Labeling Automata with Polynomials *

Olga Kouchnarenko and Sophie Pinchinat †

Abstract

We present a behavioral model for discrete event systems based on a polynomial approach. We define *implicit transition systems* with associated combinators of parallel composition and event hiding, as well as an equivalence criterion called *symbolic bisimulation*. We also establish congruence properties of this equivalence. Further on, we explain the methodology applied to the synchronous language SIGNAL, which domains of application range over the many fields of embedded systems, robotics, etc.

Keywords: Dynamic systems, event calculus, polynomials, behavioral equivalence, symbolic bisimulation, synchronous languages, model-checking.

1 Introduction

Dynamic systems are systems that evolve according to their environment. In general, an evolution of the system, in a given state, depends on an input event (some information given by the environment); this evolution leads to some instantaneous output event and to state changes.

Synchronous languages have been designed to ease the programmer's task when dealing with such systems; they provide some primitives for concurrency and communication. They can be of different kinds. The most popular ones that have been designed in France are: ESTEREL [BC84] an imperative language, LUSTRE [Pla88] and SIGNAL [BLJ91] based on declarative approach. These languages naturally bear a semantics in terms of discrete event systems, and their control part concerns boolean valued signals. The synchronous features allow one to express synchronization constraints between the different (output and internal) events of the system and the input events of its environment. Hence, any operational based semantics of such systems leads to automata labeled by combinations of atomic events. Such a labels describes an acceptable solution of the system constraints. It will be called a *composed event*, or simply an event. For SIGNAL language, a event is a vector of atomic events which respect the constraints given (by the equations of) the program.

In order to model the behavior, instead of considering explicitly all possible events for a given state change, we develop a formalism where actions of the automata are polynomials. These automata will be called *implicit transition systems* (or *ILTS* for short). These polynomials are based on several variables (one for each atomic event) with coefficients in $\mathbf{Z}_3$, according to the following encoding: an atomic boolean event can either be *absent*, then encoded by 0, or *present*

*Work supported by the Esprit SYRF Project 22703.

†EP-ATR Group, IRISA, F-35042, Rennes, France, e-mail: {okouchna, pinchina}@irisa.fr

and equal to *true*, encoded by 1, or *present* and equal to *false*, encoded by -1. The solutions of a polynomial are composed events. ILTS naturally possess an interpretation in terms of classical labeled transition systems, but they permit to avoid the transitions enumeration one would get by describing explicitly each event labeled transition. Moreover, the algebraic theory of polynomials can be used to define a language of actions which offers simple definitions for *parallel composition* and *event hiding*, both combinators widely used to design complex systems.

Further on, we propose a behavioral equivalence, called *symbolic bisimulation* over ILTS with good properties; it has the congruence property w.r.t. the combinators and it is tightly related to well-known strong bisimulation (see Theorems 1 and 2).

The developed theory is currently applied to the language SIGNAL: the options of the compiler plugged with the basic functions of the verification tool SIGALI [DLB97] allow us to perform polynomial handling for symbolic bisimulation computation.

The paper is organized as follows: Section 2 introduces the action language based on algebraic geometry of polynomials; Section 3 presents the models and the associated theory; finally, in Section 4, the use of the developed theory is illustrated by the concrete application to the language SIGNAL using the tool SIGALI. Section 5 is devoted to the conclusion and some perspectives.

2 Basic notions of algebraic geometry

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 $x \in \mathbf{Z}_3$[1]. Let $\bar{Z}$ be a finite set of m distinct variables $Z_1, ..., Z_m$. We denote by $\mathbf{Z}_3[\bar{Z}]$ (or $\mathbf{Z}_3[Z_1, ..., Z_m]$) the set of polynomials over variables $Z_1, ..., Z_m$ which coefficients range over $\mathbf{Z}_3$ with typical elements $P(\bar{Z})$ (or P for short), $P_1(\bar{Z}), \ldots$.
We recall that $(\mathbf{Z}_3[\bar{Z}], +, *)$ is a ring. In the following, a *(an algebraic) variety of* $\mathbf{Z}_3^m$ is any subset V of $\mathbf{Z}_3^m$.

