Available online at www.sciencedirect.com
ScienceDirect
Journal of Symbolic Computation 42 (2007) 443–476
Journal of Symbolic Computation
www.elsevier.com/locate/jsc

Generating all polynomial invariants in simple loops[☆]

E. Rodríguez-Carbonell[a,*], D. Kapur[b]

[a] Software Department, Technical University of Catalonia, Jordi Girona, 1-3 08034 Barcelona, Spain
[b] Department of Computer Science, University of New Mexico, Albuquerque, NM 87131-0001, USA

Received 22 September 2005; accepted 18 January 2007
Available online 25 January 2007

Abstract

This paper presents a method for automatically generating all polynomial invariants in simple loops. It is first shown that the set of polynomials serving as loop invariants has the algebraic structure of an ideal. Based on this connection, a fixpoint procedure using operations on ideals and Gröbner basis constructions is proposed for finding all polynomial invariants. Most importantly, it is proved that the procedure terminates in at most $m+1$ iterations, where m is the number of program variables. The proof relies on showing that the irreducible components of the varieties associated with the ideals generated by the procedure either remain the same or increase their dimension at every iteration of the fixpoint procedure. This yields a *correct* and *complete* algorithm for inferring conjunctions of polynomial equalities as invariants. The method has been implemented in Maple using the Groebner package. The implementation has been used to automatically discover non-trivial invariants for several examples to illustrate the power of the technique.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Loop invariant; Ideal of polynomials; Gröbner basis

1. Introduction

Program verification based on Floyd–Hoare's inductive assertion method, using preconditions, postconditions and loop invariants, was considered a major research problem in the seventies, leading to the development of many program verification systems. However, limited

☆ This is a full version of the article that appeared in the ISSAC 2004 proceedings.

* Corresponding author.
E-mail addresses: erodri@lsi.upc.edu (E. Rodríguez-Carbonell), kapur@cs.unm.edu (D. Kapur).
URLs: http://www.lsi.upc.edu/~erodri (E. Rodríguez-Carbonell), http://www.cs.unm.edu/~kapur (D. Kapur).

progress was made in achieving the goal of mechanical verification of (even partial) correctness of programs because:

- theorem provers, needed to establish the validity of the verification conditions, were not sufficiently powerful;
- programs had to be annotated with loop invariants, for which user's intervention was critical; the few tools developed then (German and Wegbreit, 1975) for this purpose were not effective.

Nonetheless, for life-critical systems it is still imperative to verify properties of programs (Hoare, 2003). With substantial progress in automated reasoning, several verification techniques have emerged in the form of static analysis of programs (type checking, type inference, extended static checking, abstract interpretation, etc.), model checking, as well as applications of theorem proving to the verification of software and hardware. However, the annotation burden remains. Our work attempts to deal with the problem of automatically generating loop invariants, which is still unsolved.

In Rodríguez-Carbonell and Kapur (2005), an abstract framework for finding loop invariants was presented. Properties of the language used for expressing invariants were identified so that a generic correct and complete procedure for computing loop invariants could be formulated.

In this paper, which is an extended version of Rodríguez-Carbonell and Kapur (2004b), we instantiate the approach proposed in Rodríguez-Carbonell and Kapur (2005) when invariants are expressed as conjunctions of polynomial equalities. We consider programs composed by *simple* loops, i.e., unnested loops. It is shown that for a given loop, the set $\{p\}$ of polynomials such that $p = 0$ is an invariant, i.e., p evaluates to 0 whenever the control flow reaches the loop entry point, is a polynomial ideal; this ideal is henceforth called the *invariant polynomial ideal* of the loop. A fixpoint procedure for computing such ideal of polynomials is proposed, which is shown to be *correct* and *complete*. If a loop does not have any non-trivial polynomial invariant, the procedure will generate the polynomial 0 (which is equivalent to *true*) as invariant.

The main result of this paper is a proof of termination of the procedure for finding polynomial invariants when assignment statements appearing in a loop are restricted to be *solvable* (which generalise affine assignments) and have positive rational eigenvalues (this technical restriction is motivated later in the paper). It is shown that the invariant ideal is computed in at most $m + 1$ iterations, where m is the number of program variables. The proof of termination uses techniques from algebraic geometry to analyse the variety associated with the ideal approximating, at every iteration of the procedure, the invariant polynomial ideal of the loop. It is shown that the irreducible components of the varieties associated with the generated ideals either remain the same or increase their dimension at every iteration. Thus, at each step, either the invariant polynomial ideal has already been computed, or the minimum of the dimensions of the non-invariant irreducible components of the associated variety increases.

This proof of termination requires that assignments are solvable mappings with positive rational eigenvalues. However, we are unaware of any example for which the procedure does not terminate. We thus conjecture that the requirement of solvable mappings possessing positive rational eigenvalues is unnecessary. In fact, it is also proved in the paper that, if the assignment statements in the body of a loop commute (i.e., the order in which assignments are executed does not affect the result), the procedure for discovering invariants terminates in at most $n + 1$ iterations, where n is the number of assignments in the body of the loop; this latter proof does not require the positiveness or rationality of the eigenvalues of the assignments.

The procedure for discovering invariants has been implemented in Maple using the Groebner package for manipulating ideals (see Cox et al. (1996) for theoretical details). The implementation has been successfully applied to several non-trivial programs to automatically generate conjunctions of polynomial equalities as loop invariants. Some of these examples are used in this paper to illustrate the key concepts of the approach.

The rest of the paper is organised as follows. After reviewing related work in the next subsection, we introduce the theoretical notions in Section 2. In Section 3 we describe in detail the kind of loops we are going to consider. In Section 4 it is shown that the set of invariant polynomials of a loop has the algebraic structure of an ideal, which immediately suggests that polynomial ideal theory and algebraic geometry can give insight into the problem of finding loop invariants. Section 5 presents the procedure for generating polynomial invariants, expressed in terms of ideals. Section 6 gives the proofs of termination of the invariant generation procedure. We show in Section 7 how to implement this procedure using Gröbner bases. We illustrate the method in Section 8 with some examples; a table is also given providing information on programs successfully analysed with our implementation. Finally, Section 9 summarises the contributions of the paper and gives an overview on future research.

1.1. Background and related work

The problem of discovering invariants is a cornerstone in the verification of systems. For this reason, invariant generation has been a major research goal since the seventies (Wegbreit, 1974, 1975; German and Wegbreit, 1975; Katz and Manna, 1976; Cousot and Cousot, 1976; Suzuki and Ishihata, 1977; Dershowitz and Manna, 1978). More specifically, the synthesis of invariant affine equalities between program variables, a particular case of invariant polynomial equalities, was first addressed in Karr (1976).

Recently, the interest in automatically deriving invariants of imperative programs has seen a resurgence. For example, in Gulwani and Necula (2005) a unified framework for random interpretation is proposed, which allows the extension of randomised intraprocedural analysers to context-sensitive interprocedural analysers; further, this framework is instantiated, e.g., for the discovery of invariant affine equalities.

Among the recent work on invariant generation, a remarkable amount of literature has been devoted to the class of polynomial equality invariants. For example, for programs with affine assignments, Müller-Olm and Seidl (2004b) have proposed an interprocedural method for computing polynomial equalities of bounded degree as invariants. The same authors Müller-Olm and Seidl (2004a) have also designed a technique for discovering all the polynomial invariants of bounded degree in a program with polynomial assignments and disequality tests.

In Sankaranarayanan et al. (2004) have also presented a method for discovering invariant polynomials. Their technique is an instance of the so-called constraint-based invariant generation approach, as opposed to classical abstract interpretation: it starts with a template polynomial with undetermined coefficients; then, by imposing that the template is invariant, a system of constraints is obtained by means of the Gröbner basis algorithm and heuristics; finally, the system of constraints is solved and each solution for the values of the coefficients yields an invariant. Kapur has proposed a related approach using quantifier elimination (Kapur, 2003). Unlike these techniques, a major aspect of our work is that an implementation of our method has been performed, which has been successfully applied on many examples (see Sections 7.1 and 8 for details).

Finally, in Rodríguez-Carbonell and Kapur (2004a, 2007) we have presented an approach based on abstract interpretation for generating polynomial invariants of bounded degree. Whereas the technique can be applied to nested loops and takes into account tests in conditional statements and loops, in order to guarantee termination it is necessary to employ a widening operator, which may miss invariants. A similar method has been suggested by Colón (2004) based on the concept of pseudo-ideal.

In contrast to the aforementioned techniques, from a theoretical viewpoint the approach proposed in this paper has the advantage of not requiring any a priori bound on the degrees of the polynomials to be generated. This allows us to find *all* polynomial invariants, which is regarded as a "challenging open problem" (Müller-Olm and Seidl, 2004a), at the cost of restricting the structure of the programs to which the method can be applied.

2. Preliminaries

Given a field $\mathbb{K}$, let $\mathbb{K}[\boldsymbol{x}] = \mathbb{K}[x_1, \ldots, x_m]$ denote the ring of polynomials in the variables $\boldsymbol{x} = (x_1, \ldots, x_m)$ with coefficients from $\mathbb{K}$. An *ideal* is a non-empty set $I \subseteq \mathbb{K}[\boldsymbol{x}]$ that is closed under addition and is such that if $p \in \mathbb{K}[\boldsymbol{x}]$ and $q \in I$, then $pq \in I$. Given a set of polynomials $S \subseteq \mathbb{K}[\boldsymbol{x}]$, the *ideal spanned by* S is

$$\left\{ q \in \mathbb{K}[\boldsymbol{x}] \mid \exists k \geq 1 \;\; q = \sum_{j=1}^{k} p_j q_j \text{ with } p_j \in \mathbb{K}[\boldsymbol{x}], q_j \in S \right\}.$$

This is the minimal ideal containing S, and we denote it by $\langle S \rangle_{\mathbb{K}[\boldsymbol{x}]}$ or simply by $\langle S \rangle$. For an ideal $I \subseteq \mathbb{K}[\boldsymbol{x}]$, a set $S \subseteq \mathbb{K}[\boldsymbol{x}]$ such that $I = \langle S \rangle$ is called a *basis* of I, and we say that S *generates* I. Given two ideals $I, J \subseteq \mathbb{K}[\boldsymbol{x}]$, their *intersection* $I \cap J$ is an ideal; however, there is no general expression for a basis of $I \cap J$ in terms of generators of I and J.

For any set S of polynomials in $\mathbb{K}[\boldsymbol{x}]$, the *variety* of S over $\mathbb{K}^m$ is defined as its set of zeros, $\mathbf{V}(S) = \{\boldsymbol{\omega} \in \mathbb{K}^m \mid p(\boldsymbol{\omega}) = 0 \; \forall p \in S\}$. When taking varieties we can assume S to be an ideal, since $\mathbf{V}(\langle S \rangle) = \mathbf{V}(S)$. For $A \subseteq \mathbb{K}^m$, the ideal $\mathbf{I}(A) = \{p \in \mathbb{K}[\boldsymbol{x}] \mid p(\boldsymbol{\omega}) = 0 \; \forall \boldsymbol{\omega} \in A\}$ is called the *ideal of* A. We write $\mathbf{IV}(S)$ instead of $\mathbf{I}(\mathbf{V}(S))$.

Ideals and varieties are dual concepts, in the sense that, given any two ideals I, J, $\mathbf{V}(I \cap J) = \mathbf{V}(I) \cup \mathbf{V}(J)$ and, if $I \subseteq J$, then $\mathbf{V}(I) \supseteq \mathbf{V}(J)$. Further, for $A, B \subseteq \mathbb{K}^m$ (in particular, if A, B are varieties), then $\mathbf{I}(A \cup B) = \mathbf{I}(A) \cap \mathbf{I}(B)$ and $A \subseteq B$ implies $\mathbf{I}(A) \supseteq \mathbf{I}(B)$. For any ideal I, the inclusion $I \subseteq \mathbf{IV}(I)$ always holds; $\mathbf{IV}(I)$ represents the largest set of polynomials with the same zeros as I.[1] Since any I satisfying $I = \mathbf{IV}(I)$ is the ideal of the variety $\mathbf{V}(I)$, we say that any such I is an *ideal of variety*. For any $A \subseteq \mathbb{K}^m$, it can be seen that the ideal $\mathbf{I}(A)$ is an ideal of variety, i.e., $\mathbf{IVI}(A) = \mathbf{I}(A)$. For further detail on these concepts, see Cox et al. (1996).

A *polynomial mapping* is a vector of polynomials $\boldsymbol{g} = (g_1, \ldots, g_m) \in \mathbb{K}[\boldsymbol{x}]^m$. In particular, a mapping $\boldsymbol{g}$ is said to be *affine* if it is of the form $\boldsymbol{g}(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{b}$, where A is an $m \times m$ matrix with coefficients in $\mathbb{K}$, and $\boldsymbol{b} \in \mathbb{K}^m$.

A polynomial mapping $\boldsymbol{g} \in \mathbb{K}[\boldsymbol{x}]^m$ is *invertible* if $\exists \boldsymbol{g}' \in \mathbb{K}[\boldsymbol{x}]^m$ such that $\boldsymbol{g}'(\boldsymbol{g}(\boldsymbol{x})) = \boldsymbol{g}(\boldsymbol{g}'(\boldsymbol{x})) = \boldsymbol{x}$; unless otherwise stated, $\boldsymbol{g}^{-1}$ denotes this mapping $\boldsymbol{g}'$. Given a set $S \subseteq \mathbb{K}[\boldsymbol{x}]$ and a polynomial mapping $\boldsymbol{g} \in \mathbb{K}[\boldsymbol{x}]^m$, we define their *composition* as $S \circ \boldsymbol{g}(\boldsymbol{x}) = \{p(g_1(\boldsymbol{x}), \ldots, g_m(\boldsymbol{x})) \in \mathbb{K}[\boldsymbol{x}] \mid p \in S\}$. If $I \subseteq \mathbb{K}[\boldsymbol{x}]$ is an ideal and $\boldsymbol{g}$ is an invertible polynomial mapping, then $I \circ \boldsymbol{g}(\boldsymbol{x})$ is also an ideal.

[1] If the field $\mathbb{K}$ is algebraically closed then $\mathbf{IV}(I) = \mathrm{Rad}(I)$, the *radical* of I, which is the set of polynomials p such that there exists $k \in \mathbb{N}$ satisfying $p^k \in I$.

3. Programming model

In this section we present our programming model: the domain of variables, the structure of loops and the expressions allowed in programs. Below, we give the intuition behind this programming model and, in particular, the rationale for the restrictions imposed on assignments.

Since our goal is to automatically generate polynomial equalities as invariants, we must approximate tests, as well as assignments, by polynomials. In the case of tests, we have decided to ignore them, since including conditions easily leads to non-computability: in Müller-Olm and Seidl (2004) it is proved that the set of all affine equality invariants is not computable if affine equality tests are allowed; in particular, the set of all polynomial equality invariants is not computable if polynomial equality tests are permitted. Moreover, despite the loss of precision involved with this abstraction, we were able to employ the generated invariants in order to prove the partial correctness of many programs, some of which are shown in Sections 7 and 8.

Regarding assignments, if arbitrary polynomial mappings are allowed, their repeated applications can lead to non-polynomial effects. For example, even if an assignment is as simple as $x := 2x$, its repeated application gives $x = 2^k x_0$ after k applications on the starting value x_0 of x. However, in the case of affine mappings, their repeated applications can be expressed in terms of polynomials multiplied by exponentials of the eigenvalues of the transformation matrices. Further, these exponentials can be related to each other using auxiliary variables and polynomial relations. It will be shown that for affine mappings, their repeated applications can indeed be expressed in a "polynomial manner".

In fact, the requirement that the assignments be affine can be relaxed. To that end we introduce the notion of *solvable* mappings; these are recursively built using blocks of variables, starting with a block of variables whose assignments are expressed as affine transformations. Moreover, the proof of termination of the proposed procedure needs yet another condition, namely that the eigenvalues of the transformation matrices associated with the assignments be positive and rational.

3.1. Simple loops

Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)$ be the program variables, which are assumed to take rational values. Regarding loop structure, the commands we allow in loops are assignments and conditional statements. Programs may have to be abstracted so that guards in loops and conditional statements can be ignored; in such cases, it is often useful to represent a conditional statement `if` $\boldsymbol{B}(\boldsymbol{x})$ `then` $\boldsymbol{x} := \boldsymbol{f}(\boldsymbol{x})$ `else` $\boldsymbol{x} := \boldsymbol{g}(\boldsymbol{x})$ as $\boldsymbol{x} := \boldsymbol{f}(\boldsymbol{x})$ `or` $\boldsymbol{x} := \boldsymbol{g}(\boldsymbol{x})$ (thus ignoring the boolean condition $\boldsymbol{B}(\boldsymbol{x})$). Thus we assume that loops have the following simple form:

```
while ? do
    $\boldsymbol{x} := \boldsymbol{f}_1(\boldsymbol{x})$;
    or
    ...
    or
    $\boldsymbol{x} := \boldsymbol{f}_n(\boldsymbol{x})$;
end while,
```

where each $\boldsymbol{x} := \boldsymbol{f}_i(\boldsymbol{x})$ is a vector of simultaneous assignments of all the variables, i.e., $\boldsymbol{f}_i : \mathbb{Q}^m \to \mathbb{Q}^m$ $(1 \le i \le n)$, and ? means that the exit condition is ignored. Any sequence of successive assignments of variables can be transformed into this form without loss of generality (with the variables not being assigned getting the identity map).

3.2. Solvable mappings

In this section we introduce the concept of *solvable* mapping, which generalises that of affine mapping. Intuitively, a solvable mapping $\boldsymbol{g}$ is a polynomial mapping such that the recurrence $\boldsymbol{x_{s+1}} = \boldsymbol{g}(\boldsymbol{x_s})$ can be *solved* effectively and such that its solution (which is given by the general power $\boldsymbol{g}^s$) has a "polynomial structure", i.e., it can be expressed as a vector of polynomials, possibly using new variables related by polynomial equations (see the example below).

