Control of polynomial dynamic systems : an example

Bruno Dutertre, Michel Le Borgne

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE

Control of Polynomial Dynamic Systems: an Example

BRUNO DUTERTRE, MICHEL LE BORGNE

N° 2193

janvier 1994

PROGRAMME 2

Calcul symbolique,

programmation

et génie logiciel

ISSN 0249-6399

Control of Polynomial Dynamic Systems: an Example

Bruno Dutertre *, Michel Le Borgne **

Programme 2 — Calcul symbolique, programmation et génie logiciel
Projet EP-ATR

Rapport de recherche n° 2193 — janvier 1994 — 28 pages

Abstract: This paper presents the application of algebraic techniques to the control of a discrete event system. The system is described using Signal, a data-flow language. An equational model is then elaborated. Control objectives are expressed as invariance and reachability properties of the model. Control equations are synthesized using algebraic tools: ideals, varieties and principal generators.

Key-words: Discrete Event Systems, Control, Polynomial Methods

(Résumé : tsvp)

*Royal Holloway, University of London, Egham, Surrey TW20 0EX, United Kingdom
**IRISA , Campus de Beaulieu, 35042 Rennes Cedex, France

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 84 71 71

Contrôle de systèmes dynamiques polynomiaux: un exemple

Résumé : Ce rapport présente, à travers un exemple, l'application de techniques algébriques au contrôle de systèmes à évènements discrets. Le système est décrit à l'aide du langage SIGNAL , un langage flot de données conçu pour les applications temps réel et développé à l'IRISA. A partir de cette description, un modèle équationnel polynomial est obtenu. Des objectifs de contrôle sont alors définis en terme d'invariance et d'atteignabilité. Les équations de contrôle sont obtenues en utilisant des outils algébriques: idéaux, variétés et générateurs principaux.

Mots-clé : Systèmes à évènements discrets, Contrôle, Méthodes polynomiales

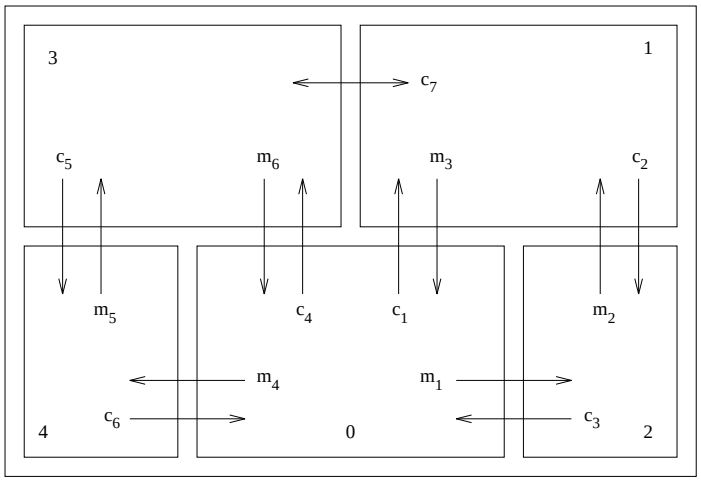

Figure 1: The Cat and Mouse problem

1 Introduction

The control theory of discrete event systems (DES) introduced by Ramadge and Wonham is based on formal languages and automata [11]. DES are modelled by languages over a finite alphabet; each element of the alphabet representing a possible event. The basic control problem is to supervise a DES by inhibiting certain events in order to satisfy a control objective given by another language [10].

This paper presents an algebraic approach to similar control problems. DES are modelled using SIGNAL, a data-flow language; from this description is derived an equational representation called a polynomial dynamic system. Controllers are then synthesized using algebraic techniques. The method is applied to the 'cat and mouse' example initially introduced in [12].

The paper is organized as follows. Section 2 briefly describes the example; section 3 presents SIGNAL and its use to model DES; section 4 introduces polynomial dynamic systems and algebraic tools; the resolution of the control problem is presented in section 5.

2 The cat and mouse example

A cat and a mouse are placed in a maze shown in figure 1. The animals can move through doors represented by arrows in the figure. The doors $C_1, \ldots, C_7$ are exclusively for the cat and the doors $M_1, \ldots, M_6$ are exclusively for the mouse. Each doorway can be traversed in only one direction, with the exception of C_7. A sensor associated with each door signals the passages and a control mechanism allows each door, except C_7, to be opened or closed.

Initially, the cat is in room 2 and the mouse in room 4. The problem is to control the doors such that the two following objectives are guaranteed:

- The cat and the mouse never occupy the same room simultaneously.
- It is always possible for the animals to return to the initial position.

The control must also permit as much freedom of movement as possible.

3 SIGNAL

SIGNAL is a synchronous data-flow language dedicated to real-time applications and reactive systems. It is based on an equational approach: programs are sets of constraints relating different sequences of values. An extensive presentation of SIGNAL can be found in [9]. A development environment including a graphical editor, a compiler and a verification tool is available [2].

In this paper, SIGNAL is not used as a programming language but as a support for describing DES. For this purpose, the language is restricted to boolean values. In general, other data types are available and SIGNAL can be used to describe hybrid systems [1].

As in the classical language-based framework, the principle of the modelling is to specify the sequence of events a DES can produce. However, SIGNAL relies on different notions of events and traces and on a different composition operation.

3.1 Events, Traces, and Processes

Given a finite set A of symbols called ports and a domain of value $\mathcal{D}$, let $\perp$ be a new symbol such that $\perp \notin \mathcal{D}$ and set $\mathcal{D}_\perp = \mathcal{D} \cup \{\perp\}$. Each element of A represents a communication channel on which a DES can receive or emit values. The symbol $\perp$ denotes the absence of value. An event ε is a mapping from A to $\mathcal{D}_\perp$; it represents an observation of the value present, or the absence of value, on each of the system's ports. If a value $x \in \mathcal{D}$ is present on a port a then $\varepsilon(a) = x$ else $\varepsilon(a) = \perp$.

For example, the movements of the mouse in the maze can be represented by events over the set of ports $M = \{M_1, \ldots, M_6\}$ (the sensors associated with the mouse's doors) with domain $\mathcal{D} = \{T\}$[1]. When the mouse passes door M_i an event ε_i is observed, defined by

$$\varepsilon_i(M_i) = T, \quad \text{and} \quad \varepsilon_i(M_j) = \perp \text{ if } i \neq j.$$

The successive moves of the mouse produce a sequence of such events. Note that, if there is no movement, nothing is observed. The sequence never contains the event $\perp_M$ such that

$$\forall i, \ \perp_M (M_i) = \perp \ .$$

[1] In this particular case, any singleton domain can be chosen; only the presence/absence of value is meaningful.

This is a general property in SIGNAL. Given a set of ports A, the event $\perp_A$ defined by

$$\forall a \in A,\ \perp_A (a) = \perp$$

is *invisible*; it never appears in the sequence of events one can observe.

An evolution of a system is then represented by a sequence of visible events called a *trace* and a system is modelled by a set of traces called a *process*.

In the sequel, only processes with boolean domain $\mathcal{D} = \{T, F\}$ will be considered. $\mathcal{B}_A$ will denote the set of all possible boolean traces over a set of ports A. For a trace $T = (\varepsilon_t)_{t \in \mathbb{N}}$ in $\mathcal{B}_A$ and a port $a \in A$, the notation $a(t)$ will be used instead of $\varepsilon_t(a)$.

The sequence $(a(t))_{t \in \mathbb{N}}$ is called a *signal*; signals are given the same name as their port.

3.2 Primitive operators

In SIGNAL, elementary processes are constructed using a set of primitive operators – synchronization, functions, delay, undersampling, and merging – which are presented in this section.

3.2.1 Synchronization

Given ports $\{a_1, \ldots, a_n\}$, the synchronization process denoted

$$\texttt{synchro}\{a_1, \ \ldots, \ a_n\}$$

is the process which admits any events where a value is present on all the ports. Its traces are defined by:

$$\forall t \in \mathbb{N},\ a_1(t) \neq \perp, a_2(t) \neq \perp, \ldots, a_n(t) \neq \perp .$$

All the ports carry values simultaneously; the signals are said to be *synchronous*.

