WORD-LEVEL TRAVERSAL OF FINITE STATE MACHINES USING ALGEBRAIC GEOMETRY

Xiaojun Sun*, Priyank Kalla*, Florian Enescu†
*Electrical & Computer Engineering, University of Utah, Salt Lake City, UT
†Mathematics & Statistics, Georgia State University, Atlanta, GA
{xiaojuns, kalla}@ece.utah.edu, fenescu@gsu.edu

***Abstract*—Reachability analysis is a tool for formal equivalence and model checking of sequential circuits. Conventional techniques are mostly bit-level, in that the reachable states, transition relations and property predicates are all represented using Boolean variables and functions. The problem suffers from exponential space and time complexities; therefore, some form of abstraction is desirable. This paper introduces a new concept of implicit state enumeration of finite state machines (FSMs) performed at the word-level. Using algebraic geometry, we show that the state-space of a sequential circuit can be encoded, canonically, as the zeros of a word-level polynomial $\mathcal{F}(S)$ over the Galois field $\mathbb{F}_{2^k}$, where $S = \{s_0, ..., s_{k-1}\}$ is the word-level representation of a k-bit state register. Subsequently, concepts of elimination ideals and Gröbner bases can be employed for FSM traversal. The paper describes the complete theory of word-level FSM traversal and demonstrates the feasibility of the approach with experiments over a set of sequential circuit benchmarks.**

I. INTRODUCTION

In verification of sequential circuits, designers face problems of errors in specification models and implementations. These errors are often modeled as "bad states" in the underlying finite state machine (FSM) of the sequential circuit. Existence of errors can be identified by detecting whether these bad states are reachable from certain initial states. This requires some form of *reachability analysis* [1]; it is required or performed by various techniques such as (symbolic) simulation [2], model checking [3], property directed reachability (PDR) [4], etc. Contemporary techniques are mostly bit-level, in that the transition relations, sets of states, property predicates are mostly given in terms of bit-level or Boolean variables. This often makes the representations too large to handle.

This paper presents a new approach to sequential verification that performs reachability analysis of an arbitrary FSM while operating *implicitly at the word-level*. The approach is based on concepts from computational algebraic geometry. Essentially, the transition relation of the FSM and the set of initial states is represented by way of multivariate polynomials over the Galois field $\mathbb{F}_{2^k}$, where k corresponds to the number of state elements, or state register bits $\{s_0, \dots, s_{k-1}\}$. Subsequently, using concepts from algebraic geometry, such as projection of varieties, elimination ideals, quotients of ideals and Gröbner Bases (GB), a word-level polynomial $\mathcal{F}(S)$ is computed, where $S = \{s_0, \dots, s_{k-1}\}$ is a word-level representation of the k state register bits. The set of reachable states is encoded as the zeros of this word-level polynomial.

This approach can be considered as a way of *word-level abstraction of the state space* of the FSM. Abstraction plays a key role in addressing the scalability issues with sequential verification [5], often by reducing the FSM's state-space to analyze. It is usually done by combining sets of states with similar properties. In this work, the k bit-level state variables are bundled together as one word-level variable, so the reachable state space can be effectively encoded by a word-level constraint expression. Such a word-level abstraction may help overcome the state explosion problem.

Contribution: Any Boolean function over k-bit vectors $f : \mathbb{B}^k \rightarrow \mathbb{B}^k$ can be uniquely mapped to a polynomial function over k-bit words in the Galois field $\mathbb{F}_{2^k}$, i.e. as polynomial functions $f : \mathbb{F}_{2^k} \rightarrow \mathbb{F}_{2^k}$ [6]. Therefore, the transition relations and sets of states (Boolean encoding) can be represented as (word-level) elements over $\mathbb{F}_{2^k}$. FSM traversal requires operations such as image computations, set intersections, unions and complements, equivalence tests, etc. All these operations can be implemented using computational algebraic geometry over Galois fields. Moreover, since $\mathbb{F}_{2^k} \supset \mathbb{F}_2$, it provides a *unified framework* for both bit-level ($\mathbb{B} \equiv \mathbb{F}_2$) and word-level ($\mathbb{F}_{2^k}$) constraints. Furthermore, values of state variables can be represented as solutions (varieties) to a finite set of polynomials (ideals). GB techniques can be subsequently applied to transform such polynomial systems into a *canonical form*, facilitating least fixed-point test (convergence). In this work, all these concepts are applied in a synergistic fashion for word-level reachability analysis. The approach can be generalized to FSMs with different number of inputs and state register bits, as any Boolean function $f : \mathbb{B}^n \rightarrow \mathbb{B}^m$ can be modeled as polynomial function $f : \mathbb{F}_{2^k} \rightarrow \mathbb{F}_{2^k}$, where $k = LCM(n,m)$.

Previous Work: The work of [7] derives a word-level polynomial representation $Z = \mathcal{F}(A)$ for a combinational circuit C with k-bit inputs $A = (a_0, \dots, a_{k-1})$ and outputs $Z = (z_0, \dots, z_{k-1})$. They make use of a reduced GB computation of the polynomials of the circuit to derive the word-level polynomial. We draw inspirations from [7] and further extend the results with fundamental algebraic geometry concepts to perform word-level FSM traversal. In our previous work [8], GB techniques have been applied to verify sequential normal-basis Galois field multipliers. This technique is inapplicable to reachability analysis of arbitrary FSMs. The use of algebraic geometry has been proposed for model checking [9] [10]; however, these approaches are a straight-forward application of *bit-level* Boolean Gröbner basis engines in lieu of BDDs or SAT solvers. In contrast, ours is a truly word-level and geometric approach to reachability analysis.

Paper Organization: The paper is organized as follows. The following section covers preliminary concepts. Section III describes the theory of word-level reachability computations and presents a complete algorithm for word-level FSM traversal based on various algebraic geometry concepts. Section IV

This work is sponsored in part by NSF grants CCF-1320335, CCF-1320385 and CCF-1619370.

978-1-5090-4270-8/16/$31.00©2016 IEEE

states some improvements which can be made on our approach to avoid high computational complexity. Section V describes our initial set of experiments that demonstrate the validity of our approach. The limitations of the approach are analyzed and the work currently underway is described. Finally, Section VI concludes the paper.

