An Implementation of an Efficient Algorithm for Bisimulation Equivalence

Jean-Claude Fernandez
IMAG–LGI
BP53X
38041 GRENOBLE Cedex
FRANCE
Tel : 76 51 49 15
e-mail fernand@imag.imag.fr

November 13, 1989

Abstract

We present an efficient algorithm for bisimulation equivalence. Generally, bisimulation equivalence can be tested in $O(mn)$ for a labeled transition system with m transitions and n states. In order to come up with a more efficient algorithm, we establish a relationship between bisimulation equivalence and the relational coarsest partition problem, solved by Paige & Tarjan in $O(m \log n)$ time. Given an initial partition and a binary relation, the problem is to find the coarsest partition compatible with them. Computing bisimulation equivalence can be viewed both as an instance and as a generalization of this problem: an instance, because only the universal partition is considered as an initial partition and a generalization since we want to find a partition compatible with a family of binary relations instead of one single binary relation. We describe how we have adapted the Paige & Tarjan algorithm of complexity $O(m \log n)$ to minimize labeled transition systems modulo bisimulation equivalence. This algorithm has been implemented in C and is used in Aldébaran, a tool for the verification of concurrent systems.

1 Introduction

Bisimulation equivalence plays a central role in the verification of concurrent systems based on equivalence relations between labeled transition systems [12]. Many theories of equivalence for concurrent systems have been proposed in the literature. All these equivalences are stronger than trace equivalence and weaker than strong bisimulation equivalence, or shortly, bisimulation equivalence. For example, observational [12], acceptance [8], failure [13] and testing equivalences [9], [10] belong to this class of equivalences. Usually, the problem of deciding these equivalences for two labeled transition systems can be reduced to the one of computing bisimulation equivalence between canonical forms of these systems [5]. Indeed, the computation of bisimulation can be used for reducing to a canonical form with respect to the number of states and for comparing canonical forms. Thus, an efficient algorithm computing bisimulation equivalence reveals itself quite useful for deciding the other equivalence relations [2], [3], [10], [11].

Kanellakis and Smolka [10] studied the connection between the relational coarsest partition problem and the bisimulation equivalence. They proposed an algorithm running in $O(mn)$ time. For the case in which the image set sizes are bounded by a constant c, they gave an algorithm running in $O(c^2 n \log n)$ time by generalization of the Hopcroft algorithm computing the minimum state deterministic finite automaton. In [11], the connection between the relational coarsest partition problem and parametrized bisimulations is stated. More recently, Paige and Tarjan proposed an algorithm to solve the relational coarsest partition problem in $O(m \log n)$ time. We present an adapted version of this last algorithm computing the coarsest partition problem with respect to the family of binary relations $(T_a)_{a\in A}$ instead of one binary relation.

In section 2 we recall properties of bisimulation relations. The greatest bisimulation can be obtained as a maximal fixpoint of a monotonic operator on the binary relations on Q [12]. This maximal fixpoint is an equivalence relation on Q. In section 3, we describe the many relational coarsest partition problem and the relationship between the solution and the greatest bisimulation. We give a formal specification of the many relational coarsest partition problem from a characteristic property of the compatibility of a partition with a family of binary relations. This allows us to derive an algorithm which is correct by construction. This algorithm is described in section 4. In addition, we present measures performed on Aldébaran, a tool for the verification of concurrent systems [5], using this algorithm.

2 Bisimulations

A labeled transition system is a quadruple $S = (Q, A, T, q_0)$, where Q is a set of states, A is a finite set of actions, $T \subseteq Q \times A \times Q$ is the transition relation and q_0 is the initial state. For each $a \in A$, the transition relation T_a is considered to be either a binary relation on Q: $T_a = \{(p, q) \mid (p, a, q) \in T\}$ or a function $Q \rightarrow 2^Q$: $T_a[p] = \{q \mid (p, a, q) \in T\}$. We also use the notations: $p \xrightarrow{a} q$ for $(p, a, q) \in T$, $T_a^{-1}[q] = \{p \mid (p, a, q) \in T\}$ and $T_a^{-1}[B] = \cup\{T_a^{-1}[q] \mid q \in B\}$ for $B \subseteq Q$. $\mid X \mid$ denotes the number of elements of the set X. T is image-finite if $\forall a \in A \,.\, \forall q \in Q \,.\, T_a^{-1}[q]$ is finite. By convention, n denotes the number of elements of Q, m denotes the number of elements of T and c the maximum for $a \in A$ and $q \in Q$ of the image set sizes $\mid T_a[q] \mid$.

In order to compare or to minimize labeled transition systems, we recall the notion of bisimulation. Intuitively, two states p and q are bisimilar if for each state p' reachable from p by execution of an action a there is a state q', reachable from q by execution of the same action a such that p' and q' are bisimilar and symmetrically.

Definition 2.1 *Given a labeled transition system S = (Q, A, T, q_0), a binary relation $\rho \subseteq Q \times Q$ is a* bisimulation *if and only if:*

$\forall (p_1, p_2) \in \rho \, . \, \forall a \in A \, .$
$\forall r_1 \, . \, (p_1 \xrightarrow{a} r_1 \Rightarrow \exists r_2 \, . \, (p_2 \xrightarrow{a} r_2 \wedge (r_1, r_2) \in \rho)) \;\; \wedge$
$\forall r_2 \, . \, (p_2 \xrightarrow{a} r_2 \Rightarrow \exists r_1 \, . \, (p_1 \xrightarrow{a} r_1 \wedge (r_1, r_2) \in \rho))$

The set of bisimulations on Q, ordered by inclusion has a maximal element, which may be obtained as a maximal fixpoint of an operator Ψ [14]:
For each $\rho \in Q \times Q$, we can define $\Psi(\rho) \subseteq Q \times Q$ as
$\Psi(\rho) = \{(p_1, p_2) \mid \forall a \in A \, .$
$\forall r_1 \, . \, (p_1 \xrightarrow{a} r_1 \Rightarrow \exists r_2 \, . \, (p_2 \xrightarrow{a} r_2 \wedge (r_1, r_2) \in \rho)) \;\; \wedge$
$\forall r_2 \, . \, (p_2 \xrightarrow{a} r_2 \Rightarrow \exists r_1 \, . \, (p_1 \xrightarrow{a} r_1 \wedge (r_1, r_2) \in \rho))\}.$

