Consiglio Nazionale delle Ricerche

ISTITUTO DI ELABORAZIONE DELLA INFORMAZIONE

PISA

Compositionality and Bisimulation

Alessandro Fantechi, Stefania Gnesi, Gioia Ristori

Nota Interna B-4-33
Agosto 1990

COMPOSITIONALITY AND BISIMULATION

A. Fantechi* , S. Gnesi*, G. Ristori**

* Istituto di Elaborazione dell'Informazione - C.N.R. - via S. Maria 46, I 56100 Pisa.
** Consorzio Pisa Ricerche- via Risorgimento 9, I 56100 Pisa.

ABSTRACT

This paper presents some results on the limits of applying a temporal semantics approach to CCS like language. This semantics is defined using a compositional approach, to provide modularity in the verification process, by which it is possible to associate a CTL* formula to each language construct. We show that such semantics cannot reach the discriminating power of bisimulation for CCS.

1. INTRODUCTION

In last years there has been a development of logics for the description of properties of concurrent processes specified by an action prefix language and with an associated operational semantics. Among these languages we can consider CCS [Mil 80] and its derivatives; the usefulness of associating suitable logics, such as modal or temporal logics to behavioural languages is due to the use of logic deductive methods to prove properties of concurrent systems, in particular for what concerns both the equivalence verification and the proof of properties.

In this context we find several attempts to define particular logics or to use existing logics in order to study the relations between the logic and the behavioural worlds: the *adequacy* of a logic w.r.t. an operational semantics [Hen 85, Pnu 85] is the key concept to establish these relations.
Adequacy however, is too weak when compositionality is required to provide modularity in the specification and verification of concurrent programs. In order to guarantee compositionality a denotational semantics can be defined using the logic itself as the denotation domain. This has been done in [Gra 86], in which an ad hoc branching logic has been defined for a CCS subset, or in [Bar 84, Bar 87], where a compositional temporal semantics of simple CSP and CCS - like languages has been given using a linear version of temporal logic.

The scope of this paper is to investigate the possibility of giving a temporal semantics to a CCS-like language, completely in accordance to bisimulation semantics, within the two requirements of i) maintaining compositionality and ii) using a standard temporal logic (by "standard" logics we intend branching or linear temporal logics as defined usually in the literature, where no operator is added with the explicit purpose of mimicking a process language construct), such as the logic CTL* [Cla 86, Eme 86].

Actually, it turns out that these requirements are too strong and do not allow to express bisimulation,

This work has been partially supported by CNR Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo, project LAMBRUSCO.

which can be expressed only by releasing one of them. When both requirements are enforced we can at most reach a weaker equivalence, simulation, which is insensitive to computation interruptions, but which still preserves the branching structure of a process.

In order to show this we have started from a subset of CCS - from now on $\mathcal{AP}$, with only the action prefixing, non deterministic choice, and recursion - providing a compositional branching temporal semantics that is fully abstract with respect to the bisimulation semantics. When we enrich $\mathcal{AP}$ with the full synchronization operator, that can be considered as the simplest parallel composition operator, we have the interesting result that it is not possible to define a compositional temporal semantics, fully abstract with respect to the bisimulation semantics, even using CTL* as the target logic. Rather, we reach a simulation equivalence. This equivalence appears to be the strongest one with respect to which it is possible to give a fully abstract compositional temporal semantics using standard logics.

2. SYNTAX AND SEMANTICS

CCS (Calculus for Communicating Systems) [Mil 80] describes the behaviour of processes and their parallel composition, employing a finite alphabet of observable actions, a finite alphabet of co-names and a silent action τ. In this context we will consider some simple languages derived from CCS in which only observable actions (with no co-names) are considered.
The syntax of these subsets, which we shall refer to as $\mathcal{AP}$, $\mathcal{AP}_i$, and $\mathcal{AP}_S$, is sketched in Figure 1, where a ranges on a finite non-empty set of actions Σ, x ranges on a finite set of variables X and p, q are terms of the proper subset.

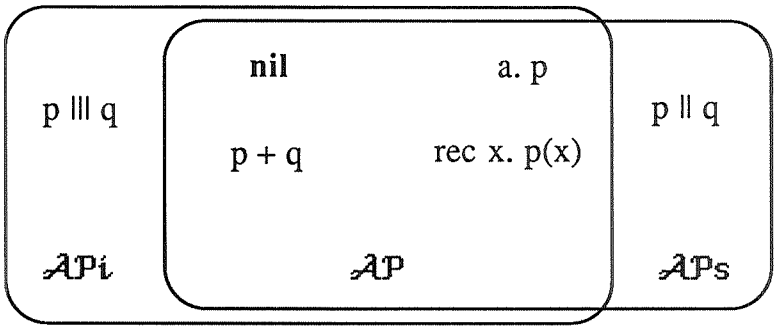

Figure 1.

We suppose that recursion is well-guarded, i.e. any occurrence of x in p is in an action prefix expression, as a.x.

The semantics of these languages is based on the concept of "Labelled Transition Systems" (LTS in the following). A LTS is a triple $(\mathcal{P},\Sigma, \{R_a, a\in \Sigma\})$ such that $\mathcal{P}$ is a set of processes, Σ is a set of actions, $R_a \subseteq \mathcal{P} \times \mathcal{P}$. We will use the notation $p\text{—}a\rightarrow q$ to mean that $(p,q)\in R_a$ and we will say that p is able to perform action 'a' and transform in q. We will use p, q (with indexes) as ranging over processes, to be understood as the behaviour expression representing a process in the language referred to in that context.
The transitional semantics for the operators of the three languages is given in Table 1.

$$a.p \xrightarrow{a} p$$

$$\frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'} \qquad \frac{q \xrightarrow{a} q'}{p + q \xrightarrow{a} q'}$$

$$\frac{p \xrightarrow{a} p'}{p \,|||\, q \xrightarrow{a} p' \,|||\, q} \qquad \frac{q \xrightarrow{a} q'}{p \,|||\, q \xrightarrow{a} p \,|||\, q'}$$

$$\frac{p \xrightarrow{a} p' \text{ and } q \xrightarrow{a} q'}{p \,||\, q \xrightarrow{a} p' \,||\, q}$$

$$\frac{p(\textbf{rec } x.p) \xrightarrow{a} p'}{\textbf{rec } x.p \xrightarrow{a} p'}$$

Table 1

On LTSs several equivalence relations are defined; among them we consider the *bisimulation*, the *simulation* and the *2/3 bisimulation* equivalences:

Definition 2.1: *A simulation R is a binary relation on $\mathcal{P}$ such that whenever pRq and $a \in \Sigma$ then: $p \xrightarrow{a} p' \Rightarrow \exists q'.\ q \xrightarrow{a} q'$ and p'Rq'.*
A process q is said to simulate a process p if and only if there exists a simulation R with pRq and we write p<q. Two processes are said simulation equivalent if p<q and q<p, written p <> q [Lar 85].

Definition 2.2: *A binary relation R on $\mathcal{P}$ is a bisimulation if and only if both R and R'= {(q, p) : (p, q)∈R} are simulations.*
Two processes p and q are said bisimulation (observational) equivalent if and only if there exists a bisimulation R with pRq, and we write $p \approx q$ [Par 81].

Definition 2.3: *A 2/3 bisimulation is a binary relation R on $\mathcal{P}$ such that whenever pRq and $a \in \Sigma$ then:*
i) $p \xrightarrow{a} p' \Rightarrow \exists q'.\ q \xrightarrow{a} q'$ *and* $p'Rq'$
ii) $q \xrightarrow{a} \Rightarrow p \xrightarrow{a}$.
Two processes are said 2/3 bisimulation equivalent if and only if there exists a 2/3 bisimulation R with pRq and a 2/3 bisimulation R' with qR'p, and we write $p \equiv q$ [Lar 88].

We will consider only "strong" equivalences because we don't take into account silent actions; in this case bisimulation is a congruence with respect to all the CCS operators. Hence from now on we will use *strong* (observational) *congruence* as a synonym for bisimulation equivalence.

