A Partial Ordering Semantics for CCS

Pierpaolo Degano, Rocco De Nicola*, Ugo Montanari

Abstract. A new operational semantics for "pure" CCS is proposed that considers the parallel operator as a first class one, and permits a description of the calculus in terms of partial orderings. The new semantics (also for unguarded agents) is given in the SOS style via the partial ordering derivation relation. CCS agents are decomposed into sets of sequential sub-agents, and the new derivations which relate sets of sub-agents describe their actions and their causal dependencies. The computations obtained by animating sets of sub-agents via the partial ordering derivation relation are "observed" either as interleaving or partial orderings of events. Interleavings coincide with Milner's many step derivations, and "linearizations" of partial orderings are all and only interleavings. In order to obtain more abstract semantics, we introduce two relations of partial ordering observational equivalence and congruence that preserve concurrency. These are finer than Milner's exactly in that they distinguish interleaving of sequential nondeterministic agents from their concurrent execution.

Febbraio 1988

* Istituto di Elaborazione dell'Informazione, C.N.R., Pisa

ADDR: Corso Italia, 40 - 56100 PISA - Italy TEL: +39 - 50 - 510111 TLX: 590291 DIPISA I E_MAIL: dipisa.UUCP
1. Introduction

Many different models have been proposed to specify systems whose subparts can progress in parallel, synchronize and exchange messages. These models can be compared by considering how they describe the fact that events (atomic actions, synchronizations, communications) can be performed by subparts of a system concurrently, i.e., independently from each other. If we take this standpoint, the various models of concurrency can be divided into two broad groups: those based on interleaving and those based on true concurrency.

Models based on interleaving express concurrency among events by saying that they may occur in any order. Thus, a total ordering among possibly spatially separated and causally independent events is imposed: a global clock and global states are assumed. Their proposers, among which [Mil80 and 83, Niv82, Lam83, AB84, BHR84, BK84, Mne85, Hen87], stress the simplicity of the underlying mathematics as a sufficient reason to stick to this approach, since it permits easier reasoning about concurrent systems and proving most of their properties.

Models based on true concurrency use, instead, partial ordering of events where concurrency is represented as absence of ordering. In this framework, where no global clock is assumed, the behaviour of a system is expressed in terms of causal relations between the events performed by subparts of its distributed state. Their proposers (some references are [Maz77, Lam78, Pet80, Wink80, NPW81, Shi82, DK83, GR83, BS86, Pra86, BC87, DM87a and b]) claim that these models offer a more faithful picture of reality, and that some liveness properties of concurrent systems can be better understood and studied in this framework.

A classical representative of the models based on interleaving is Milner's Calculus of Communicating Systems, CCS for short, [Mil80]. It relies on a small number of operators which are used to build CCS terms. These are considered as agents that may perform certain actions to become other agents. The operational semantics of the calculus is given through labelled transition systems, and the fact that agent $E_0$ evolves to $E_1$ by performing an action $\mu$ is rendered by $E_0 \xrightarrow{\mu} E_1$. The technique used (Structured Operational Semantics or SOS [Plo81]) relies on the well-known idea of describing the behaviour of systems by sequences of transitions between configurations. Transitions of compound systems are defined in a syntax-driven way, via axioms and inference rules.

Since the original version of CCS was geared towards the interleaving approach, its semantics does not consider the operator for parallel composition of processes "||" as primitive: given any finite process containing $\l$, there always exists another process without $\l$ which exhibits the same behaviour.

In this paper a new operational semantics for CCS is proposed that considers instead the parallel operator as a first class one, and offers a partial ordering semantics for the calculus. Again, the operational semantics is given in the SOS style, but a different transition relation, called partial ordering derivation relation, is defined. It relates subparts of CCS agents, rather than their whole global state, and carries information about causal dependencies. CCS agents are decomposed into sets of sequential processes, called grapes, and the new transitions not only describe the actions
agents may perform when in a given state, but they also express the causal relation among subparts of agents when the global state changes. The partial ordering derivation relation is defined via inference rules which are in direct correspondence with those of [Mii80]. Thus, also the deduction of either transitions follows the same pattern.

The new transitions have the form $I_1 - [\mu, \mathcal{R}] \rightarrow I_2$ where $I_1$ and $I_2$ represent sets of grapes, and $\mathcal{R}$ is a relation giving additional information about the causal relations among agents. The grapes in $I_1$ perform the action $\mu$ evolving to those in $I_2$, thus we say that the grapes of $I_1$ cause, through $\mu$, those in $I_2$. The information about other grapes caused by grapes in $I_1$, but not by $\mu$, is recorded in $\mathcal{R}$. The intended dynamic meaning is that set of grapes $I_1$ occurring in the current state can be replaced, after showing an event labelled by $\mu$, by all grapes in $I_2$ and all grapes related by $\mathcal{R}$ to obtain the new state.

As an example, consider the CCS agent $(\alpha.NIL|\beta.NIL)+\gamma.NIL$, which may evolve to $NIL|\beta.NIL$ after resolving the nondeterministic choice (expressed by $+$) in favour of $\alpha$. In the interleaving approach, this will be rendered as

$$(\alpha.NIL|\beta.NIL)+\gamma.NIL \longrightarrow_{\alpha} NIL|\beta.NIL.$$  

We will write it as

$$(\alpha.NIL|\beta.NIL)+\gamma.NIL \rightarrow \alpha \rightarrow (\alpha.NIL|\beta.NIL)+\gamma.NIL \leq idl|\beta.NIL) \rightarrow \{NIL|lid)$$

where $(\alpha.NIL|\beta.NIL)+\gamma.NIL$, NIL|lid and idl|\beta.NIL are grapes.

In this way we describe the fact that grape $(\alpha.NIL|\beta.NIL)+\gamma.NIL$ causes both grape idl|\beta.NIL and the event labelled by $\alpha$ which in turn causes grape NIL|lid. Note that the possibility idl|\beta.NIL has to perform $\beta$ independently of the occurrence of $\alpha$ is implied by the absence of any causal relation between $\alpha$ and idl|\beta.NIL. The $\alpha$-derivation of grape $(\alpha.NIL|\beta.NIL)+\gamma.NIL$ is shown in Fig. 1.1. It should be noted that every derivation of the original calculus can always be recovered from our partial ordering derivation, simply by “putting together” its initial and final sets of grapes.

In the example above, we get NIL|\beta.NIL by putting together the two grapes NIL|lid and idl|\beta.NIL, caused by $\{(\alpha.NIL|\beta.NIL)+\gamma.NIL\}$.

![Figure 1.1](image-url)

**Figure 1.1.**

A transition of the partial ordering operational semantics. Grapes are represented by labelled boxes, events by labelled circles and the causal relation is expressed through its Hasse diagram growing downwards. It represents

$$(\alpha.NIL|\beta.NIL)+\gamma.NIL \rightarrow \alpha \rightarrow (\alpha.NIL|\beta.NIL)+\gamma.NIL \leq idl|\beta.NIL) \rightarrow \{NIL|lid).$$
A transition of the above form may look a bit unnatural. We are used to conceive labelled transitions as relations between a set of processes and an action, and between that action and all the new processes. In the transition above, grape $\text{idl}\beta.NIL$ is, instead, directly related to grape $(\alpha.NIL|\beta.NIL) + \gamma.NIL$. This happens because the evolution of a nondeterministic process like the latter requires first choosing one of the alternatives, and then performing an action of the chosen grapes. A possible way of describing in detail the above $\alpha$-transition is drawn in Fig. 1.2.a): first, a choice-event, labelled by $?$, causes two concurrent grapes $\alpha.NIL|\text{idl}$ and $\text{idl}\beta.NIL$; then, the former performs an $\alpha$. It is however important noticing that the decision and the action are to be considered as a single indivisible action in order to be faithful to the original semantics. Since CCS has no mechanisms for defining atomic actions from sequences, we are left with two alternatives: either to hide the decision, and obtain transitions like that of Fig. 1.1, or to incorporate the decision into the action itself, and give rise to usual transitions (Fig. 1.2.b). In [DDM85, 87b], we have followed the latter approach, but it gives rise to an operational semantics that does not take fully into account the possible parallelism of CCS agents. For example, independencies are lost between some concurrent actions in $+$-context; in the case of agent $(\alpha.NIL|\beta.NIL) + \gamma.NIL$ a causal relation between $\alpha$ and $\beta$ is forced, thus identifying this agent with $\alpha.\beta.NIL + \beta.\alpha.NIL + \gamma.NIL$.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig12.png}
\caption{Alternative descriptions of the $\alpha$-transition of agent $(\alpha.NIL|\beta.NIL) + \gamma.NIL$.}
\end{figure}

A \textit{computation} is a sequence of sets of grapes (i.e., system states corresponding to CCS agents), and of partial ordering derivations (i.e., system transitions). A computation of agent $(\alpha.NIL|\beta.NIL) + \gamma.NIL$ is

$$
\xi = \{ (\alpha.NIL|\beta.NIL) + \gamma.NIL \}
\{ (\alpha.NIL|\beta.NIL) + \gamma.NIL \}[\alpha, \{ (\alpha.NIL|\beta.NIL) + \gamma.NIL \leq \text{idl}\beta.NIL \}] \rightarrow \{ \text{NIL|idl} \}
\{ \text{NIL|idl}, \text{idl}\beta.NIL \}
\{ \text{idl}\beta.NIL \} \rightarrow \{ \emptyset \} \rightarrow \{ \text{idlNIL} \}
\{ \text{NIL|idl}, \text{idlNIL} \}.
$$

We can extract from computations either sequences or partial orderings of actions. In the first case we keep track of the temporal ordering in which the actions have been performed (in our example $\alpha$ followed by $\beta$, i.e., $\alpha\beta$); while in the second case we keep track of the causal dependencies.
among the actions (in our example $\alpha$ concurrent with $\beta$). By "observing" computations in either way and by taking into account their initial and final sets of grapes we obtain interleaving or partial ordering many step derivations. On passing, we remark that our approach is indeed operational since we build our many step derivations by composing elementary steps and then abstracting. This differs from other approaches [e.g., BC87, BS86], where transition systems are used to directly associate partial orderings to agents: the notion of elementary step and the possibility of growing computations from them are lost in favour of a more denotational flavour.

The two kinds of derivations provide us with a firm ground for studying the relationships between the interleaving and the partial ordering approaches. Actually, the natural direct correspondence between our partial ordering derivation relation and Milner's allows us to prove that his many step derivations coincide with our interleaving derivations. This results guarantees also that the original interleaving operational semantics of CCS is immediately retrievable from ours. Furthermore, we will see that, given a computation, "linearizing" the causal relation of its partial ordering many step derivation results in the set of sequences which are all and only the interleaving many step derivations. This property, called complete concurrency, amounts to saying that two concurrent events can be generated in either ordering, and plays a crucial role in relating the interleaving and partial ordering semantics of CCS. Back to our example, all and only the linearizations of the partial ordering of the derivation obtained from $S$ (a concurrent with $B$) are exactly Milner's many-step derivations associated to agent $(\alpha.NIL|B.NIL)+\gamma.NIL$ (when the same side of the $+$ is chosen), i.e., $\alpha\beta$ and $\beta\alpha$.

When the behaviour of concurrent systems is described through a relation between their states, all their internal states are to be taken into account. Often, only some of them, however, are relevant for actual system analysis. So operational descriptions of this kind end up specifying too many details, and introducing unnecessary and unnatural differentiations. A remedy advocated by Milner is to consider concurrent systems as black boxes, to assume some actions as internal, thus invisible, and therefore to describe system behaviours only in terms of visible actions. To this aim, notions of observational equivalence and congruence are introduced which are based on experimentations, and permit to abstract from unwanted details [Mil80 and 83, HM85]. Because of the intrinsically sequential nature of the experiments allowed, concurrency is still not a primitive notion of the theory. Here, we introduce a new notion of partial ordering observation through which we can define notions of observational equivalence and congruence that preserve concurrency.

Like Milner's, our starting point is the notion of bisimulation [Par81]: two agents are equivalent if they are able to perform the same partial orderings of visible actions, evolving to equivalent agents. The new relations of partial ordering observational equivalence and congruence are finer than Milner's exactly in that they distinguish interleaving of sequential nondeterministic processes from their concurrent execution. As a matter of fact, the two equivalences and the two congruences coincide when dealing with nondeterministic sequential processes only.
We started some years ago our investigation on a partial ordering approach to the semantics of concurrent languages, and reported our intermediate results in a number of papers [DDM85, 87a, 87b, DGM87, DM87a and b]. As already mentioned, the semantics for CCS proposed in the first three papers is however not completely satisfactory. There, we keep a one-to-one correspondence between set of grapes reachable through derivations and agents, between the new rules and Milner’s, and between the proofs of the derivations, but we do not always permit concurrent execution of intuitively independent actions. In [DDM87c] we solve this problem at the price of a more complex notion of distributed state, and of a less natural set of rules. Actually, due to a distributed treatment of the choice operator, a decomposition relation is introduced which causes a combinatorial explosion of the possible sets of grapes, and the loss of the one-to-one correspondence between states, i.e., sets of grapes, and CCS agents. Moreover, when dealing with unguarded recursion the decomposition relation may originate infinite sets of infinite grapes, and to deduce them, problems arise similar to those about perpetual processes in Logic Programming.

In this paper we are able to give a full account of parallelism, yet maintaining a syntactic one-to-one correspondence between the interleaving and the partial ordering approaches. We keep a centralized treatment of choice thus avoiding state explosion. Moreover, we straightforwardly deal with unguarded recursion. The causal relation among events may in this case be infinitely branching, thus reflecting the possible unbounded parallelism (see also Fig. 4.4).

The rest of the paper is organized as follows. Section 2 surveys the original interleaving semantics of CCS which relies on the derivation relation and on the notion of bisimulation. Section 3 defines the new partial ordering derivation relation on sets of sub-agents rather than on whole agents. The partial ordering many step derivation relation is introduced in Section 4 and compared with Milner’s. Using this new relation, partial ordering observational equivalence and congruence are defined in the same section, and shown to be finer than the original ones, yet coincident on sequential non-deterministic agents. Finally, Section 5 discusses the relationships between this work and other proposals of true concurrent semantics for CCS.

