1 Introduction

Concurrent and distributed systems are becoming of great practical importance as computer science takes advantage of the opportunities of new technologies. When modelling computations of distributed systems, it is quite important to deal with properties of their behaviour which are local, in that they are defined on subparts of the system, making explicit use of the distributed structure of states. In order to reason about local properties it is necessary to identify the actions which can affect subparts of the system, and so information on causal dependencies is needed. For instance, let us consider a distributed system which after some evolution performs an action causing part of the system to starve. Here it is vital to identify, for a fixed evolution, which part of the system starves, and which does not. Moreover, one would like to identify the actions that can have influenced the occurrence of the starvation.

Many models for concurrent systems have been proposed in the literature. Much work on the semantics of distributed systems has been based on the interleaving approach [Ho 85, Mil 89, BK 84]. In this class of models, a global state is assumed, and the evolution of a system is described in terms of sequences of global states. As a consequence, dealing with local properties is impossible because the corresponding information has been lost in the abstract behaviour.

Instead, true concurrency or partial ordering models [Re 85, NPW 81, DM 87, Pr 86, BC 88] describe the behaviour of distributed systems in terms of the events they may perform, and the constraints on their occurrence: a partial ordering represents the causal dependencies among events, while concurrency is represented by the absence of ordering. True concurrency models provide a more faithful account of distributed computations. They are well suited to handle properties which explicitly refer to the information about distributed activities.

Several research efforts have been devoted to the relation between programming logics and observational models of concurrency. As shown by Abramsky [Ab 88], in the interleaving approach this relation is an extension of the classical Stone duality theorem for boolean algebras. Such duality clarifies the relationships between equivalence classes of computations and properties (described through programming logics) of processes. Presently, it is not clear whether or not this duality holds for the partial ordering approach.

With respect to process description languages, the true concurrency approach has not yet received a completely satisfactory treatment when compared with the results based on interleaving. True concurrency operational semantics have been developed only recently. The basic idea is to provide an interpretation of the language in terms of Petri Nets [DDM 88a, Ol 87, Go 88], Labelled Event Structures [Win 82, DDM 88b], Causal Trees [DD 89], and so on.

Since these descriptions are too concrete, certain behavioural equivalences are introduced by extending the techniques introduced within the interleaving framework [DDM 87, vGG 89, RT 88]. In this way, the problem of finding a truly concurrent semantics is reduced to the problem of defining equivalence classes of programs and computations which express particular aspects of system behaviour with respect to certain notions of observation.

One of the drawbacks of truly concurrent semantics is that the research on obtaining logics equipped with proof systems which emphasize the non sequential properties of processes is at a very preliminary stage. In particular, little is known on the relations (adequacy, expressiveness results) between non interleaving models and logical languages (see [DF 90] for some...
preliminary results on the logical characterization of concurrency preserving behavioural equivalences).

Moreover, in the case of observational semantics (both interleaving and truly concurrent), the standard representatives of the equivalence classes of programs and computations, if any are actually defined, sometimes do not yield minimal realizations, i.e. they do not form a transition system. The minimal realization would represent the most reduced operational semantics with respect to a notion of program transformation which preserves the observable behaviour of programs. Notice that this is a typical situation in automata theory [Gog 72], and in abstract data type specification [GGM 76, Wa 79]. Minimal realizations and the associated transformations are very convenient (think for instance of equivalent circuits in electronics) since they provide an intuitive and suggestive way of handling abstract semantics, and they are quite useful in practice.

The difficulty of having a minimal realization of truly concurrent operational semantics depends on the fact that observable behaviours of machines are inherently incremental: they are obtained by composing the elementary steps of the machines and their observations. On the contrary, truly concurrent behavioural equivalences are defined by observing just the global outcome of computations. The main problem is that in the case of partial ordering models there is no obvious operation of sequential composition of partial orders which preserves all the information on causal dependencies.

This paper aims at solving these problems. Our first starting point is the definition of the semantics of process description languages in terms of categories of transition systems with algebraic structure both on states and transitions [FM 90, Fe 90]. In this approach, the observation mechanism of computations is handled by a labelling or typing technique: every computation is labelled (typed) with its observations. In this framework, we consider behavioural equivalences based on the notion of bisimulation [Pa 81]. In [FM 90, Fe 90] it is shown that the strong observational congruence [Mil 80] (the simplest bisimulation equivalence) can be characterized in an algebraic way by considering special simplification morphisms which preserve the algebraic structure of states and transitions, the observations and the transitions outgoing from any state. It turns out that the strong observational congruence is characterized by a universal property of finality: the terminal object is a transition system whose states and transitions are congruence classes of agents and computations, i.e. a minimal realization.

This schema can be applied only to behavioural equivalences which are at the same time congruences and bisimulations. For instance this schema does not work in the case of Weak Observational Congruence (an equivalence which forgets about internal invisible moves) because the weak congruence is not a bisimulation 1. Bisimulation equivalences which are also congruences can be characterized in a denotational setting [Ab 88]. Furthermore, in the functional case, they capture the notion of dynamic reconfiguration of the structure of machines. The Dynamic Bisimulation [MS 90] is a bisimulation which tests the observable behaviour of processes also when they are dynamically embedded in the same context. The dynamic bisimulation is the coarsest interleaving weak bisimulation which is also a congruence.

