GROEBNER BASES COMPUTATION IN BOOLEAN RINGS FOR SYMBOLIC MODEL CHECKING

Quocnam Tran[1] & Moshe Y. Vardi
Rice University, Houston, Texas

ABSTRACT

Model checking is an algorithmic approach for automatically verifying whether a hardware or software system functions correctly. Typically, computation is carried over Boolean algebras using binary decision diagrams (BDDs) or satisfiability (SAT) solvers. In this paper we show that computation for model checking can also be carried over the dual Boolean rings of the Boolean algebras by means of efficient polynomial and Groebner basis (GB) computation. We also show how all operations required for model checking can be implemented by means of Groebner bases.

KEYWORDS

Groebner basis, Boolean ring, model checking, automated formal verification.

I. INTRODUCTION

Automated formal verification is an active research direction aiming at increasing the productivity of system designers. In digital circuit design, each design has to be verified with respect to numerous aspects before production can be started, for example, low level design rules such as layout, timing; high level design rules such as logic level, firmware, and functional correctness, which is the aim of this work.

In automated formal verification, *mathematical techniques* are used to guarantee the correctness of a design with respect to some specified behavior. Automated formal verification is based on model checking [4], [11], [23], temporal logic [26], automata-theoretic techniques [21], [32]. It has the ability to discover subtle flaws resulted from improbable events. In fact, it is one of the most successful applications of automated reason in computer science.

Model checking has several attractive features. Once the Kripke model of the system and the temporal logic specifications have been defined, the process is fully automated. If a computation that violates the specification is found in the Kripke model, it can be displayed to the designer to aid the debugging process. In addition, the model checker can formally certify that the specification holds [24], [25].

The first step in formal verification is the representation of formal specification of the design consisting of a description of the desired behavior. As sequential systems capture time-variant behavior, it is not possible to describe their properties completely in the framework of conventional propositional formulas. In a temporal logic, temporal operators are additionally provided, which can be used to express time-variant dependencies. A widely used specification language for designs is Temporal Logic [26], which is a modal logic (i.e. logic to reason about many possible world with its semantics based on Kripke structures).

In linear temporal logic (LTL), time is treated as if each moment in time has a unique possible future. (An alternative approach is to use branching time (CTL) [12].) Linear temporal formulas are constructed from a set $\wp$ of atomic propositions using usual Boolean connectives as well as temporal connectives $\mathbf{X}$ ("next"), $\mathbf{G}$ ("always"), $\mathbf{F}$ ("eventually") and $\mathbf{U}$ ("until").

We define the semantics of temporal formulas with respect to a Kripke structure $K = (\mathbb{N}, \leq, \pi)$, where $\mathbb{N}$ is the set of natural numbers, $\leq \subseteq \mathbb{N}^2$ is the standard linear order, and $\pi : \mathbb{N} \to 2^{\wp}$ is a function that defines what propositions are true at a certain time instant.

In model checking, we assume that the specification is given in terms of properties expressed by temporal formulas. For example, the LTL formula $\mathbf{G}(\textit{request} \rightarrow \mathbf{F}\ \textit{grant})$, which refers to the atomic propositions *request* and *grant*, specifies that every state in the computation in which *request* holds is followed by some state in the future in which *grant* holds.

Using temporal logic as a specification language for systems quite naturally leads to the idea of model checking. In fact, model checking has become one of the most active studied automated formal verification techniques [11], [12], [27]. In model checking, a temporal logic specification is checked against a Kripke model of the finite state system (i.e. the implementation). For realistic designs, the number of states of the system can be very large, 10^{20} or higher, and hence explicit traversal of the state space becomes infeasible. Perhaps, the biggest hurdle for a practical use of model checking is the state explosion problem [31]. To relieve the state explosion problem, many approaches have been proposed. Symbolic model checking using ordered binary decision diagrams (BDD) is a successful and widely used techniques for verifying properties of concurrent hardware and software systems. The underlying model for BDD was studied in the 1950s and 1970s [1], [22]. This basic data structure has been widely used in many areas of computer-aided VLSI design after Bryant's substantial improvement [7] in that he added some ordering restrictions as well as sophisticated reduction mechanism to the model. In symbolic model checking, the state space is represented implicitly using symbolic means and propositional logic formulas are manipulated using BDDs [19]. Symbolic model checking succeeded in checking systems with unprecedented large state spaces.

Even though BDDs are a canonical representation of Boolean formulas and they can succinctly represent many Boolean functions, the efficiency of the BDD representation for a Boolean function very much depends on finding a good order of variables in the Boolean formula. Efficient finding of a good variable ordering or converting a BDD from one order to another order are still open problems. To make the situation worse, there are functions which do not have any succinct BDD representation [7]. Unpredictable blow-ups in memory usage can occur if the current variable order is unsuitable for the current Boolean formula. Practically, there are many systems for which the BDDs are too large to make model checking feasible.