3.2.2 Functions

Any boolean function f can be extended to a function on boolean traces. The equation

$$b \ \texttt{:= f}(a_1, \ldots, a_n)$$

represents the process over the ports $\{a_1, \ldots, a_n, b\}$ whose traces are defined by:

$$\forall t \in \mathbb{N},\ \begin{cases} a_1(t) \neq \perp, \ldots, a_n(t) \neq \perp \\ b(t) = f(a_1(t), \ldots, a_n(t)). \end{cases}$$

The first equation means that all the input ports carry values simultaneously and the second specifies the relation between the value over port b and the values over input ports.

3.2.3 Delay

The delay operator allows past values of a signal to be accessed. It is noted

$$b \texttt{ := } a \texttt{ \$1 init } x$$

where x is a boolean initialization value. The traces of this process are defined by

$$b(0) = x \quad \text{and} \quad \forall t \in \mathbb{N},\ b(t+1) = a(t).$$

This process behaves like a shift register. Initially, the port b holds the value x and subsequently each value on b is equal to the previous value on a. As in the case of functions and synchronizations, the signals are synchronous.

The initialization is optional. If no initial value is specified, $b(0)$ can be either true or false; the constraint $b(0) = x$ is replaced by $b(0) \neq \perp$.

3.2.4 Undersampling

Given two ports a and b, the undersampling operator selects some values of a depending on the values of the boolean port b. The traces of the process

$$c \texttt{ := } a \texttt{ when } b$$

are defined by:

$$\forall t \in \mathbb{N},\ \begin{cases} a(t) \neq \perp & \text{or} \quad b(t) \neq \perp \\ c(t) = a(t) & \text{if} \quad b(t) = T \\ c(t) = \perp & \text{if} \quad b(t) \neq T. \end{cases}$$

The value observed on c is equal to the value on a if, simultaneously, T is present on b. If a carries no value or if b is either absent or false, no value is present on c. Contrarily to the preceding operators, there is no synchronization constraint on a and b. For any event, a value can be present on a alone, on b alone, or simultaneously on both ports.

3.2.5 Merging

The last operator, **default**, merges two sequences of values. Given three ports a, b and c, the traces of the process

$$c \texttt{ := } a \texttt{ default } b$$

are defined by

$$\forall t \in \mathbb{N},\ \begin{cases} a(t) \neq \perp & \text{or} \quad b(t) \neq \perp \\ c(t) = a(t) & \text{if} \quad a(t) \neq \perp \\ c(t) = b(t) & \text{if} \quad a(t) = \perp . \end{cases}$$

The value observed on c is the same as on a if one is present else it is equal to the value on b. Like the undersampling, this operator does not impose any synchronization constraint between a and b.

3.3 Composition

The primitive operators are used to specify elementary processes in an equational manner. More complex systems can be constructed by composing these processes.

Consider two processes with disjoint set of ports A and B. For example, A and B can be the sensors recording respectively the movements of the cat and the movements of the mouse. There is no necessity that both systems evolve synchronously (that the cat and the mouse always move together). One can observe on $A \cup B$, events of the form $(\varepsilon_A, \perp_B)$, where ε_A is a visible event on A and $\perp_B$ the invisible event on B (only the cat is moving), or events of the form $(\perp_A, \varepsilon_B)$ (only the mouse is moving). But there is no reason either that events $(\varepsilon_A, \varepsilon_B)$ must be forbidden; it may happen that both animals move simultaneously.

According to this principle, if a trace $T_A = (\varepsilon_t)_{t\in\mathbb{N}}$ is observed on A and a trace $T_B = (\mu_t)_{t\in\mathbb{N}}$ is observed on B, a possible global observation might be:

$$\begin{array}{lcccccc} A: & \varepsilon_1 & \varepsilon_2 & \perp_A & \perp_A & \varepsilon_3 & \dots \\ B: & \perp_B & \mu_1 & \mu_2 & \mu_3 & \perp_B & \dots \end{array}$$

The projection of this trace on A is equal to T_A with invisible events $\perp_A$ freely inserted between each pair $(\varepsilon_t, \varepsilon_{t+1})$. In the same way, the projection on B is equal to T_B with invisible events $\perp_B$ inserted. Any trace over $A \cup B$ which satisfies these two properties can be a global observation.

This principle of composition can be generalized to processes with non-disjoint set of ports. Such systems communicate and interact with each other through their common ports. The traces produced by each process must be identical when projected on the common ports. For example, if $A = \{a, b\}$ and $B = \{b, c\}$, the two following traces T_A and T_B can be observed on A and B as the global system evolves:

$$T_A: \begin{array}{lccccccc} a: & T & T & \perp & F & \perp & T & \dots \\ b: & \perp & \perp & T & \perp & F & T & \dots \end{array}$$

$$T_B: \begin{array}{lccccccc} b: & T & F & \perp & T & \perp & \perp & \dots \\ c: & T & \perp & T & F & F & F & \dots \end{array}$$

For these two traces, the sequences of visible values on b are identical. The behaviour of the global system can be determined from the two observations:

$$T_{A\cup B}: \begin{array}{lcccccccc} a: & T & T & \perp & F & \perp & \perp & T & \dots \\ b: & \perp & \perp & T & \perp & F & \perp & T & \dots \\ c: & \perp & \perp & T & \perp & \perp & T & F & \dots \end{array}$$

As previously, the projection of this trace on A (resp. B) is equal to the trace T_A (resp. T_B) with invisible events inserted.

More formally, given two sets of ports A and B, and a domain $\mathcal{D}$, an event over $A \cup B$ is a mapping

$$\varepsilon : A \cup B \longrightarrow \mathcal{D}_\perp;$$

the restrictions of ε to A and B are denoted $\varepsilon_{/A}$ and $\varepsilon_{/B}$ respectively. This notation is extended to traces. Let $T = (\varepsilon_t)_{t \in \mathbb{N}}$ be a trace over $A \cup B$; $T_{/A}$ and $T_{/B}$ denote the sequences of events,

$$T_{/A} = (\varepsilon_{t\,/A})_{t \in \mathbb{N}} \quad \text{and} \quad T_{/B} = (\varepsilon_{t\,/B})_{t \in \mathbb{N}}.$$

In general, $T_{/A}$ and $T_{/B}$ are not traces because they can contain invisible events, but two traces noted $T_{/A}\downarrow$ and $T_{/B}\downarrow$ can be extracted from $T_{/A}$ and $T_{/B}$ by simply deleting the invisible events.

The composition of two processes can now be formally defined. Given a set of traces $\mathcal{H}_A$ on ports A and a set of traces $\mathcal{H}_B$ on ports B, the composition of $\mathcal{H}_A$ and $\mathcal{H}_B$ is the process over ports $A \cup B$ defined by

$$\mathcal{H}_{A \cup B} \quad = \quad \{T \in \mathcal{B}_{A \cup B} \;/\; T_{/A}\downarrow \in \mathcal{H}_A \text{ and } T_{/B}\downarrow \in \mathcal{H}_B\}.$$

Syntactically, the composition of two SIGNAL processes P_1 and P_2 is written $(|P_1|P_2|)$.

3.4 The Maze Example

3.4.1 Modelling the rooms

To illustrate the preceding notions, let us consider the possible movements of the mouse into or out of room 1. The mouse can come into this room through the door M_2 and go out through the door M_3 (cf. figure 1).

The two sensors associated with these doors are represented by two ports `M2` and `M3`. Each time a door is traversed a boolean value *true* is assumed to be emitted on the associated sensor. The signals `M2` and `M3` can then be either *true* or *absent*, never *false*.