The second starting point is the notion of Concatenable Processes. Although Petri Non Sequential Processes [GR 83] have information on both causality and distribution, they lack an operation of sequential composition. The problem of having incremental descriptions of

---

1The states α, τ, β, nil and α, β, nil are observationally congruent, where τ is the invisible action, but the states τ, β, nil and β, nil they reach after performing an α transition are not. Thus weak observational congruence is not a bisimulation relation.
Non Sequential Processes has been successfully tackled by Degano, Meseguer and Montanari [DMM 89]. A Petri Net is seen as a graph with a monoidal operation expressing parallel composition of places and transitions. The free category generated by the graph is introduced and certain axioms are used to define a quotient on its morphisms. Indeed, morphisms can be seen as terms of an algebra with two operations, parallel and sequential composition. The congruence classes of these terms can be represented by Concatenable Processes, which are based on Non Sequential Processes but have extra information which allows sequential composition to be defined. However, Concatenable Processes are still unsatisfactory since the Petri Nets they are based on are not labelled by actions, and since they retain information about the intermediate states.

The basic idea of the paper is the definition of an algebraic theory of process description languages (models and logics) where information on causal dependencies and distribution is properly taken into account, and which allows an incremental approach to the description of computations. We take CCS [Mil 80] as a case study.

In our framework, a transition system (model) for the CCS language consists of a typed algebra [MSS 89]. The algebraic structure on elements of type \textit{state} is given by the language itself; other elements describe transitions, and computations. Instead of considering a single model of the language, we consider a collection of such models: each model represents a specific abstract machine (an interpreter) for the language. The collection of models forms a category, where the morphisms preserve the algebraic structure and represent relations between different abstract machines.

CCS models are not necessarily free models of the presentation given: nonfreeness may reflect a particular interpretation of the operations. This allows us to identify a particular model in the collection of models which plays the role of the model of the observations: the interpretation of the operations on computations implements a calculus. By giving different interpretations to the operations we can define several calculi of computations. Because we are interested in a truly concurrent semantics, the calculus of computations will be a calculus of partial orderings.

We introduce the algebra of \textit{Concatenable Concurrent Histories}, which are essentially Concatenable Processes with action labelled events and without information about the intermediate states. For the purpose of the present paper, it is important to know that the operation of sequential composition of computations is one of the basic operations of the algebra of Concatenable Concurrent Histories. This algebra is closely related to the model of Concurrent Histories, developed by Degano and Montanari [DM 87].

We show that in the category of CCS models there is an object for which the computations have the structure of the algebra of Concatenable Concurrent Histories. This model constitutes our algebra of observations. As a consequence the observations are incremental. Intuitively, an observation is just the partial ordering of the events performed plus extra information about spatial distribution of the initial and final states of the computation.

Observing the computations of a CCS model means finding a morphism from it to the model of observations. This construction builds a category whose objects are CCS models with computations labelled by Concatenable Concurrent Histories, and whose morphisms preserve the observations as well as the algebraic structure. The labelling construction is defined in categorical terms: it is an instance of the general \textit{Comma Category} operation [ML 71].
The category of CCS abstract machines where causality is observed incrementally provides the formal apparatus to introduce and study the features of behavioural equivalences (and congruences) together with the logics which describe properties of computations.

In this paper we consider behavioural equivalences based on the notion of bisimulation [Pa 81]. As in the case of interleaving semantics, we first introduce a bisimulation equivalence over the elements of type state of the initial object (the most concrete semantics). We then prove that this equivalence is also a congruence, and we show that it gives rises to a minimal realization. In other words, the quotient of the initial object with respect to such a congruence is a CCS model, i.e. an abstract machine of the language, which is a terminal object in the appropriate subcategory of CCS models.

To prove this universal property of finality, we take a subcategory, whose objects are the images of the initial object under some fixed type of simplification mappings and whose morphisms are the simplification mappings themselves. The minimal realization (when it exists) is the final object of this subcategory. In this paper the simplification mappings which we consider are strict transition preserving homomorphisms, a variant of the transition preserving homomorphisms introduced in [DDM 88a, AD 89, FM 90]. We prove that when the observations are Concatenable Concurrent History the final object exists, and, moreover, the unique mapping from the initial to the final object fully characterizes the bisimulation congruence.

In our framework, logics which describe properties of the observable behaviour of computations can be automatically derived by considering Dynamic Logics [Ha 84] which are special modal logics where the modalities are parameterized by terms of the calculus of computations. Notice that the modal schema (axiom) which corresponds to the notion of incremental description of computations is \([t_1][t_2] \varphi \rightarrow [t_1; t_2] \varphi\). This feature can be profitably exploited for defining proof systems emphasizing the non sequential aspects of computations. This topic will be subject of further studies. In this paper, we simply introduce a modal logic (in the style of Hennessy-Milner Logic [HM 85]) whose modalities are parameterized by Concatenable Concurrent Histories. We show that our collection of models provides the right framework to give an interpretation of this logic and to understand the relations with the observational semantics. In fact, it turns out that the equivalence induced by the logic coincides with the equivalence induced by the final object.