Given $\boldsymbol{g} \in \mathbb{Q}[\boldsymbol{x}]^m$ and a subvector of the variables $\boldsymbol{w} \subseteq \boldsymbol{x}$, we write $g_{\boldsymbol{w}} = (g_j)_{x_j \in \boldsymbol{w}} : \mathbb{Q}^m \to \mathbb{Q}^{|\boldsymbol{w}|}$. For instance, for the mapping $\boldsymbol{g}(a, b, p, q) = (a-1, b, p, q+bp)$:

$$g_a(a,b,p,q) = a-1,$$
$$g_{(a,b,p)}(a,b,p,q) = (a-1,b,p),$$
$$g_q(a,b,p,q) = q+bp.$$

Definition 1. Let $\boldsymbol{g} \in \mathbb{Q}[\boldsymbol{x}]^m$ be a polynomial mapping. $\boldsymbol{g}$ is *solvable* if there exists a partition of $\boldsymbol{x}$ into subvectors of variables, $\boldsymbol{x} = \boldsymbol{w_1} \cup \cdots \cup \boldsymbol{w_k}$, $\boldsymbol{w_i} \cap \boldsymbol{w_j} = \emptyset$ if $i \neq j$, such that $\forall j : 1 \le j \le k$ we have

$$g_{\boldsymbol{w_j}}(\boldsymbol{x}) = M_j \boldsymbol{w_j}^T + \boldsymbol{P_j}(\boldsymbol{w_1}, \ldots, \boldsymbol{w_{j-1}}),$$

where $M_j \in \mathbb{Q}^{|\boldsymbol{w_j}| \times |\boldsymbol{w_j}|}$ is a matrix and $\boldsymbol{P_j}$ is a vector of $|\boldsymbol{w_j}|$ polynomials in the ring $\mathbb{Q}[\boldsymbol{w_1}, \ldots, \boldsymbol{w_{j-1}}]$. For $j = 1$, $\boldsymbol{P_1}$ must be a constant vector, implying that $g_{\boldsymbol{w_1}}$ is an affine mapping.

The *eigenvalues* of $\boldsymbol{g}$ are the eigenvalues of the matrices M_j, $1 \le j \le k$.

In our programming model, assignment mappings have to be solvable with positive rational eigenvalues. We show in Section 8 that there are many programs that satisfy this requirement.

Notice that any affine mapping $\boldsymbol{g}(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{b}$ is solvable, since we can take $\boldsymbol{w_1} = \boldsymbol{x}$, $M_1 = A$, $\boldsymbol{P_1} = \boldsymbol{b}$, and then the eigenvalues of $\boldsymbol{g}$ are the eigenvalues of A. Consider for example the following loop, which is an abstraction of a program that computes the product of two integers x and y:

```
    $(a, b, p, q)$:=$(x, y, 1, 0)$;
while ? do
    $(a, b, p, q) := (a/2, b/2, 4p, q)$;
    or $(a, b, p, q) := (a-1, b, p, q+bp)$;
    or $(a, b, p, q) := (a, b-1, p, q+ap)$;
    or $(a, b, p, q) := (a-1, b-1, p, q+(a+b-1)p)$;
end while
```

For instance, the first assignment mapping $\boldsymbol{g_1}(a,b,p,q) = (a/2, b/2, 4p, q)$ is affine and therefore solvable; its eigenvalues are $\{1/2, 4, 1\}$. Also the non-linear mapping $\boldsymbol{g_2}(a,b,p,q) = (a-1, b, p, q+bp)$ is solvable: we can take $\boldsymbol{w_1} = (a,b,p)$, $M_1 = \textit{diagonal}(1,1,1)$, $\boldsymbol{P_1} = (-1,0,0)$, as $(g_2)_{(a,b,p)} = (a-1,b,p)$; and then $\boldsymbol{w_2} = (q)$, with $M_2 = (1)$ and $\boldsymbol{P_2} = (bp)$, as $(g_2)_q = q+bp$. In this case the eigenvalues of $\boldsymbol{g_2}$ are just $\{1\}$.

To motivate the term *solvable*, let us compute ${\boldsymbol{g_2}}^s$ for an arbitrary $s \in \mathbb{N}$, which is the effect of applying the second assignment s times. This is equivalent to explicitly solving the recurrence $(a_{s+1}, b_{s+1}, p_{s+1}, q_{s+1}) = \boldsymbol{g_2}(a_s, b_s, p_s, q_s)$, whose (symbolic) solution is ${\boldsymbol{g_2}}^s(a_0, b_0, p_0, q_0)$. We first solve the recurrence for a_s, b_s, p_s (notice the correspondence with the partition of the variables above):

$$\begin{cases} a_{s+1} &= a_s - 1 \\ b_{s+1} &= b_s \\ p_{s+1} &= p_s \end{cases} \qquad \Longrightarrow \qquad \begin{cases} a_s &= a_0 - s \\ b_s &= b_0 \\ p_s &= p_0 \end{cases}$$

Now, as $q_{s+1} = q_s + b_s p_s$, by plugging in the expressions for the variables that have already been solved we get the recurrence $q_{s+1} = q_s + b_0 p_0$. The solution to this equation is $q_s = q_0 + b_0 p_0 s$, and thus ${\boldsymbol{g_2}}^s(a, b, p, q) = (a - s, b, p, q + bps)$. Notice that, in this case, we have obtained a vector of polynomials in the program variables a, b, p, q and in the auxiliary variable s. In Section 7, this observation is generalised to all solvable mappings with positive rational eigenvalues.

4. Ideals of invariant polynomials

In this section we give the definition of invariant polynomial for the loops we are considering, and we also see that the algebraic structure of an ideal is the natural object when studying them.

Intuitively, an invariant polynomial is a polynomial that evaluates to 0 at any program state at the loop entry point. For example, in the loop

```
    $(a,\ b,\ c)$:=$(0,\ 1,\ 1)$;
    while ? do
        $(a,\ b,\ c)$:=$(a+1,\ b+c+2,\ c+2)$;
    end while
```

it can be seen that the polynomials $c - 2a - 1$ and $b - (a + 1)^2$ always yield 0 when evaluated at the loop entry point, and so are invariant.

We write the tuple of right-hand sides of loop assignments $\boldsymbol{f_1}, \ldots, \boldsymbol{f_n}$ as $(\boldsymbol{f})$. Consider the set of strings over the alphabet $[n] = \{1, \ldots, n\}$, which we denote by $[n]^*$. For every string $\sigma \in [n]^*$ we inductively define the mapping $(\boldsymbol{f})^\sigma$ as

$$(\boldsymbol{f})^\lambda(\boldsymbol{x}) = \boldsymbol{x}, \quad (\boldsymbol{f})^{\sigma.i}(\boldsymbol{x}) = \boldsymbol{f_i}(\,(\boldsymbol{f})^\sigma(\boldsymbol{x})\,),$$

where λ is the empty string. Each string σ represents an execution path of the loop, and $(\boldsymbol{f})^\sigma$ maps initial states of the loop to states after executing the path σ.

Definition 2. Given a set of initial conditions $I_0 \subseteq \mathbb{Q}[\boldsymbol{x}]$, a polynomial $p \in \mathbb{Q}[\boldsymbol{x}]$ is *invariant* at the loop entry point with respect to I_0 if

$$\forall \sigma \in [n]^* \ \forall \boldsymbol{\omega} \in \mathbf{V}(I_0), \quad p((\boldsymbol{f})^\sigma(\boldsymbol{\omega})) = 0.$$

Notice that, if the polynomial p is invariant with respect to I_0, then it is invariant with respect to $\langle I_0 \rangle$, as $\mathbf{V}(I_0) = \mathbf{V}(\langle I_0 \rangle)$. So we can assume that I_0 is always an ideal. In the example above, we have $I_0 = \langle a, b - 1, c - 1 \rangle$.

The following result shows that the set of all polynomial invariants with respect to a given I_0 is an ideal:

Proposition 3. *Given an ideal $I_0 \subseteq \mathbb{Q}[\boldsymbol{x}]$,*

$$I_\infty := \{p \in \mathbb{Q}[\boldsymbol{x}] \mid \forall \sigma \in [n]^* \ \ \forall \boldsymbol{\omega} \in \mathbf{V}(I_0) \ \ p((\boldsymbol{f})^\sigma(\boldsymbol{\omega})) = 0\}$$

is an ideal.

Proof. As $0 \in I_\infty$, the sum of two polynomials in I_∞ is in I_∞, and the product of an arbitrary polynomial by a polynomial in I_∞ is in I_∞ too, I_∞ is an ideal. □

We will refer to I_∞ as the *invariant polynomial ideal* of the loop. By Hilbert's basis theorem, I_∞ has a finite basis. The conjunction of polynomial equations corresponding to the polynomials in any of its bases, completely describes the invariants of the loop. The key challenge is in computing I_∞. The rest of the paper addresses this issue.

5. Invariant generation procedure

In this section we describe a fixpoint procedure which, given the assignment mappings $\boldsymbol{f_1}, \ldots, \boldsymbol{f_n}$ and an ideal I_0 of polynomials satisfied by the initial values, returns the invariant polynomial ideal I_∞ on termination. We will refer to it as the *Invariant Generation Procedure*.

In order to have a one-to-one correspondence between ideals and varieties, we need that all ideals below, including I_0, be ideals of variety, i.e., $I = \mathbf{IV}(I)$. The Invariant Generation Procedure[2] is as follows:

```
**Input:**
    • The solvable mappings with positive rational eigenvalues $\boldsymbol{f_1}, \ldots, \boldsymbol{f_n}$
    of the assignments.
    • An ideal $I_0$ of polynomials satisfied by the initial values such that
    $I_0 = \mathbf{IV}(I_0)$.

**Output:**
    • The invariant polynomial ideal $I_\infty$.

var $I, I'$ : ideals in $\mathbb{Q}[\boldsymbol{x}]$ end var

$I := I_0$
do
    $I' := I$
    $I := \bigcap_{s \in \mathbb{N}} \bigcap_{i=1}^{n} I \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})$
while $I' \neq I$
return $I$
```

The assignment mappings are invertible, as solvable mappings with positive rational eigenvalues are invertible (see Appendix A.2).

Theorem 5 below ensures that, on termination, the result, i.e., the ideal stored in the variable I, is *correct*, in the sense that all polynomials contained in it are invariant for the loop, and *complete*, in the sense that it does not miss any polynomial invariant.

[2] This procedure can be seen in the abstract interpretation framework (Cousot and Cousot, 1977) as an accelerated forward propagation using ideals of polynomials as abstract values. For the sake of self-containedness, we have not used this approach.

Let us denote the ideal stored in the variable I at the N-th iteration by I_N. We need the following lemma:

Lemma 4. $\forall N \in \mathbb{N}$, $I_\infty \subseteq I_N$.

Proof. Let us prove the lemma by induction over the number of iterations N. If $N = 0$, since $\forall \boldsymbol{\omega} \in \mathbf{V}(I_0)\ \forall p \in I_\infty\ p(\boldsymbol{\omega}) = 0$, we have $I_\infty \subseteq \mathbf{IV}(I_0) = I_0$.

Now let us prove the inductive step. If $I_\infty \subseteq I_N$, as $I_{N+1} = \bigcap_{s\in\mathbb{N}} \bigcap_{i=1}^{n} I_N \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})$, it is enough to prove that $I_\infty \subseteq \bigcap_{s\in\mathbb{N}} \bigcap_{i=1}^{n} I_\infty \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})$. We have to show that for any $s \in \mathbb{N}$ and $1 \le i \le n$, $I_\infty \subseteq I_\infty \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})$, or equivalently that $\forall p \in I_\infty$, $p(\boldsymbol{f_i}^{s}(\boldsymbol{x})) \in I_\infty$. But this is the case, as $\forall \sigma \in [n]^*$, $\forall \boldsymbol{\omega} \in \mathbf{V}(I_0)$,

$$p(\boldsymbol{f_i}^{s}((\boldsymbol{f})^{\sigma}(\omega))) = p((\boldsymbol{f})^{\sigma.\overbrace{i,\dots,i}^{s \text{ times}}}(\omega)) = 0. \quad \square$$

Theorem 5. *If the Invariant Generation Procedure terminates, $I = I_\infty$.*

Proof. By Lemma 4, $I \supseteq I_\infty$. Below, we show that $I \subseteq I_\infty$. We prove that $\forall p \in I\ \forall \sigma \in [n]^*$ $\forall \boldsymbol{\omega} \in \mathbf{V}(I_0)\ p((\boldsymbol{f})^{\sigma}(\boldsymbol{\omega})) = 0$ by induction over the length of σ.

If $\sigma = \lambda$, then $\forall p \in I\ \forall \boldsymbol{\omega} \in \mathbf{V}(I_0)\ p((\boldsymbol{f})^{\lambda}(\boldsymbol{\omega})) = 0$ as $I \subseteq I_0$. Now, assume that the claim holds for strings of length $\le k$. Given σ of length $k+1$, we can write $\sigma = \tau.i$ for certain $\tau \in [n]^*$ of length k and i such that $1 \le i \le n$. But if the procedure terminates, then $I \subseteq \bigcap_{s\in\mathbb{N}} \bigcap_{i=1}^{n} I \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})$; in particular, $I \subseteq I \circ \boldsymbol{f_i}^{-1}(\boldsymbol{x})$, i.e., $\forall p \in I\ p(\boldsymbol{f_i}(\boldsymbol{x})) \in I$. Then

$$p((\boldsymbol{f})^{\sigma}(\boldsymbol{\omega})) = p((\boldsymbol{f})^{\tau.i}(\boldsymbol{\omega})) = p(\boldsymbol{f_i}((\boldsymbol{f})^{\tau}(\boldsymbol{\omega}))) = 0$$

by induction hypothesis. $\square$

6. Termination of the invariant generation procedure

In this section, we give a proof of termination of the Invariant Generation Procedure under the condition that the assignment mappings are solvable and have positive eigenvalues. For the sake of simplicity, we will work in the real field $\mathbb{R}$. As in Section 5, let I_N stand for the ideal computed at the end of the N-th iteration of the Invariant Generation Procedure.

As a motivating example for illustrating the key ideas, consider the following loop:

```
(x, y):=(0, 0);
while ? do
    (x, y):=(x + 1, y);    or    (x, y):=(x, y + 1);
end while
```

This toy program begins with the point $(0, 0)$ and then repeatedly chooses non-deterministically to move horizontally or vertically, thus covering all pairs of natural numbers $\mathbb{N} \times \mathbb{N}$.

Let us apply the procedure. In this case we have that

$$\boldsymbol{f_1}(x, y) = (x + 1, y), \quad \boldsymbol{f_1}^{-s}(x, y) = (x - s, y),$$
$$\boldsymbol{f_2}(x, y) = (x, y + 1), \quad \boldsymbol{f_2}^{-s}(x, y) = (x, y - s).$$

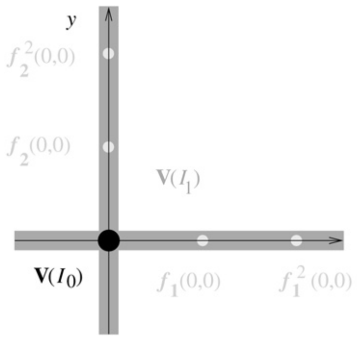

Fig. 1. Varieties of $\mathbf{V}(I_0)$ and $\mathbf{V}(I_1)$.

As both x and y are initialised to 0 before entering the loop, $I_0 = \langle x, y\rangle$. So $I_0 \circ \boldsymbol{f_1}^{-s}(x, y) = \langle x-s, y\rangle$ and $I_0 \circ \boldsymbol{f_2}^{-s}(x, y) = \langle x, y-s\rangle$. We have to compute $\bigcap_{s\in\mathbb{N}} I_0 \circ \boldsymbol{f_i}^{-s}(x, y)$ for $i = 1, 2$. The finite intersection from 0 to a certain $N \in \mathbb{N}$ is (for $i = 1$)

$$\bigcap_{s=0}^{N} I_0 \circ \boldsymbol{f_1}^{-s}(x, y) = \bigcap_{s=0}^{N} \langle x - s, y\rangle = \left\langle \prod_{s=0}^{N} (x - s), y \right\rangle,$$

as $\bigcap_{s=0}^{N} \langle x - s\rangle = \langle \prod_{s=0}^{N} (x - s)\rangle$: since $\langle x - s\rangle$ is the set of all multiples of $x - s$, $\bigcap_{s=0}^{N} \langle x - s\rangle$ is the set of common multiples of $x - s$ for $0 \le s \le N$; $\prod_{s=0}^{N} (x - s)$ is the least common multiple of $x - s$ for $0 \le s \le N$, and so it is a generator of $\bigcap_{s=0}^{N} \langle x - s\rangle$.

Now, if s ranges in $\mathbb{N}$, then $\prod_{s\in\mathbb{N}} (x - s)$ is not a polynomial anymore, and so it cannot be in the intersection. Thus $\bigcap_{s\in\mathbb{N}} I_0 \circ \boldsymbol{f_1}^{-s}(x, y) = \langle y\rangle$. Analogously, $\bigcap_{s\in\mathbb{N}} I_0 \circ \boldsymbol{f_2}^{-s}(x, y) = \langle x\rangle$. Finally, $I_1 = \langle y\rangle \cap \langle x\rangle = \langle xy\rangle$ as xy is the least common multiple of x and y. Fig. 1 shows the corresponding variety, together with the initial point and its successive images by $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$. Notice that the dimensions of both irreducible components of $\mathbf{V}(I_1)$, the two coordinate axes, are greater than the dimension of $\mathbf{V}(I_0)$, the origin.

Let us apply another iteration of the Invariant Generation Procedure. Using a similar argument as above, $\bigcap_{s\in\mathbb{N}} I_1 \circ \boldsymbol{f_1}^{-s}(x, y) = \bigcap_{s\in\mathbb{N}} I_1 \circ \boldsymbol{f_2}^{-s}(x, y) = \{0\}$. Thus $I_2 = \{0\}$. It is clear that in the following iteration, the procedure terminates yielding only the trivial invariant, which is the polynomial equation $0 = 0$. Again, notice that the dimension of $\mathbf{V}(I_2) = \mathbb{R}^2$ is greater than the dimension of $\mathbf{V}(I_1)$.

In this example, the dimension of (the variety of) the computed ideal increases at each step until termination. We show below that in general, at each step either the invariant polynomial ideal has been computed or the minimum dimension of the non-invariant irreducible components of the variety increases.

6.1. *Proof of termination*

The key concept in the proof of termination is that of *dimension* of a variety (Becker and Weispfenning, 1993):

Definition 6. Given a variety $V \neq \emptyset$, its *(Krull) dimension* is

$$\dim V = \max\{d \mid V = V_0 \supset V_1 \supset \cdots \supset V_d, \text{ with the } V_i \text{ irreducible varieties}\}.$$

Important properties are that the dimension is monotone, i.e., $V \subseteq W$ implies $\dim V \leq \dim W$, and that $\dim \mathbb{R}^m = m$.

