A Distributed Operational Semantics for CCS Based on Condition/Event Systems

Pierpaolo Degano *, *Rocco De Nicola* + *and Ugo Montanari* *,[1]

* Dipartimento di Informatica, Università di Pisa, PISA
+ Istituto di Elaborazione dell'Informazione, C.N.R., PISA

Thu, Sep 17, 1987

Abstract. A new set of inference rules for the guarded version of Milner's Calculus of Communicating Systems is proposed. They not only describe the actions agents may perform when in a given state, but also say which parts of the agents move when the global state changes. From the transition relation a Condition/Event system, called Σ_{CCS}, is immediately derived and two demonstrations are given of its adequacy as a truly concurrent and distributed CCS semantics. First, we prove that the abstract interleaving case graph of Σ_{CCS} (i.e., the case graph with arcs labelled by single actions) is homomorphic to the transition system defined by Milner's inference rules. Since the homomorphism preserves transitions, we also have that the synchronization trees of two corresponding nodes are identical. Thus, our construction gives a semantics which is consistent with the original interleaving semantics of CCS whatever behavioural equivalence is chosen. Second, we prove that the transition system expressing a multiset semantics for CCS is transition-preserving homomorphic to the abstract case graph of Σ_{CCS} (i.e., the case graph with arcs labelled by multisets of actions) and thus that the amount of parallelism exhibited by Σ_{CCS} and by multiset CCS is the same.

Running head:
A Distributed Semantics for CCS Based on C/E Systems

---

[1] Research performed in part while on leave at SRI International, Menlo Park, USA.
Research supported by Office of Naval Research Contract N00014-86-C-0450.

1. Introduction

The operational semantics of concurrent languages has often been described in terms of labelled transition systems [13]. The states are terms of the language and the transitions between states are labelled by actions [1, 14, 16]. This approach relies on the well-known idea of describing system behaviours as sequences of transitions. Terms are considered to be processes that may perform certain actions to become other processes; the fact that process p evolves to process q by performing an action α is rendered by $p \xrightarrow{\alpha} q$.

A drawback of this approach is that it describes transitions between global states only, and does not offer a full account of the causal dependencies between the actions (possibly due to independent/parallel subsystems) which are performed when passing from one state to another. As a result, transition systems provide concurrent languages with a so-called *interleaving operational semantics*, where the operator for the parallel composition of processes is not primitive: given any finite term containing this operator, another term always exists without it and exhibiting the same behaviour. In the case of CCS this property is reflected by the so-called expansion theorem [14].

A slight variation of the transition systems approach includes multisets of actions as possible transition labels; languages like Meije [1] and SCCS [15] are equipped with an operational semantics based on this generalization. Multiset labels give a direct representation of the amount of "parallelism" available in the system and help in distinguishing the behaviour of systems in some obvious cases, like $\alpha NIL|\beta NIL$ and $\alpha\beta NIL + \beta\alpha NIL$. However, causal dependencies remain non-recoverable (for instance, the behaviour of $\alpha NIL|\beta NIL + \alpha\beta NIL$ and that of $\alpha NIL|\beta NIL$ cannot be differentiated, see also Figure 6.1); weaker versions of the expansion theorem still hold. Indeed, labelled transition systems turn out to be satisfactory to describe temporal dependencies between actions or multisets of actions but not causal dependencies.

Our long term goal is to equip concurrent languages with operational definitions which are also able to describe their causal behaviour. In this paper, we show how the centralized assumption on transition systems can be relaxed to express this kind of operational semantics by letting Petri Nets play the rôle of "distributed" transition systems. Indeed, Petri Nets [10, 19] can be viewed as a generalization of transition systems, in which concurrency and causal dependency between transition occurrences are explicitly represented. To make the choice of a particular Petri net less arbitrary, we propose two properties which we consider essential for any net semantics of a language previously equipped with an interleaving and a multiset semantics. We will require that:

i) the interleaving semantics is retrievable from the net semantics;
ii) the net semantics captures all and only the parallelism present in the multiset semantics of the language.

As a case study, we extend the original interleaving semantics of Milner's Calculus of Communicating Systems (CCS) [14] to provide the language with a causality preserving semantics based on Petri Nets.

Indeed, CCS, which is a primitive language for describing communicating processes with a (deceptively) simple and well-defined operational semantics, has been a touchstone for many of the proposed theories of concurrency. On the other hand, many different kinds of Petri Nets have been defined. They can be divided into two families. The first contains those more intensional nets which represent models of real systems, and may thus have cycles. The nets in the second family can be seen as behaviours of systems, with a partial ordering structure obtained by unfolding nets of the first class. Privileged representatives of the first class are Condition/Event (C/E) Systems, where the occurrence of *events* causes changes in local states, called *conditions*. Typical elements of the second class are Occurrence Nets.

We map CCS into a minor extension of C/E Systems which we call Augmented Condition/Events (A-C/E) Systems. Actually, we consider "pure" and "guarded" CCS, i.e., the calculus without value passing and, like in the original definition of [14], with all the variables bound in recursively defined terms prefixed by an action. The proposed extension to C/E Systems permits self-loops and is needed to enable a transition for which some preconditions coincide with some postconditions. Self-loops arise naturally when considering recursive CCS agents like rec x. αx. The conditions for enabling events and the property of contact-freeness have been modified accordingly, still maintaining the properties that *an event whose preconditions hold is always enabled*, and that *no token is ever lost.* An extension similar to ours is proposed in [21].

In order to define the new CCS operational semantics, like in the traditional interleaving semantics, we follow the Structured Operational Semantics (SOS) approach [18] which provides the means of defining the transitions of a compound system in a syntax-driven way. A different transition relation is introduced which we call *partial ordering derivation relation.* It relates sequential (sub)processes of CCS agents, rather than their whole global states. Roughly, sequential processes are obtained by decomposing CCS agents, and the partial ordering derivation relation describes both the actions the sequential processes may perform and their effects. The transitions defined by the derivation relation have the form H' - K'—μ→H" where H' and H" represent the initial and final sets of sequential processes, μ is an action, and K' has a purely technical rôle. The A-C/E system, called Σ_{CCS}, is obtained straightforwardly

from the partial ordering derivation. More precisely, *sequential processes* are *conditions*; *decompositions of CCS agents* are *cases*; and *elements of the partial ordering derivation relation* (discarding the second argument K') are *events*.

Our achievement is twofold: first we equip CCS with a fully concurrent and fully distributed semantics: concurrency and causal dependencies between the actions the various agents can perform are explicitely represented. Second, we provide the basis for an adequate linguistic level for the particular class of Petri Nets which can be defined through CCS operators, and move a step toward removing one of the generally acknowledged inadequacies of Petri Nets: their lack of compositionality and modularity. Our results should also provide a framework for evaluating the expressive power of CCS in this context, and for understanding the relationships between CCS and Petri Nets, so that analytic concepts and techniques might be transferred form one theory to the other.

Our concurrent distributed semantics for CCS is adequate in that it satisfies criteria i) and ii) discussed above.

Criterion i) is shown to hold by proving that the *abstract interleaving* case graph of Σ_{CCS} (i.e., the case graph with arcs labelled by single actions) is homomorphic to the transition system defined by Milner's inference rules. Since this homomorphism preserves transitions, the synchronization trees of two corresponding nodes are *identical*. Thus, our construction gives a concurrent semantics which is consistent with the original interleaving semantics of CCS whichever behavioural equivalence is chosen.

In order to prove that criterion ii) is satisfied, we introduce a new transition system which, following the approach of Meije [1], precisely defines the multisets of independent actions the agents may perform. We then prove that the A-C/E system we associate to CCS has the same capabilities of this transition system. Formally, we prove that the multiset transition system is transition-preserving homomorphic to the *abstract* case graph of Σ_{CCS} (i.e., the case graph with arcs labelled by multisets of actions instead of sets of events). Also in this case, the synchronization trees of two corresponding nodes are identical.

The net semantics we propose is such that truly concurrent behaviour is also exhibited in the presence of nondeterminism or recursion. For instance, the case corresponding to agent $E = \alpha NIL|\beta NIL + \gamma NIL$ (see Figure 1.1) contains two conditions, and three events, labelled by α, β and γ, are enabled by them. The two conditions, which are obtained by decomposing E, are $\alpha NIL|id + \gamma NIL$ and $id|\beta NIL + \gamma NIL$. Conditions are represented by special terms which represent sequential processes and are called *grapes*. Events α and β have disjoint preconditions, and thus can fire in parallel. Conversely, both pairs $<\alpha, \gamma>$ and $<\beta, \gamma>$ have intersecting preconditions and thus cannot fire in parallel. According to Milner's transition relation we have $\alpha NIL|\beta NIL + \gamma NIL \xrightarrow{\alpha} NIL|\beta NIL$. Here when event α fires, the case $\{NIL|id, id|\beta NIL + \gamma NIL\}$ is reached which contains a *non updated* sequential process, i.e., a

process where the γNIL choice is still present. This is a necessary consequence of the *distributed control* in Petri Nets: a condition *not* involved in the transition *cannot* be modified. However, we must still make sure that the set {NIL|id, id|βNIL + γNIL} has the same behaviour as the sets of grapes obtained by decomposing agent NIL|βNIL.

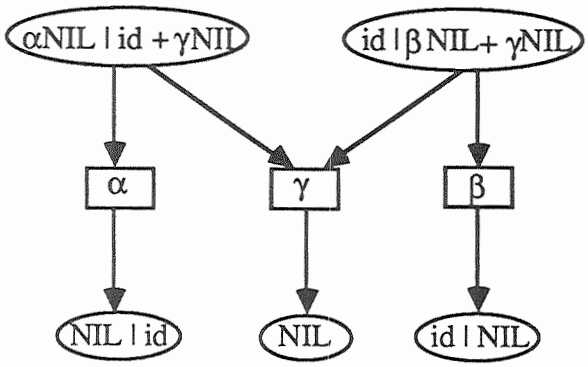

Fig. 1.1. A C/E system corresponding to CCS agent αNIL|βNIL + γNIL. The initial case consists of the two topmost conditions.

This simple example gives an idea of the difficulties we have encountered. We need to represent global decisions in a completely distributed way, and in doing so we have to give up a one-to-one correspondence between agents and cases. A rather clean result such as that presented here is in our view rather surprising. The proofs, which rely on syntax-directed techniques, are often simple.

The rest of the paper is organized as follows: Section 2 summarizes the indispensable notions of CCS, and presents our proposed extension of Petri's C/E Systems. Section 3 contains the decomposition and the partial ordering derivation relations for CCS agents. Section 4 shows how the new derivation relation naturally leads to an Augmented C/E system, the behaviour of which is in agreement with the original interleaving operational semantics and with a multiset based semantics. Finally, Section 5 compares our approach with other work on related subjects and Section 6 contains some concluding remarks.

2. Background

Here, we briefly introduce the relevant definitions for Milner's Calculus of Communicating Systems (CCS) [14], Petri's Condition/Event (C/E) Systems [10, 19] with the extensions needed to describe the operational semantics of CCS agents.

First, we recall the operators of the CCS, and then we present the traditional interleaving operational semantics of [14].

Definition 2.1.

Let $\Delta = \{\alpha, \beta, \gamma \ldots\}$ be a fixed set and $\Delta^- = \{\alpha^- \mid \alpha \in \Delta\}$, then $\Lambda = \Delta \cup \Delta^-$ (ranged over by λ) will be used to denote the set of *visible actions*. Moreover, symbol τ will be used to denote a distinguished *invisible action* not in Λ; $\Lambda \cup \{\tau\}$ will be ranged over by μ.

CCS **terms** are the terms generated by the following BNF-like grammar:

$T ::= x \mid NIL \mid \mu T \mid T\backslash\alpha \mid T[\phi] \mid T + T \mid T|T \mid rec\ x.\ T,$

where x is a variable and ϕ is a bijection of $\Lambda \cup \{\tau\}$ which preserves τ and the operation $^-$ of complementation. ♦

Following [14] we impose a slight constraint on CCS terms: all recursively defined terms are such that every occurrence of the bound variable is prefixed by an action from $\Lambda \cup \{\tau\}$. This restriction guarantees that all recursive definition of agents have a unique solution.

Definition 2.2.

Let *guarded* be the least relation on CCS terms which satisfies:

i) NIL and μT are *guarded*;

ii) if T is *guarded* then $T\backslash\alpha$, $T[\phi]$ and rec x. T are *guarded*;

iii) if T and T' are *guarded* then T + T' and T|T' are *guarded*. ♦

Definition 2.3.

CCS **agents** are *guarded* CCS terms which do not contain any free variable. ♦

In the rest of the paper we will use E and T (possibly indexed) to range over CCS agents and terms, respectively; moreover we will assume that the precedence of the operators is given by: $\backslash\alpha > [\phi] > \mu > rec > + > |$.

CCS interleaving operational semantics is based on a labelled transition system; the transition relation of this system is defined by the following set of inference rules over agents.

Definition 2.4.