In the interleaving semantics of CCS, a special action, the \(\tau\) action, is used to indicate the occurrence of invisible internal operations. Only the proper treatment of \(\tau\) actions provides us with a semantics of concurrency (as opposed to explicit time). Forgetting \(\tau\) actions in the observations means performing an abstraction operation. In the standard interleaving approach, Weak Observational Equivalence takes care of this abstraction. However, in this way we get an equivalence which is not a congruence, and therefore we do not have a minimal realization. As mentioned before, the weak observational congruence does not solve the problem since it is not a bisimulation. A minimal realization can be found by considering dynamic bisimulation [MS 90] instead, which is also a congruence.

In our framework the same approach can be applied. Invisible actions can be handled by modifying the observations so that they are forgotten, but it is still possible to distinguish between an idle system and a system performing an invisible move. Also in this case we get both an algebraic and a logical characterization of a weak partial ordering semantics.
2 The Algebra of Concatenable Concurrent Histories

In this section, we introduce the algebra of Concatenable Concurrent Histories. For the following definitions, we fix two nonintersecting alphabets $P$, $A$. Intuitively, $P$ represents the set of process names, while $A$ represents the set of action names.

**Definition 1 (Concatenable Histories)**

Concatenable Histories are labelled partial orders $(V, \leq, \ell)$ where the labelling function $\ell : V \to P \cup A$ sends the set of maximal and minimal elements to $P$ and the set of other elements to $A$. Concatenable Histories are considered up to isomorphisms of labelled partial orders.\[\square\]

The elements with labels in $P$ are called *processes*, those with labels in $A$ are called *events*.

**Definition 2 (Label Indexed Ordering Functions)**

Suppose $S$ is a set with a labelling function $\ell : S \to P$. A label indexed ordering function on the labelled set $S$ is a function $\alpha$ from $S$ to the set of natural numbers, such that for each $p \in P$ the restriction of $\alpha$ to the set of elements labelled $p$ is a bijection on the set $\{1, 2, \ldots, n_p\}$ where $n_p = |\{s \in S : \ell(s) = p\}|$.\[\square\]

**Definition 3 (Concatenable Concurrent Histories)**

A Concatenable Concurrent History is a triple $(h, \beta, \gamma)$ where $h = (V, \leq, \ell)$ is a Concurrent History for which no element is both maximal and minimal, and $\beta, \gamma$ are label indexed ordering functions on the labelled sets of minimal and maximal elements of $V$ (called origins and destinations), respectively. Concatenable Concurrent Histories are defined up to isomorphisms of labelled partial orders that preserve the label indexed ordering functions.\[\square\]

The introduction of the label indexed ordering functions allows us to discriminate between different maximal elements (and minimal elements) with the same label. Figure 1 illustrates two concatenable concurrent histories $ch_1$ and $ch_2$ ((a) and (b)); the order relation is depicted through its Hasse diagram growing downwards. Processes (resp. events) are represented as circles (boxes): all processes have the same label. Finally, the label indexed ordering functions are represented by positive numbers on processes.

The algebra of Concatenable Concurrent Histories has two operations: parallel composition and sequential composition. Let $ch_1 = (h_1, \beta_1, \gamma_1)$ and $ch_2 = (h_2, \beta_2, \gamma_2)$ be Concatenable Concurrent Histories, where $h_1 = (V_1, \leq_1, \ell_1)$ and $h_2 = (V_2, \leq_2, \ell_2)$. Without loss of generality, $V_1$ and $V_2$ are disjoint. Let $\text{Min}(h_1), \text{Max}(h_1), \text{Min}(h_2), \text{Max}(h_2)$ be the origins and destinations of $h_1$ and $h_2$, respectively.

**Definition 4 (The Algebra of Concatenable Concurrent Histories)**

The parallel composition $ch_1 \otimes ch_2$ is the Concatenable Concurrent History $((V, \leq, \ell), \beta, \gamma)$ where
Figure 1: Two Concatenable Concurrent Histories

- $V = V_1 \cup V_2$
- $\leq$ is $\leq_1 \cup \leq_2$
- $\ell(v)$ is $\ell_1(v)$ if $v \in V_1$ and is $\ell_2(v)$ if $v \in V_2$
- $\beta(v)$ is $\beta_1(v)$ if $v \in \text{Min}(h_1)$ and is $n_1(v) + \beta_2(v)$ if $v \in \text{Min}(h_2)$, where $n_1(v)$ is the number of origins of $h_1$ with label equal to $\ell_2(v)$
- $\gamma(v)$ is $\gamma_1(v)$ if $v \in \text{Max}(h_1)$ and is $n_1(v) + \gamma_2(v)$ if $v \in \text{Max}(h_2)$, where $n_1(v)$ is the number of destinations of $h_1$ with label equal to $\ell_2(v)$

The sequential composition $ch_1; ch_2$ is defined if and only if $\ell_1(\text{Max}(h_1)) = \ell_2(\text{Min}(h_2))$, intended as multisets. In this case the result of the operation is the Concatenable Concurrent History $((V, \leq, \ell, \beta, \gamma)$ where