We also recall the following basic theorem from algebraic geometry (see Cox et al. (1996) for example):

Theorem 7. *Any variety V can be uniquely expressed as a finite union of irreducible varieties V_i with $V_i \not\subseteq V_j$ for $i \neq j$ (i.e., irredundant varieties).*

The varieties V_i appearing in this unique decomposition are called the *irreducible components* of V.

In order to show termination of the Invariant Generation Procedure, we also need the following auxiliary results:

Theorem 8. *In the Invariant Generation Procedure, $\forall N \in \mathbb{N}\ I_N = \mathbf{IV}(I_N)$.*

Proof. See at the end of Appendix A.1. $\square$

Theorem 9. *Let $J \subseteq \mathbb{R}[\boldsymbol{x}]$ be a prime ideal of variety and $\boldsymbol{g} \in \mathbb{R}[\boldsymbol{x}]^m$ be a solvable mapping with positive eigenvalues. Then $\bigcap_{s \in \mathbb{N}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$ is also a prime ideal. Moreover, if $\boldsymbol{g}(\mathbf{V}(J)) \not\subseteq \mathbf{V}(J)$, then $\dim \mathbf{V}(\bigcap_{s \in \mathbb{N}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})) > \dim \mathbf{V}(J)$.*

Proof. Let us denote the ideal $\bigcap_{s \in \mathbb{N}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$ by I. The ideal I is prime by Theorem 29 in Appendix A.2 and Theorem 31 in Appendix A.3, respectively. For the second claim, let $d = \dim \mathbf{V}(J)$ and $V_0 = \mathbf{V}(J) \supset V_1 \supset \cdots \supset V_d$ be a maximal chain of irreducible varieties. Since I is prime, $\mathbf{V}(I)$ is irreducible. Moreover, $\mathbf{V}(I) \supseteq \bigcup_{s \in \mathbb{N}} g^s(\mathbf{V}(J)) \supset \mathbf{V}(J)$, as $\boldsymbol{g}(\mathbf{V}(J)) \not\subseteq \mathbf{V}(J)$ by hypothesis. So, $\mathbf{V}(I) \supset \mathbf{V}(J) \supset V_1 \supset \cdots \supset V_d$ is a chain of irreducible varieties, and therefore $\dim \mathbf{V}(I) \geq d + 1 > d = \dim \mathbf{V}(J)$. $\square$

Lemma 10. *If $J, K \subseteq \mathbb{R}[\boldsymbol{x}]$ are ideals and $g \in \mathbb{R}[\boldsymbol{x}]^m$ is an invertible polynomial mapping, then $(J \cap K) \circ \boldsymbol{g}(\boldsymbol{x}) = (J \circ \boldsymbol{g}(\boldsymbol{x})) \cap (K \circ \boldsymbol{g}(\boldsymbol{x}))$.*

Proof. Let us see $\subseteq$. Let $p \in (J \cap K) \circ \boldsymbol{g}(\boldsymbol{x})$. Then there exists $q \in J \cap K$ such that $p = q \circ \boldsymbol{g}$. Since $q \in J$, $p \in J \circ \boldsymbol{g}(\boldsymbol{x})$. And as $q \in K$, $p \in K \circ \boldsymbol{g}(\boldsymbol{x})$. So $p \in (J \circ \boldsymbol{g}(\boldsymbol{x})) \cap (K \circ \boldsymbol{g}(\boldsymbol{x}))$.

Now let us see $\supseteq$. Let $p \in (J \circ \boldsymbol{g}(\boldsymbol{x})) \cap (K \circ \boldsymbol{g}(\boldsymbol{x}))$. Then there exist $q \in J$ such that $p = q \circ \boldsymbol{g}$ and $q' \in K$ such that $p = q' \circ \boldsymbol{g}$. Thus $q = q \circ \boldsymbol{g} \circ \boldsymbol{g}^{-1} = p \circ \boldsymbol{g}^{-1} = q' \circ \boldsymbol{g} \circ \boldsymbol{g}^{-1} = q'$. So $q = q' \in J \cap K$ and $p \in (J \cap K) \circ \boldsymbol{g}(\boldsymbol{x})$. $\square$

Finally we give the proof of termination of the Invariant Generation Procedure. The main idea is as follows. We first show that $\mathbf{V}(I_{N+1})$ can be decomposed as the union of: (i) the irreducible components of $\mathbf{V}(I_N)$ that are invariant, in the sense that they are preserved by all assignment mappings; and (ii), other irreducible varieties related to the non-invariant irreducible components of $\mathbf{V}(I_N)$. Moreover, the minimum dimension of the varieties in (ii) is strictly greater than the minimum dimension of the non-invariant irreducible components of $\mathbf{V}(I_N)$. The key observation is that, if I_{N+1} is not the invariant polynomial ideal, and thus there are irreducible components of $\mathbf{V}(I_{N+1})$ that are not invariant, then these non-invariant components must appear in (ii). Therefore, the minimum dimension of the non-invariant irreducible components has increased strictly. However, this dimension cannot increase indefinitely: since there are m variables $x_1, \ldots, x_m$, we get the final bound $m + 1$.

Theorem 11. *The Invariant Generation Procedure terminates in at most $m + 1$ iterations.*

Proof. Let us fix $N \in \mathbb{N}$. We denote by $\mathrm{Irr}(\mathbf{V}(I_N)) = \{V_1, \ldots, V_k\}$ the irreducible components of $\mathbf{V}(I_N)$. We define $J_j := \mathbf{I}(V_j)$. Then the J_j are prime ideals, $\mathbf{IV}(J_j) = \mathbf{IVI}(V_j) = \mathbf{I}(V_j) = J_j$, and by Theorem 8,

$$I_N = \mathbf{IV}(I_N) = \mathbf{I}\left(\bigcup_{j=1}^{k} V_j\right) = \bigcap_{j=1}^{k} \mathbf{I}(V_j) = \bigcap_{j=1}^{k} J_j.$$

Then, using the above equation and Lemma 10,

$$\begin{aligned} I_{N+1} &= \bigcap_{i=1}^{n} \bigcap_{s\in\mathbb{N}} I_N \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x}) \\ &= \bigcap_{i=1}^{n} \bigcap_{s\in\mathbb{N}} \left(\bigcap_{j=1}^{k} J_j\right) \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x}) = \bigcap_{j=1}^{k} \bigcap_{i=1}^{n} \bigcap_{s\in\mathbb{N}} J_j \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x}), \end{aligned} \qquad (1)$$

and

$$\mathbf{V}(I_{N+1}) = \mathbf{V}\left(\bigcap_{j=1}^{k} \bigcap_{i=1}^{n} \bigcap_{s\in\mathbb{N}} J_j \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})\right) = \bigcup_{j=1}^{k} \bigcup_{i=1}^{n} \mathbf{V}\left(\bigcap_{s\in\mathbb{N}} J_j \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})\right).$$

For $1 \le i \le n$ and $1 \le j \le k$ such that $\boldsymbol{f_i}(\mathbf{V}(J_j)) \subseteq \mathbf{V}(J_j)$, by induction $\forall s \in \mathbb{N}$ $\boldsymbol{f_i}^{s}(\mathbf{V}(J_j)) \subseteq \mathbf{V}(J_j)$; and therefore $\forall s \in \mathbb{N}$,

$$J_j = \mathbf{IV}(J_j) \subseteq \mathbf{I}(\boldsymbol{f_i}^{s}(\mathbf{V}(J_j))) = \mathbf{IV}(J_j \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})) = J_j \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x}).$$

As $J_j = J_j \circ \boldsymbol{f_i}^{0}(\boldsymbol{x})$, $J_j = \cap_{s\in\mathbb{N}} J_j \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})$ and $\mathbf{V}(J_j) = \mathbf{V}(\cap_{s\in\mathbb{N}} J_j \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x}))$. Let us write $\mathrm{Inv} = \{(i, j) \mid \boldsymbol{f_i}(\mathbf{V}(J_j)) \subseteq \mathbf{V}(J_j)\}$. Now we can decompose $\mathbf{V}(I_{N+1})$ as follows:

$$\begin{aligned} \mathbf{V}(I_{N+1}) &= \bigcup_{j=1}^{k} \bigcup_{i=1}^{n} \mathbf{V}\left(\bigcap_{s\in\mathbb{N}} J_j \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})\right) \\ &= \left(\bigcup_{(i,j)\in\mathrm{Inv}} \mathbf{V}\left(\bigcap_{s\in\mathbb{N}} J_j \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})\right)\right) \cup \left(\bigcup_{(i,j)\notin\mathrm{Inv}} \mathbf{V}\left(\bigcap_{s\in\mathbb{N}} J_j \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})\right)\right) \\ &= \left(\bigcup_{(i,j)\in\mathrm{Inv}} \mathbf{V}(J_j)\right) \cup \left(\bigcup_{(i,j)\notin\mathrm{Inv}} \mathbf{V}\left(\bigcap_{s\in\mathbb{N}} J_j \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})\right)\right) \\ &= \left(\bigcup_{\substack{j=1 \\ \exists i \mid (i,j)\in\mathrm{Inv}}}^{k} \mathbf{V}(J_j)\right) \cup \left(\bigcup_{(i,j)\notin\mathrm{Inv}} \mathbf{V}\left(\bigcap_{s\in\mathbb{N}} J_j \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})\right)\right) \\ &= \left(\bigcup_{\substack{j=1 \\ \forall i \mid (i,j)\in\mathrm{Inv}}}^{k} \mathbf{V}(J_j)\right) \cup \left(\bigcup_{(i,j)\notin\mathrm{Inv}} \mathbf{V}\left(\bigcap_{s\in\mathbb{N}} J_j \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})\right)\right). \end{aligned}$$

The last equality follows from the fact that if there exist i, i' such that $(i, j) \in \mathrm{Inv}$ and $(i', j) \notin \mathrm{Inv}$, then $\mathbf{V}(J_j) \subset \mathbf{V}(\bigcap_{s\in\mathbb{N}} J_j \circ \boldsymbol{f_{i'}}^{-s}(\boldsymbol{x}))$; so $\mathbf{V}(J_j)$ is already taken into account on the right-hand side of the union.

By Theorem 9, all the varieties in this decomposition are irreducible. So the varieties in the (unique irredundant) irreducible decomposition of $\mathbf{V}(I_{N+1})$ must appear in the above union.

For each ideal I_N computed by the Invariant Generation Procedure we distinguish two cases:

(1) $\forall i, j$ such that $1 \leq i \leq n,\ 1 \leq j \leq k$ we have $(i, j) \in$ Inv, which implies $J_j = \bigcap_{s\in\mathbb{N}} J_j \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})$. Then by Eq. (1), $I_{N+1} = I_N$ and so $I_N = I_\infty$.
(2) $\exists i, j$ such that $1 \leq i \leq n,\ 1 \leq j \leq k$ and $(i, j) \notin$ Inv. Then we can define

$$\Delta(I_N) = \min\{\dim V \mid V \in \mathrm{Irr}(\mathbf{V}(I_N)) \text{ and } \exists i \text{ such that } \boldsymbol{f_i}(V) \not\subseteq V\},$$

where $\mathrm{Irr}(\mathbf{V}(I_N))$ is the set of irreducible components of $\mathbf{V}(I_N)$.

Further, if $I_{N+1} \neq I_\infty$, I_{N+1} must satisfy Case (2). Then

$$\begin{aligned}\Delta(I_{N+1}) &= \min\{\dim V \mid V \in \mathrm{Irr}(\mathbf{V}(I_{N+1})) \text{ and } \exists i \text{ such that } \boldsymbol{f_i}(V) \not\subseteq V\}\\ &\geq \min\left\{\dim \mathbf{V}\left(\bigcap_{s\in\mathbb{N}} J_j \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})\right) \mid (i, j) \notin \mathrm{Inv}\right\}\\ &> \min\{\dim \mathbf{V}(J_j) \mid \exists i \text{ such that } (i, j) \notin \mathrm{Inv}\} = \Delta(I_N),\end{aligned}$$

as by Theorem 9, $\boldsymbol{f_i}(\mathbf{V}(J_j)) \not\subseteq \mathbf{V}(J_j)$ implies $\dim \mathbf{V}(\bigcap_{s\in\mathbb{N}} J_j \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})) > \dim \mathbf{V}(J_j)$.

Finally, let us assume that the procedure takes more than $m + 1$ iterations to terminate, and we will get a contradiction. In this case we have that $\forall N : 1 \leq N \leq m$, $I_N \neq I_\infty$. As we have seen above, this implies Case (2), in particular that $\Delta(I_N) > \Delta(I_{N-1})$. Since $\Delta(I_0) \geq 0$, by induction $\Delta(I_m) \geq m$. So $I_m = \mathbb{R}[\boldsymbol{x}]$ and $\mathbf{V}(I_m) = \mathbb{R}^m$. But then it is impossible that $\exists i : 1 \leq i \leq n$ such that $\boldsymbol{f_i}(\mathbb{R}^m) \not\subseteq \mathbb{R}^m$. □

Thus, given a loop with m variables and n non-deterministic assignments which are solvable mappings with positive rational eigenvalues, the Invariant Generation Procedure takes at most $m + 1$ iterations to terminate. Nevertheless, the computational complexity of each of these iterations may be proportional to the number of assignments n and doubly exponential in the number of variables m, due to the application of the Gröbner basis algorithm. Other related approaches using Gröbner bases as well, such as Sankaranarayanan et al. (2004), also have this severe theoretical complexity bound. Similarly, the approach proposed in Kapur (2003), which relies on real quantifier elimination, also has a worst-case complexity which is doubly exponential in the number of variables m. Moreover, the method presented in Müller-Olm and Seidl (2004a) does not allow upper complexity bounds, as the proof of termination of the proposed algorithm depends on Hilbert's basis theorem.

On the other hand, the methods based on linear algebra, like Müller-Olm and Seidl (2004b) and Colón (2004), have complexity which is polynomial in the number of variables m if the degree is fixed a priori; however, they are exponential once the degree is left as a parameter.

6.2. *Commuting assignments*

In the case where a loop has many variables whereas the number of assignments in the body of the loop is small, we are able to derive a better upper bound on the number of iterations until termination provided the assignments are commuting. This condition will be satisfied, for example, if different assignments change different subsets of variables.

More specifically, in this section we prove that if assignment mappings commute, i.e., $\boldsymbol{f_i} \circ \boldsymbol{f_j}(\boldsymbol{x}) = \boldsymbol{f_j} \circ \boldsymbol{f_i}(\boldsymbol{x})$ for $1 \leq i, j \leq n$, the procedure terminates in at most $n + 1$ iterations, where n is the number of non-deterministic assignments in the body of the loop (in other words, the number of *branches* control flow may take). Notice that, unlike in the proof of termination above, we do not require any condition on the positivity of the eigenvalues of assignment mappings.

We first prove a more general fact, namely, that at the N-th iteration of the Invariant Generation Procedure, the effect of all possible compositions of assignments with $\leq N-1$ alternations has been considered. Using this general result we show that, if the assignment mappings commute, then an arbitrary execution path has the same effect as the first assignment mapping being executed first followed by the second assignment followed by the third assignment, etc. I.e., the order in which assignment mappings are executed does not matter.

Given a string $\sigma \in [n]^*$ we define $\nu(\sigma)$, the number of alternations of σ as:

- $\nu(\lambda) = -1$ (λ is the empty string),
- $\nu(i) = 0$,
- $\nu(i.j.\sigma) = \nu(j.\sigma)$ if $i = j$,
- $\nu(i.j.\sigma) = 1 + \nu(j.\sigma)$ if $i \neq j$,

$(1 \leq i, j \leq n)$.

Lemma 12. $\forall N \in \mathbb{N}$,

$$I_N = \bigcap_{\nu(\sigma)\leq N-1} I_0 \circ ((\boldsymbol{f})^{\sigma})^{-1}(\boldsymbol{x}).$$

Proof. Let us prove it by induction over N. If $N = 0$, then the only string σ such that $\nu(\sigma) \leq -1$ is $\sigma = \lambda$, the empty string. Since $I_0 \circ ((\boldsymbol{f})^{\lambda})^{-1}(\boldsymbol{x}) = I_0$, our claim holds.

Now let us assume that $N > 0$. By definition of I_N,

$$I_N = \bigcap_{i=1}^{n} \bigcap_{s\in\mathbb{N}} I_{N-1} \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x}).$$

Applying the induction hypothesis and using Lemma 10,

$$\begin{aligned}
I_N &= \bigcap_{i=1}^{n} \bigcap_{s\in\mathbb{N}} \left(\bigcap_{\nu(\sigma)\leq N-2} I_0 \circ ((\boldsymbol{f})^{\sigma})^{-1} \right) \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x}) \\
&= \bigcap_{i=1}^{n} \bigcap_{s\in\mathbb{N}} \bigcap_{\nu(\sigma)\leq N-2} (I_0 \circ ((\boldsymbol{f})^{\sigma})^{-1}) \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x}) \\
&= \bigcap_{i=1}^{n} \bigcap_{s\in\mathbb{N}} \bigcap_{\nu(\sigma)\leq N-2} I_0 \circ (((\boldsymbol{f})^{\sigma})^{-1} \circ \boldsymbol{f_i}^{-s})(\boldsymbol{x}) \\
&= \bigcap_{i=1}^{n} \bigcap_{s\in\mathbb{N}} \bigcap_{\nu(\sigma)\leq N-2} I_0 \circ (\boldsymbol{f_i}^{s} \circ (\boldsymbol{f})^{\sigma})^{-1}(\boldsymbol{x}) \\
&= \bigcap_{i=1}^{n} \bigcap_{s\in\mathbb{N}} \bigcap_{\nu(\sigma)\leq N-2} I_0 \circ ((\boldsymbol{f})^{\sigma.\overbrace{i.\cdots.i}^{s \text{ times}}})^{-1}(\boldsymbol{x}) = \bigcap_{\nu(\sigma)\leq N-1} I_0 \circ ((\boldsymbol{f})^{\sigma})^{-1}(\boldsymbol{x}). \quad \square
\end{aligned}$$

Theorem 13. *If the assignment mappings $\boldsymbol{f_i}$ commute, i.e., $\forall i, j : 1 \leq i, j \leq n$ $\boldsymbol{f_i} \circ \boldsymbol{f_j}(\boldsymbol{x}) = \boldsymbol{f_j} \circ \boldsymbol{f_i}(\boldsymbol{x})$, then the Invariant Generation Procedure terminates in at most $n+1$ iterations.*

Proof. In order to prove that the Invariant Generation Procedure terminates in at most $n+1$ iterations, we have to show that $I_{n+1} = I_n$. Now, for any string σ (in particular, if $\nu(\sigma) \leq n$), we can build a string τ of the form $1^{k_1}.2^{k_2}.\cdots.n^{k_n}$ such that $\nu(\tau) \leq n-1$ and $(\boldsymbol{f})^\sigma = (\boldsymbol{f})^\tau$ by rearranging the mappings $\boldsymbol{f_i}$, which by hypothesis commute. Then, by Lemma 12

$$I_{n+1} = \bigcap_{\nu(\sigma)\leq n} I_0 \circ ((\boldsymbol{f})^\sigma)^{-1}(\boldsymbol{x}) = \bigcap_{\nu(\tau)\leq n-1} I_0 \circ ((\boldsymbol{f})^\tau)^{-1}(\boldsymbol{x}) = I_n.$$

So the procedure terminates in at most $n+1$ iterations. □

7. Approximating with Gröbner bases

In this section, we show how the Invariant Generation Procedure can be implemented with Gröbner bases and elimination theory. We also prove that the approximations performed are precise enough so as to guarantee that we do not lose completeness.