II. Preliminaries

Definition II.1. *A Mealy finite state machine is a n-tuple $\mathcal{M} = (\Sigma, O, S, S^0, \Delta, \Lambda)$ where: i) Σ is the input label, O is the output label; ii) S is the set of states, and $S^0 \subseteq S$ is the set of initial states; iii) $\Delta: \ S \times \Sigma \rightarrow S$ is the next state transition function; iv) $\Lambda: \ S \times \Sigma \rightarrow O$ is the output function.*

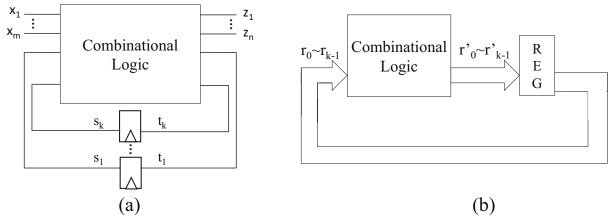


Fig. 1: FSM models of sequential circuits

Typical sequential circuits are depicted as in Fig.1(a). Primary inputs are denoted $x_1, \ldots, x_m \in \Sigma$, and the primary outputs as $z_1, \ldots, z_n \in O$. Signals $s_1, \ldots, s_k$ are the present state (PS) variables, $t_1, \ldots, t_k$ the next state (NS) variables. We can define two k-bit words denoting the PS/NS variables as there are k flip-flops in the datapath: $S = (s_1, \ldots, s_k)$, $T = (t_1, \ldots, t_k)$. Transition function at the bit-level is defined as $\Delta_i : t_i = \Delta_i(s_1, \ldots, s_k, x_1, \ldots, x_m)$.

When the primary outputs of the FSM only depend on the present states, i.e. $\Lambda: \ S \rightarrow O$, then the FSM is called a Moore machine. For example, in some cases k-bit arithmetic computations are implemented as Moore machines, where input operands are loaded into register files R and the FSM is executed for k clock cycles to obtain the result in R. We can simplify these to the model in Fig.1(b). While our approach works for both kinds of FSMs, we will consider only Mealy FSMs in the paper.

A. Algebraic geometry concepts

Let $\mathbb{B} \equiv \mathbb{F}_2 \equiv \{0,1\}$, $\mathbb{F}_q = \mathbb{F}_{2^k}$ the finite (Galois) field of $q = 2^k$ elements and $R = \mathbb{F}_{2^k}[x_1, \ldots, x_n]$ the polynomial ring in n variables with coefficients from $\mathbb{F}_{2^k}$. The field $\mathbb{F}_{2^k}$ is constructed as $\mathbb{F}_{2^k} = \mathbb{F}_2[X] \pmod{P(X)}$, where $P(X)$ is an irreducible polynomial over $\mathbb{F}_2$. Let α be a root of the irreducible polynomial $P(X)$, i.e. $P(\alpha) = 0$. Any element $A \in \mathbb{F}_{2^k}$ can be represented as $A = \sum_{i=0}^{k-1} a_i \alpha^i$, where $a_i \in \mathbb{F}_2$. The field $\mathbb{F}_{2^k}$ is therefore, a k-dimensional *extension* of the base field $\mathbb{F}_2$: so, $\mathbb{F}_{2^k} \supset \mathbb{F}_2$. Consequently, all operations of addition and multiplication in $\mathbb{F}_{2^k}$ are performed modulo the irreducible polynomial $P(\alpha)$ and coefficients are reduced modulo 2; so $-1 = +1$ in $\mathbb{F}_{2^k}$. Boolean operators in $\mathbb{B}$ are mapped to operators in $\mathbb{F}_2 \subset \mathbb{F}_{2^k}$ as:

$$a \wedge b \rightarrow a \cdot b; \quad a \oplus b \rightarrow a + b$$
$$\neg a \rightarrow 1 + a; \quad a \vee b \rightarrow a + b + a \cdot b$$

Using these mappings we can write Boolean functions as polynomials over $\mathbb{F}_{2^k}$. Every function $f : \mathbb{B}^k \rightarrow \mathbb{B}^k$ can be construed as a *polynomial function* over $f : \mathbb{F}_{2^k} \rightarrow \mathbb{F}_{2^k}$. Therefore, f can be represented by way of a unique, minimal, canonical polynomial $\mathcal{F}(X)$ [6] [7].

A polynomial $f = c_1X_1 + c_2X_2 + \cdots + c_tX_t$ is written as a finite sum of terms, where $c_1, \ldots, c_t$ are coefficients and $X_1, \ldots, X_t$ are monomials. A monomial ordering $X_1 > X_2 \cdots > X_t$ is imposed on the polynomials to process them systematically. Then, $LT(f) = c_1X_1, LM(f) = X_1$ denote the leading term and the leading monomial of f, respectively.

Ideals, Varieties and Gröbner Bases: Let $F = \{f_1, \ldots, f_s\}$ denote the given set of polynomials. An *ideal* $J \subseteq R$ generated by polynomials $f_1, \ldots, f_s \in R$ is:

$$J = \langle f_1, \ldots, f_s \rangle = \{\sum_{i=1}^{s} h_i \cdot f_i : \ h_i \in \mathbb{F}_q[x_1, \ldots, x_n]\}.$$

The polynomials $f_1, \ldots, f_s$ form the *basis* or the *generators* of J. We have to consider the set of solutions to the system of polynomials equations $f_1 = \cdots = f_s = 0$. The set of all solutions to a given system of polynomial equations $f_1 = \cdots = f_s = 0$ is called the *variety*, which depends upon the ideal $J = \langle f_1, \ldots, f_s \rangle$ generated by the polynomials. The variety is denoted by $V(J) = V(f_1, \ldots, f_s)$ and defined as:

$$V(J) = V(f_1, \ldots, f_s) = \{\mathbf{a} \in \mathbb{F}_q^n : \forall f \in J, f(\mathbf{a}) = 0\},$$

where $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{F}_q^n$ denotes a point in the affine space. We will denote by $\overline{V(J)}$ the complement of $V(J)$, where $\overline{V(J)} = (\mathbb{F}_{2^k})^n - V(J)$.

An ideal may have many generating sets; i.e. it is possible to have ideal $J = \langle f_1, \ldots, f_s \rangle = \langle h_1, \ldots, h_r \rangle = \cdots = \langle g_1, \ldots, g_t \rangle$ such that $V(f_1, \ldots, f_s) = V(h_1, \ldots, h_r) = \cdots = V(g_1, \ldots, g_t)$. A Gröbner basis (GB) is one such representation with many important properties, including the fact that it is a *canonical representation* of an ideal.