To model the occupation of the room, two supplementary boolean signals are needed: `ZEM1` carries the current status of the room and `EM1` carries the future status. At a given instant, `ZEM1` holds the value *true* if the mouse is in room 1, else it holds the value *false*; `EM1` is *true* if the room will be occupied at the next instant, *false* otherwise.

The signals `ZEM1` and `EM1` are then related by:

```
ZEM1 := EM1 $1 init false;
```

the value of `ZEM1` is initially false (the mouse is not in room 1) and then it is equal to the previous value of `EM1`.

`EM1` evolves according to the signals `M2` and `M3`, and to the current status of the room: `EM1` becomes *true* when the mouse enters through the door M_2, *false* when it leaves through the door M_3. If none of the sensor signals a passage, `EM1` is equal to the current status `ZEM1`. This is described by the process:

```
EM1 := M2 default (not M3) default ZEM1.
```

An additional constraint

```
synchro { M3, M3 when ZEM1 }
```

specifies that door M_3 can be passed only when room 1 is occupied by the mouse. In the same way,

```
synchro { M2, M2 when not ZEM1 }
```

specifies that door M_2 can be passed only when the mouse is not in room 1.

The global process modelling this room is

```
(| ZEM1 := EM1 $1 init false
 | EM1 := M2 default (not M3) default ZEM1
 | synchro { M3, M3 when ZEM1 }
 | synchro { M2, M2 when not ZEM1 }
 |)
```

A possible trace of this process is the following:

M2 :	$\perp$	T	$\perp$	$\perp$	$\perp$	$\perp$	T	$\perp$	...
M3 :	$\perp$	$\perp$	$\perp$	$\perp$	T	$\perp$	$\perp$	T	...
EM1 :	F	T	T	T	F	F	T	F	...
ZEM1 :	F	F	T	T	T	F	F	T	...

Two processes similar to the previous one are associated with each room. One describes the movement of the cat, the other the movement of the mouse. The global system is modelled as the composition of all these processes. Simultaneous movement of both animals is allowed.

3.4.2 Modelling the control mechanism

To complete the description, it is necessary to model the control mechanism. For this purpose, new boolean ports are introduced. Each of them controls the opening and closing of a particular door. For each door, the specification is a copy of the following process:

```
(| OPEN := OPENCMD $1
 | synchro { SENSOR, SENSOR when OPEN }
 |)
```

`OPEN` represents the current position of the door. When it is true, the door is open; when it is false, the door is closed. The door is controlled by the port `OPENCMD`. Sending the value *true* on `OPENCMD` will open the door at the next instant, sending *false* will close it.

`SENSOR` is the signal emitted by the sensor associated with the door. The second relation states that the doorway can be traversed, and the sensor signal emitted, only when the door is open.

The control of the system is achieved by sending suitable values on the command port `OPENCMD` in order to open or close the doors. Note that `OPEN` is not initialized, it will be the role of the controller to choose the initial position of each door.

4 Polynomial Dynamic Systems

4.1 SIGNAL coding

4.1.1 Principle

SIGNAL processes can be translated into systems of polynomial equations over $\mathcal{F}_3$, the field of integers modulo 3. The principle is to code the three possible status of a boolean port by:

$$\begin{array}{lcl} \bot & \rightarrow & 0 \\ T & \rightarrow & 1 \\ F & \rightarrow & -1. \end{array}$$

Using this coding, events are represented by vectors in ${\mathcal{F}_3}^n$, where n is the number of ports, and traces by sequences of vectors.

Consider the elementary process `C := A when B` whose traces are defined by, for all $t \in \mathbb{N}$,

$$\begin{array}{lll} A(t) \neq \bot & \text{or} & B(t) \neq \bot \\ C(t) = A(t) & \text{if} & B(t) = T \\ C(t) = \bot & \text{if} & B(t) \neq T. \end{array}$$

Let a_t, b_t and c_t denote respectively the coding in $\mathcal{F}_3$ of $A(t)$, $B(t)$ and $C(t)$. The two last relations become:

$$\text{if } \; b_t = 1 \; \text{ then } \; c_t = a_t \; \text{ else } \; c_t = 0,$$

which can be rewritten

$$c_t \;\; = \;\; a_t(-b_t - b_t^2). \tag{1}$$

The constraint $A(t) \neq \bot$ or $B(t) \neq \bot$ could also be translated to a polynomial equation, but unlike processes, we do not want to discard the null vector representing the invisible event. The processs `C := A when B` is then represented by equation (1) alone. The sequences $(a_t, b_t, c_t)_{t \in \mathbb{N}}$ satisfying this equation encode not only traces of the process but also all the sequences of events T such that $T\downarrow$ is a trace of the process. This allows us to obtain the coding of the composition of two processes by simply juxtaposing their respective polynomial representations.

For example, the process `synchro{C, D}` is translated into the equation

$$c_t^2 = d_t^2,$$

which states that c_t and d_t can either be both null (`C` and `D` are absent together) or both non-null (`C` and `D` are present together). The pair $(0, 0)$ is a solution of this equation.

The composition of this process with the previous one

```
(| C := A when B
 | synchro{C, D}
 |)
```

is represented by the two following equations:

$$\begin{aligned} c_t &= a_t(-b_t - b_t^2) \\ c_t^2 &= d_t^2. \end{aligned}$$

Since the composed process admits events where neither `C` nor `D` carry a value, the system of equations must have solutions where $c_t = 0$ and $d_t = 0$.

4.1.2 Delay

The `default` operator and the boolean functions are transformed into polynomial equations in the same manner as the `when` and `synchro` operators. The translation of the delay operator is more complex because of its dynamic behaviour.

To represent the process

`B := A $1 init` x`,`

an auxiliary sequence $(\xi_t)_{t\in\mathbb{N}}$ is necessary. This sequence memorizes the non-null values of (a_t); it is defined by the equations:

$$\begin{aligned} \xi_0 &= x \\ \xi_{t+1} &= a_t + (1 - a_t^2)\xi_t. \end{aligned}$$

The first equation reflects the initialization; the second describes the evolution of ξ_t. If a_t is different from 0 then ξ_{t+1} is equal to a_t else ξ_{t+1} stays equal to ξ_t. The value of ξ_t is then equal to the initialization until the first non-null value of a_t and then ξ_t is equal to the most recent non-null value of a_t. The new variable ξ which memorizes the value of a is called a state variable.

The value of b_t is related to a_t and ξ_t by the equation

$$b_t = \xi_t a_t^2.$$

b_t is equal to the current state of the memory ξ_t when a_t is non null (`A` is present). Note that, as required by the composition mechanism, $a_t = 0, b_t = 0$ is a solution of the equation.

4.1.3 Polynomial Dynamic Systems

By putting together the equations representing the elementary processes, any SIGNAL specification is translated to a set of equations called a polynomial dynamic system.

For example, the coding yields, for the model of room 1, a polynomial dynamic system with variables $m2$, $m3$, $em1$, and $zem1$ corresponding to the ports `M2`, `M3`, `EM1` and `ZEM1`, and a state variable ξ introduced by the delay. The system consists of

- an initialization equation

$$\xi_0 = -1,$$

- an evolution equation

$$\xi_{t+1} = em1_t + (1 - em1_t^2)\xi_t,$$

- and a system of constraint equations

$$\begin{aligned} em1_t &= m2_t + (1 - m2_t^2)[-m3_t + (1 - m3_t^2)zem1_t] \\ zem1_t &= em1_t^2 \xi_t \\ m3_t^2 &= m3_t^2(-zem1_t - zem1_t^2) \\ m2_t^2 &= m2_t^2(zem1_t - zem1_t^2). \end{aligned}$$

The state variable ξ represents the occupancy of the room; if the mouse is in room 1 at time t then $\xi_t = 1$ else $\xi_t = -1$. The constraint equations define the possible event vectors $(m2_t, m3_t, em1_t, zem1_t)$ which can occur depending on the value of ξ_t. The evolution equation gives the new state resulting from such an event.