ρ is a bisimulation if and only if $\rho \subseteq \Psi(\rho)$. Ψ is a monotonic operator on the complete lattice of binary relations on Q, under inclusion. If T is image-finite then Ψ is $\cap$–continuous [11] (i.e. $\Psi(\bigcap_{i \in I} \rho_i) = \bigcap_{i \in I} \Psi(\rho_i)$ for each decreasing sequence $\{\rho_i \mid i \in I\}$) and has a maximal fixpoint:

$$\rho_\Psi = \bigcap_{i=0}^{\infty} \Psi^i(Q \times Q)$$

which may be obtained by computing the limit of the sequence $(\rho_r)_{r \in N}$ such that:

$$\begin{aligned} \rho_0 &= Q \times Q \\ \rho_{r+1} &= \Psi(\rho_r) \end{aligned}$$

Proposition 2.1 *ρ_Ψ is an equivalence relation on (or a partition of) Q.*

Proof. ρ being an equivalence relation, $\Psi(\rho)$ is an equivalence relation. $\square$

3 Relational Coarsest Partition Problem

In this section, we consider a labeled transition system $S = (Q, A, T, q_0)$. We represent an equivalence relation ρ on Q as a partition $\rho = \{B_1, ..., B_n\}$ where each B_i represents one of its equivalence classes
(i.e. $\forall x, y \in Q \, . \; (x, y) \in \rho$ if and only if $\exists B_i \, . \, (x \in B_i \wedge y \in B_i)$).
A partition ρ' is a *refinement* of a partition ρ, (or ρ is *coarser* than ρ') $\rho' \sqsubseteq \rho$, if and only if: $\forall B' \in \rho' \, . \, \exists B \in \rho \, . \, (B' \subseteq B)$.
Consider a partition ρ_I on Q. The set of refinements of ρ_I, ordered by $\sqsubseteq$, forms a complete lattice, $\mathcal{L}(\rho_I)$ with

- ρ_I as the unique maximum element,
- $\{\{p\} \mid p \in Q\}$ as the unique minimum element,
- $\sqcap\{\rho_j \mid j \in J\} = \{\cap_{j \in J} B_j \mid B_j \in \rho_j\}$ as the greatest lower bound of $\{\rho_j \mid j \in J \;\wedge\; \rho_i \sqsubseteq \rho_I\}$,

- $\sqcup\{\rho_j \mid j \in J\} = \sqcap\{\rho \mid \rho \sqsubseteq \rho_I \ \wedge \ \forall j \in J \ . \ \rho_j \sqsubseteq \rho\}$, as the least upper bound of $\{\rho_j \mid j \in J \ \wedge \ \rho_i \sqsubseteq \rho_I\}$.

Note that $\mathcal{L}(\rho_I)$ is a sublattice of $\mathcal{L}(\{Q\})$, the complete lattice of partitions of Q.

Definition 3.1 *Given an equivalence relation $\rho = \{B_i \mid i \in J\}$ on Q, ρ is* compatible *with T_a if and only if*
$\forall i, j \in J \ . \ \forall p, q \in B_i \ . \ (T_a[p] \cap B_j \neq \emptyset \Leftrightarrow T_a[q] \cap B_j \neq \emptyset)$.

We say that ρ is compatible with T if and only if ρ is compatible with T_a, for each $a \in A$.

Proposition 3.1 *Given an equivalence relation $\rho = \{B_i \mid i \in J\}$ on Q, ρ is compatible with T if and only if it is a bisimulation.*

Proof $T_a[p] \cap B_j \neq \emptyset$ is logically equivalent to $\exists r \in B_j \ . \ p \xrightarrow{a} r$. Thus, we can rewrite the definition of compatibility as follows:
$\forall i, j \in J \ . \ \forall p, q \in B_i \ . \ \forall a \in A$
$\forall r_1 \ . \ (r_1 \in B_j \ \wedge \ p_1 \xrightarrow{a} r_1 \Rightarrow \exists r_2 \in B_j \ . \ p_2 \xrightarrow{a} r_2) \ \wedge$
$\forall r_2 \ . \ (r_2 \in B_j \ \wedge \ p_2 \xrightarrow{a} r_2 \Rightarrow \exists r_1 \in B_j \ . \ p_1 \xrightarrow{a} r_1)$.
By the fact that ρ is a partition, the above property is equivalent to the definition of bisimulation.
$\square$
Let us now consider the relational coarsest partition problem:

> Given a partition ρ of a set Q and a family of binary relations $(T_a)_{a \in A}$ over Q, find the coarsest refinement ρ' of ρ such that ρ' is *compatible* with (T_a) for each $a \in A$.

Since the space of refinements of a partition is a complete lattice, a unique coarsest partition exists. The following proposition gives a characteristic property of compatibility.

Proposition 3.2 *Given an equivalence relation ρ on Q, ρ is compatible with T if and only if*
$\forall a \in A \ . \ \forall B, B' \in \rho \ . \ (B' \subseteq T_a^{-1}[B] \ \vee B' \cap T_a^{-1}[B] \ = \emptyset)$.

Proof. The above property is logically equivalent to the following property:
$\forall a \in A \ . \ \forall B, B' \in \rho \ . \ (B' \cap T_a^{-1}[B] \ \neq \emptyset \Rightarrow B' \subseteq T_a^{-1}[B])$.
Thus, it is easy to prove that this property is equivalent to the definition of compatibility.
$\square$
Proposition 3.2 is used hereafter as a basis for the design of an algorithm computing partitions which are bisimulations. We define the properties π, π_B and $\pi_{a,B}$ for $B \subseteq Q$ and $a \in A$.

$$\begin{aligned}
\pi_{a,B}(\rho) &= \forall X \in \rho \ . \ (X \cap T_a^{-1}[B] = \emptyset \vee X \subseteq T_a^{-1}[B]) \\
\pi_B(\rho) &= \forall a \in A \ . \ \pi_{a,B}(\rho) \\
\pi(\rho_1, \rho_2) &= \forall B \in \rho_1 \ . \ \pi_B(\rho_2)
\end{aligned}$$

The following figure illustrates expression of $\pi(\rho_1, \rho_2)$ in terms of $\pi_B(\rho_2)$ and $\pi_{a,B}(\rho_2)$.

$$\underbrace{\forall B \in \rho_1 \;.\; \overbrace{\forall a \in A \;.\; \overbrace{\forall X \in \rho_2 \;.\; X \cap T_a^{-1}[B] = \emptyset \vee X \subseteq T_a^{-1}[B]}^{\pi_{a,B}(\rho_2)}}^{\pi_B(\rho_2)}}_{\pi(\rho_1,\rho_2)}$$

The property π_B corresponds to Paige's and Tarjan's notion of stability. Note that ρ is compatible with T if and only if $\pi(\rho, \rho)$. For computing such relations we define an operator Φ in the following manner:

- First, for $a \in A$ and $B \subseteq Q$, we define an operator $\Phi_{a,B}$ that refines the partition ρ with respect to the class B and the action a. This operator is such that, for any partition ρ, the property $\pi_{a,B}(\Phi_{a,B}(\rho))$ holds.
- Second, we define an operator Φ_B from the operators $\Phi_{a,B}$, for $a \in A$. This operator is such that, for any partition ρ, the property $\pi_B(\Phi_B(\rho))$ holds.
- Finally, an operator Φ is obtained from the operator Φ_B, for $B \in \rho$. This operator is such that, $\pi(\rho, \Phi(\rho, \rho))$ holds.

