Fundamenta Informaticae, vol. 16, n. 2, 1992

DYNAMIC CONGRUENCE vs. PROGRESSING BISIMULATION for CCS*

Ugo Montanari and Vladimiro Sassone

Dipartimento di Informatica – Università di Pisa
Corso Italia 40 - 56125 - Pisa - Italy
E-MAIL:{ugo,vladi}@di.unipi.it

Abstract

Weak Observational Congruence (woc) defined on CCS agents is not a bisimulation since it does not require two states reached by bisimilar computations of woc agents to be still woc, e.g. $\alpha.\tau.\beta.nil$ and $\alpha.\beta.nil$ are woc but $\tau.\beta.nil$ and $\beta.nil$ are not. This fact prevent us from characterizing CCS semantics (when τ is considered invisible) as a final algebra, since the semantic function would induce an equivalence over the agents that is both a congruence and a bisimulation.

In the paper we introduce a new behavioural equivalence for CCS agents, which is the coarsest among those bisimulations which are also congruences. We call it Dynamic Observational Congruence because it expresses a natural notion of equivalence for concurrent systems required to simulate each other in the presence of dynamic, i.e. run time, (re)configurations. We provide an algebraic characterization of Dynamic Congruence in terms of a universal property of finality.

Furthermore we introduce Progressing Bisimulation, which forces processes to simulate each other performing explicit steps. We provide an algebraic characterization of it in terms of finality, two logical characterizations via modal logic in the style of HML and a complete axiomatization for finite agents (consisting of the axioms for Strong Observational Congruence and of two of the three Milner's τ-laws).

Finally, we prove that Dynamic Congruence and Progressing Bisimulation coincide for CCS agents.

1 Introduction

Understanding concurrent systems is difficult, since many of our intuitions about sequential systems cannot be extended to concurrent and distributed systems. In particular, there is no prevalent, notion of system behaviour on which semantic constructions can be based.

Milner's *Calculus of Communicating Systems* (CCS) ([Mil80], [Mil89]) can be considered the touchstone of process description languages. Its semantics is given in two steps.

*Research supported in part by **HEWLETT–PACKARD Laboratories**, Pisa Science Center.

First, a *Labelled Transition Systems* (LTS), which constitutes the abstract machine (the interpreter) of the language, is defined in the style of Plotkin's *Structured Operational Semantics* (SOS) ([Plo81]). Then behavioural equivalences which identify processes are introduced.

A large number of such equivalences have been proposed. In fact, several properties may be interesting in the analysis of concurrent systems and each definition stresses a particular aspect of systems behaviour. For instance, if we are interested only in the actions performed by a system, we consider a simple trace equivalence; otherwise, if we allow the possibility of replacing a subsystem by an equivalent one, we must define an equivalence which is a *congruence* with respect to language constructs. Moreover, if we follow the *interleaving* approach ([Mil80], [Mil89]), i.e. if we express concurrency of actions by saying that they may be executed in any order, then we will choose to observe *sequences* of actions. In a *truly concurrent* approach ([Pet62], [NPW81], [Pra86], [DDM90]) we may want to observe *partial orderings* of actions. For an overview and a comparison of many behavioural equivalences see, for instance, [DeN87] or [GvG89].

Among the equivalences proposed in the literature, we consider those based on *bisimulation* ([Mil80], [Par81], [vGW89]). Two processes are equivalent if they not only produce the same observations, but also reach equivalent states afterwards and, in the case of *Branching Bisimulation*, pass only through equivalent intermediate states. The advantages of those equivalences, besides their operational suggestiveness, are the existence of simple axiomatizations, the elegance of the proofs and their relationship with equivalences based on logics ([Mil89]), on denotational semantics ([Ab88]) and on algebraic techniques ([BB88], [Acz87]).

Ferrari and Montanari ([FM90]) define *Strong Observational Congruence*, the simplest equivalence based on bisimulation, in an algebraic way. They see the CCS transition system as equipped with an algebraic structure on both states (the operations of the language) and transitions (an operation for every SOS inference rule). Furthermore they define a collection (in fact a subcategory) of such transition systems, where the operations are not necessarily free and where morphisms relating two transition systems are *transition preserving*, i.e. they define simplification mappings which respect, besides operations on both nodes and arcs and besides labels (including τ's) on arcs, the transitions outgoing from any state. This subcategory has an initial and a terminal element, and it turns out that the unique morphism from the former to the latter defines the coarsest equivalence on agents that is both a *congruence* and a *strong bisimulation*, i.e. *Strong Observational Congruence*.

A similar construction can be repeated by mapping computations instead of transitions, and disregarding τ's. We obtain the coarsest equivalence that is both a *congruence* and a *weak bisimulation*, but now this equivalence is *not* the *Weak Observational Congruence*, since the latter is not a weak bisimulation. Actually, Van Glabbeek ([vGl87]) shows that Weak Observational Congruence is a bisimulation, but the operational semantics of CCS he assumes is not the usual one, e.g. $\alpha.\beta \xrightarrow{\alpha} \tau.\beta$.

From these facts originated a new behavioural equivalence, *Dynamic Observational Congruence*, introduced in [MS90] and further developed in [MS91]. The presentation here is the full version of [MS91].

The basic idea of *dynamic bisimulation* is to allow at every step of bisimulation not only the execution of an action (or a sequence of actions), but also the embedding of the two agents under measurement within the same, but otherwise arbitrary, context.

Conceptually, bisimulation can be understood as a kind of *game*, where two programs try in turn to match each other's moves. When a move consists of performing some computational steps and matching a move means to produce the same *observable* behaviour, we obtain the notion of *observational equivalence.* This definition is independent of the particular observable behaviour (τ observable or not, sequences or partial orderings of actions, etc.), and it can be proved under very mild conditions that the maximal bisimulation relation *always* exists and is an equivalence ([MSg89]).

Instead of programs just being able to perform computational steps, we might consider *modular* (i.e. compositional) *software systems* which are statically configurated at time zero. In our functional setting, this means to start the game by applying an arbitrary context to both agents. The resulting semantic notion is Milner's *Observational Congruence.*

Finally we may allow *dynamic reconfiguration*: at any instant in time the structure of both software systems may be modified, but in the same way, i.e. a context can be applied to both agents. In this way we obtain our new notion of *dynamic bisimulation*, and the resulting semantic equivalence is called *Dynamic Observational Congruence.* Of course the *bisimulation game* is not just an academic fancy but is motivated by practical considerations: equivalent (in the various senses discussed above) modules can actually replace each other consistently in any real system, guaranteeing *software modularity* and *reusability.* In particular, the issue of dynamic reconfiguration is relevant for *system programming* and for applications like *software development*, where executing new programs is essential, and like *industrial automation*, where halting execution may be extremely costly.

In this paper we show that *Dynamic Observational Congruence*, is the *coarsest* bisimulation which is also a congruence, and thus it is algebraically characterized by the finality construction in the style of [FM90] outlined above.

Furthermore we introduce a new observational equivalence, *Progressing Bisimulation*, between states of a labelled transition system with a distinct action τ. The basic idea underlying Progressing Bisimulation is to force programs in the bisimulation game to match moves with explicit moves. This also justifies its name.

For Progressing Bisimulation we give an *algebraic characterization* in the category of labelled transition systems and two *modal logics* in the style of HML, one in which the modal operators may include τ's, and their meaning is that at least those τ's must appear in the models, the other in which the satisfaction relation forces agents to move. Then we provide a *complete axiomatization* for states with finite computations, consisting of the axioms for Strong Observational Congruence and of two of the three Milner τ-laws (of course $\alpha.\tau.P = \alpha.P$ is no longer true).

Finally, we show that on the CCS transition system Progressing Bisimulation coincides with Dynamic Congruence so giving it all the characterizations above.

This presentation stresses the fact that we are in presence of two distinct, general concepts, which, in the case of CCS, coincide: *Dynamic Congruence*, which makes sense on the LTS of every language and has a nice algebraic characterization and *Progressing Bisimulation*, which makes sense on every LTS with a distinct action τ and has algebraic, logical and axiomatic characterizations.

The paper is organized as follows.

In section 2 we recall the basic concepts of CCS ([Mil80], [Mil89]).

In section 3 we give the *operational definition* and an *algebraic characterization* of Dynamic Observational Congruence proving that in the category introduced in [FM90] there exists an object corresponding to it. It is easy to notice that the construction could be given on any labelled transition system in which the concept of context makes sense.

In section 4 we introduce *Progressing Bisimulation*, and give its *algebraic*, *logical* and *axiomatic* characterizations.

In section 5 we show that Dynamic Congruence and Progressing Bisimulation *coincide* in the CCS transition system, thus obtaining a full characterization of Dynamic Congruence.

In appendix A we recall some further results about CCS which are used in the paper.

The algebraic characterizations in sections 3 and 4 are given in terms of *type algebras*, introduced in [MSS90], which are an extension of the standard theory of *universal algebras* able to deal with specification problems such as partiality. In appendix B we introduce the fundamental concepts of type algebras.

In the paper, we strictly follow the notations and definitions in the references. Thus the expert reader may safely skip section 2 and the appendices.

2 Calculus of Communicating Systems

In this section we recall the basic definitions of Milner's *Calculus of Communicating Systems* (CCS). For a full introduction however, the reader is referred to [Mil80] and [Mil89].

Let $\Delta = \{\alpha, \beta, \gamma, \ldots\}$ be a fixed set of *actions*, and $\overline{\Delta} = \{\overline{\alpha} | \alpha \in \Delta\}$ the set of *complementary actions* ($\bar{}$ being the operation of complementation). $\Lambda = \Delta \cup \overline{\Delta}$ (ranged over by λ) is the set of *visible actions.* Moreover, let $\tau \notin \Lambda$ be the *invisible action* and μ range over $\Lambda \cup \{\tau\}$.

Definition 2.1 *(CCS Expressions and Agents)*
The syntax of CCS expressions *is defined as follows:*

$$E ::= \ x \mid nil \mid \mu.E \mid E \setminus \alpha \mid E[\Phi] \mid E + E \mid E|E \mid recx.E$$

where x is a variable, and Φ is a permutation of $\Lambda \cup \{\tau\}$ preserving τ and $\bar{}$.
CCS agents *(ranged over by $P, Q, \ldots$) are closed CCS expressions, i.e. expressions without free variables. A CCS agent P is* guarded *if every variable occurrence in P is within some subexpression $\lambda.Q$ of P.* □

Labelled transition systems ([Kel76]), a generalization of finite–state automata, are a standard model of computation often used to describe behaviours and give operational semantics to programming languages.

Definition 2.2 *(Labelled Transition Systems)*
A labelled transition system *is a triple* $T =< S, \longrightarrow, A >$ *where:*

- S *is a non–empty set of states;*
- $\rightarrow \subseteq S \times A \times S$ *is the transition relation;*
- A *is a non–empty set of labels.* □

As usual, we write $s \xrightarrow{a} s'$ to indicate that $(s, a, s') \in \longrightarrow$.

The operational semantics of CCS is defined in terms of *labelled transition systems* in which the states are CCS expressions, the labels are actions and the transition relation, following Plotkin's SOS style ([Plo81]), is defined by axioms and inference rules driven by the syntactic structure of expressions.

Definition 2.3 *(CCS Transition Relation)*
The CCS transition relation $\xrightarrow{\mu}$ *is defined by the following inference rules.*

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

Res) $$\frac{E_1 \xrightarrow{\mu} E_2}{E_1 \setminus \alpha \xrightarrow{\mu} E_2 \setminus \alpha} \qquad \mu \notin \{\alpha, \overline{\alpha}\}$$

Rel) $$\frac{E_1 \xrightarrow{\mu} E_2}{E_1[\Phi] \xrightarrow{\Phi(\mu)} E_2[\Phi]}$$

Sum) $$\frac{E_1 \xrightarrow{\mu} E_2}{E_1 + E \xrightarrow{\mu} E_2 \text{ and } E + E_1 \xrightarrow{\mu} E_2}$$

Com1) $$\frac{E_1 \xrightarrow{\mu} E_2}{E_1|E \xrightarrow{\mu} E_2|E \text{ and } E|E_1 \xrightarrow{\mu} E|E_2}$$

Com2) $$\frac{E_1 \xrightarrow{\lambda} F_1 \text{ and } E_2 \xrightarrow{\overline{\lambda}} F_2}{E_1|E_2 \xrightarrow{\tau} F_1|F_2}$$

Rec) $$\frac{E_1[recx.E_1/x] \xrightarrow{\mu} E_2}{recx.E_1 \xrightarrow{\mu} E_2}$$ □

The transition $P \xrightarrow{\mu} Q$ expresses that the agent P may evolve to become the agent Q performing action μ.

Computations are usually described by *multistep derivation* relations derived from relation $\xrightarrow{\mu}$. Here, we introduce the relations we will need in the following.