The Invariant Generation Procedure cannot be directly implemented because of

$$I := \bigcap_{s\in\mathbb{N}} \bigcap_{i=1}^{n} I \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x}),$$

since infinite intersection of ideals cannot be effectively computed.

By considering s as a new variable, if a general expression for each $\boldsymbol{f_i}^{-s}$ for $1 \leq i \leq n$ can be computed as a polynomial in the variables $\boldsymbol{x}, s$, then s can be eliminated. Assuming that $\boldsymbol{f_i}^{-s}(\boldsymbol{x}) \in \mathbb{Q}[s, \boldsymbol{x}]^m$ and given a basis of the ideal $I \subseteq \mathbb{Q}[\boldsymbol{x}]$, we get a basis of the ideal $I \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x}) \subseteq \mathbb{Q}[s, \boldsymbol{x}]$ by substituting the variables $\boldsymbol{x}$ by $\boldsymbol{f_i}^{-s}(\boldsymbol{x})$ in the polynomials in the basis of I. Then Gröbner bases can be used to compute the finite intersection $\bigcap_{i=1}^n I \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})$. From the ideal thus obtained, we can compute an elimination ideal eliminating s, again using a Gröbner basis algorithm with an elimination term ordering in which s is bigger than all other variables $\boldsymbol{x}$. In other words, we eliminate s from $\bigcap_{i=1}^n I \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})$.

For instance, in the example from Section 4, there is one single assignment mapping $\boldsymbol{f}(a, b, c) = (a+1, b+c+2, c+2)$. By linear algebra, we compute $\boldsymbol{f}^s(a, b, c) = (a + s, b + sc + s + s^2, c + 2s)$; the inverse $\boldsymbol{f}^{-s}(a, b, c) = (a - s, b - sc - s + s^2, c - 2s)$ is obtained by substituting s by $-s$. Since before executing the loop $a = 0$, $b = c = 1$, we have $I_0 = \langle a, b-1, c-1\rangle$. Then $I_0 \circ \boldsymbol{f}^{-s}(a, b, c) = \langle a - s, b - sc - s + s^2 - 1, c - 2s - 1\rangle$, which after elimination of the variable s yields $\langle c - 2a - 1, b - (a+1)^2\rangle$. As $c - 2a - 1$ and $b - (a+1)^2$ are invariant polynomials, the procedure has reached a fixpoint and terminates.

Nevertheless, there is a problem with this approach: the hypothesis $\boldsymbol{f_i}^{-s}(\boldsymbol{x}) \in \mathbb{Q}[s, \boldsymbol{x}]$ does not necessarily hold in general. Consider the example from Section 3; exponential terms might appear:

$$\boldsymbol{f_1}(a, b, p, q) = (a/2, b/2, 4p, q),$$
$$\boldsymbol{f_1}^{-s}(a, b, p, q) = (2^s a, 2^s b, (1/4)^s p, q).$$

Notice that $\boldsymbol{f_1}^{-s}$ is not a polynomial mapping in s, a, b, p, q. However, if new auxiliary variables are introduced to replace 2^s, $(1/2)^s$, say u, v, respectively, then

$$\boldsymbol{f_1}^{-s}(a, b, p, q) = (ua, ub, v^2 p, q),$$

subject to the polynomial relation $uv = 1$. In this case, the eigenvalues of $\boldsymbol{f_1}$ are $\{1/2, 4, 1\}$, which yield the exponential terms 2^s and $(1/4)^s$ in $\boldsymbol{f_1}^{-s}$; the variables u and v are introduced to substitute for these exponentials so that $\boldsymbol{f_1}^{-s}$ can be represented as a polynomial mapping. In general, the following result enables to express powers of solvable mappings with positive rational eigenvalues as polynomial mappings using auxiliary variables:

Theorem 14. *Let $\boldsymbol{g} \in \mathbb{Q}[\boldsymbol{x}]^m$ be a solvable mapping with positive rational eigenvalues. Then $\forall j : 1 \leq j \leq m$, $\forall s \in \mathbb{Z}$, $g_j^s(\boldsymbol{x})$, the j-th component of $\boldsymbol{g}^s(\boldsymbol{x})$ (where negative exponents mean powers of the inverse of $\boldsymbol{g}$), can be expressed as*

$$g_j^s(\boldsymbol{x}) = \sum_{l=1}^{r_j} P_{jl}(s, \boldsymbol{x})(\gamma_{jl})^s \ ,$$

where for $1 \leq j \leq m$, there exists $r_j \in \mathbb{N}$ such that for $1 \leq l \leq r_j$, $P_{jl} \in \mathbb{Q}[s, \boldsymbol{x}]$ and $\gamma_{jl} \in \mathbb{Q}^+$. Moreover, each γ_{jl} is a product of eigenvalues of $\boldsymbol{g}$.

For the proof, see Appendix A.2. To represent $\boldsymbol{g}^s$ as a polynomial mapping (or equivalently $\boldsymbol{g}^{-s}$, by substitution of s by $-s$), auxiliary variables are introduced to substitute for exponential terms (e.g., u, v for 2^s, $(1/2)^s$, respectively, in the example); these variables are eliminated by means of a suitable elimination ordering employing the polynomial relations between them (e.g., $uv - 1$ in the example).

Let us see how auxiliary variables can be employed to substitute for exponential terms in general. For any $\gamma \in \mathbb{Q}^+$, there exists a unique prime decomposition of the form $\gamma = \prod_{i=1}^k \lambda_i^{\rho_i}$, where the λ_i are primes and $\rho_i \in \mathbb{Z}$ for $1 \leq i \leq k$. Therefore we can compute a "base" $\{\lambda_1, \ldots, \lambda_k\} \subset \mathbb{N}$ of prime numbers such that for any eigenvalue γ, $\gamma = \prod_{i=1}^k \lambda_i^{\rho_i}$ for certain $\rho_i \in \mathbb{Z}$. The powers γ^s can then be expressed in terms of new variables u_i and v_i that are introduced to replace λ_i^s and λ_i^{-s}, respectively, for each λ_i:

$$\gamma^s = \prod_{i=1}^k \lambda_i^{s\rho_i} = \prod_{i=1}^k \begin{cases} u_i^{\rho_i} & \text{if } \rho_i > 0 \\ v_i^{-\rho_i} & \text{if } \rho_i < 0. \end{cases}$$

By Theorem 14, for $1 \leq i \leq n$ there exists a polynomial mapping $\boldsymbol{F_i} = \boldsymbol{F_i}(s, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{x}) : \mathbb{Q}^{1+2k+m} \to \mathbb{Q}^m$ such that $\forall s_0 \in \mathbb{N}$,

$$\boldsymbol{f_i}^{s_0}(\boldsymbol{x}) = \boldsymbol{F_i}(s_0, \boldsymbol{\lambda}^{s_0}, \boldsymbol{\lambda}^{-s_0}, \boldsymbol{x}) \quad \text{and} \quad \boldsymbol{f_i}^{-s_0}(\boldsymbol{x}) = \boldsymbol{F_i}(-s_0, \boldsymbol{\lambda}^{-s_0}, \boldsymbol{\lambda}^{s_0}, \boldsymbol{x}),$$

where $\boldsymbol{u} = (u_1, \ldots, u_k)$, $\boldsymbol{v} = (v_1, \ldots, v_k)$ and $\boldsymbol{\lambda}^{s_0} = (\lambda_1^{s_0}, \ldots, \lambda_k^{s_0})$.

In our previous example, we have taken $\{2\}$ as a base of prime numbers, and introduced variables u, v to represent 2^s and $(1/2)^s$ respectively. Moreover, $uv - 1$ is required to eliminate these new variables. In general, it is necessary to consider all polynomial relations between s, the λ_i^s and the λ_i^{-s} in order to eliminate the auxiliary variables. Proposition 32 in Appendix B shows that $L = \{u_1 v_1 - 1, \ldots, u_k v_k - 1\}$ characterises the set of all these polynomial relations.

Finally, it is possible to weaken the restriction of positiveness on the eigenvalues of assignment mappings as follows. If there is any negative eigenvalue, it is only necessary to introduce a new variable t to replace $(-1)^s$ in the $\boldsymbol{f_i}^{-s}(\boldsymbol{x})$; the equality $\left((-1)^{s_0}\right)^2 = 1 \ \forall s_0 \in \mathbb{N}$ yields the polynomial $t^2 - 1$, which plays a similar role as the polynomials $u_i v_i - 1$ above. Moreover, if any of the assignment mappings is not invertible, i.e., 0 is an eigenvalue, then the algorithm can be modified so that it can be applied also in this case, basically by performing, instead of the assignment

$$I := \bigcap_{i=1}^n \bigcap_{s \in \mathbb{N}} I \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x}),$$

the assignment

$$I := \bigcap_{i=1}^{n} \bigcap_{s \in \mathbb{N}} \Big(\langle I(\boldsymbol{x}'),\ \boldsymbol{x} - \boldsymbol{f_i}^{s}(\boldsymbol{x}') \rangle \cap \mathbb{Q}[\boldsymbol{x}] \Big),$$

where the projection is computed by elimination of variables. For the sake of simplicity, in this paper we have focused on positive rational eigenvalues.

7.1. Implementation

In the algorithm below, ideals are represented by their Gröbner bases using some term ordering. Checking that the assignment mappings $\boldsymbol{f_i}$ are solvable with positive rational eigenvalues is done using linear algebra. Then the powers $\boldsymbol{f_i}^{-s}$ are computed and expressed as polynomial mappings denoted by $\boldsymbol{F_i}$, possibly employing the parameter s and additional auxiliary variables $\boldsymbol{u}, \boldsymbol{v}$ introduced to replace exponential terms; relations among auxiliary variables are specified using $L = \{u_1 v_1 - 1, \ldots, u_k v_k - 1\}$.

```
Input:
    • The solvable mappings with positive rational eigenvalues $\boldsymbol{f_1}, \ldots, \boldsymbol{f_n}$
    of the assignments.
    • A set $S_0$ of polynomials satisfied by the initial values such that
    $I_0 = \mathbf{IV}(I_0)$, where $I_0 = \langle S_0 \rangle$.
Output:
    • A finite basis for the invariant polynomial ideal $I_\infty$.

var $S$, $S'$ : sets of polynomials in $\mathbb{Q}[\boldsymbol{x}]$ end var

1:  $S := \mathtt{GB}(S_0, \succ)$
2:  do
3:      $S' := S$
4:      $S := \mathtt{GB}(\bigcap_{i=1}^{n} \langle L \cup (S \circ \boldsymbol{F_i}(-s, \boldsymbol{v}, \boldsymbol{u}, \boldsymbol{x})) \rangle_{\mathbb{Q}[s,\boldsymbol{u},\boldsymbol{v},\boldsymbol{x}]}, \succ) \cap \mathbb{Q}[\boldsymbol{x}]$
5:  while $S' \neq S$
6:  return $S$
```

Line 4 corresponds to the assignment $I := \bigcap_{s \in \mathbb{N}} \bigcap_{i=1}^{n} I \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})$ of the algorithm in Section 5. The inverse power $\boldsymbol{f_i}^{-s}(\boldsymbol{x})$ is represented by $\boldsymbol{F_i}(-s, \boldsymbol{v}, \boldsymbol{u}, \boldsymbol{x})$, as $\forall s_0 \in \mathbb{N}$ $\boldsymbol{f_i}^{-s_0}(\boldsymbol{x}) = \boldsymbol{F_i}(-s_0, \boldsymbol{\lambda}^{-s_0}, \boldsymbol{\lambda}^{s_0}, \boldsymbol{x})$ by construction of $\boldsymbol{F_i}$. The polynomials L are added to the set of polynomials S to take into account the relationships between the powers of the exponentials. The function GB computes the reduced Gröbner basis of its input ideal, specified as a finite set of polynomials, with respect to a term ordering. The intersection of ideals is performed by using Gröbner bases methods. In particular, the intersection with $\mathbb{Q}[\boldsymbol{x}]$ corresponds to the elimination of the variables $s, \boldsymbol{u}, \boldsymbol{v}$; to that end, the term ordering $\succ$ can be either a block term ordering or a lexicographic term ordering in which $s, \boldsymbol{u}, \boldsymbol{v}$ are the biggest variables. Finally, the equality test on ideals at Line 5 is implemented by comparing the reduced Gröbner bases with respect to $\succ$, as every ideal has a unique reduced Gröbner basis once the ordering is fixed.

The following result ensures that the above implementation is correct and complete:

Theorem 15. *If the Invariant Generation Procedure terminates, the implementation also terminates in at most the same number of iterations with output S such that* $\langle S \rangle_{\mathbb{Q}[\boldsymbol{x}]} = I_\infty$.

See Appendix B. The proof is based on two facts: (i) the inclusion $I_\infty \subseteq \langle S \rangle$ always holds, and in particular, on termination; and (ii) at any iteration, the ideal I computed by the Invariant Generation Procedure includes the ideal $\langle S \rangle$ generated by the polynomials obtained in the implementation. So, if the Invariant Generation Procedure terminates, we have $I = I_\infty \subseteq \langle S \rangle \subseteq I$. Therefore all inclusions are in fact equalities, and the implementation terminates with a set of polynomials generating I_∞ in at most the same number of steps as the Invariant Generation Procedure.

A variation of the above algorithm employing additional heuristics to speed up the computation has been implemented in Maple. The implementation has been successfully used to automatically discover invariants of many non-trivial programs. Some of these are discussed below as well as in Section 8. As the reader will notice, many of these invariants are not easy to deduce by hand.

The following example comes from a program for computing the product of two integer numbers X and Y:

```
$(x, y, z)$:=$(X, Y, 0)$;
while ? do
    $(x, y, z)$:=$(2x, (y-1)/2, x+z)$;    or    $(x, y, z)$:=$(2x, y/2, z)$;
end while
```

We express the powers of the assignments as polynomial mappings as follows:

$$\begin{aligned}
&\boldsymbol{f_1}(x, y, z) = (2x, y/2 - 1/2, x + z),\\
&\boldsymbol{f_2}(x, y, z) = (2x, y/2, z),\\
&\boldsymbol{f_1}^{-s}(x, y, z) = ((1/2)^s x, 2^s y + 2^s - 1, z + ((1/2)^s - 1)x),\\
&\boldsymbol{f_2}^{-s}(x, y, z) = ((1/2)^s x, 2^s y, z),\\
&\boldsymbol{F_1}(-s, v, u, x, y, z) = (vx, uy + u - 1, z + (v - 1)x),\\
&\boldsymbol{F_2}(-s, v, u, x, y, z) = (vx, uy, z),
\end{aligned}$$

where the variables u, v represent 2^s and $(1/2)^s$ respectively. We get the following trace (to illustrate the proof of termination in Section 6, we also give, for each iteration, the dimension of the variety corresponding to the computed ideal):

```
iteration 0 ⟶    $\{x - X, y - Y, z\}$, dimension 2;
iteration 1 ⟶    $\{xz - z - zX, -XY + z + xy, yz + zyX - zXY + z\}$, dimension 3;
iteration 2 ⟶    $\{z + xy - XY\}$, dimension 4;
iteration 3 ⟶    $\{z + xy - XY\}$, dimension 4.
```

Therefore, in only 3 iterations, the algorithm terminates with the invariant polynomial $z + xy - XY$.

Consider now the following loop, which is an abstraction of a program in Knuth (1969) to find a factor of a number N with only addition and subtraction:

```
$(r,\ x,\ y)$:=$(R^2 - N,\ 2R + 1,\ 1)$;
while ? do
    $(r,\ y)$:=$(r - y,\ y + 2)$;    or    $(r,\ x)$:=$(r + x,\ x + 2)$;
end while
```

In this case we just have to add one new variable s:

$$\begin{aligned}
&\boldsymbol{f_1}(r, x, y) = (r - y, x, y + 2),\\
&\boldsymbol{f_2}(r, x, y) = (r + x, x + 2, y),\\
&\boldsymbol{F_1}(-s, r, x, y) = \boldsymbol{f_1}^{-s}(r, x, y) = (r + sy - (s + 1)s, x, y - 2s),\\
&\boldsymbol{F_2}(-s, r, x, y) = \boldsymbol{f_1}^{-s}(r, x, y) = (r - sx + (s + 1)s, x - 2s, y).
\end{aligned}$$

Using $S_0 = \{r - R^2 + N, x - 2R - 1, y - 1\}$, we get the following trace (again we also indicate the dimensions of the varieties):

`iteration 0` $\longrightarrow \{r - R^2 + N, x - 2R - 1, y - 1\}$, dimension 2;
`iteration 1` $\longrightarrow \{xy - 2yR - y - x + 2R + 1, x^2 - y^2 - 4r - 4N - 2x + 2y, y^3 + 4ry - 4yR^2 + 4yN - 3y^2 - 4r + 4R^2 - 4N + 3y - 1\}$, dimension 3;
`iteration 2` $\longrightarrow \{x^2 - y^2 - 4r - 4N - 2x + 2y\}$, dimension 4;
`iteration 3` $\longrightarrow \{x^2 - y^2 - 4r - 4N - 2x + 2y\}$, dimension 4.

So the algorithm terminates in 3 iterations as well yielding the invariant $x^2 - y^2 - 4r - 4N - 2x + 2y$.

As illustrated using the above two examples, the algorithm terminates in iterations fewer than the number of variables plus one. This was proved in Section 6 for solvable assignment mappings with rational positive eigenvalues.

We have yet to find an example for which the above procedure does not terminate. Our experience suggests to us conjecturing that insofar as the effect of assignment mappings has a "polynomial structure", i.e., can be presented using polynomials possibly involving new variables and relations on these variables, the procedure always terminates.

