Noname manuscript No.
(will be inserted by the editor)

From Affine to Polynomial: Synthesizing Loops with Branches via Algebraic Geometry

Erdenebayar Bayarmagnai · Fatemeh Mohammadi · Rémi Prébet

Abstract Ensuring software correctness remains a fundamental challenge in formal program verification. One promising approach relies on finding polynomial invariants for loops. Polynomial invariants are properties of a program loop that hold before and after each iteration. Generating such invariants is a crucial task in loop analysis, but it is undecidable in the general case. Recently, an alternative approach to this problem has emerged, focusing on synthesizing loops from invariants. However, existing methods only synthesize affine loops without guard conditions from polynomial invariants. In this paper, we address a more general problem, allowing loops to have polynomial update maps with a given structure, inequations in the guard condition, and polynomial invariants of arbitrary form.

We use algebraic geometry tools to design and implement an algorithm that computes a finite set of polynomial equations whose solutions correspond to all nondeterministic branching loops satisfying the given invariants. Furthermore, we introduce a new class of invariants for which we present a significantly more efficient algorithm. In other words, we reduce the problem of synthesizing loops to find solutions of multivariate polynomial systems with rational entries. This final step is handled in our software using an SMT solver.

E. Bayarmagnai
KU Leuven, Department of Computer Science
E-mail: erdenebayar.bayarmagnai@kuleuven.be
F.Mohammadi
KU Leuven, Department of Computer Science
E-mail: fatemeh.mohammadi@kuleuven.be
R.Prébet
Inria, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP, UMR 5668, 69342, Lyon cedex 07, France
E-mail: remi.prebet@ens-lyon.fr

1 Introduction

Loop invariants are properties that hold before and after each iteration of a loop. They play a central role in automating program verification, which guarantees program correctness prior to execution. Various well-established methods rely on loop invariants for safety verification, including the Floyd–Hoare inductive assertion technique [16] and termination verification using standard ranking functions [15]. When a loop invariant takes the form of a polynomial equation or inequality, it is referred to as a polynomial invariant. In this paper, we restrict our attention to invariants given by polynomial equations.

In this work, instead of generating polynomial invariants for a given loop, we address the reverse problem, that is synthesizing a loop that satisfies a specified set of polynomial invariants.

We consider polynomial loops $\mathcal{L}(\mathbf{a}, h, F)$ of the form

$$\begin{array}{l}\mathbf{x} := (x_1, \ldots, x_n) \leftarrow \mathbf{a} := (a_1, \ldots, a_n)\\ \textbf{while } h(\mathbf{x}) \neq 0 \textbf{ do}\\ \quad \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \xleftarrow{F} \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{pmatrix}\\ \textbf{end while}\end{array}$$

Here, x_i are the program variables with initial values a_i, $h \in \mathbb{C}[\mathbf{x}]$, and $F = (F_1, \ldots, F_n)$ is a sequence of polynomials in $\mathbb{C}[\mathbf{x}]$. The condition $h(\mathbf{x}) \neq 0$, called the *guard*, is assumed to be a single inequation for simplicity. This is without loss of generality, since a guard of the form $h_1 \neq 0, \ldots, h_k \neq 0$ can be replaced by the product condition $h_1 \cdot \ldots \cdot h_k \neq 0$. When no guard is present, we write $\mathcal{L}(\mathbf{a}, 1, F)$, which corresponds to an infinite loop.

In the following, we introduce some terminology from algebraic geometry. For further details, we refer the

reader to [10,13,14]. We denote the field of complex numbers by $\mathbb{C}$. Throughout the paper, $\mathbf{x}$ denotes the tuple of indeterminates $x_1, \ldots, x_n$, and $\mathbb{C}[\mathbf{x}]$ the multivariate polynomial ring in these variables.

Ideals. A polynomial ideal I is a subset of $\mathbb{C}[\mathbf{x}]$ that is closed under addition, $0 \in I$ and for any $f \in \mathbb{C}[\mathbf{x}]$ and $g \in I$, $fg \in I$. Given a subset S of $\mathbb{C}[\mathbf{x}]$, the ideal generated by S is

$$\langle S \rangle = \{a_1 f_1 + \ldots + a_m f_m \mid a_i \in \mathbb{C}[\mathbf{x}], f_i \in S, m \in \mathbb{N}\}.$$

By Hilbert Basis Theorem [10, Theorem 4, Chap. 2], every ideal is generated by finitely many polynomials. For a subset $X \subset \mathbb{C}^n$, the defining ideal $I(X)$ of X is the set of polynomials that vanish on X. The radical ideal of an ideal $I \subset \mathbb{C}[\mathbf{x}]$ is defined as

$$\sqrt{I} = \{f \in \mathbb{C}[\mathbf{x}] | \, f^m \in I \text{ for some } m \in \mathbb{N}\}.$$

Varieties. For $S \subset \mathbb{C}[x]$, the algebraic variety $\boldsymbol{V}(S)$ is the common zero set of all polynomials in S. Moreover, $\boldsymbol{V}(S) = \boldsymbol{V}(\langle S \rangle)$ and so every algebraic variety is the vanishing locus of finitely many polynomials.

Polynomial maps. A map $F : \mathbb{C}^n \longrightarrow \mathbb{C}^m$ is a polynomial map if there exist $f_1, \ldots, f_m \in \mathbb{C}[x_1, \ldots, x_m]$ such that

$$F(x) = (f_1(x), \ldots, f_m(x))$$

for all $x \in \mathbb{C}^n$. For simplicity, we will identify polynomial maps and their corresponding polynomials.

In the following definition, $F^{(l)}(x)$ is defined recursively as $F^{(l)}(x) = F(F^{(l-1)}(x))$ for any $l > 1$ and $F^{(0)}(x) = x$ by convention.

Definition 1 A polynomial g is an invariant of the loop $\mathcal{L}(\mathbf{a}, h, F)$ if, for any $m \in \mathbb{Z}_{\geq 0}$, either

$$g(F^{(m)}(\mathbf{a})) = 0,$$

or there exists $m \in \mathbb{Z}_{\geq 0}$ such that

$$g(F^{(m)}(\mathbf{a})) = h(F^{(m)}(\mathbf{a})) = 0,$$

and for every $0 \leq l < m$

$$g(F^{(l)}(\mathbf{a})) = 0 \text{ and } h(F^{(l)}(\mathbf{a})) \neq 0.$$

Definition 2 Let $\mathcal{L}(\mathbf{a}, h, F)$ be a polynomial loop. The set of all polynomial invariants for $\mathcal{L}(\mathbf{a}, h, F)$ is called the *invariant ideal* of $\mathcal{L}$ and is denoted by $I_{\mathcal{L}(\mathbf{a},h,F)}$.

The invariant ideal is indeed known (e.g. [4]) to be an ideal of $\mathbb{C}[x_1, \ldots, x_n]$ where $x_1, \ldots, x_n$ are the program variables. Consider polynomials $f_1, \ldots, f_n$ in $\mathbb{C}[x_1, \ldots, x_n]$, then we will denote by

$$\text{span}\{f_1, \ldots, f_n\}$$

the vector space they generate.

Definition 3 Let $\mathbf{f}_1 = (f_{1,1}, \ldots, f_{1,l_1}), \ldots, \mathbf{f}_n = (f_{n,1}, \ldots, f_{n,l_n})$ and $(F_1, \ldots, F_n)$ be sequences of polynomials in $\mathbb{C}[\mathbf{x}]$ such that for every i,

$$F_i = \sum_{j=1}^{l_i} b_{i,j} f_{i,j}$$

for some $b_{i,j} \in \mathbb{C}$. Let $\mathbf{f} = (\mathbf{f}_1 \ldots, \mathbf{f}_n)$ and define a map

$$F_{\mathbf{f},\mathbf{b}} : \mathbb{C}^n \longrightarrow \mathbb{C}^n$$

with $F_{\mathbf{f},\mathbf{b}}(x) = (F_1(x), \ldots, F_n(x))$.

The object defined below is the primary focus of this paper. We prove that it is an algebraic variety and compute the polynomial equations that define it.

Definition 4 Using the notation of Definition 3, let $h \in \mathbb{C}[\mathbf{x}]$, $\boldsymbol{g} = (g_1, \ldots, g_m)$ be a sequence of polynomials in $\mathbb{C}[\mathbf{x}]$, and $\mathbf{a} \in \mathbb{C}^n$. The *coefficient set* of the polynomial loop structured by $\mathbf{f}$ with invariants $\boldsymbol{g}$ is defined as

$$\mathscr{C}(\mathbf{a}, h, \mathbf{f}; \boldsymbol{g}) = \left\{ \mathbf{b} \in \mathbb{C}^{l_1 + \cdots + l_n} \mid \boldsymbol{g} \subset I_{\mathcal{L}(\mathbf{a},h,F_{\mathbf{f},\mathbf{b}})} \right\}.$$

In simple words, $\mathscr{C}(\mathbf{a}, h, \mathbf{f}; \boldsymbol{g})$ is the set of all coefficient vectors $\mathbf{b} \in \mathbb{C}^{l_1 + \cdots + l_n}$ such that all polynomials in $\boldsymbol{g}$ are polynomial invariants of the following loop:

```
x ← a
while h(x) ≠ 0 do
    x ← F_{f,b}(x)
end while
```

In prior works [5,24,26], restricting computations to affine loops corresponds to choose $\mathbf{f}_i = (1, x_1, \ldots, x_n)$, for all $1 \leq i \leq n$.

Related work. The computation of polynomial invariants for loops has been an active area of research over the past two decades [1,3,12,8,19,4,11,7,20]. However, computing invariant ideals is undecidable for general loops [17]. Consequently, efficient methods have been developed for restricted classes of loops, particularly those in which the assertions are linear or can be reduced to linear form.

The converse problem, namely synthesizing loops from given invariants, has received considerably less attention in the literature. Most existing work has focused on linear and affine loops. Synthesizing linear loops that satisfy given invariants was shown to be NP-hard in [23]. These studies differ primarily in the types of invariants considered: linear invariants in [28], a single quadratic polynomial in [24], and pure difference binomials in [5]. In contrast, [26] considers general polynomial invariants but lacks completeness guarantees and does not address guard conditions.

Higher-degree loops have been investigated in a more general framework in [18], which also considers polynomial inequalities as input. However, the loops synthesized by that algorithm are restricted to cases where the input invariants are inductive, that is if an invariant holds after one iteration, it continues to hold for all subsequent iterations.

Our contributions. In this work, we consider the following situation: the generation of polynomial loops with guards from arbitrary polynomial invariants (equalities). We take the first step towards this goal by generating a polynomial system whose solutions correspond exactly to loops with a given structure that satisfy the specified polynomial invariants. We then discuss different strategies that can be used to solve this system. In practice, in most cases a satisfying solution is found using an SMT solver.