Definition 2.4 *(Multistep Derivations)*

- $\overset{\epsilon}{\Longrightarrow} = (\xrightarrow{\tau})^*$, *$\epsilon$ being the null string and * the transitive closure of relations;*
- $\overset{\mu}{\Longrightarrow} = \overset{\epsilon}{\Longrightarrow} \xrightarrow{\mu} \overset{\epsilon}{\Longrightarrow}$, $\mu \in \Lambda \cup \{\tau\}$;
- $\overset{t}{\Longrightarrow} = \overset{\mu_1}{\Longrightarrow} \overset{\mu_2}{\Longrightarrow} \cdots \overset{\mu_n}{\Longrightarrow}$, $t = \mu_1\mu_2 \dots \mu_n \in (\Lambda \cup \{\tau\})^*$;
- $\overset{s}{\Longrightarrow} = \overset{\lambda_1}{\Longrightarrow} \overset{\lambda_2}{\Longrightarrow} \cdots \overset{\lambda_n}{\Longrightarrow}$, $s = \lambda_1\lambda_2 \dots \lambda_n \in \Lambda^*$. □

The semantics given by labelled transition systems is too concrete: the addition of *behavioural equivalence* equates those processes which cannot be distinguished by any external observer of their behaviour. Park's notion of *bisimulation* ([Par81]) has become the standard device for defining behavioural equivalences.

Definition 2.5 *(Strong Bisimulation)*
Let $\mathcal{R}$ be a binary relation over CCS agents.
Then Ψ, a function from relations to relations, is defined by

$(P, Q) \in \Psi(\mathcal{R})$ *if and only if* $\forall \mu \in \Lambda \cup \{\tau\}$*:*

- *whenever $P \xrightarrow{\mu} P'$ there exists Q' s.t. $Q \xrightarrow{\mu} Q'$ and $(P', Q') \in \mathcal{R}$;*
- *whenever $Q \xrightarrow{\mu} Q'$ there exists P' s.t. $P \xrightarrow{\mu} P'$ and $(P', Q') \in \mathcal{R}$.*

A relation $\mathcal{R}$ is called strong bisimulation *if and only if $\mathcal{R} \subseteq \Psi(\mathcal{R})$.*
The relation $\sim = \cup\{\mathcal{R} | \mathcal{R} \subseteq \Psi(\mathcal{R})\}$ is called Strong Observational Equivalence. □

In words, two agents are (strongly) bisimilar if they are not only capable of performing the same sequence of actions but the states they reach after the execution of a given action are (strongly) bisimilar.

Strong Observational Equivalence is the maximal set of pairs of strongly bisimilar processes and is an equivalence relation over CCS agents.

Proposition 2.6 *($\sim$ is well–defined)*
Relation $\sim$ is a strong bisimulation and an equivalence relation.

Proof. By fixed-point theory (see [Tar55]), since Ψ is a monotone mapping on the complete lattice of binary relations over CCS agents, $\sim$ is the greatest fixed-point of Ψ and so $\sim = \Psi(\sim)$, i.e. $\sim$ is the largest strong bisimulation. Moreover, it is immediate to see that $\sim$ enjoys the properties of equivalences. □

Strong Observational Equivalence can be extended to CCS expressions by saying that two expressions are strongly equivalent if so are all the agents obtained by binding their free variables to CCS agents.

However, definition 2.5 does not consider τ actions as special actions representing the occurrence of invisible internal moves. If we take into account the special status of τ actions, agents are equivalent if they can perform the same sequences of visible actions and then reach equivalent states. The notion of *Weak Observational Equivalence* implements this kind of abstraction.

Definition 2.7 *(Weak Bisimulation)*
Let $\mathcal{R}$ be a binary relation over CCS agents.
Then Φ, a function from relations to relations, is defined by

$(P, Q) \in \Phi(\mathcal{R})$ *if and only if* $\forall s \in \Lambda^*$*:*

- *whenever $P \overset{s}{\Longrightarrow} P'$ there exists Q' s.t. $Q \overset{s}{\Longrightarrow} Q'$ and $(P', Q') \in \mathcal{R}$;*
- *whenever $Q \overset{s}{\Longrightarrow} Q'$ there exists P' s.t. $P \overset{s}{\Longrightarrow} P'$ and $(P', Q') \in \mathcal{R}$.*

A relation $\mathcal{R}$ is called weak bisimulation *if and only if $\mathcal{R} \subseteq \Phi(\mathcal{R})$.*
The relation $\approx = \cup\{\mathcal{R} | \mathcal{R} \subseteq \Phi(\mathcal{R})\}$, is called Weak Observational Equivalence. □

A proposition equivalent to proposition 2.6 holds for $\approx$.

Proposition 2.8 *($\approx$ is well–defined)*
Relation $\approx$ is a weak bisimulation and is an equivalence relation. □

Weak Equivalence is extended to CCS expressions in the same way Strong Equivalence was.

An equivalence ρ is called *congruence* with respect to an operator f, if it is respected by the operator, i.e. $x\rho y$ implies $f(x)\rho f(y)$. The equivalences which are congruences with respect to all the operators of the language are very important: they provide algebras in which equivalence is mantained in every context (i.e. it is actually an equality), a property that can be exploited by algebraic techniques.

Definition 2.9 *(Context)*
A context $\mathcal{C}[\ ]$ *is a CCS expression without free variables and with exactly one "hole" to be filled by a CCS agent.* □

Relation $\sim$ is a congruence with respect all the CCS operators, that is $E \sim F$ implies $\mathcal{C}[E] \sim \mathcal{C}[F]$ for each context $\mathcal{C}[\ ]$, but it is well known that Weak Observational Equivalence is not a congruence. Indeed, we have $\tau.E \approx E$ but in general it is false that $\tau.E+E' \approx E+E'$, e.g. $\tau.\alpha.nil \approx \alpha.nil$ but $\beta.nil+\alpha.nil \not\approx \beta.nil+\tau.\alpha.nil$ because the first agent provides α and β as alternative actions, while the second agent may autonomously discard the β alternative by simply performing a τ action.

The largest congruence contained in $\approx$ is Milner's Observational Congruence.

Definition 2.10 *(Weak Observational Congruence)*
We say that $P \approx^c Q$ if and only if for any context $\mathcal{C}[\ ]$, $\mathcal{C}[P] \approx \mathcal{C}[Q]$.
Relation $\approx^c$ is called Weak Observational Congruence. □

Weak Observational Equivalence has a main drawback: it is *not* a bisimulation. As an example consider the weakly congruent agents $\alpha.\tau.nil$ and $\alpha.nil$. When $\alpha.\tau.nil$ performs an α action becoming the agent $\tau.nil$, $\alpha.nil$ can only perform an α action becoming nil: clearly $\tau.nil$ and nil are not weakly congruent but only weakly equivalent. Our definition of Dynamic Observational Congruence remedies this situation.

3 Dynamic Observational Congruence

In this section we introduce *Dynamic Bisimulation* by giving its *operational definition* and its *algebraic characterization* in the style of [FM90].

The definition is given for CCS, but it can be given for any labelled transition system in which the concept of context makes sense, in particular for the LTS corresponding to any language.

In the rest of the paper, when we use generically the term bisimulation, we actually mean weak bisimulation.

3.1 Operational definition

We want to find the coarsest bisimulation which is also a congruence. Let $\mathcal{B}$ be the set of (weak) bisimulations and $\mathcal{C}$ be the set of congruences.

Definition 3.1 *(Dynamic Bisimulation)*
Let $\mathcal{R}$ be a binary relation over CCS agents.
Then Φ_d, a function from relations to relations, is defined as follows:
$(P, Q) \in \Phi_d(\mathcal{R})$ *if and only if* $\forall s \in \Lambda^*$ *and* $\forall \mathcal{C}[\ ]$:
- *whenever* $\mathcal{C}[P] \overset{s}{\Longrightarrow} P'$ *there exists* Q' *s.t.* $\mathcal{C}[Q] \overset{s}{\Longrightarrow} Q'$ *and* $(P', Q') \in \mathcal{R}$;
- *whenever* $\mathcal{C}[Q] \overset{s}{\Longrightarrow} Q'$ *there exists* P' *s.t.* $\mathcal{C}[P] \overset{s}{\Longrightarrow} P'$ *and* $(P', Q') \in \mathcal{R}$.

A relation $\mathcal{R}$ is called dynamic bisimulation *if and only if* $\mathcal{R} \subseteq \Phi_d(\mathcal{R})$.
The relation $\approx^d = \cup\{\mathcal{R} | \mathcal{R} \subseteq \Phi_d(\mathcal{R})\}$ *is called* Dynamic Observational Equivalence. □

Proposition 3.2 *($\approx^d$ is well–defined)*
Relation $\approx^d$ is a dynamic bisimulation and is an equivalence relation. □

Just as Strong and Weak Equivalence Dynamic Equivalence is extended to CCS expressions. In the following, whenever it makes sense, we will consider only CCS agents: obviously, our results will also hold for CCS expressions, by definition of the equivalences on them.

First we show that Dynamic Bisimulations are indeed bisimulations.

Lemma 3.3 *(Dynamic Bisimulations are Weak Bisimulations)*
$\mathcal{R} \subseteq \Phi_d(\mathcal{R})$ *implies* $\mathcal{R} \subseteq \Phi(\mathcal{R})$.

Proof. Directly from the definitions of Φ and Φ_d (fixing the context $\mathcal{C}[\ ] \equiv x$). □

As a corollary, we have that $\approx^d$ refines $\approx$.

Corollary 3.4 *(Dynamic Equivalence refines Weak Equivalence)*
$\approx^d \subseteq \approx$, *where $\approx$ is Weak Observational Equivalence.* □

However, the reverse inclusion does not hold as $P \approx \tau.P$ while $P \not\approx^d \tau.P$.

We show now that $\approx^d$, Dynamic Observational Equivalence, is a congruence, and, actually, the coarsest bisimulation which is also a congruence.

Lemma 3.5 *($\Phi_d(\mathcal{R})$'s are Congruences)*
$P \, \Phi_d(\mathcal{R}) \, Q$ *if and only if* $\mathcal{C}[P] \, \Phi_d(\mathcal{R}) \, \mathcal{C}[Q]$ *for each context* $\mathcal{C}[\ ]$.

Proof. ($\Rightarrow$) Since $P \, \Phi_d(\mathcal{R}) \, Q$, fixed $\mathcal{C}[\ ]$, for any context $\mathcal{C}'[\ ]$, $\mathcal{C}'[\mathcal{C}[P]]$ and $\mathcal{C}'[\mathcal{C}[Q]]$ satisfy the conditions of definition 3.1. Therefore, $(\mathcal{C}[P], \mathcal{C}[Q]) \in \Phi_d(\mathcal{R})$.
($\Leftarrow$) Obvious. □

From this lemma, since $\Phi_d(\approx^d) = \approx^d$, it is immediate to see that $\approx^d$ is a congruence.

Corollary 3.6 *(Dynamic Equivalence is a Congruence)*
$P \approx^d Q$ if and only if $\mathcal{C}[P] \approx^d \mathcal{C}[Q]$ for each context $\mathcal{C}[\,]$. □

Proposition 3.7 *(Bisimulation and Congruence $\Rightarrow$ Dynamic Bisimulation)*
$\mathcal{R} \in \mathcal{B} \cap \mathcal{C}$ implies $\mathcal{R} \subseteq \Phi_d(\mathcal{R})$.

Proof. $\mathcal{R} \in \mathcal{B} \cap \mathcal{C}$ implies $\mathcal{R} \subseteq \Phi(\mathcal{R})$ and $P\mathcal{R}Q$ implies $\mathcal{C}[P]\mathcal{R}\mathcal{C}[Q]$ $\forall \mathcal{C}[\,]$.
Then if $(P,Q) \in \mathcal{R}$, $\forall \mathcal{C}[\,]$ $(\mathcal{C}[P], \mathcal{C}[Q]) \in \mathcal{R}$ and so $(\mathcal{C}[P], \mathcal{C}[Q]) \in \Phi(\mathcal{R})$ and so
$\forall s \in \Lambda^*$ if $\mathcal{C}[P] \overset{s}{\Longrightarrow} P'$ then $\exists Q'$ s.t. $\mathcal{C}[Q] \overset{s}{\Longrightarrow} Q'$ and $(P',Q') \in \mathcal{R}$ and
if $\mathcal{C}[Q] \overset{s}{\Longrightarrow} Q'$ then $\exists P'$ s.t. $\mathcal{C}[P] \overset{s}{\Longrightarrow} P'$ and $(P',Q') \in \mathcal{R}$.
Therefore $(P,Q) \in \Phi_d(\mathcal{R})$ and so $\mathcal{R} \subseteq \Phi_d(\mathcal{R})$. □

Hence we have:

Corollary 3.8 *($\approx^d$ is the coarsest)*
$\approx^d$ is the greatest bisimulation which is also a congruence.

