Appendix Deliverable SYRF
S. Pinchinat[*], O. Kushnarenko, H. Marchand

Full report from Rennes

1 Introduction : the context and the previous results

Last year, we have presented *Implicit Labeled Transition Systems* as intermediate models for discrete event systems. We have studied operations of parallel composition and event hiding, as well as an equivalence criterion based on strong bisimulation semantics, namely the equivalence called *symbolic bisimulation*.

The aim of the work is to rely on intensional descriptions of the systems for symbolic abstractions purposes, based on behavioral equivalences. The intensional approach we propose has the main advantage to remain at an interesting level of abstraction to handle sets (of states, of events) in which algorithms and properties can entirely be expressed, thus providing us with a high level of expressiveness. This abstract level avoids us to bother with a particular choice of sets implementation, such as BDDs, even if all operations are actually based on this representation for sets. Moreover, the intensional formalism is completely compatible with the symbolic techniques, since intensionally described sets can afterwards be implemented by any kind of methods, such as decision diagrams. However, other tools with different implementations might be considered using the same high level framework (e.g. Mathematica, Maple) [Dut92, GJ94].

Intensional models have already been the subject of previous work [LBMR96], under the name of *polynomial dynamical systems*. They were the base of the temporal logics verification tool SIGALI for the SIGNAL language we develop in our group in Rennes. The task 3.1 of the SYRF project has led us to enrich the scope of this verification tool by implementing equivalence checking, such as strong bisimulation on the basis of the intensional philosophy. This new functionality of SIGALI is of high interest since the SIGNAL language is used in a lot of areas (controller synthesis [DLB94], [ML98, MBLL98] (see also Part II of the report), robotics [MRMC98],...) where models equivalence checking is crucial. Indeed, using reduction techniques, we hope for dealing with smaller models, then improving the efficiency of the tool SIGALI.

[*]IRISA Campus de beaulieu, 35042 Rennes Cedex, FRANCE. E-mail:Sophie.Pinchinat@irisa.fr

This year, we aimed to complete the equivalence checking methodology by integrating in SIGALI the construction of a reduced model according to a given equivalence. This work has the advantage to design the reduced model still in an intensional manner, whereas in classic approaches symbolic models might become explicit when reduced [Fer84]. However, our method still has some explicit aspects features (we cannot yet avoid the equivalence class enumeration). Note that the reduced model is built "on the fly" during this unique enumeration phase.

Regarding the underlying theory we have presented last year, we would like to emphasis, this year, the new presentation of the semantic models, so-called *Intensional Labeled Transition Systems*. In this report, the models are still based on the specific domain of SIGNAL programs boolean abstraction, that is $\mathbf{Z}_3$, but the theory could be generalized to other domains (e.g. the family of finite domains $\mathbf{Z}_p$, with p prime), provided we clearly identify the hypothesis for algorithms termination (for example, the finitely branching property of the underlying explicit models) and the decidability properties that are needed to apply the techniques. But this is out of the scope of the project.

The report is organized as follows: Section 2 introduces the theoretical framework, the so-called *intensionally Labeled Transition Systems (iLTS)*, as the labels of the transitions are sets defined "intensionally". Also parallel composition and event hiding are considered as basic combinators for the modular design of systems. Next, the equivalence of *symbolic bisimulation* is defined with an associated study of some properties, s.a. a congruence theorem and other characterizations.

Then, in Section 3 the models become "fully" intensional, called *Intensional Labeled Transition Systems (ILTS)*: the set of all possible transitions of the system is itself described intensionally, just like symbolic methods proceed classically. On the basis of those models, we propose algorithms to compute the symbolic bisimulation between two ILTS (Section 3.1, but mostly a method to build the reduced model according to this equivalence (Sections 3.2.1 and 3.3).

The integration environment in which our methods are to be applied is presented Section 4: Section 4.1 presents the language SIGNAL and Section 4.2 explains how the Intensional Labeled Transition System is (automatically) derived from a SIGNAL program. The verification tool SIGALI is briefly commented in Section 4.3.

2 The basic models

In the following, we write $\mathbf{Z}_3$ for the finite field $\{-1, 0, 1\}$ in which $x^3 = x$ and $3x = 0$ for all $x \in \mathbf{Z}_3$. Let Y be a finite set of m distinct variables $Y_1, ..., Y_m$. We denote by $\mathbf{Z}_3[Y]$ (or $\mathbf{Z}_3[Y_1, ..., Y_m]$) the set of polynomials over variables $Y_1, ..., Y_m$ which coefficients range over $\mathbf{Z}_3$ with typical elements $P(Y)$ (or P for short), $P_1(Y), \ldots$. We recall that $(\mathbf{Z}_3[Y], +, *)$ is a ring.

2.1 Intensionally labeled transition systems

Definition 2.1 (Intensionally Labeled Transition Systems (iLTS)) *An* m-dimensional intensionally Labeled Transition System (or m-iLTS) *is a structure*
$T = (Q, Y, \to)$*, where*
- Q *is set of states,*

- Y *is a set of* m *variables* $Y_1, ..., Y_m$*, and*
- $\rightarrow \subseteq Q \times \mathbf{Z}_3[Y] \times Q$*. Each transition is labeled by a polynomial over the set* Y*.*

We write $q \overset{P(Y)}{\rightarrow} q'$ (or simply $q \overset{P}{\rightarrow} q'$ when it is clear from the context), instead of $(q, P(Y), q') \in \rightarrow$.
Given a polynomial $P(Y) \in \mathbf{Z}_3[Y]$, we associate its set of solutions $Sol(P) \subseteq \mathbf{Z}_3^m$, defined by

$$Sol(P) \overset{\text{def}}{\equiv} \{(y_1, ..., y_m) \in \mathbf{Z}_3^m | P(y_1, ..., y_m) = 0\}$$

Then, iLTS can be understood as an "intensional" representation of classical labeled transition systems, where the labels are tuples in $\mathbf{Z}_3^m$: each arrow of the iLTS labeled by $P(Y)$ intensionally represents as many arrows labeled by some y where $y \in Sol(P(Y))$. We call $Ext(T)$ the corresponding "extensional" labeled transition system.

Now, it is worthwhile noting that in $\mathbf{Z}_3[Y]$, polynomials $Y_1^3 - Y_1, ..., Y_m^3 - Y_m$ evaluate to zero (this is due to Fermat's Theorem). Then for any $P(Y) \in \mathbf{Z}_3[Y]$, one for instance has $Sol(P) = Sol(P + (Y_1^3 - Y_1))$, but also, $Sol(P) = Sol(-P) = Sol(P^2)$, etc... A very natural abstraction would be to consider iLTS modulo isomorphism, of course, but also modulo $\equiv$-equivalence over labels, where $P_1 \equiv P_2$ whenever $Sol(P_1) = Sol(P_2)$. To this end, we introduce the quotient ring of polynomial functions $A[Y] = \mathbf{Z}_3[Y]/_{<Y^3-Y>}$, where all polynomials $Y_i^3 - Y_i$ are identified to zero, written for short $Y^3 - Y = 0$. $A[Y]$ can be regarded as the set of polynomial functions with coefficients in $\mathbf{Z}_3$ for which the degree in each variable is lower than 2. This will be very useful for algorithmic purposes. [Dut92] showed how to define a unique representative of $Sol(P)$ (for $P \in \mathbf{Z}_3[Y]$), i.e. of each $\equiv$-equivalence class, called the *canonical generator*. This polynomial is the characteristic function of $Sol(P)$ and has at most degree 2 in each variable (see Section 3.3.1).

Lemma 1 [Dut92]
Given a polynomial $P \in \mathbf{Z}_3[Y]$*, we can compute a canonical generator* $[P]_\equiv$.

At this point, we would like to mention some useful properties for polynomials handling, namely :

Property 1 *For all* $P_1, P_2, P \in \mathbf{Z}_3[Y]$

$$Sol(P1) \cap Sol(P_2) = Sol(P_1 \oplus P_2), \textit{ where } P_1 \oplus P_2 \overset{def}{=} P_1^2 + P_2^2 \tag{1}$$

$$Sol(P_1) \cup Sol(P_2) = Sol(P_1 * P_2) \tag{2}$$

$$\mathbf{Z}_3^m \setminus Sol(P) = Sol(1 - P^2) \tag{3}$$

2.2 Operations over iLTS

The class of iTLS can be provided with the usual operations over (extensional) transition systems. Among them, the *parallel composition* and the *events hiding* play an important role in the complex systems design.

