Automatic Synthesis for Simultaneous Supervision and Control – a First Example

J. Gunnarsson, J. Plantin
Department of Electrical Engineering
Linköping University
S-581 83 Linköping, Sweden
E-mail: johan@isy.liu.se, plantin@isy.liu.se

January 10, 1996

**Submitted to the 1995 American Control Conference.
Pages: 3157–3162**

Abstract

In industry, supervision and control have often been regarded as separate problems, where the supervision problem has been solved in an ad hoc manner. It would be desirable to handle both control and supervision aspects in the design process.

In this work we investigate whether it is possible to synthesize the control law for a discrete event dynamic system, using a polynomial representation of the system and controller. By working through a relatively simple example we propose a synthesis method based on the polynomial representation and repeated Gröbner basis computations.

The supervisor objective is given in terms of forbidden states and the controller should actively be able to avoid these. We reduce the manifold of solutions by imposing a weighting function on the state space and by proposing a priority among the actuators. In this way we improve the computational performance radically and we can also control the system towards a desired state, which is our control objective.

The computational load lies in the design phase where we use Gröbner bases. The resulting controller description is suited for real time applications since the computations needed are evaluations of a limited number of polynomials.

During the design phase we also obtain a polynomial describing where in the state space the computed control law is valid. This makes it possible to examine controllability and supervisability of the system.

1 Introduction

In industry, control and supervision has often been regarded as separate problems, where the supervision has been dealt with in an ad hoc manner. There are several results in the area of supervisory control, initiated by Ramadge & Wonham [5]. However, it would be desirable to find a design method, formally

handling both control and supervision aspects in a discrete setting. By supervision we mean avoiding forbidden states and the control problem is that of achieving a desired behavior among all allowed behaviors.

We propose a method where we use polynomials over a quotient ring to represent the system and the controller. By working through an example we investigate whether it is possible to automatically synthesize the control law for a discrete event system using this representation. This paper describes the process of controller design for the example. However, the method is not specific for this example and general conclusions are drawn about the method used. For a full report, see [4].

After a short introduction to the polynomial representation we describe the model of the tank and the criteria used in the controller design. In Section 5 we deal with the computation of the control law, using Gröbner bases. In the two final sections we analyze the resulting controller and discuss the results achieved. Appendix A contains further details on the polynomial representation.

2 Polynomial Representation of DEDS

We represent a *discrete event dynamic system* (DEDS) by the mathematical description

$$\begin{aligned} f(x^+, x, u) &= 0 \\ h(y, x, u) &= 0 \end{aligned} \tag{1}$$

which denotes a set of relations between inputs u , outputs y , states x and the next state value x^+ . In a DEDS all variables are discrete and the system is event driven, i.e. “real time” is not included in the description.

Every variable in a model of a DEDS has some finite field, $\mathbb{F}_q$, as its domain. All functions $f : \mathbb{F}_q^n \rightarrow \mathbb{F}_q$ can be represented by a polynomial in $\mathbb{F}_q[X]$, i.e. the polynomial ring with coefficients from $\mathbb{F}_q$ and variables $x_1, \dots, x_n$. However, one function can be represented by an infinite number of polynomials in $\mathbb{F}_q[X]$. To get a one-to-one correspondence between functions and polynomials, Germundsson [2] introduces the quotient polynomial ring

$$R_q[X] = \mathbb{F}_q[x_1, \dots, x_n] / \langle x_1^q - x_1, \dots, x_n^q - x_n \rangle \tag{2}$$

The degree and the “length” of polynomials in $R_q[X]$ have an upper limit which gives complexity advantages.

We can use polynomials from this ring to represent (1). Other formalisms for discrete event systems can be transformed to and from this polynomial representation. This class of models includes any finite state based representation, which in its turn includes the regular language based models proposed by Ramadge & Wonham [5]. For more details on the polynomial representation, see Appendix A, [2], [3] and [4].

3 Modeling the Water Tank

We apply our design ideas to an example system: a water tank (see figure 1). This is a test example which is not entirely trivial but where it is possible to analyze the computed control laws by hand. Even if this is a simple system, important features in the process of control design are clearly visible.

