## Article

# A New Approach to Nonlinear Invariants for Hybrid Systems Based on the Citing Instances Method 

Honghui $\mathrm{He}^{\mathbf{1 , 2 , 3 , 4}}$ and Jinzhao $\mathrm{Wu}^{\mathbf{1 , 2 , 3 , *}}$<br>1 Chengdu Institute of Computer Application, Chinese Academy of Sciences, Chengdu 610041, China; HongHuiHe@vip.henu.edu.cn<br>2 University of Chinese Academy of Sciences, Beijing 100049, China<br>3 Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Guangxi University for Nationalities, Nanning 53006, China<br>4 School of Computer Software, Henan University, Kaifeng 475001, China<br>* Correspondence: wujinzhao@gxun.edu.cn

Received: 20 March 2020; Accepted: 9 April 2020; Published: 2 May 2020


#### Abstract

In generating invariants for hybrid systems, a main source of intractability is that transition relations are first-order assertions over current-state variables and next-state variables, which doubles the number of system variables and introduces many more free variables. The more variables, the less tractability and, hence, solving the algebraic constraints on complete inductive conditions by a comprehensive Gröbner basis is very expensive. To address this issue, this paper presents a new, complete method, called the Citing Instances Method (CIM), which can eliminate the free variables and directly solve for the complete inductive conditions. An instance means the verification of a proposition after instantiating free variables to numbers. A lattice array is a key notion in this paper, which is essentially a finite set of instances. Verifying that a proposition holds over a Lattice Array suffices to prove that the proposition holds in general; this interesting feature inspires us to present CIM. On one hand, instead of computing a comprehensive Gröbner basis, CIM uses a Lattice Array to generate the constraints in parallel. On the other hand, we can make a clever use of the parallelism of CIM to start with some constraint equations which can be solved easily, in order to determine some parameters in an early state. These solved parameters benefit the solution of the rest of the constraint equations; this process is similar to the domino effect. Therefore, the constraint-solving tractability of the proposed method is strong. We show that some existing approaches are only special cases of our method. Moreover, it turns out CIM is more efficient than existing approaches under parallel circumstances. Some examples are presented to illustrate the practicality of our method.


Keywords: citing instances method; comprehensive Gröbner basis; hybrid system; lattice array; invariant

## 1. Introduction

In the real world, there exist many systems exhibiting mixed discrete-continuous behavior, which cannot be described in a proper way by using either a discrete or continuous model. The notion of a hybrid automaton has been introduced for modeling such systems [1]. For example, embedded systems are often modeled as hybrid systems due to their involvement in both digital control software and analog plants, the physical process of which is often specified in the form of differential equations.

Safety verification is among the most challenging problems in verifying hybrid systems, which consists of asking whether a set of bad (unsafe) states can be reached from a set of initial states. The safety verification problem for systems described by non-linear differential equations is particularly complicated, as computing the exact reachable set is usually intractable [2-4], with the
exception of some severely restricted sub-classes, such as timed automata [5] and initialized rectangular automata [6].

However, for purposes of verification of safety properties, it often suffices to compute an over-approximation of the reachable set of states-if the over-approximation does not intersect the set of bad states, then the original system will never reach a bad state. So far, the existing approaches are mainly based on approximate reachable set computations [7-9] and abstraction [10-14].

An over-approximation of the reachable states is also called an invariant of the system. The most precise invariant of a system is its exact reach set. The standard technique for proving a safety property $\varphi$ is to generate an inductive invariant $\psi$ that implies $\varphi$. Moreover, invariants have the benefit of avoiding computing the exact reachable set of hybrid systems and are very useful for hybrid systems which are described by non-linear differential equations which cannot be solved analytically.

Some typical techniques for invariants are based on templates, which are used to search for inductive invariant assertions with standard computational techniques in algebraic geometry involving Gröbner basis [3,15]. The intuitive idea behind these techniques is that of first fixing a template assertion (i.e., a parametric polynomial with unknown coefficients and of bounded degree in the system variables) and encoding inductive conditions into constraints on unknown coefficients, such that any solution to these constraints is an inductive assertion. This technique guarantees that the invariant must be found if the system has it in that form. However, the key challenges for these techniques is how to define an inductive condition with completeness and how to efficiently compute an inductive invariant that satisfies the inductive condition. Usually, these two aspects contradict each other; that is, an inductive condition with completeness often encounters the computability or complexity problem. Sankaranarayanan et al.'s approach [3], for example, makes significant strides in generating non-linear polynomial equations as invariants of a hybrid system. Their approach consists of the following four main steps:

1. Guessing a template polynomial of fixed degree as a candidate inductive invariant.
2. Defining complete inductive conditions.
3. Utilizing a comprehensive Gröbner basis to encode complete inductive conditions into constraints on parameters and, therefore, reducing the invariant generation problem to a non-linear constraint solving problem.
4. Solving constraints on parameters. Any solution with respect to the constraints is an inductive invariant.

In order to encode the complete inductive conditions, a comprehensive Gröbner basis is involved, which is an exact but impractical encoding as the construction of the comprehensive Gröbner basis is very expensive. Hence, Sankaranarayanan et al. resorted to more tractable (but incomplete) inductive conditions (as shown in Tables 7 and 8) to avoid the explicit construction of a comprehensive Gröbner basis. However, in doing so, their approach sacrifices completeness [3]. The exact steps of [3] go as follows.

1. Guessing a template polynomial of fixed degree as candidate inductive invariant.
2. Defining alternative inductive conditions-which are incomplete but more tractable-to take the place of complete inductive conditions.
3. Utilizing a Gröbner basis to encode the alternative inductive conditions into constraints on parameters.
4. Solving the constraints. Any solution with respect to the constraints is an inductive invariant.

To address this issue, instead of computing a comprehensive Gröbner basis, this paper presents a new technique, called Citing Instances Method (CIM). By substituting real numbers for the variables in the complete inductive conditions, CIM can easily derive a set of constraints on the parameters, which can be solved more efficiently by the elimination of free variables. This idea is inspired by the Parallel Numerical Method, which was first presented in [16,17]. The paper showed that a theorem
can be proved by carrying out a series of numerical verifications of one certain instance. Their method was firstly used to prove geometric theorems; furthermore, many classic theorems, such as Simson's Theorem and Ptolemy's Theorem, were proved using this technique. Differing from the parallel numerical method, CIM is used to generate constraints on parameters. For example, it is well-known that there are many methods to generate a constraint on $\{a, b\}$ which guarantees that the identity below holds,

$$
\begin{equation*}
x^{3}-4 x^{2}+2 x+1-(x-1)\left(x^{2}+a x+b\right) \equiv 0 \tag{1}
\end{equation*}
$$

1. Zero Polynomial Theorem [3]: $x^{3}-4 x^{2}+2 x+1-(x-1)\left(x^{2}+a x+b\right) \Leftrightarrow x^{3}-4 x^{2}+2 x+1-$ $\left(x^{3}+(a-1) x^{2}+(b-a) x-b\right) \Leftrightarrow-(a+3) x^{2}+(2-b+a) x+1+b$. By the Zero Polynomial Theorem, three constraint equations are obtained synchronously, $\{a+3=0,2-b+a=0,1+b=$ $0\}$, solving $\{b=-1, a=-3\}$.
2. CIM (this paper): Take four arbitrary distinct values $\{-1,0,1,2\}$ of $x$ and substitute them for $x$ in the left side of (1), respectively. Then, equate the left side of (1) to zero (e.g., when $x=0$, then $1+b=0$ is obtained). Hence, four constraint equations can be derived independently: $\{-4-2 a+2 b=0,1+b=0,0=0,-7-2 a-b=0\}$. Solving, we get $\{b=-1, a=-3\}$. Here is an intuitive explanation of CIM: when $\{b=-1, a=-3\}$ and the degree of the polynomial in the left side of (1) is 3 , the equation must have at most three roots. However, the left side of (1) vanishes over $\{-1,0,1,2\}$, which means that the equation (1) has at least four roots. Therefore, the equation (1) must be identically equal to zero .

In this paper, by an instance, we mean a verification of the proposition by substituting numbers for the variables in the proposition. CIM exhibits two interesting features: (1) the constraint equations can be obtained in parallel. As a result, we can make hla clever use of this feature to speed up our computation. In fact, some special assignments to variables tend to generate constraints easily and some parameters may be determined in the early stage. For example, $x=0 \Rightarrow b=-1$ in (1): this solution can be used to simplify the rest of the constraint equations on the parameters. Thus, solving constraint equations becomes easier. (2) By substituting real values for the variables, all or some of the variables are, in fact, eliminated by an instance. The main source of the intractability of hybrid systems is that transition relations are first-order assertions on current-state variables and next-state variables, which doubles the number of system variables and may introduce many more free variables and encoding inductive conditions by a comprehensive Gröbner basis will be very expensive. In contrast, CIM can be used to eliminate these free variables. Generally, to automatically discover invariants, the more variables, the less tractability.

Inspired by [16,17], this paper presents a new and complete algorithm for invariant generation in hybrid systems. A lattice array, which is a key notion introduced in this paper and essentially a finite set of instances, will be directly applied to the complete inductive conditions to generate the constraint on parameters, and any solution to the constraint produces an inductive assertion. The main idea of our approach is sketched as follows:

1. Guessing a template of a fixed degree as an invariant template.
2. Defining the complete inductive conditions.
3. Applying a lattice array to the complete inductive conditions to generate the constraint on the parameters.
4. Solving the constraint. Any solution to the constraint guarantees the template an inductive invariant.

The main contributions of this paper are as follows: Firstly, we propose a complete method for constructing invariants of hybrid systems which can solve the complete inductive conditions rather than alternative incomplete inductive conditions. Secondly, CIM takes the place of a comprehensive Gröbner basis in the invariant generation process, and the tractability of constraint-solving is stronger. On one hand, CIM utilizes instances to eliminate the free variables (fewer variables leads to higher
tractability). On the other hand, the generation of constraint equations by CIM is parallelized by their independence from each other. Therefore, we can make a clever use of the parallelizability of CIM to start with constraint equations generated by special instances and spread the solutions to some parameters determined in the early stage to other constraint equations, which hence simplifies the constraint equations obtained; this whole process works a bit like the Domino Effect. Therefore, the tractability of the constraint-solving is stronger than comprehensive Gröbner basis computation. Thirdly, CIM involves less symbolic computation and that is why it requires fewer computational resources.

## Related Work

Recently, many researchers have devoted effort towards finding the non-linear invariants of hybrid systems. Based on the theory of ideals and Gröbner bases, Sankaranarayanan et al. [3] presented an approach for generating polynomial equation invariants for hybrid systems with more general (non-linear) polynomial dynamics. To control the complexity of the constraint solving, however, this method has to make a trade-off between the complexity of the invariant generation process and the strength of the resulting invariants and several stronger conditions replace the complete conditions. Differing from Sankaranarayanan et al.'s approach, the paper [18] presented a complete method that was not based on guessing a template. However, it is only complete for linear systems. Without resorting to Gröbner bases, the paper [19] implemented the promising algorithm Fastind. Although its use is limited to a discrete system, Fastind executes significantly faster than implementations using Gröbner bases. Fastind is based on remainder computations over parameterized polynomials, and is still an incomplete method. Not coming singly but in pairs, Kong et al. [15] proposed an approach to automatically generate invariant clusters for semi-algebraic hybrid systems by computing the remainder of the Lie derivative of a template polynomial with respect to its Gröbner basis. The benefit of invariant clusters is that they can precisely overapproximate trajectories of the system. Another approach considered barrier certificates based on different inductive conditions [4,20-22] which can be solved efficiently by sum-of-squares (SOS) programming. The zero level set of barrier certificates forms the boundary of the reach set of a hybrid system and, hence, is an invariant. However, this approach is limited by the conservative inductive condition. On the whole, the verification problem of hybrid systems is undecidable: it is doomed to be impossible to find a universal approach for all hybrid systems. This implies that various inductive invariants and computational methods can be proposed for different classes of hybrid systems with some simplification or restriction. Some other approaches, which focus on different features of systems, have also been proposed for the construction of inductive invariants [23-27].

The paper is organized as follows. Section 2 is devoted to details about the modeling framework and some elementary lemmas on which CIM is based. We introduce the theory of the proposed CIM and describe the algorithms in Section 3. Constraint generation by CIM and techniques for solving these constraint equations are discussed in Section 4. In Section 5, we show that the approach of [3] is a special case of CIM. In Section 6, the technique is illustrated with a few examples. Finally, Section 8 concludes our work and discusses the future work in this direction.

## 2. Preliminaries

In this section, we introduce the lemmas on which CIM is based, as well as presenting our computational model of algebraic hybrid systems. First, we clarify some notation used throughout the paper.

We denote by $\mathbb{K}\left[x_{1}, \cdots, x_{n}\right]$ the polynomial ring in $n$ indeterminates $\left\{x_{1}, \cdots, x_{n}\right\}$ over the field $\mathbb{K}$. For conciseness, we also use boldface lowercase letters to denote vectors throughout the paper (e.g., $\left.\vec{x}=\boldsymbol{x}=\left(x_{1}, \cdots, x_{n}\right)\right)$. If $n=1, \mathbb{K}[\boldsymbol{x}]$ is called a univariate polynomial ring; it is called a multivariate polynomial ring when $n>1$. Let the variables be ordered as $x_{1} \prec x_{2} \prec \cdots \prec x_{n}$. Moreover, we assume $\mathbb{K}=\mathbb{R}$ throughout the paper unless otherwise specified.