This example shows the general structure of the polynomial representation of a process. It consists of three systems of equations – initialization, evolution and constraint equations – relating state variables and event variables.

4.2 Definitions

4.2.1 Notations

From now on, in order to simplify the notations, the index t appearing in polynomial dynamic system will be suppressed, and the notation ξ' will be used instead of ξ_{t+1}.

X, Y, Z denote finite sets of variables; $\mathcal{F}_3[X]$, $\mathcal{F}_3[X, Y]$, $\mathcal{F}_3[Z]$, are rings of polynomials over these variables with coefficients in $\mathcal{F}_3$; $Q(X, Y) = 0$, $Q_0(X) = 0$, etc. are systems of polynomial equations.

4.2.2 Polynomial Dynamic Systems

Definition 1 *A polynomial dynamic system is a triple of systems of polynomial equations of the form*

$$\left\{ \begin{array}{rcl} Q(X, Y) & = & 0 \\ X' & = & P(X, Y) \\ Q_0(X) & = & 0 \end{array} \right.$$

where

- *X is a set of variables called state variables,*
- *Y is a set of variables called event variables,*
- *$Q(X, Y) = 0$ is the constraint equation,*

- $X' = P(X, Y)$ *the evolution equation, and*
- $Q_0(X) = 0$ *the initialization equation of the system.*

Let n and m be the number of variables in, respectively, X and Y. P can be considered as a function from ${\mathcal{F}_3}^{n+m}$ to ${\mathcal{F}_3}^n$, and Q and Q_0 as vectorial functions. A polynomial dynamic system implicitly defines a finite transition system with set of states ${\mathcal{F}_3}^n$ and set of events ${\mathcal{F}_3}^m$. The initial states of this automaton are the solutions of the equation $Q_0(x) = 0$. When the system is in a state $x \in {\mathcal{F}_3}^n$, any event $y \in {\mathcal{F}_3}^m$ such that $Q(x, y) = 0$ can be produced; such an event is said to be *admissible* in x. The system then evolves to the state $x' = P(x, y)$. The notation $x \xrightarrow{y} x'$ will be used as an abbreviation for

$$Q(x, y) = 0 \text{ and } x' = P(x, y).$$

A finite or infinite sequence of pairs (x_i, y_i) such that

$$x_0 \xrightarrow{y_0} x_1 \xrightarrow{y_1} x_2 \ldots$$

is a *trajectory initiated in* x_0.

4.2.3 Basic properties

The study of polynomial dynamic systems is based on properties such as liveness, invariance, reachability, recurrence [7]. In the sequel, the following notions will be useful.

Definition 2 *A set of states E is invariant for a polynomial dynamic system if, for any state x of E and any event y admissible in x, the successor state $P(x, y)$ also belongs to E.*

Definition 3 *A state x' is reachable from a state x if there exists a trajectory*

$$x_0 \xrightarrow{y_0} x_1 \xrightarrow{y_1} x_2 \ldots \xrightarrow{y_{k-1}} x_k$$

such that $x_0 = x$ and $x_k = x'$.

A set of states F is reachable from a state x if all the elements of F are reachable from x.

A set of states F is reachable from a set of states E if it is reachable from every state in E.

Definition 4 *The orbit of a polynomial dynamic system is the set of all states reachable from one of the initial states.*

If a set of states E is invariant, any trajectory initiated in E visits only elements of E. The orbit of a system is an example of invariant set.

The translation of processes to polynomial dynamical systems was initially introduced as a tool for verifying properties of SIGNAL programs [8, 6]. Reachability and invariance of desirable sets of states are examples of such properties.

4.3 Algebraic tools

4.3.1 Ideals and Varieties

In order to reason about polynomial dynamic systems, elementary algebraic notions are used. The idea is to convert geometric properties expressed in terms of set of states or set of events to equivalent equational properties. The core of this approach is the representation of sets by polynomial ideals.

Let $Z = \{Z_1, \ldots, Z_p\}$ be a set of variables, $\mathcal{F}_3[Z]$ the ring of polynomials with variables Z and E a subset of ${\mathcal{F}_3}^p$. The following set of polynomials

$$\mathcal{I}(E) \quad = \quad \{g \in \mathcal{F}_3[Z] \ / \ \forall z \in E, g(z) = 0\}$$

is an ideal of $\mathcal{F}_3[Z]$.

Reciprocally, to any set of polynomials $G \subseteq \mathcal{F}_3[Z]$ (not necessarily an ideal) is associated a subset $\mathcal{V}(G)$ of ${\mathcal{F}_3}^p$, called a variety, defined by

$$\mathcal{V}(G) \quad = \quad \{z \in {\mathcal{F}_3}^p \ / \ \forall g \in G, g(z) = 0\}.$$

The following theorem is fundamental.

Theorem 1 *For any ideal $\underline{a} \subseteq \mathcal{F}_3[Z]$ and any set $E \subseteq {\mathcal{F}_3}^p$,*

$$\mathcal{V}(\mathcal{I}(E)) \quad = \quad E, \tag{2}$$

$$\mathcal{I}(\mathcal{V}(\underline{a})) \quad = \quad \underline{a} + <Z_1^3 - Z_1, \ldots, Z_p^3 - Z_p>. \tag{3}$$

The first relation means that any subset of ${\mathcal{F}_3}^p$ is a variety and that $\mathcal{I}(E)$ is characteristic of E. If $E_1 \neq E_2$ then $\mathcal{I}(E_1) \neq \mathcal{I}(E_2)$.

The second relation allows a set of generators of $\mathcal{I}(E)$ to be easily obtained from equations defining E. If E is the set of solutions of a system of equations

$$\begin{cases} g_1(z) & = & 0 \\ & \vdots & \\ g_k(z) & = & 0, \end{cases}$$

it is clear that $E = \mathcal{V}(<g_1, \ldots, g_k>)$ and then, by relation (3),

$$\mathcal{I}(E) \quad = \quad <g_1, \ldots, g_k, Z_1^3 - Z_1, \ldots, Z_p^3 - Z_p>.$$

In the sequel, the ideal spanned by the polynomials $Z_1^3 - Z_1, \ldots, Z_p^3 - Z_p$ is denoted $<Z^3 - Z>$. This notation is used for any set of variables.

The following elementary relations transform basic geometric properties of varieties to equivalent properties of their characteristic ideals.

Property 1 *Given two subsets E_1 and E_2 of ${\mathcal{F}_3}^p$,*

$$\begin{aligned} E_1 \subseteq E_2 &\Leftrightarrow \mathcal{I}(E_2) \subseteq \mathcal{I}(E_1), \\ \mathcal{I}(E_1 \cap E_2) &= \mathcal{I}(E_1) + \mathcal{I}(E_2), \\ \mathcal{I}(E_1 \cup E_2) &= \mathcal{I}(E_1) \cap \mathcal{I}(E_2). \end{aligned}$$

Given a subset E of ${\mathcal{F}_3}^p$ and a new set of variables $Z' = \{Z'_1, \ldots, Z'_{p'}\}$,

$$\mathcal{I}(E \times {\mathcal{F}_3}^{p'}) = \mathcal{I}(E)\mathcal{F}_3[Z, Z'],$$

where $\mathcal{I}(E)\mathcal{F}_3[Z, Z']$ is the ideal spanned by $\mathcal{I}(E)$ in the ring $\mathcal{F}_3[Z, Z']$.

4.3.2 Comorphism

The evolution equation of a polynomial dynamic system also enjoys an algebraic counterpart. The equation $X' = P(X, Y)$ is of the form

$$\left\{ \begin{array}{rcl} X'_1 &=& P_1(X, Y) \\ &\vdots& \\ X'_n &=& P_n(X, Y) \end{array} \right.$$

where $P_1(X, Y), \ldots, P_n(X, Y)$ are polynomials in $\mathcal{F}_3[X, Y]$. P can then be considered as a function from ${\mathcal{F}_3}^{n+m}$ to ${\mathcal{F}_3}^n$. From P is derived a function $P^\star$ from $\mathcal{F}_3[X]$ to $\mathcal{F}_3[X, Y]$, called a *comorphism*, defined by

$$P^\star(g(X_1, \ldots, X_n)) = g(P_1(X, Y), \ldots, P_n(X, Y)).$$

The image of a polynomial $g \in \mathcal{F}_3[X]$ by $P^\star$ is obtained by subsituting $P_i(X, Y)$ for each variable X_i. $P^\star$ is a ring homorphism and its effect on ideals is described by the following theorem.

