Symbolic methods in the SIGNAL environment

S. Pinchinat and O. Kouchnarenko

Unpublished report

1 Introduction

The original multi-clock data-flow synchronous language SIGNAL [GB87, GBGM91, BLJ91] manipulates *signals* A, which denote unbounded series of typed values $(A_t)_{t\in\mathcal{T}}$, indexed by time t in a time domain $\mathcal{T}$.
An associated clock determines the set of instants at which values are present. This clock is a particular signals called `event` and is characterized only by its presence, and always has the value *true* (hence, its negation by *not* is always *false*).
The constructs of the language can be used in an equational style to specify the relations between signals, i.e. between their values and between their clocks. Systems of equations on signals are built using a parallel composition construct, thus defining *processes*. Data flow applications are activities executed over a set of instants in time. At each instant, input data is acquired from the execution environment; output values are produced according to the system of equations considered as a network of operations.

The equational nature SIGNAL makes that it relies on a formal model in terms of *polynomial dynamical equations systems (PDS)*, which can be seen as an equational representation of an automaton. This aspect is an originality of the SIGNAL approach compared to others, e.g. Esterel. This abstraction is automatically performed by the SIGNAL compiler and the resulting PDS can then be used as a formal basis for verification of SIGNAL programs.
The theory of PDS uses classical tools in algebraic geometry such as *ideals, varieties* and *morphisms* [CLO93]. This is the basis of the verification tool SIGALI [DB97], developed in the EP-ATR group. The techniques used in consist in manipulating the system of equations instead of the sets of solution, which avoids the enumeration of the state space. Each set of states is uniquely characterized by a polynomial and the operations on sets can be equivalently performed on the associated polynomials.
Therefore, a wide spectre of properties, such as liveness, invariance, reachability and attractivity can be checked.
At the moment, many algorithms for computing predicates states are available and we are developing the algorithm for the symbolic bisimulation computation.

The present report is organized as follows : Section 2 recalls the basic mathematical notions for polynomials. Section 3 introduces the models, as *implicit labeled transition systems*, parallel composition and event hiding constructs, as well as the behavioral equivalence so-called *symbolic bisimulation* with its congruence theorem w.r.t. the abovementioned

constructs. Next, in Section 4, polynomial dynamical systems are presented and provided with an implicit labeled transition system operational semantics. Verification techniques with the SIGALI tool are briefly explained and used to describe an algorithm computing the symbolic bisimulation.
Finally, Section 5 explains, through a toy example of SIGNAL program, how to use SIGALI to compute the symbolic bisimulation.

2 Preliminaries

In the following, $\mathbb{Z}/_{3\mathbb{Z}}$ denotes the finite field $\{-1,0,1\}$[1]. Let Z be a set of m variables $Z_1, ..., Z_m$.

We denote by $\mathbb{Z}/_{3\mathbb{Z}}[Z]$ (or $\mathbb{Z}/_{3\mathbb{Z}}[Z_1, ..., Z_m]$) the set of polynomials over variables $Z_1, ..., Z_m$ which coefficients range over $\mathbb{Z}/_{3\mathbb{Z}}$. For Z and U two sets of variables, we write $\mathbb{Z}/_{3\mathbb{Z}}[Z, U]$ instead of $\mathbb{Z}/_{3\mathbb{Z}}[Z \cup U]$ and for a variable Z_1, we simply write $\mathbb{Z}/_{3\mathbb{Z}}[Z_1]$ instead of $\mathbb{Z}/_{3\mathbb{Z}}[\{Z_1\}$.
We recall $\mathbb{Z}/_{3\mathbb{Z}}[Z]$ is provided with a structure of ring, with addition and multiplication. In order to simplify notations, we write Z for the set $\{Z_1, ..., Z_m\}$, and write $P(Z)$ instead of $P(Z_1, ..., Z_m)$.
In the following, a *(an algebraic) variety of* $\mathbb{Z}/_{3\mathbb{Z}}{}^m$ is any subset of $\mathbb{Z}/_{3\mathbb{Z}}{}^m$.

2.1 Varieties and ideals

Given a polynomial $P(Z_1, ..., Z_m) \in \mathbb{Z}/_{3\mathbb{Z}}[Z_1, ..., Z_m]$, we associate $Sol(P) \subseteq \mathbb{Z}/_{3\mathbb{Z}}{}^m$, its *algebraic variety* defined by :

$$Sol(P) \stackrel{\text{def}}{=} \{(z_1, ..., z_m) \in \mathbb{Z}/_{3\mathbb{Z}}{}^m | P(z_1, ..., z_m) = 0\}$$

composed of all the solutions of polynomial P.
Sol can be generalized to sets of polynomials by : for all $G \subseteq \mathbb{Z}/_{3\mathbb{Z}}[Z]$,

$$Sol(G) \stackrel{\text{def}}{=} \bigcap_{P \in G} Sol(P)$$

Now, define $LC(G) \subseteq \mathbb{Z}/_{3\mathbb{Z}}[Z]$ to be the set of all *linear combinations* of polynomials in G, that is

$$LC(G) \stackrel{\text{def}}{=} \{A_1 * P_1 + A_2 * P_2 + ... + A_t * P_t | (P_i \in G) \wedge (A_1, ..., A_t \in \mathbb{Z}/_{3\mathbb{Z}}[Z]\}.$$

$LC(G)$ is also called an *ideal* of $\mathbb{Z}/_{3\mathbb{Z}}[Z]$.