Parallel composition over iLTS imposes the compatibility of values between common events of the composed systems. From the extensional point of view, Definition 2.2 is the classical *synchronous parallel composition* as defined in ESTEREL, SIGNAL or LUSTRE languages, but the intensional approach avoids a part of the potential combinatorial explosion to compute the synchronized transitions.

Definition 2.2 (Parallel composition of iLTS) *Let $T_1 = (Q_1, Y, \rightarrow_1)$ be an m_1-iLTS and $T_2 = (Q_2, U, \rightarrow_2)$ be an m_2-iLTS with possible common variables between Y and U. The* parallel composition *of T_1 and T_2, written $T_1 \mid T_2$, is $(Q_1 \times Q_2, Y \cup U, \rightarrow)$ with*

$$(q_1, q_2) \overset{P_1(Y) \oplus P_2(U)}{\rightarrow} (q_1', q_2') \text{ whenever } \begin{cases} q_1 \overset{P_1(Y)}{\rightarrow}_1 q_1' \text{ in } T_1 \\ q_2 \overset{P_2(U)}{\rightarrow}_2 q_2' \text{ in } T_2 \end{cases} \tag{4}$$

Remark 1 *Because in $\mathbf{Z}_3$, $P_1 \oplus P_2 = 0$ iff $(P_1 = 0 \wedge P_2 = 0)$, we have $P_1 \oplus P_2 = P_2 \oplus P_1$ and $(P_1 \oplus P_2) \oplus P_3 \equiv P_1 \oplus (P_2 \oplus P_3)$. Therefore, parallel composition over iLTS is clearly commutative and associative.*

Hiding events consists in abstracting from components of the label. It helps in internalizing some communications between the composed systems that are not relevant to observe in the behavior.
Let $P \in \mathbf{Z}_3[Y]$, we shall write $\exists Y_i P$ for the polynomial $P|_{Y_i=-1} * P|_{Y_i=0} * P|_{Y_i=1}$, where $P|_{Y_i=v}$ is P obtained by instantiating any occurrence of variable Y_i by value v. The reader can check that $Sol(\exists Y_i P)$ is obtained from $Sol(P)$ by deleting the i-th component of its elements (it is a projection). Also when $\tilde{Y} \subset Y$ is some $\{Y_{i_1}, ..., Y_{i_r}\}$ we simply write $\exists \tilde{Y} P$ for $\exists Y_{i_1} ... \exists Y_{i_r} P$.

Also it is possible to define a dual variable abstraction over polynomials, based on universal quantificator: $\forall Y_i P$ is computed as $P|_{Y_i=-1} \oplus P|_{Y_i=0} \oplus P|_{Y_i=1}$ which solutions are elements of the form $(y_1, ..., y_{i-1}, y_{i+1}, ..., y_m)$ s.t. $\forall y_i, (y_1, ..., y_{i-1}, y_i, y_{i+1}, ..., y_m) \in Sol(P)$. This abstraction will be considered in other sections of this report.

Definition 2.3 (Event hiding) *Let $T = (Q, Y, \rightarrow)$ be an m-iLTS, and $Y_i \in Y$. We define the $(m-1)$-iLTS $(T \setminus \{Y_i\})$ by $(Q, Y \setminus \{Y_i\}, \rightarrow_{\setminus \{Y_i\}})$ where $q_1 \overset{\exists Y_i P}{\rightarrow} q_2$ in $T \setminus \{Y_i\}$ iff $q_1 \overset{P}{\rightarrow} q_2$ in T.*
This definition can be generalized to sets of variables $\tilde{Y} \subseteq Y$.

2.3 Symbolic bisimulation

As we aim to manipulate transition systems in an intensional way, we explain here how the classical strong bisimulation can be handle in this setting (see Theorem 3). The definition is strongly inspired from De Simone's symbolic bisimulation over reactive automata [DR94].

In order to be compared, events of two iLTS have to belong to the same space $\mathbf{Z}_3^m$. This way, we suppose without of lost of the generality, two iLTS have the same events variables.

Definition 2.4 (Symbolic Bisimulation) *Let $T_1 = (Q_1, Y, \rightarrow_1)$ and $T_2 = (Q_2, Y, \rightarrow_2)$ be two iLTS. A symbolic bisimulation between T_1 and T_2 is a binary relation $\mathcal{R} \subseteq Q_1 \times Q_2$ s.t. $q_1 \mathcal{R} q_2$ whenever*
(1) for all $q_1 \xrightarrow{P}_1 q_1'$ there exists a finite set of transitions $(q_2 \xrightarrow{P_i}_2 q_2^i)_{i \in I}$ with
(a) $(1 - P^2) * \Pi_i P_i = 0$, and
(b) $q_1' \mathcal{R} q_2^i$, for all $i \in I$, and
(2) vice versa.

Definition 2.4 can be intuitively explained as follows: this equivalence aims to capture exactly classic strong bisimulation over the underlying explicit models (as stated by Theorem 3). However, it takes advantage of the intensional feature by imposing that, for the first system, each outgoing transition of a given set (represented intensionally by P) can be mimicked in the second system, i.e. belongs to at least one outgoing "set" of transitions (one of the $Sol(P_i)$'s). Point (a) of Definition 2.4 can be rephrased as $Sol(P) \subseteq \bigcup_i Sol(P_i)$, then retrieveing the original definition of De Simone [DR94].

Theorem 2 (Compositionality) *Symbolic bisimulation is a congruence w.r.t. parallel composition and events hiding.*

Proof Symbolic bisimulation is a congruence w.r.t. parallel composition $|$ and events hiding $\backslash Y$.
We only prove the result for $|$, the remaining case is easier since symbolic bisimilar states remain bisimilar when hiding is performed.

Let $T_1 = (Q_1, Y, \rightarrow_1)$ and $T_2 = (Q_2, Y, \rightarrow_2)$ be two m-ILTS of dimension m, and let $T_3 = (Q_3, U, \rightarrow_3)$ be a k-ITLS. Let assume that there exists some symbolic bisimulation $\mathcal{R}_0$ between T_1 and T_2. We show that the relation $\mathcal{R}$ given by

$$\mathcal{R} \stackrel{\text{def}}{=} \{((p_1, p_3), (p_2, p_3)) \in (Q_1 \times Q_3) \times (Q_2 \times Q_3) \mid (p_1 \mathcal{R}_0 p_2)\}$$

is a symbolic bisimulation between $T_1 \mid T_3$ and $T_2 \mid T_3$.

Consider a pair $((q_1, q_3), (q_2, q_3)) \in \mathcal{R}$. It is folklore that we have to verify the transfer property of the symbolic bisimulation.

Suppose $(q_1, q_3) \xrightarrow{P} (q_1', q_3')$, by Definition 2.2 it means that P is some $P_1(Y) \oplus P_3(U)$ where $q_1 \xrightarrow{P_1(Y)}_1 q_1'$ in T_1 and $q_3 \xrightarrow{P_3(U)}_3 q_3'$ in T_3. By definition of $\mathcal{R}$, $q_1 \mathcal{R}_0 q_3$. Then there exists a finite set of transitions $(q_2 \xrightarrow{P_{2,i}(Y)} q_2^i)_{i \in I}$ with $q_1' \mathcal{R}_0 q_2^i$ and

$$[(1 - P_1^2(Y)) * \Pi_i P_{2,i}(Y)] = 0. \tag{5}$$

Now, proving that $(1 - (P_1(Y) \oplus P_3(U)) * \Pi_i (P_{2,i}(Y) \oplus P_3(U)) = 0$
is sufficient. Let $(y, u) \in \mathbf{Z}_3^m \times \mathbf{Z}_3^k$. If $1 - (P_1(y) \oplus P_3(u) = 0$ then the equation is obtained; otherwise, $P_1(y) \oplus P_3(u) = 0$, which entails $P_1(y) = 0$ and $P_3(u) = 0$. By