Theorem 2 *Given a set $E \subseteq {\mathcal{F}_3}^{n+m}$ and an ideal $\underline{a} \subseteq \mathcal{F}_3[X]$,*

$$\mathcal{I}(P(E)) = P^{\star -1}(\mathcal{I}(E)), \tag{4}$$

$$\mathcal{V}(P^\star(\underline{a})) = P^{-1}(\underline{a}). \tag{5}$$

In general, $P^\star(\underline{a})$ is not an ideal. However relations (5) and (3) yield

$$\mathcal{I}(P^{-1}(E)) = <P^\star(\mathcal{I}(E))> + <X^3 - X, Y^3 - Y>. \tag{6}$$

As a consequence, given a set of polynomials $g_1, \ldots, g_k$ such that $\mathcal{I}(E) = <g_1, \ldots, g_k>$, generators of $\mathcal{I}(P^{-1}(E))$ are obtained by

$$\mathcal{I}(P^{-1}(E)) = <P^\star(g_1), \ldots, P^\star(g_k), X_1^3 - X_1, \ldots, Y_1^3 - Y_1, \ldots>. \tag{7}$$

4.3.3 Example: verification of invariance

All the previous algebraic tools can be used to verify properties of polynomial dynamic systems. For example, suppose one wants to verify whether or not a given set of states E is invariant.

By definition 2, E is invariant if and only if

$$\forall x \in E, \forall y \in {\mathcal{F}_3}^m, \quad Q(x,y) = 0 \;\Rightarrow\; P(x,y) \in E.$$

Let $<Q>$ be the ideal spanned by the constraint equations and polynomials $X_i^3 - X_i, Y_i^3 - Y_i$; the variety $\mathcal{V}(<Q>)$ is the set of all pairs (x,y) such that y is admissible in x. The previous relation is equivalent to

$$E \times {\mathcal{F}_3}^m \;\cap\; \mathcal{V}(<Q>) \quad\subseteq\quad P^{-1}(E).$$

By property (1) and relation (6) the following property can be deduced.

Property 2 *A set of states E is invariant if and only if*

$$<P^\star(\mathcal{I}(E))> \quad\subseteq\quad \mathcal{I}(E)\mathcal{F}_3[X,Y] + <Q>.$$

4.4 Principal generators

The previous transformations result in relations between ideals which can be verified using formal calculus. Gröbner bases [5] is a classical tool of effective algebra capable of solving problems of this kind. It relies on particular canonical sets of generators of the ideals. Gröbner bases are general and can be used in any polynomial ring, but the computation of the canonical generators can be extremely expensive when more than a few variables are involved. For this reason, a different implementation which makes use of the peculiarities of our applications has been devised.

As a consequence of theorem 1, it is possible to work in the quotient ring

$$\mathcal{F}_3[Z] \;/{<}Z^3 - Z{>}.$$

This ring is isomorphic to the ring of functions from ${\mathcal{F}_3}^p$ to $\mathcal{F}_3$. In this new ring, any ideal $\underline{a}$ is characteristic of a variety, i.e. $\mathcal{I}(\mathcal{V}(\underline{a})) = \underline{a}$, and we still have $\mathcal{V}(\mathcal{I}(E)) = E$.

Working in the quotient ring offers two simplifications. First, any element of this ring can be represented by a polynomial of degree at most two in every variable. Second, any ideal can be spanned by a single element, called a principal generator. This results from the property:

Property 3 *Let $\underline{a}$ be an ideal in $\mathcal{F}_3[Z] \;/{<}Z^3 - Z{>}$ and $\{g_1, \ldots, g_k\}$ be a set of generators of $\underline{a}$, then the polynomial function*

$$f \quad = \quad 1 - \prod_{i=1}^{k} (1 - g_i^2)$$

is a principal generator of $\underline{a}$.

Proof: Let $\underline{b}$ be the ideal generated by f.

- The development of $\prod_{i=1}^{k}(1-g_i^2)$ yields an expression of the form $1+h$ where h is a combination of the polynomial fuctions $g_1,\ldots,g_k$. h is then an element of $\underline{a}$; $f=1-(1+h)=-h$ also belongs to $\underline{a}$ and then $\underline{b}\subseteq\underline{a}$.
- Reciprocally, for any g_i, we have

$$fg_i \quad = \quad g_i - g_i\prod_{j=1}^{k}(1-g_j^2) \quad = \quad g_i-(g_i-g_i^3)\prod_{j=1,j\neq i}^{k}(1-g_j^2).$$

  In $\mathcal{F}_3[Z]\ /<Z^3-Z>$, $g_i-g_i^3=0$ then $fg_i=g_i$. This shows that $g_i\in\underline{b}$ and so $\underline{a}\subseteq\underline{b}$. □

The link between principal generators and varieties is the following:

Property 4 *Given a subset E of $\mathcal{F}_3{}^p$, a polynomial function f is a principal generator of the ideal $\mathcal{I}(E)$ if and only if, for any element x in $\mathcal{F}_3{}^p$,*

$$\begin{aligned} x\in E \quad &\Rightarrow \quad f(x)=0\\ x\notin E \quad &\Rightarrow \quad f(x)\neq 0\end{aligned}$$

By using principal generators, the ideal computations reduce to simple polynomial operations.

Property 5 *Let E_1 and E_2 be two subsets of $\mathcal{F}_3{}^p$ and f_1 and f_2 be principal generators of, respectively, $\mathcal{I}(E_1)$ and $\mathcal{I}(E_2)$, then*

- *$1-f_1^2$ is a principal generator of $\mathcal{I}(\mathcal{F}_3{}^p-E_1)$,*
- *f_1f_2 is a principal generator of $\mathcal{I}(E_1)\cap\mathcal{I}(E_2)=\mathcal{I}(E_1\cup E_2)$,*
- *$f_1^2+f_2^2$ is a principal generator of $\mathcal{I}(E_1)+\mathcal{I}(E_2)=\mathcal{I}(E_1\cap E_2)$,*
- *$\mathcal{I}(E_1)\subseteq\mathcal{I}(E_2)\ \Leftrightarrow\ E_1\supseteq E_2$ if and only if $f_1(1-f_2^2)=0$.*

The projection is another useful operation on varieties. Suppose E is a set of pairs state/event; E is a subset of $\mathcal{F}_3{}^{n+m}$. We write $Proj_X(E)$ for the projection of E on the state components:

$$Proj_X(E) \quad = \quad \{x\in\mathcal{F}_3{}^n\ /\ \exists y\in\mathcal{F}_3{}^m,\ (x,y)\in E\}.$$

The characteristic ideal of $Proj_X(E)$ is obtained by:

$$\mathcal{I}(Proj_X(E)) \quad = \quad \mathcal{I}(E)\cap\mathcal{F}_3[X],$$

and a principal generator of this ideal can be computed from a principal generator of $\mathcal{I}(E)$ by repeated applications of the rule:

$$\exists z_1\in\mathcal{F}_3,\ f(z_1,\ldots,z_p)=0 \quad\Leftrightarrow\quad f(1,z_2,\ldots,z_p)f(-1,z_2,\ldots,z_p)f(0,z_2,\ldots,z_p)=0.$$

To implement the operations on principal generators, a powerful symbolic calculus system is needed. We use the representation of functions by ternary decision diagrams, a slight extension of BDDs [3] which are very efficient in boolean algebra and other areas [4].