For a multivariate polynomial ring, $\mathbb{K}[x]=\mathbb{K}\left[x_{1}, \cdots, x_{k-1}, x_{k}, x_{k+1}, \cdots, x_{n}\right]=$ $\mathbb{K}\left[x_{1}, \cdots, x_{k-1}, x_{k+1}, \cdots, x_{n}\right]\left[x_{k}\right]$. This means that a multivariate polynomial $p \in \mathbb{K}[\boldsymbol{x}]$ can be written as a univariate polynomial; that is, $p=\sum a_{i_{1} i_{2} \ldots i_{n}} x_{1}^{i_{1}} \ldots x_{n}^{i_{n}}=c_{m} x_{k}^{m}+c_{m-1} x_{k}^{m-1}+\cdots+c_{0}=\sum_{i=0}^{m} c_{i} x_{k}^{i}$, where $c_{i} \in \mathbb{K}\left[x_{1}, \cdots, x_{k-1}, x_{k+1}, \cdots, x_{n}\right], m=\operatorname{deg}\left(p, x_{k}\right)$. For example, $p\left(x_{1}, x_{2}, x_{3}, x_{4}\right)=x_{2}^{5}+$ $x_{3}^{4} x_{4}^{2}+\left(2 x_{2}+x_{1}\right) x_{4}^{3}$ can be considered as a univariate polynomial in $x_{2}$, namely $p\left(x_{1}, x_{2}, x_{3}, x_{4}\right)=$ $x_{2}^{5}+2 x_{4}^{3} x_{2}+x_{1} x_{4}^{3}+x_{3}^{4} x_{4}^{2}$.

An atomic algebraic assertion $\phi$ over $\mathbb{K}[x]$ has the form of $p\left(x_{1}, x_{2}, \cdots, x_{n}\right)=0$. An algebraic assertion is a conjunction of atomic algebraic assertions (i.e., $\bigwedge_{i} p_{i}\left(x_{1}, x_{2}, \cdots, x_{n}\right)=0$, where $\left.p_{i} \in \mathbb{K}[\boldsymbol{x}]\right)$. We denote by $\hat{x}$ an assignment of $x$.

### 2.1. Basic Lemmas

Lemma 1 ([16]). Let $f(x)$ and $g(x)$ be univariate polynomials of degree less than $n$. If there are $n+1$ distinct numbers $b_{0}, b_{1}, \cdots, b_{n} \in \mathbb{K}$ such that for every $b_{k}$ one has $f\left(b_{k}\right)=g\left(b_{k}\right)$ with $k=0,1, \cdots, n$. Then,

$$
\begin{equation*}
f(x) \equiv g(x) \tag{2}
\end{equation*}
$$

It is very easy to understand Lemma 1 . If $h(x)=f(x)-g(x)=0$, then $h(x)$ has $n+1$ roots at least. Meanwhile, $h(x)$ is a univariate equation, the degree of which is less than $n$ and which has $n$ roots at most. As $h(x)$ is not identical to zero, it contradicts the fact that $h(x)$ has at least $n+1$ roots. In this paper, we call the $n+1$ distinct numbers $\left\{b_{0}, b_{1}, \cdots, b_{n}\right\} n+1$ instances. Lemma 1 shows that, in order to prove that two univariate polynomials are identical, it suffices to test $n+1$ instances. Moreover, the $n+1$ instances can be chosen arbitrarily. Lemma 1 can be extended to multivariate polynomials. For this purpose, we need the following definition.

Definition 1 (Lattice Array [16]). Suppose $S_{1}, S_{2}, \ldots, S_{m} \subset \mathbb{K}$ such that $\left|S_{j}\right|=t_{j}$ with $j \in\{1, \cdots, m\}$, where $\left|S_{j}\right|$ denotes the size of $S_{j}$. We call the Cartesian product of the above $m$ subsets,

$$
S=S_{1} \times S_{2} \times \ldots \times S_{m}
$$

the m-dimensional lattice array on $\mathbb{K}$. Clearly, $S$ has $t_{1} t_{2} \ldots t_{m}$ elements.
For the convenience of discussion in what follows, we denote $|S|$ by $\left(t_{1}, t_{2}, \ldots, t_{m}\right)$.
Lemma 2 (Multi-Instances Numerical Verification [16]). Let $f\left(x_{1}, \cdots, x_{m}\right) \in \mathbb{K}\left[x_{1}, \cdots, x_{m}\right]$, $\operatorname{deg}\left(f, x_{k}\right)=n_{k}, n_{k} \in \mathbb{Z}^{+}, k=1,2, \cdots, m$. If there exists an m-dimensional lattice array $S$ of size $\left(n_{1}+1, n_{2}+1, \cdots, n_{m}+1\right)$ such that $f\left(x_{1}, \cdots, x_{m}\right)$ vanishes along $S$, i.e., for every $\left(\hat{x}_{1}, \hat{x}_{2}, \cdots, \hat{x}_{m}\right) \in$ $S, f\left(\hat{x}_{1}, \hat{x}_{2}, \cdots, \hat{x}_{m}\right)=0$, then $f$ is identically equal to zero.

In order to safely conclude whether a proposition is true or false, Multi-Instances Numerical Verification needs a finite set of instances to test it, which has requirements for both the numbers of instances and the relations between instances. The lattice array is defined for this purpose. We first assign variables freely; once variables are assigned, the instances are determined by Definition 1.

Example 1. Prove (3) by Lemma 2:

$$
\begin{equation*}
(x+y)(x-y) \equiv x^{2}-y^{2} \tag{3}
\end{equation*}
$$

1. Determining the size of the lattice array. By observation, the degrees of (3) in $x$ and $y$ are both less than or equal to 2 , such that the lattice array size is $(2+1,2+1)$.
2. Determining the members of the lattice array. Essentially, the variables $x, y$ can be assigned freely, so we can assign variables which are easy to test. For our convenience, let $x=0,1,2$ and $y=0,1,2$. Then, the nine instances in the lattice array are $(0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1)$, and $(2,2)$.
3. Substituting instances for $x$ and $y$ one by one, test whether the left side of (3) is equal to the right side of (3). If one of them leads to the left side of (3) differing from the right side of (3), it is a counterexample for (3) not being an identity. Otherwise, we can conclude that (3) is an identity.

Note: Arbitrarily taking nine points, such as $(0,0),(1,1),(3,2),(4,0),(7,1),(2,2),(9,10),(2,7)$, and $(8,9)$, can these nine points be used to safely prove (3)? Of course, the answer is no. The reason is that these nine points do not conform to the definition of a lattice array. It is easy to find a counterexample (see Figure 1): $x+2 y+3 z-7=14 x+9 y-z-15$ is an equation, rather than an equality; however, it is easy to find $(1+1,1+1,1+1)$ points (lying on the intersection) which satisfy $x+2 y+3 z-7$ (the left side) $=14 x+9 y-z-15$ (the right side).


Figure 1. Points lying on the intersection of two planes cannot consist of a lattice array.
Lemma 3 (Pseudo-Remainder Formula for Multivariate Polynomials [28]). Let $F, G \in \mathbb{K}\left[x_{1}, \ldots, x_{n}\right]$, and $x_{k}$ be a fixed variable, such that

$$
\begin{aligned}
& F=a_{m} x_{k}^{m}+a_{m-1} x_{k}^{m-1}+\ldots+a_{0} \\
& G=b_{\ell} x_{k}^{\ell}+b_{\ell-1} x_{k}^{\ell-1}+\ldots+b_{0}
\end{aligned}
$$

where $\ell=\operatorname{deg}\left(G, x_{k}\right), m=\operatorname{deg}\left(F, x_{k}\right)$, and $a_{i}, b_{i} \in \mathbb{K}\left[x_{1}, \ldots, x_{k-1}, x_{k+1} \ldots, x_{n}\right]$. Let $G \neq 0$ and $m \geq \ell$. To pseudo-divide $F$ by $G$, there exists an algorithm to obtain two polynomials, $Q$ and $R$, such that $l c\left(G, x_{k}\right)^{s} F=$ $Q G+R$, where $l c\left(G, x_{k}\right)$ is the leading coefficient of $G$ with respect to $x_{k}, 0 \leq s \leq m-\ell+1, \operatorname{deg}\left(R, x_{k}\right)<\ell$. $Q$ and $R$ are called the pseudo-quotient and the pseudo-remainder, respectively.

In our algorithm, the size of the lattice array depends on the degree of the pseudo-remainder. The following lemma is given for the purpose of determining an upper bound on the degree.

Lemma 4 ([16]). Let $f$ and $g$ be the polynomials in the variables $\left(u_{1}, u_{2}, \ldots, u_{n}, x\right)$ over a field $\mathbb{K}$ :

$$
\begin{aligned}
& f=a_{m} x^{m}+a_{m-1} x^{m-1}+\ldots+a_{0} \\
& g=b_{\ell} x^{\ell}+b_{\ell-1} x^{\ell-1}+\ldots+b_{0}
\end{aligned}
$$

where $m \geq 1, \ell \geq 1, a_{k}, b_{k}$ are polynomials in the variables $\left(u_{1}, u_{2}, \ldots, u_{n}\right)$ over $\mathbb{K}$, and $a_{m}, b_{\ell}$ are not zero polynomials. Then there exists a mechanical method to determine the polynomials $P, Q$, and $R$ in the variables $\left(u_{1}, u_{2} \ldots u_{n}\right)$ over $\mathbb{K}$ such that

$$
P f+Q g=R
$$

and $\operatorname{deg}(P, x) \leq \ell-1, \operatorname{deg}(Q, x) \leq m-1, \operatorname{deg}(R, x)=0$. The bound on $\operatorname{deg}\left(R, u_{j}\right)$ is estimated as follows:

Suppose $\ell \geq m$, and

$$
\begin{aligned}
A_{0} & =\operatorname{deg}\left(f, u_{j}\right) \\
A_{1} & =\operatorname{deg}\left(g, u_{j}\right)+(\ell-m+1) A_{0} \\
A_{k+1} & =2 A_{k}+A_{k-1}
\end{aligned}
$$

Then $\operatorname{deg}\left(R, u_{j}\right) \leq A_{m}$
In particular, if $1 \leq m \leq 2, \operatorname{deg}\left(R, u_{j}\right) \leq A_{m}$ can be simply expressed as follows:

$$
\operatorname{deg}\left(R, u_{j}\right) \leq m \operatorname{deg}\left(g, u_{j}\right)+(m \ell-m+1) \operatorname{deg}\left(f, u_{j}\right)
$$

### 2.2. Characteristic Set

The mathematical concept of a characteristic set was discovered in the late forties by J.F. Ritt [29]. In the late seventies, the Chinese mathematician Wen-Tsün Wu specialized it with modifications to commutative algebra and demonstrated its power for mechanical theorem proving [30,31].

Let $f \in \mathbb{K}\left[x_{1}, \cdots, x_{n}\right]$, the class of $f$, denoted by $\operatorname{class}(f)$, is the largest $i$ such that $x_{i}$ occurs in $f$. If $f \in \mathbb{K}$, then $\operatorname{class}(f)=0$. Let $c=\operatorname{class}(f)>0$. We call $x_{c}$, denoted by $l v(f)$, the leading variable of $f$. Considering $f$ as a polynomial in $x_{c}$, we can write $f$ as

$$
a_{n} x_{c}^{n}+a_{n-1} x_{c}^{n-1}+\cdots+a_{0}
$$

where $a_{n}, \cdots, a_{0}$ are in $\mathbb{K}\left[x_{1}, \cdots, x_{c-1}\right]$, and $a_{n} \neq 0$. We call $a_{n}$ the initial or leading coefficient of $f$ and $n$ the leading degree of $f$, denoting them as $l c(f)$ and $l d(f)$, respectively.

Definition 2 (Ascending Set [32]). A sequence of polynomials $F=\left[F_{1}, \cdots, F_{r}\right] \subseteq \mathbb{K}\left[x_{1}, \cdots, x_{n}\right]$ is said to be an ascending set (or chain), if one of the following two conditions holds:

1. $r=1$ and $F_{1}$ s not identically zero;
2. $\quad r>1$ and $0<\operatorname{class}\left(F_{1}\right)<\operatorname{class}\left(F_{2}\right), \cdots,<\operatorname{class}\left(F_{r}\right) \leq n$, and each $F_{i}$ is reduced with respect to the preceding polynomials, $F_{j}^{\prime} s(1 \leq j<i)$.

Definition 3 (Characteristic Set [32]). Let $P \subseteq K[x]$ is nonempty set of polynomials, and $I=\langle P\rangle$ be the ideal generated by $P$, ascending set $F=\left[F_{1}, \cdots, F_{r}\right]$ is the Characteristic Set of $P$, if $F \subseteq\langle P\rangle$, and $\operatorname{prem}(P, F)=\{0\}$.

The Wu-Ritt Process [32] described how to obtain such the ascending set. Besides Gröbner basis method, characteristic set provides an alternative algorithmic way for solving multivariate polynomial equations or differential equations.

### 2.3. Hybrid System

In this paper, we adopt the hybrid automata proposed in [1] as our modeling framework. Many other models for hybrid systems can be found in [33-36].

A hybrid system can be defined as follows:
Definition 4 (Hybrid System ). A hybrid system is a tuple $\mathcal{H}:\left\langle V, L, \mathcal{T}, \ell_{0}, \Theta\right\rangle$, where

- $\quad L$ is a finite set of locations (or modes);
- $\quad V$ is a set of real-valued system variables. The hybrid system state space is denoted by $\Sigma=L \times \mathbb{R}^{|V|}, a$ state is denoted by $\langle\ell, s\rangle \in \Sigma$, and $s \in \mathbb{R}^{|V|}$ is a continuous state of the variables over the real numbers;
- $\mathcal{T} \subseteq L \times L \times 2^{\mathbb{R}^{|V|}} \times\left(\mathbb{R}^{|V|} \mapsto \mathbb{R}^{|V|}\right)$ is a set of discrete transitions. A discrete transition $\tau=$ $\left\langle\ell_{1}, \ell_{2}, \rho_{\tau}, \alpha_{\tau}\right\rangle$ consists of the pre- and post-locations $\ell_{1}, \ell_{2}$, a guard $\rho_{\tau}$ (which is a boolean function of the variables $V$ ), and an action $\alpha_{\tau}$, which is a first-order assertion over $V^{\prime} \cup V$, where $V$ denotes the current-state variables and $V^{\prime}$ denotes the next-state variables;
- $\quad F: L \mapsto\left(\mathbb{R}^{|V|} \mapsto \mathbb{R}^{|V|}\right)$ is a map that maps each location $\ell \in L$ to a differential rule $f$; that is, $F(\ell)=f$, of the form $\dot{v}_{i}=f_{i}(\boldsymbol{x})$. The differential rule $F(\ell)$ specifies how the system variables evolve at the location $\ell$, which is also known as a vector field or a flow field;
- $\quad F: L \mapsto\left(\mathbb{R}^{|V|} \mapsto \mathbb{R}^{|V|}\right)$ is a map that maps each location $\ell \in L$ to a differential rule $\vec{f}$; that is, $F(\ell)=\vec{f}$, of the form $\dot{v}_{i}=f_{i}(\boldsymbol{x})$. The differential rule $F(\ell)$ specifies how the system variables evolve at the location $\ell$, which is also known as a vector field or a flow field;
- $\quad I: L \mapsto 2^{\mathbb{R}^{|V|}}$ is a map that maps each location $\ell \in L$ to a location condition (location invariant), which is an assertion over $V$ and defines all possible continuous states that the system is allowed to move to while at location $\ell$;
- $\Theta$ is an assertion specifying the initial condition; and
- $\quad \ell_{0} \in L$ is the initial location. We assume that the initial condition satisfies location invariance at the initial location; that is, $\Theta \models I\left(\ell_{0}\right)$.

