On Relating Concurrency and Nondeterminism*

Luca Aceto
INRIA - Sophia Antipolis
F-06560 Valbonne Cedex, France

Abstract

We present an intuitive preorder for a simple CCS-like language whose semantic theory allows us to relate concurrency and nondeterminism without reducing the former to the latter. The preorder over processes is induced by using an equationally defined preorder over computations in a bisimulation-like protocol. The relationships of the proposed preorder with pomset bisimulation and standard strong bisimulation equivalence are studied in detail. Moreover, we give an axiomatization of the preorder over recursion-free processes.

1 Introduction

In recent years a lot of research has been devoted to the study of notions of equivalence for Process Algebras, such as CCS [Mil89], CSP [Hoare85] and ACP [BK85]. The interest in equivalences and preorders, which relate descriptions of concurrent systems in terms of these languages, stems from the fact that process algebras are used not only for describing actual systems, but also their specifications. Notions of equivalence between descriptions are thus an important component of these languages as they allow one to formally state when (the description of) a system is a correct implementation of a given specification. Roughly, the proposals presented in the literature may be divided into two broad classes:

1. the equivalences and preorders which semantically reduce parallelism to sequential nondeterminism, and

2. those whose semantic theories treat parallelism as a primitive notion.

The equivalences which semantically reduce parallelism to sequential nondeterminism are usually called interleaving equivalences as in their associated theories concurrency between events is interpreted as their arbitrary interleaving (i.e. their possibility to occur in any temporal order). Several different notions of equivalence, based upon the interleaving approach, have been proposed in the literature.

*This work has been supported by a grant from the United Kingdom Science and Engineering Research Council as well as by the Esprit BRA project CEDISYS.
literature; the most widely used amongst them are trace equivalence [Hoare85], observational equivalence [Mil80], bisimulation equivalence [Pa81], testing equivalence [DH84], [H88a] and failure equivalence [BHR84]. Although their theories are very different from each other, all of these behavioural equivalences have much in common; the only form of abstraction supported by the above-mentioned equivalences is related to nondeterminism. Processes can only be equivalent if their behaviour is the same modulo nondeterminism.

Several researchers have recently argued that, although they allow a faithful description of the interactive behaviour of processes, the assumptions underlying the interleaving semantic models are inadequate in accounting for the nonsequential behaviour of distributed systems and that semantic theories which consider parallelism as a primitive notion are best suited for this purpose. Consequently, several equivalences for process algebras which distinguish concurrency from sequential nondeterminism have recently been proposed in the literature, e.g. distributed bisimulation [CH87], timed equivalence [H88b], pomset bisimulation [BC87,88], NMS-bisimulation [DDM86]. A variety of equivalences has been discussed in [GV87] in the context of Petri Nets [Rei85]. All of these equivalences are based on adaptations of the standard notion of bisimulation equivalence and draw a sharp line between concurrent execution of actions and their interleavings, between causal and temporal dependencies among computational events.

Although the above-mentioned equivalences differ in the degree in which they model the interplay between causality and the branching structure of processes, they identify descriptions of processes only if they can exhibit the same degree of parallelism. However, it may be argued that, whereas the interleaving equivalences are too coarse in that they forget too much of the structure of processes, the above-mentioned equivalences are perhaps too discriminating in that they do not allow one to relate "concurrent implementations" and "nondeterministic specifications" at all. As it is frequently more natural to specify the behaviour of a system in terms of a sequential nondeterministic process and more efficient to implement it in a parallel fashion, it would be helpful to have a semantic theory of processes which allows us to relate these two notions without semantically reducing parallelism to sequential nondeterminism. One possible use of this feature of the theory is in requiring that all the parallelism which is present in a specification be maintained in the implementation.

The main aim of this paper is to provide such a semantic theory for a simple CCS-like language and to show how standard tools used in defining interleaving and non-interleaving equivalences for this language may be adapted in order to reconcile both philosophies to the semantics of concurrency.

Intuitively, our proposal is based upon a preorder \( \sqsubseteq \) over processes such that, for processes \( p \) and \( q \), \( p \sqsubseteq q \) if, and only if, \( p \) and \( q \) have the same "interactive behaviour" and \( q \) is "at least as parallel as" \( p \). In order to formalize this idea, one needs to make precise the notion of "interactive behaviour" over processes and to give a formal way of measuring the degree of parallelism a process may exhibit during its evolution. In this paper we take the view that a reasonable notion of equivalence between the interactive behaviour of two concurrent, nondeterministic processes is captured by bisimulation equivalence, [Pa81]. Of course there is some arbitrariness in the choice of such an interleaving equiva-
lence as the touchstone of our approach. However, bisimulation equivalence has a simple and elegant mathematical theory, [Mil89], and its properties have been investigated in a number of papers in the literature, [HM85], [Ab87a,b]. Moreover, the simplicity of its definition will allow us to concentrate on the issues which are more relevant to the aim of this paper.

The preorder $\sim$ will be defined by means of a bisimulation-like relation which will be dependent on the second parameter mentioned above: the measure of the degree of parallelism processes may exhibit during their evolution. In order to formalize this notion, following [BC87,88], we shall drop the requirement of atomicity over the actions performed by a concurrent process and shall axiomatize a preorder over the set of actions, $\preceq_C$. Intuitively, for computations $u$ and $v$, $u \preceq_C v$ is intended to capture the fact that $u$ and $v$ correspond to the performance of the same atomic events, but the events are performed in a possibly more parallel fashion in $v$. The reasonableness of the preorder $\preceq_C$ will be justified by showing that it coincides with a simple model-theoretic relation over a suitable interpretation of computations as finite labelled posets. It will be shown how, by using the preorder $\preceq_C$ over computations in a bisimulation-like relation, it is possible to obtain, in a rather simple way, a preorder over processes $\subseteq$ which gives rise to a semantic theory in which concurrency is related to nondeterminism, but is not reduced to it.

We now give a brief outline of the remainder of the paper. In §2 we introduce the language studied in the paper and give both a standard labelled transition system semantics, [Kel76], and a pomset transition system semantics, [BC87,88], for it. This section also introduces the notion of computative study in detail in §3. The syntactic and model-theoretic properties of a preorder on the set of computations are investigated in §3. We axiomatize a preorder $\preceq_C$ over the set of computation with the intent of capturing their relative degree of parallelism, and show that it coincides with an intuitive preorder over an interpretation of computations as finite labelled posets. The preorder $\preceq_C$ on computations is used in §4 to induce one over processes, $\subseteq$, by means of a bisimulation-like protocol. We show that $\subseteq$ is a precongruence with respect to all the operators of the calculus and study its relationships with bisimulation equivalence and the pomset bisimulation equivalence [BC87,88]. An equational characterization of $\subseteq$ over the set of recursion-free processes is presented in §5. Finally, §6 presents a simple application of the semantic theory developed in the previous sections to the specification of concurrent systems. We end with a conclusion and suggestions for future work.

2 The language and its operational semantics

This section is devoted to the presentation of a simple process algebra, essentially a subset of Milner CCS [Mil89], which will be used to introduce the formalism and the intuitions underlying the semantic theory for processes presented in this paper. The process algebra, which is parameterized over a set $A$ of atomic actions $A$, consists of a facility for recursive definitions and the following operators:

- $\text{nil}$, a constant used to denote an action, the process that cannot perform any action, [Mil89]
\[(1) \; a.p \xrightarrow{a} p \]
\[(2) \; p \xrightarrow{a} p' \quad \text{implies} \quad p + q \xrightarrow{a} p' \]
\[(3) \; p \xrightarrow{a} p' \quad \text{implies} \quad p|q \xrightarrow{a} p'|q \]
\[(4) \; t[\text{rec } x \cdot t/x] \xrightarrow{a} p \quad \text{implies} \quad \text{rec } x \cdot t \xrightarrow{a} p \]

Figure 1: Axiom and rules for \(\xrightarrow{a}\)

- \(\cdot\), a unary operator, one for each \(a \in A\), used to prefix an action to a process. Intuitively, \(a.p\)
  will denote a process capable of performing action \(a\) and behaving like \(p\) thereafter;

- \(\cdot\), for \textit{nondeterministic choice};

- \(\cdot\), for \textit{parallel composition} (without communication).

Formally:

\textbf{Definition 2.1} Let \(A\) be a countable set of atomic actions, ranged over by \(a, b, \ldots\), and \(X\) be a countable set of variables, ranged over by \(x, y, \ldots\). The set of terms over \(A\) and \(X\) is generated by the following syntax:

\[t ::= \text{nil} \mid a.t \mid t + t \mid t|t \mid \text{rec } x \cdot t,\]

where \(a \in A\) and \(x \in X\). We assume the usual notions of free and bound variables in terms, with \(\text{rec } x \cdot \_\) as the binding operator. The set of closed terms (or processes) will be denoted by \(P\) (\(p, q, \ldots \in P\)) and that of finite processes, those not containing occurrences of \(\text{rec } x \cdot \_\), will be denoted by \(P_{\text{Fin}}\).

Following Milner [Mil89], a standard operational semantics may be given to \(P\) by means of Plotkin's SOS, [P81]. This can be done by defining a binary transition relation \(\xrightarrow{a}\) for each \(a \in A\). A standard way of defining \(\xrightarrow{a}\), \(a \in A\), is to stipulate that \(\xrightarrow{a}\) is the least binary relation on \(P\) which satisfies the axiom and rules in Figure 1. Several interpretations of the relations \(\xrightarrow{a}\) are possible; a standard one is the following:

\[p \xrightarrow{a} p' \text{ if } p \text{ may perform action } a \text{ and thereby become } p'.\]

This interpretation of the transition relation \(\xrightarrow{a}\) closely corresponds to Milner's experimental approach to the semantics of concurrent processes, [Mil89]. Roughly, experimenting on a process is taken to mean communicating with it and the binary relations \(\xrightarrow{a}\), \(a \in A\), are used to formalize the changes of state caused by successful \(a\)-experiments on processes by their environment. Informally, two processes that exhibit the same observable behaviour are deemed to be equivalent.
One of the most successful attempts to formalize the above notion of equivalence between processes is the notion of \textit{bisimulation equivalence}, \cite{Pa81}. A relation $\mathcal{R} \subseteq P^2$ is a bisimulation if, for each $(p, q) \in \mathcal{R}$, $a \in A$, the following conditions hold:

(a) $p \xrightarrow{a} p'$ implies, for some $q', q \xrightarrow{a} q'$ and $(p', q') \in \mathcal{R}$;

(b) $q \xrightarrow{a} q'$ implies, for some $p', p \xrightarrow{a} p'$ and $(p', q') \in \mathcal{R}$.