Milner's derivation relation $E_1 \text{—}\mu\rightarrow E_2$ is defined as the least relation satisfying the following axiom and inference rules:

Act) $\mu E \text{—}\mu\rightarrow E$

Res) $E_1 \text{—}\mu\rightarrow E_2$ **and** $\mu \notin \{\alpha,\alpha^-\}$ **implies** $E_1\backslash\alpha \text{—}\mu\rightarrow E_2\backslash\alpha$,

Rel) $E_1 \text{—}\mu\rightarrow E_2$ **implies** $E_1[\phi] \text{—}\phi(\mu)\rightarrow E_2[\phi]$

Sum) $E_1 \text{—}\mu\rightarrow E_2$ **implies** $E_1+E \text{—}\mu\rightarrow E_2$ **and** $E+E_1 \text{—}\mu\rightarrow E_2$

Com) $E_1 \text{—}\mu\rightarrow E_2$ **implies** $E_1|E \text{—}\mu\rightarrow E_2|E$ **and** $E|E_1 \text{—}\mu\rightarrow E|E_2$

$E_1 \text{—}\lambda\rightarrow E_2$ **and** $E'_1 \text{—}\lambda^-\rightarrow E'_2$ **implies** $E_1|E'_1 \text{—}\tau\rightarrow E_2|E'_2$

Rec) $T[\text{rec x. } T/x] \text{—}\mu\rightarrow E$ **implies** $\text{rec x. } T \text{—}\mu\rightarrow E$. ♦

This relation completely specifies the CCS operational semantics which, given an agent, determines the actions (and sequences of actions) it may perform, and the new agents which are obtained as results. We aim at defining a Petri net which, in addition to the above information, also carries information about the causal dependencies and possible concurrency among the actions performed by every CCS agent.

We now introduce the class of Petri Nets which will be used in the rest of the paper to describe CCS. We call this class Augmented Condition/Event (A-C/E) Systems since it is obtained by straightforwardly extending C/E Systems [19]. Indeed, A-C/E Systems are obtained by slightly relaxing the condition for enabling events and by removing the simplicity condition.

Although the simplicity requirement could be kept, we drop it since we want to naturally represent situations of pure nondeterminism, such as that expressed in CCS by agent $\alpha E + \beta E$, which may evolve to E in two ways, firing an event either observed as α or as β.

The actual extension to C/E Systems consists in relaxing the conditions under which a transition may occur. This is mainly because we want to consider as enabled an event the pre- and postset of which are not disjoint. Indeed, loops arise when dealing with CCS agents involving recursion, e.g., rec x. αx.

Definition 2.5.

A **net** is a triple <B, E; F>, where

- B is the set of **conditions**;
- E is the set of **events**;
- $B \cap E = \emptyset$;
- $F \subseteq (B \times E) \cup (E \times B)$ is the **flow relation**. ♦

Definition 2.6.

Given a net N = <B, E; F>, let $x,y \in B \cup E$,

- $\cdot x$ denotes $\{y \mid yFx\}$, called **preset**, and $x\cdot$ denotes $\{y \mid xFy\}$, called **postset**;
- x is **isolated** if $\cdot x \cup x\cdot = \emptyset$;
- a subset $c \subseteq B$ is called **case**. ♦

Definition 2.7.

Given a net N = <B, E; F> and a case c,

- an event e is **a-enabled** by c if and only if $\cdot e \subseteq c$ and $e\cdot \subseteq \cdot e \cup B\text{-}c$.

Given $c_1, c_2 \subseteq B$ and $G \subseteq E$, the **a-step** $c_1 [[G\rangle c_2$, is defined if

- $\forall e \in G$, e is a-enabled by c_1;
- $\forall e_1, e_2 \in G$, $e_1 \neq e_2$ implies $\cdot e_1 \cap \cdot e_2 = e_1\cdot \cap e_2\cdot = \emptyset$;
- $c_2 = (c_1 - \cdot G) \cup G\cdot$. ♦

Definition 2.8.

An **Augmented Condition/Event system** (A-C/E system) is a quadruple Σ = <B, E; F, C>, where

- <B, E; F> is a net with no isolated element and $B \cup E \neq \emptyset$;
- $C \subseteq 2^B$ is an equivalence class of the reachability relation $R=(r \cup r^{-1})^*$, being $r \subseteq 2^B \times 2^B$, where $c_1\ r\ c_2$ if and only if $\exists G \subseteq E$ such that $c_1 [[G\rangle c_2$;
- $\forall e \in E$, $\exists c \in C$ such that e is a-enabled by c. ♦

Definition 2.9.

An A-C/E system <B, E; F, C> is **a-contact-free** if and only if $\forall e \in E$, $\forall c \in C$

- $\cdot e \subseteq c$ implies $e\cdot \subseteq \cdot e \cup B\text{-}c$;
- $e\cdot \subseteq c$ implies $\cdot e \subseteq e\cdot \cup B\text{-}c$. ♦

Definition 2.10.

Given an A-C/E system Σ = <B, E; F, C>, if $P = \{(c_1, G, c_2) \mid c_1 [[G\rangle c_2$ is an a-step of $\Sigma\}$ then the labelled graph $\Phi = (C, P)$ is called the **case graph** of Σ. ♦

The only differences between C/E and A-C/E Systems concern simplicity and event enabling: the former must be simple (formally, $\cdot x = \cdot y$ and $x\cdot = y\cdot$ imply $x=y$) while the latter may not be; enabling in C/E Systems requires pre- and postsets to be disjoint (formally, $\cdot e \subseteq c$ and $e\cdot \subseteq B\text{-}c$), a-enabling does not.

The following propositions and definition clarify and characterize a-contact-freeness.

Property 2.1.
Given an a-contact-free A-C/E system <B, E; F, C> and a case c in C, an event is a-enabled if and only if its preset is a subset of c. ♦

Definition 2.11.
A case c of a net N = <B, E; F> is **critical** if there exists an event e such that either
$\cdot e \subseteq c$ and $(c\text{-}\cdot e) \cap e\cdot \neq \emptyset$ or $e\cdot \subseteq c$ and $(c\text{-}e\cdot) \cap \cdot e \neq \emptyset$. ♦

Note that, in a critical case, the preset of an event holds which, if fired, would cause the "loss of some tokens". Actually, the case reached after firing would be the union of $c\text{-}\cdot e$ and $e\cdot$ and, if the two sets were intersecting, some token would be lost. Alternatively, the dual property might hold.

Proposition 2.2. (*a system is a-contact-free iff no token is lost in the token game*)
An A-C/E system Σ = <B, E; F, C> is a-contact-free if and only if no case in C is critical.
Proof. Immediate. ♦

It is easy to define a non contact-free C/E system which shows that the if-part of the above proposition is not true for standard C/E Systems. Indeed, the set of (contact-free) C/E Systems is a proper subset of (a-contact-free) A-C/E Systems, as shown by Fig. 2.1. below.

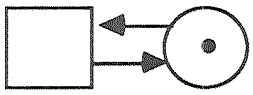

Fig. 2.1. A non contact-free A-C/E system which is a-contact-free.

3. A Partial Ordering Derivation Relation for CCS

In this section we present a new set of inference rules for CCS which relate parts of CCS agents, rather than whole global states. These rules are not only able to express the temporal dependencies between the actions performed by the agents but also make it possible to recover causal dependencies and the parallelism of CCS agents. In fact, we decompose these agents into sets of sequential processes, called grapes, and then we define the new relation between sets of grapes; the new transition relation, together with the actions an agent E may perform, also specifies which sequential processes of E perform these actions. The transitions have the form H' - K'—$\mu\rightarrow$ H", and their intuitive meaning is that set of grapes H' may become set H" by performing action μ. Set of grapes K' has only a technical meaning: it forbids deductions of incorrect transitions; it will be ignored in the construction of the A-C/E system mirroring CCS. The new axioms and inference rules are in direct correspondence with those of Section 2. Before defining the new transition relation we need to show how sequential processes are obtained from agents.

Definition 3.1. (*defining CCS sequential processes*)
A **grape** is a term defined by the following BNF-like grammar

G ::= NIL | μE | G\α | G[ϕ] | G + G | id|G | G|id | let x = T in G

where E, T, \α and [ϕ] have the standard CCS meaning.
We will use I, J, H, K (possibly indexed) to denote sets of grapes and g, g_1, ... to denote single grapes. ♦

Intuitively speaking, a grape represents a subagent of a CCS agent, together with its access path. The latter is used to take into account the context in which sequential processes operate. This information is crucial in many occasions. For example, it allows us to differentiate the behaviour of processes like (αNIL | α^-NIL)\$\alpha$ and (αNIL)\$\alpha$ | (α^-NIL)\$\alpha$. We have an operator on grapes for each CCS operator and we keep the same name for all the operators apart from that for parallel composition. This is replaced by two unary operators, lid and idl, which are tags recording that there are other processes that can perform actions concurrently with those of the given sequential process. We derive sets of grapes from CCS agents by using a ternary relation, decrel (E, I, J), which relates agents and sets of grapes. The second argument I represents the distributed state corresponding to agent E. The rôle of the third argument is purely technical: it allows agents to be recovered uniquely from sets of grapes and avoids wrong decompositions of them.

The decomposition relation is defined below. In this relation and in what follows we understand constructors as extended to operate on sets, e.g., $J\backslash\alpha=\{g\backslash\alpha \mid g\in J\}$. Moreover, everywhere in the paper, we will use the symbol $\cup$ to represent *disjoint* set union.

Definition 3.2. (*decomposing CCS agents into sequential processes*)
Relation decrel between CCS terms and two sets of grapes is the least relation satisfying the following axioms and inference rules.

Nil) decrel(NIL, {NIL}, {NIL})

Act) decrel(μE, {μE}, {μE})

Res) decrel(E, I, J) **implies** decrel(E\α, I\α, J\α)

Rel) decrel(E, I, J) **implies** decrel(E[ϕ], I[ϕ], J[ϕ])

Sum) decrel(E', I', I') **and** decrel(E", I", I") **implies** decrel(E'+E", I'+I", I'+I")

Com) decrel(E', I', J') **and** decrel(E", I", J") **implies** decrel(E'|E", I'|id ∪ id|I", J)
where J is either Ø or J'|id or id|J" or J'|id ∪ id|J"

Rec) decrel(T[rec x. T/x], I, I) **implies** decrel(rec x. T, let x=T in I, let x=T in I)

Ups) decrel(E', I' ∪ J', J') **and** decrel(E, I, I) **and** I'≠Ø
implies decrel(E', I' ∪ I+J', I+J')
and decrel(E', I' ∪ J'+I, J'+I)

Upr) decrel(E, I ∪ J[rec x. T/x], J[rec x. T/x]) **and** I≠Ø
implies decrel(E, I ∪ let x=T in J, let x=T in J) ♦

The decomposition goes inside the structure of agents and only stops when a process prefixed by an action or the NIL process are encountered, since these cannot be considered but atomic sequential processes. All the other rules allow a set of grapes I to be built whose correspondence with agent E is immediate except for rules Ups) and Upr). These two rules are needed to allow generation of those grapes which may come into existence when only parts of the global state of an agent perform an action. Here, we diverge substantially from the classical transition system approach since we want to give a distributed account of the transitions of agents and to assume *independent loci of control*. In fact, transition systems carry with them the assumption of a *centralised control*, and in our case this assumption forces *all* concurrent processes which occur in the *same* "+" or "rec" context to participate in the decision concerning the choice or the unwinding. An immediate consequence of distributed control is that subparts of the global states are left behind and are non-updated with respect to the choice or to the unwinding. So, Ups) and Upr) allow sets of grapes to be extended by "ornating" *proper* subsets of I with recursive definitions or alternative sets resulting from the

decomposition of other agents. It should be noted that rules Ups) and Upr) can be applied only after rule Com) with J equal to either Ø or J'|id or id|J" has been applied to differentiate the second and the third argument of decrel. Indeed, if we always have I = J then relation decrel can be seen as a total injective function from agents to sets of grapes which coincides with the decomposition function dec' proposed in [6]. There, it was also shown that given any set of grapes which can be generated by dec', it is always possible to recover the CCS agent which has generated it simply as the unique unifier of the grapes representing the sequential processes, when all the "id" atoms are seen as distinct variables. Actually, those sets I such that decrel(E, I, I) holds, represent *completely updated* states of CCS agents. However, as mentioned above, decrel also allows those states of CCS agents which are only partially updated to be generated.

Example 3.1 shows how different sets of grapes can be obtained by decomposing the same agent αNIL | βNIL. On the other hand, Example 3.2 shows a possible decomposition of an agent (((rec x. αx+βx) | rec x. αx+γx) | rec x. α⁻x)\α which will be used as a running example in the rest of the paper.

Example 3.1.

a) decrel (αNIL | βNIL, {αNIL|id, id|βNIL}, {αNIL|id, id|βNIL});

b) decrel (αNIL | βNIL, {αNIL|id, id|βNIL+NIL}, {id|βNIL+NIL});

c) decrel (αNIL | βNIL,{αNIL|id, let x=NIL in id|βNIL+NIL},
{let x=NIL in id|βNIL+NIL}).

Let us completely work out the proof of the second predicate:

1. decrel (αNIL, {αNIL}, {αNIL}) and decrel ((βNIL, {βNIL}, {βNIL}) by Act);
2. decrel (αNIL | βNIL, {αNIL|id, id|βNIL}, {id|βNIL}) by Com) and 1;
3. decrel (αNIL | βNIL, {αNIL|id, id|βNIL+NIL}, {id|βNIL+NIL}) by Ups) and 2.