Proof. From the definition of $\approx^d$ and from the previous proposition we have that $\approx^d \supseteq \bigcup\{\mathcal{R} | \mathcal{R} \in \mathcal{B} \cap \mathcal{C}\}$. On the other hand, by proposition 3.2 and lemma 3.3 we have that $\approx^d \in \mathcal{B}$ and by corollary 3.6 we have that $\approx^d \in \mathcal{C}$.
So, $\approx^d \in \mathcal{B} \cap \mathcal{C}$ and therefore $\approx^d = \bigcup\{\mathcal{R} | \mathcal{R} \in \mathcal{B} \cap \mathcal{C}\}$. □

3.2 Algebraic characterization

In this subsection we show that $\approx^d$ has a corresponding object in **CatLCCS**, the category of CCS computations whose construction is due to Ferrari and Montanari ([FM90], [F90]). Here, the construction of **CatLCCS** is based on [FMM91].

As we have seen in section 2, the operational semantics of CCS is defined by deductive systems. Now, we structure those systems as *type algebras* ([MSS90], see also appendix B), i.e. algebras in which types are assigned to elements, and which contain, as special elements, the types themselves.

Types allow us to distinguish between three kinds of elements: agents, typed by *state* and denoted by $u, v, w \ldots$, transitions, typed by $\rightarrow$ and denoted by t and computations, typed by $\Rightarrow$ and denoted by c. In the following $x : y$ will indicate that x has type y.

Operations *source*() and *target*(), which respectively give source and target state (elements of type *state*) and a function *label*() which gives an observation in $\Lambda^* \cup \{\tau\}$, are defined on elements typed by $\rightarrow$ or $\Rightarrow$.

We write $t : u \xrightarrow{\mu} v$ to denote an element of type $\rightarrow$ with *source*$(t) = u$, *target*$(t) = v$ and *label*$(t) = \mu$. Similarly, we write $c : u \overset{s}{\Longrightarrow} v$.

A computation with empty observation will be indicated by $c : u \overset{\epsilon}{\Longrightarrow} v$, while we will write $u \Rightarrow v$ when we are not interested in the observation.

The definition of CCS models should be given by listing an appropriate *presentation* and by stating that CCS models are the models of that presentation. Such a definition would be long and boring, since it deals with the formal definition of guarded CCS agents.

Therefore, we prefer to give only the most significant axioms, i.e. those defining the structure of transitions and computations. The interested reader can find the rigorous definition in [FMM91].

Definition 3.9 *(CCS Models)*
A CCS model *is a type algebra (with multityping) (see definition B.1) where elements typed* state *have the algebraic structure of guarded CCS agents.*
Moreover, there is an operation on transitions for each rule in the CCS transition system, an operation idle and an operation $\_;\_$*. They satisfy the following:*

$$[\mu, v >: \mu.v \xrightarrow{\mu} v$$

$$\frac{t : u \xrightarrow{\mu} v}{t[\Phi] : u[\Phi] \xrightarrow{\Phi(\mu)} v[\Phi]} \qquad\qquad \frac{t : u \xrightarrow{\mu} v}{t \setminus \alpha : u \setminus \alpha \xrightarrow{\mu} v \setminus \alpha}\ \mu \notin \{\alpha, \overline{\alpha}\}$$

$$\frac{t : u \xrightarrow{\mu} v}{t <\!+\, w : u + w \xrightarrow{\mu} v} \qquad\qquad \frac{t : u \xrightarrow{\mu} v}{w +\!> t : w + u \xrightarrow{\mu} v}$$

$$\frac{t : u \xrightarrow{\mu} v}{t \rfloor w : u|w \xrightarrow{\mu} v|w} \qquad\qquad \frac{t : u \xrightarrow{\mu} v}{w \lfloor t : w|u \xrightarrow{\mu} w|v}$$

$$\frac{t_1 : u_1 \xrightarrow{\lambda} v_1 \ \mathit{and}\ t_2 : u_2 \xrightarrow{\overline{\lambda}} v_2}{t_1|t_2 : u_1|u_2 \xrightarrow{\tau} v_1|v_2}$$

$$\frac{t : u \xrightarrow{\lambda} v}{t : u \xRightarrow{\lambda} v} \qquad \frac{t : u \xrightarrow{\tau} v}{t : u \xRightarrow{\epsilon} v} \qquad idle(v) : v \xRightarrow{\epsilon} v \qquad \frac{c_1 : u \xRightarrow{s_1} v \ \mathit{and}\ c_2 : v \xRightarrow{s_2} w}{c_1; c_2 : u \xRightarrow{s_1 s_2} w}$$

Finally, a CCS model satisfies the following equations:

$$recx.u = u[recx.u/x] \qquad c_1; (c_2; c_3) = (c_1; c_2); c_3 \qquad \frac{c : u \Rightarrow v}{idle(u); c = c = c; idle(v)} \qquad \square$$

Note that the way in which we defined the operations also defines *source*, *target* and *label*. Note also that there are no rules and operations for recursion which is instead handled by imposing the axiom above on states. Moreover τ's are completely forgotten in the observations.

As an example, the term $[\alpha, u >\rfloor \beta.v <\!+\, \gamma.w$ has type (actually type, *source*, *target* and *label*) $\alpha.u|\beta.v + \gamma.w \xrightarrow{\alpha} u|\beta.v$ and the term $u\lfloor[\beta, v >$ has type $u|\beta.v \xrightarrow{\beta} u|v$. Hence $[\alpha, u >\rfloor \beta.v <\!+\, \gamma.w; u\lfloor[\beta, v >$ has type $\alpha.u|\beta.v + \gamma.w \xRightarrow{\alpha\beta} u|v$.

Definition 3.10 *(CCS Morphisms)*
A CCS morphism *is a type homomorphism (see definition B.2) of CCS models.* □

A CCS morphism obviously respects types (by definition) and respects *source* and *target* since they are operations of the algebra. Moreover, it is easy to see that CCS morphisms respect the function *label*, because the latter is completely determined by the algebraic structure of computations, which, in turn, is preserved.

Definition 3.11 *(The Category* **CatLCCS***)*
CatLCCS *is the category whose objects are CCS models and whose morphisms are CCS morphisms.* □

Here we recall the definitions of initial and terminal object in a category.

Definition 3.12 *(Inital and Terminal Objects)*
Given a category **C**, *an object I of* **C** *is* initial *in* **C** *if and only if for any object C of* **C** *there exists a unique morphism in* **C** *from I to C.*
Dually, an object T of **C** *is* terminal *in* **C** *if and only if for any object C of* **C** *there exists a unique morphism in* **C** *from C to T* □

Initial and terminal objects are uniquely determined up to isomorphisms. This allows us to speak about *the* initial and *the* terminal object in a category.

The following result is a general result on categories of type algebras (see proposition B.6).

Proposition 3.13 *(Initiality in* **CatLCCS***)*
CatLCCS *has an initial object $\Im$.* □

We can think of $\Im$ as a term algebra whose element typed by *state* are exactly the guarded CCS agents.

Weak Observational Congruence cannot be characterized algebraically in **CatLCCS**, even though Ferrari and Montanari ([FM90]) showed that this is the case in a category of unfoldings constructed from it. This impossibility is basically due to the fact that the definition of morphism implies that, from congruent states, corresponding transitions lead to congruent states and this is *not* the case for Weak Observational Congruence.
The situation is different for $\approx^d$.
In the following we shall use $[\![P]\!]_C$ to denote the state to which agent P evaluates in a particular CCS model C. Clearly, $[\![\ ]\!]_C$ is the evaluation morphism from $\Im$ to C. In order to avoid confusion we will denote by $[\![P]\!]$ the agent P viewed as an element of $\Im$.

Lemma 3.14 *(CatLCCS morphisms respect contexts)*
If h is a CatLCCS morphism then $h([\![P]\!]_C) = h([\![Q]\!]_C)$ implies $h([\![\mathcal{C}[P]]\!]_C) = h([\![\mathcal{C}[Q]]\!]_C)$ for each context $\mathcal{C}[\ ]$.

Proof. The thesis follows directly from the fact that h respects the algebraic structure of elements. In fact, for each context $\mathcal{C}[\ ]$ there is a corresponding unary derived operation $[\![\mathcal{C}]\!]_C$ in each CCS model C defined on the constructable elements of type *state* of C as $[\![\mathcal{C}]\!]_C([\![P]\!]_C) = [\![\mathcal{C}[P]]\!]_C$.
Therefore, given any CatLCCS morphism $h : C \to C'$, since derived operations are preserved by these morphisms, we have that $h([\![\mathcal{C}[P]]\!]_C) = h([\![\mathcal{C}]\!]_C([\![P]\!]_C)) = [\![\mathcal{C}]\!]_{C'}(h([\![P]\!]_C)) = [\![\mathcal{C}]\!]_{C'}(h([\![Q]\!]_C)) = h([\![\mathcal{C}]\!]_C([\![Q]\!]_C)) = h([\![\mathcal{C}[Q]]\!]_C)$. □

Next, we want to give an algebraic characterization of $\approx^d$ in **CatLCCS**, i.e. we want to find a CCS model C such that exactly the pairs of dynamically bisimilar agents evaluate to the same element of C, or, in other words, such that the congruence which is induced on $\Im$ by the unique morphism from $\Im$ to C, is exactly $\approx^d$.

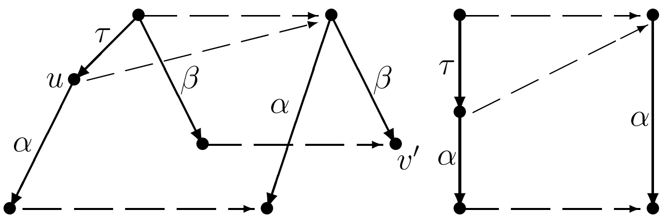
In order to do that, we restrict our attention to a particular kind of abstraction morphisms, which, roughly speaking, preserve transitions in the sense that they do not map an element having a μ-labelled transition to an element without any such transition. Such a condition has an evident relation with bisimulations: indeed, we will show that the congruences these morphisms induce on $\Im$ are dynamic bisimulations.

This will bring us to identify the object we are looking for and a subcategory of **CatLCCS** in which it is the terminal object.

Definition 3.15 *(Transition Preserving Homomorphism)*
A CatLCCS morphism $h : C \to C'$ is a transition preserving homomorphism*, tp-homomorphism for short, if and only if:*

- *$h : C \to C'$ is a surjective CatLCCS morphism;*
- *$t' \in C'$, $t' : h(u) \Rightarrow v'$ implies $\exists t \in C$, $t : u \Rightarrow v$ with $h(t) = t'$.* □

Example 3.16 *(no tp-homomorphism maps $\tau.\alpha + \beta$ to $\alpha + \beta$ or $\tau.\alpha$ to α)*



(A) **(B)**

The figures show two morphisms which are not tp-homomorphisms.
In case (A) we have $t : h(u) \overset{\beta}{\Longrightarrow} v'$ but $u \overset{\beta}{\not\Longrightarrow}$.
In case (B) the morphism cannot respect the algebra, for if it did we would have $h(\tau.\alpha) + h(\beta) = h(\alpha) + h(\beta)$ and so $h(\tau.\alpha + \beta) = h(\alpha + \beta)$ which is case (A). □

Proposition 3.17 *(tp-homomorphism $\Rightarrow$ Dynamic Bisimulation)*
If $h : \Im \to C$ is a tp-homomorphism then $h([\![P]\!]) = h([\![Q]\!])$ implies $P \approx^d Q$.

