_2 = \underline{c'}.\underline{c'}.\mathrm{dec}.\mathbf{0} + \overline{c'}.\mathbf{0}$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\{dec.\mathbf{0}\} \xrightarrow{dec} \emptyset}\,(\text{pref})}{\{\underline{c'}.dec.\mathbf{0}\} \xrightarrow{c'dec} \emptyset}\,(\text{s-pref})}{\{\underline{c'}.\underline{c'}.dec.\mathbf{0}\} \xrightarrow{c'c'dec} \emptyset}\,(\text{s-pref})}{\{s_2\} \xrightarrow{c'c'dec} \emptyset}\,(\text{sum}_1) \qquad \dfrac{}{\{\overline{c'}.\mathbf{0}\} \xrightarrow{\overline{c'}} \emptyset}\,(\text{pref})}{\{s_2\} \xrightarrow{\overline{c'}} \emptyset}\,(\text{sum}_2)}{\{s_2, s_2\} \xrightarrow{c'dec} \emptyset}\,(\text{s-com}) \qquad \dfrac{\dfrac{}{\{\overline{c'}.\mathbf{0}\} \xrightarrow{\overline{c'}} \emptyset}\,(\text{pref})}{\{s_2\} \xrightarrow{\overline{c'}} \emptyset}\,(\text{sum}_2)}{\{s_2, s_2, s_2\} \xrightarrow{dec} \emptyset}\,(\text{s-com})$$

$t'' = \{s_2\} \xrightarrow{\overline{c'}} \emptyset$, but both are not labeled with a sequence in $\mathcal{A}$. However, these transitions can be composed by means of rule (s-com), as shown in Table 11, to produce transition $t_2 = 3 \cdot s_2 \xrightarrow{\mathrm{dec}} \emptyset$, which does not add any new reachable place. So, $S_p = \{s_1, s_2\}$ and $T_p = \{t_1, t_2\}$.

4.6 The CRW example

Let us consider process CRW of Sect. 3.4. The net associated to CRW is $\mathrm{Net(CRW)} = (S_{\mathrm{CRW}}, A_{\mathrm{CRW}}, T_{\mathrm{CRW}}, m_0)$ we are going to construct, where the initial marking is $m_0 = \mathrm{dec(CRW)} = \mathrm{dec}((\nu l, u)(R \mid R \mid R \mid W \mid W \mid L \mid L \mid L)) = \mathrm{dec}(R \mid R \mid R \mid W \mid W \mid L \mid L \mid L)\{l'/l\}\{u'/u\} = 4 \cdot rd \oplus 3 \cdot lk \oplus 2 \cdot wr$, where $rd = l'.\mathrm{read}.u'.R'$ (with $R' \stackrel{\mathrm{def}}{=} l'.\mathrm{read}.u'.R'$), $lk = \overline{l'}.\overline{u'}.L'$ (with $L' \stackrel{\mathrm{def}}{=} \overline{l'}.\overline{u'}.L'$) and $wr = \underline{l'}.\underline{l'}.l'.\mathrm{write}.\underline{u'}.\underline{u'}.u'.W'$ (with $W' \stackrel{\mathrm{def}}{=} \underline{l'}.\underline{l'}.l'.\mathrm{write}.\underline{u'}.\underline{u'}.u'.W'$).

One of the two possible initial transitions is $wr \oplus 3 \cdot lk \xrightarrow{\tau} wr' \oplus 3 \cdot lk'$, where $wr' = \mathrm{write}.\underline{u'}.\underline{u'}.u'.W'$ and $lk' = \overline{u'}.L'$. After such a transition, no reader can read, as all the locks are busy. The other possible initial transition is $rd \oplus lk \xrightarrow{\tau} rd' \oplus lk'$, where $rd' = \mathrm{read}.u'.R'$. From place wr', one transition is derivable, namely $wr' \xrightarrow{\mathrm{write}} wr''$, where $wr'' = \underline{u'}.\underline{u'}.u'.W'$. From place rd', one transition is derivable, namely $rd' \xrightarrow{\mathrm{read}} rd''$, where $rd'' = u'.R'$. Finally, two further transitions are derivable: $rd'' \oplus lk' \xrightarrow{\tau} rd \oplus lk$ and $wr'' \oplus 3 \cdot lk' \xrightarrow{\tau} wr \oplus 3 \cdot lk$. The resulting P/T Petri net Net(CRW) is depicted in Fig. 2.

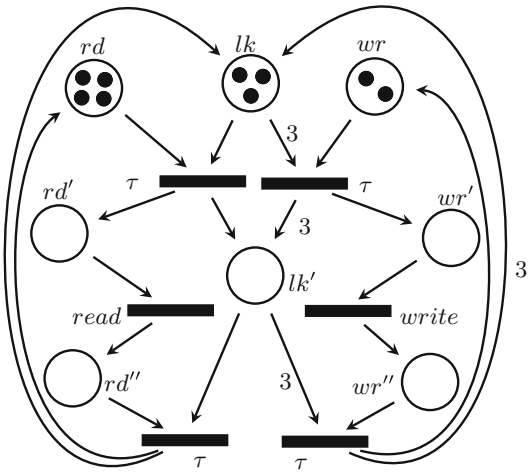

Fig. 2 The net for the concurrent readers/writers problem

5 Soundness

In this section, we prove that the operational net semantics is sound w.r.t. the operational LTS semantics: for any process $p \in \mathcal{P}$, the LTS rooted in p is bisimilar to the rooted LTS IMG(Net(dec(p))). First, some auxiliary lemmata.

Lemma 5 *Transition $t = (m, \sigma, m')$ is derivable by the rules in Table 9 if and only if transition $t' = (m\{a'/a\}, \sigma\{a'/a\}, m'\{a'/a\})$ is derivable by the rules.*

Proof By induction on the proof of t. □

Lemma 6 *Let $t_1 = (m_1, \sigma, m_1')$ be derivable by the rules in Table 9, and let p be a process such that* $\mathrm{dec}(p)[t_1\rangle\mathrm{dec}(p')$. *If $p \equiv q$, then there exists a transition $t_2 = (m_2, \sigma, m_2')$ such that* $\mathrm{dec}(q)[t_2\rangle\mathrm{dec}(q')$, *with $q' \equiv p'$.*

Proof By induction on the proof of $q \equiv p$ and then on the proof of t_1. The base cases are the three axioms in Table 4. For axiom **E1** (associativity), we have that $\mathrm{dec}(p \mid (q \mid r)) = \mathrm{dec}(p) \oplus \mathrm{dec}(q) \oplus \mathrm{dec}(r) = \mathrm{dec}((p \mid q) \mid r)$, so that transition t_2 is exactly t_1. Similarly, $\mathrm{dec}(p \mid q) = \mathrm{dec}(p) \oplus \mathrm{dec}(q) = \mathrm{dec}(q \mid p)$ and $\mathrm{dec}(A) = \mathrm{dec}(q)$ if $A \stackrel{\mathrm{def}}{=} q$. So, for the base cases, the thesis follows trivially. For substitutivity of prefixing, we assume that $p = \mu.p_1$, $q = \mu.q_1$ and $p_1 \equiv q_1$; in such a case, $t_1 = (\{\mu.p_1\}, \mu, \mathrm{dec}(p_1))$ is the only transition enabled at $\mathrm{dec}(p)$; the required transition t_2 is $(\{\mu.q_1\}, \mu, \mathrm{dec}(q_1))$. For substitutivity of strong prefixing, we assume that $p = \underline{\alpha}.p_1$, $q = \underline{\alpha}.q_1$ and $p_1 \equiv q_1$; in such a case, $t_1 = (\{\underline{\alpha}.p_1\}, \alpha \diamond \sigma', \mathrm{dec}(p_1'))$ is derivable if $t_1' = (\{p_1\}, \sigma', \mathrm{dec}(p_1'))$ is derivable; by induction, as $p_1 \equiv q_1$, there exists transition $t_2' = (\{q_1\}, \sigma', \mathrm{dec}(q_1'))$, with $p_1' \equiv q_1'$; then, by (s-pref), also transition $t_2 = (\{\underline{\alpha}.q_1\}, \alpha \diamond \sigma', \mathrm{dec}(q_1'))$ is derivable, as required. For substitutivity of choice, we assume that $p = p_1 + p_2$, $q = q_1 + q_2$ and $p_i \equiv q_i$, for $i =$

1, 2; in such a case, $t_1 = (\{p_1 + p_2\}, \sigma, \mathrm{dec}(p_1'))$ is derivable if (w.l.o.g., we assume p_1 moves) $t_1' = (\{p_1\}, \sigma, \mathrm{dec}(p_1'))$ is derivable by (sum_1); by induction, as $p_1 \equiv q_1$, there exists transition $t_2' = (\{q_1\}, \sigma, \mathrm{dec}(q_1'))$, with $p_1' \equiv q_1'$; then, by (sum_1), also transition $t_2 = (\{q_1 + q_2\}, \sigma, \mathrm{dec}(q_1'))$ is derivable, as required.