- $V = V_1 \cup V_2 \setminus (\text{Max}(h_1) \cup \text{Min}(h_2))$
- $\leq$ is the restriction to $V \times V$ of the transitive closure of $\leq_1 \cup \leq_2 \cup \{(w,v) : w \in \text{Max}(h_1), v \in \text{Min}(h_2), \ell_1(w) = \ell_2(v), \gamma_1(w) = \beta_2(v)\}$
- $\ell(v)$ is $\ell_1(v)$ if $v \in V_1$ and is $\ell_2(v)$ if $v \in V_2$
- $\beta(v) = \beta_1(v)$
- $\gamma(v) = \gamma_2(v)$

Figure 2 shows the result of the parallel composition $ch_1 \circ ch_2$ (a), and the result of the sequential composition $ch_1 ; ch_2$ (b) of the two histories depicted in Figure 1(a) and 1(b).
3 Algebraic Models for CCS

In this section we provide an algebraic semantics for CCS in terms of typed algebras [MSS 89]. Let $\Delta$ be the alphabet of actions, and $\overline{\Delta}$ the alphabet of complementary actions (with $\Delta = \overline{\Delta}$). Let $\tau \notin \Delta \cup \overline{\Delta}$ be the invisible action, and let $\Lambda = \Delta \cup \overline{\Delta} \cup \{\tau\}$ (ranged over by $\mu$) be the set of actions. We first recall that a CCS expression $E$ has the following syntax,

$$E := \text{nil} \mid x \mid \mu.E \mid E \alpha \mid E[\Phi] \mid E + E \mid E \mid \text{rec } x.E$$

where $x$ is a variable belonging to a set $\text{Var}$ of variables, and $\Phi$ is a permutation of $\Lambda$ fixing $\tau$ and the operation of complementation. A CCS agent is a CCS expression without free variables. A guarded CCS agent is a CCS agent where each occurrence of a variable is within a subexpression $\mu.E$, for some CCS expression $E$.

We introduce now an algebraic model of CCS using the theory of typed algebras. The idea is to represent CCS agents and computations as elements of an algebra equipped with a binary typing relation, which assigns types to elements. Types are just (special) elements of the algebra. The typing information allows us to identify the elements of the algebra which are agents, and the elements which are computations. In the following, we will use $u \rightarrow \mu \rightarrow v$ to indicate a special operator having $u$, $v$ and $\mu$ as arguments. Terms $u \rightarrow \mu \rightarrow v$ will indicate the type of transitions with $u$ as source, $v$ as target and action $\mu$ as label. Similarly we will define $u \Rightarrow v$ to be the type of computations. In the presentation we will use $x : t$ to indicate that the element $x$ has type $t$. 

Figure 2: Parallel (a) and sequential composition (b) of the histories in Figure 1(a) and 1(b)
Definition 5 (CCS Model)

A CCS model is a typed algebra (with multityping) which is a model of the following presentation. An element of this algebra is typed state provided that it is a CCS expression which is both closed and guarded\(^2\). Moreover, there is an operator for each of the SOS-style rules in the operational semantics of CCS and CCS models satisfy the following.

\[
\begin{align*}
\text{Definition 5 (CCS Model)}
\end{align*}
\]

\[
\begin{align*}
&v : \text{state} \\
&\mu, v >: \mu. v - \mu \rightarrow v
\end{align*}
\]

\[
\begin{align*}
&\begin{array}{l}
t : u - \mu \rightarrow v \\
\end{array} \\
&t[\Phi] : u[\Phi] - \Phi(\mu) \rightarrow v[\Phi]
\end{align*}
\]

\[
\begin{align*}
&\begin{array}{l}
t : u - \mu \rightarrow v, w : \text{state} \\
\end{array} \\
&t < + w : u + w - \mu \rightarrow v
\end{align*}
\]

\[
\begin{align*}
&\begin{array}{l}
t : u - \mu \rightarrow v, w : \text{state} \\
\end{array} \\
&w[ t ] : w - u - \mu \rightarrow v | v
\end{align*}
\]

\[
\begin{align*}
&t_1 : u_1 - \lambda \rightarrow v_1, t_2 : u_2 - \lambda \rightarrow v_2 \\
&t_1 | t_2 : u_1 | u_2 - \tau \rightarrow v_1 | v_2
\end{align*}
\]

\[
\begin{align*}
&\begin{array}{l}
t : u - \mu \rightarrow v \\
\end{array} \\
&\text{idle}(v) : v \Rightarrow v
\end{align*}
\]

\[
\begin{align*}
&c_1 : u \Rightarrow v, c_2 : v \Rightarrow w \\
&c_1; c_2 : u \Rightarrow w
\end{align*}
\]

Finally, we require that the following equations\(^3\) are satisfied.

\[
\begin{align*}
&\text{recox}. u : \text{state} \\
&\text{recox}. u = u[\text{recox}. u / x]
\end{align*}
\]

\[
\begin{align*}
&c : u \Rightarrow v \\
&c = \text{idle}(u); c = c_1; \text{idle}(v)
\end{align*}
\]

Definition 6 Let \(M_1\) and \(M_2\) be CCS models. A morphism from \(M_1\) to \(M_2\) is a morphism of algebras that respects the typing. CCS models with their morphisms define the category \(\text{CatCCS}\). \(\Box\)

Proposition 7 \(\text{CatCCS}\) has an initial object \(I\). \(\Box\)

This proposition follows from a general result on categories of typed algebras [MSS 89]. The elements of \(I\) with type state are guarded CCS agents (modulo the equation on recursion), and the elements of type \(u \Rightarrow v\) are the proofs of computations from \(u\) to \(v\).

