Extended Transition Systems for Parametric Bisimulation
(Extended Abstract)

P. Inverardi, C. Priami, D. Yankelevich

Nota Interna IEI B4-60
Novembre 1992
1. Introduction

In the last years, a wide spectrum of semantic models for concurrency has been developed. Any model deals with different aspects of distributed concurrent systems. Moreover, there is not a general accepted model. Indeed, different approaches may be used to specify different properties (or views) of the system. For example, the final user of a system may just want to know how the system behaves in terms of its possible temporal sequences of actions, and thus the interleaving semantics [Mil89] is suitable for him. On the other hand, the designers of a system usually need information about causal and spatial dependencies between the actions of the system, and thus a truly concurrent semantics is more adequate for them.

A formalism able to describe many models in a single framework provides a general basis that permits to:

i. classify and compare the descriptive power of such models on firm grounds;
ii. reuse general definitions and results
iii. describe different views of a system without changing almost nothing of the underlying theory.

These considerations leads to the development of parametric theories [DDM92a, DDM92b, DP92, MY92, FGM91]. These approaches, starting from a very detailed description of systems given in the SOS style [Plo81], provide abstraction mechanisms to recover many models presented in the literature. The parametric approach of [DDM92a, DDM92b] roughly consists of the following four steps.

i. Define elementary transitions which describe the immediate evolution of the system;
ii. Construct computations of the system as paths in the transition system and give them a tree structure by ordering them by prefix;
iii. Introduce observations over computations to abstract from unwanted details, and decorate the trees above with observations, thus obtaining observation trees;
iv. Compare observation trees to determine which systems have an equivalent behaviour by means of a bisimulation relation.

Following this approach, many semantics can be captured just by slightly changing the observations of the third step (see [MY92]).

An alternative approach [DP92] take proved transition systems [BC88], where transitions are labelled with their proofs, as privileged transition system. Computations are defined as sequences of transitions, and are organized as trees, called proved trees. Their arcs are labelled with the proofs of the transitions and then are observed to recover many semantic models. Classical bisimulation is defined over proved trees for comparing the behaviour of systems.
In [FGM91] transition are considered to be terms of an algebra. The description of a system is driven to a tree-structured normal form in an axiomatic way. Also in this case, abstraction mechanisms allows to retrieve many models for the semantics of concurrent systems.

All approaches outlined above relies on tree structure to describe the behaviour of a system. Thus, they can finitely describe only systems having finite behaviour. This fact is a major drawback to perform bisimulation-based verifications. In fact, most of the approaches to automatic verification uses algorithms which rely on the transition system representation of the system [IP91, Kor91]. Moreover, almost any example of distributed systems has possibly infinite behaviours, and therefore it cannot be checked by using tree-like representations.

In this paper we propose a new kind of transition systems, called Extended Transition Systems (ETS). This model gives a parametric finite representation of a finite state system, by unifying many of the ideas of [DDM92a, DDM92b] from one side and of [BC88, DP92] from the other. Indeed, we start from the proved transition system and we label each node \( n \) with a regular expression describing all computations from the initial state to \( n \). Hence, the number of the states in our representation does not increase with respect to the original transition system. We define observations over ETS and a notion of parametric bisimulation. Moreover, we show that our parametric bisimulation coincides with the equivalences defined in [DDM92a, DDM92b, DP92]. Also, we give some results to compute bisimulation for a class of observations, called incremental, and we briefly discuss how this approach can be extended to other observations. An immediate application of the results of this paper is the development of parametric tools for verification of concurrent systems like [IPY92] also for finite state systems.

The paper is organized as follows. The following section recall the background of parametric theories. In Section 3 the extended transition systems are introduced and a notion of parametric bisimulation is defined over them. Also, we show that our equivalence coincides with the ones of [DDM92a, DDM92b, DP92]. Finally, Section 4 discusses how regular expressions can be observed.

2. Background

In this section we briefly recall the basic notions of the parametric theory based on the so-called observation trees [DDM92a, DDM92b], on proved transition systems [BC88] and on proved trees [DP92]. We recall the basic definitions of that theory by considering systems specified in the well-known Calculus of Communicating Systems (CCS) [Mil89], the basic notions of which follow.

**Definition 2.1 (CCS)**

Let \( \Delta \) be a set of names (ranged over by \( \alpha, \beta, \ldots \)). Let \( " \) be an involution on \( \Delta \), and call complementary names the elements of the set \( \Delta " \). Then \( \Lambda = \Delta \cup \Delta " \) is the set of labels (ranged over by \( \lambda \)) and \( \Lambda = \Lambda \cup \{ \tau \} \) is the set of actions (ranged over by \( \mu \)). The set \( T \) of closed CCS terms is expressed by the following BNF-grammar:

\[
t ::= \text{nil} | x | u | t | (t | t) | (t + t) | (t \cdot u) | (t \cdot \alpha) | (t \cdot \phi) | \text{rec } x.t
\]

with the restriction that the term \( t \) in \( \text{rec } x.t \) is well-guarded, and where \( \phi \) is a relabelling function preserving \( \tau \) and ".

