Affine Loop Invariant Generation via Matrix Algebra

Yucheng Ji[1,2], Hongfei Fu[2(✉)], Bin Fang[1], and Haibo Chen[1,2]

[1] OS Kernel Lab, Huawei Technologies, Shanghai, China
{jiyucheng,fangbin11,hb.chen}@huawei.com
[2] Shanghai Jiao Tong University, Shanghai, China
fuhf@cs.sjtu.edu.cn

Abstract. Loop invariant generation, which automates the generation of assertions that always hold at the entry of a while loop, has many important applications in program analysis and formal verification. In this work, we target an important category of while loops, namely affine while loops, that are unnested while loops with affine loop guards and variable updates. Such a class of loops widely exists in many programs yet still lacks a general but efficient approach to invariant generation. We propose a novel matrix-algebra approach to automatically synthesizing affine inductive invariants in the form of an affine inequality. The main novelty of our approach is that (i) the approach is general in the sense that it theoretically addresses all the cases of affine invariant generation over an affine while loop, and (ii) it can be efficiently automated through matrix-algebra (such as eigenvalue, matrix inverse) methods.

The details of our approach are as follows. First, for the case where the loop guard is a tautology (i.e., '**true**'), we show that the eigenvalues and their eigenvectors of the matrices derived from the variable updates of the loop body encompass all meaningful affine inductive invariants. Second, for the more general case where the loop guard is a conjunction of affine inequalities, our approach completely addresses the invariant-generation problem by first establishing through matrix inverse the relationship between the invariants and a key parameter in the application of Farkas' lemma, then solving the feasible domain of the key parameter from the inductive conditions, and finally illustrating that a finite number of values suffices for the key parameter w.r.t a tightness condition for the invariants to be generated.

Experimental results show that compared with previous approaches, our approach generates much more accurate affine inductive invariants over affine while loops from existing and new benchmarks within a few seconds, demonstrating the generality and efficiency of our approach.

1 Introduction

An *invariant* is a logical assertion at a certain program location that always holds whenever the program executes across that location. Invariants are indispensable parts of program analysis and formal verification, and thus the generation of

S. Shoham and Y. Vizel (Eds.): CAV 2022, LNCS 13371, pp. 257–281, 2022.
https://doi.org/10.1007/978-3-031-13185-1_13

invariants has been key to the proof and analysis of crucial properties like reachability [3,6,15], time complexity [9] and safety [2,32]. To ease program analysis and formal verification, there has been a long thread of research on approaches to automatic generation of invariants, including constraint solving [10,12,27], recurrence analysis [17,24,29,31], abstract interpretation [13,14], logical inference [18,19,38], dynamic analysis [33,39], and machine learning [20,23,44]. To guarantee that an assertion is indeed an invariant, the widely-adopted paradigm is to generate an *inductive invariant* that holds for the first execution and for every periodic execution to the particular program location [12,32]. In this work, we consider an important subclass of invariants called *numerical invariants* which are assertions over the numerical values taken by the program variables, and are closely related to many common vulnerabilities like integer overflow, buffer overflow, division by zero and array out-of-bound. More specifically, we consider *affine* inductive invariants in the form of an affine inequality over program variables, and focus on affine while loops that have affine loop guards (as a conjunction of affine inequalities) and affine updates for the program variables but do not have nested loops.

To automate the generation of affine inductive invariants, we adopt the *constraint-solving* based approach with three steps. First, it establishes a template with unknown parameters for the target invariants. Second, it collects constraints derived from the inductive conditions. Finally, it solves the unknown parameters to get the desired invariants. Prior work in this space [12,37] leverages Farkas' lemma to provide a sound and complete characterization for the inductive conditions and then generates the affine inductive invariants either by the complete approach of quantifier elimination [12] or through several heuristics [37]. Specifically, the StInG invariant generator [40] implements the approach in [37], and the InvGen invariant generator [22] integrates abstract interpretation as well as the approach in [37]. Furthermore, a recent effort [34] leverages eigenvalues and eigenvectors for inferring a restricted class of invariants. Finally, some recent work considers decidable logic fragments that directly verify properties of loops [4,11,28,30]. Compared with other approaches such as machine learning and dynamic analysis, constraint solving has a theoretical guarantee on the correctness and accuracy of the generated invariants, yet typically at the cost of higher runtime complexity.

