ELSEVIER

Theoretical Computer Science 141 (1995) 195–252

Theoretical Computer Science

On the implementation of concurrent calculi in net calculi: two case studies☆

Roberto Gorrieri[a,*], Ugo Montanari[b]

[a] *Dipartimento di Matematica, Università di Bologna, Piazza di Porta S. Donato 5, I-40100 Bologna, Italy*
[b] *Dipartimento di Informatica, Università di Pisa, Corso Italia 40, I-56125 Pisa, Italy*

Received April 1992; revised February 1994
Communicated by G. Rozenberg

Abstract

Concurrent calculi, such as CCS, are defined in terms of labelled transition system; similarly, here we introduce the notion of *net* (or *distributed*) *calculus*, which is defined through a place/transition Petri net. As a first case study, a simple calculus of nets, called SCONE, is proposed. Relationships between SCONE and the subset of CCS without restriction and relabelling, called RCCS, are studied by showing that RCCS can be implemented in SCONE through a suitable mapping from the transition system for RCCS to net for SCONE. In particular, the complex CCS operation of global choice is implemented in terms of the SCONE finer grained operation of local choice, making explicit the fact that certain CCS transition are implemented as SCONE computations to be executed atomically. The result is a finite net implementation for RCCS agents. By making the quotient of the RCCS transition system w.r.t. the implementation mapping, we induce a truly concurrent semantics for RCCS. The second case study is then concerned with an extension dealing with restriction and relabelling. The resulting net calculus, called SCONE$^+$, is exploited as an implementation language for full CCS. The truly concurrent semantics induced by the implementation mapping is proved to coincide with the so-called "permutation-of-transitions" sematics.

1. Introduction

Among the various approaches to the semantics of concurrency, we distinguish two: the so-called "interleaving" approach and the "truly concurrent" one. The main merit of the former is its well-established theory. A concurrent system is described by a

☆ This paper reports in a coherent way some preliminary results which appeared as [25, 26], also reported as Chs. 7–9 of Gorrieri's [23] Ph.D. Thesis.

The first author has been partially supported by Exprit BRA no. 9102 COORDINATION, while the second author by Esprit BRA no. 3011 CEDISYS. Both authors have been also partially supported by Progetto Finalizzato, Obiettivo Lambrusco.

* Corresponding author. Email: gorrieri@cs.unibo.it.

SSDI 0304-3975(94)00073-R

term of an algebra, which gives rise to a labelled transition system. The states are themselves terms and the transitions are defined by means of a deductive system in structural inductive form, as proposed by Plotkin [39] with his *structural operational semantics* (SOS for short). Equivalences among states/terms are defined according to a suitable notion of observation and the useful result is that observational congruences often have nice axiomatizations. Unfortunately, there is a serious drawback: this approach relies on the well-known idea of describing system behaviour as sequences of transitions, a too simplistic view in many practical cases when information about distribution in space, about causal dependency or about fairness must be provided. On the other hand, in the "truly concurrent" approach, which started from the pioneering work of Petri [38], this kind of information can be easily given, but net theory has not yet reached a completely satisfactory theoretical treatment if compared with the firm results coming from the interleaving side. Rephrasing and extending the ideas developed for the interleaving approach to the "truly concurrent" case can be considered the main goal of a branch of concurrency research. The present paper aims at giving a contribution in this direction.

Petri nets, at least in their usual formulation, are not very suited for modular description of concurrent systems, because no general theory of composition and decomposition is available as there is no (finite) syntax generating them. Therefore, we shall restrict our attention to a particular class of nets possessing such a syntax, which thus forms naturally a language for nets. We are interested not only in defining a language whose formulae specify distributed systems, but also in describing the nets representing their operational behaviour as a *calculus*. Hence, extending Plotkin's paradigm to distributed systems, formulae of the language would denote markings of the net, while net transitions would be defined by means of a syntax-driven deductive system.

The problem is now to find a sufficiently general framework where concurrent calculi and net (or distributed) calculi can be described uniformly; this would permit an easy comparison between the two approaches and, in particular, the possibility of implementing concurrent calculi in net calculi in a (hopefully) direct way. Recently, several attempts have been made towards a unifying approach to concurrency. In particular, the work developed in Pisa is mainly concerned with the definition of a uniform algebraic framework in which specifications based both on transition systems and on Petri nets fit rather naturally [10]. This investigation started with [32], where the basic model of place/transition Petri nets has received a simple algebraic description by showing that a P/T net can be *statically* described as a directed graph equipped with a commutative monoidal operation $\oplus$ on nodes (union of markings), and *dynamically* as a graph with also two operations on transitions (parallel $\otimes$ and sequential; composition), together with suitable axioms for identifying those computations which are observationally identical.

By observing that transition system are nothing but directed graphs with labelled transitions, we discover that the notion of graph is a possible unifying mathematical tool for investigating the relationship between the two approaches. Moreover, SOS

specifications [39] can be described in algebraic form: the transition system is a two-sorted algebra with states and transitions as sorts [35, 18]. The states are the language terms, and the (proofs of the) transitions, being defined through deductive systems, can be represented as terms of an algebra having the axioms as generators and the inference rules as operations. Thus, the other common link between the two approaches is the algebraic structure for nodes and transitions. Indeed, SOS specifications and Petri nets are both specializations of the graph concept obtained by adding (different) algebraic structures on nodes and transitions: thus, graphs defined as two-sorted algebras represent the uniform framework we were looking for.

A calculus for nets can be introduced by defining an algebra for the nodes of the graph in such a way that it can be seen as a free algebra of markings, generated by the places. Therefore, the algebra must possess, among others, also the commutative monoidal operator $\oplus$ of union of markings.

As a first case study, we present a simple calculus of nets (SCONE). The operators generating places are prefixing and nondeterministic choice. The operators generating net transitions are prefixing, internal choice, and synchronization. The axioms in the deductive system of the calculus (prefix and internal choice) are the generators of the algebra and the inference rule (synchronization) is its sole operation, building a new transition from a pair of given transitions.

The semantics of SCONE is the semantics of a P/T net. Among the various semantic notions, we mention nonsequential processes (unfoldings of the net from an initial marking) [21] and commutative processes [3]. In [16] these notions are given an intuitive algebraic axiomatization on the algebra of net computations [32], where actually a slight refinement of classical nonsequential processes, called *concatenable processes*, is considered. To our aims, we choose concatenable processes because they faithfully represent causal dependencies and are equipped with an operation of sequential composition.

RCCS – the subset of CCS [34] without restriction and relabelling – can be implemented in SCONE, by means of a suitable mapping from its transition system (in algebraic form) to the Petri net of SCONE. This can be seen as a *denotational semantics* for RCCS with SCONE as semantic domain. RCCS agents are mapped to SCONE markings by considering prefixed $(\mu.E)$ and nondeterministically composed $(E+E')$ agents as places and by interpreting parallel composition as the multiset union of places. The implementation of RCCS transitions is less immediate and has been influenced by the categorical formulation of Petri nets in [32], which provides a flexible tool for relating system descriptions at different levels of abstraction by means of *implementation* morphisms in the category of net computations: a net transition can be mapped to an entire net computation. As a matter of fact, the combinators of SCONE are sometimes more elementary than those of RCCS, so that a RCCS transition may be mapped to a SCONE computation. Thus, the semantic morphism maps transitions to net computations by mapping basic operators of (the algebra of) RCCS to derived operators of (the algebra of) SCONE. A striking example

is concerned with the implementation of the external nondeterminism of RCCS in terms of the internal nondeterminism of SCONE. A natural consequence of our algebraic approach is that the implementation mapping does not affect the granularity of the execution: since a RCCS transition is executed atomically, the execution of the SCONE computation implementing it must be atomic as well. In other words, the distributed implementation of a RCCS agent is not a SCONE net, rather an implementation morphism on a SCONE net.

The relevance of the result is that this mapping can be seen as an instance of a general algebraic methodology for implementing parallel languages (also in interleaving form) in others (even distributed). Indeed, the second example illustrates this methodology in the more complex case of full CCS.

As already observed, e.g., [41], the transition system representation of an agent is usually larger than its net representation. As a matter of fact, not all the RCCS agents have a finite transition system representation. We prove that for any marking v, the SCONE subnet reachable from v is always *finite* (but not safe). Hence, by means of the implementation mapping, we show that any RCCS agent is always implemented on a finite net.

The second case study is concerned with a proper distributed treatment of full CCS. To this aim, we define a new calculus for nets, SCONE$^+$, extending SCONE with restriction and relabelling. It is exploited as the target machine through which we give a distributed implementation of CCS.

In order to deal with restriction correctly, parallel composition is modelled through a syntactic construction which leads, as side effect, to a 1-safe P/T net representation of the reachable subnet implementing a CCS agent. The price to pay is that such subnets may be infinite.

It is interesting to study not only the target of the implementation mapping, but also the effect on the source. We prove that the semantics of CCS agents as factorization of their computations w.r.t. the implementation mapping is the "permutation-of-transitions" semantics proposed by [5, 18]. Our semantics is consistent and complete with respect to theirs, in this case. On the contrary, in the case of RCCS, the "quotient" semantics is complete only under some mild assumptions.

Since Boudol and Castellani have proved in [6] that the "permutation-of-transitions" semantics is equivalent to a variant of Winskel's [42] event structure semantics and to a variant of Degano et al.'s [12] distributed choice net semantics for CCS, indirectly we get a nice correspondence result among all these different truly concurrent semantics for CCS.

The paper is organized as follows. Sections 2–4 introduce background material. It comprises of an introduction to the basic definitions of category theory used throughout the paper (see [29] for more details), an account of the algebraic formulation of P/T Petri nets and a brief exposition of CCS in its algebraic formulation. The proposed simple calculus of nets is introduced in Section 5. In Section 6 we describe the implementation mapping from RCCS to SCONE and then, in Section 7, we prove that the induced semantics is consistent with respect to Milner's and also faithfully

represents causality. SCONE$^+$ is introduced (Section 8)[1] together with the implementation mapping (Section 9). Hence, the comparison with Boudol and Castellani semantics is reported in Section 10. Section 11 is concerned with the relationships of our investigations with some related works. In particular, we discuss the issue of atomicity in giving semantics to CCS [24, 13], other proposals for giving finite net representation to RCCS agents [19, 22], and some recent results on the connections between Petri nets and the chemical abstract machine (CHAM) [2].

2. Graphs, categories and monoidal structures

A (directed) *graph* G is a tuple $N=(V,T,\partial_0,\partial_1)$, where V is the class of *states* (or nodes), T of *transitions* (or arcs), and $\partial_0,\partial_1: T\to V$ are two functions, called *source* and *target*, respectively. We denote an arc t, such that $\partial_0(t)=u$ and $\partial_1(t)=v$, by the shorthand $t: u\to v$. A *graph morphism* from G to G' is a pair of functions $\langle f,g\rangle, f: T\to T'$ and $g: V\to V'$, which preserve the source and the target functions: $g(\partial_0(t))=\partial'_0(f(t))$ and $g(\partial_1(t))=\partial'_1(f(t))$ for all transitions t.

A *category* $C=(V,T,\partial_0,\partial_1,\mathrm{id},;)$ is a graph $(V,T,\partial_0,\partial_1)$, where the states in V and the transitions in T are usually called *objects* and *arrows* (or *morphisms*), respectively, with in addition:

- an operation $\mathrm{id}: V\to T$ called *identity*, such that $\partial_0(\mathrm{id}(v))=v=\partial_1(\mathrm{id}(v))$,
- a *partial* operation; $T\times T\to T$ called *composition*, assigning to each pair of arrows t and t', such that[2] $\partial_0(t')=\partial_1(t)$, an arrow $t;t'$ such that $\partial_0(t;t')=\partial_0(t)$, $\partial_1(t;t')=\partial_1(t')$
- and the operations satisfy the two axioms (which hold if both members are defined)

$$t;(t';t'')=(t;t');t'' \qquad \mathrm{id}(u);t=t=t;\mathrm{id}(v)$$

Moreover, for any pair (u,v), the class $C[u,v]=\{t \mid t\in T,\ \partial_0(t)=u,\ \partial_1(t)=v\}$ is actually a set. Let C,D be two categories. A (covariant) *functor* $F: C\to D$ is a pair of functions $F_V: V_C\to V_D$ and $F_T: T_C\to T_D$, such that for each $t,t'\in T_C$

- $\partial_{iD}(F_T(t))=F_V(\partial_{iC}(t))$ $(i=0,1)$
- $F_T(\mathrm{id}(v))=\mathrm{id}(F_V(v))$
- $F_T(t;t')=F_T(t);F_T(t')$.

Let $F,G: C\to D$ be two functors. A *natural transformation* $\tau: F\to G$ is a function from V_C to T_D assigning to an object u in C an arrow τ_u in D such that

- $\partial_0(\tau_u)=F_V(u)$ and $\partial_1(\tau_u)=G_V(u)$
- $\forall t$ in T_C with $\partial_0(t)=u$ and $\partial_1(t)=v$ $\quad \tau_u;G_T(t)=F_T(t);\tau_v$

[1] The reader interested only in this case study can skip most of Section 3, focusing mainly on Definition 3.14.

[2] Note that composition ; is defined following the interpretation of sequential composition of transitions (diagrammatic order), which is obvious in computer science, but is in contrast with the tradition in mathematics.

A transformation τ such that each component τ_u is invertible in D is called a natural *isomorphism.* A *strict monoidal* category $(C, \otimes, e)$ consists of

- a category C;
- a (left and right) identity object e (i.e. $e \otimes t = t = t \otimes e$);
- a bifunctor $\otimes : C \times C \to C$, thus satisfying the functoriality axioms[3]

$$(t_1 \otimes t_2);(t_1' \otimes t_2') = (t_1 ; t_1') \otimes (t_2 ; t_2') \qquad \mathrm{id}(u) \otimes \mathrm{id}(v) = \mathrm{id}(u \otimes v)$$

and, additionally, $\otimes$ is associative, with e as neutral element on objects and $\mathrm{id}(e)$ on arrows.

A *symmetric* strict monoidal category $(C, \otimes, e, \gamma)$ is a strict monoidal category enriched with a natural isomorphism $\gamma_{u,v} : u \otimes v \to v \otimes u$ such that the following two equations hold:

$$\gamma_{u,v} ; \gamma_{v,u} = \mathrm{id}(u \otimes v) \qquad (\gamma_{u,v} \otimes \mathrm{id}(w)) ; (\mathrm{id}(v) \otimes \gamma_{u,w}) = \gamma_{u, v \otimes w}$$

Let us consider two arrows $t : u \to v$, $t' : u' \to v'$. An interesting consequence of stating that γ is a natural isomorphism is

$$\gamma_{u,u'} ; (t' \otimes t) = (t \otimes t') ; \gamma_{v,v'}$$

meaning that the factors can be exchanged in any monoidal composition of arrows, provided that suitable exchanges are sequentially composed before and after.

A *strictly symmetric* strict monoidal category $(C \otimes, e)$ is a symmetric strict monoidal category where $\gamma_{u,v}$ is the identity $\mathrm{id}(u \otimes v)$; hence, $\otimes$ is commutative, too.

3. An algebraic view to Petri nets

We assume the reader is familiar with the basic concepts of net theory (see, e.g. [40]). By Petri nets we mean (capacity free) *place/transition* nets, where the flow relation is *finite* (the pre- and post-sets of transitions are always finite multisets) and every transition has a nonempty pre-set. This section is devoted to recalling the categorical approach to Petri nets as graphs with a monoidal structure proposed in [32, 16]. The basic idea is to consider a transition t as an arrow from its pre-set to its post-set, i.e. $t : {}^\bullet t \to t^\bullet$, and the set of the multisets of places as the free commutative monoid over the set of places. In other words, a Petri net is a graph whose set of nodes is a free commutative monoid.

Definition 3.1 (*Finite multiset, union and empty multiset*). Given a set S, a *finite multiset* over S is a function $M : S \to \mathbb{N}$ such that the set $\{s \in S \mid M(s) \neq 0\}$ is finite. The *multiplicity* of an element s in M is given by $M(s)$. Given two multisets M and M'

[3] The first equality holds whenever both members are defined; indeed, due to the fact that ; is a partial operation, it may happen that the left member is defined, but not the right one. For the categorical associativity axiom, instead, if one of the two members is defined, then also the other is so.

over a set S, their *union*, denoted by $M \oplus M'$, is the multiset given by $M \oplus M'(s) = M(s) + M'(s)$. The empty multiset, denoted by 0, is defined by $0(s) = 0$.

Since natural number addition is associative and commutative, then also multiset union is so. Furthermore, since 0 is the natural element of addition, 0 is the neutral element of multiset union.

Property 3.2 (*Multisets as free commutative monoids*). Given a set S, let $S^{\oplus}$ denote the set of (finite) multisets over S. With the union operator $\oplus$ and the element 0, $S^{\oplus}$ is a free commutative monoid over S.

Definition 3.3 (*Petri nets as graphs with a monoidal structure*). A *place/transition petri net* (net, in short) is a graph $N = (S^{\oplus}, T, \partial_0, \partial_1)$, where $S^{\oplus}$ is the free commutative monoid of nodes over a set of *places* S. The elements of $S^{\oplus}$, called also the *markings* of the net N, are represented as formal sums $n_1 a_1 \oplus \cdots \oplus n_k a_k$ ($a_i \in S$, n_i is a natural number) with the order of the summands being immaterial, where addition is defined by $(\bigoplus_i n_i a_i) \oplus (\bigoplus_i m_i a_i) = (\bigoplus_i (n_i + m_i) a_i)$ and 0 is its neutral element.

A *Petri net morphism* h from $N = (S^{\oplus}, T, \partial_0, \partial_1)$ to $N' = (S'^{\oplus}, T', \partial'_0, \partial'_1)$ is a graph morphism (i.e. a pair of functions $\langle f, g \rangle$, $f: T \to T'$ and $g: S^{\oplus} \to S'^{\oplus}$, preserving source and target) where g is a monoid morphism (i.e. leaving 0 fixed and respecting $\oplus$). With this definition of morphism, nets form a category, called *Petri*, which is equipped with products and coproducts [29].

We define an algebra of (finite) *computations* by considering, as generators, the transitions in T and also a set of transitions, called *symmetries*, defined below. Moreover, the operations of the algebra are the associative sequentialization (partial) operation ; and a monoidal operation $\otimes$ on transitions, which is interpreted as parallel composition. By imposing suitable axioms on the operations of sequential and parallel composition (yielding a monoidal category) we can define equivalence classes of net computations. In [16] it is shown that some of the semantic notions on Petri nets can be naturally characterized in this axiomatic approach. Here, we will present the category every morphism of which turns out to be an equivalence class of computations all evaluating to the same *concatenable* process [16] (a slight variation of the Goltz/Reisig *nonsequential* process [21]).

Definition 3.4 (*Symmetries*). Let us consider a finite set I labelled by a function $l: I \to S$ which associates to every element x a label/place $l(x)$. When defined up to isomorphisms (i.e. up to label-preserving bijections), set I corresponds to an element $u = n_1 a_1 \oplus \cdots \oplus n_k a_k$ in $S^{\oplus}$, where $n_i = |\{x \mid l(x) = a_i\}|$, $i = 1, \ldots, k$.

A *symmetry* p of the labelled set I is a bijective endofunction $p: I \to I$ which is label-preserving, i.e. such that $l(x) = l(p(x))$. We can associate it to u and write $p: u \to u$. It is clear that, by choosing a linear order for each of the sets $\{x \mid l(x) = a_i\}$, $i = 1, \ldots, k$, p can be expressed as a vector of permutations. Given $u = n_1 a_1 \oplus \cdots \oplus n_k a_k$

in $S^{\oplus}$, a symmetry $p: u \to u$ is a vector of permutations $\langle \sigma_{a_1}, \ldots, \sigma_{a_k} \rangle$ with $\sigma_{a_i} \in \Pi(n_i)$, i.e. σ_{a_i} is a permutation of n_i elements ($|\sigma_{a_i}| = n_i$).

We define three operations on symmetries: the sequential composition $p; p'$ of two symmetries, the parallel composition $p \otimes p'$ and the interchange of two objects u and v, which gives rise to the symmetry $\gamma(u, v)$. Let $p: u \to u$ and $p': u \to u$; then

$$p; p': u \to u = \langle \sigma_{a_1}; \sigma'_{a_1}, \ldots, \sigma_{a_k}; \sigma'_{a_k} \rangle \quad \text{where } \sigma; \sigma'(x) = \sigma'(\sigma(x)).$$

Let $p: u \to u$ and $p': v \to v$; then

$$p \otimes p': u \oplus v \to u \oplus v = \langle \sigma_{a_1} \otimes \sigma'_{a_1}, \ldots, \sigma_{a_k} \otimes \sigma'_{a_k} \rangle$$

where

$$\sigma \otimes \sigma'(x) = \begin{cases} \sigma(x) & \text{if } 0 < x \leqslant |\sigma|, \\ \sigma'(x - |\sigma|) + |\sigma| & \text{otherwise.} \end{cases}$$

The interchange symmetry $\gamma(u, v): u \oplus v \to v \oplus u$, associated to permutation $\{1 \to 2, 2 \to 1\}$ and to $u = n_1 a_1 \oplus \cdots \oplus n_k a_k$, and $v = m_1 a_1 \oplus \cdots \oplus m_k a_k$, is the vector of permutations $\langle \sigma_{a_1}, \ldots, \sigma_{a_k} \rangle$ defined by

$$\sigma_{a_j}(x) = \begin{cases} m_j + x & \text{if } x \leqslant n_j, \\ x - n_j & \text{if } x > n_j. \end{cases}$$

Example 3.5. A suggestive graphical representation of a symmetry p on $3a \oplus 2b$ where $\sigma_a = \{1 \to 2, 2 \to 3, 3 \to 1\}$ and $\sigma_b = \{1 \to 2, 2 \to 1\}$ is depicted in the first operand of Fig. 1(a). The intuition behind the three operations defined above can be easily grasped from Fig. 1. To obtain the sequential comosition $p; p'$ of two symmetries on u, we have simply to follow the threads of the permutations. Supposing $p: u \to u$ and $q: v \to v$, the *parallel* composition $p \otimes q: u \oplus v \to u \oplus v$ of the two symmetries is obtained by putting side by side the permutations regarding the same place. Exchanging the summands in a node gives rise to an *interchange* symmetry; if $u = 2a \oplus b$ and $v = 3a \oplus 2b$, then the interchange symmetry is formally denoted by $\gamma(u, v) = \langle \sigma_a, \sigma_b \rangle$, where $\sigma_a = \{1 \to 4, 2 \to 5, 3 \to 1, 4 \to 2, 5 \to 3\}$ and $\sigma_b = \{1 \to 3, 2 \to 1, 3 \to 2\}$.

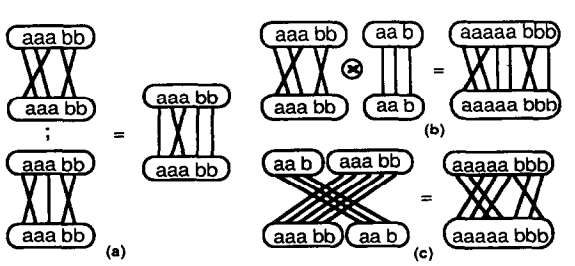

Fig. 1. Three operations on symmetries.

Both $\otimes$ and ; are associative but not commutative. If ι_{a_i} denotes the identity permutation, then for each u in $S^{\oplus}$ the symmetry $\langle \iota_{a_1}, \ldots, \iota_{a_k} \rangle : u \to u$ is the identity transition $\mathrm{id}(u)$. Furthermore, $\otimes$ and ; satisfy the functoriality equations:

$$(p \otimes p'); (q \otimes q') = (p; q) \otimes (p'; q') \qquad \mathrm{id}(u) \otimes \mathrm{id}(v) = \mathrm{id}(u \oplus v)$$

Finally, the interchange $\gamma(u, v)$ is a natural transformation

$$\gamma(u, v); (p \otimes q) = (q \otimes p); \gamma(u, v)$$

satisfying also the axioms of symmetric strict monoidal categories

$$\gamma(u, v); \gamma(v, u) = \mathrm{id}(u \oplus v) \qquad (\gamma(u, v) \oplus \mathrm{id}(w)); (\mathrm{id}(v) \otimes \gamma(u, w)) = \gamma(u, v \oplus w).$$

Given a set S, let Sym_S be the graph whose nodes are the elements of the commutative monoid $S^{\oplus}$ and whose transitions are symmetries with the above-defined operations of $\otimes$ and ;. Then, Sym_S is a category, because identities do exist and ; is the arrow composition; Sym_S is strict monoidal, because the pair $\langle \oplus, \otimes \rangle$ is a bifunctor ($\oplus$ on nodes and $\otimes$ on transitions) which is associative and has the neutral element; Sym_S is symmetric since the *interchange symmetry* associated to the permutation $\gamma = \{1 \to 2, 2 \to 1\}$, $\gamma(u, v) : u \otimes v \to v \otimes u$, is the required natural isomorphism.

