Aldébaran: A Tool for Verification of Communicating Processes

Jean-Claude Fernandez [*]

Abstract

Aldébaran is a tool for verifying communicating systems, represented as labeled transition systems. Verification techniques are based on the comparison of two labeled transition systems according to an equivalence relation. Strong bisimulation, weak bisimulation, acceptance model equivalence and safety equivalence are supported by Aldébaran. Communicating systems are described hierarchically by parallel composition of processes. Synchronization (or communications) between labeled transition systems set in parallel are determined by a synchronization algebra. To allow partial synchrony, a restriction operator is defined. To verify external specification, an abstract mechanism is used. A major goal of the tool is to provide different equivalence relations and efficient algorithms implementing these.

1 Introduction

Aldébaran is a tool for verifying communicating processes, represented by labeled transition system. It is intended to be useful in the verification of communicating finite state processes by model checking or by means of equivalence relations. In the first case, one check that a given program, viewed as a labeled transition system over which a temporal logic formula is interpreted, satisfies it. In the second one, semantics of process algebra is usually given as a transition relation on terms together with a congruence relation on terms that is preserved by a transition relation. Aldébaran allows the reduction and the comparison of labeled transition systems with respect to the following equivalence: strong bisimulation [12], [15], weak bisimulation (or observational equivalence) [12], acceptance model equivalence [9] and safety equivalence [16]. Let $\sim_R$ be one of these equivalences. For each labeled transition system, a normal form with respect to $\sim_R$ may be obtained by applying a set of transformations on the labeled transition system. Thus, two labeled transition systems are equivalents, with respect to $\sim_R$, if and only if they have the same normal form. Since these equivalences are weaker than the strong bisimulation, the normal form may be chosen minimal in number of states.

[*] Tel : 76 51 49 15 e-mail `fernand@imag.imag.fr`

Thus, an efficient algorithm computing strong bisimulation reveals itself quite useful. It is an adapted version of Paige & Tarjan algorithm that refines a partition until it is preserved by transition relation. For example, initial partition may be determined by state predicates. Another feature of Aldébaran is an attempt for controlling state explosion that occurs in parallelism. We propose a method for minimizing space requirement translating a program into a labeled transition system.

Let $States$ be a set of states, A a set of names, τ a particular name, not in A, which represents an internal or hidden action. A_τ denotes the set $A \cup \{\tau\}$. A labeled transition system is a quadruple $S = (Q, A_\tau, T, q_0)$ where Q is the subset of $States$ reachable from q_0 with respect to T (i.e if q is reachable from q_0 with respect to T and $(q, a, q') \in T$ then q' is reachable from q_0 with respect to T). $T \subseteq Q \times A_\tau \times Q$ the transition relation and q_0 the initial state. For each label a and each state q, we consider the image set $T_a[q] = \{q' \in Q \mid (q, a, q') \in T\}$. We extend this notation for sets of states: $T_a[B] = \cup\{T_a[q] \mid q \in B\}$. T^{-1} denotes the inverse relation $T^{-1} = \{(q', a, q) \mid (q, a, q') \in T\}$. $T_a^{-1}[q]$ and $T_a^{-1}[B]$ denotes respectively the image set of q by T_a^{-1} and the image set of B by T_a^{-1}. We also use the notation $p \xrightarrow{a}_T q$ for $(p, a, q) \in T$. We consider a particular class of labeled transition systems: the regular labeled transition systems.

This paper is organized as follows: in part 2, we describe the set of transformations on the labeled transition systems for strong bisimulation, observational equivalence, acceptance model equivalence and safety equivalence in order to obtain normal form. In part 3, we present our adaptation of the Paige & Tarjan Algorithm that computes the relational coarsest partition problem for a labeled transition system. In part 4, three operators on the labeled transition systems are defined: parallel composition, restriction [12] and abstraction [3] for designing communicating systems. A communicating system is defined as an expression whose constructors are labeled transition systems, parallel composition, restriction and abstraction. In part 5, we presents functionalities of Aldébaran.

2 Equivalences on Labeled Transitions Systems

We recall the definitions of four equivalences on regular labeled transition systems: strong bisimulation, observational equivalence, acceptance model equivalence and safety equivalence. For each of them, a set of transformations may be defined in order to obtain a normal form, which is a canonical element of an equivalence class. Each equivalence relation $\sim_R$ defined on a labeled transition system may be extended to an equivalence relation between labeled transition systems in the following manner: let $S_i = (Q_i, A_\tau, T_i, q_i)$, for $i = 1, 2$ be two labeled transition systems such that $Q_1 \cap Q_2 = \emptyset$ (if it is not the case, this condition may be easily obtained by renaming the set of the states of one labeled transition system). $S = (Q_1 \cup Q_2 \cup \{q_0\}, A_\tau, T_1 \cup T_2, q_0)$ is the labeled transition system obtained from the union of S_1 and S_2. If $q_1 \sim_R q_2$ then $S_1 \sim_R S_2$. Furthermore, we have the following property: if $NF_{\sim_R}(S)$ denotes the normal form of S with respect to $\sim_R$ then $S_1 \sim_R S_2$ if and only if

$NF_{\sim_R}(S_1) \sim NF_{\sim_R}(S_2)$ where $\sim$ denotes the strong bisimulation (see definition latter). In fact, $\sim_R$ and $\sim$ agree on normal forms.

Before defining these equivalences, we give some notations and a definition. Let $S = (Q, A_\tau, T, q_0)$ be a labeled transition system, P be a subset of Q and B be a subset of A_τ

- For a set X, $\mid X \mid$ denotes the number of elements of X,
- for a set X, 2^X denotes the set of subsets of X,
- B^* denotes the set of strings over B,
- $p \overset{a_1 \dots a_n}{\longrightarrow}_T q$ abbreviates that there exist $p_1 \dots p_n$ such that $p = p_0$, $p_i \overset{a_{i+1}}{\longrightarrow}_T p_{i+1}$, $0 \le i < n$ and $p_n = q$,
- $p \overset{\epsilon}{\Longrightarrow}_T q$ abbreviates $p \overset{\tau^n}{\longrightarrow}_T q$, for $n \le 0$,
- $p \overset{a}{\Longrightarrow}_T q$ abbreviates $p \overset{\epsilon}{\Longrightarrow}_T p_1 \overset{a}{\longrightarrow}_T q_1 \overset{\epsilon}{\Longrightarrow}_T q$,
- $init(p, T) = \{a \in A \mid \exists q \in Q \ \wedge \ p \overset{a}{\longrightarrow}_T q\}$,
- $init_\tau(p, T) = \{a \in A_\tau \mid \exists q \in Q \ \wedge \ p \overset{a}{\longrightarrow}_T q\}$,
- $init(P, T) = \cup_{p \in P}\{init(p, T)\}$,
- $c(p, T) = \{q \mid p \overset{\epsilon}{\Longrightarrow}_T q\}$.

Definition 2.1 *Let ρ and ρ' be partition of Q. ρ' is a refinement of ρ, $\rho' \subseteq \rho$ if and only if $\forall B' \in \rho' \,.\, \exists B \in \rho \,.\, (B' \subseteq B)$.*

For each equivalence, we recall its definition, characteristic property in order to obtain the normal form of a labeled transition system, its extension to labeled transition systems and the definition of the normal form.

2.1 Strong Bisimulation

Intuitively, two states p and q are bisimilars 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 action a such that p' and q' are bisimilars.