In order to have a very concrete model for the description of distributed concurrent systems, the operational semantics of CCS is given in the SOS style [Pl81] through the proved transition system (PTS) [BC88]. The transitions of PTS are labelled by encodings of their proofs.\(^1\) More

\(^1\) Alternatively, the proofs themselves are the transitions, that make it possible to define a powerful algebra of transitions [MY89, FM90].
precisely, we consider a variant of the Boudol and Castellani’s PTS, where the representation of synchronizations requires two rules for restriction as in [DP92]. Moreover, a rule for relabelling is introduced. First, we define the alphabet for the labelling of transitions, and then we report the proved transition system.

Definition 2.2 (Proof Terms)
Let $\Theta \in \{\text{Nil}, \text{Nil}_1, +_0, +_1, \text{Nil}_0, \text{Nil}_1, \text{Nil}_0\}^*$. The set of proof terms is $\Theta = \{\vartheta \mu | \mu \in A\} \cup \{\vartheta \lambda_0 \lambda, \vartheta \lambda_1 \lambda | \lambda \in \Lambda\}$, ranged over by $\Theta$.

Definition 2.3 (Proved Transition System, PTS [BC88])
The proved transition system $\text{PTS} = (T, \Theta, \rightarrow)$, where $T$ is the set of CCS terms, $\Theta$ is the set of proof terms and the transition relation $\rightarrow$ is defined by the following axiom and rules:

- **Act**
  \[
  \mu, t \rightarrow \mu t \rightarrow t
  \]

- **Sum**
  \[
  \frac{1 \rightarrow \theta \rightarrow t'}{t + t'' \rightarrow \theta \rightarrow t'}
  \]

- **Rel**
  \[
  \frac{1 \rightarrow \theta \rightarrow t'}{t(\cdot) \rightarrow \theta \rightarrow t(\cdot)}
  \]

- **Res**
  \[
  \frac{1 \rightarrow \theta \rightarrow t'}{t, \mu \rightarrow \theta \rightarrow t, \mu}
  \]

- **Asyn**
  \[
  \frac{1 \rightarrow \theta \rightarrow t'}{t || t' \rightarrow \theta \rightarrow t' || t'}
  \]

- **Syn**
  \[
  \frac{t_0 \rightarrow \theta \rightarrow t', t_1 \rightarrow \theta \rightarrow t'_1}{t_0 || t_1 \rightarrow \theta \rightarrow t'_0 || t'_1}
  \]

- **Rec**
  \[
  \frac{t[\text{rec } x, t'(x)] \rightarrow \theta \rightarrow t'}{t[\text{rec } x, t'(x)] \rightarrow \theta \rightarrow t'}
  \]

As it is already noted by Boudol and Castellani, it is immediate to recover the classical labelled transition system of CCS [Mil89] from the above one. It is sufficient to delete the proof part of the labels of the transitions. We define the operational meaning of each CCS term $t$ as the portion of the PTS that is generated by the term $t$, denoted by $[t]_{\text{PTS}}$. Alternatively, the operational meaning of a CCS term $t$ can be defined as the unfolding from $t$ of the proved transition system as in [DP92]. The obtained tree is called proved tree and it is denoted by $[t]_{\text{PT}}$.

Example 2.1 (Operational Construction of Proved Transition Systems)
Consider the CCS term $t = \text{rec } x. \alpha \beta x \mid \text{rec } x. \gamma x$. The PTS generated by the operational rules of Definition 2.3 is illustrated in Figure 2.1.

![Fig. 2.1 - Proved Transition System generated by the term rec x. \alpha \beta x \mid rec x. \gamma x](image)

The formal definition of proved trees follows, where the standard syntax for trees as summations is assumed.
Definition 2.4 (Proved Tree)
The proved tree of a CCS term \( t \) is \( [t]_{pt} = \Sigma_{i \in I} \theta_i [t_i]_{pt} \), where, \( \forall i \in I, t \xrightarrow{\theta_i} t_i \) is a transition of the proved transition system. In the sequel, isomorphic trees will be identified. The set of proved trees generated by CCS terms will be denoted by \( PT \).

Given a tree, an arc is uniquely determined by its label and its depth. Indeed, proved trees cannot be simplified applying the axiom \( x + x = x \), stating idempotence of summation, as tag \( +0 \) is different from \( +1 \). Therefore, in the formal summation \( \Sigma_{i \in I} \theta_i [t_i]_{pt} \), \( \forall i \in I, \exists i \), we have \( \theta_i \neq \theta_j \). Note also that the set \( I \) is always finite, i.e. the trees are finitely branching, because CCS terms are guarded.

Now, we define computations as sequences of transitions, i.e. paths in the PTS.