Proposition 3.6. *Sym_S is a symmetric strict monoidal category, strictly symmetric on objects.*

Definition 3.7 (*From a net to the category of its processes*). Given a net $N = (S^{\oplus}, T, \partial_0, \partial_1)$, the category $\mathscr{P}[N]$ of its processes is defined as follows. The objects of $\mathscr{P}[N]$ are the nodes of N, i.e. $S^{\oplus}$. $\mathscr{P}[N]$ includes Sym_S as a subcategory, and has as additional arrows only those defined by the following rules of inference:

$$\frac{t : u \to v \text{ in } N}{t : u \to v \text{ in } \mathscr{P}[N]}$$

$$\frac{\alpha : u \to v \quad \alpha' : u' \to v' \text{ in } \mathscr{P}[N]}{\alpha \otimes \alpha' : u \oplus u' \to v \oplus v' \text{ in } \mathscr{P}[N]} \qquad \frac{\alpha : u \to v \quad \beta : v \to w \text{ in } \mathscr{P}[N]}{\alpha; \beta : u \to w \text{ in } \mathscr{P}[N]}$$

and axioms expressing the fact that the arrows form a monoid:

$$(\alpha \otimes \beta) \otimes \delta = \alpha \otimes (\beta \otimes \delta) \qquad \alpha \otimes \mathrm{id}(0) = \alpha$$

the fact that $\mathscr{P}[N]$ is a category:

$$\alpha; \mathrm{id}(\partial_1(\alpha)) = \mathrm{id}(\partial_0(\alpha)); \alpha = \alpha \qquad (\alpha; \beta); \delta = \alpha; (\beta; \delta)$$

and the fact that $\langle \oplus, \otimes \rangle$ is a bifunctor:

$$(\alpha \otimes \alpha'); (\beta \otimes \beta') = (\alpha; \beta) \otimes (\alpha'; \beta') \qquad \mathrm{id}(u) \otimes \mathrm{id}(v) = \mathrm{id}(u \oplus v).$$

Also, there is an axiom stating that generators (i.e. the transitions of N) are symmetrical:

$t;p=t$ where $t:u\to v$ in N and $p:v\to v$ in Sym_S,

$p;t=t$ where $t:u\to v$ in N and $p:u\to u$ in Sym_S.

Finally, the fact that the *interchange symmetry* $\gamma(u_1,u_2)$, associated to the permutation $\gamma=\{1\to 2, 2\to 1\}$, defines the natural isomorphism, where $\alpha_i:u_i\to v_i$ $(i=1,2)$:

$$\gamma(u_1,u_2);(\alpha_2\otimes\alpha_1)=(\alpha_1\otimes\alpha_2);\gamma(v_1,v_2).$$

Theorem 3.8 (Degano et al. [16]). *Given a net $N=(S^{\oplus},T,\partial_0,\partial_1)$, the category $\mathscr{P}[N]$ of its process is the quotient of the symmetric, strict monoidal category freely generated by the net N (where the monoidal operation is denoted by $\otimes$ on arrows and by $\oplus$ on objects, the operator of arrow composition is ;, and $\gamma(u,v)$ is the natural isomorphism of commutativity), determined by the axioms (Ψ) below, where $t:u\to v$ belongs to N and p is a computation involving natural isomorphisms only:*

Objects:	$u\oplus v=v\oplus u$	*as a whole* (Ψ).
Isomorphisms:	$\gamma(a,b)=\mathrm{id}(a\oplus b)$ $\forall a,b\in S,\ a\neq b$	
Transitions:	$t:p=t$ *where* $p:v\to v$ *is a symmetry*	
	$p;t=t$ *where* $p:u\to u$ *is a symmetry*	

This theorem states that $\mathscr{P}[N]$ can be obtained by freely generating the symmetric, strict monoidal category from the net $N=(S^{\oplus},T,\partial_0,\partial_1)$, considering a bifunctor $(\oplus,\otimes)$ which is commutative up to isomorphism. The latter is specified by the natural isomorphism γ. Then, by identifying the nodes $u\oplus v$ and $v\oplus u$ which differ only by such an isomorphism. As a consequence, the "vector" $u\oplus v$ becomes now a multiset. Analogously, the arrow $\gamma(u,v):u\oplus v\to v\oplus u$, which is a (global) permutation, must be transformed into a symmetry (a vector of local permutations). This is obtained by adding the equation $\gamma(a,b)=\mathrm{id}(a\oplus b)$, which states that the permutation $\gamma(a,b)$ is in fact the identity of $a\oplus b$ if a and b are tokens on different places. The last equations state that token exchanges do not affect net transitions.

The arrows of $\mathscr{P}[N]$ are equivalence classes of net computations: an arrow α represents the *observation* out of any computation in the equivalence class of α. Such an observation is a *concatenable process* [16].

Definition 3.9 (*Label-indexing ordering*). Given a set S with a labelling function $l:S\to S'$, a *label-indexed ordering function* is a family $\beta=\{\beta_a\}$, $a\in S'$, of bijections, where $\beta_a:[a]\to\{1,\ldots,|[a]|\}$, with $[a]=\{b\in S\mid l(b)=a\}$.

Definition 3.10 (*Plain processes*). A *plain process* for a net N is a morphism $\varphi=\langle f,g\rangle$ in *Petri* from a finite occurrence net P to N. The functions f and g map transitions and places of P to transitions and places of N, respectively. The places of P which are

minimal in the partial ordering associated to it are called *origins*, the maximal places are called *destinations*. We give plain processes a categorical structure, by taking as morphisms

$$\delta:(\varphi: P\to N)\to(\varphi': P'\to N)$$

between processes φ and φ' those *Petri* morphisms

$$\rho: P\to P'$$

between the supporting occurrence nets such that $\rho;\varphi'=\varphi$.

Definition 3.11 (*Concatenable processes*). A *concatenable process* for a net N is a triple $C=\langle\varphi,\theta,\kappa\rangle$, where:

- $\varphi=\langle f,g\rangle$ is a plain process for N;
- θ and κ are label-indexed ordering functions on the origins and destinations, respectively, where the labelling function is g restricted to the respective domains.

Isomorphic[4] concatenable processes are identified. Furthermore, we can associate to every concatenable process C of N two multisets of places as follows. The multisets $O(C)$ and $D(C)$ are defined as

$$O(C)=\sum_{i=1}^{k} n_i a_i \quad \text{and} \quad D(C)=\sum_{i=1}^{k} m_i a_i,$$

where a_i are places of N, and n_i and m_i are the numbers of origins b and destinations c of P, respectively, such that $g(b)=a_i=g(c)$.

A concatenable process is essentially a finite nonsequential process [21] with, additionally, *ordered* labels on both origins and destinations. This means that the origins (destinations) of a concatenable process mapped on the same place are distinguished by imposing an ordering on them.[5]

We can picture a concatenable process C of N as an arrow $C: O(C)\to D(C)$. Also concatenable processes may easily be turned into a monoidal category. Indeed, we show how concatenable processes can be associated to net transitions and symmetries, and also how parallel composition and sequential compositions can be defined. Thus we have an *algebra* of concatenable processes.

Given a transition $t: n_1 a_1 \oplus \cdots \oplus n_k a_k \to n'_1 a_1 \oplus \cdots \oplus n'_k a_k$ in N, let P be the occurrence net with $I\cup I'$ as set of places, where $I=\{\langle i,j,0\rangle \mid i=1,\ldots,k,\ j=1,\ldots,n_i\}$, $I'=\{\langle i,h,1\rangle \mid i=1,\ldots,k,\ h=1,\ldots,n'_i\}$, and t' as unique transition with $\partial_0(t')=I$ and $\partial_1(t')=I'$. Then a concatenable process for t is $C=\langle\varphi,\theta,\kappa\rangle$, where $\varphi=\langle f,g\rangle$ is such

[4] Two concatenable processes C and C' are isomorphic if there is a plain process isomorphism $\langle f'', g''\rangle$ from P to P' preserving the label-indexed functions, namely with $\theta(b)=\theta'(g''(b))$ and $\kappa(b)=\kappa'(g''(b))$.

[5] Usually, in depicting a process, the ordering on process places mapped to the same net place is implicitly given by the "space" (left-to-right) ordering. If a token is both origin and destination (i.e. isolated), then it should be explicitly equipped with the ordering annotation (see the representation of a symmetry in Fig. 3).

that $f(t')=t$ and $g(\langle i,j,0\rangle)=g(\langle i,h,1\rangle)=a_i$, $i=1,\ldots,k$, $j=1,\ldots,n_i$, $h=1,\ldots,n'_i$, and the label-indexed ordering functions on the origins and destinations of P are given by $\theta_{a_i}(\langle i,j,0\rangle)=j$ and $\kappa_{a_i}(\langle i,h,1\rangle)=h$.

Given a symmetry $p=\langle\sigma_{a_1},\ldots,\sigma_{a_k}\rangle: n_1a_1\oplus\cdots\oplus n_ka_k\to n_1a_1\oplus\cdots\oplus n_ka_k$, let P be the occurrence net having $I=\{\langle i,j\rangle \mid i=1,\ldots,k,\ j=1,\ldots,n_i\}$ as set of places, and no transitions. A concatenable process for p is $C=\langle\varphi,\theta,\kappa\rangle$ where $\varphi=\langle\emptyset,g\rangle$ is such that $g(\langle i,j\rangle)=a_i$, and the ordering functions on the origins and destinations of P are such that $\theta_{a_i}(\langle i,j\rangle)=j$ and $\kappa_{a_i}(\langle i,j\rangle)=\sigma_{a_i}(j)$, respectively.

In the following definitions of parallel and sequential composition of processes, let $C=\langle\varphi,\theta,\kappa\rangle: n_1a_1\oplus\cdots\oplus n_ka_k\to m_1a_1\oplus\cdots\oplus m_ka_k$ and $C'=\langle\varphi',\theta',\kappa'\rangle: n'_1a_1\oplus\cdots\oplus n'_ka_k\to m'_1a_1\oplus\cdots\oplus m'_ka_k$ be two concatenable processes for N, with P and P' as occurrence nets of φ and φ', respectively. We define $C\otimes C'=\langle\varphi'',\beta'',\gamma''\rangle:(n_1+n'_1)a_1\oplus\cdots\oplus(n_k+n'_k)a_k\to(m_1+m'_1)a_1\oplus\cdots\oplus(m_k+m'_k)a_k$, where φ'' is the coproduct of φ and φ' in the category of plain processes, $\theta''=\langle\theta_{a_1}\cup(n_1+\theta'_{a_1}),\ldots,\theta_{a_k}\cup(n_k+\theta'_{a_k})\rangle$ and $\kappa''=\langle\kappa_{a_1}\cup(n_1+\kappa'_{a_1}),\ldots,\kappa_{a_k}\cup(n_k+\kappa'_{a_k})\rangle$.

Finally, $C;C'$ is defined only if $D(C)=O(C')=m_1a_1\oplus\cdots\oplus m_ka_k$. Let us define a plain process $\varphi^-=\langle\emptyset,g^-\rangle$ from the occurrence net $P^-=(S^{-\oplus},\emptyset,\emptyset,\emptyset)$ with $S^-=\{\langle i,j\rangle \mid i=1,\ldots,k,j=1,\ldots,m_i\}$ and $g^-(\langle i,j\rangle)=a_i$. Two plain process morphisms ρ,ρ' from φ^- to φ and φ', respectively, are induced by the two functions s and s' from S^- to the destinations of P and to the origins of P', respectively, that satisfy the following equations:

$$g(s(\langle i,j\rangle))=a_i=g'(s'(\langle i,j\rangle)) \quad\text{and}\quad \kappa(s(\langle i,j\rangle))=j=\theta'(s'(\langle i,j\rangle)).$$

It is not difficult to see that s and s', and therefore ρ and ρ', are defined uniquely. Also, it is easy to verify that in the category of processes the pushout construction involving ρ and ρ' exists, yielding a new process φ'' which is obtained, roughly speaking, as the disjoint union of φ and φ', where for all x, the pairs $s(x)$ and $s'(x)$ have been identified. Then,

$$C;C'=\langle\varphi'',\beta,\gamma'\rangle: n_1a_1\oplus\cdots\oplus n_ka_k\to m'_1a_1\oplus\cdots\oplus m'_ka_k.$$

Theorem 3.12 ($\mathscr{P}[N]\cong\mathscr{CP}[N]$ (Degano et al. [16]). (i) *Given a net N, concatenable processes on it form a symmetric strict monoidal category $\mathscr{CP}[N]$ satisfying the equation (Ψ).*

(ii) *There is a unique homomorphism H from $\mathscr{P}[N]$ to $\mathscr{CP}[N]$ preserving all the operations and leaving N fixed when viewed as a subnet of $\mathscr{P}[N]$ and of $\mathscr{CP}[N]$ via the obvious inclusions. Furthermore, homomorphism H is actually an isomorphism.*

This theorem proves that any arrow of $\mathscr{P}[N]$ can be represented as a concatenable process and any term of the algebra can be *evaluated* to the concatenable process representing its equivalence class. Vice versa, any concatenable process may be denoted by a term of the algebra $\mathscr{P}[N]$.

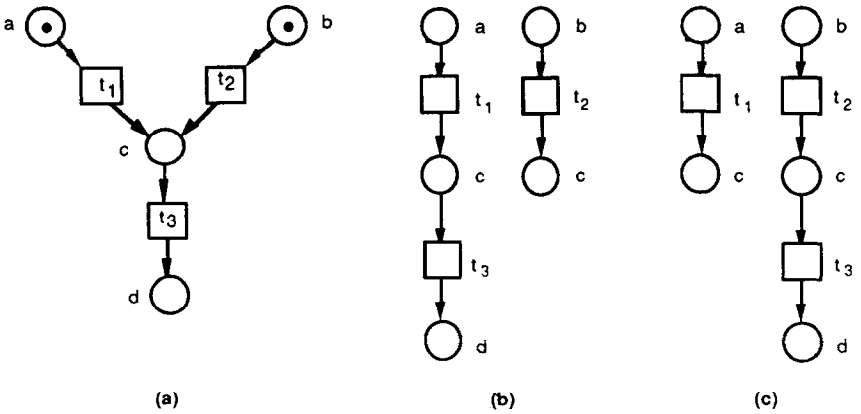

Fig. 2. A net in (a) and two of its processes in (b) and (c).

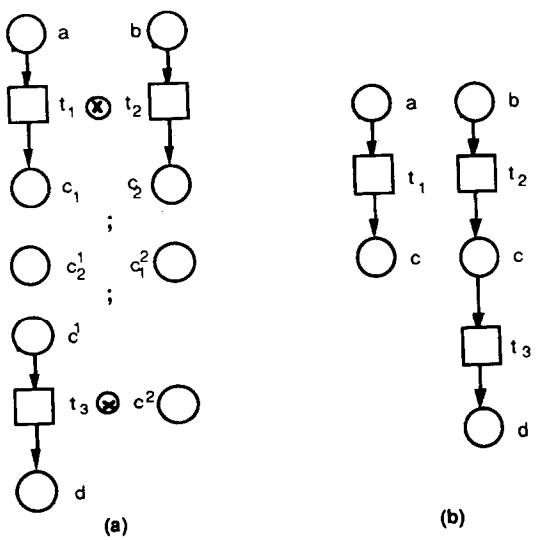

Fig. 3. (a) The evaluation in $\mathscr{CP}[N]$ of a term of $\mathscr{P}[N]$; the corresponding result is in (b).

Example 3.13. Let us consider the net in Fig. 2(a). Fig. 2(b) and (c) show two of its processes. Term $(t_1 \otimes t_2);(t_3 \otimes \mathrm{id}(c))$ corresponds to starting with a token in place a and a token in place b and to executing transitions t_1 and t_2 simultaneously. One of the tokens is then left in c, while the other (the one produced by t_1) is used for executing transition t_3. This term of $\mathscr{P}[N]$ corresponds to the process in Fig. 2(b). Similarly, the term $(t_1 \otimes \mathrm{id}(b));(\mathrm{id}(c) \otimes t_2);(t_3 \otimes \mathrm{id}(c))$ is related to the same process.

Fig. 3 illustrates through an example that a formal *evaluation* of terms of concatenable processes can be naturally provided. In this figure, the considered term is $(t_1 \otimes t_2);p;(t_3 \otimes \mathrm{id}(c))$, with $p=\langle \sigma_a, \sigma_b, \sigma_c \rangle$ and $\sigma_a=\sigma_b=\emptyset$ and $\sigma_c=\{1\to 2, 2\to 1\}$.

The algebraic characterization of concatenable processes can be conveniently simplified in the case of 1-safe nets [3]. Because each place contains at most one token, any symmetry $\gamma_{u,v}$ collapses to the identity $\mathrm{id}(u \oplus v)$. Hence, $\otimes$ becomes commutative and category $\mathscr{P}[N]$ becomes *strictly* symmetric. As in the second case study the nets exploited as implementation for CCS agents are 1-safe, we provide also this convenient characterization, which coincides incidentally with the axiomatization of Best/Devillers processes [3], provided in [16] where they are called *commutative* process.

Definition 3.14 (*From a net to the category of its commutative processes*). Given a net $N=(S^{\oplus}, T, \partial_0, \partial_1)$, the category $\mathscr{T}[N]$ of its *commutative* processes is the strictly symmetric strict monoidal category freely generated by N. Explicitly, the category $\mathscr{T}[N]$ is defined by the following rules of inference:

$$\frac{t: u \to v \text{ in } N}{t: u \to v \text{ in } \mathscr{T}[N]} \qquad \frac{u \text{ in } S^{\oplus}}{\mathrm{id}(u): u \to u \text{ in } \mathscr{T}[N]}$$

$$\frac{\alpha: u \to v \;\; \alpha': u' \to v' \text{ in } \mathscr{T}[N]}{\alpha \otimes \alpha': u \oplus u' \to v \oplus v' \text{ in } \mathscr{T}[N]} \qquad \frac{\alpha: u \to v \;\; \beta: v \to w \text{ in } \mathscr{T}[N]}{\alpha; \beta: u \to w \text{ in } \mathscr{T}[N]}$$

and axioms expressing the fact that the arrows form a commutative monoid:

$$(\alpha \otimes \beta) \otimes \delta = \alpha \otimes (\beta \otimes \delta) \qquad \alpha \otimes \beta = \beta \otimes \alpha \qquad \alpha \otimes \mathrm{id}(0) = \alpha$$

the fact that $\mathscr{T}[N]$ is a category:

$$\alpha; \mathrm{id}(\partial_1(\alpha)) = \mathrm{id}(\partial_0(\alpha)); \alpha = \alpha \qquad (\alpha; \beta); \delta = \alpha; (\beta; \delta)$$

and the functoriality of $\otimes$:

$$(\alpha \otimes \alpha'); (\beta \otimes \beta') = (\alpha; \beta) \otimes (\alpha'; \beta') \qquad \mathrm{id}(u) \otimes \mathrm{id}(v) = \mathrm{id}(u \oplus v)$$

Theorem 3.15. *The quotient of category $\mathscr{P}[N]$ with respect to the axiom below is $\mathscr{T}[N]$:*

Isomorphism: $\gamma(a, a) = \mathrm{id}(a \oplus a) \quad \forall a \in S$

Proof. This axiom[6] is the complement of the axiom given in Theorem 3.8. All symmetries are collapsed to identities with the relevant side effect of making $\otimes$ commutative on arrows. □

[6] This axiom is an algebraic description of the swap construction in [3]. As $\otimes$ is associative and also commutative up to natural isomorphism, then the permutation to nonconsecutive a's can always be reduced to the permutation of consecutive a's, provided that suitable interchange symmetries are sequentially composed before and after.

Of course, in the case of 1-safe nets, the axiom above holds "vacuously"; hence, concatenable and commutative[7] processes coincide for 1-safe nets.

4. A calculus of communicating systems

4.1. The classic approach to CCS and RCCS

We begin by recalling briefly a few definitions about Milner's CCS (we assume the reader is familiar with [34]). Let $\Lambda=\{\alpha,\beta,\gamma\ldots\}$ be a set of *action names*, $\Lambda^-=\{\alpha^-,\beta^-,\gamma^-\ldots\}$ the set of action *conames* and τ a special *silent action*. We will call $\Lambda=\Lambda\cup\Lambda^-$ the set of *visible actions* ranged over by λ, while $\mathcal{M}=\Lambda\cup\{\tau\}$ the set of *actions* ranged over by μ. The set of recursive terms over the signature $\Sigma_{CCS}=\bigcup_{n\geq 0}\Sigma_n$ is defined by the following BNF-like notation:

$$E :: x \mid op(E_1,E_2,\ldots,E_k) \mid \mathrm{rec}\,x.E$$

where x is any element in a (possibly infinite) set of variables *Var*, rec x.– is the binding construct, $op\in\Sigma_k$ and the signature Σ_{CCS} consists of the following operators:

$$\Sigma_0=\{\mathrm{nil}\},$$

$$\Sigma_1=\{\mu. \mid \mu\in\mathcal{M}\}\cup\{\backslash\alpha \mid \alpha\in\Lambda\}\cup\{[\Phi] \mid \Phi \text{ is a permutation of } \mathcal{M} \text{ preserving } {}^- \text{ and } \tau\},$$

$$\Sigma_2=\{|,+\},$$

$$\Sigma_n=\emptyset,\quad \forall n>2,$$

with the agreement to write the set of unary operators $\{\mu. \mid \mu\in\mathcal{M}\}$ in prefix form, the other unary operators in postfix form and the binary operators in infix form. We assume the reader is familiar with the usual notions of *free* and *bound* variables and of *syntactic substitution*. A term rec $x.E$ is *locally guarded* if every occurrence of x in E is inside the scope of a μ-prefixing. A term E is *guarded* if every recursive subterm of E is locally guarded. We denote by $CCS_{\mathcal{M}}$ the set of *closed* (i.e. without free variables) guarded terms, also called *CCS agents*, which will be ranged over by the variable E, with abuse of notation. For the sake of brevity, the ending constant nil is often omitted, as in $\alpha|\beta$.

The operational semantics of CCS is defined in terms of *labelled transition systems* [27], LTS for short. An LTS is a triple $(S,\mathcal{M},\{\xrightarrow{\mu} \mid \mu\in\mathcal{M}\})$ where S is a set of *states*,

[7] In general, commutative processes seem to supply too abstract a description of computations of place/transition nets. Because of commutativity of $\otimes$, multiple tokens in a place s become definitely "indistinguishable", thus making it impossible to describe correctly the causal dependencies between the transitions producing tokens in s and the transitions consuming these tokens. For instance, the net in Fig. 2(a) after the execution of t_1 and t_2, we have two tokens in c which cannot be distinguished; hence, both concatenable processes in Fig. 2(b) and (c) are considered equivalent, i.e. correspond to the same commutative process (see [3, 16] for more details).

$\mathcal{M}$ is a set of *actions* and each $\xrightarrow{\mu}$ is a binary *transition relation* on S. We will write $s \xrightarrow{\mu} s'$ instead of $(s,s') \in \xrightarrow{\mu}$. Hence, an LTS is a graph with labelled transitions. Every transition $s \xrightarrow{\mu} s'$ specifies that the system in the state s can transit to the state s' by performing the action μ.

A relevant breakthrough in the definition of operational semantics for languages was due to Plotkin [39] with his structured operational semantics (SOS for short). According to this technique, the terms of the language constitute themselves the states, and the transitions are defined by means of a deductive system in structural inductive form, making the definition of an abstract machine rather an easy task. The labelled transition system $\mathcal{T}_{\mathrm{CCS}} = (\mathrm{CCS}_{\mathcal{M}}, \mathcal{M}, \{\xrightarrow{\mu} \mid \mu \in \mathcal{M}\})$ defining the CCS operational semantics has agents as states and the transition relations are defined as follows.

Definition 4.1. The transition relation $\{\xrightarrow{\mu} \mid \mu \in \mathcal{M}\}$ is defined as the least relation satisfying the following axiom and inference rules:

(Act) $\quad \mu.E \xrightarrow{\mu} E$

(Res) $\dfrac{E \xrightarrow{\mu} E'}{E\backslash\alpha \xrightarrow{\mu} E'\backslash\alpha}$ when $\mu \notin \{\alpha, \alpha^-\}$ $\qquad$ (Rel) $\dfrac{E \xrightarrow{\mu} E'}{E[\Phi] \xrightarrow{\Phi\mu} E'[\Phi]}$

(Sum_1) $\dfrac{E_1 \xrightarrow{\mu} E_1'}{E_1 + E_2 \xrightarrow{\mu} E_1'}$ $\qquad$ (Sum_2) $\dfrac{E_2 \xrightarrow{\mu} E_2'}{E_1 + E_2 \xrightarrow{\mu} E_2'}$

(Asyn_1) $\dfrac{E_1 \xrightarrow{\mu} E_1'}{E_1 \mid E_2 \xrightarrow{\mu} E_1' \mid E_2}$ $\qquad$ (Asyn_2) $\dfrac{E_2 \xrightarrow{\mu} E_2'}{E_1 \mid E_2 \xrightarrow{\mu} E_1 \mid E_2'}$

(Syn) $\dfrac{E_1 \xrightarrow{\lambda} E_1' \;\; E_2 \xrightarrow{\lambda^-} E_2'}{E_1 \mid E_2 \xrightarrow{\tau} E_1' \mid E_2'}$ $\qquad$ (Rec) $\dfrac{E[\mathrm{rec}\, x.E/x] \xrightarrow{\mu} E_1}{\mathrm{rec}\, x.E \xrightarrow{\mu} E_1}$

The subset of CCS which does not comprise restriction and relabelling we call RCCS. Hence, syntactically, $\Sigma_{\mathrm{RCCS}} = \Sigma_{\mathrm{CCS}} - (\{\backslash\alpha, \alpha \in \Delta\} \cup \{[\Phi], \Phi$ is a permutation of $\mathcal{M}$ preserving $^-$ and $\tau\})$, and the closed guarded RCCS terms form the set $\mathrm{RCCS}_{\mathcal{M}}$. Semantically, the LTS $\mathcal{T}_{\mathrm{RCCS}} = (\mathrm{RCCS}_{\mathcal{M}}, \mathcal{M}, \{\xrightarrow{\mu} \mid \mu \in \mathcal{M}\})$ is obtained by Definition 4.1, by dropping rules (Res) and (Rel).