Definition II.2. [**Gröbner Basis**] *[11]: For a monomial ordering $>$, a set of non-zero polynomials $G = \{g_1, g_2, \cdots, g_t\}$ contained in an ideal J, is called a Gröbner basis for J iff $\forall f \in J$, $f \neq 0$ there exists $i \in \{1, \cdots, t\}$ such that $LM(g_i)$ divides $LM(f)$; i.e., $G = GB(J) \Leftrightarrow \ \forall f \in J : f \neq 0, \exists g_i \in G : LM(g_i) \mid LM(f)$.*

The famous Buchberger's algorithm [12], which is now textbook knowledge [11], is used to compute a GB. Operating on input $F = \{f_1, \ldots, f_s\}$, and subject to the imposed term order $>$, the algorithm computes $G = GB(J) = \{g_1, \ldots, g_t\}$. One application of GB is that it can work as a *quantification procedure* [13]. For this purpose, we introduce the concepts of *vanishing polynomials*, and *elimination ideals*.

Fermat's little theorem over $\mathbb{F}_q$: For any $\alpha \in \mathbb{F}_q, \alpha^q = \alpha$. Therefore, the polynomial $x^q - x$ vanishes $(= 0)$ over $\mathbb{F}_q$, and is called the vanishing polynomial of $\mathbb{F}_q$. Denote by $J_0 = \langle x_1^q - x_1, \ldots, x_n^q - x_n \rangle$ the ideal of all vanishing polynomials in R.

Gröbner bases can be used to *eliminate* (i.e. quantify) variables from an ideal. Given ideal $J = \langle f_1, \ldots, f_s \rangle \subset R$, the l^{th} elimination ideal J_l is the ideal of $\mathbb{F}_q[x_{l+1}, \ldots, x_n]$ defined by $J_l = J \cap \mathbb{F}_q[x_{l+1}, \ldots, x_n]$. Variable elimination can be achieved by computing a Gröbner basis of J w.r.t. elimination orders:

Theorem II.1. *(Elimination Theorem [11]) Let $J \subset \mathbb{F}_{2^k}[x_1, \ldots, x_n]$ be an ideal and let G be a Gröbner basis of J with respect to a lexicographic (LEX) ordering where $x_1 > x_2 > \cdots > x_n$. Then for every $0 \leq l \leq n$, the set $G_l = G \cap \mathbb{F}_{2^k}[x_{l+1}, \ldots, x_n]$ is a Gröbner basis of the l-th elimination ideal J_l.*

Example II.1. *Consider polynomials $f_1 : x^2 - y - z - 1$; $f_2 : x - y^2 - z - 1$; $f_3 : x - y - z^2 - 1$ and ideal $J = \langle f_1, f_2, f_3 \rangle \subset \mathbb{C}[x, y, z]$. Gröbner basis $G = GB(J)$ w.r.t. LEX term order equals to $g_1 : x - y - z^2 - 1$; $g_2 : y^2 - y - z^2 - z$; $g_3 : 2yz^2 - z^4 - z^2$; $g_4 : z^6 - 4z^4 - 4z^3 - z^2$. From observation, we find that the polynomial g_4 only contains variable z (x, y eliminated), and polynomials g_2, g_3, g_4 only contain variables y, z (x eliminated). According to theorem II.1, $G_1 = G \cap \mathbb{C}[y, z] = \{g_2, g_3, g_4\}$ is the Gröbner basis of the 1^{st} elimination ideal of J and $G_2 = G \cap \mathbb{C}[z] = \{g_4\}$ is the 2^{nd} elimination ideal of J, respectively.*

Application to word-level abstraction: Assume that we are given a combinational logic block C with input $A = (a_0, \ldots, a_{k-1})$ and output $Z = (z_0, \ldots, z_{k-1})$. We can describe this circuit with an elimination ideal $J + J_0$ in $\mathbb{F}_{2^k}$, where J is the ideal generated by the polynomials corresponding to the gates of the circuit and J_0 is the ideal of vanishing polynomials. The authors of [7] showed that for any combinational logic block, a canonical word-level polynomial representation can be derived through a GB computation with elimination orders. They used a lexicographic (elimination) term order with variables ordered as "bit-level variables of the circuit" $>$ "word-level output" $Z >$ "word-level input" A, and computed $G = GB(J + J_0)$ to derive the canonical word-level polynomial abstraction (cf. Theorem 4.2 in [7]).

Conceptually, we use a similar strategy. By representing the transition relations and sets of initial states using the ideal $J + J_0$, and computing a Gröbner basis w.r.t. an elimination order with *"bit-level variables"* $>$ *"present-state word"* $>$ *"next-state word"*, the reachable states can be encoded as polynomials in word-level state variables.

With this background, we now describe how to perform word-level FSM traversal.

III. FSM Reachability using Algebraic Geometry

We use symbolic state reachability with algebraic geometry concepts. It is an abstraction based on word operand definition of datapaths in circuits, and it can be applied to arbitrary FSMs by bundling a set of bit-level variables together as one or several word-level variables. The abstraction polynomial, encoding the reachable state space of the FSM, is obtained through computing a GB over $\mathbb{F}_{2^k}$ of the polynomials of the circuit using an elimination term order based on Theorem II.1.

Conceptually, the state-space of a FSM is traversed in a breadth-first manner, as shown in Algorithm 1. The algorithm operates on the FSM $\mathcal{M} = (\sum, O, S, S^0, \Delta, \Lambda)$ underlying a sequential circuit. In such cases, the transition function Δ and the initial states are represented and manipulated using Boolean representations such as BDDs or SAT solvers. The variables $from, reached, to, new$ represent characteristic functions of sets of states. Starting from the initial state $from^i = S^0$, the algorithm computes the states reachable in 1-step from $from^i$ in each iteration. In line 4 of algorithm 1, the *image computation* is used to compute the reachable states in every execution step.