Varieties and ideals. Given a polynomial $P(\bar{Z}) \in \mathbf{Z}_3[\bar{Z}]$, we associate the subset $Sol(P) \subseteq \mathbf{Z}_3^m$, called its *algebraic variety*, defined by : $Sol(P) \stackrel{\text{def}}{=} \{(z_1, ..., z_m) \in \mathbf{Z}_3^m | P(z_1, ..., z_m) = 0\}$ the set all the solutions of polynomial P, and we generalize it to sets of polynomials by : for all $G \subseteq \mathbf{Z}_3[\bar{Z}]$, $Sol(G) \stackrel{\text{def}}{=} \bigcap_{P \in G} Sol(P)$.

A set $I \subseteq \mathbf{Z}_3[\bar{Z}]$ is an *ideal* if it is closed by linear combinations of its elements, i.e. $I = \{A_1 * P_1 + A_2 * P_2 + ... + A_t * P_t | (P_1, ..., P_t \in I) \wedge (A_1, ..., A_t \in \mathbf{Z}_3[\bar{Z}])\}$. Let I_1 and I_2 be two ideals, we write $I_1 + I_2$ the ideal generated by $\{P_1 + P_2 | P_1 \in I_1 \wedge P_2 \in I_2\}$.
For any $G \subseteq \mathbf{Z}_3[\bar{Z}]$, we write $< G >$ for the ideal generated by G, i.e. the smallest ideal which contains G. Because $Sol(G) = Sol(< G >)$ for all set G of polynomials, and without loss of generality, we can consider ideals of polynomials instead of simply sets of polynomials. For all variety $V \subseteq \mathbf{Z}_3^m$, the set $I(V)$ will denote the ideal $I(V) \stackrel{\text{def}}{=} \{P \in \mathbf{Z}_3[\bar{Z}] | \forall\ \bar{z} \in V,\ P(\bar{z}) = 0\}$.

[1] The present paper can be generalized to any field of the form $\mathbf{Z}_p$.

Henceforth, there is a natural correspondence between ideals and algebraic varieties:

$$\text{for any variety } V, \quad Sol(I(V)) = V, \tag{1}$$

$$\text{and, for all ideal } I, \quad I(Sol(I)) = I + < \{Z_1^3 - Z_1, ..., Z_m^3 - Z_m\} > . \tag{2}$$

The second equation shows that the correspondence between ideals and varieties can be made unique modulo the ideal generated by $\{Z_1^3 - Z_1, ..., Z_m^3 - Z_m\}$. Therefore in the following, we consider the quotient ring $\mathbf{Z}_3[\bar{Z}]/ < \{Z_1^3 - Z_1, ..., Z_m^3 - Z_m\} >$ instead of simply $\mathbf{Z}_3[\bar{Z}]$, but we still write $\mathbf{Z}_3[\bar{Z}]$. Now the equation (2) becomes $I(Sol(I)) = I$. Also, each variety $V \subseteq \mathbf{Z}_3^m$ can be represented by a unique ideal $I(V)$ and the following holds: for all varieties V_1 and V_2,

$$I(V_1 \cap V_2) = I(V_1) + I(V_2), \quad I(V_1 \cup V_2) = I(V_1) \cap I(V_2), \text{ and } V_1 \subseteq V_2 \Leftrightarrow I(V_2) \subseteq I(V_1). \tag{3}$$

Generators. As proved in [LBBLG91], it is possible to represent any ideal by a unique polynomial called *the canonical generator*.