According to Milner's two-step approach, agents must be identified if they give rise to the same *behaviour*. Such a notion is defined in terms of the more elementary notion of *observation*: the behaviour of an agent is what can be observed from it. The observation out of a transition $s \xrightarrow{\mu} s'$ is action μ. A standard tool for defining behavioural equivalences of this kind relies upon the notion of *interleaving strong bisimulation* [37,34]: s and s' are equivalent iff, for all $\mu \in \mathcal{M}$, each μ-sucessor of s is

equivalent to some μ-successor of s', and vice versa. Maximal strong bisimulation is a congruence over CCS agents. Among the various equations which are sound for strong congruence, we mention that parallel composition is associative, commutative and having nil as neutral element.

4.2. An algebraic view of CCS operational semantics

Now we present the operational definition in terms of a graph $N_{CCS} = (V_{CCS}, T_{CCS}, \partial_0, \partial_1)$ with labelled transitions for CCS, where both V_{CCS}, T_{CCS} are algebras. Following [35, 18, 17], an SOS specification, and thus its associated transition system, can be described as a two-sorted algebra, where the sorts are states and transitions.

As far as states are concerned, it is immediate to observe that the CCS terms form an algebra. However, only a part of them is relevant for the operational semantics: the closed guarded terms, called *agents*, which are the states of Milner's transition system.[8]

Definition 4.2 (*The algebra of CCS states*). The set V_{CCS} of CCS states, ranged over by u, v, w, is obtained by making the quotient of CCS agents through the following axiom which captures the essence of recursion, i.e. that of "unfolding":

$$\text{rec } x.E = E[\text{rec } x.E/x]$$

In other words, the set of nodes V_{CCS} is composed of all the recursive terms, freely generated by the syntax modulo the recursion axiom, which are closed and guarded.

The CCS transitions in SOS style, having the format $v \xrightarrow{\mu} v'$, have been defined by a set of axioms and inference rules, i.e. by a deductive system. Here, we characterize the set of transitions as an algebra: the axioms represent the set of the generators and the inference rules are the operations. In this way, the terms of the algebra T_{CCS} denote the *proofs* of the transitions in the corresponding SOS specification (see also [5] for a transition system of proved transitions). Furthermore, every term of T_{CCS} is labelled with an action in $\Lambda \cup \{\tau\}$. To help intuition, a transition is represented in the format

$$t : v \xrightarrow{\mu} v', \quad \text{where } t \text{ is a proof term and } v \xrightarrow{\mu} v' \text{ is the corresponding SOS triple.}$$

In CCS not all transitions can be synchronized, but only those labelled by complementary actions. Nonetheless, we want to define a total operation of synchronization; thus, we introduce a special symbol $*$, labelling error transitions. The choice

[8] In a sense, the algebra is partial, or better it is total but we restrict our attention only those terms which are well-typed, i.e. closed and guarded. Recently, *typed algebras* and *equational type logic* have been proposed to this aim [30]; therefore, we could more rigorously redefine the algebra in this setting, which however gives rise sometimes to rather boring definitions when nontrivial examples are taken into account.

of a CCS-like synchronization algebra is part of our case study, but also different synchronization algebras (and different operators) might be considered as well.

Definition 4.3 (*The algebra of transitions*). T_{CCS} is the free algebra generated by the following constants (determined by **Act**) and operations, where $t: v_1 \xrightarrow{\mu} v_2$ and $t': v_1' \xrightarrow{\mu'} v_2'$ range over transitions and v over V_{CCS}.

(Act)	$[\mu, v\rangle : \mu.v \xrightarrow{\mu} v$	for any $\mu \in \Lambda \cup \{\tau\}$
(Sum <)	$t < + v : v_1 + v \xrightarrow{\mu} v_2$	
(Sum >)	$v + > t : v + v_1 \xrightarrow{\mu} v_2$	
(Res)	$t\backslash\alpha : v_1\backslash\alpha \xrightarrow{\mu'} v_2\backslash\alpha$	with $\mu' :=$ if $\mu \notin \{\alpha, \alpha^-\}$ then μ else*
(Rel)	$t[\Phi] : v_1[\Phi] - \Phi(\mu) \rightarrow v_2[\Phi]$	
(Com-)	$t \rfloor v : v_1 \mid v \xrightarrow{\mu} v_2 \mid v$	
(Com-⌊)	$v \lfloor t : v \mid v_1 \xrightarrow{\mu} v \mid v_2$	
(Sync)	$t \mid t' : v_1 \mid v_1' \xrightarrow{\mu''} v_2 \mid v_2'$	with $\mu'' :=$ if $\mu' = \mu^-$ then τ else $*$

where $*$ is a special error symbol. We restrict our attention to transitions which are not $*$-labelled.[9]

In order to properly define graph $N_{CCS} = (V_{CCS}, T_{CCS}, \partial_0, \partial_1)$, functions ∂_0 and ∂_1 remain to be defined. Nonetheless, their definition is implicityly given in the algebra of transitions: if $t: u - \mu \rightarrow v$, then $\partial_0(t) = u$ and $\partial_1(t) = v$. With $N_{CCS}(E)$ we denote the subgraph of N_{CCS} reachable from E.

As already mentioned, a term of the algebra denotes a derivation of a transition in the SOS deductive system in Definition 4.1. In general, an SOS transition has associated more than one proof term, as many derivations can give rise to the same SOS triple. For example, the SOS transition $\alpha.\mathrm{nil} + \alpha.\mathrm{nil} \xrightarrow{\alpha}$ nil has two possible derivations, denoted by the proof terms $[\alpha, \mathrm{nil}\rangle < + \alpha$ and $\alpha + > [\alpha, \mathrm{nil}\rangle$.

Graph N_{CCS} induces an obvious LTS $\mathcal{N}_{CCS}$, where the set of states is V_{CCS}, $\mathcal{M}$ the set of labels and a (SOS) transition $v_1 \xrightarrow{\mu} v_2$ does exist iff there exists a (proved) transition $t: v_1 \xrightarrow{\mu} v_2$. This means that the set of transition in $\mathcal{N}_{CCS}$ is isomorphic to the quotient of the algebra of transitions in N_{CCS} with respect to the following conditional axiom, where $t: v_1 \xrightarrow{\mu} v_2$ and $t': v_1' \xrightarrow{\mu'} v_2'$:

$$t = t' \quad \text{if } v_1 = v_1',\ v_2 = v_2' \text{ and } \mu = \mu'.$$

It is immediate to observe that $\mathcal{N}_{CCS}$ and $\mathcal{T}_{CCS}$ are bisimilar. Indeed, even though V_{CCS} is the quotient of the states of $\mathcal{T}_{CCS}$ via the recursion axiom, the states involved in

[9] Also in this case we should exploit typed algebras for dealing with the inherently partial operation of synchronization. For the sake of simplicity in the exposition, we prefer to work with an explicit representation of the error element.

the axiom are bisimilar in $\mathscr{T}_{CCS}$. On the other hand, transitions in $\mathscr{N}_{CCS}$ and $\mathscr{T}_{CCS}$ are the same, up to the recursion axiom. Therefore, we can safely continue our investigation considering the algebraic view of CCS operational semantics in place of its classic set-theoretic formulation.

5. SCONE: a simple calculus of nets

In this section we introduce our simple calculus of nets (SCONE), following as much as possible the algebraic formulation of Plotkin's paradigm, exemplified in the previous section.

Definition 5.1 (*The algebra of SCONE markings*). The set of recursive terms over $\Sigma_{SCONE}=\bigcup_{n\geq 0}\Sigma_n$ is defined by the following BNF-like notation:

$$M ::= x \mid op(E_1, E_2, \ldots, E_k) \mid \mathrm{rec}\, x.M$$

where x is any element in a (possibly infinite) set of variables Var, rec x.– is the binding construct, $op\in\Sigma_k$ and the signature Σ_{SCONE} consists of the following operators:

$$\Sigma_0=\{\mathrm{nil}\}, \qquad \Sigma_1=\{\mu. \mid \mu\in\mathscr{M}\},$$

$$\Sigma_2=\{\oplus, +\}, \qquad \Sigma_n=\emptyset, \quad \forall n>2.$$

The algebra is quotiented by the following axioms:

$$M\oplus M'=M'\oplus M \qquad M\oplus(M'\oplus M'')=(M\oplus M')\oplus M'' \qquad M\oplus \mathrm{nil}=M,$$

$$\mathrm{rec}\, x.M=M[\mathrm{rec}\, x.M/x]$$

Only a subset of the terms is relevant for the operational semantics: those closed and guarded, which are the markings of our net. Markings are ranged over by u, v, w (with abuse of notation).

The set of nodes V_{SCONE} is composed of all the closed, guarded terms, freely generated by the syntax modulo the axioms. Let S_{SCONE} be the set of the terms in V_{SCONE} generated by the following syntax:

$$N ::= \mu.M \mid M+M$$

Thus, $V_{SCONE}=(S_{SCONE})^{\oplus}$. V_{SCONE} is the free commutative monoid of nodes over a set of *places*[10] S_{SCONE}, having nil as neutral element. Hence, V_{SCONE} has, on nodes, the algebraic structure of a net.

Intuitively, $\mu.v$ is the place from which a μ-labelled transition reaches marking v; $v+v'$ is a place from which two choice transitions reach v and v', respectively; $v\oplus v'$ is the multiset union of v and v'. Term nil, being the neutral element, is not considered a place, rather, it denotes absence of a place.

[10] Because of the recursion axiom, rec $x.v$ is a place if and only if $v[\mathrm{rec}\, x.v/x]$ is so.

The general syntactical form of SCONE transitions is $t: v \xrightarrow{\mu} v'$, where v and v' are the source and the target of the transition, respectively, μ is its label, and t is a term of the algebra of transition proofs, whose operations are in a one-to-one correspondence with the inference rules of the calculus and whose generators are the axioms of the calculus.

Definition 5.2 (*Algebra of SCONE Transitions*). Let ε be a special unobservable action, $\varepsilon \notin \mathcal{M}$. The transitions in T_{SCONE} are generated by the following constants and operation (determined by *sync*), where $t: v_1 \xrightarrow{\mu} v_2$ and $t': v_1' \xrightarrow{\mu'} v_2'$ range over transitions and v over V_{SCONE}.

(*act*) $[\mu, v\rangle : \mu.v \xrightarrow{\mu} v$ for any $\mu \in \Lambda \cup \{\tau\}$

(*sum-<*) $v \lll + v' : v + v' \xrightarrow{\varepsilon} v$

(*sum->*) $v' + \ggg v : v' + v \xrightarrow{\varepsilon} v$

(*sync*) $t \mid t' : v_1 \oplus v_1' \xrightarrow{\mu''} v_2 \oplus v_2'$ with $\mu'' := \text{if } \mu' = \mu^- \text{ then } \tau \text{ else } *$

where ε is a special unobservable action and $*$ is the error symbol; furthermore, the operation of synchronization is subject to the following axiom of commutativity:

$$t \mid t' = t' \mid t.$$

The generators of the algebra are of two kinds: action prefixing and local, internal choice transitions. The only operation for building new transitions from existing ones is synchronization. The intuition behind $t_1 \mid t_2$ is that it is a new transition, whose source and target are the multiset union of the two and whose label is the synchronization of the two. This commutativity axiom is imposed because both transitions $t_1 \mid t_2$ and $t_2 \mid t_1$ have the same pre-set, the same post-set and the same label, and there is no observable reason for considering them different.

It could be interesting to give a look at the shape of net transitions. Transitions may have several post-places (sometimes none), but either one pre-place (in the case of action prefixing and internal choice) or two pre-places (in the case of synchronization[11] of two transitions). Moreover, also loop transitions are allowed, due to recursion. Transitions which are $*$-labelled may have more than two pre-places; however, the study of an algebra of shape-constructors representing all possible net transitions (and in general the study of an algebra generating richer classes of nets) is outside the scope of the paper.

SCONE is a place/transition Petri net $N_{\text{SCONE}} = (V_{\text{SCONE}}, T_{\text{SCONE}}, \partial_0, \partial_1)$ because V_{SCONE} is the free commutative monoid over S_{SCONE}, T_{SCONE} is the set of transitions and $\partial_0, \partial_1 : T_{\text{SCONE}} \to V_{\text{SCONE}}$ are defined as usual: given a transition $t: v \xrightarrow{\mu} v'$, $\partial_0(t) = v$ and $\partial_1(t) = v'$.

[11] Of course, this shape does strictly depend on the chosen CCS synchronization algebra. We recall that such a choice is motivated only by our interest in showing the implementation mapping from CCS to SCONE in the next section.

Being SCONE a net, we can apply the algebraic construction of Section 3 to gain the symmetric strict monoidal category $\mathscr{P}[N_{\text{SCONE}}]$ of its computations observed as concatenable processes. In this setting, the notion of reachability of places and transitions can be naturally defined in $\mathscr{P}[N_{\text{SCONE}}]$: a place s (transition t) is reachable from a marking v if and only if there exists a computation ξ starting from v and ending u such that s occurs in u (t is a subterm of ξ).

SCONE enjoys a nice property: the subpart of the global SCONE net reachable starting from a certain marking v is finite, for any v. Given a marking v, let $N_v=(S_v^{\oplus}, T_v, \partial_{0v}, \partial_{1v})$ denote the subnet of N_{SCONE} reachable from v. For a finite subnet we mean a net where the set S_v, contained in S_{SCONE}, and the set T_v, contained in T_{SCONE}, are finite. Therefore, we have to first define how to associate a finite set S_v to a SCONE marking v; then, to prove that all the computations starting from v reach markings in $S_v^{\oplus}$. This is a sufficient condition because, by construction, the set T_v is finite. Infact, a place $v'+v''$ has exactly two transitions starting from it, while a place $\mu.v'$ has one (local) μ-transition, plus a set of synchronization transitions, one for each place of the form $\mu^{-}.v''$ in S_v. By a pure combinatorial analysis, since S_v is always finite, there must be a finite set of reachable transitions. Therefore, $N_v=(S_v^{\oplus}, \partial_{0v}, \partial_{1v})$ is definitely a finite net.

Definition 5.3 (*Places associated to a marking*). Let v be a closed, guarded marking. Let $[\![v]\!]$ be the set of the subterms of v defined below:

$$[\![\text{nil}]\!]=\emptyset \qquad [\![x]\!]=\emptyset$$

$$[\![\mu.v]\!]=\{\mu.v\}\cup[\![v]\!] \qquad [\![v+v']\!]=\{v+v'\}\cup[\![v]\!]\cup[\![v']\!]$$

$$[\![v\oplus v']\!]=[\![v]\!]\cup[\![v']\!] \qquad [\![\text{rec}\,x.v]\!]=[\![v]\!]\,[\text{rec}\,x.v/x]$$

where, in the rule of recursion, the free variable x is to be replaced by $\text{rec}\,x.v$.

For instance, $[\![\text{rec}\,x.\alpha.x]\!]=[\![\alpha.x]\!]\,[\text{rec}\,x.x/x]=(\{\alpha.x\}\cup[\![x]\!])[\text{rec}\,x.\alpha./x]=\{\alpha.x\}$ $[\text{rec}\,x.\alpha.x/x]=\{\alpha.(\text{rec}\,x.\alpha.x)\}$; moreover, $[\![\text{rec}\,x.(\alpha.x\oplus\beta.x)]\!]=[\![\alpha.x\oplus\beta.x]\!]\,[\text{rec}\,x.(\alpha.x\oplus\beta.x/x]=(\{\alpha.x\}\cup\{\beta.x\})[\text{rec}\,x.(\alpha.x\oplus\beta.x)/x]=\{\alpha.\text{rec}\,x.(\alpha.x\oplus\beta.x),\beta.\text{rec}\,x.(\alpha.x\oplus\beta.x)\}$

Lemma 5.4. *Let v be a closed and guarded marking. Then, the following hold:*

(i) $[\![v]\!]\subseteq S_{\text{SCONE}}$,

(ii) $[\![v]\!]$ *is finite,*

(iii) $v'\in[\![v]\!]$ *implies v' is a closed and guarded place,*

(iv) $v\in[\![v]\!]^{\oplus}$, *i.e. v is a marking on the set of places $[\![v]\!]$,*

(v) $v'\in[\![v]\!]^{\oplus}$ *if and only if* $[\![v']\!]\subseteq[\![v]\!]$.

Proof. Immediate by structural induction. □

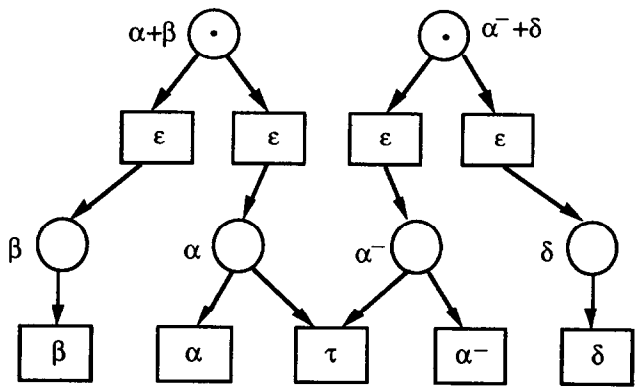

Fig. 4. The relevant SCONE subnet for the marking $(\alpha+\beta)\oplus(\alpha^-+\delta)$.

Theorem 5.5 ($[\![v]\!]$ is closed w.r.t. computations). *For any computation* $\xi\in\mathscr{P}[N_{\text{SCONE}}]$, $\partial_0(\xi)=v$, $\partial_1(\xi)=w$, *we have that* $w\in[\![v]\!]^{\oplus}$.

Proof. By induction on the structure of ξ. The base cases are symmetries and net transitions. For symmetries, the thesis is immediate by Lemma 5.4(iv). For net transitions, we have to proceed by induction on the structure of the transition. If t is $[\mu, v\rangle : \mu.v \xrightarrow{\mu} v$, then $v\in[\![\mu.v]\!]^{\oplus}$ since, by Definition 5.3, $[\![\mu.v]\!]=\{\mu.v\}\cup[\![v]\!]$ and $v\in[\![v]\!]^{\oplus}$ by Lemma 5.4(iv). With a similar argument, the thesis holds also when t is $v\ll+v' : v+v'\xrightarrow{\varepsilon}v$ and $v'+\gg v : v'+v\xrightarrow{\varepsilon}v$. In the case of $t\,|\,t' : u\oplus u'\xrightarrow{\tau}v\oplus v'$, we have by inductive hypothesis that $v\in[\![u]\!]^{\oplus}$ and $v'\in[\![u']\!]^{\oplus}$. Thus, $v\oplus v'\in([\![u]\!]\cup[\![u']\!])^{\oplus}$, from which $v\oplus v'\in[\![u\oplus u']\!]^{\oplus}$ by Definition 5.3. If the computation is $\xi_1;\xi_2$, where $\partial_0(\xi_1)=v$, $\partial_1(\xi_1)=u$, and $\partial_0(\xi_2)=u$, $\partial_1(\xi_2)=w$, then $w\in[\![v]\!]^{\oplus}$ by observing that $u\in[\![v]\!]^{\oplus}$ and we $w\in[\![u]\!]^{\oplus}$ by inductive hypothesis; in fact, by Lemma 5.4(v) $[\![w]\!]\subseteq[\![u]\!]\subseteq[\![v]\!]$, from which the thesis follows. The last case considers the computation $\xi_1\otimes\xi_2$, where $\partial_0(\xi_1)=v_1$, $\partial_1(\xi_1)=w_1$, and $\partial_0(\xi_2)=v_2$, $\partial_1(\xi_2)=w_2$. By inductive hypothesis we have $w_1\in[\![v_1]\!]^{\oplus}$ and $w_2\in[\![v_2]\!]^{\oplus}$. Like in the case of synchronization, $w_1\oplus w_2\in[\![v_1\oplus v_2]\!]^{\oplus}$. □

Example 5.6. The reachable subnet for the making $(\alpha+\beta)+(\alpha^-+\delta)$ is depicted[12] in Fig. 4.

Example 5.7. The second example shows the subnet reachable from $\gamma+(\alpha\oplus\beta)$, depicted in Fig. 5. This marking corresponds to the RCCS agent $\gamma+(\alpha|\beta)$, which is

[12] Of course, when describing nets graphically, we abandon the presentation of nets as graphs with algebraic structure for a more traditional representation as bipartite graphs. Note that transitions labelled by ε denote local choices. Note also that, since nil is the neutral element, it does not have a corresponding place.

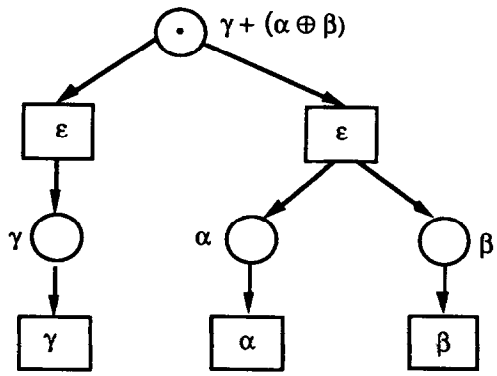

Fig. 5. The SCONE subnet for $\gamma+(\alpha \oplus \beta)$.

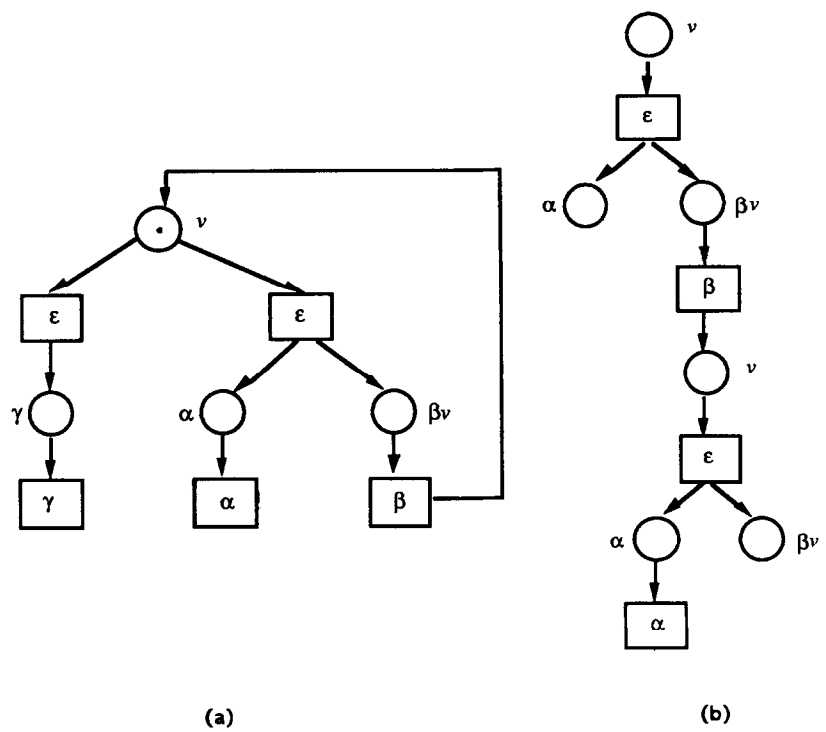

Fig. 6. (a) The SCONE subset for the place $v=\mathrm{rec}\, x.\gamma+(\alpha \oplus \beta.x)$. (b) The process in $\mathscr{P}[N_{\mathrm{SCONE}}]$ associated to the computation $(\gamma+\gg(\alpha \oplus \beta v)); (\alpha \otimes [\beta, v\rangle); (\alpha \otimes (\gamma+\gg(\alpha \oplus \beta v\rangle)))); (\alpha \otimes ([\alpha, \mathrm{nil}\rangle \otimes \beta v))$.

considered a difficult agent to model in terms of net theory because of an interweaving of nondeterministic and parallel operators (see Example 10.1).

Example 5.8. The third example shows the net (in Fig. 6(a)) reachable from the place corresponding to the evaluation of the recursive term $v=\mathrm{rec}\, x.\gamma+(\alpha \oplus \beta.x)$. In

Fig. 6(b) is shown a graphical representation of the process in $\mathscr{P}[N_{\text{SCONE}}]$ associated to the computation[13]

$$(\gamma + \gg (\alpha \oplus \beta v)); (\alpha \otimes [\beta, v\rangle); (\alpha \otimes (\gamma + \gg (\alpha \oplus \beta v))); (\alpha \otimes ([\alpha, \text{nil}\rangle \otimes \beta v)).$$

This example also illustrates that the finiteness result stated in Theorem 5.5 is also due to the fact that the nets are not 1-safe. Indeed, in this example many tokens may be stored in the place α when playing the token game from the initial 1-safe marking v.