♦

Example 3.2. Let agent E be (((rec x. αx+βx) | rec x. αx+γx) | rec x. α⁻x)\α. We have decrel(E, I, I), where

I = { (((let x=αx+βx in α(rec x.αx+βx)+βrec x.αx+βx)|id)|id)\α,
((id|let x=αx+γx in α(rec x.αx+γx)+γrec x.αx+γx)|id)\α,
(id|let x=α⁻x in α⁻rec x.α⁻x)\α }

♦

At this point the rôle played by the third operand of decrel can be explained more clearly. This operand acts as a controller of relation decrel and guides the decomposition process to obtain only those sets of grapes which exhibit the same behaviour. In fact, it makes possible to derive from

decrel(αNIL | βNIL, {αNIL |id, id| βNIL}, {id| βNIL})
that
decrel(αNIL | βNIL, {αNIL |id, id| βNIL + γNIL}, {id| βNIL + γNIL}) holds.
However, it is not possible to derive from the latter predicate
decrel(αNIL | βNIL, {αNIL |id + γNIL, id| βNIL + γNIL}, {id| βNIL + γNIL}).
This derivation would be wrong, since {αNIL |id + γNIL, id| βNIL + γNIL} has a behaviour different from that of αNIL | βNIL; indeed, it corresponds to αNIL | βNIL + γNIL.

The next definition precisely characterizes the class of sets of grapes which can be obtained by decomposing CCS agents. We call *regular* any set of grapes representing a distributed, possibly *not updated*, state of a CCS agent, and we call *complete* any set of grapes representing a *completely updated* state. Lemma 3.1 and Theorem 3.1 stress the relationships between the three arguments of the relation decrel.

Definition 3.3.
A set I of grapes is

- **regular** if and only if there exists a CCS agent E such that decrel(E, I, J);
- **complete** if and only if there exists a CCS agent E such that decrel(E, I, I).

Often, we will use I (possibly indexed) to denote a regular set of grapes. ♦

Lemma 3.1.
Let decrel(E, I, J), then

i) If I contains a grape g of the form G + G, then J consists of all the grapes in I of the same form.
ii) If I contains a grape g of the form let x=T in G, then J consists of all the grapes in I of the same form.
iii) If decrel(E, I, J) then I cannot contain a grape of the form G + G and a grape of the form let x=T in G at the same time.

Proof. **i)** All the grapes in J have the same operator at the top level (considering |id and id| as the same operator) and J is a subset of I. So, to prove the claim, it is sufficient to show that any grape g belonging to I having the form G + G must belong to J as well. Let us consider any proof of decrel(E, I, J). If, by absurdum, g does not belong to J, then the last rule used must have been either Ups) or Upr). In fact Sum) would imply I = J, and none of the remaining rules and axioms would unify with a set I containing even a single grape of the form G + G. But both rules Ups) and Upr) preserve the grapes not in J. Thus g belongs, by backward induction, to all the goals of the proof, which uses only Ups) and Upr). But this is false since the proof must use an axiom.
A similar proof applies to **ii)**.
Claim **iii)** follows trivially from i) and ii). ♦

The next theorem is particularly important since it guarantees the soundness of decrel in that to different agents are associated different sets of grapes, and in that it is not possible to find proper subsets of sets of grapes representing a distributed state which directly correspond to a CCS agent, i.e., no proper subset of a regular set is complete.

Theorem 3.1.

i) If decrel(E', I, J') and decrel(E", I, J") then E' = E".

ii) If decrel(E, I, J), then decrel(E', I', I') does not hold for every I' strictly contained in I.

Proof. Both properties are proved by induction on the length of the proof of I.
For **i)**, we have to prove that the binding for E in the last step is unique. Let us set as goal decrel(E, I, J) where I is the given set and E and J are variables. We have to see which pairs of rule heads unify, once the second argument has been instantiated to I. It can be seen immediately that all clauses except for Ups) and Upr) do not pairwise unify. This includes axioms Nil) and Act) which provide the basis of the induction. Com) might be applied in three ways, but in all cases the subgoals will be the same, with the same binding for E and I. Thus if I does not contain grapes of the form G + G or let x=T in G, precisely one rule will match. But set I cannot contain grapes of both kind because of Lemma 3.1.iii). If G contains a grape of the form let x=T in G then J is known, by Lemma 3.1.ii). If J = I, then only Rec) applies, else only Upr) applies. If G contains a grape of the form G + G, then again J is known, by Lemma 3.1.i). If and only if J = I then Sum) applies. Otherwise, let us decompose the cartesian product J = J' + J" into its components J' and J". If J' is complete then the first implication of Ups) applies and if J" is complete then the second implication of Ups) applies. It is always possible to tell which one is complete, because the other set will be smaller than a regular set with a shorter proof than I, and thus is not complete by the inductive hypothesis for ii).
Proving inductively property **ii)** for I is trivial. The basis is obvious, since in Nil) and Act) the I argument is a singleton, and the empty set is not complete. Furthermore, given any proper subset I' of I, only the same goal reduction of I can be used, if any exists at all; this generates a subgoal which has also a smaller I, and it is possible to use the induction hypothesis for proving the claim. The only apparent exception is when the last rule for I was Ups) or Upr), and in I' all the grapes with last constructor + or let-in respectively have been removed. In this case I' would be also a subset of the regular set of grapes which I reduces to, and the inductive hypothesis can again be applied. ♦

Theorem 3.2.

For any CCS agent E there exists always a set of grapes I such that decrel(E, I, J').

Proof. Structural induction can be used to show that by using the first seven rules of Definition 3.2 it is always possible to find the wanted I. The base cases are NIL and αE. The proof terminates since rule Rec) unfolds E once for every rec-context. This relies on the fact that E is guarded. Notice that no rule for x is needed. ♦

Below we define our new partial ordering derivation relation which is given in the SOS style.

Definition 3.4. (*partial ordering derivation relation*)
The **partial ordering derivation relation** $H' - K' \xrightarrow{\mu} H''$ is defined as the least relation satisfying the following inference rules:

act) $\mathrm{decrel}(E, I, I)$ **implies** $\{\mu E\} - \{\mu E\} \xrightarrow{\mu} I$

res) $H' - K' \xrightarrow{\mu} H''$ **and** $\mu \notin \{\alpha, \alpha^-\}$ **implies** $H' \backslash \alpha - K' \backslash \alpha \xrightarrow{\mu} H'' \backslash \alpha$

rel) $H' - K' \xrightarrow{\mu} H''$ **implies** $H'[\phi] - K'[\phi] \xrightarrow{\mu} H''[\phi]$

sum) $H \cup K' - K' \xrightarrow{\mu} H''$ **and** $\mathrm{decrel}(E, I, I)$ **implies** $H \cup K' + I - K' + I \xrightarrow{\mu} H''$
and $H \cup I + K' - I + K' \xrightarrow{\mu} H''$

com) $H' - K' \xrightarrow{\mu} H''$ **implies** $H'|id - X \xrightarrow{\mu} H''|id$ **where** $X = \emptyset$ or $K'|id$
and $id|H' - X \xrightarrow{\mu} id|H''$ **where** $X = \emptyset$ or $id|K'$

$H_1' - K_1' \xrightarrow{\lambda} H_1''$ **and** $H_2' - K_2' \xrightarrow{\lambda^-} H_2''$
implies $H_1'|id \cup id|H_2' - X \xrightarrow{\tau} H_1''|id \cup id|H_2''$
where $X = \emptyset$ or $K_1'|id$ or $id|K_2'$ or $K_1'|id \cup id|K_2'$

rec) $H \cup K'[\mathrm{rec}\ x.T/x] - K'[\mathrm{rec}\ x.\ T/x] \xrightarrow{\mu} H''$
implies $H \cup \mathrm{let}\ x=T\ \mathrm{in}\ K' - \mathrm{let}\ x=T\ \mathrm{in}\ K' \xrightarrow{\mu} H''$. ♦

The basic meaning of $H' - K' \xrightarrow{\mu} H''$ is that set of grapes H' evolves to set H'' by performing action μ. As in the definition of decrel, the extra argument K', with $K' \subseteq H'$, forbids incorrect derivations. Set K' represents the most out-of-date grapes within H'. No extra argument is required on the right member of the transition, since the rewritten grapes are all in the most updated form.

We now comment on our rules in detail. Separation of concerns suggests to ignore in a first discussion the second argument K' and to concentrate on the evolution of H' to H'' via μ. In rule **act**), a single grape is rewritten as a set of grapes, since the occurrence of the action makes the (possible) parallelism of E explicit. The rules **res**) and **rel**) and the first two rules for **com**)

simply state that if a set of grapes H' can be rewritten as H" via μ, then we can combine the access paths of the grapes in both sets with either path constructors .\α, .[ϕ], .lid or idl., and still obtain a derivation, labelled, say, by μ'. Clearly, when dealing with restriction we have that μ' is μ, but the inference is possible only if $\mu \notin \{\alpha, \alpha^-\}$; in **rel**) μ' is $\phi(\mu)$ and in the first two rules of **com**) μ' is simply μ. The second rule for **com**) is the synchronization rule.

To comment **sum**), let us first consider the case H = Ø. If so, before enabling K' to perform its action, we must make sure that this alternative has not been discarded by the occurrence of actions of I. Certainly, we must avoid situations where, given two pairs of non intersecting sets of grapes $J'_1, J'_2 \subseteq J'$ and $J''_1, J''_2 \subseteq J''$, the two sets $J'_1 + J''_1$ and $J'_2 + J''_2$ (both contained in the sum J' + J") perform successive actions via both the right and the left operands of +. For this reason, a set of grapes is enabled to perform an action only if it is alternative to a *complete* set of grapes. When H ≠ Ø a similar reasoning applies; we only have to consider that the transition is the result of a communication between H and K' and that the presence of alternative I does not influence the overall outcome.

The rule for **rec**) determines the transitions of a recursively defined agent by considering the transitions of the set of grapes obtained by unwinding the relevant part of the recursively defined grapes. Again we have that if H = Ø and K'[rec x. T/x]) may evolve to H" via a particular action, then also grape let x = T in K' may evolve, via the same action, to H"; and if H ≠ Ø considerations similar to **sum**) apply.

Let us now take into account argument K', stressing again that it contains the most out-of-date grapes of H' which perform the action. As expected, in rule **act**) we have K' = H'. In rule **sum**) the claim is obvious when H = Ø; otherwise exactly K' + I (or I + K') is given as second argument of the consequent of the rule. Quite a similar reasoning applies to **rec**). In the first rule of **com**) we infer from a derivation involving set H' a derivation concerning this set when plugged into a | context. In this case, the most out-of-date grapes belong either to H' or to the context or to both; hence, we have that the second argument of the consequent is K'|id or empty or K'|id respectively. Symmetrically, in the second rule of **com**). In the third rule the most out-of-date grapes may belong to either set H' or H" or both, and the second argument in the consequent is accordingly set. An inductive reasoning vindicates the rôle of K' in the partial ordering derivation relation. An example of a derivation follows which is not deducible by using K' as a controller and is deducible when K' is ignored.

$\{\alpha|id + \beta NIL, id|\alpha^- + \gamma NIL\} - K' \xrightarrow{\tau} \{NIL|id, id|NIL\}$.

Further comments on this issue can be found in Section 5.

We can now give additional motivations for our decomposition relation by showing how the direct correspondence between agents and sets of grapes resulting from a rewriting is lost whenever a pair of parallel processes is put either in a sum or in a recursion context.

As an example, consider again agent

$E = \{\alpha NIL \mid \beta NIL + \gamma NIL\}$

of Figure 1.1; when decomposed, E gives rise to

$\{\alpha NIL|id + \gamma NIL\} \cup \{id|\beta NIL + \gamma NIL\}$

and when action α is performed the global state moves to

$\{NIL|id\} \cup \{id|\beta NIL + \gamma NIL\}$.

This set does not directly correspond to any CCS agent although it will always exhibit the same behaviour as the set

$\{NIL|id\} \cup \{id|\beta NIL\}$

which corresponds to agent

$NIL|\beta NIL$.

In fact, when using Milner's derivation relation to perform action α the choice of process γNIL is discarded by *both* components αNIL and βNIL. In our distributed relation, instead, we discard γNIL *only* from the grape which has actually contributed to action α. The same happens when E is put into a rec context such as rec x. $\alpha NIL|\beta NIL$. In fact, when dealing with a recursive definition we unwind *only* the set of grapes which performs an action whereas Milner always unwinds the *whole* agent.

It is now necessary to extend the derivation relation to regular sets of grapes (notation $\succ\!\!\rightarrow\mu\rightarrow$). This extension is crucial for proving that our new semantics meets criteria i) and ii) discussed in the introduction.

Definition 3.5.

seq) $H' \cup I \succ\!\!\rightarrow\mu\rightarrow H'' \cup I$ **if and only if** $decrel(E, H' \cup I, J')$ **and** $H' - K' \rightarrow\mu\rightarrow H''$. ♦

The next theorem establishes the relationships between our distributed approach and Milner's. Claim i) guarantees that, given a derivation $H' - K' \rightarrow\mu\rightarrow H''$, H' is always a subset of a regular set of grapes, and thus there always exists an agent that originates a distributed state to which the above derivation can be applied. Claims ii) and iii) say that derivations are independent of those grapes which are concurrent with the rewritten ones, but inactive. This result should evidence the asynchrony of our derivation relation. In other words, we have defined a rewriting system, rather than a transition system. Claim i) provides the basis for proving that every event of the A-C/E system for CCS defined in the next section has a case enabling it; parts ii) and iii) are essential for showing contact-freeness of that system. Claim iv) shows that it is possible to derive the proofs of $I' \succ\!\!\rightarrow\mu\rightarrow I''$ and of $decrel(E'', I'', J')$ from the proofs of $decrel(E', I', J')$ and of $E' \rightarrow\mu\rightarrow E''$. Part v) shows that the converse also holds.