The novelty of our approach lies in that it completely addresses the constraints derived from Farkas' lemma by matrix methods, thus ensuring both generality and efficiency. In detail, this paper makes the following contributions (due to the page limit, the current paper is abridged. The full version is available at [25]):

- For affine while loops with tautological guard, we prove that the affine inductive invariants are determined by the eigenvalues and eigenvectors of the matrices that describe variable updates in the loop body.
- For affine while loops whose loop guard is a conjunction of affine inequalities, we solve the affine inductive invariants by first deriving through matrix inverse a formula with a key parameter in the application of Farkas' lemma,

then solving the feasible domain of the key parameter from the inductive conditions, and finally showing that it suffices to choose a finite number of values for the key parameter if one imposes a tightness condition on the invariants.
– We generalize our results to affine while loops with non-deterministic updates and to bidirectional affine invariants. A continuity property on the invariants w.r.t. the key parameter is also proved for tackling the numerical issue arising from the computation of eigenvectors. Experimental results on existing benchmarks and new benchmarks arising from linear dynamical systems demonstrate the generality and efficiency of our approach.

1.1 Related Work

Constraint Solving. There have been several prior approaches [12,37] using constraint solving for invariant generation based on Farkas' lemma. Compared to the approach in [12] that uses quantifier elimination to solve the constraints from Farkas' lemma, our approach is more efficient since it only involves the matrix computation. Compared with [37] that uses several heuristics, our approach is more general and complete in addressing all the cases in affine invariant generation. While the approach in [34] also uses eigenvectors, it is restricted to the subclass of equality and convergent invariants. In contrast, our approach targets at general affine inductive invariants over affine while loops. Other prior work [4,11,28,30] considers to have a decidable logic for unnested affine while loops with tautological guard but no conditional branches. Compared with them, our approach handles general affine while loops and targets at invariant generation.

Abstract Interpretation. A long thread of research to infer inductive invariants is using *abstract interpretation* [1,7,22,35] framework which constructs sound approximations for program semantics. In a nutshell, it first establishes an abstract domain for the specific form of properties to be generated, and then performs fixed-point computation in the abstract domain. Abstract interpretation generates invariants whose precision depends on the abstract domain and abstract operators, except for rare special cases [21,37].

Recurrence Analysis. Another closely-related technique is *recurrence analysis* [8,17,24,29,31]. The main idea is transforming the problem of invariant generation into a recurrence relation problem and then solve the latter one. The main limitation of recurrence analysis is that it requires the underlying recurrence relation to have a closed-form solution. This requirement, unfortunately, does not hold for the general case of affine inductive invariants over affine while loops.

Logical Inference. Invariants could also be obtained through *logical inference*, such as abductive inference [16], Craig interpolation [18], ICE learning [19,43], random search [38], etc. These approaches, however, cannot provide any theoretical guarantee on the accuracy of the generated numerical invariants. In contrast, our approach essentially addresses this issue.

Dynamic Analysis. *Dynamic analysis* [33,39] has also been exploited to invariant generation. The major process is first to collect the execution traces of a particular program by running it multiple times, and then guess the invariants based on these traces. As indicated in its process, dynamic analysis provides no guarantee on the correctness or accuracy of the inferred invariants, yet still pays the price of running the program at a large amount of time.

Machine Learning. There is a recent trend of applying *machine learning* [20,23,44] to solve the invariant-generation problem. Such approaches first establish a (typically large) training set of data, then use training approaches such as neural networks to generate invariants. Compared to our approach, those approaches require a large training set, while still having no theoretical guarantee on the correctness or accuracy. Specifically, such approaches cannot produce specific numerical values (e.g., eigenvalues) that are required to handle some examples in this work.

2 Preliminaries

In this section, we specify the class of affine while loops and define the affine-invariant-generation problem over such loops. Throughout the paper, we use $V = \{x_1, ..., x_n\}$ to denote the set of program variables in an affine while loop; we abuse the notation V so that it also represents the current values (before the execution of the loop body) of the original variables in V, and use the primed variables $V' := \{x' \mid x \in V\}$ for the next values (after the execution of the loop body). Furthermore, we denote by $\mathbf{x} = [x_1, ..., x_n]^{\mathrm{T}}$ the vector variable that represents the current values of the program variables, and by $\mathbf{x}' = [x'_1, ..., x'_n]^{\mathrm{T}}$ the vector variable for the next values.

An *affine while loop* is a while loop without nested loops that has affine updates in each assignment statement and possibly multiple conditional branches in the loop body. To formally specify the syntax of it, we first define affine inequalities and assertions, program states and satisfaction relation between them as follows.

Affine Inequalities and Assertions. An *affine inequality* ϕ is an inequality of the form $\mathbf{c}^{\mathrm{T}} \cdot \mathbf{y} + d \leq 0$ where $\mathbf{c}$ is a real vector, $\mathbf{y}$ is a vector of real-valued variables and d is a real scalar. An *affine assertion* is a finite conjunction of affine inequalities. An affine assertion is *satisfiable* if it is true under some assignment of real values to its variables. Given an affine assertion ψ over vector variable $\mathbf{x}$, we denote by ψ' the affine assertion obtained by substituting $\mathbf{x}$ in ψ with its next-value variable $\mathbf{x}'$.