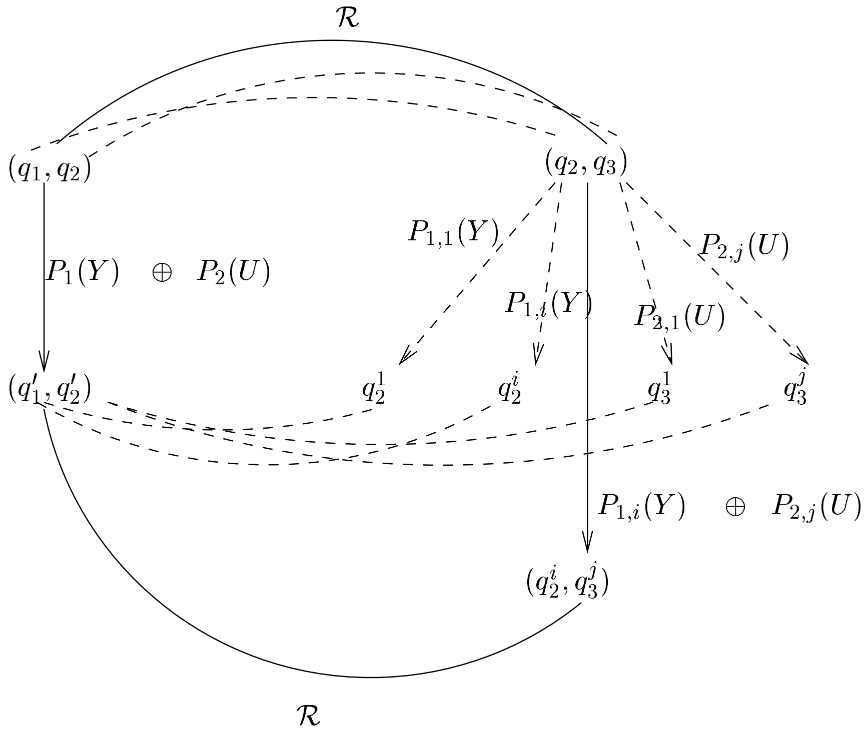

Figure 1: Congruence property for $|$

Equation (5), $\Pi_i(P_{2,i}(y) = 0$ which implies that $P_{2,i_0}(y) = 0$ for some i_0. Therefore, $\Pi_i(P_{2,i}(y) \oplus P_3(u)) = 0$ and we are done. The couples (q_2^i, q_3) are then the good candidates to conclude the proof. See Figure 1.

Transitions starting from (q_2, q_3) are dealt similarly. $\square$

We show here that symbolic bisimulation is an alternative view of strong bisimulation when intensional models are considered. At this stage, we assume the reader is familiar with the class of (extensional) labeled transition systems, as well as with the equivalence of *strong bisimulation*. However, we recall:

Definition 2.5 [Par81, Mil89b] (Strong bisimulation) *Given two transition systems (labeled over some set A) $t_1 = (Q_1, A, \rightarrow_1)$ and $t_2 = (Q_2, A, \rightarrow_2)$, a bisimulation between t_1 and t_2 is a binary relation $\rho \subseteq Q_1 \times Q_2$ s.t. $(q_1, q_2) \in \rho$ whenever*

(1) *for all $a \in A$, for all transition $q_1 \xrightarrow{a}_1 q'_1$ there exists a state q'_2 s.t. $q_2 \xrightarrow{a}_2 q'_2$ and $(q'_1, q'_2) \in \rho$, and*

(2) *vice versa.*

Symbolic bisimulation between iLTS corresponds to strong bisimulation between the extensional labeled transition systems:

Theorem 3 *Let T_1 and T_2 be two iLTS. Then there exists a symbolic bisimulation between T_1 and T_2 iff there exists a strong bisimulation between $Ext(T_1)$ and $Ext(T_2)$ (actually, by keeping the same relation over states).*

Proof Let T_1 and T_2 be two ILTS. Then there exists a symbolic bisimulation between T_1 and T_2 iff there exists a strong bisimulation between $Ext(T_1)$ and $Ext(T_2)$.

$\Rightarrow$) Let $\mathcal{R}$ be a symbolic bisimulation between T_1 and T_2. We show that $\mathcal{R}$ is a strong bisimulation between $Ext(T_1)$ and $Ext(T_2)$. Let $q_1 \mathcal{R} q_2$ and let $q_1 \xrightarrow{y}_1 q_1'$ in $Ext(T_1)$. Then there exists $q_1 \xrightarrow{P}_1 q_1'$ with $P(y) = 0$. By definition of $\mathcal{R}$, there exists some indexes i such that $q_2 \xrightarrow{P_i} q_2^{(i)}$, $(1-P^2) * \Pi_i P_i = 0$ and $q_1' \mathcal{R} q_2^{(i)}$. Because $P(y) = 0$, then $(1-P^2) * \Pi_i P_i = 0$ applied to y entails $P_i(y) = 0$ for some i which proves then $q_2 \xrightarrow{y}_2 q_2^{(i)}$, and we are done. Transition $q_2 \xrightarrow{y}_2 q_2'$ is dealt similarly.

$\Leftarrow$) Let us show that a strong bisimulation ρ is a symbolic bisimulation. Let $q_1 \rho q_2$ and let $q_1 \xrightarrow{P}_1 q_1'$ in T_1. Because $q_1 \rho q_2$, for each $y_0 \in Sol(P)$ there exists $q_2 \xrightarrow{y_0}_2 q_2^{y_0}$ in $Ext(T_2)$ with $q_1' \rho q_2^{y_0}$. In T_2 there exists some P^{y_0} s.t. $q_2 \xrightarrow{P^{y_0}}_2 q_2^{y_0}$ and $P^{y_0}(y_0) = 0$.
Consider now the polynomial $\Pi_{y \in Sol(P)} P^y(Y)$[1]. Clearly, $\forall y_0,\ P(y_0) = 0 \Rightarrow \Pi_{y \in Sol(P)} P^y(y_0) = 0$, then $(1 - P^2) * \Pi_{y \in Sol(P)} P^y(Y)$ evaluates to zero. Then the q_2^y are the good candidates. $\square$

3 Intensional Labeled Transition Systems

Intensional approach for labels offers a "compact" way to talk about sets of transitions in the system. However, we would like to reinforce this method in such a way that the whole system, and not only its sets of labels, can be itself described intensionally. For this purpose, a structure over the states is unavoidable. We propose the fairly standard structure of tuples for states where values ranges over booleans (in our setting it means values 1 and -1). This is classically used in symbolic verification methods [Cou91].

Intuitively, the set of transitions will be given by a polynomial, which generalizes the iLTS approach. Section 4 explains how this formalism can be obtained for "free" (provided we have specified the system in a the high level language, in our case the SIGNAL language) from the real systems to be compared and reduced modulo symbolic (or equivalently, strong) bisimulation.

Definition 3.1 *An* (n, m)-dimensional Intensional Labeled Transition System (or ILTS for short) *is a structure $S = (X, X', Y, \mathcal{T})$ where $X = \{X_1, ..., X_n\}$ and $X' = \{X_1', ..., X_n'\}$ are two sets of* (source and target) states variables, *$Y = \{Y_1, ..., Y_m\}$ is a set of labels variables and $\mathcal{T}(X, Y, X')$ is a polynomial in $\mathbf{Z}_3[X, Y, X']$ describing the legal transitions.*

Given some source state $x = (x_1, ..., x_n) \in \mathbf{Z}_3^n$ and some target state $x' = (x_1', ..., x_n') \in \mathbf{Z}_3^n$, the set $Sol(\mathcal{T}(x, Y, x'))$ denotes all the possible labels of transitions from state x to state

[1] note that $Sol(P)$ is finite.

x'. When states are viewed extensionally, we retrieve the iLTS of in Section 2.1, which in turn can be interpreted as a classical labeled transition system.

Actually, ILTS are also known under the name of **(non deterministic) Polynomial Dynamical Systems** [LBBLG91] which are usually described by a set of polynomial equations (here over the field $\mathbf{Z}_3$) of the form:

$$\begin{cases} Init(X) = 0 \\ \mathcal{T}(X, Y, X') = 0 \end{cases}$$

where $Init(X) = 0$ describes the set of initial states of the system.

Parallel composition and event hiding can be defined at the full intensional level.

Event hiding, say over variable Y_i, is simply obtained by considering the systm

$$S \setminus \{Y_i\} \stackrel{\text{def}}{=} (X, X', Y \setminus \{Y_i\}, \exists Y_i \mathcal{T}(X, Y, X'))$$

Parallel composition, written $S_1 \mid S_2$ can be defined by

$$S_1 \mid S_2 \stackrel{\text{def}}{=} (X^1 \cup X^2, X'^1 \cup X'^2, Y, \mathcal{T}_1 \oplus \mathcal{T}_2)$$

We do not enter the details here, but one should take care of the variable names to represent states when parallel composition is considered. In order to match the Definition 2.2 of parallel composition between iLTS, we always assume that the two systems to be composed have distinct state variable sets. Otherwise, some state variables renaming needs being performed before composition.