Definition 3.2 *For $a \in A$ and $B \subseteq Q$, we define the operator $\Phi_{a,B}$ as follows:*
$\Phi_{a,B}(\rho) = \{X \cap T_a^{-1}[B] \mid X \in \rho\} \cup \{X - T_a^{-1}[B] \mid X \in \rho\}$.

Proposition 3.3 properties of $\Phi_{a,B}$
Let a, a_1 and a_2 be elements of A, B, B_i, B_j subsets of Q and ρ, ρ_1, ρ_2 partitions of Q.

(i) $\Phi_{a,B}(\rho)$ *is a partition of* Q*,*

(ii) monotonicity*:* $\rho_1 \sqsubseteq \rho_2 \Rightarrow \Phi_{a,B}(\rho_1) \sqsubseteq \Phi_{a,B}(\rho_2)$

(iii) $\Phi_{a,B}(\rho)$ *is a refinement of* ρ*,*

(iv) $\Phi_{a_1,B_i} \circ \Phi_{a_2,B_j} = \Phi_{a_2,B_j} \circ \Phi_{a_1,B_i}$*,*

(v) $\pi_{a,B}(\Phi_{a,B}(\rho))$*.*

(vi) $\pi_{a,B}(\rho) \Leftrightarrow \Phi_{a,B}(\rho) = \rho$*.*

Proof. (i) and (iii): An element of ρ either is in $\Phi_{a,B}(\rho)$ or is split into two pieces, each of them belonging to $\Phi_{a,B}(\rho)$.
(ii) holds by properties of set operators.
(iv) Let $X \in (\Phi_{a_1,B_i} \circ \Phi_{a_2,B_j})(\rho)$, $Y \in \rho$ and $X \subseteq Y$. We must consider four cases :
– $X = Y \cap T_{a_1}^{-1}[B_i] \cap T_{a_2}^{-1}[B_j] = Y \cap T_{a_2}^{-1}[B_j] \cap T_{a_1}^{-1}[B_i]$,
– $X = (Y \cap T_{a_1}^{-1}[B_i]) - T_{a_2}^{-1}[B_j] = (Y - T_{a_2}^{-1}[B_j]) \cap T_{a_1}^{-1}[B_i]$,

– $X = (Y - T_{a_1}^{-1}[B_i]) \cap T_{a_2}^{-1}[B_j] = (Y \cap T_{a_2}^{-1}[B_j]) - T_{a_1}^{-1}[B_i]$,
– $X = (Y - T_{a_1}^{-1}[B_i]) - T_{a_2}^{-1}[B_j] = (Y - T_{a_2}^{-1}[B_j]) - T_{a_1}^{-1}[B_i]$.
In the four cases, $X \in (\Phi_{a_2,B_j} \circ \Phi_{a_1,B_i})(\rho)$.Thus,
$\Phi_{a_1,B_i} \circ \Phi_{a_2,B_j}(\rho) \sqsubseteq \Phi_{a_2,B_j} \circ \Phi_{a_1,B_i}(\rho)$. $\Phi_{a_2,B_j} \circ \Phi_{a_1,B_i}(\rho) \sqsubseteq \Phi_{a_1,B_i} \circ \Phi_{a_2,B_j}(\rho)$ follows by symmetric reasoning.

(v) holds by construction of $\Phi_{a,B}$.
(vi) $\Rightarrow$:From (iii), it is sufficient to prove that $\pi_{a,B}(\rho) \Rightarrow \rho \sqsubseteq \Phi_{a,B}(\rho)$. Let $X \in \rho$. Since $\pi_{a,B}(\rho)$, we have $\forall a \in A \,.\, X \subseteq T_a^{-1}[B] \vee X \cap T_a^{-1}[B] = \emptyset$. In the first case, $X \cap T_a^{-1}[B] = X$, and in the second case, $X - T_a^{-1}[B] = X$. Thus, $X \in \Phi_{a,B}(\rho)$.
$\Leftarrow$: This follows from (v). □

Given a subset B of Q and a partition ρ, the sequence of refinements with respect to B and a, for $a \in A$, may be computed in any order (property (iv) of proposition 3.3). We define an operator Φ_B which refines a partition for all $a \in A$, with respect to B.

Definition 3.3 *Let $a_1, ..., a_n$ be the elements of the set A. For $B \subseteq Q$, we define an operator Φ_B such that:*
$\Phi_B = \Phi_{a_1,B} \circ ... \circ \Phi_{a_n,B}$.

Proposition 3.4 properties of Φ_B
Let B, B_1 and B_2 be subsets of Q and ρ, ρ_1, ρ_2 partitions of Q.

(i) $\Phi_B(\rho)$ *is a partition of Q,*

(ii) monotonicity: $\rho_1 \sqsubseteq \rho_2 \Rightarrow \Phi_B(\rho_1) \sqsubseteq \Phi_B(\rho_2)$,

(iii) $\Phi_B(\rho)$ *is a refinement of* ρ,

(iv) $\Phi_{B_1} \circ \Phi_{B_2} = \Phi_{B_2} \circ \Phi_{B_1}$,

(v) $\pi_B(\Phi_B(\rho))$.