Definition 2.2 *Given a labeled transition system $S = (Q, A_\tau, T, q_0)$, a binary relation $\rho \subseteq Q \times Q$ is a* bisimulation *if and only if:*
$\forall (p_1, p_2) \in \rho \,.\, \forall a \in A_\tau \,.$
$\forall r_1 \,.\, (p_1 \overset{a}{\longrightarrow}_T r_1 \Rightarrow \exists r_2 \,.\, (p_2 \overset{a}{\longrightarrow}_T r_2 \ \wedge \ (r_1, r_2) \in \rho)) \ \wedge$
$\forall r_2 \,.\, (p_2 \overset{a}{\longrightarrow}_T r_2 \Rightarrow \exists r_1 \,.\, (p_1 \overset{a}{\longrightarrow}_T r_1 \ \wedge \ (r_1, r_2) \in \rho))$

The set of bisimulations on Q, ordered by inclusion has a maximal element denoted by $\sim$ which may be obtained as the limit of a decreasing sequence of relations $\sim_i$, for $i \in N$ [12] ($\sim = \cap_{i \in N} \sim_i$):

- $\forall p, q \in Q \,.\, p \sim_0 q$,
- $p_1 \sim_{i+1} p_2$ if and only if $\forall a \in A_\tau \,.$
  $\forall r_1 \,.\, (p_1 \xrightarrow{a}_T r_1 \Rightarrow \exists r_2 \,.\, (p_2 \xrightarrow{a}_T r_2 \;\wedge\; r_1 \sim_i r_2)) \wedge$
  $\forall r_2 \,.\, (p_2 \xrightarrow{a}_T r_2 \Rightarrow \exists r_1 \,.\, (p_1 \xrightarrow{a}_T r_1 \;\wedge\; r_1 \sim_i r_2))\}$.

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

Proof. By the fact that if $\sim_i$ is an equivalence relation then $\sim_{i+1}$ is an equivalence relation. $\square$

The connection with the relational coarsest partition problem was established in [4], [6] and [11]. Let ρ be a partition of Q. ρ is *compatible* with T if and only if the following property π holds:
$\pi(\rho) : \forall a \in A_\tau \,.\, \forall B, B' \in \rho \,.\, (B' \subseteq T_a^{-1}[B] \;\vee\; B' \cap T_a^{-1}[B] \; = \emptyset)$.
Given an initial partition ρ_0, the set of partition ordered by refinement as a maximal element, which may be obtained as the maximal fixed-point of an operator Φ [6]:
$\Phi(\rho_0) = \cup\{\rho \mid \rho \subseteq \rho_0 \;\wedge\; \pi(\rho)\}$. Thus, $\sim = \Phi(\sim_0)$.

Definition 2.3 *Let* $S = (Q, A_\tau, T, q_0)$ *be a labeled transition system and* ρ *an equivalence relation which is a bisimution, the quotient labeled transition system denoted by* S/ρ *is defined as follows:*
$S/\rho \;=\; (Q/\rho, A_\tau T/\rho, q_0)$ *where:*

- Q/ρ *is the set of equivalence classes,* $Q/\rho \;=\; \{B \mid\subseteq Q, \forall p, q \in B \,.\, (p, q) \in \rho)$
- $(B', a, B) \in T/\rho$ *if and only if* $T_a^{-1}[B] \cap B' \neq \emptyset$
- $[q_0]$ *is the equivalence class of* q_0.

The T/ρ *definition is correct by the fact that* ρ *is compatible with* T.

Definition 2.4 *Let* $S = (Q, A_\tau, T, q_0)$ *be a labeled transition system. The normal form* $NF_\sim(S)$ *of* S *with respect to* $\sim$ *is* $S/\sim$.

Proposition 2.2 *Let* $S_i = (Q_i, A_\tau, T_i, q_i)$, *for* $i = 1, 2$ *be two labeled transition systems.*

$S_1 \sim S_2$ if and only if $NF_\sim(S_1) \;\; \sim \;\; NF_\sim(S_2)$.

Proof : By transitivity of $\sim$. $\square$

2.2 Observational Equivalence

Observational Equivalence is based on the idea of unobservable (or internal) action [12], labeled by τ. From the notion of bisimulation, we can derive a weaker bisimulation in the following manner [12]:

Definition 2.5 *Given a labeled transition system $S = (Q, A_\tau, T, q_0)$, a binary relation $\rho \subseteq Q \times Q$ is a* weak bisimulation *if and only if:*
$\forall (p_1, p_2) \in \rho \,.\, \forall a \in A_\tau^* \,.$
$\forall r_1 \,.\, (p_1 \overset{a}{\Longrightarrow}_T \; r_1 \Rightarrow \exists r_2 \,.\, (p_2 \overset{a}{\Longrightarrow}_T \; r_2 \;\wedge\; (r_1, r_2) \in \rho)) \;\wedge$
$\forall r_2 \,.\, (p_2 \overset{a}{\Longrightarrow}_T \; r_2 \Rightarrow \exists r_1 \,.\, (p_1 \overset{a}{\Longrightarrow}_T \; r_1 \;\wedge\; (r_1, r_2) \in \rho))$

As in the case of strong bisimulation equivalence, the observational equivalence $\approx$ may be obtained as the limit of a decreasing sequences of relations $\approx_i$, for $i \in N$ [12] ($\approx = \cap_{i \in N} \approx i$) by replacing in the definition $\xrightarrow{a}_T$ by $\overset{a}{\Longrightarrow}_T$, for $a \in A_\tau^*$:

- $\forall p, q \in Q \,.\; p \approx_0 q$,
- $p_1 \approx_{i+1} p_2$ if and only if $\forall a \in A_\tau^* \,.$
  $\forall r_1 \,.\, (p_1 \overset{a}{\Longrightarrow}_T \; r_1 \Rightarrow \exists r_2 \,.\, (p_2 \overset{a}{\Longrightarrow}_T \; r_2 \;\wedge\; r_1 \approx_i r_2)) \wedge$
  $\forall r_2 \,.\, (p_2 \overset{a}{\Longrightarrow}_T \; r_2 \Rightarrow \exists r_1 \,.\, (p_1 \overset{a}{\Longrightarrow}_T \; r_1 \;\wedge\; r_1 \approx_i r_2))\}$.

We introduce now the notion of *pre-normal form* from which the normal form is obtained by minimizing the number of states.

Definition 2.6 *(Pre-normal form) Let $S = (Q, A_\tau, T, q_0)$ be a labeled transition system. A* pre-normal form *of S with respect to $\approx$, $PNF_\approx(S)$, may be obtained in the following manner: $PNF_\approx(S) = (Q, A_\tau, T', q_0)$ where*

$q \xrightarrow{a}_{T'} q'$ *if and only if* $q \overset{a}{\Longrightarrow}_T q'$

Proposition 2.3 $S \;\; \approx \;\; PNF_\approx(S)$.

Proof By definition of $\approx$. $\square$

Proposition 2.4 *$PNF_{\approx}(S_1) \ \sim \ PNF_{\approx}(S_2)$ if and only if $S_1 \ \approx \ S_2$.*

Proof $\Rightarrow$:
By the fact that $\approx \subseteq \sim$ [12] we have $S_1 \approx PNF_{\approx}(S_1) \approx PNF_{\approx}(S_2) \approx S_2$.
$\Leftarrow$:
Conversely, we have to prove that $q_1 \ \approx \ q_2$, where q_1 and q_2 are considered respectively as initial state of S_1 and S_2, implies $q_1 \ \sim \ q_2$ where q_1 and q_2 are considered respectively as initial state of $PNF_{\approx}(S_1)$ and $PNF_{\approx}(S_2)$. By induction on $i \in N$: suppose that $p_1 \approx_i p_2$, where p_1 and p_2 are considered respectively as states of S_1 and S_2, implies $p_1 \sim_i p_2$, where p_1 and p_2 are considered respectively as states of $PNF_{\approx}(S_1)$ and $PNF_{\approx}(S_2)$, and suppose that $p_1 \approx_{i+1} p_2$, where p_1 and p_2 are considered respectively as states of S_1 and S_2. We have to prove that $p_1 \sim_{i+1} p_2$, where p_1 and p_2 are considered respectively as states of $PNF_{\approx}(S_1)$ and $PNF_{\approx}(S_2)$. By definition of $\approx_{i+1}$, we have $\forall a \in A_\tau^*$.
$\forall r_1 . (p_1 \overset{a}{\Longrightarrow}_{T_1} \ r_1 \Rightarrow \exists r_2 . (p_2 \overset{a}{\Longrightarrow}_{T_2} \ r_2 \ \wedge \ r_1 \approx_i r_2)) \ \wedge$
$\forall r_2 . (p_2 \overset{a}{\Longrightarrow}_{T_2} \ r_2 \Rightarrow \exists r_1 . (p_1 \overset{a}{\Longrightarrow}_{T_1} \ r_1 \ \wedge \ r_1 \approx_i r_2))$.
By induction hypothesis and definition of pre-normal form we have $\forall a \in A_\tau^*$.
$\forall r_1 . (p_1 \xrightarrow{a}_{T'_1} \ r_1 \Rightarrow \exists r_2 . (p_2 \xrightarrow{a}_{T'_2} \ r_2 \ \wedge \ r_1 \sim_i r_2)) \ \wedge$
$\forall r_2 . (p_2 \xrightarrow{a}_{T'_2} \ r_2 \Rightarrow \exists r_1 . (p_1 \xrightarrow{a}_{T'_1} \ r_1 \ \wedge \ r_1 \sim_i r_2))$. $\Box$

We define the normal form with respect to observational equivalence just as the composition of pre-normal form and normal form with respect to strong bisimulation.