The transition and dynamic structures of the hybrid system define a set of trajectories. A trajectory is a sequence of states starting from a state $\left\langle\ell_{0}, \hat{x}_{0}\right\rangle \in \Theta$, where $\Theta \in \Sigma$ is an initial state set, consisting of a series of interleaved continuous flows and discrete transitions. During the continuous flows, the system evolves following the vector fields at some location $\ell$ until the invariant condition $I(\ell)$ is violated. At some state $\langle\ell, \hat{x}\rangle$, if there is a discrete transition $\tau$ from location $\ell$ to $\ell^{\prime}$ such that $\rho_{\tau}(\hat{x})=$ true, then the discrete transition can be taken.

Definition 5 (Trajectory). A trajectory of a hybrid system $\mathcal{H}$ is an infinite sequence of states $\langle\ell, \hat{x}\rangle \in L \times \mathbb{R}^{|V|}$ of the form

$$
\left\langle\ell_{0}, \hat{x}_{0}\right\rangle,\left\langle\ell_{1}, \hat{x}_{1}\right\rangle,\left\langle\ell_{2}, \hat{x}_{2}\right\rangle, \ldots,
$$

such that
Initiation: $\hat{x}_{0} \models \Theta$ specifies an initial state.
Furthermore, for each consecutive state pair $\left\langle\ell_{i}, \hat{x}_{i}\right\rangle,\left\langle\ell_{i+1}, \hat{x}_{i+1}\right\rangle$, one of the two consecution conditions below is satisfied.

Discrete Consecution: there exists a transition $\tau:\left\langle\ell_{1}, \ell_{2}, \rho_{\tau}, \alpha_{\tau}\right\rangle \in \mathcal{T}$ such that $\tau$ is enabled; that is, $\rho_{\tau}\left(\hat{x}_{i}\right)=$ true and $\hat{x}_{i+1}=\alpha_{\tau}\left(\hat{x}_{i}\right)$. Or,

Continuous Consecution: $\ell_{i}=\ell_{i+1}=\ell$; in other words, the location does not change and there exists a time interval, $\delta>0$, along which a smooth (continuous and differentiable to all orders) function $\Psi:[0, \delta] \mapsto \mathbb{R}^{|V|}$ exists, such that $\Psi$ evolves from $\hat{x}_{i}$ to $\hat{x}_{i+1}$ according to the differential rule at location $\ell$ while satisfying the location condition $I(\ell)$. Formally,

1. $\Psi(0)=\hat{x}_{1}, \Psi(\delta)=\hat{x}_{2}$ and $\forall t \in[0, \delta], \Psi(t) \models I(\ell)$,
2. $\forall t \in[0, \delta],\langle\Psi(t), \dot{\Psi}(t)\rangle \vDash F(\ell)$.

A hybrid state $\langle\ell, \hat{x}\rangle$ is called a reachable state if it appears in some trajectory of $\mathcal{H}$.
A linear constraint over $V$ is an inequality of the form $a_{1} x_{1}+\ldots+a_{n} x_{n}+b \leq 0$, and a linear assertion over a set of variables $V$ is a conjunction of linear constraints over $V$. The set of points satisfying a linear assertion forms a polyhedron.

A non-linear constraint is an inequality of the form $P \leq 0$, where $P$ is a polynomial in $\left\{x_{1}, \ldots, x_{n}\right\}$. The constraint is said to be algebraic if $P=0$. A non-linear assertion is a conjunction of non-linear constraints. The set of points satisfying a non-linear assertion is called a semi-algebraic set .

Throughout the paper, given an assertion $\psi$ over the variables $V, \psi^{\prime}$ denotes the assertion obtained by replacing each variable $v \in V$ by $v^{\prime} \in V^{\prime}$.

General hybrid systems can be specialized into algebraic hybrid systems such that, for each transition $\tau$, the transition relation $\rho_{\tau}$ is an algebraic assertion over $V^{\prime} \cup V$ and the initial condition $\Theta$ is an algebraic assertion over $V$.

Definition 6 (Algebraic Hybrid Systems ). An algebraic hybrid system is a hybrid system $\left\langle V, L, \mathcal{T}, \ell_{0}, \Theta\right\rangle$, where:

1. For each transition $\tau:\left\langle\ell_{1}, \ell_{2}, \rho_{\tau}, \alpha_{\tau}\right\rangle, \alpha_{\tau}$ is an algebraic assertion.
2. The initial condition $\Theta$ and the location conditions $I(\ell)$ are also algebraic assertions.
3. Each rule $F(\ell)$ is of the form $\wedge \dot{v}_{i}=f_{i}\left(v_{1}, \ldots, v_{n}\right)$, where $f_{i} \in \mathbb{R}\left[v_{1}, \ldots, v_{n}\right]$.

This paper focuses on algebraic hybrid systems.
Definition 7 (Invariant). An invariant of a hybrid system $\mathcal{H}$ at a location $\ell$ is an assertion $\psi$ such that, for any reachable state $\langle\ell, \hat{x}\rangle$ at $\ell, \hat{x} \models \psi$.

Template polynomials play an important role in Sankaranarayanan et al.'s approach. Given a degree $d$, a template polynomial is essentially a generic degree $d$ polynomial; that is, $p=$ $\sum_{i_{1}+\cdots+i_{n} \leq d} \mathcal{L}\left(a_{1}, \cdots, a_{\ell}\right) x_{1}^{i_{1}} \ldots x_{n}^{i_{n}}$, where $\left\{a_{1}, \cdots, a_{\ell}\right\}$ are coefficients to be decided. Sankaranarayanan et al. treat these coefficients $\left\{a_{1}, \cdots, a_{\ell}\right\}$ as unknowns and encode inductive conditions by a Gröbner basis to generate constraints on the coefficients such that any solution corresponds to an inductive assertion. Formally,

Definition 8 (Template). Let $\boldsymbol{a}=\left(a_{1}, \cdots, a_{l}\right)$ be template variables and $\mathcal{L}(\boldsymbol{a})$ be the polynomial of the form $\sum_{\beta} c_{\beta} a^{\beta}$, where each $c_{\beta}$ is a real-valued coefficient $\boldsymbol{\beta}=\left(\beta_{1}, \cdots, \beta_{n}\right), \beta_{n} \geq 0$. Ad-degree template over $\{\boldsymbol{a}, \boldsymbol{x}\}$ is a polynomial in the variables $x$ with coefficients $\mathcal{L}(\boldsymbol{a})$ (i.e., $\left.p=\sum_{i_{1}+\cdots+i_{n} \leq d} \mathcal{L}\left(a_{1}, \cdots, a_{\ell}\right) x_{1}^{i_{1}} \ldots x_{n}^{i_{n}}\right)$.

Hybrid systems generally consist of many locations and, hence, an invariant can be seen as a mapping to map each location to an assertion which is true under any system state reaching the location. Thus, the following two definitions come very naturally.

Definition 9 (Algebraic Assertion Map ). Given a domain of algebraic assertions $\mathcal{D}$, an algebraic assertion map for an algebraic hybrid system is a map $\eta: L \rightarrow \mathcal{D}$ that associates each location of the system with an algebraic assertion, where each algebraic assertion $\eta(\ell)$ is of the form $p=0$.

For simplicity, we shall use $\eta(\ell)$ to denote both the algebraic assertion $p=0$ and the polynomial $p$, as long as it will not cause any ambiguity.

Definition 10 (Template Map). Let $\mathcal{H} \equiv\left\langle V, L, \mathcal{T}, \ell_{0}, \Theta\right\rangle$ be an algebraic hybrid system. Assuming a set of template variables $\boldsymbol{a}=\left(a_{1}, \cdots, a_{l}\right)$, a template map over $\mathcal{H}$ is a map $\eta: L \rightarrow \mathcal{L}(\boldsymbol{a})[V]$ that maps each location in $L$ to a template over $\{a, V\}$.

It is a well-known fact, from the pioneering work of Floyd and Hoare [37,38], that if $\eta$ is an inductive assertion map, then $\eta(\ell)$ is invariant at $\ell$. In fact, all known invariant generation methods are inductive assertion generation methods.

Definition 11 (Inductive Algebraic Assertion Map ). An inductive algebraic assertion map $\eta(\ell)$ is a map that associates with each location $\ell \in L$ an assertion $\eta(\ell)$ that holds initially and is preserved by all discrete transitions and continuous flows. More formally, an inductive assertion map satisfies the following requirements:

Initiation: The algebraic assertion at $\ell_{0}$ subsumes the initial condition; that is, $\Theta=\eta\left(\ell_{0}\right)$.
Discrete Consecution: For each transition $\tau:\left\langle\ell_{i}, \ell_{j}, \rho_{\tau}, \alpha_{\tau}\right\rangle$ starting from a state satisfying $\eta\left(\ell_{i}\right)$, taking $\tau$ leads to a state satisfying $\eta\left(\ell_{j}\right)$. Formally, $\eta\left(\ell_{i}\right) \wedge \rho_{\tau} \wedge \alpha_{\tau}=\eta\left(\ell_{j}\right)^{\prime}$.

Continuous Consecution: For every location $\ell \in L$ and states $\left\langle\ell, \hat{x}_{1}\right\rangle,\left\langle\ell, \hat{x}_{2}\right\rangle$ such that $\hat{x}_{1}$ evolves from $\hat{x}_{2}$ according to the differential rule $F(\ell)$ at $\ell$, if $\hat{x}_{1} \models \eta(\ell)$, then $\hat{x}_{2} \models \eta(\ell)$. If $\eta(\ell)$ is an assertion of the form
$f_{\ell}(\boldsymbol{x})=0$ and $f_{\ell} \in \mathbb{R}\left[v_{1}, \ldots, v_{|V|}\right]$ is a real-valued smooth function, we can express continuous consecution by the following condition:

$$
I(\ell)(x) \wedge\left(f_{\ell}(x)=0\right) \models \dot{f}_{\ell}(\boldsymbol{x})=0
$$

Note that $\dot{f}_{\ell}$ denotes the Lie derivative of $f(\ell)$ along the vector field $F(\ell)$.
An important concept used in this paper is the Lie derivative. In our context, the Lie derivative evaluates the change of a scalar function $\varphi(x)$ along the flow of a vector field of the form $\dot{x}=f(x)$, where $\boldsymbol{f}(\boldsymbol{x})=\left(f_{1}(\boldsymbol{x}), \ldots, f_{n}(\boldsymbol{x})\right)$. Formally,

Definition 12 (Lie Derivative).

$$
\mathcal{L}_{f} \varphi(\boldsymbol{x}) \triangleq \frac{\partial \varphi}{\partial x} f(\boldsymbol{x})=\sum_{i=1}^{n} \frac{\partial \varphi}{\partial x_{i}} \dot{x}_{i}=\sum_{i=1}^{n} \frac{\partial \varphi}{\partial x_{i}} f_{i}(\boldsymbol{x})
$$

## 3. Theory of CIM

Assuming $\left\{a_{1}, \cdots, a_{l}\right\}$ are the parameters to be decided, there exists a system of $k$ polynomial equations that involve $\{l+d+k\}$ indeterminates in formula (4); that is, $\left\{a_{1}, \cdots, a_{l}, u_{1}, \cdots, u_{n}, x_{1}, \cdots, x_{k}\right\}$. Let the $\{l+d+k\}$ indeterminates be ordered as $x_{1} \prec x_{2} \prec \cdots \prec$ $x_{k} \prec u_{1}, \cdots, \prec u_{n} \prec a_{1}, \cdots, \prec a_{l}$. By Wu-Ritt's Algorithm [39], the indeterminates are divided into two categories: dependent variables $\left(x_{1}, \cdots, x_{k}\right)$ and independent variables $\left(u_{1}, \cdots, u_{n}, a_{1}, \cdots, a_{l}\right)$.

$$
\begin{equation*}
\bigwedge_{m=1}^{k} f_{m}\left(a_{1}, \cdots, a_{l}, u_{1}, \cdots, u_{n}, x_{1}, \cdots, x_{k}\right)=0 \tag{4}
\end{equation*}
$$

However, for the convenience of discussion, this paper divides the $\{l+d+k\}$ indeterminates into three categories: $\boldsymbol{x}=\left(x_{1}, \cdots, x_{k}\right), \boldsymbol{u}=\left(u_{1}, \cdots, u_{n}\right)$, and $\boldsymbol{a}=\left(a_{1}, \cdots, a_{l}\right)$, which are called dependent variables, independent variables, and template variables (parameters), respectively. The ring of polynomials with coefficients in $\mathbb{K}$ can also be written as $\mathbb{K}[\boldsymbol{a}, \boldsymbol{u}, \boldsymbol{x}]$.