(vi) $\pi_B(\rho) \Leftrightarrow \Phi_B(\rho) = \rho$

(vii) $\Phi_{B_1} \circ \Phi_{B_2}(\rho) \sqsubseteq \Phi_{B_1 \cup B_2}(\rho)$,

(viii) $\Phi_{B_1} \circ \Phi_{B_2} \circ \Phi_{B_1 \cup B_2} = \Phi_{B_1} \circ \Phi_{B_2}$,

Proof. (i)–(iv) follow from properties of operator $\Phi_{a,B}$.
(v) $\Phi_B(\rho) = (\Phi_{a_1,B} \circ ... \circ \Phi_{a_i,B} \circ ... \circ \Phi_{a_n,B})(\rho)$, by definition. From the definition of $\Phi_B(\rho)$ and property (v) of proposition 3.3, we get $\pi_{a_1,B}(\Phi_B(\rho))$. Furthermore, from property (iv) of proposition 3.3, we deduce: $\Phi_B(\rho) = (\Phi_{a_i,B} \circ ... \circ \Phi_{a_1,B} \circ ... \circ \Phi_{a_n,B})(\rho)$. Thus, $\forall a \in A \,.\, \pi_{a,B}(\Phi_B(\rho))$.
(vi) $\pi_B(\rho) \Leftrightarrow \forall a \in A \,.\, \pi_{a,B}(\rho) \quad \Leftrightarrow \forall a \in A \,.\, \rho = \Phi_{a,B}(\rho)$, from property (vi) of proposition 3.3. Thus, $\rho = \Phi_B(\rho)$.
(vii) first, we prove that $(\Phi_{a,B_1} \circ \Phi_{a,B_2})(\rho) \sqsubseteq \Phi_{a,B_1 \cup B_2}(\rho)$ (1):

let $X \in (\Phi_{a,B_1} \circ \Phi_{a,B_2})(\rho)$, $Y \in \rho$ and $X \subseteq Y$. We must consider four cases:
$X = Y \cap T_a^{-1}[B_1] \cap T_a^{-1}[B_2] \subseteq Y \cap (T_a^{-1}[B_1] \cup T_a^{-1}[B_2]) = Y \cap (T_a^{-1}[B_1 \cup B_2]) \in \Phi_{a,B_1 \cup B_2}(\rho)$
$X = (Y \cap T_a^{-1}[B_1]) - T_a^{-1}[B_2] \subseteq Y \cap T_a^{-1}[B_1] \subseteq Y \cap T_a^{-1}[B_1 \cup B_2] \in \Phi_{a,B_1 \cup B_2}(\rho)$
$X = (Y \cap T_a^{-1}[B_2]) - T_a^{-1}[B_1] \subseteq Y \cap T_a^{-1}[B_2] \subseteq Y \cap T_a^{-1}[B_1 \cup B_2] \in \Phi_{a,B_1 \cup B_2}(\rho)$
$X = (Y - T_a^{-1}[B_1]) - T_a^{-1}[B_2] = Y - (T_a^{-1}[B_1] \cup T_a^{-1}[B_2]) = Y - T_a^{-1}[B_1 \cup B_2] \in \Phi_{a,B_1 \cup B_2}(\rho)$
then, from the fact that the Φ_{a_i,B_j} commute with each other we have:
$\Phi_{B_1} \circ \Phi_{B_2}(\rho) = \Phi_{a_1,B_1} \circ ... \circ \Phi_{a_n,B_1} \circ \Phi_{a_1,B_2} \circ ... \circ \Phi_{a_n,B_2}(\rho) =$
$\Phi_{a_1,B_1} \circ \Phi_{a_1,B_2} \circ ... \circ \Phi_{a_i,B_1} \circ \Phi_{a_i,B_2} \circ ... \circ \Phi_{a_n,B_1} \circ \Phi_{a_n,B_2}(\rho) \sqsubseteq$
$\Phi_{a_1,B_1 \cup B_2} \circ ... \circ \Phi_{a_i,B_1 \cup B_2} \circ ... \circ \Phi_{a_n,B_1 \cup B_2} = \Phi_{B_1 \cup B_2}(\rho)$ (by monotonicity of Φ_{a_i,B_j} and (1)).

(viii) Let ρ be a partition of Q. From (iii) and (iv), we have
$(\Phi_{B_1} \circ \Phi_{B_2} \circ \Phi_{B_1 \cup B_2})(\rho) \sqsubseteq \Phi_{B_1} \circ \Phi_{B_2}(\rho)$. Conversely, from (ii), (v), (vi) and (vii) $(\Phi_{B_1} \circ \Phi_{B_2})(\rho)$= $(\Phi_{B_1} \circ \Phi_{B_2} \circ \Phi_{B_1} \circ \Phi_{B_2})(\rho) \sqsubseteq (\Phi_{B_1} \circ \Phi_{B_2} \circ \Phi_{B_1 \cup B_2})(\rho)$

□

Given a partition ρ, the sequence of refinements with respect to B, for $B \in \rho$, may be computed in any order (property (iv) of proposition 3.4). When a class B is split into B_1 and B_2, refining with respect to B is useless (property (viii) of proposition 3.4).

Definition 3.4 *Let $\rho = \{B_i \mid 1 \leq i \leq n\}$ be a family of subsets of Q and ρ' be partition of Q. We define an operator Φ such that:*
$\Phi(\rho, \rho') = (\Phi_{B_1} \circ ... \circ \Phi_{B_n})(\rho')$

Proposition 3.5 properties of Φ
Let B be a subset of Q and ρ, ρ_1, ρ_2 partitions of Q.

- *(i)* $\Phi(\rho_1, \rho_2)$ *is a partition of Q,*
- *(ii)* left–monotonicity: $\rho_1 \sqsubseteq \rho_2 \Rightarrow \Phi(\rho_1, \rho) \sqsubseteq \Phi(\rho_2, \rho)$,
- *(iii)* right–monotonicity: $\rho_1 \sqsubseteq \rho_2 \Rightarrow \Phi(\rho, \rho_1) \sqsubseteq \Phi(\rho, \rho_2)$,
- *(iv)* $\Phi(\rho_1, \rho_2)$ *is a refinement of* ρ_2,
- *(v)* $\pi(\rho_1, \Phi(\rho_1, \rho_2))$,
- *(vi)* $\pi(\rho_1, \rho_2) \Leftrightarrow \Phi(\rho_1, \rho_2) = \rho_2$
- *(vii) if* $B \in \rho_1$ *and* $\pi_B(\rho_2)$ *then* $\Phi(\rho_1, \rho_2) = \Phi(\rho_1 - \{B\}, \rho_2)$.

Proof. (i)–(vi) follow from properties of operator Φ_B.
(vii) By definition of Φ, we have: $\Phi(\rho_1, \rho_2) = \Phi_B(\Phi(\rho_1 - \{B\}, \rho_2))$. It is easy to see that: $\forall \rho_1, \rho_2$, partitions of Q $\rho_1 \sqsubseteq \rho_2 \wedge \pi_B(\rho_2) \Rightarrow \pi_B(\rho_1)$. From $\pi_B(\rho_2)$ and $\Phi(\rho_1 - \{B\}, \rho_2) \sqsubseteq \rho_2$, we have $\pi_B(\Phi(\rho_1 - \{B\}, \rho_2))$. From property (vi) of proposition 3.4, we deduce $\Phi_B(\Phi(\rho_1 - \{B\}, \rho_2)) = \Phi(\rho_1 - \{B\}, \rho_2)$.