Definition 2.7 *(normal form) Let $S = (Q, A_\tau, T, q_0)$ be a labeled transition system. A* normal form *of S with respect to $\approx$ is given by $NF_{\approx}(S) = (NF_{\sim} \ o \ PNF_{\approx})(S)$.*

We obtain the key property for observational equivalence: two labeled transition systems are equivalent with respect to observational equivalence if and only if their normal form are equivalent with respect to strong bisimulation.

Proposition 2.5 *$NF_{\approx}(S_1) \ \sim \ NF_{\approx}(S_2)$ if and only if $S_1 \ \approx \ S_2$*

Proof By proposition 2.4 and transitivity of $\sim$. $\Box$

2.3 Acceptance model equivalence

In this section, we present acceptance model [9],[10]. These model are used as a semantic domain for Theorical CSP [5].

Definition 2.8 *An* acceptance model *[10] is a subset of $A \times 2^{A_\tau}$ satisfying the following properties (denote by m_s the* acceptance set *$m_s \ = \ \{X \mid (s, X) \in m\}$).*

- $\forall (s,X) \in A \times 2^{A_\tau} . \forall a \in \ . X[(s,X) \in m \Rightarrow \exists Y \in 2^{A_\tau} \ \wedge \ (sa,Y) \in m]$,
- $\forall a \in A . \forall s \in A^* . \forall Y \in 2^{A_\tau} . [(sa,Y) \in m \Rightarrow \exists X \in 2^{Act} . [a \in X \ \wedge \ (s,X) \in m]]$,
- $\forall s \in A^* . m_s$ *is finite,*
- *if* $s \neq \epsilon$ *then* $\forall X . (X \in m_s \ \Rightarrow \ \tau \notin X)$,
- $\mid \{X \in m_\epsilon \mid \tau \notin X\} \mid \leq 1$,
- *if* $\exists X \in m_\epsilon \tau \in X$ *then* $\emptyset \notin m_\epsilon$.

We can associate an acceptance model to a state of a labeled transition system.

Definition 2.9 *Let* $S = (Q, A_\tau, T, q_0)$ *be a labeled transition system and* p *a state of* Q. *An acceptance model* m *may be obtained in the following manner:*
$m(p,T) \ = \ \{(s,X) \mid \exists q \in Q \ \wedge \ p \overset{s}{\Longrightarrow}_T q \ \wedge$
$(\text{init}(q,T) \neq \emptyset \ \vee \ \text{init}(c(q,T),T) = \emptyset) \ \wedge \ X = \text{init}(q,T)\} \cup \{(\epsilon, \text{init}_\tau(p,T))\}$

The acceptance model associated with S *is* $m(q_0,T)$. *Thus, two labeled transition systems* $S_i = (Q_i, A_\tau, T_i, q_i)$ *for* $i = 1,2$ *are equivalent with respect to acceptance model if and only if* $m(q_1,T_1) \ = \ m(q_2,T_2)$.

It is well known that acceptance model equivalence is not a (weak) bisimulation. We can characterize acceptance model equivalence as the limit of a decreasing sequence.

Definition 2.10 *Let* $S = (Q, A_\tau, T, q_0)$ *be a labeled transition system. For* $i \in N$, *we define relation* $\approx_i^{ac}$ *by*

- $\forall p,q \in Q . p \approx_0^{ac} q$,
- $p_1 \approx_i^{ac} p_2$ *if and only if* $\forall s \in A^* . \ \mid s \mid < i \Rightarrow$
  $[(\forall r_1 . (p_1 \overset{s}{\Longrightarrow}_T \ r_1 \ \wedge \ (\text{init}(r_1,T) \neq \emptyset \ \vee \ \text{init}(c(r_1,T),T) = \emptyset)) \Rightarrow$
  $\exists r_2 . (p_2 \overset{s}{\Longrightarrow}_T \ r_2 \ \wedge \ \text{init}(r_1,T) = \text{init}(r_2,T))) \ \wedge$
  $(\forall r_2 . (p_2 \overset{s}{\Longrightarrow}_T \ r_2 \ \wedge \ (\text{init}(r_2,T) \neq \emptyset \ \vee \ \text{init}(c(r_2,T),T) = \emptyset)) \Rightarrow$
  $\exists r_1 . (p_1 \overset{s}{\Longrightarrow}_T \ r_1 \ \wedge \ \text{init}(r_1,T) = \text{init}(r_2,T)))]$.
- $\approx^{ac} = \cap_{i \in N} \approx_i^{ac}$

Proposition 2.6 *Let* $S = (Q, A_\tau, T, q_0)$ *be a labeled transition system and* p, $q \in Q$.

$$m(p,T) \;=\; m(q,T) \text{ if and only if } p \;\approx^{ac}\; q.$$

Proof By definition of $m(p,T)$ and $\approx^{ac}$. $\Box$

As for the observational equivalence, we introduce a pre-normal form from which the normal form is obtained by minimizing the number of states.

Definition 2.11 *Let $S = (Q, A_\tau, T, q_0)$ be a labeled transition system. A* pre-normal form *of S with respect to $\approx^{ac}$, $PNF_{\approx^{ac}}(S)$, may be obtained in the following manner:*
$PNF_{\approx}(S) = (Q', A_\tau, T', q')$ where

- $\tau \in \mathrm{init}_\tau(q_0, T)$ *if and only if* $q' \notin Q \;\wedge\; q' \in Q' \;\wedge\; Q' \subseteq Q \cup \{q'\}$,
- $\tau \notin \mathrm{init}_\tau(q_0, T)$ *if and only if* $q' = q_0 \;\wedge\; Q' \subseteq Q$,
- $\mathrm{init}_\tau(q', T') \;=\; \mathrm{init}_\tau(q_0, T)$,
- $q \in c(q_0, T) \;\wedge\; (\mathrm{init}(q_0, T) \neq \emptyset \;\vee\; \mathrm{init}(c(q_0, T), T) = \emptyset)$ *if and only if* $q' \xrightarrow{\tau}_{T'} q$,
- $q \xrightarrow{a}_{T'} q_1$ *if and only if* $q \xrightarrow{a\tau^*}_{T} q_1 \;\wedge\; (\mathrm{init}(q_1, T) \neq \emptyset \;\vee\; \mathrm{init}(c(q_1, T) = \emptyset)$,
- $(q' \xrightarrow{\tau}_{T'} q_1 \;\wedge\; q_1 \xrightarrow{a}_{T'} q_2 \;\wedge\; q' \xrightarrow{a}_{T'} q_3)$ *if and only if* $q_1 \xrightarrow{a}_{T'} q_3 \;\wedge\; q' \xrightarrow{a}_{T'} q_2)$,
- $(p \xrightarrow{a_1}_{T'} p_1 \;\wedge\; p_1 \xrightarrow{a_2}_{T'} p_2 \;\wedge\; p \xrightarrow{a_1}_{T'} q_1 \;\wedge\; q_1 \xrightarrow{a_2}_{T'} q_2)$ *if and only if* $p_1 \xrightarrow{a_2}_{T'} q_2 \;\wedge\; q_1 \xrightarrow{a_2}_{T'} p_2)$.