3.1 Computation of Symbolic Bisimulation

Assume given two ILTS $S_1 = (X^1, X'^1, Y, \mathcal{T}_1)$ and $S_2 = (X^2, X'^2, Y, \mathcal{T}_2)$, then the following algorithm computes the greatest symbolic bisimulation between S_1 and S_2:

Algorithm 1 *1. Define the polynomial $R_0(X^1, X^2) = 0$.*

2. Compute iteratively until stabilization the sequence of polynomials $(R_k(X^1, X^2))_k$ defined by:

$R_{k+1}(X^1, X^2)$ is the canonical generator of the $\equiv$-class of

$$\begin{cases} R_k(X^1, X^2) \\ \oplus\ \forall X'^1 \forall Y[(1 - \mathcal{T}_1(X^1, Y, X'^1) * \exists X'^2(\mathcal{T}_2(X^2, Y, X'^2) \oplus R_k(X'^1, X'^2))] \\ \oplus\ \forall X'^2 \forall Y[(1 - \mathcal{T}_2(X^2, Y, X'^2) * \exists X'^1(\mathcal{T}_1(X^1, Y, X'^1) \oplus R_k(X'^1, X'^2))] \end{cases} \quad (6)$$

Call $R(X^1, X^2)$ the result.

In order to prove the termination and correctness of this algorithm, we first recall some classic definitions and results (cf. [BBK87, Mil89a, Mil90]).

Definition 3.2 (Projective Equivalences) *Let $t_1 = (Q_1, A, \rightarrow_1)$ and $t_2 = (Q_2, A, \rightarrow_2)$ be two transition systems (labeled over some set A). For $k \in \mathbf{N}$, we define the relations $\equiv_k \subseteq Q_1 \times Q_2$ by:*

- $q_1 \equiv_0 q_2$ *for all* $(q_1, q_2) \in Q_1 \times Q_2$,
- $q_1 \equiv_{k+1} q_2$ *iff*
  1. *for all* $q_1 \xrightarrow{a} q_1'$, *there exists* $q_2 \xrightarrow{a} q_2'$ *s.t.* $q_1' \equiv_k q_2'$,
  2. *reciprocally, for each* $q_2 \xrightarrow{a} q_2'$, *there exists* $q_1 \xrightarrow{a} q_1'$ *s.t.* $q_1' \equiv_k q_2'$.

Theorem 4 [Mil89a, Mil90] *Let $t_1 = (Q_1, A, \rightarrow_1)$ and $t_2 = (Q_2, A, \rightarrow_2)$ be two finite transition systems (labeled over some set A). For all $q_1 \in Q_1$ and $\forall q_2 \in Q_2$, we write $q_1 \underline{\leftrightarrow} q_2$ whenever there exists a bisimulation between t_1 and t_2 which relates q_1 and q_2. Then there exists $p \in \mathbf{N}$ s.t. $\equiv_p = \equiv_{p+1} = \equiv_{p+2} = \ldots$, and $\underline{\leftrightarrow} = \bigcap_{k=0,1,\ldots,p} \equiv_k$*

We are now ready to state:

Theorem 5 (Termination and Correctness) *The algorithm terminates and at the end, $R(x^1, x^2) = 0$ iff $x_1 \underline{\leftrightarrow} x_2$.*

Proof By Theorem 4, it is enough to show that $R_k(x^1, x^2) = 0$ iff $x_1 \equiv_k x_2$, for all $k \in I\!N$. We make the proof by induction over k.

1. For $k = 0$, it is obvious.
2. Suppose $R_{k+1}(x^1, x^2) = 0$.
   We consider a pair $(x^1, x^2) \in Sol(R_{k+1}(X^1, X^2))$. From the definition of $\oplus$ operator and Properties (1), (2) and (3), Equation (6) is equivalent to say that (x^1, x^2) belongs to:

$$
\begin{array}{l}
Sol(R_k(X^1, X^2)) \\
\bigcap \ \{(x^1, x^2) \mid \forall x'^1, \forall y \ \mathcal{T}_1(x^1, y, x'^1) = 0 \Rightarrow \exists x'^2 \ \mathcal{T}_2(x^2, y, x'^2) = 0 \ \wedge \ R_k(x'^1, x'^2) = 0\} \\
\bigcap \ \{(x^1, x^2) \mid \forall x'^2, \forall y \ \mathcal{T}_2(x^2, y, x'^2) = 0 \Rightarrow \exists x'^1 \ \mathcal{T}_1(x^1, y, x'^1) = 0 \ \wedge \ R_k(x'^1, x'^2) = 0\}
\end{array}
$$

   which by induction hypothesis for R_k and Definition 3.2, is in turn equivalent to say that

$$
\begin{array}{l}
x^1 \equiv_k x^2 \\
\wedge \ \forall x'^1, \forall y, x^1 \xrightarrow{y} x'^1 \ \Rightarrow \ \exists x'^2, x^2 \xrightarrow{y} x'^2 \ \wedge \ x'^1 \equiv_k x'^2 \ , \\
\wedge \ \forall x'^2, \forall y \ x^2 \xrightarrow{y} x'^2 \ \Rightarrow \ \exists x'^1, x^1 \xrightarrow{y} x'^1 \ \wedge \ x'^1 \equiv_k x'^2
\end{array}
$$

   in other words, $x^1 \equiv_{k+1} x^2$.

$\square$

Expression 6 can be made simpler when deterministic systems are to be compared. This is the case in Section 4, when our theory is applied to the synchronous language SIGNAL. Indeed, in this case the computation of R can be performed according to some standard computation of a *greatest invariant*:

Definition 3.3 *Given $S = (X, X', Y, \mathcal{T})$ a (n,m)-ILTS, a set of states $E \in \mathbf{Z}_3^n$ is* **invariant** *iff*

$$\forall x, x' \in \mathbf{Z}_3^m, \forall y \in \mathbf{Z}_3^m (x \in E \wedge \mathcal{T}(x, y, x') = 0) \text{ implies } x' \in E$$

Algorithm 2 1. *Compute the admissible events from a given state in each system: for system S_1, compute the canonical generator of $A_1(X^1, Y)$ of $[\exists X'^1 \mathcal{T}_1(X^1, Y, X'^1)]_\equiv$, and similarly compute $A_2(X^2, Y)$ for S_2.*

2. *Compute the canonical generator $D_0(X^1, X^2)$ of $[\forall Y (A_1(X^1, Y) - A_2(X^2, Y))]_\equiv$.*
   Solutions of D_0 are pair of states (x^1, x^2) which accept the same labels on their output transitions, i.e. which have the same admissible *events.*

3. *Now the greatest invariant has to be computed. We iteratively compute polynomial D_k until stabilization as follows:*

$$\begin{array}{l} D_{k+1}(X^1, X^2) \text{ is the canonical generator of the } \equiv\text{-class of} \\ \forall X'^1 \forall X'^2 \forall Y [(1 - (\mathcal{T}_1(X^1, Y, X'^1) \oplus \mathcal{T}_2(X^1, Y, X'^1)) * D_k(X'^1, X'^2)] \end{array} \quad (7)$$

Call $D(X^1, X^2)$ the result.