2. CCS and its Interleaving Semantics

This section contains a brief introduction to “pure” CCS, i.e., the calculus without value passing. First, we shall introduce the syntax of the calculus, then we will present the traditional interleaving semantics and the observational equivalence and congruence of [Mil80] refined in [Mil84].

Definition 2.1. (agents)
Let
- \( \Delta = \{ \alpha, \beta, \gamma \ldots \} \) be a fixed set and \( \Delta^- = \{ \alpha^- | \alpha \in \Delta \} \), assuming \( (\alpha^-)^- = \alpha \);
- \( \Lambda = \Delta \cup \Delta^- \) (ranged over by \( \lambda \)) be the set of visible actions;
- \( \tau \in \Lambda \) be a distinguished invisible action, and let \( \Lambda \cup \{ \tau \} \) be ranged over by \( \mu \).
The CCS agents, ranged over by E, consists of all closed terms (i.e., terms without free variables) which can be generated by the following BNF-like grammar

\[ E ::= x | N | E \alpha | E[\phi] | E + E | E \epsilon | \text{rec } x. E, \]

where \( x \) is a variable and \( \phi \) is a permutation of \( \Lambda \cup \{ \tau \} \) which preserves \( \tau \) and the operation \( - \) of complementation.

We assume that the precedence among operators is \( \alpha > [\phi] > \mu > \text{rec} > + > \epsilon \).

CCS has a two level semantics: the first level describes the behaviour of agents through an abstract machine and the second level forgets their internal structure by identifying those machines which all exhibit the same external behaviour.

The first level, i.e., the interleaving operational semantics, is based on a labelled transition system with a transition relation defined via a set of transition rules. The relation, called \textit{derivation relation} and denoted by \( -\mu\rightarrow \), relies on the intuition that agent \( E_0 \) may evolve to become agent \( E_1 \) either by reacting to a \( \lambda \)-stimulus from its environment \( (E_0 \rightarrow \lambda \rightarrow E_1) \) or by performing an internal action which is independent of the environment \( (E_0 \rightarrow \epsilon \rightarrow E_1) \).

\textbf{Definition 2.2. (transitions)}

\textit{Milner's derivation relation} \( E_0 \rightarrow \mu \rightarrow E_1 \) is defined as the least relation satisfying the following axiom and inference rules.

- \textbf{Act)} \( \mu E \rightarrow \mu E \)
- \textbf{Res)} \( E_0 \rightarrow \mu \rightarrow E_1 \) implies \( E_0 \alpha \rightarrow \mu \rightarrow E_1 \alpha, \mu \notin \{ \alpha, \alpha' \} \)
- \textbf{Rel)} \( E_0 \rightarrow \mu \rightarrow E_1 \) implies \( E_0[\phi] \rightarrow \phi(\mu) \rightarrow E_1[\phi] \)
- \textbf{Sum)} \( E_0 \rightarrow \mu \rightarrow E_1 \) implies \( E_0 + E \rightarrow \mu \rightarrow E_1 \) and \( E + E_0 \rightarrow \mu \rightarrow E_1 \)
- \textbf{Com)} \( E_0 \rightarrow \mu \rightarrow E_1 \) implies \( E_0 E \rightarrow \mu \rightarrow E_1 E \) and \( E E_0 \rightarrow \mu \rightarrow E_1 E \)
- \( E \epsilon \rightarrow \lambda \rightarrow E_1 \) and \( E' \epsilon \rightarrow \lambda \rightarrow E'_1 \) implies \( E E_0 \epsilon \rightarrow \tau \rightarrow E_1 E'_1 \)
- \textbf{Rec)} \( E_0 [\text{rec } x. E_0 / x] \rightarrow \mu \rightarrow E_1 \) implies \( \text{rec } x. E_0 \rightarrow \mu \rightarrow E_1 \).

Hereeto, we will use the following conventions to talk about sequences of actions and sequences of visible actions:
- \( E = \varepsilon \rightarrow E' \), \( \varepsilon \) being the null string of \( \Lambda^* \), stands for \( E \rightarrow \sigma_0 \rightarrow E' \), \( n \geq 0 \);
- \( E = \mu \rightarrow E' \), stands for there exist \( E_1 \) and \( E_2 \) such that \( E = \varepsilon \rightarrow E_1 \rightarrow \mu \rightarrow E_2 = \varepsilon \rightarrow E' \);
- \( E = s \rightarrow E' \), \( s = \lambda_1 \ldots \lambda_n \in \Lambda^+ \), stands for there exist \( E_i \), \( 0 < i < n \), such that \( E = E_1 = \lambda_1 = \cdots = \lambda_2 = \cdots = \lambda_n = \varepsilon \rightarrow E_n = E' \);
- the relation \( = s \rightarrow \), \( s \in \Lambda^* \) will be referred to as \textit{many step derivation}.

The derivation relation of Definition 2.2 completely specifies the operational semantics of CCS; the second level of CCS semantics is obtained by abstracting from unwanted details. To this purpose, a notion of bisimulation is introduced which is then used to define an equivalence relation on agents. Agents which are observationally equivalent can then be identified.
We can define a bisimulation relation \( R \) between CCS agents which consists of all those pairs of agents related via \( =\Rightarrow \) to equal (up to \( R \)) agents. Loosely speaking, two agents \( E_0 \) and \( E_1 \) are considered as bisimilar, written \( E_0 \sim E_1 \), if and only if there exists a bisimulation \( R \) containing the pair \( \langle E_0, E_1 \rangle \) and guaranteeing that \( E_0 \) and \( E_1 \) are able to perform equal sequences of visible actions evolving to equal (up to \( R \)) agents.

**Definition 2.3. (observational equivalence)**

1. If \( R \) is a binary relation between CCS agents, then \( \Psi \), a function from relations to relations, is defined as follows: 
   
   \( \langle E_0, E_1 \rangle \in \Psi(R) \) if, for every \( s \in \Lambda^* \),
   
   i) whenever \( E_0 \Rightarrow s \Rightarrow E_0 \) there exists \( E'_0 \) such that \( E_1 \Rightarrow s \Rightarrow E'_1 \) and \( \langle E'_0, E'_1 \rangle \in R \)
   
   ii) whenever \( E_1 \Rightarrow s \Rightarrow E'_1 \) there exists \( E'_0 \) such that \( E_0 \Rightarrow s \Rightarrow E'_0 \) and \( \langle E'_0, E'_1 \rangle \in R \).

2. A relation \( R \) is a bisimulation if \( R \subseteq \Psi(R) \).

3. Relation \( = = \cup \{ R \mid R \subseteq \Psi(R) \} \), is called observational equivalence.

**Proposition 2.1.**

- Function \( \Psi \) is monotonic on the lattice of relations under inclusion.
- Relation \( = = \) is a bisimulation and an equivalence relation.

Below we propose two pairs of equivalent processes. The first shows that the equivalence based on bisimulation succeeds in ignoring the internal structure of agents; the second shows that concurrent and nondeterministic processes may be identified.

**Example 2.1.**

a. \( \alpha.(\beta.NIL + \tau.\gamma.NIL) + \alpha.\gamma.NIL \sim \alpha.(\beta.NIL + \tau.\gamma.NIL) \);

b. \( \alpha.NIL \mid \beta.NIL \sim \alpha.\beta.NIL + \beta.\alpha.NIL \).

Here, the relevant bisimulations are

a. \{ \( \langle \alpha.(\beta.NIL + \tau.\gamma.NIL) + \alpha.\gamma.NIL, \alpha.(\beta.NIL + \tau.\gamma.NIL) \rangle, \langle \beta.NIL, \beta.NIL \rangle, \langle \tau.\gamma.NIL, \tau.\gamma.NIL \rangle, \langle \gamma.NIL, \gamma.NIL \rangle, \langle NIL, NIL \rangle \} \}

b. \{ \( \langle \alpha.NIL \mid \beta.NIL, \alpha.\beta.NIL + \beta.\alpha.NIL \rangle, \langle NIL \mid \beta.NIL, \beta.NIL \rangle, \langle \alpha.NIL \mid NIL, \alpha.NIL \rangle, \langle NIL \mid NIL, NIL \rangle \} \}.

Rather than equivalence relations we need congruences which guarantee that equivalent agents can be interchangeably plugged into any context, without affecting the overall behaviour. It is well known that observational equivalence is not preserved by \( + \)-contexts, and thus in [Mil80] and [Mil84] this relation is strengthened to a congruence. The definition below characterizes observational congruence without making any explicit use of contexts.

**Definition 2.4. (observational congruence, [Mil84])**

\( E_0 \equiv E_1 \) iff

i) whenever \( E_0 \xrightarrow{\mu} E'_0 \), there exists \( E'_1 \) such that \( E_1 \xrightarrow{\mu} E'_1 \) and \( E_0 = E'_0 \)

ii) whenever \( E_1 \xrightarrow{\mu} E'_1 \), there exists \( E'_0 \) such that \( E_0 \xrightarrow{\mu} E'_0 \) and \( E'_0 = E'_1 \).
This definition shows exactly in what respect observational congruence differs from observational equivalence; however, it has the disadvantage of needing explicit concatenations of visible and invisible actions. We aim at giving a similar definition of congruence where partial orderings of events are considered instead of interleavings of events; while string concatenation is trivial, problems arise when a general notion of concatenation on partial orderings is needed. Thus, we introduce below a less elegant characterization of observational congruence the pattern of which will be followed in defining the partial ordering one in Section 4. The alternative congruence uses again observational equivalence, but takes into account only non empty initial sequences of silent actions.

Definition 2.5. (another characterization of observational congruence)

\( E_0 \preceq E_1 \) iff

i) \( E_0 = E_1 \)

ii) whenever \( E_0 \rightarrow_{\tau} E_0' \), there exists \( E_1' \) such that \( E_1 = \tau \rightarrow E_1' \) and \( E_0 = E_1' \)

iii) whenever \( E_1 = \tau \rightarrow E_1' \), there exists \( E_0' \) such that \( E_0 = \tau \rightarrow E_0' \) and \( E_0' = E_1' \).

It is easy showing that the two context independent equivalences defined above are indeed the same congruence.

Proposition 2.2. (the two context independent equivalences are the same congruence)

\( E_0 \preceq E_1 \) if and only if \( E_0 \preceq E_1 \)

Proof. Immediate, by definition of observational equivalence and by noticing that \( E_0 \rightarrow_{\tau} E_0' \) implies \( E_0 = \tau \rightarrow E_0' \).

3. Defining the Partial Ordering Derivation Relation

In this section we define the partial ordering derivation relation \( I_1 \llbracket [\mu, R] \rrbracket I_2 \), which generalizes Milner's derivation relation \( E_1 \rightarrow_{\mu} E_2 \), and allows us to obtain a notion of many step derivation based on partial orderings.

We first need to single out those sub-agents of a given CCS agent which can reasonably be considered as single entities, in that they may perform actions in isolation.

Definition 3.1. (defining CCS sequential agents)

A grape is a term defined by the following BNF-like grammar

\[
G ::= E \mid \text{id} \mid G \text{id} \mid G \alpha \mid G[\phi]
\]

where \( E, \alpha, [\phi] \) have the standard CCS meaning.
Intuitively speaking, a grape represents a sub-agent 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 on many occasions. For example, it allows us to differentiate the behaviour of processes like \((\alpha.\beta.\text{NIL} \mid \alpha^- . \text{NIL})\alpha\) and \((\alpha.\beta.\text{NIL})\alpha \mid (\alpha^- . \text{NIL})\alpha\). 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, \text{lid} and \text{idl}, which are tags recording that there are other processes that can perform actions concurrently with those of the given sequential process.

A CCS agent is decomposed by function \text{dec} into a set of grapes.

Definition 3.2. (decomposing a CCS agent into its sequential agents)

Function \text{dec} decomposes a CCS agent into a set of grapes and is defined by structural induction as follows:

\[
\begin{align*}
\text{dec}(\text{NIL}) &= \{\text{NIL}\} \\
\text{dec}(E \cdot \alpha) &= \text{dec}(E)\alpha \\
\text{dec}(E_1 + E_2) &= \{E_1 + E_2\} \\
\text{dec}(\text{rec } x.\ E) &= \{\text{rec } x.\ E\}.
\end{align*}
\]

In this definition, and from now onwards, the application of a syntactic constructor to a set of grapes is defined as applying the constructor elementwise, e.g., \(\Lambda \alpha = \{g \cdot \alpha \mid g \in \Lambda\}\).

The decomposition goes inside the structure of agents and stops when a process prefixed by an action or the NIL process are encountered, since these cannot be considered but atomic sequential processes. It also stops when a sum or a recursion is encountered; this choice is debatable. For example, if we take agent \(\alpha.\text{NIL}\beta.\text{NIL} + \gamma.\text{NIL}\) it is not immediate whether it should be considered as a single sequential process, or rather as two sequential processes, namely \(\alpha.\text{NIL}\text{lid} + \gamma.\text{NIL}\) and \(\text{idl}\beta.\text{NIL} + \gamma.\text{NIL}\). We take here the first standing and assume that, in order to resolve the choice between the two sides of a +, all concurrent processes on the same side must agree on being chosen. A similar situation arises with recursively defined agents, where all concurrent agents in the rec body must unwind at the same time.

The above assumption of centralized control contrasts with that of [DDM87c]. There, a decomposition relation \text{decr} is defined which does not consider as sequential those agents having + and rec as top-level operators, and goes inside the structure of agents even in this cases. In the case of +, this results in a cartesian product of the sequential components of the alternative agents, thus yielding a combinatorial explosion of the number of generated grapes, and the loss of the one-to-one correspondence between states and CCS agents. Indeed, the alternatives present in all grapes are discarded by the occurrence of a transition only in those grapes affected by it. Nevertheless, the alternatives still present in the remaining grapes are meaningless and will never be taken. Decomposing the above agent \(\alpha.\text{NIL}\beta.\text{NIL} + \gamma.\text{NIL}\) through \text{decr} gives exactly the
set of grapes \{\alpha.NIL, i.dl\beta.NIL + \gamma.NIL\}. When the action \alpha is performed, state
\{NIL, i.dl\beta.NIL + \gamma.NIL\} is reached where the \gamma.NIL choice is still present, yet useless and
misleading. In the case of rec, this approach gives rise to problems when dealing with unguarded
recursion; actually infinite sets of infinite grapes may be generated.