\(^2\) These conditions can be easily expressed explicitly using typed algebras.

\(^3\) Variable substitution can be easily stated in typed algebras.
4 Observing True Concurrency Incrementally

As it stands, the objects of the category CatCCS do not include any mechanism to observe computations, in that the elements of type $u \Rightarrow v$ represent CCS computations but there is no further abstraction corresponding to what can be detected by an external observer. In our approach [Fe 90, FM 90], the observation mechanism is handled by an operation of labelling (typing) which is internal to the category. We select a specific object of CatCCS to be the model of observations, where the operations are suitably interpreted. Defining an observation mechanism corresponds to selecting a morphism from a CCS model to the model of the observations.

We can now define an observation mechanism for CCS which takes distributed and causal information into account. This is done by choosing, as the model of observations, the CCS model $H$ for which the computations have the structure of the algebra of Concatenable Concurrent Histories.

**Definition 8 (Observing Computations)**

Let CatCCS : $H$ be the category whose objects are pairs $(C, \ell : C \rightarrow H)$ where $C$ is a CCS model, and $\ell$ is a CatCCS morphism. The morphisms of CatCCS : $H$ are maps $\psi$: $(C_1, \ell_1 : C_1 \rightarrow H) \rightarrow (C_2, \ell_2 : C_2 \rightarrow H)$ such that $\psi$ is a CatCCS morphism from $C_1$ to $C_2$ and $\ell_2(\psi(c)) = \ell_1(c)$ for all elements $c$ in $C_1$. □

The definition above expresses that the morphisms of CatCCS : $H$ are morphisms of CatCCS which preserve the labelling (typing). The initial object of CatCCS : $H$ is $(I, \ell_I)$ where $\ell_I$ is the unique morphism from $I$ to $H$.

This is a simple and general construction which enables us to have many different observation mechanisms: it is sufficient to have a different model of observations. For instance, we can have the standard interleaving semantics for CCS by taking a particular algebra of observations whose elements of type $u \Rightarrow v$ are strings of actions [FM 90].

We can now introduce the formal definition of the CCS model $H$ which encodes information about distribution and causality by exploiting the expressive power of the algebra of Concatenable Concurrent Histories.

We first need to introduce some notation. Intuitively, the elements of type state of $H$ represent distributed states of a system of processes. We use $[n]$ to indicate a distributed state having $n$ sequential processes; for instance $[2]$ indicates a distributed state with two sequential processes.

With CCH we indicate the algebra of Concatenable Concurrent Histories with the alphabet $P$ of processes being a singleton, and the alphabet $A$ of events being the set $\Lambda$ of CCS actions. We say that an element of CCH is bipartite if all its elements are either minimal or maximal, and none are both minimal and maximal. We write $idle_1$ for the bipartite history with just two elements and $idle_n$ for the parallel composition of $n$ copies of this, for $n > 1$. We write $\wedge_n$ for the bipartite history with $n + 1$ elements, $n$ of which are maximal, and $\vee_n$ for the bipartite history with $n + 1$ elements, $n$ of which are minimal. We write $t_n$ for the Concatenable Concurrent History with three linearly ordered elements, where the middle...
one is labelled $\mu$, for $\mu \in \Lambda$. Finally, a history $c$ of CCH, with $n$ minimal and $m$ maximal elements, will be typed $[n] \Rightarrow [m]$.

In order to define the CCS model $H$ of causal observations it is enough to define the interpretation of the operations. 4 We start by giving the interpretation of the elements yielding types:

- $\llbracket state \rrbracket = state$,
- $\llbracket u - \mu \rightarrow v \rrbracket = \llbracket u \rrbracket - \mu \rightarrow \llbracket v \rrbracket$,
- $\llbracket u \Rightarrow v \rrbracket = \llbracket u \rrbracket \Rightarrow \llbracket v \rrbracket$

If $u$ is an element of type $state$ then we define the interpretation of $u$, $\llbracket u \rrbracket$, to be as follows.

- $\llbracket nil \rrbracket = [1]$
- $\llbracket \mu . u \rrbracket = [1]$
- $\llbracket u \setminus \alpha \rrbracket = \llbracket u \rrbracket$
- $\llbracket u[\Phi] \rrbracket = \llbracket u \rrbracket$
- $\llbracket [u_1 + u_2] \rrbracket = [1]$
- $\llbracket [u_1] \mid [u_2] \rrbracket = \llbracket u_1 \rrbracket \oplus \llbracket u_2 \rrbracket$ with $[n] \oplus [m] = [n + m]$

Intuitively the interpretation of the elements of type $state$ detects the number of sequential components which can be considered as autonomous processes. Notice that applying the nondeterministic choice operator gives a *global state*, i.e. a state with just one autonomous component. This assumption corresponds to having a centralized mechanism to deal with non deterministic choices (see [DDM 88c, DDM 89] for a deeper discussion on this topic).

The interpretation of operations yielding elements of type $u - \mu \rightarrow v$ is as follows.