Because clearly $Sol(G) = Sol(LC(G))$, and without loss of generality, we shall consider ideals of polynomials instead of simply sets of polynomials. Reciprocally, for all variety $V \subseteq \mathbb{Z}/_{3\mathbb{Z}}{}^m$, the set $I(V)$ will denote

$$I(V) \stackrel{\text{def}}{=} \{P \in \mathbb{Z}/_{3\mathbb{Z}}[Z] | P(z) = 0\}$$

[1]The present paper can be generalized to any field $\mathbb{Z}/_{p\mathbb{Z}}$.

Henceforth, there is a natural correspondence between ideals and algebraic varieties. One can easily prove that :

Proposition 1 *For all variety V,*

$$Sol(I(V)) = V, \tag{1}$$

and, for all ideal G,

$$I(Sol(G)) = G + LC(\{Z_1^3 - Z_1, ..., Z_m^3 - Z_m\}) \tag{2}$$

Equation (2) shows that the correspondence between ideals and varieties can be made unique modulo linear combinations of polynomials $Z_1^3 - Z_1, ..., Z_m^3 - Z_m$. This is what we shall do in the following, by considering the quotient ring $\mathbb{Z}/_{3\mathbb{Z}}[Z]/LC(\{Z_1^3 - Z_1, ..., Z_m^3 - Z_m\})$ instead of simply $\mathbb{Z}/_{3\mathbb{Z}}[Z]$ (However, we still write $\mathbb{Z}/_{3\mathbb{Z}}[Z]$). Then Equation (2) becomes $I(Sol(G)) = G$.

2.2 Generators

In this theory, any ideal can be represented by an unique polynomial called *canonical generator*, with algebraic properties. The theoretical foundations are not described here but can be found in [Dut92].

Definition 2.1 (Generator) *Let $G \subseteq \mathbb{Z}/_{3\mathbb{Z}}[Z]$ be an ideal. $g \in \mathbb{Z}/_{3\mathbb{Z}}[Z]$ is called a* generator *of G whenever $g(z) = 0$ iff $z \in Sol(G)$, and it is* canonical *whenever $g^2 = g$.*

Set-theoretical constructs can be expressed in the dual framework of polynomials :

Proposition 2 *Let g_1 and g_2 be two canonical generators of some ideals G_1 and G_2 respectively. Then :*

- *$1 - g_1^2$ is the canonical generator of $({\mathbb{Z}/_{3\mathbb{Z}}}^n \setminus Sol(G_1))$,*
- *$g_1 \oplus g_2$ is the canonical generator $Sol(G_1) \cap Sol(G_2)$ (where we define $f \oplus h = (f^2 + h^2)^2$),*
- *$g_1^2 * g_2^2$ is the canonical generator $Sol(G_1) \cup Sol(G_2)$,*
- *$Sol(G_1) \subseteq Sol(G_2)$ iff $(1 - g_1^2) * g_2$ evaluates to zero.*

Finally, we introduce *existential abstractions* (or existential quantifications) of polynomials w.r.t. some variables. This construct will be useful for the definition of *events hiding* construct over the models.

Definition 2.2 (Existential abstraction) *For $V \subseteq {\mathbb{Z}/_{3\mathbb{Z}}}^m$ and a sub-order of indexes $\iota = \{i_1, ..., i_k\} \subseteq \{1, ..., m\}$, we define $V \setminus \iota$ as the set of all vectors $(z_{i_1}, ..., z_{i_k})$ obtained by deleting from the vectors $(z_1, ..., z_m)$ of V all components of indexes in ι.*

For example,
$\{(1, z_2, z_3, z_4)|z_i \in \mathbb{Z}/_{3\mathbb{Z}}\} \setminus \{3,4\} = \{(1, z_2)|\exists z_3, z_4 \in \mathbb{Z}/_{3\mathbb{Z}}, (1, z_2, z_3, z_4) \in V\}$.
This set construct can be dually defined over polynomials by : let $\bar{Z} \subseteq Z$. Then we can define a mapping $\exists\bar{Z} : \mathbb{Z}/_{3\mathbb{Z}}[Z] \to \mathbb{Z}/_{3\mathbb{Z}}[Z \setminus \bar{Z}]$ s.t. $Sol(\exists\bar{Z}P) = Sol(P) \setminus \iota$ where $\iota = \{i|Z_i \in \bar{Z}\}$.

We refer to [Dut92] for the complete definition and the computation of a canonical generator of $Sol(\exists\bar{Z}P)$.

3 The models

3.1 Implicit labeled transition systems

Definition 3.1 (Implicit labeled transition systems (ILTS)) *An* m-dimensional implicit labeled transition system (or m-ILTS) *is a structure* $T = (Q, Z, \to)$ *where*

- Q *is set of states,*
- Z *is a set of* m *variables* $Z_1, ..., Z_m$*, and*
- $\to \subseteq Q \times \mathbb{Z}/_{3\mathbb{Z}}[Z] \times Q$*. Each transition of* T *is labeled by a polynomial over the vector of variables* Z*.*

  We write $q \stackrel{P(Z)}{\to} q'$ *(or simply* $q \stackrel{P}{\to} q'$*), instead of* $(q, P(Z), q') \in \to$*, and* $q \stackrel{P(Z)}{\to}$ *whenever* $q \stackrel{P(Z)}{\to} q'$ *for some* q'*.*

  Also $q \stackrel{z}{\to} q'$ *means that there exists* $q \stackrel{P(Z)}{\to} q'$ *with* $P(z) = 0$ *and* $q \stackrel{z}{\to}$ *means* $q \stackrel{z}{\to} q'$ *for some* q'*.*

3.2 Explicit/Implicit labeled transition systems

Given an m-ILTS T, we can *explicit* it to get the (explicit) underlying transition system labeled over $\mathbb{Z}/_{3\mathbb{Z}}{}^m$: each transition $q \stackrel{P}{\to} q'$ gives rise to all transitions $q \stackrel{y}{\to} q'$ for each $y \in Sol(P)$. Call $Exp(T)$ the obtained "explicit" labeled transition system.