As it is well-known, the largest such relation is $\sim = \bigcup \{ \mathcal{R} \mid \mathcal{R} \text{ is a bisimulation} \}$. The following proposition is standard.

**Proposition 2.1** $\sim$ is a congruence over $P$.

The aim of this paper is to investigate whether it is possible to carry out the programme sketched in the introduction employing the tools of Milner's approach to the semantics of concurrency: structural operational semantics and bisimulation.

In order to relate concurrency and nondeterminism in a way that does not semantically reduce parallelism to sequential nondeterminism, we shall take a more liberal view of the computational steps of a concurrent system than the one underlying the operational semantics given in terms of the labelled transition system $(P, \{ \xrightarrow{a} \mid a \in A \})$. Underlying an interleaving description of the semantics of concurrent processes is the idea that processes evolve from one (global) state to another by performing atomic actions. Following \cite{BC87,88}, we shall assume that concurrent processes evolve by performing actions which are no longer atomic in space and time, \cite{Ca88}. The computational steps considered in what follows will be transitions of the form $p \xrightarrow{u} q$, where $u$ is a syntactically deterministic process (a syntactic notation for a \textit{partially ordered multiset}, or \textit{pomset} in Pratt and Gischer's terminology \cite{Pr86, Gi84}). The operational semantics for the set of processes $P$ will thus be defined by means of what G. Boudol and I. Castellani call a \textit{pomset transition system}.

**Definition 2.2** The set of computations, Comp, is generated by the following syntax:

$$u ::= \text{nil} \mid a.u \mid u|u,$$

where $a \in A$. Comp will be ranged over by $u, u, v, \ldots$.

As already mentioned above, a computation is thus a syntactically deterministic process. This idea of computational step is already implicit in Winskel's notion of \textit{configuration} for Event Structures, \cite{Win87}, and has been studied at length in \cite{BC87,88}. We can now define a pomset transition system semantics for $P$ following \cite{BC87,88}.
Definition 2.3 For each \( u \in \text{Comp} \), \( \xrightarrow{u} \) is the least binary relation over \( P \) which satisfies the following axiom and rules:

1. \( a.p \xrightarrow{\text{a.nil}} p \)
2. \( p \xrightarrow{u} p' \) implies \( a.p \xrightarrow{\text{a.u}} p' \)
3. \( p \xrightarrow{u} p' \) implies \( p + q \xrightarrow{u} p' \)
   \( q + p \xrightarrow{u} q' \)
4. \( p \xrightarrow{u} p' \) implies \( p|q \xrightarrow{u} p'|q \)
   \( q|p \xrightarrow{u} q'|p' \)
5. \( p \xrightarrow{u} p', q \xrightarrow{u} q' \) imply \( p|q \xrightarrow{\text{a.u}} p'|q' \)
6. \( t|\text{rec} x. t/x | \xrightarrow{u} p \) implies \( \text{rec} x. t \xrightarrow{u} p \).

Comments about these rules may be found in the above-quoted references. Notationally, occurrences of \( \text{nil} \) will be often omitted in the context \( a.n\text{il} \), both in processes and computations.

The pomset operational semantics defined above allows one to clearly distinguish causal dependencies from temporal ones, as the following examples show.

Example 2.1

1. \( a.b + b.a \xrightarrow{a} b \xrightarrow{b} \text{nil} \)
   \( \xrightarrow{b} a \xrightarrow{a} \text{nil} \)
   \( \xrightarrow{\text{a.b}} \text{nil} \)
   \( \xrightarrow{\text{b.a}} \text{nil}. \)
2. \( a|b \xrightarrow{a} \text{nil}|b \xrightarrow{b} \text{nil}|\text{nil} \)
   \( \xrightarrow{b} a|\text{nil} \xrightarrow{a} \text{nil}|\text{nil} \)
   \( \xrightarrow{a|b} \text{nil}|\text{nil}. \)

Note how the process \( a|b \) is capable of performing actions \( a \) and \( b \) in any temporal order (represented in this setting by the composition of the relations \( a \rightarrow \) and \( b \rightarrow \)), but it is not capable of performing them in a single computational step labelled \( a.b \) or \( b.a \) (which corresponds to a causal dependence between the two actions). Thus there is a whole set of transitions which allow one to differentiate the processes \( a|b \) and \( a.b + b.a \), which are typically equated by interleaving semantic theories, [Mil89], [DH84].

The strength of a pomset transition system view of processes has been exploited by G. Boudol and I. Castellani, who define a standard bisimulation equivalence on \( (P, \{ \xrightarrow{u} | u \in \text{Comp} \}) \) up to a simple equational theory of computations. The resulting equivalence \( \approx_{P_{\text{om}}} \), which they call pomset bisimulation, clearly distinguishes concurrency from sequential nondeterminism and its properties are investigated in [BC87,88]. A similar notion of equivalence has been presented in [GV87] in the setting of Petri Nets. However, as pointed out in the introduction, it might be argued that these equivalences are too discriminating. In fact, they do not allow one to relate sequential nondeterministic processes with concurrent ones at all, which is a drawback as it is frequently more natural to specify a system
in a nondeterministic fashion, but it might be more efficient to implement it in a parallel way. The aim of this work is to show how, using the notions of pomset transition system and bisimulation, it is possible to define a preorder on \( P \) which is compatible with both an interleaving view of processes and the idea that concurrency should not be reduced to sequential nondeterminism.

3 A preorder on computations

This section is devoted to the introduction of the basic tool used in this paper for relating nondeterminism and concurrency and to a study of its elementary properties. The tool for relating nondeterminism and concurrency in the behaviour of the processes in \( P \) will be a preorder over computations \( \leq_C \). Intuitively, for computations \( u \) and \( v \), \( u \leq_C v \) is intended to capture the fact that \( u \) and \( v \) correspond to the performance of the same atomic actions, but these actions are performed in a more parallel fashion in the computation \( v \).

Formally, let \( \leq_C \) denote the least precongruence over \( \text{Comp} \) which satisfies the following set of axioms \( C \):

\[
\begin{align*}
(PAR1) & \quad z|\text{nil} \quad = \quad z \\
(PAR2) & \quad z|y \quad = \quad y|z \\
(PAR3) & \quad (z|y)|z \quad = \quad z|(y|z) \\
(SEQ) & \quad a.(z|y) \quad \leq \quad a.z|y.
\end{align*}
\]

Of the above axioms, axiom (SEQ) is the interesting one; it expresses the intuitive fact that a computation in which action \( a \) causes both \( z \) and \( y \) is "more sequential" than a computation in which action \( a \) only causes \( z \). Its relevance is probably best illustrated by means of an example.

Example 3.1 Using the axioms in \( C \) it is possible to give a formal proof of the intuitive statement "\( a.b \) is more sequential than \( a|b \)" as follows:

\[
\begin{align*}
\ a.b.\text{nil} \quad =_C \quad a.\bar{a}.(\text{nil}|\text{nil}) & \quad \text{by (PAR1) and substitutivity} \\
\leq_C \quad a.(\text{nil}|b) \quad & \quad \text{by (SEQ) and (PAR2)} \\
\leq_C \quad a|b & \quad \text{by (SEQ)}.
\end{align*}
\]

The reasonableness of the preorder \( \leq_C \) over computations will be demonstrated by showing that the set of axioms \( C \) completely axiomatizes a natural preorder on an interpretation of computations as finite posets labelled on \( A \).

It is clear from the definition of the pomset transition system semantics for \( P \) that there are computations \( u \in \text{Comp} \) such that \( \overrightarrow{u} = \emptyset \). For instance, \( p \overrightarrow{\text{nil}} \) or \( p \overrightarrow{\text{nil}} \) for no \( p \) and \( a \). The computations \( u \in \text{Comp} \) such that \( p \overrightarrow{u} \) for some \( p \in P \) will be referred to as relevant.

Definition 3.1 Let \( R\text{Comp} \) be the subset of \( \text{Comp} \) generated by the following syntax:

\[ u ::= a.\text{nil} \mid a.u \mid u|u, \]

where \( a \in A \).
The following lemma states that every relevant computation \( u \) is a member of \( R\text{Comp} \).

**Lemma 3.1** For each \( p \in P \), \( u \in \text{Comp} \), \( p \xrightarrow{u} \) implies \( u \in R\text{Comp} \).

It is now easy to see that \( R\text{Comp} \) is indeed the set of relevant computations mentioned above. This follows from the above lemma and the observation that, for each \( u \in R\text{Comp} \), \( u \xrightarrow{u} \). The next definition and lemma express some properties of \( \leq_G \) over some important subclasses of computations.

**Definition 3.2** (Sequential and Maximally Parallel Computations) A computation \( u \in \text{Comp} \) is said to be:

1. sequential if \( u =_G a_0, \ldots, a_n \) for some \( n \geq 0 \) and \( \{a_0, \ldots, a_n\} \subseteq A \);
2. maximally parallel if \( u =_G \prod_{i \in I} a_i \), where \( I \) is a finite nonempty index set and \( \{a_i \mid i \in I\} \subseteq A \).

The notation \( \prod_{i \in I} u_i \) stands for \( u_{i_1} \ldots u_{i_k} \) (where \( I = \{i_1, \ldots, i_k\}, k \geq 0 \)) and is justified by axioms \((\text{PAR2})-(\text{PAR3})\). By convention \( \prod_{i \in \emptyset} u_i \equiv \text{nil} \).

**Lemma 3.2** Let \( u, v \in \text{Comp} \) be such that \( u \leq_G v \). Then:

1. if \( v \) is sequential then \( u =_G v \);
2. if \( v \equiv a_0, \ldots, a_n \) and \( u \in R\text{Comp} \) then \( u \equiv v \);
3. if \( u \) is maximally parallel then \( u =_G v \);
4. if \( u \equiv \text{nil} \) and \( v \in R\text{Comp} \) then \( u \equiv v \).