More precisely, in Section 2 we recall the definition of invariant sets and relevant results from [25]. Section 3 introduces a method for simultaneously identifying multiple polynomial invariants (Proposition 3) and applies it to show that the set $\mathscr{C}(\mathbf{a}, h, \mathbf{f}; \boldsymbol{g})$, the collection of all coefficients of polynomial maps of loops satisfying the polynomial invariants $\boldsymbol{g}$ with respect to $\mathbf{f}$, forms an algebraic variety. We also present Algorithm 2, which computes the defining polynomials of $\mathscr{C}(\mathbf{a}, h, \mathbf{f}; \boldsymbol{g})$. Section 4 discusses several strategies for solving the resulting polynomial systems and highlights their limitations. Finally, in Section 5 we describe our implementation of the algorithms and report the experimental results.

Extended version. This paper is an extended version of the paper [40], published in the Proceedings of the 14th ACM SIGPLAN International Workshop on the State of the Art in Program Analysis. It expands on the earlier work by providing extended results and introducing new contributions. Specifically, Section 3.2 generalizes our previous work by addressing branching loops with nondeterministic conditional statements, whereas the earlier paper focused only on single-path loops. Section 3.3 introduces a new class of invariants, and we develop Algorithm 3 to compute polynomial equations that characterize all branching loops with a given structure that satisfy these invariants. In contrast to Algorithm 2, Algorithm 3 avoids costly Gröbner basis computation. In addition, when the given invariants are affine, Proposition 8 shows that the set of coefficients of the update map for a loop satisfying these invariants forms an affine space. Algorithm 4 is then used to compute an affine basis for this space.

2 Preliminaries

Here, we introduce the main objects of this paper and extend key results from [25, 1].

2.1 Invariant sets for a single polynomial map

Definition 5 Let $F : \mathbb{C}^n \longrightarrow \mathbb{C}^n$ be a polynomial map and $X \subset \mathbb{C}^n$. The invariant set of (F, X) is defined as:

$$S_{(F,X)} = \{x \in X \mid \forall m \in \mathbb{N}, F^{(m)}(x) \in X\}.$$

The following proposition, adapted from [25, Proposition 2.3], is the key result enabling the invariant sets to be computed algebraically.

Proposition 1 *Let $X \subseteq \mathbb{C}^n$ be an algebraic variety and consider a polynomial map $F : \mathbb{C}^n \longrightarrow \mathbb{C}^n$. We define*

$$X_m = X \cap F^{-1}(X) \cap \ldots \cap F^{-m}(X)$$

for all $m \in \mathbb{N}$. Then, there exists $N \in \mathbb{N}$ such that $X_N = X_{N+1}$, and for any such index $X_N = S_{(F,X)}$.

Building on Proposition 1, the following algorithm computes invariant sets by means of an iterative outer approximation that converges in finitely many steps. The algorithm requires the following procedures:

- `Compose`: given two sequences of polynomials $\boldsymbol{g} = (g_1, \ldots, g_m)$ and $F = (F_1, \ldots, F_n)$ in $\mathbb{Q}[\mathbf{x}]$, it returns
$$g_1\big(F_1(\mathbf{x}), \ldots, F_n(\mathbf{x})\big),\ \ldots,\ g_m\big(F_1(\mathbf{x}), \ldots, F_n(\mathbf{x})\big).$$
- `InRadical`: given a sequence of polynomials $\widetilde{\boldsymbol{g}}$ and a finite set $S \subset \mathbb{Q}[\mathbf{x}]$, it outputs `True` if $\widetilde{\boldsymbol{g}} \subset \sqrt{\langle S \rangle}$, and `False` otherwise.

These procedures are standard routines in symbolic computation and the latter one can be carried out efficiently using techniques such as Gröbner bases [10]. For further details, we refer the reader to [1].

Algorithm 1 InvariantSet

Input: sequences $\boldsymbol{g}$ and $F = (F_1, \ldots, F_n)$ in $\mathbb{Q}[\mathbf{x}]$.
Output: polynomials of common zero-set $S_{(F, \boldsymbol{V}(\boldsymbol{g}))}$.

```
S ← {g};
g̃ ← Compose(g, F);
while InRadical(g̃, S) == False do
    S ← S ∪ {g̃};
    g̃ ← Compose(g̃, F);
end while
return S;
```

The termination and correctness of Algorithm 1 follow from Proposition 1 and [25, Theorem 2.4].

Example 1 Consider the polynomial map and the algebraic variety:

$$F = (2x_1 - 3x_2,\, x_1 + x_2) \text{ and } X = \boldsymbol{V}(x_1^2 - x_2^2 + x_1x_2),$$

respectively. On input F and $g = x_1^2 - x_2^2 + x_1x_2$, Algorithm 1 computes the invariant set of (F, X) through the following steps:

- At Step 1, $S \leftarrow \{g\}$. At Step 2, $\widetilde{g}$ is set to
$$\mathsf{Compose}(g, F) = 5x_1^2 - 15x_1x_2 + 5x_2^2.$$
- At Step 3, a Gröbner basis of the ideal $\langle g,\, 1 - t\widetilde{g}\rangle$ is computed which makes the following decision
$$\mathsf{InRadical}(\widetilde{g}, S) = \texttt{False}.$$
- At Step 4, $S \leftarrow \{g,\, g \circ F\}$. At Step 5, $\widetilde{g}$ is updated as
$$\mathsf{Compose}(\widetilde{g}, F) = -5x_1^2 - 35x_1x_2 + 95x_2^2.$$
- This time, $\mathsf{InRadical}(\widetilde{g}, S) = \texttt{True}$, so the while-loop terminates.

Therefore, the invariant set is $S_{(F,\boldsymbol{V}(g))} = \boldsymbol{V}(g,\, g \circ F)$.

2.2 Invariants sets for multiple polynomial loops

In many cases, we are interested not in a single polynomial loop but in systems with multiple loops or updates. This subsection generalizes the notion of invariant sets to the setting of multiple polynomial loops.

Definition 6 Let $F_1, \ldots, F_k : \mathbb{C}^n \to \mathbb{C}^n$ be polynomial maps. For any $m \in \mathbb{N}$ and any sequence $i_1, \ldots, i_m \in [k]$, define the polynomial map

$$F_{i_1,\ldots,i_m}(\mathbf{x}) = F_{i_m}\big(F_{i_{m-1}}(\cdots F_{i_1}(\mathbf{x}))\big).$$

The next definition extends Definition 5 to the setting of multiple polynomial maps.

Definition 7 Consider the set $X \subseteq \mathbb{C}^n$ and the polynomial maps $F_1, \ldots, F_k : \mathbb{C}^n \to \mathbb{C}^n$. The invariant set of $((F_1, \ldots, F_k), X)$ is defined as

$$\Big\{x \in X \mid \forall m \geq 1, \forall i_1, \ldots, \forall i_m \in [k],\ F_{i_m}\big(\ldots(F_{i_1}(x)\big) \in X\Big\}$$

where $[k] = \{1, \ldots, k\}$. We note it $S_{((F_1,\ldots,F_k),X)}$.

The next proposition introduce an algorithm for computing invariant sets of multiple polynomial maps. For more details, this is an adaptation of in [1, Theorem 3.14].

Proposition 2 *Let $F_1, \ldots, F_k$ and $\boldsymbol{g}$ be sequences of polynomial in $\mathbb{Q}[\mathbf{x}]$. Define* InvariantSetBranch *the modified version of Algorithm 1, obtained by replacing*

- $\mathsf{Compose}(\boldsymbol{g}, F)$ *with the tuple*
$$\big(\mathsf{Compose}(\boldsymbol{g}, F_1), \ldots, \mathsf{Compose}(\boldsymbol{g}, F_k)\big)$$
in Step 2;, and
- $\mathsf{Compose}(\widetilde{\boldsymbol{g}}, F)$ *with the tuple*
$$\big(\mathsf{Compose}(\widetilde{\boldsymbol{g}}, F_1), \ldots, \mathsf{Compose}(\widetilde{\boldsymbol{g}}, F_k)\big)$$
in Step 5.

Then, on input the F_i and $\boldsymbol{g}$, InvariantSetBranch *computes polynomials whose common zero-set is the invariant set of $((F_1, \ldots, F_k), \boldsymbol{V}(\boldsymbol{g}))$.*

3 From loops to polynomial systems

In this section, we present a method for identifying all (branching) loops that satisfy given polynomial invariants by formulating the problem as a polynomial system over invariant sets. This reduces loop synthesis to solving a polynomial system, which will be examined in detail in the next section.

3.1 Single-path loops

We first consider *single-path loops*, i.e. loops defined by a single polynomial update map $F : \mathbb{C}^n \to \mathbb{C}^n$ with an inequation $h(x) \neq 0$ in the guard condition. This case provides the basic setting in which to study invariant sets, before extending the results to branching loops in Section 3.2.

The following lemma, which is an immediate consequence of Definition 5, shows that invariant sets are preserved under intersections of algebraic varieties.

Lemma 1 *Let $F : \mathbb{C}^n \to \mathbb{C}^n$ be a polynomial map, and let $X_1, \ldots, X_m \subseteq \mathbb{C}^n$ be algebraic varieties. Then*

$$S_{(F,\, X_1 \cap \cdots \cap X_m)} \;=\; S_{(F,X_1)} \cap \cdots \cap S_{(F,X_m)}.$$

We now provide a necessary and sufficient condition for determining whether a given set of polynomials forms invariants of a loop. Furthermore, the next proposition extends [1, Proposition 2.7], where the statement was established for a single polynomial invariant.

Proposition 3 *Let $h, g_1, \ldots, g_m \in \mathbb{C}[\mathbf{x}]$, and let z be a new indeterminate. Define*

$$X = \boldsymbol{V}(zg_1, \ldots, zg_m) \subset \mathbb{C}^{n+1}.$$

Let $F : \mathbb{C}^n \to \mathbb{C}^n$ be a polynomial map, and define

$$G_h(\mathbf{x}, z) = (F(\mathbf{x}),\, zh(\mathbf{x})).$$

Then, for $\mathbf{a} \in \mathbb{C}^n$, one has $\boldsymbol{g} \subseteq I_{\mathcal{L}(\mathbf{a},h,F)}$ if and only if $(\mathbf{a}, 1) \in S_{(G_h,X)}$.

Proof For each $i \in \{1, \ldots, m\}$, let $X_i = \boldsymbol{V}(zg_i) \subset \mathbb{C}^{n+1}$ be an algebraic variety. By [1, Proposition 2.7], we have

$$g_i \in I_{\mathcal{L}(\mathbf{a},h,F)} \iff (\mathbf{a}, 1) \in S_{(G_h,X_i)}.$$

Hence, $g_1, \ldots, g_m \in I_{\mathcal{L}(\mathbf{a},h,F)}$ if and only if

$$(\mathbf{a}, 1) \in S_{(G_h,X_1)} \cap \cdots \cap S_{(G_h,X_m)}.$$

By Lemma 1, it follows that

$$S_{(G_h,X)} = S_{(G_h,X_1)} \cap \cdots \cap S_{(G_h,X_m)}.$$

Hence, $g_1, \ldots, g_m \in I_{\mathcal{L}(\mathbf{a},h,F)}$ if and only if $(\mathbf{a}, 1) \in S_{(G_h,X)}$. □

The next proposition provides a necessary and sufficient condition for loops with a given structure to satisfy specified polynomial invariants.