Proof. We show that $\mathcal{R} = \{< P, Q > | h([\![P]\!]) = h([\![Q]\!])\}$ is a dynamic bisimulation.
Let $(P, Q) \in \mathcal{R}$ and $\mathcal{C}[\ ]$ be a context.
If $\mathcal{C}[P] \overset{s}{\Longrightarrow} P'$ then there exists $t : [\![\mathcal{C}[P]]\!] \overset{s}{\Longrightarrow} [\![P']\!]$ in $\Im$. Then $h(t) : h([\![\mathcal{C}[P]]\!]) \overset{s}{\Longrightarrow} w$ with $w = h([\![P']\!])$. Now, since $h([\![\mathcal{C}[P]]\!]) = h([\![\mathcal{C}[Q]]\!])$ (by lemma 3.14), we have $h(t) = t' : h([\![\mathcal{C}[Q]]\!]) \overset{s}{\Longrightarrow} w$ and so by definition of tp-homomorphism there exists $t'' : [\![\mathcal{C}[Q]]\!] \overset{s}{\Longrightarrow} [\![Q']\!]$ with $h([\![Q']\!]) = w$. Hence, there exists Q' such that $\mathcal{C}[Q] \overset{s}{\Longrightarrow} Q'$ and since $h([\![P']\!]) = w = h([\![Q']\!])$, it results $(P', Q') \in \mathcal{R}$.
Simmetrically, if $\mathcal{C}[Q] \overset{s}{\Longrightarrow} Q'$ then there exists $\mathcal{C}[P] \overset{s}{\Longrightarrow} P'$ such that $(P', Q') \in \mathcal{R}$.
Therefore, by definition of Φ_d, $(P, Q) \in \Phi_d(\mathcal{R})$ and so $\mathcal{R} \subseteq \Phi_d(\mathcal{R})$. □

Proposition 3.18 *(Dynamic Congruence $\Rightarrow$ tp-homomorphism)*
There exists an object $\Im/_{\approx^d}$ of **CatLCCS** *such that the unique morphism $h_{\approx^d} : \Im \to \Im/_{\approx^d}$ is a tp-homomorphism. Moreover, $P \approx^d Q$ implies $h_{\approx^d}([\![P]\!]) = h_{\approx^d}([\![Q]\!])$.*

Proof. Let Θ be the type congruence over $\Im$ (see definition B.3) given by
$[\![P]\!] \equiv_\Theta [\![Q]\!]$ if and only if $P \approx^d Q$;
$(t_1 : u_1 \overset{s}{\Longrightarrow} v_1) \equiv_\Theta (t_2 : u_2 \overset{s}{\Longrightarrow} v_2)$ if and only if $u_1 \equiv_\Theta u_2$ and $v_1 \equiv_\Theta v_2$,
where $:_\Theta$ is the relation obtained by adding to the typing relation of $\Im$ the new pairs

induced by $\equiv_\Theta$, i.e. by giving also type $\rightarrow$ to computations which are equated to transitions. $\Im/_{\approx^d}$ is obtained as the quotient of $\Im$ modulo Θ.
The canonical map $h_{\approx^d} : \Im \rightarrow \Im/_{\approx^d}$ which sends each element in its congruence class is a tp-homomorphism.
Morphism $h_{\approx^d}$ is a CatLCCS morphism because Θ is a congruence (that derives from the fact that $\approx^d$ is a congruence). Moreover, $h_{\approx^d}$ is obviously surjective.
Let $t_1 : h_{\approx^d}(u) \Rightarrow w$ be an element of $\Im/_{\approx^d}$. We have to show that there exists an element t of $\Im$ such that $t : u \Rightarrow v$. By construction we have $t_1 = h_{\approx^d}(\overline{t}) = [\overline{t}]_\Theta$ for some $\overline{t}$ in $\Im$ and for each $t' : u' \Rightarrow v'$ in the congruence class $[\overline{t}]_\Theta$ we have $u \equiv_\Theta u'$. Certainly there exists an element with source state u in $[\overline{t}]_\Theta$, otherwise $h_{\approx^d}(u)$ should not be the source state of t_1. □

Thus, we have that the equivalence induced by $h_{\approx^d}$ on $\Im$ coincides with $\approx^d$.

Corollary 3.19 *(Correspondence with Dynamic Congruence)*
$P \approx^d Q$ if and only if $h_{\approx^d}(\llbracket P \rrbracket) = h_{\approx^d}(\llbracket Q \rrbracket)$. □

Finally, we prove that $\Im/_{\approx^d}$ is the terminal object in the subcategory of the objects reachable from $\Im$ through tp-homomorphisms.

Proposition 3.20 *($\Im/_{\approx^d}$ is terminal)*
The subcategory of **CatLCCS** *consisting of all objects reachable from $\Im$ through tp-homomorphisms, and having morphisms which are tp-homomorphisms has $\Im/_{\approx^d}$ as a terminal object.*

Proof. By initiality of $\Im$, for each C in the subcategory there exists a unique tp-homomorphism $h : \Im \rightarrow C$ and the canonical map $h_{\approx^d}$ is the unique tp-homomorphism from $\Im$ to $\Im/_{\approx^d}$.
Consider the map $k : C \rightarrow \Im/_{\approx^d}$ defined as $k(e) = h_{\approx^d}(e')$ for e' such that $h(e') = e$. Since $h(\llbracket P \rrbracket) = h(\llbracket Q \rrbracket)$ implies $P \approx^d Q$ which implies $h_{\approx^d}(\llbracket P \rrbracket) = h_{\approx^d}(\llbracket Q \rrbracket)$, we have that $h(t) = h(t')$ implies $h_{\approx^d}(t) = h_{\approx^d}(t')$ and $h(u) = h(u')$ implies $h_{\approx^d}(u) = h_{\approx^d}(u')$ and so k is well defined and is a tp-homomorphism. For any tp-homomorphism $k' : C \rightarrow \Im/_{\approx^d}$ we have $h; k = h_{\approx^d} = h; k'$ and then, since h is surjective, $k = k'$.
Therefore, there exists an unique tp-homomorphism from C to $\Im/_{\approx^d}$. □

4 Progressing Bisimulation

In this section we introduce a new bisimulation, *Progressing Bisimulation*, on the states of a labelled transition system with a distinct label τ.

We will give an *algebraic characterization* of such a bisimulation and two *modal logical languages*, in the style of HML, adequate with respect to it. Furthermore we will provide a *complete axiomatization* of Progressing Equivalence for states with finite computations.

In the next section we will see that, when the transition system is the CCS transition system, Progressing Bisimulation coincides with Dynamic Congruence, thus completing its characterization for CCS.

We reiterate our two distinct results: the first, concerning Dynamic Congruence and guided by the concept of context, and the second concerning Progressing Bisimulation and its algebraic, logical and axiomatic characterizations. Both bisimulations are very general and go beyond CCS semantics, even though Dynamic Congruence is perhaps better justified in terms of practical considerations. Moreover, in the case of CCS they coincide, giving us plenty of characterizations of Dynamic Congruence.

4.1 Operational definition and Algebraic characterization

Definition 4.1 *(Progressing Bisimulation)*
Let $T =< S, \longrightarrow, \Lambda \cup \{\tau\} >$ be a labelled transition system and $\mathcal{R}$ be a binary relation over the states of T.
Then Φ_p, a function from relations to relations, is defined as follows:

$(s, r) \in \Phi_p(\mathcal{R})$ *if and only if* $\forall \mu \in \Lambda \cup \{\tau\}$:

- *whenever* $s \xrightarrow{\mu} s'$ *there exists* r' *s.t.* $r \overset{\mu}{\Longrightarrow} r'$ *and* $(s', r') \in \mathcal{R}$;
- *whenever* $r \xrightarrow{\mu} r'$ *there exists* s' *s.t.* $s \overset{\mu}{\Longrightarrow} s'$ *and* $(s', r') \in \mathcal{R}$.

A relation $\mathcal{R}$ is called progressing bisimulation, *if and only if* $\mathcal{R} \subseteq \Phi_p(\mathcal{R})$.
The relation $\approx^p = \cup\{\mathcal{R} | \mathcal{R} \subseteq \Phi_p(\mathcal{R})\}$ *is called* Progressing Equivalence. □

Proposition 4.2 *($\approx^p$ is well–defined)*
Relation $\approx^p$ is a progressing bisimulation and is an equivalence relation. □

Now we introduce an algebraic model of a labelled transition system. As for CCS models (definition 3.9) the definition of LTS models could be given more formally. The notations used here are those defined in the previous section.

Definition 4.3 *(LTS Models)*
An LTS model *is a type algebra (with multityping) where elements typed* state *are a set, i.e. they do not have any particular algebraic structure.*
Partial operations idle and _; _ are defined so that they satisfy:

$$\frac{t : u \xrightarrow{\lambda} v}{t : u \overset{\lambda}{\Longrightarrow} v} \qquad \frac{t : u \xrightarrow{\tau} v}{t : u \overset{\epsilon}{\Longrightarrow} v} \qquad idle(v) : v \overset{\epsilon}{\Longrightarrow} v \qquad \frac{c_1 : u \overset{s_1}{\Longrightarrow} v \text{ and } c_2 : v \overset{s_2}{\Longrightarrow} w}{c_1; c_2 : u \overset{s_1 s_2}{\Longrightarrow} w}$$

Moreover, an LTS model satisfies the following equations:

$$c_1; (c_2; c_3) = (c_1; c_2); c_3 \qquad \frac{c : u \Rightarrow v}{idle(u); c = c = c; idle(v)}$$ □

Clearly, given an LTS model, elements typed by $\rightarrow$ represent transitions, elements typed by $\Rightarrow$ represent sequences of transitions (computations) and elements typed by *state* represent states of the transition system.

Definition 4.4 *(LTS Morphisms)*
An LTS morphism is a type homomorphism of LTS models that respects labelling. □

LTS morphisms also respect types, *source* and *target*.

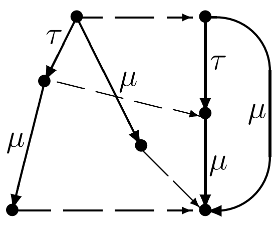
Definition 4.5 *(The Category* **LTS***)*
LTS *is the category whose objects are LTS models and whose morphisms are LTS morphisms.* □

We introduce a new kind of morphism which, besides preserving transitions, prevents τ-transitions to be mapped to *idle*-transitions. We refer to them as *progressing transition preserving morphisms.*

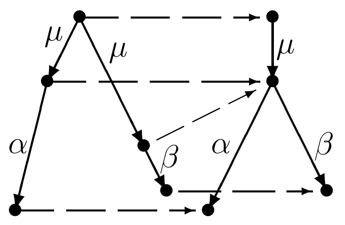
Definition 4.6 *(Progressing Transition Preserving Morphism)*
An LTS morphism $h : T \to T'$ is a progressing transition preserving morphism*, ptp-morphism for short, if and only if:*

- $h : T \to T'$ *is a surjective LTS morphism;*
- $t' \in T'$, $t' : h(s) \Rightarrow r'$ *implies* $\exists t \in T$, $t : s \Rightarrow r$ *with* $h(t) = t'$*;*
- $h(t) = idle(h(s))$ *implies* $t = idle(s)$. □

Example 4.7 *(ptp-morphisms map $\tau.\mu + \mu$ to $\tau.\mu$ but not $\mu.\alpha + \mu.\beta$ to $\mu.(\alpha + \beta)$)*



(C)



(D)

Cases (A) and (B) of example 3.16 are not ptp-morphisms, the first because it does not preserve transitions, the second because it maps a not idle to an idle transition.
In case (C) we have a ptp-morphism mapping a $\xrightarrow{\mu}$ transition to a computation $\xrightarrow{\tau};\xrightarrow{\mu} = \overset{\mu}{\Longrightarrow}$, while the morphism in case (D) is not a ptp, for it does not preserve transitions. □

In the following we will point out the correspondence between ptp-morphisms and progressing bisimulations. The approach runs parallel to the one in section 3.2 and the proofs, when formally identical, are omitted.

Proposition 4.8 *(ptp-morphism ⇒ Progressing Bisimulation)*
If $h : T \to T'$ is a ptp-morphism then $h(s) = h(r)$ implies $s \approx^p r$. □

Proposition 4.9 *(Progressing Equivalence ⇒ ptp-morphism)*
There exists an object $T/_{\approx^p}$ of **LTS** *and a ptp-morphism $h_{\approx^p} : T \to T/_{\approx^p}$ such that $s \approx^p r$ implies $h_{\approx^p}(s) = h_{\approx^p}(r)$.* □

Thus, we have:

Corollary 4.10 *(Correspondence with Progressing Equivalence)*
$s \approx^p r$ if and only if $h_{\approx^p}(s) = h_{\approx^p}(r)$. □

Similar to the case of $\Im/_{\approx^d}$, $T/_{\approx^p}$ is characterized by a property of finality in the subcategory of **LTS** defined by ptp-morphisms.

Proposition 4.11 *($T/_{\approx^p}$ is terminal)*
The subcategory of **LTS** *consisting of all objects reachable from T through ptp-morphisms, and having morphisms which are ptp-morphisms, has $T/_{\approx^p}$ as a terminal object.*

Proof. It is easy to show that $h_{\approx^p}$ is the unique ptp-morphism from T to $T/_{\approx^p}$, by the properties of congruence classes. The existence and unicity of a ptp-morphism from a generic object of the subcategory to $T/_{\approx^p}$ can be shown as in proposition 3.20. □

4.2 Logical characterization

In this subsection we design two modal logical languages in the style of HML which are adequate with respect to Progressing Bisimulation, i.e. two states are progressing equivalent if and only if they cannot be distinguished by any formula of the language.

The proofs follow Milner's scheme in [Mil89].