The *transition function* Δ is given by Boolean equations of the flip-flops of the circuit: $t_i = \Delta_i(s, x)$, where t_i is a next state variable, s represents the present state variables and x represents the input variables. The *transition relation of the FSM* is then represented as:

$$T(s, x, t) = \prod_{i=1}^{n} (t_i \overline{\oplus} \Delta_i) \tag{1}$$

where n is the number of flip flops, and $\overline{\oplus}$ is XNOR operation. Let $from$ denote the set of initial states, then the image of the initial states, under the transition function Δ is finally computed as:

$$to = \text{Img}(\Delta, from) = \exists_s \; \exists_x \; [T(s, x, t) \cdot from] \tag{2}$$

Here, $\exists x(f)$ represents the *existential quantification of f w.r.t. variable x.*

ALGORITHM 1: BFS Traversal for FSM Reachability

```
Input: Transition functions $\Delta$, initial state $S^0$
1 $from^0 = reached = S^0$;
2 repeat
3     $i \leftarrow i + 1$;
4     $to^i \leftarrow \text{Img}(\Delta, from^{i-1})$;
5     $new^i \leftarrow to^i \cap \overline{reached}$;
6     $reached \leftarrow reached \cup new^i$;
7     $from^i \leftarrow new^i$;
8 until $new^i == 0$;
9 return $reached$
```

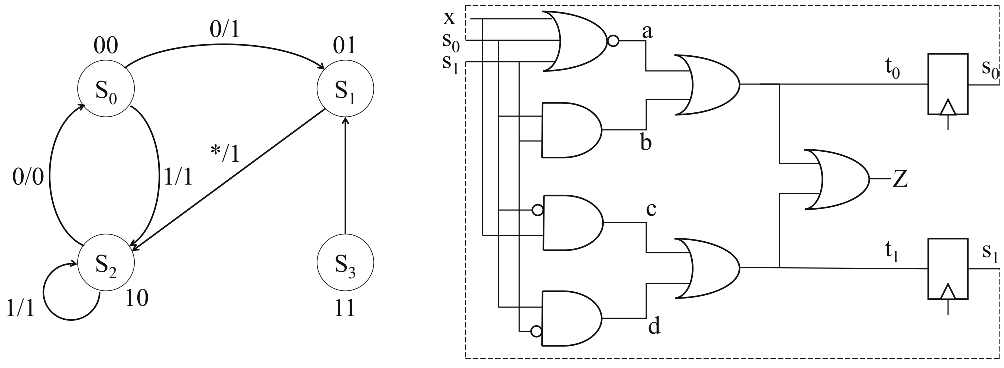
Let us describe the application of the algorithm on the FSM circuit of Fig. 2. *We will first describe its operation at the Boolean level, and then describe how this algorithm can be implemented using algebraic geometry at word level.*

In Line 1 of the BFS algorithm, assume that the initial state is S_3 in Fig.2(b), which is encoded as $S_3 = \{11\}$. Using Boolean variables s_0, s_1 for the present states, $from^0 = s_0 \cdot s_1$ is represented as a Boolean formula.

Example III.1. *For the circuit in Fig. 2 (b), we have the transition functions of the machine as:*

$$\begin{aligned} \Delta_1 &: t_0 \overline{\oplus} ((\overline{x \vee s_0 \vee s_1}) \vee s_0 s_1) \\ \Delta_2 &: t_0 \overline{\oplus} (\overline{s_0} x \vee \overline{s_1} s_0) \\ from &: from^0 = s_0 \cdot s_1 \end{aligned}$$

When the formula of Eqn. (2) is applied to compute 1-step reachability, $to = \exists_{s_0, s_1, x}(\Delta_1 \cdot \Delta_2 \cdot from^0)$, we obtain $to = \overline{t_0} \cdot t_1$, which denotes the state $S_1 = \{01\}$ reached in 1-step from S_3.



(a) (b)

Fig. 2: The example FSM and the gate-level implementation.

In the next iteration, the algorithm uses state $S_1 = \{01\}$ as the current (initial) state, and computes $S_2 = \{10\} = t_0 \cdot \overline{t_1}$ as the next reachable state, and so on.

Our objective is to model the transition functions Δ as a polynomial ideal J, and to perform the image computations using Gröbner bases over Galois fields. *This requires to perform quantifier elimination; which can be accomplished using the GB computation over $\mathbb{F}_{2^k}$ using elimination ideals* [13]. Finally, the set union, intersection and complement operations are also to be implemented in algebraic geometry.

FSM Traversal at word-level over $\mathbb{F}_{2^k}$: The state transition graph (STG) shown in Fig.2(a) uses a 2-bit Boolean vector to represent 4 states $\{S_0, S_1, S_2, S_3\}$. We map these states to elements in $\mathbb{F}_{2^2}$, where $S_0 = 0, S_1 = 1, S_2 = \alpha, S_3 = \alpha + 1$. Here, we take $P(X) = X^2 + X + 1$ as the irreducible polynomial to construct $\mathbb{F}_4$, and $P(\alpha) = 0$ so that $\alpha^2 + \alpha + 1 = 0$.

Initial state: In Line 1 of Alg.1, the initial state is specified by means of a corresponding polynomial $f = \mathcal{F}(S) = S - 1 - \alpha$. Notice that if we consider the ideal generated by the initial state polynomial, $I = \langle f \rangle$, then its variety $V(I) = 1 + \alpha$ corresponds to the state encoding $S_3 = \{11\} = 1 + \alpha$, where a polynomial in word-level variable S encodes the initial state.

Set operations: In Lines 5 and 6 of Alg. 1, we need **union**, **intersection** and **complement** of varieties over $\mathbb{F}_{2^k}$, for which we again use algebraic geometry concepts.

Definition III.1. *(***Sum/Product of Ideals** *[11]) If $I = \langle f_1, \dots, f_r \rangle$ and $J = \langle g_1, \dots, g_s \rangle$ are ideals in R, then the* **sum** *of I and J is defined as $I + J = \langle f_1, \dots, f_r, g_1, \dots, g_s \rangle$. Similarly, the* **product** *of I and J is $I \cdot J = \langle f_i g_j \mid 1 \le i \le r, 1 \le j \le s \rangle$.*

Theorem III.1. *If I and J are ideals in R, then $\mathbf{V}(I + J) = \mathbf{V}(I) \bigcap \mathbf{V}(J)$ and $\mathbf{V}(I \cdot J) = \mathbf{V}(I) \bigcup \mathbf{V}(J)$.*

In Line 5 of Alg. 1, we need to compute the complement of a set of states. Assume that J denotes a polynomial ideal whose variety $V(J)$ denotes a set of states. We require the computation of another polynomial ideal J', such that $V(J') = \overline{V(J)}$. We show that this computation can be performed using the concept of **ideal quotient**:

Definition III.2. *(***Quotient of Ideals***) If I and J are ideals in a ring R, then $I : J$ is the set $\{f \in R \mid f \cdot g \in I, \forall g \in J\}$ and is called the* **ideal quotient** *of I by J.*