We assume the reader is familiar with the class of (explicit) transition systems as well as with the equivalence relation over states called *strong bisimulation* (see [Par81, Mil89] for example). Strong bisimulation will be used in the next section to explain symbolic bisimulation. We briefly recall that :

Definition 3.2 (Strong bisimulation) *Let* $t = (Q, A, \to)$ *be a transition system labeled over some set* A*. A bisimulation over* t *is a binary relation* $\rho \subseteq Q \times Q$ *s.t. for all* $(q_1, q_2) \in \rho$*, for all* $a \in A$*, if* $q_1 \stackrel{a}{\to} q_1'$ *there exists* $q_2 \stackrel{a}{\to} q_2'$ *with* $(q_1', q_2') \in \rho$*.*

It is well known that there exists a greatest bisimulation relation, which yields an equivalence relation between states. In the following, we write $\underline{\leftrightarrow}_t$ (or simply $\underline{\leftrightarrow}$) for the greatest strong bisimulation over states of t.

3.3 Operations over ILTS

The class of ITLS can be provided with many operations. Among them, the *parallel composition* and the *events hiding* offer powerful constructs to design more complex systems from simple ones.

Parallel composition of Definition 3.3 is very standard, and corresponds to the synchronous parallel composition of the underlying explicit transition system, see Remark 1.

Definition 3.3 (Parallel composition of ILTS) *Let $T_1 = (Q_1, Z, \to_1, I_1)$ be an m_1-ILTS and $T_2 = (Q_2, U, \to_2, I_2)$ be an m_2-ILTS, with possible common variables between Z and U. We assume also given a* compatibility *relation $co(Q_1, Q_2)$ over $Q_1 \times Q_2$ (We simply write co when it is clear from the context).*
The parallel composition *of T_1 and T_2, written $T_1 \mid T_2$, is $(co, Z \cup U, \to)$ where $(q_1, q_2) \overset{P_1(Z)\oplus P_2(U)}{\to} (q_1', q_2')$ in $T_1 \mid T_2$ whenever $(q_1', q_2') \in co$, $q_1 \overset{P_1(Z)}{\to}_1 q_1'$ in T_1 and $q_2 \overset{P_2(U)}{\to}_2 q_2'$ in T_2.*

Remark 1 *Very informally, it can be shown that given T_1 and T_2 two ILTS, there exists a construct $||$ over labeled transition systems s.t. $Exp(T_1 \mid T_2) = Exp(T_1) \; || \; Exp(T_2)$. Actually, $||$ construct can be expressed by*

$$\frac{q_1 \overset{z}{\to} q_1' \quad q_2 \overset{u}{\to} q_2' \quad z \text{ and } u \text{ agree on common components}}{(q_1, q_2) \overset{z \sqcup u}{\to} (q_1', q_2')}$$

where $z \sqcup u$ is the fusion of the vectors z and u according to common components.

Events hiding enables one to abstract from internal or communicating events that can be used to compose systems.

Definition 3.4 (Events hiding) *Let $T = (Q, Z, \to)$ be an m-ILTS, and let $\bar{Z} \subseteq Z$. We define $(T \setminus \bar{Z})$ by*

$$(T \setminus \bar{Z}) \overset{def}{=} (Q, Z \setminus \bar{Z}, \to_{\setminus \bar{Z}}) \quad \text{where } q_1 \overset{P'}{\to}_{\setminus \bar{Z}} q_2 \text{ iff } P' = \exists \bar{Z} P \wedge q_1 \overset{P}{\to} q_2$$

Example 1 *Figure 1 shows an example of an ILTS, in which M_k, P_i and Q_j are polynomials over variables Y_1, Y_2, that are not described here. An example is given in Section 5.2. We obtain labels like $(Y_1 - 1) \oplus (Y_2 - 1) \oplus \exists Y_3 (Y_3 - P_1(Y_1, Y_2)) \oplus \exists Y_4 (Y_4 - P_2(Y_1, Y_2))$. When hiding events Y_3 and Y_4, we get roughly the same ILTS but expressions $(Y_3 - P_1(Y_1, Y_2)), \ldots$ are removed from the polynomial labels.*

3.4 Symbolic bisimulation

We propose here an equivalence relation over sets of ILTS which definition is strongly inspired from DeSimone's symbolic bisimulation over reactive automata [DR94]

Definition 3.5 (Symbolic Bisimulation) *Let $T = (Q, Z, \to)$ be a ILTS. A symbolic bisimulation over T is a symmetric relation $\mathcal{R} \subseteq Q \times Q$ s.t. $q_1 \mathcal{R} q_2$ whenever for all $q_1 \overset{P}{\to} q_1'$ there exists a finite set of transitions $(q_2 \overset{P_i}{\to} q_2^i)_{i \in I}$ with*