For substitutivity of parallel composition, we assume that $p = p_1 \mid p_2$, $q = q_1 \mid q_2$, $p_i \equiv q_i$ for $i = 1, 2$, and $\mathrm{dec}(p)[t_1\rangle\mathrm{dec}(p')$. We have three subcases: (i) $\mathrm{dec}(p_1)[t_1\rangle\mathrm{dec}(p_1')$, with $p' = p_1' \mid p_2$; or (ii) $\mathrm{dec}(p_2)[t_1\rangle\mathrm{dec}(p_2')$, with $p' = p_1 \mid p_2'$; or (iii) neither of the previous two cases, i.e., ${}^\bullet t_1$ is not contained in $\mathrm{dec}(p_1)$ or $\mathrm{dec}(p_2)$. In the first case, by induction (since $p_1 \equiv q_1$), there exists t_2 such that $\mathrm{dec}(q_1)[t_2\rangle\mathrm{dec}(q_1')$, with $p_1' \equiv q_1'$, and so $\mathrm{dec}(q)[t_2\rangle\mathrm{dec}(q_1' \mid q_2)$, with $q_1' \mid q_2 \equiv p_1' \mid p_2$; the second case is symmetric, hence omitted.

In the third case, there exist two transitions, say t_1' and t_2', with $l(t_1') = \sigma_1$, $l(t_2') = \sigma_2$ and $\mathrm{Sync}(\sigma_1, \sigma_2, \sigma)$, such that, by rule (s-com), t_1 is $t_1'|t_2' = ({}^\bullet t_1' \oplus {}^\bullet t_2', \sigma, t_1'^\bullet \oplus t_2'^\bullet)$. By using the three axioms of the structural congruence $\equiv$ at the top level only (that we know can be used safely), we can find two processes $\overline{p}_1, \overline{p}_2$ such that $p \equiv \overline{p}_1 \mid \overline{p}_2$, $\mathrm{dec}(\overline{p}_1)[t_1'\rangle\mathrm{dec}(\overline{p}_1')$, $\mathrm{dec}(\overline{p}_2)[t_2'\rangle\mathrm{dec}(\overline{p}_2')$. Since $p \equiv q$, we can find two processes $\overline{q}_1, \overline{q}_2$ such that $q \equiv \overline{q}_1 \mid \overline{q}_2$ and $\overline{p}_i \equiv \overline{q}_i$ for $i = 1, 2$. Then, induction can be applied to conclude that there exist two transitions t_1'' and t_2'' such that $\mathrm{dec}(\overline{q}_1)[t_1''\rangle\mathrm{dec}(\overline{q}_1')$, $\mathrm{dec}(\overline{q}_2)[t_2''\rangle\mathrm{dec}(\overline{q}_2')$, with $\overline{p}_i' \equiv \overline{q}_i'$ for $i = 1, 2$. So, by rule (s-com), transition $t_2 = t_1''|t_2'' = ({}^\bullet t_1'' \oplus {}^\bullet t_2'', \sigma, t_1''^\bullet \oplus t_2''^\bullet)$ is derivable; hence, $\mathrm{dec}(q) = \mathrm{dec}(\overline{q}_1) \oplus \mathrm{dec}(\overline{q}_2)[t_2\rangle\mathrm{dec}(\overline{q}_1' \mid \overline{q}_2')$, with $\overline{q}_1' \mid \overline{q}_2' \equiv \overline{p}_1' \mid \overline{p}_2' \equiv p'$.

For substitutivity of restriction, we assume that $p = (\nu a)p_1$, $q = (\nu a)q_1$, $p_1 \equiv q_1$, and $\mathrm{dec}(p)[t_1\rangle\,\mathrm{dec}(p')$, with $l(t_1) = \sigma$ and $a, \bar{a} \notin n(\sigma)$. Since $\mathrm{dec}(p) = \mathrm{dec}(p_1)\{a'/a\}$, t_1 has the form $(m_1\{a'/a\}, \sigma, m_1'\{a'/a\})$, and $t_1' = (m_1, \sigma, m_1')$ is derivable by Lemma 5; moreover, $\mathrm{dec}(p_1)[t_1'\rangle\mathrm{dec}(p_1')$, with $\mathrm{dec}(p') = \mathrm{dec}(p_1')\{a'/a\} = \mathrm{dec}((\nu a)p_1')$. By induction, as $p_1 \equiv q_1$, there exists $t_2' = (m_2, \sigma, m_2')$ such that $\mathrm{dec}(q_1)[t_2'\rangle\mathrm{dec}(q_1')$, with $q_1' \equiv p_1'$. By Lemma 5, also $t_2 = (m_2\{a'/a\}, \sigma, m_2'\{a'/a\})$ is derivable, with $\mathrm{dec}(q) = \mathrm{dec}(q_1)\{a'/a\}\,[t_2\rangle\,\mathrm{dec}(q_1')\{a'/a\} = \mathrm{dec}((\nu a)q_1')$, where $(\nu a)q_1' \equiv (\nu a)p_1'$ as required. □

Proposition 16 *For any process $p \in \mathcal{P}$, if $p \xrightarrow{\sigma} p'$ then there exist $t \in T_p$ and $p'' \equiv p'$ such that $\mathrm{dec}(p)[t\rangle\mathrm{dec}(p'')$ with $l(t) = \sigma$.*

Proof The proof is by induction on the proof of $p \xrightarrow{\sigma} p'$. The base case is axiom (Pref), hence $p = \mu.q$, $\sigma = \mu$ and $p' = q$. The thesis follows by noting that axiom (pref) ensures that $\mathrm{dec}(p) = \{\mu.q\} \xrightarrow{\mu} \mathrm{dec}(q) = \mathrm{dec}(p')$ is in T_p.

If rule (S-pref) is the last rule used to derive $p \xrightarrow{\sigma} p'$, then $p = \underline{\alpha}.q$, $q \xrightarrow{\sigma'} p'$ and $\sigma = \alpha \diamond \sigma'$. The inductive hypothesis on the premise of rule (S-pref) ensures that there exist a transition $t = (m, \sigma', m')$ and a process $p'' \equiv p'$ such that $\mathrm{dec}(q)[t\rangle\mathrm{dec}(p'')$ with $l(t) = \sigma'$. Since q must be sequential, then $m = \mathrm{dec}(q) = \{q\}$ and $m' = \mathrm{dec}(p'')$. Hence the thesis follows by rule (s-pref): $t = (\{q\}, \sigma', \mathrm{dec}(p''))$ implies $\mathrm{dec}(p) = \{\underline{\alpha}.q\} \xrightarrow{\alpha\diamond\sigma'} \mathrm{dec}(p'')$.

If rule (Sum_1) is the last rule applied to derive transition $p \xrightarrow{\sigma} p'$, then $p = p_1 + p_2$ and $p_1 \xrightarrow{\sigma} p'$. The inductive hypothesis on the premise of rule (Sum_1) ensures that there exist a transition $t = (m, \sigma', m')$ and a process $p'' \equiv p'$ such that $\mathrm{dec}(p_1)[t\rangle\mathrm{dec}(p'')$ with $l(t) = \sigma$. Since p_1 is sequential, $m = \mathrm{dec}(p_1) = \{p_1\}$ and $m' = \mathrm{dec}(p'')$. Hence, the thesis follows by rule (sum_1): t implies $\mathrm{dec}(p_1 + p_2) \xrightarrow{\sigma} \mathrm{dec}(p'')$. Symmetrically, if (Sum_2) is the last rule applied.