Figure 1: The water tank.

Variable	Domain	Quantity
x	$\mathbb{F}_7$	Water level
u_1	$\mathbb{F}_2$	Pump signal
$u_2 - u_4$	$\mathbb{F}_2$	Valve signals
ϕ	$\mathbb{F}_3$	Net flow
w	$\mathbb{F}_2$	Outflow disturbance
d	$\mathbb{F}_3$	Pump failure

Table 1: The variables and their domains.

3.1 System Description

We want to control the water level, x , in the tank. The tank has one inlet and two outlets, which are controlled by valves that are either on or off. The inlet is supplied by a pump that is either on or off and in one of the outlets there is a measurable but uncontrollable flow out of the tank. Apart from the normal control of the tank we also want to handle a possible pump failure at the inlet. The failure is modeled as a three-valued disturbance, d , acting on the pump. When $d = 0$ the pump is stuck and when $d = 2$ the pump is running, regardless of the value of u_1 . Only when $d = 1$, the pump obeys the control signal, u_1 . The disturbance, d , is assumed to be measurable. For later use we also define the net flow in the tank, ϕ . All variables and their corresponding domains are shown in table 1.

The supervisor objective is to prevent the tank from drying up or overflowing. In addition we have a control objective of keeping the level as close to the middle of the tank as possible. We want to compute a control law that uses only the pump as long as that is sufficient. The valves could be regarded as “emergency actuators”.

3.2 Deriving the Model

It is essential to divide the system into subsystems, if possible. In this case we can use the net flow ϕ to write the tank model as

$$\begin{aligned} x^+ &= f_1(x, \phi) \\ \phi &= f_2(u_1, u_2, u_3, u_4, w, d) \end{aligned} \quad (3)$$

where x^+ denotes the next value of the level. By doing this we can synthesize smaller parts and thereby reduce complexity.

Since x takes values between zero and six we have chosen to represent all signals by polynomials in $R_7[Z]$, where Z is the variable set $\{x, \phi, u_1, u_2, u_3, u_4, w, d\}$. Some of the variables are only two- or three-valued but these can be embedded in $R_7[Z]$, using a method explained in Appendix, A.4. The set of polynomials describing the embedding will according to definition A.5, be denoted Λ_7 .

3.3 Polynomial description of the system

Using equation (19) and (20) in Appendix A we get a polynomial representation of the system (3). An indication of the complexity and the structure of the polynomials is given by the following short hand notation

$$\begin{aligned} f_1 &= 6\phi + \{21 \text{ terms}\} + 6\phi^3 x^6 \\ f_2 &= 1 + 3du_2 + \{9 \text{ terms}\} + d^2 u_1 u_2 u_3 u_4 w \end{aligned} \quad (4)$$

where $f_1 \in R_7[x, \phi]$ and $f_2 \in R_7[u_1, u_2, u_3, u_4, w, d]$. Together these polynomials tell us how x^+ depends on the control signals and the disturbances.

In (4) we have a mathematical model of the water tank. The question now is how to specify the control/supervisor objectives and how to compute the desired control law.

4 Design Criteria

The supervisor objective is to avoid $x = 0$ and $x = 6$. Given the present level and the disturbances w and d , we want to find a control law that guarantees that we never reach the forbidden levels, specified by the *logical polynomial* (see Appendix, A.1)

$$p(x) = 1 + x + 6x^2 + x^3 + 6x^4 + x^5 + 5x^6 \quad (5)$$

This polynomial is equal to zero for all values of x , except $x = 0$ and $x = 6$.