The properties of (relevant) computations stated in the above lemma will be very useful in relating the notion of preorder developed in §4 with Milner's strong bisimulation equivalence \( \sim \). Before giving the definition of our preorder which relates concurrency to nondeterminism, we shall show that the preorder \( \leq_G \) on computations, defined in this section by purely syntactical means, has indeed an intuitive model-theoretic characterization. The remaining part of this section is devoted to the study of such a model-theoretic characterization of \( \leq_G \) over \( \text{Comp} \). Computations will be interpreted as finite posets labelled on \( A \). A preorder, denoted by \( \preceq \), which is intended to capture the intuition conveyed by the defining axioms of \( \leq_G \), is defined over the interpretation of \( \text{Comp} \) and it is shown to coincide with \( \leq_G \).

### 3.1 Labelled posets

The following definition introduces the main object of study of this section.

**Definition 3.3** (A-posets) A finite \( A \)-labelled partially ordered set (A-poset) is a triple \( \ell = (E, \leq, l) \) where:

- \( E \) is a finite set of events (the vertices of the poset),
- \( \leq \) is a partial order on \( E \) (the causality relation), and
• \( l : E \rightarrow A \) is a labelling function.

In what follows, \( A \)-posets will be always considered up to isomorphism. Isomorphism classes of \( A \)-posets (for suitable label sets \( A \)) are often met under different names in the literature; for example, Pratt calls them pomsets (an acronym for partially ordered multisets), [Gi84], [Pr86], and Grabowski calls them partial words, [Gr81].

The following notions about posets will be useful in what follows.

**Definition 3.4** Let \( \mathcal{E} = (E, \leq, l) \) be an \( A \)-poset. Then:

- \( \leadsto_{\mathcal{E}} = \text{def} \ E^2 - (\leq \cup \leq^{-1}) \) is the concurrency relation over \( \mathcal{E} \). The subscript \( \mathcal{E} \) will be often dropped whenever the \( A \)-poset we are referring to is clear from the context;
- \( \text{Min}(\mathcal{E}) = \text{def} \ \{ e \in E \mid \forall e' \in E \ e' \leq e \implies e' = e \} \) will denote the set of minimal events in \( \mathcal{E} \);
- for each \( e \in E \), \( \uparrow e \) will denote, with abuse of notation, both
  - (a) \( \uparrow e = \text{def} \ \{ e' \in E \mid e \leq e' \} \), and
  - (b) \( \uparrow e = \text{def} \ (\uparrow e, \leq, l((\uparrow e))) \), the substructure of \( \mathcal{E} \) corresponding to \( \uparrow e \).

Due to the restricted form of sequential composition, action-prefixing, allowed in the signature for computations, the \( A \)-posets which are interpretations of computations will have a particularly simple order structure.

**Definition 3.5 (Separable \( A \)-Posets)** An \( A \)-poset \( \mathcal{E} = (E, \leq, l) \) is separable iff, for each \( e, e' \in E \), \( e \leadsto e' \) implies \( e \leq e'' \) and \( e' \leq e'' \) for no \( e'' \in E \).

We can now define a preorder over \( A \)-posets which will be shown to give a model-theoretic characterization of the relation \( \leq_G \) defined syntactically over computations. The definition of the preorder is based upon the notion of projection presented in the following definition.

**Definition 3.6 (The Model-Theoretic Preorder)**

1. Let \( \mathcal{E}_i = (E_i, \leq_i, l_i), \ i = 1, 2, \) be two \( A \)-posets. A projection of \( \mathcal{E}_2 \) onto \( \mathcal{E}_1 \) is a function \( h : E_2 \rightarrow E_1 \) such that:
   - (i) \( h \) is bijective,
   - (ii) \( \forall e, e' \in E_2 \ e \leq e' \) implies \( h(e) \leq_1 h(e') \), i.e. \( h \) is monotonic, and
   - (iii) \( \forall e \in E_2 \ l_2(e) = l_1(h(e)) \), i.e. \( h \) is label-preserving.

2. The relation \( \triangleleft \) over the set of \( A \)-posets is given by:

\[
\mathcal{E}_1 \triangleleft \mathcal{E}_2 \iff \text{there exists a projection } h \text{ of } \mathcal{E}_2 \text{ onto } \mathcal{E}_1.
\]

**Lemma 3.3** \( \triangleleft \) is a preorder over the set of \( A \)-posets.
Intuitively, $\mathcal{E}_1 < \mathcal{E}_2$ means that all the causal dependencies among the events in the $A$-poset $\mathcal{E}_2$ can be bijectively embedded into causal dependencies in $\mathcal{E}_1$ (i.e. $\mathcal{E}_1$ has “at least as many causal dependencies” as $\mathcal{E}_2$). This is intended to capture the intuition that $\mathcal{E}_1 < \mathcal{E}_2$ iff “$\mathcal{E}_2$ is at least as parallel as $\mathcal{E}_1$”.

**Example 3.2** $A$-posets will be drawn, up to isomorphism, as Hasse diagrams growing rightwards. As we are concerned with labelled structures, the events will be left anonymous. The following $A$-posets are related by $\triangleleft$:

```
  a → b → c  \triangleleft  a → b  \triangleleft  b
              \quad c
```

On the other hand, $\frac{\text{a}}{\text{b}} \not\triangleleft \frac{\text{a} \rightarrow \text{b}}{\text{c}}$.

The following proposition establishes some important properties of projection functions.

**Proposition 3.1** Let $\mathcal{E}_i = (E_i, \leq_i, l_i)$, $i = 1, 2$, be two $A$-posets. Assume that $h : E_2 \rightarrow E_1$ is a projection function. Then:

1. for each $e \in E_2$, $h(e) \in \text{Min} (\mathcal{E}_1)$ implies $e \in \text{Min} (\mathcal{E}_2)$;
2. assume $\mathcal{E}_1$ is separable. Then:
   1. $\{ \mathcal{E}_2(e) \mid e \in \text{Min} (\mathcal{E}_1) \}$, where $\mathcal{E}_2(e) \overset{\text{def}}{=} \{ \hat{e} \in E_2 \mid e \leq_1 h(\hat{e}) \}$, is a partition of $E_2$.
      Moreover, for each $e \in \text{Min} (\mathcal{E}_1)$, $h[\mathcal{E}_2(e)]$ is a projection of $\mathcal{E}_2(e)$ onto $\uparrow e$.
   2. For each $e_1, e_2 \in E_2$, $e_1 \leq_2 e_2 \in \mathcal{E}_2(e)$ and $e \in \text{Min} (\mathcal{E}_1)$ imply $e_1 \in \mathcal{E}_2(e)$.

**Proof:** Assume that $\mathcal{E}_1$ and $\mathcal{E}_2$ are $A$-posets and that $h : E_2 \rightarrow E_1$ is a projection. We only give the proof of (2).

1. Assume that $\mathcal{E}_1$ is separable.
   1. We prove, first of all, that $\{ \mathcal{E}_2(e) \mid e \in \text{Min} (\mathcal{E}_1) \}$ is a partition of $E_2$.
      - We show that $E_2 = \bigcup \{ \mathcal{E}_2(e) \mid e \in \text{Min} (\mathcal{E}_1) \}$. It is sufficient to prove that, for each $e' \in E_2$, there exists $e \in \text{Min} (\mathcal{E}_1)$ such that $e' \in \mathcal{E}_2(e)$. Assume $e' \in E_2$. By the finiteness of $E_1$, $\text{Min} (\mathcal{E}_1) \neq \emptyset$ and, for each $\hat{e} \in E_1$, there exists $e \in \text{Min} (\mathcal{E}_1)$ such that $e \leq_1 \hat{e}$. Thus $e \leq_1 h(e')$, for some $e \in \text{Min} (\mathcal{E}_1)$. This implies $e' \in \mathcal{E}_2(e)$.
      - We show that, for $e, e' \in \text{Min} (\mathcal{E}_1)$, $e \neq e'$ implies $\mathcal{E}_2(e) \cap \mathcal{E}_2(e') = \emptyset$. Assume, towards a contradiction, that $e, e' \in \text{Min} (\mathcal{E}_1)$, $e \neq e'$ and $e \in \mathcal{E}_2(e) \cap \mathcal{E}_2(e')$. By the definition of $\mathcal{E}_2(e)$, we have that $e \leq_1 h(e)$ and $e' \leq_1 h(e)$. As $e, e' \in \text{Min} (\mathcal{E}_1)$ and $e \neq e'$ imply $e \sim_\mathcal{E}_1 e'$, the above assumption violates the hypothesis that $\mathcal{E}_1$ is separable. Hence, for each $e, e' \in \text{Min} (\mathcal{E}_1)$, $e \neq e'$ implies $\mathcal{E}_2(e) \cap \mathcal{E}_2(e') = \emptyset$. 
We have thus shown that \( \{ \xi_2(e) \mid e \in \operatorname{Min}(\xi_1) \} \) is a partition of \( E_2 \). In order to show that, for each \( e \in \operatorname{Min}(\xi_1) \), \( h[\xi_2(e)] \) is a projection of \( \xi_2(e) \) onto \( \uparrow e \), it is sufficient to prove that \( h[\xi_2(e)] \) is surjective. Assume \( \bar{e} \in \uparrow e \), i.e. \( e \preceq \bar{e} \). As \( h \) is bijective, there exists \( e' \in E_2 \) such that \( h(e') = \bar{e} \). It follows, by the definition of \( \xi_2(e) \), that \( e' \in \xi_2(e) \). Thus \( h[\xi_2(e)] \) is surjective. All the other properties of projection maps follow from the hypothesis that \( h \) is itself a projection.

(iii) Assume, towards a contradiction, that \( \bar{e} \in \xi_2(e) \), \( e' \preceq \bar{e} \) and \( e' \notin \xi_2(e) \). Then, by (i), there exists \( \hat{e} \in \operatorname{Min}(\xi_1) \) such that \( e' \in \xi_2(\hat{e}) \). It must be the case that \( \hat{e} \neq e \). Then, by the definition of \( \xi_2(\cdot) \), \( e \preceq_1 h(\hat{e}) \) and \( \bar{e} \preceq_1 h(e') \). By the monotonicity of \( h \), \( e' \preceq \bar{e} \) implies \( h(e') \preceq_1 h(\bar{e}) \). Thus, by transitivity, \( \bar{e} \preceq_1 h(\bar{e}) \). As \( e \neq \bar{e} \) and \( e, \bar{e} \in \operatorname{Min}(\xi_1) \), we have that \( e \sim_\xi \bar{e} \). This contradicts the hypothesis that \( \xi_1 \) is separable.