If rule (Par_1) is the last rule used to derive $p \xrightarrow{\sigma} p'$, then $p = p_1 \mid p_2$ and $p_1 \xrightarrow{\sigma} p_1'$. The inductive hypothesis on the premise of (Par_1) ensures that there exist a transition $t = (m_1, \sigma, m_1')$ and a process $p_1'' \equiv p_1'$ such that $\mathrm{dec}(p_1)[t\rangle\mathrm{dec}(p_1'')$ with $l(t) = \sigma$. Hence, the thesis then follows by additivity: $\mathrm{dec}(p_1 \mid p_2) = \mathrm{dec}(p_1) \oplus \mathrm{dec}(p_2)[t\rangle\mathrm{dec}(p_1'') \oplus \mathrm{dec}(p_2) = \mathrm{dec}(p_1'' \mid p_2)$, with $p_1'' \mid p_2 \equiv p_1' \mid p_2$. Symmetrically, if rule (Par_2) is the last rule applied.

If rule (S-Com) is the last rule used to derive $p \xrightarrow{\sigma} p'$, then $p = p_1 \mid p_2$, $p' = p_1' \mid p_2'$, $p_1 \xrightarrow{\sigma_1} p_1'$, $p_2 \xrightarrow{\sigma_2} p_2'$ and $\mathrm{Sync}(\sigma_1, \sigma_2, \sigma)$. The inductive hypothesis on the premises of (S-Com) ensures that there exist two transitions, t_1 and t_2, and two processes, $p_1'' \equiv p_1'$ and $p_2'' \equiv p_2'$, such that $\mathrm{dec}(p_1)[t_1\rangle\mathrm{dec}(p_1'')$ with $l(t_1) = \sigma_1$, and $\mathrm{dec}(p_2)[t_2\rangle\mathrm{dec}(p_2'')$ with $l(t_2) = \sigma_2$. Hence, by rule (s-com), t_1 and t_2 imply transition $t_1|t_2 = ({}^\bullet t_1 \oplus {}^\bullet t_2, \sigma, t_1^\bullet \oplus t_2^\bullet)$; note that we are sure that the executability of t_1 and t_2 on their respective markings ensures that also the compound transition $t_1|t_2$ is executable on the union of the two markings, i.e., $\mathrm{dec}(p_1) \oplus \mathrm{dec}(p_2)[t_1|t_2\rangle\mathrm{dec}(p_1'') \oplus \mathrm{dec}(p_2'')$. Hence: $\mathrm{dec}(p_1 \mid p_2)[t_1|t_2\rangle\mathrm{dec}(p_1'' \mid p_2'')$, with $p_1'' \mid p_2'' \equiv p_1' \mid p_2'$.

If rule (S-Res) is the last rule used to derive $p \xrightarrow{\sigma} p'$, then $p = (\nu a)p_1$, $p' = (\nu a)p_1'$, $p_1 \xrightarrow{\sigma} p_1'$ and $a, \bar{a} \notin n(\sigma)$. The inductive hypothesis on the premise of rule (S-Res) ensures that there exist a transition $t = (m, \sigma, m')$ and a process $p_1'' \equiv p_1'$ such that $\mathrm{dec}(p_1)[t\rangle\mathrm{dec}(p_1'')$ with $l(t) = \sigma$. Note that $\mathrm{dec}((\nu a)p_1) = \mathrm{dec}(p_1)\{a'/a\}$ for a' restricted. Note also that, by Lemma 5, if t is derivable by the net rules, then also transition $t' = (m\{a'/a\}, \sigma, m'\{a'/a\})$ is derivable by the net rules. Therefore, since $m\{a'/a\} \subseteq \mathrm{dec}(p_1)\{a'/a\}$, we have that $\mathrm{dec}((\nu a)p_1) = \mathrm{dec}(p_1)\{a'/a\}\,[t'\rangle\,\mathrm{dec}(p_1'')\{a'/a\} = \mathrm{dec}((\nu a)p_1'')$, with $(\nu a)p_1'' \equiv (\nu a)p_1'$.

If rule (Cong) is the last rule used to derive $p \xrightarrow{\sigma} p'$, then $p \equiv q, q \xrightarrow{\sigma} q'$ and $q' \equiv p'$. The inductive hypothesis on the premise ensures the existence of a transition $t' = (m', \sigma, m'')$

and a process $q'' \equiv q'$ such that $\mathrm{dec}(q)[t'\rangle\mathrm{dec}(q'')$ with $l(t') = \sigma$. By Lemma 6, there exists a transition t, with $l(t) = \sigma$, such that $\mathrm{dec}(p)[t\rangle\mathrm{dec}(p'')$ and $p'' \equiv q''$, hence, by transitivity, $p'' \equiv p'$. □

Proposition 17 *For any process $p \in \mathcal{P}$, if there exists $t \in T_p$ such that $\mathrm{dec}(p)[t\rangle\mathrm{dec}(p')$ with $l(t) = \sigma$, then $p \xrightarrow{\sigma} p'$.*

Proof By induction on the definition of $\mathrm{dec}(p, I)$ and then by induction on the proof of t. The base cases are empty. The first base case is $\mathrm{dec}(\mathbf{0}, I) = \emptyset$ and so no transition t is enabled. Similarly, the second base case is $p = A$, with $A \in I$ (hence, $\mathrm{dec}(A, I) = \emptyset$). The other cases follow.

$\mathrm{dec}(\mu.q, I) = \{\mu.q\}$. By axiom (pref), the only derivable transition is $t = (\{\mu.q\}, \mu, \mathrm{dec}(q))$. By axiom (Pref), $\mu.q \xrightarrow{\mu} q$, and so the thesis follows trivially.

$\mathrm{dec}(\underline{\alpha}.q, I) = \{\underline{\alpha}.q\}$. By rule (s-pref), a transition $t = (\{\underline{\alpha}.q\}, \alpha \diamond \sigma', \mathrm{dec}(q'))$ is derivable only if a transition $t' = (\{q\}, \sigma', \mathrm{dec}(q'))$ is derivable. The inductive hypothesis on the premise t' of the rule (s-pref) ensures that $q \xrightarrow{\sigma'} q'$. By rule (S-Pref), $\underline{\alpha}.q \xrightarrow{\alpha\diamond\sigma'} q'$, as required.

$\mathrm{dec}(p_1 + p_2, I) = \{p_1 + p_2\}$. By rule ($\mathrm{sum}_1$), transition $t = (\{p_1 + p_2\}, \sigma, \mathrm{dec}(p'))$ is derivable only if transition $t' = (\{p_1\}, \sigma, \mathrm{dec}(p'))$ is derivable. The inductive hypothesis on the premise t' of the rule (sum_1) ensures that $p_1 \xrightarrow{\sigma} p'$. By rule ($\mathrm{Sum}_1$), $p_1 + p_2 \xrightarrow{\sigma} p'$, as required. Symmetrically, if rule (sum_2) is used.

$\mathrm{dec}(p_1 \mid p_2, I) = \mathrm{dec}(p_1, I) \oplus \mathrm{dec}(p_2, I)$. Given a transition t, with $l(t) = \sigma$, such that $\mathrm{dec}(p_1 \mid p_2, I)[t\rangle\mathrm{dec}(p')$, three cases are possible. If t is enabled at $\mathrm{dec}(p_1, I)$ and so $\mathrm{dec}(p_1, I)[t\rangle\mathrm{dec}(p_1')$, then $p' = p_1' \mid p_2$ and, by induction, we know that $p_1 \xrightarrow{\sigma} p_1'$. Hence the thesis follows by rule (Par). Symmetrically, if t is enabled at $\mathrm{dec}(p_2)$. The third case is when transition t is such that ${}^\bullet t$ is not contained in $\mathrm{dec}(p_1, I)$ or in $\mathrm{dec}(p_2, I)$. In such a case, there exist two transitions, say t_1 and t_2, with $l(t_1) = \sigma_1$, $l(t_2) = \sigma_2$ and $\mathrm{Sync}(\sigma_1, \sigma_2, \sigma)$, such that, by rule (s-com), t is $t_1|t_2 = ({}^\bullet t_1 \oplus {}^\bullet t_2, \sigma, t_1{}^\bullet \oplus t_2{}^\bullet)$. The marking $\mathrm{dec}(p_1, I) \oplus \mathrm{dec}(p_2, I)$ can be equivalently represented as $\mathrm{dec}(q_1, I) \oplus \mathrm{dec}(q_2, I)$ such that $p_1 \mid p_2 \equiv q_1 \mid q_2$, $\mathrm{dec}(q_1, I)[t_1\rangle\mathrm{dec}(q_1')$ and $\mathrm{dec}(q_2, I)[t_2\rangle\mathrm{dec}(q_2')$. Then, induction can be applied to conclude that $q_1 \xrightarrow{\sigma_1} q_1'$ and $q_2 \xrightarrow{\sigma_2} q_2'$; hence, by rule (S-Com), also $q_1 \mid q_2 \xrightarrow{\sigma} q_1' \mid q_2'$ is derivable, and by rule (Cong), $p_1 \mid p_2 \xrightarrow{\sigma} q_1' \mid q_2'$, as required.