By formulating the supervisor objective in this way we get a manifold of solutions. It turns out that it is computationally inefficient to compute the entire manifold. Even if it would be possible to express it with a polynomial relation, this would be of little practical use since we want the control laws to be functions of the measured variables. One possibility is to reduce the manifold of solutions by imposing more requirements on the system. A reasonable approach to do this is to weight the values of x . A control objective which gives us a possibility to specify desired states could then be formulated as reducing the weight of the next value of x compared to the present value. If we can still

x	0	1	2	3	4	5	6
$J(x)$	3	2	1	0	1	2	3

Table 2: The weighting function, $J(x)$.

guarantee that the level never reaches $x = 0$ or $x = 6$, this is just one way of picking a single solution. If we want the level to tend to the middle of the tank, the weighting function can be chosen as in table 2 and it can be represented by a polynomial $J(x) \in R_7[x]$.

Use this polynomial to weight the new values of the state variable

$$J(x^+) = J(f_1(x, \phi)) \quad (6)$$

We want to find the value of ϕ that minimizes $J(x^+)$ with respect to the ordering $0 < 1 < \dots < 6$, i.e.

$$\phi = \arg \min_{\phi \in \mathbb{F}_3} J(f_1(x, \phi)) \quad (7)$$

Now let the polynomial $p_1(x, \phi, m)$ be defined as

$$p_1(x, \phi, m) = J(f_1(x, \phi)) + m - J(x) \quad (8)$$

If there exists a $\phi \in \mathbb{F}_3$ that decreases the weight of the state with m steps, that value of ϕ is defined as the solution to

$$p_1(x, \phi, m) = 0 \quad (9)$$

This solution is valid only in the case when $m < q - J_{max}$, where $J_{max} = \arg \max_{x \in \mathbb{F}_7} J(x)$ and m , q and J_{max} and the max - and $<$ -operations are regarded as in $\mathbb{N}$. The ordering will otherwise be destroyed, and there will be false solutions.

By this construction we will find a solution, ϕ , if one exists. If we can express ϕ explicitly, the values of the actuator signals are then the solutions to $p_2 = 0$ where p_2 is defined as

$$p_2(\phi, u_1, u_2, u_3, u_4, w, d) = f_2 - \phi \quad (10)$$

5 Computation of the Control Laws

The two polynomials p_1 and p_2 express relations between the variables, implicitly describing the control laws. We want to find explicit control laws

$$u_i = K_{u_i}(x, w, d), \quad i = 1, \dots, 4 \quad (11)$$

where the u_i are functions of x , w and d .

In order to do this, we first need to express ϕ as a function of x , using the polynomial p_1 . One way of doing that is to compute a Gröbner basis (see Appendix, A.2) with lexical ordering, where we rank ϕ the highest and use m as a parameter. The Gröbner basis will then contain a polynomial that is *linear* in ϕ , see Appendix, A.3.

5.1 Computing the Desired Net Flow

Compute the Gröbner basis

$$\mathcal{GB}_a = GB_7(\{p_1(x, \phi, m), \lambda_7^3(\phi)\} \mid_{m=1}) \quad (12)$$

with lexicographic order $\phi > x$. This Gröbner basis contains a number of polynomials

$$\mathcal{GB}_a = \{g_1(\phi, x), g_2(x), \dots, g_i(x)\} \quad (13)$$

By the construction of p_1 in (8) we have no ambiguity for $m = 1$, i.e. ϕ is a linear variable w.r.t. $\langle p_1, \lambda_7 \rangle$ and the first polynomial in $\mathcal{GB}_a$ can be written

$$\phi - k_{\phi_a}(x) \quad (14)$$

The other polynomials in $\mathcal{GB}_a$ define where this solution is valid. Let the logical polynomial $v_{\phi_a}(x)$ denote the *valid area* for $k_{\phi_a}(x)$. We compute $v_{\phi_a}(x)$ as

$$v_{\phi_a}(x) = g_2(x) \wedge g_3(x) \wedge \dots \wedge g_i(x) \quad (15)$$

In this case it turns out that the solution is valid for all x , except $x = 3$. This is due to the fact that we have required the weight of the state to decrease by one and this is not possible when $x = 3$. This means that we have to search for another solution in the case when $v_{\phi_a} \neq 0$. Compute

$$\mathcal{GB}_b = GB_7(\{p_1(x, \phi, m), \lambda_7(\phi), \neg v_{\phi_a}(x)\} \mid_{m=0}) \quad (16)$$

to get

$$\phi - k_{\phi_b}(x) \quad (17)$$

with corresponding polynomial $v_{\phi_b}(x)$, denoting the valid area for $k_{\phi_b}(x)$ (disjoint from $v_{\phi_a}(x)$).