6. Implementing RCCS into SCONE

Relating different languages whose operational semantics have been defined in terms of graphs with algebraic structure is now an easy task: we have simply to give a *denotational semantics* of the first in terms of the second, i.e. we have to define a (two-sorted theory) morphism between the two graphs. In this way, not only the terms of the languages (the nodes of the graph) are mapped, but also their operational behaviour is *pointwise* translated. Furthermore, if the target language is a Petri net calculus, then we get a distributed implmentation for the language.

Here, we provide RCCS with a distributed implementation over the net of SCONE. Formally, the mapping is a pair of functions $\langle f, g\rangle$, where g maps RCCS states to SCONE markings and f maps RCCS transitions to arrows of $\mathscr{P}[N_{\text{SCONE}}]$. While the definition of g is immediate, f is nontrivial, as some RCCS operations have no obvious counterpart in SCONE. For instance, any RCCS external choice transition is implemnted as a SCONE computation composed of internal choice(s) ending with an action prefix. This idea is related to the notion of implementation morphism of [32]. The analogy is expressed by the fact that our denotational semantics maps *basic* operators of the algebra of RCCS transitions to *derived* operators of the algebra of SCONE computations as well as an implementation morphism maps transitions to net computations. As the implementation of an agent E via $\langle f, g\rangle$ is an algebraic theory in $\mathscr{P}[N_{\text{SCONE}}]$, the *underlying net* for E is the SCONE net composed of those places and transitions used to build $\langle f, g\rangle (N_{\text{RCCS}}(E))$.

Let us try to define the mapping with an example, just to point out some technical problems. Consider the RCCS agent $(\alpha + \beta)|(\alpha^- + \delta)$, which can be mapped to the SCONE marking $(\alpha + \beta) \oplus (\alpha^- + \delta)$ (see Example 5.6). The RCCS transition

$$([\alpha, \text{nil}\rangle < + \beta)|([\alpha^-, \text{nil}\rangle < + \delta)$$

which represents the synchronization of α and α^- (thus an elementary step in the transition system for RCCS) should be mapped to the net computation (thus to a derived operator)

$$(\alpha \ll \beta \otimes \alpha^- \ll + \delta); ([\alpha, \text{nil}\rangle | [\alpha^-, \text{nil}\rangle)$$

[13] In the following, for the sake of simplicity, we often use the coercion "state for its identity", i.e, u for $\text{id}(u)$.

where firstly the internal choices are executed in parallel and then the synchronization is performed.

We work out how the mapping is defined by induction on the syntax. For CCS generators the mapping is trivial: $f([\alpha, \mathrm{nil}\rangle)=[\alpha, \mathrm{nil}\rangle$, $f([\alpha^-, \mathrm{nil}\rangle)=[\alpha^-, \mathrm{nil}\rangle$. Then, in the case of nondeterministic choice we have an interesting situation of nontrivial mapping:

$$f([\alpha, \mathrm{nil}\rangle < + \beta)=\alpha \lll + \beta; f([\alpha, \mathrm{nil}\rangle)=\alpha \lll \beta; [\alpha, \mathrm{nil}\rangle,$$

$$f([\alpha^-, \mathrm{nil}\rangle < + \delta)=\alpha^- \lll + \delta; f([\alpha^-, \mathrm{nil}\rangle)=\alpha^- \lll + \delta; [\alpha^-, \mathrm{nil}\rangle.$$

The choice operator is mapped to a derived operator, i.e. to a suitable combination of local choice and sequential composition, so that a global choice CCS transition is implemented as a sequence of (at least two) transitions, the first of which is a local choice, resulting in a many-step computation of SCONE. Indeed, any global choice can be seen as composed of two steps: the choice of the subcomponents and the execution of an action from the selected components. However, in order to preserve the correct semantics of the language, these steps are to be executed *atomically* (see Section 11), and the mapping f is the right mean to express this notion. Finally.

$$f(([\alpha, \mathrm{nil}\rangle < + \beta) | ([\alpha^-, \mathrm{nil}\rangle < + \delta)) = f([\alpha, \mathrm{nil}\rangle < + \beta) | f([\alpha^-, \mathrm{nil}\rangle < + \delta).$$

Now notice that $f([\alpha, \mathrm{nil}\rangle < + \beta) | f([\alpha^-, \mathrm{nil}\rangle < + \delta)$ is not defined in $\mathscr{P}[N_{\mathrm{SCONE}}]$, since the operator of synchronization is not defined *for computations*, but only for net transitions! Therefore, we should define an algebra $\mathscr{G}_{N_{\mathrm{SCONE}}}$, obtained enriching the algebra of category $\mathscr{P}[N_{\mathrm{SCONE}}]$ with the operator of synchronization, and expressing which net computation the term $f(t_1) | f(t_2)$ should represent.

6.1. *Transactions and synchronization*

Defining an operator of synchronization for concatenable processes is a difficult task, since the operation seems to be intrinsically nondeterministic. As an example, it is not clear what should be the result of the synchronization of a transition t with the parallel composition $t' \otimes t''$. We might say that either the synchronization is not possible, or t can be synchronized with either t' or t'', or even with both, but apparently there is no sensible choice. Luckily, in the present case, only a restricted family of concatenable processes is interesting for synchronization: the processes in $\mathscr{P}[N_{\mathrm{SCONE}}]$ which are the targets of RCCS transitions according to function f. For this family, it turns out that a *deterministic* operation of synchronization can be defined which exactly reflects our intuition about CCS synchronization.

The class of the relevant processes we call *transactions*, is formed by those concatenable processes $C=\langle \varphi, \theta, \kappa\rangle$ where, additionally, the underlying plain process has the property that there is a (unique) basic transition, called *commit*, which is greater than all the others in the partial ordering.

Definition 6.1 (*Net transactions and RCCS transactions*). A *net transaction* for a net N is an equivalence class of terms in $\mathscr{P}[N]$ such that there are no terms in the class of the form $\eta;(u \otimes t \otimes t');p$, i.e. with two concurrent final[14] transitions. A *RCCS transaction* is a net transaction for SCONE where all the transitions are ε-labelled, with the exception of the commit transition. A RCCS transaction with the μ-labelled commit transition is also called a μ-transaction.

We will often use the convenient shorthand $\eta;(u \otimes t);p$ – where η is a computation, t is the commit, u an identity and p a symmetry – for a transaction (but also computations which are not transactions can have the same form). Indeed, it is easy to see that any computation ξ can be always reduced to the format $p_0;(u_1 \otimes t_1);p_1;,\ldots;p_{n-1};(u_n \otimes t_n);p_n$ by applying functoriality of $\otimes$. The commit transition, being caused by all the others, will always be the last one. Now we want to prove that there is some standard representative for transactions. To this aim, we need some auxiliary definitions and results.

Definition 6.2 (*Merge symmetry*). A symmetry $p: u \oplus v \to u \oplus v$, $p = \langle \sigma_{a_1}, \ldots, \sigma_{a_k} \rangle$ is a (u,v)-*merge*, where $u = n_1 a_1 \oplus \cdots \oplus n_k a_k$ and $v = m_1 a_1 \oplus \cdots \oplus m_k a_k$, iff $\sigma_{a_h}(i) \leqslant \sigma_{a_h}(j)$ whenever $1 \leqslant i \leqslant j \leqslant n_h$ or $n_h + 1 \leqslant i \leqslant j \leqslant n_h + m_h$, $h = 1, \ldots, k$.

A merge symmetry is an arbitrary merge of two identity symmetries, the former on u and the latter on v. As the condition $\sigma_{a_h}(i) \leqslant \sigma_{a_h}(j)$ holds in the two intervals $1 \leqslant i \leqslant j \leqslant n_h$ and $n_h + 1 \leqslant i \leqslant j \leqslant n_h + m_h$, we are sure that no exchange is possible within u or within v. Indeed, any symmetry on $u \oplus v$ can be seen as composed of two local exchaging symmetries followed by a (u,v)-merge.

Lemma 6.3 (*Unique decomposition of symmetries*). *Given u and v, any symmetry $p: u \oplus v \to u \oplus v$ can be uniquely decomposed as $p = (p_1 \otimes p_2);p'$ with $p_1: u \to u$, $p_2: v \to v$, and p' being a (u,v)-merge.*

Lemma 6.4 (Unique decomposition of transaction, up to). *Any transaction ξ can be uniquely decomposed in right-standard form $\eta';(u \otimes t);p'$ where $t: w \to v$ and p' is a (u,v)-merge, up to equivalence of η'.*

Proof. Given a transaction in the form $\eta;(u \otimes t);p$, symmetry p can be decomposed as $(p_1 \otimes p_2);p'$ due to Lemma 6.3; hence, $\xi = \eta;(u \otimes t);(p_1 \otimes p_2);p' = \eta;(p_1 \otimes w);(u \otimes t;p_2));p' = \eta';(u \otimes t);p'$ due to the axiom stating that generators absorb symmetries. The decomposition $\eta';(u \otimes t);p'$ is unique, up to equivalence of η'. In fact, two equivalent right-standard decompositions $\eta;(u \otimes t);p$ and $\eta';(u' \otimes t');p'$ must evaluate to the same concatenable process $C = \langle \varphi, \theta, \kappa \rangle$. Since no endomorphism may map the commit to another transition, then $u = u'$ and $t = t'$.

[14] As the commit transition is caused by all the others, there exists no transition concurrent with the commit.

Furthermore, the induced labelling κ on the destinations which are post-conditions of t uniquely determines $p=p'$. Finally, η and η', which must evaluate to the same concatenable process, need not be the same term in general. □

Of course, we are not forced to leave t on the right; indeed, for any transaction ξ there is also a left-standard form $\eta'';(t\otimes u);p''$ where p'' is a (v,u)-merge.

With this notion in mind, a natural deterministic definition of synchronization between two transactions consists of putting in parallel the two processes but synchronizing the two commits to become the commit for the resulting process. This operation is associative, and commutative up to natural isomorphism [23, p. 214]. In the next definition we introduce a new algebra by enriching $\mathscr{P}[N_{\text{SCONE}}]$ with a derived operation [ ] of synchronization defined on standard representatives of transactions, expressing the fact that the synchronization of two transactions is again a transaction.

Definition 6.5 (*from* $\mathscr{P}[N_{\text{SCONE}}]$ *to* $\mathscr{G}_{N_{\text{SCONE}}}$). $\mathscr{G}_{N_{\text{SCONE}}}=(V_{\text{SCONE}},\mathsf{T},\partial_0,\partial_1)$ is the same graph[15] as $\mathscr{P}[N_{\text{SCONE}}]$ with the extra partial[16] operation [ ] defined on transactions. The operation is subject to the following axiom, which defines it as a derived operator inside the algebra of $\mathscr{P}[N_{\text{SCONE}}]$: if two transactions $\xi=\eta;(u\otimes t);p$ and $\xi'=\eta';(t'\otimes u');p'$ are in standard form (right- and left-respectively), then

$$\xi[\,]\xi'=\eta\otimes\eta';u\otimes(t|t')\otimes u';p\otimes p'.$$

Before entering into the details of the results we prove, we will clarify the definition of this operation. The simpler case is when the RCCS transactions consist of exactly one step and there are no symmetries. In such a case, the operation states that identities do not participate to synchronizations.

$$(v\otimes t)[\,](t'\otimes v')=v\otimes(t|t')\otimes v',\quad t,t'\in T_{\text{SCONE}}.$$

This example also shows that for net transitions, which are transactions in standard form, $t'[\,]t'=t|t'$. The synchronization operation represents a kind of composition of transactions where the two commits are synchronized. This is one of the few deterministic ways of synchronizing two net computations, and certainly the only one meaningful for RCCS transactions.

Theorem 6.6 (*Synchronizations of transactions are transactions*) (see Fig. 7). (i) *Given two transactions* ξ *and* ξ', $\xi[\,]\xi'$ *is a transaction.*

(ii) *Given a* λ*-transaction* ξ *and a* λ^{-}*-transaction* ξ', $\xi[\,]\xi'$ *is a* τ*-transaction.*

[15] To be more rigorous, we should say that the two algebras are different but induce the same underlying graph.

[16] Note that this kind of partiality is different of that of sequential composition. In fact, [ ] is defined only on a restricted subset of computations (i.e. transactions), whilst ; is defined on all the computations provided that the constraints on source and target states are respected.

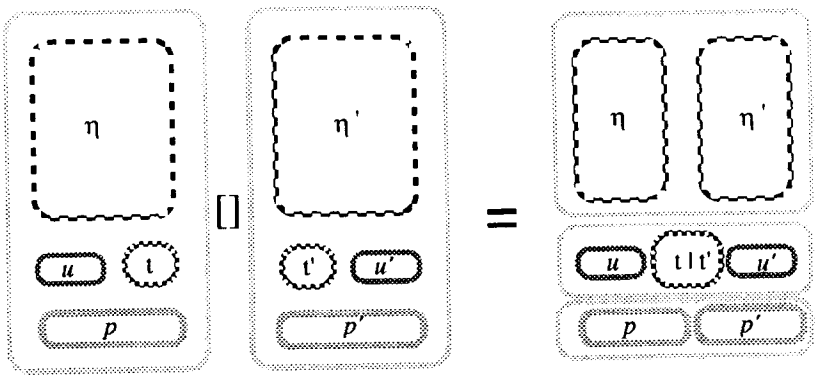

Fig. 7. A graphical representation of the synchronization operation.

Proof. (i) Let $\xi=\eta;(u\otimes t);p$ and $\xi'=\eta';(t'\otimes u');p'$ be transactions. By the synchronization axiom, we know that $\xi[\,]\xi'$ is the computation $\eta\otimes\eta';u\otimes(t|t')\otimes u';p\otimes p'$ which is a transaction, since $t|t'$ is the maximal transition in the associated concatenable process. In fact, $t|t'$ is caused by all the transitions which caused either t or t'. (ii) Immediate. □

6.2. *Mapping RCCS transitions to SCONE computations*

Now, we are ready to define the evaluation morphism from the two-sorted algebra of RCCS to the two-sorted algebra of $\mathscr{G}_{N_{\text{SCONE}}}$.

Definition 6.7 (*Implementing RCCS in SCONE*). The pair $\langle f,g\rangle: N_{\text{RCCS}}\to\mathscr{G}_{N_{\text{SCONE}}}$, $f: T_{\text{RCCS}}\to\mathsf{T}$ and $g: V_{\text{RCCS}}\to V_{\text{SCONE}}$, is defined as follows, where $t: u\to w$.

$g(\text{nil})=\text{nil}$ $\quad g(x)=x$

$g(\mu.v)=\mu.g(v)$ $\quad g(\text{rec}\,x.v)=\text{rec}\,x.g(v)$

$g(v+v')=g(v)+g(v')$ $\quad g(v|v')=g(v)\oplus g(v')$

$f([\mu,v\rangle)=[\mu,g(v)\rangle$

$f(t<+v)=g(u)\lll+g(v);f(t)$ $\quad f(v+>t)=g(v)+\ggg g(u);f(t)$

$f(v\lfloor t)=g(v)\otimes f(t)$ $\quad f(t\rfloor v)=f(t)\otimes g(v)$

$f(t_1|t_2)=f(t_1)[\,]f(t_2)=(\eta_1\otimes\eta_2);u_1\otimes(t_1|t_2)\otimes u_2;(p_1\otimes p_2)$

where $f(t_1)=\eta_1;(u_1\otimes t_1);p_1$ and $f(t_2)=\eta_2;(t_2\otimes u_2);p_2$ are in standard form.

It is easy to see that function g is surjective (given any marking M, transform all the occurrences of $\oplus$ into $|$, thus gaining a RCCS agent), but not injective (due to the fact that $|$ is implemented via the commutative monoidal operation $\oplus$). Moreover,

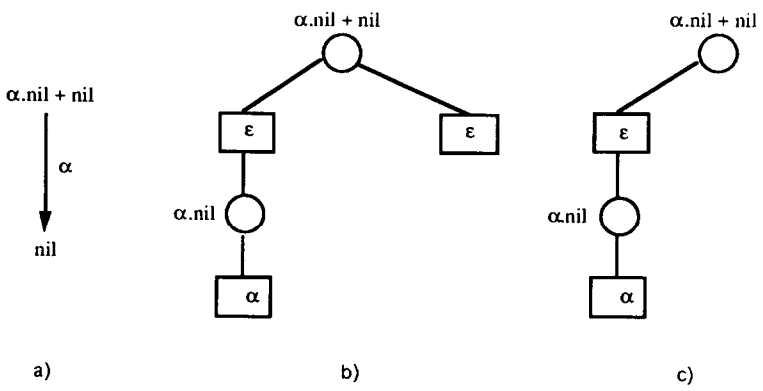

Fig. 8.

function f is neither injective (e.g. due to the noninjectivity of g, $[\mu, g(v)\rangle$ and $[\mu, g(v')\rangle$ can be the same SCONE transition) nor surjective. This latter case is very interesting, as it illustrates that certain SCONE transitions – e.g. $\alpha.\mathrm{nil} \lll +\beta.\mathrm{nil}$ – cannot be the image of any RCCS transition, rather only a part of its image. Furthermore, we want to show that certain internal choice transitions are not even part of the image of any RCCS transition.

Example 6.8. Consider the RCCS agent $E = \alpha.\mathrm{nil} + \mathrm{nil}$ and Fig. 8. The LTS $N_{\mathrm{CCS}}(E)$, in (a), is composed of one transition only, $[\alpha, \mathrm{nil}\rangle < +\mathrm{nil}$. Its associated marking is $\alpha.\mathrm{nil} + \mathrm{nil}$, a single place. However, the subnet reachable from this place, in (b), has more transitions. For example, $\alpha.\mathrm{nil} \lll +\mathrm{nil}$ is a choice transition which is only part of the image of $[\alpha, \mathrm{nil}\rangle < +\mathrm{nil}$. Conversely, $\alpha, \mathrm{nil} + \gg \mathrm{nil}$ is not even part of the image. The subnet in (c) represents the underlying net of $\langle f, g\rangle(N_{\mathrm{CCS}}(E))$. Notice, however, that this net is *not* the implementation, but only its support. The implementation needs also function f which specifies that e.g., $\alpha.\mathrm{nil} \lll +\mathrm{nil}$ is not a legal transition per se.

Proposition 6.9. *For each RCCS transition t, $f(t)$ is always a RCCS transaction.*

Proof. By structural induction (symmetric cases are omitted).

- $f([\mu, v\rangle) = [\mu, g(v)\rangle$: In this case the net computation is simply a net transition, which is of course a RCCS transaction.
- $f(t < +v) = g(u) \lll +g(v); f(t)$: By inductive hypothesis, $f(t)$ is a RCCS transaction. As the ε-transition $g(u) \lll +g(v)$ causes all the transitions of $f(t)$, and in particular its commit, the thesis holds.
- $f(t \rfloor v) = f(t) \otimes g(v)$: As $f(t)$ is a RCCS transaction, $f(t) \otimes g(v)$ is so, because $g(v)$ generates no transition in the process associated to the transaction.
- $f(t_1 | t_2) = f(t_1) [] f(t_2)$: By inductive hypothesis and Theorem 6.6. □

Proposition 6.10. *The pair* $\langle f,g\rangle : N_{\mathrm{RCCS}} \to \mathscr{G}_{N_{\mathrm{SCONE}}}$ *is a graph morphism, i.e.* $g(\partial_{i_{\mathrm{RCCS}}}(t)) = \partial_{i_{\mathrm{SCONE}}}(f(t))$, $i=0,1$, *for all transitions* t.

Proof. By structural induction (only two relevant cases are reported). For the sake of simplicity, ∂_{i_R} denotes $\partial_{i_{\mathrm{RCCS}}}$, while ∂_{i_S} denotes $\partial_{i_{\mathrm{SCONE}}}$.

$$\langle \partial_{0_S}(f(t<+v)), \partial_{1_S}(f(t<+v))\rangle$$
$$= \langle \partial_{0_S}(g(u) \lll +g(v); f(t)), \partial_{1_S}(g(u) \lll +g(v); f(t))\rangle$$
$$= \langle \partial_{0_S}(g(u) \lll +g(v)), \partial_{1_S}(f(t))\rangle = \langle g(u)+g(v), g(\partial_{1_R}(t))\rangle = \langle g(u+v), g(\partial_{1_R}(t))\rangle$$
$$= \langle g(\partial_{0_R}(t)+v), g(\partial_{1_R}(t<+v))\rangle = \langle g(\partial_{0_R}(t<+v)), g(\partial_{1_R}(t<+v))\rangle,$$
$$\langle \partial_{0_S}(f(t \,\lrcorner\, v)), \partial_{1_S}(f(t \,\lrcorner\, v))\rangle$$
$$= \langle \partial_{0_S}(f(t) \otimes g(v)), \partial_{1_S}(f(t) \otimes g(v))\rangle,$$
$$= \langle \partial_{0_S}(f(t)) \oplus \partial_{0_S}(g(v)), \partial_{1_S}(f(t)) \oplus \partial_{1_S}(g(v))\rangle$$
$$= \langle g(\partial_{0_S}(t)) \oplus g(v), g(\partial_{1_R}(t)) \oplus g(v)\rangle = \langle g(\partial_{1_R}(t)|v), g(\partial_{1_R}(t)|v)\rangle$$
$$= \langle g(\partial_{0_R}(t \,\lrcorner\, v)), g(\partial_{1_R}(t \,\lrcorner\, v))\rangle. \quad \square$$

Example 6.11. Consider $E=(\alpha+\beta)|(\alpha^-+\delta)$ and the SCONE subnet in Fig. 4. The initial marking of the subnet we are interested in is $g(E)=(\alpha+\beta)\oplus(\alpha^-+\delta)$. Transitions are mapped to computations as follows:

- $f([\sigma,\mathrm{nil}\rangle)=[\sigma,\mathrm{nil}\rangle$ for $\sigma\in\{\alpha,\beta,\alpha^-,\delta\}$,
- $f([\alpha,\mathrm{nil}\rangle < +\beta)=\alpha \lll \beta; f([\alpha,\mathrm{nil}\rangle)=\alpha \lll \beta; [\alpha,\mathrm{nil}\rangle$ and similarly for the other choices,
- $f(([\alpha,\mathrm{nil}\rangle < +\beta) \,\lrcorner\, (\alpha^-+\delta))=f([\alpha,\mathrm{nil}\rangle < +\beta) \otimes g(\alpha^-+\delta)$
  $=(\alpha \lll +\beta; [\alpha,\mathrm{nil}\rangle) \otimes (\alpha^-+\delta)$ and similarly for other asynchronous moves,
- $f(([\alpha,\mathrm{nil}\rangle < +\beta)|([\alpha^-,\mathrm{nil}\rangle < +\delta))=f([\alpha,\mathrm{nil}\rangle < +\beta)[\,]f([\alpha^-,\mathrm{nil}\rangle < +\delta)$
  $=(\alpha \lll +\beta; [\alpha,\mathrm{nil}\rangle)[\,](\alpha^- \lll +\delta; [\alpha^-,\mathrm{nil}\rangle)$
  $=(\alpha \lll +\beta \otimes \alpha^- \lll +\delta); ([\alpha,\mathrm{nil}\rangle|[\alpha^-,\mathrm{nil}\rangle)$.

Summing up,

$$f([\alpha,\mathrm{nil}\rangle < +\beta|[\alpha^-,\mathrm{nil}\rangle < +\delta)=(\alpha \lll \beta \otimes \alpha^- \lll +\delta); ([\alpha,\mathrm{nil}\rangle|[\alpha^-,\mathrm{nil}\rangle),$$

i.e. the choices are executed in parallel and then the synchronization is performed. Of course, the choices can also be done in any order, as proved through the following identification:

$$(\alpha \lll +\beta \otimes (\alpha^-+\delta)); (\alpha \otimes (\alpha^- \lll +\delta)); ([\alpha,\mathrm{nil}\rangle|[\alpha^-,\mathrm{nil}\rangle)$$
$$= ((\alpha \lll +\beta; \alpha) \otimes (\alpha^-+\delta; \alpha^- \lll +\delta)); ([\alpha,\mathrm{nil}\rangle|[\alpha^-,\mathrm{nil}\rangle) \quad \text{(applying functoriality)}$$
$$= (\alpha \lll +\beta \otimes \alpha^- \lll +\delta); ([\alpha,\mathrm{nil}\rangle)|[\alpha^-\mathrm{nil}\rangle) \quad \text{(cancelling identities)}$$
$$= ((\alpha+\beta; \alpha \lll +\beta) \otimes (\alpha^- \lll +\delta; \alpha^-)); ([\alpha,\mathrm{nil}\rangle)|[\alpha^-,\mathrm{nil}\rangle) \quad \text{(introducing identities)}$$
$$= ((\alpha+\beta) \otimes \alpha^- \lll +\delta); (\alpha \lll +\beta \otimes \alpha^-); ([\alpha,\mathrm{nil}\rangle)|[\alpha^-\mathrm{nil}\rangle) \quad \text{(applying functoriality)}$$

In the previous section we have pointed out that, for any SCONE marking v, its reachable subnet N_v is finite. Here we have mapped the whole RCCS transition system to the whole[17] SCONE net, via a graph morphism $\langle f,g\rangle$. As an immediate corollary we have that the subnet implementing any RCCS agent is always finite.