Theorem 6 *Assume given two ILTS $S_1 = (X^1, X'^1, Y, \mathcal{T}_1)$ and $S_2 = (X^2, X'^2, Y, \mathcal{T}_2)$, and let $R(X^1, X^2)$ (resp. $D(X^1, X^2)$) be the result of Algorithm 1 (resp. Algorithm 2), then $R(X^1, X^2) = D(X^1, X^2)$.*

Proof Let us write $\mathcal{R}, \mathcal{D}, \mathcal{D}_0, ...$ for the sets denoted by polynomials $R, D, D_0, ...$ Because the systems are deterministic, it can be shown that

$$\text{"}\mathcal{R} \text{ is an invariant of } S_1 \mid S_2\text{"} \quad (8)$$

(Note that in general, Equation (8) does not hold.) Now, additionally, the fact that $\mathcal{R} \subseteq \mathcal{D}_0$ entails $\mathcal{R} \subseteq \mathcal{D}$. Indeed, by construction, $\mathcal{D}$ is the greatest invariant contained in $\mathcal{D}_0$.
Now to show that $\mathcal{D} \subseteq \mathcal{R}$ it is sufficient to prove $\mathcal{D}$ is a bisimulation, since $\mathcal{R}$ is the greatest bisimulation. Let $(x^1, x^2) \in \mathcal{D}$ and let $x^1 \xrightarrow{y} x'^1$. Because $(x^1, x^2) \in \mathcal{D}_0$, there exists $x^2 \xrightarrow{y} x'^2$. Then $(x^1, x^2) \xrightarrow{y} (x'^1, x'^2)$ is a transition of $S_1 \mid S_2$, and because $\mathcal{D}$ is invariant, we have $(x'^1, x'^2) \in \mathcal{D}$, and we are done. $\square$

3.2 Algorithms to build the reduced system

3.2.1 Building the minimal ILTS

In this section, we assume given an (n,m)-ILTS $S = (X, X', Y, \mathcal{T})$ for which the greatest symbolic (auto-)bisimulation $R(X^1, X^2)$ has already been computed. We also assume that the number of equivalence classes is known, say k. It is then enough to have $l = [log_3(k)] + 1$ [2].

It is now possible to define a (l,m)-ILTS which represents the *reduction* of S w.r.t. R, assuming we are provided with an encoding ϕ of the partition $\{[x]_R | x \in \mathbf{Z}_3^n\}$, where an encoding is defined as follows:

[2] $[\alpha]$ denotes the greatest integer less than or equal to *alpha*.

Definition 3.4 *Given $n \in I\!N$, a variable set $X = \{X_1, ..., X_n\}$ and a partition $\Pi = (\Pi_i)_{i\in\{1,...,k\}}$ of $\mathbf{Z}_3^n$ we say that ϕ is an encoding of Π over the variable set $Z = \{Z_1, ..., Z_l\}$ if $\phi \in \mathbf{Z}_3[Z, X]$ and it satisfies:*

1. *for all $x \in \mathbf{Z}_3^n$, $\phi(Z, x) = 0$ has a unique solution, which encodes the unique subset Π_x containing x, and*

2. *Conversely, given $z \in \mathbf{Z}_3^l$, if $Sol(\phi(z, X)) \neq \emptyset$ then it matches one $\Pi_i \in \Pi$.*

Clearly, when the cardinal of Π is some 3^r ($r \geq 1$), it is possible to define an encoding ϕ over a variable set $Z = \{Z_1, ..., Z_{r+1}\}$, each $z \in \mathbf{Z}_3^{r+1}$ encoding exactly one Π_i. Otherwise, there might be some z left which cannot correspond to any Π_i (think of the case when $k = 2$, then $\Pi = \{\Pi_1, \Pi_2\}$. One variable Z_1 is sufficient, leading to three points $0, 1, -1 \in \mathbf{Z}_3$ to be injectively mapped onto Π). We refer to Section 3.3 for the construction of ϕ when the partition is induced by the symbolic (auto-)bisimulation relation R.

Assuming such ϕ exists, we can now define $\mathcal{T}'(Z, Y, Z')$ by:

$$\mathcal{T}'(Z, Y, Z') = \exists X \exists X'(\phi(Z, X) \oplus \phi(Z', X') \oplus \mathcal{T}(X, Y, X')) \tag{9}$$

and call S_R the (l, m)-ILTS $(Z, X', Y, \mathcal{T}')$.
Intuitively speaking, $\mathcal{T}'(z, y, z') = 0$ whenever there exists one x in the class encoded by z and one x' in the class encoded by z' such that $x \xrightarrow{y} x'$ holds in the original system S. This is exactly the definition of the quotient model. Moreover, when initial states are taken into account in the system, thus defined by some polynomial $Init(X)$, the corrsponding $Init'(Z)$ in the reduced model can be computed by $Init'(Z) = \exists X \phi(Z, X) \oplus Init(X)$.

Theorem 7 *Relation $\phi(Z, X)$ is the greatest bisimulation between S and S_R.*

Proof The proof that ϕ is a bisimulation is easy. It is the greatest because we have proved that R is the greatest auto-bisimulation (see Theorem 3). $\square$

3.3 An algorithm to obtain an encoding ϕ

3.3.1 Implementation of polynomials

To implement the operations on polynomials, a powerful symbolic calculus system is available. Moreover, it is possible to implement the canonical representative of $\equiv$-classes by ternary decision diagrams (TDD) [LBBLG91], a slight extension of Binary Decisions Diagrams (BDDs) [Bry86, BRB90], which are very efficient in boolean algebra and other areas ([Cou91] for boolean circuits verification). Like for the BDD, we obtain in this way a canonical representation of the polynomial functions.

Given the ring of polynomial functions $A[X] = \mathbf{Z}_3[X]/_{<X^3-X>}$ and a variable X_i let us define the polynomial functions: $e_i^1 = -X_i^2 - X_i$, $\quad e_i^2 = -X_i^2 + X_i$, $\quad e_i^3 = 1 - X_i^2$. The three polynomial functions associated with each variable have interesting properties:

$$\begin{cases} (e_i^\alpha)^2 = e_i^\alpha \text{ for } \alpha = 1, 2, 3 \\ e_i^\alpha e_i^\beta = 0 \text{ for all } \alpha \neq \beta \\ e_i^1 + e_i^2 + e_i^3 = 1 \end{cases}$$

Proposition 8 *If $P(X)$ is an element of $A[X_1, \dots, X_n]$ then there exists a unique triple (P_1, P_2, P_3) of elements of $A[X_2, \dots, X_n]$ such that: $P(X) = e_1^1 P_1 + e_2^1 P_2 + e_1^3 P_3$. Moreover, these three polynomials have the following form:*

$$P_1 = P(1, X_2, \dots, X_n), \ P_2 = P(-1, X_1, \dots, X_n), \ P_3 = P(0, X_1, \dots, X_n,).$$

From an algebraic point of view the families $\{e_i^1, e_i^2, e_i^3\}$ are families of orthogonal idempotents and the following corollary states that the algebra $A[X_1, \dots, X_n]$ is finitely generated by families of orthogonal idempotents.

Corollary 9 *Every element of $A[X_1, \dots, X_n]$ has a unique decomposition:*

$$P(X) = \sum_{\alpha \in [1,2,3]^n} a_\alpha e_1^{\alpha_1} \dots e_n^{\alpha_n} \text{ where, } a_\alpha \in \mathbf{Z}_3$$

The proof is straightforward using a recurrence on the number of variables.
The decomposition of the polynomial functions using the basis of monomials $e_1^{\alpha_1} \dots e_n^{\alpha_n}$ is not interesting for an efficient implementation of the sum and product of polynomial functions since it needs 3^n coefficients. It is possible to have a more compact representation but at the price of choosing an order on the variables. Moreover the efficiency of the representation will depend on the chosen order.

Definition 3.5 *Given a polynomial function P in $A[X]$, and an order $X_1 \prec X_2 \prec \dots \prec X_n$ on the variables, a h-expression of P is:*

- *either $P(X) = c_1 e_i^1 + c_2 e_i^2 + c_3 e_i^3$ where $c_i \in \mathbf{Z}_3$*
- *or $P(X) = e_i^1 P_1 + e_i^2 P_2 + e_i^3 P_3$ where the P_i are h-expressions with variables greater than X_i.*

An h-expression may be pictured as a ternary tree. For example the polynomial function:

$$P(X_1, X_2, X_3) = X_1^2X_2^2X_3^2 - X_1^2X_2^2X_3 - X_1^2X_2X_3^2 + X_1^2X_2X_3 - X_1^2X_3^2 - X_1^2X_3 - X_1X_2^2X_3^2 + X_1X_2^2X_3 + X_1X_2X_3^2 - X_1X_2X_3 + X_1X_3^2 + X_1X_3 + X_2^2X_3^2 - X_2^2X_3 - X_2X_3^2 + X_2X_3 - X_3^2 - X_3$$

is represented as in Figure 2.