Definition 4.12 *(The language $PL^{\approx^p}$)*
$PL^{\approx^p}$ is the smallest class of formulae containing the following:

- *If $\varphi \in PL^{\approx^p}$ then $\forall s \in \Lambda^*$ $\langle\langle s\rangle\rangle_p\, \varphi \in PL^{\approx^p}$*
- *If $\varphi \in PL^{\approx^p}$ then $\neg\varphi \in PL^{\approx^p}$*
- *If $\varphi_i \in PL^{\approx^p}$ $\forall i \in I$ then $\bigwedge_{i\in I} \varphi_i \in PL^{\approx^p}$, where I is an index set.* □

Definition 4.13 *(Satisfaction relation)*
The validity of a formula $\varphi \in PL^{\approx^p}$ at state r ($r \models \varphi$) is inductively defined as follows:

- *$r \models \langle\langle s\rangle\rangle_p\, \varphi$ if and only if $\exists r'$ s.t. $r \overset{\overline{s}}{\Longrightarrow} r'$ and $r' \models \varphi$ where $\overline{s} = \begin{cases} s \text{ if } s \neq \epsilon \\ \tau \text{ otherwise} \end{cases}$*
- *$r \models \neg\varphi$ if and only if* not $r \models \varphi$
- *$r \models \bigwedge_{i\in I} \varphi_i$ if and only if $\forall i \in I$ $r \models \varphi_i$.* □

Now, we introduce a language whose modal operator may consider τ's, with the meaning that at least those τ's must be observed in the models.
In the following, A^+ will mean $A^* \setminus \{\epsilon\}$.

Definition 4.14 *(The language $PL_\tau^{\approx^p}$)*
$PL_\tau^{\approx^p}$ is the smallest class of formulae containing the following:

- *If $\varphi \in PL_\tau^{\approx^p}$ then $\forall t \in (\Lambda \cup \{\tau\})^+$ $\langle\langle t\rangle\rangle_\tau\, \varphi \in PL_\tau^{\approx^p}$*
- *If $\varphi \in PL_\tau^{\approx^p}$ then $\neg\varphi \in PL_\tau^{\approx^p}$*
- *If $\varphi_i \in PL_\tau^{\approx^p}$ $\forall i \in I$ then $\bigwedge_{i\in I} \varphi_i \in PL_\tau^{\approx^p}$, where I is an index set.* □

Definition 4.15 *(Satisfaction relation)*
The validity of a formula $\varphi \in PL_\tau^{\approx^p}$ at state r ($r \models_\tau \varphi$) is inductively defined as follows:

- *$r \models_\tau \langle\langle t\rangle\rangle_\tau\, \varphi$ if and only if $\exists r'$ s.t. $r \overset{t}{\Longrightarrow} r'$ and $r' \models_\tau \varphi$*
- *$r \models_\tau \neg\varphi$ if and only if* not $r \models_\tau \varphi$
- *$r \models_\tau \bigwedge_{i\in I} \varphi_i$ if and only if $\forall i \in I$ $r \models_\tau \varphi_i$.* □

The languages so given look very similar. The difference between them is that $PL^{\approx^p}$ does not consider τ's in its modal operator, but takes care of them in the satisfaction relation, while the reverse holds for $PL_\tau^{\approx^p}$. The language used is a matter of taste: indeed, we will show that the equivalences they induce on CCS agents coincide.

Proposition 4.16 *($PL_\tau^{\approx^p}$ is more discriminating than $PL^{\approx^p}$)*
For each $\varphi \in PL^{\approx^p}$ there exists $\psi \in PL_\tau^{\approx^p}$ such that $\forall r\ r \models \varphi \Leftrightarrow r \models_\tau \psi$.

Proof. By structural induction on φ.
The interesting case is when $\varphi \equiv \langle\!\langle s\rangle\!\rangle_p \varphi'$. By induction $\exists \psi' \in PL_\tau^{\approx^p}$ such that $\forall r\ r \models \varphi' \Leftrightarrow r \models_\tau \psi'$. Let $\psi \equiv \langle\!\langle s\rangle\!\rangle_\tau \psi'$ if $s \neq \epsilon$ otherwise $\psi \equiv \langle\!\langle \tau\rangle\!\rangle_\tau \psi'$.
Obviously, $\psi \in PL_\tau^{\approx^p}$ and the thesis holds. □

Proposition 4.17 *($PL^{\approx^p}$ is more discriminating than $PL_\tau^{\approx^p}$)*
For each $\psi \in PL_\tau^{\approx^p}$ there exists $\varphi \in PL^{\approx^p}$ such that $\forall r\ r \models_\tau \psi \Leftrightarrow r \models \varphi$.

Proof. By structural induction on ψ.
Again, the interesting case is when $\psi \equiv \langle\!\langle t\rangle\!\rangle_\tau \psi'$. By induction $\exists \varphi' \in PL^{\approx^p}$ such that $\forall r\ r \models_\tau \psi' \Leftrightarrow r \models \varphi'$. Let $t \equiv s_1 \tau s_2 \cdots \tau s_n$, where $s_i \in \Lambda^*$.
Then it is easy to see that $\varphi \equiv \langle\!\langle s_1\rangle\!\rangle_p \langle\!\langle \epsilon\rangle\!\rangle_p \langle\!\langle s_2\rangle\!\rangle_p \cdots \langle\!\langle \epsilon\rangle\!\rangle_p \langle\!\langle s_n\rangle\!\rangle_p \varphi'$, where only the $s_i \neq \epsilon$ have a corresponding $\langle\!\langle s_i\rangle\!\rangle_p$, satisfies the thesis. □

From the previous two propositions, it is easy to infer the following corollary.

Corollary 4.18 *($PL^{\approx^p}$ and $PL_\tau^{\approx^p}$ induce the same equivalence)*
$\forall \varphi \in PL^{\approx^p}\ s \models \varphi \Leftrightarrow r \models \varphi$ if and only if $\forall \psi \in PL_\tau^{\approx^p}\ s \models_\tau \psi \Leftrightarrow r \models_\tau \psi$. □

In the rest of this subsection we will show that the equivalences induced by $PL^{\approx^p}$ and $PL_\tau^{\approx^p}$ coincide with $\approx^p$.

In order to appreciate the difference between $\sim$, $\approx^p$ and $\approx$, it is useful to observe that, replacing $r \xRightarrow{t} r'$ by $r \xrightarrow{t} r'$ in the satisfaction relation clause for $\langle\!\langle t\rangle\!\rangle_\tau \varphi$, we obtain the HML for $\sim$ and that, redefining $\overline{s}$ to be simply s in the clause for $\langle\!\langle s\rangle\!\rangle_p \varphi$, we obtain the HML for $\approx$.

We start by defining a concept of *depth* for formulas which induces a natural stratification on $PL_\tau^{\approx^p}$ useful to prove the results in this section. In the following λ and κ will range over *ordinal* numbers and ϑ will denote the class of all *ordinals.*

Definition 4.19 *(Depth of $PL_\tau^{\approx^p}$ formulas)*
The depth *of a formula $\varphi \in PL_\tau^{\approx^p}$ is an ordinal number defined inductively as follow:*

- *$depth(\langle\!\langle t\rangle\!\rangle_\tau \varphi) = depth(\varphi) + 1$;*
- *$depth(\neg\varphi) = depth(\varphi)$;*
- *$depth(\bigwedge_{i\in I} \varphi_i) = \sup_{i\in I}\{depth(\varphi_i)\}$.* □

Definition 4.20 *(Stratification of $PL_\tau^{\approx^p}$)*
For each ordinal κ, $PL_{\tau\kappa}^{\approx^p} = \{\varphi \in PL_\tau^{\approx^p} | depth(\varphi) \leq \kappa\}$. □

In what follows, we will use a slightly modified version of $\approx^p$. This is convenient from the technical point of view and, moreover, gives us a chance to formulate another definition of $\approx^p$.

Definition 4.21 *(Function Φ_s)*
Let $\mathcal{R}$ be a binary relation over the state of an LTS.
Then Φ_s, a function from relations to relations, is defined as follows:
$(s, r) \in \Phi_s(\mathcal{R})$ *if and only if* $\forall t \in (\Lambda \cup \{\tau\})^+$*:*

- *whenever* $s \xrightarrow{t} s'$ *there exists* r' *s.t.* $r \overset{t}{\Longrightarrow} r'$ *and* $(s', r') \in \mathcal{R}$*;*
- *whenever* $r \xrightarrow{t} r'$ *there exists* s' *s.t.* $s \overset{t}{\Longrightarrow} s'$ *and* $(s', r') \in \mathcal{R}$. □

Function Φ_s has the same maximal fixed-point of Φ_p, that is it only redefines $\approx^p$, as the following shows.

Proposition 4.22 *(Φ and Φ_s have the same pre-fixed-points)*
$\mathcal{R} \subseteq \Phi_p(\mathcal{R})$ *if and only if* $\mathcal{R} \subseteq \Phi_s(\mathcal{R})$.

Proof. ($\Rightarrow$) If $\mathcal{R}$ is a progressing bisimulation then if $(s, r) \in \mathcal{R}$ we have that $(s, r) \in \Phi_p(\mathcal{R})$. Hence, if $s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} \cdots \xrightarrow{t_n} s'$ then $r \overset{t_1}{\Longrightarrow} r_1 \overset{t_2}{\Longrightarrow} \cdots \overset{t_n}{\Longrightarrow} r'$ with (s_i, r_i), $(s', r') \in \mathcal{R}$, that is $r \overset{t}{\Longrightarrow} r'$. The same if $r_1 \xrightarrow{t_1} r_2 \xrightarrow{t_2} \cdots \xrightarrow{t_n} r'$. Therefore, by definition of Φ_s, we have $(s, r) \in \Phi_s(\mathcal{R})$ and so $\mathcal{R} \subseteq \Phi_s(\mathcal{R})$.
($\Leftarrow$) If $(s, r) \in \mathcal{R}$ then $\forall \mu \in \Lambda \cup \{\tau\}$ $s \xrightarrow{\mu} s'$ implies $r \overset{\mu}{\Longrightarrow} r'$ with $(s', r') \in \mathcal{R}$ and $r \xrightarrow{\mu} r'$ implies $\exists s \overset{\mu}{\Longrightarrow} s'$ with $(s', r') \in \mathcal{R}$.
Therefore $(s, r) \in \Phi_p(\mathcal{R})$ and so $\mathcal{R} \subseteq \Phi_p(\mathcal{R})$. □

Corollary 4.23 *(Φ and Φ_s have the same maximal fixed-point)*
$\bigcup\{\mathcal{R} | \mathcal{R} \subseteq \Phi_p(\mathcal{R})\} = \approx^p = \bigcup\{\mathcal{R} | \mathcal{R} \subseteq \Phi_s(\mathcal{R})\}$. □

We give now a stratified version of $\approx^p$ and relate the discriminating power of each strate to the discriminating power of the corresponding strate of $PL_\tau^{\approx^p}$ formulas.

Definition 4.24 *(Stratification of $\approx^p$)*

i. $s \approx_0^p r$ *holds for each* s *and* r*;*

ii. $s \approx_{\kappa+1}^p r$ *holds if and only if* $\forall t \in (\Lambda \cup \{\tau\})^+$

- *whenever* $s \xrightarrow{t} s'$ *there exists* r' *s.t.* $r \overset{t}{\Longrightarrow} r'$ *and* $s' \approx_\kappa^p r'$*;*
- *whenever* $r \xrightarrow{t} r'$ *there exists* s' *s.t.* $s \overset{t}{\Longrightarrow} s'$ *and* $s' \approx_\kappa^p r'$*;*

iii. *For each limit ordinal* λ, $s \approx_\lambda^p r$ *if and only if for all* $\kappa < \lambda$ $s \approx_\kappa^p r$ *holds.* □

Proposition 4.25 *(Stratification of $\approx^p$)*
$\approx^p = \bigcap_{\kappa \in \vartheta} \approx_\kappa^p$.

Proof. Note that Φ_s is a monotone mapping on the complete lattice of the binary relations over the states S of an LTS with the order given by set inclusion. Note also that $\Phi_s(\approx_\kappa^p) = \approx_{\kappa+1}^p$ for each ordinal κ and $\bigcap_{\kappa<\lambda} \Phi_s(\approx_\kappa^p) = \approx_\lambda^p$ for each limit ordinal λ. So, using standard notations and results from fixed-point theory, we have $\approx^p = \bigcup\{\mathcal{R} | \mathcal{R} \subseteq \Phi_s(\mathcal{R})\} = gfp(\Phi_s) = \bigcap_{\kappa \in \vartheta} \Phi_s^\kappa(S \times S) = \bigcap_{\kappa \in \vartheta} \approx_\kappa^p$. □

The following establishes the important correspondence between $PL^{\approx^p}_{\tau\kappa}$ and $\approx^p_\kappa$.