In Sankaranarayanan et al.'s approach, encoding complete inductive conditions into the constraint by a comprehensive Gröbner basis will lead to computability problems. In our approach, CIM is used to take the place of a comprehensive Gröbner basis in encoding the complete inductive conditions. Concretely, given an algebraic hybrid system $\mathcal{H}$, we first define a $d$-degree polynomial as template (i.e., $\sum_{i_{1}+\cdots+i_{n} \leq d} \mathcal{L}(\boldsymbol{a}) x_{1}^{i_{1}} \ldots x_{n}^{i_{n}}$, where $\boldsymbol{a}=\left(a_{1}, \cdots, a_{l}\right)$ are the template variables to be decided). CIM is an algorithm for finding all the assignments $\hat{\boldsymbol{a}}=\left(\hat{a}_{1}, \cdots, \hat{a}_{\ell}\right)$ to the template variables $\boldsymbol{a}$ that guarantee the truth of a formula of the form:

$$
\begin{equation*}
\forall(\boldsymbol{u}, \boldsymbol{x}): \bigwedge_{m=1}^{k} f_{m}(\hat{\boldsymbol{a}}, \boldsymbol{u}, \boldsymbol{x})=0 \models G(\hat{\boldsymbol{a}}, \boldsymbol{u}, \boldsymbol{x})=0 \tag{5}
\end{equation*}
$$

where $\boldsymbol{a}=\left(a_{1}, \cdots, a_{l}\right), \boldsymbol{u}=\left(u_{1}, \cdots, u_{n}\right)$, and $\boldsymbol{x}=\left(x_{1}, \cdots, x_{k}\right), f_{m}, G \in \mathcal{L}(\boldsymbol{a})[\boldsymbol{u}, \boldsymbol{x}]$ are all algebraic assertions. The indeterminates in formula (5) are divided into three groups: $a$ denotes the template variables (parameters), $\boldsymbol{u}$ denotes independent variables, and $\boldsymbol{x}$ denotes dependent variables, as they are constrained by $k$ polynomial equations in the hypothesis of formula (5).

Note: For an invariant generation, the implication (5) will contain more system variables, as transition relations are algebraic constraints on both current-state and next-state variables, which doubles the number of system variables and introduces many more free variables.

### 3.1. Basic Lemma and Theorem

We first prove a lemma.

Lemma 5. Let $R \in \mathcal{L}(\boldsymbol{a})[\boldsymbol{u}]$ with $\boldsymbol{u}=\left(u_{1}, \cdots, u_{n}\right)$, such that $\operatorname{deg}\left(R, u_{i}\right)=d_{i} ; S_{R}=S_{1} \times S_{2} \times \cdots \times S_{n}$ be an $n$-dimensional lattice array of size $\left(d_{1}+1, \cdots, d_{n}+1\right)$; and $\Omega$ be the following system of equations over $\boldsymbol{a}$ :

$$
\begin{equation*}
\Omega: \quad R\left(\boldsymbol{a}, \hat{\boldsymbol{u}}_{i}\right)=0, i=1, \cdots,\left|S_{R}\right| \tag{6}
\end{equation*}
$$

where $\hat{\boldsymbol{u}}_{i} \in S_{R}$. Then, for any vector $\hat{\boldsymbol{a}}, \hat{\boldsymbol{a}}$ is a solution to $\Omega$ iff $\hat{\boldsymbol{a}}$ satisfies the following identity:

$$
\begin{equation*}
R(\hat{\boldsymbol{a}}, \boldsymbol{u}) \equiv 0 \tag{7}
\end{equation*}
$$

Proof. $\Longrightarrow$. Suppose that $\hat{\boldsymbol{a}}$ is a solution to the system of equations (6); that is,

$$
\left\{\begin{array}{l}
R\left(\hat{\boldsymbol{a}}, \hat{\boldsymbol{u}}_{1}\right)=0  \tag{8}\\
R\left(\hat{\boldsymbol{a}}, \hat{\boldsymbol{u}}_{2}\right)=0 \\
\cdots \\
R\left(\hat{\boldsymbol{a}}, \hat{\boldsymbol{u}}_{\left|S_{R}\right|}\right)=0
\end{array}\right.
$$

In other words, $R(\hat{\boldsymbol{a}}, \boldsymbol{u})$ vanishes over $S_{R}$. By Lemma 2, $R(\hat{\boldsymbol{a}}, \boldsymbol{u}) \equiv 0$.
$\Longleftarrow$. Suppose that there exists $\hat{\boldsymbol{a}}$ which satisfies $R(\hat{\boldsymbol{a}}, \boldsymbol{u}) \equiv 0$, then $R(\hat{\boldsymbol{a}}, \boldsymbol{u})$ must vanish over $S_{R}$; that is, for every $\hat{\boldsymbol{u}}_{i} \in S_{R}$,

$$
R\left(\hat{\boldsymbol{a}}, \hat{\boldsymbol{u}}_{i}\right)=0
$$

In other words, (8) holds; that is, $\hat{a}$ is a solution to (6).
Example 2. ( from [3]) Compute the constraint on $\left\{a_{1}, a_{2}, a_{3}\right\}$ such that $p=\left(2 a_{2}+3\right) x_{1} x_{2}^{2}+3 a_{3} x_{2}+4\left(a_{3}+\right.$ $\left.a_{1}+10\right) \equiv 0$.

1. By observation, the lattice array size of $p$ is $(1+1,2+1)$. Let $x_{1}=0,1$ and $x_{2}=-1,0,1$. Then, six instances in the lattice array are $(0,-1),(0,0),(0,1),(1,-1),(1,0),(1,1)$.
2. For each instance, the constraint is obtained by substituting, which consists of the six equations shown in Table 1.

Table 1. The six constraint equations generated in parallel.

| $x_{1}$ | $x_{2}$ | $\boldsymbol{a}_{1}, \boldsymbol{a}_{2}, \boldsymbol{a}_{\mathbf{3}}$ |
| :---: | :---: | :---: |
| 0 | 0 | $4 a_{3}+a_{1}+10=0$ |
| 0 | 1 | $7 a_{3}+a_{1}+10=0$ |
| 0 | -1 | $a_{3}+a_{1}+10=0$ |
| 1 | 0 | $4 a_{3}+a_{1}+10=0$ |
| 1 | 1 | $7 a_{3}+2 a_{2}+a_{1}+13=0$ |
| 1 | -1 | $a_{3}+2 a_{2}+a_{1}+13=0$ |

3. Solving the above six equations: $a_{1}=-10, a_{2}=-\frac{3}{2}, a_{3}=0$, as in [3].

Problem: Given a formula in the form of (5), how do we compute all the possible values $\hat{\boldsymbol{a}}$ for $\boldsymbol{a}$ for which (5) holds?

The traditional method is to construct a Gröbner basis [3] of $\left\{f_{1}, \cdots, f_{k}\right\}$. Let $I=$ $\operatorname{Ideal}\left(\left\{f_{1}, \cdots, f_{k}\right\}\right)$ be the ideal generated by $\left\{f_{1}, \cdots, f_{k}\right\}$ and $G$ be a polynomial. By Hilbert's Nullstellensatz [40], the formula (5) holds is equivalent to that there exists an integer $m \geq 1$ such that $G^{m}$ belongs to $I$. Therefore, according to Hilbert's Nullstellensatz, to compute all the possible $\hat{a}$ that make the formula (5) hold, one has to enumerate all the $m \geq 1$ to find all the $\hat{a}$ that make $G^{m} \in I$ based on Gröbner basis, which is apparently not possible because there is an infinite number of $m$.

For the above reason, Sankaranarayanan et al. chose to set $m=1$ to find all the $\hat{a}$ that make $G \in I$ in [3]. There are two drawbacks to this approach. Firstly, since $G \in I$ is just a sufficient condition for the formula (5) to hold, the approach can only find part of the solutions to (5) which is not complete.

Secondly, computing $\hat{a}$ in this way involves constructing a comprehensive Gröbner basis [41], a variant of the Gröbner basis, which is very expensive (double exponential in the dimension of the variables). To avoid this issue, the basic idea of CIM is "turning the difficulty of quality into the complexity of quantity". We now introduce our Theorem 1:

Theorem 1. Given $x_{1} \prec x_{2} \prec \cdots \prec x_{k} \prec u_{1}, \cdots, \prec u_{n} \prec a_{1}, \cdots, \prec a_{l}$, let $F=\left\{f_{i}, i=1, \cdots, k\right\}$ be defined as in formula (5) and $F_{i}\left(\boldsymbol{a}, u, x_{1}, \cdots, x_{i}\right), i=1, \cdots, k$ be the ascending set of $F$ obtained by Wu-Ritt's Algorithm [39]. Then, there must exist a sequence of polynomials $P_{i}(\boldsymbol{a}, \boldsymbol{u}, \boldsymbol{x}), i=1, \cdots, k, Q(\boldsymbol{a}, \boldsymbol{u}, \boldsymbol{x})$, and $R(\boldsymbol{a}, \boldsymbol{u})$, such that the following equation holds:

$$
\begin{equation*}
\sum_{i=1}^{k} P_{i} F_{i}+Q G=R \tag{9}
\end{equation*}
$$

Moreover, for any assignment $\hat{\boldsymbol{a}}$ to $\boldsymbol{a}$, the formula (5) holds if $R(\hat{\boldsymbol{a}}, \boldsymbol{u}) \equiv 0$ and the following system of equations has no solution

$$
\left\{\begin{align*}
F_{i}\left(\hat{\boldsymbol{a}}, \boldsymbol{u}, x_{1}, \cdots, x_{i}\right) & =0, \quad i=1, \cdots, k  \tag{10}\\
Q(\hat{\boldsymbol{a}}, \boldsymbol{u}, \boldsymbol{x}) & =0
\end{align*}\right.
$$

Proof. For the convenience of proof, let formula (11) denote the hypothesis of (5) and formula (12) denote the conclusion of the formula (5):

$$
\begin{gather*}
\bigwedge_{m=1}^{k} f_{m}(\boldsymbol{a}, \boldsymbol{u}, \boldsymbol{x})=0  \tag{11}\\
G(\boldsymbol{a}, \boldsymbol{u}, \boldsymbol{x})=0 \tag{12}
\end{gather*}
$$

Now, we show how to construct such a series of polynomials $P_{i}, Q$, and $R \in \mathcal{L}(\boldsymbol{a})[\boldsymbol{u}]$ satisfying equation (9).

By applying Lemma 4 to $F_{k}$ and $G$, we can obtain $\widetilde{P}_{k} F_{k}+Q_{k} G=R_{k}\left(\boldsymbol{a}, \boldsymbol{u}, x_{1}, \ldots, x_{k-1}\right)$. Similarly, by applying Lemma 4 to $F_{k-i}$ and $R_{k-i}$ repeatedly, we get the following sequence of equations,

$$
\begin{align*}
& \widetilde{P}_{k} F_{k}+Q_{k} G=R_{k} \\
& \widetilde{P}_{k-1} F_{k-1}+Q_{k-1} R_{k}=R_{k-1} \\
& \widetilde{P}_{k-2} F_{k-2}+Q_{k-2} R_{k-1}=R_{k-2}  \tag{13}\\
& \ldots \ldots \ldots \ldots \\
& \widetilde{P}_{2} F_{2}+Q_{2} R_{3}=R_{2} \\
& \widetilde{P}_{1} F_{1}+Q_{1} R_{2}=R_{1}
\end{align*}
$$

where $R_{1} \in \mathcal{L}(\boldsymbol{a})[\boldsymbol{u}], R_{i} \in \mathcal{L}(\boldsymbol{a})\left[\boldsymbol{u}, x_{1}, \cdots, x_{i-1}\right], i=2, \cdots, k$ and $\widetilde{P}_{i}$ and $Q_{i} \in \mathcal{L}(\boldsymbol{a})\left[\boldsymbol{u}, x_{1}, \cdots, x_{i}\right], i=$ $1, \cdots, k$. Note that $R_{1}$ does not contain the variable $x_{i}$ any more.

Next, by a sequence of substitutions of $R_{k-i}$ in (13), in a top-down order, we can easily derive the following equation:

$$
\begin{equation*}
\widetilde{P}_{1} F_{1}+\sum_{j=2}^{k}\left(\prod_{i=1}^{j-1} Q_{i}\right) \widetilde{P}_{j} F_{j}+\left(\prod_{j=1}^{k} Q_{j}\right) G=R_{1} . \tag{14}
\end{equation*}
$$

Now, let $R=R_{1}, P_{1}=\widetilde{P}_{1}, P_{j}=\left(\prod_{i=1}^{j-1} Q_{i}\right) \widetilde{P}_{j}, j=2, \ldots, k$, and $Q=\left(\prod_{j=1}^{k} Q_{j}\right)$. Then, we can obtain the following equation:

$$
\sum_{i=1}^{k} P_{i} F_{i}+Q G=R
$$

Suppose that there exists an assignment $\hat{\boldsymbol{a}}$ to $\boldsymbol{a}$ that satisfies $R(\hat{\boldsymbol{a}}, \boldsymbol{u}) \equiv 0$ and the system of equations (10) has no solution; we prove the formula (5) holds. As $R \equiv 0$, we get

$$
\begin{equation*}
\sum_{i=1}^{k} P_{i} F_{i}=-Q G \tag{15}
\end{equation*}
$$

In addition, as the system of equations (10) has no solution, for any $(\hat{\boldsymbol{u}}, \hat{x}), F_{i}(\hat{\boldsymbol{a}}, \hat{\boldsymbol{u}}, \hat{\boldsymbol{x}})=0$ we can deduce that $Q(\hat{\boldsymbol{a}}, \hat{\boldsymbol{u}}, \hat{\boldsymbol{x}}) \neq 0$. Then, by the equation (15), we deduce that $G(\hat{\boldsymbol{a}}, \hat{\boldsymbol{u}}, \hat{\boldsymbol{x}})=0$ as well. Therefore, formula (5) holds. Note that, in Wu-Ritt's algorithm, $Q=\prod_{j=1}^{k} Q_{j} \neq 0$ is called the non-degenerate condition [39].