This completes the proof of statement (2). \( \square \)

The main consequence of the above proposition is that if \( h : E_2 \rightarrow E_1 \) is a projection of \( \xi_2 = (E_2, \leq_2, l_2) \) onto \( \xi_1 = (E_1, \leq_1, l_1) \) and \( \xi_1 \) is separable then \( h \) determines a partition \( \equiv_h = \{ \xi_2(e) \mid e \in \operatorname{Min}(\xi_1) \} \) of \( E_2 \) such that, for each \( e \in \operatorname{Min}(\xi_1) \), \( \uparrow e \sim_\xi \xi_2(e) \). This property of projection functions will be very useful in relating \( \leq_C \) and \( \prec \). In fact, we can give another useful characterization of the partition \( \equiv_h \) as follows:

- for each \( e \in \operatorname{Min}(\xi_1) \), \( \operatorname{Min}(\xi_2, e) \overset{\text{def}}{=} \{ e' \in \operatorname{Min}(\xi_2) \mid e \preceq_1 h(e') \} \),
- for each \( e \in \operatorname{Min}(\xi_1) \), \( [\operatorname{Min}(\xi_2, e)] = \{ e' \in E_2 \mid \exists \bar{e} \in \operatorname{Min}(\xi_2, e) : \bar{e} \preceq_2 e' \} \), the upper-closure of \( \operatorname{Min}(\xi_2, e) \).

**Proposition 3.2** If \( \xi_1 \) is separable then, for each \( e \in \operatorname{Min}(\xi_1) \), \( \xi_2(e) = [\operatorname{Min}(\xi_2, e)] \).

**Proof:** Assume \( e' \in \xi_2(e) \), i.e. \( e \preceq_1 h(e') \). As \( E_2 \) is finite, there exists \( \bar{e} \in \operatorname{Min}(\xi_2) \) such that \( \bar{e} \preceq_2 e' \). By proposition 3.1, we also have that \( \bar{e} \in \xi_2(e) \). Hence \( e' \in [\operatorname{Min}(\xi_2, e)] \). The converse inclusion easily follows by the monotonicity of \( h \). \( \square \)

The import of the above proposition is that the partition \( \equiv_h \) can be obtained as the upper-closure of a partition over the minimal elements of \( \xi_1 \); this partition being \( \equiv_{\xi_1} = \{ \operatorname{Min}(\xi_2, e) \mid e \in \operatorname{Min}(\xi_1) \} \).

### 3.2 Interpretation of computations as A-posets

In order to interpret computations as A-posets, we shall have to impose the structure of a \( \Sigma^{\text{Comp}} \) algebra over the set of A-posets, where \( \Sigma^{\text{Comp}} \) is obviously given by \( \{ \text{nil}, | \} \cup \{ a_- | a \in A \} \). This is all that is needed to give an interpretation of \( \text{Comp} \) in terms of A-posets.

Let \( \xi_i = (E_i, \leq_i, l_i), i = 1, 2 \), be two A-posets. Then:

- \( a \xi_1 = (E_i, \leq_i, l_i) \), where
(1) \( E = E_1 \cup \{ r \} \), with \( r \notin E_1 \),

(2) \( \forall e, e' \in E \ e \leq e' \text{ iff } e = r \text{ or } e \leq_1 e' \),

(3) for each \( e \in E \)

\[
\ell(e) = \begin{cases} 
  a & \text{if } e = r \\
  l_1(e) & \text{otherwise}.
\end{cases}
\]

- \( \ell_1 | \ell_2 = (E, \leq, l) \), where

(1) \( E = E_1 \uplus E_2 \), the disjoint union of \( E_1 \) and \( E_2 \),

(2) \( \leq = \{(i, e), (j, e') \ | \ i = j \text{ and } e \leq_1 e' \} \ (\leq_1 \uplus \leq_2) \),

(3) \( l(i, e) = l_i(e) \ (l = l_1 \uplus l_2) \).

- \( \text{NIL} = (0, 0, 0) \).

The unique homomorphism from \( \text{Comp} \) to the \( \Sigma^{\text{Comp}} \)-algebra of \( A \)-posets will be denoted by \( [\cdot] \). It can be given the usual inductive characterization as follows:

\[
\begin{align*}
[nil] &= \text{NIL} \\
[a.u] &= a.[u] \\
[u|v] &= [u][v].
\end{align*}
\]

The following lemma, which can be easily shown by structural induction over \( u \in \text{Comp} \), states that the interpretations of computations are always separable \( A \)-posets.

**Lemma 3.4** For each \( u \in \text{Comp} \), \( [u] \) is a separable \( A \)-poset.

Let \( \triangleleft \) denote, with abuse of notation, the preorder over \( \text{Comp} \) defined by:

\[
\forall u, v \in \text{Comp} \ u \triangleleft v \text{ iff } [u] \triangleleft [v].
\]

We shall now proceed to show that \( \triangleleft \) and \( \leq_C \) coincide over \( \text{Comp} \), i.e. that \( \leq_C \) completely axiomatizes \( \triangleleft \). This result will hopefully provide a good motivation for the choice of \( \leq_C \) as our preorder over computations. First of all we show that \( \triangleleft \) is a \( \Sigma^{\text{Comp}} \)-precongruence over the algebra of \( A \)-posets (and, consequently, over \( \text{Comp} \)).

**Proposition 3.3** \( \triangleleft \) is a precongruence over the algebra of \( A \)-posets.

The next lemma states the soundness of the syntactically defined relation \( \leq_C \) with respect to \( \triangleleft \).

**Lemma 3.5** (Soundness) For each \( u, v \in \text{Comp} \), \( u \leq_C v \) implies \( u \triangleleft v \).

We now concentrate on proving that \( \leq_C \) is indeed complete with respect to \( \triangleleft \) over \( \text{Comp} \), i.e. that \( u \triangleleft v \) implies \( u \leq_C v \). The proof of completeness relies as usual on the isolation of normal forms for computations.

**Definition 3.7** (Normal Forms for Computations) The set of normal forms \( \text{NF} \) is the least subset of \( \text{Comp} \) which satisfies:
* nil $\in$ NF,

* $\Pi_{i=0}^n a_i, u_i \in$ NF if $n \geq 0$ and, for each $0 \leq i \leq n$, $u_i \in$ NF.

The following normalization lemma, whose proof requires axioms (PAR1)-(PAR3) only, states that each computation $u$ can be reduced to a normal form $nf(u)$ using the axioms in $C$.

**Lemma 3.6 (Normalization)** For each $u \in$ Comp, there exists a normal form $nf(u)$ such that $u =_C nf(u)$.

The following lemma, which is based upon Propositions 3.1 and 3.2, states a fundamental decomposition property which will be used in the proof of the completeness theorem.

**Lemma 3.7 (Decomposition Lemma)** Assume $u \equiv \Pi_{i=0}^n a_i, u_i < v \equiv \Pi_{j=0}^m b_j, v_j$. Then there exists a partition $\{J_i \mid 0 \leq i \leq n\}$ of $J = \{0, \ldots, m\}$ such that, for each $0 \leq i \leq n$, $a_i, u_i < \Pi_{j \in J_i} b_j, v_j$.

**Proof:** By the definition of $\leq$, $u < v$ iff $[u] < [v]$ iff $\Pi_{i=0}^n a_i, [u_i] < \Pi_{j=0}^m b_j, [v_j]$. By the definition of the operations over $A$-posets, $Min([u]) = Min(\Pi_{i=0}^n a_i, [u_i]) = \{e_0, \ldots, e_n\}$, where $e_i$ denotes the least element of $[a_i, [u_i]]$. By lemma 3.4, $[u]$ is a separable $A$-poset. By propositions 3.1 and 3.2, a projection $h$ of $[v]$ onto $[u]$ establishes a partition of the event set of $[v]$. Moreover, this partition is actually determined by a partition $\{J_i \mid 0 \leq i \leq n\}$ over the set of minimal elements $\{f_0, \ldots, f_m\}$ of $[v]$, where $f_i$ is the least element of $[b_j, [v_j]]$. By propositions 3.1 and 3.2, for each $0 \leq i \leq n$,

$$a_i, [u_i] = \uparrow e_i < [v](e_i) = [Min([v], e_i)] = \prod_{j \in J_i} b_j, [v_j].$$

We can now prove the promised completeness theorem.

**Theorem 3.1 (Completeness)** For each $u, v \in$ Comp, $u < v$ iff $u \leq_C v$.

**Proof:** The if implication is just lemma 3.5. Assume $u < v$. By the normalization lemma we may assume, wlog, that $u$ and $v$ are nfs. The proof proceeds by induction of the combined size of $u$ and $v$ (axioms (PAR2)-(PAR3) are used throughout the proof). If $u \equiv \text{nil}$ then it is easy to see that also $v \equiv \text{nil}$. The claim is then trivial.

Otherwise assume that $u \equiv \Pi_{i=0}^n a_i, u_i$ and $v \equiv \Pi_{j=0}^m b_j, v_j$, with $n \geq 0$. If $u < v$ then, by the decomposition lemma, there exists a partition $\{J_i \mid 0 \leq i \leq n\}$ of $J = \{0, \ldots, m\}$ such that, for each $i \in \{0, \ldots, n\}$, $a_i, u_i < \Pi_{j \in J_i} b_j, v_j$. We show that, for each $0 \leq i \leq n$, $a_i, u_i \leq_C \Pi_{j \in J_i} b_j, v_j$. The claim will then follow by substitutivity and (PAR2)-(PAR3).