8. Examples

The procedure discussed in Section 7.1 has been implemented in Maple. Below we show some of the loops whose polynomial invariants have been successfully computed using this implementation. Again, to illustrate the ideas presented in the proof of termination in Section 6, we give, for each iteration, the corresponding dimension.

Example 16. The next loop is a version of a program taken from Petter (2004):

```
    (x, y):=(0, 0);
while ? do
        (x, y):=(x + y^5, y + 1);
end while
```

For this example we get the following trace:

`iteration 0` $\longrightarrow \{x, y\}$, dimension 0;
`iteration 1` $\longrightarrow \{-12x + 2y^6 - 6y^5 + 5y^4 - y^2\}$, dimension 1;
`iteration 2` $\longrightarrow \{-12x + 2y^6 - 6y^5 + 5y^4 - y^2\}$, dimension 1.

Finally, the invariant $12x = 2y^6 - 6y^5 + 5y^4 - y^2$ is obtained. Notice that, though the degree of the polynomial is high, the procedure takes just 2 iterations to generate it.

Example 17. The following loop has been extracted from Cousot and Cousot (1977):

```
    $(i, j)$:=$(2, 0)$;
while ? do
    $(i, j)$:=$(i + 4, j)$;   or   $(i, j)$:=$(i + 2, j + 1)$;
end while
```

For this case we get the following ideals:

`iteration 0` $\longrightarrow \{j, i - 2\}$, dimension 0;
`iteration 1` $\longrightarrow \{ji - 2j - 2j^2\}$, dimension 1;
`iteration 2` $\longrightarrow \{0\}$, dimension 2;
`iteration 3` $\longrightarrow \{0\}$, dimension 2.

After 3 iterations, the algorithm stabilises but only a trivial loop invariant is generated. Since our technique is complete, it can be asserted that there are *no* non-trivial invariant polynomial equalities for this loop. This is consistent with the results obtained by Cousot and Halbwachs, who did not find any linear invariant equalities for this example.

Example 18. The next example, taken from Dijkstra (1976), is a version of Euclid's algorithm that computes at the same time the least common multiple and the greatest common divisor of two natural numbers a and b:

```
$(x, y, u, v)$:=$(a, b, b, a)$;
while $x \neq y$ do
    if $x > y$
        $(x, y, u, v)$:=$(x - y, y, u, u + v)$;
    else
        $(x, y, u, v)$:=$(x, y - x, u + v, v)$;
    end if
end while
```

If we apply the Invariant Generation Procedure to the above program (ignoring conditions), we get the invariant $ux + vy - 2ab$ in 4 iterations:

`iteration 0` $\longrightarrow \{x - a, y - b, u - b, v - a\}$, dimension 2;
`iteration 1` $\longrightarrow \{y + u - 2b, x + v - 2a, uv - ua - vb + ba\}$, dimension 3;
`iteration 2` $\longrightarrow$
$\{ux - 2ba + vy, -2xb + xy + uv - 2vb - 2ya - 2ua + 6ba, 2yua - 2yba + vy^2 - u^2v + 2u^2a - 6uba + 2uvb + 4b^2a - 2yvb\}$, dimension 4;
`iteration 3` $\longrightarrow \{ux + vy - 2ba\}$, dimension 5;
`iteration 4` $\longrightarrow \{ux + vy - 2ba\}$, dimension 5.

The invariant $ux + vy - 2ab$ is fundamental in order to prove that on termination $\text{lcm}(a, b) = (u + v)/2$.

Example 19. The following program is yet another version of Euclid's algorithm. It computes the greatest common divisor of two natural numbers together with Bezout's coefficients:

```
$(a, b, p, q, r, s)$:=$(x, y, 1, 0, 0, 1)$;
while $a \neq b$ do
    if $a > b$
```

```
            $(a, b, p, q, r, s)$:=$(a - b, b, p - q, q, r - s, s)$;
        else
            $(a, b, p, q, r, s)$:=$(a, b - a, p, q - p, r, s - r)$;
        end if
    end while
```

`iteration 0`$\longrightarrow$
$\{a - x, b - y, p - 1, q, r, s - 1\}$, dimension 2;
`iteration 1` $\longrightarrow$
$\{s - 1, p - 1, qr, -a + x + ry, br - a + x, qx - b + y, qa - b + y, ba - bx - ay + xy\}$, dimension 3;
`iteration 2` $\longrightarrow$
$\{sp - s - p + 1, qr - p - s + 2, br + x - sa, qx - b + sy, bp - qa - y, xp - a + ry, -sa + sx + sry + a - x - ry, sqa - qa - sb + sy + b - y, s^2ya - sba + xsb - asy - xsy + ba - bx + xy\}$, dimension 4;
`iteration 3` $\longrightarrow$
$\{-sp + 1 + qr, br + x - sa, qx - b + sy, bp - qa - y, xp - a + ry\}$, dimension 5;
`iteration 4` $\longrightarrow$
$\{-sp + 1 + qr, br + x - sa, qx - b + sy, bp - qa - y, xp - a + ry\}$, dimension 5.

In this case, in 4 iterations, the procedure yields the invariant:

$$1 + qr = sp \wedge rb + x = sa \wedge qx + sy = b \wedge \; aq + y = bp \wedge px + ry = a,$$

which can be used to prove that, on termination, (p, r) and (q, s) are Bezout's coefficients for x and y.

As mentioned earlier, the algorithm in Section 7.1 has been implemented in Maple, employing additional heuristics to speed up the computation. For instance, when there are two or more assignments, looking for polynomial invariants for all the branches together from the very beginning requires computing a lot of intersections of ideals at the same time. In order to avoid this, we can first find invariants for one branch; then find invariants for two branches, the previous and another one; and so on, until considering all possible branches.[3] This implementation in Maple has been successfully used to automatically discover invariants of many non-trivial programs. The table in Fig. 2 gives a representative list of the examples attempted so far. There is a row for each program; the columns provide the following information (for those programs which are formed by a sequence of loops of the kind considered here, the data for each loop is provided, except for the timings, which are added up):

- 1st column is the name of the program.
- 2nd column states what the program does.
- 3rd column gives the citation from where the program was picked (the entry $(\star)$ is for the examples developed by the authors[4]).
- 4th column gives the number of variables in the loop.

[3] However, this strategy increases the theoretical number of iterations of the procedure: as we execute the basic algorithm n times, each of which takes at most $m + 1$ iterations, we may need up to $nm + n$ iterations to terminate. Still, this upper bound is far from being reached in the experimental evaluation shown in Fig. 2.

[4] Program `prod4` is the example presented in Section 3.2.

Name	Function	Source	m	n	$\sharp$	d	Ite.	Time	Other
dijkstra	$\sqrt[2]{}$	(Dijkstra, 1976)	4	1-2	2-1	1-2	2-3	1.5	1.4
divbin	division	(Kaldewaij, 1990)	4	1-2	2-1	1-2	2-3	2.1	1.2
freire1	$\sqrt[2]{}$	(Freire, 2002)	3	1	1	2	2	0.7	0.5
freire2	$\sqrt[3]{}$	(Freire, 2002)	4	1	3	2	2	0.7	1.0
cohencu	cube	(Cohen, 1990)	4	1	4	2	2	0.7	1.2
fermat	factor	(Bressoud, 1989)	5	2	1	2	4	0.8	1.1
wensley	division	(Wegbreit, 1974)	5	2	3	2	4	1.1	1.1
euclidex	gcd	($\star$)	8	2	5	2	5	1.4	2.1
lcm	lcm	(Dijkstra, 1976)	6	2	1	2	5	1.0	1.3
prod4	product	($\star$)	6	4	1	3	7	2.1	4.8
knuth	factor	(Knuth, 1969)	8	4	1	3	7	55.4	2.8
petter1	power sum	(Petter, 2004)	2	1	1	2	2	1.0	0.5
petter2	power sum	(Petter, 2004)	2	1	1	3	2	1.1	0.8
petter3	power sum	(Petter, 2004)	2	1	1	4	2	1.3	4.2
petter4	power sum	(Petter, 2004)	2	1	1	5	2	1.3	TO
petter5	power sum	(Petter, 2004)	2	1	1	6	2	1.4	TO

Fig. 2. Table of examples.

- 5th column gives the number of branches in the body of the loop.
- 6th column gives the number of polynomials in the invariant.
- 7th column gives the maximum degree of the polynomials in the invariant.
- 8th column gives the number of times the main loop of the Invariant Generation Procedure is executed.
- 9th column gives the time (in seconds) taken by the implementation of the algorithm in Maple, running on a Pentium 4 with a 3.4 GHz. processor and 2 Gb of memory.
- 10th column gives the time (in seconds) taken by an implementation of the method in Rodríguez-Carbonell and Kapur (2004a) running on the same machine (timeouts are set to 300 s and are represented by TO; the degree bound that has been taken is the maximum degree from the 7th column).

In general, the implementation in Maple of the proposed method works quite fast: it took just over 2 s to analyse all of the examples, except for `knuth`; in this particular case, the algorithm in Rodríguez-Carbonell and Kapur (2004a) is better, while for the rest, either both algorithms perform similarly, or the one presented here is better. More specifically, it can observed that, for the sequence of programs `petter1`, etc., the behaviour of the proposed method is more robust and no timeouts are obtained. One can draw the conclusion that, for programs with a single branch, like `petter1`, etc., the approach developed in this paper tends to work better than the other one, especially when the degree of the invariants is high.

9. Conclusions

The main contributions of this paper are:

(1) We prove that the invariant polynomial ideal of a loop *is computed in at most* $m + 1$ *steps*, where m is the number of program variables.
(2) If assignment mappings commute, i.e., $f_i \circ f_j(x) = f_j \circ f_i(x)$ for $1 \le i, j \le n$, we show that the invariant polynomial ideal *is computed in at most* $n + 1$ *steps*, where n is the number of branches in the loop body.

(3) We explain how the procedure for computing the invariant polynomial ideal can be approximated using Gröbner bases. And moreover, we prove that this approximation is exact, i.e., the algorithm computes the invariant ideal of the loop.
(4) The algorithm has been implemented in Maple and successfully used to compute invariant polynomial equalities for many non-trivial examples. Some of these examples are discussed in the paper.

For future work, we are interested in exploring the proposed research along several directions:

- study more general conditions under which the Invariant Generation Procedure terminates: since we are unaware of any example for which the procedure does not terminate, we conjecture that the requirement of solvable mappings possessing positive rational eigenvalues is unnecessary. We are particularly interested in extending the proof of termination to the cases where the eigenvalues of the assignment mappings may be negative or null.
- enrich the programming model so as to consider nested loops and procedure calls, as well as tests in conditional statements and loops. In particular, the approach presented in this paper could be merged with the method described in Rodríguez-Carbonell and Kapur (2004a), which can handle both nested loops and tests; the former would accelerate the fixpoint computation of the latter while avoiding the application of widening, thus leading to an improvement on the timing and precision of the overall analysis.
- identify other languages to which the ideas here presented apply and which are rich enough to specify properties of data structures such as arrays, records, pointers, etc.
- integrate these and other techniques for mechanically inferring loop invariants, together with theorem proving components, into a tool for program verification.

Acknowledgements

This research was partially supported by an NSF ITR award CCR-0113611, the Prince of Asturias Endowed Chair in Information Science and Technology at the University of New Mexico and an FPU grant from the Spanish Secretaría de Estado de Educación y Universidades, ref. AP2002-3693. The authors would also like to thank A. Chtcherba, G. Godoy, R. Nieuwenhuis, A. Oliveras and the anonymous referees of previous versions of this paper for their comments.

Appendix A. Auxiliary results for the proof of termination

Below we prove in detail all of the auxiliary results required in the proof of termination of the Invariant Generation Procedure.

A.1. $I = \mathbf{IV}(I)$ Is invariant in the invariant generation procedure

The following results are aimed at showing that $\forall N \in \mathbb{N}$, I_N is an ideal of variety; in other words, that $I = \mathbf{IV}(I)$ is invariant in the Invariant Generation Procedure. In order to prove that, we have to show that it holds at the beginning, and that it is preserved at each step of the procedure. To that end, we will see that the property is closed under intersection of ideals and under the mapping between ideals $J \mapsto \bigcap_{s \in \mathbb{N}} J \circ \boldsymbol{f}^{-s}(\boldsymbol{x})$.

The following lemma shows that the property of being an ideal of variety is closed under intersection:

Lemma 20. *If $J, K \subseteq \mathbb{R}[\boldsymbol{x}]$ are ideals of variety, then $J \cap K$ is also an ideal of variety.*

Proof. $\mathbf{IV}(J \cap K) = \mathbf{I}(\mathbf{V}(J) \cup \mathbf{V}(K)) = \mathbf{IV}(J) \cap \mathbf{IV}(K) = J \cap K$. □

The goal of the following four results is to prove that being an ideal of variety is preserved by $J \mapsto \bigcap_{s\in\mathbb{N}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$. First we need the next lemma, which shows the dual relation between the composition $\circ$ and the concept of variety of an ideal.

Lemma 21. *Given an invertible polynomial mapping $\boldsymbol{g} \in \mathbb{R}[\boldsymbol{x}]^m$ and an ideal $J \subseteq \mathbb{R}[\boldsymbol{x}]$, $\mathbf{V}(J \circ \boldsymbol{g}(\boldsymbol{x})) = \boldsymbol{g}^{-1}(\mathbf{V}(J))$.*

Proof. Let us see the $\subseteq$ inclusion. Let $\boldsymbol{\omega} \in \mathbf{V}(J \circ \boldsymbol{g}(\boldsymbol{x}))$, and take any $p \in J$. Then $p(\boldsymbol{g}(\boldsymbol{\omega})) = 0$ since $p \circ \boldsymbol{g} \in J \circ \boldsymbol{g}(\boldsymbol{x})$. So $\boldsymbol{g}(\boldsymbol{\omega}) \in \mathbf{V}(J)$, and thus $\boldsymbol{\omega} \in \boldsymbol{g}^{-1}(\mathbf{V}(J))$.

For the other inclusion, let $\boldsymbol{\omega} \in \boldsymbol{g}^{-1}(\mathbf{V}(J))$. Then $\boldsymbol{g}(\boldsymbol{\omega}) \in \mathbf{V}(J)$. Now let us take any $p \in J \circ \boldsymbol{g}(\boldsymbol{x})$. Then there exists $q \in J$ such that $p = q \circ \boldsymbol{g}$, and $p(\boldsymbol{\omega}) = q(\boldsymbol{g}(\boldsymbol{\omega})) = 0$ since $\boldsymbol{g}(\boldsymbol{\omega}) \in \mathbf{V}(J)$ and $q \in J$. Therefore $\boldsymbol{\omega} \in \mathbf{V}(J \circ \boldsymbol{g}(\boldsymbol{x}))$. □

Now we show that being an ideal of variety is closed under the operator $\circ$:

Lemma 22. *If $J \subseteq \mathbb{R}[\boldsymbol{x}]$ is an ideal of variety and $\boldsymbol{g} \in \mathbb{R}[\boldsymbol{x}]^m$ is an invertible polynomial mapping, then $J \circ \boldsymbol{g}(\boldsymbol{x}) = \mathbf{IV}(J \circ \boldsymbol{g}(\boldsymbol{x}))$.*

Proof. It is enough to show that $J \circ \boldsymbol{g}(\boldsymbol{x}) \supseteq \mathbf{IV}(J \circ \boldsymbol{g}(\boldsymbol{x}))$, since the other inclusion is trivial. Let $p \in \mathbf{IV}(J \circ \boldsymbol{g}(\boldsymbol{x}))$. First, let us show that $p \circ \boldsymbol{g}^{-1}(\boldsymbol{x}) \in \mathbf{IV}(J)$: indeed, given any $\boldsymbol{\omega} \in \mathbf{V}(J)$ we have $p \circ \boldsymbol{g}^{-1}(\boldsymbol{\omega}) = 0$ since $p \in \mathbf{I}(\boldsymbol{g}^{-1}(\mathbf{V}(J)))$ by Lemma 21. But then $p \circ \boldsymbol{g}^{-1}(\boldsymbol{x}) \in \mathbf{IV}(J) = J$, which implies that $p(\boldsymbol{x}) = (p \circ \boldsymbol{g}^{-1}) \circ \boldsymbol{g}(\boldsymbol{x}) \in J \circ \boldsymbol{g}(\boldsymbol{x})$. □

Lemma 23. *If $J \subseteq \mathbb{R}[\boldsymbol{x}]$ is an ideal of variety and $\boldsymbol{g} \in \mathbb{R}[\boldsymbol{x}]^m$ is an invertible polynomial mapping, then*

$$\bigcap_{s\in\mathbb{N}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x}) = \mathbf{I}\left(\bigcup_{s\in\mathbb{N}} \mathbf{V}(J \circ \boldsymbol{g}^{-s}(\boldsymbol{x}))\right).$$

Proof. Let us see the $\supseteq$ inclusion. Let $p \in \mathbf{I}(\bigcup_{s\in\mathbb{N}} \mathbf{V}(J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})))$. Let us assume that $p \notin \bigcap_{s\in\mathbb{N}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$ and we will get a contradiction. Under this hypothesis there exists $s_0 \in \mathbb{N}$ such that $p \notin J \circ \boldsymbol{g}^{-s_0}(\boldsymbol{x})$. Since $p \in \mathbf{I}(\bigcup_{s\in\mathbb{N}} \mathbf{V}(J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})))$, in particular $p \in \mathbf{IV}(J \circ \boldsymbol{g}^{-s_0}(\boldsymbol{x})) = J \circ \boldsymbol{g}^{-s_0}(\boldsymbol{x})$ (by Lemma 22), which is impossible.