Program States. A *program state* $\mathbf{v}$ is a real vector $\mathbf{v} = [v_1, ..., v_n]^{\mathrm{T}}$ such that each v_i is a concrete value for the variable x_i (in the vector variable $\mathbf{x}$). We say that a program state $\mathbf{v}$ satisfies an affine inequality $\phi = \mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} + d \leq 0$, written as $\mathbf{v} \models \phi$, if it holds that $\mathbf{c}^{\mathrm{T}} \cdot \mathbf{v} + d \leq 0$. Likewise, $\mathbf{v}$ satisfies an affine assertion ψ if it satisfies every conjunctive affine inequality in ψ. Furthermore, given an affine

assertion ψ with both $\mathbf{x}$ and $\mathbf{x}'$, we say that two program states $\mathbf{v}, \mathbf{v}'$ satisfy ψ, written as $\mathbf{v}, \mathbf{v}' \models \psi$, if ψ is true when one substitutes $\mathbf{x}$ by $\mathbf{v}$ and $\mathbf{x}'$ by $\mathbf{v}'$.

We then illustrate the syntax of (unnested) affine while loops as follows.

Affine While Loops. We consider affine while loops that take the form:

$$
\begin{array}{l}
\textbf{initial condition}\ \ \theta : \mathbf{R} \cdot \mathbf{x} + \mathbf{f} \leq \mathbf{0} \\
\textbf{while}\ \ G : \mathbf{P} \cdot \mathbf{x} + \mathbf{q} \leq \mathbf{0}\ \ \textbf{do} \\
\quad \textbf{case}\ \ \psi_1 : \mathbf{T}_1 \cdot \mathbf{x} - \mathbf{T}'_1 \cdot \mathbf{x}' + \mathbf{b}_1 \leq \mathbf{0}\ \ (\tau_1)\,; \\
\qquad\qquad \vdots \\
\quad \textbf{case}\ \ \psi_k : \mathbf{T}_k \cdot \mathbf{x} - \mathbf{T}'_k \cdot \mathbf{x}' + \mathbf{b}_k \leq \mathbf{0}\ \ (\tau_k)\,; \\
\textbf{end}
\end{array}
\qquad (\dagger)
$$

where (i) θ is an affine assertion that specifies the initial condition for inputs and is given by the real matrix $\mathbf{R}$ and vector $\mathbf{f}$, (ii) G is an affine assertion serving as the loop guard given by the real matrix $\mathbf{P}$ and vector $\mathbf{q}$, and (iii) each ψ_j is an affine assertion that represents a conditional branch, with the relationship between the current-state vector $\mathbf{x}$ and the next-state vector $\mathbf{x}'$ given by the affine assertion $\tau_j := \mathbf{T}_j \cdot \mathbf{x} - \mathbf{T}'_j \cdot \mathbf{x}' + \mathbf{b}_j \leq \mathbf{0}$ with transition matrices $\mathbf{T}_j, \mathbf{T}'_j$ and vector $\mathbf{b}_j$. In this work, we always assume that the rows of $\mathbf{R}$ are linearly independent (this condition means that every variable x_i has one independent initial condition attached to it, which holds in most situations such as a fixed initial program state), such that $\mathbf{R}^{\mathrm{T}}$ is left invertible; we denote its left inverse as $(\mathbf{R}^{\mathrm{T}})^{-1}_{\mathrm{L}}$.

The execution of an affine while loop is as follows. First, the loop starts with an arbitrary initial program state $\mathbf{v}^*$ that satisfies the initial condition θ. Then in each loop iteration, the current program state $\mathbf{v}$ is checked against the loop guard G. In the case that $\mathbf{v} \models G$, the loop arbitrarily chooses a conditional branch ψ_j satisfying $\mathbf{v} \models \psi_j$, and sets the next program state $\mathbf{v}'$ non-deterministically such that $\mathbf{v}, \mathbf{v}' \models \tau_j$; the next program state $\mathbf{v}'$ is then set as the current program state. Otherwise (i.e., $\mathbf{v} \not\models G$), the loop halts immediately.

Now we define affine inductive invariants over affine while loops. Informally, an affine inductive invariant is an affine inequality satisfying the initiation and consecution conditions which mean that the inequality holds at the start of the loop (initiation) and is preserved under every iteration of the loop body (consecution).

Affine Inductive Invariants. An *affine inductive invariant* for an affine while loop (†) is an affine inequality Φ that satisfies the initiation and consecution conditions as follows:

- (**Initiation**) θ implies Φ, i.e., $\mathbf{v} \models \theta$ implies $\mathbf{v} \models \Phi$ for all program states $\mathbf{v}$;
- (**Consecution**) for all program states $\mathbf{v}, \mathbf{v}'$ and every ψ_j, τ_j $(1 \leq j \leq k)$ in (†), we have that $(\mathbf{v} \models G \wedge \mathbf{v} \models \Phi \wedge \mathbf{v}, \mathbf{v}' \models \tau_j) \Rightarrow \mathbf{v}' \models \Phi'$.