Example III.2. *In $\mathbb{F}_q[x, y, z]$, ideal $I = \langle xz, yz \rangle$, ideal $J = \langle z \rangle$. Then*

$$\begin{aligned} I : J &= \{f \in \mathbb{F}_q[x, y, z] \mid f \cdot z \in \langle xz, yz \rangle\} \\ &= \{f \in \mathbb{F}_q[x, y, z] \mid f \cdot z = Axz + Byz\} \\ &= \{f \in \mathbb{F}_q[x, y, z] \mid f = Ax + By\} \\ &= \langle x, y \rangle \end{aligned}$$

We can now obtain the complement of a variety through the following results which are stated and proved below:

Lemma III.1. *Let $f, g \in \mathbb{F}_{2^k}[x]$, then $\langle f : g \rangle = \left\langle \frac{f}{gcd(f,g)} \right\rangle$.*

Proof. Let $d = gcd(f, g)$. So, $f = df_1, g = dg_1$ with $gcd(f_1, g_1) = 1$. Note that $f_1 = \frac{f}{gcd(f,g)}$.

Take $h \in \langle f : g \rangle$. According to the Def. III.2, $hg \in \langle f \rangle$, which means $hg = f \cdot r$ with $r \in \mathbb{F}_{2^k}[x]$. Therefore, $hdg_1 = df_1 r$ and $hg_1 = f_1 r$. But considering $gcd(g_1, f_1) = 1$ we have the fact that f_1 divides h. Hence $h \in \langle f_1 \rangle$.

Conversely, let $h \in \langle f_1 \rangle$. Then $h = s \cdot f_1$, where $s \in \mathbb{F}_{2^k}[x]$. So, $hg = hdg_1 = sf_1 dg_1 = sg_1 f \in \langle f \rangle$. Therefore, $h \in \langle f : g \rangle$. □

Theorem III.2. *Let J be an ideal generated by a single univariate polynomial in variable x over $\mathbb{F}_{2^k}[x]$, and let the vanishing ideal $J_0 = \langle x^{2^k} - x \rangle$. Then*

$$V(J_0 : J) = V(J_0) - V(J),$$

where all the varieties are considered over the field $\mathbb{F}_{2^k}$.

Proof. Since $\mathbb{F}_{2^k}[x]$ is a principal ideal domain, $J = \langle g \rangle$ for some polynomial $g \in \mathbb{F}_{2^k}[x]$. Let $d = gcd(g, x^{2^k} - x)$. So, $g = dg_1, x^{2^k} - x = df_1$, with $gcd(f_1, g_1) = 1$. Then $J_0 : J = \langle f_1 \rangle$ by Lemma III.1.

Let $x \in V(J_0) - V(J)$. From the definition of set complement, we get $x \in \mathbb{F}_{2^k}$ while $g(x) \neq 0$.

Since $x^{2^k} = x$, we see that either $d(x) = 0$ or $f_1(x) = 0$. Considering $g(x) \neq 0$, we can assert that $d(x) \neq 0$. In conclusion, $f_1(x) = 0$ and $x \in V(f_1)$.

Now let $x \in V(f_1)$, we get $f_1(x) = 0$. So, $x^{2^k} - x = 0$ gives $x \in V(J_0) = \mathbb{F}_{2^k}$ which contains all elements in the field.

Now we make an assumption that $x \in V(g)$. Then $g(x) = 0 = d(x)g_1(x)$ which means either $d(x) = 0$ or $g_1(x) = 0$.

If $g_1(x) = 0$, then since $f_1(x) = 0$ we get that f_1, g_1 share a root. This contradicts the fact that $gcd(f_1, g_1) = 1$.

On the other hand, if $d(x) = 0$, then since $f_1(x) = 0$ and $x^{2^k} - x = df_1$, we get that $x^{2^k} - x$ has a double root. But this is impossible since the derivative of $x^{2^k} - x$ is -1.

So, $x \notin V(g)$ and this concludes the proof. □

Let $x^{2^k} - x$ be a vanishing polynomial in $\mathbb{F}_{2^k}[x]$. Then $V(x^{2^k} - x) = \mathbb{F}_{2^k}$ i.e. the variety of vanishing ideal contains all possible valuations of variables, so it constitutes the **universal set**. Subsequently, based on Theorem III.2, the **absolute complement** $V(J')$ of a variety $V(J)$ can be computed as:

Corollary III.1. *Let $J \subseteq \mathbb{F}_{2^k}[x]$ be an ideal, and $J_0 = \langle x^{2^k} - x\rangle$. Let J' be an ideal computed as $J' = J_0 : J$. Then*

$$V(J') = \overline{V(J)} = V(J_0 : J)$$

With Corollary III.1, we are ready to demonstrate the concept of word-level FSM traversal over $\mathbb{F}_{2^k}$ using algebraic geometry. The algorithm is given in Alg. 2. Note that in the algorithm, $from^i, to^i, new^i$ are *univariate polynomials in variables S or T* only, due to the fact that they are the result of a GB computation with an elimination term order, where the bit-level variables are abstracted and quantified away.

ALGORITHM 2: Algebraic Geometry based FSM Traversal

```
Input: The circuit's characteristic polynomial ideal $J_{ckt}$, initial
       state polynomial $\mathcal{F}(S)$, and LEX term order: bit-level
       variables $x, s, t >$ PS word $S >$ NS word $T$
1  $from^0 = reached = \mathcal{F}(S)$;
2  repeat
3      $i \leftarrow i + 1$;
4      $G \leftarrow$GB($\langle J_{ckt}, J_v, from^{i-1}\rangle$);
       /* Compute Gröbner basis with elimination
          term order: $T$ smallest                  */
5      $to^i \leftarrow G \cap \mathbb{F}_{2^k}[T]$;
       /* There will be a univariate polynomial
          in $G$ denoting the set of next states
          in word-level variable $T$                */
6      $\langle new^i\rangle \leftarrow \langle to^i\rangle + (\langle T^{2^k} - T\rangle : \langle reached\rangle)$;
       /* Use quotient of ideals to attain
          complement of reached states, then use
          sum of ideals to attain an
          intersection with next state             */
7      $\langle reached\rangle \leftarrow \langle reached\rangle \cdot \langle new^i\rangle$;
       /* Use product of ideals to attain a
          union of newly reached states and
          formerly reached states                  */
8      $from^i \leftarrow new^i(S \setminus T)$;
       /* Start a new iteration by replacing
          variable $T$ in newly reached states
          with current state variable $S$          */
9  until $\langle new^i\rangle == \langle 1\rangle$;
   /* Loop until a fixpoint reached: newly
      reached state is empty                       */
10 return $\langle reached\rangle$
```