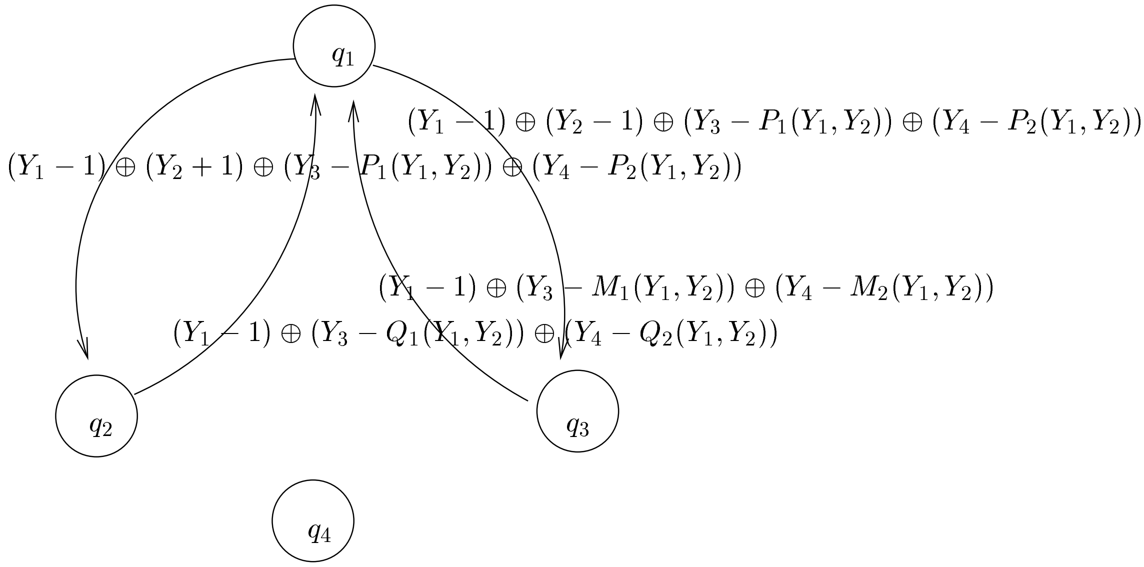

Figure 1: An example of ILTS

- $(1 - P^2) * \Pi_i P_i = 0$ *(which means* $Sol(P) \subseteq \bigcup_i Sol(P_i)$*), and*
- $q_1' \mathcal{R} q_2^i$*, for all* $i \in I$

There exists a greatest symbolic bisimulation over T*, written* $\sim$ *in the following. Notice that* $\sim$ *is an equivalence relation over the set of states* Q*.*

Proposition 3 (Compositionality) *Symbolic bisimulation is a congruence w.r.t. parallel composition* $|$ *and events hiding* $\backslash \bar{Z}$*.*

Proof We only prove the result for $|$, the remaining case is easier since symbolic bisimilar states in some ITLS T remain bisimilar in $(T \setminus \bar{Z})$.

Let $T_1 = (Q_1, Z, \rightarrow_1, I_1)$ and $T_2 = (Q_2, Z, \rightarrow_2, I_2)$ be two ILTS.
Define $\mathcal{R} \stackrel{\text{def}}{=} \{((q_1, q_2), (q_3, q_4)) \in co \times co | (q_1 \sim q_3) \wedge (q_2 \sim q_4)\}$.

Now, let $(q_1, q_2) \stackrel{P_1(Z) \oplus P_2(U)}{\rightarrow} (q_1', q_2')$. Because $q_1 \sim q_3$, and $q_2 \sim q_4$, then there exist a finite set of transitions $(q_3 \stackrel{P_{1,i}(Z)}{\rightarrow} q_3^i)_{i \in I}$ with $q_1 \mathcal{R} q_3^i$ and

$$(1 - P_1^2(Z)) * \Pi_i P_{1,i}(Z) = 0, \tag{3}$$

and a finite set of transitions $(q_4 \stackrel{P_1^j(U)}{\rightarrow} q_4^j)_{j \in J}$ with $q_2 \mathcal{R} q_4^j$ and

$$(1 - P_2^2(U)) * \Pi_j P_{2,j}(U) = 0 \tag{4}$$

By (3), (4) and the distributivity of $\oplus$ over $*$, we have

$$\Pi_i P_{1,i}(Z) \oplus \Pi_j P_{2,j}(U) = \Pi_{i,j}(P_{1,i}(Z) \oplus P_{2,j}(U))$$

the couples (q_3^i, q_4^j) are then the good candidates to conclude the proof. See Figure 2.