In bounded model checking (BMC), a limited model

[1] On sabbatical leave from Lamar University, Beaumont, Texas.

checking problem is considered: only *counterexamples* of a fixed length are sought for. By letting the length k grow incrementally, we can prove that the system contains no counterexamples of length k or less [4], [5]. For a finite system, the method is complete if one lets the bound grow large enough, for example greater the diameter of the explicit state automaton. However, finding a tight bound for k is a hard problem.

Bounded model checking uses the same basic idea as symbolic model checking using BDD in that the state space of the system is represented implicitly by Boolean formulas. However, instead of manipulating the Boolean formulas using BDDs, bounded model checking transforms the model and property specifications into a propositional satisfiability (SAT) problem. Given a system M, a temporal logic formula ψ and a bound k, a Boolean formula is constructed, which is satisfiable if and only if M has a counterexample of length k to ψ. A SAT solver such as SATO or zChaff is used to perform the query. In BMC, the growth of the size of the Boolean formulas can be known in advanced, but predicting the running times of the SAT solver is difficult. There are quite some successful industrial applications of BMC [5], [6] but BMC is only good at finding shallow counterexamples.

It is worthwhile to investigate alternative symbolic representations for Boolean sets that could be used for model checking. New approaches would hopefully supplement existing symbolic model checking methods.

The paper is organized as follows: In the next section, we present some algebraic preliminaries. In Section III, we present our method for direct Groebner basis computation in Boolean rings and show how all operations required for model checking can be implemented by means of Groebner bases.

II. Algebraic Preliminaries

In this section, we summarize some basic facts about Boolean rings and the method of Groebner bases. We also summarize efforts for using these algebraic concepts in propositional proof systems and model checking. Furthermore, we point out the obstacles for an *effective use* of these concepts for model checking.

A. Boolean rings

Boolean algebras, which were introduced by Boole in the 1850's to codify the laws of thought, have become a popular topic of research since then. The discovery in 1930's of the duality between Boolean algebras and Boolean spaces by Stone [10], [28] was a major breakthrough of the field. Stone also proved that Boolean algebras and Boolean rings are the same in the sense that one can convert from one algebraic structure to the other.

Definition 1: A ring $\mathbf{R} = \langle R, +, \cdot, 0, 1\rangle$ is Boolean if $\mathbf{R}$ satisfies $x^2 \approx x, \forall x \in R$.

Lemma 1: If $\mathbf{R}$ is a Boolean ring, then $\mathbf{R}$ is commutative and $x + x \approx 0$ [10].

Every Boolean algebra $(R, \wedge, \vee)$ gives rise to a ring $(R, +, \cdot)$ by defining $a+b = (a \wedge \neg b) \vee (b \wedge \neg a)$ and $a \cdot b = a \wedge b$. The zero element of this ring coincides with the 0 of the Boolean algebra; the multiplicative identity element of the ring is the 1 of the Boolean algebra. Conversely, if a Boolean ring $\mathbf{R}$ is given, we can turn it into a Boolean algebra by defining $x \vee y = x+y+x \cdot y$, $x \wedge y = x \cdot y$ and $\neg x = x+1$. Since these two sets of operations are invertible from each other, we can say that every Boolean ring arises from a Boolean algebra, and vice versa. Furthermore, a map $f : A \to B$ is a homomorphism of Boolean algebras if and only if it is a homomorphism of Boolean rings. The categories of Boolean rings and Boolean algebras are equivalent. By using these translations, there exists a Boolean polynomial for each Boolean formula and vice versa.

B. The Method of Groebner Bases

Once the Boolean formulas have been converted into an equivalent system of polynomials in the corresponding Boolean ring, one can use the results from symbolic computation to perform calculation on the polynomial system. In this section, we give a short introduction to basic facts on admissible term orders and the method of Groebner bases. We refer to [8], [9], [29], [33] for missing details.

Let K be a computable field and $K[x_1, \ldots, x_n]$ the polynomial ring in n variables over K. In the context of Boolean rings, K is restricted to a finite field or even a field with two elements 0 and 1. We denote the set of power products in the variables $x_1, x_2, \ldots, x_n$ by $[X]$.

Definition 2: A total order $\prec$ on $[X]$ is called an admissible term order (or a term order for short) iff

1) $\forall t \in [X] \setminus \{1\}$, $1 = x_1^0 \cdot x_2^0 \cdots x_n^0 \prec t$, and
2) $\forall s, t, u \in [X]$, $s \prec t \Rightarrow s \cdot u \prec t \cdot u$.

Let I be an ideal in $K[X]$, f be a non-zero polynomial in $K[X]$ and $\prec$ be a term order on $[X]$. We are particularly interested in elimination term orders and efficient term orders (see [30], [33] for classes of these term orders). We denote

- $\mathrm{lpp}_\prec(f)$ the leading power product of f with respect to $\prec$.
- $\mathrm{lc}_\prec(f)$ the leading coefficient of f with respect to $\prec$.
- $f_\prec = \mathrm{lc}_\prec(f) \cdot \mathrm{lpp}_\prec(f)$ the initial term of f with respect to $\prec$.
- $\langle I_\prec \rangle$ the ideals generated by $\{f_\prec : f \in I\}$
- $V(I) = \{u_0 \in K^n : f(u_0) = 0, \text{ for all } f \text{ in } I\}$ the *variety* of I.