$\mathrm{dec}((\nu a)p_1, I) = \mathrm{dec}(p_1, I)\{a'/a\}$. A transition t, with $l(t) = \sigma \in \mathcal{A}$ and $a, \bar{a} \notin n(\sigma)$, is such that $\mathrm{dec}(p_1, I)\{a'/a\}[t\rangle\mathrm{dec}(p_1')\{a'/a\}$ if transition t has the form $(m\{a'/a\}, \sigma, m'\{a'/a\})$; so transition $t' = (m, \sigma, m')$ is derivable by Lemma 5 and $\mathrm{dec}(p_1, I)[t'\rangle\mathrm{dec}(p_1')$. By induction, we have $p_1 \xrightarrow{\sigma} p_1'$, and so by rule (S-Res), also $(\nu a)p_1 \xrightarrow{\sigma} (\nu a)p_1'$ is derivable, as required.

$\mathrm{dec}(A, I) = \mathrm{dec}(p, I \cup \{A\})$ if $A \stackrel{\mathrm{def}}{=} p$ and $A \notin I$. Then, if there exists t, with $l(t) = \sigma$, such that $\mathrm{dec}(A, I)[t\rangle\mathrm{dec}(p')$, then also $\mathrm{dec}(p, I \cup \{A\})[t\rangle\mathrm{dec}(p')$. By induction, we can assume that transition $p \xrightarrow{\sigma} p'$ is derivable. Hence, by rule (Cong), also $A \xrightarrow{\sigma} p'$ is derivable, too. □

We are now ready to state the soundness theorem: the interleaving marking graph associated to $Net(p)$ is bisimilar to the LTS rooted in p.

Theorem 5 (Soundness) *For any process $p \in \mathcal{P}$, $p \sim \mathrm{dec}(p)$.*

Proof If the relation $R = \{(p, \mathrm{dec}(q)) \mid p, q \in \mathcal{P} \wedge p \equiv q\}$ is a bisimulation, then the thesis follows trivially, as $p \equiv p$. On the one hand, if $p \xrightarrow{\sigma} p'$, then, by Proposition 16, there exist a transition t, with $l(t) = \sigma$, and a process p'', with $p'' \equiv p'$, such that $\mathrm{dec}(p)[t\rangle\mathrm{dec}(p'')$, and $(p', \mathrm{dec}(p'')) \in R$. On the other hand, if $\mathrm{dec}(q)[t\rangle\mathrm{dec}(q')$, with $l(t) = \sigma$, then, by Proposition 17, we have $q \xrightarrow{\sigma} q'$; as $p \equiv q$, by rule (Cong), $p \xrightarrow{\sigma} q'$, and $(q', \mathrm{dec}(q')) \in R$, as required.

6 A process term for any finite P/T net

In this section we address the following problem: given a finite, statically reduced, P/T Petri net system $N(m_0)$, labeled on $\mathcal{L} \cup \{\tau\}$, can we single out a finite-net multi-CCS process $p_{N(m_0)}$ such that $\mathrm{Net}(p_{N(m_0)})$ and $N(m_0)$ are isomorphic? The answer to this question is positive; hence, finite-net multi-CCS can represent all finite, statically reduced, P/T Petri nets, up to net isomorphism.

The translation from nets to processes we are going to present defines a constant C_i in correspondence to each place s_i; the definition of the constant C_i contains an addend composed of a new bound name y_i, which is used in order to distinguish syntactically all the constants bodies, so that no fusion of two constants to the same place is possible when applying the reverse step from the generated process term to its associated net (see Example 8). Moreover, the translation considers a bound name x_i^j for each pair (s_i, t_j), where s_i is a place and t_j is a transition; such bound names are used to synchronize all the components participating in transition t_j. The constant C_i, associated to place s_i, has a summand c_i^j for each transition t_j, which may be $\mathbf{0}$ when s_i is not in the pre-set of t_j. Among the many places in the pre-set of t_j, the one with minimal index (as we assume that places are indexed) plays the role of *leader* of the multi-party synchronization (i.e., the process performing the atomic sequence of inputs x_i^j to be synchronized with single outputs $\bar{x}_i^j$ performed by the other *servant* participants).

Definition 17 Let $N(m_0) = (S, A, T, m_0)$—with $S = \{s_1, \ldots, s_n\}$, $A \subseteq \mathcal{L} \cup \{\tau\}$, $T = \{t_1, \ldots, t_k\}$, and $l(t_j) = a_j$

for $j = 1, \ldots, k$—be a finite P/T net system. Function INet$(-)$, from finite P/T net systems to well-formed, finite-net multi-CCS processes is defined as

$$\mathrm{INet}(N(m_0)) = (\nu L)(\underbrace{C_1 | \cdots | C_1}_{m_0(s_1)} | \cdots | \underbrace{C_n | \cdots | C_n}_{m_0(s_n)})$$

where $L = \{y_1, \ldots, y_n\} \cup \{x_1^1, \ldots, x_n^1, x_1^2, \ldots, x_n^2, \ldots, x_1^k, \ldots, x_n^k\}$, and each C_i has a defining equation

$$C_i \stackrel{\mathrm{def}}{=} c_i^1 + \cdots + c_i^k + y_i.\mathbf{0}$$

where each c_i^j, for $j = 1, \ldots, k$, is equal to

- $\mathbf{0}$, if $s_i \notin {}^\bullet t_j$;
- $a_j.\Pi_j$, if ${}^\bullet t_j = \{s_i\}$;
- $\overline{x}_i^j.\mathbf{0}$, if ${}^\bullet t_j(s_i) > 0$ and ${}^\bullet t_j(s_{i'}) > 0$ for some $i' < i$ (i.e., s_i is not the leader for the synchronization on t_j);
- $\underbrace{\underline{x}_{i+1}^j.\cdots.\underline{x}_{i+1}^j}_{{}^\bullet t_j(s_{i+1})} . \cdots . \underbrace{\underline{x}_n^j.\cdots.\underline{x}_n^j}_{{}^\bullet t_j(s_n)}.a_j.\Pi_j$, if ${}^\bullet t_j(s_i) = 1$ and s_i is the leader of the synchronization (i.e., ${}^\bullet t_j(s_{i'}) > 0$ for no $i' < i$, while ${}^\bullet t_j(s_{i'}) > 0$ for some $i' > i$);
- $\overline{x}_i^j.\mathbf{0} + \underbrace{\underline{x}_i^j.\cdots.\underline{x}_i^j}_{{}^\bullet t_j(s_i)-1} . \underbrace{\underline{x}_{i+1}^j.\cdots.\underline{x}_{i+1}^j}_{{}^\bullet t_j(s_{i+1})} . \cdots . \underbrace{\underline{x}_n^j.\cdots.\underline{x}_n^j}_{{}^\bullet t_j(s_n)}.a_j.\Pi_j$, otherwise (i.e., s_i is the leader and ${}^\bullet t_j(s_i) \geq 2$).

Finally, process Π_j is defined as $\Pi_j = \underbrace{C_1 | \cdots | C_1}_{t_j^\bullet(s_1)} | \cdots | \underbrace{C_n | \cdots | C_n}_{t_j^\bullet(s_n)}$, meaning that $\Pi_j = \mathbf{0}$ if $t_j^\bullet = \emptyset$.