Proposition 4.26 *($PL^{\approx^p}_{\tau\kappa}$ is adequate w.r.t. $\approx^p_\kappa$)*
For each $\kappa \in \vartheta$, $s \approx^p_\kappa r$ if and only if $\forall \varphi \in PL^{\approx^p}_{\tau\kappa}$ $s \models_\tau \varphi \Leftrightarrow r \models_\tau \varphi$.

Proof. ($\Rightarrow$) Assume that $s \approx^p_\kappa r$ and that $s \models_\tau \varphi$, where $depth(\varphi) \leq \kappa$. We show that $r \models_\tau \varphi$. The critical case is when $\varphi \equiv \langle\langle t\rangle\rangle_\tau \varphi'$, where $depth(\varphi') = \lambda < \kappa$. By assumption $\exists s \overset{t}{\Longrightarrow} s'$ and $s' \models_\tau \varphi'$ and since $\lambda + 1 \leq \kappa$ we have $s \approx^p_{\lambda+1} r$, so $\exists r \overset{t}{\Longrightarrow} r'$ and $s' \approx^p_\lambda r'$. Hence, by inductive hypothesis, we have $r' \models_\tau \varphi'$ and so $r \models_\tau \langle\langle t\rangle\rangle_\tau \varphi' \equiv \varphi$ as required. Simmetrically, if $r \models_\tau \varphi$ then $s \models_\tau \varphi$.
($\Leftarrow$) Assume that $s \not\approx^p_\kappa r$. We can find a formula $\varphi \in PL^{\approx^p}_{\tau\kappa}$ such that $s \models_\tau \varphi$ and $r \not\models_\tau \varphi$. If κ is a limit ordinal then, by definition, $s \not\approx^p_\lambda r$ for some $\lambda < \kappa$ and by induction we have φ. Otherwise $\kappa = \lambda + 1$ and for some t and s' we have $s \xrightarrow{t} s'$ and for every r' if $r \overset{t}{\Longrightarrow} r'$ then $s' \not\approx^p_\lambda r'$. Let $\{r_i | i \in I\}$ be the set of the t-derivates of r. By induction $\forall i \in I$ $\exists \varphi_i \in PL^{\approx^p}_{\tau\lambda}$ such that $s' \models_\tau \varphi_i$ and $r_i \not\models_\tau \varphi_i$.
Defining $\varphi \equiv \langle\langle t\rangle\rangle_\tau \bigwedge_{i\in I} \varphi_i$ we are done, since $\varphi \in PL^{\approx^p}_{\tau\kappa}$, $s \models_\tau \varphi$ and $r \not\models_\tau \varphi$. □

Finally, as corollaries we obtain the results.

Corollary 4.27 *($PL^{\approx^p}_{\tau}$ is adequate w.r.t. $\approx^p$)*
$s \approx^p r$ if and only if $\forall \varphi \in PL^{\approx^p}_{\tau}$ $s \models_\tau \varphi \Leftrightarrow r \models_\tau \varphi$. □

Corollary 4.28 *($PL^{\approx^p}$ is adequate w.r.t. $\approx^p$)*
$s \approx^p r$ if and only if $\forall \varphi \in PL^{\approx^p}$ $s \models \varphi \Leftrightarrow r \models \varphi$. □

4.3 Axiomatic characterization

Going back to CCS, in this subsection we give a complete axiomatization of $\approx^p$ for finite CCS agents. It is worth noticing that every labelled transition system with finite computations can be represented by a finite sequential CCS agent in a straightforward way.

Obviously, to carry on a proof with axioms and equational reasoning, we need an observational equivalence which, is actually a congruence. For the moment let us assume that $\approx^p$ is a congruence with respect the CCS operators: in the next section we will prove that this is indeed the case (see proposition 5.4).

The proofs follow Milner's scheme in [Mil89].

Let us begin relating $\approx^p$ on CCS agents to $\sim$ and $\approx^c$.

Lemma 4.29 *(Strong Bisimulations are Progressing Bisimulations)*
$\mathcal{R} \subseteq \Psi(\mathcal{R})$ implies $\mathcal{R} \subseteq \Phi_p(\mathcal{R})$.

Proof. Directly from the definitions of Ψ and Φ_p. □

Corollary 4.30 *(Strong Congruence refines Progressing Bisimulation)*
$\sim \subseteq \approx^p$, where $\sim$ is Strong Observational Congruence. □

Thus $\approx^p$ inherits the properties stated in propositions A.1 and A.2, i.e. Monoid laws and the Expansion Theorem. Concerning the τ-laws (see proposition A.3) we have:

Proposition 4.31 *(Progressing Bisimulation and τ-laws)*
i. $P + \tau.P \approx^p \tau.P$
ii. $\alpha.(P + \tau.Q) + \alpha.Q \approx^p \alpha.(P + \tau.Q)$
iii. $\alpha.\tau.P \not\approx^p \alpha.P$

Proof. *i)* and *ii)* can be showed simply exhibiting opportune progressing bisimulations. For *iii)* it sufficies to consider the agents $\alpha.\tau.nil$ and $\alpha.nil$. □

Definition 4.32 *(Finite and Serial Agents)*
A CCS agent is finite *if it does not contain Recursion and it is* serial *if it contains no Parallel Composition, Restriction or Relabelling.* □

It is clear that with the use of the *expansion theorem* every finite agent can be equated to a finite serial agent. Therefore, a complete axiomatization for finite and serial agents can be considered an axiomatization for finite agents (considering the expansion theorem as an axiom scheme). In the rest of the subsection every CCS agent must be considered finite and serial.

We introduce a new set of axioms very similar to the ones given for Strong and Weak Observational Congruence: it contains the monoid laws and two of the three τ-laws.

Definition 4.33 *(Axiom System $\mathcal{A}$)*

$\mathbf{A_1} : P + Q = Q + P$	$(\mathbf{T_1} : \mu.\tau.P = \mu.P)$
$\mathbf{A_2} : P + (Q + R) = (P + Q) + R$	$\mathbf{T_2} : P + \tau.P = \tau.P$
$\mathbf{A_3} : P + P = P$	$\mathbf{T_3} : \mu.(P + \tau.Q) + \mu.Q = \mu.(P + \tau.Q)$
$\mathbf{A_4} : P + nil = P$	

$$\mathcal{A} = \{\mathbf{A_1}, \mathbf{A_2}, \mathbf{A_3}, \mathbf{A_4}\} \cup \{\mathbf{T_2}, \mathbf{T_3}\}$$ □

We will prove that $\mathcal{A}$ is a complete set of axioms when $=$ is understood as $\approx^p$, i.e. two agents are progressing equivalent if and only if they can be proved equal by the axioms of $\mathcal{A}$ and the usual equational deduction. The fact that the equality of P and Q can be proved from $\mathcal{A}$ by equational deduction will be indicated in the following by $\mathcal{A} \vdash P = Q$.

In order to fully understand the relation between $\sim$, $\approx^p$ and $\approx^c$, it is useful to remind that $\{\mathbf{A_1}, \mathbf{A_2}, \mathbf{A_3}, \mathbf{A_4}\}$ and $\mathcal{A} \cup \{\mathbf{T_1}\}$ are complete axiomatizations, respectively, of $\sim$ and $\approx^c$.

Definition 4.34 *(Standard Form and Depth)*
P is in Standard Form *(SF) if*

i. $P \equiv nil$ *or* $P \equiv \sum_i \mu_i.P_i$ *where each* P_i *is also in Standard Form;*
ii. *whenever* $P \overset{\mu}{\Longrightarrow} P'$ *then* $P \xrightarrow{\mu} P'$.

The depth *of P is the maximum number of nested prefix in P.* □

Lemma 4.35 *(Lifting Lemma)*
If $P \overset{\mu}{\Longrightarrow} P'$ then $\mathcal{A} \vdash P = P + \mu.P'$.

Proof. We proceed by induction on the structure of P.
Consider three cases for $P \overset{\mu}{\Longrightarrow} P'$:
case 1) $\mu.P'$ is a summand of P. Then the conclusion holds by $\mathbf{A_3}$.
case 2) $\mu.Q$ is a summand of P and $Q \overset{\tau}{\Longrightarrow} P'$.
Then by induction $\mathcal{A} \vdash Q = Q + \tau.P'$,

$$\begin{aligned}
\text{so } \mathcal{A} \vdash P &= P + \mu.Q && \text{(by } \mathbf{A_3}\text{)}\\
&= P + \mu.(Q + \tau.P') && \text{(by induction)}\\
&= P + \mu.(Q + \tau.P') + \mu.P' && \text{(by } \mathbf{T_3}\text{)}\\
&= P + \mu.P' && \text{(by previous step reversed).}
\end{aligned}$$

case 3) $\tau.Q$ is a summand of P and $Q \overset{\mu}{\Longrightarrow} P'$.
Then by induction $\mathcal{A} \vdash Q = Q + \mu.P'$,

$$\begin{aligned}
\text{so } \mathcal{A} \vdash P &= P + \tau.Q && \text{(by } \mathbf{A_3}\text{)}\\
&= P + \tau.Q + Q && \text{(by } \mathbf{T_2}\text{)}\\
&= P + \tau.Q + Q + \mu.P' && \text{(by induction)}\\
&= P + \mu.P' && \text{(by previous step reversed)}
\end{aligned}$$

□

Lemma 4.36 *(Reduction to Standard Form)*
For any P there is P' in SF with the same depth of P such that $\mathcal{A} \vdash P = P'$.

Proof. We can assume $P \equiv nil$ or $P \equiv \sum_i \mu_i.P_i$ and that the same condition holds for each P_i, since, by using $\mathbf{A_4}$, every *nil* summand can be eliminated. We proceed by induction on the structure of P. In the base case $P \equiv nil$ that is in SF, otherwise for each summand $\mu.Q$ of P we can assume by induction that Q is already in SF without depth increase. In every case in which $P \overset{\mu_i}{\Longrightarrow} P_i$ but not $P \xrightarrow{\mu_i} P_i$, P_i must be in SF since it must be a subexpression of some summand of P. So $P' \equiv P + \sum_i \mu_i.P_i$ is in SF and its depth is the one of P. So, by lemma 4.35, we have $\mathcal{A} \vdash P = P'$. □

Proposition 4.37 *($\mathcal{A}$ is a complete axiomatization of $\approx^p$)*
$P \approx^p Q$ if and only if $\mathcal{A} \vdash P = Q$.

Proof. ($\Leftarrow$) From corollary 4.30 and proposition 4.31, it follows that axioms $\mathcal{A}$ are all true when $=$ is understood as $\approx^p$ and that is enough.
($\Rightarrow$) We can assume, by the previous lemma 4.36, that P and Q are in SF; the proof proceeds by induction on the sum of the depths of P and Q. The base of induction is obvious: $P \equiv nil \equiv Q$.
Let $\mu.P'$ be a summand of P. Then Q has a summand $\mu.Q'$ provably equal to $\mu.P'$. In fact $P \xrightarrow{\mu} P'$ implies that $\exists Q \overset{\mu}{\Longrightarrow} Q'$, so $Q \xrightarrow{\mu} Q'$ and so $\mu.Q'$ is a summand of Q and $P' \approx^p Q'$. Hence, by induction, $\mathcal{A} \vdash P' = Q'$. Similarly, each summand of Q can be proved equal to a summand of P. Finally, by using $\mathbf{A_3}$ to eliminate the duplicate summands, we conclude $\mathcal{A} \vdash P = Q$. □

5 Dynamic and Progressing Bisimulations

In this section we show that Dynamic Congruence and Progressing Bisimulation *coincide* when $\approx^p$ is considered on CCS.

This gives many characterizations to $\approx^d$: in fact, we have two characterizations by finality through particular kinds of *abstraction morphisms* (the one encoding the CCS algebra into states and transitions, the other just considering the naked labelled transition system), two logical characterizations via HML-like modal languages and, finally, an axiomatization for finite agents, besides the two operational characterizations given by the *bisimulation game.*

To carry on the proof, we need the technical tool introduced in the next definition. In the following the juxtaposition of relations will denote their composition, formally defined by $a\mathcal{R}\mathcal{R}'c$ if and only if $\exists b$ such that $a\mathcal{R}b\mathcal{R}'c$.

Definition 5.1 *(Progressing Bisimulations up to $\approx^p$)*
$\mathcal{R}$ is a progressing bisimulation up to $\approx^p$ *if and only if*

$(P,Q) \in \mathcal{R}$ if and only if $\forall \mu \in \Lambda \cup \{\tau\}$:

- *whenever $P \xrightarrow{\mu} P'$ there exists Q' s.t. $Q \overset{\mu}{\Longrightarrow} Q'$ and $P' \approx^p \mathcal{R} \approx^p Q'$;*
- *whenever $Q \xrightarrow{\mu} Q'$ there exists P' s.t. $P \overset{\mu}{\Longrightarrow} P'$ and $P' \approx^p \mathcal{R} \approx^p Q'$.* □

Lemma 5.2 *(Progressing up to $\approx^p$* vs. *Progressing Bisimulations)*
If $\mathcal{R}$ is a progressing bisimulation up to $\approx^p$ then $\approx^p \mathcal{R} \approx^p$ is a progressing bisimulation.