Definition 3: Let $f, g, h \in K[X], F \subset K[X]$. We say that $g \neq 0$ is reducible by $f \neq 0$ to h, denoted by $g \to_f h$, iff there are power products $s, t \in [X]$ such that s has a non-vanishing coefficient c in g, $s = \mathrm{lpp}_\prec(f) \cdot t$, and $h = g - \frac{c}{\mathrm{lc}_\prec(f)} \cdot t \cdot f$. We say that g is *reducible* with respect to F iff there exists a sequence of polynomials $f_1, f_2, \ldots f_k \in F$ such that $g \to_{f_1} \to_{f_2} \cdots \to_{f_k} g'$ for some g' in $K[X]$. If g' is not reducible, we say that g' is a *normal form* of g with respect to F.

Let G be a finite subset of $K[X] \setminus \{0\}$, $\prec$ be a term order on $[X]$, and $I \supseteq G$ be an ideal in $K[X]$. Then G is a Groebner basis of I with respect to $\prec$ iff $\langle G_\prec \rangle = \langle I_\prec \rangle$. Furthermore, G is called a minimal Groebner basis iff $\mathrm{lpp}_\prec(f) \nmid \mathrm{lpp}_\prec(g)$ for all $f, g \in G, f \neq g$. G is called a reduced Groebner basis iff for all $f, g \in G, f \neq g$, f is not reducible by g. G is normed iff $\mathrm{lc}_\prec(g) = 1$, for all $g \in G$.

Theorem 1: [8] Every ideal I in $K[X]$ has a unique finite normed reduced Groebner basis.

Without loss of generality, from now on we assume that reduced Groebner bases are normed. Given an ideal I and a term order $\prec$, we denote the reduced Groebner basis of I with respect to $\prec$ by $\mathrm{GB}(I, \prec)$. The following lemma gives us many different ways to check whether or not a set of polynomials is a Groebner basis.

Lemma 2: Let I be an ideal in $K[X]$, $\prec$ a term order, $F \subset K[X]$, and $\langle F \rangle = I$. The following statements are equivalent:

1) F is a Groebner basis of I with respect to $\prec$.

2) g is reducible to 0 with respect to F, for all $g \in I \setminus \{0\}$.
3) g is reducible with respect to F, for all $g \in I \setminus \{0\}$.
4) $\rightarrow_F$ is a Church-Rosser reduction relation.

Let $f, g \in K[X]$, $t = \text{lcm}(\text{lpp}_\prec(f), \text{lpp}_\prec(g))$. $\text{cp}(f,g) = (\frac{t}{\text{lc}_\prec(f)\cdot\text{lpp}_\prec(f)} \cdot f, \frac{t}{\text{lc}_\prec(g)\cdot\text{lpp}_\prec(g)} \cdot g)$ is called the critical pair of f and g. The difference of the elements of the critical pair $\frac{t}{\text{lc}_\prec(f)\cdot\text{lpp}_\prec(f)} \cdot f - \frac{t}{\text{lc}_\prec(g)\cdot\text{lpp}_\prec(g)} \cdot g$, denoted by S-pol$(f,g)$, is called the S-polynomial of f and g.

Algorithm 1: Buchberger's algorithm [8].
Input: a finite set $F \subset K[X]$ and an order $\prec$.
Output: a Groebner basis G of F w.r.t. $\prec$.

Step-1 $G \leftarrow F$
$C \leftarrow \{\{g_1, g_2\} : g_1, g_2 \in G, g_1 \neq g_2\}$
Step-2 While not all pairs in C are marked
choose an unmarked pair $\{g_1, g_2\}$;
mark $\{g_1, g_2\}$;
$h \leftarrow$ normal form of S-pol(f,g) w.r.t. G ;
if $h \neq 0$ then
$C \leftarrow \{\{g, h\} : g \in G\}$;
$G \leftarrow G \cup \{h\}$;
return G.

For our work, we also need supplementary polynomials, which are the polynomials in the ideal with necessary monomials for reduction calculation.

Algorithm 2: Finding supplementary polynomials [15], [16]
Input: $f \in K[X]$ and $G \subset K[X]$
Output: the set of supplementary polynomials

- $M \leftarrow$ set of terms of f.
- While $M \neq \emptyset$
  - dequeue the highest term h in M
  - if $h = t \cdot lpp(g)$ for some term t and $g \in G$
    - add $t \cdot g$ into the set of supplementary polynomials
    - add the terms of $t \cdot g$ into M except the leading term

C. Related works

Algebraic reasoning such as Groebner basis-based or Hilbert's Nullstellensatz-based reasoning, has been used in propositional proof systems [3], [13]. For an appropriate measure of proof size, [13] showed that the Groebner basis algorithm finds a proof of a tautology in time polynomial in the size of the smallest such proof. Furthermore, the Groebner basis-based system polynomially simulates Horn clause resolution, and quasi-polynomially simulates resolution. On the other hand, there are simple tautologies that have polynomial-size Groebner proofs but require exponential-size resolution proofs. In comparison with Nullstellensatz-based system [3], [13] showed that there is a family of tautologies that have degree 3 Groebner refutations, but they require $\Theta(\sqrt{n})$ degree Nullstellensatz refutations, where n is the number of variables. Thus, there is an exponential separation between the Groebner basis-based and the Hilbert's Nullstellensatz-based systems.