Proposition 4 *Let $\mathbf{f}_i = (f_{i,1}, \ldots, f_{i,l_i})$ be a sequence of polynomials in $\mathbb{C}[\mathbf{x}]$ for every $i \leq n$, and define $\mathbf{f} = (\mathbf{f}_1, \ldots, \mathbf{f}_n)$. Let $z, y_{i,1}, \ldots, y_{i,l_i}$ be new indeterminates for every $i \leq n$, and let $h \in \mathbb{C}[\mathbf{x}]$. Let H_h be the map*

$$\begin{aligned} \mathbb{C}^{n+\sum_{j=1}^{n} l_j+1} &\longrightarrow \mathbb{C}^{n+\sum_{j=1}^{n} l_j+1} \\ (\mathbf{x}, \mathbf{y}, z) &\longmapsto \left(\sum_{i=1}^{l_1} y_{1,i} f_{1,i}(\mathbf{x}), \ldots, \sum_{i=1}^{l_n} y_{n,i} f_{n,i}(\mathbf{x}),\ \mathbf{y},\ zh(\mathbf{x}) \right). \end{aligned}$$

Let $\boldsymbol{g} = (g_1, \ldots, g_m)$ be a sequence of polynomials in $\mathbb{C}[\mathbf{x}]$, viewed as elements of $\mathbb{C}[\mathbf{x}, \mathbf{y}, z]$, and define

$$X = \boldsymbol{V}(zg_1, \ldots, zg_m) \subset \mathbb{C}^{n+\sum_{j=1}^{n} l_j+1}.$$

Then, for any $\mathbf{a} \in \mathbb{C}^n$,

$$\mathscr{C}(\mathbf{a}, h, \mathbf{f}; \boldsymbol{g}) = \left\{ \mathbf{b} \in \mathbb{C}^{\sum_{j=1}^{n} l_j} \;\middle|\; (\mathbf{a}, \mathbf{b}, 1) \in S_{(H_h,X)} \right\}.$$

Proof Write $H_h(\mathbf{x}, \mathbf{y}, z) = \big(H(\mathbf{x}, \mathbf{y}),\ zh(\mathbf{x})\big)$, where

$$H(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{l_1} y_{1,i} f_{1,i}(\mathbf{x}),\ \ldots,\ \sum_{i=1}^{l_n} y_{n,i} f_{n,i}(\mathbf{x}),\ \mathbf{y} \right).$$

By Proposition 3, $g_1, \ldots, g_m$ are polynomial invariants of $\mathcal{L}((\mathbf{a}, \mathbf{b}), h, H)$ if and only if $(\mathbf{a}, \mathbf{b}, 1)$ is contained in $S_{(H_h,X)}$. Since the value of $\mathbf{y}$ remains fixed at $\mathbf{b}$ under iteration, for all $N \in \mathbb{N}$, we have

$$H^N(\mathbf{x}, \mathbf{b}) = \big(F^N_{\mathbf{f},\mathbf{b}}(\mathbf{x}),\ \mathbf{b}\big).$$

Hence, $\boldsymbol{g} \subset I_{\mathcal{L}(\mathbf{a},h,F_{\mathbf{f},\mathbf{b}})}$ if and only if $\boldsymbol{g} \subset I_{\mathcal{L}((\mathbf{a},\mathbf{b}),h,H)}$, which in turn holds if and only if $(\mathbf{a}, \mathbf{b}, 1) \in S_{(H_h,X)}$. □

Since the invariant set is an algebraic variety, Proposition 4 implies that $\mathscr{C}(\mathbf{a}, h, \mathbf{f}; \boldsymbol{g})$ is likewise an algebraic variety. Its defining equations are obtained by substituting $\mathbf{x} = \mathbf{a}$ and $z = 1$ into the system that defines the invariant set.

The following algorithm computes a generating set of polynomials for $\mathscr{C}(\mathbf{a}, h, \mathbf{f}; \boldsymbol{g})$.

Algorithm 2 GenerateLoops

Input: $h, g_1, \ldots, g_m, f_{1,1}, \ldots, f_{1,l_1}, \ldots, f_{n,1}, \ldots, f_{n,l_n}$ in $\mathbb{Q}[\mathbf{x}]$ and $\mathbf{a} \in \mathbb{Q}^n$.

Output: A set of polynomials $\{P_1, \ldots, P_s\}$ such that $\mathscr{C}(\mathbf{a}, h, \mathbf{f}; g)$ is $\boldsymbol{V}(P_1, \ldots, P_s)$

1: $H_h \leftarrow \left(\sum_{i=1}^{l_1} y_{1,i} f_{1,i}(\mathbf{x}), \ldots, \sum_{i=1}^{l_n} y_{n,i} f_{n,i}(\mathbf{x}), \mathbf{y}, zh(\mathbf{x}) \right)$

2: $\{Q_1, \ldots, Q_s\} \leftarrow \mathsf{InvariantSet}((zg_1, \ldots, zg_m), H_h)$;

3: $\{P_1(\mathbf{y}), \ldots, P_s(\mathbf{y})\} \leftarrow \{Q_1(\mathbf{a}, \mathbf{y}, 1), \ldots, Q_s(\mathbf{a}, \mathbf{y}, 1)\}$

4: **return** $\{P_1, \ldots, P_s\}$;

Theorem 1 *Let $\mathbf{a} \in \mathbb{Q}^n$, $h \in \mathbb{C}[\mathbf{x}]$, $\boldsymbol{g} = (g_1, \ldots, g_m)$, and $\mathbf{f}_1 = (f_{1,1}, \ldots, f_{1,l_1}), \ldots, \mathbf{f}_n = (f_{n,1}, \ldots, f_{n,l_n})$ be sequences of polynomials in $\mathbb{Q}[\mathbf{x}]$. Set $\mathbf{f} = (\mathbf{f}_1, \ldots, \mathbf{f}_n)$. Given input $(\mathbf{a}, h, \mathbf{f}, \boldsymbol{g})$, Algorithm 2 outputs a set of polynomials whose vanishing locus is $\mathscr{C}(\mathbf{a}, h, \mathbf{f}; \boldsymbol{g})$.*

Proof We prove the correctness of Algorithm 2. Let $X = \boldsymbol{V}(zg_1, \ldots, zg_m) \subset \mathbb{C}^{n+l_1+\cdots+l_n+1}$. At Step 2, by [25, Theorem 2.4], InvariantSet outputs

$$\{Q_1(\mathbf{x}, \mathbf{y}, z), \ldots, Q_s(\mathbf{x}, \mathbf{y}, z)\},$$

such that $S_{(H_h,X)}$ is the common zero-set of $\{Q_1, \ldots, Q_s\}$. Hence, by Proposition 4, $g_1, \ldots, g_m$ are polynomial invariants of $\mathcal{L}(\mathbf{a}, h, F_{\mathbf{f},\mathbf{b}})$ if and only if

$$Q_1(\mathbf{a}, \mathbf{b}, 1) = \cdots = Q_s(\mathbf{a}, \mathbf{b}, 1) = 0.$$

Consequently, $g_1, \ldots, g_m \in I_{\mathcal{L}(\mathbf{a},h,F_{\mathbf{f},\mathbf{b}})}$ if and only if

$$P_1(\mathbf{b}) = \cdots = P_s(\mathbf{b}) = 0,$$

which establishes the correctness of Algorithm 2. □

In the examples below, to distinguish program variables from the coefficients of the update maps, we denote the coefficients by λ instead of y.

Example 2 Consider the polynomial map

$$F(x_1, x_2, x_3) = (\lambda_1 x_1^3 + \lambda_2 x_2^2,\ \lambda_3 x_1 + \lambda_4 x_2^2,\ \lambda_5 x_1),$$

and the polynomial invariants $g_1 = x_2^2 - x_1$ and $g_2 = x_3^3 + 2x_2^2 - x_1$. Our goal is to determine all loops of

this form with precondition $(x_1, x_2, x_3) = (1, 1, -1)$ and postcondition $(x_2^2 - x_1 = 0,\ x_3^3 + 2x_2^2 - x_1 = 0)$:

$(x_1, x_2, x_3) \leftarrow (1, 1, -1)$
while true **do**
$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \xleftarrow{F} \begin{pmatrix} \lambda_1 x_1^3 + \lambda_2 x_2^2 \\ \lambda_3 x_1 + \lambda_4 x_2^2 \\ \lambda_5 x_1 \end{pmatrix}$
end while

Let $F = (F_1, F_2, F_3)$ and $\boldsymbol{g} = \{g_1, g_2\}$, and define

$$\mathbf{f} = \{\{x_1^3, x_2^2\},\ \{x_1, x_2^2\},\ \{x_1\}\}.$$

From the loop structure we have

$$F_1 \in \operatorname{span}\{x_1^3, x_2^2\},\ F_2 \in \operatorname{span}\{x_1, x_2^2\},\ F_3 \in \operatorname{span}\{x_1\}.$$

Thus, the input to Algorithm 2 is $(\{1\}, \boldsymbol{g}, \mathbf{f}, \{1, 1, -1\})$. Let $X = \boldsymbol{V}(zg_1, zg_2) \subset \mathbb{C}^9$ and define the map

$$H_h(\mathbf{x}, \mathbf{y}, z) = (y_1 x_1^3 + y_2 x_2^2,\ y_3 x_1 + y_4 x_2^2,\ y_5 x_1,\ \mathbf{y},\ z).$$

At Step 2, given input H_h and X, `InvariantSet` outputs polynomials

$$\{Q_1(\mathbf{x}, \mathbf{y}, z), \ldots, Q_6(\mathbf{x}, \mathbf{y}, z)\}.$$

whose vanishing locus is the invariant set $S_{(H_h, X)}$. Substituting the initial values of the loop into these polynomials yields four nonzero polynomials $P_1(\mathbf{y}), \ldots, P_4(\mathbf{y})$, whose vanishing locus is $\mathscr{C}((1, 1, -1), 1, \mathbf{f}; \boldsymbol{g})$:

$\boldsymbol{P_1} = (y_3 + y_4)^2 - y_1 - y_2, \quad \boldsymbol{P_2} = y_5^3 + 2(y_3 + y_4)^2 - y_1 - y_2,$
$\boldsymbol{P_3} = 2y_3^4 y_4^2 + 8y_3^3 y_4^3 + 12y_3^2 y_4^4 + 8y_3 y_4^5 + 2y_4^6 + y_1^3 y_5^3 + 3y_1^2 y_2 y_5^3 + 3y_1 y_2^2 y_5^3 + y_2^3 y_5^3 + 4y_1 y_3^3 y_4 + 4y_2 y_3^3 y_4 + 8y_1 y_3^2 y_4^2 + 8y_2 y_3^2 y_4^2 + 4y_1 y_3 y_4^3 + 4y_2 y_3 y_4^3 - y_1^4 - 3y_1^3 y_2 - 3y_1^2 y_2^2 - y_1 y_2^3 + 2y_1^2 y_3^2 + 4y_1 y_2 y_3^2 + 2y_2^2 y_3^2 - y_2 y_3^2 - 2y_2 y_3 y_4 - y_2 y_4^2,$

$\boldsymbol{P_4} = y_3^4 y_4^2 + 4y_3^3 y_4^3 + 6y_3^2 y_4^4 + 4y_3 y_4^5 + y_4^6 + 2y_1 y_3^3 y_4 + 2y_2 y_3^3 y_4 + 4y_1 y_3^2 y_4^2 + 4y_2 y_3^2 y_4^2 + 2y_1 y_3 y_4^3 + 2y_2 y_3 y_4^3 - y_1^4 - 3y_1^3 y_2 - 3y_1^2 y_2^2 - y_1 y_2^3 + y_1^2 y_3^2 + 2y_1 y_2 y_3^2 + y_2^2 y_3^2 - y_2 y_3^2 - 2y_2 y_3 y_4 - y_2 y_4^2.$

Remark 1 This algorithm can be straightforwardly extended to select an initial value in addition to the update map of the loop. Indeed, when no initial values are specified, Step 2 of Algorithm 2 produces a sequence of polynomials

$$(Q_1, \ldots, Q_s) \subset \mathbb{Q}[\mathbf{x}, \mathbf{y}, z]$$

such that $\mathbf{b} \in \mathscr{C}(\mathbf{a}, h, \mathbf{f}; \boldsymbol{g})$ if and only if

$$Q_1(\mathbf{a}, \mathbf{b}, 1) = \cdots = Q_s(\mathbf{a}, \mathbf{b}, 1) = 0.$$

Consequently, Algorithm 2 applies even in the absence of initial values.