From the definition above, it can be observed that an affine inductive invariant is an invariant, in the sense that every program state traversed (as a current state

at the start or after every loop iteration) in some execution of the underlying affine while loop will satisfy the affine inductive invariant.

From now on, we abbreviate affine while loops as affine loops and affine inductive invariants as affine invariants.

Problem Statement. In this work, we study the problem of automatically generating affine invariants over affine loops. Our aim is to have a complete mathematical characterization on all such invariants and develop efficient algorithms for generating these invariants.

3 Affine Invariants via Farkas' Lemma

Affine invariant generation through Farkas' lemma is originally proposed in [12, 37]. Farkas' lemma is a fundamental result in the theory of linear inequalities that leads to a complete characterization for the affine invariants. Since our approach is based on Farkas' lemma, we present a detailed account on the approaches in [12,37], and point out the weakness of each of the approaches.

Theorem 1 (Farkas' Lemma). *Consider the following affine assertion S over real-valued variables $y_1, \ldots, y_n$:*

$$S : \begin{bmatrix} a_{11}y_1 + \ldots + a_{1n}y_n + b_1 \leq 0 \\ \vdots \\ a_{k1}y_1 + \ldots + a_{kn}y_n + b_k \leq 0 \end{bmatrix}$$

when S is satisfiable, it entails a given affine inequality

$$\phi : c_1y_1 + \ldots + c_ny_n + d \leq 0$$

if and only if there exist non-negative real numbers $\lambda_0, \ldots, \lambda_k$ such that (i) $c_j = \sum_{i=1}^{k} \lambda_i a_{ij}$ for $1 \leq j \leq n$ and (ii) $d = (\sum_{i=1}^{k} \lambda_i b_i) - \lambda_0$.

The application of Farkas' lemma can be visualized by a table form as follows:

$$\begin{array}{c|c} \lambda_0 & -1 \leq 0 \\ \lambda_1 & a_{11}y_1 + \ldots + a_{1n}y_n + b_1 \leq 0 \\ \vdots & \vdots \qquad\qquad \vdots \\ \lambda_k & a_{k1}y_1 + \ldots + a_{kn}y_n + b_k \leq 0 \end{array} \Big\} (S)$$
$$\hline \quad c_1y_1 + \ldots + c_ny_n + d \leq 0 \quad (\phi) \qquad (\ddagger)$$

The intuition of the table form above is that one first multiplies the λ_i's on the left to their corresponding affine inequalities (in the same row) on the right, and then sums these affine inequalities together to obtain the affine inequality at the bottom. In this paper, we will call the table form as *Farkas table*.

Given an affine loop as (†), the approaches in [12,37] first establish a template $\Phi : c_1x_1 + \ldots + c_nx_n + d \leq 0$ for an affine invariant where $c_1, \ldots, c_n, d$ are

the unknown coefficients. Second, they establish constraints for the unknown coefficients from the initiation and consecution conditions for an affine invariant, as follows.

Initiation. By Farkas' lemma, the initiation condition can be solved from the Farkas table (‡) with $S := \theta$ and $\phi := \Phi$:

$$\begin{array}{c|l} \lambda_0^{\mathrm{I}} & -1 \leq 0 \\ \boldsymbol{\lambda} & \mathbf{R} \cdot \mathbf{x} + \mathbf{f} \leq \mathbf{0}\ (\theta) \\ \hline & \mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} + d \leq 0\ (\Phi) \end{array} \qquad (\#)$$

Here we rephrase the affine inequalities in θ and Φ with the condensed matrix forms $\mathbf{R} \cdot \mathbf{x} + \mathbf{f} \leq \mathbf{0}$ and $\mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} + d \leq 0$; we also use $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_k]^{\mathrm{T}}$ to denote the non-negative parameters in the leftmost column of (‡).

Consecution. The consecution condition can be solved by handling each conditional branch (specified by τ_j, ψ_j in (†)) separately. By Farkas' lemma, we treat each conditional branch by the Farkas table (‡) with $S := \Phi \wedge G \wedge \tau_j$ and $\phi := \Phi'$:

$$\begin{array}{c|l} \mu & \mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} + d \leq 0\ (\Phi) \\ \lambda_0^{\mathrm{C}} & -1 \leq 0 \\ \boldsymbol{\xi} & \mathbf{P} \cdot \mathbf{x} + \mathbf{q} \leq \mathbf{0}\ (G) \\ \boldsymbol{\eta} & \mathbf{T}_j \cdot \mathbf{x} - \mathbf{T}'_j \cdot \mathbf{x}' + \mathbf{b}_j \leq \mathbf{0}\ (\tau_j) \\ \hline & \mathbf{c}^{\mathrm{T}} \cdot \mathbf{x}' + d \leq 0\ (\Phi') \end{array} \qquad (*)$$

Note that the Farkas table above contains quadratic constraints as we multiply an unknown non-negative parameter μ to the unknown invariant $\mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} + d \leq 0$ in the table. The Farkas tables for all conditional branches are grouped conjunctively together to represent the whole consecution condition.