Now let us see the other inclusion. Let $p \in \bigcap_{s\in\mathbb{N}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$. Then for any $\boldsymbol{\omega} \in \bigcup_{s\in\mathbb{N}} \mathbf{V}(J \circ \boldsymbol{g}^{-s}(\boldsymbol{x}))$ there exists $s_0 \in \mathbb{N}$ such that $\boldsymbol{\omega} \in \mathbf{V}(J \circ \boldsymbol{g}^{-s_0}(\boldsymbol{x}))$. Since $p \in \bigcap_{s\in\mathbb{N}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x}) \subseteq J \circ \boldsymbol{g}^{-s_0}(\boldsymbol{x})$, $p(\boldsymbol{\omega}) = 0$. So $p \in \mathbf{I}(\bigcup_{s\in\mathbb{N}} \mathbf{V}(J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})))$. □

Finally, the next lemma shows that the property $J = \mathbf{IV}(J)$ is preserved by $J \mapsto \bigcap_{s\in\mathbb{N}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$:

Lemma 24. *If $J \subseteq \mathbb{R}[\boldsymbol{x}]$ is an ideal of variety and $g \in \mathbb{R}[\boldsymbol{x}]^m$ is an invertible polynomial mapping, then*

$$\bigcap_{s\in\mathbb{N}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x}) = \mathbf{IV}\left(\bigcap_{s\in\mathbb{N}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})\right).$$

Proof. By Lemma 23,

$$\mathbf{IV}\left(\bigcap_{s\in\mathbb{N}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})\right) = \mathbf{IVI}\left(\bigcup_{s\in\mathbb{N}} \mathbf{V}(J \circ \boldsymbol{g}^{-s}(\boldsymbol{x}))\right)$$
$$= \mathbf{I}\left(\bigcup_{s\in\mathbb{N}} \mathbf{V}(J \circ \boldsymbol{g}^{-s}(\boldsymbol{x}))\right) = \bigcap_{s\in\mathbb{N}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x}). \quad \square$$

Theorem 8. *In the Invariant Generation Procedure, $\forall N \in \mathbb{N}\ I_N = \mathbf{IV}(I_N)$.*

Proof. Let us prove it by induction over N. For $N = 0$ it is true by construction. Now let us consider the case $N > 0$. By induction hypothesis, $I_{N-1} = \mathbf{IV}(I_{N-1})$. And by Lemma 24, for $1 \le i \le n$:

$$\bigcap_{s\in\mathbb{N}} I_{N-1} \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x}) = \mathbf{IV}\left(\bigcap_{s\in\mathbb{N}} I_{N-1} \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x})\right).$$

Then, by Lemma 20, $I_N = \mathbf{IV}(I_N)$. $\square$

A.2. Powers of solvable mappings

Given a solvable mapping $\boldsymbol{g}$, we can compute its s-th *natural* power, which is the s-fold composition of $\boldsymbol{g}$: $\boldsymbol{g}^s = \boldsymbol{g} \circ \boldsymbol{g} \circ \cdots \circ \boldsymbol{g}$ (s times). These $\boldsymbol{g}^s$ have the structure of sums of products of polynomials and exponentials (Lemma 25 and Proposition 26). In order to define *real* powers of solvable mappings (not necessarily *natural* powers), we use this expression of sums of products of polynomials and exponentials to extend the definition; for the expression to make sense, it is not necessary that s be a natural number, but it may be any real number (note that, however, in the real case we lose the original meaning of s-fold composition[5]).

After defining real powers of solvable mappings this way, it is proved that real powers behave like any reasonable definition of "power" should (Lemma 27 and Proposition 28), i.e., $\boldsymbol{g}^0$ is the identity and $\boldsymbol{g}^{s+t} = \boldsymbol{g}^s \circ \boldsymbol{g}^t$ (just like $2^0 = 1$ and $2^{s+t} = 2^s \cdot 2^t$). Theorem 14 is a corollary of Lemma 27 and Proposition 28 that is required in Section 7.

Finally it is shown in Theorem 29 that, if J is an ideal of variety and $\boldsymbol{g}$ is a solvable mapping with positive eigenvalues, then

$$\bigcap_{s\in\mathbb{N}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x}) = \bigcap_{s\in\mathbb{R}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x}),$$

which is required in the proof of termination.

In all this subsection, let $\mathbb{K}$ be either $\mathbb{Q}$ or $\mathbb{R}$. First we need the following lemma, which describes the solutions of the recurrences that arise when computing natural powers of solvable mappings. We will extensively use the theory of generating functions, linear recurrences with constant coefficients and rational functions, and we are not giving all the details; the interested reader may consult, e.g., (Stanley, 1997, p. 200, Section 4) to fully understand the proof.

[5] This is similar to combinatorial numbers: By definition, $C(n, k) = n \cdot (n-1) \cdots (n-k+1)/k!$. Though for the combinatorial meaning it is required $n, k \in \mathbb{N}$, using the expression $n \cdot (n-1) \cdots (n-k+1)/k!$ we can define $C(n, k)$ for $n \in \mathbb{R}$, e.g., $C(3/2, 2) = (3/2 \cdot 1/2)/2 = 3/8$.

Lemma 25. *Consider a recurrence*

$$\begin{pmatrix} x_1^{(s+1)} \\ \vdots \\ x_h^{(s+1)} \end{pmatrix} = M \begin{pmatrix} x_1^{(s)} \\ \vdots \\ x_h^{(s)} \end{pmatrix} + \boldsymbol{Q}(s, \boldsymbol{y}),$$

where $M \in \mathbb{K}^{h \times h}$ is a matrix with eigenvalues in $\mathbb{K}$ and $\boldsymbol{Q}$ is a vector of h functions of the form $\sum_{l=1}^{r} Q_l(s, \boldsymbol{y})\mu_l^s$, where for $1 \le l \le r$, $Q_l \in \mathbb{K}[s, \boldsymbol{y}]$ and $\mu_l \in \mathbb{K}$ (the $\boldsymbol{y}$ variables represent parameters). Then the solutions of the recurrence have the form:

$$x_j^{(s)} = \sum_{l=1}^{r_j} P_{jl}(s, \boldsymbol{y}, \boldsymbol{x}^{(0)})(\gamma_{jl})^s,$$

where for $1 \le j \le h$, there exists $r_j \in \mathbb{N}$ such that for $1 \le l \le r_j$, $P_{jl} \in \mathbb{K}[s, \boldsymbol{y}, \boldsymbol{x}^{(0)}]$ and the $\gamma_{jl} \in \mathbb{K}$ are either eigenvalues of M or belong to the set of μ_l bases of exponentials in $\boldsymbol{Q}(s, \boldsymbol{y})$.

Proof. By linear algebra, $\exists S, J \in \mathbb{K}^{h \times h}$ such that $\det(S) \neq 0$ and $J = S^{-1}MS$ is the Jordan normal form of M. By making a change of variables and splitting the variables into independent sets, we can assume without loss of generality that M has the structure of a Jordan block, i.e., for a certain λ eigenvalue of M

$$M = \begin{pmatrix} \lambda & & & \\ 1 & \lambda & & \\ & & \ddots & \\ & & 1 & \lambda \end{pmatrix}.$$

We denote by $X_j(z)$ the generating function of the sequence $(x_j^{(s)})_{s \in \mathbb{N}}$. Since the components of $\boldsymbol{Q}(s, \boldsymbol{y})$ are linear combinations of exponentials with polynomial coefficients, the corresponding generating functions are rational functions $U_j(z, \boldsymbol{y})/V_j(z)$ such that the roots of the V_j are the μ_l bases of exponentials in $\boldsymbol{Q}(s, \boldsymbol{y})$.

From the recurrence we get the following system of equations for the X_j:

$$\begin{pmatrix} \frac{X_1(z) - x_1^{(0)}}{z} \\ \vdots \\ \frac{X_h(z) - x_h^{(0)}}{z} \end{pmatrix} = M \begin{pmatrix} X_1(z) \\ \vdots \\ X_h(z) \end{pmatrix} + \begin{pmatrix} \frac{U_1(z,y)}{V_1(z)} \\ \vdots \\ \frac{U_h(z,y)}{V_h(z)} \end{pmatrix}.$$

The solution to this system is

$$\begin{pmatrix} X_1(z) \\ \vdots \\ X_h(z) \end{pmatrix} = (I - zM)^{-1} \left(\begin{pmatrix} x_1^{(0)} \\ \vdots \\ x_h^{(0)} \end{pmatrix} + z \begin{pmatrix} \frac{U_1(z,\boldsymbol{y})}{V_1(z)} \\ \vdots \\ \frac{U_h(z,\boldsymbol{y})}{V_h(z)} \end{pmatrix} \right),$$

where

$$(I - zM)^{-1} = \begin{pmatrix} \frac{1}{1-\lambda z} & & & \\ \frac{z}{(1-\lambda z)^2} & \frac{1}{1-\lambda z} & & \\ \vdots & \ddots & \ddots & \\ \frac{z^{h-1}}{(1-\lambda z)^h} & \cdots & \frac{z}{(1-\lambda z)^2} & \frac{1}{1-\lambda z} \end{pmatrix}.$$

Therefore, the generating functions X_j are also rational functions with poles which are either eigenvalues of M or μ_l bases of exponentials in $\boldsymbol{Q}(s, \boldsymbol{y})$. From the theory of rational generating functions, we get that the solutions to the recurrence have the form as in the statement of the lemma. $\square$

Now we can characterise natural powers of solvable mappings by using the equivalence of computing powers and solving recurrences:

Proposition 26. *Let $\boldsymbol{g} \in \mathbb{K}[\boldsymbol{x}]^m$ be a solvable mapping with eigenvalues in $\mathbb{K}$. Then $\forall j : 1 \le j \le m$ $\forall s \in \mathbb{N}$ $g_j^s(\boldsymbol{x})$, the j-th component of $\boldsymbol{g}^s(\boldsymbol{x})$, can be expressed as*

$$g_j^s(\boldsymbol{x}) = \sum_{l=1}^{r_j} P_{jl}(s, \boldsymbol{x})(\gamma_{jl})^s \ ,$$

where for $1 \le j \le m$, there exists $r_j \in \mathbb{N}$ such that for $1 \le l \le r_j$, $P_{jl} \in \mathbb{K}[s, \boldsymbol{x}]$ and each $\gamma_{jl} \in \mathbb{K}$ is a product of eigenvalues of $\boldsymbol{g}$.

Proof. The statement is equivalent to the following one. Given a solvable mapping $\boldsymbol{g} \in \mathbb{K}[\boldsymbol{x}]^m$ with eigenvalues in $\mathbb{K}$, we have to prove that the general solution of the recurrence $\boldsymbol{x}^{(s+1)} = \boldsymbol{g}(\boldsymbol{x}^{(s)})$ has the form for $1 \le j \le m$:

$$x_j^{(s)} = \sum_{l=1}^{r_j} P_{jl}(s, \boldsymbol{x}^{(0)})(\gamma_{jl})^s, \quad 1 \le j \le m, \ s \ge 0,$$

where for $1 \le j \le m$, there exists $r_j \in \mathbb{N}$ such that for $1 \le l \le r_j$, $P_{jl} \in \mathbb{K}[s, \boldsymbol{x}^{(0)}]$ and each of the $\gamma_{jl} \in \mathbb{K}$ is a product of eigenvalues of $\boldsymbol{g}$.

Since $\boldsymbol{g}$ is solvable, there exists a partition of the set of variables $\boldsymbol{x}$, $\boldsymbol{x} = \bigcup_{i=1}^k \boldsymbol{w_i}$ with $\boldsymbol{w_i} \cap \boldsymbol{w_j} = \emptyset$ if $i \ne j$, such that $\forall i : 1 \le i \le k$ we have

$$g_{\boldsymbol{w_i}}(\boldsymbol{x}) = M_i \boldsymbol{w_i}^T + \boldsymbol{P_i}(\boldsymbol{w_1}, \ldots, \boldsymbol{w_{i-1}}),$$

where $M_i \in \mathbb{K}^{|\boldsymbol{w_i}| \times |\boldsymbol{w_i}|}$ is a matrix and $\boldsymbol{P_i}$ is a vector of $|\boldsymbol{w_i}|$ polynomials with coefficients in $\mathbb{K}$ and depending on the variables in $\boldsymbol{w_1}, \ldots, \boldsymbol{w_{i-1}}$.

Let us prove the proposition by induction over i, the counter of the sets in the partition. By renaming the variables, we can assume without loss of generality that there exist $0 = h_0 \le h_1 \le h_2 \le \cdots \le h_k = m$ such that $\forall i : 1 \le i \le k$ $\boldsymbol{w_i} = \{x_{h_{i-1}+1}, x_{h_{i-1}+2}, \ldots, x_{h_i}\}$.

For $i = 0$ we want to prove that $\forall j$ such that $1 \le j \le h_1$, $x_j^{(s)}$ has the form like in the statement. For the first h_1 variables we have the recurrence:

$$\begin{pmatrix} x_1^{(s+1)} \\ \vdots \\ x_{h_1}^{(s+1)} \end{pmatrix} = M_1 \begin{pmatrix} x_1^{(s)} \\ \vdots \\ x_{h_1}^{(s)} \end{pmatrix} + \boldsymbol{P_1},$$

where M_1 is a matrix and $\boldsymbol{P_1}$ is a constant vector. By Lemma 25, the $x_j^{(s)}$ have the desired form for $1 \le j \le h_1$. Moreover, since $\boldsymbol{P_1}$ is constant, for $1 \le j \le h_1$ the bases of exponentials in $x_j^{(s)}$ are eigenvalues of M_1, and therefore eigenvalues of $\boldsymbol{g}$.

Now for $i > 0$ we have the recurrence:

$$\begin{pmatrix} x_{h_{i-1}+1}^{(s+1)} \\ \vdots \\ x_{h_i}^{(s+1)} \end{pmatrix} = M_i \begin{pmatrix} x_{h_{i-1}+1}^{(s)} \\ \vdots \\ x_{h_i}^{(s)} \end{pmatrix} + \boldsymbol{P_i}(x_1^{(s)}, \dots, x_{h_{i-1}}^{(s)}).$$

By induction hypothesis, $\forall j : 1 \le j \le h_{i-1}$ $x_j^{(s)}$ has the form like in the statement. Therefore, if $\forall j : 1 \le j \le h_{i-1}$ we plug the solution $x_j^{(s)}$ in $\boldsymbol{P_i}(x_1^{(s)}, \dots, x_{h_{i-1}}^{(s)})$, we get that $\boldsymbol{P_i}(x_1^{(s)}, \dots, x_{h_{i-1}}^{(s)})$ is a vector of functions of the form

$$\sum_{l=1}^{r} Q_l(s, x_1^{(0)}, \dots, x_{h_{i-1}}^{(0)}) \mu_l^s,$$

where for $1 \le l \le r$, $Q_l \in \mathbb{K}[s, x_1^{(0)}, \dots, x_{h_{i-1}}^{(0)}]$ and each of the $\mu_l \in \mathbb{K}$ is a product of eigenvalues of $\boldsymbol{g}$ (since $\forall j : 1 \le j \le h_{i-1}$ the bases of exponentials in the solutions $x_j^{(s)}$ are products of eigenvalues of $\boldsymbol{g}$). By Lemma 25 again, $\forall j : h_{i-1} < j \le h_i$ the $x_j^{(s)}$ have the required form, and the bases of exponentials appearing in them are either eigenvalues of M_i or bases of exponentials in $\boldsymbol{P_i}(x_1^{(s)}, \dots, x_{h_{i-1}}^{(s)})$; in either case, the bases of exponentials in the $x_j^{(s)}$ are products of eigenvalues of $\boldsymbol{g}$, which is what we wanted to see. □

Given a solvable mapping $\boldsymbol{g} \in \mathbb{R}[\boldsymbol{x}]^m$ with positive eigenvalues, the exponential terms $(\gamma_{jl})^s$ in the proposition above are well-defined for any $s \in \mathbb{R}$. Thus, it is possible to extend the powers of a solvable mapping with positive eigenvalues $\boldsymbol{g}^s$ to general $s \in \mathbb{R}$ by using the right-hand side formula in Proposition 26. In order to show that $\boldsymbol{g}^s$ for $s \in \mathbb{R}$ is well-defined, in the sense that $\boldsymbol{g}^0(\boldsymbol{x}) = \boldsymbol{x}$ and $\boldsymbol{g}^{s+t}(\boldsymbol{x}) = \boldsymbol{g}^s(\boldsymbol{g}^t(\boldsymbol{x}))$ for any $s, t \in \mathbb{R}$, we need the following lemma:

Lemma 27. *Let $\varphi : \mathbb{R} \to \mathbb{R}$ be a function of the form $\varphi(s) = \sum_{\gamma \in \Gamma} p_\gamma(s)\gamma^s$ for a certain $\Gamma \subset \mathbb{R}^+$ which is finite and such that $\forall \gamma \in \Gamma$, $p_\gamma \in \mathbb{R}[s]$ and $p_\gamma \ne 0$. If $\forall n \in \mathbb{N}$ $\varphi(n) = 0$, then $\Gamma = \emptyset$ (and therefore $\varphi \equiv 0$).*

Proof. Let us assume that $\Gamma \ne \emptyset$ and we will get a contradiction. Let $\gamma_* = \max_{\gamma \in \Gamma} \gamma$. Then $\forall \gamma \in \Gamma, \gamma \ne \gamma_*$ implies $\gamma < \gamma_*$. So

$$\lim_{s \to \infty} \frac{\varphi(s)}{\gamma_*^s} = \lim_{s \to \infty} p_{\gamma_*}(s).$$

And since $\forall n \in \mathbb{N}$ $\varphi(n) = 0$, we have that $\lim_{s \to \infty}(\varphi(s)/\gamma_*^s) = \lim_{s \to \infty} p_{\gamma_*}(s) = 0$, which implies $p_{\gamma_*} = 0$. But this is impossible. □

Now we can show well-definedness of real powers of solvable mappings with positive eigenvalues:

Proposition 28. *Let $\boldsymbol{g} \in \mathbb{R}[\boldsymbol{x}]^m$ be a solvable mapping with positive eigenvalues. Then $\forall \boldsymbol{\omega} \in \mathbb{R}^m$ $\boldsymbol{g}^0(\boldsymbol{\omega}) = \boldsymbol{\omega}$ and $\forall s, t \in \mathbb{R}$, $\boldsymbol{g}^{s+t}(\boldsymbol{\omega}) = \boldsymbol{g}^s(\boldsymbol{g}^t(\boldsymbol{\omega}))$.*

Proof. Since $0 \in \mathbb{N}$, $\forall \boldsymbol{\omega} \in \mathbb{R}^m$ $g^0(\omega) = \omega$ by Proposition 26. Now let us fix $\boldsymbol{\omega} \in \mathbb{R}^m$ and define $\boldsymbol{G}(s, t) := \boldsymbol{g}^{s+t}(\boldsymbol{\omega}) - \boldsymbol{g}^s(\boldsymbol{g}^t(\boldsymbol{\omega}))$. By Proposition 26 again, $\forall s, t \in \mathbb{N}$, $\boldsymbol{G}(s, t) = \boldsymbol{0}$. Let us fix $s \in \mathbb{N}$. Then all the components of $\boldsymbol{G}(s, t)$ are of the form like in Lemma 27. Since $\forall t \in \mathbb{N}$ $\boldsymbol{G}(s, t) = \boldsymbol{0}$, we get that $\forall t \in \mathbb{R}$ $\boldsymbol{G}(s, t) = \boldsymbol{0}$. So $\forall s \in \mathbb{N}$ $\forall t \in \mathbb{R}$, $\boldsymbol{G}(s, t) = \boldsymbol{0}$. Now if we fix

$t \in \mathbb{R}$, again by using the same argument we get $\forall s \in \mathbb{R}\ \boldsymbol{G}(s,t) = \boldsymbol{0}$. Thus finally $\forall s, t \in \mathbb{R}$, $\boldsymbol{G}(s,t) = \boldsymbol{0}$. $\square$

In particular, the above proposition implies that $\boldsymbol{g}^{-1}$ is the inverse of $\boldsymbol{g}$. Moreover, Theorem 14 from Section 7 follows immediately:

Theorem 14. *Let $\boldsymbol{g} \in \mathbb{Q}[\boldsymbol{x}]^m$ be a solvable mapping with positive rational eigenvalues. Then $\forall j : 1 \le j \le m\ \forall s \in \mathbb{Z}\ g_j^s(\boldsymbol{x})$, the j-th component of $\boldsymbol{g}^s(\boldsymbol{x})$ (negative exponents mean powers of the inverse of $\boldsymbol{g}$), can be expressed as*

$$g_j^s(\boldsymbol{x}) = \sum_{l=1}^{r_j} P_{jl}(s,\boldsymbol{x})(\gamma_{jl})^s \,,$$

where for $1 \le j \le m$, there exists $r_j \in \mathbb{N}$ such that for $1 \le l \le r_j$, $P_{jl} \in \mathbb{Q}[s,\boldsymbol{x}]$ and $\gamma_{jl} \in \mathbb{Q}^+$. Moreover, each γ_{jl} is a product of eigenvalues of $\boldsymbol{g}$.