3. BRANCHING TEMPORAL LOGIC

We choose a standard branching time logic, CTL* [Eme 86], with the addition of a maximal fixed point constructor, to give the branching temporal semantics to our language, since CTL* is adequate with respect to the strong congruence on CCS [Bro 88].
The atomic predicates specify events and are borrowed from the action set Σ. The formulae in this logic

are interpreted on the states of a model structure M which can be seen as a tree. A model M is a 4-tuple (S, s_0, R, L), where:

_ S is a set of states;

_ $s_0 \in S$;

_ R is a relation on S which defines the structure of the model (a tree); the properties of R are:

i) $\forall s \in S.\ (s, s) \notin R$ and $(s, s_0) \notin R$

ii) $\forall s, s' \in S.\ (s, s') \in R \Rightarrow (\forall s'' \in S - \{s\}.(s'', s') \notin R)$

iii) $\forall s \in S - \{s_0\}.\ \exists\ s' \in S.(s', s) \in R$;

_ L is a mapping, $L: S \rightarrow 2^{Prop}$, from S to the powerset of Prop (the set of atomic propositions, $Prop = \Sigma \cup \{\mathbf{true}, \mathbf{false}\}$). We have:

i) $\forall s \in S.$ **true** $\in L(s)$ and **false** $\notin L(s)$

ii) $\forall s \in S, \forall a, b \in \Sigma.$ if $a \neq b$ and $a \in L(s)$ then $b \notin L(s)$. (This means that only one action can be performed in a transition, and implies that $(a \wedge b) \equiv$ **false** if $a \neq b$).

Note that we can reach the state s' from s via the occurrence of an action a if and only if $(s, s') \in R$ and $a \in L(s)$ and that we allow both finite and infinite computation paths.

Table 2 shows the operators of CTL* used in the following to provide branching temporal semantics and what means for a state s in a model M to satisfy a formula ϕ, (denoted by $s \models_M \phi$).

Informally, EX ϕ means that *if exist next states* then *exists a next state* which satisfies ϕ. The formula EX! ϕ means that *exists a next state* which satisfies ϕ. Moreover, AX ϕ means that *if exist next states* then *all next states* satisfy ϕ; the formula AX! ϕ means that *exists a next state* and *all next states* satisfy ϕ. Also, we have that every state satisfies **true** and no state satisfies **false**.

Actually, the pair of strong next operators can be derived by the weak ones by duality, and vice versa. We have defined explicitly strong and weak next operators to emphasize that we deal with finite and infinite paths, that represent partial or complete computations; on the contrary, having only infinite paths implies that weak operators coincide with the corresponding strong ones.

Propositional calculus operators

$s \models_M p$	iff	$p \in L(s)$ $\quad (p \in Prop)$
$s \models_M \neg\phi$	iff	not $(s \models_M \phi)$
$s \models_M \phi \vee \psi$	iff	$(s \models_M \phi)$ or $(s \models_M \psi)$
$s \models_M \phi \wedge \psi$	iff	$(s \models_M \phi)$ and $(s \models_M \psi)$

Branching temporal operators

$s \models_M AX\,\phi$	iff	$R(s) = \{\}$ or for all $s' \in R(s)$: $s' \models_M \phi$
$s \models_M EX\,\phi$	iff	$R(s) = \{\}$ or exists $s' \in R(s)$: $s' \models_M \phi$
$s \models_M AX!\,\phi$	iff	$R(s) \neq \{\}$ and for all $s' \in R(s)$: $s' \models_M \phi$
$s \models_M EX!\,\phi$	iff	$R(s) \neq \{\}$ and exists $s' \in R(s)$: $s' \models_M \phi$
$s \models_M \nu x.\,\Phi(x)$	iff	$s \models_M \bigwedge_{n \geq 0} \Phi^n(\mathbf{true})$

Table 2.

Note that by Φ^k (**true**) we denote the temporal formula $\Phi(\Phi(...\Phi(\textbf{true})...))$ k-times. The maximal fixed point constructor $\nu x.\ \Phi(x)$ denotes the maximal solution to $x=\Phi(x)$, which exists if the functional $\Phi(x)$ is monotonic and continuous. The monotonicity requirement can be ensured by the appearance of the temporal formula x in Φ under an even number of negations [Ban 89] and continuity derives from the continuity of all logical operators.

4. ADEQUACY AND EXPRESSIVENESS

In order to define a temporal semantics for $\mathcal{AP}$ we need some preliminary definitions. As we are interested in correct logic reasoning on processes of a process language $\mathcal{P}$, with an associated operational semantics, in a particular (temporal) logic $\mathcal{T}$, we need to identify a process p ($p \in \mathcal{P}$) with the set of all properties (formulae in $\mathcal{T}$) which it satisfies.
This requires the definition of a binary relation $\models \subseteq \mathcal{P} \times \mathcal{T}$, usually called *satisfaction relation*, written: $p \models \phi$ and read "p satisfies the property ϕ". The definition of this relation induces an equivalence between processes which enjoy the same properties. Formally, we define:

$$\mathcal{F}(p) = \{\ \phi : \phi \in \mathcal{T} \wedge\ p \models \phi\}$$

hence

$$p \equiv_{\mathcal{T}} q \quad \text{iff} \quad \mathcal{F}(p) = \mathcal{F}(q)$$

Now, if $\approx\ \subseteq \mathcal{P} \times \mathcal{P}$ is the equivalence induced by the native operational semantics of the process language $\mathcal{P}$, it is possible to define a relation between $\approx$ and $\equiv_{\mathcal{T}}$:

A logic $\mathcal{T}$ is **adequate** [Hen 85] w.r.t. an equivalence ($\approx$) defined on a given process language $\mathcal{P}$, if for every pair of processes p and q:

$$p \equiv_{\mathcal{T}} q \quad \text{iff} \quad p \approx q$$

However we may require a stronger relationship between a process language and a logic: a way to characterize with a unique finite formula in $\mathcal{T}$ all the properties satisfied by a given process p ($p \in \mathcal{P}$); in the context of this paper this corresponds to define a *temporal semantics* [Pnu 81], i.e. a particular denotational semantics in which a mathematical function associates to each $p \in \mathcal{P}$ a temporal logic formula expressing the properties of its execution sequences.

When a denotational semantics is defined on a language provided with an operational semantics it is necessary to establish a link between them; as a first relation we can speak about *simple abstractness*: a semantics $\mathcal{D}$ is *simply abstract* with respect to the operational semantics defining $\approx$, if and only if:

$$\forall p, q \in \mathcal{P}.\quad \mathcal{D}[p] = \mathcal{D}[q] \quad \text{iff} \quad p \approx q$$

The notion of simple abstractness of a logic semantics w.r.t. a given operational semantics can be related to the logic by mean of the following definition:

A logic $\mathcal{T}$ is **expressive** [Pnu 85] w.r.t. an equivalence relation ($\approx$) defined on a given process language $\mathcal{P}$, if it is adequate and for every process p ($p \in \mathcal{P}$) there exists a *characteristic formula* (or *logic semantics*) $\mathcal{L}(p) \in \mathcal{T}$ such that:

$$\mathcal{L}(p) = \mathcal{L}(q) \quad \text{iff} \quad p \approx q.$$

We can therefore say that a logic $\mathcal{T}$ is expressive w.r.t. an equivalence relation $\approx$, defined on a given process language $\mathcal{P}$, iff it is possible to define a semantics $\mathcal{L}$ for $\mathcal{P}$ such that the denotation domain of $\mathcal{L}$ is $\mathcal{T}$ and $\mathcal{L}$ is simply abstract with respect to the operational semantics defining $\approx$.

In order to provide modularity in the specification and verification of concurrent programs, semantics should be compositional [DeR 85]. A semantics $\mathcal{F}$ is *compositional* with respect to an n-adic syntactic operator $\mathcal{O}p$ if and only $\exists$ f: $\mathcal{D}_{en}{}^{n} \rightarrow \mathcal{D}_{en}$. $\forall p_1, p_2, .., p_n \in \mathcal{P}$. $\mathcal{F}(\mathcal{O}p(p_1, p_2, .., p_n)) = f(\mathcal{F}(p_1), \mathcal{F}(p_2), .., \mathcal{F}(p_n))$, where $\mathcal{P}$ is the object language and $\mathcal{D}_{en}$ is the set representing the range of the semantic function $\mathcal{F}$ [Fis 87].