Definition 2.1 (Generator, canonical generator) *Let I be an ideal of $\mathbf{Z}_3[\bar{Z}]$. $g \in \mathbf{Z}_3[\bar{Z}]$ is called a* generator *of I whenever $g(z) = 0$ iff $z \in Sol(I)$, which means $< g >= I$, and it is* canonical *whenever $g^2 = g$.*

For all ideal $I \subseteq \mathbf{Z}_3[\bar{Z}]$ generated by $P_1, \ldots, P_t$, the canonical generator can be obtained by:

$$g = \bigoplus_{i=1}^{t} P_i \quad \text{where } \oplus \text{ is defined by } f \oplus h \stackrel{\text{def}}{=} (f^2 + h^2)^2 \text{ (it is associative).} \tag{4}$$

Set-theoretical constructs used in Equations (1)–(3) can be expressed in the dual framework of polynomials. Properties given below allow us to implement sets by canonical generators.
Let g_1 and g_2 be two generators (not necessarily canonical) of some ideals I_1 and I_2 respectively, and let $V_1 = Sol(I_1)$ and $V_2 = Sol(I_2)$,

$$1 - g_1^2 \text{ is the canonical generator of } (\mathbf{Z}_3^n \setminus V_1), \tag{5}$$

$$g_1 \oplus g_2 \text{ is the canonical generator } V_1 \cap V_2, \tag{6}$$

$$g_1^2 * g_2^2 \text{ is the canonical generator } V_1 \cup V_2, \tag{7}$$

$$V_1 \subseteq V_2 \text{ iff } (1 - g_1^2) * g_2 \text{ evaluates to zero.} \tag{8}$$

Finally, we introduce the *existential abstractions* (or existential quantifications) over polynomials w.r.t. some variables. This construct will be useful for the definition of *events hiding* presented in Section 3.

Definition 2.2 (Existential abstraction) *For any $V \subseteq \mathbf{Z}_3^m$ and a sub-order of indexes $\iota = \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, m\}$, we define $V \setminus \iota$ as the set of all elements $(z_{j_1}, \ldots)$ with indexes $j_k \in \{1, \ldots, m\} \setminus \iota$ obtained by deleting from the elements $\{z_1, ..., z_m\}$ of V all components indexed in ι. It is a projection.*

For instance, $\{(1, z_2, z_3, z_4) | z_i \in \mathbf{Z}_3\} \setminus \{3, 4\} = \{(1, z_2) | \exists z_3, z_4 \in \mathbf{Z}_3, (1, z_2, z_3, z_4) \in V\}$.
Existential abstraction can be dually defined over polynomials as follows: for any $\tilde{Z} \subseteq \bar{Z}$, we can define a mapping $\exists \tilde{Z} : \mathbf{Z}_3[\bar{Z}] \to \mathbf{Z}_3[\bar{Z} \setminus \tilde{Z}]$ s.t. $Sol(\exists \tilde{Z} P) = Sol(P) \setminus \{i | Z_i \in \tilde{Z}\}$. Actually, thanks to Equation 4, it is enough to describe the computation of $\exists \tilde{Z}$ for any generator g: if $\tilde{Z} = \{Z_i\}$ for some i then $\exists Z_i g$ can be computed by $\exists Z_i g = g_{|z_i=0} * g_{|z_i=1} * g_{|z_i=-1}$. When $\tilde{Z}$ contains more than one variable, $\exists Z_i g$ can be iteratively computed using Equation above.

3 Polynomial encoding of actions

This section introduces *implicit labeled transition systems*, *parallel composition* and *event hiding*, as well as the *symbolic bisimulation* behavioral equivalence.

Definition 3.1 (Implicit Labeled Transition Systems (ILTS)) *An* m-dimensional implicit labeled transition system (or m-ILTS) *is a structure* $T = (Q, \bar{Z}, \rightarrow)$, *where*
- Q *is set of states,*
- $\bar{Z}$ *is a set of* m *variables* $Z_1, ..., Z_m$, *and*
- $\rightarrow \subseteq Q \times \mathbf{Z}_3[\bar{Z}] \times Q$. *Each transition is labeled by a polynomial over the set of variables* $\bar{Z}$.

We write $q \overset{P(\bar{Z})}{\rightarrow} q'$ (or simply $q \overset{P}{\rightarrow} q'$), instead of $(q, P(\bar{Z}), q') \in \rightarrow$.