Finally, the following is the main result of this subsection:

Theorem 29. *Let $J \subseteq \mathbb{R}[\boldsymbol{x}]$ be an ideal of variety and $\boldsymbol{g} \in \mathbb{R}[\boldsymbol{x}]^m$ be a solvable mapping with positive eigenvalues. Then*

$$\bigcap_{s\in\mathbb{N}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x}) = \bigcap_{s\in\mathbb{R}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x}).$$

Proof. It is obvious that $\bigcap_{s\in\mathbb{N}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x}) \supseteq \bigcap_{s\in\mathbb{R}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$. Let us see the other inclusion. Let $p \in \bigcap_{s\in\mathbb{N}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$. So $p \in \mathbf{I}(\bigcup_{s\in\mathbb{N}} \mathbf{V}(J \circ \boldsymbol{g}^{-s}(\boldsymbol{x}))) = \mathbf{I}(\bigcup_{s\in\mathbb{N}} \boldsymbol{g}^s(\mathbf{V}(J)))$ by Lemmas 21 and 23.

We want to see that $p \in \bigcap_{s\in\mathbb{R}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$, or equivalently that $\forall s \in \mathbb{R}$, $p \in J \circ \boldsymbol{g}^{-s}(\boldsymbol{x}) = \mathbf{IV}(J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})) = \mathbf{I}(\boldsymbol{g}^s(\mathbf{V}(J)))$. So we have to prove that $\forall \boldsymbol{\omega} \in \mathbf{V}(J)$ and $\forall s \in \mathbb{R}$ then $(p \circ \boldsymbol{g}^s)(\boldsymbol{\omega}) = 0$.

Now fix any $\boldsymbol{\omega} \in \mathbf{V}(J)$ and consider the function $\varphi_{\boldsymbol{\omega}} : \mathbb{R} \to \mathbb{R}$, $\varphi_{\boldsymbol{\omega}}(s) = p(\boldsymbol{g}^s(\boldsymbol{\omega}))$. Since by hypothesis $p \in \mathbf{I}(\bigcup_{s\in\mathbb{N}} \boldsymbol{g}^s(\mathbf{V}(J)))$, we have that $\forall s \in \mathbb{N}\ \varphi_{\boldsymbol{\omega}}(s) = p(\boldsymbol{g}^s(\boldsymbol{\omega})) = 0$. By Lemma 27, $\varphi_{\boldsymbol{\omega}} \equiv 0$. Therefore $\forall \boldsymbol{\omega} \in \mathbf{V}(J)$ and $\forall s \in \mathbb{R}$, $(p \circ \boldsymbol{g}^s)(\boldsymbol{\omega}) = 0$. $\square$

A.3. Primality

We recall that an ideal $J \subseteq \mathbb{R}[\boldsymbol{x}]$ is *prime* if, given polynomials $p, q \in \mathbb{R}[\boldsymbol{x}]$, $p \cdot q \in J$ implies that either $p \in J$ or $q \in J$. The following two results show that primality is preserved under the mapping $J \mapsto \bigcap_{s\in\mathbb{R}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$. More precisely, we will prove that if $J \subseteq \mathbb{R}[\boldsymbol{x}]$ is a prime ideal of variety and $g \in \mathbb{R}[\boldsymbol{x}]^m$ is a solvable mapping with positive eigenvalues, then

$$\bigcap_{s\in\mathbb{R}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$$

is also a prime ideal.

Lemma 30. *If $J \subseteq \mathbb{R}[\boldsymbol{x}]$ is a prime ideal and $g \in \mathbb{R}[\boldsymbol{x}]^m$ is an invertible polynomial mapping, then $J \circ \boldsymbol{g}(\boldsymbol{x})$ is also a prime ideal.*

Proof. Let p, q be such that $p \cdot q \in J \circ \boldsymbol{g}(\boldsymbol{x})$. We have to see that either $p \in J \circ \boldsymbol{g}(\boldsymbol{x})$ or $q \in J \circ \boldsymbol{g}(\boldsymbol{x})$. As J is prime, $p \cdot q \in J \circ \boldsymbol{g}(\boldsymbol{x}) \Rightarrow (p \cdot q) \circ \boldsymbol{g}^{-1}(\boldsymbol{x}) = p \circ \boldsymbol{g}^{-1}(\boldsymbol{x}) \cdot q \circ \boldsymbol{g}^{-1}(\boldsymbol{x}) \in J \Rightarrow p \circ \boldsymbol{g}^{-1}(\boldsymbol{x}) \in J$ or $q \circ \boldsymbol{g}^{-1}(\boldsymbol{x}) \in J \Rightarrow p \in J \circ \boldsymbol{g}(\boldsymbol{x})$ or $q \in J \circ \boldsymbol{g}(\boldsymbol{x})$. $\square$

Theorem 31. *Let $J \subseteq \mathbb{R}[\boldsymbol{x}]$ be a prime ideal of variety and $\boldsymbol{g} \in \mathbb{R}[\boldsymbol{x}]^m$ be a solvable mapping with positive eigenvalues. Then $\bigcap_{s\in\mathbb{R}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$ is also a prime ideal.*

Proof. Let p, q be such that $pq \in \bigcap_{s\in\mathbb{R}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$. We have to see that either $p \in \bigcap_{s\in\mathbb{R}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$ or $q \in \bigcap_{s\in\mathbb{R}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$. But $pq \in \bigcap_{s\in\mathbb{R}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$ implies that $\forall s \in \mathbb{R}$ $pq \in J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$; and as $J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$ is prime by Lemma 30, we have that either $p \in J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$ or $q \in J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$. Since $J = \mathbf{IV}(J)$, we have $p \in J \circ \boldsymbol{g}^{-s}(\boldsymbol{x}) \Leftrightarrow p \in \mathbf{IV}(J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})) \Leftrightarrow \forall \boldsymbol{\omega} \in \mathbf{V}(J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})) = \boldsymbol{g}^s(\mathbf{V}(J))$, $p(\boldsymbol{\omega}) = 0 \Leftrightarrow \forall \boldsymbol{\omega} \in \mathbf{V}(J)$, $p(\boldsymbol{g}^s(\boldsymbol{\omega})) = 0$; and similarly for q.

Now let us distinguish two cases. Let us first assume that $\exists s^* \in \mathbb{R}$ such that $\forall n \in \mathbb{N}\ \exists s_n^*$ such that $|s^* - s_n^*| < 1/(n+1)$ and $p \in J \circ \boldsymbol{g}^{-s_n^*}(\boldsymbol{x})$. Let us take an arbitrary $\boldsymbol{\omega} \in \mathbf{V}(J)$ and define the function $\varphi_{\boldsymbol{\omega}} : \mathbb{R} \to \mathbb{R}$, $\varphi_{\boldsymbol{\omega}}(s) = p(\boldsymbol{g}^s(\boldsymbol{\omega}))$. Clearly $\varphi_{\boldsymbol{\omega}}$ is analytical. But since $\forall n \in \mathbb{N}$ we have that $p \in J \circ \boldsymbol{g}^{-s_n^*}(\boldsymbol{x})$, then $\forall n \in \mathbb{N}$ $p(\boldsymbol{g}^{s_n^*}(\boldsymbol{\omega})) = 0$; and moreover, $s_n^* \to s^*$. As $\varphi_{\boldsymbol{\omega}}$ is analytical, $\varphi_{\boldsymbol{\omega}}(s) = 0\ \forall s \in \mathbb{R}$. Since $\boldsymbol{\omega} \in \mathbf{V}(J)$ is arbitrary, we have that $\forall \boldsymbol{\omega} \in \mathbf{V}(J)$ $\forall s \in \mathbb{R}$ $p(\boldsymbol{g}^s(\boldsymbol{\omega})) = \varphi_{\boldsymbol{\omega}}(s) = 0$. This implies that $\forall s \in \mathbb{R}$ $p \in J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$, or equivalently $p \in \bigcap_{s\in\mathbb{R}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$.

Now let us assume the contrary. So let us assume that $\forall s^* \in \mathbb{R}\ \exists n \in \mathbb{N}$ such that $\forall s_n^* \in (s^*-1/(n+1), s^*+1/(n+1))$, then $p \notin J \circ \boldsymbol{g}^{-s_n^*}(\boldsymbol{x})$; but this implies that $q \in J \circ \boldsymbol{g}^{-s_n^*}(\boldsymbol{x})$. Let us take any $s^* \in \mathbb{R}$. Given $\boldsymbol{\omega} \in \mathbf{V}(J)$ we define the analytical function $\psi_{\boldsymbol{\omega}} : \mathbb{R} \to \mathbb{R}$, $\psi_{\boldsymbol{\omega}}(s) = q(\boldsymbol{g}^s(\boldsymbol{\omega}))$. Then there exists $n^* \in \mathbb{N}$ such that $\forall s \in (s^* - 1/(n^*+1), s^* + 1/(n^*+1))$, $q(\boldsymbol{g}^s(\boldsymbol{\omega})) = \psi_{\boldsymbol{\omega}}(s) = 0$. Thus, since $\psi_{\boldsymbol{\omega}}$ is analytical, $\psi_{\boldsymbol{\omega}}(s) = 0\ \forall s \in \mathbb{R}$. Following a similar argument as above, we get that $q \in \bigcap_{s\in\mathbb{R}} J \circ \boldsymbol{g}^{-s}(\boldsymbol{x})$ in this case. □

Appendix B. Correctness and completeness of the implementation

First of all, we need the following technical lemma, which shows that the set of all polynomial relations between the powers of the eigenvalues and their inverses can be characterised:

Proposition 32. *Let $\lambda_1, \ldots, \lambda_k$ be different prime numbers. Let $L = \{u_1v_1 - 1, \ldots, u_kv_k - 1\}$. Then L generates the set of all polynomial relations between the powers of these prime numbers, i.e.,*

$$\langle L\rangle_{\mathbb{Q}[s,\boldsymbol{u},\boldsymbol{v}]} = \{p \in \mathbb{Q}[s,\boldsymbol{u},\boldsymbol{v}] \mid \forall s_0 \in \mathbb{N}\ \ p(s_0, \boldsymbol{\lambda}^{s_0}, \boldsymbol{\lambda}^{-s_0}) = 0\}.$$

Proof. The $\subseteq$ inclusion is obvious. Now let us prove $\supseteq$. Let p be such that $\forall s_0 \in \mathbb{N}$ $p(s_0, \boldsymbol{\lambda}^{s_0}, \boldsymbol{\lambda}^{-s_0}) = 0$. Let us take any term ordering $\succ$ and let us divide p into L. Then we get polynomials $r, p_1, \ldots, p_k \in \mathbb{Q}[s,\boldsymbol{u},\boldsymbol{v}]$ such that

$$p(s,\boldsymbol{u},\boldsymbol{v}) = r(s,\boldsymbol{u},\boldsymbol{v}) + \sum_{i=1}^{k} p_i(s,\boldsymbol{u},\boldsymbol{v}) \cdot (u_iv_i - 1).$$

We want to show that $r = 0$. Let us assume that $r \neq 0$ and we will get a contradiction. We can write

$$r(s,\boldsymbol{u},\boldsymbol{v}) = \sum_{\boldsymbol{\alpha},\boldsymbol{\beta}\in\mathbb{N}^k} P_{\boldsymbol{\alpha},\boldsymbol{\beta}}(s)\boldsymbol{u}^{\boldsymbol{\alpha}}\boldsymbol{v}^{\boldsymbol{\beta}},$$

where $\boldsymbol{u}^{\boldsymbol{\alpha}} = \prod_{i=1}^k u_i^{\alpha_i}$, $\boldsymbol{v}^{\boldsymbol{\beta}} = \prod_{i=1}^k v_i^{\beta_i}$ and $P_{\boldsymbol{\alpha},\boldsymbol{\beta}} \in \mathbb{Q}[s]$ are polynomials such that at least one of them is not null, and only finitely many of them are not null.

Then $\forall s_0 \in \mathbb{N}$ we have that

$$0 = p(s_0, \boldsymbol{\lambda}^{s_0}, \boldsymbol{\lambda}^{-s_0}) = r(s_0, \boldsymbol{\lambda}^{s_0}, \boldsymbol{\lambda}^{-s_0}) = \sum_{\boldsymbol{\alpha},\boldsymbol{\beta}\in\mathbb{N}^k} P_{\boldsymbol{\alpha},\boldsymbol{\beta}}(s_0) \prod_{i=1}^{k} \lambda_i^{(\alpha_i-\beta_i)s_0}.$$

Given $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{N}^k$, let us define $\lambda_{\boldsymbol{\alpha},\boldsymbol{\beta}} = \prod_{i=1}^{k} \lambda_i^{\alpha_i-\beta_i}$. Then the above equation can be expressed as $\forall s_0 \in \mathbb{N}$

$$0 = \sum_{\boldsymbol{\alpha},\boldsymbol{\beta}\in\mathbb{N}^k} P_{\boldsymbol{\alpha},\boldsymbol{\beta}}(s_0)\lambda_{\boldsymbol{\alpha},\boldsymbol{\beta}}^{s_0}. \tag{B.1}$$

Now let us see that, $\forall \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\delta} \in \mathbb{N}^k$, if $(\boldsymbol{\alpha}, \boldsymbol{\beta}) \neq (\boldsymbol{\gamma}, \boldsymbol{\delta})$ then $\lambda_{\boldsymbol{\alpha},\boldsymbol{\beta}} \neq \lambda_{\boldsymbol{\gamma},\boldsymbol{\delta}}$. Let us assume the contrary, i.e., that $\lambda_{\boldsymbol{\alpha},\boldsymbol{\beta}} = \lambda_{\boldsymbol{\gamma},\boldsymbol{\delta}}$ and we will get a contradiction. If $\lambda_{\boldsymbol{\alpha},\boldsymbol{\beta}} = \lambda_{\boldsymbol{\gamma},\boldsymbol{\delta}}$, then $\prod_{i=1}^{k} \lambda_i^{\alpha_i-\beta_i-\gamma_i+\delta_i} = 1$. As the λ_i are different prime numbers, we necessarily have that $\alpha_i - \beta_i - \gamma_i + \delta_i = 0$ for $1 \le i \le k$. Moreover, since no monomial of r can be divided by the $u_i v_i$ by the properties of the division algorithm, either $\alpha_i = 0$ or $\beta_i = 0$, and either $\gamma_i = 0$ or $\delta_i = 0$. If $\alpha_i = 0$, then $\beta_i = \delta_i - \gamma_i$, and as $\beta_i \ge 0$ and either $\gamma_i = 0$ or $\delta_i = 0$, we get that $0 = \gamma_i = \alpha_i$ and $\beta_i = \delta_i$. The case $\beta_i = 0$ is symmetric. So $\lambda_{\boldsymbol{\alpha},\boldsymbol{\beta}} = \lambda_{\boldsymbol{\gamma},\boldsymbol{\delta}}$ implies $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\boldsymbol{\gamma}, \boldsymbol{\delta})$.

Let $\boldsymbol{\alpha}^*, \boldsymbol{\beta}^* \in \mathbb{N}^k$ be such that $\lambda_{\boldsymbol{\alpha}^*,\boldsymbol{\beta}^*} = \max\{\lambda_{\boldsymbol{\alpha},\boldsymbol{\beta}} | P_{\boldsymbol{\alpha},\boldsymbol{\beta}} \neq 0\}$. Notice that $\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*$ are well-defined, since $r \neq 0$ by hypothesis. By definition, and as $(\boldsymbol{\alpha}, \boldsymbol{\beta}) \neq (\boldsymbol{\gamma}, \boldsymbol{\delta})$ implies $\lambda_{\boldsymbol{\alpha},\boldsymbol{\beta}} \neq \lambda_{\boldsymbol{\gamma},\boldsymbol{\delta}}$, we have that $(\boldsymbol{\alpha}, \boldsymbol{\beta}) \neq (\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$ implies $\lambda_{\boldsymbol{\alpha},\boldsymbol{\beta}} < \lambda_{\boldsymbol{\alpha}^*,\boldsymbol{\beta}^*}$.