In this case a stronger link between the denotational semantics and the operational one can be defined: the denotational semantics should be *fully abstract* with respect to the operational semantics. Let $\mathcal{P}$ be a set of processes; a semantics $\mathcal{D}$ is *fully abstract* with respect to the operational semantics defining an equivalence $\approx$, if and only if it is simply abstract and compositional.

Also, we can define a stronger relation between a process language and a logic:
A logic $\mathcal{T}$ is **fully expressive** [Pnu 85] w.r.t. an equivalence relation ($\approx$) defined on a given process language $\mathcal{P}$, if it is adequate and for every process p ($p \in \mathcal{P}$) there exists a *compositional logic semantics* $\mathcal{L}(p) \in \mathcal{T}$ such that:

$$\mathcal{L}(p) = \mathcal{L}(q) \quad \text{iff} \quad p \approx q$$

As before we can state that a logic $\mathcal{T}$ is fully expressive w.r.t. an equivalence relation $\approx$, defined on a given process language $\mathcal{P}$, iff it is possible to define a semantics $\mathcal{L}$ for $\mathcal{P}$ such that the denotation domain of $\mathcal{L}$ is $\mathcal{T}$ and $\mathcal{L}$ is fully abstract with respect to the operational semantics defining $\approx$.

Note that if a semantics $\mathcal{L}$ for $\mathcal{P}$ is compositional and fully abstract (or simply abstract - in this case the two concepts coincide) with respect to the operational semantics defining the equivalence $\approx$, then $\approx$ is also a congruence on $\mathcal{P}$.

5. EXPRESSIVENESS OF CTL*

In this section we will give several results which show the expressiveness of the logic CTL*+ν with respect to different language operators and different equivalence relations defined on the process languages.
As a first observation, we recall that CTL*+ν is *adequate* with respect to the strong congruence. In [Bro 88] it is shown that CTL* is adequate with respect to the bisimulation equivalence; adequacy is not lost when enriching the logic with a new operator, in this case the maximal fixed point.
We can also note the strict correspondence between the models of the Logics and the Labelled Transitions Systems; they differ only because in the former labels are associated to states while in the latter they are associated to arcs. We can easily transform each other shifting the information from states to arcs or viceversa.

Theorem 1: *CTL*+ν is fully expressive for $\mathcal{AP}$ with respect to the strong congruence .*
Proof (by construction): We will give a denotational semantics to $\mathcal{AP}$ where the domain of denotations is CTL*+ν in order to produce a semantics fully abstract with respect to strong congruence.
The branching temporal semantics of $\mathcal{AP}$ is shown in Table 3. It is defined by means of the semantics

function $\mathcal{M}$ and auxiliary functions $\mathcal{S}$, $\mathcal{T}$.

$\mathcal{AP}$ semantics

$\mathcal{M}(p)$	$= \mathcal{S}(p) \wedge AX\mathcal{T}(p)$, for $p \neq$ **rec** x.p', $p \neq x$		
$\mathcal{M}$(**rec** x. p)	$= \nu x.\mathcal{M}(p)$		
$\mathcal{M}(x)$	$= x$		
$\mathcal{T}$(**nil**)	= **false**	$\mathcal{T}$(a. p)	$= a \wedge \mathcal{M}(p)$
$\mathcal{T}(p+q)$	$= \mathcal{T}(p) \vee \mathcal{T}(q)$	$\mathcal{T}$(rec x.p)	$= \mathcal{T}(p(\text{rec } x.p))$
$\mathcal{S}$(**nil**)	= **true**	$\mathcal{S}$(a. p)	$= EX!(a \wedge \mathcal{M}(p))$
$\mathcal{S}(p+q)$	$= \mathcal{S}(p) \wedge \mathcal{S}(q)$	$\mathcal{S}$(rec x.p)	$= \mathcal{S}(p(\text{rec } x.p))$

Table 3

Bisimulation is preserved by $\mathcal{M}$; actually, since we have a compositional constraint, which involves contexts, the proof of fully abstractness reduces to prove: *$\mathcal{M}(p)=\mathcal{M}(q)$ if and only if $p \approx q$*. For the if-direction of the proof we show that the $\mathcal{M}$ function respects the axioms of bisimulation on $\mathcal{AP}$ [Mil 89]. For the other implication we show the correspondence between the transitions of a process p and its $\mathcal{M}$-formula using inductive arguments. For the complete proof see the appendix. □

Example 1:

Consider the processes *rec x.(a. b. x+ a. **nil**)* and *rec x.(a. b. x)*:

$\mathcal{M}$(rec x. (a. b. x + a. **nil**)) $\equiv \nu x.\ (EX!(a \wedge AX!(b \wedge x)) \wedge EX!(a \wedge AX\textbf{false}) \wedge AX((a \wedge AX!(b \wedge x)) \vee (a \wedge AX\textbf{false})))$

$\mathcal{M}$(rec x. (a. b. x)) $\equiv \nu x.\ (AX!(a \wedge AX!(b \wedge x)))$

(In the formulae above we have applied the property $EX!\phi \wedge AX\phi \equiv AX!\phi$).

The two formulae are not equivalent and the two processes are not congruent.

Example 2:

Consider the processes *rec x. (a. a. x)* and *rec x. (a. x)*:

$\mathcal{M}$(rec x. (a. a. x)) $\equiv \nu x.\ EX!(a \wedge (EX!(a \wedge x) \wedge AX(a \wedge x))) \wedge AX(a \wedge (EX!(a \wedge x) \wedge AX(a \wedge x))) =$
$EX!(a \wedge (EX!(a) \wedge AX(a))) \wedge AX(a \wedge (EX!(a) \wedge AX(a))) \wedge \ldots\ldots = EX!(a) \wedge AX(a) \wedge EX!(a \wedge (EX!(a) \wedge AX(a))) \wedge AX(a \wedge (EX!(a) \wedge AX(a))) \wedge \ldots\ldots = \nu x.\ EX!(a \wedge x) \wedge AX(a \wedge x) = \mathcal{M}$(rec x. (a. x))

The two formulae are equivalent and the two processes are congruent.

As soon as we enrich the language with even the simplest parallel composition operator, such as the full synchronization operator, we have the general result that shows that any compositional temporal semantics given induces an equivalence weaker than bisimulation:

Theorem 2: *Let | be an associative operator such that the idempotence law, $p \mid p \approx p$, where $\approx$ is a defined equivalence on processes, does not hold. Then it is not possible to find a temporal semantics for the enriched language $\mathcal{AP}+\{|\}$ fully abstract w.r.t. $\approx$ making use of a compositional method and a standard logic as CTL*.*

Proof:

Let $\mathcal{F}_L$ a temporal semantics function for the terms of $\mathcal{AP}+\{|\}$, $\mathcal{F}_L$: $\mathcal{AP}+\{|\} \rightarrow$ CTL*.

In order to satisfy the compositionality requirement we must find a function γ such that $\mathcal{F}_{\mathcal{L}}(p \mid q) = \gamma(\mathcal{F}_{\mathcal{L}}(p), \mathcal{F}_{\mathcal{L}}(q))$.
Let $\gamma = \sigma \cdot \rho$, where:
- $\sigma \cdot \rho(x, y)$ stands for $\sigma(\rho(x, y))$
- σ is a composition of unary operators (X and $\neg$) and path quantifiers (A and E)
- the outermost operator of ρ is binary.

We show firstly that σ is an empty sequence or a sequence of quantifiers and *not* operators. To see that, let $\sigma = \sigma_1 \cdot X \cdot \sigma_2$; since $\mid$ is associative, that is $(p \mid q) \mid r \approx p \mid (q \mid r)$, we must have $\gamma(\gamma(x, y), z) = \gamma(x, \gamma(y, z))$. But such equality will be generally false because, in the left part, the argument x is prefixed by X one more time than in the right part (the converse holds for z). So, σ can only be an empty sequence or a sequence of quantifiers and *not* operators.
Now, let $\gamma = \sigma \cdot \rho$ such that $\rho(x, y) = op(\gamma_1(x, y), \gamma_2(x, y))$, where *op* is a binary operator, $op \in \{\mathcal{U}, \wedge, \vee\}$. We show that $op \neq \mathcal{U}$; in fact, if we still consider the associative law $(p \mid q) \mid r \approx p \mid (q \mid r)$, we must have $P \approx Q$, where:

$P = \sigma(\gamma_1(\gamma(x, y), z)\ \mathcal{U}\ \gamma_2(\gamma(x, y), z))$

$Q = \sigma(\gamma_1(x, \gamma(y, z))\ \mathcal{U}\ \gamma_2(x, \gamma(y, z)))$.