Definition 2.5 (Computations)
Let \( [t]_{pt} \) be the proved transition system generated by \( t \) and let \( t_0 \xrightarrow{\theta} t_1 \) be a transition of \( [t]_{pt} \). Then, \( t_0 \) is called the source of the transition and \( t_1 \) its target. The set \( C \) of computations of \( [t]_{pt} \) is given by all sequences \( t \xrightarrow{\theta_0} t_1 \xrightarrow{\theta_1} t_2 \xrightarrow{\theta_2} t_3 \xrightarrow{\theta_3} \ldots \) starting from \( t \) and such that the target of any transition coincides with the source of the next one. In the sequel, we use the notation \( t = \xi \Rightarrow t_j \), where \( \xi = \theta_0 \theta_1 \theta_2 \) ranges over \( C \). The empty computation is denoted by \( \varepsilon \). Recall that the operator \( \ldots \Rightarrow \ldots \) is used for the sequential composition of transitions. Also, source and target are extended in the obvious way to computations.

The next step of the described methodology consists in the generation of a tree-like structure by ordering by prefix all computations outgoing from a starting state (corresponding to an agent) as the basic model of concurrent distributed systems. These structures are called observable trees [DDM92b]. Roughly speaking, an observable tree generated by an agent \( t \) is a tree, the nodes of which are the computations of \( t \). The formal definition is the following.

Definition 2.6 (Observable Trees)
Let \( t \) be a CCS term and let \( N = \{ \xi \mid t = \xi \Rightarrow t_1 \} \) with \( t_1 \) a reachable state = of \( [t]_{pt} \) \( \cup \{ \varepsilon \} \) be the set of all computations with source \( t \) and leading to any reachable state of \( [t]_{pt} \) Moreover, let \( \leq_{pre} \) be an ordering relation over \( N \) such that \( \xi' \leq_{pre} \xi'' \) iff \( \xi'' = \xi' \xi \xi' \xi'' \) with \( \xi \) possibly empty, \( \xi, \xi', \xi'' \in N \). Then, the observable tree generated by \( t \) is \( [t]_{ot} = \langle N, \leq_{pre} \rangle \).

Let \( n, n', n'' \in N \). Then, a node \( n' \) is an immediate successor of a node \( n \) iff \( n <_{pre} n' \) and there is no \( n'' \) such that \( n <_{pre} n'' <_{pre} n' \). Also, \( n' \) is a successor of \( n \) iff \( n \leq_{pre} n' \). Finally, \( n' \) is a proper successor of \( n \) iff \( n' \) is a successor of \( n \) and \( n' \neq n \).

Note that the relation over computations corresponds to the prefix ordering over them. Also, recall that the observable trees are the integral variant of the so-called proved trees [DP92], where arcs are labelled by proved transitions.

Depending on the properties to investigate of a distributed concurrent system described through a CCS agent, many details contained in the nodes of the corresponding observable tree may be superfluous. Recall that from the proved transition system, almost all models for the semantics of concurrent system proposed in the literature can be retrieved by abstracting from unwanted details [DP92]. As an example, if we are simply interested in the temporal ordering of the events [Mil89], it suffices to get as computations the sequences of action labels. Instead, if causality [DD89], locality [BCHK92], or both of them [Kie91] are of interest also the concurrent structure of an agent, represented by parallel operators, must be recorded in the computations [DP92, MY92, DDM90]. However, the computations that we use as nodes of observable trees contains many more information than the one mentioned above. Therefore, an abstraction step, called observation, is needed in order to get only the relevant information. Following the
approaches of [DDM92a, DP92, MY92], the observation of an observable tree essentially consists in labelling its nodes with encodings of the relevant information that is contained in the corresponding computations. Roughly speaking, only the needed aspects of the experiment described by a node are made visible to the observer. More precisely, we have the following definition.

**Definition 2.7 (Observation Function)**
Let \( \langle N, \leq_{\text{pre}} \rangle \) be an observable tree and let \( \langle D, \leq \rangle \) be a partial ordering. Then, an observation function is a monotone function \( O : \langle N, \leq_{\text{pre}} \rangle \to \langle D, \leq \rangle \) such that \( n \leq_{\text{pre}} n' \), with \( n, n' \in N \), implies \( O(n) \leq O(n') \).

Now, we report the definition of observed tree, i.e., an observable tree with nodes labelled by the information that is extracted from computations.

**Definition 2.8 (Observed Trees)**
Let \( \langle N, \leq_{\text{pre}} \rangle \) be an observable tree generated by a CCS term \( t \) and let \( O \) be an observation function. Then \( \langle N, \leq_{\text{pre}}, O \rangle \) is the observed tree corresponding to \( \langle N, \leq_{\text{pre}} \rangle \) according to \( O \).

The last step of the methodology sketched in the Introduction consists in comparing observed trees through bisimulations in order to determine which terms of the language have an equivalent behaviour according to a selected observation. Hence, we report the definition of bisimulation on observed trees.