Theorem 3.3.

i) If H' - K' —μ→H" then there exists I such that H' ∪ I is regular;

ii) H' ∪ I regular and H' - K' —μ→H" implies that I and H" are disjoint and that I ∪ H" is regular;

iii) H" ∪ I regular and H' - K'—μ→H" implies that I and H' are disjoint and that I ∪ H' is regular;

iv) if decrel(E', I', J') and I' >—μ→ I" then there exists E" such that E'—μ→E" and decrel(E", I", J");

v) if decrel(E', I', J') and E' —μ→ E" then there exists I" such that I' >—μ→ I" and decrel(E", I", J").

Proof.

i) Given a proof of H' - K' —μ→ H", it is easy to derive a proof of decrel(E', H' ∪ I, J'). Rules Act), Res), Rel) and Com) immediately correspond to rules act), res), rel) and com). Rule Sum) corresponds to sum) if H' = Ø; otherwise Ups) corresponds to sum). Similarly for Rec), Upr) and rec).

In order to prove claims (ii-v) it suffices showing the connection between decrel, —μ→ between CCS agents, >—μ→ between regular sets of grapes, and —μ→ between sets of grapes. To this aim we introduce a new relation encompassing all of them which we denote by [-μ-> (E' - I' - J' - H' - K' [-μ-> E" - I" - J" - H").

Act.

decrel(E, I, I) **implies** μE - {μE} - {μE} - {μE} - {μE} [-μ-> E - I - I - I

Res.

E' - I' - J' - H' - K' [-μ-> E" - I" - J" - H" **and** μ ∉ {α, α-}

implies E'\α - I'\α - J'\α - H'\α - K'\α [-μ-> E"\α - I"\α - J"\α - H"\α

Rel.

E' - I' - J' - H' - K' [-μ-> E" - I" - J" - H"

implies E'[ϕ] - I'[ϕ] - J'[ϕ] - H'[ϕ] - K'[ϕ] [-μ-> E"[ϕ] - I"[ϕ] - J"[ϕ] - H"[ϕ]

Sum.

E' - I'∪H' - I'∪H' - H' - H' [-μ-> E" - I'∪H" - I' - H" **and** dcrl(E, I, I)

implies E'+E - I'+I∪H'+I - I'+I∪H'+I - H'+I - H'+I [-μ-> E" - I'+I∪H" - I'+I - H"

and E+E' - I+I'∪I+H' - I+I'∪I+H' - I+H' - I+H' [-μ-> E" - I+I'∪H" - I+I' - H"

Com.

E' - I' - J' - H' - K' [-μ-> E" - I" - J" - H" **and** decrel(E, I, J)

implies E'|E - I'|id ∪ id|I - X' - H'|id - Y' [-μ-> E"|E - I"|id ∪ id|I - X" - H"|id

where X', Y' and X" are either all Ø; or J'|id, K'|id and J"|id;
or id|J, Ø and id|J; or J'|id ∪ id|J, K'|id and J"|id ∪ id|J

and

E|E' - id|I' ∪ I|id - X' - id|H' - Y' [-μ-> E|E" - id|I" ∪ I|id - K" - id|H"

where X', Y' and X" are either all Ø; or id|J', id|K' and id|J";
or J|id, Ø and id|J; or id|J' ∪ J|id, id|K' and id|J" ∪ J|id

E_1' - I_1' - J_1' - H_1'- K_1' [-λ-> $E_1"$ - $I_1"$ - $J_1"$ - $H_1"$ **and**

E_2' - I_2' - J_2' - H_2'- K_2' [-λ^{-}-> $E_2"$ - $I_2"$ - $J_2"$ - $H_2"$

implies $E_1'|E_2'$ - $I_1'|id \cup id|I_2'$ - X' - $H_1'|id \cup id|H_2'$ - Y' [-τ->

$E_1"|E_2"$ - $I_1"|id \cup id|I_2"$ - $J_1"|id \cup id|J_2"$ - $H_1"|id \cup id|H_2"$

where X', Y' and X" are either all Ø; or $J_1'|id$, $K_1'|id$ and $J_1"|id$;
or $id|J_2'$, $id|K_2'$ and $id|J_2"$; or $J_1'|id \cup id|J_2'$, $K_1'|id \cup id|K_2'$ and $J_1"|id \cup id|J_2"$

Rec.

T'[rec x. T'/x] - I'[rec x. T'/x] ∪ H'[rec x. T'/x] - I'[recx. T'/x] ∪ H'[rec x. T'/x] -
H'[rec x. T'/x] - H'[rec x. T'/x]

[-μ-> E" - I'[rec x. T'/x]∪H" - I'[rec x.T'/x] - H"

implies

rec x.T' - let x=T' in I'∪let x=T' in H' - let x=T' in I'∪let x=T' in H' - let x=T' in H'
- let x=T' in H'

[-μ-> E" - let x=T' in I'∪H" - let x=T' in I' - H"

Ups.

E' - I'∪J'∪K'∪H' - J'∪K' - K'∪H' - K' [-μ-> E" - I'∪J'∪H" - J' - H"

and decrel(E, I, I) **and** I'∪H' ≠ Ø

implies

E' - I'∪J'+I∪K'+I∪H' - J'+I∪K'+I - K'+I∪H' - K'+I [-μ->E" - I'∪J'+I∪H" - J'+I - H"

and

E' - I'∪I+J'∪I+K'∪H' - I+J'∪I+K' - I+K'∪H' - I+K' [-μ-> E" - I'∪I+J'∪H" - I+J' - H"

Upr.

E' - I'∪J'[rec x. T/x]∪K'[rec x. T/x]∪ H' - J'[rec x. T/x]∪K'[rec x. T/x] - K'[rec x. T/x]∪H' - K'[rec x. T/x]

[-μ-> E" - I'∪J'[rec x. T/x]∪H" - J'[rec x. T/x] - H" **and** I'∪H' ≠ Ø

implies

E' - I'∪ let x=T in J'∪let x=T in K'∪H' - let x=T in J'∪let x=T in K' - let x=T in K'∪H' - let x=T in K'

[-μ-> E" - I'∪let x=T in J'∪H" - let x=T in J' - H"

By examining the rules one by one, it is easy to see that from proofs of decrel(E', H' ∪ I, J') and of H' - K'—μ→H" it is possible to derive a proof of

E' - H' ∪ I - J' - H' - K' [-μ-> E" - H" ∪ I - J" - H".

It is then possible to see that I and H" are disjoint. Furthermore, from this proof it is possible to derive proofs of E' —μ→ E" and of decrel(E", H" ∪ I, J"). Similarly, from the proofs of decrel(E", H" ∪ I, J") and of H'—μ→H" it is possible to derive a proof of E' - H' ∪ I - J' - H' - K' [-μ-> E" - H" ∪ I - J" - H". It is then possible to see that I and H' are disjoint. From this proof it is possible to derive a proof of decrel(E', H' ∪ I, J'). Finally, from the proofs of decrel(E', I', J') and of E' —μ→ E" it is possible to derive a proof of E' - I' - J' - H' - K' [-μ-> E" - I" - J" - H"; and from this, in turn, proofs of I' >—μ→I" and of decrel(E", I", J").

Let us look to the rules for E' - I' - J' - H' - K' [-μ-> E" - I" - J" - H" a little more in detail. The meanings of the arguments are as follows. The agent E' and alternatively the grapes I' form the initial state of the transition. Similarly E" and I" represent the final state. Grapes J' and J" are the most out-of-date grapes of I' and I" respectively. Grapes H' and H" are those grapes of I' and I" respectively which are active in the transition: the remaining grapes stay idle. Finally, K' are the most out-of-date grapes of I' participating in the transition. Thus the following conditions hold:

- J' ⊆ I', J" ⊆ I";
- H' ⊆ I', H" ⊆ I";
- J'∩ H' = K', J" ∩ H" = Ø.

We now comment in detail Ups., one of the most complicate rules given above. The basic idea is that the regular set of grapes which is the second argument of the left member of [-μ-> can be partitioned in two ways: those most out-of-date and those not (a distinction relevant for the proofs for decrel of both left and right members of [-μ->), and those participating in the move and those not (relevant for the proofs of —μ→ between sets of grapes and —μ→ between CCS agents). Thus we partition the regular set in four parts (see also Figure 3.1):

I': the grapes which are neither most out-of-date nor participating in the move;

J': grapes most out-of-date and not participating;

K': grapes most out-of-date and participating;

H': grapes not most out-of-date and participating.

This explains the left member of the first premise. Obviously, I' and J' are also part of the right member of the first premise, since they are left unchanged by the move. The other premises contain the requirement that I be complete, the condition that I' ∪ H' must be not empty, otherwise rule Sum. should be used, as happens in decrel for rule Ups). Finally, in the consequences of the rule, only the most out-of-date grapes (namely, J' and K') are operated upon with the constructor "+I" or "I+" (see again Figure 3.1). This is consistent with Ups) of decrel.

♦

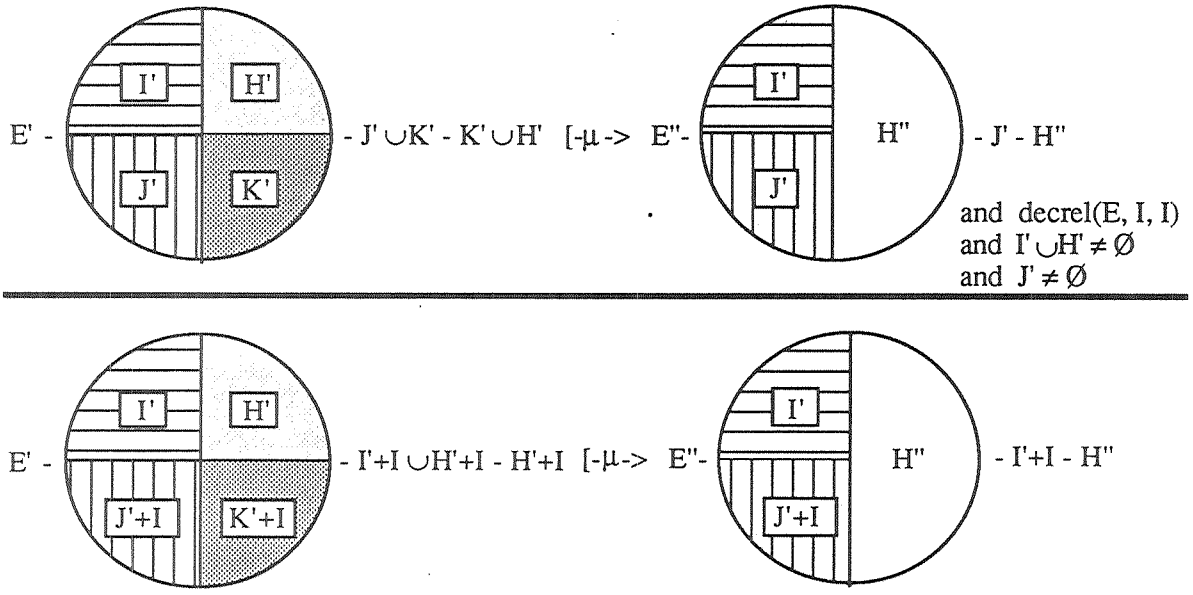

Fig. 3.1. A graphical representation of the first implication of rule **Ups.**

While the reverse implication of v) above is true (we practically have shown it while proving part iii)), the reverse implication of iv) does not hold. In fact, if we take

E' = NIL|αNIL and E" = NIL|NIL,

we have that

NIL|αNIL —α→ NIL|NIL and

decrel(NIL|NIL, {NIL|id,id|NIL+NIL}, {id|NIL+NIL}),

but there is no I' such that

decrel(NIL|αNIL, I', J') and I' >—α→{NIL|id, id|NIL+NIL}.

The next theorem states that relation decrel permits to associate to CCS agents only those sets of grapes which are *reachable* from *complete sets of grapes* via partial ordering derivations. Therefore, decrel precisely characterizes the distributed states which are reachable through executions of CCS agents.

Theorem 3.4.
Set I_2 is regular if and only if there exists an agent E and a sequence of actions $s = \mu_1\mu_2...\mu_n$, $n \geq 0$, such that decrel(E, I_1, I_1) and $I_1 \rightarrowtail \mu_1 \rightarrow I'_1 \rightarrowtail \mu_2 \rightarrow ... \ I'_{n-1} \rightarrowtail \mu_n \rightarrow I_2$ (denoted by $I_1 \rightarrowtail s \rightarrow I_2$).