Since $\gamma(x, y) = \sigma(\gamma_1(x, y) \mathcal{U} \gamma_2(x, y))$ and $w_1 \mathcal{U} (w_2 \mathcal{U} w_3) \neq (w_1 \mathcal{U} w_2) \mathcal{U} w_3$, we have that generally $P \neq Q$.
The reasonings above apply inductively to γ_1 and γ_2.
Finally, we are reduced to ρ, and hence γ, being a composition of boolean operators only and path quantifiers (Note that γ must be a composition of $\wedge$, $\vee$, $\neg$ operators because a state formula cannot be quantified again); in this case we get immediately that $\gamma(x, x) \in \{x, \neg x, \mathbf{true}, \mathbf{false}\}$; if we examine the first possibility, i.e. $\gamma(x, x) = x$, we can see that this corresponds to state $p \mid p \approx p$, so we have a contradiction. For the other possibilities, they are clearly not acceptable generally as a valid meaning for $p \mid p$. □

The last part of the proof clearly shows that problems with bisimulation arise when we consider $\mathcal{AP}_i$ and $\mathcal{AP}_S$:

Theorem 3: *It is not possible to find a temporal semantics for $\mathcal{AP}_i$, $\mathcal{AP}_S$ and CCS making use of a compositional method and a standard logic as CTL*, with atomic action predicates, without losing the full abstractness with respect to bisimulation semantics.*

Proof: Straightforward by applying Theorem 2 to the parallel composition operators present in the relevant language. □

Actually, this theorem applies also to 2/3 bisimulation semantics, for which the idempotence law does not hold for the composition operators of $\mathcal{AP}_i$, $\mathcal{AP}_S$ and CCS.
Let us however try to give a compositional temporal semantics, using CTL*, to $\mathcal{AP}_S$, in Table 4.

$\mathcal{AP}_S$ *semantics*

$\mathcal{M}'(p)$	$= AX\,\mathcal{B}(p)$, for $p \neq$ **rec** x.p'(x), $p \neq x$
$\mathcal{M}'$(**rec** x. p(x))	$= \nu x.\mathcal{M}'(p(x))$
$\mathcal{M}'(x)$	$= x$
$\mathcal{B}$(**nil**)	$=$ **false**
$\mathcal{B}(a.\ p)$	$= a \wedge \mathcal{M}'(p)$
$\mathcal{B}(p+q)$	$= \mathcal{B}(p) \vee \mathcal{B}(q)$
$\mathcal{B}$(**rec** x. p(x))	$= \mathcal{B}(p(\text{rec x. } p(x)))$
$\mathcal{B}(p \mid q)$	$= \mathcal{B}(p) \wedge \mathcal{B}(q)$

Table 4

The following example shows that this semantics is indeed not consistent with strong congruence:

Example 3: For the processes *a. b.* ***nil*** + *a.* ***nil*** and *a. b. nil* we have:
$\mathcal{M}'$(a. b. **nil** + a. **nil**) ≡ AX((a ∧ AX(b ∧ AX**false**)) ∨ (a ∧ AX**false**))
$\mathcal{M}'$(a. b. **nil**) ≡ AX(a ∧ AX(b ∧ AX**false**))
Since ∀ϕ. AXϕ ∨ AX **false** ≡ AXϕ, we have that the two formulae are equivalent.

As expected, bisimulation is not preserved by $\mathcal{M}'$ in this example; actually, we can give the following result:

Theorem 4 : *The semantics $\mathcal{M}'$ for $\mathcal{AP}_S$ is fully abstract with respect to simulation semantics. As a consequence, CTL* is expressive w.r.t. simulation equivalence on $\mathcal{AP}_S$.*

Proof sketch: Again, for compositionality, it is enough to prove $\mathcal{M}'(p) = \mathcal{M}'(q)$ *if and only if* $p <> q$. We prove that a process p is simulated by a process q if and only if $\mathcal{M}'(p)$ *implies* $\mathcal{M}'(q)$, showing the correspondence between transitions and logic formulae. For the complete proof see the appendix. Note that Theorem 2 does not apply in this case since p || p <> p holds for simulation equivalence.

Since in the hierarchy of branching equivalences [Gro 88] the 2/3 bisimulation equivalence is immediately above simulation equivalence, *the simulation equivalence is the strongest equivalence that we can characterize with standard logic if we consider synchronization operators on processes, maintaining the compositionality constraint.*

6. CONCLUSIONS

The temporal semantics approach presented can be considered inside a research context in which several attempts to provide expressive logics for CCS-like process languages have been developed [Hen 85, Pnu 85, Gra 86]. Our approach differs from previous attempts because it considers together the compositionality constraint, the use of a logic with standard operators and the application to a language with recursion and parallel composition operators. At least one of this aspects is missing in each of the referred approach. Another characteristic of our approach is that we consider a "bounded" temporal logic and consequently we can characterize both equivalences that cover infinite computation properties (liveness) as maximal trace equivalence and bisimulation equivalence, and equivalences that cover partial properties as trace equivalence and simulation equivalence.
We have shown the impossibility, even when considering simple forms of parallel composition (interleaving, full synchronization), of giving a temporal semantics fully abstract w.r.t. strong congruence inside standard logics as CTL*, maintaining the compositionality constraint. We consider more interesting to work inside standard logics, in order to use tools already developed for them, rather than using Hennessy-Milner Logic with recursion [Lar 88a], or μ-calculus [Koz 83], that however do not add expressive power.
The result can be extended to all those action prefix languages which have a parallel composition operator: as an example, in [Fan 90] we have considered the same result for what concern the LOTOS specification language.
Within CTL*+V we have instead succeeded in defining a compositional temporal semantics for a subset of CCS, fully abstract with respect to a weaker equivalence, simulation, which still preserves the branching structure of the processes.

REFERENCES

[Ban 89] B. Banieqbal, H. Barringer: "Temporal Logic Fixed Point Calculus", Proceedings Colloquium on Temporal Logic and Specification, Altrincham, England, 1987. Lecture Notes in Computer Science, vol.398, pp. 62-74, 1989.

[Bar 84] H. Barringer, R. Kuiper, A.Pnueli: "Now you may Compose Temporal Logic Specifications", Proc. 16th ACM Symposium on the Theory of Computing, 1984, pp. 51-63.

[Bar 87] H. Barringer, "Using Temporal Logic in the Compositional Specifications of Concurrent Systems", in A. Galton ed., Temporal Logics and their Applications, Academic Press, 1987, pag.53-90.

[Bro 88] M. C. Browne, E. M. Clarke, O. Grümberg, "Characterizing Finite Kripke Structures in Propositional Temporal Logic", Theoretical Computer Science, vol. 59, pp. 115-131, 1988.

[DeR 85] W. P. De Roever, "The quest of compositionality - a survey of assertion-based proof systems for concurrent programs. Part 1: concurrency based on shared variables", in E. J. Neuhold and G. Chroust (eds) Formal Models of Programming, 1985, North-Holland: Elsevier, pp.181-205

[Eme 86] E. A. Emerson, J. Halpern, ""Sometime" and "Not Never" Revisited: On Branching versus Linear Time Temporal Logic", Journal of ACM, vol. 33, January 1986, pp. 151-178.

[Fan 90] A. Fantechi, S. Gnesi, G. Ristori, " Compositional Logic Semantics and LOTOS", Proc. of the 10th Intl. IFIP WG 6.1 Symposium on Protocol Specification, Testing and Verification, Ottawa, June 1990, North-Holland.

[Fis 87] M.D. Fisher, "Temporal Logics for Abstract Semantics", University of Manchester, Technical Report UMCS-87-12-4, December 1987.

[Gra 86] S. Graf, J. Sifakis, "A Logic for the Description of Non-deterministic Programs and Their Properties", Information and Control vol.68, 1986, pp.254-270.

[Gro 88] J.F. Groote, F.W. Vaandrager, "Structured Operational Semantics and Bisimulation as a Congruence", CWI, Amsterdam, Technical Report CS-R8845, November 1988.