Groebner bases considered as a Church-Rosser reduction relation or a term rewriting system has been used for propositional satisfiability in [17], [18]. Techniques from algebraic geometry have also been proposed for symbolic model checking in lieu of BDDs [2].

However, the obstacles for an *effective use* of these concepts for model checking are that: (1) In model checking, one has to deal with a large number of variables. (2) Intermediate polynomials may have a doubly exponential degree and require exponential space [20]. (3) Traditional work in Groebner basis computation is focused on general polynomials where special care must be given for the coefficient swell problem when the coefficients of intermediate polynomials will get very big. (4) All of the current implementations of Buchberger's algorithm for Groebner basis computation, which either based on conventional representation of polynomial or matrices [15], are not efficient enough for model checking.

III. Direct Groebner Bases Computation in Boolean Rings

In this section, we present an approach to overcome the obstacles just mentioned. We first present the data structures, techniques and algorithms for direct Groebner basis computation in Boolean rings where the degree of intermediate polynomials will never go up. It is well-known that polynomial reduction is the costliest part for any practical implementation of Buchberger's algorithm for Groebner basis computation. We follow Buchberger's algorithm in a novel way in that extremely efficient bit operations are used for polynomial reduction. Similar to Faugere's approach [15], [16] in that Gaussian elimination from linear algebra has been used for polynomial reductions in Buchberger's algorithm, we use bit operations for polynomial reduction. Such representations would presumably allow efficient model checking for different classes of systems than BDDs and BMC, and hence supplement existing symbolic model checking methods.

We illustrate this approach with one simple example. In this example we want to verify that $(p \vee q) \wedge (s \vee t) \wedge (\neg p \vee \neg t) \wedge (\neg p \vee \neg s) \wedge (\neg q \vee \neg t) \rightarrow (q \wedge s)$ is a tautology. Equivalently, we can show that any solution for the sub-formula on the left hand side of the implication is also a solution for the sub-formula on the right hand side. By converting the Boolean formula into its Boolean ring counterpart, we obtain six polynomials for the five disjunctions on the left hand side and the conjunction on the right hand side of the formula

(1)	$p.q + p + q + 1 = 0$
(2)	$s.t + s + t + 1 = 0$
(3)	$p.t = 0$
(4)	$p.s = 0$
(5)	$q.t = 0$
(6)	$q.s + 1 = 0$

From the theory of Groebner bases, we know that a Boolean formula is not satisfiable if and only if the Groebner basis of the polynomials has 1 as a member of the basis [8], [9]. As we will explain more in Section III-B that in our approach, validity checking can be translated into satisfiability checking easily.

We follow Buchberger's algorithm for calculating the Groebner bases using a term order, for example lexicographic term order with $p < q < s < t$. The reduced Groebner basis of the first five Boolean polynomials w.r.t. the chosen term order is $G = \{t, s + 1, q + 1, p\}$.

The crux of our novel approach is that we do not use the costly polynomial reductions during the calculation of the Groebner basis nor in the last polynomial reduction as usual. Instead, we take into account the specific structure of polynomials in Boolean rings by representing each polynomial as a binary number of length m, where m is the number of monomials of the involved polynomials and their supplementary polynomials.

The least significant digit corresponds to the smallest term and the most significant digit corresponds to the leading term. By using this representation, polynomial reduction can be done by fast binary operations. For example, the reduction of polynomial (6) with respect to G is performed as follows:

Polynomial	Binary representation
(6)	101
supplementary polynomial	110
$(q+1)$	011
result (using XOR)	000

In this example, the only supplementary polynomial is $q \cdot (s+1) = q \cdot s + q$. By performing just only one XOR bit operation on the binary representations, we obtain 0. The result means that the formula is a tautology. Besides verifying whether a system satisfies a formula, we can also enumerate all solutions for the formula. For example, all solutions for the example can be immediately found from a Groebner basis computation with respect to an elimination term order. The reduced Groebner basis $\{t, s+1, q+1, p\}$ has all solutions for the system. When we use a term order for elimination such as lexicographic term order, the Groebner basis will have a very special form in that the polynomial will be guaranteed in a triangular shape. This triangular form give us the set of all solutions by assigning and substituting values of one variable at a time.