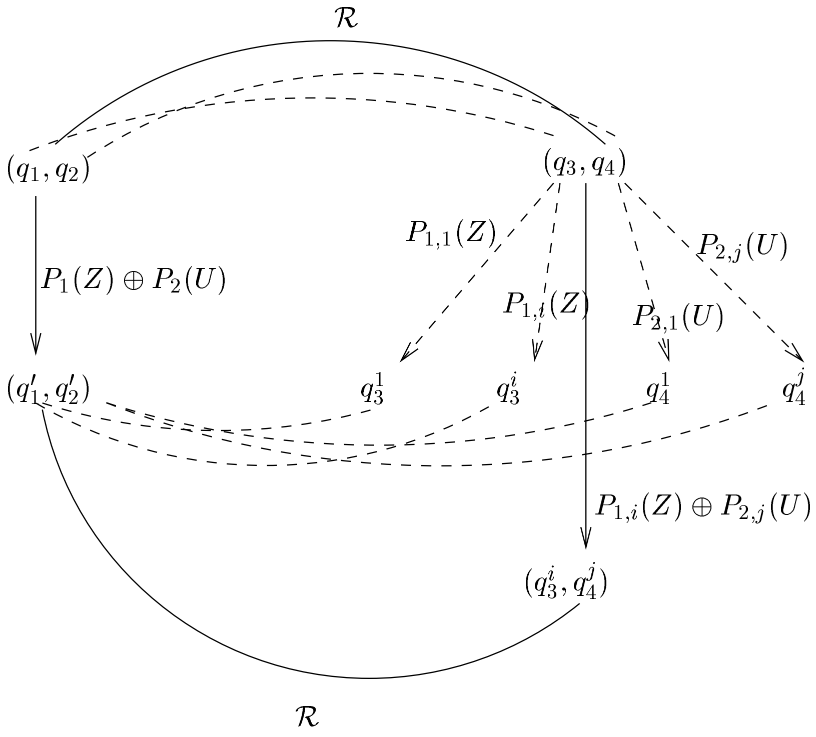

Figure 2: Congruence property for $|$

Transitions from (q_3, q_4) are dealt similarly. $\square$

3.5 Other characterizations of symbolic bisimulation

Symbolic bisimulation between ILTS coincides with strong bisimulation between the corresponding explicit labeled transition systems :

Proposition 4 *Let $T = (Q, Z, \rightarrow)$ be an ILTS. Then for all $q_1, q_2 \in Q$, $q_1 \sim q_2$ in T iff $q_1 \underline{\leftrightarrow}_{Exp(T)} q_2$.*

Proof $\Rightarrow$) We show that $\sim$ is a strong bisimulation in $Exp(T)$.

Let $q_1 \sim q_2$ and let $q_1 \xrightarrow{y} q_1'$ in $Exp(T)$. Then there exists $q_1 \xrightarrow{P} q_1'$ with $P(y) = 0$. By definition of $\sim$, there exists $q_2 \xrightarrow{P_i} q_2^{(i)}$ s.t. $(1 - P^2) * \Pi_i P_i = 0$ and $q_1' \sim q_2^{(i)}$. Because $P(y) = 0$, then $(1 - P^2) * \Pi_i P_i = 0$ applies to y entails $P_i(y) = 0$ for some i and then $q_2 \xrightarrow{y} q_2^{(i)}$, and we are done.

$\Leftarrow$) We show that $\underline{\leftrightarrow}$ is a symbolic bisimulation.

Let $q_1 \underline{\leftrightarrow} q_2$ and let $q_1 \xrightarrow{P} q_1'$ in T. Because $q_1 \underline{\leftrightarrow} q_2$, for each $y_0 \in Sol(P)$ there exists $q_2 \xrightarrow{y_0} q_2^{y_0}$ with $q_1' \underline{\leftrightarrow} q_2^{y_0}$. In T there exists some P^{y_0} s.t. $q_2 \xrightarrow{P^{y_0}} q_2^{y_0}$ and $P^{y_0}(y_0) = 0$.
Consider now the polynomial $\Pi_{y \in Sol(P)} P^y(Y)$. 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. The q_2^y are the good candidates. $\square$

Notice that one could get another proof of Proposition 3 for $|$ construct by Proposition 4 and Remark 1.

It can also be established that our notion symbolic bisimulation coincides with the one of DeSimone, also called symbolic bisimulation, provided an encoding of the ILTS into a reactive automata, but this is out of the scope of this report.

4 Polynomial Dynamical Systems

4.1 Definitions

Definition 4.1 (Polynomial Dynamical Systems) *A* polynomial dynamical system (PDS for short) *(over the finite Galois field $\mathbb{Z}/_{3\mathbb{Z}}$) is a system of polynomial equations of the form*

$$S =< E^S(X, Y, X') > \tag{5}$$

where X, Y, X' are sets of indexed variables $\{X_1, X_2, ..., X_n\}$, $\{Y_1, Y_2, ..., Y_m\}$ and $\{X_1', X_2', ..., X_n'\}$ respectively.

$E^S(X, Y, X')$ is called the evolution polynomial*. We write $statedim(S) \stackrel{\text{def}}{=} n$ and $eventdim(S) \stackrel{\text{def}}{=} m$, and denote by $dim(S)$ the couple $(statedim(S), eventdim(S))$. We write $S(n, m)$ to denote a PDS S with $dim(S) = (n, m)$.*

We omit the superscript S when it is clear from the context.

PDS are naturally provided with an operational semantics yielding an ILTS :

Definition 4.2 (Operational semantics of PDS) *Let $S(n, m) =< E(X, Y, X') >$ be a PDS over $\mathbb{Z}/_{3\mathbb{Z}}$. We define $TS(S) \stackrel{def}{=} (\mathbb{Z}/_{3\mathbb{Z}}{}^n, Y, \rightarrow)$ (we use $x, x', (x_1, ..., x_n), x^1, x^2, ...$ for typical states of $TS(S)$, i.e. elements of $\mathbb{Z}/_{3\mathbb{Z}}{}^n$) where $x \xrightarrow{P} x'$ whenever $P(Y) = E(x, Y, x')$.*

Notice that any $TS(S)$ is finite with $3^{statedim(S)}$ states (not all those states are necessarily reachable).