[Hen 85] M. Hennessy, R. Milner, "Algebraic Laws for Nondeterminism and Concurrency", Journal of ACM, vol. 32, n. 1, January 1985, pp. 137-161.

[Lar 88a] K. Larsen, "Proof Systems for Hennessy-Milner Logic with Recursion ", Proc. CAAP'88, Lecture Notes in Computer Science, vol. 299.

[Koz 83] D. Kozen, "Results on the Propositional μ-calculus", Theoretical Computer Science, vol. 27, 1983.

[Mil 80] R. Milner, "A Calculus of Communicating Systems", Lecture Notes in Computer Science vol. 92, 1980.

[Mil 89] R. Milner, "A Complete Axiomatisation for Observational Congruence of Finite-state Behaviours", Inf. Comput., 81 (1989) 222-247.

[Mol 89] F. Moller, "The nonexistence of Finite Axiomatizations for CCS Congruences", Proceedings Logic in Computer Science 1990, Philadephia, PA, IEEE, June 1990.

[Pnu 81] A. Pnueli, "The Temporal Semantics of Concurrent Programs", Theoretical Computer Science, vol.13, 1981, pp.45-60.

[Pnu 85] A. Pnueli, "Linear and Branching Structures in the Semantics and Logic of Reactive Systems" Proceedings of 12th ICALP, Lecture Notes in Computer Science vol. 194, July 1985.

[Sti 88] C. Stirling, "Expressibility and Definability in Branching and Linear Time Temporal Logic", REX School / Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Noordwijkerhout, the Netherlands, May-June 1988.

APPENDIX

a) Proof of Theorem 1

Lemma 1.a: *$p \not\xrightarrow{a} \forall a$ iff $\mathcal{S}(p) \equiv$ true.*

Proof (by structural induction):

(only if)

i) p = **nil**

nil $\not\xrightarrow{a} \forall a$ and $\mathcal{S}$(**nil**) = **true**.

ii) p = p' + p"

p' + p" $\not\xrightarrow{a} \forall a$ implies p' $\not\xrightarrow{a} \forall a$ and p" $\not\xrightarrow{a} \forall a$; we have, for the induction hypothesis: $\mathcal{S}(p') \equiv$ **true** and $\mathcal{S}(p") \equiv$ **true** and then $\mathcal{S}(p'+ p") = \mathcal{S}(p') \wedge \mathcal{S}(p") \equiv$ **true**.

iii) p = rec x.q(x)

rec x.q(x) $\not\xrightarrow{a} \forall a$ implies q(rec x.q(x)) $\not\xrightarrow{a} \forall a$; thus, for the induction hypothesis, $\mathcal{S}$(q(recx.q(x))) $\equiv$ **true** and then $\mathcal{S}$(rec x.q(x)) = $\mathcal{S}$(q(rec x.q(x))) $\equiv$**true**.

(if)

i) $\mathcal{S}$(**nil**) =**true** and **nil** $\not\xrightarrow{a} \forall a$

ii) $\mathcal{S}(p'+ p") = \mathcal{S}(p') \wedge \mathcal{S}(p") \equiv$ **true**. Therefore we have $\mathcal{S}(p') \equiv$ **true** and $\mathcal{S}(p") \equiv$ **true** and for the induction hypothesis p' $\not\xrightarrow{a} \forall a$ and p" $\not\xrightarrow{a} \forall a$; so p'+ p" $\not\xrightarrow{a} \forall a$.

iii) $\mathcal{S}$(rec x.p(x)) $\equiv$ **true**.

Since $\mathcal{S}$(p(rec x.p(x))) = $\mathcal{S}$(rec x.p(x)) we have $\mathcal{S}$(p(rec x.p(x))) $\equiv$ **true** and for the induction hypothesis p(rec x.p(x)) $\not\xrightarrow{a} \forall a$; so rec x.p(x) $\not\xrightarrow{a} \forall a$.

Lemma 1.b: *$p \not\xrightarrow{a} \forall a$ iff $\mathcal{T}(p) \equiv$ false.*

Proof: is analogous to that of Lemma 1.a.

Lemma 2.a: *$p \xrightarrow{a} p'$ iff $\mathcal{S}(p) = EX!(a \wedge \mathcal{M}(p)) \wedge s$.*

Proof (by structural induction):

(only if)

i) p = a. p'

a. p' $\xrightarrow{a}$ p' and $\mathcal{S}$(a. p') = EX!(a $\wedge$ $\mathcal{M}$(p'))

ii) $p = p_1 + p_2$

If $p_1 + p_2 \xrightarrow{a} p'$ then $p_1 \xrightarrow{a} p'$ or $p_2 \xrightarrow{a} p'$; because the induction hypothesis we have:

$\mathcal{S}(p_1) = EX!(a \wedge \mathcal{M}(p')) \wedge s'$

or

$\mathcal{S}(p_2) = EX!(a \wedge \mathcal{M}(p')) \wedge s"$.

Therefore $\mathcal{S}(p_1 + p_2) = \mathcal{S}(p_1) \wedge \mathcal{S}(p_2) = EX!(a \wedge \mathcal{M}(p')) \wedge s$.

iii) p = rec x.q(x)

We have rec x.q(x) $\xrightarrow{a}$ p' iff q(rec x.q(x)) $\xrightarrow{a}$ p'. Then $\mathcal{S}$(q(rec x.q(x))) = EX!(a $\wedge$ $\mathcal{M}$(p'))$\wedge$ s and $\mathcal{S}$(rec x.q(x)) = $\mathcal{S}$(q(rec x.q(x))) = EX!(a $\wedge$ $\mathcal{M}$(p')) $\wedge$ s.

(if)

i) p = a. p'

We have a. p' $\xrightarrow{a}$ p' and $\mathcal{S}$(a. p') = EX!(a $\wedge$ $\mathcal{M}$(p')).

ii) $p = p_1 + p_2$;

$\mathcal{S}(p_1 + p_2) = \mathcal{S}(p_1) \wedge \mathcal{S}(p_2) = EX!(a \wedge \mathcal{M}(p')) \wedge s$; then $\mathcal{S}(p_1) = EX!(a \wedge \mathcal{M}(p')) \wedge s_1$ or $\mathcal{S}(p_2) = EX!(a \wedge \mathcal{M}(p')) \wedge s_2$.

For the induction hypothesis we have $p_1 \xrightarrow{a} p'$ or $p_2 \xrightarrow{a} p'$ so $p_1 + p_2 \xrightarrow{a} p'$.

iii) p = rec x.q(x);
$\mathcal{S}$(rec x.q(x)) = $\mathcal{S}$(q(rec x.q(x))) = EX!(a $\wedge$ $\mathcal{M}$(p')) $\wedge$ s; for the induction hypothesis we have q(rec x.q(x))—a→p' and so rec x.q(x)—a→p'.

Lemma 2.b: *p—a→p' iff* $\mathcal{T}(p) = (a \wedge \mathcal{M}(p')) \vee t$.
Proof: is analogous to that of Lemma 2.a.

Lemma 3.a: *If p—a_i→p_i (i = 1, .., n) are the (only) transitions of p then:*

$$\mathcal{S}(p) \equiv \wedge_{i=1..n}\ EX!(a_i \wedge \mathcal{M}(p_i)) .$$

Proof (by contraddiction):
From the Lemma 2.a. we have $\mathcal{S}$(p) = $\wedge_{i=1..n}$ EX!(a_i $\wedge$ $\mathcal{M}$(p_i)) $\wedge$ s. If the thesis not hold then we have s = EX! (b $\wedge$ $\mathcal{M}$(q)) $\wedge$ s' and for all i=1, .., n $a_i \neq b$ or $\mathcal{M}(p_i) \neq \mathcal{M}(q)$. But for the Lemma 2.a. p—b→q so we have a contraddiction.

Lemma 3.b: *If p—a_i→p_i (i = 1, .., n) are the (only) transitions of p then:*

$$\mathcal{T}(p) \equiv \vee_{i=1..n}(a_i \wedge \mathcal{M}(p_i))\ ".$$

Proof: is analogous to that of Lemma 3.a.