Lemma 2.1 *Let $S = (Q, A_\tau, T, q_0)$ be a labeled transition system and $PNF_{\approx^{ac}}(S) = (Q', A_\tau, T', q')$ its pre-normal form. Then, $\forall q \in Q'$. $[q \neq q' \;\Rightarrow\; \mathrm{init}(q, T) = \mathrm{init}(q, T')]$.*

Proof First, we have by definition of pre-normal form : $\mathit{init}(q, T') \subseteq \mathit{init}(q, T)$.
Conversely, let $a \in \mathit{init}(q, T)$. $\exists q_1 \in Q$ such that $q \xrightarrow{a\tau^*}_{T} q_1$.
If $\mathit{init}(q_1, T) \neq \emptyset \;\vee\; \mathit{init}(c(q_1, T), T) = \emptyset$ then $q \xrightarrow{a}_{T'} q_1$ and consequently, $a \in \mathit{init}(q, T')$.
If $\mathit{init}(q_1, T) = \emptyset \;\wedge\; \mathit{init}(c(q_1, T), T) \neq \emptyset$ then $\exists q_2$ such that $\mathit{init}(q_2, T) \neq \emptyset \;\wedge\; q_1 \xrightarrow{\tau^*}_{T} q_2$.
Thus $q \xrightarrow{a}_{T'} q_2$ and consequently, $a \in \mathit{init}(q, T')$.
$\Box$

Proposition 2.7 $S \;\approx^{ac}\; PNF_{\approx^{ac}}(S)$.

Proof By definition of $\approx^{ac}$ and pre-normal form.
$\Box$

Proposition 2.8 *Let $S_i = (Q_i, A_\tau, T_i, q_i)$, for $i = 1, 2$ be two labeled transition systems and $PNF_{\approx^{ac}}(S_i) = (Q'_i, A, T'_i, q'_i)$ for $i = 1, 2$.*
$PNF_{\approx^{ac}}(S_1) \sim PNF_{\approx^{ac}}(S_2)$ if and only if $S_1 \approx^{ac} S_2$.

Proof
$\Rightarrow$: By transitivity of $\approx^{ac}$.
$\Leftarrow$: By induction on N, we prove that $q'_1 \approx^{ac} q'_2 \Rightarrow q'_1 \sim_i q'_2$.

- if $init_\tau(q'_1, T'_1) = init_\tau(q'_2, T'_2) = \emptyset$ then they are no transitions from q'_1 and q'_2. Thus, $S_1 \approx^{ac} S_2$.
- For $i = 0$, the proof is obvious.
- Suppose that $i > 0$. By definition of the normal form, we have $init_\tau(q'_1, T'_1) = init_\tau(q'_2, T'_2)$. Let $a \in A_\tau$ and $p_1 \in Q'_1$ such that $q'_1 \xrightarrow{a}_{T'_1} p_1$. We want to prove that $\exists p_2 \in Q'_2$ such that $q'_2 \xrightarrow{a}_{T'_2} p_2$ and $p_1 \approx^{ac} p_2$. Indeed, if a such p_2 exists, then from induction hypothesis we deduce that $p_1 \sim_i p_2$. Similarly, we prove the symmetrical condition and we obtain that $q'_1 \sim_{i+1} q'_2$.
- We prove that $\exists p_2$ such that $q'_2 \xrightarrow{a}_{T'_2} p_2$ and $p_1 \approx^{ac} p_2$. From $q'_1 \approx^{ac} q'_2$ and $q'_1 \xrightarrow{a}_{T'_1} p_1$ we deduce that $\exists p_2$ such that $q'_2 \xrightarrow{a}_{T'_2} p_2$ and $init(p_1, T'_1) = init(p_2, T'_2)$. If $init(p_1, T'_1) = \emptyset$, then $init(p_2, T'_2) = \emptyset$. In this case, we have $p_1 \approx^{ac} p_2$.
  If $init(p_1, T'_1) \neq \emptyset$ then $\exists s \in A^+$ and $r_1 \in Q'_1$ such that $p_1 \xrightarrow{s}_{T'_1} r_1$. Thus, $\exists r_2 \in Q'_2$ such that $q'_2 \xrightarrow{as}_{T'_2} r_2$ and $init(r_1, T'_1) = init(r_2, T'_2)$. Let $p'_2 \in Q'_2$ such that $q'_2 \xrightarrow{a}_{T'_2} p'_2$ and $p'_2 \xrightarrow{s}_{T'_2} r_2$. Since $init(p_1, T'_1) = init(p_2, T'_2)$, by construction of pre-normal form, we have $p_2 \xrightarrow{s}_{T'_2} r_2$. Thus, $p_1 \approx^{ac} p_2$.

□

Definition 2.12 *(normal form) Let $S = (Q, A_\tau, T, q_0)$ be a labeled transition system. A* normal form *of S with respect to $\approx^{ac}$ $PNF_{\approx^{ac}}(S) = NF_\sim$ o $PNF\approx^{ac}$*

Proposition 2.9 *$NF_{\approx^{ac}}(S_1) \sim NF_{\approx^{ac}}(S_2)$ if and only if $S_1 \approx^{ac} S_2$*

Proof Obvious. □

2.4 Safety equivalence

Safety equivalence was introduced in [16] and is an equivalence relation that preserves the *safety* property. This equivalence is interesting in connection with a temporal logic.

Definition 2.13 *Let $S = (Q, A_\tau, T, q_0)$ be a labeled transition system and let $\rho \subseteq Q \times Q$. Then ρ is a safety preorder if ρ satisfies the following property:*
$\forall (p_1, p_2) \in \rho \,.\, \forall a \in A$.
$\forall r_1 \,.\, (p_1 \xrightarrow{\tau^* a}_T r_1 \Rightarrow \exists r_2 \,.\, (p_2 \xrightarrow{\tau^* a}_T r_2 \wedge (r_1, r_2) \in \rho)) \wedge$
$\forall r_1 \,.\, (p_1 \overset{\epsilon}{\Longrightarrow}_T r_1 \Rightarrow (r_1, p_2) \in \rho)$

The set of safety-preorder on Q, ordered by inclusion has a maximal element denoted by $\sqsubseteq^{saf}$ which may be obtained as the limit of a decreasing sequence of relation $\sqsubseteq_i^{saf}$, for $i \in N$ [16]:

- $\forall p, q \in Q \,.\, p \sqsubseteq_0^{saf} q$,
- $p_1 \sqsubseteq_{i+1}^{saf} p_2$ if and only if $\forall a \in A_\tau$.
  $\forall r_1 \,.\, (p_1 \xrightarrow{\tau^* a}_T r_1 \Rightarrow \exists r_2 \,.\, (p_2 \xrightarrow{\tau^* a}_T r_2 \wedge r_1 \sqsubseteq_i^{saf} r_2)) \wedge$
  $\forall r_1 \,.\, (p_1 \overset{\epsilon}{\Longrightarrow}_T r_1 \Rightarrow r_1 \sqsubseteq_i^{saf} r_2)$.

Proposition 2.10 *Let $S = (Q, A_\tau, T, q_0)$ be a labeled transition system. Then*

$$p \xrightarrow{\tau^*}_T q \;\Rightarrow\; q \sqsubseteq^{saf} p.$$

Proof By induction on $i \in N$. Suppose that $p \xrightarrow{\tau^*}_T q \;\Rightarrow\; q \sqsubseteq_i^{saf} p$, for $i > 0$ (the case $i = 0$ is obvious). If $p \xrightarrow{\tau^*}_T q$ then $\forall a \in A_\tau \,.\, \forall r \in Q \,.\, q \xrightarrow{\tau^* a}_T r \;\Rightarrow\; p \xrightarrow{\tau^* a}_T r$. Furthermore, we have $r \sqsubseteq_i^{saf} r$. If $q \xrightarrow{\tau^*}_T r$ then $p \xrightarrow{\tau^*}_T r$ and $r \sqsubseteq_i^{saf} p$, by induction hypothesis. Thus, by definition of $\sqsubseteq_{i+1}^{saf}$ we deduce that $q \sqsubseteq_{i+1}^{saf} p$. □

The relation $\sqsubseteq^{saf}$ may be characterized as a weak simulation:

Proposition 2.11 *Let $S = (Q, A_\tau, T, q_0)$ be a labeled transition system and let p_1, p_2 be two states of Q. Then*

$$p_1 \sqsubseteq^{saf} p_2 \text{ if and only if } \forall a \in A_\tau \,.\, \forall r_1 \,.\, (p_1 \xrightarrow{\tau^* a}_T r_1 \Rightarrow \exists r_2 \,.\, (p_2 \xrightarrow{\tau^* a}_T r_2 \wedge r_1 \sqsubseteq^{saf} r_2)).$$

Proof We denote by $\preceq$ the following relation :

$p_1 \preceq p_2$ *if and only if* $\forall a \in A_\tau$.