Example 3.1.
\[
\text{dec}(((\text{rec} x. \alpha \cdot x + \beta \cdot x) | \text{rec} x. \alpha \cdot x + \gamma \cdot x) | \text{rec} x. \alpha \cdot x) | \alpha) =
\{
((\text{rec} x. \alpha \cdot x + \beta \cdot x) i.dl (i.dl), ((i.dl(\text{rec} x. \alpha \cdot x + \gamma \cdot x) i.dl), (i.dl(\text{rec} x. \alpha \cdot x) | \alpha).
\]

Example 3.2.
\[
\text{dec}(((\alpha.NIL | \gamma.NIL + \theta.NIL) | (\alpha-NIL | \delta.NIL + \upsilon.NIL)) | \beta.NIL) =
\{
((\alpha.NIL | \gamma.NIL + \theta.NIL) i.dl (i.dl, (i.dl(\alpha-NIL | \delta.NIL + \upsilon.NIL)) i.dl, i.dl\beta.NIL).\]

We now define a correspondence between CCS agents and sets of grapes, more precisely with
the sets of their sequential sub-agents.

Definition 3.3.
A set of grapes \(I\) is complete if there exists a CCS agent \(\varepsilon\) such that \(\text{dec}(\varepsilon) = I\).

Full information about a CCS agent \(\varepsilon\) is retained in \(\text{dec}(\varepsilon)\), since the following property
holds.

Property 3.1.
Function \(\text{dec}\) is injective and thus defines a bijection between CCS agents and complete sets of
grapes.

Proof. Immediate by induction.

Note that the inverse function of \(\text{dec}\) is standard unification, provided that distinct variables are
substituted for each occurrence of \(i.dl\), and \(\mu E\), \((E_1+E_2)\) and \(\text{rec} \ x. \ E\) are considered atomic.
In other words, the unique unifier of a complete set of grapes \(I\) is the CCS agent of which \(I\) is the
decomposition.

Sets of grapes will play the rôle of states in our partial ordering derivations, which are defined
below. First, we need some notation used to describe the causal relation between sets of grapes.

Notation.
Let \(\mathcal{R}\) be a binary relation, by \(\mathcal{R} \downarrow 1\) we understand the set \(\{x \mid \langle x,y \rangle \in \mathcal{R}\}\) and by \(\mathcal{R} \downarrow 2\) the set
\(\{y \mid \langle x,y \rangle \in \mathcal{R}\}\).
Furthermore, we consider operators to be extended on \(\mathcal{R}\) too, e.g.,
\(\mathcal{R} \downarrow i.dl = \{\langle x.i.dl, y.i.dl \rangle \mid \langle x,y \rangle \in \mathcal{R}\}\).
The partial ordering derivation relation $I_1 - [\mu, \mathcal{R}] \rightarrow I_2$ is defined via axioms and inference rules in direct correspondence with those of Milner's $E_1 \rightarrow \mu \rightarrow E_2$. In this new relation, sets of grapes ($I_1$ and $I_2$), rather than agents, are source and target of the arrow, and $\mathcal{R}$ is a binary relation on grapes. Still, the intuitive meaning of $I_1 - [\mu, \mathcal{R}] \rightarrow I_2$ is that $I_1$ may become $I_2$ by performing action $\mu$; thus, we say that the grapes of $I_1$ cause through $\mu$ those in $I_2$ (also written as $I_1 \subseteq [\mu \subseteq I_2$). The information about other grapes which can be caused by $I_1$ but not by $\mu$ is recorded in $\mathcal{R}$. More precisely, if $g_1 \leq g_2 \in \mathcal{R}$, we have that $g_1 \in I_1$, $g_2 \in I_2$ and that $g_1$, but not action $\mu$, causes grape $g_2$. As a whole, we may say that the derivation $I_1 - [\mu, \mathcal{R}] \rightarrow I_2$ replaces the grapes of $I_1$ with those of $I_2 \cup \mathcal{R}\downarrow 2$ showing $\mu$. Thus, $\mathcal{R}$ records that there are agents that may perform actions concurrently with $I_2$.

In order to make examples more readable, we now introduce a graphical representation of $I_1 - [\mu, \mathcal{R}] \rightarrow I_2$, already informally used in the introductory section. The causal relation is represented through its Hasse diagram growing downwards (the lines representing the transitive closure are omitted), and since sets $I_1$ and $I_2 \cup \mathcal{R}\downarrow 2$ may be intersecting, distinct instances of grapes in $I_1$, $I_2$ and $\mathcal{R}\downarrow 2$ are considered. Derivation $[\tau \cdot \{g_1 \leq g_4, g_2 \leq g_5\}] \rightarrow \{g_6, g_7\}$ is shown in Fig. 3.1.b); for a denotation of the $g_i$'s, see Example 3.3.

**Definition 3.4. (partial ordering derivation relation)**

The partial ordering derivation relation $I_1 - [\mu, \mathcal{R}] \rightarrow I_2$ is defined as the least relation satisfying the following axiom and inference rules.

- **act)** $([\mu E] - [\mu, \emptyset] \rightarrow \text{dec}(E))$
- **res)** $I_1 - [\mu, \mathcal{R}] \rightarrow I_2 \quad \Rightarrow \quad I_1\backslash \alpha - [\mu, \mathcal{R}\alpha] \rightarrow I_2\backslash \alpha$ if $\mu \notin \{\alpha, \alpha'\}$
- **rel)** $I_1 - [\mu, \mathcal{R}] \rightarrow I_2 \quad \Rightarrow \quad I_1[\emptyset] - [\emptyset(\mu), \mathcal{R}[\emptyset]] \rightarrow I_2[\emptyset]$
- **sum)** $\text{dec}(E_1) \cdot I_3 - [\mu, \mathcal{R}] \rightarrow I_2 \quad \Rightarrow \quad \{E + E_1\} - [\mu, \{E + E_1\} \leq (I_3 \cup \mathcal{R}\downarrow 2)] \rightarrow I_2$
  and $\{E_1 + E\} - [\mu, \{E_1 + E\} \leq (I_3 \cup \mathcal{R}\downarrow 2)] \rightarrow I_2$
- **com)** $I_1 - [\mu, \mathcal{R}] \rightarrow I_2 \quad \Rightarrow \quad I_1 \backslash \mu \rightarrow [\mu, \mathcal{R} \backslash \mu] \rightarrow I_2 \backslash \mu$
  and $\text{id}_{I_1} \rightarrow [\mu, \mathcal{R}] \rightarrow \text{id}_{I_2}$
- $I_1 - [\lambda, \mathcal{R}] \rightarrow I_2$ and $I_1' - [\lambda', \mathcal{R}'] \rightarrow I_2' \quad \Rightarrow \quad I_1 \backslash \mu \rightarrow [\mu, \mathcal{R} \backslash \mu] \rightarrow I_2 \backslash \mu \rightarrow \text{id}_{I_2}$
  and $I_1' \backslash \lambda \rightarrow [\lambda', \mathcal{R}'] \rightarrow I_2' \backslash \lambda \rightarrow \text{id}_{I_2'}$
- **rec)** $\text{dec}(E[\text{rec } x, E/x]) \cdot I_3 - [\mu, \mathcal{R}] \rightarrow I_2 \quad \Rightarrow \quad \{\text{rec } x, E\} - [\mu, \{\text{rec } x, E\} \leq (I_3 \cup \mathcal{R}\downarrow 2)] \rightarrow I_2$.

We can now shortly comment our axiom and rules. In axiom act), a single grape is rewritten as a set of grapes, since the firing of the action makes explicit the (possible) parallelism of $E$. For every grape in $\text{dec}(E)$ is caused by $\mu$, obviously relation $\mathcal{R}$ is empty. Rules res) and rel) and the
first two rules for \( \text{com} \) simply state that if a set of grapes \( I_1 \) can be rewritten as \( I_2 \) via \( \mu \), then we can combine the access paths of the grapes in both sets with either path constructors \( \alpha_\cdot, [\phi]_\cdot, \id \) or \( \id_\cdot \), and still obtain a derivation, labelled, say, by \( \mu' \). When dealing with restriction we have that \( \mu' \) is \( \mu \), but the inference is possible only if \( \mu \in \{ \alpha, \alpha^{-}\} \); in \( \text{rel} \) \( \mu' \) is \( \phi(\mu) \) and in the first two rules of \( \text{com} \) \( \mu' \) is simply \( \mu \). Relation \( R \) is accordingly modified. The third rule for \( \text{com} \) is the synchronization rule; of course it takes care that relations \( R \) and \( R' \) are (modified and) unioned.

A derivation generated by the first implication of rule \( \text{sum} \) can be understood as consisting of two steps. Starting from the singleton \( (E_1 + E) \) the first step discards alternative \( E \) and decomposes \( E_1 \) into the union of suitable sets of grapes \( I_1 \) and \( I_3 \); the second step (the condition of the inference rule) rewrites \( I_1 \) as \( I_2 \), leaving \( I_3 \) idle (see also Fig. 1.2.a). The grapes in \( I_3 \) are originated by \( E_1 + E \) but not caused by \( \mu \), so we add \( (E_1 + E) \leq I_3 \) to \( R \). Moreover, all the grapes which are caused by some grape in \( \text{dec}(E_1) \cdot I_3 \) and not by \( \mu \) (namely, those in \( R \cdot I_2 \)) are still caused by \( (E_1 + E) \) and not by \( \mu \), thus we also have that \( (E_1 + E) \leq (I_3 \cdot R \cdot I_2) \) is in \( R \). The net effect of these two steps is then rewriting the singleton \( (E_1 + E) \) into the set \( I_2 \) and labelling the arrow with the pair \( [\mu, (E_1 + E) \leq (I_3 \cdot R \cdot I_2)] \). Analogously for the second rule of \( \text{sum} \).

The intuition behind rule \( \text{rec} \) is similar to that behind \( \text{sum} \). Obviously, the first step consists now in unwinding the recursive agent in all the occurrences of the bound variable, rather than discarding one of the alternatives.

The way we deal with nondeterministic choice and recursion shows that our transition rules still assume a centralised control. For instance, all the concurrent sequential processes which occur in an argument of \( + \) must participate in and are affected by the decision.

![Figure 3.1](image-url)

The graphical representation of \( \text{rec } x \cdot \alpha x \rightarrow (\alpha, \emptyset) \rightarrow \text{rec } x \cdot \alpha x \) (in a),
and of \( (g_1, g_2) \rightarrow (\tau, (g_1, g_4, g_5, g_6, g_7)) \rightarrow (g_6, g_7) \) (in b).

The following property clarifies the structure of the derivations and stresses the asynchrony of the partial ordering derivation relation. Indeed, the underlying model of standard CCS derivations is a transition system, while Definition 3.4 introduces a rewriting system. As a matter of fact, \( E_1 \rightarrow \mu \rightarrow E_2 \) is a transition, i.e., \( E_1 \) and \( E_2 \) are global states, while \( I_1 \rightarrow [\mu, R] \rightarrow I_2 \) can be interpreted as a rewriting rule, since the grapes involved there are only those processes of the current state which are active in the step. The correspondence between the two derivation relations is stated in Theorem 3.1.
Property 3.2.
Given $I_1 - [\mu, R] \rightarrow I_2$ in the partial ordering derivation relation, we have
- $R \downarrow 1 \subseteq I_1$;
- $R \downarrow 2 \cap I_2 = \emptyset$;
- for every set of grapes $I$, $I_1 \cup I$ is complete if and only if $I_2 \cup R \downarrow 2 \cup I$ is complete.

Proof. Immediate by induction.

Example 3.3. Let us consider the agent of Example 3.2

$E_1 = ((\alpha.NIL \land \gamma.NIL + \theta.NIL) \land (\alpha : \gamma.NIL \land \delta.NIL + \nu.NIL)) \land \beta.NIL$
and the agent

$E = \eta.NIL$.
Furthermore, let

$g_1 = ((\alpha.NIL \land \gamma.NIL + \theta.NIL)\langle id\rangle\langle id\rangle)$;
$g_2 = (\langle id\rangle(\alpha.NIL \land \gamma.NIL + \theta.NIL)\langle id\rangle\langle id\rangle)$;
$g_3 = \langle id\rangle\langle id\rangle\beta.NIL$;
$g_4 = (\langle id\rangle\langle id\rangle\gamma.NIL)\langle id\rangle\langle id\rangle$;
$g_5 = (\langle id\rangle\langle id\rangle\delta.NIL)\langle id\rangle\langle id\rangle$;
$g_6 = (\langle id\rangle\langle id\rangle\nu.NIL)\langle id\rangle\langle id\rangle$;
$g_7 = (\langle id\rangle\langle id\rangle\eta.NIL)\langle id\rangle\langle id\rangle$;

and

$R = \{g_1 \leq g_2, g_2 \leq g_5\}$

By using the first inference rule sum, from the partial ordering derivation

$\{g_1, g_2\} \rightarrow [\tau, R] \rightarrow \{g_6, g_7\}$, with $R = \{g_1 \leq g_2, g_2 \leq g_5\}$ (Fig. 3.2.a)
we can deduce

$\{E_1 + E\} \rightarrow [\tau, (E_1 + E) \leq (g_3 \cup R \downarrow 2)] \rightarrow \{g_6, g_7\}$ (Fig. 3.2.b)

In fact, we have that $\text{dec}(E_1) = \{g_1, g_2, g_3\}$ (see Example 3.2), thus $I_3$ contains $g_3$, and
$R \downarrow 2 = \{g_4, g_5\}$.

![Figure 3.2.](image)

The partial ordering derivations $\text{dec}(E_1) \cdot (g_3) = \{g_1, g_2\} \rightarrow [\tau, (g_1 \leq g_4, g_2 \leq g_5)] \rightarrow \{g_6, g_7\}$ (in part a),
and $\{E_1 + E\} \rightarrow [\tau, (E_1 + E) \leq (g_3, g_4, g_5)] \rightarrow \{g_6, g_7\}$ (in part b).