5 Control Problems

5.1 Controlled Polynomial Dynamic Systems

After immediate simplifications, the polynomial dynamic system obtained from the cat and mouse example can be written

$$S : \begin{cases} Q(X,Y,U) & = & 0 \\ X' & = & P(X,Y,U) \\ Q_0(X) & = & 0. \end{cases}$$

The state variables X represent the occupation of the rooms and the position of the doors. The initial state is only partially defined; the initial occupation of the rooms is determined but the position of the doors is free.

The event variables are of two different natures. The variables Y represent the signals received from the sensors, the variables U are the commands signals to control the doors.

A system of this form is called a controlled polynomial dynamic system; Y and U are the set of, respectively, uncontrolled and controlled event variables.

Let n, m, and p be the respective dimension of X, Y, and U. The trajectories of a controlled system are sequences (x_t, y_t, u_t) in ${\mathcal{F}_3}^n \times {\mathcal{F}_3}^m \times {\mathcal{F}_3}^p$ such that $Q_0(x_0) = 0$ and, for all t,

$$\begin{aligned} Q(x_t, y_t, u_t) &= 0 \\ x_{t+1} &= P(x_t, y_t, u_t). \end{aligned}$$

The events (y_t, u_t) include an uncontrolled component y_t and a controlled one u_t. We have no direct influence on the y_t part which depends only on the state x_t. On the contrary, we have full control on u_t, we can choose any value of u_t which is admissible, i.e. such that $Q(x_t, y_t, u_t) = 0$. The chosen value determines the next state x_{t+1} and indirectly influences the possible values for y_{t+1}.

To distinguish the two components, a vector $y \in {\mathcal{F}_3}^m$ is called an *event* and a vector $u \in {\mathcal{F}_3}^p$ a *command*. This leads to a new notion of admissibility: an event y is *admissible* in a state x if there exists a command u such that $Q(x, y, u) = 0$; such a command is said to be *compatible* with y in x.

5.2 Controllers

A dynamic system of the same form as S can be controlled by first selecting a particular initial state x_0 and then by chosing suitable values for $u_1, u_2, u_3, \ldots$ For example, in the cat and mouse case, the choice of x_0 corresponds to the initial positioning of the doors and each u_t is a command to open or close the doors.

There are different ways to determine the possible command values. A simple case is where u_t can be determined from x_t and y_t uniquely. In this case, the controller does not

need any memory of the values preceding x_t and y_t; such a controller is called a *static controller*.

Definition 5 *A static controller is a system of two equations:*

$$\begin{aligned} C(X,Y,U) &= 0 \\ C_0(X) &= 0. \end{aligned}$$

The equation $C_0(X) = 0$ determines the initial states the controller can select. The other equation describes the possible commands which can be chosen; when the controlled system is in state x,and an event y occur, any command u such that

$$Q(x,y,u) = 0 \quad \text{and} \quad C(x,y,u) = 0$$

can be selected.

The control is not necessarily deterministic. Several initial states may be chosen and at each instant several commands may be suitable.

The behaviour of a system S under the supervision of a static controller (C, C_0) is modelled by the composed system

$$S_c : \begin{cases} Q(X,Y,U) &= 0 \\ C(X,Y,U) &= 0 \\ X' &= P(X,Y,U) \\ Q_0(X) &= 0 \\ C_0(X) &= 0. \end{cases}$$

To control the physical system modelized by S (the maze) it suffices to connect it to S_c as in figure 2. S_c receives the signals Y, its internal state X is the same as the actual state of the controlled system[2], and S_c controls the maze through the signals U. Since the state of the maze is not directly available, it must be reconstructed; both S and the control equations (C, C_0) are necessary.

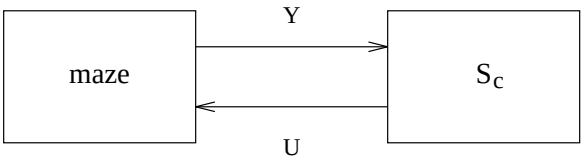

Figure 2: Feedback control

Not any pair (C, C_0) can constitute an acceptable controller. On the one hand, it is necessary that S_c can be initialized; the equations $C_0(X) = 0$ and $Q_0(X) = 0$ must have common solutions. On the other hand, due to the uncontrollability of Y, any event that the system S can produce must also be admissible by S_c.

[2]There is no observability issue here. The initial state is known and S is a perfect model.

Definition 6 *A static controller (C, C_0) is acceptable for a system S if the two following conditions are satisfied:*

1. *The initial constraints $C_0(X) = 0$ and $Q_0(X) = 0$ have common roots.*
2. *For any state x of the orbit of S_c, any event y admissible in x for S is also admissible in x for S_c.*

5.3 Synthesis of Controllers

Given a polynomial dynamic system

$$S : \begin{cases} Q(X,Y,U) & = & 0 \\ X' & = & P(X,Y,U) \\ Q_0(X) & = & 0. \end{cases}$$

and a control objective. The problem is to calculate two constraints $C_0(X) = 0$ and $C(X,Y,U) = 0$ such that (C, C_0) is an acceptable controller for S and the system

$$S_c : \begin{cases} Q(X,Y,U) & = & 0 \\ C(X,Y,U) & = & 0 \\ X' & = & P(X,Y,U) \\ Q_0(X) & = & 0 \\ C_0(X) & = & 0 \end{cases}$$

satisfies the given control objective.

For the cat and mouse example, the objective is twofold:

- The cat and the mouse must never occupy the same room.
- It is always possible for the animals to return to the initial position.

This objective is a combination of an invariance and a reachability property. Let G_1 be the set of states such that the cat and the mouse do not occupy the same room and G_2 be the set of states such that the mouse is in room 4 and the cat in room 2. In both sets, the position of the doors is unspecified. The control problem is to find C and C_0 such that, for the system S_c,

(a1) the orbit is included in G_1,

(a2) G_2 is reachable from the orbit.

The condition (a1) states that any accessible state x of S_c belongs to G_1. The condition (a2) states that from any state x accessible by S_c, there exists a trajectory initiated in x which reaches G_2. The first objective will be called *ensuring the invariance of* G_1, the second *ensuring the reachability of* G_2.

5.3.1 Ensuring invariance

Consider first, the simpler case of an acceptable controller which guarantee only objective (a1). Suppose there exists a solution (C, C_0) and let O be the orbit of S_c.

From definition 4, it is clear that O is invariant for S_c:

$$\forall x \in O, \forall y \in {\mathcal{F}_3}^m, \forall u \in {\mathcal{F}_3}^p, \quad Q(x,y,u) = 0 \text{ and } C(x,y,u) = 0 \;\Rightarrow\; P(x,y,u) \in O.$$

Now let x be an element of O and y an event admissible in x by the system S. Since the controller is acceptable, y is also admissible in x for S_c; there exists a command u such that

$$Q(x,y,u) = 0 \text{ and } C(x,y,u) = 0,$$

and, for this command, $P(x,y,u)$ belongs to O. For the system S, O possesses a property similar to invariance. We say that O is control-invariant.