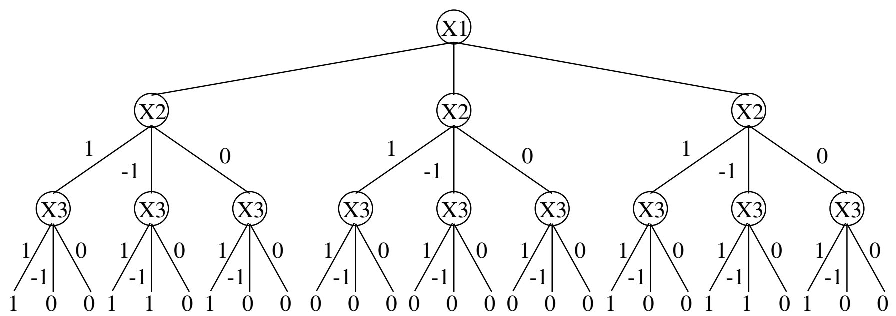

Figure 2: An example of ternary tree

where the left son of a subtree with root X_i represents the e_i^1 factor, the middle son the e_i^2 factor and the right son the e_i^3 factor. The leaves are labeled with the numerical coefficients. A polynomial function has several h-expressions. All h-expressions have nice properties with respect to polynomial operations.

Proposition 10 *Given an operation op, (sum, product...), and two h-expressions of two polynomials (with the same order on the variables):*

$$P(X) = e_1^1 P_1 \ + e_1^2 P_2 \ + e_1^3 P_3, \ P'(X = e_1^1 P_1' \ + e_1^2 P_2' \ + e_1^3 P_3'$$

we have $P \ op \ P' \ = \ e_1^1(P_1 \ op \ P_1') \ + \ e_1^2(P_2 \ op \ P_2') \ + \ e_1^3(P_3 \ op \ P_3')$

This orthogonality property would allow efficient implementations of polynomial operations if the number of monomials in such h-expressions were not, as previously noticed, two big in general for practical use.
Two ideas give the clue to an efficient implementation of polynomial functions: the first one is to reduce the h-expressions of the form $P(X) = e_i^1 P_1 \ + e_i^2 P_2 \ + e_i^3 P_3$ by eliminating the idempotents e_i^α when $P_1 = P_2 = P_3$ and replacing the former expression by the common value P_1. This is possible since the sum of these idempotents is 1 and the resulting expression is still a h-expression. The second idea comes from the fact that in the tree representation of the h-expressions, many subtrees are identical. Thus it is tempting to have a representation sharing the common subtrees. The two ideas lead to a generalisation of Bryant graphs [Bry86, CM90, Cou91] and fortunately these graphs retain in some way the orthogonality property with respect to polynomial operations. The graph representation of the previous example is given Figure 3.

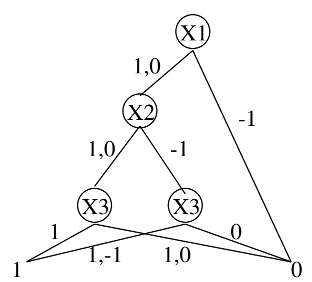

Figure 3: An example of TDD

The efficiency of this representation for elementary operations on polynomial functions (sum, product, composition) allows us to use principal generators to represent $Sol(P)$. The various algorithms (sets inclusion checking, largest invariant computation ...) can be implemented using this representation.

3.3.2 Building relation ϕ

The goal of this section is to explain how to obtain relation ϕ, upon which the construction of the reduced ILTS (Section 3.2.1) relies on. It is the key point of our method and it is worthwhile noting that this step is dependant on the choice for sets/polynomials implementation.

Intuitively speaking, the starting point is to work on the TDD representation of relation $R(X^1, X^2) \subseteq \mathbf{Z}_3^n \times \mathbf{Z}_3^n$ denoting the symbolic auto-bisimulation of a given (n,m)-ILTS (see Section 3.1). More precisely, we shall define an algorithm based on the representation of R by a TDD with the particular reorder of the variables $X_1^1 \prec X_2^1 \prec ... \prec X_n^1 \prec$

$X_1^2 \prec X_2^2 \prec ... \prec X_n^2$. In this TDD, at the end of every path x^1 (over variables in X^1) the subtree corresponds the R-equivalence class of x^1. Besides, this property is used to compute "on the fly" the number of R-equivalence classes, namelly k ; also, during this unique enumeration of the state space of the reduced model, we introduce one by one the variables Z_i that are necessary to represent the reduced ILTS (as described Section 3.2.1), and mostly we build the useful encoding $\phi(Z, X)$, which is the key object for the definition of the reduced ILTS.

Remark 2 *In order to be more efficient and because in our systems an initial (set of) state(s) is defined (by some polynomial $Init(X)$), we only consider equivalence classes that are reachable, i.e. which contain at least one reachable state. This is be performed on the fly: when a new equivalence class is met in TDD R, it is taken into account only if its intersection with the "orbite" of the system is non empty. This polynomial representing this orbite is computed by a classical fixed-point computation (it is a forward-computation from $Init(X)$).*

We assume the set of R-equivalence classes, say C, can be traversed on the basis of the TDD of R (according to the order $X_1^1 \prec X_2^1 \prec ... \prec X_n^1 \prec X_1^2 \prec X_2^2 \prec ... \prec X_n^2$).
The working data structure is a list which elements are of type:

```
struct var_cell_
{
    tdd_indexindex_type novar;    /* indexindex */
    tdd_index_type      rangvar;  /* index */
    tdd                 B1;       /* branches of the TDD */
    tdd                 B2;
    tdd                 B3;
    struct var_cell_    *next;    /* next element in the ``stack''-list */
};
```

`novar` is the index of the variable at the root of the slot.
`rangvar` is the maximal index of the variables in the list.
`B1`, `B2`, `B3`, when not `NULL`, are pointers to already built TDDs.
A TDD is built only when its three branches defined.
We do not give here the full description of this algorithm, but the following example is proposed to illustrate the method:

Example 1 *In the Figure 4, already built TDD are drawn as triangles with solid bold lines, whereas not already built ones are drawn in dashed bold lines.*
When a TDD is built (then the corresponding tree is full) and when a new class is considered, the arrival of a new class, say C_{10} in the example, will force to introduce one more variable for the encoding. here variable Z_3.
Now, when classes C_{10}, C_{11} and C_{12} have been treated, the 4-th TDD at root Z_1 has been attached to branch $B1$ of the Structure (2).
Structure (1) is not yet a TDD (not built yet), but will be when classes C_{14} and C_{15} will be plugged, i.e. when its three branches will be allocated. Similarly for Structure (2) and Structure (3).

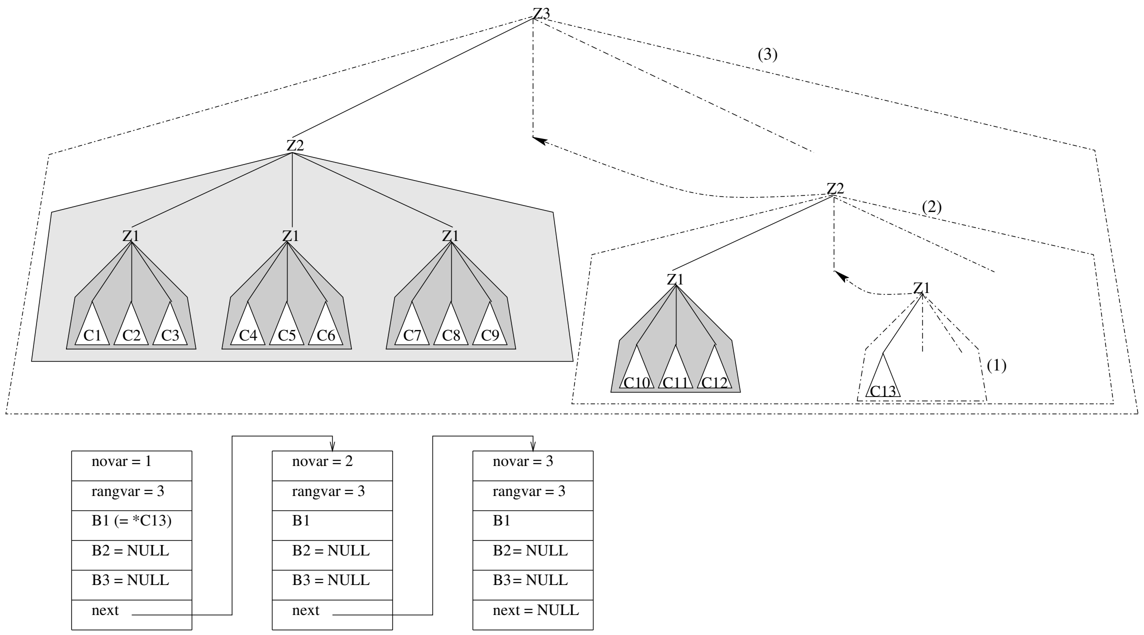

Figure 4: Construction of $\phi(Z, X)$

Assume now, the set of classes has cardinal 16, *i.e. the last class is* C_{16}*. The idea is now run a completion for the remaining non allocated branches by pointing them to TDD* **1**, *see Figure 5.*

4 Integration environment

4.1 The Signal language

In order to derive an ILTS models from a textual specification, we use the high level equational data-flow synchronous language Signal [BLJ91][3]. This language is widely used to specify and to implement reactive systems as well as to verify their properties. There exists a lot of examples using the Signal environment: among them, a production cell [ALGMR95], a power transformer station controller [LBMR96], an experiment with reactive data-flow tasking in active robot vision [MRMC98], ...