**Definition 2.9 (Bisimulation on Observed Trees)**
Let \( \langle N, \leq_{\text{pre}}, O \rangle \) and \( \langle N', \leq_{\text{pre}}, O \rangle \) be two observed trees generated by the CCS terms \( t \) and \( t' \), respectively. Let \( R \) be a symmetric binary relation on \( N \cup N' \). Assuming that \( n, n_1, n_2 \in N \) and \( n', n'_1, n'_2 \in N' \), the relation \( R \) is a strong bisimulation iff
\[
n_1 R n'_1 \text{ implies that } O(n_1) = O(n'_1) \quad \text{and for every immediate successor } n_2 \text{ of } n_1 \text{ there exists an immediate successor } n'_2 \text{ of } n'_1 \text{ such that } n_2 R n'_2 \text{ and for any } n', n'_1 \leq n' \leq n'_2, \text{ it is } O(n') = O(n'_1) \quad \text{or } O(n') = O(n'_2).
\]

Analogously, the relation \( R \) is a weak bisimulation iff
\[
n_1 R n'_1 \text{ implies that } O(n_1) = O(n'_1) \quad \text{and for every immediate successor } n_2 \text{ of } n_1 \text{ there is a successor } n'_2 \text{ of } n'_1 \text{ such that } n_2 R n'_2 \text{ and for any } n', n'_1 \leq n' \leq n'_2, \text{ it is } O(n') = O(n'_1) \quad \text{or } O(n') = O(n'_2).
\]

The Example 2.2 shows all the steps of the described methodology, first reporting the operational construction of observable trees, and then illustrating how these trees are observed with the partial ordering observation of events.

**Example 2.2 (Construction of Observable Trees and their Observation)**
Consider the CCS term \( t = \alpha.(\beta.\text{nil} \mid \delta.\text{nil}) \). According to the operational semantics the observable

---

**Figure 2.2 - Observable and Observed Tree generated by the CCS term** \( t = \alpha.(\beta.\text{nil} \mid \delta.\text{nil}) \).
tree originated by $t$ is depicted in Figure 2.2, where the computations are the nodes of the tree. If we are interested e.g. to causality, we label the node of the tree with the partial ordering of events generated by $t$. For instance, the computation $\alpha \parallel \beta | \delta$ says that the action $\alpha$ causes both the event $\beta$ and $\delta$, but these two last actions are independent because one happens in the left part of a parallel composition and the other in the right part. Hence, the labelling of the corresponding node.

In the sequel, the reader is assumed familiar with regular expressions and with the usual notation for denoting them.

3. Extended Transition Systems

In this section we extend the theory of observable and observed trees [DDM92] in order to cope with finite state processes. More precisely, we define node-labelled graphs called extended transition systems where any node $n$ is a CCS term that can be labelled with a regular expression which encodes all computations that lead from the starting state to $n$. Therefore, since any state is a term of the language and not a computation as in [DDM92], it is possible to have finite parametric representations of regular processes. For the sake of readability, we first introduce an intermediate step which consists in restating the definition of bisimulation in order to consider the history of computations. This new definition is proved to be equivalent to Definition 2.9 for observed trees. Then, extended transition systems are defined and equipped with a notion of bisimulation. Finally, the bisimulation over extended transition systems is showed equivalent to the definition of the intermediate step. The following fact obviously derive from formal languages theory.

**Fact 3.1 (Computations and Regular Expressions)**

Let $t$ be a CCS term. If $[t]_{\text{plS}}$ is finite state, then each computation $\xi$ with source $t$ in $[t]_{\text{plS}}$ is a regular language over the alphabet $\Theta$ which does not contain $\tau$.

The first step of the presentation is the definition of bisimulation that keeps track of the history of computations. Before, we need some notation.

**Notation 3.1 (Sequences of moves)**

Let $[t]_{\text{plS}}$ be a proved transition system. Then, we use $t_0 \Rightarrow t_k$ if there is a computation $t_0 \rightarrow t_1 \rightarrow \ldots \rightarrow t_{k-1} \rightarrow t_k$ in $[t]_{\text{plS}}$ with $\theta_i = \tau$ or $\theta_i = \varphi \cdot \theta_0 \lambda^*, \varphi \neq \lambda^*$, $0 \leq i \leq k-1$. Also, $t_0 \Rightarrow \varphi = t_k$ denotes $t_0 \Rightarrow t_1 \rightarrow \ldots \rightarrow t_{k-1} \rightarrow t_k$, with $\theta_i = \varphi \mu$.

The following definition is an intermediate step towards the introduction of extended transition systems. It defines parametric bisimulation on proved transition systems by constructing the bisimulation associating to each analyzed node all possible computations leading to it. This definition is clearly not effective even when dealing with finite state systems since to a node can be associated infinitely many computations.