The weakness of the approaches presented in [12,37] lies at the treatment of the quadratic constraints from the consecution condition. The approach in [12] addresses the quadratic constraints by quantifier elimination that guarantees the theoretical completeness but typically has high runtime complexity. The approach in [37] solves the quadratic constraints by several heuristics that guess possible values for the key parameter μ in $(*)$ which causes non-linearity, hence losing completeness. Our approach considers to address parameter μ through matrix-based methods (eigenvalues and eigenvectors, matrix inverse, etc.), which is capable of efficiently generating affine invariants (as compared with quantifier elimination in [12]) while still ensuring theoretical completeness (as compared with the heuristics in [37]).

4 Single-Branch Affine Loops with Deterministic Updates

For the sake of simplicity, we first consider the affine invariant generation for a simple class of affine loops where there are no conditional branches in the loop body and the updates of the next-value vector $\mathbf{x}'$ are deterministic.

Formally, an affine loop with deterministic updates and a single branch takes the following form:

$$\textbf{initial condition}\ \ \theta : \mathbf{R}\cdot\mathbf{x}+\mathbf{f}\le\mathbf{0}$$
$$\textbf{while}\ \ G\ \ \textbf{do}\ \ \mathbf{x}'=\mathbf{T}\cdot\mathbf{x}+\mathbf{b};\ \ \textbf{end}$$

For the loop above, we aim at *non-trivial* affine invariants, i.e., affine invariants $\mathbf{c}^{\mathrm{T}}\cdot\mathbf{x}+d\le 0$ with $\mathbf{c}\neq\mathbf{0}$. We summarize our results below.

1. When the loop guard is '**true**', there are only finitely many independent non-trivial invariants $\mathbf{c}^{\mathrm{T}}\cdot\mathbf{x}+d\le 0$ where $\mathbf{c}$ is an eigenvector of the transpose of the transition matrix $\mathbf{T}$.
2. When the loop guard is not a tautology, there can be infinitely many more non-trivial invariants $\mathbf{c}^{\mathrm{T}}\cdot\mathbf{x}+d\le 0$ with $\mathbf{c}$ given by a direct formula in μ; in this case we derive the *feasible domain* of μ and select finitely many optimal ones (which we call *tight choices*) among them.

In Sect. 4.1, we first derive the constraints from the initiation (#) and consecution (∗) conditions satisfied by the invariants. Then we solve these constraints for the tautological loop guard case in Sect. 4.2 and the single-constraint loop guard case in Sect. 4.3. Finally we generalize the results to the multi-constraint loop guard case in Sect. 4.4.

4.1 Derived Constraints from the Farkas Tables

We first derive the constraints from the Farkas tables as follows:

Initiation. Recall the Farkas table (#) for initiation. We first compare the coefficients of $\mathbf{x}$ above and below the horizontal line in (#), and obtain

$$\boldsymbol{\lambda}^{\mathrm{T}}\cdot\mathbf{R}=\mathbf{c}^{\mathrm{T}}\ \Rightarrow\ \mathbf{R}^{\mathrm{T}}\cdot\boldsymbol{\lambda}=\mathbf{c}. \tag{1}$$

Then by comparing the constant terms in (#), we have:

$$-\lambda_0^{\mathrm{I}}+\boldsymbol{\lambda}^{\mathrm{T}}\cdot\mathbf{f}=d\ \Rightarrow\ \mathbf{f}^{\mathrm{T}}\cdot\boldsymbol{\lambda}-d=\lambda_0^{\mathrm{I}}\ge 0. \tag{2}$$

Note that $\mathbf{R}^{\mathrm{T}}$ has left inverse $(\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1}$, thus constraint (1) is equivalent to $\boldsymbol{\lambda}=(\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1}\cdot\mathbf{c}$. Plugging it into (2) yields

$$\mathbf{f}^{\mathrm{T}}\cdot(\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1}\cdot\mathbf{c}-d=\lambda_0^{\mathrm{I}}\ge 0. \tag{3}$$

Consecution. The Farkas table (∗) for consecution in the case of single-branch affine loops with deterministic updates is as follows:

μ	$\mathbf{c}^{\mathrm{T}}\cdot\mathbf{x}$	$+\ d\le 0$	(Φ)
λ_0^{C}		$-\ 1\le 0$	
$\boldsymbol{\xi}$	$\mathbf{P}\cdot\mathbf{x}$	$+\ \mathbf{q}\le\mathbf{0}$	(G)
$\boldsymbol{\eta}$	$\mathbf{T}\cdot\mathbf{x}-\mathbf{x}'$	$+\ \mathbf{b}=\mathbf{0}$	(τ)
	$\mathbf{c}^{\mathrm{T}}\cdot\mathbf{x}'$	$+\ d\le 0$	(Φ')