$\forall r_1 \,.\, (p_1 \xrightarrow{\tau^* a}_T r_1 \Rightarrow \exists r_2 \,.\, (p_2 \xrightarrow{\tau^* a}_T r_2 \wedge r_1 \preceq r_2))$.
Obviously, we have $\sqsubseteq^{saf} \subseteq \preceq$. Conversely, by induction on $i \in N$, we prove that

$p_1 \preceq p_2 \;\Rightarrow\; p_1 \sqsubseteq_i^{saf} p_2$ This property if true for $i = 0$. If $p_1 \preceq p_2$ and $p_1 \xrightarrow{\tau^*}_T r_1$, we have to prove that $r_1 \sqsubseteq_i^{saf} p_2$. We have $r_1 \sqsubseteq_i^{saf} p_1$ by the above proposition and $p_1 \sqsubseteq_i^{saf} p_2$ by induction hypothesis. Thus, by transitivity of $\sqsubseteq_i^{saf}$, we have $r_1 \sqsubseteq_i^{saf} p_2$. $\Box$

Definition 2.14 *Let $S = (Q, A_\tau, T, q_0)$ be a labeled transition system and $p, q \in Q$. We define safety equivalence:*

$$p \approx^{saf} q \text{ if and only if } p \sqsubseteq^{saf} q \;\wedge\; q \sqsubseteq^{saf} p.$$

We introduce a pre-normal form for safety equivalence:

Definition 2.15 *Let $S = (Q, A_\tau, T, q_0)$ be a labeled transition system. A pre-normal form of S, $PNF_{\approx^{saf}}(S)$, with respect to $\approx^{saf}$ may be obtained in the following manner:*
$PNF_{\approx^{saf}}(S) = (Q', A_\tau, T', q')$ where

- $q' = c(q_0, T)$
- $[p_1 \xrightarrow{\tau^* a}_T p_2 \;\wedge\; \forall p_3 \,.\, (p_1 \xrightarrow{\tau^* a}_T p_3 \;\Rightarrow\; p_2 \not\sqsubseteq^{saf} p_3)]$ *if and only if* $c(p_1, T) \xrightarrow{a}_{T'} c(p_2, T)$.

Proposition 2.12 *Let $S = (Q, A_\tau, T, q_0)$ be a labeled transition system and $PNF_{\approx^{saf}}(S) = (Q', A_\tau, T', q')$ its pre-normal form. Then*

$$c(p, T) \in Q' \;\Rightarrow\; p \approx^{saf} c(p, T).$$

Proof Suppose that $c(p, T) \in Q'$.
- It is easy to see that $c(p, T) \sqsubseteq^{saf} p$.
- Conversely, we prove that $p \sqsubseteq_i^{saf} c(p, T)$ by induction on $i \in N$.
if $p \xrightarrow{\tau^* a}_T p_2 \;\wedge\; c(p, T) \xrightarrow{a}_{T'} c(p_2, T)$ then $p_2 \sqsubseteq_i^{saf} c(p_2, T)$ since $c(p_2, T) \in Q'$. If $p \xrightarrow{\tau^* a}_T p_2 \;\wedge\; \neg(c(p, T) \xrightarrow{a}_{T'} c(p_2, T))$ then $\exists p_3 (p \xrightarrow{\tau^* a}_T p_3 \;\wedge\; p_2 \sqsubseteq_i^{saf} p_3 \;\wedge\; (c(p, T) \xrightarrow{a}_{T'} c(p_3, T)$. Thus, by transitivity and by induction hypothesis, we have $p_2 \sqsubseteq_i^{saf} c(p_3, T)$. $\Box$

From this proposition, we deduce:

Proposition 2.13 *Let $S = (Q, A_\tau, T, q_0)$ be a labeled transition system. Then*

$$S \approx^{saf} PNF_{saf}(S).$$

Proposition 2.14 *Let $S_i = (Q_i, A_\tau, T_i, q_i)$, for $i = 1, 2$ be two labeled transition systems and $PNF_{\approx^{saf}}(S_i) = (Q'_i, A, T'_i, q'_i)$ for $i = 1, 2$.*
$PNF_{\approx^{saf}}(S_1) \;\sim\; PNF_{\approx^{saf}}(S_2)$ if and only if $S_1 \;\approx^{saf}\; S_2$.

Proof $\Rightarrow$: By transitivity of $\approx^{saf}$.
$\Leftarrow$:We prove that $c(p_1,T) \approx^{saf} c(p_2,T)$ if and only if $c(p_1,T) \sim c(p_2,T)$. Suppose that $c(p_1,T) \approx^{saf} c(p_2,T)$ and $p_1 \xrightarrow{a}_{T'_1} r_1$, for some $a \in A$ and some $q \in Q$. Then From $c(p_1,T) \sqsubseteq^{saf} c(p_2,T)$, we deduce $\exists r_2(p_2 \xrightarrow{a}_{T'_2} r_2 \wedge r_1 \sqsubseteq^{saf} r_2))$. From $c(p_2,T) \sqsubseteq^{saf} c(p_1,T)$, we deduce $\exists r_3(p_1 \xrightarrow{a}_{T'_1} r_3 \wedge r_1 \sqsubseteq^{saf} r_3))$. Thus, we have $r_1 \sqsubseteq^{saf} r_2 \sqsubseteq^{saf} r_3$ and by definition of pre-normal form, we have $r1 \approx^{saf} r_3$. We can conclude:
$\forall a \in A$.
$\forall r_1 . (p_1 \xrightarrow{a}_{T'_1} r_1 \Rightarrow \exists r_2 . (p_2 \xrightarrow{a}_{T'_2} r_2 \wedge r_1 \approx^{saf} r_2)) \wedge$
$\forall r_2 . (p_2 \xrightarrow{a}_{T'_2} r_2 \Rightarrow \exists r_1 . (p_1 \xrightarrow{a}_{T'_1} r_1 \wedge r_1 \sqsubseteq^{saf} r_2))$.

Since $\sim$ is the greatest relation satisfying the previous requirement we conclude that $c(p_1,T) \sim c(p_2,T)$. $\Box$

3 Efficient algorithm for strong bisimulation equivalence

Efficient algorithm for deciding strong bisimulation equivalence are based on the Relational Coarsest Partition Problem [11], [14]. Paige & Tarjan [14] proposed an algorithm that computes the Relational Coarsest Partition Problem in $O(m \log n)$ time and $O(m)$ space. We present an adapted version of the Paige & Tarjan algorithm by considering a family of relation instead of one relation.

3.1 Relational Coarsest Partition Problem

In this section, we consider a labeled transition system $S = (Q, A_\tau, T, q_0)$where Q is finite. We represent an equivalence relation ρ on the set Q as a partition $\rho = \{B_1, ..., B_n\}$ where the B_i represent its equivalence classes.

An equivalence relation ρ on Q is compatible with T if and only if
$\forall a \in A_\tau . \forall B, B' \in \rho . (B' \subseteq T_a^{-1}[B] \vee B' \cap T_a^{-1}[B] = \emptyset)$. We consider the relational coarsest partition problem [14]:

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

We adapt the Paige & Tarjan algorithm on the following manner. For a partition ρ and subset $X \subseteq Q$, let *split*(B, ρ) be the refinement of ρ obtained by replacing, for each a in A, each block $X \in \rho$ such that $X \not\subseteq T_a^{-1}[B] \wedge X \cap T_a^{-1}[B] \neq \emptyset\}$ by the blocks $X_1 = X \cap T_a^{-1}[B]$ and $X_2 = X - T_a^{-1}[B]$. Note that if $\mid A \mid = 1$, we obtain the same definition of the Paige & Tarjan function *split*. A set $B \subseteq Q$ is called a *splitter* if and only if $\rho \neq$ *split*(B, ρ). When

$\rho = split(B, \rho)$ ρ is *stable* with respect to B. We state properties of function *split* without proof. For more details, see [6].