3.2 Generalization to branching loops

In this section, we extend the results on single-path loops to *branching loops* $\mathcal{L}(\mathbf{a}, h, (F_1, \ldots, F_k))$, which are defined by a nondeterministic conditional statement with k branches:

$(x_1, \ldots, x_n) := (a_1, \ldots, a_n)$
while $h \neq 0$ **do**
  if $*$ **then**
    $(x_1, \ldots, x_n) \leftarrow F_1(x_1, \ldots, x_n)$
  ⋮
  else if $*$ **then**
    $(x_1, \ldots, x_n) \leftarrow F_i(x_1, \ldots, x_n)$
  ⋮
  else
    $(x_1, \ldots, x_n) \leftarrow F_k(x_1, \ldots, x_n)$
  end if
end while

Here $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{C}^n$, $h \in \mathbb{C}[\mathbf{x}]$, and

$$F_1, \ldots, F_k : \mathbb{C}^n \to \mathbb{C}^n$$

are polynomial maps. When no guard h is specified, we simply write $\mathcal{L}(\mathbf{a}, 1, (F_1, \ldots, F_k))$.

The following lemma, an immediate consequence of Definition 7, shows that invariant sets remain closed under intersections in the branching case.

Lemma 2 *Let $F_1, \ldots, F_k : \mathbb{C}^n \to \mathbb{C}^n$ be polynomial maps and let $X_1, \ldots, X_m \subseteq \mathbb{C}^n$ be algebraic varieties. Denote $\mathbf{F} = (F_1, \ldots, F_k)$. Then*

$$S_{(\mathbf{F},\, X_1 \cap \cdots \cap X_m)} \;=\; S_{(\mathbf{F}, X_1)} \cap \cdots \cap S_{(\mathbf{F}, X_m)}.$$

The invariant ideal $I_{\mathcal{L}(\mathbf{a}, h, (F_1, \ldots, F_k))}$ of a branching loop is defined in [1, Definition 3.11] as the set of all polynomial invariants of the loop. The following proposition extends Proposition 3; the proof is the same, except that [1, Proposition 3.15] is invoked in place of [1, Proposition 2.7].

Proposition 5 *Let $F_1, \ldots, F_k : \mathbb{C}^n \to \mathbb{C}^n$ be polynomial maps and let $h, g_1, \ldots, g_m \in \mathbb{C}[\mathbf{x}]$. Let z be a new indeterminate, and define the variety*

$$X = \boldsymbol{V}(zg_1, \ldots, zg_m) \subset \mathbb{C}^{n+1}.$$

For each $i \leq k$, define the polynomial map

$$G_{i,h}(\mathbf{x}, z) = \big(F_i(\mathbf{x}),\ zh(\mathbf{x})\big).$$

Then

$$\boldsymbol{g} \subseteq I_{\mathcal{L}(\mathbf{a}, h, (F_1, \ldots, F_k))} \iff (\mathbf{a}, 1) \in S_{((G_{1,h}, \ldots, G_{k,h}), X)}.$$

We now define the set of branching loops, specified by multiple polynomial maps, an initial value, and a guard condition, that satisfy a given collection of polynomial invariants. The following definition extends Definition 4.

Definition 8 For each $i \leq k$ and $j \leq n$, let $\mathbf{f}_{i,j} = (f_{i,j,1}, \ldots, f_{i,j,n_{i,j}})$ be a sequence of polynomials in $\mathbb{C}[\mathbf{x}]$ and let $h \in \mathbb{C}[\mathbf{x}]$. Let $\boldsymbol{g} = (g_1, \ldots, g_m)$ be a sequence of polynomials in $\mathbb{C}[\mathbf{x}]$, and set $\mathbf{f}_i = (\mathbf{f}_{i,1}, \ldots, \mathbf{f}_{i,n})$ for each $i \leq k$.

The set of branching loops structured by $(\mathbf{f}_1, \ldots, \mathbf{f}_k)$ and satisfying the invariants $\boldsymbol{g}$ is the coefficient set $\mathscr{C}(h, (\mathbf{f}_1, \ldots, \mathbf{f}_k); \boldsymbol{g})$ defined by

$$\left\{(\mathbf{a}, \mathbf{b}_1, \ldots, \mathbf{b}_k) \in \mathbb{C}^M \;\middle|\; \boldsymbol{g} \subseteq I_{\mathcal{L}(\mathbf{a}, h, (F_{\mathbf{f}_1, \mathbf{b}_1}, \ldots, F_{\mathbf{f}_k, \mathbf{b}_k}))}\right\},$$

where $M = n + \sum_{i=1}^{k} \sum_{j=1}^{n} n_{i,j}$.

Thus, $\mathscr{C}(h, (\mathbf{f}_1, \ldots, \mathbf{f}_k); \boldsymbol{g})$ consists of all choices of initial values and coefficients that yield branching loops satisfying the polynomial invariants $\boldsymbol{g}$.

Hence, to synthesize a branching loop structured by $(\mathbf{f}_1, \ldots, \mathbf{f}_k)$ that satisfies $\boldsymbol{g}$, it suffices to identify a vector in $\mathscr{C}(h, (\mathbf{f}_1, \ldots, \mathbf{f}_k); \boldsymbol{g})$.

We now show that $\mathscr{C}(h, (\mathbf{f}_1, \ldots, \mathbf{f}_k); \boldsymbol{g})$ is an invariant set, analogous to Proposition 4 for the single-map case. The proof is similar and therefore omitted.

Proposition 6 *Using the notation of Definition 8, let z and $y_{i,j,l}$ be new indeterminates for every $i \leq k$, $j \leq n$, and $l \leq n_{i,j}$. For each $i \leq k$, define the polynomial map $H_{i,h} : \mathbb{C}^{M+1} \to \mathbb{C}^{M+1}$ in $\mathbb{C}[\mathbf{x}, \mathbf{y}, z]$ such that $H_{i,h}(\mathbf{x}, \mathbf{y}, z)$ is equal to*

$$\left(\sum_{l=1}^{n_{i,1}} y_{i,1,l} f_{i,1,l}(\mathbf{x}),\ \ldots,\ \sum_{l=1}^{n_{i,n}} y_{i,n,l} f_{i,n,l}(\mathbf{x}), \mathbf{y},\ z h(\mathbf{x})\right).$$

Define $X = \boldsymbol{V}(zg_1, \ldots, zg_m) \subset \mathbb{C}^{M+1}$, and set $\mathbf{b} = (\mathbf{b}_1, \ldots, \mathbf{b}_k)$. Then $\mathscr{C}(h, (\mathbf{f}_1, \ldots, \mathbf{f}_k); \boldsymbol{g})$ is the set

$$\left\{(\mathbf{a}, \mathbf{b}) \in \mathbb{C}^M \;\middle|\; (\mathbf{a}, \mathbf{b}, 1) \in S_{((H_{1,h}, \ldots, H_{k,h}), X)}\right\}.$$

Therefore, by computing a defining set of polynomials for the invariant set $S_{((H_{1,h}, \ldots, H_{k,h}), X)}$ and then substituting $z = 1$, we obtain defining equations for $\mathscr{C}(h, (\mathbf{f}_1, \ldots, \mathbf{f}_k); \boldsymbol{g})$. An algorithm for computing these equations in the branching (multi-map) setting is provided by Proposition 2. The following example illustrates the procedure step by step, yielding a defining set of polynomials for $\mathscr{C}(h, (\mathbf{f}_1, \ldots, \mathbf{f}_k); \boldsymbol{g})$.

Example 3 Consider branching loops with a nondeterministic conditional statement involving two branches, given by

$$F_1(x_1, x_2) = (\lambda_1 x_1 + \lambda_2,\ \lambda_3 x_2),$$

$$F_2(x_1, x_2) = (\lambda_4 x_2 + \lambda_5,\ \lambda_6 x_1),$$

that satisfy the polynomial invariant $g = 2x_1 - x_2^2$. Hence, we aim to compute all branching loops of the following form that preserve g:

```
(x1, x2) := (a1, a2)
while 1 ≠ 0 do
    if ∗ then
        (x1, x2) ←F1 (λ1x1+λ2, λ3x2)
    else
        (x1, x2) ←F2 (λ4x2+λ5, λ6x1)
    end if
end while
```

Let $\mathbf{f}_1 = \{\{x_1, 1\}, \{x_2\}\}$ and $\mathbf{f}_2 = \{\{x_2, 1\}, \{x_1\}\}$. Then $\mathcal{L}((a_1, a_2), 1, (F_1, F_2))$ satisfies the polynomial invariant g if and only if

$$(a_1, a_2, \lambda_1, \ldots, \lambda_6) \in \mathscr{C}(1, (\mathbf{f}_1, \mathbf{f}_2), g).$$

Let $X = \boldsymbol{V}(zg) \subset \mathbb{C}^9$, and define the polynomial maps

$$H_{1,h}(\mathbf{x}, \mathbf{y}, z) = (y_1 x_1 + y_2,\ y_3 x_2,\ \mathbf{y},\ z),$$

$$H_{2,h}(\mathbf{x}, \mathbf{y}, z) = (y_4 x_2 + y_5,\ y_6 x_1,\ \mathbf{y},\ z).$$