Lemma 4.a: *If* $\mathcal{S}(p) \equiv \wedge_{i=1..n} EX!(a_i \wedge \mathcal{M}(p_i))$ *then* $\exists\, q_1, \ldots, q_m$ *such that:*

i) $\forall i \in \{1,..,n\} \exists j \in \{1,..,m\}$: *p—$a_i$→$q_j$ and* $\mathcal{M}(p_i) \equiv \mathcal{M}(q_j)$

ii) $\forall j \in \{1,..,m\} \exists i \in \{1,..,n\}$: *p—$a_i$→$q_j$ and* $\mathcal{M}(p_i) \equiv \mathcal{M}(q_j)$.

Proof:
Since $\mathcal{S}$(p)≠**true** we have, for the Lemma 2.a, that $\exists$ q_1, .., q_m, b_1, ..,b_m such that the transitions of p are p—b_j→q_j, (j = 1, .., m) and then, for the Prop 3.a., we have $\mathcal{S}$(p) ≡ $\wedge_{j=1..m}$ EX!(b_j $\wedge$ $\mathcal{M}$(q_j)). Hence $\wedge_{i=1..n}$ EX!(a_i $\wedge$ $\mathcal{M}$(p_i)) ≡ $\wedge_{j=1..m}$ EX!(b_j $\wedge$ $\mathcal{M}$(q_j)) and this implies the thesis.

Lemma 4.b: *If* $\mathcal{T}(p) \equiv \vee_{i=1..n}(a_i \wedge \mathcal{M}(p_i))$ *then* $\exists\, q_1, \ldots, q_m$ *such that:*

i) $\forall i \in \{1,..,n\} \exists j \in \{1,..,m\}$: *p—$a_i$→$q_j$ and* $\mathcal{M}(pi) \equiv \mathcal{M}(qj)$

ii) $\forall j \in \{1,..,m\} \exists i \in \{1,..,n\}$: *p—$a_i$→$q_j$ and* $\mathcal{M}(pi) \equiv \mathcal{M}(qj)$.

Proof: analogous to that of Lemma 4.a.

Lemma 5: *p—a/→*$\forall a$ *iff* $\mathcal{M}(p) \equiv AX$ ***false***"
Proof: immediate from the definition of $\mathcal{M}$(p) and from Lemmas 1.a, 1.b.

Lemma 6: $\mathcal{M}(p) \neq AX$ ***false*** *implies* $\exists p_1, \ldots, p_m, a_1, .., a_m$:

$$\mathcal{M}(p) \equiv \wedge_{i=1..m} EX!(a_i \wedge \mathcal{M}(p_i)) \wedge AX(\vee_{i=1..m}(a_i \wedge \mathcal{M}(p_i))).$$

Proof: immediate from the definition of $\mathcal{M}$(p) and from Lemma 3.a, 3.b, 5.

Lemma (*): $\mathcal{M}(p) \equiv \mathcal{M}(q)$ *implies* $p \approx q$.
Proof: (by induction on formulae)
For the Lemma 6 we have only two cases:
i) $\mathcal{M}$(p) ≡ $\mathcal{M}$(q) ≡ AX **false**
For the Prop 5. we have p—a/→$\forall$a and q—a/→$\forall$a so p ≈ q.

ii) $\mathcal{M}(p) \equiv \mathcal{M}(q) \equiv \wedge_{i=1..m} EX!(a_i \wedge \mathcal{M}(p_i)) \wedge AX(\vee_{i=1..m} (a_i \wedge \mathcal{M}(p_i)))$

Let $p \text{—} b \rightarrow p'$; then for the Prop 3. $\exists$ i$\in$ {i, ..,m} such that a_i=b and $\mathcal{M}(p') \equiv \mathcal{M}(p_i)$, and, for the Prop 4., $\exists$ q' such that $q \text{—} b \rightarrow q'$ and $\mathcal{M}(q') \equiv \mathcal{M}(p_i)$; thus we have $\mathcal{M}(p') \equiv \mathcal{M}(q')$ and, for the induction hypothesis, $p' \approx q'$. We can repeat this reasoning for any transition of p and q and so we have $p \approx q$.

Lemma (**): *$p \approx q$ implies $\mathcal{M}(p) \equiv \mathcal{M}(q)$.*

Proof: we will apply the semantic functions $\mathcal{M}$, $\mathcal{S}$ and $\mathcal{T}$ to left and right parts of bisimulation's equational axioms [Hen 85] and we will show that functions $\mathcal{M}$, $\mathcal{S}$ and $\mathcal{T}$ preserve the equalities.

i) p + (q + r) = (p + q) + r

We have:

$\mathcal{M}(p + (q + r)) = \mathcal{S}(p+(q+ r)) \wedge AX(\mathcal{T}(p+ (q+r))) = \mathcal{S}(p) \wedge \mathcal{S}(q+ r) \wedge AX(\mathcal{T}(p) \vee \mathcal{T}(q+ r)) = \mathcal{S}(p) \wedge \mathcal{S}(q) \wedge \mathcal{S}(r) \wedge AX(\mathcal{T}(p) \vee \mathcal{T}(q) \vee \mathcal{T}(r)) \equiv \mathcal{S}(p+ q) \wedge \mathcal{S}(r) \wedge AX(\mathcal{T}(p+ q) \vee \mathcal{T}(r)) \equiv \mathcal{S}((p+q)+ r) \wedge AX(\mathcal{T}((p+ q)+r)) = \mathcal{M}((p+q)+r)$.

ii) p+ q = q+ p

$\mathcal{M}(p+ q) = \mathcal{S}(p+ q) \wedge AX(\mathcal{T}(p+ q)) = \mathcal{S}(p) \wedge \mathcal{S}(q) \wedge AX(\mathcal{T}(p) \vee \mathcal{T}(q)) \equiv \mathcal{S}(q) \wedge \mathcal{S}(p) \wedge AX(\mathcal{T}(q) \vee \mathcal{T}(p)) = \mathcal{S}(q+ p) \wedge AX(\mathcal{T}(q+ p)) = \mathcal{M}(q+ p)$.

iii) p + **nil** = p

$\mathcal{M}(p+ \textbf{nil}) = \mathcal{S}(p) \wedge \mathcal{S}(\textbf{nil}) \wedge AX(\mathcal{T}(p) \vee \mathcal{T}(\textbf{nil})) = \mathcal{S}(p) \wedge \textbf{true} \wedge AX(\mathcal{T}(p) \vee \underline{\text{false}})) \equiv \mathcal{S}(p) \wedge AX(\mathcal{T}(p)) = M(p)$.

iv) rec x.p(x) = p(rec x.p(x))

$\mathcal{M}(\text{rec x.p(x)}) = \nu x.\, \mathcal{M}(p(x)) = \nu x.\, \mathcal{S}(p(x)) \wedge AX(\mathcal{T}(p(x))) \equiv \mathcal{S}(p(\text{rec x.p(x)})) \wedge AX(\mathcal{T}(p(\text{rec x.p(x)}))) = \mathcal{M}(p(\text{rec x.p(x)}))$.

Theorem 1: *The semantics $\mathcal{M}$ for AP is fully abstract with respect to bisimulation semantics, i.e.$\mathcal{M}(p) \equiv \mathcal{M}(q)$ if and only if $p \approx q$.*

Proof: follow by Lemmas (*) and (**).

b) Proof of Theorem 2

Lemma 7: *$p \text{—} a_i \rightarrow q_i$ (i=1, ..,n) implies $\mathcal{B}(p) \equiv \vee_{i=1,..,n} (a_i \wedge \mathcal{M}'(q_i))$.*

Proof: (by structural induction)

i) a; $q \text{—} a \rightarrow q$ and $\mathcal{B}(a.\, q) = a \wedge \mathcal{M}'(q)$

ii) $p_1 + p_2 \text{—} a_i \rightarrow q_i$ i$\in$ I={1,..,n};

then $\exists$ I1,I2 such that I1 $\cup$ I2 = I and $\forall i_1 \in$ I1, $\forall i_2 \in$ I2 . $p_1 \text{—} a_{i_1} \rightarrow q_{i_1}$ and $p_2 \text{—} a_{i_2} \rightarrow q_{i_2}$.