Example 7 Consider the net $N(m_0)$ of Fig. 3, where transition t_1 is labeled with a, t_2 with b and t_3 with c. Applying the translation above, we obtain the well-formed, finite-net multi-CCS process

$$\mathrm{INet}(N(m_0)) = (\nu L)(C_1 \mid C_1 \mid C_1 \mid C_2 \mid C_2)$$

where $L = \{y_1, y_2, y_3\} \cup \{x_1^1, x_2^1, x_3^1, x_1^2, x_2^2, x_3^2, x_1^3, x_2^3, x_3^3\}$, and

$$\begin{aligned} C_1 &\stackrel{\mathrm{def}}{=} (\overline{x}_1^1.\mathbf{0} + \underline{x}_1^1.a.C_1) + (\overline{x}_1^2.\mathbf{0} + \underline{x}_1^2.\underline{x}_1^2.\underline{x}_2^2.b.\mathbf{0}) \\ &\quad + \underline{x}_2^3.\underline{x}_2^3.c.C_3 + y_1.\mathbf{0} \\ C_2 &\stackrel{\mathrm{def}}{=} \mathbf{0} + \overline{x}_2^2.\mathbf{0} + \overline{x}_2^3.\mathbf{0} + y_2.\mathbf{0} \\ C_3 &\stackrel{\mathrm{def}}{=} \mathbf{0} + \mathbf{0} + \mathbf{0} + y_3.\mathbf{0} \end{aligned}$$

Note that INet$(N(m_0))$ is a *finite-net multi-CCS process*: in fact, the restriction operator occurs only at the top level, applied to the parallel composition of a number of constants; each constant has a body that is sequential and restriction-free. Note also that INet$(N(m_0))$ is a *well-formed* process: in fact, each *strong prefix* is an input x_i^j, and any sequence ends with an action $a_j \in A$ which is either an input or τ; hence, no synchronization of sequences is possible. Therefore, the following proposition holds by Theorem 4 and Proposition 12.

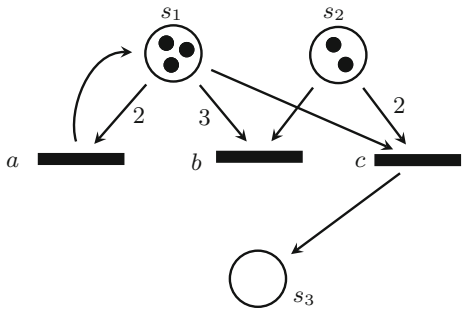

Fig. 3 A simple net

Proposition 18 *For any finite P/T Petri net $N(m_0) = (S, A, T, m_0)$, the net Net*(INet$(N(m_0))$) *is a finite, statically reduced, P/T net.*

Example 8 In order to explain the role of addend y_i in the body of constant C_i, for $i = 1, \ldots, n$, let us assume that they are omitted in Definition 17. Now, let us consider the net $N(\{s_1, s_2\}) = (\{s_1, s_2, s_3, s_4\}, \{a\}, \{(s_1, a, s_3), (s_2, a, s_4)\}, \{s_1, s_2\})$, which has four places and two transitions. The finite-net multi-CCS term INet$(N(\{s_1, s_2\}))$ would be $(\nu L)(C_1 \mid C_2)$, where $L = \{x_1^1, x_2^1, x_1^2, x_2^2\}$ and

$$\begin{aligned} C_1 &\stackrel{\mathrm{def}}{=} a.C_3 + \mathbf{0} & C_2 &\stackrel{\mathrm{def}}{=} \mathbf{0} + a.C_4 \\ C_3 &\stackrel{\mathrm{def}}{=} \mathbf{0} + \mathbf{0} & C_4 &\stackrel{\mathrm{def}}{=} \mathbf{0} + \mathbf{0} \end{aligned}$$

but now Net(INet$(N(\{s_1, s_2\}))$) is the net $(\{a.C_3 + \mathbf{0}, \mathbf{0} + a.C_4, \mathbf{0} + \mathbf{0}\}, \{a\}, \{(a.C_3 + \mathbf{0}, a, \mathbf{0} + \mathbf{0}), (\mathbf{0} + a.C_4, a, \mathbf{0} + \mathbf{0})\}, \{a.C_3 + \mathbf{0}, \mathbf{0} + a.C_4\})$, which has three places only, as the two distinct places s_3 and s_4 are now mapped to the same place $\mathbf{0} + \mathbf{0}$. This fusion cannot happen when we include the additional addend y_i in the body of each constant C_i.

Now we are ready to state our main result, the so-called *representability theorem*.

Theorem 6 *(Representability theorem) Let $N(m_0) = (S, A, T, m_0)$ be a finite, statically reduced, P/T net system such that $A \subseteq \mathcal{L} \cup \{\tau\}$, and let $p =$* INet$(N(m_0))$. *Then, Net(p) is isomorphic to $N(m_0)$.*

Proof Let $N(m_0) = (S, A, T, m_0)$ be a finite, statically reduced, P/T net system, with $S = \{s_1, \ldots, s_n\}$, $A \subseteq \mathcal{L} \cup \{\tau\}$,

$T = \{t_1, \ldots, t_k\}$ and $l(t_j) = a_j$ for $j = 1, \ldots, k$. The associated finite-net multi-CCS process is

$$\mathrm{INet}(N(m_0)) = (\nu L)(\underbrace{C_1|\cdots|C_1}_{m_0(s_1)}|\cdots|\underbrace{C_n|\cdots|C_n}_{m_0(s_n)})$$

where $L = \{y_1, \ldots, y_n\} \cup \{x_1^1, \ldots, x_n^1, x_1^2, \ldots, x_n^2, \ldots, x_1^k, \ldots, x_n^k\}$, and for each place s_i we have a corresponding constant $C_i \stackrel{\mathrm{def}}{=} (\sum_{j=1}^{k} c_i^j) + y_i.\mathbf{0}$, defined as in Definition 17. For notational convenience, $(\sum_{j=1}^{k} c_i^j) + y_i.\mathbf{0}$ is denoted by p_i, i.e., $C_i \stackrel{\mathrm{def}}{=} p_i$; for the same reason, we use p to denote $\mathrm{INet}(N(m_0))$.

Let $\rho = \{L'/L\}$ be a substitution that maps each bound name x_i^j (or y_i) to its corresponding restricted name $x_i'^j$ (or y_i') in $\mathcal{L}'$, for $i = 1, \ldots, n$ and $j = 1, \ldots, k$. Let $\mathrm{Net}(p) = (S', A', T', m_0')$. Then, $m_0' = \mathrm{dec}(p)$ is the multiset

$$\begin{aligned}&\mathrm{dec}((\nu L)(\underbrace{C_1|\cdots|C_1}_{m_0(s_1)}|\cdots|\underbrace{C_n|\cdots|C_n}_{m_0(s_n)}))\\ &= \mathrm{dec}(\underbrace{C_1|\cdots|C_1}_{m_0(s_1)}|\cdots|\underbrace{C_n|\cdots|C_n}_{m_0(s_n)})\rho\\ &= m_0(s_1)\cdot p_1\rho \oplus \cdots \oplus m_0(s_n)\cdot p_n\rho.\end{aligned}$$

because $C_i \stackrel{\mathrm{def}}{=} p_i$ for $i = 1, \ldots n$ and so $\mathrm{dec}(C_i) = \{p_i\}$. Hence, the initial places are all of the form $p_i\rho$, where such a place is present in m_0' only if $m_0(s_i) > 0$.

Note that, by Definition 17, any transition $t' \in T'$ is such that $t'^{\bullet} = \mathrm{dec}(\Pi_j)$ for some suitable j, so that each statically reachable place s_i' in S' is of the form $p_i\rho$, which are all distinct because each p_i contains one distinguishing summand $y_i.\mathbf{0}$. Hence, there is a bijection $f : S \to S'$ defined by $f(s_i) = s_i' = p_i\rho$, which is the natural candidate isomorphism function. To prove that f is an isomorphism, we have to prove that:

1. $f(m_0) = m_0'$,
2. $t = (m, a, m') \in T$ implies $f(t) = (f(m), a, f(m')) \in T'$, and
3. $t' = (m_1', a, m_2') \in T'$ implies there exists $t = (m_1, a, m_2) \in T$ such that $f(t) = t'$, i.e., $f(m_1) = m_1'$ and $f(m_2) = m_2'$.