The original multi-clock data-flow synchronous language Signal manipulates a set of *signals*; each signal `A` denotes an unbounded series of typed values $(\mathtt{A}_t)_{t \in \mathcal{T}}$, indexed by time t in a time domain $\mathcal{T}$. $\perp$ is a particular value which denotes the absence of the signal. We call *clock of* `A` the set of instants t when `A` is not absent, i.e. $\mathtt{A}_t \neq \perp$. Two signals with the same clock are called *synchronous*. The *kernel*-language Signal is based on four operations, defining primitive processes by equations, and a parallel composition to combine equations, as well as a signal hiding to internalize them.

[3]developed in the EP-ATR research Group of the IRISA/INRIA Institute.

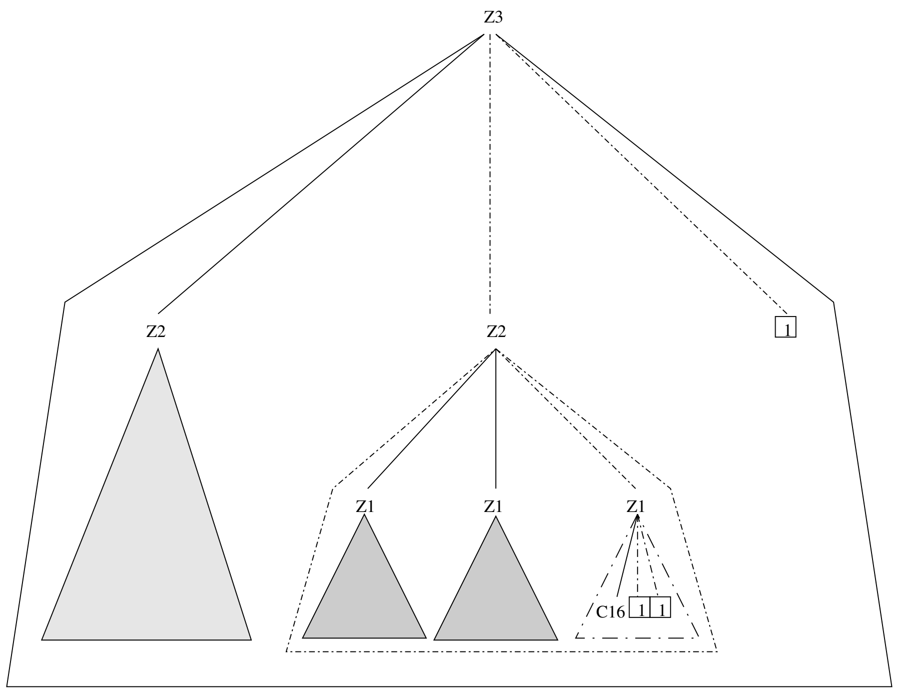

Figure 5: Final stage of the construction

In order to simplify the presentation, we shall restrict to the boolean fragment of SIGNAL language; that is the type domain is *true*, *false* or *absent*. The constructors of the language are equations of the form $\mathtt{A} :=< \mathtt{expression} >$, as well as a parallel composition and an event hiding.

• **Static synchronous operator** $\mathtt{A} := \mathtt{p}(\mathtt{A_1}, \ldots, \mathtt{A_n})$ is a boolean function of data $\mathtt{A_1}, \ldots, \mathtt{A_n}$ at each instant t. This instruction requires all referred variables to have the same clock.

• **Deterministic merge operator**, written $\mathtt{A} := \mathtt{A1}\,\mathtt{default}\,\mathtt{A2}$, A has the value of A1 when A1 is present, otherwise it has the value of A2. Its clock is the union of those of A1 and A2.

• **Selection operator** of the form $\mathtt{A} := \mathtt{A1}\,\mathtt{when}\,\mathtt{B}$ links A with A1 when the boolean B has value *true*. The result can be seen as a down-sampling of a signal A1. The clock of A is the intersection of that of A1 and the set of instants when boolean B has value *true*.

• **Delay** (a dynamic synchronous operator) $\mathtt{A} := \mathtt{B}\,\$1$ gives access to the last value of signal B. A and B have equal clocks. The memorizing of last values will give raise to states (see below).

• **Parallel composition** of processes is noted $|$ and consists in the conjunction of the equations (systems); it is then associative and commutative.

• **Signal hiding** $\backslash\{\mathtt{A}\}$ hides any occurrence of signal A; it is internalized.

4.2 Associated Intensional Labeled Transition System

Logical SIGNAL programs can be translated into polynomials equations over $\mathbf{Z}_3$, following the principle of coding the possible values of a boolean signal `A` by a variable a: values for a will respectively be 1, -1 and 0 and are respectively interpreted by "`A` is *present* and *true*", "`A` is *present* and *false*", "`A` is *absent*"[4]. Therefore, any signal `A` can be associated its *clock* a^2, and two synchronous signals `A` and `B` satisfy $a^2 = b^2$.

Operators	Clock equations	Evaluations
$\mathtt{A} := \mathtt{not}\ \mathtt{A}_1$	$a^2 = {a_1}^2$	$a = -a_1$
$\mathtt{A} := \mathtt{A}_1\ \mathtt{and}\ \mathtt{A}_2$	$a^2 = {a_1}^2 = {a_2}^2$	$a = a_1 a_2(a_1 a_2 - a_1 - a_2 - 1)$ $a_1^2 = a_2^2$
$\mathtt{A} := \mathtt{A}_1\ \mathtt{or}\ \mathtt{A}_2$	$a^2 = {a_1}^2 = {a_2}^2$	$a = a_1 a_2(1 - a_1 a_2 - a_1 - a_2)$ $a_1^2 = a_2^2$
$\mathtt{A} := \mathtt{A}_1\ \mathtt{default}\ \mathtt{A}_2$	$a^2 = {a_1}^2 + (1 - {a_1}^2){a_2}^2$	$a = a_1 + (1 - {a_1}^2)a_2$
$\mathtt{A} := \mathtt{A}_1\ \mathtt{when}\ \mathtt{B}$	$a^2 = {a_1}^2(-b - b^2)$	$a = a_1(-b - b^2)$
$\mathtt{A} := \mathtt{B}\$1$	$a^2 = b^2$	$x' = b + (1 - b^2)x$ $a = b^2 x$

Table 1: Synchronization constraints and the boolean signal evaluation

Table 1 shows how the programs are transformed into polynomial equations (we refer to [LBBLG89, LBBLG91] for more details), leading to an ILTS models semantics. Nevertheless, the delay operator \$ deserves some extra explanations. A delay requires to memorize the last value (then different from 0) of the signal into a (state) variable, say x. Translating $\mathtt{A} := \mathtt{B}\,\$1$, imposes to introduce two auxiliary equations: (1) $x' = b + (1 - b^2)x$, where x' denotes the next value of state variable x, expresses the dynamics of the system. (2) $a = b^2 x$ delivers the value of the delayed signal according to the memorization in state variable x.

The translation of Table 1 is automatically performed by the SIGNAL compiler. The automata semantics can then be used as a basis for the verification of SIGNAL programs. As an example, according to Table 1, the implicit transition of the ILTS associated to a simple program of the form $A := B\$1$, is given by

$$\mathcal{T}(X, \{a, b\}, X') = (X' - [b + (1 - b^2) * X]) \oplus (a - b^2 * X) \oplus (a^2 - b^2)$$

4.3 The "extended" SIGALI tool

The tool SIGALI [DLB97], offering algebraic polynomial computations was developed. It relies on an implementation of polynomials by Ternary Decision Diagram (TDD) (for three valued logics) in the same spirit of BDD [Bry89], but where the paths in the data structures are decorated by values in $\{-1, 0, 1\}$ instead of $\{0, 1\}$. This tool performs classic temporal logics specification verifications[5], whereas until now, SIGNAL programs

[4] General SIGNAL programs, with other type values, can also be treated by only coding information of presence of absence of non-boolean signals.