For the induction hypothesis we have:

$\mathcal{B}(p_1) \equiv \vee_{i \in I1} (ai \wedge \mathcal{M}'(qi))$ and $\mathcal{B}(p_2) \equiv \vee_{i \in I2} (ai \wedge \mathcal{M}'(qi))$: therefore $\mathcal{B}(p_1 + p_2) = \mathcal{B}(p_1) \vee \mathcal{B}(p_2) \equiv \vee_{i \in I1} (ai \wedge \mathcal{M}'(qi)) \vee \vee_{i \in I2} (ai \wedge \mathcal{M}'(qi)) \equiv \vee_{i=1,..,n} (a_i \wedge \mathcal{M}'(q_i))$.

iii) $p_1 \,\|\, p_2 \text{—} a_i \rightarrow q_i$ i$\in$ I={1,..,n};

we have then:

$\exists$ $b_1, ..,b_m, p_{11}, ..,p_{1m}, c_1, ..,c_t, p_{21}, ..,p_{2t}$, (m, t $\geq$ n) such that :

*) $p_1 \text{—} b_j \rightarrow p_{1j}$ (j=1, ..,m)

*) $p_2 \text{—} c_k \rightarrow p_{2k}$ (k=1, ..,t)

*) $\{b_1, ..,b_m\} \cap \{c_1, ..,c_t\} = \{a_1, ..,a_n\}$

*) $\forall i \exists j, k : a_i = b_j = c_k$ and $q_i = p_{1j} \,\|\, p_{2k}$.

For the induction hypothesis we have:

$\mathfrak{B}(p_1) \equiv \vee_{j=1,..,m} (b_j \wedge \mathfrak{M}'(p_{1j}))$

$\mathfrak{B}(p_2) \equiv \vee_{k=1,..,t} (c_k \wedge \mathfrak{M}'(p_{2k}))$

and then:

$\mathfrak{B}(p_1 \| p_2) = \mathfrak{B}(p_1) \wedge \mathfrak{B}(p_2) \equiv (\vee_{j=1,..,m} (b_j \wedge \mathfrak{M}'(p_{1j}))) \wedge (\vee_{k=1,..,t} (c_k \wedge \mathfrak{M}'(p_{2k})))$.

Since for any model $(a \wedge b) \equiv$ <u>false</u> if $a \neq b$, and $\mathfrak{M}'(p) \wedge \mathfrak{M}'(q) \equiv \mathfrak{M}'(p \| q)$, we have:

$(\vee_{j=1,..,m} (b_j \wedge \mathfrak{M}'(p_{1j}))) \wedge (\vee_{k=1,..,t} (c_k \wedge \mathfrak{M}'(p_{2k}))) \equiv \vee_{b_j=c_k} (b_j \wedge \mathfrak{M}'(p_{1j}) \wedge \mathfrak{M}'(p_{2k})) \equiv \vee_{i=1,..,n} (a_i \wedge \mathfrak{M}'(q_i))$.

iv) rec x.p(x)—a_i→q_i i∈ I={1,..,n};

then we have p(rec x.p(x))—a_i→q_i i∈ I={1,..,n}and for the induction hypothesis $\mathfrak{B}$(p(rec x.p(x))) ≡ $\vee_{i=1,..,n} (a_i \wedge \mathfrak{M}'(q_i))$; so $\mathfrak{B}$(rec x.p(x)) = $\mathfrak{B}$(p(rec x.p(x))) ≡ $\vee_{i=1,..,n} (a_i \wedge \mathfrak{M}'(q_i))$.

Lemma 8: *p—a_i/→ ∀ a iff $\mathfrak{B}(p) \equiv$ false.*

<u>Proof</u>: (by structural induction)

(only if)

i) **nil**—a/→ ∀a and $\mathfrak{B}$(**nil**) = **false**.

ii) $p_1 + p_2$ —a/→ ∀a;

then we have p_i—a/→ ∀a (i=1, 2) and for the induction hypothesis $\mathfrak{B}(p_i) \equiv$ **false**; therefore $B(p_1 + p_2) = B(p_1) \vee B(p_2) \equiv$ **false** ∨ **false** ≡ **false**.

iii) $p_1 \| p_2$ —a/→ ∀a;

there are two cases:

*) p_i —a/→ ∀a , for i=1 or i=2; then $B(p_i) \equiv$ **false** and so $\mathfrak{B}(p_1 \| p_2) = \mathfrak{B}(p_1) \wedge \mathfrak{B}(p_2) \equiv$ **false** ∧ $\mathfrak{B}(p_j) \equiv$ **false**.

*) p_1—a_i→p_{1i} (i=1, ..,n) and p_2—b_j→p_{2j} (j=1, ..,m); since $p_1 \| p_2$ —a/→ ∀a we have $a_i \neq b_j$ ∀i, j and so, for the Lemma 7, $\mathfrak{B}(p_1) \equiv \vee_i (a_i \wedge \mathfrak{M}'(q_i))$ and $\mathfrak{B}(p_2) \equiv \vee_j (b_j \wedge \mathfrak{M}'(p_{2j}))$. Then $(\vee_i (a_i \wedge \mathfrak{M}'(q_i))) \wedge (\vee_j (b_j \wedge \mathfrak{M}'(p_{2j}))) \equiv$ **false** and this implies $a_i \neq b_j$ ∀i, j.

iv) rec x.p(x)—a/→ ∀a; therefore p(rec x.p(x))—a/→ ∀a and for the induction hypothesis $\mathfrak{B}$(p(rec x.p(x))) ≡ **false**.

(if)

i) $\mathfrak{B}$(**nil**) = **false** and **nil**—a/→ ∀a.

ii) Let $\mathfrak{B}(p_1 + p_2) = \mathfrak{B}(p_1) \vee \mathfrak{B}(p_2) \equiv$ **false**. Then $\mathfrak{B}(p_1) \equiv$ **false** and $\mathfrak{B}(p_2) \equiv$ false so for the induction hypothesis p_1—a/→ ∀a and p_2—a/→ ∀a; therefore $p_1 + p_2$—a/→ ∀a.

iii) Let $\mathfrak{B}(p_1 \| p_2) = \mathfrak{B}(p_1) \wedge \mathfrak{B}(p_2) \equiv$ **false**. Then we have two cases:

*) $\mathfrak{B}(p_i) \equiv$ **false** for i=1 or i=2; in this case, for the induction hypothesis, p_i—a/→ ∀a and so $p_1 \| p_2$—a/→ ∀a.

*) $\mathfrak{B}(p_i) \neq$ **false**, i=1, 2. From the implication (⇒) we argue that ∃a_1, ..,a_n, p_{11}, ..,p_{1n}, b_1, .., b_m, p_{21}, .., p_{2m} such that p_1—a_i→p_{1i} (i=1, ..,n) and p_2—b_j→p_{2j} (j=1, ..,m); then, from Lemma 7, we have $\mathfrak{B}(p_1) \equiv (\vee_i (a_i \wedge \mathfrak{M}'(p_{1i})))$ and $\mathfrak{B}(p_2) \equiv \vee_j (b_j \wedge \mathfrak{M}'(p_{2j}))$ and so: $\mathfrak{B}(p_1 \| p_2) = \mathfrak{B}(p_1) \wedge \mathfrak{B}(p_2) \equiv (\vee_i (a_i \wedge \mathfrak{M}'(p_{1i}))) \wedge (\vee_j (b_j \wedge \mathfrak{M}'(p_{2j}))) \equiv$ **false** ; this implies $a_i \neq b_j$ ∀i,j and then $p_1 \| p_2$—a/→ ∀a.

iv) Let $\mathfrak{B}$(rec x.p(x)) = $\mathfrak{B}$(p(rec x.p(x))) ≡ **false** ; we have then p(rec x.p(x))—a/→ ∀a and so rec x.p(x)—a/→ ∀a.

Lemma 9: *If* $(a \wedge \mathcal{M}'(p)) \Rightarrow \mathcal{B}(q)$ *then* $\exists q'$ *such that* $q \text{—}a\rightarrow q'$ *and* $\mathcal{M}'(p) \Rightarrow \mathcal{M}'(q')$.

Proof: (by structural induction)

i) $q = a.\ q' \text{—}a\rightarrow q'$

since $\mathcal{B}(a.\ q') = a \wedge \mathcal{M}'(q')$ and $a \wedge \mathcal{M}'(p) \Rightarrow \mathcal{B}(q)$ we have $\mathcal{M}'(p) \Rightarrow \mathcal{M}'(q')$.

ii) $q = p_1 + p_2$

From the hypothesis we have $a \wedge \mathcal{M}(p) \Rightarrow \mathcal{B}(p_1 + p_2) = \mathcal{B}(p_1) \vee \mathcal{B}(p_2)$.