Definition 7 *A set of states E is control-invariant for a system*

$$S : \left\{ \begin{array}{ccc} Q(X,Y,U) & = & 0 \\ X' & = & P(X,Y,U) \\ Q_0(X) & = & 0. \end{array} \right.$$

if, for every state x in E and every event y admissible in x, there exists a command u such that

$$Q(x,y,u) = 0 \quad \textit{and} \quad P(x,y,u) \in E.$$

As previously, let $<Q>$ be the ideal spanned by the constraint equations and $\mathcal{V}(<Q>)$ the associated variety. The projection of $\mathcal{V}(<Q>)$ on components X, Y is the set of all pairs (x,y) such that y is admissible in x for S:

$$Proj_{X,Y}(\mathcal{V}(<Q>)) \quad = \quad \{(x,y) \;/\; \exists u \in {\mathcal{F}_3}^p, Q(x,y,u) = 0\}.$$

The characteristic ideal of this set is

$$\mathcal{I}(Proj_{X,Y}(\mathcal{V}(<Q>))) \quad = \quad <Q> \cap \mathcal{F}_3[X,Y].$$

Let E be a control-invariant set of states, by definition, we have

$$\begin{array}{l} \forall x \in E, \forall y \in {\mathcal{F}_3}^m, \quad (x,y) \in Proj_{X,Y}(\mathcal{V}(<Q>)) \Rightarrow \\ \qquad \exists u \in {\mathcal{F}_3}^p, \; Q(x,y,u) = 0 \text{ and } \mathrm{P(x,y,u)} \in \mathrm{E}, \end{array}$$

or, equivalently,

$$(E \times {\mathcal{F}_3}^m) \cap Proj_{X,Y}(\mathcal{V}(<Q>)) \quad \subseteq \quad Proj_{X,Y}(\mathcal{V}(<Q>) \cap P^{-1}(E)).$$

This inclusion is easily translated into the following property.

Property 6 *A set of states E is control-invariant if and only if*

$$(<Q> + <P^\star(\mathcal{I}(E))>) \cap \mathcal{F}_3[X,Y] \quad \subseteq \quad \mathcal{I}(E)\mathcal{F}_3[X,Y] + (<Q> \cap \mathcal{F}_3[X,Y]).$$

If there exists an acceptable controller which guarantees the invariance of G_1, the orbit O of S_c is included in G_1 and is control-invariant for S. We also have $O \cap \mathcal{V}(<Q_0>) \neq \emptyset$. The essential result is that these three properties are sufficient.

Theorem 3 *Given a controlled system S and a set G of states of S, there exists an acceptable, static controller which guarantees the invariance of G if and only if there exists a set of states O such that:*

(b1) $O \subseteq G$,

(b2) $O \cap \mathcal{V}(<Q_0>) \neq \emptyset$,

(b3) *O is control-invariant for S.*

Proof: We have already shown that the conditions are necessary. Conversely, suppose there exists a set O which satisfies the three properties. Let C_0 be a principal generator of $\mathcal{I}(O)$ and $C = P^\star(C_0)$. By construction, we have

- $C_0(x) = 0 \quad \Leftrightarrow \quad x \in O$,
- $C(x,y,u) = 0 \quad \Leftrightarrow \quad P(x,y,u) \in O$.

It follows that the orbit of S_c is included in O and hence in G; the controller (C, C_0) ensures the invariance of G.

Now, let x be a state in the orbit of S_c and y an event admissible in x for S. x is also an element of O and since O is control-invariant, there exists a command u such that

$$Q(x,y,u) = 0 \quad \text{and} \quad P(x,y,u) \in O.$$

This is equivalent to

$$Q(x,y,u) = 0 \quad \text{and} \quad C(x,y,u) = 0.$$

y is then also admissible in x for S_c. Since the condition $O \cap \mathcal{V}(<Q_0>) \neq \emptyset$ means that $C_0(X) = 0$ and $Q_0(X) = 0$ have common solutions, the controller (C, C_0) is acceptable. □

The proof gives an algorithm to obtain a controller ensuring the invariance of G_1. It suffices to find a control-invariant subset of G_1 which satisfies the initialization condition (b2).

G_1 contains at least one control-invariant subset, the empty set. It is easy to see that the union of two control-invariant sets is also control-invariant. As a consequence, there exists

a greatest control-invariant subset of G_1. Let O be this subset; if O satisfies the condition (b2) then control equations can be obtained from generators of $\mathcal{I}(O)$ else no subset of O can satisfy condition (b2) and the problem has no solution.

Let Q' be a principal generator of $<Q> \cap \mathcal{F}_3[X,Y]$; for any pair (x,y), we have:

$$Q'(x,y) = 0 \quad \Leftrightarrow \quad \exists u \in {\mathcal{F}_3}^p, Q(x,y,u) = 0.$$

The computation of O uses the operator δ defined by, for any set of states E,

$$\delta(E) \quad = \quad \{x \ / \ \forall y, \quad Q'(x,y) = 0 \ \Rightarrow \ \exists u, Q(x,y,u) = 0 \text{ and } P(x,y,u) \in E\}.$$

From definition 7, it is clear that E is control-invariant if and only if $E \subseteq \delta(E)$.

The greatest control-invariant subset of G_1 is obtained by constructing the sequence $(E_i)_{i \in \mathbb{N}}$ defined by:

$$\begin{aligned} E_0 &= G_1 \\ E_{i+1} &= E_i \cap \delta(E_i). \end{aligned}$$

The sequence is decreasing. Since all sets E_i are finite, there exists an index j such that $E_{j+1} = E_j$. The set E_j is the greatest control-invariant subset of G_1; O is equal to E_j.

In practice, we transform this computation to an equivalent sequence of principal generators $(g_i)_{i \in \mathbb{N}}$ and g_j is a principal generator of $\mathcal{I}(O)$.

5.3.2 Ensuring reachability

Now, we consider controllers ensuring both control objectives, the invariance of G_1 and the reachability of G_2.

Suppose there exists such a controller and let O' be the orbit of S_c for this controller. O' satisfies the same three conditions as previously and a supplementary one: any state $x \in O'$ is the origin of a trajectory in S_c which reaches G_2. But any trajectory

$$x_0 \xrightarrow{y_0} x_1 \xrightarrow{y_1} \ldots \xrightarrow{y_{k-1}} x_k$$

of S_c is also a trajectory of S where all the visited states belongs to O'.

For a set of states E, let $Pre(E)$ be the set of predecessors of E;

$$\begin{aligned} Pre(E) &= \{x \ / \ \exists y, \exists u, \ Q(x,y,u) = 0 \text{ and } P(x,y,u) \in E\} \\ &= Proj_X(\mathcal{V}(<Q>) \cap P^{-1}(E)). \end{aligned}$$

The sequence of sets $(D_i)_{i \in \mathbb{N}}$ defined by

$$\begin{aligned} D_0 &= E \cap G_2 \\ D_{i+1} &= D_i \cup (E \cap Pre(D_i)), \end{aligned}$$

converges whatever the set E. The fixed point is the set of states x which are origin of a trajectory of S leading to G_2 and containing only states of E. This set is denoted $Reach(E, G_2)$.

The supplementary property of the orbit O' can now be rewritten

$$O' \subseteq Reach(O', G_2).$$

The following theorem is similar to theorem 3.

Theorem 4 *Given a controlled system S, and two set of states G_1 and G_2, there exists an acceptable, static controller which guarantees the invariance of G_1 and the reachability of G_2 if and only if there exists a set of states O' such that:*

(c1) $O' \subseteq G_1$,

(c2) $O' \cap \mathcal{V}(<Q_0>) \neq \emptyset$,

(c3) O' *is control-invariant for* S,

(c4) $O' \subseteq Reach(O', G_2)$.

Proof: The demonstration is similar to that of theorem 3. If such an O' exists, let C_0 be a principal generator of $\mathcal{I}(O')$ and $C = P^\star(C_0)$. From theorem 3, the pair (C, C_0) is an acceptable controller ensuring the invariance of G_1.

Let x be an element of O', since x belongs to $Reach(O', G_2)$, there exists a trajectory of S

$$x_0 \stackrel{y_0,u_0}{\longrightarrow} x_1 \stackrel{y_1,u_1}{\longrightarrow} \ldots \stackrel{y_{k-1},u_{k-1}}{\longrightarrow} x_k$$

with $x_0 = x$, $x_k \in G_2$ and $\forall i, 0 \leq i \leq k, x_i \in O'$. For all the indices $i \in [0 \ldots k-1]$, we have

$$Q(x_i, y_i, u_i) = 0 \quad \text{and} \quad P(x_i, y_i, u_i) \in O'.$$

This is equivalent to

$$Q(x_i, y_i, u_i) = 0 \quad \text{and} \quad C(x_i, y_i, u_i) = 0$$

and shows that the trajectory is also a trajectory of S_c initiated in x and reaching G_2. Hence (C, C_0) ensures the reachability condition. □

The computation of a controller follows the same principle as in the invariance case. It consists in finding the greatest control-invariant subset O' of G_1 which also satisfies $O' \subseteq Reach(O', G_2)$.