[5] However, we can also edal with numerical aspects using ADDs, Arithmetic Decision Diagrams [ML98]

equivalence checking was made extensionally: the little tool SIGAUTO, developed two years ago, exports the TDD generated by SIGALI in order to plug other verification tools, e.g. such as ALDEBARAN [Fer84] or FCTOOLS [BRRD96]. So the result can be submitted to the tool sets for further analysis, graphical depiction, bisimulation computation, quotient computation, etc. The plug-in is achieved with the package OPEN/CAESAR.
Obviously, the size of the generated transition systems limits the extensional methods. For instance, *the flexible manufacturing cell* [MBLL98] is represented by a transition system with 58 state variables and 11 event variables; that is to say, this fexible manufacturing cell can be represented by an automaton of 2^{58} possible states (also the set of labels has cardinal 2^{11}).

The intensional methods for bisimulation checking, as proposed in this paper, perfectly fits the spirit of the tool SIGALI: the "polynomial language" provides the programmer with an intermediate language to describe algorithms over sets, in an intensional way, without bothering with the underlying implementation.
We have then improved the ILTS models semantics by implementing the algorithm of Section 3 for the symbolic bisimulation equivalence checking and the model. Although the implementation phase is achieved, the functionalities are still under testing. For this reason, we postpone to the next report the real applications of our methods.

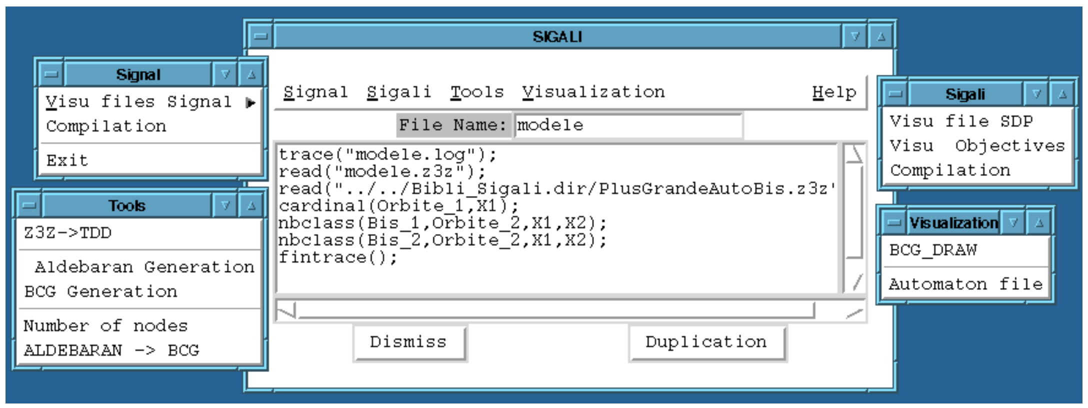

Figure 6: The Sigali tool

References

[ALGMR95] T.P. Amagbegnon, P. Le Guernic, H. Marchand, and E. Rutten. Signal : The specification of a generic, verified production cell controller. *Formal Development of Reactive Systems – Case Study, Production Cell, Lecture Notes in Computer Science 891, chapitre VII*, pages 115–129, January 1995.

[BBK87] J. C. M. Baeten, J. A. Bergstra, and J. W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. In *Proc. PARLE'87, vol. II: Parallel Languages, Eindhoven, LNCS 259*, pages 94–111. Springer-Verlag, June 1987.

[BLJ91] A. Benveniste, P. Le Guernic, and C. Jacquemot. Synchronous programming with events and relations: the SIGNAL language and its semantics. *Science of Computer Programming*, 16:103–149, 1991.

[BRB90] K. S. Brace, R. L. Rudell, and R. E. Bryant. Efficient implementation of a bdd package. *27th ACM/IEEE design automation conference*, pages 40–45, 1990.

[BRRD96] A. Bouali, A. Ressouche, V. Roy, and R. De Simone. The fc2tools set. In *Proc. of the 5th Int. Conf. AMAST'96, Munich, Germany, LNCS 1101*, July 1996.

[Bry86] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[Bry89] R. E. Bryant. Symbolic boolean manipulation with ordered binary decision diagrams. *CM computing Surweys*, pages 293–318, September 1989.

[CM90] O Coudert and J.-C. Madre. A verified frame work for the formal verification of sequential circuit. In *Conference on Computer- Aided design, IEEE*, pages 126–130, November 1990.

[Cou91] O. Coudert. Siam, une boite à outils pour la preuve formelles de systèmes séquentielles. *Thèse de ENS des télécommunications*, October 1991.

[DLB94] B. Dutertre and M. Le Borgne. Control of polynomial dynamic systems : an example. Technical Report 2193, Institut National de Recherche en Informatique et en Automatique (INRIA), January 1994.

[DLB97] B. Dutertre and M. Le Borgne. Sigali: un système de calcul formel pour la vérification de programmes signal. Technical report, Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA), July 1997.

[DR94] R. De Simone and A. Ressouche. Compositional semantics of esterel and verification by compositional reductions. *Proc. CAV'94, LNCS 818*, 1994.

[Dut92] B. Dutertre. *Spécification et preuve de systèmes dynamiques*. PhD thesis, Université de Rennes I, September 1992.

[Fer84] J.-Cl. Fernandez. *ALDEBARAN: un Système de Vérification par Réduction de Processus Communicants*. Thèse de Doctorat, Univ. Joseph Fourier-Grenoble I, France, July 1984.

[GJ94] J. Gunnarsson and Plantin J. Control law synthesis for a discrete event system. Technical Report LiTH-ISY-R-, Linköping University, 1994.

[HLR93] N. Halbwachs, F. Lagnier, and P. Raymond. Synchronous observers and the verification of reactive systems. In *Proc. AMAST'93, Enschede, NL*, June 1993.

[LBBLG89] M. Le Borgne, A. Benveniste, and P. Le Guernic. Polynomial ideal theoretic methods in discrete events and hybrid dynamical systems. In *Proc. of the 28th IEEE Conference on Decision and Control*, volume 3, pages 2695–2700, December 1989.

[LBBLG91] M. Le Borgne, A. Benveniste, and P. Le Guernic. Polynomial dynamical systems over finite fields. In G. Jacob and F. Lamnabhi-Lagarrigue, editors, *Algebraic Computing in control*, volume 165 of *Lecture Notes in Control and Information Sciences*, pages 212–222, March 1991.

[LBMR96] M. Le Borgne, H. Marchand, and E. Rutten. Formal verification of SIGNAL programs: Application to a power tranformar station controller. In *Proc. of the 5th Int. Conf. AMAST'96, Munich, Germany, LNCS 1101*, pages 270–285, July 1996.

[MBLL98] H. Marchand, P. Bournai, M. Le Borgne, and P. Le Guernic. A design environment for discrete-event controllers based on the signal language. In *1998 IEEE International Conf. On Systems, Man, And Cybernetics*, San Diego, California, USA, October 1998.

[Mil89a] R. Milner. *Communication and Concurrency.* Prentice Hall Int., 1989.

[Mil89b] R. Milner. A complete axiomatisation for observational congruence of finite-state behaviours. *Information and Computation*, 81(2):227–247, 1989.

[Mil90] R. Milner. Operational and algebraic semantics of concurrent processes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, vol. B*, chapter 19, pages 1201–1242. Elsevier Science Publishers, 1990.

[ML98] H. Marchand and M. Le Borgne. Partial order control of discrete event systems modeled as polynomial dynamical systems. In *1998 IEEE International Conference On Control Applications*, Trieste, Italie, September 1998.

[MRMC98] E. Marchand, E. Rutten, H. Marchand, and F. Chaumette. Specifying and verifying active vision-based robotic systems with the signal environment. *Int. Journal of Robotics Research*, 17(4):418–432, April 1998.

[Par81] D. Park. Concurrency and automata on infinite sequences. In *Proc. 5th GI Conf. on Th. Comp. Sci., LNCS 104*, pages 167–183. Springer-Verlag, March 1981.

[RMC97] E. Rutten, E. Marchand, and F. Chaumette. An experiment with reactive data-flow tasking in active robot vision. *Software – Practice & Experience*, 27(5):599–621, May 1997.