Remark 1. According to Theorem 1, we know that the solutions $\hat{\boldsymbol{a}}$ to $\boldsymbol{a}$ that satisfy $R(\hat{\boldsymbol{a}}, \boldsymbol{u}) \equiv 0$ and the system of equations (10) have no solution satisyfing formula (5) as well. Therefore, the idea for finding $\hat{\boldsymbol{a}}$ to make formula (5) hold is that we must first solve the constraint on $\boldsymbol{a}$ derived from $R(\boldsymbol{a}, \boldsymbol{u}) \equiv 0$ to find all the solutions $\hat{\boldsymbol{a}}$ and decide whether the system of equations (10) has a solution, given $\boldsymbol{a}=\hat{\boldsymbol{a}}$. For the first step, we use Lemma 5. Regarding whether the non-degenerate condition is true or not, we can make use of the criterion presented in [39] (i.e., $I_{1} I_{2} \cdots I_{k} \neq 0, I_{i}$ is the initial of $F_{i}$ ). From the above analysis, we can see that the solution to $R(\boldsymbol{a}, \boldsymbol{u}) \equiv 0$ cannot ensure that formula (5) holds. If the non-degenerate condition is false, formula (5) may or may not hold, which is crucial to the theorem's proof by CIM or by Wu-Ritt's Algorithm [39]. Fortunately, for invariant generation, it only means we need another step to verify the obtained invariant. Generally, verifying invariants is less expensive than generating invariants, which also means that our invariant generation algorithm is a two-phase algorithm.

### 3.2. Generate the Constraint for Implication by CIM

By Lemma 5, in order to generate a constraint over the template variables $\boldsymbol{a}$ that guarantees $R(\boldsymbol{a}, \boldsymbol{u})$ to be identical to zero, we only need test whether $R(\boldsymbol{a}, \boldsymbol{u})$ vanishes over a lattice array $S_{R}$ of size $\left(\operatorname{deg}\left(R, u_{1}\right)+1, \operatorname{deg}\left(R, u_{2}\right)+1, \cdots, \operatorname{deg}\left(R, u_{n}\right)+1\right)$, where the upper bound of $\operatorname{deg}\left(R, u_{i}\right)$ can be estimated by Lemma 4.

More specifically, take $\hat{\boldsymbol{u}} \in S_{R}$, substitute for $\boldsymbol{u}$ in (11) and (12), compute $R(\boldsymbol{a})$ by the division algorithm (Lemma 4), and then equate $R(\boldsymbol{a})=0$ to obtain the constraint equations. This process is similar to Wu's division method, but is simpler as the $\boldsymbol{u}=\left(u_{1}, \cdots, u_{n}\right)$ have been replaced by the numbers (instances). However, the aforementioned benefits come at a cost. The benefits are that the free variables are eliminated, the cost is that the implication (5) turns into $\left|S_{R}\right|$ simpler implications without free variables. This is the main idea of CIM: "turning the difficulty of quality into the complexity of quantity". If (5) is too difficult to deal with, CIM is meaningful. We outline our algorithm for generating constraints by computing $R$ in Algorithm 1.

```
Algorithm 1: Generating constraint for implication by \(R\).
    input : The hypothesis of implication \(f_{m}\left(a_{1}, \cdots, a_{\ell}, u_{1}, \cdots, u_{n}, x_{1}, \cdots x_{k}\right)\), the conclusion of implication
                \(G\left(a_{1}, \cdots, a_{\ell}, u_{1}, \cdots, u_{n}, x_{1}, \cdots, x_{k}\right)=0\), and \(A_{i}\) the upper bound of \(\operatorname{deg}\left(R, u_{i}\right), U\) is the set of independent variables, MaxA
                is the maximum of \(A_{i}\), inst is a an \(n\)-dimension vector of the form \(\left\langle\hat{u}_{1}, \cdots, \hat{u}_{n}\right\rangle, L A\) is the set of \(n\)-dimension vector.
    output : \(C\), the set of constraints on template variables \(\left\{a_{1}, \cdots, a_{\ell}\right\}\), a log file to record the process details.
    /*by default \(C\) and \(L A\) are \(\varnothing * /\);
    \(C=\varnothing\);
    \(L A=\varnothing\);
    \(/ *\) The nested loop structure aims to generate a lattice array for independent variables of \(\left\{u_{1}, \cdots, u_{n}\right\}\) of size (Max \(A^{n}\) ); for simplicity,
        \(u_{i}=0, \cdots\), MaxA \(^{*} /\)
    for \(i=0\) to MaxA do
        /*generate an instance*/
        foreach \(u\) in \(U\) do
                \(u=i ;\)
                inst \([u]=i\);
        end
        /*add the instance to Lattice Array*/
        \(L A=L A+i n s t ;\)
    end
    foreach inst in \(L A\) do
        /*instantiate \(\left\{u_{1}, \cdots, u_{n}\right\}\) to real values*/
        foreach \(u\) in \(U\) do
        \(u=\) inst \([u]\);
        end
        /*substitute real values for \(\left\{u_{1}, \cdots, u_{n}\right\}\) in \(\left\{f_{1}, \cdots, f_{k}\right\}^{*} /\)
        \(\bigwedge_{m=1}^{k} f_{m}^{\prime}\left(a_{1}, \cdots, a_{\ell}, x_{1}, \cdots, x_{k}\right) \leftarrow \bigwedge_{m=1}^{k} f_{m} ;\)
        /*substitute real values for \(\left\{u_{1}, \cdots, u_{n}\right\}\) in \(G^{*} /\)
        \(G^{\prime}\left(a_{1}, \cdots, a_{\ell}, x_{1}, \cdots, x_{k}\right) \leftarrow G ;\)
        /*transform \(\bigwedge_{m=1}^{k} f_{m}^{\prime}\) into ascending set of \(\bigwedge_{m=1}^{k} F_{m}\) by Wu-Ritt's algorithm*/
        \(\bigwedge_{m=1}^{k} F_{m}\left(a_{1}, \cdots, a_{\ell}, x_{1}, \cdots, x_{k}\right)=0 \leftarrow \bigwedge_{m=1}^{k} f_{k}^{\prime}\)
        /*compute \(R^{*}\) /
        \(R\left(a_{1}, \cdots, a_{\ell}\right)=\) compute \(R\left(\left\{F_{1}, \ldots, F_{k}\right\}, G\right)\);
        \(/ *\) Generate constraints on \(a_{1}, \cdots, a_{\ell}{ }^{*} /\)
        \(C=C \cup\left\{R\left(a_{1}, \cdots, a_{\ell}\right)=0\right\} ;\)
        /*write \(\log\) to record the details*/
        writelog \(\left(\boldsymbol{u},\left\{F_{1}, \ldots, F_{k}\right\},\left\{f_{1}^{\prime}, \ldots, f_{k}^{\prime}\right\}, G^{\prime}, R, C\right)\);
        /*utilize \(C\) to simplify the implication*/
        Simplify \(\left(f_{1}, \cdots, f_{k}, G\right)\) by \(C\);
    end
    return \(C\);
```

1. $\boldsymbol{u}=\left\{u_{1}, \cdots, u_{n}\right\}$ and $\boldsymbol{x}=\left\{x_{1}, \cdots, x_{k}\right\}$ are independent variables and dependent variables, respectively.
2. Lines 5-13 aim to generate Lattice Array.
3. Lines 16-22 aim to substitute real values for $\left\{u_{1}, \cdots, u_{n}\right\}$ in $f_{i}$ and $G$ and, hence, eliminate the independent variables. It is obvious that instances will simplify the computation of proposition (5).
4. Lines 14-33 are a ForEach loop, hence, the algorithm is easy to parallelize.
5. To make Algorithm 1 simpler, we can use $\operatorname{Max}\left(\operatorname{deg}\left(f_{m}, u_{i}\right), \operatorname{deg}\left(G, u_{i}\right)\right)$ to take the place of $A_{i}$, the upper bound of $\operatorname{deg}\left(R, u_{i}\right)$, which is not allowed for theorem proving, as $\left|S_{R}\right|$ is crucial to theorem proving by CIM. However, for invariant generation, it only means we need to verify the obtained constraint in the second phase (Section 4).

## Complexity Analysis

Assuming that the degree of each variable in $f_{i}$ is no more than $d$ (i.e., $\operatorname{deg}\left(f_{i}\right) \leq d, 1 \leq i \leq k$ ), and no more than $\delta$ in $g$ (i.e., $\operatorname{deg}(g) \leq \delta$ ), our method consists of three main steps. In the first step, we transform the equations (11) into triangular sets (i.e., equations in triangular form) by Wu-Ritt's method. By [32], the complexity of computing the characteristic set of (11) is $\mathcal{O}\left(k^{\mathcal{O}(\ell+n+k)}(d+\right.$ $1)^{\mathcal{O}^{(\ell+n+k)^{3}}}$; however, CIM eliminates the independent variables $\left\{u_{1}, \cdots, u_{n}\right\}$, such that the complexity of computing the characteristic set for every instance is $\mathcal{O}\left(k^{\mathcal{O}(\ell+k)}(d+1)^{\mathcal{O}^{\left((\ell+k)^{3}\right)}}\right)$. As the size of the lattice array is $n^{(d+1)}$ at most, the total complexity is $\mathcal{O}\left(n^{(d+1)} k^{\mathcal{O}(\ell+k)}(d+1)^{\mathcal{O}^{(\ell+k)^{3}}}\right)$. The second step
is the process of constructing $R$ by applying the pseudo-division algorithm to $f$ and $g$, which can be decomposed into a series of steps to eliminate the highest power in $x_{i}$. Assuming $\operatorname{deg}\left(F_{i}\right) \leq$ $\lambda$, by [32], the complexity of computing $R$ is $\mathcal{O}\left(\delta^{\mathcal{O}(\ell+n+k)}(\lambda+1)^{\mathcal{O}^{((\ell+n+k) k)}}\right)$ at most, and CIM eliminates the independent variables $\left\{u_{1}, \cdots, u_{n}\right\}$ and, so, the complexity of computing $R$ is $\mathcal{O}\left(n^{(d+1)} \delta^{\mathcal{O}(\ell+k)}(\lambda+1)^{\left.\mathcal{O}^{(\ell+k) k}\right)}\right.$. Thus, complexities of Wu-Ritt's method and CIM in steps 1 and 2 are $\mathcal{O}\left(k^{\mathcal{O}(\ell+n+k)^{2}}(d+1)^{\mathcal{O}^{(\ell+n+k)^{4}}}\right)$ and $\mathcal{O}\left(n^{(d+1)} k^{\mathcal{O}(\ell+k)^{2}}(d+1)^{\mathcal{O}^{(\ell+k)^{4}}}\right)$, respectively. Therefore, according to the above analysis, we can see that our approach is exponential in the degree of the involved polynomials, the number of variables as well as the number of the involved polynomials. However, according to [42], the complexity of computing Gröbner basis is $d^{2^{(\ell+n+k+\mathcal{O}(1))}}$, which is double exponential. Therefore, the approach in [3] is obviously more expensive than CIM approach.

## 4. Illustration of CIM

In this section, we illustrate CIM by an example taken from [3], and discuss some skills in computing $R$ by different kinds of implications generated in every step. Note that the main idea of CIM is "turning the difficulty of quality into the complexity of quantity" and, hence, can solve the complete inductive conditions, under which the approach by a comprehensive Gröbner basis in [3] is intractable.

$$
\begin{aligned}
& V=\left\{y, v_{y}, \delta\right\} \\
& L=\{l\} \\
& \mathcal{T}=\{\tau\}, \text { where } \\
& \tau:\left\langle l, l,\left[\begin{array}{c}
\delta>0 \wedge y=0 \wedge y^{\prime}=y \\
\wedge v_{y}^{\prime}=\frac{v_{y}}{2} \wedge \delta^{\prime}=0
\end{array}\right]\right\rangle \\
& \Theta=\left(y=0 \wedge v_{y}=16 \wedge \delta=0\right) \\
& F(l)=\left(\dot{y}=v_{y} \wedge \dot{\delta}=1 \wedge v_{y}=-10\right) \\
& I(l)=y \leq 0 \\
& \ell_{0}=l
\end{aligned}
$$

Example 3 (Bouncing Ball, from [3]). Figure 2 shows a graphical representation of a ball bouncing on a soft floor, which can be modeled as a hybrid system. The variable y represents the position of the ball (obviously, $y=0$ represents the ball being at floor level), $v_{y}$ represents its velocity, and $\delta$ denotes the time elapsed since its last bounce. A bounce is modeled by the transition $\tau$, in which the velocity $v_{y}$ of the ball is halved and the ball reverses direction.


Figure 2. The hybrid automaton for a bouncing ball.
Our approach consists of the following steps:

1. Guessing a template of fixed degree as an invariant template.
2. Defining the complete inductive conditions.
3. Applying a lattice array to the complete inductive conditions to generate the constraint on the parameters
4. Solving the constraint. Any solution to the constraint guarantees the template an inductive invariant

Different from the existing approaches, we use CIM to encode the complete inductive conditions. Step 1. Predefine the template map
Just as in [3], we set the degree of the invariant at 2, as there is only one location $l$. We set $\eta(l)$ to be a generic quadratic form on $\left\{y, v_{y}, \delta\right\}$, as follows:

$$
\eta(l):\left[\begin{array}{l}
a_{1} y^{2}+a_{2} v_{y}^{2}+a_{3} \delta^{2}+a_{4} y v_{y}+a_{5} v_{y} \delta \\
+a_{6} y \delta+a_{7} y+a_{8} v_{y}+a_{9} \delta+a_{10}
\end{array}\right]
$$

Step 2. Generate constraint for the initial condition

$$
\begin{equation*}
\left(y=0 \wedge v_{y}=16 \wedge \delta=0\right) \models \eta(l) \tag{16}
\end{equation*}
$$

In (16), there are three variables, which are constrained by three equations in $\Theta$. Thus, there are no independent variables. By Algorithm 1, $R=a_{10}+256 a_{2}+16 a_{8}$ and, then, we obtain the constraint $a_{10}+256 a_{2}+16 a_{8}=0$; the same result as in [3].