**Definition 3.1 (Parametric Bisimulation on PTS)**

Let $[t]_{\text{plS}}$ and $[t']_{\text{plS}}$ be two proved transition systems, and let $O$ be an observation function. Given two nodes $t_1$ and $t'_1$ of $[t]_{\text{plS}}$ and $[t']_{\text{plS}}$ respectively, let $t = \xi \Rightarrow t_1$ and $t' = \xi' \Rightarrow t'_1$. Then, a symmetric binary relation $R$ on the union of the pairs ⟨node, computation⟩ of the two transition systems is a strong parametric bisimulation iff $t_1 R t'_1$ implies that whenever $(t_1, t'_1, \xi, \xi') \in R$ then

- $t_1 \Rightarrow_\theta t_2$ implies that there are $t_2$ and $\theta'$ such that $t'_1 \Rightarrow_\theta' t'_2$, $O(\xi; \theta) = O(\xi'; \theta')$ and $t_2 R t'_2$;
- $t'_1 \Rightarrow_\theta t'_2$ implies that there are $t_2$ and $\theta$ such that $t_1 \Rightarrow_\theta t_2$, $O(\xi; \theta) = O(\xi'; \theta')$ and $t_2 R t'_2$.
Analogously, the relation $R$ is a weak parametric bisimulation iff $t_1 R t_1'$ implies that whenever $(t_1, t_1', \xi, \xi') \in R$

- $t_1 \rightarrow t_2$ implies that there are $t_1'$ and $\theta'$ such that $t_1'=\theta'=t_2'$, $O (\xi; \theta) = O (\xi'; \theta')$ and $t_2 R t_2'$;
- $t_1' \rightarrow t_2$ implies that there are $t_2$ and $\theta$ such that $t_1=\theta=t_2$, $O (\xi; \theta) = O (\xi'; \theta')$ and $t_2 R t_2'$.

Finally, $[t]_{pts}$ and $[t']_{pts}$ are strong or weak bisimilar iff $t R t'$. As usual, we take the congruences induced by the largest above relations, and denote them by $\sim_p$ and $\approx_p$, respectively.

In order to show the soundness of the above definition we have the following theorem which relates bisimulation defined on observation tree (OT), proved trees (PT) and proved transition systems (PTS).

**Theorem 3.1** *(Soundness of Parametric Bisimulation on PTS)*

Given two CCS terms $t$ and $t'$. Then, $[t]_{OT} = [t']_{OT}$ iff $[t]_{pt} \sim_p [t']_{pt}$ iff $[t]_{pts} \approx_p [t']_{pts}$.

**Proof:**

$[t]_{pt} \sim_p [t']_{pt}$ iff $[t]_{pts} \approx_p [t']_{pts}$ follows noticing that the definition of parametric bisimulation on proved trees (see Appendix) compares proved transitions, while the one on proved transition systems compares computations. The implications $[t]_{OT} = [t']_{OT}$ iff $[t]_{pt} \approx_p [t']_{pt}$ follows from the isomorphism between proved trees and observation trees [DP92], reported also in the Appendix.

We now introduce a new kind of transition systems that are labelled on nodes by ordered summation of regular expressions instead of being labelled on arcs by proof of transitions. However, the fact of labelling transitions or nodes does not change very much the equivalences or the theory. Note that it is possible to identify any node $t'$ of a transition system $[t]_{pts}$ with the set $\mathcal{C} (t') = \{ \xi_i | t \rightarrow^* t' \text{ and } \xi_i \text{ is a computation in } [t]_{pts} \}$ of all computations that have as target the considered node. Hence, from Fact 3.1 easily follows that $\mathcal{C} (t')$ is an ordered summation of regular expressions $\sum_i \gamma_i r_i$, where each $r_i$ does not contain $+$. We are now able to define the extended transition systems.

**Definition 3.2** *(Extended Transition System)*

Let $t$ be a CCS term and $[t]_{pts}$ its corresponding PTS. The extended transition system (ETS) originated by $t$ and denoted by $[t]_{ets}$ is obtained simply by labelling any node $t'$ of $[t]_{ets}$ with $\mathcal{C} = \sum_i r_i$; the ordered summation of regular expressions over the alphabet of proof terms representing the computations from $t$ to $t'$. The regular expressions in the summation are identified by their position and denoted in an array-like style as $r[i]$.

Note that given a finite state proved transition system, the corresponding extended transition system can be constructed by using an algorithm which generates a regular expression from an automata. We use as automata the proved transition system originated by a CCS term and as language the proofs of transitions. The regular expression that we associate to a node $t'$ is the one obtained by choosing it as acceptance state of the automata. This approach is very similar to ideas of path expressions [Ta81a], where regular expressions are used to describe all paths between two nodes of a directed graph. An algorithm to construct path expressions is presented in [Ta81b].

The next step introduces the notion of bisimulation over the ETS. Note that differently from Definition 3.1, the following definition provides an effective checking algorithm when dealing with finite state systems. In fact, in this case the number of nodes to be examined is finite as well as the size of their labels.