We can now compute ϕ as a *function* of x as

$$\phi = K_\phi(x) = k_{\phi_a}(x) \neg v_{\phi_a}(x) + k_{\phi_b}(x) \neg v_{\phi_b}(x) \quad (18)$$

with valid area $V_\phi = v_{\phi_a} \vee v_{\phi_b}$.

It turns out that $V_\phi \equiv \text{TRUE}$. Therefore the valid area covers all values of x and there is a solution in ϕ for every $x \in \mathbb{F}_7$. If we are able to compute the u_i as functions of ϕ , w and d , using p_2 , we can express the actuator signals in the desired form of equation (11) by using equation (18).

5.2 Finding an Unambiguous Control Law

In order to compute the u_i as functions of ϕ , w and d we need the solution to equation (10) to be unambiguous. Obviously, this is not the case with the water tank. There are several control actions that, given values of w and d , give the same net flow ϕ . Therefore we need to make a priority among the actuators. In this case it is natural to try to control the tank by using the pump, if possible, and only use the valves if necessary.

Due to physical causes, there is a possibility that some of the actuators can not affect the behavior of the system. In terms of the Gröbner basis this means

that the value of the corresponding variable will be unspecified. In order to guarantee that there is only one solution to (10) in these cases, we need extra constraints in the Gröbner basis computations.

In the case of the water tank we have, for example, that when $d = 0$ or $d = 2$ we cannot use u_1 for control, so for these values of d we assign u_1 its default value. There are three similar cases to account for and all these have to be specified by extra constraints. For details on this, see [4].

5.3 Computation of the Actuator Signals

The Gröbner basis computations will not be described in detail here. They are performed in a manner similar to that described in Section 5.1, with the extra constraints included in the polynomial set.

We start by computing a Gröbner basis for the case when we only use the pump for control, setting the other actuators to default values. In some cases the choice of default values is quite natural, but we can of course define them the way we want. Here we let the default values of the valves be as in normal operation (the pump working), that is: u_2 open, u_3 closed and u_4 open. The default value for the pump is chosen to be zero.

The first Gröbner basis computation gives us a valid area, v_1 , for the first control law. We then continue with the computation of a second Gröbner basis in the area $\neg v_1$, using u_1 and u_2 for the control (letting u_3 and u_4 take default values). In this way we finally get four expressions for each u_i , valid in four disjoint areas and these can be combined into one control law, K_{u_i} , for each actuator. These control laws will have the total valid area $V_u = v_1 \vee v_2 \vee v_3 \vee v_4$, where v_i denotes the valid area in case i .

We have now computed all the control laws and by substitution we can obtain the desired form (11). However, to reduce complexity, we avoid the substitution as long as possible.

6 Analysis of the Design

After the design phase we need to make sure that both the control objectives and the supervisor objectives are achieved. We will focus on

- *Controllability.* Is it possible to fulfill the control objectives by actions on the inputs?
- *Supervisability.* Given an initial state which is allowed, can we guarantee that we will never reach a forbidden state?

6.1 Controllability

The logical polynomial V_u represents the values of the variables ϕ, w, d , for which we have an appropriate control law. If this polynomial is **FALSE** for some values of ϕ, w, d it means that the system is not controllable.

Evaluating V_u for all values of ϕ, w and d , we find that V_u is **FALSE** for the case $\{\phi = 2, d = 0\}$ (independent of w). This corresponds to the case when there is a pump failure and, at the same time, the level in the tank is required

to increase. Looking at the physical system in figure 1 there is no doubt that it is impossible to fulfill that requirement.

The **FALSE** value forces us to do a re-engineering of either the system itself (adding some actuators) or the objectives stated for the control design. In this example we choose the latter.