- $\llbracket [\mu , [n] >] \rrbracket = t_\mu \land_n$
- $\llbracket \land \alpha \rrbracket = \llbracket t \rrbracket$
- $\llbracket t[\Phi] \rrbracket$ is the Concatenable Concurrent History obtained from $\llbracket t \rrbracket$ by relabelling each element labelled $\mu$ ($\mu \in \mathbb{M}$) with the label $\Phi(\mu)$
- $\llbracket t < + [k] \rrbracket = \llbracket [k] + > t \rrbracket = 2 \land \llbracket t \rrbracket$ if $\llbracket t \rrbracket : [n] - \mu \rightarrow [m]$
- $\llbracket t | [n] \rrbracket = \llbracket t \rrbracket \otimes idle_n$
- $\llbracket [n] | [t] \rrbracket = idle_n \otimes \llbracket t \rrbracket$

4The clauses expressing the interpretation of the operations can be understood as equations of typed algebras. Thus the CCS model $H$ is the initial algebra of the extended presentation given by adding these equations to the original presentation.
\[ [t_1|t_2] = (\eta_1 \otimes \eta_2; (idle_{k_1} \otimes (V_2; t_1; \land_2) \otimes idle_{k_2}); (\gamma_1 \otimes \gamma_2) \text{ where } [t_1]: u_1 - \lambda \to v_1 = (\eta_1; (idle_{k_1} \otimes t_\mu); \gamma_1), \text{ and } [t_2]: u_2 - \lambda \to v_2 = ((\eta_2; t_\mu \otimes idle_{k_2}); \gamma_2) \text{ and } \eta_2, \gamma_1, \eta_2, \gamma_2 \text{ are bipartite.} \]

The interpretation of operations yielding elements of type \( u \mapsto v \) is

- \([idle([n])]) = idle_n\]
- \([c_1; c_2] = [c_1]; [c_2]\]

Clearly, the elements which have type \( u \mapsto v \) for some \( u, v \) are histories in CCH. We can comment briefly on the interpretation of the operations. The interpretation of the operation \([\mu, u > \mapsto \) expresses that after the action \( \mu \) a fork operation is performed, making explicit the distributed structure of state \( u \). Moreover, the non deterministic operation requires first an action of choice between the two alternatives, and then the execution of the chosen alternative. Finally, in the interpretation of the synchronization operation we have adopted a normal form of computations like the one introduced in [GM 90]. It is straightforward to check that with these interpretations of the operations, \( H \) is indeed a CCS model.

**Example 9 (The Synchronization Law)**

Consider the CCS agent \( E = (\alpha.nil | \beta.nil) | ((\alpha.nil | \delta.nil) + \gamma) \). The synchronization of the transitions labelled \( \alpha \) and \( \alpha \) is represented in \( H \) by

\( [\alpha; (idle_1 \otimes t_\alpha); \alpha] | (\land_2; (t_\pi; idle_1)) \) = \( (\geq \otimes \land_2); (idle_1 \otimes (V_2; t_\pi; \land_2) \otimes idle_1); (\geq \otimes idle_2) \)

where \( \geq \) has two origins \( p_1 \) and \( p_2 \), two destinations \( q_1 \) and \( q_2 \), and no events, with \( p_1 \leq q_2 \), \( p_2 \leq q_1 \), \( \beta(p_i) = \gamma(q_i), i = 1, 2 \). See Figure 3 for a pictorial representation (the labelled indexed ordering functions are the obvious ones, so we do not represent the numbering on processes). □

## 5 Bisimulation Semantics and Minimal Realization

In this section we introduce a truly concurrent observational semantics for the CCS language by means of the notion of bisimulation. We then show that the bisimulation semantics is fully characterized by a minimal realization.

**Definition 10 (Bisimulation)**

The maximal bisimulation on guarded CCS agents (represented by the set of elements of \( I \) of type state) is the greatest equivalence relation \( R \) such that \( u R u \) if and only if \( \ell_H(u) = \ell_H(u') \) and (ii) for every \( t: u \Rightarrow v, t \neq idle(u) \), there is \( t': u' \Rightarrow v', t' \neq idle(u') \), such that \( v R v' \), and \( \ell_H(t') = \ell_H(t') \). □

The first condition in the definition of bisimulation ensures that equivalent agents have the same distributed structure, i.e. they are labelled with the same state of \( H \). Using standard
techniques we can prove the maximal bisimulation exists and it is the union of all bisimulations. We indicate with $\sim_H$ the maximal bisimulation equivalence. When restricted to sequential agents (agents without the $|$) the equivalence $\sim_H$ coincides with the strong observational congruence. In particular, Milner axiomatization [Mil 89] is consistent and complete for finite sequential agents.