Example 2 *Consider S in the form*
$< [X_1' - (\epsilon_1(X_1, X_2, Y_1, Y_2) + (1 - Y_1^2) * X_1)] \oplus [X_2' - (\epsilon_2(X_1, X_2, Y_1, Y_2) + (1 - Y_1^2) * X_2)] \oplus [Y_2^2 - Y_1^2] \oplus [Y_3 - \epsilon_1(X_1, X_2, Y_1, Y_2)] \oplus [Y_4 - \epsilon_2(X_1, X_2, Y_1, Y_2)] >$,
where $\epsilon_1(X_1, X_2, Y_1, Y_2)$ and $\epsilon_2(X_1, X_2, Y_1, Y_2)$ are defined as follows:

$$
\begin{aligned}
\epsilon_1(X_1,X_2,Y_1,Y_2) &= [(X_1+1)\oplus(X_2+1)\oplus P_1(Y_1,Y_2)]\\
&* \ [(X_1+1)\oplus(X_2-1)\oplus Q_1(Y_1,Y_2)]\\
&* \ [(X_1-1)\oplus(X_2+1)\oplus M_1(Y_1,Y_2)]\\
\epsilon_2(X_1,X_2,Y_1,Y_2) &= [(X_1+1)\oplus(X_2+1)\oplus P_2(Y_1,Y_2)]\\
&* \ [(X_1+1)\oplus(X_2-1)\oplus Q_2(Y_1,Y_2)]\\
&* \ [(X_1-1)\oplus(X_2+1)\oplus M_2(Y_1,Y_2)]
\end{aligned}
$$

The reader can draw the corresponding ILTS and find out it is the one of Example 1.

4.2 Verification techniques - Model checking

The algebraic geometry of polynomials makes it possible to perform implicit model-checking techniques over the underlying explicit transition system. All these techniques are fully described in [Dut92]. The techniques are already implemented in a tool called SIGALI developed in the EP-ATR group, see [DB97] for more details. The main feature of the tool is that it handles polynomials instead of sets of states, i.e. varieties (which are dual by Section 2). The most important feature is the implementation [DB97] of polynomials by Ternary Decision Diagram (TDD) in the same spirit of BDD, but where the paths in the data structures are labeled with values $\{-1,0,1\}$ instead of only $\{0,1\}$ as for BDD's.

The properties that are checkable are any property expressible in the branching-time temporal logic μ-calculus [Koz83]. Henceforth, all classical fixed-point based properties can be iteratively computed such as reachability, attractivity ,...

In this article, we mainly focus on the notion of *greatest invariant* since it will be used in the next section for the computation of the symbolic bisimulation : take a labeled transition system with set of states Q, and $V \subseteq Q$. We aim to compute the set of states $\mathcal{GI}(V)$ called *the greatest invariant contained in* V defined by $\{x \in Q | x \in V \wedge \forall x \xrightarrow{y} x', x' \in V\}$. This can be done implicitly by the tool SIGALI in the PDS $S(n,m) =< E(X,Y,X') >$, provided we have a generator g representing V, i.e. $Sol(g) = V$. SIGALI computes a canonical generator $\hat{g}$ of $\mathcal{GI}(V)$.
We do not describe here the algorithm to $\hat{g}$; the reader can for example look at [Dut92].

4.3 Computing the symbolic bisimulation

Thanks to Proposition 4, an algorithm for symbolic bisimulation can be obtained from an algorithm for strong bisimulation.

As the dual notion of Definition 2.2 (for existential abstraction), we introduce here a *universal abstraction* over polynomials :

Definition 4.3 (Universal abstraction) *Given a set of variables* $Z = \{Z_1, ..., Z_k\}$, *a polynomial* $P \in \mathbb{Z}/_{3\mathbb{Z}}[Z]$, *and some* $Z_l \in Z$, *we define* $\forall Z_l P \in \mathbb{Z}/_{3\mathbb{Z}}[Z \setminus \{Z_l\}]$ *to be s.t.*

$$Sol(\forall Z_l P) = \{(z_1, ..., z_{l-1}, z_{l+1}, ..., z_k) | \forall z_l \in \mathbb{Z}/_{3\mathbb{Z}}\, , P(z_1, ..., z_k) = 0\}.$$

It is easy to generalize the construct to any subset $\bar{Z}$ of Z, to obtain $\forall \bar{Z} P$. Again we refer to [Dut92] for the detailed definitions.

Definition 4.4 (Admissible events) *Given $S(n,m) =< E(X,Y,X') >$ a PDS, define $Adm(X,Y) \stackrel{def}{=} \exists X' E(X,Y,X')$. Then, for $x \in \mathbb{Z}/_{3\mathbb{Z}}{}^n$, $Adm(x,Y) = \{y \in \mathbb{Z}/_{3\mathbb{Z}}{}^m | x \xrightarrow{y}\}$ is the set of all admissible events from state x.*

Algorithm for strong bisimulation

Let $T = (Q = \mathbb{Z}/_{3\mathbb{Z}}{}^n, Y, \rightarrow)$ be the ILTS associated to S (i.e. $TS(S)$). In order to characterize the (greatest) strong bisimulation of T, we use SIGALI to execute the following stages :

1. Define new sets of variables $X^1 = \{X_1^1, ..., X_n^1\}$ and $X^2 = \{X_1^2, ..., X_n^2\}$.
2. Compute $R_0(X^1, X^2) \in \mathbb{Z}/_{3\mathbb{Z}}[X^1, X^2]$, that is a generator of the (symmetric) binary relation $\mathcal{R}_0 \subseteq Q \times Q$, s.t. $(x^1, x^2) \in \mathcal{R}_0$ whenever $x^1 \xrightarrow{y}$ iff $x^2 \xrightarrow{y}$.