Let $i \in \{0, \ldots, n\}$. As $a_i, u_i < \Pi_{j \in J_i} b_j, v_j$, there exists a projection $h$ of $[\Pi_{j \in J_i} b_j, v_j]$ onto $[a_i, u_i]$. Let $e$ denote the least element of $[a_i, u_i]$, which is labelled by $a_i$. By proposition 3.1, there exists $e' \in \text{Min}([\Pi_{j \in J_i} b_j, v_j])$ such that $h(e') = e$. Assume, wlog, that $e'$ is the least element of $[b_k, u_k]$, $k \in J_i$. As $h$ is label-preserving, $b_k = a_i$. It is easy to see that, by the construction of $[\Pi_{j \in J_i} b_j, v_j]$, $h - \{(e', e)\}$ is a projection which establishes

$$u_i < v_h \prod_{j \in J_i \sim \{k\}} b_j, v_j.$$
We may now apply the inductive hypothesis to obtain $u_i \leq_C v_k | \prod_{j \in J_i - \{k\}} b_j, v_j$. Then

$$a_i, u_i \leq_C a_i, (v_k | \prod_{j \in J_i - \{k\}} b_j, v_j)$$

by substitutivity

$$\leq_C a_i, v_k | \prod_{j \in J_i - \{k\}} b_j, v_j$$

by (SEQ)

$$= C \prod_{j \in J_i} b_j, v_j$$

by (PAR2)-(PAR3) and $b_k = a_i$.

This completes the proof of the theorem. □

We shall end this section with a result which expresses an intuitive property of the preorder $\leq_C$ which will be useful in the following section.

**Proposition 3.4 (Linearization)** Let $u, v \in \text{Comp}$. Assume that $a_0, \ldots, a_n \leq_C u$ and $b_0, \ldots, b_m \leq_C v$. Then $a_0, \ldots, a_n, b_0, \ldots, b_m \leq_C u|v$.

4 Relating nondeterminism and concurrency

In §3 we have introduced a syntactic preorder $\leq_C$ over the set of computations $\text{Comp}$ and studied some of its elementary properties. The aim of this section is to show how $\leq_C$ can be used to induce an interesting preorder over the set of processes $\mathcal{P}$. This preorder will allow us to relate sequential nondeterministic processes with concurrent ones, which may be thought of as their implementations. However, this preorder does not reduce concurrency to sequential nondeterminism as it will be clear from what follows. The preorder on $\mathcal{P}$ will be defined by means of the by now well-established notion of bisimulation, [Pa81].

Let $\mathcal{R}$ denote the set of binary relations over $\mathcal{P}$. The functional $\mathcal{F} : \mathcal{R} \rightarrow \mathcal{R}$ is defined, for each $\mathcal{R} \in \mathcal{R}$, as follows:

$$(p, q) \in \mathcal{F}(\mathcal{R}) \text{ if, for each } u \in \text{Comp},$$

1. $p \xrightarrow{u} p'$ implies, for some $q' \in \mathcal{P}$ and $v \in \text{Comp}$ such that $u \leq_C v$, $q \xrightarrow{u} q'$ and $(p', q') \in \mathcal{R}$;
2. $q \xrightarrow{u} q'$ implies, for some $p' \in \mathcal{P}$ and $v \in \text{Comp}$ such that $v \leq_C u$, $p \xrightarrow{u} p'$ and $(p', q') \in \mathcal{R}$.

A relation $\mathcal{R} \in \mathcal{R}$ will be called an $\mathcal{F}$-bisimulation if $\mathcal{R} \subseteq \mathcal{F}(\mathcal{R})$. Let $\mathcal{L}$ denote $\bigcup \{ \mathcal{R} : \mathcal{R} \subseteq \mathcal{F}(\mathcal{R}) \}$.

The following proposition is then standard.

**Proposition 4.1** $\mathcal{F}$ is a monotonic endofunction over the complete lattice $(\mathcal{R}, \subseteq)$ and $\mathcal{L}$ is its largest fixed-point.

Intuitively, for processes $p$ and $q$, $p \subseteq q$ if, for each computational step $u$,

- if $p$ may perform $u$ and thereby be transformed into $p'$ then $q$ may perform a computation $v$, which is a "possibly more parallel version" of $u$, and enter a state $q'$ such that $p' \subseteq q'$; conversely
- if $q \xrightarrow{u} q'$ then there exist a computation $v$, which is a "possibly more sequential version" of $u$, and a state $p'$ such that $p \xrightarrow{u} p'$ and $p' \subseteq q'$.
This informal explanation of the nature of $\subseteq$ should give a hint of the importance of the preorder on computations $\leq_C$ in this setting. Before giving a few examples of the relationships among processes that can be established using $\subseteq$, examples that will highlight the rôle played by $\leq_C$ in relating process behaviours, we show that $\subseteq$ is indeed a preorder. However, it is easy to check that $Id_{\mathcal{P}}$ is an $\mathcal{F}$-bisimulation and that the composition of $\mathcal{F}$-bisimulations is again an $\mathcal{F}$-bisimulation. As a corollary of these observations we obtain the following:

Fact 4.1 $\subseteq$ is a preorder over $\mathcal{P}$.

A few examples are now in order.

Example 4.1

1) $p \equiv a \cdot b + b. a \subseteq a \cdot b + a \cdot b + b. a \equiv q$. In fact, it is easy to see that the relation

$$\mathcal{R} = \{(p, q), (b, nil|b), (a, a|nil), (nil, nil|nil)\} \cup Id_{\mathcal{P}}$$

is an $\mathcal{F}$-bisimulation. The key point is that the move $q \xrightarrow{\alpha|b} nil|nil$ can be matched by $p \xrightarrow{\beta|b} nil$ as $a \cdot b \leq_C a|b$ (or, equivalently, by $p \xrightarrow{\beta|a} nil$).

2) $q \subseteq a|b \equiv r$. In fact, it is easy to check that the relation

$$\mathcal{S} = \{(q, r), (b, nil|b), (a, a|nil), (nil, nil|nil)\} \cup Id_{\mathcal{P}}$$

is an $\mathcal{F}$-bisimulation.

3) $q \not\subseteq p$. In fact $q \xrightarrow{\alpha|b} nil|nil$, but $p \xrightarrow{\beta|b} \not\in \mathcal{F}$ and $p \xrightarrow{\beta|a}$.

4) $r \subseteq q$. In fact, $q \xrightarrow{\alpha|b} nil|nil$, but $r \xrightarrow{\beta|a}$. By lemma 3.2 this is sufficient to establish that, for no $v \leq_C a|b$, $r \xrightarrow{v}$.

5) $r \not\subseteq p$. In fact, $r \xrightarrow{\alpha|b} nil|nil$, but $p \xrightarrow{\beta|b}$ and $p \xrightarrow{\beta|a}$. Again, by lemma 3.2 this is sufficient to establish that, for no $v$ such that $a|b \leq_C v$, $p \xrightarrow{v}$.

Proposition 4.2 Let $p, q, r \in \mathcal{P}$. Assume $p \subseteq q$. Then $a \cdot p \subseteq a \cdot q$, $p + r \subseteq q + r$ and $p \cdot r \subseteq q \cdot r$.

The above proposition tells us that $\subseteq$ is a well-behaved preorder with respect to all the operators of the algebra apart from recursion. In order to show that $\subseteq$ is also preserved by recursive contexts, we first extend it to open terms.

Definition 4.1 A closed substitution is a map $\rho : X \rightarrow \mathcal{P}$. Then, for open terms $t_1, t_2, t_1 \subseteq t_2$ if, for every closed substitution $\rho$, $t_1 \rho \subseteq t_2 \rho$.

The following proposition, whose proof follows standard lines [Mil89], states that $\subseteq$ is closed with respect to recursive contexts.

Proposition 4.3 Let $t_1 \subseteq t_2$. Then $rec \cdot t_1 \subseteq rec \cdot t_2$. 

The preorder $\subseteq$ that we have just defined enjoys some interesting relationships with Milner's strong bisimulation equivalence $\sim$. As already mentioned in the introduction, $\sim$ will be used to assess the reasonableness of our proposal. Intuitively, it will be required that $\subseteq$ relates processes $p$ and $q$ only when $p \sim q$, i.e. only when $p$ and $q$ have the same "interactive behaviour". Moreover, given the intuition that we are trying to capture, it might be expected that, for sequential nondeterministic processes, $\subseteq$ coincides with $\sim$. We shall now show that both these properties are true of $\subseteq$. These results should hopefully reinforce $\subseteq$ as a reasonable notion of preorder over $\mathcal{P}$.

Lemma 4.1 For each $p \in \mathcal{P}$, $a \in A$, $p \xrightarrow{a, \text{nil}} p'$ iff $p \xrightarrow{a} p'$.

The above lemma is all that is needed to prove that $\subseteq$ is contained in $\sim$.

Theorem 4.1 For each $p, q \in \mathcal{P}$, $p \subseteq q$ implies $p \sim q$.

Proof: It is sufficient to show that $\subseteq$ is an interleaving bisimulation. Assume $p \subseteq q$ and $p \xrightarrow{a} p'$.

Then, by lemma 4.1, $p \xrightarrow{a, \text{nil}} p'$. As $p \subseteq q$, there exist $v \in \text{Comp}$ and $q'$ such that $a, \text{nil} \leq_C v, q \xrightarrow{u} q'$ and $p' \subseteq q'$. By lemma 3.2, $v \equiv a, \text{nil}$ and thus, by lemma 4.1, $q \xrightarrow{a} q'$.

Assume $q \xrightarrow{a} q'$. Then, again by lemma 4.1, $q' \xrightarrow{a, \text{nil}} q'$. As $p \subseteq q$, there exist $v \leq_C a, \text{nil}$ and $p'$ such that $p \xrightarrow{u} p'$ and $p' \subseteq q'$. By lemma 3.2, $v \equiv a, \text{nil}$ and, by lemma 4.1, $p \xrightarrow{a} p'$. Thus $\subseteq$ is an interleaving bisimulation and $p \sim q$. \hfill $\square$

It follows from the above theorem and examples that the containment is indeed strict. Intuitively, it may be expected that $\subseteq$ and $\sim$ coincide over processes that do not contain occurrences of the parallel composition operator. We shall call these processes sequential and the subset of sequential processes will be denoted by $\mathcal{P}_\text{Seq}$.

Lemma 4.2 Let $p \in \mathcal{P}_\text{Seq}$. Then, for each $u \in \text{Comp}$, $p \xrightarrow{u}$ implies $u$ has the form $a_0, \ldots, a_n, \text{nil}$.

In order to prove that $\subseteq$ coincides with $\sim$ over $\mathcal{P}_\text{Seq}$, we shall need to relate derivations of the form $p \xrightarrow{u} q$, $u \equiv a_0, \ldots, a_n$, with Milner's multi-step derivations $p \xrightarrow{a_0} \cdots \xrightarrow{a_n} q$. This is done in the following lemma.