Theorem 3.1. (correspondence between Milner’s and partial ordering derivations)
We have a derivation $E_0 - [\mu, R] \rightarrow E_1$ if and only if there exist a relation $R$ and a set of grapes $I$ such that $\text{dec}(E_0) \cdot I \rightarrow [\mu, R] \rightarrow \text{dec}(E_1) \cdot (R \downarrow 2 \cup I)$.
Proof. Given a derivation of either kind, use the structure of its deduction to obtain the derivation of the other kind.

4. Partial Ordering Many Step Derivations

In this section we concatenate the derivations given in Section 3 to define computations from which the partial ordering many step derivations for CCS are obtained. The partial orderings of events of these derivations express the complete causal dependencies among the performed events. In order to relate our many step derivations with Milner’s, we also introduce total orderings on events that reflect the temporal relation among them. Eventually, the two relations of partial ordering observational equivalence and congruence are defined which are based on bisimulation and on the previously given many step derivations.

The next definition introduces three orderings of events which will be used to capture the relevant information about behaviours of agents.

**Definition 4.1. (orderings of events)**

Let $A$ be a countable set of event labels.

i) A *partial ordering (po)* of events is a triple $h = \langle S, 1, \leq \rangle$, where
- $S$ is a finite set of events;
- $1: S \to A$ is a labelling function;
- $\leq$ is a partial ordering relation on $S$, called *causal relation*.

ii) A *total ordering (to)* of events is a po of events $t = \langle S, 1, \leq \rangle$ such that $\leq$ is total. In this case we will use $<$ for $\leq$, and call it *temporal relation*.

iii) A *mixed ordering (mo)* of events is a quadruple $d = \langle S, 1, \leq, < \rangle$, where $\langle S, 1, \leq \rangle$ is a po and $\langle S, 1, < \rangle$ is a to of events.

Two events $e_1$ and $e_2$ are *concurrent* if neither $e_1 \leq e_2$ nor $e_2 \leq e_1$.

Two orderings of events will be identified if isomorphic, i.e., if there is a label- and order-preserving bijection between their events.

A to of events (up to isomorphism) will be identified with the sequence of the labels of its events.

Figure 4.1. shows a partial ordering of events, with the conventions that events are represented by circles with their labels inside, and that the partial ordering $\leq$ is represented by its Hasse diagram growing downwards. So, we have that event $e_1$, labelled by $\alpha$, has no relation with all the others, thus it is concurrent with them all. The event labelled by $\gamma$ dominates, i.e., *causes* the remaining events. Note that the labelling function is not injective.
We will now introduce our notion of computation, as a finite sequence of complete sets of grapes and of partial ordering derivations.

Definition 4.2. (computation)
A sequence

\[ \xi = (G_0, I_1 \rightarrow \mu_1, R_1 \rightarrow \Gamma_1, G_1, \ldots, G_{n-1}, I_n \rightarrow \mu_n, R_n \rightarrow \Gamma_n, G_n) \]

is a computation if

i) • \( G_i \) is a complete set of grapes, \( 0 \leq i \leq n \), and
   • \( I_1 \rightarrow [\mu_1, R_1] \rightarrow \Gamma_1 \) is in the partial ordering derivation relation, \( 0 < i \leq n \);

ii) • \( I_i \subseteq G_{i-1} \), and
   • \( G_i = (G_{i-1} - I_i) \cup R_i \downarrow 2 \cup \Gamma_i, 0 < i \leq n. \)

As noted in the previous section, the elements of the partial ordering derivation relation are rewriting rules which are applied in the computation above. States are (represented as) complete sets of grapes. This is essentially due to our assumption of having a centralized control. Indeed, function \( dec \) induces and Theorem 3.1 establishes this natural one-to-one correspondence between the states of the original interleaving and of the partial ordering computations. In the \( i \)th step, state \( G_{i-1} \) evolves to \( G_i \) by applying \( I_1 \rightarrow [\mu_1, R_1] \rightarrow \Gamma_1 \) in such a way that the set of grapes \( I_i \) (contained in \( G_{i-1} \)) is rewritten as \( R_i \downarrow 2 \cup \Gamma_i \), while the grapes in \( G_{i-1} - I_i = G_i - (R_i \downarrow 2 \cup \Gamma_i) \) stay idle.

Note also that our notion of computation coincides with Milner's, when a single step is performed, because of the correspondence between his and our derivation relation established by Theorem 3.1. Hence, hereto we will write \( E_0 \rightarrow \mu \rightarrow E_1 \) to denote a single-step computation of either nature.

From computations we generate a mixed ordering of events recording all their temporal and causal dependencies. This ordering is obtained in three steps: first an event (labelled by \( \mu \)) is associated to every derivation (labelled by \( \mu \)) with the obvious causal dependencies; then the grapes (of two successive states) which stay idle in a derivation are identified; and finally all grapes and all events labelled by \( \tau \) are removed to get the wanted ordering.
Definition 4.3. (initial derivation)

Given two CCS agents $E_0$ and $E_1$, we define the initial derivation, written

$$E_0 = \phi \Rightarrow E_1,$$

iff there exist a computation

$$\xi = \{G_0 \ I_1 \rightarrow [\mu_1, \mathcal{R}_1] \rightarrow I_1' \ G_1 \ldots G_{n-1} \ I_{n-1} \rightarrow [\mu_n, \mathcal{R}_n] \rightarrow I_n' \ G_n\}$$

where

- $G_0 = \text{dec}(E_0)$;
- $G_n = \text{dec}(E_1)$;
- $d = \langle S, I, \leq, < \rangle$ is the mo of events labelled on $A$, generated by the following procedure $PO$.

**Procedure PO($\xi$)**

1. Consider the set of instances of grapes $\{g_{j,i} \mid g_j \in G_i\}$;
2. For every occurrence $I_i \rightarrow [\mu_i, \mathcal{R}_i] \rightarrow I'_i$, generate an event $e_i$ and let
   
   $$g_{j,i-1} \leq e_i \leq g_{k,i},$$
   
   for all $g_j \in I'_i$ and $g_k \in I_i$;
   $$g_{j,i-1} \leq g_{k,i},$$
   
   for all $g_j, g_k \in \mathcal{R}_i$;
3. Identify $g_{j,i-1}$ with $g_{j,i}$ for all $g_j \in I_i$;
4. Close reflexively and transitively $\leq$;
5. Let $l(e_i) = \mu_i$; $S = \{e_i \mid l(e_i) \neq \tau\}$; restrict $\leq$ on $S$; and, for all $e_i, e_j \in S$, let $e_i \leq e_j$, $1 \leq i \leq n$.

An initial derivation contains complete information about the evolution of agents. In particular, it records the initial and final agents, the performed events, their generation ordering (expressed through $<$), and their causal dependencies (expressed through $\leq$). This information is extracted from a computation $\xi$ by Procedure PO. This procedure considers the different occurrences of grapes and of derivations in $\xi$ as different objects. The second index $i$ of $g_{j,i}$ belonging to the $i$th complete set of grapes is used to this purpose. In correspondence with the $i$th derivation, the procedure generates an event $e_i$. So, the indexes of events can be used to recover at once the temporal dependencies. The grapes that are not rewritten by a derivation, and thus occur in two successive complete sets of grapes, are identified, since they stand for the same sub-agent. The causal relation $\leq$ is then obtained by closing reflexively and transitively the causal orderings in the derivations. The mixed ordering of events labelling the initial derivation is eventually determined by keeping only the events labelled by visible actions, and by restricting the orderings on them. Below, Example 4.1 shows an application of Procedure PO.

Before giving the example, we state a fundamental theorem about our operational semantics of CCS, and derive from initial derivations the notions of partial ordering and interleaving many step derivations. The property expressed by Theorem 4.1 (called complete concurrency) relates the total and the partial orderings obtained from computations. More precisely, the first part of the theorem states that, given a computation, the events in the derived mixed ordering of events $\langle S, I, \leq, < \rangle$ are generated in a total temporal ordering that is, of course, compatible with the causal ordering ($\leq \subseteq <$). The second and crucial part says that these events can be generated by
different computations (with the same initial and final set of grapes) in all temporal orderings $\prec$ compatible with the causal one ($\leq \subseteq \prec$), namely $\leq$ is complete. Shortly, completeness amounts to saying that any two concurrent events can be generated in either temporal order. As we will see later, complete concurrency plays a crucial rôle in relating the notions of partial ordering many step derivations with Milner's, and therefore in proving that partial ordering observational equivalence and congruence are finer than Milner's.

**Notation.**

Given a partial ordering and a total ordering of events $\langle S, 1, \leq \rangle$ and $\langle S, 1, \prec \rangle$, we write $\leq \subseteq \prec$ if $s \leq s'$ implies $s < s'$ or $s = s'$.

**Theorem 4.1. (complete concurrency)**

Given two CCS agents $E_0$ and $E_1$ and a mo of events $d = \langle S, 1, \leq, \prec \rangle$ such that $E_0 \equiv d \Rightarrow E_1$, we have that

- $\leq \subseteq \prec$;
- $\forall \prec' \text{ such that } \leq \subseteq \prec', \text{ there exists an initial derivation } E_0 \equiv d' \Rightarrow E_1$,
  with $d' = \langle S, 1, \leq, \prec' \rangle$.

**Proof.** The proof of the first claim is immediate, the proof of the second one is reported in Appendix A.

A further abstraction step can now be performed, by considering derivations where only the temporal or the causal relations are kept.

**Definition 4.4. (po and interleaving many step derivations)**

Given two CCS agents $E_0$ and $E_1$, we have

$E_0 \equiv h \Rightarrow E_1$, \hspace{1em} (called partial ordering (po) many step derivation)

and

$E_0 \equiv \Rightarrow E_1$, \hspace{1em} (called interleaving many step derivation)

iff there exists an initial derivation

$E_0 \equiv d \Rightarrow E_1$,

with $d = \langle S, 1, \leq, \prec \rangle$, $h = \langle S, 1, \leq \rangle$ and $t = \langle S, 1, \prec \rangle$.

It is worth noting that we build our many step derivations by composing elementary steps and then abstracting. This differs from the approach of [BC87] and [BS86], where transition systems are used to directly associate partial orderings to agents: the notion of elementary step and the possibility of growing computations from them are lost in favour of a more denotational flavour.

The next example shows the rôle of relation $\mathcal{R}$ (see the third derivation of the computation) and how causal dependencies are transmitted through $\tau$, and in general how partial ordering of events are obtained from computations.
Example 4.1.
Consider the CCS agent 

\[ E = \alpha.NIL \parallel (\beta.NIL \parallel \gamma.( (\beta.\beta.NIL \parallel \eta.NIL + \theta.NIL) \parallel \delta.NIL)); \]

the grapes

\[ g_0 = \alpha.NIL\text{idle}; \quad g_1 = \text{idl}(\beta\text{NIL}\text{idle}); \]
\[ g_2 = \text{idl}(\text{idl}\gamma.( (\beta.\beta.NIL \parallel \eta.NIL + \theta.NIL)\parallel \delta.NIL)); \quad g_3 = \text{NIL}\text{idle}; \]
\[ g_4 = \text{idl}(\text{idl}\gamma.( (\beta.\beta.NIL \parallel \eta.NIL + \theta.NIL)\parallel \delta.NIL)\text{idle}); \quad g_5 = \text{idl}(\text{idl}\delta.NIL); \]
\[ g_6 = \text{idl}(\text{NIL}\text{idle}); \quad g_7 = \text{idl}(\text{idl}(\beta.NIL\text{idle})\text{idle})\];
\[ g_8 = \text{idl}(\text{idl}\eta.NIL\text{idle})\];
\[ g_9 = \text{idl}(\text{idl}\text{NIL})\];
\[ g_{10} = \text{idl}(\text{idl}(\text{NIL}\text{idle})\text{idle})\];

and the complete sets of grapes.

\[ G_0 = \{ g_0, g_1, g_2 \}; \quad G_1 = \{ g_3, g_1, g_2 \}; \]
\[ G_2 = \{ g_3, g_1, g_4, g_5 \}; \quad G_3 = \{ g_3, g_6, g_7, g_8, g_5 \}; \]
\[ G_4 = \{ g_3, g_6, g_7, g_8, g_9 \}; \quad G_5 = \{ g_3, g_6, g_10, g_8, g_9 \}; \]
\[ G_6 = \{ g_3, g_6, g_{10}, g_{11}, g_9 \}. \]

We have that \( G_0 = \text{dec}(E), \) from which the following computation will start.

\[ \xi = \{ G_0 \quad \{ g_0 \} \rightarrow [\alpha, \emptyset] \rightarrow \{ g_3 \} \quad G_1 \quad \{ g_2 \} \\
\quad \{ g_1, g_4 \} \rightarrow [\gamma, \emptyset] \rightarrow \{ g_4, g_5 \} \quad G_2 \}
\]
\[ G_3 \quad \{ g_5 \} \rightarrow [\delta, \emptyset] \rightarrow \{ g_9 \} \quad G_4 \]
\[ \{ g_7 \} \rightarrow [\beta, \emptyset] \rightarrow \{ g_{10} \} \quad G_5 \quad \{ g_8 \} \rightarrow [\eta, \emptyset] \rightarrow \{ g_{11} \} \quad G_6 \].\]

Computation \( \xi \) generates the following po and interleaving many step derivations

\[ \alpha.NIL \parallel (\beta.NIL \parallel \gamma.( (\beta.\beta.NIL \parallel \eta.NIL + \theta.NIL) \parallel \delta.NIL) \Rightarrow h = \Rightarrow \]
\[ \text{NIL} \parallel (\text{NIL} \parallel ((\text{NIL} \parallel \text{NIL}) \parallel \text{NIL})), \]
\[ \alpha.NIL \parallel (\beta.NIL \parallel \gamma.( (\beta.\beta.NIL \parallel \eta.NIL + \theta.NIL) \parallel \delta.NIL) \Rightarrow t = \Rightarrow \]
\[ \text{NIL} \parallel (\text{NIL} \parallel ((\text{NIL} \parallel \text{NIL}) \parallel \text{NIL})), \]

where \( h = \langle S, l, \leq \rangle \) and \( t = \langle S, l, < \rangle, \) with

- \( S = \{ e_1, e_2, e_4, e_5, e_6 \}; \)
- \( l(e_1) = \alpha, \ l(e_2) = \gamma, \ l(e_4) = \delta, \ l(e_5) = \beta, \ l(e_6) = \eta; \)
- \( e_2 \leq e_4, \ e_2 \leq e_5, \ e_2 \leq e_6, \ e_1 \leq e_1; \)
- \( e_1 < e_2 < e_4 < e_5 < e_6. \)

Figure 4.2 depicts the computation \( \xi; \) Figure 4.3.a) is an intermediate snapshot in getting the po many step derivation (after the execution of Step 3 of Procedure PO); Figure 4.3.b) shows h.
Figure 4.2.
A graphical representation of the computation in Example 4.1.