6.2 The Closed Loop System

From f_1, f_2 and the K_{u_i} we can derive a closed loop description. The closed loop system must be properly defined, and to fulfill this the control laws must also be defined for the non valid area. Let us do that in two ways:

1. Let the control laws take their default values also outside the valid area. (A simple and intuitive strategy when the physical causes of the non valid area are unknown.)
2. It seems smarter and more careful to let the valve u_4 be closed. (To be sure that the level is not decreasing.)

These two ways of handling the control laws will generate the closed loop descriptions $G_{c1}(x, w, d)$ and $G_{c2}(x, w, d)$ respectively.

The computation of the closed loop system $x^+ = G_c(x, w, d)$ is straightforward using simple substitutions. It turns out that G_{c2} is independent of w which means that the control laws eliminates the influence of the disturbance signal w in that case.

The polynomials are hard to interpret as they are. One way of gaining insight would be to substitute all possible values of the variables and derive a table of the closed loop system. For larger systems such a method would be of little use, and as we will see below it is possible to analyze the closed loop behavior, using the polynomial description.

6.3 Supervisability

One way of checking supervisability is to test *backward reachability* for our closed loop systems. If we from all forbidden states move backwards one step with all possible input signals, and the states reached are all forbidden, we know that it is impossible to reach a non forbidden state in any number of backward steps. Thus the forbidden states are not forward reachable from a non forbidden state. If the polynomial describing the allowed states is simpler than $p(x)$, it would of course be better to test forward reachability from the allowed states instead.

We use Gröbner bases to compute the backward reachable states. These states will be described by a polynomial in x .

For G_{c1} we get the polynomial $3x + 4x^2$ which has $\{x = 0, x = 1\}$ as roots, and we see that this system does not fulfill our objectives, since $x = 1$ is in the non forbidden area. For G_{c2} we have $x = 0$ as the only root showing that no non forbidden states are reached. Therefore G_{c2} is a robust design for this model of disturbances and forbidden states. The supervisability analysis shows that it is important to deal appropriately with the control behavior outside the valid area.

7 Conclusions

The example has shown that polynomial methods can be used to formally handle both control and supervision aspects in a discrete setting. Even if some steps were manipulated by hand in this example, the results indicate that the design process could be stated as an algorithm. This means that it would be possible to generate control code automatically, given a DEDS model and control/supervisor objectives.

7.1 The Design Method

The design method used produces control laws that are *functions* of the measurable variables. In order to achieve this we must have a way to choose one of, possibly, many control laws fulfilling the specification. Using ideas from LQ-design in the continuous domain, we propose the use of a weighting function, where the forbidden states are given the highest weight. This method allows assignment of weights to groups of states and it also opens the possibility to specifications of another kind than “forbidden states”.

We also propose a priority among the actuators. This priority sometimes is a natural priority, otherwise we can regard it as a design criteria. It could for example be used to distinguish emergency actions from normal control, i.e. handle supervision aspects.

To further reduce the number of possible control laws we impose constraints that eliminate all control actions that have no effect on the system. Since these constraints seems to be physically intuitive it is possible that they can be generated algebraically in the modeling process.

The functions representing the control laws can be analyzed symbolically. Another advantage is that we get a valid area for our control laws, which makes it possible to examine the controllability of the design. If there is a need for re-engineering the controllability analysis indicates what parts need to be re-engineered.

7.2 Computational Aspects

The key issue in the design process is to reduce the complexity as much as possible. This is done by careful modeling and by imposing requirements on the behavior of the system that reduces the number of possible control laws.

In the modeling it is very important to divide the system into subsystems with as few interacting variables as possible. An open question is how the choice of the finite field affects the complexity of the model.

The computations described in this paper are done by a straightforward implementation of Buchberger’s algorithm [1] in *Mathematica*. The seven Gröbner bases needed in this example were computed in a total time of half an hour on an LX SparcStation, which is acceptable considering that the algorithm is designed for general polynomials. Since the problem domain is discrete there are a lot of optimizing possibilities to consider, e.g. extending existing algorithms for boolean equations.