Example III.3. *We apply Algorithm 2 to the example shown in Fig. 2 to execute the FSM traversal. Let the initial state $from^0 = \{00\}$ in $\mathbb{B}^2$ or $0 \in \mathbb{F}_4$. Polynomially, it is written as $from^0 = S - 0$. In the first iteration, we compose an ideal J with*

$$f_1 : t_0 - (xs_0s_1 + xs_0 + xs_1 + x + s_0 + s_1 + 1)$$
$$f_2 : t_1 - (xs_0 + x + s_0s_1 + s_0)$$
$$f_3 : S - s_0 - s_1\alpha; \quad f_4 : T - t_0 - t_1\alpha$$

$J_{ckt} = \langle f_1, f_2, f_3, f_4\rangle$, and the vanishing polynomials:

$$f_5 : x^2 - x; \quad f_6 : s_0^2 - s_0, \quad f_7 : s_1^2 - s_1$$
$$f_8 : t_0^2 - t_0, \quad f_9 : t_1^2 - t_1; \quad f_{10} : S^4 - S, \quad f_{11} : T^4 - T$$

with $J_v = \langle f_5, f_6, \ldots, f_{11}\rangle$.

Compute $G = GB(J)$ for $J = J_{ckt} + J_0 + \langle from^0\rangle$, with an elimination term order

$$\underbrace{\{x, s_0, s_1, t_0, t_1\}}_{\textit{all bit-level variables}} > \underbrace{S}_{\textit{(PS word)}} > \underbrace{T}_{\textit{(NS word)}}.$$

The resulting GB G contains a polynomial generator with only T as the variable. In Line 5, assign it to the next state

$$to^1 = T^2 + (\alpha + 1)T + \alpha.$$

Note that the roots or variety of $T^2 + (\alpha + 1)T + \alpha$ is $\{1, \alpha\}$, denoting the states $\{01, 10\}$.

Since the formerly reached state "reached $= T$", its complement is computed using Corollary III.1

$$\langle T^4 - T\rangle : \langle T\rangle = \langle T^3 + 1\rangle.$$

$V(\langle T^3 + 1\rangle) = \{1, \alpha, \alpha + 1\}$ denoting the states $\{01, 10, 11\}$. Then the newly reached state set in this iteration is

$$\langle T^3 + 1, T^2 + (\alpha + 1)T + \alpha\rangle = \langle T^2 + (\alpha + 1)T + \alpha\rangle$$

We add these states to formerly reached states

$$\begin{aligned} reach &= \langle T\rangle \cdot \langle T^2 + (\alpha + 1)T + \alpha\rangle \\ &= \langle T \cdot T^2 + (\alpha + 1)T + \alpha\rangle \\ &= \langle T^3 + (\alpha + 1)T^2 + \alpha T\rangle \end{aligned}$$

i.e. states $\{00, 01, 10\}$. We update the present states for next iteration

$$from^1 = S^2 + (\alpha + 1)S + \alpha.$$

In the second iteration, we compute the reduced GB with the same term order for ideal $J = J_{ckt} + J_v + \langle from^1\rangle$. It includes a polynomial generator

$$to^2 = T^2 + \alpha T$$

denotes states $\{00, 10\}$. The complement of reached is

$$\langle T^4 - T\rangle : \langle T^3 + (\alpha + 1)T^2 + \alpha T\rangle = \langle T + 1 + \alpha\rangle$$

(i.e. states $\{11\}$). We compute the newly reached state

$$\langle T^2 + \alpha T, T + 1 + \alpha\rangle = \langle 1\rangle$$

Since the GB contains the unit ideal, it means the newly reached state set is empty, thus a fixpoint has been reached. The algorithm terminates and returns

$$reached = \langle T^3 + (\alpha + 1)T^2 + \alpha T\rangle$$

which, as a Gröbner basis of the elimination ideal, canonically encodes the final reachable state set.

Significance of using GB: A reduced GB is a unique, minimal and *canonical* representation of the circuit's function. Starting from a certain initial state and using a reduced GB to represent the transition function, reachable states can be computed and represented canonically. Then it becomes possible to identify when a fixpoint is reached (termination of the algorithm) by performing an equality check of polynomial ideals. Moreover, the GB computation is also used as a quantification procedure. As the GB is computed w.r.t. an elimination term order with "bit-level variables" > "present-state word" S > "next-state word" T, the set of reachable states are encoded, canonically, using a univariate polynomial in T, quantifying away the rest of the variables.

IV. Improving Our Approach

A. Simplify the Gröbner Basis Computation

In Alg. 2, a Gröbner basis is computed for each iteration to attain the word-level polynomial representation of the next states. In practice, for a sequential circuit with complicated structure and large size, Gröbner basis computation is intractable. To overcome the high computational complexity of computing a GB, we describe a method that computes a GB of a smaller subset of polynomials. The approach draws inspirations from [7]. According to Prop.2 in [7], if the GB is computed using *refined abstraction term order* (RATO), there will be only one pair of polynomials $\{f_w, f_g\}$ such that their leading monomials are not relatively prime, i.e.

$$gcd(LM(f_w), LM(f_g)) \neq 1$$

As a well-known fact from Buchberger's algorithm, the S-polynomial (Spoly) pairs with relatively prime leading monomials will always reduce to 0 modulo the basis and have no contribution to the Gröbner basis computation. Therefore, by removing the relatively prime polynomials from J_{ckt}, the Gröbner basis computation is transformed to the reduction of $Spoly(f_w, f_g)$ modulo J_{ckt}. More specifically, we turn the GB computation into one-step multivariate polynomial division, and the obtained remainder r will only contain bit-level inputs and word-level output.