□

The set of partitions of Q, ordered by refinement is a complete lattice. From the operator Φ, we define hereafter an operator $\tilde{\Phi}$ on the complete lattice of partitions of Q. $\tilde{\Phi}$ is shown to be $\sqcap$–continuous. The maximal fixpoint of $\tilde{\Phi}$, for a given partition ρ_I, is the coarsest refinement of ρ_I compatible with the transition relation.

Proposition 3.6 *Let* $\tilde{\Phi}(X) = \Phi(X, X)$.

(i) monotonicity*:* $\rho_1 \sqsubseteq \rho_2 \Rightarrow \tilde{\Phi}(\rho_1) \sqsubseteq \tilde{\Phi}(\rho_2)$,

(ii) if T^{-1} *is* image-finite *then* $\tilde{\Phi}$ *is* $\sqcap$*–continuous,*

(iii) Given an initial partition ρ*, the maximal fixpoint of* $\tilde{\Phi}$*, i.e. the coarsest partition compatible with* $(T_a)_{a\in A}$ *and* ρ*, is the limit of the sequence:*

$$\begin{aligned} \rho_0 &= \rho \\ \rho_{r+1} &= \tilde{\Phi}(\rho_r) \end{aligned}$$

Proof. We prove the property (ii) $\sqcap$–continuity. By monotonicity of $\tilde{\Phi}$, we have
$\tilde{\Phi}(\sqcap_{i\in I}\rho_i) \sqsubseteq \sqcap_{i\in I}\tilde{\Phi}(\rho_i)$.
Conversely, we prove: $\sqcap_{i\in I}\tilde{\Phi}(\rho_i) \sqsubseteq \tilde{\Phi}(\sqcap_{i\in I}\rho_i)$. Let $x \in \sqcap_{i\in I}\tilde{\Phi}(\rho_i)$, $a \in A$ and $y \in \sqcap_{i\in I}\rho_i$ such that $x \cap T_a^{-1}[y] \neq \emptyset$. We have to prove that $x \subseteq T_a^{-1}[y]$. There exist two decreasing sequences $(x_i)_{i\in I}$ and $(y_i)_{i\in I}$ such that $\forall i \in I \,.\, x_i \in \tilde{\Phi}(\rho_i) \wedge\ \cap_{i\in I}x_i = x$ and $\forall i \in I \,.\, y_i \in \rho_i \wedge \cap_{i\in I}y_i = y$. We have $x \cap T_a^{-1}[y] \neq \emptyset \Leftrightarrow \forall i \in I \,.\, x_i \cap T_a^{-1}[y_i] \neq \emptyset$. By property (v) of proposition 3.5, $\forall i \in I \,.\, x_i \in \tilde{\Phi}(\rho_i) \wedge x_i \cap T_a^{-1}[y_i] \neq \emptyset \wedge y_i \in \rho_i$ implies $x_i \subseteq T_a^{-1}[y_i]$. Furthermore, from the fact that T_a^{-1} is $\cap$–continuous [17], $\cap_{i\in I}x_i \subseteq \cap_{i\in I}T_a^{-1}[y_i] = T_a^{-1}[\cap_{i\in I}y_i] = T_a^{-1}[y]$. We conclude: $x \subseteq T_a^{-1}[y]$.

□

The following proposition establishes that the maximal fixpoints of Ψ (see section 2) and $\tilde{\Phi}$ are the same.

Proposition 3.7 *Given a labeled transition system* $S = (Q,\ A,\ T,\ q_0)$*, the maximal fixpoint* ρ_Ψ *of the operator* Ψ *is the maximal fixpoint* $\rho_{\tilde{\Phi}}$*, where* $\tilde{\rho}_0 = \{Q\}$.

Proof. We have $\rho_\Psi \sqsubseteq \rho_{\tilde{\Phi}}$ since $\pi(\rho_\Psi, \rho_\Psi)$. Conversely, we have $\rho_{\tilde{\Phi}} \sqsubseteq \Psi(\rho_{\tilde{\Phi}})$ thus $\rho_{\tilde{\Phi}} \sqsubseteq \rho_\Psi$.
□

4 Solution

In this section, the Paige & Tarjan algorithm is adapted in order to compute the maximal fixpoint of $\tilde{\Phi}$. The resulting algorithm has the same complexity as the original one. The major difference between the two algorithms lies on the fact that a refinement step, i.e. the computation of Φ_B, is reduced to the computation of $\Phi_{a,B}$ in the original one. In other words, a refinement step of our algorithm consists of repeating the Paige's and Tarjan's refinement step, for each $a \in A$. Let $S = (Q, A, T, q_0)$ be a finite-state labeled transition system, ρ_I be

a partition of Q, $n = \mid Q \mid$ and $m = \mid T \mid$. We suppose that for all a in A, the image set sizes $\mid T_a[p] \mid$ are uniformly bounded by a constant c. The rest of the section is organized as follows: first, we develop an abstract algorithm for computing the maximal fixpoint of $\tilde{\Phi}$. Then, we show how this algorithm can be implemented in $O(mn)$ time, from the properties of Φ, Φ_B and $\Phi_{a,B}$. Finally, from this algorithm we derive the adaptation of the Paige & Tarjan algorithm.
The maximal fixpoint of $\tilde{\Phi}$ is the limit of the sequence:

$$\begin{aligned} \rho_0 &= \rho_I \\ \rho_{r+1} &= \Phi(\rho_r, \rho_r) \end{aligned}$$

From property (v) of proposition 3.5, we have $\pi(\rho_r, \rho_{r+1})$. Thus, from property (viii) of proposition 3.5 the following sequence has the same limit as the previous one:

$$\begin{aligned} \rho_0 &= \rho_I \\ W_0 &= \rho_I \\ \rho_{r+1} &= \Phi(W_r, \rho_r) \\ W_{r+1} &= \rho_{r+1} - \rho_r \end{aligned}$$

We can derive an abstract algorithm for computing this limit:

$W, \rho = \rho_I, \rho_I$
`repeat`
  $W, \rho = \Phi(W, \rho) - \rho, \Phi(W, \rho)$ (1)
`until` $W = \emptyset$