Since model checking problems usually involves hundred of variables and requires heavy memory usage, we need extremely fast basic operations for Boolean polynomials. To accommodate this special requirement, we use *dense representation* to represent a multivariate polynomial as an array of bit vectors of size n, where n is the number of variables. This representation allows us to reduce polynomial operations to extremely fast bit operations. Our special interest is for the multiplication of a monomial with a polynomial. For example, dense representation data structure is used for monomial s and polynomial $(s{\cdot}t{+}t{+}1)$ with respect to the total degree order with $p < q < s < t$. In this manner, the polynomials are represented as a list of binary numbers where each bit corresponds to a variable (with an exception for the constant) following the order of the variables:

s: | 0 | 1 | 0 | 0 |

$s \cdot t + t + 1$: | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |

Multiplication of monomial s with polynomial $(s \cdot t + t + 1)$ can be done by performing OR bit operations of the binary number for s with each term of the polynomial $s \cdot t + t + 1$ obtaining $s \times (s{\cdot}t{+}t{+}1)$. Notice that the degree of polynomials will never go up.

1	1	0	0	1	1	0	0	0	1	0	0

After removing the duplicates, the result is

s: | 0 | 1 | 0 | 0 |

In the next sections, we will show how all operations required for model checking can be implemented by means of Groebner bases.

A. Groebner Bases for Symbolic Model Checking

In Section II we showed how Boolean formulas are translated into polynomials over the corresponding Boolean ring. During the construction of polynomials for a formula, efficient operations on sub-systems such as union, intersection and complement are required. Fortunately, such corresponding operations on Boolean polynomials are much easier:

Proposition 1: [Basic Operations]

1) Complementation: Given a set of Boolean polynomials $I = \{f_1, f_2, \ldots, f_i, \ldots, f_n\} \subset K[X]$, a set of polynomials for the complement $\overline{V(I)} = K[X] \setminus V(I)$ is $\prod_{i=1}^{n} I_i$, where $I_i = \{f_1, f_2, \ldots, f_i + 1, \ldots, f_n\}$.
2) Union: Given two sets of Boolean polynomials $I, J \subset K[X]$, a set of polynomials for $V(I) \cup V(J)$ is $I \cdot J = \{f \cdot g : f \in I, g \in J\}$. This operation can be done in $O(n^2)$, where $n = \max(|I|, |J|)$.
3) Intersection: Given two sets of Boolean polynomials $I, J \subset K[X]$, a set of polynomials for $V(I) \cap V(J)$ is $I \cup J = \{f : f \in I \vee f \in J\}$. This operation can be done in $O(n)$, where $n = \max(|I|, |J|)$.

Proof: $\alpha \notin V(I)$ iff $\exists f \in I$ s.t. $f(\alpha) \neq 0 \Leftrightarrow \exists f \in I$ s.t. $f(\alpha)+1 = 0$. Therefore, $\overline{V(I)} = \cup_{i=1}^{n} V(I_i)$. This result shows that in the Boolean ring setting the complement of a variety is a variety. Clearly, for the purpose of validity checking, one does not want to multiply the ideals together for the cost of $O(n^n)$. Since $\overline{V(I)} = \cup_{i=1}^{n} V(I_i) = \emptyset$ if and only if $V(I_i) = \emptyset$ for all $i = 1..n$, checking each of the varieties can be done with many calculations being repeated. ■

We now explain how to translate temporal formulas such as $\mathbf{EX}\phi$ into polynomial systems.

Image computation with respect to a subset of states C is a central operation for reachability in symbolic model checking. Although the initial set C and the image are of acceptable size, the sizes of the intermediate BDDs during the computation may explode. Let $T \subset K^n \times K^n$ be the transition relation of the system, where a point $(b_1, \ldots, b_n, b_1^{'}, \ldots, b_n^{'}) \in T$ if and only if there is a transition from the state represented by $(b_1, \ldots, b_n)$ to the state represented by $(b_1^{'}, \ldots, b_n^{'})$. Let $I = \langle \{f_1, f_2, \ldots, f_k\} \rangle \subset K[x_1, \ldots, x_n, x_1^{'}, \ldots, x_n^{'}]$ be the ideal generated by the set of solutions $\{(b_1, \ldots, b_n, b_1^{'}, \ldots, b_n^{'}) \in T : (b_1, \ldots, b_n) \in C\}$ using the results in Proposition 1. The image operation can be done as follows:

Algorithm 3: [Image]
Given: a transition relation T, a set $C \subset K^n$ and the ideal I of polynomials w.r.t. T and C.
Find: $\{b' \in K^n : \exists b \in C, (b, b') \in T\}$

Step-1 build a Groebner basis G_I for the ideal I with respect to an efficient term order such as the total degree term order.
Step-2 Convert G_I into a Groebner basis G_I' for the ideal I with respect to an elimination term order, where $x_1 \succ \ldots \succ x_n \succ x_1^{'} \succ \ldots \succ x_n^{'}$.
Step-3 return $G_I' \cap K[x_1^{'}, \ldots, x_n^{'}]$.