Figure 4.3.
An intermediate step (part a) in getting the po (part b) of the many step derivation of Ex. 4.1.
The identified instances of grapes $g_{j,i-1}$ and $g_{j,i}$ are depicted as $g_{j,i-1}$.
Another example follows which shows how unguarded recursion is naturally dealt with in our framework. It also gives evidence that unguardedness may lead to infinitely branching partial orderings that reflect unbounded parallelism.

**Example 4.2.** Consider the unguarded recursive agent  \( \text{rec } x. \alpha.\text{NIL}lx \). It originates computation

\[
\xi = \{ \text{rec } x. \alpha.\text{NIL}lx \}
\]

\[
\begin{align*}
\text{rec } x. \alpha.\text{NIL}lx &\rightarrow \{ \alpha, \text{rec } x. \alpha.\text{NIL}lx \leq \text{idlrec } x. \alpha.\text{NIL}lx \} \rightarrow \text{NILlid} \\
\text{NILlid}, \text{idlrec } x. \alpha.\text{NIL}lx &\rightarrow \{ \alpha, \text{idlrec } x. \alpha.\text{NIL}lx \leq \text{idl(idlrec } x. \alpha.\text{NIL}lx) \} \rightarrow \text{idl(NILlid)} \\
\text{NILlid}, \text{idl(NILlid)}, \text{idl(idlrec } x. \alpha.\text{NIL}lx) &\rightarrow \ldots \\
\text{idl(NILlid)} &\rightarrow \text{NILlid}, \text{idl(NILlid)}, \text{idl(idlrec } x. \alpha.\text{NIL}lx) \\
\end{align*}
\]

(Note that, according to Milner, we would have

\[
\text{rec } x. \alpha.\text{NIL}lx \rightarrow \text{NILlid}, \text{idlrec } x. \alpha.\text{NIL}lx \rightarrow \text{NILlid}, \text{idlrec } x. \alpha.\text{NIL}lx.
\]

We have that rec x. \( \alpha.\text{NIL}lx = h \Rightarrow \text{NILlid}, \text{idlrec } x. \alpha.\text{NIL}lx \), where po of events h consists of two concurrent events labelled by \( \alpha \). Figure 4.4 shows an intermediate step in getting h.

![Figure 4.4](image)

**Figure 4.4.** An intermediate step in getting the po of the many step derivation of Example 4.2.

We already noted that, due to the assumption of centralized control, there is a natural bijection between the states of the interleaving and of the partial ordering computations: the states of partial ordering computations are all and only complete sets of grapes, i.e., decompositions of CCS agents. The following theorem shows that interleaving many step derivations essentially coincide with Milner's, since total orderings of events are considered as the sequences of the labels of their events.

**Theorem 4.2.** *(deriving Milner's many step derivations from initial ones)*

Given two CCS agents E and E', we have Milner's many step derivation

\[
E = s \Rightarrow E'
\]

if and only if there exists an interleaving many step derivation

\[
E \equiv t \Rightarrow E' \text{ and } t = s.
\]
Proof. Let
\[ \xi = (G_0 \rightarrow \{\mu_1, \mathcal{S}_1\} \rightarrow I_1 \ G_1 \ldots G_{n-1} \ I_n \rightarrow \{\mu_n, \mathcal{S}_n\} \rightarrow I_n \ G_n) \]
be the computation such that \( E \models \Rightarrow E' \) holds. We build, by inducting on the length of \( \xi \), a sequence of Milner's derivation rules
\[ E = E_0 \rightarrow \mu_1 \rightarrow E_1 \rightarrow \mu_2 \rightarrow E_2 \ldots E_{n-1} \rightarrow \mu_n \rightarrow E_n = E' \]
such that \( E \models \Rightarrow E' \).
When there is no derivation, the claim follows trivially. Assume inductively that the thesis holds at the \( i \)th step: we have then that there exists an agent \( E_{i-1} \) such that \( G_{i-1} = \text{dec}(E_{i-1}) \). By Theorem 3.1 and by definition of computation, there exists a set of grapes \( \Gamma \) such that
\[ G_{i-1} \cdot \Gamma = I_i, G_i \cdot (\mathcal{S}_i \downarrow 2 \cup \Gamma) = I'_i, \text{ and } I_i \rightarrow [\mu_i, \mathcal{S}_i] \rightarrow I'_i \text{ if } E_{i-1} \rightarrow \mu_i \rightarrow E_i, \text{ with } \text{dec}(E_i) = G_i. \]
Since both many step derivations forget \( \tau \)'s, the proof of the inductive step follows.
The proof of the only if part is symmetric.

A consequence of the two theorems above is that Milner's many step derivations can be easily recovered from po many step derivations, the former being just linearizations of the latter. In other words, the original interleaving operational semantics for CCS is immediately retrievable from ours, since there is a syntactical bijection between the two kinds of derivation. This very basic correspondence guarantees that it is possible to carry over the partial ordering approach to the extensional semantics for CCS defined so far (e.g., see [Mi80 and 84, DH84]). The retrievability of Milner's many step derivations from ours is precisely stated by the following theorem.

**Theorem 4.3.** *(Milner's many step derivations are interleavings of po many step derivations)*
Given two CCS agents \( E_0 \) and \( E_1 \),

i) if there exists a po many step derivation
\[ E_0 \models h \Rightarrow E_1, \text{ where } h = <S, l, \leq>, \]
then, for all \( s = <\mathcal{S}, l, \leq> \) such that \( \leq \subseteq < \), we have Milner's many step derivation
\[ E_0 \models s \Rightarrow E_1; \]

ii) for all Milner's many step derivation
\[ E_0 \models s \Rightarrow E_1, \]
there exists a po of events \( h = <\mathcal{S}, l, \leq> \), with \( \leq \subseteq < \) (\( s \) is considered as to of events \( <\mathcal{S}, l, \leq> \)), such that
\[ E_0 \models h \Rightarrow E_1. \]

**Proof.** Theorem 4.2 relates Milner's many step derivations with our interleaving many step derivation. Then, Theorem 4.1 suffices to prove the claim.

So far, we have abstracted from actual computations to get many step derivations by forgetting the intermediate states of computations, the actual temporal ordering in which their events have been generated, and the events labelled by \( \tau \). Still, two agents syntactically different are considered as semantically different, also when their observable behaviour is the same. Thus, we will further abstract from the syntactic structure of agents by defining an equivalence relation over
them. Once more, our partial ordering observational equivalence is based on the notion of bisimulation, but this time it is defined in terms of partial orderings ($E_1 * E_2$) rather than sequences of actions ($=s=>$). The following definition thus rephrases Definition 2.3.

**Definition 4.5. (po observational equivalence)**

1. If $R$ is a binary relation between CCS agents and $h$ is a po of events, then $\Theta$, a function from relations to relations, is defined as follows: \( E_1, E_2 \in \Theta(R) \)

   i) whenever $E_1 = h => E_1'$ there exists $E_2'$ such that $E_2 = h => E_2'$ and $\langle E_1, E_2' \rangle \in R$

   ii) whenever $E_2 = h => E_2'$ there exists $E_1'$ such that $E_1 = h => E_1'$ and $\langle E_1', E_2' \rangle \in R$.

2. A relation $R$ is a **bisimulation** if $R \subseteq \Theta(R)$.

3. Relation $\equiv = \cup \{ R | R \subseteq \Theta(R) \}$, is called partial ordering (po) observational equivalence.

**Proposition 4.1.**

- Function $\Theta$ is monotonic on the lattice of relations under inclusion.
- Relation $\equiv$ is a bisimulation and an equivalence relation.

**Example 4.3.** It is easy to verify, for every agent $E$, that

a) the following equivalences hold

   i) $\alpha.E \equiv \alpha.\tau.E$

   ii) $\alpha.E \equiv \tau.\alpha.E$

   iii) $\alpha.(\beta.E + \tau.\gamma.E) + \alpha.\gamma.E \equiv \alpha.(\beta.E + \tau.\gamma.E)$

   (note that (i) and (iii) are two of the $\tau$-laws of [Mil80]);

b) it is not true that $\alpha.NIL \equiv \beta.NIL \equiv \alpha.\beta.NIL + \beta.\alpha.NIL$.

Not surprisingly, the above defined partial ordering observational equivalence is finer than observational equivalence.

**Theorem 4.4. (po equivalence is finer than observational equivalence)**

Given two CCS agents $E_0$ and $E_1$, we have that

$E_0 \equiv E_1$ implies $E_0 \approx E_1$, but not vice versa.

**Proof.** To show that $\equiv$ implies $\approx$ it suffices proving that, given any bisimulation relation $R_{po}$ based on po of events and such that $\langle E_1, E_2 \rangle \in R_{po}$, it is possible to define a new relation $R_{int}$ based on to of events and such that $\langle E_1, E_2 \rangle \in R_{int}$ and $R_{int} \subseteq \Theta(R_{int}).$

This is easily done, since $E \equiv h => E'$ implies $E \equiv h => E'$, for all $t \supseteq h$, by Theorem 4.1. Thus, we can choose $R_{int}$ to be $R_{po}$ itself.

The claim follows, by applying Theorem 4.2 which establishes the one-to-one correspondence between interleaving many step derivations and Milner's.

**Example 4.4.b** (together with Theorem 4.2) shows that $E_0 \approx E_1$ does not imply $E_0 \equiv E_1$. 

23
Theorem 4.5.
Partial ordering observational equivalence is preserved by all operators except for +.

Proof. See Appendix B.

We will now refine the notion of po observational equivalence so that the new relation is preserved under all contexts. The following definition follows the pattern of the context independent characterization of observational congruence given in Section 2. Again, two agents are congruent if they are equivalent and, whenever one may perform at least a τ, the other may do as well, becoming equivalent agents. We need a definition first; recall that \( E_0 \rightarrow^\tau E_1 \) denotes a single-step computation in which action \( \tau \) has been performed.

Definition 4.6. (non empty sequences of silent transitions)

\[ E \equiv \tau \Rightarrow E' \text{ if and only if there exist } E_0, E_1, 0 < i < n, \text{ such that } E = E_0, E' = E_n, \text{ and } E_i \rightarrow^\tau E_{i+1}. \]

Definition 4.7. (po congruence)

Two CCS agents \( E_0 \) and \( E_1 \) are partial ordering (po) observational congruent, written as \( E_0 \equiv_c E_1 \), if and only if

i) \( E_0 \equiv E_1 \);

ii) whenever \( E_0 \equiv \tau \Rightarrow E_0' \), there exists an agent \( E_1' \) such that \( E_1 \equiv \tau \Rightarrow E_1' \) and \( E_0' \equiv E_1' \);

iii) whenever \( E_1 \equiv \tau \Rightarrow E_1' \), there exists an agent \( E_0' \) such that \( E_0 \equiv \tau \Rightarrow E_0' \) and \( E_0' \equiv E_1' \).

Theorem 4.6. (\( \equiv_c \) is preserved by all contexts)
Relation \( \equiv_c \) is a congruence.

Proof. Let \( E, E_0, E_1 \) be CCS agents. The proof proceeds by case analysis on the operators of CCS, under the hypothesis that there exists a bisimulation relation \( R \) containing the pair \( \langle E_0, E_1 \rangle \), i.e., \( E_0 \equiv_c E_1 \).

The proof in cases act), res), rel), com) and rec) is immediate since item i) has been established by Theorem 4.5, and proving items ii) and iii) is trivial.

sum) The only difficult part of proving that \( E_0 + E \equiv_c E_1 + E \) (and that \( E + E_0 \equiv_c E + E_1 \)) is showing that \( E_0 + E \equiv E_1 + E \), i.e., when \( E_0 + E \equiv \tau \Rightarrow E_0' \) also \( E_1 + E \equiv \tau \Rightarrow E_1' \) with \( E_0' \equiv E_1' \); and viceversa. When \( E_0 \) moves via visible actions, the proof is trivial. When \( E_0 \equiv \tau \Rightarrow E_0 \), also \( E_1 \equiv \tau \Rightarrow E_1 \), for \( E_0 \equiv_c E_1 \) by hypothesis (in particular, note that item ii) holds); and viceversa.

As expected, partial ordering congruence is finer than observational congruence; furthermore they coincide when dealing with nondeterministic sequential processes only.
Corollary 4.1. (po congruence is finer than observational congruence)
Given two CCS agents $E_0$ and $E_1$, we have that

$E_0 \equiv_{po} E_1$ implies $E_0 \equiv_{obs} E_1$, but not viceversa.

Proof. The implication follows from Theorems 4.2, 4.3 and 4.4. Example 4.4.b) shows also that the reverse implication does not hold.

Not surprisingly, the equivalence and congruence relations coincide with the original relations introduced in [Mil84] when they are restricted to sequential nondeterministic processes.

Theorem 4.7. (po congruence and observational congruence coincide on sequential processes)
Let $SEQ$ be the set of CCS agents in which $I$ does not occur.
The restriction of $\equiv_{po}$ to $SEQ \times SEQ$ coincides with the restriction of $\equiv_{obs}$ to $SEQ \times SEQ$.