(i) *split* is monotone in its second argument; that is, $\rho_1 \subseteq \rho_2 \Rightarrow split(B, \rho_1) \subseteq split(B, \rho_2)$,

(ii) $split(B, \rho)$ is a refinement of ρ

(iii) $split(B_1, split(B_2, \rho)) = split(B_2, split(B_1, \rho))$,

(iv) $\forall a \in A_\tau \;.\; \forall X \in split(B, \rho) \;.\; (X \cap T_a^{-1}[B] = \emptyset \;\vee\; X \subseteq T_a^{-1}[B])$

(v) $split(B_1, split(B_2, split(B_1 \cup B_2, \rho))) = split(B_2, split(B_1, \rho))$,

3.2 Solution

The adapted algorithm has the same complexity that the original one. The major difference between the two algorithms lies on the fact that a refinement step, i.e. the computation of *split*, is made with only one element of A in the original one. We present data structures for our algorithm, as they are implemented in Aldébaran. Their selection is application-dependent: we have designed our algorithm to be the basis of a verification tool for concurrent systems, in which a state has generally a few number of successors. In particular we take this fact into account for the implementation of $info_B$ (see definition latter). Let $S = (Q, A_\tau, T, q_0)$ be a labeled transition system, ρ be a partition of Q, $n = \mid Q \mid$, and $m = \mid T \mid$. We suppose that for all a in A_τ, the image set sizes $\mid T_a[p] \mid$ are uniformly bounded by a constant c.

Let us consider the case in which $\mid A_\tau \mid = 1$. Paige & Tarjan presented an algorithm that compute the coarsest refinement of ρ in $O(m \log n)$ time and in $O(m)$ space [14]. Three basic ideas are behind their approach:

- we can compute $split(B, \rho)$ in $\mid T_a^{-1}[B] \mid$ time,
- if ρ is stable with respect to B , $B_1 \subseteq B$ and $B_1 \in \rho$, then Hopcroft's "process the smaller half" idea may be exploited in order to perform a refinement step with respect to B_1 and $B - B_1$. From property (iv) of operator *split* each set X is either a subset of $T_a^{-1}[B]$ or disjoint from it. The refinement step consists of adding in $split(B, \rho)$, for each $X \in \rho$, the following sets:

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

This decomposition may be obtained by searching through only one of the sets – actually, the smaller –, B_1 say, and using the map $\mathit{info}_B(a,p) = \mid T_a[p] \cap B \mid$, for all $p \in Q$. X_1, X_2, X_3, info_{B_1} and info_{B_2} can be computed in time $\mid T_a^{-1}[B_1] \mid$.

The sets X_1, X_2 and X_3 are computed by applying one of the three following rules:

(i) if $\mathit{info}_{B_1}(a,p) = \mathit{info}_B(a,p)$ then $X_1 := X_1 \cup \{p\}$

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

(iii) if $0 < \mathit{info}_{B_1}(a,p) < \mathit{info}_B(a,p)$ then $X_3 := X_3 \cup \{p\}$

- if ρ is unstable with respect to B then a refinement step with respect to B consists of adding in $\mathit{split}(B,\rho)$, for each $X \in \rho$, the following sets:

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

For the general case, a refinement step consists in repeating the previous one for each $a \in A$.

3.3 Algorithm

Several data structures are required to represent states, classes and splitters. Each state p points to a list of couples $(a, T_a^{-1}[p])$, where $T_a^{-1}[p]$ is a list of its elements. 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*. A splitter is a subset B of Q which is either a class (*simple splitter*) or a union of classes (*compound splitter*) such that ρ is stable with respect to B. The refinement step with respect to B is performed according to (2) in the first case whereas it is performed according to (1) 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 an 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.
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 p into a new class. (The elements of a same class are moved into the same new class.) Make each new class points to its associated old class. During the scan of B', compute the list 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, create a new node X_{12} and make it point to X and X_1. Make X and X_1 point back to X_{12} and make X_{12} the new leaf in place of X. (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 i W.
Case 2: B is a compound splitter $B_1 \cup B_2$ (suppose that $\mid X_1 \leq X_2 \mid$)
A refinement step is performed as follows:
Step 1 Remove B from W.
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 file B'. Make info_{B_2} points to info_B. If B_1 or B_2 are nodes, add them in 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 $X = X_i$ for some $i = 1, 2, 3$, then make X points to X_i else add the non-null classes X_i in ρ. Update W in same manner that in the case simple except that two nodes X_{123} and X_{23} may be created such that X_{123} is the father of X_1 and X_{23} and X_{23} is the father of X_2 and X_3.

4 Composition

Communicating systems are described as a hierarchical set of sequential processes interconnected through communication ports. We consider a labeled transition systems algebra with three operators: *parallel composition*, *restriction* and *abstraction*. Communications are described by structuring A_τ with a synchronization product, parametrizing parallel composition [17]. So, synchrony, asynchrony, communication by "rendez-vous" and broadcasting are expressible with the same operator. To allow partial synchrony, a restriction operator $\backslash a$, [12], is introduced on the labeled transition systems to hide port names labeled by a. To verify external specifications, an abstract mechanism τ_I, [3], is used to rename by τ the arcs labeled by an action $a \in I$.

We consider the problem of minimizing space requirement when translating of a term into a labeled transition system. Let us consider a the abstract tree of a term: each node which is not a leaf represents an operator and each leaf represents a labeled transition system. The result of the translation (at the root) is obtained by elaborating partial result associated with each node. Variation of space requirement depends on the order in which the partial results associated with the nodes are elaborated. We present a method for improving space requirement:

1. compute inherited and synthesized attributes for each node,
2. construct a *graph of communications*. Vertices of the graph are the leaves and there is an edge between two vertices if and only if the two associated labeled transition systems communicate. Each edge is labeled by an action set.
3. reduce the graph of communications by applying iteratively the following procedure until only one vertex remains in the graph:
   (a) select and compose two vertices,
   (b) minimize the result by applying some equivalence transformation,
   (c) replace in the graph the result of this composition.

We can derive several methods from this one by defining criteria to select, in the reduction procedure, the two labeled transition system to compose. In this section, we point out several algebraic properties of operators, used for constructing graph of communications. We define background for graphs of communications and we shortly describe algorithms for constructing a graph of communications from a term. We define a criterion for the select procedure in the graph of communications reduction.

4.1 Composition, restriction, abstraction

We define composition of two labeled transition systems with a synchronization algebra and a product of labeled transition systems. A *synchronization algebra* [17] is given by a binary, commutative and associative operation $\bullet$ on A_τ with extra distinguished elements 0 and χ. The binary operation $\bullet$ describes communication or synchronization between labeled transition systems. 0 and χ have been introduced in order to obtain a complete operation rather than a partial one. $a \bullet b = 0$ when no synchronization is possible between an action a and an action b. χ is introduced to allow definition of synchrony or asynchrony. We extend the set A_τ in $A_\tau^\chi = A_\tau \cup \{0, \chi\}$.

Definition 4.1 *A synchronization algebra $L = (A_\tau^\chi, \bullet)$ satisfies:*

1. *$\bullet$ is associative and commutative,*
2. *$\forall a \in A_\tau \,.\, a \bullet 0 = 0$,*
3. *$\chi \bullet \chi = \chi$ and $a \bullet b = \chi \;\Rightarrow\; a = \chi \,\vee\, b = \chi$, for $a, b \in A_\tau$*
4. *$\forall a \in A_\tau \,.\, a \bullet \tau = 0$,*
5. *$\chi \bullet \tau = \tau$,*

6. $\forall a \in A_\tau \,.\, a \bullet \chi = 0$ *or* $a \bullet \chi = a$

Condition (3) expresses that synchronization between internal and external actions is impossible. Condition (4) expresses that internal actions occur asynchronously. Condition (5) defines a synchronous or asynchronous product. In [17], parallel composition is defined by means of Cartesian product on labeled transition systems and synchronization algebra. We extend transition relation in order to define directly parallel composition: let $S = (Q, A_\tau, T, q_0)$ be a labeled transition system

$$T^\chi = T \cup \{(q, \chi, q) \mid q \in Q\}.$$

Definition 4.2 *(Parallel composition) Let $S_i = (Q_i, A_\tau, T_i, q_i)$ for $i = 1, 2$ be two labeled transition systems, let L be a synchronous algebra and $<,>$: States $\times$ States $\rightarrow$ States a partial bijection. We can define parallel composition of S_1 and S_2, as follows:*
$S_1 \|_L S_2 = (Q, A_\tau, T, q_0)$ where:
$q_0 =< q_1, q_2 >$
Q and T are defined by induction:
(i) $< q_1, q_2 > \in Q$

$$\text{(ii) } < p_1, p_2 > \in Q \wedge p_1 \xrightarrow{a}_{T_1^\chi} p_1' \wedge p_2 \xrightarrow{b}_{T_2^\chi} p_2' \wedge a \bullet b \neq 0 \Rightarrow \begin{cases} < p_1', p_2' > \in Q \\ \textit{and} \\ < p_1, p_2 > \xrightarrow{a \bullet b}_T < p_1', p_2' > \end{cases}$$