Example IV.1. *After adding intermediate bit-level signal a, b, c, d, the elimination ideal for example circuit (Ex.III.3) can be rewritten LEX order with RATO:*

$$\begin{aligned}(t_0, t_1) &> (a, b, c, d)\\ &> T > (x, s_0, s_1)\end{aligned}$$

We can write down all polynomial generators of J_{ckt}:

$$f_1 : a + xs_0s_1 + xs_0 + xs_1 + x + s_0s_1 + s_0 + s_1 + 1$$

$$\begin{array}{ll} f_2 : b + s_0s_1 & f_3 : c + x + xs_0 \\ f_4 : d + s_0s_1 + s_0 & f_5 : t_0 + ab + a + 1 \\ f_6 : t_1 + cd + c + d & f_7 : t_0 + t_1\alpha + T \end{array}$$

From observation, the only pair which is not relatively prime is (f_5, f_7), thus the critical candidate polynomial pair is (f_w, f_g), where

$$f_w = t_0 + a \cdot b + a + b, \quad f_g = t_0 + t_1\alpha + T$$

Result after reduction is:

$$\begin{aligned}&Spoly(f_w, f_g) \xrightarrow{J+J_0}_{+} T + s_0s_1x + \alpha s_0s_1\\ &\quad + (1+\alpha)s_0x + (1+\alpha)s_0 + s_1x + s_1 + (1+\alpha)x + 1\end{aligned}$$

The remainder contains only bit-level inputs (x, s_0, s_1) and word-level output T.

The remainder from *Spoly* reduction contains bit-level PS variables, and our objective is to get a polynomial containing only word-level PS variables. One possible method is to rewrite bit-level variables in term of word-level variables, i.e.

$$s_i = \mathcal{G}(S) \tag{3}$$

Then we can substitute all bit-level variables with the word-level variable and obtain a word-level expression. The authors of [7] propose a method to construct a system of equations and solution to the system consists of Eqn.(3).

Example IV.2. Objective*: Abstract polynomial $s_i + \mathcal{G}_i(S)$ from $f_0 : s_0 + s_1\alpha + S$.*

First, compute $f_0^2 : s_0 + s_1\alpha^2 + S^2$. Apparently variable s_0 can be eliminated by operation

$$\begin{aligned}f_1 &= f_0 + f_0^2\\ &= (\alpha^2 + \alpha)s_1 + S^2 + S\end{aligned}$$

Now we can solve univariate polynomial equation $f_1 = 0$ and get solution

$$s_1 = S^2 + S$$

Using this solution we can easily solve equation $f_0 = 0$. The result is

$$s_0 = \alpha S^2 + (1+\alpha)S$$

More formally, polynomial expressions for s_i in terms of S can be obtained by setting up and solving the following system of equations:

$$\begin{bmatrix} S \\ S^2 \\ S^{2^2} \\ \vdots \\ S^{2^{k-1}} \end{bmatrix} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{k-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(k-1)} \\ 1 & \alpha^4 & \alpha^8 & \cdots & \alpha^{4(k-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{2^{k-1}} & \alpha^{2\cdot 2^{k-1}} & \cdots & \alpha^{(k-1)\cdot 2^{k-1}} \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ \vdots \\ s_{k-1} \end{bmatrix} \tag{4}$$

Treat $\mathbf{s}$ as a vector of k unknowns $s_0, \ldots, s_{k-1}$, then Eqn.(4) can be solved by using Cramer's rule or Gaussian elimination. In other words, we can obtain $s_i = \mathcal{G}(S)$ by solving Eqn.(4) symbolically.

In this approach we get word-level variable representation for each bit-level PS variables. By substitution, a new polynomial in word-level PS/NS variables could be obtained.

After processing with RATO and bit-to-word conversions, we get a polynomial in the form of $f_T = T + \mathcal{F}(S, x)$ denoting the **transition function**. We include a polynomial in S to define the present states f_S, as well as the set of vanishing polynomials for primary inputs $J_0^{PI} = \langle x_1^2 - x_1, \dots, x_n^2 - x_n \rangle$. Using elimination term order with $S > x_i > T$, we can compute a GB of the elimination ideal $\langle f_T, f_S \rangle + J_0^{PI}$. This GB contains a univariate polynomial denoting next states. The improved algorithm is depicted in Alg. 3.

ALGORITHM 3: Refined Algebraic Geometry based FSM Traversal

```
Input: Input-output circuit characteristic polynomial ideal $J_{ckt}$,
       initial state polynomial $\mathcal{F}(S)$
1  $from^0 = reached = \mathcal{F}(S)$;
2  $f_T$ =Reduce($Spoly(f_w, f_g), J_{ckt}$);
   /* Compute S-poly for the critical pair,
      then reduce it with circuit ideal under
      RATO                                          */
3  Eliminate bit-level variables in $f_T$;
4  repeat
5      $i \leftarrow i + 1$;
6      $G \leftarrow$GB($\langle f_T, from^{i-1} \rangle + J_0^{PI}$);
       /* Compute Gröbner basis with elimination
          term order: $T$ smallest; $J_0^{PI}$ covers all
          possible inputs from PIs                  */
7      $to^i \leftarrow G \cup \mathbb{F}_{2^k}[T]$;
       /* There will be a univariate polynomial
          in $G$ denoting next state in
          word-level variable $T$                   */
8      $\langle new^i \rangle \leftarrow \langle to^i \rangle + (\langle T^{2^k} - T \rangle : \langle reached \rangle)$;
       /* Use quotient of ideals to attain
          complement of reached states, then use
          sum of ideals to attain an
          intersection with next state              */
9      $\langle reached \rangle \leftarrow \langle reached \rangle \cdot \langle new^i \rangle$;
       /* Use product of ideals to attain a
          union of newly reached states and
          formerly reached states                   */
10     $from^i \leftarrow new^i(S \setminus T)$;
       /* Start a new iteration by replacing
          variable $T$ in newly reached states
          with current state variable $S$           */
11 until $\langle new^i \rangle == \langle 1 \rangle$;
   /* Loop until a fixpoint reached: newly
      reached state is empty                        */
12 return $\langle reached \rangle$
```

V. Experiment results

We have implemented our traversal algorithm in 3 parts: the first part implements polynomial reductions (division) of the Gröbner basis computations, under the term order derived from the circuit as Line 2 in Alg. 3. This is implemented with our customized data structure in C++. The second part implements the bit-level to word-level abstraction to attain transition functions at the word-level using the Singular symbolic algebra computation system [v. 3-1-6] [14], as Line 3 in Alg. 3; and the third part executes the reachability checking iterations using Singular as well. With our tool implementation, we have performed experiments to analyze reachability of several FSMs. Our experiments run on a desktop with 3.5GHz Intel Core$^{\text{TM}}$ i7-4770K Quad-core CPU, 32 GB RAM and 64-bit Ubuntu Linux OS. The experiments are shown in Table I.