Lemma 4.3 Let $p \in \mathcal{P}_\text{Seq}$. Then:

1. $p \xrightarrow{a} p' \xrightarrow{u} q$ implies $p \xrightarrow{a, u} q$;
2. for $u \equiv a_0, \ldots, a_n$, $p \xrightarrow{u} q$ iff $p \xrightarrow{a_0} \cdots \xrightarrow{a_n} q$.

We can now show that $\subseteq$ and $\sim$ coincide over $\mathcal{P}_\text{Seq}$, i.e. that for sequential processes $\subseteq$ reduces to Milner's strong bisimulation equivalence.

Theorem 4.2 For each $p, q \in \mathcal{P}_\text{Seq}$, $p \subseteq q$ iff $p \sim q$. 
Proof: The only if implication follows by Theorem 4.1. In order to prove that, for \( p, q \in \mathcal{P}_{\text{seq}} \), \( p \sim q \) implies \( p \sim q \), it is sufficient to show that the relation \( \mathcal{R} = \cap \mathcal{P}_{\text{seq}}^2 \) is an \( \mathcal{F} \)-bisimulation. Assume \( (p, q) \in \mathcal{R} \).

Suppose that \( p \xrightarrow{u} p' \). Then, by lemma 4.2, \( u \equiv a_0 \ldots a_n \text{nil} \). By lemma 4.3, \( p \xrightarrow{u} p' \) implies \( p \xrightarrow{a_0} \ldots \xrightarrow{a_n} p' \). As \( p \sim q \), it is easy to see that there exists \( q' \) such that \( q \xrightarrow{a_0} \ldots \xrightarrow{a_n} q' \) and \( p' \sim q' \). By lemma 4.3, \( q \xrightarrow{u} q' \). Obviously, \( u \leq C u \) and, as \( p', q' \in \mathcal{P}_{\text{seq}} \), \( (p', q') \in \mathcal{R} \). The other defining clause of \( \mathcal{R} \) is checked by a similar argument. Thus \( \mathcal{R} \) is an \( \mathcal{F} \)-bisimulation. \( \square \)

Indeed, one can prove a slightly stronger statement than Theorem 4.2. Our aim is to show that, for \( p \in \mathcal{P}_{\text{seq}} \) and \( q \in \mathcal{P} \), \( p \sim q \) implies \( p \sim q \). This statement is useful for verification purposes and gives a pleasing proof technique for \( \mathcal{F} \) which will be used in \$6\). The following lemma will be useful in the proof of the above mentioned result.

**Lemma 4.4** Let \( p \in \mathcal{P} \). Then the following statements hold:

1. \( p \xrightarrow{\sigma} p' \xrightarrow{u} p \) implies \( p \xrightarrow{v} q \) for some \( v \) such that \( a.u \leq C v \);
2. \( p \xrightarrow{\sigma} q, \sigma \equiv a_0 \ldots a_n \) implies \( p \xrightarrow{u} q \) for some \( u \) such that \( a_0 \ldots a_n \leq C u \);
3. \( p \xrightarrow{u} q \) implies \( p \xrightarrow{\sigma} q, \sigma \equiv a_0 \ldots a_n \) for some \( a_0 \ldots a_n \leq C u \).

**Proof:** We prove each statement in turn.

1. By induction on the proof of the derivation \( p \xrightarrow{\sigma} p' \). The only interesting case is the following:

   * \( p \equiv p_1[p_2 \xrightarrow{\sigma} p'_1[p_2] \) because \( p_1 \xrightarrow{\sigma} p'_1 \). We proceed by analyzing the move \( p'_1[p_2 \xrightarrow{u} q \).

     By the pomset operational semantics, \( p'_1[p_2 \xrightarrow{u} q \) iff

     a. \( p'_1 \xrightarrow{u} p''_1 \) and \( q \equiv p''_1[p_2 \), or

     b. \( p'_1 \xrightarrow{u} p''_2 \) and \( q \equiv p''_2[p_2 \), or

     c. \( p'_1 \xrightarrow{u} p''_1, p_2 \xrightarrow{u} p''_2, u \equiv u_1 \) and \( q \equiv p''_1[p_2[1]

     If (a) holds then, by the inductive hypothesis, \( p_1 \xrightarrow{v} p''_1 \) for some \( v \) such that \( a.u \leq C v \).

     Hence, by the pomset operational semantics, \( p_1[p_2 \xrightarrow{v} p''_1[p_2 \equiv q \).

     If (b) holds then, as by lemma 4.1 \( p_1 \xrightarrow{a} p'_1 \) implies \( p_1 \xrightarrow{a, \text{nil}} p'_1 \), by the pomset operational semantics \( p_1[p_2 \xrightarrow{a} p'_1[p_2 \equiv q \).

     Moreover,

     \[
     a.u \equiv C a.(\text{nil}|u) \quad \text{by (PAR1) and substitutivity}
     \leq C a.\text{nil}|u \quad \text{by (SEQ)}
     \]

     If (c) holds then, by the inductive hypothesis, \( p_1 \xrightarrow{v} p''_1 \) for some \( v \) such that \( a.u_1 \leq C v \).

     Hence, by the pomset operational semantics, \( p_1[p_2 \xrightarrow{v} p''_1[p_2 \equiv q \).

     Moreover,

     \[
     a.u \equiv a.(u_1|u_2) \leq C a.u_1|u_2 \quad \text{by (SEQ)}
     \leq C v|u_2 \quad \text{by substitutivity}.
     \]
(2) By induction on \( n \). The basis of the induction easily follows by lemma 4.1. For the inductive step, assume \( p \xrightarrow{a} q \), \( \sigma \equiv a_0 \ldots a_n \), \( n \geq 1 \). Then \( p \xrightarrow{a_0} p' \xrightarrow{a_1 \ldots a_n} q \), for some \( p' \). By the inductive hypothesis, \( p' \xrightarrow{u} q \) for some \( u \) such that \( a_1 \ldots a_n \leq_C u \). By statement (1) of the lemma, \( p \xrightarrow{a_0} p' \xrightarrow{u} q \) implies \( p \xrightarrow{u} q \) for some \( u \) such that \( a_0.u \leq_C v \). By transitivity and substitutivity, \( a_0 \ldots a_n \leq_C a_0.u \leq_C v \).

(3) By induction on the proof of the derivation \( p \xrightarrow{u} q \). The only interesting case is the following:

- \( p \equiv p_1|p_2 \xrightarrow{u} q \). By symmetry and the pomset operational semantics, we may restrict ourselves to considering the following two cases.

  (a) \( p_1 \xrightarrow{u} q_1 \) and \( q_1|p_2 \equiv q \). By the inductive hypothesis, \( p_1 \xrightarrow{a_0 \ldots a_n} q_1 \) for some \( a_0 \ldots a_n \leq_C u \). Then \( p_1|p_2 \xrightarrow{a_0 \ldots a_n} q_1|p_2 \equiv q \).

  (b) \( p_1 \xrightarrow{u_1} q_1 \), \( p_2 \xrightarrow{u_2} q_2 \), \( q \equiv q_1|q_2 \) and \( u \equiv u_1|u_2 \). By the inductive hypothesis, there exist \( \sigma = a_0 \ldots a_n \leq_C u_1 \) and \( \omega = b_0 \ldots b_m \leq_C u_2 \) such that \( p_1 \xrightarrow{\sigma} q_1 \) and \( p_2 \xrightarrow{\omega} q_2 \). Obviously, \( p_1|p_2 \xrightarrow{u_1|u_2} q_1|q_2 \equiv q \). It remains to be shown that \( a_0 \ldots a_n.b_0 \ldots b_m \leq_C u_1|u_2 \), but this follows from proposition 3.4. \( \square \)

**Theorem 4.3** Let \( p \in \mathcal{P}_{\text{Seq}} \) and \( q \in \mathcal{P} \). Then \( p \sim q \) iff \( p \equiv q \).

**Proof:** Again, the if implication follows from Theorem 4.1. In order to prove the only if implication, it is sufficient to show that the relation

\[ \mathcal{R} := \{ (p, q) \mid p \in \mathcal{P}_{\text{Seq}} \text{ and } p \sim q \} \]

is an \( \sim \)-bisimulation. We check that the \( \sim \)-bisimulation clauses are met for \( (p, q) \in \mathcal{R} \).

- Assume \( p \xrightarrow{u} p' \). Then, by lemma 4.2, \( p \equiv a_0 \ldots a_n \). By lemma 4.3, \( p \xrightarrow{u} p' \) implies \( p \xrightarrow{a_0 \ldots a_n} p' \). As \( p \sim q \), there exists \( q' \) such that \( q \xrightarrow{a_0 \ldots a_n} q' \) and \( p' \sim q' \). By statement (2) of the previous lemma, \( q \xrightarrow{a_0 \ldots a_n} q' \) implies \( q \xrightarrow{u} q' \) for some \( u \) such that \( u \leq_C v \). Moreover, \( (p', q') \in \mathcal{R} \) as \( p' \in \mathcal{P}_{\text{Seq}} \).

- Assume \( q \xrightarrow{v} q' \). Then, by statement (3) of the above lemma, \( q \xrightarrow{a_0 \ldots a_n} q' \) for some \( u \equiv a_0 \ldots a_n \leq_C v \). As \( p \sim q \), there exists \( p' \) such that \( p \xrightarrow{a_0 \ldots a_n} p' \) and \( p' \sim q' \). As \( p \in \mathcal{P}_{\text{Seq}} \), by lemma 4.3, \( p \xrightarrow{a_0 \ldots a_n} p' \) implies \( p \xrightarrow{u} p' \). Moreover \( (p', q') \in \mathcal{R} \) as \( p' \in \mathcal{P}_{\text{Seq}} \).

Hence \( \mathcal{R} \) is an \( \sim \)-bisimulation. \( \square \)

A rather pleasing consequence of this result is that, whenever trying to establish that a process \( q \) is a correct implementation of a sequential specification \( p \), i.e. that \( p \equiv q \), it is sufficient to exhibit a standard strong bisimulation containing them.
Pomset bisimulation equivalence, $\approx_{P_{om}}$, has been proposed in [BC87,88] as a modification of the standard notion of bisimulation equivalence which clearly distinguishes concurrency from sequential nondeterminism. As the definition of the preorder $\sqsubseteq$ presented in this section has much in common with that of $\approx_{P_{om}}$, it is natural to investigate the relationships between the kernel of $\sqsubseteq$, $\simeq$, and $\approx_{P_{om}}$.