Proof. If $E_0$ is in $SEQ$ and $E_0 =_{s} \Rightarrow E_1$, then $\text{dec}(E_0)$ is a singleton and such are all the intermediate sets of grapes in the computation. All partial ordering many step derivations $E_0 =_{s} S, I, \leq_{s} \Rightarrow E_1$ are such that $\leq$ is a total ordering of events. Thus, we have that $E_0 \equiv_{po} E_1$ iff $E_0 =_{obs} E_1$.

5. Conclusions and Related Work

In this paper we have presented a partial ordering semantics for CCS based on a set of rewriting rules, given in the SOS style, and on a notion of observational congruence. A rewriting rule describes the evolution of sets of sequential sub-agents which are obtained by decomposing CCS agents, and expresses the causal relation among the initial sub-agents, the performed action, and the resulting sub-agents. The congruence abstracts from internal behaviour still distinguishing concurrent execution of actions from their nondeterministic interleavings, and preserving information about the causal relation among them.

To make the choice of a particular true concurrent semantics less arbitrary, in [DDM87c] we put forward two criteria we consider essential for assessing any new partial ordering semantics of a language previously equipped with an interleaving one:

i) the interleaving semantics must be retrievable from the partial ordering semantics;

ii) the partial ordering semantics must capture all and only the parallelism present in the language, as expressed, e.g., through a multiset operational semantics.

In this section, we will discuss adequacy of our semantics and its relationships with other work aiming at the same target, by checking whether they satisfy the above criteria, and by discussing the discriminating power of the proposed behavioural equivalences.
Theorem 4.3 guarantees that our semantics satisfies criterion i). It should be noted that there indeed exists a direct syntactic correspondence between agents and the sets of grapes reachable through derivations, between Milner's rules and ours, and, finally, between the proofs of either derivations. In fact, criterion i) is shown to hold by a straightforward structural induction. We would like to stress that another by-product of the direct correspondence is that proof techniques developed for the interleaving approach can be borrowed.

We have not proved criterion ii), but we claim it. The actual proof requires introducing, as done in [DDM87c], a multiset transition system where transitions are labelled by multisets of actions, rather than single actions. The new transition system makes explicit the concurrency of CCS agents by describing the effect of performing concurrent actions at the same time. The multiset operational semantics can be defined by extending and modifying the inference rule for communication between agents, so that a multiset of actions could be performed and pairs of complementary actions could be synchronized (see for something on this line [Mil83, AB84]). Once multiset transitions have been defined, the proof that criterion ii) is met requires a long and tedious work, analogous to the one followed in establishing the relationships between partial ordering and interleaving semantics. More precisely, one has to define a mixed ordering containing also sequences of multisets of actions; to prove the multiset counterpart of complete concurrency; and to eventually prove that the partial ordering equivalence implies the multiset one.

We claim also that multiset equivalence is coarser than the partial ordering one, since it does not respect causal dependencies, as shown by the following example.

Agents
\[ \alpha.NIL|\beta.NIL + \alpha.\beta.NIL \text{ and } \alpha.NIL|\beta.NIL \]
are multiset equivalent, but they are not partial ordering equivalent.

Although this example and Theorem 4.7 may support partial ordering congruence as the basis for true concurrent semantics, there are also grounds on which this congruence may not be considered completely satisfactory. Indeed, it does not completely respect branching time. For example the following agents are partial ordering congruent
\[ \alpha.\beta.NIL|\beta'.(\gamma.NIL+\delta.NIL) + \alpha.NIL|\gamma.NIL + \alpha.\gamma.NIL \]
\[ \alpha.\beta.NIL|\beta'(\gamma.NIL+\delta.NIL) + \alpha.NIL|\gamma.NIL. \]
Nevertheless, the first agent may cause via an \( \alpha \) either (\( \gamma.NIL+\delta.NIL \)) or just \( \gamma.NIL \), while the second has no choice.

When branching time is felt as important, an alternative approach can be followed. In [DDM87a], we introduced Nondeterministic Measurement Systems (NMS) and defined a bisimulation relation over them. An NMS is the tree consisting of the computations of a transition system ordered by prefix; its nodes are labelled by what is observed of the corresponding computation. Now, if we take as observations (a slight variation of) the partial orderings of events labelling the derivations and use bisimulation over NMS, we obtain an observational congruence which respects branching time. Indeed, the two agents above would differentiated by this NMS congruence. It is not difficult to prove that the alternative congruence suggested above is finer than the one defined in this paper; the additional discriminating power comes from the information about the structure of the computation NMS record.
There have been many attempts to define a partial ordering operational semantics for CCS. In many cases, however, either proper subsets of CCS have been considered or the interleaving semantics is not the standard CCS one. Our attempts [DDM85, 87a, 87b and 87c] have been already summarized in the Introduction. De Cindio et al. [DDPS83] map into a subclass of Petri Nets a version of CCS which does not allow generation of unboundedly many agents in parallel, like rec x. α.NIL | β.x. Goltz and Mycroft [GM84] give a denotational semantics of CCS in terms of Occurrence Nets and an operational semantics in terms of Place/Transitions Nets which do not satisfy criterion i) (see [DDM87b] for an example). Winskel in [Win82] and in [Win87] proposes two partial ordering denotational semantics for CCS based on Event Structures and on Petri Nets. He claims that the interleaving semantics agrees with Milner's without giving any formal statement of the satisfaction of any criteria similar to i) and ii) above. In [Old87] and other manuscripts, Olderog refines the approach of [DDM87b] to give a distributed account of + and rec; he uses a slightly modified version of our decomposition function and proposes a set of derivation rules very similar to those of [DDM87c] to obtain a partial ordering semantics of a language (CCSP) with many similarities with CCS, but without the restriction operator. Satisfaction of criterion i) is proved, but more involved and less general conditions are stated in place of criterion ii), and not formally proved.

There are also several papers which aim at providing languages traditionally equipped with interleaving based semantics with partial ordering preserving behavioural equivalences. Castellani and Hennessy [CH85] provide a fragment of CCS with a semantics based on rewriting rules and bisimulation. Synchronization and restriction are not considered and only single-step derivations are defined. Their observational equivalence appears incomparable with ours even for the common sub-language. However, the relationships have not yet been fully investigated. In [GV87] van Gabbleek and Vaandrager propose a Petri Net semantics for finite ACP processes and define two congruence relations (pomset and generalized pomset bisimulation) which seem to coincide with our partial ordering and NMS equivalence, respectively. Boudol and Castellani [BC87] consider an algebra of labelled event structures (without restriction and communication) and define a congruence relation which, we feel, coincides with pomset bisimulation and with the congruence introduced in this paper. None of the above operational approaches considers a language with operators for both recursion and restriction. It is not clear to us how and whether their results could be extended to cope with such an explosive mixture.

The results presented in this paper certainly require further improvements and extensions. Obviously, the relationships among the various partial ordering equivalences should be assessed, and other notions of equivalence defined and studied. E.g., it should be worthwhile to extend to true concurrent models those equivalence or pre-order relations already introduced for interleaving models and proved interesting [DH84, OH86]. More generally, criteria are to be established for judging the adequacy and feasibility of equivalence relations for concurrent systems. Also, the search for (complete?) proof systems should continue.
Appendix A

The proof of complete concurrency is based on the following steps.

i) Procedure TPO (Total PO) is defined which extends Procedure PO in that it builds mixed orderings of events containing also events labelled by \( \tau \); we will call the ordering generated in this way observation of the given computation.

ii) We show that, given two consecutive concurrent events originated by a computation, there always exists another computation which generates them in the reverse order. More precisely, given a two-step computation \( \{G_0 I_1 \rightarrow [\mu_1, \mathcal{R}_1] \rightarrow I_1', G_1 I_2 \rightarrow [\mu_2, \mathcal{R}_2] \rightarrow I_2' G_2 \} \), the two events originated by it can also be generated in the reverse order, provided that no grape of \( I_1' \) is used by the second po derivation, namely when \( I_1' \cap I_2 = \emptyset \) (note that \( \mathcal{R}_2 \cup I_2 \cap I_2' \) may be non empty).

iii) The result above is further extended to any set of concurrent events.

iv) We prove that discarding the events labelled by \( \tau \) does not affect the overall result.

Please, recall that two isomorphic orderings of events will be considered identical.

Definition A.1. (observation)

Given two CCS agents \( E_0 \) and \( E_1 \), with \( \text{dec}(E_0) = G_0 \) and \( \text{dec}(E_1) = G_n \), and computation

\[
\xi = \{G_0 I_1 \rightarrow [\mu_1, \mathcal{R}_1] \rightarrow I_1' G_1 ... G_n-1 I_n \rightarrow [\mu_n, \mathcal{R}_n] \rightarrow I_n' G_n \}
\]

we call observation of \( \xi \) the mo of events \( o = \langle \mathcal{S}, I, \leq, \prec \rangle \), labelled on \( \Lambda \cup \{\tau\} \), generated by the following procedure TPO.

Procedure TPO(\( \xi \))

1. Consider the set of instances of grapes \( \{g_{j,i} \mid g_j \in \mathcal{G}_i\} \);
2. For every occurrence \( I_i \rightarrow [\mu_i, \mathcal{R}_i] \rightarrow I_i' \), generate an event \( e_i \) and let
   \[
   g_{j,i-1} \leq e_i \leq g_{k,i} \quad \text{for all } g_j \in \mathcal{G}_i \text{ and } g_k \in I_i';
   \]
   \[
   g_{j,i-1} \leq g_{k,i'} \quad \text{for all } g_j \in I_i \text{ and } g_k \in \mathcal{R}_i';
   \]
3. identify \( g_{j,i-1} \) with \( g_{j,i} \) for all \( g_j \in I_i \);
4. close reflexively and transitively \( \leq \);
5. let \( S = \{e_1, e_2\}; \forall e_j \in S. \text{ let } l(e_j) = \mu_j; \text{ restrict } \leq \text{ on } S; \text{ and } \forall e_i, e_j \in S, \text{ let } e_i < e_j, 1 \leq i+j \leq n. \)

Lemma A.1.

Given a two-step computation

\[
\{G_0 I_1 \rightarrow [\mu_1, \mathcal{R}_1] \rightarrow I_1' G_1 I_2 \rightarrow [\mu_2, \mathcal{R}_2] \rightarrow I_2' G_2 \}
\]

with \( I_1' \cap I_2 = \emptyset \),
i) its observation is \( o = \langle \mathcal{S}, I, \leq, \prec \rangle \), where
   \[
   \mathcal{S} = \{e_1, e_2\} \quad l(e_1) = \mu_1; \quad l(e_2) = \mu_2 \quad e_1 \leq e_1; \quad e_2 \leq e_2 \quad e_1 \leq e_2,
   \]
   i.e., the two events are concurrent;
ii) There always exists a computation

\[
\{G_0 I_2 \rightarrow [\mu_2, \mathcal{R}_2] \rightarrow I_2' G_1 I_1 \rightarrow [\mu_1, \mathcal{R}_1] \rightarrow I_1' G_2 \}
\]

with observation \( o' = \langle I_1', I, \leq, \prec \rangle \), where
   \[
   e_2 < e_1;
   \]
iii) The same causal dependencies are obtained among the elements of \( G_0 \), those of \( G_1 \) and events \( e_1 \) and \( e_2 \) after Step 4 of Procedure TPO applied either to \( \xi \) or \( \xi' \).
Proof. The proof of the first claim is immediate. Items ii) and iii) are proved by induction on the structure (and number of) the grapes belonging to \( R_1 \cup I_2 \), induced by structure of the deduction of \( po \) derivations.

For proving the base step we must consider the following two exhaustive cases.
The first case arises when \( R_1 \cup I_2 = \emptyset \): the two events can be generated in either ordering, since, by hypothesis, \( \Gamma_1 \cap I_2 = \emptyset \). Obviously, item iii) holds true.
The second case may arise only when a recursive agent is considered in which the bound variable occurs unguarded. It might lead to having transitions \( \Gamma_1 \vdash [\mu_1,R_1] \rightarrow \Gamma_1 \), where \( R_1 \cup I_2 \neq \emptyset \).
The representative is \( x.\alpha.\text{x} \) (see Example 4.2) and in this case, \( I_2 \) is obtained by plugging \( I_1 \) into \( x.\alpha.\text{x} \) context. A derivation with at least two unwindings is to be used for obtaining the desired transition. (In the example, we have to deduce

\[
\text{rec } x.\alpha.\text{x} \implies \{\alpha, \text{rec } x.\alpha.\text{x} \leq \{\alpha.\text{id}l(\text{id}l(\text{rec } x.\alpha.\text{x})])\} \rightarrow \text{id}l(\text{NIL}l)l),
\]

and to use it as first step in place of

\[
\text{rec } x.\alpha.\text{x} \implies \{\alpha, \text{rec } x.\alpha.\text{x} \leq \text{id}l\text{rec } x.\alpha.\text{x} \rightarrow \text{NIL}l).
\]

Let us prove the inductive step. We will call equivalent two computations the observation of which differ only in the total ordering, and such that the same causal relation among the elements of \( G_0 \), those of \( G_2 \) and events \( e_1 \) and \( e_2 \) is determined after Step 4 of Procedure TPO.

act). The axiom of Definition 3.4 can never be used in the last step of the proof of

\( I_1 \vdash [\mu_1,R_1] \rightarrow \Gamma_1 \), since in this case we would have \( \Gamma_1 \cap I_2 \neq \emptyset \) contradicting the hypothesis.

res). If, from

\[
\{G_0 I_1 \vdash [\mu_1,R_1] \rightarrow \Gamma_1 G_1 I_2 \vdash [\mu_2,R_2] \rightarrow \Gamma_2 G_2\}
\]
we can generate the equivalent computation

\[
\{G_0 I_2 \vdash [\mu_2,R_1] \rightarrow \Gamma_2 G_1 I_1 \vdash [\mu_1,R_2] \rightarrow \Gamma_1 G_2\},
\]
then from

\[
\{G_0 \alpha I_1 \alpha \vdash [\mu_1,R_1 \alpha] \rightarrow \Gamma_1 \alpha G_1 \alpha I_2 \alpha \vdash [\mu_2,R_2 \alpha] \rightarrow \Gamma_2 \alpha G_2 \alpha\}
\]
we generate the equivalent computation

\[
\{G_0 \alpha I_2 \alpha \vdash [\mu_2,R_1 \alpha] \rightarrow \Gamma_2 \alpha G_1 \alpha I_1 \alpha \vdash [\mu_1,R_2 \alpha] \rightarrow \Gamma_1 \alpha G_2 \alpha\}.
\]