Corollary 6.12. *For any RCCS agent E, the SCONE subnet underlying $\langle f,g\rangle$ $(N_{\mathrm{RCCS}}(E))$ is finite.*

Proof. The net underlying $\langle f,g\rangle$ $(N_{\mathrm{CCS}}(E))$ is a subnet of $N_v=(S_v^{\oplus}, T_v, \partial_{0v}, \partial_{1v})$, where $v=g(E)$, by Proposition 6.10. N_v is finite by Theorem 5.5. □

7. Distributed semantics of RCCS

The semantics of RCCS is investigated via the mapping $\langle f,g\rangle: N_{\mathrm{RCCS}} \to \mathscr{G}_{N_{\mathrm{SCONE}}}$, which induces a quotient of states and computations of N_{RCCS}. The quotient on states is determined by the axioms stating that | is a commutative, monoidal, with nil as neutral element, as $\oplus$ is so. Some transitions are identified (e.g. $[\alpha,\mathrm{nil}\rangle\rfloor\beta$ and $\beta\lfloor[\alpha,\mathrm{nil}\rangle$), even if an axiomatic characterization within RCCS is not so easy. Here we show that if we extend homomorphically f to RCCS computations, the mapping will equate all the computations obtained by permuting transitions generating independent events.

Proposition 7.1. *Let v and v' be two states in V_{RCCS}. If $g(v)=g(v')$ then v and v' are interleaving strong bisimulation equivalent.*

Proof. By induction on the structure of v. It is based on the fact that the associativity, commutativity and "nil as neutral element" properties of the parallel operator | hold for strong congruence. □

Proposition 7.2. *Let t and t' be two transitions in T_{RCCS}. If $f(t)=f(t')$ then t and t' have the same label.*

Proof. By Proposition 6.9, $f(t)=f(t')$ is a RCCS transaction. It is easy to prove by induction that the labels of t and the commit of $f(t)$ are the same. Since the commit is unique, the thesis follows. □

It is not easy to characterize the identifications, induced by f, on RCCS transitions. Consider, e.g., two states v and v' such that $g(v)=g(v')$; then, $f([\mu,v\rangle)=f([\mu,v'\rangle)$, as

[17] To be precise, this is untrue as certain SCONE transitions are never executable, as shown in Example 6.8. As f maps transitions to μ-transactions, the part of SCONE we are interested in is the part covered by μ-transactions.

for $f([\mu,\alpha|\beta\rangle)=[\mu,\alpha\oplus\beta\rangle=[\mu,\beta\oplus\alpha\rangle=f([\mu,\beta|\alpha\rangle)$. Associativity of $\otimes$ induces further identifications, e.g., $f((t\rfloor v)\rfloor w)=f(t\rfloor(v|w))$. Moreover, because of the synchronization axiom, we have that, e.g., $f((v\lfloor t)|t')=f(v\lfloor(t|t'))$. As a consequence, it may seem that we could axiomatize these identifications within the algebra of RCCS transitions: a conditional axiom for action prefixing and nine axioms relating the three RCCS operators for parallel composition in all the possible ways for associativity. However, some form of commutativity is also possible. For example, $f([\alpha,\mathrm{nil}\rangle|[\alpha^-,\mathrm{nil}\rangle)=f([\alpha^-,\mathrm{nil}\rangle|[\alpha,\mathrm{nil}\rangle)$ due to the commutativity of the SCONE synchronization. Nonetheless, in general $f(t_1|t_2)\neq f(t_2|t_1)$, e.g. $[\alpha,\mathrm{nil}\rangle|(\alpha\lfloor[\alpha^-,\mathrm{nil}\rangle)$ and $(\alpha\lfloor[\alpha,\mathrm{nil}\rangle)|[\alpha,\mathrm{nil}\rangle$ do not give rise to the same transaction, because the synchronized α is different in the two transitions. Even if commutativity of SCONE synchronization is included, other identifications based on some form of commutativity, not easily expressible within the algebra of RCCS, are possible; e.g. $f(([\alpha,\mathrm{nil}\rangle|[\alpha^-,\mathrm{nil}\rangle)\rfloor\alpha)=f([\alpha,\mathrm{nil}\rangle|(\alpha\lfloor[\alpha^-,\mathrm{nil}\rangle))$. This example shows the intuitive fact that, when the first α is to be synchronized with α^-, the relative position of the second α w.r.t. α^- is irrelevant. As a matter of fact, an axiomatization for the identifications on transitions is already available! It is enough to interpret the algebra of RCCS inside the algebra of $\mathscr{G}_{N_{\mathrm{SCONE}}}$ as specified by the implementation morphism $\langle f,g\rangle$. In this way, we exploit the finer grain of the operations in the algebra of $\mathscr{G}_{N_{\mathrm{SCONE}}}$. Indeed, two CCS transitions t and t' are identified if and only if they give rise to the same CCS transaction, i.e. if we can prove $f(t)=f(t')$ in the equational theory consisting of the axioms in $\mathscr{P}[N_{\mathrm{SCONE}}]$ together with those arising from the definition of $\langle f,g\rangle$, which define the CCS operations as derived operations in the algebra of $\mathscr{P}[N_{\mathrm{SCONE}}]$.

Once verified by Propositions 6.10, 7.1 and 7.2 that interleaving bisimulation semantics is respected, we would like to check whether the semantics induced by the implementation morphism is sound w.r.t. the intuitive notion of causality. To this aim, we can homomorphically extend the implementation morphism $\langle f,g\rangle$ also to RCCS computations, and then observe what kind of identifications are made on them. We will show that whenever two RCCS computations are different only for the ordering of causally independent transitions, they are identified and, vice versa (even if only under some mild assumption) whenever two RCCS computations are identified, they differ only for the ordering of causally independent transitions.

Definition 7.3 (*Category of RCCS computations*). Let $\mathbf{Cat}(N_{\mathrm{RCCS}})$ denote the category obtained by adding an identity arc to each node of N_{RCCS} and closing freely w.r.t. the (partial) operation $-;-$ of sequential composition of its transitions, adding the usual categorial axioms (where, of course, the equality holds whenever both members are defined)

$$t;(t';t'')=(t;t');t'' \qquad \mathrm{id}(u);t=t=t;\mathrm{id}(v)$$

Note that the algebraic structure of RCCS has not been extended to computations. The arrows of category $\mathbf{Cat}(N_{\mathrm{RCCS}})$ are only computations composed of N_{RCCS} transitions.

The mapping $\langle f,g\rangle : N_{\mathrm{RCCS}} \to \mathscr{G}_{N_{\mathrm{SCONE}}}$ can be extended homomorphically to become a mapping from $\mathbf{Cat}(N_{\mathrm{RCCS}})$ to $\mathscr{G}_{N_{\mathrm{SCONE}}}$ by further adding the equation $f(t_1;t_2)=f(t_1);f(t_2)$. In this way we obtain a quotient of RCCS (states and) computations.

Definition 7.4. Category $\mathbf{Con}_{\mathrm{RCCS}}$ is obtained from $\mathbf{Cat}(N_{\mathrm{RCCS}})$ by the quotient map induced by $\langle f,g\rangle$.

Example 7.5. Let us consider the RCCS term $(\alpha+\beta)|(\alpha^-+\delta)$ and the net in Fig. 4. The RCCS computation

$$(([\alpha,\mathrm{nil}\rangle < +\beta)\rfloor(\alpha^-+\delta));(\mathrm{nil}\lfloor([\alpha^-,\mathrm{nil}\rangle < +\delta)) : (\alpha+\beta)|(\alpha^-+\delta) \xrightarrow{\alpha.\alpha^-} \mathrm{nil}|\mathrm{nil}$$

denotes the execution of an α followed by an α^-. It is mapped to the SCONE computation

$$((\alpha \ll +\beta);[\alpha,\mathrm{nil}\rangle) \otimes (\alpha^-+\delta);((\alpha^- \ll +\delta);[\alpha^-,\mathrm{nil}\rangle),$$

which, by functoriality and cancelling identities, is equivalent to the parallel execution

$$((\alpha \ll +\beta);[\alpha,\mathrm{nil}\rangle) \otimes ((\alpha^- \ll +\delta);[\alpha^-,\mathrm{nil}\rangle),$$

which, by introducing identities and applying functoriality, is equal to their execution in reverse order

$$(\alpha+\beta) \otimes ((\alpha^- \ll +\delta);[\alpha^-,\mathrm{nil}\rangle);((\alpha \ll +\beta);[\alpha,\mathrm{nil}\rangle).$$

Notice that this net computation is the image of the RCCS computation

$$((\alpha+\beta)\lfloor([\alpha^-,\mathrm{nil}\rangle < +\delta));(([\alpha,\mathrm{nil}\rangle < +\beta)\rfloor\mathrm{nil}) : (\alpha+\beta)|(\alpha^-+\delta) \xrightarrow{\alpha.\alpha^-} \mathrm{nil}|\mathrm{nil},$$

thus inducing an identification between the former and the latter RCCS computations.

A relation χ, called *concurrency relation*, between computations of length two (denoted as t_1 **then** t_2) relates computations differing just for permuting the order of independent transitions. This relation proposed in [17, 18], rephrases in algebraic terms a previous proposal by Boudol and Castellani [5, 6]. Nonetheless, the present formulation is, in our view, a bit simpler and more direct. First of all, [6] uses SOS transitions labelled by their proofs, whilst here we directly introduce a more manageable algebra of transitions (proofs themselves are transitions); moreover, whilst they define the permutation equivalence in two steps (first the definition of a concurrency relation on transitions outgoing from the same state, and then of the real equivalence on diamond computations), here the equivalence is obtained in one single step by (conditional) axioms since sequential composition is an operation of the algebra.

Definition 7.6 (*Concurrency relation*). Let $\_\mathbf{then}\_\chi\_\mathbf{then}\_$ be a quaternary relation on transitions of N_{RCCS} defined as the least $1,2 \leftrightarrow 3.4$ commutative[18] relation satisfying the following axiom and inference rules, where $\partial_0(t_i)=u_i$, $\partial_1(t_i)=v_i$, $i=1,\ldots,4$, and $\partial_0(t)=u$, $\partial_1(t)=v$.

$$t_1 \rfloor u_2 \mathbf{\ then\ } v_1 \lfloor t_2 \chi u_1 \lfloor t_2 \mathbf{\ then\ } t_1 \rfloor v_2$$

$$\frac{t_1 \mathbf{\ then\ } t_2 \chi t_3 \mathbf{\ then\ } t_4}{t_1 < w \mathbf{\ then\ } t_2 \chi t_3 < + w \mathbf{\ then\ } t_4} \qquad \frac{t_1 \mathbf{\ then\ } t_2 \chi t_3 \mathbf{\ then\ } t_4}{w + > t_1 \mathbf{\ then\ } t_2 \chi w + > t_3 \mathbf{\ then\ } t_4}$$

$$\frac{t_1 \mathbf{\ then\ } t_2 \chi t_3 \mathbf{\ then\ } t_4}{t_1 \rfloor w \mathbf{\ then\ } t_2 \rfloor w \chi t_3 \rfloor w \mathbf{\ then\ } t_4 \rfloor w} \qquad \frac{t_1 \mathbf{\ then\ } t_2 \chi t_3 \mathbf{\ then\ } t_4}{w \lfloor t_1 \mathbf{\ then\ } w \lfloor t_2 \chi w \lfloor t_3 \mathbf{\ then\ } w \lfloor t_4}$$

$$\frac{t_1 \mathbf{\ then\ } t_2 \chi t_3 \mathbf{\ then\ } t_4}{t_1 | t \mathbf{\ then\ } t_2 \rfloor v \chi t_3 \rfloor u \mathbf{\ then\ } t_4 | t} \qquad \frac{t_1 \mathbf{\ then\ } t_2 \chi t_3 \mathbf{\ then\ } t_4}{t | t_1 \mathbf{\ then\ } v \lfloor t_2 \chi u \lfloor t_3 \mathbf{\ then\ } t | t_4}$$

$$\frac{t_1 \mathbf{\ then\ } t_2 \chi t_3 \mathbf{\ then\ } t_4 \quad \text{and} \quad t'_1 \mathbf{\ then\ } t'_2 \chi t'_3 \mathbf{\ then\ } t'_4}{t_1 | t'_1 \mathbf{\ then\ } t_2 | t'_2 \chi t_3 | t'_3 \mathbf{\ then\ } t_4 | t'_4}$$

Proposition 7.7. *Given four transitions* t_1, t_2, t_3 *and* t_4 *in* N_{RCCS} *such that* $t_1 \mathbf{\ then\ } t_2 \chi t_3 \mathbf{\ then\ } t_4$, *the following hold*:

(i) $t_1 ; t_2$ *and* $t_3 ; t_4$ *are defined*;

(ii) $\partial_0(t_1)=\partial_0(t_3)$ *and* $\partial_1(t_2)=\partial_1(t_4)$;

(iii) t_1 *and* t_4 (t_2 *and* t_3) *have the same label.*

Proof. Immediate by induction on the proof of $t_1 \mathbf{\ then\ } t_2 \chi t_3 \mathbf{\ then\ } t_4$. □

The concurrency relation singles out a "diamond" in the transition system N_{RCCS} which is due to the different order of execution of independent transitions. The axiom algebraically singles out the basic diamonds, and the other rules reproduce the diamonds in all the other possible contexts.

Theorem 7.8 (Completeness w.r.t. the truly concurrent semantics for RCCS). *Given four basic transitions of* $\mathbf{Cat}(N_{\mathrm{RCCS}})$, *i.e.* t_1, t_2, t_3 *and* t_4 *in* N_{RCCS}, *then we have*

$$t_1 \mathbf{\ then\ } t_2 \chi t_3 \mathbf{\ then\ } t_4 \text{ implies } f(t_1 ; t_2) = f(t_3 ; t_4)$$

The proof, long and boring, proceed by induction on the proof of $t_1 \mathbf{\ then\ } t_2 \chi t_3 \mathbf{\ then\ } t_4$. It is reported in [23, p. 222–224]. The proof is essentially the same we will give in the next part (Theorem 10.3), where, even if full CCS is considered, the proof is less heavy because we can exploit the simpler algebraic characterization of concatenable processes in the case of 1-safe nets.

[18] Namely, $t_1 \mathbf{\ then\ } t_2 \chi t_3 \mathbf{\ then\ } t_4$ iff $t_3 \mathbf{\ then\ } t_4 \chi t_1 \mathbf{\ then\ } t_2$.

The reverse of the above theorem is false, in general, as the following example shows. Let us consider the four N_{RCCS} transitions below:

$$t_1=([\alpha,\mathrm{nil}\rangle)|[\alpha^-,\mathrm{nil}\rangle)\rfloor\alpha \qquad t_2=(\mathrm{nil}|\mathrm{nil})\lfloor[\alpha,\mathrm{nil}\rangle$$

$$t_3=([\alpha,nil\rangle\rfloor\alpha)|[\alpha^-,\mathrm{nil}\rangle \qquad t_4=(\mathrm{nil}\lfloor[\alpha,\mathrm{nil}\rangle)\rfloor\mathrm{nil}$$

It is immediate to verify that $f(t_1;t_2)=f(t_3;t_4)$. Nonetheless, t_1 **then** $t_2\,\chi\,t_3$ **then** t_4 is false. Indeed, Proposition 7.7 ensures that $\partial_0(t_1)=\partial_0(t_3)$ and $\partial_1(t_2)=\partial_1(t_4)$ whenever the χ relation holds, which is trivially false in this case. However, the reverse of Theorem 7.8 holds if the constraint $\partial_0(t_1)=\partial_0(t_3)$ is imposed. Also this proof is not reported, because it is essentially the same we provide for the next study (Theorem 10.4), where this constraint is satisfied as g is injective.

Theorem 7.9 (Consistency w.r.t the truly concurrent semantics for RCCS). *Let t_1,t_2,t_3 and t_4 be four different transitions in N_{RCCS} such that $\partial_0(t_1)=\partial_0(t_3)$. Then*

$$f(t_1;t_2)=f(t_3;t_4) \quad \textit{implies} \quad t_1\ \textbf{then}\ t_2\,\chi\,t_3\ \textbf{then}\ t_4$$

Finally, let us examine what happens in case commutative processes are used in place of the more concrete concatenable processes. First of all, we observed in footnote 7 that causality is not well-represented and so we cannot specify what a transaction (and its commit) is. Consequently, we should restrict to the RCCS sublanguage without synchronization (called RCCS$^-$), as its corresponding operation $[\,]$ cannot be defined in $\mathfrak{T}[N_{\mathrm{SCONE}}]$. Hence, $\langle h,g\rangle: N_{\mathrm{RCCS}^-}\to\mathfrak{T}[N_{\mathrm{SCONE}}]$ is defined as $\langle f,g\rangle$ of Definition 6.7, where the case of synchronization is omitted.

When using h, Theorem 7.8 obviously holds, because $f(t_1;t_2)=f(t_3;t_4)$ implies $h(t_1;t_2)=h(t_3;t_4)$. Conversely, Theorem 7.9 does no longer hold.

Example 7.10. Consider the agent $\alpha.\gamma|\beta.\gamma$ and the net in Fig. 2 (interpreting $a=\alpha.\gamma$, $b=\beta.\gamma$, $c=\gamma$, $l(t_1)=\alpha$, $l(t_2)=\beta$ and $l(t_3)=\gamma$, where $l(t)$ is the labelling of t, and ignoring place $d=\mathrm{nil}$). Assume that transition $[\alpha,\gamma>\,\rfloor\beta.\gamma$ is executed. Then, the reached state is $\gamma|\beta.\gamma$. The four RCCS transitions are:

$$t_1=[\gamma,nil\rangle\rfloor\beta.\gamma \qquad t_2=\mathrm{nil}\lfloor[\beta,\gamma\rangle$$

$$t_3=\gamma\lfloor[\beta,\gamma\rangle \qquad t_4=\gamma\lfloor[\gamma,\mathrm{nil}\rangle$$

It is clear that $[\alpha,\gamma\rangle\otimes\beta.\gamma;h(t_1;t_2)$ corresponds to the concatenable process in Fig. 2(b), whilst $[\alpha,\gamma\rangle\otimes\beta.\gamma;h(t_3;t_4)$ to the concatenable process in Fig. 2(c). Nonetheless, they are the same commutative process. Hence, $h(t_1;t_2)=h(t_3;t_4)$; nonetheless, t_1 **then** $t_2\,\chi\,t_3$ **then** t_4 cannot be proved (observe, e.g., that $\partial_1(t_2)\neq\partial_1(t_4)$).

This example shows that commutative processes are not appropriate when one wants to use SCONE as a system model for executions of RCCS$^-$ agents: the intuitive notion of causality is not respected and computations which should be intuitively

distinguished are identified instead. However, if one is interested only in detecting diamonds due to the different order of execution of independent transitions in the RCCS$^-$ transition system, then also the quotient induced by $\langle h, g\rangle$ is consistent and complete. Indeed, the following proposition can be proved similarly to Theorem 10.4; notice that the conditions $\partial_0(t_1)=\partial_0(t_3)$ and $\partial_1(t_2)=\partial_1(t_4)$ are crucial for the *only if* part, as Example 7.10 shows.

Proposition 7.11. *Let t_1, t_2, t_3 and t_4 be four different transitions in N_{RCCS} such that $\partial_0(t_1)=\partial_0(t_3)$ and $\partial_1(t_2)=\partial_1(t_4)$:*

$$h(t_1;t_2)=h(t_3;t_4) \text{ if and only if } t_1 \textbf{ then } t_2 \,\chi\, t_3 \textbf{ then } t_4$$

8. SCONE$^+$: a calculus of nets

With this section we begin the presentation of our second case study. There are some major differences when considering full CCS. First of all, we have to cope with the complexity of other operations. Restriction and relabelling may be modelled by means of a syntactic construction which leads, as a side effect, to a 1-safe P/T net representation of the reachable subnet implementing a CCS agent. However, the resulting net associated to a CCS agent may be infinite.

On the other hand, we can exploit the simpler algebra of concatenable processes in the case of 1-safe nets. Indeed, the rather complex treatment of token exchanges can be removed in favour of a more accessible semantics which assumes commutative the operation of parallel composition of processes.

Now we introduce our enriched simple calculus of nets, we call SCONE$^+$. Since the definition of sensible SCONE$^+$ operations depends on the possibility of mapping correctly CCS on the SCONE$^+$ net, we explain through an example why, in our opinion, some auxiliary operators (w.r.t those included in SCONE) are needed. The example concerns the troublesome interplay between parallelism and restriction. To be more concrete, let us consider the CCS agent $(\alpha|\alpha^-)\backslash\alpha$ which cannot perform asynchronously the two actions α and α^-, but only the τ-labelled communication step. The mapping for the restriction operator should be defined by function g from CCS states to SCONE$^+$ markings as follows:

$$g(v\backslash\alpha)=g(v)\backslash\alpha.$$

Thus, the SCONE$^+$ marking associated to the CCS agent $(\alpha|\alpha^-)\backslash\alpha$ might become

$$g((\alpha|\alpha^-)\backslash\alpha)=g(\alpha|\alpha^-)\backslash\alpha=(\alpha\oplus\alpha^-)\backslash\alpha.$$

On the other side, SCONE$^+$ must have a distributive axiom for restriction of the form

$$(v\oplus v')\backslash\alpha=v\backslash\alpha\oplus v'\backslash\alpha$$

as, otherwise, $(v \oplus v')\backslash\alpha$ would represent a single place, in constrast with the intuition that actions executed by v and v' are neither causally dependent, nor in conflict; indeed, whenever a parallel operator is present, we should get a multiset union of places from its components. However, the distributive axiom would induce false equalities, e.g. $(\alpha|\alpha^-)\backslash\alpha = \alpha\backslash\alpha|\alpha^-\backslash\alpha$ where the latter agent is deadlocked. Thus, our interpretation of | as multiset union is too simplistic now. Disjoint union is the answer to our problem. We introduce two unary operators for both nodes and transitions, $\mathbf{id}|\_$ and $\_|\mathbf{id}$ (called right and left context), with the intuition that $v|\mathbf{id}$ makes v the left part of a larger system. The mapping is

$$g(v|v') = g(v)|\mathbf{id} \oplus \mathbf{id}|g(v')$$

and now distributivity of restriction w.r.t. multiset union preserves the intended semantics. In our example, we get $g((\alpha|\alpha^-)\backslash\alpha) = (\alpha|\mathbf{id})\backslash\alpha \oplus (\mathbf{id}|\alpha^-)\backslash\alpha$, which represents two places, each one independently stuck but able to cooperate for synchronization. The idea of using these auxiliary context operators for correctly dealing with the interplay between restriction and parallel composition dates back to [11] and has been used by other authors [36, 41].

Definition 8.1 (*The algebra of SCONE$^+$ markings*). The SCONE$^+$ terms are generated by the following syntax:

$$M ::= 0 \mid \text{nil} \mid x \mid \mu.M \mid M+M \mid M\backslash\alpha \mid M[\Phi] \mid M|\mathbf{id} \mid \mathbf{id}|M \mid M \oplus M \mid \text{rec}\, x.M$$

The algebra is quotiented by the following axioms:

$$M \oplus M' = M' \oplus M \qquad M \oplus (M' \oplus M'') = (M \oplus M') \oplus M'' \qquad M \oplus 0 = M$$

$$(M \oplus M')\backslash\alpha = M\backslash\alpha \oplus M'\backslash\alpha \qquad 0\backslash\alpha = 0$$

$$(M \oplus M')[\Phi] = M[\Phi] \oplus M'[\Phi] \qquad 0[\Phi] = 0$$

$$(M \oplus M')|\mathbf{id} = M|\mathbf{id} \oplus M'|\mathbf{id} \qquad 0|\mathbf{id} = 0$$

$$\mathbf{id}|(M \oplus M') = \mathbf{id}||M \oplus \mathbf{id}|M' \qquad \mathbf{id}|0 = 0$$

$$rec\, x.M = M[\text{rec}\, x.M/x]$$

Only a subset of the terms, however, is relevant for the operational semantics: the closed, guarded terms (ranged over by u, v, w, with abuse of notation) which are the markings of the net.

The set of nodes $V_{\text{SCONE}+}$ is composed of all the closed, guarded terms, freely generated by the syntax modulo the axioms. Let $S_{\text{SCONE}+}$ be the set of the terms in $V_{\text{SCONE}+}$ generated by the following syntax:

$$N ::= \text{nil} \mid \mu.M \mid M+M \mid N\backslash\alpha \mid N[\Phi] \mid N|\mathbf{id} \mid \mathbf{id}|N$$

Thus, $V_{\text{SCONE}+}=(S_{\text{SCONE}+})^{\oplus}$, i.e. $V_{\text{SCONE}+}$ is the free commutative monoid of nodes over a set of places $S_{\text{SCONE}+}$, having 0 as neutral element.[19] $V_{\text{SCONE}+}$ has, on nodes, the algebraic structure of a net.

Intuitively, $\mu.v$ and $v+v'$ are places with the same intuition as before; $v\backslash\alpha$ may perform any transition of v, provided that it is not labelled α or α^-; $v[\Phi]$ performs the transitions of v, where the label has been relabelled by Φ; $v|\mathbf{id}$ can execute the same transitions as v, and makes explicit that v is part of a larger system connected on its own right, and symmetrically for $\mathbf{id}|v$ (see the synchronization operator on transition); $v \oplus v'$ is the multiset union of the two markings v and v'.