Step 3. Encode the discrete consecution
By Definition 11, the discrete consecution can be expressed by the following implication:

$$
\begin{align*}
& \eta(l)=0 \wedge \rho_{\tau} \wedge \alpha_{\tau} \models \eta(l)^{\prime}=0 \\
& \rho_{\tau}: y=0 \\
& \alpha_{\tau}:\left\{\begin{array}{l}
y^{\prime}=y \\
\delta^{\prime}=0 \\
v_{y}^{\prime}=-\frac{v_{y}}{2}
\end{array}\right. \tag{17}
\end{align*}
$$

1. In (17), there are six variables that are constrained by five equations. We might as well assume that $\left\{y, y^{\prime}, v_{y}^{\prime}, \delta, \delta^{\prime}\right\}$ are dependent variables, and $\left\{v_{y}\right\}$ is the independent variable. The degree of $\left\{v_{y}\right\}$ is 2 and the size of the lattice array is ( $2+1$ ). and, so, three implications are obtained (Table 2).
2. Applying Algorithm 1. We get three simpler implications and three corresponding $R$ by Lemma 3 (Table 2).

Table 2. Implications and $R$ generated by Citing Instances Method (CIM) on the encoding initial condition.

| $v_{y}$ | Implication | $\boldsymbol{R}$ |
| :---: | :---: | :---: |
| 0 | $a_{3} \delta^{2}+a_{9} \delta+a_{10}=0 \models a_{10}=0$ | $a_{10}$ |
| 1 | $a_{3} \delta^{2}+a_{5} \delta+a_{9} \delta+a_{10}+a_{2}+a_{8}=0 \models a_{10}+\frac{1}{4} a_{2}-\frac{1}{2} a_{8}=0$ | $a_{10}+\frac{1}{4} a_{2}-\frac{1}{2} a_{8}$ |
| 2 | $a_{3} \delta^{2}+2 a_{5} \delta+a_{9} \delta+a_{10}+4 a_{2}+2 a_{8}=0 \models a_{10}+a_{2}-a_{8}=0$ | $a_{10}+a_{2}-a_{8}$ |

At last, we get three constraint equations $\left\{a_{10}=0, a_{10}+\frac{1}{4} a_{2}-\frac{1}{2} a_{8}=0, a_{10}+a_{2}-a_{8}=0\right\}$. Solving, we obtain $\left\{a_{10}=0, a_{2}=0, a_{8}=0\right\}$.

Step 4. Encode the Continuous Consecution
By Definition 11, the continuous consecution can be expressed by the implication

$$
\begin{equation*}
I(l) \wedge(\eta(l)=0) \models(\eta \dot{l}(l)=0) \tag{18}
\end{equation*}
$$

where $\eta(l)$ is the Lie Derivative of $\eta(l)$ with respect to $F(l)$,

$$
\eta \dot{(l)}=\frac{\partial \eta(l)}{\partial y} \dot{y}+\frac{\partial \eta(l)}{\partial \delta} \dot{\delta}+\frac{\partial \eta(l)}{\partial v_{y}} \dot{v}_{y}
$$

that is,

$$
\begin{align*}
\eta(\ell) & =a_{4} v_{y}^{2}+2 a_{1} y v_{y}+a_{6} \delta v_{y}+\left(-20 a_{2}+a_{5}+a_{7}\right) v_{y}  \tag{19}\\
& +\left(2 a_{3}-10 a_{5}\right) \delta+\left(-10 a_{4}+a_{6}\right) y+\left(a_{9}-10 a_{8}\right)
\end{align*}
$$

In (18), there are three variables which are constrained by one equations (we ignore $I(l)=y \geq 0$ ). We might as well assume that $\left\{v_{y}\right\}$ is the dependent variable and that $\{y, \delta\}$ are independent variables. The degree of $y$ and $\delta$ is 2 and the size of the lattice array is ( $2+1,2+1$ ). Applying Algorithm 1 to (18). Thus, nine implications are obtained (Table 3). We analyze some implications to illustrate the domino effect:

1. $\quad 0=0 \models a_{4} v_{y}^{2}+\left(a_{5}+a_{7}\right) v_{y}+a_{9}=0$

By Algorithm 1, $F_{i}$ is a constant, the consequent of implication is $R$, i.e., $R=a_{4} v_{y}^{2}+\left(a_{5}+a_{7}\right) v_{y}+a_{9}$, and the constraint equations are $\left\{a_{5}=-a_{7}, a_{4}=0, a_{9}=0\right\}$. We will use this solution to simplify the remaining eight implications and the same below. Thus, the resulting implications become more and more simple (Table 4), we call it the domino effect, which is the benefit of parallelism in CIM.
2. $-a_{7} v_{y}+2 a_{3}=0 \mid=a_{6} v_{y}+2 a_{3}+10 a_{7}=0$

By Algorithm $1, R=2 a_{3} a_{7}+10 a_{7}^{2}+2 a_{3} a_{6}$ and $2 a_{3} a_{7}+10 a_{7}^{2}+2 a_{3} a_{6}=0$ is the constraint equation. However, by comprehensive Gröbner basis, we need to compute the remainder under two situations: $a_{7}=0$ and $a_{7} \neq 0$. If $a_{7}$ is polynomial, situation becomes very complicated. This is why Sankaranarayanan et al. defined Alternative Consecution Relations to eliminate template variables in the antecedent of implication to avoid construction of comprehensive Gröbner basis. However, every coin has two sides. In Wu-Ritt's Algorithm, $a_{7}=0$ means non-degenerate condition is false, the solution to $2 a_{3} a_{7}+10 a_{7}^{2}+2 a_{3} a_{6}=0$ may or may not satisfy the formula (5), which is crucial to the theorem's proof by CIM. Fortunately, for invariant generation, it only means we need another step to verify the obtained invariant. Generally, verifying invariants is less expensive than generating invariants.
3. $a_{1}+a_{7}=0=2 a_{1} v_{y}+a_{6}=0$

By Algorithm 1, $a_{1}+a_{7}$ is a constant, so $R=2 a_{1} v_{y}+a_{6}$ and $a_{1}=0, a_{6}=0$. We continue to simplify the implications(Table 5).
4. $-a_{7} v_{y}+a_{3}=0 \models 2 a_{3}+10 a_{7}=0$

By Algorithm 1, $R=2 a_{3}+10 a_{7}$ and the constraint is $a_{3}=-5 a_{7}$, continue to simplify the implications(Table 6).

Table 3. Implications generated by CIM on the encoding continuous consecution.

| $\delta$ | $\mathbf{y}$ | Implication |
| :---: | :---: | :---: |
| 0 | 0 | $0=0 \models a_{4} v_{y}^{2}+\left(a_{5}+a_{7}\right) v_{y}+a_{9}=0$ |
| 0 | 1 | $a_{4} v_{y}+a_{1}+a_{7}=0$ <br> $\models a_{4} v_{y}^{2}+2 a_{1} v_{y}+\left(a_{5}+a_{7}\right) v_{y}-10 a_{4}+a_{6}+a_{9}=0$ |
| 0 | 2 | $2 a_{4} v_{y}+4 a_{1}+2 a_{7}=0$ <br> $\models a_{4} v_{y}^{2}+4 a_{1} v_{y}+\left(a_{5}+a_{7}\right) v_{y}-20 a_{4}+2 a 6+a_{9}=0$ |
| 1 | 0 | $a_{5} v_{y}+a_{3}+a_{9}=0$ <br> $\models a_{4} v_{y}^{2}+a_{6} v_{y}+\left(a_{5}+a_{7}\right) v_{y}+2 a_{3}-10 a_{5}+a_{9}=0$ |
| 1 | 1 | $a_{4} v_{y}+a_{5} v_{y}+a_{1}+a_{3}+a_{6}+a_{7}+a_{9}=0$ <br> $\models a_{4} v_{y}^{2}+2 a_{1} v_{y}+a_{6} v_{y}+\left(a_{5}+a_{7}\right) v_{y}+2 a_{3}-10 a_{5}-10 a_{4}+a_{6}+a_{9}=0$ |
| 1 | 2 | $2 a_{4} v_{y}+a_{5} v_{y}+4 a_{1}+a_{3}+2 a_{6}+2 a_{7}+a_{9}=0$ <br> $\models a_{4} v_{y}^{2}+4 a_{1} v_{y}+a_{6} v_{y}+\left(a_{5}+a_{7}\right) v_{y}+2 a_{3}-10 a_{5}-20 a_{4}+2 a_{6}+a_{9}=0$ |

Table 3. Cont.

| $\delta$ | $\mathbf{y}$ | Implication |
| :---: | :---: | :---: |
| 2 | 0 | $2 a_{5} v_{y}+4 a_{3}+2 a_{9}=0$ <br> $\models a_{4} v_{y}^{2}+2 a_{6} v_{y}+\left(a_{5}+a_{7}\right) v_{y}+4 a_{3}-20 a_{5}+a_{9}=0$ |
| 2 | 1 | $a_{4} v_{y}+2 a_{5} v_{y}+a_{1}+4 a_{3}+2 a_{6}+a_{7}+2 a_{9}=0$ <br> $\models a_{4} v_{y}^{2}+2 a_{1} v_{y}+2 a_{6} v_{y}+\left(a_{5}+a_{7}\right) v_{y}+4 a_{3}-20 a_{5}-10 a_{4}+a_{6}+a_{9}=0$ |
| 2 | 2 | $2 a_{4} v_{y}+2 a_{5} v_{y}+4 a_{1}+4 a_{3}+4 a_{6}+2 a_{7}+2 a_{9}=0$ <br> $=a_{4} v_{y}^{2}+4 a_{1} v_{y}+2 a_{6} v_{y}+\left(a_{5}+a_{7}\right) v_{y}+4 a_{3}-20 a_{5}-20 a_{4}+2 a_{6}+a_{9}=0$ |

Table 4. Domino effect (1).

| $\delta$ | $\mathbf{y}$ | Implication |
| :---: | :---: | :---: |
| 0 | 0 | $0=0=0=0$ |
| 0 | 1 | $a_{1}+a_{7}=0 \models 2 a_{1} v_{y}+a_{6}=0$ |
| 0 | 2 | $4 a_{1}+2 a_{7}=0 \models 4 a_{1} v_{y}+2 a 6=0$ |
| 1 | 0 | $-a_{7} v_{y}+a_{3}=0 \models a_{6} v_{y}+2 a_{3}+10 a_{7}=0$ |
| 1 | 1 | $-a_{7} v_{y}+a_{1}+a_{3}+a_{6}+a_{7}=0 \models 2 a_{1} v_{y}+a_{6} v_{y}+2 a_{3}+10 a_{7}+a_{6}=0$ |
| 1 | 2 | $-a_{7} v_{y}+4 a_{1}+a_{3}+2 a_{6}+2 a_{7}=0 \models 4 a_{1} v_{y}+a_{6} v_{y}+2 a_{3}+10 a_{7}+2 a_{6}=0$ |
| 2 | 0 | $-2 a_{7} v_{y}+4 a_{3}=0=2 a_{6} v_{y}+4 a_{3}+20 a_{7}=0$ |
| 2 | 1 | $-2 a_{7} v_{y}+a_{1}+4 a_{3}+2 a_{6}+a_{7}=0 \models 2 a_{1} v_{y}+2 a_{6} v_{y}+4 a_{3}+20 a_{7}+a_{6}=0$ |
| 2 | 2 | $-2 a_{7} v_{y}+4 a_{1}+4 a_{3}+4 a_{6}+2 a_{7}=0=4 a_{1} v_{y}+2 a_{6} v_{y}+4 a_{3}+20 a_{7}+2 a_{6}=0$ |

Table 5. Domino effect (2).

| $\delta$ | $\mathbf{y}$ | Implication |
| :---: | :---: | :---: |
| 0 | 0 | $0=0 \models 0=0$ |
| 0 | 1 | $a_{7}=0 \models 0=0$ |
| 0 | 2 | $2 a_{7}=0 \models 0=0$ |
| 1 | 0 | $-a_{7} v_{y}+a_{3}=0 \models 2 a_{3}+10 a_{7}=0$ |
| 1 | 1 | $-a v_{7}+a_{3}+a_{7}=0 \models 2 a_{3}+10 a_{7}=0$ |
| 1 | 2 | $-a_{7} v_{y}+a_{3}+2 a_{7}=0 \models 2 a_{3}+10 a_{7}=0$ |
| 2 | 0 | $-2 a_{7} v_{y}+4 a_{3}=0 \models 4 a_{3}+20 a_{7}=0$ |
| 2 | 1 | $-2 a_{7} v_{y}+4 a_{3}+a_{7}=0 \models 4 a_{3}+20 a_{7}=0$ |
| 2 | 2 | $-2 a_{7} v_{y}+4 a_{3}+2 a_{7}=0 \models 4 a_{3}+20 a_{7}=0$ |

At last, the final constraint equations in the template variables are

$$
\begin{gathered}
a_{\{1,2,4,6,8,9,10\}}=0, \\
a_{5}+a_{7}=0, \\
5 a_{7}+a_{3}=0,
\end{gathered}
$$

and the corresponding invariant is $y=v_{y} \delta+5 \delta^{2}$ [3].