Now we divide Eq. (B.1) into $(\lambda_{\boldsymbol{\alpha}^*,\boldsymbol{\beta}^*})^{s_0}$ and get that $\forall s_0 \in \mathbb{N}$

$$0 = \sum_{\boldsymbol{\alpha},\boldsymbol{\beta}\in\mathbb{N}^k} P_{\boldsymbol{\alpha},\boldsymbol{\beta}}(s_0)\left(\frac{\lambda_{\boldsymbol{\alpha},\boldsymbol{\beta}}}{\lambda_{\boldsymbol{\alpha}^*,\boldsymbol{\beta}^*}}\right)^{s_0}.$$

Taking limits, $0 = \lim_{s_0\to\infty} P_{\boldsymbol{\alpha}^*,\boldsymbol{\beta}^*}(s_0)$, which contradicts $P_{\boldsymbol{\alpha}^*,\boldsymbol{\beta}^*} \neq 0$. □

We also need the following lemma. It intuitively means that $\boldsymbol{F_i}(-s, \boldsymbol{v}, \boldsymbol{u}, \boldsymbol{x})$ and $\boldsymbol{F_i}(s, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{x})$ are "inverses modulo $\langle L\rangle$" (where the $\boldsymbol{F_i}$ refer to the mappings from Theorem 14 in Section 7):

Lemma 33. *For $1 \le i \le n$ and $\forall q \in \mathbb{Q}[\boldsymbol{x}]$ we have that*

$$q(\boldsymbol{x}) - q(\boldsymbol{F_i}(s, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{F_i}(-s, \boldsymbol{v}, \boldsymbol{u}, \boldsymbol{x}))) \in \langle L\rangle_{\mathbb{Q}[s,\boldsymbol{u},\boldsymbol{v},\boldsymbol{x}]}.$$

Proof. We can write

$$q(\boldsymbol{x}) - q(\boldsymbol{F_i}(s, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{F_i}(-s, \boldsymbol{v}, \boldsymbol{u}, \boldsymbol{x}))) = \sum_{\boldsymbol{\alpha}\in\mathbb{N}^m} R_{\boldsymbol{\alpha}}(s, \boldsymbol{u}, \boldsymbol{v})\, \boldsymbol{x}^{\boldsymbol{\alpha}},$$

where $\boldsymbol{x}^{\boldsymbol{\alpha}} = \prod_{j=1}^{m} x_j^{\alpha_j}$ and $R_{\boldsymbol{\alpha}} \in \mathbb{Q}[s, \boldsymbol{u}, \boldsymbol{v}]$ are polynomials such that only a finite number of them are different from 0. Then $\forall s_0 \in \mathbb{N}$

$$\begin{aligned} &q(\boldsymbol{x}) - q(\boldsymbol{F_i}(s_0, \boldsymbol{\lambda}^{s_0}, \boldsymbol{\lambda}^{-s_0}, \boldsymbol{F_i}(-s_0, \boldsymbol{\lambda}^{-s_0}, \boldsymbol{\lambda}^{s_0}, \boldsymbol{x}))) \\ &\quad = q(\boldsymbol{x}) - q(\boldsymbol{f_i}^{s_0}(\boldsymbol{f_i}^{-s_0}(\boldsymbol{x}))) = 0. \end{aligned}$$

Therefore $\forall s_0 \in \mathbb{N}$ we have

$$\sum_{\boldsymbol{\alpha}\in\mathbb{N}^m} R_{\boldsymbol{\alpha}}(s_0, \boldsymbol{\lambda}^{s_0}, \boldsymbol{\lambda}^{-s_0})\, \boldsymbol{x}^{\boldsymbol{\alpha}} = 0\,,$$

which implies that $\forall \boldsymbol{\alpha} \in \mathbb{N}^m$ $R_{\boldsymbol{\alpha}} \in \langle L\rangle_{\mathbb{Q}[s,\boldsymbol{u},\boldsymbol{v}]}$. Thus

$$q(\boldsymbol{x}) - q(\boldsymbol{F_i}(s, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{F_i}(-s, \boldsymbol{v}, \boldsymbol{u}, \boldsymbol{x}))) \in \langle L\rangle_{\mathbb{Q}[s,\boldsymbol{u},\boldsymbol{v},\boldsymbol{x}]}. \quad \square$$

The following result intuitively means that we do not lose invariant polynomials in our approximation:

Proposition 34. *The inclusion $I_\infty \subseteq \langle S\rangle_{\mathbb{Q}[\boldsymbol{x}]}$ holds in all the executions of the implementation.*

Proof. The inclusion holds at the beginning as $I_\infty \subseteq \mathbf{IV}(\langle S_0\rangle) = \langle S_0\rangle = \langle S\rangle$. It remains to be seen that the inclusion is preserved at each iteration. From now on, $\langle\rangle$ means $\langle\rangle_{\mathbb{Q}[s,\boldsymbol{u},\boldsymbol{v},\boldsymbol{x}]}$. It suffices to see that

$$I_\infty \subseteq \bigcap_{i=1}^{n} \langle L \cup (I_\infty \circ \boldsymbol{F_i}(-s,\boldsymbol{v},\boldsymbol{u},\boldsymbol{x}))\rangle.$$

So for $1 \le i \le n$ we have to see that $I_\infty \subseteq \langle L \cup (I_\infty \circ \boldsymbol{F_i}(-s,\boldsymbol{v},\boldsymbol{u},\boldsymbol{x}))\rangle$. Given $q \in I_\infty$, we want to show that $q \in \langle L \cup (I_\infty \circ \boldsymbol{F_i}(-s,\boldsymbol{v},\boldsymbol{u},\boldsymbol{x}))\rangle$.

First, we show that $q(\boldsymbol{F_i}(s,\boldsymbol{u},\boldsymbol{v},\boldsymbol{x})) \in \langle L \cup I_\infty\rangle$. If we divide $q(\boldsymbol{F_i}(s,\boldsymbol{u},\boldsymbol{v},\boldsymbol{x}))$ into a Gröbner basis $p_1,\ldots,p_K$ of I_∞ with respect to any term ordering $\succ$, we get $R, Q_1,\ldots,Q_K \in \mathbb{Q}[s,\boldsymbol{u},\boldsymbol{v},\boldsymbol{x}]$ such that

$$q(\boldsymbol{F_i}(s,\boldsymbol{u},\boldsymbol{v},\boldsymbol{x})) = R + \sum_{j=1}^{K} Q_j p_j.$$

We want to show $R \in \langle L\rangle$. As $q \in I_\infty$, $\forall s_0 \in \mathbb{N}$, $q(\boldsymbol{F_i}(s_0,\boldsymbol{\lambda}^{s_0},\boldsymbol{\lambda}^{-s_0},\boldsymbol{x})) = q(\boldsymbol{f_i}^{s_0}(\boldsymbol{x})) \in I_\infty$. But the remainder obtained when dividing $q(\boldsymbol{f_i}^{s_0}(\boldsymbol{x})) \in I_\infty$ into $p_1,\ldots,p_K$ is 0. As $p_1,\ldots,p_K$ is a Gröbner basis, it can be proved that $\forall s_0 \in \mathbb{N}$, $R(s_0,\boldsymbol{\lambda}^{s_0},\boldsymbol{\lambda}^{-s_0},\boldsymbol{x}) = 0$.

We write $R(s,\boldsymbol{u},\boldsymbol{v},\boldsymbol{x}) = \sum_{\boldsymbol{\alpha}\in\mathbb{N}^m} R_{\boldsymbol{\alpha}}(s,\boldsymbol{u},\boldsymbol{v})\,\boldsymbol{x}^{\boldsymbol{\alpha}}$, where $\boldsymbol{x}^{\boldsymbol{\alpha}} = \prod_{j=1}^{m} x_j^{\alpha_j}$ and $R_{\boldsymbol{\alpha}} \in \mathbb{Q}[s,\boldsymbol{u},\boldsymbol{v}]$ are polynomials such that only a finite number of them are different from 0. We have that $\forall s_0 \in \mathbb{N}$

$$0 = R(s_0,\boldsymbol{\lambda}^{s_0},\boldsymbol{\lambda}^{-s_0},\boldsymbol{x}) = \sum_{\boldsymbol{\alpha}\in\mathbb{N}^m} R_{\boldsymbol{\alpha}}(s_0,\boldsymbol{\lambda}^{s_0},\boldsymbol{\lambda}^{-s_0})\,\boldsymbol{x}^{\boldsymbol{\alpha}}.$$

So $\forall \boldsymbol{\alpha} \in \mathbb{N}^m$ $\forall s_0 \in \mathbb{N}$, $R_{\boldsymbol{\alpha}}(s_0,\boldsymbol{\lambda}^{s_0},\boldsymbol{\lambda}^{-s_0}) = 0$. By Proposition 32, $R_{\boldsymbol{\alpha}} \in \langle L\rangle_{\mathbb{Q}[s,\boldsymbol{u},\boldsymbol{v}]}$. So $R \in \langle L\rangle$ and $q(\boldsymbol{F_i}(s,\boldsymbol{u},\boldsymbol{v},\boldsymbol{x})) \in \langle L \cup I_\infty\rangle$. As $q(\boldsymbol{F_i}(s,\boldsymbol{u},\boldsymbol{v},\boldsymbol{x})) \in \langle L \cup I_\infty\rangle$ and $L \subset \mathbb{Q}[s,\boldsymbol{u},\boldsymbol{v}]$, substituting $\boldsymbol{x}$ by $\boldsymbol{F_i}(-s,\boldsymbol{v},\boldsymbol{u},\boldsymbol{x})$,

$$q(\boldsymbol{F_i}(s,\boldsymbol{u},\boldsymbol{v},\boldsymbol{F_i}(-s,\boldsymbol{v},\boldsymbol{u},\boldsymbol{x}))) \in \langle L \cup (I_\infty \circ \boldsymbol{F_i}(-s,\boldsymbol{v},\boldsymbol{u},\boldsymbol{x}))\rangle.$$

From Lemma 33, $q(\boldsymbol{x}) - q(\boldsymbol{F_i}(s,\boldsymbol{u},\boldsymbol{v},\boldsymbol{F_i}(-s,\boldsymbol{v},\boldsymbol{u},\boldsymbol{x}))) \in \langle L\rangle$. Therefore $q(\boldsymbol{x}) \in \langle L \cup (I_\infty \circ \boldsymbol{F_i}(-s,\boldsymbol{v},\boldsymbol{u},\boldsymbol{x}))\rangle$. □

Finally, the last theorem implies trivially that the implementation is correct and complete:

Proof of Theorem 15. Let us denote by I_N the ideal computed at the end of the N-th iteration in the Invariant Generation Procedure; and, analogously, let S_N be the set of polynomials computed at the end of the N-th iteration in the implementation.

First, we prove that $\forall N \in \mathbb{N}$, $\langle S_N\rangle \subseteq I_N$. Then the termination of the Invariant Generation Procedure will imply a chain of equalities that will yield the theorem.

So let us prove that $\forall N \in \mathbb{N}$, $\langle S_N\rangle \subseteq I_N$ by induction on N. If $N = 0$ there is nothing to prove, since by definition $I_0 = \langle S_0\rangle_{\mathbb{Q}[\boldsymbol{x}]}$.

If $N > 0$,

$$I_N = \bigcap_{s\in\mathbb{N}} \bigcap_{i=1}^{n} I_{N-1} \circ \boldsymbol{f_i}^{-s}(\boldsymbol{x}),$$

$$\langle S_N \rangle = \mathbb{Q}[\boldsymbol{x}] \cap \left(\bigcap_{i=1}^{n} \langle L \cup (S_{N-1} \circ \boldsymbol{F_i}(-s, \boldsymbol{v}, \boldsymbol{u}, \boldsymbol{x})) \rangle_{\mathbb{Q}[s,\boldsymbol{u},\boldsymbol{v},\boldsymbol{x}]} \right).$$

Given $q \in S_N$, for $1 \le i \le n$ and $s_0 \in \mathbb{N}$ we have to show that $q \in I_{N-1} \circ \boldsymbol{f_i}^{-s_0}(\boldsymbol{x})$. By induction hypothesis, it is enough to see that $q \in \langle S_{N-1} \rangle \circ \boldsymbol{f_i}^{-s_0}(\boldsymbol{x})$.

Now, if $S_{N-1} = \{p_1, \ldots, p_l\}$ there exist polynomials $P_r, L_j \in \mathbb{Q}[s, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{x}]$ for $1 \le r \le l$ and $1 \le j \le k$ such that

$$q(\boldsymbol{x}) = \sum_{r=1}^{l} P_r(s, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{x})\ p_r(\boldsymbol{F_i}(-s, \boldsymbol{v}, \boldsymbol{u}, \boldsymbol{x})) + \sum_{j=1}^{k} L_j(s, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{x})\ (u_j v_j - 1).$$

For any $s_0 \in \mathbb{N}$, by evaluating conveniently the auxiliary variables,

$$\begin{aligned} q(\boldsymbol{x}) &= \sum_{r=1}^{l} P_r(s_0, \boldsymbol{\lambda}^{s_0}, \boldsymbol{\lambda}^{-s_0}, \boldsymbol{x})\ p_r(\boldsymbol{F_i}(-s_0, \boldsymbol{\lambda}^{-s_0}, \boldsymbol{\lambda}^{s_0}, \boldsymbol{x})) \\ &\quad + \sum_{j=1}^{k} L_j(s_0, \boldsymbol{\lambda}^{s_0}, \boldsymbol{\lambda}^{-s_0}, \boldsymbol{x}) \cdot 0 = \sum_{r=1}^{l} P_r(s_0, \boldsymbol{\lambda}^{s_0}, \boldsymbol{\lambda}^{-s_0}, \boldsymbol{x})\ p_r(\boldsymbol{f_i}^{-s_0}(\boldsymbol{x})). \end{aligned}$$

So $q \in \langle S_{N-1} \rangle \circ \boldsymbol{f_i}^{-s_0}(\boldsymbol{x})$ indeed. Therefore $\forall N \in \mathbb{N}$, $\langle S_N \rangle \subseteq I_N$.

Now, if the Invariant Generation Procedure terminates in N iterations, by Theorem 5 $I_N = I_{N-1} = I_\infty$. Then, by Proposition 34, $\langle S_{N-1} \rangle \subseteq I_{N-1} = I_\infty \subseteq \langle S_N \rangle$. But $\langle S_N \rangle \subseteq \langle S_{N-1} \rangle$ clearly holds. So $\langle S_{N-1} \rangle = \langle S_N \rangle$, which implies $S_N = S_{N-1}$ as both are Gröbner bases. Thus, the implementation terminates in at most the same number of iterations as the Invariant Generation Procedure with $\langle S \rangle = I_\infty$. $\square$

References

Becker, T., Weispfenning, V., 1993. Gröbner Bases. A Computational Approach to Commutative Algebra. Springer-Verlag.

Bressoud, D.M., 1989. Factorization and Primality Testing. Springer-Verlag.

Cohen, E., 1990. Programming in the 1990s. Springer-Verlag.

Colón, M., 2004. Approximating the algebraic relational semantics of imperative programs. In: International Symposium on Static Analysis. SAS 2004. In: Lecture Notes in Computer Science, vol. 3148. Springer-Verlag, pp. 296–311.

Cousot, P., Cousot, R., 1976. Static determination of dynamic properties of programs. In: Robinet, B. (Ed.), Proceedings of the 2nd International Symposium on Programming, April, pp. 106–130.

Cousot, P., Cousot, R., 1977. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 238–252.

Cox, D., Little, J., O'Shea, D., 1996. Ideals, Varieties and Algorithms. An Introduction to Computational Algebraic Geometry and Commutative Algebra. Springer-Verlag.

Dershowitz, N., Manna, Z., 1978. Inference rules for program annotation. In: Proceedings of the 3rd International Conference on Software Engineering, pp. 158–167.

Dijkstra, E., 1976. A Discipline of Programming. Prentice Hall.

Freire, P., 2002. www.pedrofreire.com/crea2_en.htm?.

German, S., Wegbreit, B., 1975. A synthesizer of inductive assertions. IEEE Transactions on Software Engineering 1 (1), 68–75.

Gulwani, S., Necula, G.C., 2005. Precise interprocedural analysis using random interpretation. In: Palsberg, J., Abadi, M. (Eds.), Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. ACM, pp. 324–337.

Hoare, T., 2003. The verifying compiler: A grand challenge for computing research. Journal of the ACM 50 (1), 63–69.

Kaldewaij, A., 1990. Programming. The Derivation of Algorithms. Prentice-Hall.

Kapur, D., 2003. Automatically generating loop invariants using quantifier elimination. Technical Report TR-CS-2003-58, Department of Computer Science, UNM. Also in the Proc. of 10th International IMACS Conference on Applications of Computer Algebra, ACA 2004, Lamar, TX, July 2004.

Karr, M., 1976. Affine relationships among variables of a program. Acta Informatica 6, 133–151.

Katz, S., Manna, Z., 1976. Logical analysis of programs. Communications of the ACM 19 (4), 188–206.

Knuth, D.E., 1969. The Art of Computer Programming. In: Seminumerical Algorithms, vol. 2. Addison-Wesley.

Müller-Olm, M., Seidl, H., 2004. A note on Karr's algorithm. In: 31 Int. Coll. on Automata, Languages and Programming, ICALP. In: LNCS, vol. 3142. Springer-Verlag, pp. 1016–1028.

Müller-Olm, M., Seidl, H., 2004a. Computing polynomial program invariants. Information Processing Letters (IPL) 91 (5), 233–244.

Müller-Olm, M., Seidl, H., 2004b. Precise interprocedural analysis through linear algebra. In: ACM SIGPLAN Principles of Programming Languages, POPL 2004, pp. 330–341.

Petter, M., 2004. Berechnung von polynomiellen invarianten. Master's Thesis, Fakultät für Informatik, Technische Universität München. Available at http://www2.cs.tum.edu/˜petter/da.

Rodríguez-Carbonell, E., Kapur, D., 2004a. An abstract interpretation approach for automatic generation of polynomial invariants. In: International Symposium on Static Analysis. SAS 2004. In: Lecture Notes in Computer Science, vol. 3148. Springer-Verlag, pp. 280–295.

Rodríguez-Carbonell, E., Kapur, D., 2004b. Automatic generation of polynomial loop invariants: Algebraic foundations. In: International Symposium on Symbolic and Algebraic Computation 2004. ISSAC04. ACM Press, pp. 266–273.

Rodríguez-Carbonell, E., Kapur, D., 2005. Program verification using automatic generation of invariants. In: 1st International Colloquium on Theoretical Aspects of Computing. ICTAC'04. In: Lecture Notes in Computer Science, vol. 3407. Springer-Verlag, pp. 325–340.

Rodriguez-Carbonell, E., Kapur, D., 2007. Automatic generation of polynomial invariants of bounded degree using abstract interpretation. Science of Computer Programming 64 (1), 54–75.

Sankaranarayanan, S., Sipma, H.B., Manna, Z., 2004. Non-linear loop invariant generation using Gröbner bases. In: ACM SIGPLAN Principles of Programming Languages, POPL 2004, pp. 318–329.

Stanley, R., 1997. Enumerative Combinatorics, vol. 1. Cambridge University Press.

Suzuki, N., Ishihata, K., 1977. Implementation of an array bound checker. In: Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages. ACM Press, pp. 132–143.

Wegbreit, B., 1974. The synthesis of loop predicates. Communications of the ACM 17 (2), 102–112.

Wegbreit, B., 1975. Property extraction in well-founded property sets. IEEE Transactions on Software Engineering 1 (3), 270–285.