Definition 8.2 (*Algebra of SCONE$^+$ transitions*). Let $\Gamma=\mathcal{M}\cup\{\varepsilon\}$, ranged over by γ, where ε is a special unobservable and unrestricted action preserved by any relabelling function Φ. The transitions in $T_{\text{SCONE}+}$ are generated by the following constants (determined by *act*, *sum-*$<$, *sum-*$>$) and operations (determined by *res*, *rel*, */id*, *id/*, *sync*), where $t: v_1 \xrightarrow{\gamma} v_2$ and $t': v_1' \xrightarrow{\gamma'} v_2'$ range over transitions and v over $V_{\text{SCONE}+}$:

(*act*)	$[\mu.v\rangle : \mu.v \xrightarrow{\mu} v$	for any $\mu\in\mathcal{M}$
(*sum-*$<$)	$v \lll + v' : v + v' \xrightarrow{\varepsilon} v$	
(*sum-*$>$)	$v' + \ggg v : v' + v \xrightarrow{\varepsilon} v$	
(*res*)	$t\backslash\alpha : v_1\backslash\alpha \xrightarrow{\gamma''} v_2\backslash\alpha$	with $\gamma'' :=$ if $\gamma\notin\{\alpha,\alpha^-\}$ then γ else $*$
(*rel*)	$t[\Phi] : v_1[\Phi] \xrightarrow{\Phi(\gamma)} v_2[\Phi]$	
(*/id*)	$t\|\mathbf{id} : v_1\|\mathbf{id} \xrightarrow{\gamma} v_2\|\mathbf{id}$	
(*id/*)	$\mathbf{id}\|t : \mathbf{id}\|v_1 \xrightarrow{\gamma} \mathbf{id}\|v_2$	
(*sync*)	$t\|t' : v_1\|\mathbf{id} \oplus \mathbf{id}\|v_1' \xrightarrow{\gamma''} v_2\|\mathbf{id} \oplus \mathbf{id}\|v_2'$	with $\gamma'' =$ if $\gamma'=\gamma^-$ then τ else $*$

where $*$ is the error symbol. Again, we will consider only transitions which are not $*$-labelled.

The generators of the algebra are of two kinds: action prefixing ($[\mu, v\rangle : \mu.v \xrightarrow{\mu} v$) and local choice transitions ($v \lll + v' : v + v' \xrightarrow{\varepsilon} v$ and $v' + \ggg v : v' + v \xrightarrow{\varepsilon} v$). The operations for restriction, relabelling and left-, right-context are trivial. The intuition behind $t_1|t_2$ is that it is a new transition, whose source and target are the *disjoint* (because of tags "**id**|" and "|**id**") *set union* (because of $\oplus$) of the two and whose label is the synchronization of the two.

SCONE$^+$ is a P/T net $N_{\text{SCONE}+}=(V_{\text{SCONE}+}, T_{\text{SCONE}+}, \partial_0, \partial_1)$ because $V_{\text{SCONE}+}$ is the free commutative monoid over the set of places $S_{\text{SCONE}+}$.

Example 8.3. The reachable subnet for $(\alpha+\beta)|\mathbf{id} \oplus \mathbf{id}|(\alpha^-+\delta)$ is depicted in Fig. 9(a). Note the similarities with the net for the SCONE marking $(\alpha+\beta)\oplus(\alpha^-+\delta)$ reported

[19] Notice that nil is not the neutral element in this case. To this aim, we have added a new element 0, which will be the image of no CCS state.

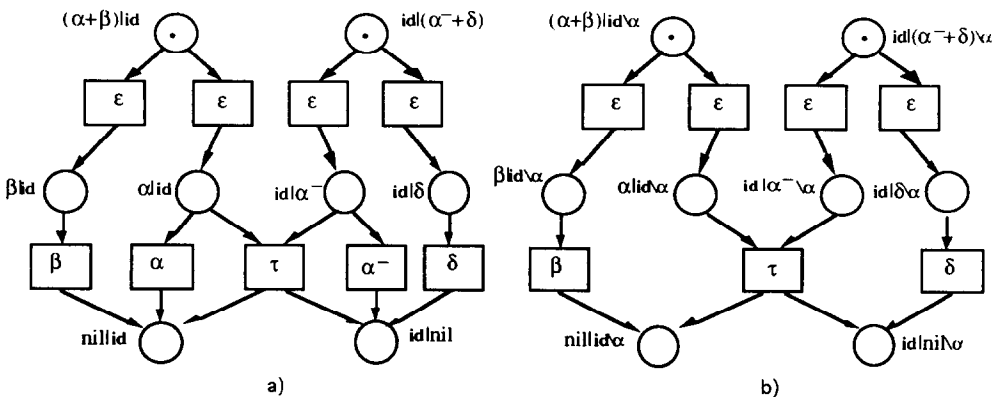

Fig. 9. The SCONE$^+$ subnet for the marking $(\alpha+\beta)|\mathbf{id} \oplus \mathbf{id}|(\alpha^- + \delta)$ in (a) and $((\alpha+\beta)|\mathbf{id} \oplus \mathbf{id}|(\alpha^- + \delta))\backslash\alpha$ in (b).

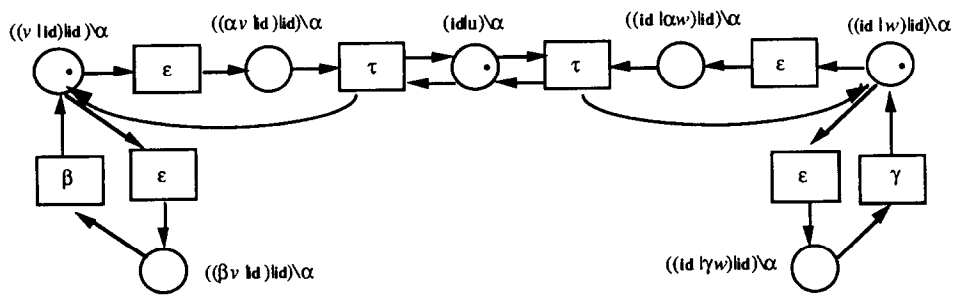

Fig. 10. A finite SCONE$^+$ subnet for a recursive behaviour.

in Fig. 4. Fig. 9(b) reports the net for the SCONE$^+$ marking $((\alpha+\beta)|\mathbf{id} \oplus \mathbf{id}|(\alpha^- + \delta))\backslash\alpha$.

Example 8.4. Fig. 10 represents the finite subnet reachable from the marking corresponding to the CCS term

$$E = (((\mathrm{rec}\, x.\alpha x + \beta x)|\mathrm{rec}\, x.\alpha x + \gamma x)|\mathrm{rec}\, x.\alpha^- x)\backslash\alpha$$

The initial marking is composed of three places:

$(\mathbf{(}v|\mathbf{id)}|\mathbf{id}\backslash\alpha$ where $v = \mathrm{rec}\, x.\alpha x + \beta x$

$(\mathbf{(id}|w)|\mathbf{id}\backslash\alpha$ where $w = \mathrm{rec}\, x.\alpha x + \gamma x$

$\mathbf{(id}|u)\backslash\alpha$ where $u = \mathrm{rec}\, x.\alpha^- x$

Example 8.5. The initial part of the infinite reachable subnet for the recursive term $v = \mathrm{rec}\, x.\gamma + (\alpha|\mathbf{id} \oplus \mathbf{id}|\beta.x)$, which will correspond to the CCS agent $\mathrm{rec}\, x.\gamma + (\alpha|\beta.x)$ (see also Example 5.8), is drawn in Fig. 11.

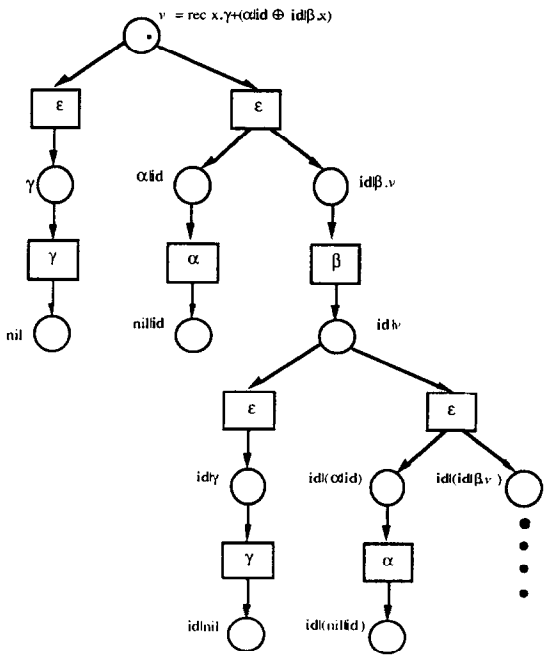

Fig. 11. The initial part of the infinite SCONE$^+$ subnet for $v=\mathrm{rec}\,x.\gamma+(\alpha|\mathbf{id}\oplus\mathbf{id}|\beta.x)$.

9. Distributed implementation of CCS

In analogy to what was done in Section 6, here we map CCS states and transitions into SCONE$^+$ markings and computations, thus providing CCS with a distributed implementation.

9.1. Implementing states

Definition 9.1 (*From CCS states to SCONE$^+$ markings*). Let $g: V_{\mathrm{CCS}}\to V_{\mathrm{SCONE}+}$ be defined as follows:

$$g(\mathrm{nil})=\mathrm{nil} \qquad g(x)=x$$

$$g(\mu.v)=\mu.g(v) \qquad g(\mathrm{rec}\,x.v)=\mathrm{rec}\,x.g(v)$$

$$g(v+v')=g(v)+g(v') \qquad g(v\backslash\alpha)=g(v)\backslash\alpha$$

$$g(v[\Phi])=g(v)[\Phi] \qquad g(v|v')=g(v)|\mathbf{id}\oplus\mathbf{id}|g(v')$$

Mapping g is injective and invertible; any place in $g(v)$ occurs at most once, and from the set of places composing $g(v)$ we can recover the original CCS agents v by

means of a sort of unification procedure, where various occurrences of **id** should be intended as different variables. Indeed, g is a decomposition function which keeps track of the topological structure of subsystem interconnections *via* the auxiliary operators $\_|\mathbf{id}$ and $\mathbf{id}|\_$. A marking which is image of a CCS state is called *complete* [13]. Complete markings enjoy a nice property: a marking reachable from a complete marking is, in turn, complete. Therefore, since complete markings are 1-safe, the reachable subnet is 1-safe.

Definition 9.2. A marking w is *complete* if there exists a CCS term v such that $g(v)=w$.

Property 9.3 (i) If a marking v is complete, then it is composed of a *set* of places (i.e. it is 1-safe).

(ii) Function g defines a bijection between CCS terms and complete markings of SCONE$^+$.

Theorem 9.4. *Given a complete marking v and a computation $\xi: v \to u$ in $\mathscr{T}[N_{\mathrm{SCONE}^+}]$, u is a complete marking.*

Proof. By induction on the structure of the complete marking v. If $v=\mathrm{nil}$, then ξ can only be $id(\mathrm{nil})$ and the thesis holds trivially. If $v=\mu.v'$, then v' is also a complete marking by definition of g; thus, the thesis holds for v' by inductive hypothesis; hence, also for v because the unique initial transition $[\mu.v'\rangle$ reaches v'. If $v=v'+v''$, then v' and v'' are also complete; thus, the thesis holds for both v' and v'' by inductive hypthesis; hence, the thesis holds for v because the only two initial transitions from $v'+v''$ reach v' and v'', respectively. If $v=v'|\mathbf{id} \oplus \mathbf{id}|v''$, then v' and v'' are complete and the thesis holds by inductive hypothesis for v' and v''. Note that whenever u' and u'' are complete, then $u'|\mathbf{id} \oplus \mathbf{id}|u''$ is complete; therefore, any transition enabled by $v'|\mathbf{id}$ reaches a marking which is completed with $\mathbf{id}|v''$ (similarly for any transition enabled by $\mathbf{id}|v''$ and also for synchronizations, needing tokens from both sides). Sequential composition and (disjoint, i.e. $\_|\mathbf{id} \otimes \mathbf{id}|\_$) parallel composition of enabled computations satisfying the thesis produce enabled computations satisfying the thesis. Hence, the thesis holds for v. The simple cases for restriction and relabelling are omitted. No case for recursion is needed because of the recursion axiom.

Corollary 9.5. *The SCONE$^+$ subnet reachable from a complete marking is 1-safe.*

9.2. Implementing transitions

As the part of SCONE$^+$ relevant to our aims is a 1-safe net, the algebraic construction of Definition 3.14, yielding the strictly symmetric strict monoidal category $\mathscr{T}[N_{\mathrm{SCONE}^+}]$, can be exploited, as we know that for 1-safe nets this simple algebraic characterization is equivalent to the one for concatenable processes. Let us

try to define the mapping from CCS transitions to the arrows of $\mathscr{T}[N_{\text{SCONE}^+}]$. We start with some examples, pointing out some technical problems. Formally, the mapping is to be defined in a purely syntax-driven way. However, some CCS transitions have no obvious counterpart in SCONE$^+$. For example, consider $(\alpha|\beta)\backslash\alpha$; its sole transition is $(\alpha\lfloor[\beta,\text{nil}\rangle)\backslash\alpha$; let us try to map it to a SCONE$^+$ computation:

$$f(\alpha\lfloor[\beta,\text{nil}\rangle)=\alpha|\mathbf{id}\otimes\mathbf{id}|f([\beta,\text{nil}\rangle)=\alpha|\mathbf{id}\otimes\mathbf{id}|[\beta,\text{nil}\rangle$$

which is not a transition, rather a computation (parallel composition of a net transition with an identity), and thus restriction must be extended to computations:

$$f(\alpha\lfloor[\beta,\text{nil}\rangle\backslash\alpha)=(\alpha|\mathbf{id}\otimes\mathbf{id}|[\beta,\text{nil}\rangle)\backslash\alpha$$

Indeed, restriction is not defined in $\mathscr{T}[N_{\text{SCONE}^+}]$. Similar arguments also hold for relabelling and for the two unary context operators. Therefore, we should define an algebra $\mathscr{G}_{N_{\text{SCONE}^+}}$ obtained enriching the algebra of category $\mathscr{T}[N_{\text{SCONE}^+}]$ with the auxiliary $\mathbf{id}|\_$ and $\_|\mathbf{id}$, restriction and relabelling, and expressing which net computations the terms $\mathbf{id}|f(t), f(t)|\mathbf{id}, f(t)\backslash\alpha$ and $f(t)[\Phi]$ should represent. For these operators, the solution is immediate: it is enough to add a distributive axiom, e.g.

$$(t_1\otimes t_2)\backslash\alpha=t_1\backslash\alpha\otimes t_2\backslash\alpha,$$

stating that the restriction of a parallel execution of two transitions is the parallel execution of the restricted transitions. Back to our example,

$$f(\alpha\lfloor[\beta,\text{nil}\rangle\backslash\alpha)=(\alpha|\mathbf{id}\otimes\mathbf{id}|[\beta,\text{nil}\rangle)\backslash\alpha=(\alpha|\mathbf{id})\backslash\alpha\otimes(\mathbf{id}|[\beta,\text{nil}\rangle)\backslash\alpha$$

which is the parallel composition of a net transition and of an identity.

The next problem is concerned with nondeterminism, and its solution is in perfect analogy with the solution presented in Section 6. Any external choice is seen as composed of two steps: the choice of a subcomponent and the execution of an action from the selected component.

Further example presents the harder problem of synchronization. As expected, the mapping of a CCS transition $t_1|t_2$ should be the net computation $f(t_1|t_2)=f(t_1)|f(t_2)$, but unfortunately $f(t_1)|f(t_2)$ is not defined in $\mathscr{T}[N_{\text{SCONE}^+}]$. Therefore, the algebra $\mathscr{G}_{N_{\text{SCONE}^+}}$ should be further enriched with the synchronization operation. Also in this case, a deterministic synchronization operation is defined through the transaction concept. As in Section 6, a transaction is a concatenable process such that there is a net transition, called the commit, which is larger than all the others in the partial ordering.

Definition 9.6 (*Net transactions and CCS transactions*). A *net transaction* for a net N is an equivalence class of terms in $\mathscr{T}[N]$ such that there are no terms in the class of the form $\eta;(u\otimes t\otimes t')$, i.e. with two concurrent final transitions. A *CCS transaction*, also called μ-transaction, is a net transaction for the net of SCONE$^+$ where all the transitions are labelled by ε, with the exception of the μ-labelled commit transition.

The shorthand $\eta;(u \otimes t)$ for a transaction can be conveniently used, where η is a computation, t is the commit and u an identity (but also net computations which are not transactions can have the same form).[20] This is a standard form, as the decomposition $\eta;(u \otimes t)$ is unique, up to equivalence of η (provable similarly to what we did in Section 6). With this notion in mind, a natural deterministic definition of synchronization between two transactions consists of putting in parallel the two processes but synchronizing the two commit transitions to become the commit for the resulting process.

In the following definition we introduce a new algebra by enriching $\mathscr{T}[N_{\text{SCONE}^+}]$ with some derived operations (i.e. by extending these operations to net computations). Among them, the most important is the operator $[\,]$ of synchronization, expressing the intuitive fact that the synchronization of two transactions is again a transaction. In this way, a generalized notion of computation is defined.

Definition 9.7 (*from* $\mathscr{T}[N_{\text{SCONE}^+}]$ *to* $\mathscr{G}_{N_{\text{SCONE}^+}}$). $\mathscr{G}_{N_{\text{SCONE}^+}} = (V_{\text{SCONE}^+}, \mathbf{T}, \partial_0, \partial_1)$ is the same graph as $\mathscr{T}[N_{\text{SCONE}^+}]$ with the extra four operations $\_\backslash\alpha$, $\_[\Phi]$, $\_|\mathbf{id},\mathbf{id}|\_$ and the partial operation $\_[\,]\_$ on transactions. These operations are subject to the following axioms, which reduce them to the fundamental operations inside the algebra of $\mathscr{T}[N_{\text{SCONE}^+}]$:

$$(\xi \otimes \xi')\backslash\alpha = \xi\backslash\alpha \otimes \xi'\backslash\alpha \qquad (\xi;\xi')\backslash\alpha = \xi\backslash\alpha;\xi'\backslash\alpha$$

$$(\xi \otimes \xi')[\Phi] = \xi[\Phi] \otimes \xi'[\Phi] \qquad (\xi;\xi')[\Phi] = \xi[\Phi];\xi'[\Phi]$$

$$(\xi \otimes \xi')|\mathbf{id} = \xi|\mathbf{id} \otimes \xi'|\mathbf{id} \qquad (\xi;\xi')|\mathbf{id} = \xi|\mathbf{id};\xi'|\mathbf{id}$$

$$\mathbf{id}|(\xi \otimes \xi') = \mathbf{id}|\xi \otimes \mathbf{id}|\xi' \qquad \mathbf{id}|(\xi;\xi') = \mathbf{id}|\xi;\mathbf{id}|\xi'$$

$$\mathrm{id}(v)\backslash\alpha = \mathrm{id}(v\backslash\alpha) \qquad \mathrm{id}(v)[\Phi] = \mathrm{id}(v[\Phi])$$

$$\mathrm{id}(v)|\mathbf{id} = \mathrm{id}(v|\mathbf{id}) \qquad \mathbf{id}|\mathrm{id}(v) = \mathrm{id}(\mathbf{id}|v)$$

Let $\xi = \eta;(u \otimes t)$ and $\xi' = \eta';(u' \otimes t')$ be two transactions, then

$$\xi[\,]\xi' = (\eta|\mathbf{id} \otimes \mathbf{id}|\eta');(u|\mathbf{id} \otimes \mathbf{id}|u' \otimes t|t')$$

Proposition 9.8 (Synchronizations of transactions are transactions). (i) *Given two transactions* ξ *and* ξ', $\xi[\,]\xi'$ *is a transaction.*

(ii) *Given a* λ*-transaction* ξ *and a* λ^-*-transaction* ξ', $\xi[\,]\xi'$ *is a* τ*-transaction.*

Definition 9.9 (*From CCS transitions to SCONE*$^+$ *generalized computations*). Let $f: T_{\text{CCS}} \to \mathbf{T}$ be defined as follows, where g is the mapping of Definition 9.1. and $t: u \to w$:

$$f([\mu,v\rangle) = [\mu,g(v)\rangle$$

$$f(t < + v) = g(u) \ll + g(v);f(t) \qquad f(v + > t) = g(v) + \gg g(u);f(t)$$

[20] Indeed, it is easy to see that any computation ξ can be always reduced to the format $(u_1 \otimes t_1);\cdots;(u_n \otimes t_n)$ by applying functoriality of the commutative parallel operation $\otimes$.

$$f(v\lfloor t)=g(v)|\mathbf{id}\otimes\mathbf{id}|f(t) \qquad f(t\rfloor v)=f(t)|\mathbf{id}\otimes\mathbf{id}|g(v)$$

$$f(t\backslash\alpha)=f(t)\backslash\alpha \qquad f(t[\Phi])=f(t)[\Phi]$$

$$f(t_1|t_2)=f(t_1)[\,]f(t_2)=(\eta_1|\mathbf{id}\otimes\mathbf{id}|\eta_2);(u_1|\mathbf{id}\otimes\mathbf{id}|u_2\otimes t|t')$$

where $f(t_1)=\eta_1;(u_1\otimes t)$ and $f(t_2)=\eta_2;(u_2\otimes t')$

It is an easy task to prove, by structural induction, that f is injective; this is essentially due to the fact that g is injective (hence, no two CCS prefix transitions can be confused). Nonetheless, f is not surjective; the instance reported in Example 6.8 applies also to CCS (adding the place for nil).

Proposition 9.10. *For each CCS transition t, $f(t)$ is always a CCS transaction.*

Of course, also the reverse is true, i.e. any CCS transaction is the image of a CCS transition. The following important proposition states that $\langle f,g\rangle$ is a graph morphism.

Proposition 9.11. *The pair $\langle f,g\rangle: N_{\mathrm{CCS}}\to\mathscr{G}_{N_{\mathrm{SCONE}^+}}$ is a graph morphism, i.e. $g\circ\partial_{i_{\mathrm{CCS}}}=\partial_{i_{\mathrm{SCONE}^+}}\circ f$, $i=0,1$.*

Proof. By structural induction (only two cases are reported). For the sake of simplicity, $\partial_{i_{\mathrm{SCONE}^+}}$ is shortened to $\partial_{i_{s^+}}$ and $\partial_{i_{\mathrm{CCS}}}$ to ∂_{i_c}.

$$\begin{aligned}
&\langle\partial_{0_{s^+}}(f(t\backslash\alpha)),\partial_{1_{s^+}}(f(t\backslash\alpha)\rangle\\
&\quad=\langle\partial_{0_{s^+}}(f(t)\backslash\alpha),\partial_{1_{s^+}}(f(t)\backslash\alpha)\rangle=\partial_{0_{s^+}}(f(t))\backslash\alpha,\partial_{1_{s^+}}(f(t))\backslash\alpha\rangle\\
&\quad=\langle g(\partial_{0_c}(t))\backslash\alpha,g(\partial_{1_c}(t))\backslash\alpha\rangle=\langle g(\partial_{0_c}(t)\backslash\alpha),g(\partial_{1_c}(t)\backslash\alpha)\rangle\\
&\quad=\langle g(\partial_{0_c}(t\backslash\alpha)),g(\partial_{1_c}(t\backslash\alpha))\rangle,\\
&\langle\partial_{0_{s^+}}f(t_1|t_2)),\partial_{1_{s^+}}(f(t_1|t_2))\rangle\\
&\quad=\langle\partial_{0_{s^+}}(f(t_1)[\,]f(t_2)),\partial 1_{0_{s^+}}(f(t_1)[\,]f(t_2))\rangle\\
&\quad=\langle\partial_{0_{s^+}}(f(t_1))|\mathbf{id}\oplus\mathbf{id}|\partial_{0_{s^+}}(f(t_2)),\partial_{1_{s^+}}(f(t_1))|\mathbf{id}\oplus\mathbf{id}|\partial_{1_{s^+}}(f(t_2))\rangle\\
&\quad=\langle g(\partial_{0_c}(t_1))|\mathbf{id}\oplus\mathbf{id}|g(\partial_{0_c}(t_2)),g(\partial_{1_c}(t_1))|\mathbf{id}\oplus\mathbf{id}|g(\partial_{1_c}(t_2))\rangle\\
&\quad=\langle g(\partial_{0_c}(t_1|t_2)),g(\partial_{1_c}(t_1|t_2))\rangle. \qquad\square
\end{aligned}$$

10. Distributed semantics of CCS

CCS semantics is investigated through the implementation mapping $\langle f,g\rangle$: $N_{\mathrm{CCS}}\to\mathscr{G}_{N_{\mathrm{SCONE}^+}}$, in order to study the effects on the source N_{CCS} of the identifications induced by $\langle f,g\rangle$. Since g and f are both injective, no different CCS agents are mapped

to the same marking nor CCS transitions are identified. Thus, in this case the quotient of N_{CCS} through $\langle f,g\rangle$ is N_{CCS} itself. Nonetheless, the mapping will equate (all and only) those computations obtained by permuting transitions generating independent events. To prove this fact, we can homomorphically extend the implementation morphism $\langle f,g\rangle$ also to CCS computations, and then observe what kind of identifications are made on them.