Pre-image computation with respect to a subset of states C is another central operation in symbolic model checking for formulas such as $\mathbf{EX}\phi$. Let $I = \langle \{f_1, f_2, \ldots, f_k\} \rangle \subset K[x_1, \ldots, x_n, x_1^{'}, \ldots, x_n^{'}]$ be the ideal generated by the set of solutions $\{(b_1, \ldots, b_n, b_1^{'}, \ldots, b_n^{'}) \in T : (b_1', \ldots, b_n') \in C\}$. The pre-image operation can be done as follows:

Algorithm 4: [Pre-Image]
Given: a transition relation T, a set $C \subset K^n$ and the ideal I of polynomials w.r.t. T and C.
Find: $\{b \in K^n : \exists b' \in C, (b, b') \in T\}$

Step-1 build a Groebner basis G_I for the ideal I with respect to an efficient term order.

Step-2 Convert G_I into a Groebner basis G'_I for the ideal I with respect to an elimination term order, where $x_1 \prec \ldots \prec x_n \prec x_1^{'} \prec \ldots \prec x_n^{'}$.

Step-3 return $G'_I \cap K[x_1, \ldots, x_n]$.

Fix-point computation for CTL formulas such as $\mathbf{EF}f = \mu Z.(f \vee \mathbf{EX}Z)$ can be translated using Algorithm III-A for pre-image computation and the quantification operators in QBF $\exists x, f \equiv f|_{x \leftarrow 0} \vee f|_{x \leftarrow 1}$.

Since the two ideals are the same if and only if their Groebner bases w.r.t. a term order are the same, the convergence of fix-point computation can be detected.

To illustrate the main ideas, we use an example in [2] in that one has to verify properties of a mutual exclusion protocol. The protocol is defined as follows:

$t \leftarrow 0$;

process p_0	**process** p_1
$s_0 \leftarrow$nc;	$s_1 \leftarrow$nc;
while 1	while 1
$t' \leftarrow (t = 0 \wedge s_0 =$c?$\neg t$: $t)$	$t' \leftarrow (t = 1 \wedge s_1 =$c?$\neg t$: $t)$
$s'_0 \leftarrow$(case	$s'_1 \leftarrow$(case
s_0=nc: {r, nc},	s_1=nc: {r, nc},
s_0=r $\wedge$ s_1=nc: c,	s_1=r $\wedge$ s_0=nc: c,
s_0=r$\wedge s_1$=r$\wedge t = 0$: c,	s_1=r$\wedge s_0$=r$\wedge t = 0$: c,
s_0=c: {c, nc}	s_1=c: {c, nc}
default: s_0);	default: s_1);
$t \leftarrow t'$;	$t \leftarrow t'$;
$s_0 \leftarrow s'_0$;	$s_1 \leftarrow s'_1$;

To describe the system, we use one variable x_1 for t, two variables x_2, x_3 for s_0, two variables x_4, x_5 for s_1, and one variable x_6 for keeping track of the running process. We encode the enumerated variables s_0 and s_1 by setting the corresponding pair of bits to $(0, 0)$ for nc, $(0, 1)$ for r, and $(1, 0)$ for c.

The transition relation T can be constructed based on the assignments made by processes p_0 and p_1. The assignments for each process can be represented by a set of polynomials as

$I_{p_0} = \langle (x_6+1)(x_2+1)(x_3+1)(x_2^{'}+1)+1,\ (x_6+1)(x_2+1)x_3(x_4+1)(x_5+1)x_2^{'}(x_3^{'}+1)+1,\ (x_6+1)(x_2+1)x_3(x_4+1)x_5(x_1+1)x_2^{'}(x_3^{'}+1)+1,\ (x_6+1)x_2(x_3+1)(x_3^{'}+1)+1,\ (x_6+1)(x_2+1)x_3(x_4+1)x_5x_1(x_2^{'}+1)x_3^{'}+1,\ (x_6+1)(x_2+1)x_3x_4(x_5+1)(x_2^{'}+1)x_3^{'}+1,\ (x_6+1)(x_1+1)x_2(x_3+1)x_1^{'}+1,\ (x_6+1)((x_1+1)x_2(x_3+1)+1)(x_1+x_1^{'}+1)+1,\ (x_6+1)(x_4+x_4^{'}+1)+1, (x_5+x_5^{'}+1)+1\rangle$.

$I_{p_1} = \langle (x_6)(x_2+1)(x_3+1)(x_2^{'}+1)+1,\ (x_6)(x_2+1)x_3(x_4+1)(x_5+1)x_2^{'}(x_3^{'}+1)+1,\ (x_6)(x_2+1)x_3(x_4+1)x_5(x_1+1)x_2^{'}(x_3^{'}+1)+1,\ (x_6)x_2(x_3+1)(x_3^{'}+1)+1,\ (x_6)(x_2+1)x_3(x_4+1)x_5x_1(x_2^{'}+1)x_3^{'}+1,\ (x_6)(x_2+1)x_3x_4(x_5+1)(x_2^{'}+1)x_3^{'}+1,\ (x_6)(x_1+1)x_2(x_3+1)x_1^{'}+1,\ (x_6)((x_1+1)x_2(x_3+1)+1)(x_1+x_1^{'}+1)+1,\ (x_6)(x_4+x_4^{'}+1)+1, (x_5+x_5^{'}+1)+1\rangle$.