By Proposition 6, $(a_1, a_2, \lambda_1, \ldots, \lambda_6) \in \mathscr{C}(1, (\mathbf{f}_1, \mathbf{f}_2), g)$ if and only if $(a_1, a_2, \lambda_1, \ldots, \lambda_6, 1) \in S_{(H_{1,h}, H_{2,h}), X}$. On input $H_{1,h}, H_{2,h}$ and X, InvariantSetBranch outputs 31 polynomials

$$\{Q_1(\mathbf{x}, \mathbf{y}, z), \ldots, Q_{31}(\mathbf{x}, \mathbf{y}, z)\},$$

whose vanishing locus is $S_{(H_{1,h}, H_{2,h}), X}$. Therefore, the vanishing locus of the equations

$$Q_1(\mathbf{x}, \mathbf{y}, 1) = 0, \ldots, Q_{31}(\mathbf{x}, \mathbf{y}, 1) = 0,$$

is $\mathscr{C}(1, (\mathbf{f}_1, \mathbf{f}_2), g)$. Below we present six sample polynomials among these:

$$2x_1 - x_2^2;\quad 2y_1 x_1 + 2y_2 - y_3^2 x_2^2;\quad 2y_4 x_2 + 2y_5 - y_6^2 x_1^2;$$
$$2y_1^2 x_1 + 2y_1 y_2 + 2y_2 - y_3^4 x_2^2;\ 2y_1 y_4 x_2 + 2y_1 y_5 + 2y_2 - y_3^2 y_6^2 x_1^2$$
$$- y_1^2 y_6^2 x_1^2 - 2y_1 y_2 y_6^2 x_1 - y_2^2 y_6^2 + 2y_3 y_4 x_2 + 2y_5.$$

Remark 2 Similarly to Remark 1, we can extend this approach (and in particular the previous algorithm) to also synthesize guard inequations. Indeed, using the notation of Definition 8, let z, $y_{i,j,l}$ and $w_1, \ldots, w_r$ be new indeterminates for every $i \leq k$, $j \leq n$, and $l \leq n_{i,j}$. Let $h = (h_1, \ldots, h_r)$ be a sequence of polynomials in $\mathbb{Q}[\mathbf{x}]$ given as input. We look for guard polynomial is of the form

$$h(\mathbf{x}, \mathbf{w}) = w_1 h_1(\mathbf{x}) + \cdots + w_r h_r(\mathbf{x})$$

where $\mathbf{w} = (w_1, \ldots, w_r)$. For each $i \leq k$, define polynomial maps $H_{i,h}(\mathbf{x}, \mathbf{y}, z, \mathbf{w})$

$$\Big(\sum_{l=1}^{n_{i,1}} y_{i,1,l} f_{i,1,l}(\mathbf{x}), \ldots, \sum_{l=1}^{n_{i,n}} y_{i,n,l} f_{i,n,l}(\mathbf{x}), \mathbf{y}, zh(\mathbf{x},\mathbf{w}), \mathbf{w}\Big).$$

Define $X = \boldsymbol{V}(zg_1, \ldots, zg_m)$. By computing defining equations for the invariant set

$$S_{((H_{1,h},\ldots,H_{k,h}),X)}$$

and then substituting $z = 1$, we obtain equations

$$P_1(\mathbf{x},\mathbf{y},\mathbf{w}) = 0, \ldots, P_s(\mathbf{x},\mathbf{y},\mathbf{w}) = 0$$

such that $(\mathbf{a},\mathbf{b})$ is contained in $\mathscr{C}(h(\mathbf{x},\mathbf{c}),(\mathbf{f}_1,\ldots,\mathbf{f}_k);\boldsymbol{g})$ if and only if $P_1(\mathbf{a},\mathbf{b},\mathbf{c}) = \cdots = P_s(\mathbf{a},\mathbf{b},\mathbf{c}) = 0$.

3.3 Universally inductive invariants

Algorithm 2 relies on Gröbner basis computation, whose worst case complexity can be doubly exponential in the number of variables [21]. Consequently, when the template of the update map contains many generators, Algorithm 2 may fail to terminate within a reasonable time. Moreover, the degrees of the polynomial equations computed by Algorithm 2 can grow exponentially.

In this subsection, we address these issues by focusing on a special class of invariants. For this class, we rely only on linear algebra to generate polynomial equations characterizing loops that satisfy the given invariants, and the degrees of these equations are bounded above by the degrees of the invariants themselves.

3.3.1 General case

In this subsection, we consider polynomial invariants of the form $g(\mathbf{x}) - g(\mathbf{a})$, which hold for every initial value $\mathbf{a} \in \mathbb{C}^n$. Equivalently, $g(\mathbf{x}) - g(\mathbf{a})$ is an invariant of $\mathcal{L}(\mathbf{a}, h, (F_1,\ldots,F_k))$ for all $\mathbf{a} \in \mathbb{C}^n$.

Definition 9 Let $F_1,\ldots,F_k : \mathbb{C}^n \to \mathbb{C}^n$ be polynomial maps, and let $h \in \mathbb{C}[\mathbf{x}]$. A polynomial $g(\mathbf{x})$ is called a *universally inductive invariant* of $\mathcal{L}(h,(F_1,\ldots,F_k))$ if $g(\mathbf{x}) - g(\mathbf{a})$ is an invariant of $\mathcal{L}(\mathbf{a},h,(F_1,\ldots,F_k))$ for every $\mathbf{a} \in \mathbb{C}^n$. The set of all universally inductive invariants of $\mathcal{L}(h,(F_1,\ldots,F_k))$ is denoted by $UI_{\mathcal{L}(h,(F_1,\ldots,F_k))}$.

We now define the coefficient set that characterizes all branching loops structured by $(\mathbf{f}_1,\ldots,\mathbf{f}_k)$ which satisfy a prescribed collection of universally inductive invariants.

Definition 10 Using the notation of Definition 8, the set of all branching loops structured by $(\mathbf{f}_1,\ldots,\mathbf{f}_k)$ that satisfy the universally inductive invariants $\boldsymbol{g}$ is defined by the coefficient set $U\mathscr{C}(h,(\mathbf{f}_1,\ldots,\mathbf{f}_k);\boldsymbol{g})$, which is

$$\Big\{(\mathbf{b}_1,\ldots,\mathbf{b}_k) \in \mathbb{C}^M \;\Big|\; \boldsymbol{g} \subseteq UI_{\mathcal{L}(h,(F_{\mathbf{f}_1,\mathbf{b}_1},\ldots,F_{\mathbf{f}_k,\mathbf{b}_k}))}\Big\},$$

where $M = \sum_{i=1}^{k}\sum_{j=1}^{n} n_{i,j}$.

The following proposition, stated in [1, Corollary 4.4], provides a necessary and sufficient condition for a polynomial to be a universally inductive invariant. For completeness, we include a brief proof.

Proposition 7 *Let $F_1,\ldots,F_k : \mathbb{C}^n \to \mathbb{C}^n$ be polynomial maps, and let $h(\mathbf{x}) \in \mathbb{C}[\mathbf{x}]\backslash\{0\}$. Then a polynomial $g(\mathbf{x})$ is a universally inductive invariant if and only if*

$$g(F_i(\mathbf{x})) = g(\mathbf{x}) \quad \textit{for every } i \le k.$$

Proof ($\Rightarrow$) Suppose g is a universally inductive invariant. Let $\mathbf{a} \notin \boldsymbol{V}(h)$. By the definition of invariant polynomials,

$$g(F_i(\mathbf{a})) - g(\mathbf{a}) = 0$$

for every $i \le k$. Hence $g(F_i(\mathbf{a})) = g(\mathbf{a})$ for all $\mathbf{a} \notin \boldsymbol{V}(h)$. Now let $\mathbf{a} \in \boldsymbol{V}(h)$. Since $h(\mathbf{x}) \neq 0$, there exists a sequence $\{\mathbf{a}_n\}$ with $\mathbf{a}_n \notin \boldsymbol{V}(h)$ such that $\lim_{n\to\infty} \mathbf{a}_n = \mathbf{a}$. Because $g(F_i(\mathbf{a}_n)) = g(\mathbf{a}_n)$ and both g and F_i are continuous, we conclude that $g(F_i(\mathbf{a})) = g(\mathbf{a})$ for all $\mathbf{a} \in \boldsymbol{V}(h)$. Thus, $g(F_i(\mathbf{a})) = g(\mathbf{a})$ for every $\mathbf{a} \in \mathbb{C}^n$, and so $g(F_i(\mathbf{x})) = g(\mathbf{x})$ as polynomials, for all $i \le k$.

($\Leftarrow$) Conversely, assume $g(F_i(\mathbf{x})) = g(\mathbf{x})$ for every $i \le k$. Then for any $i_1,\ldots,i_m \in [k]$ and all $\mathbf{a} \in \mathbb{C}^n$, we obtain

$$g(F_{i_m}(\cdots F_{i_1}(\mathbf{a}))) - g(\mathbf{a}) = 0.$$

Hence $g(\mathbf{x}) - g(\mathbf{a}) \in I_{\mathcal{L}(\mathbf{a},h,(F_1,\ldots,F_k))}$, and g is a universally inductive invariant. □

An invariant is called *inductive* if, once it holds after a single iteration, it continues to hold for all subsequent iterations. By Proposition 7, we immediately obtain:

Corollary 1 *Every universally inductive invariant is inductive.*

Algorithm 3 takes as input a sequence $\boldsymbol{g}$ of polynomials in $\mathbb{Q}[\mathbf{x}]$ together with the structure $(\mathbf{f}_1,\ldots,\mathbf{f}_k)$ of polynomial maps of a loop, and outputs polynomials defining $U\mathscr{C}(h,(\mathbf{f}_1,\ldots,\mathbf{f}_k);\boldsymbol{g})$. The auxiliary procedure `coef` takes as input a sequence of polynomials in $\mathbb{Q}[\mathbf{x},\mathbf{y}]$ and returns the coefficients of the monomials in the variables $\mathbf{x}$. The correctness of Algorithm 3 follows immediately from Proposition 7.

Algorithm 3 ComputeLoopsUniversal

Input: A sequence $\boldsymbol{g} = (g_1, \dots, g_m)$ of polynomials, and polynomials $f_{i,j,l} \in \mathbb{Q}[\mathbf{x}]$ for all $i \le k$, $j \le n$, and $l \le n_{i,j}$.

Output: A set of polynomials $C_1, \dots, C_t$ such that $U\mathscr{C}(h, (\mathbf{f}_1, \dots, \mathbf{f}_k); \boldsymbol{g})$ is $\boldsymbol{V}(C_1, \dots, C_t)$

1: **for** $i \in [k]$ **do**
2: $F_i \leftarrow \left(\sum_{l=1}^{n_{i,1}} y_{i,1,l} f_{i,1,l}(\mathbf{x}),\ \dots,\ \sum_{l=1}^{n_{i,n}} y_{i,n,l} f_{i,n,l}(\mathbf{x}) \right)$
3: **end for**
4: $\{C_1, \dots, C_t\} \leftarrow \{\mathsf{coef}(g_i - g_i \circ F_j) \mid i \in [m],\ j \in [k]\}$
5: **return** $\{C_1, \dots, C_t\}$

As a concrete application, we demonstrate how our method can be used to synthesize loops that generate Markov triples.