The multiple-assignment prescribes that $\Phi(W, \rho) - \rho$ and $\Phi(W, \rho)$ must be computed before executing the assignments $W = \Phi(W, \rho) - \rho$ and $\rho = \Phi(W, \rho)$. The elements of W are called *splitters*. From the definition of Φ and the properties (v) and (viii) of the proposition 3.4, if an element of W is split into subblocks, we need not partition with respect to B. Thus, we consider $W_1 \cup W$ instead of W, W_1. We can replace the computation of Φ appearing in (1) by the following one:
$W_1 = W$
`for each` $B \in W_1$
 $\rho, W, W_1 = \Phi_B(\rho), \Phi_B(\rho) - \rho - \{B\}, \Phi_B(\rho) \cap W_1 - \{B\}$
From property (iv) of proposition 3.4, the refinements steps may be performed in any order. We can transform the abstract algorithm into the following one:

$W, \rho = \rho_I, \rho_I$
`repeat`
 `choose any` $B \in W$
  `replace` ρ `by` $\Phi_B(\rho)$
  `replace` W `by` $(\Phi_B(\rho) - \rho) \cup (\Phi_B(\rho) \cap W) - \{B\}$

Finally, from the definition of $\Phi_{a,B}$, we obtain the following algorithm in which W and $\Phi_{a,B}$ are computed at the same time:

$W, \rho := \rho_I, \rho_I$

`repeat`

`choose and remove any` B `in` W

$$\left.\begin{array}{l} \texttt{for each } a \in A \\ \left.\begin{array}{l} I_{a,B} = \{X \in \rho \mid X \cap T_a^{-1}[B] \neq \emptyset \ \wedge \ X \not\subseteq T_a^{-1}[B]\} \\ I_{a,B}^{1,2} = \{X \cap T_a^{-1}[B] \mid X \in I_{a,B}\} \cup \{X - T_a^{-1}[B] \mid X \in I_{a,B}\} \\ \rho = \rho - I_{a,B} \cup I_{a,B}^{1,2} \\ W = W - I_{a,B} \cup I_{a,B}^{1,2} \end{array}\right\} \Phi_{a,B}(\rho) \end{array}\right\} \Phi_B(\rho)$$

`until` $W = \emptyset$

This algorithm can be implemented in $O(mn)$ time: W and $\Phi_{a,B}$ can be computed in $O(\mid B \mid + \sum_{q \in B} \mid T_a^{-1}[q] \mid)$ time [15]. The algorithm terminates after at most $n-1$ steps [15]. The total cost of the algorithm is obtained by summing over all blocks B used for refinement and over all elements in such blocks [1,15].

Let us consider the case in which $\mid A \mid = 1$. Paige & Tarjan presented an algorithm that computes the coarsest refinement of ρ in $O(m \log n)$ time and in $O(m)$ space [15]. In order to reduce the size of W, they generalize the Hopcroft's algorithm [1] that minimizes the number of states of a deterministic finite automaton. Intuitively, the basic idea is to keep track of how blocks of the partition are split into subblocks at each refinement step. Thus, a splitter B is either a class (*simple splitter*) or a union of classes (*compound splitter*) such that ρ is stable with respect to B. A splitter is regarded as a *set expression*. Their structures consist of the binary associative operator $\cup$, which operands are either elements of the current partition ρ or further expressions. A *subexpression* is either a splitter or a proper subexpression. If X , Y are unions of classes of the current partition, we write $X \preceq Y$ to mean that X is a subexpression of Y. Notice that an element of ρ occurs at most in one expression of W. We describe hereafter how the computation of $\Phi_{a,B}$ is improved.

- If ρ is stable with respect to a splitter B (i.e. the property $\pi_{a,B}(\rho)$ holds) and $B_1 \subseteq B$, then Hopcroft's "process the smaller half" idea may be exploited in order to perform the refinement step with respect to B_1 and $B - B_1$. From property (v) of proposition 3.3 each set $X \in \rho$ is either a subset of $T_a^{-1}[B]$ or disjoint from it. The refinement step consists of the transformation of ρ with Φ_B by replacing each $X \in \rho \ \wedge \ X \subseteq T_a^{-1}[B]$ by the following sets:

$$\begin{array}{rcl} X_1 & = & (X \cap T_a^{-1}[B_1]) - T_a^{-1}[B - B_1] \\ X_2 & = & (X \cap T_a^{-1}[B - B_1]) - T_a^{-1}[B_1] \\ X_3 & = & X \cap T_a^{-1}[B_1] \cap T_a^{-1}[B - B_1] \end{array} \qquad (2)$$

  X_1 (resp. X_2 and X_3 is the subset of X whose successors are in B_1 (resp. in $B - B_1$ and together in B and $B - B_1$. This decomposition may be obtained by searching through the smaller set only, B_1 say, and using the map $info_B(a,p) \ = \mid T_a[p] \cap B \mid$, for all $p \in Q$. X_1, X_2, X_3, $info_{B_1}$ and $info_{(B-B_1)}$ can be computed in time $\mid T_a[B_1] \mid$. The sets X_1, X_2 and X_3 are computed by applying one of the three following rules:

  (i) if $info_{B_1}(a,p) \ = \ info_B(a,p)$ then $X_1 \ := \ X_1 \cup \{p\}$

(ii) if $info_{B_1}(a,p) = 0$ then $X_2 := X_2 \cup \{p\}$

(iii) if $0 < info_{B_1}(a,p) < info_B(a,p)$ then $X_3 := X_3 \cup \{p\}$

Suppose that B is a compound splitter and B_1 is a subexpression of B. The following code computes $\Phi_{a,B}$ and W:

`compute the maps` $info_{B_1}$ `and` $info_{(B-B_1)}$
`for each set` X such that $X \in \rho \wedge X \subseteq T_a^{-1}[B]$
`replace` X `by`$X_1, X_2 and X_3$ as described in (2)
`update` W`in the following manner`
`if` $X \preceq Y$`then substitute` $(X_1 \cup X_2 \cup X_3)$ `for` X`in` Y
`else add` $X_1 \cup X_2 \cup X_3$ `to` W

- if ρ is unstable with respect to B then the refinement step consists of the transformation of ρ with Φ_B by replacing each $X \in \rho$ by the following sets:

$$\begin{aligned} X_1 &= X \cap T_a^{-1}[B_1] \\ X_2 &= X - T_a^{-1}[B_1] \end{aligned} \qquad (3)$$

Suppose that B is a sample splitter. The following code computes $\Phi_{a,B}$ and W:

`compute the map` $info_B$
`for each set` X such that $X \in \rho \wedge X \not\subseteq T_a^{-1}[B] \wedge X \cap T_a^{-1}[B] \neq \emptyset$
`replace` X `by`$X_1 and X_2$ as described in (3)
`update` W`in the following manner`
`if` $X \preceq Y$`then substitute` $(X_1 \cup X_2)$ `for` X`in` Y
`else add` $X_1 \cup X_2$ `to` W

For the general case in which $\mid A \mid > 1$, the stability is expressed by the property π_B. A refinement step consists in repeating the previous one for each $a \in A$.