The property we want to check is $\mathbf{EF}f$, where $f \equiv (s_0 =$c$) \wedge (s_1 =$c$)$. The temporal formula can be translated into the least fixed point of $\mu y.f \vee \mathbf{EX}y$ where f can be represented by $I_f = \langle x_2(x_3+1)+1,\ x_4(x_5+1)+1\rangle$. Groebner basis computation found the least fixed point of $\lambda y.f \vee \mathbf{EX}y$ as $I_{\mathbf{EF}f} = \langle x_2+1, x_3, x_4+1, x_5\rangle$. The initial condition can be represented by $I_{init} = \langle x_2, x_3, x_4, x_5, x_6\rangle$. Another simple Groebner basis computation for $V(I_{\mathbf{EF}f}) \cap V(I_{init})$ show that the constant polynomial 1 is the Groebner basis. That means $\mathbf{EF}f$ is false in the initial states.

B. Groebner Bases for LTL Model Checking

Given an LTL formula such as $\varphi = \mathbf{E}\phi$ and a Kripke structure M, we construct a Groebner basis whose solutions are exactly the states satisfying the temporal formula as follows: We construct a Büchi automaton T for the path formula ϕ. T is a Kripke structure and includes every path that satisfies ϕ. By composing T with M, we find a set of paths that appear in both T and M. A state in M will satisfy $\mathbf{E}\phi$ if and only if it is the start of a path in the composition that satisfies ϕ. The procedures in Section III-A and [14] are used to find these states.

C. Groebner Bases for Bounded Model Checking

A bounded model checker restricts the general model checking problem to a bounded one. Instead of checking whether the system M violate a property ψ, we check whether the system M has any counterexample of length k for ψ. Let $\pi = s_0.s_1 \ldots$ be an infinite path. We say that π is a (k, l)-loop if $\pi = s_0.s_1 \ldots s_{l-1}\ (s_l \ldots s_k)^{\omega}$. For finite state systems, all LTL counterexamples can be captured by a finite (k, l)-loop [32]. Using the operators defined in the previous sections, one can construct a Groebner basis whose solutions are exactly the states satisfying the formula $\varphi = I(s_0) \wedge \wedge_{i=1}^{k} T(s_{i-1}, s_i) \wedge |[\neg\psi]|_k$, where $I(s_0)$ is the predicate for initial states, $T(s_{i-1}, s_i)$ is the transition relations and $|[\neg\psi]|_k$ is the formula constraint [4], [5].

Algorithm 5: [Unsatisfiability Checking]

Given: a system M, an LTL formula ψ and a bound k .

Verify: M has no counterexample of length k for ψ.

Step-1 build a Groebner basis G_φ for the formula φ with respect to an efficient term order.

Step-2 check if $V(G_\varphi) = \emptyset$ or $1 \in G_\varphi$.

Enumerating all counterexamples can be done in our new approach by converting G_φ into a Groebner basis G'_φ for the formula φ with respect to an elimination term order. The reduced Groebner basis G'_φ has all counterexamples for the system. In fact, there are many results from symbolic computation for efficient conversion of Groebner bases from one term order to the other [30]. When we use an elimination term order such as lexicographic term order, the Groebner basis will have a very special form in that the polynomial will be guaranteed in a triangular shape. This triangular form give us the set of all solutions by assigning and substituting values of one variable at a time.

D. Discussion and Complexity

Symbolic computation is an active and rich area with enormous activity and progress in the last twenty years. An approach to symbolic model checking making use of results from symbolic computation seems to have considerable promise, both as a supplement to existing methods and as a way to bring a large body of powerful mathematical machinery to bear on the model checking problems. Our approach has many advantages as follows:

1) Once the Groebner basis of a system is calculated, many properties can be checked without recalculating the basis again.
2) In contrast to the case of BDD-based and SAT-based model checking methods, we know how to convert a Groebner basis representation from one order of variables into the other order efficiently. This is a very important feature of our approach [29], [30].

3) The method works for both model checking and bounded model checking (BMC).
4) The method can check for unsatisfiability and satisfiability.
5) Polynomial reductions are replaced by extremely efficient binary bit operations.

An alternative representation for symbolic model checking might lead to new theoretical insights into the practicality and guidance for choosing methods for verifying properties of the systems. Recently, we have been able to prove that the complexity of algorithms for Groebner basis computation in Boolean rings is in PSPACE.

REFERENCES

[1] S. B. Akers. Binary decision diagrams. *IEEE Transactions on Computers*, 1978.

[2] G. S. Avrunin. Symbolic model checking using algebraic geometry. In *CAV '96: Proceedings of the 8th International Conference on Computer Aided Verification*, pages 26–37, London, UK, 1996. Springer-Verlag.