Proposition 4.1 *Let L be a synchronous algebra, then $\|_L$ is associative and commutative.*

Proof. By associativity and commutativity of synchronous product. □

Example A synchronization algebra for CCS, [12], without passing value.
For each label a, there is a complementary label $\bar{a}$ such that $a \bullet \bar{a} = \tau$. The following conditions are added to conditions of definition 4.1:
$a \bullet b = 0$ if and only if $b \neq \bar{a}$ (for $a \in \mathcal{A}$, only a and $\bar{a}$ can be synchronized)
$a \bullet \chi = a$ (asynchrony).
We define a *restriction* operator on labeled transition systems $\backslash a$ such that all transitions labeled by a is removed.

Definition 4.3 *Let $S = (Q, A_\tau, T, q_0)$ be a labeled transition system and let $a \in A$. We define $S \backslash a = (Q', A_\tau, T', q_0)$ where:*
$T' = T \setminus \{(p, a, q) \mid (p, a, q) \in T\}$
Q' is the set of states reachable from q_0 via T'.

An *abstraction* operator τ_I is defined in order to rename in τ the arcs labeled by $a \in I$. This definition is similar to the definition of abstraction in ACP.

Definition 4.4 *Let $S = (Q, A_\tau, T, q_0)$ be a labeled transition system and let $I \subseteq A$. We define $\tau_I(S) = (Q, A_\tau, T', q_0)$ where:*
$T' = \{(p, \tau, q) \mid (p, a, q) \in T \wedge a \in I\} \cup \{(p, a, q) \mid (p, a, q) \in T \wedge a \notin I\}$
Q' is the set of states reachable from q_0 via T'.

We exhibit properties of these operators on labeled transition systems in order to construct a *graph of communications.*

Notation 1 *Let $S = (Q, A_\tau, T, q_0)$ be a labeled transition system. Let L be a synchronization algebra, $A, B \subseteq \mathcal{A}$ and a, a_1 and a_2 elements of A.*

1. *The synchronization product is extended to subsets of $\mathcal{A}$:*
   $A \bullet B = \{a \bullet b \mid a \in A \wedge b \in B \wedge a \bullet b \neq \tau \wedge a \bullet b \neq 0\}$

2. *Restriction operator is extended to subsets of $\mathcal{A}$:*
   $(S \backslash a_1) \backslash a_2 = S \backslash \{a_1, a_2\}$

3. $\mathcal{A}(S) = \{a \in \mid \exists p, q \in Q \ \wedge \ p \xrightarrow{a}_T q\}$.

4. $Com(a) = \{b \mid b \in \mathcal{A} \wedge a \bullet b \neq 0\}$

5. $$Com(A) = \bigcup_{a \in A} Com(a)$$

Proposition 4.2 *Let $S_i = (Q_i, A_\tau, T_i, q_i)$ for $i = 1, 2$ be two labeled transition systems, let L be a synchronous algebra and let I_1, I_2 be subsets of A.*

1. $(S_1 \backslash I_1) \backslash I_2 = S_1 \backslash (I_1 \cup I_2)$

2. $(S_1 \backslash I_1) = S_1$ *if* $I_1 \cap \mathcal{A}(S_1) = \emptyset$

3. $(S_1 \|_L S_2) \backslash a = (S_1 \backslash a) \|_L S_2$ *if*
   $(a \notin \mathcal{A}(S_1) \bullet \mathcal{A}(S_2) \wedge a \notin \mathcal{A}(S_2) \wedge (a \in \mathcal{A}(S_1) \Rightarrow Com(a) \cap \mathcal{A}(S_2) = \emptyset)) \vee$
   $(a \in \mathcal{A}(S_1) \bullet \mathcal{A}(S_2) \wedge ((a = b \bullet c) \Rightarrow a = b = c))$

4. $(S_1 \|_L S_2) \backslash a = (S_1 \backslash a) \|_L (S_2 \backslash a)$ *if*
   $$\begin{aligned} (a \notin \mathcal{A}(S_1) \bullet \mathcal{A}(S_2) \quad & \wedge \quad (a \in \mathcal{A}(S_1) \Rightarrow Com(a) \cap \mathcal{A}(S_2) = \emptyset) \\ & \wedge \quad (a \in \mathcal{A}(S_2) \Rightarrow Com(a) \cap \mathcal{A}(S_1) = \emptyset)) \vee \\ (a \in \mathcal{A}(S_1) \bullet \mathcal{A}(S_2) \wedge ((a = b \bullet c) & \Rightarrow a = b = c)). \end{aligned}$$

5. *Properties (1) - (iv) hold when abstraction operator is substituted for restriction operator,*

6. $(\tau_{\{a\}}(S))\backslash\{b\} = \tau_{\{a\}}(S\backslash\{b\})$,

7. $(\tau_{\{a\}}(S))\backslash\{a\} = (\tau_{\{a\}}(S))$

8. $\tau_{\{a\}}(S\backslash\{a\}) = S\backslash\{a\}$.

Proof. By set properties. □

4.2 From a term to labeled transition system

We consider in this section a congruence relation $\sim_R$ and an abstract syntax for terms:

$$t ::= S \mid t\|_L t \mid t\backslash I \mid \tau_I$$

where S, t, I ranges over respectively labeled transition systems, terms, subsets of A.

4.2.1 Computing attributes

Each node is decorated with two inherited attributes *hid* ad *rel*, denoting respectively the set of hidden labels and the set of labels renamed in τ, and by two synthesized attributes *ste*, *vis*, denoting respectively the set of labeled transition systems on the leaves and the set of visible labels. The following table summarizes the semantic rules computing each attribute:

production rules	*semantic rules*
$t \rightarrow S = (A, A_\tau, T, q_0)$	$t.vis := calA$ $t.ste := \{S\}$
$t \rightarrow \tau_I(t_1)$	$t.vis := (t_1.vis) - I$ $t.ste := t_1.ste$ $t.hid := (t.hid) - I$ $t_1.hid := t.hid$ $t.rel := (t.rel) \cup I$ $t_1.rel := t.rel$
$t \rightarrow t_1 \backslash I$	$t.vis := (t_1.vis) - I$ $t.ste := t_1.ste$ $t.hid := (t.hid) \cup I$ $t_1.hid := t.hid$ $t.rel := (t.rel) - I$ $t_1.rel := t.rel$
$t \rightarrow t_1 \|_L t_2$	$t.vis := (t_1.vis) \cup (t_2.vis) \cup (t_1.vis \bullet t_2.vis)$ $t.ste := (t_1.ste) \cup (t_2.ste)$ $t_1.hid := t.hid$ $t_2.hid := t.hid$ $t_1.rel := t.rel$ $t_2.rel := t.rel$

4.2.2 Transforming the tree

Rules of proposition 4.1 are used in order to transform the abstract tree. Rules 1 and 2 are oriented left to right whereas rules 3 and 4 are oriented right to left. When one of the rules 3 or 4 fails, a graph of communications is associated with the nodes restriction or abstraction.

4.2.3 Generating the graph of communications

A *Graph of communications* is an undirected graph $(E, V, \rightarrow)$ associated with each node $\backslash$, τ_I and with the root. $E \subseteq A$ is the set of hidden actions associated with the node. v is the set of vertices. A vertex is a leaf of abstract tree and there is an edge between two vertices v_i and v_j if and only if:

$$\exists a \in v_i.vis \ \wedge \ \exists b \in v_j.vis \ \wedge \ a \bullet b \neq 0.$$

This edge is labeled by the set v_{ij} (i.e., visible actions after composition of $v = v_i$ and v_j), subset of $v_i \cup v_j \cup (v_i \bullet v_j)$. v_{ij} is obtained by removing from v the elements a of E occurring only in the communication of v_i and v_j.