**Example 11** (Result of Observing Distribution and Causality)

We have that $\alpha.nil + \alpha.nil = \alpha.nil$, but the two agents $(\alpha.nil | \beta.nil) + (\alpha.nil | \beta.nil)$ and $(\alpha.nil | \beta.nil)$ are not identified because of the global control mechanism for the non deterministic choice. The states $\alpha.nil | \alpha.nil$ and $\alpha.nil | \beta.nil$ are not identified. However, it is easy to convince oneself that this must be the case. In fact, assume the two states above are the intermediate states of computations starting from $\beta.nil | \gamma.nil$ and $\beta.nil | \gamma.nil$. The identification of the intermediate states (after the parallel execution of the actions $\alpha$ and $\beta$) would imply the impossibility of detecting the correct cause of the action $\alpha$. □

**Example 12** (Expressive Power)

The two CCS agents $E_1 = \alpha.(\beta.nil + \gamma.nil) + \alpha.nil | \beta.nil$, $E_2 = \alpha.(\beta.nil + \gamma.nil) + \alpha.nil | \beta.nil + \alpha.\beta.nil$ are indistinguishable by Pomset Bisimulation Equivalence [BC 88]. However, here they are distinguished. This is because the agent $E_2$ can perform a computation $t_\alpha$ ending in a state from which a $t_\gamma$ computation is impossible. The agent $E_1$ cannot do this because the computation $t_\alpha$ is impossible from the state $\alpha.nil | \beta.nil$, although the computation $t_\alpha \otimes idle_1$ is possible. □

We now consider the subcategory of $\text{CatCCS:H}$ of the admissible behaviours, $\text{Beh}_H\text{CCS}$, whose objects are CCS models typed on $H$, and whose morphisms express a notion of simplification. This class of models has an initial model $(I, \ell_1)$ which is the free observational...
semantics, while the most reduced model (if it exists) corresponds to the most abstract observational semantics.

As simplification morphisms we consider *strict transition preserving morphisms* (simply stp-morphisms). Stp-morphisms are program transformations such that two elements of type state are mapped together only if the computations starting from them are mapped together.

**Definition 13 (Strict Transition Preserving Homomorphism)**

A CatCCS:H morphism

\[ h : (C_1, \ell_1 : C_1 \rightarrow H) \rightarrow (C_2, \ell_2 : C_2 \rightarrow H) \]

is a stp-morphism iff (i) \( h \) is surjective; (ii) if \( t_2 \) is an element of \( C_2 \) with type \( h(u_1) \Rightarrow v_2 \) then there is an element \( t_1 \) of \( C_1 \) with type \( u_1 \Rightarrow v_1 \) such that \( h(t_1) = t_2 \) and \( h(v_1) = v_2 \); and (iii) if \( t \) is an element of \( C_1 \) with type \( u \Rightarrow u \) then \( h(t) = \text{idle}(h(u)) \) implies \( t = \text{idle}(u) \).

There is a closed relationship between strict transition preserving homomorphisms and bisimulation relations which are also congruences for the algebraic structure of CCS models. Indeed, the congruence induced by a stp-homomorphism (the kernel) is a bisimulation. Moreover, the quotient mapping with respect to a bisimulation congruence is a stp-morphism.

**Theorem 14 (The Minimal Realization Theorem)**

The category Beh\(_H\) CCS has a final object, a minimal realization of CCS (with observations in \( H \)).

**Theorem 15 (Characterization Theorem)**

The congruence induced on the set of guarded CCS agents by the unique stp-morphism from \((I, \ell_I)\) to the final object of Beh\(_H\) CCS coincides with \( \sim_H \).

### 6 Logical Characterization

So far, we have defined an algebraic observational semantics for CCS. In this section, we introduce a (logical) language to express properties of computations, such that the discriminating power of the language is exactly that of the equivalence \( \sim_H \), thus reflecting a sort of duality between the two representations.

The language of properties takes the form of a Modal Logic HML(\( H \)) in the style of Hennessy-Milner Logic \[HM 85\] whose modalities are \( <h> \) where \( h \) is an element of \( H \) of type \( u \Rightarrow v \) for some \( u: \text{state} \), \( v: \text{state} \).

**Definition 16 (HML(\( H \)) Logic)**
The syntax of the modal logic $HML(H)$ is

$$\psi ::= \text{TRUE} \lor \neg \psi \lor \bigwedge_{j \in J} \psi_j \land h > \psi$$

where $J$ is a (possibly infinite) nonempty set of indices, and $h$ ranges over the set of elements of $H$ which have type $u \Rightarrow v$ for some $u, v : \text{state}$.

We define the satisfaction relation $\models$ for $HML(H)$ on the set of guarded elements of CCS models as follows:

- $u \models \text{TRUE}$ for $u : \text{state}$
- $u \models [n]$ if and only if $\ell(u) = [n]$
- $u \models \neg \psi$ if and only if $u \not\models \psi$
- $u \models \bigwedge_{j \in J} \psi_j$ if and only if $u \models \psi_j$ for each $j \in J$
- $u \models < h > \psi$ if and only if there is $t$ with type $u \Rightarrow v$, $t \neq \text{idle}(u)$, such that $\ell(t) = h$ and $v \models \psi$.

We comment briefly on the definition of the logic. The family of atomic formulae $[n]$ is introduced because we need to describe the distributed structure of states, namely a state $u$ satisfies the atomic formula $[n]$ if and only if it has $n$ autonomous components. (For instance, a state with two autonomous components satisfies the formula $[2]$. In the interleaving case this kind of atomic formula does not provide any discriminating power since a global state is assumed, i.e. the formula $[1]$ is always satisfied.

The satisfaction relation naturally induces an equivalence relation $\equiv_H$ on the elements of $I$ of type $\text{state}$. We say that $v_1 \equiv_H v_2$ if and only if $(v_1 \models \psi) \iff (v_2 \models \psi)$ for all formulae $\psi$ of $HML(H)$.