Let $=P$ denote the least congruence over $Comp$ which satisfies axioms (PAR1)-(PAR3). Then $\approx_{P_{om}}$ is defined as the largest binary, symmetric relation over $P$ which satisfies the following condition:

$$p \approx_{P_{om}} q \text{ if, for each } u \in \text{Comp},$$

$$p \xrightarrow{u} p' \text{ implies, for some } q' \text{ and } u =P u, q \xrightarrow{u} q' \text{ and } p' \approx_{P_{om}} q'. $$

It is easy to see that, because $=P \subseteq \leq_C$ over $Comp$, $\approx_{P_{om}}$ is indeed an $\mathcal{F}$-bisimulation. As a consequence of this observation we have the following:

**Fact 4.2** For each $p, q \in P$, $p \approx_{P_{om}} q$ implies $p \approx q$.

However, the converse implication does not hold, i.e. $\simeq$ is strictly weaker than $\approx_{P_{om}}$. Consider, in fact, the following two processes:

$$p \equiv a.b.c + c.a.b + a|b|c,$$

$$q \equiv p + a.b|c.$$ 

Obviously, $p \not\approx_{P_{om}} q$ as $q \xrightarrow{a.b|c}$, but $p \xrightarrow{u}$ for no $u =P a.b|c$. However, we do have that $p \simeq q$. This essentially depends on the fact that, because $a.b.c \leq_C a.b|c \leq_C a|b|c$, the transition $q \xrightarrow{a.b|c} \text{nil|nil}$ can be matched “in a more sequential fashion” by $p \xrightarrow{a.b|c} \text{nil}$ and “in a more parallel one” by $p \xrightarrow{a|b|c} \text{nil|nil|nil}$.

## 5 Algebraic characterization of the preorder

The purpose of this section is to axiomatize the preorder $\sqsubseteq$ defined in the previous section over the set of finite processes $P_{Fin}$. As noted by G. Boudol and I. Castellani in [BC87,88], the interpretation of processes given by a pomset transition system semantics is just an ordinary labelled transition system over a set of actions. The only difference being that the actions are “structured”.

Following M. Hennessy and R. Milner [HM85], there is a standard way of axiomatizing bisimulation-like relations over ordinary, finite, acyclic labelled transition systems. Their method involves the reduction of terms to so-called sumforms over the set of actions into consideration. As in our pomset transition system semantics for the language $P$ processes evolve by performing computations and computations are not in the signature for processes themselves, this method is not directly applicable to the language $P$.

In order to apply Hennessy and Milner’s technique to provide an axiomatization for the preorder $\sqsubseteq$ over $P_{Fin}$, we thus need to extend the language $P$ to $P'$, where $P'$ is built as $P$ with the additional
formation rule:
\[ u \in \text{Comp} \text{ and } p \in P' \text{ imply } u : p \in P'. \]

Thus the signature of the language \( P \) has been extended by allowing prefixing operators of the form \( u : - \), for \( u \in \text{Comp} \). The language \( P' \) thus allows one to prefix computations to processes and this is what will be needed to define a suitable set of sumforms for \( P' \). The operational semantics for \( P' \) is obtained by extending the rules in definition 2.3 with the axiom
\[
(PRE) \quad u : p \xrightarrow{u} p.
\]

It is easy to see that \( \subseteq \) can be conservatively extended to the language \( P' \) and that the following proposition holds:

**Proposition 5.1** \( \subseteq \) is a \( P'_{\text{Fin}} \)-precongruence.

**Definition 5.1 (Sumforms)** The set of sumforms over \( \text{Comp} \), \( \text{SF}(\text{Comp}) \), is the least subset of \( P'_{\text{Fin}} \) which satisfies:

1. \( \text{nil} \in \text{SF}(\text{Comp}) \),
2. \( u \in \text{Comp} \text{ and } p \in \text{SF}(\text{Comp}) \text{ imply } u : p \in \text{SF}(\text{Comp}) \),
3. \( p, q \in \text{SF}(\text{Comp}) \text{ imply } p + q \in \text{SF}(\text{Comp}) \).

In order to give a complete axiomatization for \( \subseteq \) over \( P'_{\text{Fin}} \) (and, consequently, over \( P_{\text{Fin}} \)), it will be sufficient to devise a set of axioms which allow us to reduce terms in \( P'_{\text{Fin}} \) to sumforms and which are complete for \( \subseteq \) over \( \text{SF}(\text{Comp}) \). Formally, the theory we shall consider is the two-sorted theory consisting of the set of axioms \( C \) over \( \text{Comp} \), the set of axioms \( A \) given in Figure 2 together with the following rule
\[
(SUB) \quad u \leq_C v, p \leq q \text{ imply } u : p \leq v : q.
\]

Note that the axioms in \( C \) are only needed to relate computations; indeed, (PAR1)-(PAR3) are valid for processes as well (although we have no use for them in the equational theory for processes) whilst (SEQ) is not sound for processes. In fact, \( a.b \leq_C a|b \), but \( a.b \not\subseteq a|b \). It is worth pointing out that all the axioms in \( A \) have an equational nature, i.e., they essentially express properties of the kernel of \( \subseteq \). Thus all the essence of the preorder \( \subseteq \) is captured by the inequational theory of computations and namely by axiom (SEQ). Axiom (R1) expresses the interplay between the two kinds of prefixing operators in the language \( P' \). It will be used in reducing processes to sumforms to eliminate occurrences of the action-prefixing operators \( a\ldots \) in terms in favour of the new computation-prefixing operators \( u : \ldots \).

From now onwards, sumforms will be written as \( \sum_{i \in I} u_i : p_i \), where, for each \( i \in I \), \( u_i \in \text{Comp} \) and \( p_i \) is a sumform. This notation is justified by axioms (A1)-(A2). If \( I = \emptyset \) then, by convention, \( \sum_{i \in I} u_i : p_i \equiv \text{nil} \). Let \( \leq_A \) be the least precongruence over \( P'_{\text{Fin}} \) generated by the set of axioms in Figure 2 and rule (SUB). The following proposition, whose proof is omitted, states the soundness of the proof system with respect to \( \subseteq \).
The set of axioms $C$

\begin{align*}
(PAR1) \quad x | nil &= x \\
(PAR2) \quad x | y &= y | x \\
(PAR3) \quad (x | y) | z &= x | (y | z) \\
(SEQ) \quad a. (x | y) &\leq a. x | y
\end{align*}

The set of axioms $A$

\begin{align*}
(A1) \quad (x + y) + z &= x + (y + z) \\
(A2) \quad x + y &= y + x \\
(A3) \quad x + nil &= x \\
(A4) \quad x + x &= x \\
(R1) \quad a. (\sum_{i \in I} u_i : p_i) &= (a. nil) : (\sum_{i \in I} u_i : p_i) + \sum_{i \in I} (a. u_i) : p_i \\
(R2) \quad (\sum_{i \in I} u_i : p_i) (\sum_{j \in J} v_j : q_j) &= \sum_{i \in I} u_i : (p_i | \sum_{j \in J} v_j : q_j) + \sum_{j \in J} v_j : (\sum_{i \in I} u_i : p_i | q_j) \\
&\quad + \sum_{i \in I, j \in J} (u_i | v_j) : (p_i | q_j)
\end{align*}

Figure 2: A complete set of axioms for $\subseteq$

**Proposition 5.2 (Soundness)** For each $p, q \in P_{Fin}^I$, $p \leq_A q$ implies $p \subseteq q$.

As usual, the key of the proof of completeness is to show that each term may be reduced to a suitable normal form using the axioms. Following [HM85], our normal forms will be the sunforms of definition 5.1.

**Lemma 5.1 (Normalization)** For each $p \in P_{Fin}^I$, there exists a sumform $sf(p)$ such that $p =_A sf(p)$.

We can now prove the promised completeness theorem.

**Theorem 5.1 (Completeness)** For each $p, q \in P_{Fin}^I$, $p \subseteq q$ iff $p \leq_A q$.

**Proof:** The if implication follows by Proposition 5.2. Assume then that $p, q \in P_{Fin}^I$ and $p \subseteq q$. The proof is by induction on the combined size of the terms. By the normalization lemma we may assume, wlog, that $p$ and $q$ are sumforms. Thus assume that $p = \sum_{i \in I} u_i : p_i$ and $q = \sum_{j \in J} v_j : q_j$.

- First of all we show that $p + q \leq_A q$. As $p \subseteq q$, for each $i \in I$ there exists $h(i) \in J$ such that $u_i \leq C v_{h(i)}$ and $p_i \subseteq q_{h(i)}$. As $p_i$ and $q_{h(i)}$ are sumforms and their combined size is less than that of $p$ and $q$, we may apply the inductive hypothesis to obtain $p_i \leq_A q_{h(i)}$. By rule (SUB), $u_i \leq C v_{h(i)}$ and $p_i \leq_A q_{h(i)}$ imply $u_i : p_i \leq_A v_{h(i)} : q_{h(i)}$. As this is the case for each $i \in I$, by substitutivity

  \[ p \equiv \sum_{i \in I} u_i : p_i \leq_A \sum_{i \in I} v_{h(i)} : q_{h(i)}. \]
Thus,

\[ p + q \leq_A \sum_{i \in I} v_{k(i)} : q_{h(i)} + q \]

by substitutivity

\[ =_A q \]

by (A1)-(A4).