**Definition 3.3** *(Parametric Bisimulation on Extended Transition Systems)*

Let $[t]_{ets}$ and $[t']_{ets}$ be two ETS, and let $O$ be an observation function. Given two nodes $t_1$ and $t_1'$
of \([t]_{\text{ets}}\) and \([t']_{\text{ets}}\), let \(\Sigma_i\) and \(\Sigma_j\) be their corresponding labels. Then, a relation \(R_e\) on the nodes of the transition systems is a **strong parametric bisimulation** if and only if \(t_1 R_e t'\) implies that whenever \((t_1, t'_1, r[h], r'[k]) \in R_e\) then

- \(t_1 \to t_2\) implies that there exist \(t'_2\) labelled by \(\mathcal{R}'_2\) such that \(t'_1 \to t'_2\), \(O (r_2[h]) = O (r'_2[k])\) and \(t_2 R_e t'_2\), where \(r_2[h]\) and \(r'_2[k]\) are the \(h\)th and the \(k\)th regular expression in the summations of \(t_2\) and \(t'_2\), respectively;

- \(t'_1 \to t'_2\) implies that there exist \(t_2\) and \(\mathcal{R}_2\) such that \(t'_1 \to t'_2\), \(O (r_2[h]) = O (r'_2[k])\) and \(t_2 R_e t'_2\), where \(r_2[h]\) and \(r'_2[k]\) are the \(h\)th and the \(k\)th regular expression in the summations of \(t_2\) and \(t'_2\), respectively.

Analogously, the relation \(R_e\) on the nodes of the transition systems is a **weak parametric bisimulation** if and only if \(t_1 R_e t'\) implies that whenever \((t_1, t'_1, r[h], r'[k]) \in R_e\) then

- \(t_1 \to t_2\) implies that there exist \(t'_2\) and \(\mathcal{R}'_2\) such that \(t'_1 \to t'_2\), \(O (r_2[h]) = O (r'_2[k])\) and \(t_2 R_e t'_2\), where \(r_2[h]\) and \(r'_2[k]\) are the \(h\)th and the \(k\)th regular expression in the summations of \(t_2\) and \(t'_2\), respectively;

- \(t'_1 \to t'_2\) implies that there exist \(t_2\) and \(\mathcal{R}_2\) such that \(t'_1 \to t'_2\), \(O (r_2[h]) = O (r'_2[k])\) and \(t_2 R_e t'_2\), where \(r_2[h]\) and \(r'_2[k]\) are the \(h\)th and the \(k\)th regular expression in the summations of \(t_2\) and \(t'_2\), respectively.

Finally, \([t]_{\text{ets}}\) and \([t']_{\text{ets}}\) are bisimilar if \(t R_e t'\). As usual, we take the congruences induced by the largest above relations, and denote them again by \(=_{\text{p}}\) and \(=_{\text{w}}\).

Note that the above definition of parametric bisimulation deals with the observation of a regular expression. This topic is the argument of the next section.

The soundness of Definition 3.3 is guaranteed by the following theorem.

**Theorem 3.2 (Soundness of Parametric Bisimulation on ETS)**

Given two CCS terms \(t\) and \(t'\), \([t]_{\text{ets}} =_{\text{p}} [t']_{\text{ets}}\) iff \([t]_{\text{pts}} =_{\text{p}} [t']_{\text{pts}}\).

**Outline of the proof:**

Given a node \(t_i\) of \([t]_{\text{ets}}\), the regular expression \(\mathcal{R}_i\) associated to \(t_i\) is a finite representation of all computations from \(t\) to \(t_i\). Hence, a step of comparison in ETS corresponds to many (possibly infinite) steps of comparison in PTS. Assume that \([t]_{\text{ets}} =_{\text{p}} [t']_{\text{ets}}\). This means that there is no computation that, once observed, differentiates the behaviour of \(t\) and \(t'\). Thus, \([t]_{\text{pts}} =_{\text{p}} [t']_{\text{pts}}\). Viceversa, if \([t]_{\text{pts}} =_{\text{p}} [t']_{\text{pts}}\), there is no computation that, once observed, differentiates the behaviour of \(t\) and \(t'\). Therefore, if we consider the sets of computation leading to a given node globally, i.e., we consider the corresponding regular expression, the equivalence relation still holds. Thus, \([t]_{\text{ets}} =_{\text{p}} [t']_{\text{ets}}\).

---

4. Observations and Regular Languages

In this section we discuss the problem of dealing with the equivalence of labels in the extended transition systems. Indeed, since the labels to be compared in the ETS are regular expressions that in general describe an infinite set of computations, we want to determine if the (possibly infinite) set of observations of two labels are the same or not. More precisely, if \(O\) is the obvious extension of an observation function to sets, we want to know if \(O(\mathcal{R}) = O(\mathcal{R}')\), where \(\mathcal{R}\) and \(\mathcal{R}'\) are the regular expressions labelling two nodes of an ETS. As a first result, we report the following fact.