Proof. The if-part is an immediate consequence of Theorem 3.3.ii).
For proving the only if-part, let us consider the relation decrel' defined by the same clauses Nil), Act), Res), Rel), Sum), and Rec) of decrel, but where

Com') decrel'(E', I', J') **and** decrel'(E", I", J")
implies decrel'(αE'|αE", I'|id ∪ id|I", ∅)
and decrel'(αE'|E", I'|id ∪ id|I", id|J")
and decrel'(E'|αE", I'|id ∪ id|I", J'|id)
and decrel'(E'|E", I'|id ∪ id|I", J'|id ∪ id|J")

Ups') decrel'(E', I' ∪ J', J') **and** decrel'(E, I, I) **and** I' ≠ ∅
implies decrel'(E+E', I' ∪ I+J', I+J')
and decrel'(E'+E, I' ∪ J'+I, J'+I)

Upr') decrel'(TT/x], I ∪ J[rec x. T/x], J[rec x. T/x]) **and** I ≠∅
implies decrel'(rec x. T, I ∪ let x=T in J, let x=T in J)

Given a proof of decrel(E_2, I_2, J_2) a similar proof that there exists an E_1 such that decrel'(E_1, I_2, J_2) can be immediately exhibited. We now show that if decrel(E_1, I_1, I_1), then $I_1 \rightarrowtail \alpha^n \rightarrow I_2$, $n \geq 0$. The proof is by induction on the length of the proof of decrel'(E_1, I_2, J_2). The basis is trivial: $E_1 = E_2$, $I_1 = I_2$, $n = 0$. The induction step is immediate if the last clause is anything except Com'), Ups') or Upr'): it is sufficient to apply the same clause to all the proofs of the transitions in the sequence $I_1 \rightarrowtail \alpha^n \rightarrow I_2$. The induction step is also easy if the fourth alternative of Com') is used (it is the same as the fourth alternative of Com)). To give a proof for the first alternative of Com'), let us assume that

decrel'(E_1', I_2', J_2'), decrel(E_1', I_1', I_1') and $I_1' \rightarrowtail \alpha^h \rightarrow I_2'$

decrel'(E_1", I_2", J_2"), decrel(E_1", I_1", I_1") and $I_1'' \rightarrowtail \alpha^k \rightarrow I_2''$

then we have

$\mathrm{decrel'}(\alpha E_1'|\alpha E_1'', I_2'|\mathrm{id} \cup \mathrm{id}|I_2'', \emptyset)$,

$\mathrm{decrel}(\alpha E_1'|\alpha E_1'', \{\alpha E_1'|\mathrm{id}\} \cup \{\mathrm{id}|\alpha E_1''\}, \emptyset)$, and

$\{\alpha E_1'|\mathrm{id}\} \cup \{\mathrm{id}|\alpha E_1''\} \rightarrowtail\!\!\!\!\!-\alpha\!\!\!\!\!\rightarrow \{\alpha E_1'|\mathrm{id}\} \cup \mathrm{id}|I_1''$

$\rightarrowtail\!\!\!\!\!-\alpha\!\!\!\!\!\rightarrow I_1'|\mathrm{id} \cup \mathrm{id}|I_1'' \rightarrowtail\!\!\!\!\!-\alpha^{h+k}\!\!\!\!\!\rightarrow I_2'|\mathrm{id} \cup \mathrm{id}|I_2''$

Similarly, for giving a proof for the second alternative of Com'), let us assume that

$\mathrm{decrel'}(E_1', I_2', J_2')$, $\mathrm{decrel}(E_1', I_1', I_1')$ and $I_1' \rightarrowtail\!\!\!\!\!-\alpha^{h}\!\!\!\!\!\rightarrow I_2'$

$\mathrm{decrel'}(E_1'', I_2'', J_2'')$, $\mathrm{decrel}(E_1'', I_1'', I_1'')$ and $I_1'' \rightarrowtail\!\!\!\!\!-\alpha^{k}\!\!\!\!\!\rightarrow I_2''$

then we have

$\mathrm{decrel'}(\alpha E_1'|E_1'', I_2'|\mathrm{id} \cup \mathrm{id}|I_2'', \mathrm{id}|J_2'')$,

$\mathrm{decrel}(\alpha E_1'|E_1'', \{\alpha E_1'|\mathrm{id}\} \cup \mathrm{id}|I_1'', \{\alpha E_1'|\mathrm{id}\} \cup \mathrm{id}|I_1'')$ and

$\{\alpha E_1'|\mathrm{id}\} \cup \mathrm{id}|I_1'' \rightarrowtail\!\!\!\!\!-\alpha\!\!\!\!\!\rightarrow I_1'|\mathrm{id} \cup \mathrm{id}|I_1'' \rightarrowtail\!\!\!\!\!-\alpha^{h+k}\!\!\!\!\!\rightarrow I_2'|\mathrm{id} \cup \mathrm{id}|I_2''$

A symmetric proof applies to the third alternative of Com').

For Ups'), we assume

$\mathrm{decrel'}(E_1', I_2' \cup J_2', J_2')$, $\mathrm{decrel'}(E, I, I)$ and $I_2' \neq \emptyset$; $\mathrm{decrel}(E_1', I_1', I_1')$ and

(*) $I_1' \rightarrowtail\!\!\!\!\!-\alpha^{n}\!\!\!\!\!\rightarrow I_2' \cup J_2'$.

We must have

$\mathrm{decrel'}(E+E_1', I_2' \cup I+J_2', I+J_2')$

$\mathrm{decrel}(E+E_1', I+I_1', I+I_1')$ and

$I+I_1' \rightarrowtail\!\!\!\!\!-\alpha^{n}\!\!\!\!\!\rightarrow I_2' \cup I+J_2'$

The first statement is immediate and the second descends from the fact that decrel'(E, I, I) if and only if decrel(E, I, I). To prove the last statement we will of course use (*), but we need to further assume the following inductive hypothesis:

$I_1' \rightarrowtail\!\!\!\!\!-\alpha^{n}\!\!\!\!\!\rightarrow I_2' \cup J_2'$ implies $I+I_1' \rightarrowtail\!\!\!\!\!-\alpha^{n}\!\!\!\!\!\rightarrow I_2' \cup I+J_2'$.

It is easy to see that this assertion goes through all clauses immediately except for Com'). The proof of the first alternative of Com') does not need the inductive hypothesis. We have to prove that

$\{\alpha E_1'|\mathrm{id}\} \cup \{\mathrm{id}|\alpha E_1''\} \rightarrowtail\!\!\!\!\!-\alpha\!\!\!\!\!\rightarrow I_1'|\mathrm{id} \cup \{\mathrm{id}|\alpha E_1''\}$

$\rightarrowtail\!\!\!\!\!-\alpha\!\!\!\!\!\rightarrow I_1'|\mathrm{id} \cup \mathrm{id}|I_1'' \rightarrowtail\!\!\!\!\!-\alpha^{h+k}\!\!\!\!\!\rightarrow I_2'|\mathrm{id} \cup J_2'|\mathrm{id} \cup \mathrm{id}|I_2'' \cup \mathrm{id}|J_2''$

implies

$I + \{\alpha E_1'|\mathrm{id}\} \cup I + \{\mathrm{id}|\alpha E_1''\} \rightarrowtail\!\!\!\!\!-\alpha\!\!\!\!\!\rightarrow I_1'|\mathrm{id} \cup I + \{\mathrm{id}|\alpha E_1''\}$

$\rightarrowtail\!\!\!\!\!-\alpha\!\!\!\!\!\rightarrow I_1'|\mathrm{id} \cup \mathrm{id}|I_1'' \rightarrowtail\!\!\!\!\!-\alpha^{h+k}\!\!\!\!\!\rightarrow I_2'|\mathrm{id} \cup J_2'|\mathrm{id} \cup \mathrm{id}|I_2'' \cup \mathrm{id}|J_2''$

which is obvious. In order to prove the assertion for the second alternative of Com'), let us

inductively assume that

(+) $I_1' \rightarrowtail\!\!\!\!\!\!\rightarrow^{\alpha^k} I_2' \cup J_2'$ implies $I'+I_1' \rightarrowtail\!\!\!\!\!\!\rightarrow^{\alpha^k} I_2' \cup I+J_2'$

and we have to prove that

$$\{\alpha E_1'|id\} \cup id|I_1'' \rightarrowtail\!\!\!\!\!\!\rightarrow^{\alpha} I_1'|id \cup id|I_1'' \rightarrowtail\!\!\!\!\!\!\rightarrow^{\alpha^{h+k}} I_2'|id \cup J_2'|id \cup id|I_2'' \cup id|J_2''$$

implies

$$I+\{\alpha E_1'|id\} \cup I+id|I_1'' \rightarrowtail\!\!\!\!\!\!\rightarrow^{\alpha} I_1'|id \cup I+id|I_1'' \rightarrowtail\!\!\!\!\!\!\rightarrow^{\alpha^{h+k}} I_2'|id \cup J_2'|id \cup id|I_2'' \cup I+id|J_2''$$

which is immediate from (+).

The third alternative of Com') is symmetric. For proving the assertion of the fourth alternative let us inductively assume that

(&)
$I_1' \rightarrowtail\!\!\!\!\!\!\rightarrow^{\alpha^h} I_2' \cup J_2'$ implies $I+I_1' \rightarrowtail\!\!\!\!\!\!\rightarrow^{\alpha^h} I_2' \cup I+J_2'$

$I_1'' \rightarrowtail\!\!\!\!\!\!\rightarrow^{\alpha^k} I_2'' \cup J_2''$ implies $I+I_1'' \rightarrowtail\!\!\!\!\!\!\rightarrow^{\alpha^k} I_2'' \cup I+J_2''$

and we have to prove that

$$I_1'|id \cup id|I_1'' \rightarrowtail\!\!\!\!\!\!\rightarrow^{\alpha^{h+k}} I_2'|id \cup J_2'|id \cup id|I_2'' \cup id|J_2''$$

implies

$$I+I_1'|id \cup I+id|I_1'' \rightarrowtail\!\!\!\!\!\!\rightarrow^{\alpha^{h+k}} I_2'|id \cup I+J_2'|id \cup id|I_2'' \cup I+id|J_2''$$

which is immediate from (&).

Similar arguments apply for the second implication of Ups') and for Upr'). ♦

The proof of the above theorem is constructive. In order to exemplify how to find the CCS agent which can be transformed via a sequence of derivations into a given regular set of grapes, let us follow the pattern of the above proof in the simple case of I_2 = {NIL|id, id|NIL + NIL}. We have

decrel(NIL|NIL, {NIL|id, id|NIL + NIL}, {id|NIL + NIL}).

Now, if NIL|NIL is "decomposed" with decrel', we get

decrel'(αNIL|NIL + NIL, {NIL|id, id|NIL + NIL}, {id|NIL + NIL}).

The required complete set of grapes can now be obtained by

decrel(αNIL|NIL+NIL, {αNIL|id+NIL, id|NIL+NIL}, {αNIL|id+NIL, id|NIL+NIL}).

In fact, we have that

{αNIL|id + NIL, id|NIL + NIL} $\rightarrowtail\!\!\!\!\!\!\rightarrow^{\alpha}$ {NIL|id, id|NIL + NIL}.

4. CCS as an Augmented C/E System

4.1. From CCS to an Augmented C/E System

In this section we build Σ_{CCS}, the non-simple, a-contact-free A-C/E system which behaves like CCS agents. We use as starting point the partial ordering transition relation defined in the previous section.

Hereto, we will use the following notation:

- $\mathcal{A}$ denotes the set of CCS agents;
- $\mathcal{T}$ denotes the set of transitions $E_1 \text{—}\mu\rightarrow E_2$ belonging to Milner's derivation relation;
- $\mathcal{G}$ denotes the set of all grapes;
- $\mathcal{R}$ denotes the set of regular subsets of $\mathcal{G}$;
- $\mathcal{D}$ denotes the set of derivations $H' \text{—}\mu\rightarrow H''$ such that $H' - K' \text{—}\mu\rightarrow H''$ belongs to the partial ordering derivation relation.

Definition 4.1. (*from CCS to the corresponding a-contact-free A-C/E system*)
Let $\Sigma_{CCS} = \langle B, E; F, C\rangle$, where

- $B = \mathcal{G}$;
- $E = \mathcal{D}$;
- g_1 F $(H' \text{—}\mu\rightarrow H'')$ and $(H' \text{—}\mu\rightarrow H'')$ F g_2,
  for all $g_1 \in H'$, $g_2 \in H''$, and $H' \text{—}\mu\rightarrow H'' \in E$;
- $C = \mathcal{R}$.

Property 4.1. (Σ_{CCS} *is an A-C/E system*)
i) C is an equivalence class of the reachability relation;
ii) Every event has a case in C which enables it.

Proof. i) The regular sets of grapes are closed with respect to forward and backward firings by Theorem 3.3.ii) and iii). They are connected with respect to the reachability relation since, given any pair of regular sets I_2' and I_2'', by Theorem 3.4 it is possible to find agents E_1' and E_1'' such that

decrel(E_1', I_1', I_1') and $I_1' \text{ >—s'}\rightarrow I_2'$
decrel(E_1'', I_1'', I_1'') and $I_1'' \text{ >—s''}\rightarrow I_2''$

where s and s' are sequences of actions; and thus if we consider the regular (and complete) set $\{\alpha E_1' + \alpha E_1''\}$ we have

$\{\alpha E_1' + \alpha E_1''\} \succ\!\!\!\rightarrow \alpha s' \rightarrow I_2'$ and $\{\alpha E_1' + \alpha E_1''\} \succ\!\!\!\rightarrow \alpha s'' \rightarrow I_2''$.

ii) Follows from Theorem 3.3.i). ♦

Property 4.2.

The A-C/E system Σ_{CCS} is a-contact-free.

Proof. Immediate from Theorem 3.3.ii) and iii). ♦

Example 4.1.

Let us consider the CCS agent of Example 3.2:

$E = (((\text{rec } x.\ \alpha x+\beta x) \mid \text{rec } x.\ \alpha x+\gamma x) \mid \text{rec } x.\ \alpha^- x)\backslash\alpha$.

Fig. 4.1 shows the relevant part of Σ_{CCS} corresponding to E, containing only the cases c such that decrel(E, I, I) holds and I r* c, the conditions in these cases, and the events enabled by them. Actually, it has only one case c_0, graphically represented by marking those conditions which hold in it.