- We show that \( p \leq_A p+q \). As \( p \leq_A q \), for each \( j \in J \) there exists \( k(j) \in I \) such that \( u_{k(j)} \leq_C v_j \) and \( p_{k(j)} \leq q_j \). By the inductive hypothesis, \( p_{k(j)} \leq_A q_j \) and, by rule (SUB), \( u_{k(j)} : p_{k(j)} \leq_A v_j : q_j \). As this is the case for each \( j \in J \), by substitutivity

\[ \sum_{j \in J} u_{k(j)} : p_{k(j)} \leq_A \sum_{j \in J} v_j : q_j \equiv q. \]

Thus, again by substitutivity, \( p + \sum_{j \in J} u_{k(j)} : p_{k(j)} \leq_A p + q \) and, by (A1)-(A4),

\[ p =_A p + \sum_{j \in J} u_{k(j)} : p_{k(j)} \leq_A p + q. \]

Thus we have proved that \( p \leq_A p + q \leq_A q \) and, by transitivity, \( p \leq_A q \). \( \square \)

6 A simple example

This section is devoted to the discussion of a simple example of application of the theory developed in the previous sections to the specification of concurrent systems. The example is based upon the definition of an \( n \)-ary semaphore, which admits any sequence of get’s and put’s in which the number of get’s minus the number of put’s lies in the range 0 to \( n \) inclusive. The following description of an \( n \)-ary semaphore is taken from [Mil89]:

\[
\begin{align*}
\text{Sem}_n(0) & \iff \text{get} \cdot \text{Sem}_n(1) \\
\text{Sem}_n(k) & \iff \text{get} \cdot \text{Sem}_n(k + 1) + \text{put} \cdot \text{Sem}_n(k - 1) \quad \text{if} \ 0 < k < n \\
\text{Sem}_n(n) & \iff \text{put} \cdot \text{Sem}_n(n - 1),
\end{align*}
\]

where get and put correspond to Dijkstra’s \( P \) and \( V \) operations on a semaphore, respectively. The simplest semaphore is the unary one, whose definition reduces to

\[
\text{Sem} \iff \text{get} \cdot \text{put} \cdot \text{Sem}.
\]

Let us assume now that we want to give the specification of a \((n + 1)\)-semaphore, \( n \geq 1 \), which is to be implemented on a multiprocessor having at least \( k \) processors, for some \( k > 1 \). Moreover, we want to specify the \((n + 1)\)-semaphore in such a way that an implementation be “forced” to exploit this feature of our system, i.e. we require that the degree of parallelism of any actual implementation be “at least \( k \)” in some formal sense. This is not possible in the semantic theory of standard bisimulation equivalence because of the reduction of parallelism to sequential nondeterminism it enforces, but, as shown in what follows, it can be done by using the preorder \( \leq \) as a formal “implementation ordering” over \( P \). As already remarked, the semantic statement \( p \leq q \) will be interpreted as meaning that \( q \) is
an implementation of \( p \) and we require implementations to exhibit at least all the parallelism present in the specification.

One way to incorporate the above-given requirement into the specification of our semaphore is to stipulate, for instance, that

\[
SPEC \Leftrightarrow Sem_{n-k+2}(0)|Sem|\cdots|Sem,
\]

\((k-1)\)-times

where \( Sem_{n-k+2}(0) \) is the sequential nondeterministic description of an \((n-k+2)\)-semaphore. It is easy to see that \( SPEC \) is indeed functionally equivalent to an \((n+1)\)-semaphore in the theory of bisimulation equivalence as

\[
SPEC \sim Sem_{n+1}(0).
\]

If during the implementation process it is realized, for instance by means of performance tests, that our multiprocessor system is capable of supporting the efficient, concurrent execution of more than \( k \) processes, the efficiency of the semaphore might be increased by maximizing the parallelism in its implementation. For instance, \( Sem_{n-k+2}(0) \) might itself be implemented by

\[
Sem^{n-k+2} \Leftrightarrow Sem|\cdots|Sem,
\]

\((n-k+2)\)-times

the system consisting of \((n-k+2)\) unary semaphores running in parallel. In order to show that

\[
IMP \Leftrightarrow Sem|\cdots|Sem.
\]

\((n+1)\)-times

is a correct implementation of \( SPEC \) with respect to \( \sqsubseteq \), i.e. that \( SPEC\sqsubseteq IMP \), it is sufficient to prove that

\[
Sem_{n-k+2}(0) \sqsubseteq Sem|\cdots|Sem,
\]

\((n-k+2)\)-times

as the claim will then follow by substitutivity. A proof of (1) can be given by exhibiting an \( \mathcal{R} \)-bisimulation \( \mathcal{R} \) containing the pair \((Sem_{n-k+2}(0),Sem^{n-k+2})\).

Moreover, as \( Sem_{n-k+2}(0) \) is in \( \mathcal{F}_{seq} \), by Theorem 4.3 it is indeed sufficient to check that the two systems are bisimilar. This can be done by using the standard techniques supported by the theory of bisimulation equivalence.

This proof technique supported by the preorder \( \sqsubseteq \) is quite pleasing because refinements of specifications into implementations frequently proceed by substituting parallel processes for sequential nondeterministic ones in a context. Hence, in most practical cases, checking that \( SPEC\sqsubseteq IMP \) reduces to showing bisimilarity between a sequential nondeterministic process \( SEQ \) and a process \( IMPSEQ \) or to a combination of such proofs. Moreover, several automated tools for checking bisimilarity among labelled transition systems are now available, e.g. [CPS89].
Note, however, that $Sem_{n+1}(0)$ would not be a correct implementation of $SPEC$ with respect to $\sim$. In fact, $SPEC \not\sim Sem_{n+1}(0)$ as $SPEC \not\leadsto get^k$, where

$$get^k \equiv get \ldots \cdot get, \quad k\text{-times}$$

whilst $Sem_{n+1}(0) \rightarrow u$ for no $u$ such that $get^k \leq_C u$. Intuitively, the specification permits $k$ concurrent accesses to the semaphore, but $Sem_{n+1}(0)$ permits only one access at the time. This formalizes the intuitive fact that $Sem_{n+1}(0)$ does not capture all the parallelism which is present in the specification $SPEC$.

We hope that this very simple example shows at least that, by using $\sim$ as a formal implementation ordering, it is possible to require that implementations preserve all the parallelism already present in the specification of a system. As already remarked, this is not possible in the theory of standard bisimulation equivalence. On the other hand, semantic statements of the form $SPEC \sim IMP$ may be often shown by employing the elegant and compositional proof techniques supported by such an interleaving equivalence.

7 Conclusion

In this paper, we have presented a semantic theory for a simple CCS-like language which allows to relate concurrency and nondeterminism in the behaviour of concurrent processes without semantically reducing the former to the latter. The theory is based on a simple behavioural preorder, $\subseteq$, which relates processes $p$ and $q$, $p \subseteq q$, if $p$ and $q$ have the same observable behaviour and $q$ is at least as parallel as $p$. The preorder has been defined by means of a slight modification of the notion of pomset bisimulation equivalence [BC87,88], obtained by parameterizing it with respect to a natural preorder over computations. This preorder on computations measures their relative degree of parallelism. Moreover, we have shown how this preorder can be axiomatized over the set of recursion-free processes by means of standard techniques used in the literature to give equational characterizations of bisimulation-like relations.

The semantic theory for processes developed in this paper enjoys some pleasing relationships with that of strong bisimulation equivalence and supports many of the proof techniques which are familiar from the theory of such an interleaving equivalence. On the other hand, it has the advantage of allowing to take into consideration the relative degree of parallelism between specifications and implementations, for instance by forcing implementations to exhibit at least all the parallelism which is present in the specification. This is not possible in the theory of standard strong bisimulation because of the semantic reduction of parallelism to sequential nondeterminism it enforces.

As it is often the case with bisimulation-like relations, the preorder presented in this paper may be given a pleasing logical characterization by means of a variation of Hennessy-Milner logic, $HML$. Let $HML(Comp)$ denote an infinitary Hennessy-Milner logic whose modalities are indexed by computations $u \in Comp$ (infinite conjunctions and disjunctions are needed as the pomset transition system
The satisfaction relation $\models \subseteq \mathcal{P} \times \text{IEL}(\text{Comp})$ is defined in a standard way following the above given references. The only interesting clauses of its definition are the ones regarding the modalities $\langle u \rangle$ and $[u]$; these read as follows:

$$
p \models \langle u \rangle \phi \quad \text{iff} \quad \exists v \in \text{Comp}, q \in \mathcal{P} \text{ such that } u \leq_C v, p \xrightarrow{v} q \text{ and } q \models \phi

p \models [u] \phi \quad \text{iff} \quad \forall v \in \text{Comp} \quad v \leq_C u \text{ and } p \xrightarrow{v} q \implies q \models \phi.
$$

It is then possible to prove using standard techniques that, for processes $p$ and $q$,

$$p \leq_C q \quad \text{iff, for each } \phi \in \text{IEL}(\text{Comp}), p \models \phi \implies q \models \phi.$$

This paper is undoubtedly just a first attempt at giving a semantic theory for processes which reconciles both an interleaving and a "truly concurrent" approach to the semantics of concurrency. More work remains to be done in extending the results presented in this paper to languages incorporating features like communication, silent moves and restriction/hidden operations as these concepts have been shown to be of great significance in the specification of realistic systems [Hoare85], [Mil89]. The approach followed in this paper relied on a pomset operational semantics [BC87,88] and a bisimulation-like preorder. It is a challenging open problem to study whether the methods used in this paper extend to more powerful languages or whether alternative forms of operational semantics, e.g. the distributed operational semantics of [CH87] or the timed operational semantics of [H88b], are more viable in general.

We end this conclusion with a brief discussion of related work. The preorder presented in this paper may be seen as an instance of the general scenario developed in [Thom87]1. There the author studies a bisimulation preorder over Labelled Transition Systems induced by a given, uninterpreted preorder on actions. Our $\leq_C$ may be seen as arising from this setting by interpreting all the parameters of Thomsen's proposal and, in particular, by using $\leq_C$ as the preorder on actions.

The preorder on computations $\leq_C$ used in the paper is related to the notion of augmentation presented by several authors in the literature on pomsets and partial words [Gr81], [Gi84], [Pr86]. Essentially, for pomsets $P$ and $Q$, $P$ is an augmentation of $Q$ if $P$ is obtained from $Q$ by adding more causal dependencies to those of $Q$. This notion is intuitively related to the model-theoretic counterpart of $\leq_C$, $\prec$.

Recently, S. Abramsky has independently proposed a notion of "concurrent refinement" over a rich process language which is closely related to the one presented in this paper, see [Ab90]. In particular, Abramsky also uses Gischer's augmentation preorder over pomsets to induce a preorder over processes. However, the paper [Ab90] is mostly concerned with a deep study of pomset bisimulation equivalence and the refinement preorder over processes is not studied in detail.

---

1I thank Kim Larsen and Arne Skou for pointing out this reference.
8 References


[CH89] I. Castellani and M. Hennessy, Distributed Bisimulations, J. ACM, October 1989