**Fact 4.1 (Languages and Observations)**

Let \(\mathcal{R}\) and \(\mathcal{R}'\) be two regular expressions, and let \(L(\mathcal{R})\) and \(L(\mathcal{R}')\) be the languages denoted by \(\mathcal{R}\) and \(\mathcal{R}'\), respectively. Then, \(L(\mathcal{R}) = L(\mathcal{R}')\) implies \(O(\mathcal{R}) = O(\mathcal{R}')\).
Clearly, the above fact gives a sufficient condition for checking equivalences by using well-known techniques from compiler and formal languages theory. There are cases when the languages of two regular expressions are not equal, but their observation in a particular model must coincide. In order to deal with these cases we distinguish observations in incremental and non-incremental, according to [DDM92b].

Definition 4.1 (Incremental Observations)
Let $\xi_1, \xi_2^1, \xi_2, \xi_2^2$ be computations and let $O$ be an observation function. Then, $O$ is called incremental if $O(\xi_1) = O(\xi_1^1)$ and $O(\xi_2) = O(\xi_2^2)$ imply $O(\xi_1 \cdot \xi_1^1) = O(\xi_2 \cdot \xi_2^2)$, where $\cdot$ denote concatenation of computations.

For any incremental observation $O$ it is very easy to translate the structure of regular expressions of computations into observations. More precisely, we have the following lemma.

Lemma 4.1 (Languages and Incremental Observations)
Let $O$ be an incremental observation function and let $O(\alpha) = \alpha_1$ and $O(\alpha) = \alpha_1'$. Then, $O(\alpha) = O(\alpha_1)$ if and only if $L(\alpha_1 \cdot) = L(\alpha_1')$.

Proof:
Since $O$ preserves the constructors of regular expressions +, . and * (see Definition 4.1), it is possible to concatenate observations and to compute finite iterations of them from a basic value. Hence, the observation of a regular expression is itself a regular expression on a different alphabet. Finally, two regular expressions are equivalent if their languages are equal.

Example 4.1 (Observing Extended Transition Systems)
Consider CCS term $t = \text{rec } x. \alpha \cdot \beta \cdot x \cdot \text{rec } y. \gamma \cdot x$. The ETS is illustrated in Figure 2.1. The corresponding ETS is reported in Figure 4.1(a). Also, consider the CCS term $t' = \text{rec } x. (\gamma \cdot x + \alpha \cdot (\text{rec } y. (\gamma \cdot \gamma + \beta \cdot x)))$. The ETS of $t'$ is shown in Figure 4.1(b). Clearly, $[t]$ and $[t']$ are not bisimilar in any observation that distinguishes the parallel composition from interleaving and nondeterminism. Hence, let $O$ be the classical interleaving observation function, that is incremental. We have $O(\alpha) = O(\alpha_1)$ and also $O(\alpha) = O(\alpha_1')$. Thus, by application of Lemma 4.1 the bisimulation that proves the equivalence $[t]$ and $[t']$ is the following:

\[
\{ (p, q), (\alpha \cdot \alpha) \cdot (\beta \cdot \beta), (\alpha \cdot \alpha) \cdot (\beta \cdot \beta), (\gamma \cdot \gamma) \cdot (\alpha \cdot \alpha) \cdot (\beta \cdot \beta), (\alpha \cdot \alpha) \cdot (\beta \cdot \beta), (\alpha \cdot \alpha) \cdot (\beta \cdot \beta), (\alpha \cdot \alpha) \cdot (\beta \cdot \beta) \}
\]

Figure 4.1 - ETS corresponding to the terms $t = \text{rec } x. \alpha \cdot \beta \cdot x \cdot \text{rec } y. \gamma \cdot x$ (a) and $t' = \text{rec } x. (\gamma \cdot x + \alpha \cdot (\text{rec } y. (\gamma \cdot \gamma + \beta \cdot x)))$ (b).
For non-incremental observations, the situation is more complex. Indeed, these observations do not preserve the structure of computations and therefore Lemma 4.1 does not apply. In order to cope with this problem, we decompose any non-incremental observation into an incremental one and an abstraction function $f$. Thus, we use Lemma 4.1 for the incremental component of the observation and then we apply $f$ during the comparison steps. The existence of the decomposi-
tion is ensured by the following fact.

**Fact 4.2 (Decomposition of non-incremental Observations)**

Let $O$ be a non-incremental observation function. Then, there exists an incremental observation function $O'$ and an abstraction function $f$ such that $O = f \cdot O'$, where $\cdot$ denotes function composition.

In the following example we show how it is possible to obtain a non-incremental observation like partial orders from a more concrete and incremental one like spatial histories [FGM91] (also called concurrent histories in [DM87]).