Proof. Let suppose $P \approx^p \mathcal{R} \approx^p Q$, that is $P \approx^p P''\mathcal{R}Q'' \approx^p Q$.
If $P \xrightarrow{\mu} \overline{P}'$ then $P'' \overset{\mu}{\Longrightarrow} \overline{P}''$ with $\overline{P}' \approx^p \overline{P}''$, and so, by repeated applications of the definition of progressing bisimulation up to $\approx^p$, $\exists Q'' \overset{\mu}{\Longrightarrow} \overline{Q}''$ with $\overline{P}'' \approx^p \mathcal{R} \approx^p \overline{Q}''$. So $\exists Q' \overset{\mu}{\Longrightarrow} \overline{Q}'$, with $\overline{Q}'' \approx^p \overline{Q}'$. Therefore $\overline{P}' \approx^p \overline{P}'' \approx^p \mathcal{R} \approx^p \overline{Q}'' \approx^p \overline{Q}'$, that is $\overline{P}' \approx^p \mathcal{R} \approx^p \overline{Q}'$. Then $\approx^p \mathcal{R} \approx^p \subseteq \Phi_p(\approx^p \mathcal{R} \approx^p)$. □

Proposition 5.3 *(Progressing Bisimulations up to $\approx^p \Rightarrow \approx^p$)*
If $\mathcal{R}$ is a progressing bisimulation up to $\approx^p$ then $\mathcal{R} \subseteq \approx^p$.

Proof. For the previous lemma $\approx^p \mathcal{R} \approx^p$ is a progressing bisimulation and so $\approx^p \mathcal{R} \approx^p \subseteq \approx^p$. Since the identity relation over CCS expressions $Id_{CCS} \subseteq \approx^p$, we have $\mathcal{R} \subseteq Id_{CCS}\, \mathcal{R}\, Id_{CCS} \subseteq \approx^p \mathcal{R} \approx^p$. □

Now, we can show that, as mentioned in section 4.3, $\approx^p$ is a congruence.

Proposition 5.4 *($\approx^p$ is a congruence)*
Let E, F be CCS expressions and $E \approx^p F$. Then

i. $\mu.E \approx^p \mu.F$
ii. $E + Q \approx^p F + Q$
iii. $E|Q \approx^p F|Q$
iv. $E[\Phi] \approx^p F[\Phi]$
v. $E \setminus \alpha \approx^p F \setminus \alpha$
vi. $rec x.E \approx^p rec x.F$

Proof.

i. Trivial.

ii. Trivial.

iii. $\mathcal{R} = \{(P_1|Q, P_2|Q) | P_1 \approx^p P_2\}$ is a progressing bisimulation.
Let $(P_1|Q, P_2|Q) \in \mathcal{R}$ and $P_1|Q \xrightarrow{\mu} P'$. We have the following three cases.
case 1) $P_1 \xrightarrow{\mu} \overline{P}_1$ and $P' \equiv \overline{P}_1|Q$. Since $P_1 \approx^p P_2$, $\exists P_2 \overset{\mu}{\Longrightarrow} \overline{P}_2$ with $\overline{P}_1 \approx^p \overline{P}_2$ and so $P_2|Q \overset{\mu}{\Longrightarrow} \overline{P}_2|Q \equiv P''$ and $(P', P'') \in \mathcal{R}$.
case 2) $Q \xrightarrow{\mu} Q'$ and $P' \equiv P_1|Q$. Then $P_2|Q \overset{\mu}{\Longrightarrow} P_2|Q' \equiv P''$ and $(P', P'') \in \mathcal{R}$.
case 3) $\mu = \tau$, $P_1 \xrightarrow{\lambda} \overline{P}_1$, $Q \xrightarrow{\overline{\lambda}} Q'$ and $P' \equiv \overline{P}_1|Q'$. Then $P_2 \overset{\lambda}{\Longrightarrow} \overline{P}_2$ and $\overline{P}_1 \approx^p \overline{P}_2$ and $P_2|Q \overset{\tau}{\Longrightarrow} \overline{P}_2|Q' \equiv P''$ and $(P', P'') \in \mathcal{R}$.
Symmetrically, if $P_2|Q \xrightarrow{\mu} P''$ then $\exists P_1|Q \overset{\mu}{\Longrightarrow} P'$ and $(P', P'') \in \mathcal{R}$.
So $(P_1|Q, P_2|Q) \in \Phi_p(\mathcal{R})$ and therefore $\mathcal{R} \subseteq \Phi_p(\mathcal{R})$.
The omitted proofs of points *iv.* and *v.* follow this same scheme.

iv. $\mathcal{R} = \{(P_1[\Phi], P_2[\Phi]) | P_1 \approx^p P_2\}$ is a progressing bisimulation.

v. $\mathcal{R} = \{(P_1 \setminus \alpha, P_2 \setminus \alpha) | P_1 \approx^p P_2\}$ is a progressing bisimulation.

vi. $\mathcal{R} = \{(G[recx.E/x], G[recx.F/x]) | G$ contains at most the variable x$\}$ is a progressing bisimulation up to $\approx^p$. Let $G[recx.E/x] \xrightarrow{\mu} P'$.
We shall prove by induction on the depth of the inference by which the action is inferred, that $\exists G[recx.F/x] \overset{\mu}{\Longrightarrow} Q''$ and $\exists Q' \approx^p Q''$ with $(P', Q') \in \mathcal{R}$.
We proceed by cases on the form of G.
case $G \equiv x$. Then $G[recx.E/x] \equiv recx.E \xrightarrow{\mu} P'$, and so $E[recx.E/x] \xrightarrow{\mu} P'$ by a shorter inference. Therefore, by induction, $E[recx.F/x] \overset{\mu}{\Longrightarrow} Q''$ and $\exists Q' \approx^p Q''$ with $(P', Q') \in \mathcal{R}$. But $E \approx^p F$, so $F[recx.F/x] \overset{\mu}{\Longrightarrow} \overline{Q}''$ and so $recx.F \overset{\mu}{\Longrightarrow} \overline{Q}''$ with $\overline{Q}'' \approx^p Q'' \approx^p Q'$ as required.
case $G \equiv \mu.G'$. Then $G[recx.E/x] \equiv \mu.G'[recx.E/x] \xrightarrow{\mu} G'[recx.E/x] \equiv P'$ and $G[recx.F/x] \equiv \mu.G'[recx.F/x] \overset{\mu}{\Longrightarrow} G'[recx.F/x] \equiv Q''$ and $(P', Q'') \in \mathcal{R}$.
case $G \equiv G_1 + G_2$. Directly from the inductive hypothesis.
case $G \equiv G_1|G_2$.
(1) $G_1[recx.E/x] \xrightarrow{\mu} \overline{P}'$ and $P' \equiv \overline{P}'|G_2[recx.E/x]$.
By induction $\exists G_1[recx.F/x] \overset{\mu}{\Longrightarrow} \overline{Q}''$ and $\exists \overline{Q}' \approx^p \overline{Q}''$ with $(\overline{P}', \overline{Q}') \in \mathcal{R}$. Hence, for some H, $\overline{P}' \equiv H[recx.E/x]$ and $\overline{Q}' \equiv H[recx.F/x]$, therefore $P' \equiv (H|G_2)[recx.E/x]$, $Q' \equiv (H|G_2)[recx.F/x]$ and so $(P', Q') \in \mathcal{R}$.
(2) $G_2[recx.E/x] \xrightarrow{\mu} \overline{P}'$ and $P' \equiv G[recx.E/x]|\overline{P}'$. Symmetric to point (1).
(3) $\mu = \tau$ and $G_1[recx.E/x] \xrightarrow{\lambda} \overline{P}'$ and $G_2[recx.E/x] \xrightarrow{\overline{\lambda}} \overline{P}''$.
Analogous to point (1).
case $G \equiv G_1 \setminus \alpha$. Then $G[recx.E/x] \equiv G_1[recx.E/x] \setminus \alpha \xrightarrow{\mu} P'$. Hence, by a shorter inference, $G_1[recx.E/x] \xrightarrow{\mu} \overline{P}'$ with $P' \equiv \overline{P}' \setminus \alpha$. By induction $\exists G_1[recx.F/x] \overset{\mu}{\Longrightarrow} \overline{Q}''$ and $\exists \overline{Q}' \approx^p \overline{Q}''$ with $(\overline{P}', \overline{Q}') \in \mathcal{R}$. Hence, $\exists G[recx.F/x] \overset{\mu}{\Longrightarrow} Q'' \equiv \overline{Q}'' \setminus \alpha$ and $\exists Q' \equiv \overline{Q}' \setminus \alpha \approx^p Q''$ with $(P', Q') \in \mathcal{R}$.
case $G \equiv G_1[\Phi]$. As in the previous case.
case $G \equiv recy.G_1$. As in the previous case.

In the same way, we have that if $G[recx.F/x] \xrightarrow{\mu} Q'$ then $\exists G[recx.E/x] \overset{\mu}{\Longrightarrow} P''$ and $\exists P' \approx^p P''$ with $(P', Q') \in \mathcal{R}$.Therefore $\mathcal{R}$ is a progressing bisimulation up to $\approx^p$. Now, choosing $G \equiv x$ we obtain $(recx.E, recx.F) \in \mathcal{R}$ and so, by proposition 5.3, $recx.E \approx^p recx.F$. □

Clearly, with that we have proved that replacing in any expression E a subexpression with a *progressing equivalent* one we obtain an expression E' *progressing equivalent* to E.

Proposition 5.5 *(Progressing Bisimulations are Weak Bisimulations)*
$\mathcal{R} \subseteq \Phi_p(\mathcal{R})$ *implies* $\mathcal{R} \subseteq \Phi(\mathcal{R})$.

Proof. Suppose that $P\mathcal{R}Q$. Then, by hypothesis, $P\Phi_p(\mathcal{R})Q$. If $P \overset{s}{\Longrightarrow} P'$ with $s \in \Lambda^*$ then $P(\xrightarrow{\tau})^*\overline{P}_1 \xrightarrow{s_1} P_1 \cdots \overline{P}_n \xrightarrow{s_n} P_n(\xrightarrow{\tau})^*P'$ and, by repeated applications of the definition of Φ_p, we have $Q \overset{\tau}{\Longrightarrow} \overline{Q}_1 \overset{s_1}{\Longrightarrow} Q_1 \cdots \overline{Q}_n \overset{s_n}{\Longrightarrow} Q_n \overset{\tau}{\Longrightarrow} Q'$ and $\overline{P}_i\mathcal{R}\overline{Q}_i$, $P_i\mathcal{R}Q_i$ and $P'\mathcal{R}Q'$, that is $Q \overset{s}{\Longrightarrow} Q'$.
So $(P, Q) \in \Phi(\mathcal{R})$ and so $\mathcal{R} \subseteq \Phi(\mathcal{R})$. □

Finally, we can show the result.

Proposition 5.6 *(Dynamic and Progressing Bisimulations coincide)*
$\approx^d = \approx^p$.

Proof. We shall prove that $\approx^d \subseteq \approx^p$ showing that $\approx^d \subseteq \Phi_p(\approx^d)$.
Suppose $P \approx^d Q$ and $P \xrightarrow{\mu} P'$. If $\mu \neq \tau$ then applying the definition of Φ_d fixing the context $\mathcal{C}[\ ] \equiv x$ we obtain that $\exists Q \overset{\mu}{\Longrightarrow} Q'$ and $P' \approx^d Q'$. If $\mu = \tau$ we have that $\exists Q \overset{\epsilon}{\Longrightarrow} Q'$ in which at least one τ move must be done and $P' \approx^d Q'$. Actually, if it were not the case we could find a context for which the definition of $\approx^d$ does not hold: let $\overline{P}$ be an agent not dynamically equivalent to each α-derivate of P', if there exists any, otherwise $\overline{P} \equiv nil$. Then the context $\mathcal{C}[\ ] \equiv x + \alpha.\overline{P}$ is such that $\mathcal{C}[P] \equiv P + \alpha.\overline{P} \xrightarrow{\tau} P'$ and $\mathcal{C}[Q] \equiv Q + \alpha.\overline{P} \overset{\epsilon}{\Longrightarrow} Q + \alpha.\overline{P}$ and $P' \not\approx^d Q + \alpha.\overline{P}$ since P' cannot match the move $Q + \alpha.\overline{P} \xrightarrow{\alpha} \overline{P}$. Therefore it must be $\exists Q \overset{\tau}{\Longrightarrow} Q'$.
Symmetrically, if $Q \xrightarrow{\mu} Q'$ then $\exists P \overset{\mu}{\Longrightarrow} P'$ and $P' \approx^d Q'$.
Then by definition of Φ_p we have $(P, Q) \in \Phi_p(\approx^d)$ and so $\approx^d \subseteq \Phi_p(\approx^d)$.
In order to get the other inclusion, it is enough to observe that $\approx^p$ is a bisimulation and a congruence, and therefore, since $\approx^d$ is the coarsest such relation, $\approx^p \subseteq \approx^d$. □

Acknowledgements
We wish to thank GianLuigi Ferrari for many interesting discussions, which have significantly contributed to our understanding of the subject and which gave us the idea of Dynamic Observational Congruence. Special thanks to Rocco De Nicola for his suggestions on the organization of the paper.