Here the transition matrix $\mathbf{T}$ is a $n \times n$ square matrix, and $\mathbf{b}$ is a n-dimensional vector. Since τ contains only equalities, the components $\eta_1, ..., \eta_n$ of the vector parameter $\boldsymbol{\eta}$ do not have to be non-negative (while the components $\xi_1, ..., \xi_n$ of $\boldsymbol{\xi}$ and μ must be non-negative). In this table, by comparing the coefficients of $\mathbf{x}'$ above and below the horizontal line, we easily get $-\boldsymbol{\eta} = \mathbf{c}$. Then we substitute $\boldsymbol{\eta}$ by $-\mathbf{c}$ and compare the coefficients of $\mathbf{x}$ above and below the horizontal line. We get

$$\mu \cdot \mathbf{c}^{\mathrm{T}} + \boldsymbol{\xi}^{\mathrm{T}} \cdot \mathbf{P} - \mathbf{c}^{\mathrm{T}} \cdot \mathbf{T} = \mathbf{0}^{\mathrm{T}} \;\Rightarrow\; \mu \cdot \mathbf{c} - \mathbf{T}^{\mathrm{T}} \cdot \mathbf{c} + \mathbf{P}^{\mathrm{T}} \cdot \boldsymbol{\xi} = \mathbf{0}. \tag{4}$$

We also compare the constant terms and get

$$\mu \cdot d - \lambda_0^{\mathrm{C}} + \boldsymbol{\xi}^{\mathrm{T}} \cdot \mathbf{q} - \mathbf{c}^{\mathrm{T}} \cdot \mathbf{b} = d \;\Rightarrow\; (\mu - 1)d - \mathbf{b}^{\mathrm{T}} \cdot \mathbf{c} + \mathbf{q}^{\mathrm{T}} \cdot \boldsymbol{\xi} = \lambda_0^{\mathrm{C}} \geq 0. \tag{5}$$

The rest of this section is devoted to solving the invariants $\Phi : \mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} + d \leq 0$ which satisfy all constraints (1)–(5).

4.2 Loops with Tautological Guard

We first consider the simplest case where the loop guard is '**true**':

$$\begin{array}{l} \textbf{initial condition}\ \ \theta : \mathbf{R} \cdot \mathbf{x} + \mathbf{f} \leq \mathbf{0} \\ \textbf{while true do}\ \ \mathbf{x}' = \mathbf{T} \cdot \mathbf{x} + \mathbf{b};\ \ \textbf{end} \end{array} \tag{$\diamond$}$$

In order for completely solving the non-linear constraints, we take three steps:

1. choose the correct μ, thus turn the non-linear constraints into linear ones;
2. use linear algebra method to solve out the vector $\mathbf{c}$;
3. with μ and $\mathbf{c}$ known, find out the feasible domain of d and determine the optimal value of it. Here 'optimality' is defined by the fact that all invariants with other d's in this domain are implied by the invariant with the 'optimal' d.

Step 1 and Step 2. We address the values of $\mu, \mathbf{c}$ by eigenvalues and eigenvectors in the following proposition:

Proposition 1. *For any non-trivial invariant* $\mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} + d \leq 0$ *of the loop* ($\diamond$)*, we have that* $\mathbf{c}$ *must be an eigenvector of* $\mathbf{T}^{\mathrm{T}}$ *with a non-negative eigenvalue* μ*.*

Proof. Since the loop guard is a tautology, we take the parameter $\boldsymbol{\xi}$ to be $\mathbf{0}$ in (4):

$$\mu \cdot \mathbf{c} - \mathbf{T}^{\mathrm{T}} \cdot \mathbf{c} = \mathbf{0}.$$

It's obvious that μ must be a non-negative eigenvalue of $\mathbf{T}^{\mathrm{T}}$ and $\mathbf{c}$ is the corresponding eigenvector. □

Example 1. (Fibonacci numbers). Consider the sequence $\{s_n\}$ defined by initial condition $s_1 = s_2 = 1$ and recursive formula $s_{n+2} = s_{n+1} + s_n$ for $n \geq 1$. If we use variables (x_1, x_2) to represent (s_n, s_{n+1}), then the sequence can be written as a loop:

$$\textbf{initial condition } \theta : \mathbf{R} \cdot \mathbf{x} + \mathbf{f} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \mathbf{0}$$
$$\textbf{while true do } \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \mathbf{T} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \mathbf{b} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \mathbf{0}; \textbf{ end}$$