From items (2) and (3) above, it follows that $A = A'$.

Proof of 1: Let $m_0 = k_1 \cdot s_1 \oplus k_2 \cdot s_2 \oplus \cdots \oplus k_n \cdot s_n$, where $k_i = m_0(s_i) \geq 0$ for $i = 1, \ldots, n$. The mapping via f of the initial marking m_0 is $f(m_0) = k_1 \cdot f(s_1) \oplus k_2 \cdot f(s_2) \oplus \cdots \oplus k_n \cdot f(s_n) = k_1 \cdot p_1\rho \oplus k_2 \cdot p_2\rho \oplus \cdots \oplus k_n \cdot p_n\rho = \mathrm{dec}(\underbrace{C_1|\cdots|C_1}_{k_1 \text{ times}}|\cdots|\underbrace{C_n|\cdots|C_n}_{k_n \text{ times}})\rho = \mathrm{dec}(p) = m_0'$.

Proof of 2: we prove that, for $j = 1, \ldots, k$, if $t_j = (m, a, m') \in T$, then $t_j' = (f(m), a, f(m')) \in T'$. From transition t_j, we can derive the two processes $P_j = (\underbrace{C_1\rho|\cdots|C_1\rho}_{{}^{\bullet}t_j(s_1)}|\cdots|\underbrace{C_n\rho|\cdots|C_n\rho}_{{}^{\bullet}t_j(s_n)})$ and $P_j' = (\underbrace{C_1\rho|\cdots|C_1\rho}_{t_j^{\bullet}(s_1)}|\cdots|\underbrace{C_n\rho|\cdots|C_n\rho}_{t_j^{\bullet}(s_n)})$ such that $f({}^{\bullet}t_j) = \mathrm{dec}(P_j)$ and $f(t_j^{\bullet}) = \mathrm{dec}(P_j')$. According to Definition 17, for each $C_i = p_i$, we have a summand c_i^j in p_i, with $Q_j = (\underbrace{c_1^j\rho|\cdots|c_1^j\rho}_{{}^{\bullet}t_j(s_1)}|\cdots|\underbrace{c_n^j\rho|\cdots|c_n^j\rho}_{{}^{\bullet}t_j(s_n)})$. By inspecting the shape of t_j and the definition of the various c_i^j's, one can get convinced that $(\mathrm{dec}(Q_j), l(t_j), \mathrm{dec}(P_j'))$ is a derivable transition. Hence, since each p_i is a summation containing the summand c_i^j, also $(\mathrm{dec}(P_j), l(t_j), \mathrm{dec}(P_j'))$ is a derivable transition and belongs to T', as required.

Proof of 3: We prove that if $t_j' = (m_1', a, m_2') \in T'$, then there exists a transition $t_j = (m_1, a, m_2) \in T$ such that $f(m_1) = m_1'$ and $f(m_2) = m_2'$. This is proved by case analysis on the shape of the marking m_1'.

If m_1' is a singleton, then $m_1' = \{p_i\rho\}$ for some $i = 1, \ldots, n$, and so $t_j' = \{p_i\rho\} \xrightarrow{a} m_2'$. According to Definition 17, such a transition is derivable by the rules only if, among the many summands composing $p_i\rho$, there exists a summand $c_i^j\rho = a.\Pi_j\rho$, which is possible only if in $N(m_0)$ we have a transition t_j with ${}^{\bullet}t_j = \{s_i\}$, $f(\{s_i\}) = \{p_i\rho\}$, $f(t_j^{\bullet}) = \mathrm{dec}(\Pi_j\rho) = m_2'$ and $l(t_j) = a$, as required.

Otherwise, if $m_1' = k_1 \cdot p_1\rho \oplus \cdots \oplus k_n \cdot p_n\rho$ and i is the least index such that $k_i > 0$, then in deriving transition $t_j' = m_1' \xrightarrow{a} m_2'$, one of the k_i processes $p_i\rho$ acts as the leader of the synchronization, and all the other participants are servants. If $k_i = 1$, then, by Definition 17, p_i has a summand c_i^j defined as $\underbrace{\underline{x}_{i+1}^j.\cdots.\underline{x}_{i+1}^j}_{k_{i+1} \text{ times}}.\cdots.\underbrace{\underline{x}_n^j.\cdots.\underline{x}_n^j}_{k_n \text{ times}}.a_j.\Pi_j$, where $\Pi_j = \underbrace{C_1|\cdots|C_1}_{h_1 \text{ times}}|\cdots|\underbrace{C_n|\cdots|C_n}_{h_n \text{ times}}$..

This summand c_i^j will synchronize with all the other components of m_1', as each other may only contribute with an action of the form $\overline{x}_k^j$ for $k = i+1, \ldots, n$, being the unique synchronizable summand of p_k. Therefore, transition t_j' is possible only if transition $t_j = (m_1, a, m_2)$ is in T, where $m_1 = k_1 \cdot s_1 \oplus \cdots \oplus k_n \cdot s_n$, with $f(m_1) = m_1'$, and $m_2 = h_1 \cdot s_1 \oplus \cdots \oplus h_n \cdot s_n$ is such that $f(m_2) = \underbrace{C_1\rho|\cdots|C_1\rho}_{h_1 \, times}|\cdots|\underbrace{C_n\rho|\cdots|C_n\rho}_{h_n \, times}$, as required.

Similarly, if $k_i \geq 2$, then p_i has a summand c_i^j defined as

$\overline{x}_i^j.\mathbf{0} + \underbrace{\underline{x}_i^j.\cdots.\underline{x}_i^j}_{k_i-1 \text{ times}} . \underbrace{\underline{x}_{i+1}^j.\cdots.\underline{x}_{i+1}^j}_{k_{i+1} \text{ times}} .\cdots. \underbrace{\underline{x}_n^j.\cdots.\underline{x}_n^j}_{k_n \text{ times}} .a_j.\Pi_j$, such that the other $k_i - 1$ instances of p_i can contribute to a synchronization with the first p_i by means of the summand $\overline{x}_i^j.\mathbf{0}$ in c_i^j. □

Example 9 Function INet(−) can be applied to any finite P/T net $N(m_0)$. However, if $N(m_0)$ is not *statically reduced*, the representability theorem does not hold. Let us consider the net $N(\{s_1\}) = (\{s_1, s_2\}, \{a\}, \{(s_1, a, \emptyset), (s_2, a, \emptyset)\}, \{s_1\})$. Clearly such a net is not statically reduced because place s_2 is not statically reachable from the initial marking. The finite-net multi-CCS term INet($N(\{s_1\})$) would be $(\nu L)(C_1)$, where $L = \{y_1, y_2\} \cup \{x_1^1, x_2^1, x_1^2, x_2^2\}$ and

$$C_1 \stackrel{\text{def}}{=} a.\mathbf{0} + \mathbf{0} + y_1.\mathbf{0} \qquad C_2 \stackrel{\text{def}}{=} \mathbf{0} + a.\mathbf{0} + y_2.\mathbf{0}$$

but now Net(INet($N(\{s_1\})$)) is the net $(\{a.\mathbf{0}+\mathbf{0}+y_1.\mathbf{0}\}, \{a\}, \{(a.\mathbf{0} + \mathbf{0} + y_1.\mathbf{0}, a, \emptyset)\}, \{a.\mathbf{0} + \mathbf{0} + y_1.\mathbf{0}\})$, which has one place and one transition only, i.e., it is isomorphic to the subnet of $N(\{s_1\})$ statically reachable from the initial marking $\{s_1\}$.

Remark 3 In the classic definition of Petri nets (see, e.g., [7,30,31]), the transition labeling is given with actions taken from a set A of *unstructured* actions; hence, our assumption that $A \subseteq \mathcal{L} \cup \{\tau\}$ is in analogy with this tradition.