4.1 Algorithm

Several data structures are required to represent states, classes, splitters. Each state p points to a list of couples $(a, T_a^{-1}[p])$, where $T_a^{-1}[p]$ is represented as a list. This allows scanning of the set $T_a^{-1}[p]$ in time proportional to its size. Each class of ρ has an associated integer giving its size and points to a list its elements. Each state points to its predecessor in its class (this allows deletion in $O(1)$ time) and to the class containing it. We maintain a set W of *splitters*. The refinement step with respect to B is performed according to (3) in the first case whereas it is performed according to (2) in the second one. A compound splitter B is represented as a binary tree with the $info_B$ map associated with the root, and has B_1 and B_2 as children if $B = B_1 \cup B_2$. For each class, we maintain a piece of information which indicates whether it is in W or it is a leaf of a compound splitter. For each $p \in Q$ and each $a \in A$, we maintain a list of couples $(B, info_B)$which has at most c elements. The space needed for the data structures is $O(m)$. The algorithm consists of repeating the refinement step with respect to B until $W = \emptyset$.

Case 1: B is a class
A refinement step is performed as follows:
Step 1 Remove the element B from W.
For each a in A, perform the following two steps:
Step 2 (compute the set $I = \{X_1 \mid \exists X \in \rho \wedge X_1 = X \cap T_a^{-1}[B] \neq \emptyset\}$). Copy the elements of B into a temporary set B'. For each state p in $T_a^{-1}[B]$ move this p into a new class. (The elements of a same class are moved into the same new class). Make each new class point to its associated old class. During the scan of B', compute the map $info_B$.
Step 3 (update ρ and W). After the step 2, each old class X contains the elements $X - T_a^{-1}[B]$. For each X_1 in I perform the following statements:
If $X = X_1$ (this is performed in $O(1)$ time by the comparison between the numbers of the elements of the old and new classes) make X point to X_1.
For the case $X \neq X_1$, make each element of the new class point to X_1 by scanning X_1, add X_1 to ρ and update W in the following manner: if X is in W then add X_1 to W. If X is a leaf of a compound splitter, it is replaced by a subtree whose root is the new node X_{12} and whose leaves are X and X_1: make X_{12} point to X and X_1 and make X and X_1 point back to X_{12}. (This is performed in O(1) time since the old class points to its father). If X is not in W and X is not a leaf then create a new node X_{12} as previously and add it to W.
Case 2: B is a compound splitter $B_1 \cup B_2$ (suppose that $\mid X_1 \mid \leq \mid X_2 \mid$)
A refinement step is performed as follows:
Step 1 Remove B from W.
For each a in A, perform the following two steps:
Step 2 Compute the maps $info_{B_1}$ by scanning the leaves of B_1. During the same scanning, decrement $info_B$, compute the set $I = \{X \mid X \in \rho \wedge X \subseteq T_a^{-1}[B]\}$ and copy elements of the leaves in a temporary set B'. After scanning all the leaves of B_1, mark $info_{B_2}$ as being $info_B$. If B_1 or B_2 are nodes, add them to W.
Step 3 For each X in I, perform the following statement:
split X in X_1, X_2 and X_3 by using $info_B$ and $info_{B_1}$. if $X = X_i$ (i.e. X is not split) for some $i = 1, 2, 3$, then make X point to X_i else add the non-null classes among X_1, X_2 and X_3 to ρ. Update W in same manner that in the simple case except that if all the classes X_i are non-null, then two nodes X_{123} and X_{23} are created such that X_{123} points to X_1 and X_{23} and X_{23} points to X_2 and X_3.

4.2 Example

Consider the following labeled transition system (Q, A, T, q_0):

- $Q = \{0, 1, 2, 3, 4, 5\}$
- $A = \{a, b, c\}$
- $T_a[0] = \{1\}$, $T_a[1] = \{2\}$, $T_a[2] = \{1\}$
- $T_b[0] = \{3\}$, $T_b[1] = \{3, 4\}$, $T_b[2] = \{4\}$

- $T_c[3] = \{5\}$, $T_c[4] = \{5\}$

We start with universal partition $\mho = \{B_0\}$ where $B_0 = \{0,1,2,3,4,5\}$ and $W = \{B_0\}$. We decompose $info_b$ in $info_{a,B}$, $info_{b,B}$ and $info_{c,B}$. We represent $info_{\alpha,B}$ with its graph.

1. refinement with respect to B_0

   (a) label a
   $T_a^{-1}[B_0] = \{0,1,2\}$
   $B_1 = \{0,1,2\}$
   $B_2 = \{3,4,5\}$
   $info_{a,B_0} = (0,1)(1,1),(2,1)$
   $\rho = \{B_1, B_2\}$
   $W = \{(B_0, B_1, B_2)\}$

   (b) label b
   $T_b^{-1}[B_0] = \{0,1,2\}$
   ρ and W are not modified
   $info_{b,B_0} = (0,1)(1,2),(2,1)$

   (c) label c
   $T_c^{-1}[B_0] = \{3,4\}$
   $B_3 = \{3,4\}$
   $B_4 = \{5\}$
   $info_{a,B_0} = (3,1),(4,1)$
   $\rho = \{B_1, B_3, B_4\}$
   $W = \{(B_0, B_1, B_2),(B_2, B_3, B_4)\}$

2. refinement with respect to (B_0, B_1, B_2)
   $info_{a,B_1} = (0,1)(1,1),(2,1)$, $info_{b,B_1} = \emptyset$, $info_{c,B_1} = \emptyset$
   $info_{a,B_2} = \emptyset$, $info_{b,B_2} = info_{b,B_0}$, $info_{c,B_2} = info_{c,B_0}$
   ρ is not modified and $W = \{(B_2, B_3, B_4)\}$

3. refinement with respect to (B_2, B_3, B_4) $\mid B_4 \mid < \mid B_3 \mid$
   $info_{a,B_4} = info_{b,B_4} = \emptyset$, $info_{c,B_4} = info_{c,B_0}$
   The partition is not modified and $W = \emptyset$.