Example 4 (Markov triples [41]) In this example, we synthesize a loop of the following form that generates a solution to the Markov equation

$$x_1^2 + x_2^2 + x_3^2 - 3x_1x_2x_3 = 0.$$

$(x_1, x_2, x_3) := (1, 1, 2)$
while $1 \neq 0$ **do**
  if $*$ **then**
    $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \xleftarrow{F_1} \begin{pmatrix} \lambda_1 x_1 + \lambda_2 x_2 \\ \lambda_3 x_1 x_2 + \lambda_4 x_3 + \lambda_5 x_1^2 \\ \lambda_6 x_2 + \lambda_7 x_3 \end{pmatrix}$
  else
    $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \xleftarrow{F_2} \begin{pmatrix} \lambda_8 x_1 + \lambda_9 x_2 \\ \lambda_{10} x_2 x_3 + \lambda_{11} x_1 + \lambda_{12} x_2^2 \\ \lambda_{13} x_2 + \lambda_{14} x_3 \end{pmatrix}$
  end if
end while

Let $g = x_1^2 + x_2^2 + x_3^2 - 3x_1x_2x_3$. We therefore consider branching loops $\mathcal{L}((1,1,2), 1, (F_1, F_2))$ that satisfy the universal polynomial invariant g with structure

$$F_1(\mathbf{x}) = (\lambda_1 x_1 + \lambda_2 x_2, 3x_1x_2 + \lambda_4 x_3 + \lambda_5 x_1^2, \lambda_6 x_2 + \lambda_7 x_3),$$

$$F_2(\mathbf{x}) = (\lambda_8 x_1 + \lambda_9 x_2, \lambda_{10} x_2 x_3 + \lambda_{11} x_1 + \lambda_{12} x_2^2, \lambda_{13} x_2 + \lambda_{14} x_3)$$

By Proposition 7, we have

$$g(\mathbf{x}) = g(F_1(\mathbf{x})), \quad g(\mathbf{x}) = g(F_2(\mathbf{x})).$$

Hence, the coefficients of the monomials in the variables $\mathbf{x}$ appearing in the polynomial pairs $(g(\mathbf{x}), g(F_1(\mathbf{x})))$ and $(g(\mathbf{x}), g(F_2(\mathbf{x})))$ must be equal. Hence, we obtain the following system of 32 polynomial equations:

$$\begin{cases} 3\lambda_1\lambda_3\lambda_6 + 3\lambda_2\lambda_5\lambda_6 - \lambda_3^2 = 0, \\ 3\lambda_8\lambda_{10}\lambda_{13} + 3\lambda_8\lambda_{12}\lambda_{14} = 0 \\ \lambda_5^2 = 0, \\ \dots \\ \lambda_{14}^2 - 1 = 0, \\ \lambda_1^2 - 1 = 0. \end{cases}$$

Using the Z3 solver, we obtain the following solution to the system above:

$$\lambda_2 = \lambda_5 = \lambda_7 = \lambda_8 = \lambda_{12} = \lambda_{13} = 0, \quad \lambda_3 = \lambda_{10} = 3$$

$$\lambda_1 = \lambda_4 = \lambda_6 = \lambda_9 = \lambda_{11} = \lambda_{14} = -1$$

3.3.2 Affine case

We now restrict attention to affine universally inductive invariants. We show that $U\mathscr{C}(h, (\mathbf{f}_1, \dots, \mathbf{f}_k); \boldsymbol{g})$ is an affine space whenever $\boldsymbol{g}$ consists of affine polynomials. Furthermore, we present an algorithm for computing an affine basis of $U\mathscr{C}(h, (\mathbf{f}_1, \dots, \mathbf{f}_k); \boldsymbol{g})$.

Proposition 8 *Let $\boldsymbol{g} = (g_1, \dots, g_m)$ be affine polynomials. Then the coefficient set $U\mathscr{C}(h, (\mathbf{f}_1, \dots, \mathbf{f}_k); \boldsymbol{g})$ is an affine space.*

Proof Let $(\mathbf{b}_1, \dots, \mathbf{b}_k) \in U\mathscr{C}(h, (\mathbf{f}_1, \dots, \mathbf{f}_k); \boldsymbol{g})$. Then, by Proposition 7, the polynomials $g_1, \dots, g_m$ are universally inductive invariants of $\mathcal{L}(h, (F_{\mathbf{f}_1, \mathbf{b}_1}, \dots, F_{\mathbf{f}_k, \mathbf{b}_k}))$ if and only if

$$g_i(F_j(\mathbf{x})) = g_i(\mathbf{x}) \quad \text{for every } i \le m,\ j \le k.$$

This condition requires that the coefficients of the monomials in $g_i(F_j(\mathbf{x}))$ coincide with those in $g_i(\mathbf{x})$. Since $g_1, \dots, g_m$ are linear, these coefficients are themselves linear polynomials in $\mathbb{C}[\mathbf{b}_1, \dots, \mathbf{b}_k]$. Consequently, $U\mathscr{C}(h, (\mathbf{f}_1, \dots, \mathbf{f}_k); \boldsymbol{g})$ is the solution set of a system of linear equations, and therefore an affine space. □

Proposition 8 shows that, in the affine case, the synthesis problem reduces to a problem of linear algebra.

Let $\boldsymbol{g}$ be a sequence of affine polynomials. The following algorithm computes a vector v and a basis of a vector space V such that $U\mathscr{C}(h, (\mathbf{f}_1, \dots, \mathbf{f}_k); \boldsymbol{g})$ is the affine space $v + V$.

We now describe the auxiliary procedures used in Algorithm 4:

– sol takes as input a sequence of affine polynomials $(l_1(\mathbf{y}), \dots, l_r(\mathbf{y}))$ in $\mathbb{Q}[\mathbf{y}]$ and returns a solution to the linear system

$$l_1(\mathbf{y}) = 0,\ \dots,\ l_r(\mathbf{y}) = 0.$$

– solveHom takes as input a sequence of affine polynomials $(l_1(\mathbf{y}), \dots, l_r(\mathbf{y}))$ in $\mathbb{Q}[\mathbf{y}]$ and outputs a basis for the solution space of the corresponding homogeneous system, namely

$$l_1(\mathbf{y}) = 0,\ \dots,\ l_r(\mathbf{y}) = 0.$$

The correctness of Algorithm 4 follows immediately from Proposition 8.

Algorithm 4 ComputeLoopsLinearUniversal

Input: A sequence $\boldsymbol{g} = (g_1, \dots, g_m)$ of affine polynomials, and polynomials $f_{i,j,l} \in \mathbb{Q}[\mathbf{x}]$ for all $i \le k$, $j \le n$, and $l \le n_{i,j}$.

Output: A vector $v \in \mathbb{Q}^M$ and a sequence of vectors B such that $U\mathscr{C}(h, (\mathbf{f}_1, \dots, \mathbf{f}_k); \boldsymbol{g}) = v + V$, where V is the vector space generated by B.

1: $C \leftarrow \mathsf{ComputeLoopsUniversal}(\boldsymbol{g}, \mathbf{f}_1, \dots, \mathbf{f}_k)$
2: $v \leftarrow \mathsf{sol}(C)$
3: $B \leftarrow \mathsf{solveHom}(C)$
4: **return** $\{v, B\}$

Example 5 Consider branching loops with a nondeterministic conditional statement involving two branches, defined by

$$F_1(x_1, x_2) = (\lambda_1 x_1^2 + \lambda_2 x_1 + \lambda_3 x_2,\ \lambda_4 x_1^2 + \lambda_5 x_1 + \lambda_6 x_2),$$

$$F_2(x_1, x_2) = (\lambda_7 x_1 + \lambda_8 x_2,\ \lambda_9 x_1 + \lambda_{10} x_2),$$

such that $g = x_1 - x_2 + 1$ is a universally inductive invariant of $\mathcal{L}(1, (F_1, F_2))$. By Proposition 7, this requires

$$g(\mathbf{x}) - g(F_1(\mathbf{x})) = 0, \qquad g(\mathbf{x}) - g(F_2(\mathbf{x})) = 0.$$

Collecting the coefficients of the monomials in $\mathbf{x}$ yields the linear system:

$$\begin{cases} \lambda_4 - \lambda_1 = 0, \\ \lambda_5 - \lambda_2 + 1 = 0, \\ \lambda_6 - \lambda_3 - 1 = 0, \\ \lambda_9 - \lambda_7 + 1 = 0, \\ \lambda_{10} - \lambda_8 - 1 = 0. \end{cases}$$

A particular solution is $v = (0, 1, -1, 0, 0, 0, 1, -1, 0, 0)$. Solving the associated homogeneous system yields a sequence B of basis vectors:

$$(1,0,0,1,0,0,0,0,0,0), \quad (0,1,0,0,1,0,0,0,0,0),$$

$$(0,0,1,0,0,1,0,0,0,0), \quad (0,0,0,0,0,0,1,0,1,0),$$

$$(0,0,0,0,0,0,0,1,0,1).$$

Therefore, g is a universally inductive invariant of the loop $\mathcal{L}(1, (F_1, F_2))$ if and only if $(\lambda_1, \dots, \lambda_{10}) \in v + \mathrm{span}(B)$.

4 Solving Polynomial Systems

In the previous section, we saw that Algorithms 2 and 3 produce systems of multivariate polynomials whose solutions correspond precisely to loops with the prescribed structure and invariants. Thus, the problem of loop synthesis reduces to solving polynomial systems.

Since our goal is to obtain loops that can be represented finitely and exactly on a computer, we focus on *rational solutions*, i.e., solutions with coefficients in $\mathbb{Q}$. Unlike the situation over $\mathbb{C}$ (where Hilbert's Nullstellensatz applies [30], see also [10, Chap. 4, §1]) or over $\mathbb{R}$ (where the Tarski–Seidenberg theorem applies [32,31], see e.g. [9]), deciding whether a polynomial equation has a rational solution is a long-standing open problem in number theory [33]. For integer solutions, this corresponds to Hilbert's Tenth Problem, which is known to be undecidable [33].

Given these difficulties, we outline three main strategies for addressing the problem, although none is fully satisfactory or complete.

4.1 Exploiting the structure of polynomial systems

Although no general algorithm is known for computing (or even deciding the existence of) rational solutions to multivariate polynomial systems by exact methods, many practical instances can still be solved by leveraging structural properties. For example, consider the polynomial system obtained at the end of Example 2. While this case may seem simple or narrowly tailored, all benchmarks reported in Section 5 can be handled in a similar way. More generally, decompositions of varieties with additional structure can sometimes be reduced to combinatorial problems; see, e.g., [36,37].