In the following, we assume the reader is familiar with labeled transition systems (LTS) as well as with the notion of strong bisimulation as a good commonly accepted notion of behavioral equivalence over LTS. Strong bisimulation is widely used in the verification field of reactive systems. (However, we refer to [Par81, Mil89] for the definition).

ILTS can be understood as a "compact" representation of classical labeled transition systems for which labels are elements of $\mathbf{Z}_3^m$: each arrow of the ILTS of the form $q \overset{P(\bar{Z})}{\rightarrow} q'$ implicitly represents all the transitions $q \overset{\bar{z}}{\rightarrow} q'$ where $\bar{z} \in Sol(P(\bar{Z}))$[2]. It gives then a way to represent a set of events V by the canonical generator of $I(V)$; this concrete representation of sets allows us to easily define the parallel composition and the hiding of events, whereas using very sets would drastically complicate these description.

The *parallel composition* and the *events hiding* are indispensable constructs to design compositionally complex systems from simple ones.

Definition 3.2 (Parallel composition of ILTS) *Let* $T_1 = (Q_1, \bar{Z}, \rightarrow_1)$ *be an* m_1*-ILTS and* $T_2 = (Q_2, \bar{U}, \rightarrow_2)$ *be an* m_2*-ILTS with possible common variables between* $\bar{Z}$ *and* $\bar{U}$. *The* parallel composition *of* T_1 *and* T_2, *written* $T_1 \mid T_2$, *is* $(Q_1 \times Q_2, \bar{Z} \cup \bar{U}, \rightarrow)$ *where* $(q_1, q_2) \overset{P_1(\bar{Z}) \oplus P_2(\bar{U})}{\rightarrow} (q_1', q_2')$ *in* $T_1 \mid T_2$ *whenever* $q_1 \overset{P_1(\bar{Z})}{\rightarrow}_1 q_1'$ *in* T_1 *and* $q_2 \overset{P_2(\bar{U})}{\rightarrow}_2 q_2'$ *in* T_2.

In Definition 3.2, expression $P_1(\bar{Z}) \oplus P_2(\bar{U})$ imposes synchronization as well as compatibility constraints between actions of T_1 and T_2. We can easily retrieve, from Definition 3.2, the synchronous parallel composition over the underlying explicit transition systems, but the representation is still remaining "compact", without transitions explosion.

Hiding events permits to abstract from internal actions or communications between the composed systems.

Definition 3.3 (Events hiding) *Let* $T = (Q, \bar{Z}, \rightarrow)$ *be an* m*-ILTS, and let* $\tilde{Z} \subseteq \bar{Z}$. *We define the* events hiding (w.r.t. $\tilde{Z}$) *by* $(T \setminus \tilde{Z}) \overset{def}{=} (Q, \bar{Z} \setminus \tilde{Z}, \rightarrow_{\setminus \tilde{Z}})$ *where* $q_1 \overset{g}{\rightarrow}_{\setminus \tilde{Z}} q_2$ *iff* g *is the canonical generator of* $< \exists \tilde{Z} P >$ *and* $q_1 \overset{P}{\rightarrow} q_2$.

[2]We could have restricted the labels of the ILTS to canonical generators

We propose now an equivalence relation over ILTS which is strongly inspired from DeSimone's symbolic bisimulation over reactive automata [DR94].

Definition 3.4 (Symbolic Bisimulation) *Let $T = (Q, \bar{Z}, \rightarrow)$ be an 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 $q_1 \overset{P}{\rightarrow} q_1'$ there exists a finite set of transitions $(q_2 \overset{P_i}{\rightarrow} q_2^i)_{i \in I}$ with*
- $(1 - P^2) * \Pi_i P_i = 0$ *(which means $Sol(P) \subseteq \bigcup_{i \in I} Sol(P_i)$), 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.

We have proved the congruence theorem for the symbolic bisimulation and we have compared it with the well-known strong bisimulation relation.