This is obtained by constructing the sequence $(E_i)_{i \in \mathbb{N}}$ as follows:

$$\begin{aligned} E_0 &= G_1 \\ E_{i+1} &= Reach(E_i \cap \delta(E_i), G_2). \end{aligned}$$

From the construction of $Reach$, for any set E, $Reach(E, G_2)$ is included in E; this implies that

$$E_{i+1} \subseteq E_i \cap \delta(E_i) \subseteq E_i.$$

The sequence is decreasing and then stationary; there exists j such that $E_j = E_{j+1}$ and then

$$E_j \subseteq E_j \cap \delta(E_j) \subseteq E_j.$$

This verifies

$$\begin{aligned} E_j &\subseteq \delta(E_j) \\ E_j &\subseteq Reach(E_j, G_2). \end{aligned}$$

E_j is control-invariant, is included in G_1 and satisfies condition (c4). It can also be shown that E_j is the greatest such set.

As previously, all these computations have equivalents in terms of ideals and principal generators.

5.4 Minimally restrictive control

In this section, we write (C, C_0) for the controller obtained by the preceding method. This controller guarantees the invariance of G_1 and the reachability of G_2. Suppose (D, D_0) is another acceptable controller ensuring the same objectives, and let S_c and S_d be the two following systems.

$$S_c : \begin{cases} Q(X,Y,U) = 0 \\ C(X,Y,U) = 0 \\ X' = P(X,Y,U) \\ Q_0(X) = 0 \\ C_0(X) = 0 \end{cases} \qquad S_d : \begin{cases} Q(X,Y,U) = 0 \\ D(X,Y,U) = 0 \\ X' = P(X,Y,U) \\ Q_0(X) = 0 \\ D_0(X) = 0. \end{cases}$$

From the construction of C_0 and C, it can be shown that the orbit of S_d is included in the orbit of S_c. We then have the following properties:

$$\begin{aligned} Q_0(x) = 0 \text{ and } D_0(x) = 0 &\Rightarrow Q_0(x) = 0 \text{ and } C_0(x) = 0 \\ Q(x,y,u) = 0 \text{ and } D(x,y,u) = 0 &\Rightarrow Q(x,y,u) = 0 \text{ and } C(x,y,u) = 0. \end{aligned}$$

This means that S_c can simulate S_d. Any initial state which can be chosen by S_d can also be selected by S_c. Any state x accessible by S_d is also accessible by S_c and, in this state, any command u that S_d can emit in response to an event y can also be emitted by S_c.

For the cat and mouse example, a controller (D, D_0) which closes all the doors is acceptable and guarantees the objectives. Since S_c can simulate this controller, it can also close all the doors in the initial position. In order to allow the animals to move as freely as possible, we have to discard such restrictive behaviours of S_c.

Suppose the system S_c is in a state x and receives an event y. The controller can choose any command u such that

$$Q(x, y, u) = 0 \quad \text{and} \quad C(x, y, u) = 0.$$

Let u_1 and u_2 be two of the possible commands, and x_1 and x_2 the two corresponding successor states:

$$\begin{aligned} x_1 &= P(x, y, u_1), \\ x_2 &= P(x, y, u_2). \end{aligned}$$

A minimally restrictive control can be enforced by adopting the following strategy: Let Adm_{x_1} and Adm_{x_2} be the set of events admissible in, respectively, x_1 and x_2. Three cases are possible.

- If Adm_{x_1} is a strict subset of Adm_{x_2}, there are more movements possible in x_2 than in x_1. The controller must choose the command u_2 rather than u_1.
- On the contrary, if $Adm_{x_2} \subset Adm_{x_1}$, it must choose u_1.
- If none of the previous conditions is satisfied, u_1 and u_2 are incomparable, both can be chosen.

The choice between different initial positions is based on the same principle.

Formally, the above strategy is based on a strict order relation between different states. It is possible to define this strict order in terms of ideals and principal generators. The result is a polynomial function R such that, for any pair of states (x_1, x_2),

$$R(x_1, x_2) = 0 \quad \Leftrightarrow \quad Adm_{x_1} \subset Adm_{x_2}.$$

A new controller can be computed using the function R. The possible initial states are the maximal states (for the relation R) amongst all the solutions of the equation

$$Q_0(X) = 0 \text{ and } C_0(X) = 0.$$

Similarly, the choice of commands for a pair (x, y) is reduced such that the successor states are maximal for R.

This control strategy reduces the possible initial states and the possible transitions of the system. Although it preserves the invariance of G_1, it could affect the reachability of G_2. However, in the case of the cat and mouse problem, the new controller still ensure both objectives. A state x is composed of two elements, one is the occupation of the room, the other is the position of the doors, and the control strategy does not reduce the reachability of the rooms.

6 Conclusion

An application of algebraic techniques to the control of discrete event systems has been presented. The approach is based on the model of dynamical systems over a finite field. A high-level language – SIGNAL – facilitates the description of such systems. Control objectives are expressed in terms of geometric properties of sets of states, algebraic manipulations based on ideals and principal generators allow us to synthesize control equations.

Several advantages arise from the use of a high-level language, especially when the systems to model are complex. The structuring facilities of SIGNAL permits a modular descriptions of DES and, with a little effort, the systems can be simulated. Another interesting point would be to translate back controllers to SIGNAL processes, thus allowing easy implementation and simulation.

A major problem of the classical approach to modelling DES, based on formal languages and automata, is the combinatorial explosion experienced when composing automata. By using equational modelling, we hope to overcome, to a certain extent, this problem. The example showed that another limitation of the language models, due to the interleaving of events (the animals cannot move simultaneoulsy), could also be overcome.

The controllers presented in this paper are static. More general control systems – polynomial dynamic systems which possess their own state variables – are currently being investigated. They are necessary when control objectives are more complex than in the example. Other control aspects such as observability or decentralized control are also a subject for future studies.

References

[1] A. Benveniste and P. Le Guernic. Hybrid dynamical systems theory and the SIGNAL language. *IEEE Transactions on Automatic Control*, 35(5):535–546, May 1990.

[2] P. Bournai and P. Le Guernic. Un environnement graphique pour le langage SIGNAL. Technical Report PI 741, IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France, 1993.

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

[4] R. E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys*, 24(3):293–318, September 1992.

[5] B. Buchberger. Some properties of Gröbner bases for polynomial ideals. *ACM SIGSAM bull.*, 10(4):19–24, 1976.

[6] B. Dutertre, M. Le Borgne, A. Benveniste, and P. Le Guernic. Discrete event systems and synchronous languages: an example. In *Proc. of IFAC93 conference*, Sydney, 1993. To appear.

[7] M. Le Borgne, A. Benveniste, and P. Le Guernic. Dynamical systems over galois fields and deds control problems. In *Proc. 30th IEEE Conf. on Decision and Control*, volume 3, pages 1505–1509. IEEE Control System Society, 1991.

[8] M. Le Borgne, B. Dutertre, A. Benveniste, and P. Le Guernic. Dynamical systems over galois fields. In *Proc. ECC93*, Groningen, 1993.

[9] P. Le Guernic, T. Gautier, M. Le Borgne, and C. Le Maire. Programming real-time applications with SIGNAL. *Proceedings of the IEEE*, 79(9):1321–1336, September 1991.

[10] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, 25(1):206–230, January 1987.

[11] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, January 1989.

[12] W. M. Wonham and P. J. Ramadge. On the supremal controllable sublanguage of a given language. *SIAM J. Control and Optimization*, 25(3):637–659, May 1987.

Unité de recherche INRIA Lorraine, Technôpole de Nancy-Brabois, Campus scientifique,
615 rue de Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399