Example 6 The variety $\boldsymbol{V}(P_1, \dots, P_4)$ can be decomposed into finitely many irreducible components, which in turn split the system into simpler subsystems. This decomposition can be carried out using classical methods from computer algebra [10, Chap. 4, §6]; here we use the command `minimalPrimes` from Macaulay2 [6]. We obtain the following five irreducible components:

$$\begin{aligned} \mathbf{V}_1 &= \boldsymbol{V}(y_5,\ y_3 + y_4,\ y_1 + y_2) \\ \mathbf{V}_2 &= \boldsymbol{V}(y_5 + 1,\ y_3 + y_4 - 1,\ y_1 + y_2 - 1) \\ \mathbf{V}_3 &= \boldsymbol{V}(y_5 + 1,\ y_3 + y_4 + 1,\ y_1 + y_2 - 1) \\ \mathbf{V}_4 &= \boldsymbol{V}(y_3 + y_4 - 1,\ y_1 + y_2 - 1,\ y_5^2 - y_5 + 1) \\ \mathbf{V}_5 &= \boldsymbol{V}(y_3 + y_4 + 1,\ y_1 + y_2 - 1,\ y_5^2 - y_5 + 1) \end{aligned}$$

Thus, $\mathcal{L}((1,1,-1), 1, F)$ satisfies the polynomial invariants $\{g_1, g_2\}$ if and only if $(\lambda_1, \dots, \lambda_5)$ lies in one of the above components. The simple structure of the defining polynomials makes the problem fully solvable.

For the first three components, which involve only linear polynomials, the problem reduces to standard linear algebra, where efficient methods are available (see e.g., [34]). More specifically, for parameters $\mu_1, \mu_2 \in \mathbb{Q}$, we obtain the following loop maps:

$$\mathbf{V}_1\colon F_1(\mathbf{x}) = \big(\mu_1(x_1^3 - x_2^2),\ \mu_2(x_1 - x_2^2),\ 0\big)$$
$$\mathbf{V}_2\colon F_2(\mathbf{x}) = \big(\mu_1 x_1^3 + (1-\mu_1)x_2^2,\ \mu_2 x_1 + (1-\mu_2)x_2^2,\ -1\big)$$
$$\mathbf{V}_3\colon F_3(\mathbf{x}) = \big(\mu_1 x_1^3 - (1+\mu_1)x_2^2,\ \mu_2 x_1 + (1-\mu_2)x_2^2,\ -1\big)$$

For the last two components, no rational solution exists, since the polynomial $y_5^2 - y_5 + 1$ has no rational roots.

Another important case arises when the polynomial system computed by Algorithm 2 and Algorithm 3 (or one of its subsystems) has only finitely many solutions. Geometrically, this means that the associated variety (or one of its irreducible components) consists of finitely many points, i.e., it has *dimension zero*. In this situation, such systems can be reduced (see, e.g., [35, §3], [38]) to polynomial systems with rational coefficients of the form

$$Q_1(x_1) = 0, \qquad x_i = Q_i(x_1) \ \text{ for all } i \geq 2.$$

Finding all rational solutions then reduces to solving a univariate polynomial equation, which can be done algorithmically: either through efficient factoring [22, Theorem 15.21] to extract the linear factors in $\mathbb{Q}[x_1]$, or via modular arithmetic-based methods [39].

Thus, whenever the system has finitely many solutions, all rational ones can be effectively computed. Practical implementations exist, such as AlgebraicSolving.jl[1], which is based on the `msolve` library [2]. Moreover, many benchmarks in the next section fall into this category, which we believe is common when the degree and support of F are comparable.

4.2 Numerical methods

Numerical approaches to solving polynomial systems, such as HomotopyContinuation.jl [29], provide powerful tools for approximating solutions. These methods rely on numerical algebraic geometry, in particular on homotopy continuation. Unlike symbolic techniques based on Gröbner bases or resultants, numerical methods can efficiently handle large and complex systems. When applied to the systems generated by Algorithm 2 and Algorithm 3, they yield numerical solutions that can then be checked against nearby integers or rationals to identify valid exact solutions.

[1] `https://algebraic-solving.github.io`

Example 7 Consider the polynomial system $\{P_1, P_2, P_3, P_4\}$ from Example 2. Since it involves five variables and only four equations, it has either no solutions or infinitely many. A common strategy is to augment the system with a random linear form with integer coefficients. Using HomotopyContinuation.jl, we obtain 15 numerical real solutions to the augmented system, 12 of which correspond to valid integer solutions.

Despite their efficiency, numerical methods have limitations. They provide approximate solutions that may lack exact algebraic structure and require additional validation. They can also struggle with singular solutions, and homotopy continuation methods may occasionally miss solutions altogether.

4.3 Satisfiability Modulo Theories (SMT) solvers

The SMT solvers decide the satisfiability of logical formulas that combine Boolean logic with background theories such as integer or real arithmetic. In particular, the existence of integer or rational solutions to a polynomial system can be expressed as

$$\exists x_1 \ldots \exists x_n\ (f_1(x_1, \ldots, x_n) = 0) \wedge \ldots \wedge (f_m(x_1, \ldots, x_n) = 0).$$

SMT solvers benefit from automated reasoning, efficient decision procedures, and the ability to integrate multiple logical theories. However, they face difficulties with high-degree polynomials and, due to undecidability, cannot guarantee general solutions over the integers. Nevertheless, as shown in Section 5, the SMT solver Z3 successfully finds integer solutions for most of the polynomial systems generated by Algorithms 2 and Algorithm 3.

5 Implementation and Experiments

5.1 Setup

We implemented Algorithm 2, Algorithm 3 and Algorithm 4 in Macaulay2 [6]. The prototype builds on [25], and the source code is publicly available at:

`https://github.com/Erdenebayar2/Synthesizing_Loops.git`

All experiments were conducted on a laptop equipped with a 4.8 GHz Intel i7 processor, 16 GB RAM, and a 25 MB L3 cache. After generating polynomial systems using Algorithms 2 and 3, we used the SMT solver Z3 [27] to search for common nonzero integer solutions.

We evaluated our approach on benchmarks from [5, 24, 26]. The benchmark suite is publicly available at:

`https://github.com/Erdenebayar2/Synthesizing_Loops/software/loops`

Notations. In the following tables, "n" denotes the number of program variables, "m" the number of polynomial invariants in $\boldsymbol{g}$, and "d" their maximal degree. "D" is the maximal degree among the polynomials in $\mathbf{f}_1, \ldots, \mathbf{f}_n$. Let $\mathbf{f}_1 = (f_{1,1}, \ldots, f_{1,l_1}), \ldots, \mathbf{f}_n = (f_{n,1}, \ldots, f_{n,l_n})$ be sequences of polynomials in $\mathbb{C}[\mathbf{x}]$, and set $l = l_1 + \cdots + l_n$. In other words, "l" denotes the total number of generators for the update maps of the loops.

5.2 Experimental Results

Table 1 reports the execution times for Algorithm 2 and for Z3, both of which are run on the polynomial systems generated by Algorithm 2. All timings are given in seconds, with a timeout (indicated by **TL**) of 300 seconds. The label "**F**" indicates that Z3 failed to find an integer solution, while "**NI**" means that no input was passed to Z3 because Algorithm 2 itself timed out.

In most cases, once Algorithm 2 terminates, Z3 finds a common nonzero integer solution within 0.3 seconds. Thus, searching for integer solutions is not a bottleneck. We also observe that providing additional polynomial invariants generally accelerates the termination of Algorithm 2. Overall, these results demonstrate that the main computational challenge lies in generating the polynomial systems, rather than in solving them with Z3.

Polynomial map				D=1, $l=3$		D=1, $l=4$		D=1, $l=5$		D=2, $l=2$		D=2, $l=3$	
Benchmark	n	m	d	Alg. 2	Z3	Alg. 2	Z3	Alg. 2	Z3	Alg. 2	Z3	Alg. 2	Z3
Ex1.2 [5]	2	1	4	0.02	**F**	31.1	**F**	**TL**	**NI**	0.01	0.06	**TL**	**NI**
Ex1.1 [5]	3	2	3	0.01	0.06	0.04	0.06	4.4	0.2	0.01	0.06	0.018	0.07
Ex1.1Ineq	3	2	3	0.009	0.06	11.4	0.06	**TL**	**NI**	0.008	0.05	0.03	0.05
Ex5.2[24]	2	1	2	0.03	0.17	0.16	**TL**	**TL**	**NI**	0.01	0.07	0.13	0.07
sum1 [26]	3	2	2	0.02	0.06	0.13	0.06	1.1	0.06	0.01	0.05	0.02	0.06
square [26]	2	1	2	0.01	0.06	0.5	**TL**	**TL**	**NI**	0.01	0.06	0.015	0.26
square_conj [26]	3	2	2	0.019	0.06	0.14	0.09	0.39	0.06	0.007	0.05	0.015	0.06
fmi1 [26]	2	1	2	1.19	0.06	161.74	0.06	**TL**	**NI**	237.1	0.06	**TL**	**NI**
fmi2 [26]	3	2	2	0.01	0.06	0.015	0.06	0.017	0.07	0.009	0.07	0.01	0.07
fmi3 [26]	3	2	2	0.01	0.06	0.03	0.05	0.39	0.09	0.01	0.06	0.2	0.11
intcbrt [26]	3	2	2	0.25	0.07	16.6	0.08	**TL**	**NI**	0.01	0.06	0.65	0.17
cube_square [26]	3	1	3	0.01	0.07	0.018	**TL**	**TL**	**NI**	0.15	0.06	0.96	106
Ex3	2	1	2	0.014	0.09	0.02	0.1	0.034	0.1	0.13	0.1	0.07	0.12
markov_triples	3	1	2	5.61	0.1	36.2	0.1	134.1	0.6	1.35	0.1	**TL**	**NI**

Table 1 Timings for Algorithm 2 in seconds;

Table 2 presents the polynomial systems produced by Algorithm 2, which define $\mathscr{C}(\mathbf{a}, h, \mathbf{f}; \boldsymbol{g})$. Similarly, Table 3 shows the systems produced by Algorithm 3, which define $U\mathscr{C}(h, \mathbf{f}; \boldsymbol{g})$, together with the execution times for Z3 on those systems. For each choice of "D" and "l", we report the number "s" of nonzero polynomials and whether the system has finitely many solutions. In Tables 2 and 3, "Id" indicates that, within the given structure, the only loop satisfying the specified invariants is the identity map.

Macaulay2 [6] computes the irreducible decomposition of the generated varieties within a few seconds in most of the cases shown in Tables 2 and 3. In many examples, the irreducible components are defined by linear equations. Furthermore, Table 2 shows that even when the dimension is 0, the varieties are never empty; they always consist of finitely many points.