Theorem 1 (Compositionality [KPH98]) *Symbolic bisimulation is a congruence w.r.t. parallel composition and events hiding.*

Theorem 2 (Expressiveness [KPH98]) *Symbolic bisimulation between ILTS coincides with strong bisimulation between the underlying explicit labeled transition systems.*

The two theorems above stress that Definition 3.4 is a relevant one for systems behavior. It can also be established that our notion of symbolic bisimulation coincides with the one of DeSimone, also called symbolic bisimulation, provided a non-trivial encoding of the ILTS (three-valued logics) into a reactive automata (two-valued logics), but this is out of the scope of this paper.

4 Applications

ILTS models are applied for the verification of systems described in the equational data-flow synchronous language SIGNAL[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 [RMC97], ...

We briefly explain how each SIGNAL program can be translated into an equational system of polynomials with coefficients in $\mathbf{Z}_3$, called *Polynomial Dynamical System* in[LBBLG91], which in turn can be given an ILTS models semantics.
The original multi-clock data-flow synchronous language SIGNAL manipulates a set of *signals*; each signal A denotes an unbounded sequence of typed values $(A_t)_{t \in \mathcal{T}}$, 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. $A_t \neq \perp$. Two signals with the same clocks 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 signals

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

hiding to internalize them.

• **Static synchronous operator** $\texttt{A} := \texttt{p}(\texttt{A}_1, \dots, \texttt{A}_\texttt{n})$ is a transformation of data at each instant t. This instruction requires that all referred variables to have the same clock.
• **Deterministic merge operator**, written `A := A1 default A2`, has the value of `A1` when it is present, or otherwise it has the value of `A2` if it is present and `A1` not. Its clock is the union of those of `A1` and `A2`.
• **Selection operator** of the form `A := A1 when B` links of a signal `A1` with the boolean `B` which must be equal to *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) `ZA := A $1` gives access to the last value of signal `A`. `A` and `ZA` have equal clocks.
• **Parallel composition** of processes is noted $|$ and consists in the conjunction of the equations (systems); it is then associative and commutative.
• **Event hiding** $\backslash\{\texttt{A}\}$ hides any occurrence of signal `A`; it is internalized.

As we already said, the SIGNAL programs are translated into equational systems over $\mathbf{Z}_3$. The principle is to code the three possible values of a boolean signal `A` by a signal variable a : `A` is *present* and *true*, `A` is *present* and *false*, `A` is *absent* are translated into $a = +1, a = -1, a = 0$ respectively. For the non-boolean signals, we only code the fact that the signal is *present* or *absent* by 1 and 0 respectively. Therefore, for any signal `A`, its *clock* is characterized by $a^2 \neq 0$, and two synchronous signals `A` and `B` satisfy $a^2 = b^2$.
Now, each SIGNAL operator can be encoded by polynomial equations in $\mathbf{Z}_3$. Because of the difference among boolean and non-boolean signals encodings, we give table in which the second column presents the encoding of operators over non-boolean signals, and the third one gives the boolean signal evaluation as well as their synchronization constraints.

Operators	Clock equations non-boolean signals	Evaluations boolean signals
$\texttt{A} := \texttt{not A}_1$		$a = -a_1$
$\texttt{A} := \texttt{A}_1 \texttt{ and } \texttt{A}_2$		$a = a_1a_2(a_1a_2 - a_1 - a_2 - 1)$ $a_1^2 = a_2^2$
$\texttt{A} := \texttt{A}_1 \texttt{ or } \texttt{A}_2$		$a = a_1a_2(1 - a_1a_2 - a_1 - a_2)$ $a_1^2 = a_2^2$
$\texttt{A} := \texttt{p}(\texttt{A}_1, \dots, \texttt{A}_\texttt{n})$	$a^2 = {a_1}^2 = \dots {a_n}^2$	
$\texttt{A} := \texttt{A}_1 \texttt{ default } \texttt{A}_2$	$a^2 = {a_1}^2 + (1 - {a_1}^2){a_2}^2$	$a = a_1 + (1 - {a_1}^2)a_2$
$\texttt{A} := \texttt{A}_1 \texttt{ when B}$	$a^2 = {a_1}^2(-b - b^2)$	$a = a_1(-b - b^2)$
$\texttt{ZA} := \texttt{A\$1}$	$za^2 = a^2$	$x' = a + (1 - a^2)x$ $za = a^2x$