However, if we want to be more generous and consider Petri nets labeled over the set $\text{Act} = \mathcal{L} \cup \overline{\mathcal{L}} \cup \{\tau\}$ of *structured* actions and co-actions, the extension of the representability theorem to this larger class of nets is not trivial. First of all, we note that the translation in Definition 17 is no longer accurate; consider the Petri net $N(\{s_1, s_2\}) = (\{s_1, s_2\}, \{a, \bar{a}\}, \{(s_1, a, \emptyset), (s_2, \bar{a}, \emptyset)\}, \{s_1, s_2\})$, then INet($N(\{s_1, s_2\})$) is $(\nu L)(C_1 \mid C_2)$, with $L = \{y_1, y_2\} \cup \{x_1^1, x_2^1, x_1^2, x_2^2\}$ and

$$C_1 \stackrel{\text{def}}{=} a.\mathbf{0} + \mathbf{0} + y_1.\mathbf{0} \qquad C_2 \stackrel{\text{def}}{=} \mathbf{0} + \bar{a}.\mathbf{0} + y_2.\mathbf{0},$$

but now Net(INet($N(\{s_1, s_2\})$)) contain also an additional synchronization transition $(\{a.\mathbf{0} + \mathbf{0} + y_1.\mathbf{0}, \mathbf{0} + \bar{a}.\mathbf{0} + y_2.\mathbf{0}\}, \tau, \emptyset)$, which has no counterpart in the original net $N(\{s_1, s_2\})$. We conjecture that a possible solution is to be based on the introduction an additional operator in the language: the relabeling operator of CCS [25]—$[b_1/a_1, \ldots, b_n/a_n]$, which relabels each action a_i into b_i—to be used only at the top level. The procedure is as follows:

- First, relabel each transition t_j of the original net $N(m_0)$, labeled with an input action a_j, to a new, not in use, input action a_j^j, yielding a new renamed net $N'(m_0)$.
- Then, compute the associated process INet($N'(m_0)$), according to Definition 17; note that in Net(INet($N'(m_0)$)) no additional synchronization transitions are introduced, because, by renaming, no pair of transitions in $N'(m_0)$ are labeled with a matching pair of actions/co-actions.
- Then, consider the process INet($N'(m_0)$)$[a_1/a_1^1, \ldots, a_k/a_k^k]$ and compute its associated net: it will be isomorphic to the original net $N(m_0)$. □

7 Conclusion

The class of finite-net multi-CCS processes represents a language for describing finite, statically reduced, P/T Petri nets. This is not the only language expressing finite P/T nets: the first (and only other) one is Mayr's PRS [22], which however is rather far from a typical process algebra as its basic building blocks are rewrite rules (or net transitions) instead of actions and, for instance, it does not contain any scope operator like restriction.

A bit pretentiously, we claim that well-formed, finite-net Multi-CCS is *the* language for finite Petri nets. The main argument defending this claim is that the parallel operator $- \parallel -$ of a language able to express Petri nets has to be

- *permissive*: in a process $p \parallel q$, the actions p can perform cannot be prevented by q. This requirement is necessary because P/T Petri nets are permissive as well, meaning that if a transition t is enabled at a marking m, then t is also enabled at a marking $m' \supseteq m$; the parallel operator of Multi-CCS is permissive, while this is not the case for other parallel operators, such as the CSP one $p \parallel_A q$ [20];
- Moreover, the parallel operator $- \parallel -$ is to be ACI (*associative, commutative, with an identity*), because the decomposition of a parallel process into a marking has to reflect that a marking is a (finite) *multiset*; also in this case, the parallel operator of Multi-CCS is ACI, while this is not the case for other parallel operators, notably the CSP one.
- Moreover, the parallel operator should be able to express multi-party synchronization, because a net transition, which may have a preset of any size, can be generated by means of a synchronization among many participants, actually as many as are the tokens in its preset. The Multi-CCS parallel operator can model multi-party synchronization, by means of the interplay with the strong prefixing operator. Other process algebras offer parallel operators with multi-party synchronization capabilities, but in Multi-CCS multi-party synchronization is "programmable", meaning that we can prescribe the order in which the various participants are to interact, independently of the syntactic position they occupy within the global term and without resorting to a global synchro-

nization function, as in the case of some ACP dialects [2].

The multi-party synchronization discipline has been chosen as simple as possible: a sequence can synchronize with a complementary action at a time, in the exact order they occur in the sequence. About sequentialization operators, we note that prefixing cannot be replaced by ACP-like sequential composition, because a language with recursion and sequential composition can express all the context-free languages, while finite P/T nets cannot [19,30]; hence, sequential composition is too powerful to express only finite P/T nets. As we have chosen a CCS-like naming convention, the scoping operator, which can occur syntactically only at the top level, is the CCS restriction operator.

Summing up, any other language, if any, able to represent all and only finite P/T Petri nets should possess these necessary features, which, altogether, seem to be exclusive of finite-net Multi-CCS, or that at least are very rare in the panorama of process algebras.

Our calculus is given a net semantics in terms of *unsafe*, finite P/T nets, improving over previous work. Degano et al. [8] and Olderog's approach [28] is operational like ours, but somehow complementary in style, as it builds directly over the SOS semantics of CCS. Their construction generates *safe* P/T nets which are finite only for regular CCS processes (i.e., processes where restriction and parallel composition cannot occur inside recursion). On the contrary, here we give finite P/T net semantics to a calculus strictly larger than regular CCS, as parallel composition can occur in the body of recursively defined constants. Similar concerns are for PBC [4], whose semantics is given in terms of *safe* P/T nets only. Nonetheless, PBC can express multiway synchronization by means of its relabeling operators, and so, in principle, if equipped with an unsafe semantics, it might also serve as a language expressing all unsafe, finite P/T nets. The first paper defining a net semantics in terms of *unsafe* P/T nets is [13], where the approach is denotational and the considered language is limited to CCS without restriction. Our technique is somehow indebted to the earlier work of Busi and Gorrieri [5] on giving labeled net semantics to the π-calculus [26] in terms of P/T nets with inhibitor arcs; our solution simplifies their approach for finite-net Multi-CCS because we do not need inhibitors. In particular, already in that paper it is observed that finite-net π-calculus processes originate finite P/T net systems (with inhibitor arcs). Similar observations on the interplay between parallel composition and restriction in recursive definitions, in different contexts, has been done also by others, e.g., [1]. Also important is the work of Meyer [23,24] in providing an unlabeled P/T net semantics for a fragment of the π-calculus; the main difference is that his semantics may offer a finite net representation also for some processes where restriction occurs inside recursion, but the price to pay is that the resulting net semantics may be incorrect from a causality point of view; for instance, in process $(\nu c)(a.c.\mathbf{0} \mid b.\bar{c}.\mathbf{0}) \mid (\bar{a}.\mathbf{0} \mid \bar{b}.\mathbf{0})$ the two synchronizations on a and b are causally dependent in his semantics.

Denotational net semantics for unsafe Petri nets are rare. Besides the work by Goltz [13] mentioned above, we know also of [3], where CSP [20] is given a denotational net semantics in terms of so-called *open* nets, a reactive extension of ordinary Petri nets, with the limitation that parallel composition is modeled by disjoint union and arc weight can only be 1. Future work will be devoted to define compositional (i.e., denotational in style) unsafe P/T net semantics for finite-net Multi-CCS, generalizing work of Goltz [13] and Taubner [33].

We conclude this overview of related literature by noting the differences of this paper with respect to its earlier version [18]. First, the definition of finite-net Multi-CCS is a bit simpler now, in order to capture the minimal language capable of representing all and only finite P/T nets. Second, the net $\mathrm{Net}(p)$ associated to a process p is statically reduced: this ensures that $\mathrm{Net}(p)$ and $\mathrm{Net}(p \mid p)$ are the same unmarked net, but with a different initial marking; on the contrary, in [18] $\mathrm{Net}(p)$ was only dynamically reduced. Third, the finiteness theorem was wrongly stated in [18]: in fact, $\mathrm{Net}(p)$ is finite not for all finite-net processes, but only for *well-formed* finite-net processes. Fourth, the construction of the finite-net process $p = \mathrm{INet}(N(m_0))$ from the finite P/T net system $N(m_0)$ is inaccurate in [18], as $\mathrm{Net}(p)$ may have more transitions than $N(m_0)$; as the previous construction used too few bound names, it was impossible to link precisely the tokens consumed by a transition to the actual place from which these tokens are to be consumed.