Similarly to what was done in Definition 7.3, let $\mathbf{Cat}(N_{CCS})$ denote the category of CCS computations, obtained by adding an identity arc to each node of N_{CCS} and closing freely w.r.t. the operation $\_;\_$ of composition (modulo the usual categorical axioms). Observe that the algebraic structure of CCS has not been extended to computations, i.e. the arrows of $\mathbf{Cat}(N_{CCS})$ are only computations composed of N_{CCS} transitions. The mapping $\langle f,g\rangle: N_{CCS} \to \mathscr{G}_{N_{SCONE^+}}$ can be extended homomorphically to become a mapping from $\mathbf{Cat}(N_{CCS})$ to $\mathscr{G}_{N_{SCONE^+}}$ by further adding the equation $f(t_1;t_2)=f(t_1);f(t_2)$, which is defined as $\langle f,g\rangle$ is a graph morphism. Category $\mathbf{Con}_{CCS}$ is the resulting quotient graph.

Example 10.1. An interesting test to measure the reliability of a true concurrent semantics is represented by the CCS agent $E=\gamma+(\alpha|\beta)$, where an interweaving of nondeterministic and parallel operators may cause the possible loss of causal independency between the concurrent actions α and β (see [13, 14] for more details). The place $\gamma+(\alpha|\mathbf{id}\oplus\mathbf{id}|\beta)$ is the image of E, and the two CCS transitions

$$\gamma+>([\alpha,\mathrm{nil}\rangle\rfloor\beta):\gamma+(\alpha|\beta)\xrightarrow{\alpha}\mathrm{nil}|\beta,$$

$$\gamma+>(\alpha\lfloor[\beta,\mathrm{nil}\rangle):\gamma+(\alpha|\beta)\xrightarrow{\beta}\alpha|\mathrm{nil}$$

are mapped to the following two net computations, respectively:

$$(\gamma+\gg(\alpha|\mathbf{id}\oplus\mathbf{id}|\beta));([\alpha,\mathrm{nil}\rangle|\mathbf{id}\otimes\mathbf{id}|\beta),$$

$$(\gamma+\gg(\alpha|\mathbf{id}\oplus\mathbf{id}|\beta));(\alpha|\mathbf{id}\otimes\mathbf{id}|[\beta,\mathrm{nil}\rangle).$$

The image of the CCS computation $(\gamma+>([\alpha,\mathrm{nil}\rangle\rfloor\beta));(\mathrm{nil}\lfloor[\beta,\mathrm{nil}\rangle)$ is the net computation

$$(\gamma+\gg(\alpha|\mathbf{id}\oplus\mathbf{id}|\beta));([\alpha,\mathrm{nil}\rangle|\mathbf{id}\otimes\mathbf{id}|\beta);(\mathrm{nil}|\mathbf{id}\otimes\mathbf{id}|[\beta,\mathrm{nil}\rangle),$$

where actions α and β are in fact causally independent. This net computation is equivalent to

$$(\gamma+\gg(\alpha|\mathbf{id}\oplus\mathbf{id}|\beta));(\alpha|\mathbf{id}\otimes\mathbf{id}|[\beta,\mathrm{nil}\rangle);([\alpha,\mathrm{nil}\rangle|\mathbf{id}\otimes\mathbf{id}|\mathrm{nil}),$$

which is the image of the computation $(\gamma+>(\alpha\lfloor[\beta,\mathrm{nil}\rangle));([\alpha,\mathrm{nil}\rangle\rfloor\mathrm{nil})$ of CCS. Therefore, the two CCS computations are identified by the mapping f, i.e. they denote the same arrow in $\mathbf{Con}_{CCS}$.

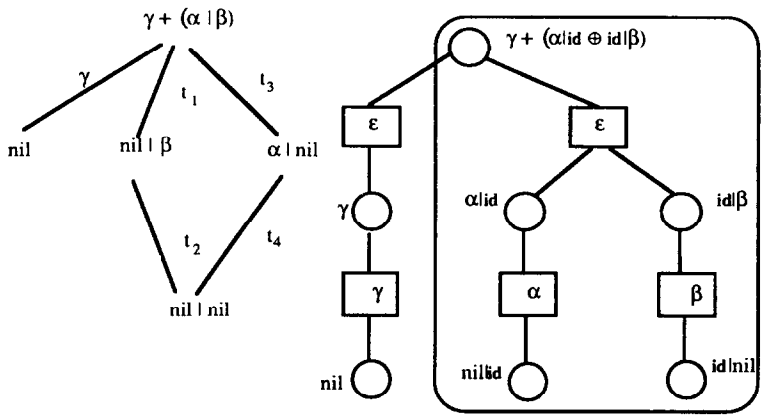

Fig. 12. The transition system and the net for the agent $\gamma+\alpha|\beta$. Notice that both $t_1;t_2$ and $t_3;t_4$ are mapped to the same process, enclosed in the box.

Now we want to prove that the identifications on CCS computations due to the implementation mapping are the same obtained *via* a set of axioms proposed in [18, 17]. The concurrency relation χ introduced in Definition 7.6. must be extended to the further operators of restriction and relabelling.

Definition 10.2 (*Concurrency relation*). Let $\_$ **then** $\_\,\chi\,\_$ **then** $\_$ be the quaternary relation on transition of N_{CCS} defined as the least $1,2 \leftrightarrow 3,4$ commutative relation satisfying the following inference rules, in addition to those in Definition 7.6:

$$\frac{t_1 \textbf{ then } t_2 \,\chi\, t_3 \textbf{ then } t_4}{t_1\backslash\alpha \textbf{ then } t_2\backslash\alpha \,\chi\, t_3\backslash\alpha \textbf{ then } t_4\backslash\alpha} \qquad \frac{t_1 \textbf{ then } t_2 \,\chi\, t_3 \textbf{ then } t_4}{t_1[\Phi] \textbf{ then } t_2[\Phi] \,\chi\, t_3[\Phi] \textbf{ then } t_4[\Phi]}$$

Similarly to Proposition 7.7, we can prove that, if we are given four transitions t_1, t_2, t_3 and t_4 in N_{CCS} such that t_1 **then** $t_2 \,\chi\, t_3$ **then** t_4, then $t_1;t_2$ and $t_3;t_4$ are defined, $\partial_0(t_1)=\partial_0(t_3)$ and $\partial_1(t_2)=\partial_1(t_4)$, as well as that t_1 and t_4 (t_2 and t_3) have the same label.

Theorem 10.3 (Completeness w.r.t. the truly concurrent semantics of CCS). *Given four basic transitions of* $\mathbf{Cat}(N_{CCS})$, *i.e.* t_1, t_2, t_3 *and* t_4 *in* N_{CCS}, *then we have*

$$t_1 \textbf{ then } t_2 \,\chi\, t_3 \textbf{ then } t_4 \textit{ implies } f(t_1;t_2)=f(t_3;t_4)$$

Proof. The proof is by induction on the proof of t_1 **then** t_2 χ t_3 **then** t_4. Actually, in order to be able to prove the theorem in a completely syntactical manner, we prove a stronger result:

$$t_1 \textbf{ then } t_2 \,\chi\, t_3 \textbf{ then } t_4 \text{ implies } f(t_1)=(\xi;(T' \oplus u'')) \otimes w, f(t_2)$$
$$=(v' \otimes T'') \otimes w, f(t_3)=(\xi;(u' \otimes T'')) \otimes w \text{ and } f(t_4)=(T' \otimes v'') \otimes w$$

where w and ξ are optional (but if, e.g., w is present in one of the $f(t_i)$, then it is present in all the others), and $T' : u' \to v'$, $T'' : u'' \to v''$ are transactions. Sequentially composing the two processes, we obtain $f(t_1 ; t_2) = (\xi ; (T' \otimes u'') ; (v' \otimes T'')) \otimes w = (\xi ; (T' \otimes T'')) \otimes w = (\xi ; (u' \otimes T'') ; (T' \otimes v'')) \otimes w = f(t_3 ; t_4)$. This means that, apart from a possible initial common segment (represented by the optional ξ) and some idle tokens (represented by the optional w), $t_1 ; t_2$ generates two independent (non-ε-labelled) events which are generated in reverse order by $t_3 ; t_4$. The order exchange is expressed by functoriality. The "commutativity" condition on χ holds because equality is a commutative relation.

The base case is the axiom; we have to prove that $f(t_1 \rfloor u_2 ; v_1 \lfloor t_2) = f(u_1 \lfloor t_2 ; t_1 \rfloor v_2)$. In this case, w and ξ are not present. Indeed, as $f(t_1 \rfloor u_2) = f(t_1) | \mathbf{id} \otimes \mathbf{id} | g(u_2)$, $T' = f(t_1) | \mathbf{id}$ and $u'' = \mathbf{id} | g(u_2)$; moreover, as $f(v_1 \lfloor t_2) = g(v_1) | \mathbf{id} \otimes \mathbf{id} | f(t_2)$, $v' = g(v_1) | \mathbf{id}$ and $T'' = \mathbf{id} | f(t_2)$. Finally, it is easy to observe that $f(u_1 \lfloor t_2) = u' \otimes T''$ and $f(t_1 \rfloor v_2) = T' \otimes v''$. The thesis can be proved directly as follows:

$$
\begin{aligned}
f(t_1 \rfloor u_2 ; v_1 \lfloor t_2) &= (f(t_1) | \mathbf{id} \otimes \mathbf{id} | g(u_2)) ; (g(v_1) | \mathbf{id} \otimes \mathbf{id} | f(t_2)) \\
&= (f(t_1) | \mathbf{id} ; g(v_1) | \mathbf{id}) \otimes (\mathbf{id} | g(u_2) ; \mathbf{id} | f(t_2)) \\
&= (f(t_1) ; g(v_1)) | \mathbf{id} \otimes \mathbf{id} | (g(u_2) ; (f(t_2)) \\
&= f(t_1) | \mathbf{id} \otimes \mathbf{id} | f(t_2) = (g(u_1) ; f(t_1)) | \mathbf{id} \otimes \mathbf{id} | (f(t_2) ; g(v_2)) \\
&= (g(u_1) | \mathbf{id} ; f(t_1) | \mathbf{id}) \otimes (\mathbf{id} | f(t_2) ; \mathbf{id} | g(v_2)) \\
&= (g(u_1) | \mathbf{id} \otimes \mathbf{id} | f(t_2)) ; (f(t_1) | \mathbf{id} \otimes \mathbf{id} | g(v_2)) = f(u_1 \lfloor t_2 ; t_1 \rfloor v_2).
\end{aligned}
$$

For restriction, we have to prove that $f(t_1 \backslash \alpha ; t_2 \backslash \alpha) = f(t_3 \backslash \alpha ; t_4 \backslash \alpha)$, knowing that $f(t_1 ; t_2) = f(t_3 ; t_4)$ by inductive hypothesis. Hence, the thesis holds because of the distributive axioms of restriction:

$$
\begin{aligned}
f(t_1 \backslash \alpha) &= f(t_1) \backslash \alpha = ((\xi ; (T' \otimes u'')) \otimes w) \backslash \alpha = (\xi \backslash \alpha ; (T' \backslash \alpha \otimes u'' \backslash \alpha)) \otimes w \backslash \alpha \\
f(t_2 \backslash \alpha) &= f(t_2) \backslash \alpha = ((v' \otimes T'') \otimes w) \backslash \alpha = (v' \backslash \alpha \otimes T'' \backslash \alpha) \otimes w \backslash \alpha \\
f(t_3 \backslash \alpha) &= f(t_3) \backslash \alpha = ((\xi ; (u' \otimes T'')) \otimes w) \backslash \alpha = (\xi \backslash \alpha ; (u' \backslash \alpha \otimes T'' \backslash \alpha)) \otimes w \backslash \alpha \\
f(t_4 \backslash \alpha) &= f(t_4) \backslash \alpha = ((T' \otimes v'') \otimes w) \backslash \alpha = (T' \backslash \alpha \otimes v'' \backslash \alpha) \otimes w \backslash \alpha
\end{aligned}
$$

which are all of the required form. The thesis can be also proved directly as follows:

$$
\begin{aligned}
f(t_1 \backslash \alpha ; t_2 \backslash \alpha) &= f(t_1 \backslash \alpha) ; f(t_2 \backslash \alpha) = f(t_1) \backslash \alpha ; f(t_2) \backslash \alpha = (f(t_1) ; f(t_2)) \backslash \alpha \\
&= f(t_1 ; t_2) \backslash \alpha = f(t_3 ; t_4) \backslash \alpha = (f(t_3) ; f(t_4)) \backslash \alpha = f(t_3) \backslash \alpha ; f(t_4) \backslash \alpha \\
&= f(t_3 \backslash \alpha) ; f(t_4 \backslash \alpha) = f(t_3 \backslash \alpha ; t_4 \backslash \alpha).
\end{aligned}
$$

The similar case of relabelling is omitted. In the case of the rule for nondeterminism, we have to prove that $f(w+>t_1;t_2)=f(w+>t_3;t_4)$, knowing that $f(t_1;t_2)=f(t_3;t_4)$ by inductive hypothesis:

$$f(w+>t_1)=((g(w)+\gg g(u_1));f(t_1))=(g(w)+\gg g(u_1));(\xi;(T'\otimes u''))\otimes w$$
$$=((g(w)+\gg g(u_1));\xi\otimes w);(T'\otimes u''\otimes w))$$
$$f(t_2)=v'\otimes T''\otimes w$$
$$f(w+>t_3)=((g(w)+\gg g(u_3));f(t_3))=(g(w)+\gg g(u_3));(\xi;(u'\otimes T'')\otimes w$$
$$=((g(w)+\gg g(u_3));\xi\otimes w);(u'\otimes T''\otimes w))$$
$$f(t_4)=T'\otimes v''\otimes w$$

which are all of the required form (the new ξ is $(g(w)+\gg g(u_1);\xi\otimes w)$, the new T' is $T'\otimes w$, the new T'' is T'' and the new w is 0). The thesis holds also because $u_1=\partial_0(t_1)=\partial_0(t_3)=u_3$. Directly,

$$f(w+>t_1;t_2)=((g(w)+\gg g(u_1));f(t_1));f(t_2)=(g(w)+\gg g(u_1));(f(t_1);f(t_2)$$
$$=(g(w)+\gg g(u_1));f(t_1;t_2)=(g(w)+\gg g(u_3));f(t_3;t_4)$$
$$=(g(w)+\gg g(u_1));(f(t_3);f(t_4))=((g(w)+\gg g(u_3);f(t_3));f(t_4)$$
$$=f(w+>t_3;t_4).$$

The proof for the symmetric nondete⇒rministic rule is omitted.

In the case of synchrony, we have to prove that $f(w\lfloor t_1;w\lfloor t_2)=f(w\lfloor t_3;w\lfloor t_4)$. In this case, we simply add extra idle tokens $g(w)|\mathbf{id}$. For the sake of brevity, we provide only the direct proof.

$$f(w\lfloor t_1;w\lfloor t_2)$$
$$=(g(w)|\mathbf{id}\otimes\mathbf{id}|f(t_1));(g(w)|\mathbf{id}\otimes\mathbf{id}|f(t_2))=(g(w);g(w))|\mathbf{id}\otimes\mathbf{id}|(f(t_1);f(t_2))$$
$$=g(w)|\mathbf{id}\otimes\mathbf{id}|f(t_1;t_2)=g(w)|\mathbf{id}\otimes\mathbf{id}|f(t_3;t_4))$$
$$=(g(w);g(w))|\mathbf{id}\otimes\mathbf{id}|(f(t_3);f(t_4))$$
$$\sim(g(w)|\mathbf{id};g(w)|\mathbf{id})\otimes(\mathbf{id}|f(t_3);\mathbf{id}|f(t_4))$$
$$=(g(w)|\mathbf{id}\otimes\mathbf{id}|f(t_3));(g(w)|\mathbf{id}\otimes\mathbf{id}|f(t_4))$$
$$=f(w\lfloor t_3;w\lfloor t_4).$$

The proof of the symmetric rule is omitted. In the subsequent case, we have to prove that $f(t_1|t;t_2\rfloor v)=f(t_3\rfloor u;t_4|t)$. Assume $f(t)=\eta;(t\otimes z)$ and $\partial_0(f(t))=g(\partial_0(t))=u$, $\partial_1(f(t))=g(\partial_1(t_1))=v$. Finally, let $\partial_1(\xi)=u'\oplus u''$ and $\partial_1(\eta)=u_\eta$. Let us firstly show an intermediate result:

$$f(t_1)[\,]f(t)=((\xi;(t'\otimes u''))\otimes w)[\,](\eta;(t\otimes z))$$
$$=((\xi\otimes w);(t'\otimes u''\otimes w))[\,](\eta;(t\otimes z))$$
$$=(w\otimes\xi);(w\otimes u''\otimes t')[\,](\eta;(t\otimes z))$$

$$= [((w \otimes \xi); (w \otimes u' \otimes u''))|\mathbf{id} \otimes \mathbf{id}|\eta]; [(w \otimes u'')|\mathbf{id} \otimes t'|t \otimes \mathbf{id}|z]$$
$$= [(w \otimes \xi)|\mathbf{id} \otimes \mathbf{id}|\eta]; [(w \otimes u'')|\mathbf{id} \otimes t'|t \otimes \mathbf{id}|z].$$

Thus, we have that

$$f(t_1|t; t_2 \rfloor v) = (f(t_1)[\,]f(t)); f(t_2)|\mathbf{id} \otimes \mathbf{id}|v)$$
$$= [(w \otimes \xi)|\mathbf{id} \otimes \mathbf{id}|\eta]; [(w \otimes u'')|\mathbf{id} \otimes t'|t \otimes \mathbf{id}|z];$$
$$[(w \otimes t'' \otimes v')|\mathbf{id} \otimes \mathbf{id}|v]$$
$$= [(w \otimes \xi)|\mathbf{id} \otimes \mathbf{id}|\eta]; [w|\mathbf{id} \otimes u''|\mathbf{id} \otimes t'|t \otimes \mathbf{id}|z];$$
$$[w|\mathbf{id} \otimes t''|\mathbf{id} \otimes v'|\mathbf{id} \otimes \mathbf{id}|v]$$
$$= [(w \otimes \xi)|\mathbf{id} \otimes \mathbf{id}|\eta]; [w|\mathbf{id} \otimes t''|\mathbf{id} \otimes t'|t \otimes \mathbf{id}|z]$$
$$= [(w \otimes \xi)|\mathbf{id} \otimes \mathbf{id}|\eta]; [w|\mathbf{id} \otimes t''|\mathbf{id} \otimes u'|\mathbf{id} \otimes \mathbf{id}|u_\eta];$$
$$[w|\mathbf{id} \otimes v''|\mathbf{id} \otimes t'|t \otimes \mathbf{id}|z]$$
$$= [(w \otimes \xi)|\mathbf{id} \otimes \mathbf{id}|\eta]; [(w \otimes t'' \otimes u')|\mathbf{id} \otimes \mathbf{id}|u_\eta];$$
$$[(w \otimes v'')|\mathbf{id} \otimes t'|t \otimes \mathbf{id}|z]$$
$$= [((w \otimes \xi); (w \otimes u' \otimes t''))|\mathbf{id} \otimes \mathbf{id}|\eta]; [(w \otimes v'')|\mathbf{id} \otimes t'|t \otimes \mathbf{id}|z]$$
$$= [((\xi; (u' \otimes t'')) \otimes w)|\mathbf{id} \otimes \mathbf{id}|u]; [(w \otimes v'' \otimes u')|\mathbf{id} \otimes \mathbf{id}|\eta];$$
$$[(w \otimes v'')|\mathbf{id} \otimes t'|t \otimes \mathbf{id}|z]$$
$$= (f(t_3)|\mathbf{id} \otimes \mathbf{id}|u); [(w \otimes v'' \otimes u')|\mathbf{id} \otimes \mathbf{id}|\eta];$$
$$[(w \otimes v'')|\mathbf{id} \otimes t'|t \otimes \mathbf{id}|z]$$
$$= (f(t_3)|\mathbf{id} \otimes \mathbf{id}|u); [(w \otimes v'' \otimes t')[\,]\eta; (t \otimes z)$$
$$= f(t_3 \rfloor u); (f(t_4)[\,]f(t))$$
$$= f(t_3 \rfloor u; t_4|t).$$

The proof of the symmetric rule is omitted.

Finally, in the case of synchronization, we have to prove that $f(t_1|t'_1; t_2|t'_2) = f(t_3|t'_3; t_4|t'_4)$. Assume that $f(t_1) = (\xi; (t_1 \otimes u_2)) \otimes w$, $f(t_2) = (v_1 \otimes t_2) \otimes w$, $f(t_3) = (\xi; (u_1 \otimes t_2)) \otimes w$, $f(t_4) = (t_1 \otimes v_2) \otimes w$, $f(t'_1) = (\xi'; (t'_1 \otimes u'_2)) \otimes w'$, $f(t'_2) = (v'_1 \otimes t'_2) \otimes w'$, $f(t'_3) = (\xi'; (u'_1 \otimes t'_2)) \otimes w'$ and $f(t'_4) = (t'_1 \otimes v'_2) \otimes w'$. Some intermediate results first.

$$f(t_1)[\,]f(t'_1) = ((\xi; (t_1 \otimes u_2)) \otimes w)[\,]((\xi'; (t'_1 \otimes u'_2)) \otimes w')$$
$$= ((\xi \otimes w); (t_1 \otimes u_2 \otimes w))[\,]((\xi' \otimes w'); (t'_1 \otimes u'_2 \otimes w'))$$
$$= [(\xi \otimes w)|\mathbf{id} \otimes \mathbf{id}|(\xi' \otimes w')]; [(u_2 \otimes w)|\mathbf{id} \otimes t_1|t'_1 \otimes \mathbf{id}|(u'_2 \otimes w')]$$

$$f(t_2)[\,]f(t_2') = ((v_1 \otimes t_2) \otimes w\,[\,]\,((v_1' \otimes t_2') \otimes w') = [(v_1 \otimes w)|\mathbf{id} \otimes t_2 | t_2' \otimes \mathbf{id}|(v_1' \otimes w')].$$

Let us abbreviate $\vartheta = [(\xi \otimes w)|\mathbf{id} \otimes \mathbf{id}|(\xi' \otimes w')]$. Hence

$$\begin{aligned}
f(t_1|t_1'; t_2|t_2') &= (f(t_1)[\,]f(t_1'));(f(t_2)[\,]f(t_2')) \\
&= \vartheta; [(u_2 \otimes w)|\mathbf{id} \otimes t_1 | t_1' \otimes \mathbf{id}|(u_2' \otimes w')]; [(v_1 \otimes w)|\mathbf{id} \otimes t_2 | \\
&\qquad\qquad t_2' \otimes \mathbf{id}(v_1' \otimes w')] \\
&= \vartheta; [u_2|\mathbf{id} \otimes w|\mathbf{id} \otimes t_1 | t_1' \otimes \mathbf{id}|u_2' \otimes \mathbf{id}|w']; \\
&\qquad\qquad [v_1|\mathbf{id} \otimes w|\mathbf{id} \otimes t_2 | t_2' \otimes \mathbf{id}|v_1' \otimes \mathbf{id}|w'] \\
&= \vartheta; [w|\mathbf{id} \otimes t_1 | t_1' \otimes t_2 | t_2' \otimes \mathbf{id}|w'] \\
f(t_3)[\,]f(t_3') &= ((\xi;(u_1 \otimes t_2)) \otimes w)[\,]((\xi';(u_1' \otimes t_2')) \otimes w') \\
&= ((\xi \otimes w);(u_1 \otimes t_2 \otimes w))[\,]((\xi' \otimes w');(u_1' \otimes t_2' \otimes w')) \\
&= [(\xi \otimes w)|\mathbf{id} \otimes \mathbf{id}|(\xi' \otimes w')]; [(u_1 \otimes w)|\mathbf{id} \otimes t_2 | t_2' \otimes \mathbf{id}|(u_1' \otimes w')] \\
&= \vartheta; [(u_1 \otimes w)|\mathbf{id} \otimes t_2 | t_2' \otimes \mathbf{id}|(u_1' \otimes w')] \\
f(t_4)[\,]f(t_4') &= ((t_1 \otimes v_2) \otimes w)[\,]((t_1' \otimes v_2' \otimes w') \\
&= [(v_2 \otimes w)|\mathbf{id} \otimes t_1 | t_1' \otimes \mathbf{id}|(v_2' \otimes w')] \\
f(t_3|t_3'; t_4|t_4') &= (f(t_3)[\,]f(t_3'));(f(t_4)[\,]f(t_4')) \\
&= \vartheta; [(u_1 \otimes w)|\mathbf{id} \otimes t_2 | t_2' \otimes \mathbf{id}|(u_1' \otimes w')]; \\
&\qquad\qquad [(v_2 \otimes w)|\mathbf{id} \otimes t_1 | t_1' \otimes \mathbf{id}|(v_2' \otimes w')] \\
&= \vartheta; [u_1|\mathbf{id} \otimes w|\mathbf{id} \otimes t_2 | t_2' \otimes \mathbf{id}|u_1' \otimes \mathbf{id}|w']; \\
&\qquad\qquad [v_2|\mathbf{id} \otimes w|\mathbf{id} \otimes t_1 | t_1' \otimes \mathbf{id}|v_2' \otimes \mathbf{id}|w'] \\
&= \vartheta; [w|\mathbf{id} \otimes t_2 | t_2' \otimes t_1 | t_1' \otimes \mathbf{id}|w'] \\
&= f(t_1|t_1'; t_2|t_2'). \qquad \square
\end{aligned}$$

Theorem 10.4 (Consistency w.r.t the truly concurrent semantics for CCS). *Given four different basic transitions t_1, t_2, t_3 and t_4 in N_{CCS}, then we have*

$$f(t_1;t_2) = f(t_3;t_4) \text{ implies } t_1 \,\mathbf{then}\, t_2 \,\chi\, t_3 \,\mathbf{then}\, t_4.$$

Proof. We know that $t_1;t_2 = t_3;t_4$ in $\mathbf{Con}_{\mathrm{CCS}}$ if and only if the two computations are mapped to the same concatenable process. Moreover, since we are assuming that the four transitions are different, we can prove that the process comprises only two

(non-ε-labelled) events, and these two events are causally independent. The proof of this fact is by induction on the structure of t_1 (and thus t_3).