   This can be achieved by computing with SIGALI a generator of $LC(\forall Y Adm(X^1, Y) - Adm(X^2, Y))$.
3. Define the composed system $S \times S$ of dimension $(n+n, m)$ by $S \times S \stackrel{\text{def}}{=} < E(X^1, Y, X'^1) \oplus E(X^2, Y, X'^2) >$.

   Notice that $\mathcal{R}_0$ is a predicate of $S \times S$, and that $(x^1, x^2) \xrightarrow{y} (x'^1, x'^2)$ in $S \times S$ whenever $x^i \xrightarrow{y} x'^i$ $(i = 1, 2)$ in S.
4. Compute the canonical generator of the greatest invariant in $S \times S$ contained in $\mathcal{R}_0$. In other words, compute $\hat{R}_0$.

Proposition 5 *For all $x^1, x^2 \in \mathbb{Z}/_{3\mathbb{Z}}{}^n$, $\hat{R}_0(x^1, x^2) = 0$ iff $x^1 \sim x^2$ in T.*

Proof By Proposition 4, it is enough to we show that $\hat{R}_0(x^1, x^2) = 0$ iff $x^1 \underline{\leftrightarrow}_{Exp(T)} x^2$. We leave it to the reader. □

5 Integration environment

5.1 SIGNAL language

This section briefly presents an outline of the data-flow language SIGNAL.

The *kernel*-language SIGNAL is based on four operations, defining primitive processes or equations, and a compositional operation to build more elaborate processes in the form of equations systems.

- **Static synchronous generator** `p(A1, ..., An)` is a transformation of data at an instant t. This instruction requires that all referred variables must be present simultaneously.
- **Deterministic merge generator**, written `A3 := A1 default A2`, has the value of `A1` when it is present, or otherwise that of `A2` if it is present and `A1` not. Its clock is the union (i.e. *includes* or *contains*) of those of `A1` and `A2`.

- **Selection generator** of the form `A2` := `A1 when B` relies an event on a port `A1` with the boolean `B` which must be equal to *true*; otherwise, `A2` is not emitted. The result can be seen as a down-sampling of an event on `A1`. The clock of `A2` is the intersection of (i.e. *included* in) that of `A1` and that of `b` at the value *true*.

- **Delay** (a dynamic synchronous generator) `A2` := `A1 $1 init A0` gives access to the last value of signal `A1`. `A1` and `A2` have equal clocks.

- **Parallel composition** of processes is noted | and consists in the composition of the equations (systems); it is associative and commutative ; communication between them is carried by the broadcasting of signals.

```
process Prog = (?  logical A,B ! E1,E2 )
(| lE1 := E1 $1
| lE2 := E2 $1
| E1 := false when (not(lE1) and not(lE2) and A and not(B))
default true when (not(lE1) and not(lE2) and A and B)
default false when (not(lE1) and lE2 and A )
default false when (lE1 and not(lE2) and A)
default lE1
¿ | E2 := true when (not(lE1) and not(lE2) and A and not(B))
default false when (not(lE1) and not(lE2) and A and B)
default false when (not(lE1) and lE2 and A)
default false when (lE1 and not(lE2) and A)
default lE2
|) / E1,E2,lE1,lE2
where
logical lE1 init false, lE2 init false
end
```

Figure 3: A toy SIGNAL program

Figure 3 contains an example of a SIGNAL program with (visible) signals $A, B, E1, E2$ and with two delayed signals $lE1$ and $lE2$ which valuations are the previous ones of $E1$ and $E2$ respectively. These delayed variables will give rise to two states variables, X_1 and X_2 in the derived PDS, see next section. However signals $lE1$ and $lE2$ have been hidden, hence are invisible externally.

5.2 Associated Polynomial Dynamical System

The SIGNAL programs are translated into PDS over $\mathbb{Z}/_{3\mathbb{Z}}$; this is automatically done by the SIGNAL compiler. The principle is to code the three possible values of a boolean signal A (i.e., *present* and *true*, or *present* and *false*, or *absent*) according to :

$$\begin{cases} present\&true & \mapsto & +1 \\ present\&false & \mapsto & -1 \\ absent & \mapsto & 0 \end{cases}$$

We still write A for the abstracted signal in $\mathbb{Z}/_{3\mathbb{Z}}$.

For the non-boolean signals, we only code the fact that the signal is *present* or *absent*. Hence, for any signal variable A, the value of A^2 is true if A is present and false if A is absent. A^2 is called the *clock* of A.
Thus, two synchronous signals A and B must satisfy the constraint $A^2 = B^2$ (an example is given below).
Now, each equation of a SIGNAL program can be encoded in a polynomial equation between their abstractions in $\mathbb{Z}/_{3\mathbb{Z}}$. For example, `C := A when B` can be rewritten into `C = A(-B-B`^{2}`)`.
The delay \$ operator of SIGNAL is different because it requires memorizing the past value of the signal into a state variable, say X. In order to encode `lE := E$1`, we have to introduce the following equations:

$$\begin{cases} X' &= E + (1 - E^2) * X \\ lE &= X * E^2 \end{cases}$$

where X' denotes the next value of state variable X.

We refer to [DB94] for more details about the translation of SIGNAL programs into PDS.

This way, the program `Prog` of Figure 3 can be translated into a PDS. Signals $lE1, lE2$ have been hidden on purpose as they characterize at each step the source state of the transition as well as the signals $E1$ and $E2$ which characterize the target state.