$c_0 = \{b_0, b_1, b_2\}$, where

$b_0 = (((\text{let } x=\alpha x+\beta x \text{ in } \alpha(\text{rec } x.\alpha x+\beta x)+\beta \text{rec } x.\alpha x+\beta x)\mid id)\mid id)\backslash\alpha$;

$b_1 = ((id \mid \text{let } x=\alpha x+\gamma x \text{ in } \alpha(\text{rec } x.\alpha x+\gamma x)+\gamma \text{rec } x.\alpha x+\gamma x)\mid id)\backslash\alpha$;

$b_2 = (id \mid \text{let } x=\alpha^- x \text{ in } \alpha^- \text{rec } x.\alpha^- x)\backslash\alpha$.

There are the following four transitions

$e_0 : \{b_0, b_2\} \longrightarrow\!\!\!\!\tau\rightarrow \{b_0, b_2\}$; $e_1 : \{b_0\} \longrightarrow\!\!\!\!\beta\rightarrow \{b_0\}$;

$e_2 : \{b_1\} \longrightarrow\!\!\!\!\gamma\rightarrow \{b_1\}$; $e_3 : \{b_1, b_2\} \longrightarrow\!\!\!\!\tau\rightarrow \{b_1, b_2\}$;

and the relevant part of flow relation F is

$\cdot e_0 = e_0^\cdot = \{b_0, b_2\}$; $\cdot e_1 = e_1^\cdot = \{b_0\}$;

$\cdot e_2 = e_2^\cdot = \{b_1\}$; $\cdot e_3 = e_3^\cdot = \{b_1, b_2\}$. ♦

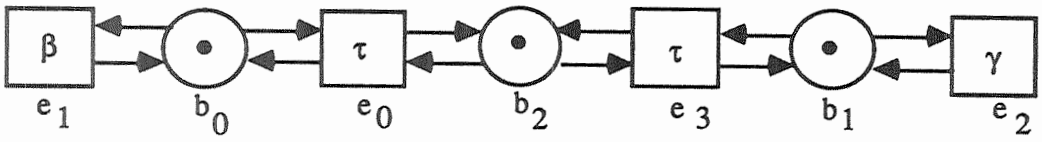

Fig. 4.1. The fragment of Σ_{CCS}, as constructed in Example 4.1.
The case, represented by marked conditions, corresponds to agent
(((rec x. αx+βx) | rec x. αx+γx) | rec x. α-x)\α decomposed in Example 3.2.
For the sake of clarity, we have labelled the events with the actions they contain.

In Figure 4.2. we see the case graph for the A-C/E system in Figure 4.1. .

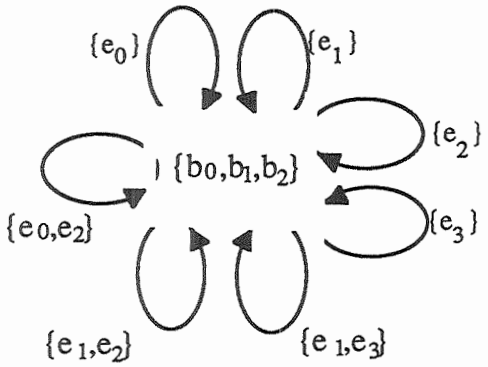

Fig. 4.2. The case graph for the A-C/E system in Figure 4.1. It is also the relevant part of the case graph of Σ_{CCS} for agent (((rec x. αx+βx) | rec x. αx+γx) | rec x. α-x)\α.
Events e_0, e_1, e_2 and e_3 contain the actions τ, β, γ, and τ, respectively.

The above construction is suggestive and straightforward, but violates the simplicity requirement for C/E Systems. Simplicity is felt as an important property within the theory of Petri Nets, being imposed by the extensionality requirement. We now sketch an alternative construction, slightly more cumbersome, that generates a system which is simple. Basically, it extends the non-simple A-C/E system obtained above by attaching a new condition to every event e (corresponding to the derivation H'—μ→H"). This condition occurs in both the pre- and the post-conditions of e itself, and is constructed as a pair <e, cond>, where cond is a tag which allows us to keep events and conditions disjoint. The A-C/E system obtained in this way, called S_{CCS}, is simple; furthermore, results similar to Properties 4.1 and 4.2 can be easily proved.

Fig. 4.3 below shows the fragment of S_{CCS} corresponding to Example 4.1.

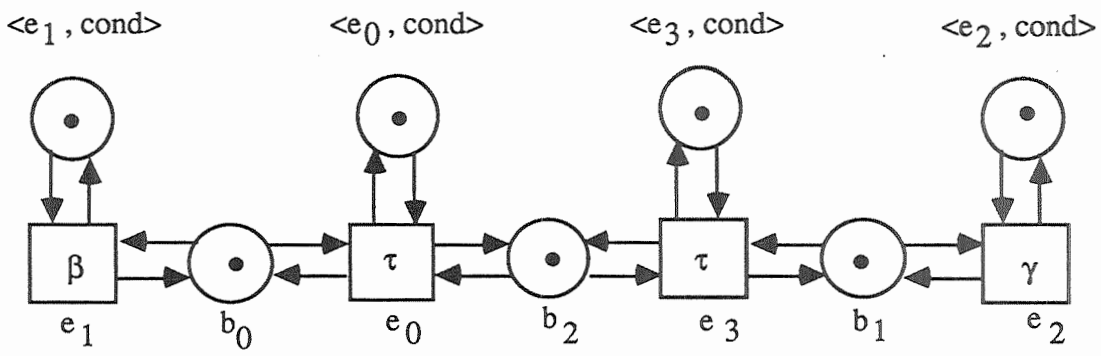

Fig. 4.3. The fragment of S_{CCS} for agent (((rec x. αx+βx) | rec x. αx+γx) | rec x. α-x)\α.

4.2. Adequacy of the Petri Net Semantics

In this section we prove that the distributed semantics we propose satisfies criteria i) and ii) discussed in the introduction. Since our framework is now more precise we can rephrase these criteria:

i) any semantics based on the original transition systems of CCS must be recoverable from Σ_{CCS}.

ii) Let c be a regular case of Σ_{CCS}, namely a set of grapes, corresponding to a CCS agent E. The multisets of actions contained in the sets of events concurrently enabled by c must coincide with the labels of the possible moves of E according to the multiset semantics of CCS.

In order to show that our semantics satisfies criterion i) we prove a stronger property: the original CCS transition system is retrievable from the A-C/E system we have built above. In fact, we assume to describe the interleaving behaviour of Σ_{CCS} by its interleaving abstract case graph, i.e., the case graph obtained from Σ_{CCS} case graph by ignoring those arcs which are labelled by more than one event, and by taking actions rather than events as labels. We will prove that there exists a transition-preserving surjective homomorphism from the interleaving abstract case graph and the original transition system for CCS, as defined in Section 2. The homomorphism guarantees that *isomorphic* labelled trees are obtained when unfolding two related nodes.

Definition 4.2. (*extracting actions from events*)
Let *abstr*: $E \rightarrow \Lambda \cup \{\tau\}$) be a function defined as $abstr(I_1 \text{—}\mu\rightarrow I_2) = \mu$. ♦

Definition 4.3. (*interleaving abstract case graph*)
Let P_{int} = $\{(c_1, abstr(e), c_2) \mid c_1\ [[\{e\}> c_2$ is a (singleton) a-step of $\Sigma_{CCS}\}$, then the labelled graph $\Phi_{int} = (C, P_{int})$ is the **interleaving abstract** case graph of Σ_{CCS}. ♦

Note that passing from Σ_{CCS} to Φ_{int} corresponds to an abstraction step, since more than one arc can be mapped together; additionally, Φ_{int} deletes arcs with parallelism. For instance, the arcs labelled by $\{e_0\}$ and $\{e_3\}$ of the case graph of Figure 4.2, both corresponding to derivations labelled by τ, are mapped into the single arc, labelled by τ, of the interleaving abstract case graph in Fig. 4.4.a); the arcs labelled by $\{e_0, e_2\}$, $\{e_1, e_2\}$ and $\{e_1, e_3\}$ are discarded.

Definition 4.4. (*CCS transition graph*)
The labelled graph $\Psi = (\mathcal{A}, \mathcal{T})$ is the **transition graph** of CCS. ♦

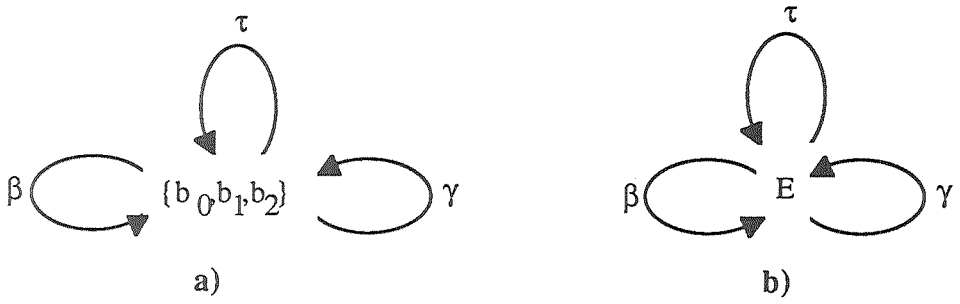

Fig. 4.4. The relevant part of the interleaving abstract case graph of Σ_{CCS} (in **a**), and of the transition graph of CCS (in **b**) relative to the agent E of Example 4.1.

Definition 4.5.
A labelled graph $\Gamma_1 = (N_1, A_1)$ is **transition-preserving (tp-) homomorphic** to a labelled graph $\Gamma_2 = (N_2, A_2)$ if and only if there exists a surjective mapping h: $N_1 \rightarrow N_2$ such that

i) $(n_1', \text{label}, n_1'') \in A_1$ implies $(h(n_1'), \text{label}, h(n_1'')) \in A_2$; and

ii) $\forall n_1'$ with $h(n_1') = n_2'$ and $(n_2', \text{label}, n_2'') \in A_2$,

$\exists n_1''$ with $h(n_1'') = n_2''$ such that $(n_1', \text{label}, n_1'') \in A_1$. ♦

Property 4.3.
Let $\Gamma_1 = (N_1, A_1)$ and $\Gamma_2 = (N_2, A_2)$ be two labelled graphs which are tp-homomorphic via a mapping h and n_1, n_2 be two nodes of Γ_1 and Γ_2 such that $n_2 = h(n_1)$; then the labelled trees obtained by unfolding Γ_1 and Γ_2 starting from n_1 and n_2 are isomorphic.

Theorem 4.1. (*interleaving Σ_{CCS} is CCS*)
The interleaving abstract case graph Φ_{int} of Σ_{CCS} is tp-homomorphic to the transition graph Ψ of CCS, via the mapping h_{ccs}: $\mathcal{R} \rightarrow \mathcal{A}$ defined as $h_{ccs}(I) = E$ iff decrel(E, I, J).

Proof. According to Theorem 3.1, decrel(E, I, J) is a function which maps regular sets of grapes to CCS agents when the second argument is given. Moreover, Theorem 3.2 guarantees that such function is surjective. Properties i) and ii) which the transition-preserving homomorphism has to satisfy are thus simply points iv) and v) of Theorem 3.3 respectively. ♦

We can now show that the synchronization tree obtained by unfolding the interleaving abstract case graph of Σ_{CCS} from any node n, corresponding to a regular set I of grapes, is isomorphic to the synchronization tree of the unique E such that decrel(E, I, I).

Corollary 4.1.
The tree obtained by unfolding the interleaving abstract case graph of Σ_{CCS} from any node n is isomorphic to the tree obtained by unfolding the transition graph of CCS from node $h_{ccs}(n)$.
♦

In Figure 4.4. we see the relevant parts of Φ_{int} (in a)) and of Ψ (in b)) for the agent E of Example 4.1: they happen to be isomorphic.

This corollary is obviously sufficient to prove that our construction meets criterion i). Additionally, it shows that the correspondence between the two models is at a very basic level since the synchronization trees of an agent and of the corresponding complete set of grapes are identical. As a consequence we have that whichever observation mechanism we define (bisimulation [14, 16], testing [9], ...), the correspondence between our net semantics and the interleaving one will be always preserved.

To prove criterion ii), we define a multiset semantics for CCS in order to be more precise about the parallelism expressed by a term. Another transition system is introduced which extends the original system in that all the interleaved transitions are kept, but also transitions labelled by multisets of concurrent actions are introduced in the spirit of SCCS [15] and Meije

[1]. We will use this new transition system, which we call *multiset transition system*, to prove that our Petri Net semantics captures all and only the possible concurrency which is expressed by a CCS term. Of course, we should first agree that the multiset transition system expresses all possible parallelism; however the rules for SCCS and Meije have stirred little controversy, and in fact it is difficult to conceive, as multiset semantics of CCS, anything different from what we propose. In order to prove criterion ii), we show that decrel defines a transition-preserving homomorphism also from the multiset transition system to the case graph of Σ_{CCS} labelled by actions rather than by events. Thus, as before, we have that there is a one-to-one correspondence between the trees obtained by unfolding the abstract case graph and CCS multiset transition system starting from nodes related via decrel.