References

[Ab88] S. Abramsky. *A Domain Equation for Bisimulation.* Technical report, Department of Computing, Imperial College, London, 1988.

[Acz87] P. Aczel. *Non-Well-Founded Sets.* CSLI Lecture Notes n. 14, Stanford University, 1987.

[BB88] D. Benson and O. Ben-Shachar. Bisimulations of Automata. *Information and Computation*, n. 79, 1988.

[DDM90] P. Degano, R. De Nicola, and U. Montanari. A Partial Ordering Semantics for CCS. *Theoretical Computer Science*, n. 75, 1990.

[DeN87] R. De Nicola. Extensional Equivalence for Transition Systems. *Acta Informatica*, n. 24, 1987.

[F90] G. Ferrari. *Unifying Models of Concurrency.* PhD thesis, TD-4/90, Dipartimento di Informatica, Università di Pisa, 1990.

[FM90] G. Ferrari and U. Montanari. Towards the Unification of Models for Concurrency. In *Proceedings of CAAP '90.* LNCS n. 431, Springer Verlag, 1990.

[FMM91] G. Ferrari, U. Montanari, and M. Mowbray. On Causality Observed Incrementally, Finally. In *Proceedings of TAPSOFT '91.* LNCS n. 493, Springer Verlag, 1991.

[GTW78] J.A. Goguen, J.W. Thatcher, and E.G. Wagner. An Initial Algebra Approach to the Specification, Correctness and Implementation of Abstract Data Types. In R. Yeh, editor, *Current Trends in Programming Methodology IV.* Prentice Hall, 1978.

[GvG89] U. Goltz and R. van Glabbeek. Equivalence Notions for Concurrent System and Refinement of Actions. In *Proceedings of MFCS '89.* LNCS n. 379, Springer Verlag, 1989.

[HM85] M. Hennessy and R. Milner. Algebraic Laws for Nondeterminism and Concurrency. *Journal of the ACM*, vol. 32, 1985.

[Kel76] R. Keller. Formal Verification of Parallel Programs. *Communications of the ACM*, vol. 7, 1976.

[Mil80] R. Milner. *A Calculus of Communicating Systems.* Lecture Notes in Computer Science, n. 92. Springer Verlag, 1980.

[Mil89] R. Milner. *Concurrency and Communication.* Prentice Hall, 1989.

[MS90] U. Montanari and V. Sassone. *Dynamic Bisimulation.* Technical Report TR 13/90, Dipartimento di Informatica, Università di Pisa, 1990.

[MS91] U. Montanari and V. Sassone. CCS Dynamic Bisimulation is Progressing. In *Proceedings of MFCS '91.* LNCS n. 520, Springer Verlag, 1991.

[MSg89] U. Montanari and M. Sgamma. Canonical Representatives for Observational Equivalences Classes. In *Proceedings of Colloquium on the Resolution of Equations in Algebraic Structures.* North Holland, 1989.

[MSS90] V. Manca, A. Salibra, and G. Scollo. Equational Type Logic. *Theoretical Computer Science*, n. 77, 1990.

[NPW81] M. Nielsen, G. Plotkin, and G. Winskel. Petri Nets, Event Structures and Domains, Part 1. *Theoretical Computer Science*, n. 13, 1981.

[Par81] D. Park. Concurrency and Automata on Infinite Sequences. In *Proceedings of GI.* LNCS n. 104, Springer Verlag, 1981.

[Pet62] C.A. Petri. *Kommunikation mit Automaten.* PhD thesis, Institut für Instrumentelle Mathematik, Bonn, FRG, 1962.

[Plo81] G. Plotkin. *A Structured Approach to Operational Semantics.* DAIMI FN-19, Computer Science Dept., Aarhus University, 1981.

[Pra86] V. Pratt. Modelling Concurrency with Partial Orders. *International Journal of Parallel Programming*, n. 15, 1986.

[Tar55] A. Tarski. A Lattice Theoretical Fixpoint Theorem and its Applications. *Pacific Journal of Mathemathics*, n. 5, 1955.

[vGl87] R. van Glabbeek. Bounded Nondeterminism and the Approximation Induction Principle in Process Algebra. In *Proceedings of STACS 87.* LNCS n. 247, Springer Verlag, 1987.

[vGW89] R. van Glabbeek and W. Weijland. Branching Time and Abstraction in Bisimulation Semantics. In *Proceedings of IFIP 11th World Computer Congress*, 1989.

A CCS Laws

In this section we point out few useful properties of Strong and Weak Observational Equivalence due to Milner ([Mil89]).

Proposition A.1 *(Monoid laws)*

i. $P + Q \sim Q + P$

ii. $P + (Q + R) \sim (P + Q) + R$

iii. $P + P \sim P$

iv. $P + nil \sim P$ $\square$

An important property is the so-called *Expansion Theorem.* In accordance with this theorem the operator for the parallel composition is expressible in terms of the non-deterministic choice operator. In the following we will use the restriction with respect to a set of actions L to denote the restriction with respect to all the actions in L applied in any order. This is well-defined because restriction is a commutative operation.

Proposition A.2 *(Expansion Theorem)*
Let $P \equiv (P_1[\Phi_1]|\cdots|P_n[\Phi_n]) \setminus L$. *Then*

$$\begin{aligned} P \;\sim\; & \textstyle\sum\{\Phi_i(\mu).(P_1[\Phi_1]|\cdots|P'_i[\Phi_i]|\cdots|P_n[\Phi_n]) \setminus L \mid P_i \xrightarrow{\mu} P'_i,\ \Phi_i(\mu) \notin L \cup \overline{L}\} \;+ \\ & \textstyle\sum\{\tau.(P_1[\Phi_1]|\cdots|P'_i[\Phi_i]|\cdots|P'_j[\Phi_j]|\cdots|P_n[\Phi_n]) \setminus L \mid \\ & \qquad\qquad P_i \xrightarrow{\lambda_1} P'_i,\ P_j \xrightarrow{\lambda_2} P'_j,\ \Phi_i(\lambda_1) = \overline{\Phi_j(\lambda_2)},\ i < j\} \end{aligned}$$ $\square$

As an instance of the Expansion Theorem we have that the CCS agents $\alpha.nil|\beta.nil$ and $\alpha.\beta.nil + \beta.\alpha.nil$ are equivalent. In other words, the parallel execution of the actions α and β is represented by the non-deterministic choice of their possible interleavings.

Since $\sim\subseteq\approx^c$, $\approx^c$ inherits the properties in the propositions A.1 and A.2. Moreover there are important properties holding for $\approx^c$ but not for $\sim$: the so-called τ-laws.

Proposition A.3 *(τ-laws)*

i. $P + \tau.P \approx^c \tau.P$

ii. $\alpha.(P + \tau.Q) + \alpha.Q \approx^c \alpha.(P + \tau.Q)$

iii. $\alpha.\tau.P \approx^c \alpha.P$ $\square$

B Type Algebras

In this section we give a brief introduction to *equational type logic* and *type algebras* introduced in [MSS90]. In this presentation we will assume the reader acquainted with standard *many–sorted algebras*: a complete exposition of their theory can be found, for instance, in [GTW78]

Type algebras are essentially one–sorted total algebras endowed with typing. Elements and sorts (or types, which in this context are synonyms) are *merged* in a single carrier equipped with a binary *typing relation*, which assigns types to elements (hence types are elements themselves). In general, an operation is defined only on elements of suitable types (*partiality*), several types can be assigned to an element (*multityping*) and operations may take type arguments or yield types.

Let $\Omega = \{\Omega_n | n \in \omega\}$ be a one–sorted signature.

Definition B.1 *(Type Algebras)*
An Ω–signed type algebra *(Ω:–algebra)* $\mathcal{A}$ *is a pair* $< A, :_\mathcal{A}>$ *where* A *is a (total)* Ω*–algebra and* $:_\mathcal{A}$ *is a binary relation (the* typing relation*) on* $|\mathcal{A}|$*, the* carrier *set of* $\mathcal{A}$*.* □

Usual notions and standard results from many–sorted algebras are straightforwardly extended to type algebras just taking into account, in the most natural way, the additional requirements induced by the extra structure of types.

Definition B.2 *(Type Algebras Homomorphisms)*
An Ω–signed type homomorphism *(Ω:–homomorphisms) between two* Ω*:–algebras* $\mathcal{A}$ *and* $\mathcal{B}$ *is an* Ω*–homomorphism* $\phi : A \rightarrow B$ *such that if* $a_1 :_\mathcal{A} a_2$ *then* $\phi(a_1) :_\mathcal{B} \phi(a_2)$*.* □

The class of Ω:–algebras together with Ω:–homomorphisms gives rise to a category, in the following called **Ω–Talg**.

Definition B.3 *(Type Algebras Congruences)*
An Ω–signed type congruence *(Ω:–congruence) on* $\mathcal{A}$ *is a pair* $\Theta = <\equiv_\Theta, :_\Theta>$ *of binary relations on* $|\mathcal{A}|$*, such that:*

i. $\equiv_\Theta$ *is a* Ω*–congruence on* A
ii. *if* $a \equiv_\Theta b$ *and* $a :_\Theta c$ *then* $b :_\Theta c$
iii. *if* $a \equiv_\Theta b$ *and* $c :_\Theta a$ *then* $c :_\Theta b$
iv. $:_\mathcal{A} \subseteq :_\Theta$ □

Given an Ω:–congruence Θ on $\mathcal{A}$, we can form the quotient algebra $\mathcal{A}/_\Theta = < A/_{\equiv_\Theta}, :_{\mathcal{A}/_\Theta}>$, whose typing relation is given by $[a]_{\equiv_\Theta} :_{\mathcal{A}/_\Theta} [b]_{\equiv_\Theta}$ if and only if $a :_\Theta b$. It is immediate to see that such a definition is well–given.

Equational logic is extended to *equational type logic* by allowing formulas which can express type assignments as well as equations. In general, equational type logic formulas are conditional formulas in which type assignments and equations may occur freely both in the premises and in the conclusions. Hence, an *equational type logic presentation* merges type and equality constraints on the elements of its models.

Definition B.4 *(ET–Formulas)*
The equational type logic *(ET–logic) of signature* Ω *has the following formulas (ET–formulas):*

atomic formulas:	$t = u$	*(equations)*
	$t : u$	*(type assignments)*
general formulas:	$\Gamma \rightarrow \alpha$	

where α *is an atomic formula, called* conclusion*,* Γ *is a finite, possibly empty, set of atomic formulas, called* assumption *and* t *and* u *are terms on* Ω *with variables.* □

Notice that in sections 3.2 and 4.1 we used the "induction-rule" style to give the *ET*–formulas defining CCS and LTS models.

Definition B.5 *(ET–Presentations)*
An equational type logic presentation *(ET–presentation) is a pair* $< \Omega, E >$*, where* E *is a finite set of ET–formulas on* Ω*.* □

The satisfaction relation for *ET*–logic is an extension of that for equational logic: a type algebra $\mathcal{A}$ satisfies an atomic formula $t = u$ if and only if for any assignment ρ of elements of $\mathcal{A}$ to variables of t and u, $\rho(t) = \rho(u)$ in $\mathcal{A}$, it satisfies $t : u$ if and only if $\rho(t) :_{\mathcal{A}} \rho(u)$ for each ρ and finally, it satisfies a general formula if and only if any assignment which makes the assumption hold, makes also the conclusion hold.

The *class of models* of the *ET*–presentation $< \Omega, E >$, denoted by $\mathbf{Talg(\Omega, E)}$, is the class of the Ω:–algebras which satisfy E. $\mathbf{Talg(\Omega, E)}$, together with Ω:–homomorphisms, forms a (full) subcategory of $\mathbf{\Omega{-}Talg}$.

Finally, as for the equational case, any *ET*–presentation admits an initial model.

Proposition B.6 *(Initiality in* $\mathbf{Talg(\Omega, E)}$*)*
$\mathbf{Talg(\Omega, E)}$ *has an initial object.* □