4.2.4 Reducing the graph of communications

This reduction is made by removing two vertices v_i and v_j and inserting the result of the composition of v_i and v_j, minimized with respect to $\sim_R$. Several criteria may be implemented in order to choose this vertex. The criterion adopted here takes into account the three following idea:

- first, we choose labeled transition systems in the same connected component.
- Second, we determine, in the same component, the set of vertices with smallest degree (i.e. the number of vertices adjacent to a vertex).
- Third, among this set we select two vertices such that v_{ij} is minimal according to the number of elements.

5 Aldébaran

The tool has been implemented in order to verify labeled transition systems. The main characteristics of Aldébaran is *efficiency*, *extensibility*, *portability* and *reusability*.

efficiency Algorithm which computes the largest bisimulation is an adaptation of the Paige & Tarjan algorithm that solves the coarsest partition problem. This algorithm is required in the minimization procedures and the decision procedures with respect to an equivalence relation.

extensibility The algorithms to transform the labeled transition systems to allow the computation of normal form with respect to an equivalence relation are based on algorithms on the graph: *depth first search* (for example transitive closure for the τ relation) and *breath first search* (for example *cross* operation in acceptance model equivalence). Thus reduction algorithms for new equivalence relations may be easyly implemented.

portability Aldébaran is written in C and runs on Unix system.

reusability Aldébaran may be interfaced with other systems which manipulate labeled transition systems. For instance, Aldébaran is interfaced with a LOTOS compiler [8] and Xesar system. Is is a part of Ocmin, a minimizer of a common object code produced by LUSTRE and ESTEREL compilers [2].

5.1 Objects

Actions They consist of strings. The string i denotes the invisible action τ.

Labeled transition system Non-deterministic sequential processes are described as labeled transition systems. A labeled transition system consist of a descriptor and a list of edges. The descriptor contains the initial state, the number of transitions, the number of states. A state is a natural number ranging into `0 .. number_of_states -1`. Parallel composition, restriction and abstraction are used for building networks (i.e., terms) and scope of action visibility. A network must be bound to a name using the **comp** command. We give hereafter the concrete syntax of terms in *Yacc* input-like.

```
net     : net '&' sub_net              /* Parallel composition */
        | sub_net
        ;
subnet  : subnet '#' '['Actionslist']'  /* Restriction */
        | subnet '$' '['Actionslist']'  /* Abstraction */
        | lts-identifier
        | '(' net ')'
        ;
```

Initial partition All the reductions (i.e., computation of a normal form) with respect to an equivalence relation are carried out according to an initial partition. It may be the universal partition or a partition obtained from basic predicates on the analyzed system. In this case, this allows reduction according to basic properties on the systems.

Synchronization algebra Parallel composition may be defined in different ways. Synchronization algebra is a very nice construct for defining various kinds of communications. A synchronization algebra must be defined, using the **def_com** command, before parsing a term.

5.2 Input/output commands

There are two kinds of input files: those which contain labeled transition system or initial partition and whose contain a network description.

Synopsis of the Aldébaran command:
aldebaran [options] name$_1$ ... name$_i$.
Aldébaran accepts two input file format : ASCII file and binary file (option **-bin**). Files whose names end with ".aut" (resp. ".gra"), are taken to be ASCII files (resp. binary files) containing a labeled transition system whereas files whose names end with ".cls" are taken to be files containing an initial partition. In this case, the file contains a list of integers. The ith element denotes the class containing the state i. There are two representations for the edges: direct or inverse (option **-inv**). The following twelves options inhibit the interactive use of Aldébaran.

bmin, omin, amin, smin The labeled transition system contained in the file name$_i$ is minimized with respect to strong bisimulation, observational equivalence, acceptance model equivalence or safety equivalence according to an initial partition if there is a file name$i$.cls.

bequ, oequ, aequ, sequ The equality of the two labeled transition system contained respectively in name$_1$ and name$_2$ is tested with respect tostrong bisimulation, observational equivalence, acceptance model equivalence or safety equivalence. The result TRUE or FALSE is displayed.

bcla, ocla, acla, scla For each labeled transition system contained in name$_i$, equivalence classes with respect to strong bisimulation, observational equivalence, acceptance model equivalence or safety equivalence are displayed.

5.3 System functionalities

The major interest of using interactively the systems lies on the experimentation of network description. However, reduction and comparison functions are available interactively.

def_com int int [Tripleslist] Defines the synchronization algebra. The first integer must be 0, for synchronous product, or 1, for asynchronous product. The second integer must takes it values in $\{0, 1, 2, 3\}$.

0 (synchronization product of CCS):
$a \bullet b = \tau$ for $b = naora = nb$
$a \bullet b = 0$ otherwise.

1 $a \bullet a = a$
$a \bullet b = 0$ for $b \neq a$.

2 $a \bullet a = \tau$
$a \bullet b = 0$ for $b \neq a$.

3 The third parameter is supplied from the user. The synchronization product is defined extensively. This parameter obeys the syntax:

```
Tripleslist     : '[' triple_list ']'
                ;

triple_list     : triple
                | triple_list ';' triple
                ;
triple          : aj ',' bj, ',' cj
```

and defines the following synchronization product: i$ai \bullet bj = cj$.

pr_com Displays the synchronization relation.

comp int int idf term Translates the term of the labeled transition systems algebra into a labeled transition system. The first integer 0, 1, 2 or 3 denotes respectively the following equivalence relation none. strong bisimulation, observational equivalence. The second integer denotes the strategy for translating a term:

0 none

1 The term is evaluated from left to right. After each sub-term evaluation, the result is minimized according to the equivalence relation.

2 A graph of communications is associated with each node restriction, abstraction and with the root after the transformation of abstract tree. Then each graph of communications is reduced and he result in substituted in the tree for the corresponding node.

help Displays the list of available commands.

ls_ste Displays the list of labeled transition systems.

pr_ste name Displays the labeled transition system bounded to name.

open_in_stream name, open_out_stream name The file name is respectively used as standard input or standard output.

close_all The input stream (resp. output stream) is `stdin` (resp. `stdout`).

def_tau string string are used as τ.

stat Displays some statistics for each command.

unstat Cancels alls statistics.

time

notime These are analogous to statistics on calls.

5.4 Example

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 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. We give two specifications in Lotos [8]. 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 ; in the second one only the *ai* are visibles. (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 to find the coarsest partition compatible with the transition relation and the universal partition.

5.5 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

5.6 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 presented an overview of Aldébaran. We have shown that it is possible to implement efficient decision procedures for some equivalence relations. We have design a verification tool which may be easily interfaced with others systems. An extension has been proposed in [13] in order to explain why two labeled transition systems are not similars according to an equivalence relation.

There are many directions for future work. One involves the definition of equivalence relations compatible with a set of properties (i.e., temporal logic formulas). Another involves strategies to control state explosion occurring in the parallel composition. An attempt in this way has been done with the graph of communications notions.

References

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

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

[3] J. A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *TCS*, 37 (1), 1985.

[4] 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.

[5] S. D. Brookes, C.A.R Hoare, and A.W. Roscoe. Theory of communicating sequential processes. *JACM*, 31(3), 1984.

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

[7] J. C. Fernandez. *Aldébaran: User's Manual.* Technical Report, LGI-IMAG Grenoble, 1988.

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

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

[10] S. Graf and J. Sifakis. *Readiness Semantics for Regular Processes with Silent Action.* Technical Report Projet Cesar RT-3, LGI-IMAG Grenoble, 1986.

[11] 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.

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

[13] L. Mounier. Equivalence des systèmes de transitions étiquetées : réduction et diagnostic. June 1989. Rapport de DEA, Grenoble.

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

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

[16] C. Rodriguez. *Spécification et validation de systèmes en XESAR.* PhD thesis, Institut National Polytechnique de Grenoble, 1988.

[17] G. Winskel. Synchronization tree. In J. Diaz, editor, *10th ICALP, LNCS 154*, 1983.