Definition 4.6. (*extending synchronization algebra to multisets*)
Let M, M' and M'' be multisets of action (notation $\{\mu_1, \ldots, \mu_n\}$) and $\cup$ denote multiset union; the **multiset synchronization relation** is the least relation which satisfies:

Synch(M', M'', M' $\cup_m$ M'') and

Synch(M', M'', M) **implies** Synch(M' $\cup$ {λ}, M'' $\cup$ {λ^-}, M $\cup$ {τ}) ♦

Definition 4.7. (*multiset derivation relation*)
The *multiset derivation relation* $E_1 \text{—}M\rightarrow E_2$, where M is a multiset of CCS actions and ϕ is extended to operate on multisets, is defined as the least relation satisfying the following axiom and inference rules:

ACT) $\mu E \text{—}\{\mu\}\rightarrow E$

RES) $E_1 \text{—} M\rightarrow E_2$ **and** $\alpha,\alpha^- \notin M$ **implies** $E_1\backslash\alpha \text{—} M\rightarrow E_2\backslash\alpha$

REL) $E_1 \text{—} M\rightarrow E_2$ **implies** $E_1[\phi] \text{—}\phi(M)\rightarrow E_2[\phi]$

SUM) $E_1 \text{—} M\rightarrow E_2$ **implies** $E_1+E \text{—} M\rightarrow E_2$ **and** $E+E_1 \text{—} M\rightarrow E_2$

COM) $E_1 \text{—} M\rightarrow E_2$ **implies** $E_1|E \text{—} M\rightarrow E_2|E$ **and** $E|E_1 \text{—} M\rightarrow E|E_2$

$E_1 \text{—}M'\rightarrow E_2$ **and** $E'_1 \text{—}M''\rightarrow E'_2$ **and** Synch(M', M'', M)

imply $E_1|E'_1 \text{—} M\rightarrow E_2|E'_2$

REC) $T[\text{rec x. T/x}] \text{—} M\rightarrow E$ **implies** rec x. T $\text{—} M\rightarrow$ E. ♦

This multiset operational semantics is very similar to that given for Meije in [1], although a richer structure than multisets is assumed there (actually a commutative monoid), and the language is slightly different. An alternative way of seeing this semantics is to consider it as a generalization of the SCCS semantics [15] in which the restriction on the synchronous execution of parallel actions is removed, so that actions can be executed asynchronously.

We now relate the operational semantics based on Σ_{CCS} to the multiset operational semantics. We will sometimes refer to CCS equipped with the multiset semantics as M-CCS.

Definition 4.8. (*abstract case graph*)
Let P_{abstr} = {(c_1, {*abstr*(e_1), ..., *abstr*(e_n)}, c_2) | c_1 [[{e_1, ..., e_n}> c_2 is an a-step of Σ_{CCS}}, then the graph Φ_{abstr} = (C, P_{abstr}) is the **abstract** case graph of the A-C/E system Σ_{CCS}. ♦

Again, passing from Σ_{CCS} to Φ_{abstr} corresponds to an abstraction step. For instance, in Fig. 4.5.a) we see the abstract case graph for the case graph in Fig. 4.2. Both arcs labelled by {e_0} and {e_3} are mapped on the same arc labelled by {τ}.

Definition 4.9. (*CCS transition graph*)
The graph $\Theta = (\mathcal{A}, \mathcal{M})$ is the **multiset transition graph** of M-CCS where $\mathcal{M}$ denotes the set of transitions E_1—M→E_2 belonging to the multiset derivation relation. ♦

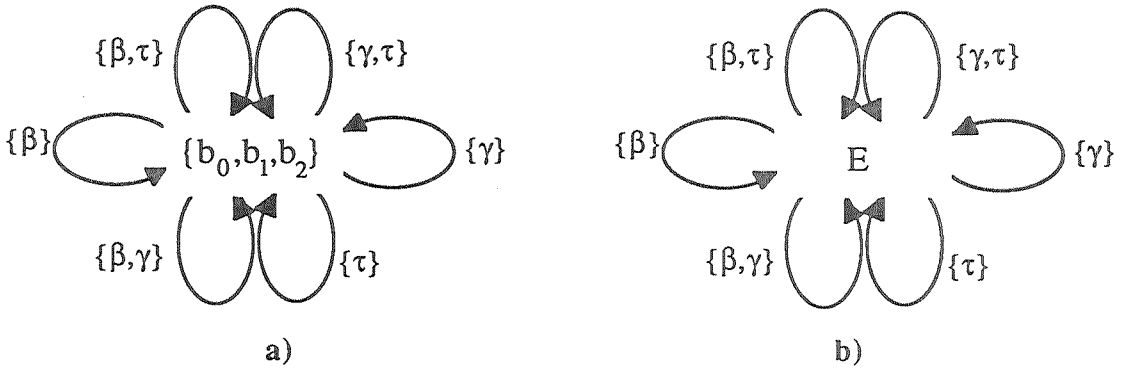

Fig. 4.5. The relevant part of the abstract case graph of Σ_{CCS} (in a), and of the multiset transition graph of CCS (in b) relative to the agent E of Example 4.1.

Theorem 4.2. (*abstract Σ_{CCS} is M-CCS*)
The abstract case graph Φ_{abstr} of Σ_{CCS} is tp-homomorphic to the multiset transition graph Θ of M-CCS via the mapping h_{CCS}: $\mathcal{R} \rightarrow \mathcal{A}$ defined as h_{CCS} (I) = E iff decrel(E, I, J).

Proof. A reasoning similar to that of Theorem 4.1 guarantees that homomorphism h_{CCS} is well-defined. The proof of properties i) and ii) of Definition 4.5 is different since the two transition systems have now larger sets of arcs. Let us prove ii) first. We start from I' to obtain the unique E' such that decrel(E', I', J'). Let us now assume that we have a proof of E'—{μ_1,...,μ_n}→ E". For every element μ_i in the multiset it is easy to extract a proof that E'—{μ_i}→E_i". But it can be seen immediately that the same proof holds for Milner's system: E'—μ_i→E_i". Using Theorem 3.1, it is now possible to derive, from each proof, a proof that H_i'—μ_i→H_i". Furthermore, it is possible to see that H_i' and H_j', H_i" and H_j" are disjoint if

$i \neq j$. Thus, all such transitions can fire at the same time, and we will have in Θ the arc (I', M, I''), with decrel(E", I", J"), $I' = (\cup_i H_i') \cup I$ and $I'' = (\cup_i H_i'') \cup I$, where I contains the grapes which do not participate to any of the transitions.
Claim i) is proved by reversing the argument. ♦

Corollary 4.2.
The tree obtained by unfolding the abstract case graph of Σ_{CCS} from any node n is isomorphic to the tree obtained by unfolding the transition graph of M-CCS from node $h_{CCS}(n)$. ♦

In Figure 4.5. we see the relevant parts of Φ_{abstr} (in a)) and of Θ (in b)) for the agent E of Example 4.1: they happen to be isomorphic.

5. Related Work

There have been many attempts to use Petri Nets to describe CCS without value passing. In many cases, however, either proper subsets of CCS have been considered or the interleaving semantics of the resulting net has not been the standard CCS one.

De Cindio et al. [3] map CCS into a subclass of Petri Nets (Superposed Automata Nets) whose elements are systems composed of interacting sequential automata. They restrict CCS syntax to avoid generation of unboundedly many agents in parallel via a recursive definition. For example, terms like rec x. αNIL | βx are not allowed.

Goltz and Mycroft [12] give a denotational semantics of CCS in terms of Occurrence Nets and an operational semantics in terms of Place/Transition Nets. In the former case, since possibly infinite Occurrence Nets are composed, operationality is lost. In the latter, only a CCS subcalculus which does not contain the restriction operator is considered. Moreover, the semantics they give is not in complete agreement with the original interleaving operational semantics; in particular, it does not respect all the causal dependencies between the actions performed by agents. See [6] for an example of this anomaly and for a discussion on its causes.

Winskel [20] also proposes a partial ordering denotational semantics for CCS which is based on Event Structures, a domain very close to Occurrence Nets and thus also extensional. Again, possibly infinite event structures must be operated upon, and thus this semantics cannot be considered as operational. Furthermore, Winskel simply claims that the interleaved semantics agrees with Milner's synchronization trees semantics without making any formal statement. In [21] and [22], a categorical interpretation of C/E Systems is proposed, and various operators are defined on them thus providing an adequate linguistic level. These operators are in close correspondence with those of CCS, and using them it should be easy to give a denotational semantics of the calculus. Nevertheless, the sum operator defined there is not in full agreement either with that defined in [14] or with operational intuitions about nondeterministic choice. In fact, a net corresponding to an agent $E_1 + E_2$ may perform actions from *both* alternative subprocesses. Winskel claims that his choice operator agrees with that of [14] if one considers only safe nets which have no event leading to the initial marking; and that all the nets obtained from CCS are of this kind. However, no complete definition of CCS semantics is given, and no proof of its relationships with the original interleaving semantics is mentioned.

An approach similar to that proposed in this paper has already been followed in a companion work by the authors [6] and by Olderog in [17] where a slightly different and simpler language is considered. However, the semantics proposed in [6] does not enjoy criterion ii). In fact, because of the choice of keeping an immediate correspondence between

sets of grapes and CCS terms and of maintaining Milner's assumption of a centralized control, it does not allow the intuitively possible concurrent execution of certain actions when a nondeterministic choice is made or a recursion is unwinded. For instance, agent $\alpha E_1|\beta E_2$ can concurrently perform actions α and β, whereas agent $(\alpha E_1|\beta E_2)+NIL$ cannot. The latter can only perform the interleavings of actions α and β and only afterwards it can concurrently perform the actions of E_1 and E_2.

Two new rules sum') and rec') for sum and recursion were introduced in the Section "Open problems" of [6], and proved unsatisfactory. The following amended version has been considered by the authors as an alternative to the four-argument derivation relation of Definition 3.4:

act")	decrel(E, I, I)	**implies**	$\{\mu E\} \sim\mu\rightarrow I$
res")	$H_1 \sim\mu\rightarrow H_2$ **and** $\mu \notin \{\alpha,\alpha^-\}$	**implies**	$H_1\backslash\alpha \sim\mu\rightarrow H_2\backslash\alpha$
rel")	$H_1 \sim\mu\rightarrow H_2$	**implies**	$H_1[\phi] \sim\phi(\mu)\rightarrow H_2[\phi]$
sum")	$H_1\cup H_3 \sim\mu\rightarrow H_2$ **and** decrel(E, I, I)	**implies**	$H_1\cup (H_3+I) \sim\mu\rightarrow H_2$
		and	$H_1\cup (I+H_3) \sim\mu\rightarrow H_2$
com")	$H_1 \sim\mu\rightarrow H_2$	**implies**	$H_1|id \sim\mu\rightarrow H_2|id$
		and	$id|H_1 \sim\mu\rightarrow id|H_2$
	$H_1 \sim\lambda\rightarrow H_2$ **and** $H'_1 \sim\lambda^-\rightarrow H'_2$	**implies**	$H_1|id\cup id|H'_1 \sim\tau\rightarrow H_2|id \cup id|H'_2$
rec")	$H_1\cup H_3[\text{rec x. T/x}]) \sim\mu\rightarrow H_2$	**implies**	$H_1\cup$ let x = T in $H_3 \sim\mu\rightarrow H_2$.

Relation $H_1 \sim\mu\rightarrow H_2$ does not coincide with the projection of relation H_1 - K $—\mu\rightarrow H_2$ defined in Section 3. A counterexample is as follows:

$$\{\alpha|id + \beta NIL, id|\alpha^- + \gamma NIL\} \sim\tau\rightarrow \{NIL|id, id|NIL\} \quad (*)$$

but not

$$\{\alpha|id + \beta NIL, id|\alpha^- + \gamma NIL\} \text{ - } K' —\tau\rightarrow \{NIL|id, id|NIL\}$$

The two relations can be proved to coincide when H_1 and H_2 in $H_1 \sim\mu\rightarrow H_2$ are always subsets of regular sets of grapes. As a matter of fact set $\{\alpha|id + \beta NIL, id|\alpha^- + \gamma NIL\}$ is not contained in any regular set of grapes. However, taking the above defined derivations $H_1 \sim\mu\rightarrow H_2$ as events would not have produced an A-C/E system, because the event corresponding to (*) would be enabled by no case.

In [17] and other manuscripts Olderog refines the approach of [6] to overcome its problems about sum and rec, and independently proposes a new set of derivation rules very similar to those defined above. These derivations are then used to associate a 1-safe P/T net to each CCS agent, rather than an A-C/E system to the whole CCS, à la Milner. In doing that, also

derivations which are not needed are generated and then thrown away by a suitable separate step in the construction. For example derivation (*) above is obtained and then discarded. Also Olderog puts forward criteria to assess the proposed semantics. However, criterion i) is stated by relying on a specific extensional semantics based on strong bisimulation rather than being independent from any extensional semantics. Its proof is rather intricate since equivalence class of nets must be considered and reachable markings, which correspond to our regular sets, must be given an additional syntactic structure. More involved and less general conditions are stated instead of criterion ii), but not formally proved.

The authors have tackled the problem of partial ordering semantics also in other papers. Particularly relevant to the work presented here are papers [4, 5, 8]. In [4] a partial ordering semantics for CCS is given in terms of Concurrent Histories, a sequential rewriting system previously developed by two of the authors. In [8], a new model for non-sequential computations, called Distributed Transition Systems, is introduced in which states are sets of processes and transitions specify which processes stay idle. There, both Condition/Event Systems and Place/Transition Nets, and CCS are modelled in terms of DTS. In [5] the model for CCS developed in [8] is used as the basis for defining new equivalence relations over the set of CCS agents, and for studying the relationships between interleaving and partial ordering observational semantics of the calculus. All the above mentioned papers use a decomposition function rather than relation decrel used in this paper, and run into problems which are similar to those of [6], in the sense that the maximal concurrency of every CCS agent is not captured. In [7] these problems are overcome by resorting to transitions with more complicated labels, which however do not lead directly to a Petri Net semantics, yet providing a fully concurrent operational semantics. The reason seems to be that in [7] we keep a *centralized locus* of control where decisions are taken when dealing with choice or recursion contexts.