Finally, an open problem for future research. Compositional equivalence-checking on finite-state process algebras, such as regular CCS, is a viable technique because the used equivalence (typically, some form of bisimilarity over finite-state LTSs) is decidable and also a congruence for the operators of regular CCS. In desire of a similar technique for finite-net Multi-CCS, one has to find an equivalence relation which is decidable over finite P/T nets and a congruence for the operators of Multi-CCS. This is not easy. Let us examine three well-known equivalences over finite P/T nets: interleaving bisimilarity (Definition 7), step bisimilarity [27] and net isomorphism (Definition 5). On the one hand, only net isomorphism is decidable for finite P/T nets, while interleaving bisimilarity and step bisimilarity are undecidable [12,21]. On the other hand, net isomorphism is not a congruence for $+$: for instance, $p = a.(\mathbf{0} + \mathbf{0})$ and $q = a.(\mathbf{0} + \mathbf{0} + \mathbf{0})$ are such that $\mathrm{Net}(p) \cong \mathrm{Net}(q)$ (both nets have only two places and one transition), but $\mathrm{Net}(p + p) \ncong \mathrm{Net}(q + p)$, as $\mathrm{Net}(p + p)$ is isomorphic to $\mathrm{Net}(p)$, while $\mathrm{Net}(q + p)$ has three places and two transitions. Moreover, interleaving bisimilarity is not a congruence for Multi-CCS parallel com-

position [19], while step bisimilarity is a congruence for all the operators of Multi-CCS [19]. Summing up, none of these three equivalences satisfies both properties: being decidable and a congruence. Of course, if we restrict our attention to finite, *bounded* nets, the situation is much better and step bisimilarity can be used to this aim, being decidable and a congruence. However, for general, unbounded finite P/T nets it is a challenging open problem to find an equivalence relation which satisfies both properties.

Acknowledgments The anonymous referees are thanked for their detailed comments and suggestions. Massimo Morara is thanked for pointing out the inaccuracy in the definition of the process INet($N(m_0)$) in [18].

References

1. Aranda, J., Valencia, F., Versari, C.: On the expressive power of restriction and priorities in CCS with replication. In: Proceedings of the FOSSACS 2009. LNCS, vol. 5504, pp. 242–256. Springer, New York (2009)
2. Baeten, J.C.M., Basten, T., Reniers, M.A.: Process algebra: equational theories of communicating processes. In: Cambridge Tracts in Theoretical Computer Science, vol. 50. Cambridge University Press, Cambridge (2010)
3. Baldan, P., Bonchi, F., Gadducci, F., Monreale, G.: Encoding synchronous interactions using labelled Petri nets. In: Proceedings of the Coordination'14. LNCS, vol. 8459, pp. 1–16. Springer, New York (2014)
4. Best, E., Devillers, R., Koutny, M.: The box algebra = Petri nets + process expressions. Inf. Comput. **178**(1), 44–100 (2002)
5. Busi, N., Gorrieri, R.: Distributed semantics for the π-calculus based on Petri nets with inhibitor arcs. J. Logic Algebraic Program. **78**(3), 138–162 (2009)
6. Courtois, P., Heymans, F., Parnas, D.: Concurrent control with readers and writers. Commun. ACM **14**(10), 667–668 (1971)
7. Desel, J., Reisig, W.: Place/Transition Petri Nets. In: Reisig, W., Rozenberg, G. (eds.) Lectures on Petri Nets I: Basic Models. LectureNotes in Computer Science, vol. 1491, pp. 122–173. Springer, New York (1998)
8. Degano, P., De Nicola, R., Montanari, U.: A distributed operational semantics for CCS based on C/E systems. Acta Inform. **26**(1–2), 59–91 (1988)
9. Degano, P., De Nicola, R., Montanari, U.: Partial ordering descriptions and observations of nondeterministic concurrent systems. In: Lecture Notes in Computer Science, vol. 354, pp. 438–466. Springer, New York (1989)
10. Degano, P., Gorrieri, R., Marchetti, S.: An exercise in concurrency: a CSP process as a condition/event system. Adv. Petri Nets (LNCS, Springer) **340**, 85–105 (1988)
11. Degano, P., Meseguer, J., Montanari, U.: Axiomatizing the algebra of net computations and processes. Acta Inform. **33**(7), 641–667 (1996)
12. Esparza, J.: Decidability and complexity of Petri net problems: an introduction. In: Reisig, W., Rozenberg, G. (eds.) Lectures on Petri Nets I: Basic Models. LectureNotes in Computer Science, vol. 1491, pp. 374–428. Springer, New York (1998)
13. Goltz, U.: On representing CCS programs by finite Petri nets. In: Proceedings of the MFCS'88. LNCS, vol. 324, pp. 339–350. Springer, New York (1988)
14. Gorrieri, R.: Language representability of finite P/T nets. In:Programming Languages with Applications to Biology and Security—Colloquium in Honour of Pierpaolo Degano for His 65th Birthday, (PLABS 2015). LNCS, vol. 9465. Springer, New York (2015) (being printed)
15. Gorrieri, R., Montanari, U.: SCONE: a simple calculus of nets. In: Proceedings of the CONCUR'90. LNCS, vol. 458, pp. 2–30. Springer, New York (1990)
16. Gorrieri, R., Montanari, U.: Towards hierarchical specification of systems: a proof system for strong prefixing. Int. J. Found. Comput. Sci. **1**(3), 277–293 (1990)
17. Gorrieri, R., Marchetti, S., Montanari, U.: A^2CCS: atomic actions for CCS. Theor. Comput. Sci. **72**(2–3), 203–223 (1990)
18. Gorrieri, R., Versari, C.: A process calculus for expressing finite place/transition Petri nets. In: Proceedings of the EXPRESS'10, EPTCS, 2010. doi:10.4204/EPTCS.41.6. arXiv:1011.6433v1
19. Gorrieri, R., Versari, C.: EATCS Text in Computer Science. Introduction to concurrency theory: transition systems and CCS. Springer, New York (2015)
20. Hoare, C.A.R.: Communicating Sequential Processes. Prentice-Hall, New York (1985)
21. Jančar, P.: Undecidability of bisimilarity for Petri nets and some related problems. Theor. Comput. Sci. **148**(2), 281–301 (1995)
22. Mayr, R.: Process rewrite systems. Inf. Comput. **156**(1–2), 264–286 (2000)
23. Meyer, R.: A theory of structural stationarity in the π-calculus. Acta Inform. **46**(2), 87–137 (2009)
24. Meyer, R., Gorrieri, R.: On the relationship between pi-calculus and finite place/transition Petri nets. In: Proceedings of the CONCUR 2009. LNCS, vol. 5710, pp. 463–480. Springer, New York (2009)
25. Milner, R.: Communication and Concurrency. Prentice-Hall, New York (1989)
26. Milner, R.: Communicating and Mobile Systems: Theπ-Calculus. Cambridge University Press, Cambridge (1999)
27. Nielsen, M., Thiagarajan, P.S.: Degrees of non-determinism and concurrency: a Petri net view. In: Proceedings of the Fourth Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'84). LNCS, vol. 181, pp. 89–117. Springer, New York (1984)
28. Olderog, E.R.: Nets, terms and formulas. In: Cambridge Tracts in Theoretical Computer Science, vol. 23. Cambridge University Press, Cambridge (1991)
29. Pomello, L., Rozenberg, G., Simone, C.: A survey of equivalence notions for net based systems. Lect. Notes Comput. Sci. **609**, 410–472 (1992)
30. Peterson, J.L.: Petri Net Theory and the Modeling of Systems. Prentice-Hall, New York (1981)
31. Reisig, W.: EATCS Monographs on TCS. Petri Nets: An Introduction. Springer, New York (1985)
32. Reisig, W., Rozenberg, G. (eds.) Lectures on Petri Nets I: Basic Models. Lecture Notes in Computer Science, vol. 1491. Springer, New York (1998)
33. Taubner, D.: Finite representations of CCS and TCSP programs by automata and Petri nets. In: Lecture notes in computer science, vol. 369. Springer, New York (1989)