Table 1: Encoding of SIGNAL operators

The delay \$ operator of SIGNAL requires to memorize the last value (different from $\perp$) of the signal into a new (state) variable, say x. In order to translate `ZA := A $1`, we have to introduce the two auxiliary equations in Table 1 where x' denotes the next value of state variable x. We refer to [LBBLG91] for more details about the translation of SIGNAL programs into equational systems.

The translation of Table 1 is automatically performed by the SIGNAL compiler. This way, any SIGNAL program can be translated into an equational system which can be seen as an automaton. This automata semantics can then be used as a basis for verification of SIGNAL programs. The techniques used in consist in manipulating the system of equations instead of the sets of solutions: each set of states is characterized by a polynomial which avoids the enumeration of the state space. The tool SIGALI [DLB97] offers algebraic polynomial computations. The most important feature is the implementation [DLB97] of polynomials by Ternary Decision Diagram (TDD) in the same spirit of BDD [Bry89], but where the paths in the data structures are labeled by values in $\{-1, 0, 1\}$ instead of $\{0, 1\}$. We have used the SIGALI tool to compute iteratively the symbolic bisimulation relation over ILTS. Because of lack of space, no example is proposed here; we refer to [KPH98].

5 Perspective and Related Approaches

In this paper, we have presented the *implicit transition systems* with parallel composition and event hiding combinators, as operational models in which actions are polynomials. Also, the symbolic bisimulation equivalence has been proved to be adequate: it is compositional and closely related to the strong bisimulation equivalence.

The main feature of the presented models is their capability to represent, in a compact way, discrete event systems. Our research group has already experienced those advantages in a French Electric Company (EDF) application for controller synthesis [LBMR96],

We are aware that the ILTS presented here are particular models that might be generalized: at some level of abstraction, labels of the ILTS are sets, since polynomials are particular implementations for sets (like polyhedra have been used for LUSTRE or boolean formulae for ESTEREL). Actually, expressing symbolic bisimulation of only by means of the abstract data type *set* might be enough, but the final implementation chosen for sets needs being known if one wishes to achieve optimized algorithms.

As we have shown, ILTS semantics applies to SIGNAL and an (under development) extension of the tool SIGALI will permit to compute the symbolic bisimulation and to reduce the systems. This application is of high interest since SIGNAL is used in a lot of areas (controller synthesis [LBMR96], robotics [RMC97],...) where the models equivalence-checking and reduction are unavoidable.

Also, we argue that ILTS are relevant in any application as long as events can be encoded in $\mathbf{Z}_3^m$ (or $\mathbf{Z}_p^m$ with p prime) with possibly other interpretations of values and where the system evolution is specified in a set-theoretical framework, such as equational systems, rewriting systems, logical clauses, lists of transitions, etc.

Finally, perspectives of work could be a proposal for rich modal logics to specify ILTS behavioral properties: for instance, "next world" modality can be decorated by a set of events, again represented by polynomials; model-checking algorithms could then be integrated in the SIGALI tool. Of course, logics expressiveness issues should be studied.

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.

[BC84] G. Berry and L. Cosserat. The ESTEREL synchronous programming language and its mathematical semantics. In *Seminar on Concurrency, Pittsburgh, LNCS 197*, pages 389–448. Springer-Verlag, July 1984.

[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.

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

[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.

[KPH98] O. Kouchnarenko, S. Pinchinat, and L. Helouet. Symbolic methods in the SIGNAL environment. Technical report, Institut National de Recherche en Informatique et en Automatique (INRIA), 1998. To appear.

[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.

[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.

[Pla88] J. A. Plaice. *Sémantique et Compilation de LUSTRE, un Langage Déclaratif Asynchrone*. Thèse de Doctorat, I.N.P. de Grenoble, France, May 1988.

[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.