There are 2 bottlenecks which restricts the performance of our tool: one bottleneck is that the polynomial reduction engine is slow when the number of gates (especially OR gates) is large; the other one is the high computational complexity of Gröbner basis engine in general. Therefore, we pick 10 FSM benchmarks of reasonable size for testing our tool. Among them "b01, b02, b06" come from ITC'99 benchmarks, "s27, s208, s386" are from ISCAS'89 benchmarks and "bbara, beecount, dk14, donfile" are from MCNC benchmarks. ISCAS benchmarks are given as *bench* format so we can directly read gate information, where ITC/MCNC FSMs are given in unsynthesized *blif* format so we first turn them into gate-level netlists using AIG based synthesizer ABC. Since the number of primary inputs (m) is relatively small, in our experiments we partition primary inputs as m single bit-level variables. To verify the correctness of our techniques and implementations, we compare the number of reachable states obtained from our tool against the results obtained from the VIS tool [15].

In Table I, # States denotes the final reachable states starting from given reset state, which given by our tool is the same with the return value of *compute_reach* in VIS. Meanwhile, from observation of the experiment run-times, we find the reduction runtime increases as the number of gates grows. Also, iterative reachability convergence check's runtime reflects both the size of present state/next state words (k) and the number of final reached states, which corresponds to the degree of polynomial *reached* in Alg.2. Although the efficiency of our initial implementation fails to compete with the BDD based FSM analyzer VIS, the experiment demonstrates the power of abstraction of algebraic geometry techniques for reachability analysis applications. Currently, we are investigating techniques that can help us overcome the complexity of the GB computation with elimination orders and speed-up our approach.

VI. Conclusion

This paper has presented a new approach to perform reachability analysis of finite state machines at the word-level. This is achieved by modeling the transition relations and sets of states by way of polynomials over finite fields $\mathbb{F}_{2^k}$, where k represents the size of the state register bits. Subsequently using the concepts of elimination ideals, Gröbner bases, and quotients of ideals, we show that the set of reachable states can be encoded, canonically, as the variety of a univariate polynomial. This polynomial is computed using the Gröbner basis algorithm w.r.t. an elimination term order. Experiments are conducted with a few FSMs that validate the concept of word-level FSM traversal using algebraic geometry.

References

[1] Herve J Touati, Hamid Savoj, Bill Lin, Robert K Brayton, and Alberto Sangiovanni-Vincentelli, "Implicit state enumeration of finite state machines using bdd's", *in Computer-Aided Design, 1990. ICCAD-90. Digest of Technical Papers., 1990 IEEE International Conference on*, pp. 130–133. IEEE, 1990.

[2] Olivier Coudert, Christian Berthet, and Jean Christophe Madre, "Verification of synchronous sequential machines based on symbolic execution", *in Automatic verification methods for finite state systems*, pp. 365–373. Springer, 1990.

[3] Jerry R Burch, Edmund M Clarke, Kenneth L McMillan, and David L Dill, "Sequential circuit verification using symbolic model checking", *in Design Automation Conference, 1990. Proceedings., 27th ACM/IEEE*, pp. 46–51. IEEE, 1990.

TABLE I: Results of running benchmarks using our tool. Parts I to III denote the time taken by polynomial divisions, bit-level to word-level abstraction and iterative reachability convergence checking part of our approach, respectively.

Benchmark	# Gates	# Latches	# PIs	# States	# iterations	Runtime (sec)			Runtime of VIS (sec)
						I	II	III	
b01	39	5	2	18	5	< 0.01	0.01	0.02	< 0.01
b02	24	4	1	8	5	< 0.01	0.01	< 0.01	< 0.01
b06	49	9	2	13	4	< 0.01	0.07	5.0	< 0.01
s27	10	3	4	6	2	< 0.01	0.01	0.02	< 0.01
s208	61	8	11	16	16	< 0.01	0.32	2.4	< 0.01
s386	118	6	13	13	3	1.0	7.6	8.2	< 0.01
bbara	82	4	4	10	6	0.04	0.01	0.04	< 0.01
beecount	48	3	3	7	3	< 0.01	0.01	0.01	< 0.01
dk14	120	3	3	7	2	45	< 0.01	0.08	< 0.01
donfile	205	5	2	24	3	12316	0.02	1.7	< 0.01

[4] Aaron R Bradley, "Sat-based model checking without unrolling", *in Verification, Model Checking, and Abstract Interpretation*, pp. 70–87. Springer, 2011.

[5] Himanshu Jain, Daniel Kroening, Natasha Sharygina, and Edmund Clarke, "Word level predicate abstraction and refinement for verifying RTL verilog", *in Proceedings of the 42nd annual Design Automation Conference*, pp. 445–450. ACM, 2005.

[6] R. Lidl and H. Niederreiter, *Finite Fields*, Cambridge University Press, 1997.

[7] T. Pruss, P. Kalla, and F. Enescu, "Efficient Symbolic Computation for Word-Level Abstraction from Combinational Circuits for Verification over Finite Fields", *IEEE Trans. on CAD*, vol. 35, pp. 1206–1218, July 2016.

[8] Xiaojun Sun, Priyank Kalla, Tim Pruss, and Florian Enescu, "Formal verification of sequential galois field arithmetic circuits using algebraic geometry", *in Design Automation and Test in Europe, DATE 2015. Proceedings*. IEEE/ACM, 2015.

[9] G. Avrunin, "Symbolic Model Checking using Algebraic Geometry", *in Computer Aided Verification Conference*, pp. 26–37, 1996.

[10] Q. Tran and M. Y. Vardi, "Gröbner Basis Computation in Boolean Rings for Symbolic Model Checking", *in IASTED Conf. on Modelling and Simulation*, 2007.

[11] D. Cox, J. Little, and D. O'Shea, *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*, Springer, 2007.

[12] B. Buchberger, *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*, PhD thesis, University of Innsbruck, 1965.

[13] S. Gao, A. Platzer, and E. Clarke, "Quantifier Elimination over Finite Fields with Gröbner Bases", *in Intl. Conf. Algebraic Informatics*, 2011.

[14] W. Decker, G.-M. Greuel, G. Pfister, and H. Schönemann, "SINGULAR 3-1-3 — A computer algebra system for polynomial computations", 2011, http://www.singular.uni-kl.de.

[15] Robert K Brayton, Gary D Hachtel, Alberto Sangiovanni-Vincentelli, Fabio Somenzi, Adnan Aziz, Szu-Tsung Cheng, Stephen Edwards, Sunil Khatri, Yuji Kukimoto, Abelardo Pardo, et al., "Vis: A system for verification and synthesis", *in Computer Aided Verification*, pp. 428–432. Springer, 1996.