In this paper, we let C/E Systems play the rôle that transition systems have in the original definition, and this allows us to contrast Petri Nets and CCS directly. As in the other papers, the technique used to define the transitions of CCS agents is based on Plotkin's SOS.

6. Conclusions

There have been many attempts to use Petri Nets to operationally describe CCS. To the best of our knowledge, however, in no case a fully satisfactory formulation has been provided and has been proved consistent with the ordinary interleaving semantics of CCS.

We have introduced Augmented Condition/Event Systems by slightly relaxing the classical (simplicity and) enabling conditions for Condition/Event Systems. One system from this class, called Σ_{CCS}, has been used to give a new operational semantics to Milner's Calculus of Communicating Systems. The interleaving case graph of Σ_{CCS} has been proved homomorphic to the transition system of CCS and the homomorphism is such that the unfoldings from two corresponding nodes are identical. We argue that our extensions do not affect the results proved for standard C/E Systems, thus non-sequential processes defined for them [19] can also be immediately defined for CCS.

The translation is in one direction only, mainly because synchronization in CCS involves only two agents, whereas in Petri Nets the set of preconditions of an event may even be an infinite set. CCS synchronization algebra needs extensions in order to cope with this problem.

We believe that lifting the restriction to *guarded* CCS terms might be conceptually easy but formally cumbersome. In fact, if we apply our decrel definition to an unguarded CCS term, the complete set of grapes we obtain may contain infinite grapes, or may be infinite, or both. For instance, from

rec x. x

we get the singleton consisting of the infinite term

{let x=x in let x=x in ...}.

More interestingly, from

rec x. α|x

we can obtain the infinite case

I = {let x=α|x in α|id, let x=α|x in id|let x=α|x in α|id, ... }

which enables infinite events in parallel.

A careful treatment of infinite sets of grapes should maintain the validity of our results also in the unguarded case. However, to our knowledge, no sufficiently well-developed theory of infinite proofs (or perpetual processes, as they are called in logic programming) is now available to concisely support the necessary formal development.

Furthermore, in order to maintain the operational character of our definition, it would be necessary to handle the infinite sets of grapes on a "by need" basis. Notice, however, that in the present development this is already the case for Σ_{CCS}: it is an infinite net, but its relevant parts can be generated on demand by a finite set of rules. In particular, it would be convenient to find a finite uniform encoding for every (possibly infinite) regular set of grapes I such that the set of possible moves I $\succ\!\!-\mu\!\!\rightarrow$ I' is decidable.

A semantics in terms of Condition/Event Petri Nets is still very intensional. As was the case for interleaving CCS, a further step is needed for defining a notion of observation and a consequent congruence on the semantic domain. In this paper we have not been concerned with this issue. However, the Petri net proposed here is a natural starting point of this further step.

In fact, a non-sequential process [19] (or, better, an action-labelled partial ordering containing only its events [7]) can be associated to every path in the case graph, providing a natural notion of partial ordering observation. From it, an observational equivalence and congruence can be derived, by extending the notion of bisimulation. This can be done either directly like in [2, 11], or on a class of partial ordering-labelled trees called NMS, like in [5].

The resulting extensional semantics of CCS do take into account the actual causal dependencies between events in a way impossible for multiset semantics, let alone interleaving semantics, thus eventually providing a firm motivation for the whole approach.

This fact can be appreciated considering the nets and the multiset transition systems associated to the following CCS agents:

a) $\alpha NIL \mid \beta NIL$ and

b) $\alpha NIL \mid \beta NIL + \alpha\beta NIL$.

Figure 6.1 shows the fragments of Σ_{CCS} relevant to these agents. Clearly the subsystem in part **b**) describes the possible causal dependency of β from α, while the other does not.

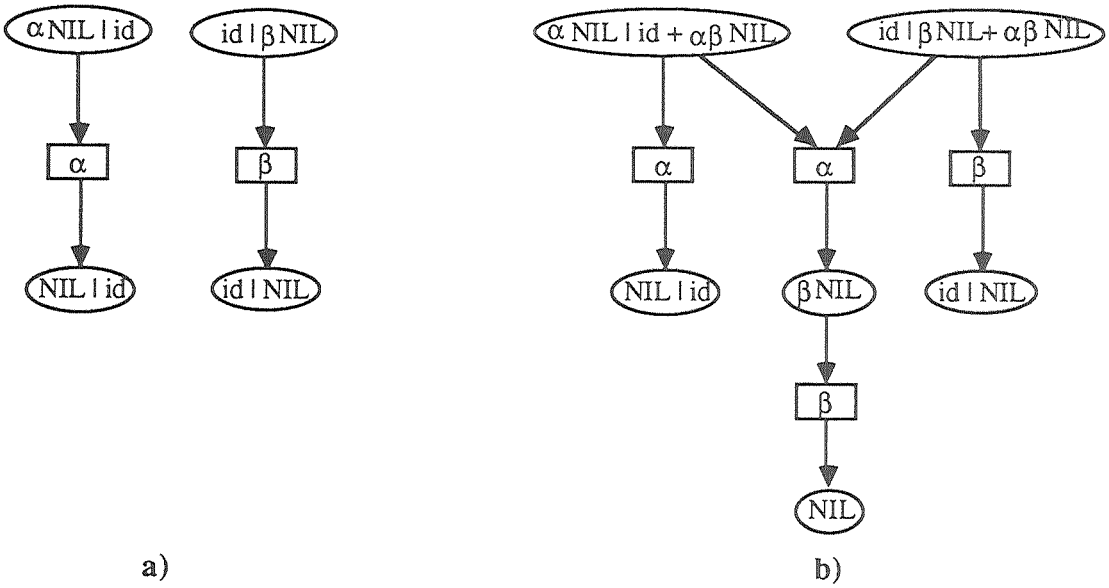

Fig. 6.1. The fragments of Σ_{CCS} relevant to the CCS agents $\alpha NIL \mid \beta NIL$ (in part **a**)) and $\alpha NIL \mid \beta NIL + \alpha\beta NIL$ (in part **b**)).

It is of course possible to define a notion of observation which can be used for giving an extensional semantics of Petri Nets which distinguishes the two C/E systems above. For example if NMS's [5] are used as the basis for defining behavioural equivalences, we have that the labelled trees corresponding to the systems above are those of Fig. 6.2; any equivalence would differentiate them, even a coarse trace equivalence considering only the sets of paths in the trees.

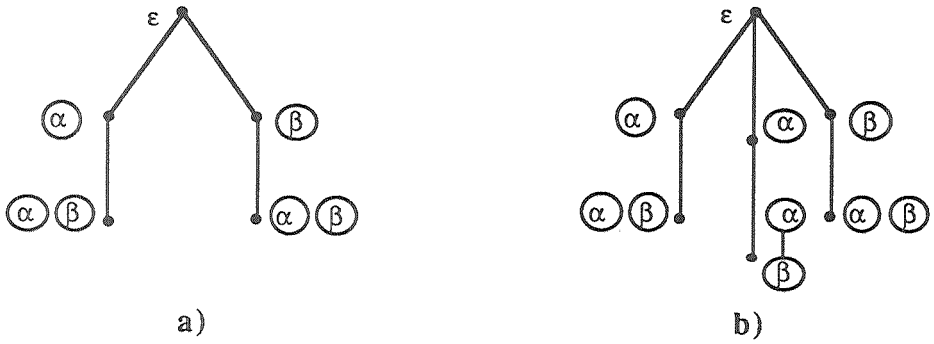

Fig. 6.2. The NMS's corresponding to the two systems of Fig. 6.1.

On the other hand the two CCS agents, αNIL | βNIL and αNIL | βNIL + αβNIL, have a very similar structure of multiset transitions:

$$\alpha\text{NIL} \mid \beta\text{NIL} \begin{cases} \text{—}\{\alpha\}\text{→ NIL} \mid \beta\text{NIL—}\{\beta\}\text{→NIL} \mid \text{NIL} \\ \text{—}\{\alpha, \beta\}\text{→ NIL} \mid \text{NIL} \\ \text{—}\{\beta\}\text{→ } \alpha\text{NIL} \mid \text{NIL—}\{\alpha\}\text{→NIL} \mid \text{NIL} \end{cases}$$

and

$$\alpha\text{NIL} \mid \beta\text{NIL} + \alpha\beta\text{NIL} \begin{cases} \text{—}\{\alpha\}\text{→ NIL} \mid \beta\text{NIL—}\{\beta\}\text{→NIL} \mid \text{NIL} \\ \text{—}\{\alpha, \beta\}\text{→ NIL} \mid \text{NIL} \\ \text{—}\{\beta\}\text{→ } \alpha\text{NIL} \mid \text{NIL—}\{\alpha\}\text{→NIL} \mid \text{NIL} \\ \\ \text{—}\{\alpha\}\text{→ } \beta\text{NIL—}\{\beta\}\text{→NIL} \mid \text{NIL} \end{cases}$$

and it is thus impossible for any reasonable extensional semantics (e.g., with an idempotent + operator) to differentiate the two CCS agents above.

Acknowledgements. The authors would like to thank E.-R. Olderog for stimulating discussions, and the anonymous referees which have contributed to the actual shaping of the paper through their detailed suggestions and pressing demands.

References

1. Austry, D. and Boudol, G. Algébre de Processus et Synchronization, *Theoret. Comput. Sci.*, **30**, pp. 91-131 (1984) .

2. Boudol, G. and Castellani, I. On the Semantics of Concurrency: Partial Orders and Transition Systems. Proc.*TAPSOFT '87* (H. Ehrig et al., eds.), LNCS 249, pp. 123-137, Springer-Verlag, 1987.

3. De Cindio, F., De Michelis, G., Pomello, L. and Simone C.: Milner's Communicating Systems and Petri Nets. In: *Selected papers from the 3rd European Workshop on Applications and Theory of Petri Nets* (A. Pagnoni and G. Rozenberg, eds.), IF 66, pp. 40-59, Springer-Verlag, 1983.

4. Degano, P., De Nicola, R. and Montanari, U.: Partial Ordering Derivations for CCS. Proc. 5th Int. Conf. on Fundamentals of Computation Theory (L. Budach, ed.), LNCS 199, pp. 520-523, Springer-Verlag, 1985.

5. Degano, P., De Nicola, R. and Montanari, U.: Observational Equivalences for Concurrency Models. In: *Formal Description of Programming Concepts III*, (M. Wirsing, ed.), pp. 105-132, North-Holland, 1987

6. Degano, P., De Nicola, R. and Montanari, U.: CCS is an (Augmented) Contact-Free C/E System. To appear in Proc. Advanced School on Mathematical Models for the Semantics of Parallelism. LNCS, Springer-Verlag, 1987.

7. Degano, P., De Nicola, R. and Montanari, U.: A Partial Ordering Semantics for CCS, forthcoming.

8. Degano, P. and Montanari, U.: Concurrent Histories: A Basis for Observing Distributed Systems. *Journal of Computer and System Sciences,* **34**, pp. 442-461 (1987).

9. De Nicola, R. and Hennessy, M.: Testing Equivalences for Processes. *Theoret. Comput. Sci.*, **34**, pp. 83-133 (1984).

10. Genrich, H.J., Lautenbach, K. and Thiagarajan, P.S.: Elements of General Net Theory. In: *Net Theory and Applications* (W. Brauer, ed.) LNCS 84, pp. 21-163, Springer-Verlag, 1980.

11. van Glabbeek, R. and Vaandrager, F.: Petri Net Models for Algebraic Theories of Concurrency. Proc. PARLE Conf. (J. W. de Bakker, A.J. Nejman and P.C. Treleaven, eds.) LNCS 259, Springer-Verlag, 1987.

12. Goltz, U. and Mycroft, A.: On the Relationships of CCS and Petri Nets. Proc. 11th ICALP (J. Paredaens, ed.), LNCS 172, pp. 196-208, Springer-Verlag, 1984.

13. Keller, R.: Formal Verification of Parallel Programs. *Communication of ACM*, **7**, pp. 561-572 (1976)

14. Milner, R.: *A Calculus of Communicating Systems*. LNCS 92, Springer-Verlag, 1980.

15. Milner, R. Calculi for Synchrony and Asynchrony, *Theoret. Comput. Sci.*, **25**, pp. 267-310 (1983).

16. Milner, R. Notes on a Calculus for Communicating Systems, in: *Control Flow and Data Flow: Concepts of Distributed Programming* (M. Broy, ed.), NATO ASI Series F: Vol. 14, pp. 205-228, Springer-Verlag, 1984.

17. Olderog, E.-R. Operational Petri Net Semantics for CCSP. In *Advances in Petri Nets 1987*, (G. Rozenberg, ed.) LNCS 266, pp. 196-223, Springer-Verlag, 1987.

18. Plotkin, G.: A Structural Approach to Operational Semantics. Technical Report DAIMI FN-19, Aarhus University, Department of Computer Science, Aarhus, 1981.

19. Reisig, W.: *Petri Nets: An Introduction*, EACTS Monographs on Theoretical Computer Science, Springer-Verlag, 1985.

20. Winskel, G.: Event Structure Semantics for CCS and Related Languages, Proc. 9th ICALP (M. Nielsen and E. M. Schmidt, eds.), LNCS 140, pp. 561-567, Springer-Verlag, 1982.

21. Winskel, G.: Categories of Models of Concurrency, (S. D. Brooks, A. W. Roscoe, G. Winskel, eds.), LNCS 197, pp. 246-267, Springer-Verlag, 1985.

22. Winskel, G.: Petri Nets, Algebras, Morphisms and Compositionality, *Info. and Co.*, **72**, 197-238 (1987).