Analogously for rel).

sum). If, from

\[
\{\text{dec}(E_1 \cdot I_3) I_1 \vdash [\mu_1,R_1] \rightarrow \Gamma_1 G_1 I_2 \vdash [\mu_2,R_2] \rightarrow \Gamma_2 G_2\}
\]
we can generate the equivalent computation

\[
\{\text{dec}(E_1 \cdot I_3) I_2 \vdash [\mu_2,R_1] \rightarrow \Gamma_2 G_1 I_1 \vdash [\mu_1,R_2] \rightarrow \Gamma_1 G_2\},
\]
then from

\[
\{(E_1 \cdot E_1) \vdash [\mu_1 \text{, } E_1 \cdot E_1 \leq (I_3 \cup R_1 \cup I_2)] \rightarrow \Gamma_1 G_1 I_2 \vdash [\mu_2,R_2] \rightarrow \Gamma_2 G_2\}
\]
we generate the equivalent computation

\[
\{(E_1 \cdot E_1) \vdash [\mu_2 \text{, } E_1 \cdot E_1 \leq (I_3 \cup R_1 \cup I_2)] \rightarrow \Gamma_2 G_1 I_1 \vdash [\mu_1,R_2] \rightarrow \Gamma_1 G_2\}.
\]
The inductive step for the two first com) rules can be proved following the same pattern of the proof of case res). Let us consider the most complicated case of the third implication: both events observed from the given computation are labelled by τ. Two further cases may arise.

i) In the first (where indexes l and r stand for left and right) we inductively assume that from
\[ \{G_0^l 1_1^l - [\lambda_1, R_1^l] \rightarrow \Gamma_1^l \quad G_1^l 1_2^l - [\lambda_2, R_2^l] \rightarrow \Gamma_2^l \quad G_1^l \} \]
we can generate the equivalent computation
\[ \{G_0^l 1_2^l - [\lambda_2, R_2^l] \rightarrow \Gamma_2^l \quad G_1^l 1_1^l - [\lambda_1, R_1^l] \rightarrow \Gamma_1^l \quad G_1^l \} \]
and that from
\[ \{G_0^r 1_1^r - [\lambda_1, R_1^r] \rightarrow \Gamma_1^r \quad G_1^r 1_2^r - [\lambda_2, R_2^r] \rightarrow \Gamma_2^r \quad G_1^r \} \]
we can generate the equivalent computation
\[ \{G_0^r 1_2^r - [\lambda_2, R_2^r] \rightarrow \Gamma_2^r \quad G_1^r 1_1^r - [\lambda_1, R_1^r] \rightarrow \Gamma_1^r \quad G_1^r \} \].
By applying the third case of rule com), we obtain that from
\[ \{G_0^l 1_1^l \cup \|idi G_0^r 1_2^l \cup \|idi I_1^l \rightarrow [\tau, R_1^l \cup \|idi I_1^r] \rightarrow \Gamma_1^l \ \|idi \cup \|idi I_1^r \}
G_1^l 1_1^r \cup \|idi I_1^r \rightarrow [\tau, R_2^l \cup \|idi I_1^r] \rightarrow \Gamma_1^r \ \|idi \cup \|idi I_1^r \] \]
we generate the equivalent computation
\[ \{G_0^l 1_2^l \cup \|idi G_0^r 1_1^l \cup \|idi I_2^r \rightarrow [\tau, R_1^l \cup \|idi I_2^r] \rightarrow \Gamma_2^l \ \|idi \cup \|idi I_2^r \}
G_1^l 1_1^r \cup \|idi I_1^r \rightarrow [\tau, R_2^l \cup \|idi I_2^r] \rightarrow \Gamma_2^r \ \|idi \cup \|idi I_2^r \] \].

ii) The proof is straightforward in the other case which occurs when we inductively assume that from
\[ \{G_0^l 1_1^l - [\lambda_1, R_1^l] \rightarrow \Gamma_1^l \quad G_1^l 1_2^l - [\lambda_2, R_2^l] \rightarrow \Gamma_2^l \quad G_1^l \} \]
we can generate the equivalent computation
\[ \{G_0^l 1_2^l - [\lambda_2, R_2^l] \rightarrow \Gamma_2^l \quad G_1^l 1_1^l - [\lambda_1, R_1^l] \rightarrow \Gamma_1^l \quad G_1^l \} \]
and that
\[ \{G_0^r 1_1^r - [\lambda_1, R_1^r] \rightarrow \Gamma_1^r \quad G_1^r 1_2^r - [\lambda_2, R_2^r] \rightarrow \Gamma_2^r \quad G_1^r \} \].

Corollary A.1. (two consecutive concurrent events can be generated in either ordering)

Given a computation ξ with observation o = (S, l, ≤, <), let e and e' be two concurrent events generated via Procedure TPO by two consecutive occurrences of po derivations. There always exist a computation ξ' with observation o' = (S, l, ≤, <'), where
\[ <' = (< - \{ e < e' \}) \cup \{ e' < e \} \].

Proof. Let
\[ \xi = \{G_0 \ldots G_i 1_i - [\mu_i, R_i] \rightarrow \Gamma_i \quad G_i 1_{i+1} - [\mu_{i+1}, R_{i+1}] \rightarrow \Gamma_{i+1} \quad G_{i+1} \ldots G_n \} \]
where events e and e' are originated by the i\textsuperscript{th} and (i+1)\textsuperscript{th} po derivations. By applying Lemma
A.1 it is easy to construct the required computation which is as follows.
$$\xi' = \{G_0 \cdots G_i \mathcal{I}_1 \mathcal{R}_i \cdots G_{i+1} \mathcal{I}_{i+1} \mathcal{R}_{i+1} \cdots G_n\}$$
The only check to be performed is that the partial ordering of the observation of $\xi'$ is indeed $\leq$. This follows immediately by noticing that the causal dependencies among the elements of $G_{i-1}$, those of $G_{i+1}$ and the events generated after Step 4 of Procedure TPO applied to computation
$$\{G_{i-1} \mathcal{I}_1 \mathcal{R}_i \cdots G_i \mathcal{I}_{i+1} \mathcal{R}_{i+1} \cdots G_{i+1}\}$$
are, by Lemma A.1.iii), exactly the same causal dependencies which result after Step 4 of TPO on computation
$$\{G_{i-1} \mathcal{I}_1 \mathcal{R}_i \cdots G_i \mathcal{I}_{i+1} \mathcal{R}_{i+1} \cdots G_n\}.$$

Lemma A.2. (two concurrent events can be generated in either ordering)

Given two CCS agents $E_0$ and $E_1$ and an observation $o = \langle S, l, \leq, < \rangle$ such that there exists a computation from $E_0$ to $E_1$ with observation $o$, we have that for all $<'$ such that $\leq <'<$, there exists a computation from $E_0$ to $E_1$, with observation $o' = \langle S, l, \leq, <' \rangle$.

Proof. Let $\Xi$ be the set of all the computations from $E_0$ to $E_1$ originating the same po of events $\langle S, l, \leq \rangle$ via Procedure TPO.

We have to prove that the observation $o' = \langle S, l, \leq, <' \rangle$ of any computation in $\Xi$ is such that $\leq <'<$. In other words, we are given the po of events $\langle S, l, \leq \rangle$ and a total ordering $<' = \{e_0, e_1, \ldots, e_k\}$ on the events of $S$ such that $\leq <'$, and we must find a computation $\xi'$ with observation $o' = \langle S, l, \leq, <' \rangle$. Let $\xi_0$ be a computation in $\Xi$. We construct a sequence of computations $\{\xi_0, \xi_1, \ldots, \xi_n\}$ as follows.

Assume that $\xi_j \in \Xi$ has observation $\xi_j = \langle S, l, \leq, <j \rangle$.

If $\xi_j = <'$, the required computation is found.

Otherwise, assume inductively that $\xi_j$ has the same $n$ first elements $<'$ has, and that $e_n$ occurs as the $(m+1)$th element, i.e., $\xi_j = \{e_0, e_1, \ldots, e_{n-1}, e'_n, \ldots, e'_{m-1}, e_n, \ldots, e_k\}$. Using Corollary A.1, it is easy to construct a computation $\xi_{j+1}$ with observation $\langle S, l, \leq, <j+1 \rangle$, where $<j+1 = \{e_0, e_1, \ldots, e_{n-1}, e'_n, e_n, e'_m, \ldots, e_k\}$. In fact $e_n$ and $e'_m$ are concurrent, for they appear in reverse order in $<j$ and $<$, which both contain $\leq$. Performing a total of $m-n$ exchanges we obtain an observation the total ordering of which is $<j+1 = <j+1$, and the inductive step is proved.

Theorem 4.1. (complete concurrency)

Given two CCS agents $E_0$ and $E_1$ and a mo of events $d = \langle S, l, \leq, < \rangle$ such that $E_0 \xrightarrow{d} E_1$, we have that

- $\leq <$;
- $\forall <'$ such that $\leq <'$, there exist an initial derivation $E_0 \xrightarrow{d'} E_1$, with $d' = \langle S, l, \leq, <' \rangle$.  

31
Proof. Let \( \xi \) be a computation, \( o = <S^0, l^0, \leq^0, <^0> \) be its observation and \( d = <S, l, \leq, <> \) be its mixed ordering of events.

Note that Procedures \( PO \) and \( TPO \) differ only in their Step 5, which is used in \( PO \) to discard events labelled by \( \tau \), i.e., \( S = S^0 - \{ e \mid l(e) = \tau \} \), and to accordingly restrict \( l, \leq, <> \).

First claim is obvious. We are left to prove the second claim: a total ordering \( '< \) on the events of \( S \) is given such that \( \leq \subseteq '< \), and we must find a computation \( \xi' \) with mixed ordering of events \( d' = <S, l, \leq, <> \). It suffices to find a \( '<^0 \) such that \( \leq^0 \subseteq '<^0 \), such that its restriction to \( S \) is \( '< \); Lemma A.2 can then be applied. Such a \( '<^0 \) does exist, since relation \( R = '< \cup '<^0 \) is a partial ordering (only the events labelled by \( \tau \) may be unrelated). In fact, a cycle in \( R \) would imply the existence of a cycle either in \( '< \) or in \( '<^0 \), for \( \leq \subseteq '< \). Indeed, we can chose as \( '<^0 \) any totalization of \( R \), obtained by adding the necessary pairs \( \mu '<^0 \tau \) or \( \tau '<^0 \mu \), and removing reflexivity. \( \diamond \)

**Appendix B**

**Theorem 4.5.** (po equivalence is a congruence in all contexts except for \(+\))

Partial ordering observational equivalence is preserved by all operators except for \(+\).

**Proof.**

Let \( E, E_0, E_1 \) be CCS agents. The proof proceeds by case analysis on the operators of CCS, under the hypothesis that there exists a bisimulation \( R \) containing the pair \( <E_0, E_1> \), i.e., \( E_0 \equiv E_1 \).

1. **act)** It suffices proving that adding to \( R \) the pair \( <\mu E_0, \mu E_1> \) results in a bisimulation. We distinguish two cases.

   - If \( \mu \neq \tau \), let us consider a computation for which \( \mu E_0 \equiv <S, l, \leq \Rightarrow E'_0 \), and call \( e \) the event corresponding to its first quadruple \( <\mu E_0, \mu E'_0> \rightarrow dec(E_0) \). There exists then another computation such that \( E_0 \equiv <S', l, \leq \Rightarrow E'_0 \), with \( S' = S - \{ e \} \) and \( \leq' = \leq - (\{ e \} \cup \{ e e' \mid e' \in S' \}) \). By hypothesis, we can always grow a computation for which \( E_1 \equiv <S', l, \leq \Rightarrow E'_1 \), with \( E'_0 \equiv E'_1 \); and from this the required computation such that \( \mu E_1 \equiv <S', l, \leq \Rightarrow E'_1 \). And vice versa.

   - If \( \mu = \tau \), obvious.

2. **res)** We have to prove that \( E_0 \not\equiv E_1 \). The proof is easy: if there exists an agent \( E'_0 \) such that \( E_0 \not\equiv h \Rightarrow E'_0 \) (i.e., for whichever computation you choose from \( dec(E_0) \) to \( dec(E'_0) \) with no quadruples of the form \( \{ \alpha, n \rightarrow l \} \)), by hypothesis \( E_0 \equiv E_1 \) we can always find an agent \( E'_1 \) such that \( E'_0 \equiv E'_1 \) and \( E_1 \not\equiv h \Rightarrow \) such that \( E'_0 \equiv E'_1 \) (i.e., there exists a computation from \( dec(E_1) \) to \( dec(E'_1) \) with no quadruples of the form \( \{ \alpha, n \rightarrow l \} \). And vice versa. Thus, the required bisimulation is \( R' = \{ E \alpha \alpha \not\equiv E \alpha \alpha \not\equiv E \in R \} \).

32
Trivial, since \( \phi \) is a permutation of \( \Lambda \cup \{ \tau \} \) that preserves \( \tau \) and \( \tau' \); the required bisimulation is \( R' = \{ \langle E_0(\phi), E_1(\phi) \rangle, \langle E_0, E_1 \rangle \in R \} \).

We only consider the case of a right \( I \) context; the other case is symmetrical.
The required bisimulation is \( R' = \{ \langle E_0I, E_1I \rangle, \langle E_0, E_1 \rangle \in R \} \). In order to support our claim, we now prove that whenever \( E_0I \equiv h \equiv E_0'I \equiv E_1I \equiv E_1'I \equiv E' \) and \( E_0I \equiv E_0'I \equiv E_1I \equiv E_1'I \equiv E' \). By a symmetric argument, \( R' \) is therefore a bisimulation.