An important aspect is that the control laws computed, easily can be translated into executable code. This means that once the system description and

the design criteria are decided, the controller code could be generated automatically. Since the computation of the actuator signals only consists of evaluating a polynomial for the measured values of the system variables, the computation can be made very fast and is therefore suited for real time controllers.

8 Acknowledgment

This work was supported by the Swedish Research Council for Engineering Sciences (TFR) and the Swedish National Board for Industrial and Technical Development (NUTEK), which is gratefully acknowledged.

A Polynomial Representation

This appendix contains mathematical details concerning the polynomial representation used in this paper. See also [2], [3] and [4].

A.1 Representing Functions and Logical Expressions with Polynomials

As shown in [2], let $f : \mathbb{F}_q^n \rightarrow \mathbb{F}_q$ be any function. The corresponding polynomial $f_p(X) \in R_q[X]$ is computed as

$$f_p(X) = \sum_{\xi \in \mathbb{F}_q^n} L_\xi(X) f(\xi) \quad (19)$$

where $L_\xi(X) = L_{\xi_1}(x_1) \cdots L_{\xi_n}(x_n) \in R_q[X]$ and

$$L_{\xi_i}(x_i) = \frac{\prod_{\nu \in \mathbb{F}_q \setminus \{\xi_i\}} (x_i - \nu)}{\prod_{\nu \in \mathbb{F}_q \setminus \{\xi_i\}} (\xi_i - \nu)} = \begin{cases} 1, & x_i = \xi_i \\ 0, & x_i \neq \xi_i \end{cases} \quad (20)$$

is the Lagrange interpolating polynomial. We then have $f(\xi) = f_p(\xi)$ for all $\xi \in \mathbb{F}_q^n$.

The formalism can also be used to represent logical conditions, i.e. functions $b : \mathbb{F}_q^n \rightarrow \{\text{TRUE}, \text{FALSE}\}$.

Definition A.1 Let a *logical polynomial*, $b(X) \in R_q[X]$, be a polynomial where the values are interpreted as logical conditions.

$$\begin{aligned} b(X) = 0 &\Rightarrow \text{TRUE} \\ b(X) \neq 0 &\Rightarrow \text{FALSE} \end{aligned} \quad (21)$$

□

If $a(X), b(X) \in R_q[X]$ represents two logical expressions, the result of AND-, OR- and NOT-operations can be computed by algebraically manipulating the polynomials a and b . This gives a natural extension of the logical operations to ideals in $R_q[X]$. See [2] for further details.

A.2 Gröbner Bases in $R_q[X]$

For each ideal in a polynomial ring there are many possible sets of polynomials that generate the ideal. Auto-reduced Gröbner bases¹ is a canonical form for ideals. They can be regarded as the “simplest” representation of an ideal w.r.t. some *term ordering*. The Gröbner bases are in a sense a nonlinear generalization of Gaussian elimination. With that interpretation the term ordering tells which of the variables will be eliminated first.

Gröbner bases are used and defined in $R_q[X]$ in analogy with the general polynomial ring $k[X]$, except that in every arithmetic operation on the polynomials in $R_q[X]$,

- degrees higher or equal than q are reduced as $x^q \rightarrow x$.
- coefficients only take values in $\mathbb{F}_q$.

The ring $R_q[X]$ is a quotient ring. Therefore the term ordering is not well defined since $\deg(fg) = \deg(f) + \deg(g)$ does not always hold. To deal with this problem formally, the Gröbner basis for an ideal I is computed in the *free* polynomial ring $\mathbb{F}_q[X]$ with the relations $x_1^q - x_1, \dots, x_n^q - x_n$ included in the set of generators for the ideal I .

Definition A.2 The Gröbner basis of the ideal $F = \langle f_1, f_2, \dots, f_n \rangle \in R_q[X]$ for some given term ordering is denoted

$$GB_q(f_1, f_2, \dots, f_n) \quad (22)$$

□

A.3 Linear Variables

When we compute Gröbner bases we get a set of polynomials relating the variables to each other. When we use this for control design we are particularly interested in whether a variable can be computed as a *function* of the other variables in the system description. This leads to the definition of *linear variables*.