Polynomial map				D=1, $l=3$		D=1, $l=4$		D=1, $l=5$		D=2, $l=2$		D=2, $l=3$	
Benchmark	n	m	d	s	#sols	s	#sols	s	#sols	s	#sols	s	#sols
Ex1.2 [5]	2	1	4	3	∞	4	∞	**TL**	**TL**	2	∞	**TL**	**TL**
Ex1.1 [5]	3	2	3	2	∞	4	∞	6	∞	4	$<\infty$	4	$<\infty$
Ex1.1Ineq	3	2	3	2	∞	4	∞	**TL**	**TL**	4	$<\infty$	4	$<\infty$
Ex5.2[24]	2	1	2	3	∞	3	∞	**TL**	**TL**	3	$<\infty$	3	∞
sum1 [26]	3	2	2	4	Id	6	∞	6	∞	2	$<\infty$	4	$<\infty$
square [26]	2	1	2	2	∞	4	∞	**TL**	**TL**	2	$<\infty$	3	∞
square_conj [26]	3	2	2	4	$<\infty$	6	∞	6	∞	4	$<\infty$	4	$<\infty$
fmi1 [26]	2	1	2	0	∞	0	∞	**TL**	**TL**	0	∞	**TL**	**TL**
fmi2 [26]	3	2	2	2	∞	4	∞	4	∞	2	$<\infty$	4	$<\infty$
fmi3 [26]	3	2	2	4	Id	4	∞	6	∞	2	$<\infty$	4	$<\infty$
intcbrt [26]	3	2	2	4	Id	4	∞	**TL**	**TL**	2	$<\infty$	4	$<\infty$
cube_square [26]	3	1	3	2	∞	3	∞	**TL**	**TL**	3	$<\infty$	5	$<\infty$
Ex3	2	1	2	3	∞	3	∞	7	∞	3	$<\infty$	7	$<\infty$
markov_triples	3	1	2	15	$<\infty$	15	$<\infty$	15	$<\infty$	7	$<\infty$	**TL**	**NI**

Table 2 Data on outputs of Algorithm 2

For every benchmark in Table 1, Algorithm 3 terminates within 0.1 seconds and Z3 within 0.3 seconds. However, in most cases the polynomial equations produced by Algorithm 3 either admit no common solution or yield only the identity loop within the specified structure. This limitation arises from the fact that Algorithm 3 assumes the invariants to be universally inductive, whereas Algorithm 2 makes no such assumption.

Accordingly, and because efficiency allows for it, in Table 3 we use significantly larger templates for the update maps of loops than in Table 1. In particular, the number of generators in Table 3 ranges from 6 to 121, compared to only 3 to 5 in Table 1. For these larger templates, Algorithm 2 fails to terminate within 300 seconds mainly because of costly Gröbner basis computations. In contrast, Algorithm 3 always terminates within 0.6 seconds because it relies solely on linear algebra computations. However, it generally produces a loop of a higher degree, which is more susceptible to numerical instability.

In each column of Table 3, the update maps are generated from all monomials up to degree D.

Polynomial map				D=1				D=2				D=3			
Benchmark	n	m	d	l	Z3	s	#sols	l	Z3	s	#sols	l	Z3	s	#sols
Ex1.2 [5]	2	1	4	6	**TL**	15	$<\infty$	12	**TL**	45	$<\infty$	20	**TL**	91	$<\infty$
Ex1.1 [5]	3	2	3	12	0.1	30	Id	30	0.1	119	∞	60	0.15	304	**TL**
Ex1.1Ineq	3	2	3	12	0.1	30	Id	30	0.1	119	∞	60	0.15	304	**TL**
Ex5.2[24]	2	1	2	6	0.1	6	∞	12	0.14	15	∞	20	**TL**	28	∞
sum1 [26]	3	2	2	12	0.15	14	∞	30	0.16	45	∞	60	0.17	104	**TL**
square [26]	2	1	2	6	0.1	6	∞	12	0.1	15	∞	20	0.1	28	∞
square_conj [26]	3	2	2	12	0.1	14	∞	30	0.16	45	∞	60	**TL**	104	**TL**
fmi1 [26]	2	1	2	6	0.14	6	∞	12	0.14	15	∞	20	0.15	28	∞
fmi2 [26]	3	2	2	12	0.15	14	∞	30	0.15	45	∞	60	0.17	104	**TL**
fmi3 [26]	3	2	2	12	0.2	14	∞	30	0.15	45	∞	60	0.17	104	**TL**
intcbrt [26]	3	2	2	12	0.17	30	Id	30	0.2	119	∞	60	0.25	304	**TL**
cube_square [26]	3	1	3	12	0.2	20	∞	30	0.19	84	**TL**	60	0.5	220	**TL**
Ex3	2	1	2	12	0.21	12	∞	24	0.2	30	∞	40	0.21	56	∞
markov_triples	3	1	2	24	**TL**	40	$<\infty$	60	**TL**	168	**TL**	121	**TL**	304	**TL**

Table 3 Data on outputs of Algorithm 3

References

1. Bayarmagnai, E., Mohammadi, F. & Prébet, R. Algebraic Tools for Computing Polynomial Loop Invariants (Extended Version). *ArXiv Preprint ArXiv:2412.14043*. (2024)
2. Berthomieu, J., Eder, C. & Safey El Din, M. msolve: A Library for Solving Polynomial Systems. *2021 International Symposium On Symbolic And Algebraic Computation*. pp. 51-58 (2021,7)
3. Hrushovski, E., Ouaknine, J., Pouly, A. & Worrell, J. Polynomial invariants for affine programs. *Proceedings Of The 33rd Annual ACM/IEEE Symposium On Logic In Computer Science*. pp. 530-539 (2018)
4. Rodríguez-Carbonell, E. & Kapur, D. Automatic generation of polynomial loop invariants: Algebraic foundations. *Proceedings Of The 2004 International Symposium On Symbolic And Algebraic Computation*. pp. 266-273 (2004)
5. Kenison, G., Kovács, L. & Varonka, A. From Polynomial Invariants to Linear Loops. *Proceedings Of The 2023 International Symposium On Symbolic And Algebraic Computation*. pp. 398-406 (2023).
6. Grayson, D. & Stillman, M. Macaulay2, a software system for research in algebraic geometry. (2002)
7. Rodríguez-Carbonell, E. & Kapur, D. Generating all polynomial invariants in simple loops. *Journal Of Symbolic Computation*. **42**, 443-476 (2007)
8. Kovács, L. Reasoning algebraically about p-solvable loops. *International Conference On Tools And Algorithms For The Construction And Analysis Of Systems*. pp. 249-264 (2008)
9. Basu, S., Pollack, R. & Roy, M. Algorithms in Real Algebraic Geometry. (Springer International Publishing, 2006)
10. Cox, D., Little, J. & O'Shea, D. Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra. (Springer Science & Business Media, 2013)
11. Rodríguez-Carbonell, E. & Kapur, D. Automatic generation of polynomial invariants of bounded degree using abstract interpretation. *Science Of Computer Programming*. **64**, 54-75 (2007)
12. Karr, M. Affine relationships among variables of a program. *Acta Informatica*. **6** pp. 133-151 (1976)
13. Kempf, G. Algebraic Varieties. (Cambridge University Press, 1993)
14. Shafarevich, I.& Reid, M. Basic algebraic geometry.(Springer, 1994)
15. Manna, Z. & Pnueli, A. Temporal verification of reactive systems: safety. (Springer Science & Business Media, 2012)
16. Floyd, R. Assigning meanings to programs. *Program Verification: Fundamental Issues In Computer Science*. pp. 65-81 (1993)
17. Hrushovski, E., Ouaknine, J., Pouly, A. & Worrell, J. On strongest algebraic program invariants. *Journal of the ACM*. **70**, 1-22 (2023)
18. Goharshady, A., Hitarth, S., Mohammadi, F. & Motwani, H. Algebro-geometric Algorithms for Template-based Synthesis of Polynomial Programs. *Proceedings Of The ACM On Programming Languages*. **7**, 727-756 (2023)
19. Kovács, L. Algebra-Based Loop Analysis. *Proceedings Of The 2023 International Symposium On Symbolic And Algebraic Computation*. pp. 41-42 (2023)
20. Oliveira, S., Bensalem, S. & Prevosto, V. Synthesizing invariants by solving solvable loops. International Symposium On Automated Technology For Verification And Analysis. pp. 327-343 (2017)
21. Mayr, E. & Meyer, A. The complexity of the word problem for commutative semi-groups and polynomial ideals. *Advances In Mathematics*. **46** pp. 305-329 (1982)
22. Von Zur Gathen, J. & Gerhard, J. Modern computer algebra. (Cambridge university press, 2013)
23. Ait El Manssour R, Kenison G, Shirmohammadi M, Varonka A. Simple Linear Loops: Algebraic Invariants and Applications. Proceedings of the ACM on Programming Languages. 2025 Jan 7;9(POPL):745-71.
24. Hitarth, S., Kenison, G., Kovács, L. & Varonka, A. Linear Loop Synthesis for Quadratic Invariants. *41st International Symposium On Theoretical Aspects Of Computer Science (STACS 2024)*. **289** pp. 41:1-41:18 (2024)
25. Bayarmagnai, E., Mohammadi, F. & Prébet, R. Algebraic Tools for Computing Polynomial Loop Invariants. *Proceedings Of The 2024 International Symposium On Symbolic And Algebraic Computation*. pp. 371-381 (2024).
26. Humenberger, A., Amrollahi, D., Bjørner, N. & Kovács, L. Algebra-Based Reasoning for Loop Synthesis. *Form. Asp. Comput.*. **34** (2022,7).
27. De Moura, L. & Bjørner, N. Z3: an efficient SMT solver. *Proceedings Of The Theory And Practice Of Software, 14th International Conference On Tools And Algorithms For The Construction And Analysis Of Systems*. pp. 337-340 (2008)
28. Srivastava, S., Gulwani, S. & Foster, J. From program verification to program synthesis. *SIGPLAN Not.*. **45**, 313-326 (2010,1).
29. Breiding, P. & Timme, S. HomotopyContinuation.jl: A Package for Homotopy Continuation in Julia. *Mathematical Software – ICMS 2018*. pp. 458-465 (2018)
30. Hilbert, D. Ueber die vollen Invariantensysteme. *Mathematische Annalen*. **42**, 313-373 (1893).
31. Seidenberg, A. A New Decision Method for Elementary Algebra. *Annals Of Mathematics*. **60**, 365-374 (1954).
32. Tarski, A. & McKinsey, J. A Decision Method for Elementary Algebra and Geometry. (University of California Press, 1951).
33. Shlapentokh, A. Defining Integers. *The Bulletin Of Symbolic Logic*. **17**, 230-251 (2011)
34. Cook, W. & Steffy, D. Solving Very Sparse Rational Systems of Equations. *ACM Trans. Math. Softw.*. **37** (2011,2).
35. Durvye, C. & Lecerf, G. A concise proof of the Kronecker polynomial system solver from scratch. *Expositiones Mathematicae*. **26**, 101-139 (2008).
36. Liwski E, Mohammadi F. Solvable and nilpotent matroids: Realizability and irreducible decomposition of their associated varieties. Journal of Symbolic Computation, 2025, p. 102484.
37. Liwski E, Mohammadi F, Prébet R. Efficient algorithms for minimal matroid extensions and irreducible decompositions of circuit varieties. arXiv preprint arXiv:2504.16632, 2025.
38. Rouillier, F. Solving Zero-Dimensional Systems Through the Rational Univariate Representation. *Applicable Algebra In Engineering, Communication And Computing*. **9**, 433-461 (1999)
39. Loos, R. Computing Rational Zeros of Integral Polynomials by p-Adic Expansion. *SIAM Journal On Computing*. **12**, 286-293 (1983)
40. Bayarmagnai, E., Mohammadi, F. & Prébet, R. Beyond Affine Loops: A Geometric Approach to Program Synthesis. *Proceedings Of The 14th ACM SIGPLAN International Workshop On The State Of The Art In Program Analysis*. pp. 1-7 (2025)
41. Cassels JW. An introduction to Diophantine approximation, 1957.