When running the compilation for program `Prog` in order to obtain the corresponding PDS, many transformations are performed, which simplify the system over $\mathbb{Z}/_{3\mathbb{Z}}$. The evolution polynomial has the final form :

$$[X_1' - (\epsilon_1(X_1, X_2, A, B) + (1 - A^2) * X_1)] \oplus [X_2' - (\epsilon_2(X_1, X_2, A, B) + (1 - A^2) * X_2)] \oplus [B^2 - A^2] \quad (6)$$

where ϵ_1 and ϵ_2 are polynomials defined by (we give the SIGNAL-like operators, but they can be translated into operations in $\mathbb{Z}/_{3\mathbb{Z}}$) :

$$\begin{aligned} \epsilon_1(X_1, X_2, A, B) = \quad & not\ (when\ not\ (X_1 * A^2)\ and\ not\ (X_2 * A^2)\ and\ A\ and\ not\ B) \\ & default\ when\ not\ (X_1 * A^2)\ and\ not\ (X_2 * A^2)\ and\ A\ and\ B \\ & default\ not(when\ not\ (X_1 * A^2)\ and\ (X_2 * A^2)\ and\ A) \\ & default\ not\ (when\ (X_1 * A^2)\ and\ not\ (X_2 * A^2)\ and\ A) \\ & default\ (X_1 * A^2) \end{aligned}$$

$$\begin{aligned} \epsilon_2(X_1, X_2, A, B) = \quad & when\ not\ (X_1 * A^2)\ and\ not\ (X_2 * A^2)\ and\ A\ and\ not\ B \\ & default\ not(when\ not\ (X_1 * A^2)\ and\ not\ (X_2 * A^2)\ and\ A\ and\ B) \\ & default\ not\ (when\ not\ (X_1 * A^2)\ and\ (X_2 * A^2)\ and\ A) \\ & default\ not(when\ (X_1 * A^2)\ and\ not\ (X_2 * A^2)\ and\ A) \\ & default\ (X_2 * A^2) \end{aligned}$$

The first (resp. second) polynomial of Expression (6) above gives the evolution law for the first (resp. second) component of the current state; the third polynomial concerns synchronization constraints between signals A and B.

Also, we obtain some initialization conditions which tell us from which state the system starts (in the example, we get $X_1 = -1 \wedge X_2 = -1$), but this is not our purpose here.

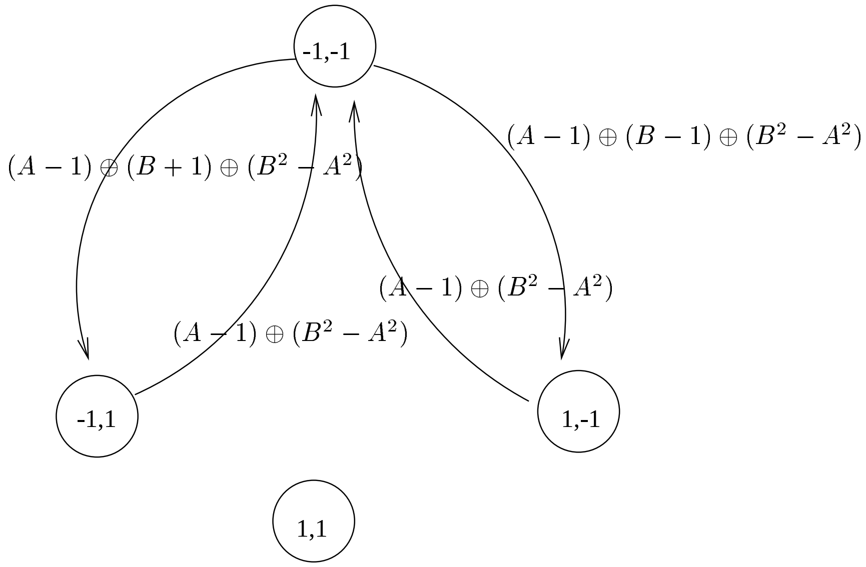

Figure 4: The ILTS associated to the PDS

Figure 4 shows the implicit labeled transition system associated to the PDS above.

When running SIGALI, relation $\hat{R}_0$ of the algorithm evaluates to zero. Therefore, all states of the system are equivalent and the quotient system contains only one state.
The possible transitions from this state to itself are obtained by computing the "union" of the polynomials labeling all outgoing transitions from any chosen state of the original ILTS. This product evaluates to $(A-1) \oplus (B^2 - A^2)$. Figure 5 shows the final reduced system.

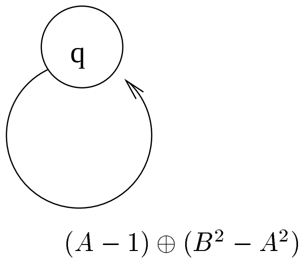

Figure 5: The reduced ILTS

References

[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.

[CLO93] D. Cox, J. Little, and D. O'Shea. *Ideals, varieties and algorithms: an introduction to computational algebraic geometry and commutative algebra.* Springer-Verlag, 1993.

[DB94] 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.

[DB97] 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.

[GB87] P. Le Guernic and A. Benveniste. Real-time, synchronous, data-flow programming: the language SIGNAL and its mathematical semantics. Research Report 620, INRIA, February 1987.

[Koz83] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27(3):333–354, 1983.

[GBGM91] P. Le Guernic, M. Le Borgne, T. Gautier, and C. Le Maire. Programming real time applications with SIGNAL. Technical Report 1446, Institut National de Recherche en Informatique et en Automatique (INRIA), Le Chesnay Cedex, France, June 1991.

[Mil89] R. Milner. A complete axiomatisation for observational congruence of finite-state behaviours. *Information and Computation*, 81(2):227–247, 1989.

[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.