Table 6. Domino effect (3).

| $\delta$ | y | Implication |
| :---: | :---: | :---: |
| 0 | 0 | $0=0 \mid=0=0$ |
| 0 | 1 | $a_{7}=0 \models 0=0$ |
| 0 | 2 | $2 a_{7}=0 \models 0=0$ |
| 1 | 0 | $-a_{7} v_{y}-5 a_{7}=0 \models 0=0$ |
| 1 | 1 | $-2 a_{7} v_{y}-4 a_{7}=0 \models 0=0$ |
| 1 | 2 | $-a_{7} v_{y}-3 a_{7}=0 \models 0=0$ |
| 2 | 0 | $-2 a_{7} v_{y}-20 a_{7}=0 \models 0=0$ |
| 2 | 1 | $-2 a_{7} v_{y}-19 a_{7}=0 \mid=0=0$ |
| 2 | 2 | $-2 a_{7} v_{y}-18 a_{7}=0 \mid=0=0$ |

## 5. A Special Case

In this section, we show that Sankaranarayanan et al.'s approach is a special case of our method.
In Sankaranarayanan et al.'s approach, the antecedent of the discrete consecution implication $\left\{\eta\left(\ell_{i}\right)=0 \wedge \rho_{\tau} \wedge \alpha_{\tau} \vDash \eta\left(\ell_{j}\right)^{\prime}=0\right\}$ and continuous consecution implication $\{I(\ell) \wedge \eta(\ell)=$ $0 \models \eta(\ell)=0\}$ contain template variables, which requires the construction of a comprehensive Gröbner basis, a variant of the Gröbner basis [41]. Unfortunately, encoding inductive conditions by comprehensive Gröbner bases is an exact but impractical approach, as the non-linear constraints produced make the constraint-solving problem intractable, even for a simple hybrid system. Sankaranarayanan et al. defined as Alternative Consecution Relations (as shown in Tables 7 and 8) to eliminate template variables in the antecedent of implication. In doing this, (17), (18), and (5) are transformed into (20), (21), and (22), respectively. Obviously, (22) is a special case of (5), the antecedent of which does not contain any more parameters. In particular, if the transition relations are separable [3]-that is, each variable in $V^{\prime}$ is expressed as a polynomial expression over the variables in $V$-we can generate constraints more easily by CIM, as the antecedent of implication has been the ascending set.

$$
\begin{gather*}
\rho_{\tau} \wedge \alpha_{\tau} \models \eta(l)^{\prime}-\lambda \eta(l)=0,  \tag{20}\\
I(l) \models \eta(l)-\lambda \eta(l)=0,  \tag{21}\\
\bigwedge_{m=1}^{k} f_{m}\left(u_{1}, \ldots, u_{n}, x_{1}, \ldots, x_{k}\right)=0 \models \\
G\left(a_{1}, \ldots, a_{\ell}, u_{1}, \ldots, u_{n}, x_{1}, \ldots, x_{k}\right)=0 . \tag{22}
\end{gather*}
$$

Table 7. Alternative discrete consecution conditions. LC: local consecution; CV: constant-value consecution; CS: constant-scale consecution; PS: polynomial-scale consecution.

| Name | Alternative Discrete Consecution Condition |
| :---: | :---: |
| LC | $\alpha_{\tau} \wedge \rho_{\tau} \vDash \eta\left(l_{j}\right)^{\prime}=0$ |
| CV | $\alpha_{\tau} \wedge \rho_{\tau} \vDash\left(\eta\left(l_{j}\right)^{\prime}-\eta\left(l_{i}\right)\right)=0$ |
| CS | $\alpha_{\tau} \wedge \rho_{\tau} \vDash\left(\eta\left(l_{j}\right)^{\prime}-\lambda \eta\left(l_{1)}\right)\right)=0$ |
| PS | $\alpha_{\tau} \wedge \rho_{\tau} \vDash\left(\eta\left(l_{j}\right)^{\prime}-p \eta\left(l_{i}\right)\right)=0$ |

Table 8. Alternative Continuous Consecution Conditions. LC: local consecution; CV: constant-value consecution; CS: constant-scale consecution; PS: polynomial-scale consecution.

| Name | Alternative Continue Consecution Condition |
| :---: | :---: |
| LC | $I(l) \models \eta(l)=0$ |
| CV | $I(l) \models(\eta(l)-\eta(l))=0$ |
| CS | $I(l) \models(\eta(l)-\lambda \eta(l))=0$ |
| PS | $I(l) \models(\eta(l)-p \eta(l))=0$ |

We apply Algorithm 1 to (20) and (21).

1. For the initial condition case, it is same as Step 1.
2. For the discrete consecution case (with $\lambda d$ ), assume that $\left\{y, y^{\prime}, v_{y}^{\prime}, \delta^{\prime}\right\}$ are dependent variables, $\left\{v_{y}, \delta\right\}$ are independent variables, and the size of lattice array is $(2+1,2+1)$.
3. For the continuous consecution case (with $\lambda c$ ), there are no dependent variables as $I(l) \geq 0$ is ignored, $\left\{v_{y}, \delta, y\right\}$ are independent variables, and the size of lattice array is $(2+1,2+1,2+1)$.
4. Solving these 37 implications, the following two groups of constraints are obtained:

$$
\begin{align*}
& a_{1}=0, a_{2}=0, a_{3}=0, a_{4}=0, a_{5}=0 \\
& a_{6}=0, a_{7}=0, a_{8}=0, a_{9}=0, a_{10}=0  \tag{23}\\
& \lambda d=\lambda d, \lambda c=\lambda c \\
& a_{1}=0, a_{2}=0, a_{3}=-5 a_{7}, a_{4}=0, a_{5}=-a_{7} \\
& a_{6}=0, a_{7}=a_{7}, a_{8}=0, \quad a_{9}=0, a_{10}=0  \tag{24}\\
& \lambda d=0, \lambda c=0
\end{align*}
$$

Obviously, (23) is a trivial solution, (24) is the same as in [3], and the corresponding invariant is $y=v_{y} \delta+5 \delta^{2}$.

Remark 2. The main shortcoming of our approach lies in that CIM tends to produce many redundant instances, which results in more time consumption than Sankaranarayanan et al.'s approach ( $4.9 \mathrm{~s} v \mathrm{~s} .3 .7 \mathrm{~s}$ in Maple, respectively). However, CIM is intrinsic to be parallelized; under parallel circumstances, our method is less time-consuming ( 0.9 s vs. 3.7 s in Maple). In addition, the generation procedure for constraint equations in our approach is simpler (which can even be done by hand), as numerical computation takes the place of symbolic computation.

## 6. Experiments

To show the practicality of CIM, we present another two application examples. One is a train system, the other is a charged particle in a magnetic field.

### 6.1. Experiment 1

Consider a train system (from [3]). Figure 3 shows a hybrid automaton modeling a train accelerating (location $l_{0}$ ), traversing at constant speed $\left(l_{1}\right)$, and decelerating $\left(l_{2}\right)$. There exist three continuous variables in the train system: $x$, the position of the train; $v$, the train's velocity; and $t$, a master clock. The system has one discrete variable $s$, representing the number of stops made so far. The initial condition is given by $\{x=s=v=t=0\}$. There are three discrete transitions $\tau_{1}, \tau_{2}$, and $\tau_{3}$; the transition relations are as follows:

$$
\begin{array}{ll}
\rho_{\tau_{1}} \wedge \alpha_{\tau_{1}}: & v=5 \wedge i d(s, x, v, t) \\
\rho_{\tau_{2}} \wedge \alpha_{\tau_{2}}: & i d(s, x, v, t)  \tag{25}\\
\rho_{\tau_{3}} \wedge \alpha_{\tau_{3}}: & {\left[v=0 \wedge s^{\prime}=s+1 \wedge t^{\prime}=t+2 \wedge i d(x, v)\right]}
\end{array}
$$

where $i d(x)$ denotes $x^{\prime}=x$.


Figure 3. Hybrid automaton for the train system.
$\eta\left(l_{i}\right), i=0,1,2$ is the template map,

$$
\begin{gathered}
\eta\left(l_{0}\right):\left[\begin{array}{l}
a_{1} x^{2}+a_{2} x s+a_{3} x t+a_{4} x v \\
+a_{5} s^{2}+a_{6} s t+a_{7} s v \\
+a_{8} t^{2}+a_{10} v^{2}+a_{11} x \\
+a_{12} s+a_{13} t+a_{14} v+a_{15}
\end{array}\right], \\
\eta\left(l_{1}\right):\left[\begin{array}{l}
b_{1} x^{2}+b_{2} x s+b_{3} x t+b_{4} x v \\
+b_{5} s^{2}+b_{6} s t+b_{7} s v \\
+b_{8} t^{2}+b_{10} v^{2}+b_{11} x \\
+b_{12} s+b_{13} t+b_{14} v+b_{15}
\end{array}\right], \\
\eta\left(l_{2}\right):\left[\begin{array}{l}
c_{1} x^{2}+c_{2} x s+c_{3} x t+c_{4} x v \\
+c_{5} s^{2}+c_{6} s t+c_{7} s v \\
+c_{8} t^{2}+c_{10} v^{2}+c_{11} x \\
+c_{12} s+c_{13} t+c_{14} v+c_{15}
\end{array}\right]
\end{gathered}
$$

Applying Algorithm 1, 2250 instances in total and the following constraints are obtained:

$$
\begin{align*}
& a_{\{1, \ldots, 9,15\}}=0, \\
& a_{10}=-\frac{1}{4} b_{11} \lambda d_{12} \lambda d_{20} \\
& a_{11}=b_{11} \lambda d_{12} \lambda d_{20} \\
& a_{12}=\frac{115}{4} b_{11} \lambda d_{12} \lambda d_{20} \\
& b_{\{1, \ldots, 9\}}=0 \\
& b_{10}=\frac{1}{2} b_{11}, b_{11}=b_{11} \\
& b_{12}=\frac{115}{4} b_{11}, b_{13}=-5 b_{11} \\
& b_{14}=-5 b_{11}, b_{15}=-\frac{75}{4} b_{11}, \\
& c_{\{1, \ldots, 9\}}=0 \\
& c_{10}=\frac{1}{2} b_{11} \lambda d_{12},  \tag{26}\\
& c_{11}=b_{11} \lambda d_{12}, \\
& c_{12}=\frac{115}{4} b_{11} \lambda d_{12}, \\
& c_{13}=-5 b_{11} \lambda d_{12} \\
& c_{14}=-5 b_{11} \lambda d_{12}, \\
& c_{15}=-\frac{75}{4} b_{11} \lambda d_{12} \\
& \lambda c_{0}=0, \lambda c_{1}=0, \lambda c_{2}=0, \\
& \lambda d_{01}=\frac{1}{\lambda d_{12} \lambda d_{20}}, \\
& \lambda d_{12}=\lambda d_{12}, \\
& \lambda d_{20}=\lambda d_{20},
\end{align*}
$$

where $\lambda c_{i}$ denotes continuous consecution $\lambda$ at location $l_{i}$, and $\lambda d_{i j}$ denotes discrete consecution $\lambda$ for translation from $l_{i}$ to $l_{j}$.

When $b_{11}=1, \lambda d_{12}=\lambda d_{20}=1$, the invariant is obtained as follows:

$$
\begin{array}{ll}
\eta\left(l_{0}\right): & v^{2}-4 x-10 v+115 s+20 t=0 \\
\eta\left(l_{1}\right): & 5 v^{2}+4 x v+115 v s-20 v t=0  \tag{27}\\
\eta\left(l_{2}\right): & 2 v^{2}+4 x-20 v+115 s-20 t+75=0
\end{array}
$$

which the same as that derived in [3].

### 6.2. Experiment 2

In this experiment, we consider a charged particle in a magnetic field (from [3]).
Figure 4 shows a charged particle in a 2D-plane with a reflecting barrier at $x=0$ and a magnetic field at $x \leq d \leq 0$. There are eight system variables in total: the particle's position $x, y$; its velocity $v_{x}, v_{y}$; a bounce counter $b$, which is incremented every time the particle collides against the barrier at $x=0$; along with the parameters $a, d$, and time $t$. There also exist three locations: magnetic, right, and left. 729 instances are generated by Algorithm 1 and the following invariants were obtained at last:

$$
\begin{align*}
& \text { left }: \\
& \left\{\begin{array}{l}
v_{y}+2=0 \\
v_{x}-2=0 \\
a(x+y)=4 b-4 a b
\end{array}\right.  \tag{27}\\
& \text { magnetic }:\left\{\begin{array}{l}
v_{y}+2=a(x-2) \\
v_{x}^{2}+v_{y}^{2}=8 \\
v_{x}-2=-a(y+2)+4 b(1-a)
\end{array}\right. \\
& \text { right }:\left\{\begin{array}{l}
v_{y}+2=0 \\
v_{x}+2=0 \\
a(x-y)=-4(b+1)(1-a)
\end{array}\right.
\end{align*}
$$

which are the same as those obtained in [3].


Figure 4. A charged particle in a magnetic field and its non-linear hybrid model.

## 7. A Clever Use of CIM

The other method is that, in order to generate constraints without computing $R$, we can make a clever use of the left side of (9); that is, we only need to make $G$ vanish over a lattice array $S_{R}$. Once $G$ vanishes over $S_{R}, R$ naturally vanishes over $S_{R}$, which means that $R \equiv 0$, according to Lemma 2 . More specifically, for each $\hat{\boldsymbol{u}} \in S_{R}$, first substitute it for $\boldsymbol{u}$ in (11) (this also means that free variables are eliminated). Then, compute the roots $r\left(a_{1}, \cdots, a_{n}\right)$ of $x$ in $a$ and, finally, substitute $r\left(a_{1}, \cdots, a_{n}\right)$ for $x$ in (12). Now, we have a polynomial constraint $G\left(a_{1}, \cdots, a_{n}\right)=0$ in $a$. Solving $G\left(a_{1}, \cdots, a_{n}\right)=0$ generates a set of solutions to $\boldsymbol{a}$ which satisfy formula (5). However, there is an issue with this approach:
$r_{i}$ is not always easy to compute even though the free variables are replaced by numbers. In that case, we can experimentally obtain the roots of dependent variables. Here is an example.

We illustrate, in this example, how to experimentally obtain the roots of dependent variables. Figure 5 shows a graphical representation of an accumulator with varying accumulation every second, which can be modeled as a hybrid system where $t$ is a continuous-time variable, $s$ is the accumulator, and $i$ is the varying accumulation; the system will terminate when $i$ increases to 100 . Obviously, Figure 5 has an invariant $s=\frac{i(i+1)}{2}$.


Figure 5. The hybrid automaton for accumulator with varying accumulation every second.
The target template invariant is a generic degree-two template polynomial: $\eta=a_{0} s^{2}+a_{1} s i+$ $a_{2} s t+a_{3} s+a_{4} i^{2}+a_{5} i t+a_{6} i+a_{7} t^{2}+a_{8} t+a_{9}$. For the discrete consecution condition, we have

$$
\eta=0 \wedge i^{\prime}-i-1=0 \wedge s^{\prime}-s-i^{\prime}=0 \wedge t=1 \wedge t^{\prime}=0 \models \eta^{\prime}=0,
$$

assuming $\left\{s, i^{\prime}, s^{\prime}, t, t^{\prime}\right\}$ are the dependent variables and $\{i\}$ is the independent variable. By CIM, we need compute the roots of $\left\{s, i^{\prime}, s^{\prime}, t, t^{\prime}\right\}$ under the lattice array $\{i=0,1,2,3\}$.