- $t_1=[\mu.v\rangle$: In this case there is no transition t_3, different from t_1, starting from $\mu.v$, and thus the premise of the thesis is not satisfied.
- $t_1=t_1<+v_1'$, *where* $t:v_1 \xrightarrow{\mu} v_2$: In this case there can be several candidate transitions t_3.
  (a) $t_3=v_1+>t'$, where $t':v_1' \xrightarrow{\mu} v_2'$. Impossible, because t_1 and t_3 generate different concatenable processes (alternative choices).
  (b) $t_3=t''<+v_1'$, where $t'':v_1'' \xrightarrow{\mu} v_2''$. The problem is then reduced to the simpler case $t;t_2=t'';t_4$ implies $t\,\mathbf{then}\,t_2\,\chi\,t''\,\mathbf{then}\,t_2\,\chi\,t''\,\mathbf{then}\,t_4$

which holds by inductive hypothesis.

- $t_1=v_1'+>t$, *where* $t:v_1 \xrightarrow{\mu} v_2$: symmetrically.
- $t_1=t^1\backslash\alpha$: In this case, all the other involved transitions must have a similar format, $t_i=t^i\backslash\alpha$, and thus the problem is reduced to the inductive case $t^1;t^2=t^3;t^4$ implies $t^1\,\mathbf{then}\,t^2\,\chi\,t^3\,\mathbf{then}\,t^4$.
- $t_1=t^1[\Phi]$: Analogously to the previous case.
- $t_1=t\rfloor u_1$, *where* $t:v_1 \xrightarrow{\mu} v_2$:

In this case there can be several candidate transitions t_3.

(a) $t_3=v_1\lfloor t'$, where $t':u_1 \xrightarrow{\mu} u_2$. In order to have $t_1;t_2=t_3;t_4$, it is necessary that t_2 produces the same event generated by t_3, and t_4 the same by t_1. This univocally forces the following assignments:

$$t_2=v_2\lfloor t' \qquad t_4=t\rfloor u_2.$$

It is patent that this case corresponds to the axiom in Definition 7.6.

(b) $t_3=t''\rfloor u_1$, where $t'':v_1 \xrightarrow{\mu} v_2''$. In order to be able to generate the same process, both t_2 and t_4 have $\rfloor$ as principal operator. Therefore, the problem is reduced to the simpler check on the left (sub) transitions.

(c) $t_3=t''|t'$. Again, the same concatenable process can be generated only if t_2 produces the same event generated by t_3, and t_4 the same by t_1. This forces the definition of the two transitions; moreover, this means that we need an inductive check on the left subtransitions.

- $t_1=u_1\lfloor t$, *where* $t:v_1 \xrightarrow{\mu} v_2$: Symmetrically, to the previous case.
- $t_1=t|t'$: There are three cases. Two of them (when t_3 is an asynchronous move) are already covered by the previous two cases. The last is when $t_3=t''|t^*$. Again, by generating the same concatenable process, t_2 and t_4 have a fixed definition; moreover, we need two checks on both the right and the left subtransitions.

Hence, we have exhaustively checked all the possible cases (recursion is ignored because of the recursion axiom), and it is easy to recognize in the proof the same axiom and rules of Definition 7.6. □

11. Comparisons and related works

In this section we will discuss the relationship with previous proposals in the area, in order to emphasize the analogies and the differences.

11.1. Algebras vs. calculi of nets

There is a relevant line of research in net theory which sets as its goal the definition of algebras of nets, yielding compositionality as its main achievement. Some of these algebras have been shown to be useful as semantic domain in a denotational net semantics for CCS-like languages. In this line, we mention only a few early proposals [28, 20, 43, 19, 9], where a Petri net can be specified by a formula of the proposed algebra.

On the contrary, here we follow the line of defining a calculus of nets, i.e. a deductive system made of axioms and inference rules, similarly to what Milner [34] did with CCS, which, in our perspective, can be considered as a calculus of transition systems. CCS is defined as a single whole transition system by means of an SOS specification; analogously, our net calculus defines a single whole net. Moreover, if one is interested in the behaviour of a particular CCS agent, the relevant piece of transition system is the part reachable from the state corresponding to the agent; similarly, we can single out a subnet corresponding to a SCONE (or SCONE$^+$) marking. This is a fairly new result in the context of nets. Indeed, the "algebra of nets" approach lacks the pleasant feature of having a single net comprising all the agent subnets. Recent ideas proposed by Degano et al. [11, 12, 14, 15] go in the direction of transforming concurrent calculi, like CCS, TCSP and ACP, into net calculi. In a sense, here we try to algebraically formalize some of the ideas developed there and in other related works [36, 41].

There are at least two main advantages in considering a calculus instead of an algebra. First, infinite nets are never produced as a whole, but only on a by-need basis; indeed, SOS specifications provide finite intensional descriptions for possibly infinite semantic objects. Second, an SOS specification is a natural guide for building an abstract interpreter of the language.

11.2. Distributed implementations of CCS

At first sight, one could think that the distributed implementation of a CCS agent E is determined by g; i.e. the implementation of E is the subnet reachable from the marking $g(E)$. This is wrong, as shown in Example 6.8, where certain transitions in the reachable subnet are not to be included. Nonetheless, one could think that the implementation of E is the net underlying $\langle f, g\rangle(N_{\mathrm{CCS}}(E))$, i.e. the support net of the image of the transition system through the mapping. Again, this is wrong, as certain computations in the net do not have a corresponding computation in the transition system. Consider the net in Fig. 9(b), which is the net underlying $\langle f, g\rangle$ $(N_{\mathrm{CCS}}(E))$ for $E=((\alpha+\beta)|(\alpha^{-}+\delta))\backslash\alpha$. It is clear that the net may deadlock (when, e.g., the left token

chooses to enable β and the right one to enable the synchronization), but no similar phenomenon happens in $N_{CCS}(E)$. As a matter of fact, the implementation of E is not simply the net underlying $\langle f,g\rangle(N_{CCS}(E))$, rather such a net plus the morphism f, which specifies which net computations are to be considered. Indeed, in our approach CCS is given two operational semantics describing the evolution of its agents at two different levels of detail. On the one hand, the CCS transition system specifies the *control level*, in the sense that it defines the atomic actions of the systems, and thus also the states which can be tested by an observer. On the other hand, its net implementation determines the *description level*, i.e. it defines the actual steps the machine has to perform, although the states the machine passes through may not be observable. In this perspective, the mapping f is the natural means of imposing an atomicity constraint on the executions of the net, resulting in what we have called net transactions (see Section 11.3 for more details).

This is a natural consequence of our algebraic approach, where the algebra of CCS transitions is mapped to an algebraic theory of SCONE$^+$ computations. Indeed, this is what happens in most denotational semantic definitions, where basic operators of the language are mapped to derived operators of the semantic domain. Often, some basic operators of the semantic domain are not in the image of the mapping. To be more explicit, consider the following simple example on the algebra $\mathscr{A}$ of natural numbers (0 is the constant and $'$ is the successor function). We define an implementation mapping $f:\mathscr{A}\to\mathscr{A}$ defined as follows:

$$f(0)=0 \qquad f(x')=f(x)''$$

Hence, f maps each number x to its double $2x$. The algebraic theory $f(\mathscr{A})$ contains all (and only) even numbers; hence, $f(\mathscr{A})$ does not contain the successor operation, but only the $+2$ operator, defined as derived in terms of the successor one. Similarly, SCONE internal choice is not an operation of the algebraic theory on $\mathscr{G}_{N_{SCONE}}$ but it is used to express its derived operators, i.e. its transactions.

The problem of giving a distributed implementation of concurrent calculi has been studied by assuming that the abstraction level of the net is the same as the one of the transition system (i.e. each action is always represented by a single transition). Some proposals by Degano, De Nicola, Montanari and Olderog [12, 36] respect such a requirement, but the price they pay is to obtain an involved net semantics because of the distribution of choice. In our view, the present solution is simpler: it rejects the assumption that the control and the description level coincide and makes an extensive use of the atomicity constraint, formalized through the implementation mapping.

As a final comment, we want to stress that our distributed implementation of CCS has the interesting effect – on the source transition system – of reducing the true concurrency of the language, defined through relation χ, to the true concurrency of a model, expressed in algebraic terms by the functoriality axioms.

11.3. Atomicity and transactions

The idea of representing the sophisticated nondeterministic operator CCS in terms of local choice has been already proposed in the literature. The problem was attacked in [24], which described a new interleaving semantics for CCS where the inference rules for nondeterminism and recursion have been replaced by the corresponding axioms of internal choice (like in SCONE) and recursion unwinding. Of course, a calculus more generous than CCS is obtained, where certain derivable transitions are forbidden in Milner's transition system. To prevent such "erroneous" transitions, a subagent performing an internal choice should have priority w.r.t. the other concurrent subagents willing to perform nonchoice actions. In this way, a sort of atomicity constraint is imposed on the calculus, which hence becomes equivalent to CCS. In fact, it represents a lower level description of CCS where each internal choice and recursion unwinding is seen as a separate move. The interesting point is that this atomicity constraint is ingrained in the syntax-driven deductive[21] system.

In [13], a distributed semantics for CCS is proposed, which looks like our present proposal. Indeed, the modelled language has internal choice only. In order to recover the correct CCS semantics, a mechanism of atomicity has to be put on the net. The authors suggested a notion of μ-transaction, essentially coincident to the one proposed in Definition 9.6, to denote the *atomic* steps on the net, with the intuition that these are the sole feasible moves on the net. In this way, a direct consistency with Milner's interleaving semantics is easily preserved and also the correct causal dependencies between concurrent actions are faithfully reproduced. The main difference between [13] and our proposal relies on the fact that restricting the behaviour of the net to μ-transactions is more naturally expressed by the morphism $f: T_{CCS} \to T$. Of course, the definition of such a morphism is possible only in the completely algebraic framework we are working in. The atomicity constraint is guaranteed by the definition of the basic CCS operations as derived operators of the SCONE$^+$ theory.

A first attempt in giving a morphism from the CCS transition system to a Petri net with internal nondeterminism only is reported in [4].

11.4. Finite net implementation of RCCS agents

The construction we have presented in this paper for giving a finite (nonsafe) representation to RCCS agents has analogies with similar proposals in the literature. In particular, Goltz [19] was the first who generalized to the case of nets the construction for the recursive combinator given by Milner in [33] for transition systems. Our proposal differs from this mainly w.r.t. the nondeterministic operation, which is centralized in our approach and distributed there. Unfortunately, her

[21] Note that we have not addressed the problem of defining a "prioritized" token game on the SCONE$^+$ net, to ensure that only CCS transactions are executable. This problem is outside the scope of the paper.

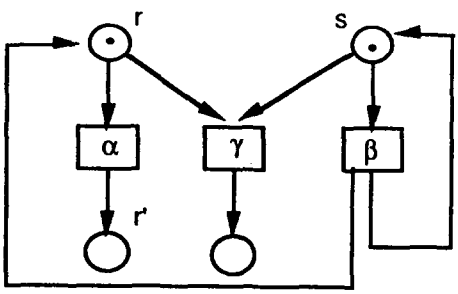

Fig. 13. The net for $\mathrm{rec}\, x.(\gamma+(\alpha \oplus \beta.x)$ as proposed in [19].

construction is sometimes inadequate, as the following counterexample (proposed by the author herself) shows. Consider the RCCS agent $\mathrm{rec}\, x.(\gamma+(\alpha|\beta.x))$ and its SCONE implementation depicted in Fig. 6(a). The net depicted in Fig. 13 is the one Goltz associates to this recursive term. Unfortunately, it is incorrect, because action γ may be enabled by tokens which remain in place r. To be more precise, let $v=\mathrm{rec}\, x.\gamma+(\alpha|\beta.x)$, which corresponds in the net to the marking $\mathrm{r} \oplus \mathrm{s}$. According to the operational semantics in Section 4, transition $\gamma+>(\alpha\lfloor[\beta,v\rangle)$ is labelled by β and reaches state $\alpha|v$, corresponding to $2\mathrm{r} \oplus \mathrm{s}$. Then, transition $(\alpha\lfloor(\gamma+>([\alpha,\mathrm{nil}\rangle\rfloor\beta.v))$ is labelled by α and reaches $\alpha|\mathrm{nil}|\beta.v$ from which actions α and β only can be performed. The simulation of these two steps on the net leads erroneously to marking $\mathrm{r} \oplus \mathrm{r}' \oplus \mathrm{s}$ where γ is enabled, too. Instead, our solution, depicted in Fig. 6(b), correctly represents the intuitive causality and the possible conflicts among the three actions.

An interesting recent report by Goltz and Rensink [22] shows that, by assuming that each action is implemented exactly as one net transition (i.e. that the abstraction levels of the transition system and of the net are the same), no finite net representation for RCCS agents is possible when a causal semantics is to be respected. Hence, it gives evidence that the assumptions of our solution are strictly necessary.

RCCS implementation mapping enjoys an interesting property. The theory $\langle f,g\rangle(N_{\mathrm{CCS}}(E))$ and its underlying net have "essentially" the same semantics: each computation on the underlying net can always be extended to become the image of a RCCS computation via f. In other words, the atomicity constraint imposed by f is somehow superfluous, as no deadlock, due to conflicting internal choices, can be reached. To help intuition, compare the differences between the nets in Figs. 4 and 9(b), where the presence of the restriction operator in the latter example produces possible deadlocks.

Is it possible to find out a finite P/T net representation for any CCS agent? It has been recognized [20, 41] that finite representations for full CCS do not exist since full CCS is "Turing-powerful" whilst finite P/T nets are not. Nonetheless, since non 1-safe representations are usually smaller, it would be interesting to discover if it is possible to deal with restriction without introducing 1-safe nets. A recent proposal [44] shows that this can be done with the help of inhibitor arcs.

11.5. Programming by multiset transformation

There has been a recent deep interest in finding inherently "truly concurrent" abstract machines which resulted in a series of proposals [1, 8, 7] ending with the *chemical abstract machine* by Berry and Boudol [2]. The basic paradigm of all these proposals can be called "programming by multiset transformation", where the sequential components of a system are organized in a multiset, each of which can autonomously proceed or interact. Anyway, Petri nets *are* abstract machines which do work by multiset transformation: indeed, the *reaction law* of the chemical abstract machine just corresponds to the definition of net transitions and the *chemical law* is simply another way of saying that the token game can be played in parallel. More abstractly, as Meseguer pointed out in [31], all these models are rewriting systems where the application of rewriting rules may be done in parallel.

If we consider SCONE and its net semantics, we can observe that it can be seen as an algebraic (hence structural) representation of the basic features of the chemical abstract machine, namely concurrency and communication. Indeed, the *parallel* rule is multiset union, *reaction* corresponds to the operation of communication, and *inaction cleanup* accounts for nil as neutral element in multiset union.

When considering SCONE$^+$, a relevant difference arises concerning the treatment of restriction. The chemical abstract machine introduces to this aim two new concepts, namely *membranes* and *airlocks*, which allow to give an environment-like structure to the system. These two concepts do not have any corresponding concept in the classical net theory. Indeed, Degano, De Nicola and Montanari proposed an alternative solution, which we have followed here: parallel composition is modelled as disjoint union *via* the auxiliary unary operators of context $\_|\mathbf{id}$ and $\mathbf{id}|\_$. It is not clear to us which of the two solutions is more amenable. On the one hand, the notion of membrane and airlock is appealing because it more faithfully describes the structure of restriction at the machine level. On the other hand, the mechanism is rather heavy (a lot of rewritings are needed in order to create an ion in a solution ready to reaction) if compared with the direct definition of communication transitions we give also in the presence of restriction.

Acknowledgement

We would like to thank the anonymous referees for their comments and suggestions which were very helpful in the preparation of the revised version of this paper. In particular, one referee suggested that we investigate the conditions under which the reverse of Theorem 7.8 (reported as Theorem 7.9 and Proposition 7.11) holds.

References

[1] J.P. Banâtre and D.Le Matayer, Programming by multiset transformation, *Comm. ACM* **36** (1993) 98–111.

[2] G. Berry and G. Boudol, The chemical abstract machine, *Theoret. Comput. Sci.* **96** (1992) 217–248.

[3] E. Best and R. Devillers, Sequential and concurrent behaviour in Petri net theory, *Theoret. Comput. Sci.* **55** (1987) 87–136.

[4] C. Blanco, Hacer Explícita la Elección Atómica de CCS Facilita la Construcción de Ordenes Parciales, Master's Thesis, ESLAI, Buenos Aires, 1988.

[5] G. Boudol and I. Catellani, Permutation of transitions: an event structure semantics for CCS and SCCS, in: *Proc. REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, Noordwijkerhout, Lecture Notes in Computer Science Vol. 354 (Springer, Berlin, 1989) 411–437.

[6] G. Boudol and I. Castellani, Three equivalent semantics for CCS, in: *Proc. 18th Ecole de Printemps sur la Semantique de Parallelism*, La Roche-Posay, Lecture Notes in Computer Science, Vol. 469 (Springer, Berlin, 1990) 96–141.

[7] N. Carriero and D. Gelerntner, Linda in context, *Comm. ACM* **32** (1989) 444–458.

[8] M. Chandy and J. Misra, *Parallel Program Design* (Addison-Wesley, Reading, MA, 1988).

[9] L. Cherkasova, Posets with non-actions: a model for concurrent nondeterministic processes, Arbeitspapiere der GMD no. 403, 1989.

[10] A. Corradini, G.L. Ferrari and U. Montanari, Transition systems with algebraic structure as models of computation, in: *Proc. 18th Ecole de Primtemps sur la Semantique de Parallelism*, La Roche-Posay, Lecture Notes in Computer Science, Vol 469 (Springer, Berlin, 1990) 185–222.

[11] P. Degano, R.De Nicola and U. Montanari, Partial ordering derivations for CCS, in: *Proc. FCT '85*, Lecture Notes in Computer Science, Vol. 199 (Springer, Berlin, 1985) 520–533.

[12] P. Degano, R.De Nicola and U. Montanari, A distributed operational semantics for CCS based on condition/event systems, *Acta Inform.* **26** (1988) 59–91.

[13] P. Degano, R.De Nicola and U. Montanari, Partial ordering description of nondeterministic concurrent systems, in: *Proc. REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, Noordwijkerhout, Lecture Notes in Computer Science, Vol. 354 (Springer, Berlin, 1989) 438–466.

[14] P. Degano, R. Nicola and U. Montanari, Partial ordering semantics for CCS, *Theoret. Comput. Sci.* **75** (1990) 223–262.

[15] P. Degano, R. Gorrieri and S. Marchetti, An exercise in concurrency: a CSP process as a condition/event system, in: *Advances in Petri Nets* 1988, *Lecture Notes in Computer Science*, Vol. 340 (Springer, Berlin, 1988) 85–105.

[16] P. Degano, J. Meseguer and U. Montanari, Axiomatizing net computations and processes, in: *Proc. Logic in Computer Science (LICS '89)*, Asilomar (1989) 175–185.

[17] G. Ferrari, Unifying models for concurrency, Ph.D. Thesis, TD 4/90, Dip. di Informatica, Pisa, 1990.

[18] G. Ferrari and U. Montanari, Towards the unification of models for concurrency, in: *Proc. Coll. on Algebra and Trees in Prog. (CAAP '90)*, Copenhagen, Lecture Notes in Computer Science, Vol. 431 (Springer, Berlin, 1990) 162–176.

[19] U. Goltz, On representing CCS programs by finite Petri nets, in: *Proc. Mathematical Foundations of Computer Science (MCFS '88)*, Lecture Notes in Computer Science, Vol. 324 (Springer, Berlin, 1988) 339–350. Full version in Über die Darstellung von CCS-Programmen durch Petrinetze, Ph.D. Thesis, RWTH Aachen, 1988.

[20] U. Goltz and A. Mycroft, On the relationships of CCS and Petri nets in: *Proc. 11th Int. Conf. on Automata, Languages and Programming (ICALP '84)*, Lecture Notes in Computer Science, Vol. 172 (Springer, Berlin, 1984) 196–208.

[21] U. Goltz and W. Reisig, The non-sequential behaviour of Petri nets, *Inform. and Control* **57** (1983) 125–147.

[22] U. Goltz and A. Rensink, Finite Petri nets as models for recursive causal behaviour, Arbeitspapiere der GMD no. 604, *Theoret. Comput. Sci.* **124** (1994) 169–179.

[23] R. Gorrieri, Refinement, atomicity and transactions for process description languages, Ph.D. Thesis, TD-2/91, Dipartimento di Informatica, Università di Pisa, 1991.

[24] R. Gorrieri, S. Marchetti and U. Montanari, A^2CCS: atomic actions for CCS, *Theoret. Comput. Sci.* **72** (1990) 203–223.
[25] R. Gorrieri and U. Montanari, SCONE: a simple calculus of nets, in: *Proc. CONCUR '90*, Amsterdam, Lecture Notes in Computer Science, Vol. 458 (Springer, Berlin, 1990) 2–30.
[26] R. Gorrieri and U. Montanari, Distributed implementation of CCS, in: *Advances in Petri Nets 1993*, Lecture Notes in Computer Science, Vol. 674 (Springer, Berlin, 1993) 244–266.
[27] R. Keller, Formal verification of parallel programs, *Comm. ACM* **19** (1976) 561–572.
[28] V. Kotov, An algebra for parallelism based on Petri nets, Lecture Notes in Computer Science, Vol. 64 (Springer, Berlin, 1978) 39–55.
[29] S. Mac Lane, *Categories for the Working Mathematicians* (Springer, Berlin, 1971).
[30] V. Manca, A. Salibra and G. Scollo, Equational type logic, *Theoret. Comput. Sci.* **77** (1990) 1–29.
[31] J. Meseguer, Rewriting as a unified model of concurrency, in: *Proc. CONCUR '90*, Amsterdam, Lecture Notes in Computer Science, Vol. 453 (Springer, Berlin, 1990) 384–400.
[32] J. Meseguer and U. Montanari, Petri nets are monoids, *Inform. and Comput.* **88** (1990) 105–155.
[33] R. Milner, A complete inference system for a class of regular behaviours, *J. Comput. System Sci.* **28** (1984) 439–466.
[34] R. Milner, *Communication and Concurrency* (Prentice-Hall, Englewood Cliffs, NJ, 1989).
[35] U. Montanari and D. Yankelevich, An algebraic view of interleaving and distributed operational semantics for CCS, in: *Proc. 3rd Conf. on Category Theory in Computer Science*, Manchester, Lecture Notes in Computer Science, Vol. 389 (Springer, Berlin, 1989) 5–20.
[36] E.-R. Olderog, *Nets, Terms and Formulas* (Cambridge Univ. Press, Cambridge, Vol. 1991).
[37] D. Park, Concurrency and automata on infinite sequence, in: *Proc. GI*, Lecture Notes in Computer Science, Vol. 104 (Springer, Berlin 1981) 167–183.
[38] C.A. Petri, Kommunication mit Automaten, Schriften des Institutes fur Instrumentelle Mathematik, Bonn, 1962.
[39] G. Plotkin, A structural approach to oprational semantics, Tech. Report DAIMI FN-19, Department of Computer Science, Aarhus University, Aarhus, 1981.
[40] W. Reisig, *Petri Nets: An Introduction*, EACTS Monographs on Theoretical Computer Science (Springer, Berlin, 1985).
[41] D. Taubner, *Finite Representation of CCS and TCSP Programs by Automata and Petri Nets*, Lecture Notes in Computer Science, Vol. 369 (Springer, Berlin, 1989).
[42] G. Winskel, Event structure for CCS and related languages, in: *Proc. 9th Inf. Conf. on Automata, Languages and Programming (ICALP '82)*, Lecture Notes in Computer Science, Vol. 140 (Springer, Berlin, 1982) 561–576.
[43] G. Winskel, Petri nets, algebras, morphisms and compositionality, *Inform. and Comput.* **72** (1987) 197–238.
[44] N. Busi, R. Gorrieri and G. Siliprandi, Distributed conflicts in communicating systems, TR-94-8, Università di Bologna, 1994.