**Example 4.2 (Decomposition of non-incremental observations)**

Consider the CCS term $t = (\alpha.\text{nil}) + \gamma.(\gamma.\text{nil} + \beta.\text{nil})) + \delta.\text{nil}$ where $\phi$ is such that $\phi(\gamma) = \alpha$. A possible computation of $t$ is $\xi = +\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{nil})\phi(\text{ni...
5. Conclusions

In this work we have defined a parametric model for describing the semantics of finite states systems. It relies on the parametric theories of observation trees [DDM92a, DDM92b] and proved trees [BC88, DP92]. A notion of parametric bisimulation is introduced over extended transition systems and it is shown that this equivalence coincide with the ones already presented in literature. The result presented in this paper are fundamental for a practical use of a parametric theory in automatic verification of systems.

Following the definitions given above, it is immediate to derive an algorithm for parametric bisimulation when incremental observations are considered. The extension to other observations is under development following the approach of decomposing any non-incremental observation into an incremental one and an abstraction function. The basic idea is use standard techniques from fix-point theory to prove properties on regular languages that may permit to make this decomposition effective.

References


Appendix

Notation (Moves and sequences of moves)

Let $T = (N, A, I)$ be an observed proved tree; then we define the following relations between nodes of $T$. We write $n \overset{a}{\rightarrow} n'$ if $(n, n') \in A$ and $I(n, n') = a$. We use $n_0 \Rrightarrow n_k$ if there is a path $<n_0, n_1, n_2, \ldots, n_{k-1}, n_k>$ of $T$ with $I(n_i, n_{i+1}) = \tau$, $0 \leq i \leq k-1$. Also, $n_0 \overset{a}{\Rightarrow} n_k$ denotes $n_0 \overset{a}{\rightarrow} n_1 \overset{a}{\rightarrow} \cdots \overset{a}{\rightarrow} n_{k-1} \overset{a}{\rightarrow} n_k$, with $I(n_i, n_{i+1}) = \mu$. Finally, we use $n_0 \overset{a}{\Rightarrow} n_k$, with $s = \lambda_1 \lambda_2 \ldots \lambda_k$, $0 \leq k$, to denote $n_0 \overset{a}{\rightarrow} n_1 \overset{a}{\rightarrow} \cdots \overset{a}{\rightarrow} n_k$. \hfill \spadesuit

In order to define the observational congruence, we introduce weak equivalence as an auxiliary definition.

**Definition A.1 (Weak Equivalence)**

Two observed trees $T$ and $T'$ are weakly bisimilar if there exists a symmetric relation $=_w$ on the nodes of $T$ and $T'$ such that

i) if $r$ and $r'$ are the roots of $T$ and $T'$, then $r =_w r'$

ii) if $n =_w n'$ and $n \overset{a}{\Rightarrow} n_1$ then $n' \overset{a}{\Rightarrow} n_2$ and $n_1 =_w n_2$

As usual, we will consider the maximal weak bisimulation only, also denoted by $=_w$. \hfill \spadesuit

We now define the (strong and) observational congruence over observed proved trees.

**Definition A.2 (Strong and Observational Congruences)**

If $T_1$ and $T_2$ are two observed trees, then we define the following two relations, where $r_1$ is the root of $T_1$ and $r_2$ is the root of $T_2$:

a) $r_1 =_s r_2$ iff $r_1 \overset{a}{\rightarrow} n_1$ then $r_2 \overset{a}{\rightarrow} n_2$ and $n_1 =_s n_2$

b) $r_1 =_o r_2$ iff $r_1 \overset{a}{\rightarrow} n_1$ then $r_2 \overset{a}{\rightarrow} n_2$ and $n_1 =_o n_2$

As usual, we take the congruences induced by the largest above relations, and denote them again by $=_s$, and $=_o$. \hfill \spadesuit

**Fact A.1 (Relations between proved trees and observation trees)**

Let a transition system $ST$ have transitions $\{t_i\}$; let $OT = \langle N, \leq, o \rangle$ be its observation tree where $o$ is the identity function (i.e., if $\sigma = t_0 t_1 \ldots t_n$ is a computation of $ST$ (= a node of $OT$) then $o(\sigma) = \sigma$); and let $PT = \langle P, A, i \rangle$ be its proved tree with nodes in $P$, arcs in $A$, labelled by the identity function $i$ (i.e., $i(t) = t$). Then

i. $PT$ is isomorphic to the tree built starting from $OT$, defined as the tree with nodes $N$, and arcs $\{\sigma, \sigma \triangleright \sigma \mid \sigma \in N, \sigma \leq \sigma\}$, labelled each by $i$.

ii. $OT$ is isomorphic to the tree $\langle C, \overset{\ast}{=} \overset{o}{\circ} \rangle$ built starting from $PT$, where $C = \{\sigma \triangleright \sigma\}$ is a (possibly empty) path of $PT$, and $\overset{\ast}{=} \overset{o}{\circ}$ is the transitive, reflexive closure of the relation induced by $A$. \hfill \spadesuit