The eigenvalues of matrix $\mathbf{T}^{\mathrm{T}}$ are $\frac{1-\sqrt{5}}{2}, \frac{1+\sqrt{5}}{2}$; only the second one is non-negative. This eigenvalue $\mu = \frac{1+\sqrt{5}}{2}$ yields eigenvector $\mathbf{c} = [c_1, \frac{1+\sqrt{5}}{2} c_1]^{\mathrm{T}}$, here c_1 is a free variable, which could be fixed in the final form of the invariant. □

Step 3. After solving μ and $\mathbf{c}$, we illustrate the feasible domain of d and its optimal value by the following proposition:

Proposition 2. *For any μ and $\mathbf{c}$ given by Proposition 1, the feasible domain of d is an interval determined by the two conditions below:*

$$d \leq \mathbf{f}^{\mathrm{T}} \cdot (\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1} \cdot \mathbf{c} \quad \textit{and} \quad (\mu - 1)d \geq \mathbf{b}^{\mathrm{T}} \cdot \mathbf{c}.$$

If the above conditions have empty solution set, then no affine invariant is available from such μ and $\mathbf{c}$; otherwise, the optimal value of d falls in one of the two choices:

$$d = \mathbf{f}^{\mathrm{T}} \cdot (\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1} \cdot \mathbf{c} \quad \textit{or} \quad (\mu - 1)d = \mathbf{b}^{\mathrm{T}} \cdot \mathbf{c}.$$

Proof. Constraint (3) provides one condition for d:

$$\mathbf{f}^{\mathrm{T}} \cdot (\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1} \cdot \mathbf{c} - d = \lambda_0^{\mathrm{I}} \geq 0 \;\Rightarrow\; \mathbf{f}^{\mathrm{T}} \cdot (\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1} \cdot \mathbf{c} \geq d;$$

while constraint (5) with $\boldsymbol{\xi} = \mathbf{0}$ provides the other condition:

$$(\mu - 1)d - \mathbf{b}^{\mathrm{T}} \cdot \mathbf{c} = \lambda_0^{\mathrm{C}} \geq 0 \;\Rightarrow\; (\mu - 1)d \geq \mathbf{b}^{\mathrm{T}} \cdot \mathbf{c}.$$

To obtain the strongest inequality $\mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} + d \leq 0$, we need to take d to be either minimal or maximal value, i.e., some boundary point of its interval; thus the invariant with this d would imply all invariants with the same $\mathbf{c}$ and other d's in this interval. The boundary is achieved when one of the two conditions achieves the equality. □

Example 2 (Fibonacci, Part 2). We continue with Example 1. Recall that $\mu = \frac{1+\sqrt{5}}{2}$, $\mathbf{c} = [c_1, \frac{1+\sqrt{5}}{2} c_1]^{\mathrm{T}}$; in this case, constraints (3) (5) (with $\boldsymbol{\xi} = \mathbf{0}$) read $-\frac{3+\sqrt{5}}{2} c_1 \geq d$ and $\frac{-1+\sqrt{5}}{2} d \geq 0$, hence yield $0 \leq d \leq -\frac{3+\sqrt{5}}{2} c_1$. The free variable c_1 must be negative here, so we choose $c_1 = -2$ and thus $\mathbf{c} = [-2, -1 - \sqrt{5}]^{\mathrm{T}}$ and $0 \leq d \leq 3 + \sqrt{5}$; there are two boundary values $d = 0$ and $d = 3 + \sqrt{5}$, where $d = 3 + \sqrt{5}$ leads to the strongest invariant:

$$\mu = (1 + \sqrt{5})/2 : \; -2x_1 - (1 + \sqrt{5})x_2 + 3 + \sqrt{5} \leq 0. \qquad \square$$

4.3 Loops with Guard: Single-Constraint Case

Here we study the loops with non-tautological guard. First of all, the eigenvalue method of Sect. 4.2 applies to this case as well; thus for the rest of Sect. 4, we always assume that μ is not any eigenvalue of $\mathbf{T}$ (and $\mathbf{c}$ is not any eigenvector of $\mathbf{T}^{\mathrm{T}}$ either) and aim for other invariants than the ones from the eigenvectors.

Let us start with the case that the loop guard consists of only one affine inequality:

$$\begin{array}{l}\textbf{initial condition}\ \ \theta : \mathbf{R}\cdot\mathbf{x}+\mathbf{f}\leq\mathbf{0}\\ \textbf{while}\ \ \mathbf{p}^{\mathrm{T}}\cdot\mathbf{x}+q\leq 0\ \ \textbf{do}\ \ \mathbf{x}'=\mathbf{T}\cdot\mathbf{x}+\mathbf{b};\ \ \textbf{end}\end{array} \qquad (\diamond')$$

where $\mathbf{p}$ is a n-dimensional real vector and q is a real number.