5 Evaluation

We present measures carried out on experimentation of Aldébaran. Aldébaran [5] is a system for verifying communicating systems, represented by labeled transition systems. It allows the reduction and the comparison of labeled transition systems with respect to the following

equivalence: bisimulation, observational, and acceptance equivalence. Various operations such as parallel compositions of labeled transition systems are also made possible by using different strategies of reductions. The algorithm presented in section 4 allows the reduction of labeled transition systems with hundred thousands of transitions in some minutes.
Aldébaran may be interfaced with other systems which manipulate labeled transition systems. Aldébaran has a sample input format which is a list of triples representing the transition relation. For instance, Aldébaran is interfaced with a LOTOS compiler [7] and a common object code produced by LUSTRE and ESTEREL compilers [4].
Aldébaran is written in C and runs on UNIX. Presently, the limit of the size of a labeled transition system on a SUN 3/60 with 50 Mega-bytes of memory, is one million transitions, because the memory cost of a transition is twenty bytes.
We give an example of reduction carried out by Aldébaran. The reduction is based on observational equivalence. Reduction with respect to observational equivalence consists of transforming the labeled transition system by computing transitive closure of the transition relation labeled by τ [10] and finding the coarsest partition with respect to the transition relation and the universal partition. The example is Milner's problem of scheduling (see [12], page 33). This example is interesting for evaluation purposes because the numbers of states, transitions and equivalence classes grow in the same proportion when the number of tasks increases. We give two specifications in Lotos [7]. We consider a ring of n elementary identical components, called *cyclers*. A cycler specification in Lotos is:

```
process CYCLER[gi, ai, bi, gi+1 ] : noexit :=
    gi ; ai ;
          ((  bi ; gi+1 ; CYCLER[ gi, ai, bi, gi+1])
              []
             ( gi+1 ; bi ; CYCLER[ gi, ai, bi, gi+1]))
endproc
```

A cycler should cycle endlessly as follows: (i) Be enabled by predecessor at gi, (ii) Receive initiation request at ai (iii) Receive termination signal at bi and enable successor at $gi+1$ in either order. We give two specifications of scheduler: the first one is such that the ai and bi are visible whereas in the second one, only the ai are visible. (This last specification expresses that the scheduler is observationally equivalent to $(a_1...a_n)^\omega$). In both cases, we give a table that summarizes the time (in seconds) spent for finding the coarsest partition compatible with the transition relation and the universal partition.

5.1 First specification

```
specification SCHEDULER [a1, ..., an, b1, ..., bn ] : noexit behaviour
 hide g1, ..., gn in
 (cycler[g1, a1, b1, g2]
       |[g1, g2]|
     (
```

```
    ...
   cycler[gi, ai, bi, gi+1]
    |[gi+1]|
    ...
      (cycler[gn, an, bn, g1] ||| g1; stop)
    ...
  ))
 where library cycler endlib
endspec
```

numbers of cyclers	number of states	number of transitions	number of classes	time
2	13	35	9	0.017s
3	37	139	25	0.05s
4	97	453	65	0.26s
5	241	1321	161	0.88s
6	577	3595	385	2.6s
7	1345	9339	897	7.28
8	3073	23465	2049	20.5s
9	6913	57687	4663	56.3s
10	15361	138111	10241	159.8s

5.2 Second specification

```
 specification SCHEDULER [a1, ..., an] : noexit behaviour
  hide g1, ..., gn, b1, ..., bn in
  (cycler[g1, a1, b1, g2]
    |[g1, g2]|
  (
    ...
   cycler[gi, ai, bi, gi+1]
    |[gi+1]|
    ...
      (cycler[gn, an, bn, g1] ||| g1; stop)
    ...
  ))
 where library cycler endlib
endspec
```

numbers of cyclers	number of states	number of transitions	number of classes	time
2	13	35	3	0.01s
3	37	325	4	0.05s
4	97	1465	5	0.15s
5	241	5851	6	0.6s
6	577	21853	7	1.9s
7	1345	78247	8	6.9s
8	3073	272209	9	24s
9	6913	927451	10	80s

Notice that in both cases, time increases quasi linearly with the number of transitions.

6 Conclusion

In this paper, we have formally established a description of bisimulation equivalence in terms of the relational coarsest partition problem. We have presented an adaptation of the Paige & Tarjan algorithm and its implementation. The new algorithm provides an efficient decision procedure for other equivalence relations requiring the computation of bisimulation equivalence.
In practice, this algorithm runs efficiently in the context of the verification of communicating systems in which a state has a few number of successors.

References

[1] A. Aho J. Hopcroft and J. Ullman. *Design and analysis of computer Algorithms.* Addison Wesley, 1974.

[2] T. Bolognesi and M. Caneve. Incremental development of a tool for equivalence verification. In *Protocol Specification, Testing and Verification VIII*, 1988.

[3] T. Bolognesi and S.A. Smolka. Fundamental results for the verification of observational equivalence. In H.Rudin and C.H. West, editors, *Protocol Specification, Testing and Verification VII*, 1987.

[4] P. Couronné, J.A. Plaice, and J.B. Saint. The lustre esterel portable format. *unpublished*, 1986.

[5] J. C. Fernandez. *Aldébaran, Un système de vérification par réduction de processus communicants.* PhD thesis, Université de Grenoble, 1988.

[6] J. C. Fernandez. *Aldébaran: Manuel de l'utilisateur.* Technical Report, LGI-IMAG Grenoble, 1988.

[7] Hubert Garavel. *Compilation et vérification de programmes LOTOS.* PhD thesis, Université Joseph Fourier de Grenoble, 1989.

[8] S. Graf. A complete inference system for an algebra of regular acceptance models. In *Mathematical Foundations of Computer Science*, 1986. LNCS, 233.

[9] M. Hennessy. Acceptance trees. *JACM*, 4, 1985.

[10] P. Kanellakis and S. Smolka. Ccs expressions, finite state processes and three problems of equivalence. In *Proceedings ACM Symp. on Principles of Distribued Computing*, 1983.

[11] K.G. Larsen. *Context–Dependent Bisimulation Between Processes.* Technical Report CST-37-86, Departement of Computer Science, University of Edimburgh, May 1986. Ph.D.

[12] R. Milner. A calculus of communication systems. In *LNCS 92*, Springer Verlag, 1980.

[13] E.R. Olderog. Specification oriented programming in tcsp. *Logics and Models of Concurrent Systems*, 13, 1985.

[14] D. Park. Concurrency and automata on infinite sequences. In *Theorical Computer Science, 5th G1-Conference*, Springer Verlag, 1985. LNCS 104.

[15] R. Paige and R. Tarjan. Three partition refinement algorithms. *SIAM J. Comput., No. 6*, 16, 1987.

[16] M. Sanderson. *Proof Techniques for CCS.* PhD thesis, University of Edimburgh, 1982. CST-19-82.

[17] J. Sifakis. A unified approach for studying the properties of transition systems. *TCS*, 3, 1982.