**Theorem 17 (Logical Characterization)**

The equivalence $\equiv_H$ coincides with $\sim_H$. □

The proof of the theorem follows the same pattern of the proofs given by Hennessy and Milner in [HM 85].

## 7 Forgetting about Internal Moves

The calculus of computations we have given treats all actions as observable. A further abstraction making some actions invisible to the external observer is possible. The standard example of abstraction from invisible actions is provided by Milner's $\tau$ action and the Weak Observational Equivalence [Mil 80, Mil 89]. The idea is that $\tau$-actions represent internal activities which do not effect the observable behaviour of processes. States (processes) are equivalent provided that they can perform the same visible computations, and then reach equivalent states. As a consequence of the abstraction from invisible moves it might happen that a transition and a computation performing several internal moves exhibit the same observable behaviour.
In our framework, the abstraction on silent moves can be obtained by considering a model where the observations of computations are either idle or have some observable action. In the case of a truly concurrent observational semantics the model of the observation has the form of the algebra of Concatenable Concurrent Histories with no action labelled with \( \tau \). We denote by \( H_W \) this model of observations. Basically, the CCS model \( H_W \) can be obtained from the CCS model \( H \) by imposing the following equations:\(^5\)

(i) \( t_\tau = \text{idle}_1 \),

(ii) \( u \xrightarrow{\tau} v = u \Rightarrow v \)

The above equations clearly express that \( \tau \) moves are invisible in the model of the observations \( H_W \).

As a further step in our construction, we build a category whose objects are CCS models labelled (typed) over this weaker calculus of computations, and whose morphisms respect the observations.

A (weak) bisimulation semantics is immediately obtained\(^6\). This observational semantics abstracts from \( \tau \) actions but it is still possible to distinguish between an idle transition and a computation performing an invisible move. We indicate with \( \cong_{H_W} \) the maximal bisimulation equivalence.

The category of admissible behaviours, \( \text{Beh}_{H_W} \text{CCS} \), is then defined by introducing stp-morphisms. Notice that because stp-morphisms are strict on identities, the simplification mapping is able to distinguish between an idle move and a computation performing an invisible move. Thus, the final element of the category \( \text{Beh}_{H_W} \text{CCS} \) fully characterizes the weak observational semantics provided by \( \cong_{H_W} \).

**Theorem 18** \( \text{Beh}_{H_W} \text{CCS} \) has a final object, a minimal realization of CCS (with observation in \( H_W \)) \( \Box \)

**Theorem 19** The congruence induced on the guarded CCS agents by the unique stp-morphism from initial to the final object coincides with \( \cong_{H_W} \) \( \Box \)

As in the case of the CCS model \( H \), we introduce the modal logic \( \text{HML}(H_W) \) where modalities are parameterized with elements of type \( u \Rightarrow v \) of the CCS model \( H_W \). We indicate with \( \equiv_{H_W} \) the equivalence induced by the logic. The logical characterization theorem still holds.

**Theorem 20** The equivalence relation \( \equiv_{H_W} \) generated by \( \text{HML}(H_W) \) coincides with \( \cong_{H_W} \). \( \Box \)

\(^5\)Recall that with \( t_\mu \) we indicate a Concatenable Concurrent History with three linearly ordered elements, where the middle one is labelled with \( \mu \).

\(^6\)Replace \( \ell_H \) of definition 10 with \( \ell_{H_W} \).
When restricted to sequential agents, this weak congruence coincides with the greatest dynamic bisimulation [MS 90]. In particular, for finite sequential agents the axiomatization consisting of the axioms for the strong observational congruence and of the second and third of Milner's $\tau$ laws is consistent and complete.

**Example 21** As an example of the congruence induced by the final object of $\text{Beh}_{Hw} \text{CCS}$, we have that the agents $E + \tau.E$ and $\tau.E$ are identified ($E$ is any agent). Notice that this is Milner's second $\tau$-law. Another law which holds is that $(\tau.E_1) \mid E_2$ and $E_1 \mid (\tau.E_2)$ (with both $E_1$ and $E_2$ initial states observed as [1]) are identified. The intuitive idea is that the observer is able to detect that the internal move has taken place, but it is not able to detect the location of such a move. The distribution of initial states is still observed, so that for instance $\tau.(\alpha.nil \mid \beta.nil)$ and $(\alpha.nil \mid \beta.nil)$ are not identified. Finally, the CCS agents $\tau.\tau.\alpha.nil$ and $\tau.\alpha.nil$ are not identified, because the $\tau$ move of the first agent should be simulated by the second agent staying idle, which is not permitted. □

**8 Conclusions**

We have introduced an theory for CCS (algebraic models and logics), where causal dependencies and distribution are properly taken into account. We have shown that observational models can be equipped with truly concurrent observations which are incremental, and that behavioural congruences can be characterized both by considering special simplification mappings, and by considering the equivalence induced by modal logics (in the style of Hennessy-Milner Logic).

We plan to extend the results of this paper, giving sufficient conditions for a category of observational models of a process description language to yield observational equivalences which can characterized by minimal realization.

**9 References**


---

If one insists that also the first $\tau$ law holds, the resulting congruence (i.e. the weak observational congruence), as observed above, is not a bisimulation and the characterization by finality does not hold.