Generally speaking, it is difficult to collect accurate continuous states during the dynamics of a system. When discrete translations are taken, however, we can easily collect continuous states and discrete states (as discrete translations are often controlled by computer software), such that we can collect the roots of $\left\{s, s^{\prime}, i^{\prime}, t, t^{\prime}\right\}$ in the log file by adding two statements to the discrete translation (Figure 6).


Figure 6. The hybrid automaton for accumulator.
The starting point is to run the system with $i=0, s=0, t=0$ and collect the roots of $\left\{s, i^{\prime}, s^{\prime}, t, t^{\prime}\right\}$ in the log file (Table 9) during every discrete transition.

Table 9. The experimentally collected hybrid states of $\left\{i, s, i^{\prime}, s^{\prime}, t, t^{\prime}\right\}$.

| $\mathbf{i}$ | $\mathbf{s}$ | $\mathbf{i}^{\prime}$ | $\mathbf{s}^{\prime}$ | $\mathbf{t}$ | $\mathbf{t}^{\prime}$ |
| :---: | :---: | :---: | :---: | :---: | :---: |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 2 | 3 | 1 | 0 |
| 2 | 3 | 3 | 6 | 1 | 0 |
| 3 | 6 | 4 | 10 | 1 | 0 |

Finally, we obtain the invariant $s=\frac{i(i+1)}{2}$. Without directly computing the roots, CIM can dramatically reduce the complexity. Similar ideas appeared in [43,44].

## 8. Conclusions

In this paper, we presented a new approach (called CIM) for invariant generation in hybrid systems. Some examples are given to illustrate how CIM works. The cornerstones of our technique are theories based on the solution number of polynomial equations and Wu-Ritt's Well-Ordering Theorem.

CIM can take the place of a comprehensive Gröbner basis for invariant generation. Furthermore, the authors are confident that most of the symbolic algebra used in existing methods can be replaced using this technique. Comparing to the well-established approaches in this field, CIM exhibits the following features which make it interesting:

1. Applying instances by essentially instantiating the free variables to real numbers. Hence, the free variables are removed.
2. Comparing to the existing approaches, CIM can solve the complete inductive conditions directly and is a complete approach.
3. According to our analysis, Wu-Ritt's method has exponential complexity while Gröbner basis method has double exponential complexity. Therefore, our approach is more efficient.
4. The main idea of CIM is "turning the difficulty of quality into the complexity of quantity". If Formula (5) is too difficult to deal with, CIM turns (5) into $\left|S_{R}\right|$ simpler implications without free variables, then CIM is meaningful.
5. Parallelization is another advantage of CIM. CIM was created to be a parallel method. To raise the computer's calculation capacity, one important method is to develop both parallel machines and parallel algorithms.

We also wish to extend CIM to the inequality field. In fact, the famous algorithm of cylindrical algebraic decomposition (CAD) [45] is a typical algorithm of proving by instances.

Many algorithms rely on the computation of (comprehensive) Gröbner basis [19]. If the whole task is too difficult to be accomplished using a Gröbner basis in one stroke, CIM can solve it by using many instances. From another perspective, CIM can be regarded as a Divide and Conquer (DAC)-type method. In Example 3, as it is too difficult to encode continuous consecution using a comprehensive Gröbner basis, we can first use CIM to obtain nine simpler implications and then apply (comprehensive) Gröbner basis, one-by-one. The combination of the CIM and (comprehensive) Gröbner basis is another interesting research goal.
Author Contributions: Conceptualization and Formal analysis, H.H. and J.W.; Investigation, H.H. All authors have read and agreed to the published version of the manuscript.
Funding: This work was partly supported by the National Natural Science Foundation of China under Grant No. 61772006, the Science and Technology Program of Guangxi under Grant No. AB17129012, the Science and Technology Major Project of Guangxi under Grant No. AA17204096,the Special Fund for Scientific and Technological Bases and Talents of Guangxi under Grant No. AD16380076, and the Special Fund for Bagui Scholars of Guangxi.
Conflicts of Interest: The authors declare no conflict of interest.

## References

1. Henzinger, T.A. The theory of hybrid automata. Available online: https://pub.ist.ac.at/~tah/Publications/ the_theory_of_hybrid_automata.pdf (accessed on 10 April 2020).
2. Alur, R. Formal verification of hybrid systems. In Proceedings of the 2011 Ninth ACM International Conference on Embedded Software (EMSOFT), Taipei, Taiwan, 9-14 October 2011; pp. 273-278.
3. Sankaranarayanan, S.; Sipma, H.B.; Manna, Z. Constructing invariants for hybrid systems. Form. Methods Syst. Des. 2008, 32, 25-55. [CrossRef]
4. Kong, H.; He, F.; Song, X.; Hung, W.N.; Gu, M. Exponential-condition-based barrier certificate generation for safety verification of hybrid systems. In Proceedings of the 25th International Conference on Computer Aided Verification (CAV2013), Saint Petersburg, Russia, 13-19 July 2013; pp. 242-257.
5. Alur, R.; Dill, D.L. A theory of timed automata. Theor. Comput. Sci. 1994, 125, 183-235. [CrossRef]
6. Henzinger, T.A.; Kopke, P.W.; Puri, A.; Varaiya, P. What's decidable about hybrid automata? J. Comput. Syst. Sci. 1998, 57, 94-124. [CrossRef]
7. Bogomolov, S.; Donzé, A.; Frehse, G. Guided search for hybrid systems based on coarse-grained space abstractions. Int. J. Softw. Tools Technol. Transf. 2015, 18, 1-19.
8. Bogomolov, S.; Frehse, G.; Grosu, R. A box-based distance between regions for guiding the reachability analysis of SpaceEx. In Proceedings of the 24th International Conference on Computer Aided Verification (CAV 2012), Berkeley, CA, USA, 7-13 July 2012; pp. 479-494.
9. Kong, H.; Bartocci, E.; Henzinge, T.A. Reachable Set Over-Approximation for Nonlinear Systems Using Piecewise Barrier Tubes. In Proceedings of the 30th International Conference on Computer Aided Verification (CAV2018), Oxford, UK, 14-17 July 2018; pp. 449-467.
10. Tiwari, A. Abstractions for hybrid systems. Form. Methods Syst. Des. 2008, 32, 57-83. [CrossRef]
11. Bogomolov, S.; Herrera, C.; Muniz, M. Quasi-dependent variables in hybrid automata. In Proceedings of the 17th Int. Workshop on Hybrid Systems: Computation and Control, Berlin, Germany, 15-17 April 2014; pp. 93-102.
12. Bogomolov, S.; Schilling, C.; Bartocci, E. Abstraction-based parameter synthesis for multiaffine systems. In Proceedings of the 11th Haifa Verification Conference (HVC 2015), Haifa, Israel, 17-19 November 2015; pp. 19-35.
13. Bogomolov, S.; Frehse, G.; Greitschus, M. Assume-guarantee abstraction refinement meets hybrid systems. In Proceedings of the 10th Haifa Verification Conference (HVC 2014), Haifa, Israel, 18-20 November 2014; pp. 116-131.
14. Boreale, M. Algorithms for exact and approximate linear abstractions of polynomial continuous systems. In Proceedings of the 21st ACM International Conference on Hybrid Systems: Computation and Control, Porto, Portugal, 11-13 April 2018; pp. 207-216.
15. Kong, H.; Bogomolov, S. Safety Verification of Nonlinear Hybrid Systems Based on Invariant Clusters. In Proceedings of the 20th ACM International Conference on Hybrid Systems: Computation and Control, Pittsburgh, PA, USA, 18-20 April 2017; pp. 163-172.
16. Zhang, J.; Yang, L.; Deng, M. The parallel numerical method of mechanical theorem proving. Theor. Comput. Sci. 1990, 74, 253-271. [CrossRef]
17. Zhang, J.; Lu, Y. Principles of parallel numerical method and single-instance method of mechanical theorem proving (in Chinese). J. Math. Pract. Theory 1989, 1, 36-45.
18. Rodríguez-Carbonell, E.; Tiwari, A. Generating Polynomial Invariants for Hybrid Systems In Proceedings of the 8th International Workshop on Hybrid Systems: Computation and Control, Zurich, Switzerland, 9-11 March 2005; pp. 590-605.
19. Cachera, D.; Jensen, T.; Jobin, A.; Kirchner, F. Inference of polynomial invariants for imperative programs: A farewell to gröbner bases. Sci. Comput. Program. 2014, 93, 89-109. [CrossRef]
20. Roux, P.; Voronin, Y. L.; Sankaranarayanan, S. Validating numerical semidefinite programming solvers for polynomial invariants. Form. Methods Syst. Des. 2018, 53, 286-312. [CrossRef]
21. Tiwari, A.; Khanna, G. Nonlinear systems: Approximating reach sets. In Proceedings of the 7th International Workshop on Hybrid Systems: Computation and Control, Philadelphia, PA, USA, 25-27 March 2004; pp. 171-174.
22. Prajna, S.; Jadbabaie, A. Safety verification of hybrid systems using barrier certificates. In Proceedings of the International Workshop on Hybrid Systems: Computation and Control, Philadelphia, PA, USA, 25-27 March 2004; pp. 271-274.
23. Johnson, T.T.; Mitra, S. Invariant synthesis for verification of parameterized cyber-physical systems with applications to aerospace systems. In Proceedings of the AIAA Infotech at Aerospace Conference, Boston, MA, USA, 19-22 August 2013; p. 4811.
24. Gulwani, S.; Tiwari, A. Constraint-based approach for analysis of hybrid systems. In Proceedings of the 20th International Conference on Computer Aided Verification, Princeton, NJ, USA, 7-14 July 2008; pp. 190-203.
25. Sassi, B.; Amin, M.; Girard, A.; Sankaranarayanan, S. Iterative computation of polyhedral invariants sets for polynomial dynamical systems. In Proceedings of the 53rd IEEE Conference on Decision and Control (IEEE CDC2014), Los Angeles, CA, USA, 15-17 December 2014; pp. 6348-6353.
26. Liu, J.; Zhan, N.; Zhao, H. Computing semi-algebraic invariants for polynomial dynamical systems. In Proceedings of the 11th International Conference on Embedded Software (EMSOFT 2011), Taipei, Taiwan, 9-14 October 2011; pp. 97-106.
27. Boreale, M. Complete Algorithms for Algebraic Strongest Postconditions and Weakest Preconditions in Polynomial ODE'S. In Proceedings of the 44th International Conference on Current Trends in Theory and Practice of Informatics, Krems, Austria, 29 January-2 February 2018; pp. 442-455.
28. Gallo, G.; Mishra, B. Wu-Ritt Characteristic Sets and Their Complexity. Discret. Comput. Geom. 1991, 6, 111-136.
29. Ritt, J.F. Differential Algebra; American Mathematical Society: Providence, RI, USA, 1950.
30. Wu, W. On the decision problem and the mechanization of theorem-proving in elementary geometry. Sci. China 1978, 29, 117-138.
31. Wu, W. Basic principles of mechanical theorem proving in elementary geometries. J. Autom. Reason. 1986, 2, 221-252.
32. Gallo, G.; Mishra, B. Efficient algorithms and bounds for Wu-Ritt characteristic sets. Eff. Methods Algebr. Geom. 1991, 94, 119-142.
33. Alur, R.; Courcoubetis, C.; Halbwachs, N.; Henzinger, T.A. The algorithmic analysis of hybrid systems. Theor. Comput. Sci. 1995, 138, 3-34. [CrossRef]
34. Maler, O.; Manna, Z.; Pnueli, A. From timed to hybrid systems. In Proceedings of the REX workshop 1991, Mook, The Netherlands, 3-7 June 1991; pp. 447-484.
35. Alur R.; Henzinger, T.A. Modularity for timed and hybrid systems. In Proceedings of the 8th InInternational Conference on Concurrency Theory (CONCUR 1997), Warsaw, Poland, 1-4 July 1997; pp. 74-88.
36. Lygeros, J.; Tomlin, C.; Sastry, S. Controllers for reachability specifications for hybrid systems. Automatica 1999, 35,349-370. [CrossRef]
37. Floyd, R.W. Assigning meanings to programs. In Proceedings of the Amer. Math. Soc. Symp. in Applied Mathematics, Providence, RI, USA, 5-7 April 1967; pp. 19-32.
38. Hoare, C.A.R. An axiomatic basis for computer programming. Commun. ACM 1983, 12, 53-56. [CrossRef]
39. Wu, W.T. Mathematics Mechanization: Mechanical Geometry Theorem-Proving, Mechanical Geometry Problem-Solving, and Polynomial Equations-Solving; Kluwer Academic Publishers: Norwell, MA, USA, 2001.
40. Cox, D.; Little, J.; O'Shea, D. Ideals, varieties, and algorithms. Am. Math. Mon. 1994, 101, 582-586.
41. Weispfenning, V. Comprehensive gröbner bases. J. Symb. Comput. 1992, 14, 669-683. [CrossRef]
42. Dubé, T.W. The structure of polynomial ideals and gröbner bases. Siam J. Comput. 1990, 19, 750-773. [CrossRef]
43. Sharma, R.; Gupta, S.; Hariharan, B.; Aiken, A.; Liang, P.; Nori, A.V. A data driven approach for algebraic loop invariants. In Proceedings of the 22nd European Conference on Programming Languages and Systems, Rome, Italy, 16-24 March 2013; pp. 574-592.
44. Padhi, S.; Sharma, R.; Millstein, T. Data-driven precondition inference with learned features. ACM Sigplan Not. 2016, 51, 42-56. [CrossRef]
45. Collins, G. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In Proceedings of the 2nd GI Conference on Automata Theory and Formal Languages, Kaiserslautern, Germany, 20-23 May 1975; pp. 134-183.
© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).