[3] P. Beame, R. Impagliazzo, J. Krajicek, T. Pitassi, and P. Pudlak. Lower bounds on Hilbert's nullstellansatz and propositional proofs. In *Proceddings of the 35th Annual Symposium on Foundations of Computer Science*, pages 794–806, Santa Fe, NM, 1994.

[4] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu. Symbolic model checking without bdds. In *Proceedings of TACAS*, volume 1579 of *LNCS*. Springer, 1999.

[5] A. Biere, E. M. Clarke, R. Raimi, and Y. Zhu. Verifiying safety properties of a power pc microprocessor using symbolic model checking without bdds. In N. Halbwachs and D. Peled, editors, *Proceedings of the 11th International Conference on Computer Aided Verification, CAV'99*, volume 1633 of *LNCS*, pages 60–71, Trento, Italy, 1999.

[6] P. Bjesse, T. Leonard, and A. Mokkedem. Finding bugs in an alpha microprocessor using satisfiability solvers. In *Proceedings of the 13th International Conference on Computer Aided Verification, CAV 2001*, number 2102 in LNCS, pages 454–464, Paris, France, 2001. Springer.

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

[8] B. Buchberger. *An Algorithm for Finding a Basis for the Residue Class Ring of a Zero-dimensional Polynomial Ideal (in German)*. PhD thesis, Institute of Mathematics, Univ. Innsbruck, Innsbruck, Austria, 1965.

[9] B. Buchberger. Groebner Bases: An algorithmic method in polynomial ideal theory. In N. K. Bose, editor, *Multidimensional Systems Theory*, chapter 6, pages 184–232. Reidel Publishing Company, Dodrecht, 1985.

[10] S. N. Burris and H. P. Sankappanavar. *A Course in Universal Algebra*. Springer-Verlag, 1981.

[11] E. Clarke and E. Emerson. Design and synthesis of syncronization of skeletons using branching time temporal logic. In *Proceedings of the IBM workshop on logics of programs*, volume 131 of *LNCS*, pages 52–71. Springer, 1981.

[12] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, 1999.

[13] M. Clegg, J. Edmonds, and R. Impagliazzo. Using Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of STOC'96*, pages 174–183, Philadelphia, PA, USA, 1996. ACM.

[14] E. Emerson and C.-L. Lei. Efficient model checking in fragments of the propositional μ-calculus. pages 267–278, 1986.

[15] J. Faugere. A new efficient algorithm for computing groebner bases without reduction to zero (f5). In *Proceedings of ISSAC'02*, 2002.

[16] J. Faugere. Efficient algorithms for computing groebner bases. Talk at Workshop B2: Efficient Computation of Groebner Bases, Special Semester on Groebner Bases and Related Methods, 2006.

[17] J. Hsiang. Refutational theorem proving using term rewriting systems. *Artificial Intelligence*, 25:255–300, 1985.

[18] G. Huang, N. Dershowitz, and J. Hsiang. Satisfiability testing using simplification in boolean rings.

[19] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic Model Checking: 10^{20} States and Beyond. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 1–33, Washington, D.C., 1990. IEEE Computer Society Press.

[20] K. Kuehnle and E. Mayr. Exponential space computation of groebner bases. In *Proceedings of ISSAC*. ACM, 1996.

[21] O. Kupferman, M. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.

[22] C. Lee. Representation of switching circuits by binary decision programs. *Bell System Technical Journal*, 38:985–999, 1959.

[23] K. L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[24] K. Namjoshi. Certifying model checker. In *Proceedings of the 13th International Conference on Computer-Aided Verification*, volume 2102 of *LNCS*, pages 2–13. Springer, 2001.

[25] D. Peled and L. Zuck. From model checking to a temporal proof. In *Proceedings of the 8th International SPIN Workshop*, volume 2057 of *LNCS*, pages 1–14, Toronto, Canada, 2001. Springer.

[26] A. Pnueli. The temporal logic of programs. In *Proceedings of 18th IEEE Symposium on Foundation of Computer Science*, pages 46–57, 1977.

[27] J. Quille and J. Sifakis. Specification and verification of concurrent systems in cesar. In *Proceedings of the 5th International Symposium on Programming*, page 1981, 337–350.

[28] M. Stone. The theory of representation for boolean algebras. *Trans. Amer. Math. Soc.*, 1936.

[29] Q.-N. Tran. A fast algorithm for Groebner basis conversion and its applications. *Journal of Symbolic Computation*, 30:451–468, 2000.

[30] Q.-N. Tran. A new class of term orders for elimination and applications. *Journal of Symbolic Computation*, 42, 2007.

[31] A. Valmari. The state explosion problem. In W. Reisig and G. Rozenberg, editors, *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets*, volume 1491 of *LNCS*, pages 429–528. Springer, 1998.

[32] M. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Science*, 32(2):182–221, 1986.

[33] F. Winkler. *Polynomial Algorithms in Computer Algebra*. Texts and Monographs in Symbolic Computation. Springer-Verlag, 1996.