This implies $a \wedge \mathcal{M}'(p) \Rightarrow \mathcal{B}(p_1)$ or $a \wedge \mathcal{M}'(p) \Rightarrow \mathcal{B}(p_2)$; for the induction hypothesis we have $\exists q'$ such that $p_1 \text{—}a\rightarrow q'$ and $\mathcal{M}'(p) \Rightarrow \mathcal{M}'(q')$ or $\exists q'$ such that $p_2 \text{—}a\rightarrow q'$ and $\mathcal{M}'(p) \Rightarrow \mathcal{M}'(q')$; therefore $p_1 + p_2 \text{—}a\rightarrow q'$ and $\mathcal{M}'(p) \Rightarrow \mathcal{M}'(q')$.

iii) $q = p_1 \parallel p_2$

From the hypothesis $(a \wedge \mathcal{M}'(p)) \Rightarrow \mathcal{B}(p_1 \parallel p_2) = \mathcal{B}(p_1) \wedge \mathcal{B}(p_2)$, so $(a \wedge \mathcal{M}'(p)) \Rightarrow \mathcal{B}(p_1)$ and $(a \wedge \mathcal{M}'(p)) \Rightarrow \mathcal{B}(p_2)$; if we apply the induction hypothesis we have:

$\exists q''$ such that $p_1 \text{—}a\rightarrow q''$ and $\mathcal{M}'(p) \Rightarrow \mathcal{M}'(q'')$

and

$\exists q'''$ such that $p_2 \text{—}a\rightarrow q'''$ and $\mathcal{M}'(p) \Rightarrow \mathcal{M}'(q''')$.

Then $p_1 \parallel p_2 \text{—}a\rightarrow q'' \parallel q'''$ and $\mathcal{M}'(p) \Rightarrow \mathcal{M}'(q'') \wedge \mathcal{M}'(q''') \equiv \mathcal{M}'(q'' \parallel q''')$.

iv) $q = \text{rec } x.p'(x)$

From the hypothesis $(a \wedge \mathcal{M}'(p)) \Rightarrow \mathcal{B}(\text{rec } x.p'(x)) = \mathcal{B}(p'(\text{rec } x.p'(x)))$; then we have, from the induction hypothesis, that $\exists q'$ such that $p'(\text{rec } x.p'(x)) \text{—}a\rightarrow q'$ and $\mathcal{M}'(p) \Rightarrow \mathcal{M}'(q')$; this implies $\text{rec } x.p'(x) \text{—}a\rightarrow q'$ and $\mathcal{M}'(p) \Rightarrow \mathcal{M}'(q')$.

Lemma 10: $(\mathcal{B}(p) \Rightarrow \mathcal{B}(q)) \Leftrightarrow (\mathcal{M}'(p) \Rightarrow \mathcal{M}'(q))$.

Proof:

(only if)

From the definition of $\mathcal{M}'$ we have $\mathcal{M}'(p) = AX(\mathcal{B}(p)) \equiv AX\ \mathbf{false} \vee AX!(\mathcal{B}(p))$

Since $(f \Rightarrow g) \Rightarrow (AX!f \Rightarrow AX!g)$

and

$AXf \Leftrightarrow AX\ \mathbf{false} \vee AX!f$

and

$(b \Rightarrow c) \Rightarrow (b \Rightarrow (c \vee d))$

we have: $(f \Rightarrow g) \Rightarrow [(AX\ \mathbf{false} \vee AX!f) \Rightarrow (AX\ \mathbf{false} \vee AX!g)]$, and then :

$(\mathcal{B}(p) \Rightarrow \mathcal{B}(q)) \Rightarrow (\mathcal{M}'(p) \Rightarrow \mathcal{M}'(q))$

(if)

We can note that:

$\{[(a \vee b) \Rightarrow (a \vee c)] \wedge (a \Rightarrow \neg b) \wedge (a \Rightarrow \neg c) \wedge (c \Rightarrow \neg a) \wedge (b \Rightarrow \neg a)\} \Rightarrow (b \Rightarrow c)$.

So we have:

$(\mathcal{M}'(p) \Rightarrow \mathcal{M}'(q)) \Rightarrow [(AX\ \mathbf{false} \vee AX!\mathcal{B}(p)) \Rightarrow (AX\ \mathbf{false} \vee AX!\mathcal{B}(q))]$

and

$AX\ \mathbf{false} \Rightarrow \neg AX!\ f\ ,\ \forall\ f$

and

$AX!\ f \Rightarrow \neg AX\ \mathbf{false}\ ,\ \forall\ f.$

Then $(\mathcal{M}'(p) \Rightarrow \mathcal{M}'(q)) \Rightarrow (\mathcal{B}(p) \Rightarrow \mathcal{B}(q))$.

Lemma (#): $p < q \Rightarrow (\mathcal{M}'(p) \Rightarrow \mathcal{M}'(q))$.

Proof: (by structural induction)

We have two cases:

*) $p \not\!\!\rightarrow \forall a$;

then $p < q$ $\forall q$ and for the Lemma 8. we have $\mathcal{B}(p) \equiv$ **false** and **false** $\Rightarrow \mathcal{B}(q)$ $\forall q$.

*) $p \rightarrow_{a_i} p_i$ (i=1, ..,n).

Since $p < q$ we have:

$\exists q_1, .., q_n$ such that $q \rightarrow_{a_i} q_i$ (i=1, ..,n) and $p_i < q_i$; then for the Lemma 7. we have:

$\mathcal{B}(p) \equiv \vee_i (a_i \wedge \mathcal{M}'(p_i))$

and

$\mathcal{B}(q) \equiv \vee_i (a_i \wedge \mathcal{M}'(q_i)) \vee (\vee_j (b_j \wedge \mathcal{M}'(q'_j)))$.

Also, for the induction hypothesis, $\mathcal{M}'(p_i) \Rightarrow \mathcal{M}'(q_i)$ (i=1, ..,n); so we have $\mathcal{B}(p) \Rightarrow \mathcal{B}(q)$ and then, for the Lemma 10., $\mathcal{M}'(p) \Rightarrow \mathcal{M}'(q)$.

Lemma (##): *$(\mathcal{M}'(p) \Rightarrow \mathcal{M}'(q)) \Rightarrow p < q$.*

Proof: (by structural induction)

We have two cases:

*) $p \not\!\!\rightarrow \forall a$;

For the Lemma 8. we have $\mathcal{B}(p) \equiv$ **false** ; therefore $\mathcal{M}'(p) = AX\,\mathcal{B}(p) \equiv AX$ **false**.

We have AX **false** $\Rightarrow \mathcal{M}'(q) \forall q$ and since $p \not\!\!\rightarrow \forall a$ $p < q$ $\forall q$.

*) $p \rightarrow_{a_i} p_i$ (i=1, ..,n);

For the Lemma7 we have $\mathcal{B}(p) \equiv \vee_i (a_i \wedge \mathcal{M}'(p_i))$ and so $\mathcal{M}'(p) \equiv AX(\vee_i (a_i \wedge \mathcal{M}'(p_i)))$.

Since for the hypothesis $\mathcal{M}'(p) \Rightarrow \mathcal{M}'(q)$ we have, for the Lemma 10, $\mathcal{B}(p) \Rightarrow \mathcal{B}(q)$, i.e. $\vee_i (a_i \wedge \mathcal{M}'(p_i)) \Rightarrow \mathcal{B}(q)$. Then, for the Lemma 9., we have:

$\exists$ $q_1, .., q_n$ such that $q \rightarrow_{a_i} q_i$ (i=1, ..,n) and $\mathcal{M}'(p_i) \Rightarrow \mathcal{M}'(q_i)$; thus, for the induction hypothesis, $p_i < q_i$, and so $p < q$.

Theorem 2: *The semantics $\mathcal{M}'$ for $\mathcal{APA}_S$ is fully abstract with respect to simulation semantics, i.e. $\mathcal{M}'(p) = \mathcal{M}'(q)$ if and only if $p <> q$.*

Proof: this result follows from Lemmas (#) and (##).