This is the most difficult case to be proved, and, in order to guide the reader in understanding the proof, we first consider the case when there is no communication between \( E_0 \) and \( E \). Then, we prove the thesis for a single-step computation consisting of a synchronization. Finally, we extend this result to the general case.

In the first case, for every computation of \( E_0I \) with no communication
\[
\xi_0 = \{(dec(E_0)lid \cup idldc(E)) \mid \mu_1, \varpi_1 \rightarrow I_1 \rightarrow G_1 \ldots \}
\]
with \( h = \langle S, I, \leq \rangle \) as label of its \( \mu \) many step derivation, we must find a computation of \( E_1I \) with no communication
\[
\xi_1 = \{(dec(E_1)lid \cup idldc(E)) \mid \mu_1, \varpi_1 \rightarrow I_1 \rightarrow G_1 \ldots \}
\]
with the same \( h \) as label of the \( \mu \) many step derivation, and \( E_0I \equiv E_0'I \equiv E_1I \equiv E_1'I \equiv E' \).

We can write each (occurrence of) complete set of grapes \( G_k \) as \( G_klid \cup idlG_k r \) (index \( l \) is for \( left \), \( r \) for \( right \)), where \( G_klid \) (\( G_k r \)) is a(n occurrence of) complete set of grapes to which \( dec(E_0) \) (\( dec(E) \)) has evolved. Now, since there is no communication, it is possible to partition \( \xi_0 \) in two parts: the first one contains those quadruples involving only grapes in \( G_klid \); the second part contains those quadruples involving only grapes in \( idlG_k r \). This can always be done, by looking whether \( I_k \subseteq G_klid \), or \( I_k \subseteq idlG_k r \).

The partition above induces a partition on \( h \) in \( h^l \) and \( h^r \), as well. The events of \( h \) are then accordingly partitioned depending on whether they correspond to the quadruples in the left or the right part of \( \xi_0 \), respectively. It is important to note that, since \( I_k \cap I_j = \emptyset \), for all \( I_k \subseteq G_klid \), \( I_j \subseteq idlG_k r \), all events of \( h^l \) are \textit{concurrent} with those of \( h^r \).

Two computations can now be generated from \( \xi_0 \), by “splitting” each quadruple in its premises, more precisely

- \( I_j \rightarrow [\mu_j, \varpi_j] \rightarrow I_j' \), with \( I_j \subseteq G_j r \), originates \( I_j \rightarrow [\mu_j, \varpi_j] \rightarrow I_j' \), where \( \varpi_jr \) and \( I_jr = I_j, I_jr = I_j, I'_j = I'_j \).
- \( I_j \rightarrow [\mu_j, \varpi_j] \rightarrow I_j' \), with \( I_j \subseteq idlG_j r \), originates \( I_j \rightarrow [\mu_j, \varpi_j] \rightarrow I_j' \), where \( idl \varpi_jr = \varpi_j, idl I_jr = I_j, idl I'_j = I'_j \).

We obtain the following computations

\[
\xi_0 = \{(dec(E_0)lid \cup idldc(E)) \mid \mu_1, \varpi_1 \rightarrow I_1 \rightarrow G_1 \ldots \}
\]
\[
\xi_1 = \{(dec(E_1)lid \cup idldc(E)) \mid \mu_1, \varpi_1 \rightarrow I_1 \rightarrow G_1 \ldots \}
\]
\[ \xi_0^1 = (\text{dec}(E_0) \ I_1^1 \rightarrow [\mu_1^1, R_1^1] \rightarrow I_1^1 \ G_1^1 \ ... \ G_{p-1}^1 \ I_p^1 \rightarrow [\mu_p^1, R_p^1] \rightarrow I_p^1 \ \text{dec}(E'_0)) \]

\[ \xi_0^r = (\text{dec}(E) \ I_1^r \rightarrow [\mu_1^r, R_1^r] \rightarrow I_1^r \ G_1^r \ ... \ G_{q-1}^r \ I_q^r \rightarrow [\mu_q^r, R_q^r] \rightarrow I_q^r \ \text{dec}(E')) \]

which give rise to po derivations which are (isomorphic to) \( h^1 \) and \( h^r \):

\[ h^x = \langle S^x, I^x, \leq^x \rangle \text{ (with } x = 1, r) \text{, where} \]

- \( S^x = \{ e_j \in S \mid e_j \text{ is generated by procedure PO in correspondence to } I_j^x \rightarrow [\mu_j^x, R_j^x] \rightarrow I_j^x \} \);
- \( I^x \) is the restriction of \( I \) to \( S^x \);
- \( \leq^x = \{ e_j \leq e_k \mid e_j, e_k \in S^x \} \).

By hypothesis, \( E_0 \) and \( E_1 \) are po equivalent, thus we can find an agent \( E'_1 \) po equivalent to \( E'_0 \), such that \( E_1 = h \Rightarrow E_1' \). This po derivation is obtained by a computation, say

\[ \xi_1^1 = (\text{dec}(E_1) \ I_1^1 \rightarrow [\mu_1^1, R_1^1] \rightarrow I_1^1 \ G_1^1 \ ... \ G_{z-1}^1 \ I_z^1 \rightarrow [\mu_z^1, R_z^1] \rightarrow I_z^1 \ \text{dec}(E'_1)) \]

Eventually, we can “put Hampty together again” first by inferring from quadruple

- \( I_j^1 \rightarrow [\mu_j^1, R_j^1] \rightarrow I_j^1 \), 1\( \leq j \leq z \), the quadruple \( I_j^1 \ lif \rightarrow [\mu_j^1, R_j^1] \ lif \rightarrow I_j^1 \ lif = I_j^1 \rightarrow [\mu_j^1, R_j^1] \rightarrow I_j^1 \);
- \( I_j^r \rightarrow [\mu_j^r, R_j^r] \rightarrow I_j^r \), 1\( \leq j \leq q \), the quadruple \( idlI_j^r \rightarrow [\mu_j^r, idlR_j^r] \rightarrow idlI_j^r = I_j^r \rightarrow [\mu_j^r, R_j^r] \rightarrow I_j^r \);

and then by generating the following computation.

\[ \xi_1 = (\text{dec}(E_1) \ lif \cup idldec(E) \ I_1 \rightarrow [\mu_1^1, R_1^1] \rightarrow I_1 \ G_1^1 \cup idldec(E) \ ... \ I_z \rightarrow [\mu_z^1, R_z^1] \rightarrow I_z \ \text{dec}(E'_1) \ lif \cup idldec(E) \ ... \ I_i \rightarrow [\mu_i^r, R_i^r] \rightarrow I_i \ \text{dec}(E'_1) \ lif \cup idldec(E')) \]

(coming from \( \xi_1^1 \))

\[ I_1 = [\mu_1^r, R_1^r] \rightarrow I_1 \ G_1^r \ lif \cup idlG_1^r \ ... \ I_i = [\mu_i^r, R_i^r] \rightarrow I_i \ \text{dec}(E'_1) \ lif \cup idldec(E')) \]

(coming from \( \xi_0^r \)).

Computation \( \xi_1 \) is indeed such that \( E_1 | E = h \Rightarrow E_1' | E' \), and thus the proof is completed in the case of no communication. Actually, composition of computations \( \xi_1^1 \) and \( \xi_0^r \) was possible because the quadruples in the former generate events which are concurrent with all those generated by the quadruples in the latter, and thus Theorem 4.1 stating complete concurrency applies.

We now consider a computation containing only a single quadruple resulting from a synchronization, i.e.,

\[ \xi_0 = (\text{dec}(E_0) \ lif \cup idldec(E) \ I \rightarrow [\tau, R] \rightarrow I \ \text{dec}(E'_0) \ lif \cup idldec(E')) \]

with empty po as label of its derivation. We must find a computation of \( E_1 | E \)

\[ \xi_1 = (\text{dec}(E_1) \ lif \cup idldec(E) \ I \rightarrow [\tau, R] \rightarrow I \ \text{dec}(E'_1) \ lif \cup idldec(E')) \]

with the same empty po labelling its derivation, and \( E'_0 | E' \equiv E'_1 | E' \).
Again, two computations can be generated, by “splitting” the quadruple in its premises, more precisely

1. \( I \rightarrow [\tau, \mathcal{R}] \rightarrow I' \) originates \( I^1 \rightarrow [\lambda, \mathcal{R}^1] \rightarrow I^1 \) and \( I^r \rightarrow [\lambda^r, \mathcal{R}^r] \rightarrow I^r \),
   where \( I^1 \text{id} \cup idl^r = I \), \( I^1 \text{id} \cup idl^r = I' \), and \( \mathcal{R}^1 \text{id} \cup idl^r = \mathcal{R} \).

We obtain the following computations

\[
\xi_0^1 = (\text{dec}(E_0) \quad I^1 \rightarrow [\lambda, \mathcal{R}^1] \rightarrow I^1 \quad \text{dec}(E_0));
\]
\[
\xi_0^r = (\text{dec}(E) \quad I^r \rightarrow [\lambda^r, \mathcal{R}^r] \rightarrow I^r \quad \text{dec}(E')).
\]

that generate one new event each, say, \( e_1 \) and \( e_r \), and have the following po derivations

\[
h^1_\tau = \langle \{e_1\}, \{1(e_1) = \lambda\}, \{e_1 \leq e_1\} \rangle;
\]
\[
h^r_\tau = \langle \{e_r\}, \{1(e_r) = \lambda^r\}, \{e_r \leq e_r\} \rangle.
\]

By inductive hypothesis \( E_0 \) and \( E_1 \) are congruent, thus we can find an agent \( E_1' \) equivalent to \( E_0 \), such that \( E_1 \equiv h^1_\tau \Rightarrow E_1'. \) This po derivation is obtained by a computation, say

\[
\xi_1^1 = (\text{dec}(E_1) \quad I^1 \rightarrow [\lambda, \mathcal{R}^1] \rightarrow I^1 \quad \text{dec}(E_1')).
\]

We can now “put Hampty together again” first by inferring from quadruples

\[
I^1 \text{id} \rightarrow [\lambda, \mathcal{R}^1 \text{id}] \rightarrow I^1 \text{id} \quad \text{and} \quad idl^r \rightarrow [\lambda^r, idl^r] \rightarrow idl^r \quad \text{the quadruple}
\]
\[
I^1 \text{id} \rightarrow [\tau, \mathcal{R}^1] \rightarrow I^1 \quad \text{id} = I \rightarrow [\tau, \mathcal{R}] \rightarrow I
\]

and then by generating the following computation

\[
\xi_1 = (\text{dec}(E_1) \text{id} \cup idl \text{dec}(E) \quad I \rightarrow [\tau, \mathcal{R}] \rightarrow I' \quad \text{dec}(E_1') \text{id} \cup idl \text{dec}(E')).
\]

Obviously \( E_1 \| E \equiv h^1_\tau \Rightarrow E_1', E'. \)

Now we can better face the general case, by using the facts proved above. Suppose we are given the following computation

\[
\xi_0 = \mathcal{A} \quad I_1 \rightarrow [\tau_1, \mathcal{R}_1] \rightarrow I'_1 \quad \mathcal{B} \quad I_2 \rightarrow [\tau_2, \mathcal{R}_2] \rightarrow I'_2 \quad \cdots \quad I_n \rightarrow [\tau_n, \mathcal{R}_n] \rightarrow I'_n \quad \mathcal{Z}
\]

where \( \mathcal{A}, \mathcal{B}, \ldots, \mathcal{Z} \) denote (possibly empty) segments of computation without communications, and the invisible actions due to the \( n \) communications carry indexes in order to uniquely pick them up. We can split \( \xi_0 \), as we did above, obtaining the following computations.
\[ \xi_0^l = \mathcal{A}^l \Gamma_1^l \rightarrow \cdots \rightarrow \Gamma_n^l ] \rightarrow \Gamma_n^l Z^l \]

\[ \xi_0^r = \mathcal{A}^r I_1^r \rightarrow \cdots \rightarrow I_n^r \rightarrow I_n^r Z^r \]

Computations \( \xi_0^l \) and \( \xi_0^r \) originate po derivations labelled by \( h^l \) and \( h^r \), respectively. As done before, we can obtain the following computation originating, by inductive hypothesis, a po derivation with label \( h^l \).

\[ \xi_1^l = \mathcal{A}^l I_1^l \rightarrow \cdots \rightarrow I_n^l \rightarrow I_n^l Z^l \]

We may now compose \( \xi_1^l \) and \( \xi_0^r \), to obtain \( \xi_1 \), by iteratively interleaving their parts without communication and "synchronizing" the quadruples with action \( \lambda_i \). In doing so, two cases may arise, depending on whether the actions used for synchronization are generated in the same order or not. More accurately, whether \( \lambda_i = (\lambda_i)^- \) or \( \lambda_i \neq (\lambda_i)^- \). In the first case, no trouble arises and the required computation is simply

\[ \xi = \mathcal{A}^l \mathcal{A}^r I_1^l \rightarrow I_1^r \rightarrow \cdots \rightarrow I_n^l \rightarrow I_n^r \rightarrow Z^l Z^r \]

where \( I_i^l \rightarrow I_i^r \rightarrow \cdots \rightarrow I_n^l \rightarrow I_n^r \rightarrow Z^l Z^r \) are obtained from \( I_i^l \rightarrow I_i^r \rightarrow \cdots \rightarrow I_n^l \rightarrow I_n^r \rightarrow Z^l Z^r \) as done above. Since computations \( \xi_0^l \) and \( \xi_1^l \) have the same label \( h \) and are sewed with \( \xi_0^l \) in the same manner, the recomposed computation \( \xi_1 \) originates the same po derivation of \( \xi_0^l \).

When there exist quadruples with \( \lambda_i = (\lambda_i)^- \), we have to re-arrange computation \( \xi_1^l \) in order to go back to the previous case, and "put Hampty together again" properly. This can always be done, since these quadruples generate the corresponding events in different total orderings, and thus these events are concurrent. Thus, we can apply Theorem 4.1 to switch transitions in \( \xi_1^l \) and still obtain a legal computation originating the same po derivation with label \( h^l \).

rec) Empty, since only closed terms are allowed.
References