Definition A.3 Given an ideal $I \subseteq R_q[y, x_1, \dots, x_n]$ and the variety $(y, x_1, \dots, x_n) \in V(I)$. If for each value of the variables $x_1, \dots, x_n$ in $V(I)$ there exists only one value of y , then the variable y is *linear* w.r.t I . □

As stated in the following theorem, Gröbner basis computations can be used to determine if a variable is linear and the corresponding function.

Theorem A.1 Given an ideal $I = \langle f_1, \dots, f_l \rangle \subseteq R_q[y_1, \dots, y_n, x_1, \dots, x_m]$. The variables $y_1, \dots, y_n$ are linear w.r.t I iff the auto-reduced Gröbner basis G of I with lex-ordering $y > x$ has the form

$$G = \{y_1 - h_1, \dots, y_n - h_n, v_1, v_2, \dots\} \quad (23)$$

¹An excellent introduction to Gröbner bases for ideals in a general polynomial ring $k[X]$ is [1].

where the polynomials $h_i, v_j \in R_q[x_1, \dots, x_m]$. □

Proof: See [4]. □

A.4 Mixing Polynomials from Different Rings

Consider a polynomial $p(X) \in R_q[X]$ where a variable $x_i \in X$ will not take values outside the interval $\{0, \dots, r_i - 1\}$, where $r_i < q$. We say that for some values of x_i there are “don’t cares”. This will help us to simplify² the polynomial $p(X)$ by finding the simplest polynomial preserving the values of $p(X)$ for all values of x_i that are not “don’t care”. The variable x_i can be considered to belong to the field $\mathbb{F}_{r_i}$.

Let us compute a logical polynomial that is TRUE only for those values of x_i that are not “don’t care”. We make the following definition.

Definition A.4 Let $\lambda_q^{r_i}(x_i) \in \mathbb{F}_q[X]$, $x_i \in X$, $r_i < q$ be a polynomial such that

$$\lambda_q^{r_i}(x_i) = \begin{cases} \text{TRUE} & , x_i = 0, \dots, r_i - 1 \\ \text{FALSE} & , x_i = r_i, \dots, q - 1 \end{cases} \quad (24)$$

□

The logical polynomial $\lambda_q^{r_i}(x_i)$ can then be used to embed variables $x_i \in \mathbb{F}_{r_i}$ in the ring $R_q[X]$.

To collect all λ -polynomials corresponding to variables containing a “don’t care” value we make the following definition.

Definition A.5 Let Λ_q denote the set containing all $\lambda_q^{r_i}(x_i)$ corresponding to all $x_i \in \mathbb{F}_{r_i}$ where $r_i < q$. □

To compute the simplified polynomial, reduce $p(X)$ w.r.t $GB_q(\Lambda_q)$.

References

- [1] D. Cox, J. Little, and D. O’Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer-Verlag, 1992.
- [2] R. Germundsson. Symbolic and algebraic methods for modeling, analysis, design and implementation of discrete systems. Technical Report LiTH-ISY-R-1477, Dept. of Electrical Engineering, Linköping University, S-581 83 Linköping, Sweden, May 1993. Lecture notes for a seminar at ITM, May 14, 1993.
- [3] R. Germundsson. *Symbolic Systems - Theory, Computation and Applications*. PhD thesis, Linköping University, September 1995.

²Here simplify means reducing the degree of the polynomial and reducing the number of monomials.

- [4] J. Gunnarsson and J. Plantin. Control law synthesis for a discrete event system. Technical Report LiTH-ISY-R-1651, Dept. of Electrical Engineering, Linköping University, S-581 83 Linköping, Sweden, August 1994. Available electronically at <http://joakim.isy.liu.se/AutomaticControl/Reports/index.html>.
- [5] P. J. Ramadge and W. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, January 1989.