We again take three steps to compute the invariants; these steps are different from the previous case:

1. we derive a formula to compute $\mathbf{c}$ in terms of μ; so for any non-negative real value μ, we get a corresponding $\mathbf{c}$;
2. however, not all μ's would produce invariants that satisfy all constraints (1)–(5). We will determine the feasible domain of μ that does so;
3. we will select finitely many μ's from its feasible domain which provide *tight invariants*; the meaning of *tightness* will be defined later. For every single μ, we will also determine the feasible domain of d and optimal value of it.

Step 1. We first establish the relationship between μ and $\mathbf{c}$ through the constraints. The initiation is still (1) (2) (3), while the consecution (4) (5) becomes:

$$\mu\cdot\mathbf{c}-\mathbf{T}^{\mathrm{T}}\cdot\mathbf{c}+\xi\cdot\mathbf{p}=\mathbf{0} \qquad (4')$$

$$(\mu-1)d-\mathbf{b}^{\mathrm{T}}\cdot\mathbf{c}+\xi\cdot q=\lambda_0^{\mathrm{C}}\geq 0 \qquad (5')$$

where the matrix $\mathbf{P}$ in (4) degenerates to vector $\mathbf{p}^{\mathrm{T}}$ and the vectors $\mathbf{q},\boldsymbol{\xi}$ in (5) both have just one component q,ξ here. Note that ξ is a non-negative parameter.

In contrast to Sect. 4.2, we assume that μ is not any eigenvalue of $\mathbf{T}$, and $\xi\neq 0$. For such μ, we have a new formula to compute $\mathbf{c}$:

Proposition 3. *For any non-trivial invariant* $\mathbf{c}^{\mathrm{T}}\cdot\mathbf{x}+d\leq 0$ *of the loop* $(\diamond')$*, we have that* $\mathbf{c}$ *is given by*

$$\mathbf{c}=\xi\cdot(\mathbf{T}^{\mathrm{T}}-\mu\cdot\mathbf{I})^{-1}\cdot\mathbf{p}\quad \textit{with}\quad \xi\geq 0 \qquad (6)$$

when μ *is fixed,* $\mathbf{c}$*'s with different* ξ*'s are proportional to each other and yield equivalent invariants.*

Proof. Since μ is not any eigenvalue of $\mathbf{T}$, the matrix $\mu\cdot\mathbf{I}-\mathbf{T}^{\mathrm{T}}$ is invertible; thus $(4')$ is equivalent to

$$(\mu\cdot\mathbf{I}-\mathbf{T}^{\mathrm{T}})\cdot\mathbf{c}=-\xi\cdot\mathbf{p}\ \Rightarrow\ \mathbf{c}=\xi\cdot(\mathbf{T}^{\mathrm{T}}-\mu\cdot\mathbf{I})^{-1}\cdot\mathbf{p}. \qquad \square$$

Example 3 (Fibonacci, Part 3). We add a loop guard $x_1 \leq 10$ to Example 1:

$$\textbf{initial condition } \theta : \mathbf{R} \cdot \mathbf{x} + \mathbf{f} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \mathbf{0}$$
$$\textbf{while } \mathbf{p}^{\mathrm{T}} \cdot \mathbf{x} + q = [1, 0] \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - 10 \leq 0 \textbf{ do}$$
$$\begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \mathbf{T} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \mathbf{b} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \mathbf{0}; \textbf{ end}$$

and search for more invariants. The formula (6) here reads

$$\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \frac{\xi}{\mu^2 - \mu - 1} \begin{bmatrix} 1 - \mu & -1 \\ -1 & -\mu \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{\xi}{\mu^2 - \mu - 1} \begin{bmatrix} 1 - \mu \\ -1 \end{bmatrix}. \qquad \square$$

Step 2. With formula (6) in hand, every non-negative value μ would give us a vector $\mathbf{c}$; the next step is to find such μ's that (1) (2) (3) (5′) are all satisfied. We call this set the *feasible domain* of μ.

Notice that (3) and (5′) are two inequalities both containing d. When the value of μ changes, there is a possibility that (3) and (5′) conflict each other, hence make no invariant available. So the feasible domain consists of such μ's that make the two inequalities compatible with each other:

Proposition 4. *For the loop (⋄′), any feasible μ falls in $[0, 1) \cup \big(K \cap [1, +\infty)\big)$, where K is the solution set to the following rational inequality of μ (which we call 'compatibility condition'):*

$$\mathbf{b}^{\mathrm{T}} \cdot (\mathbf{T}^{\mathrm{T}} - \mu \cdot \mathbf{I})^{-1} \cdot \mathbf{p} - q \leq (\mu - 1)\mathbf{f}^{\mathrm{T}} \cdot (\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1} (\mathbf{T}^{\mathrm{T}} - \mu \cdot \mathbf{I})^{-1} \cdot \mathbf{p}. \tag{7}$$

Proof. We multiply $(\mu - 1)$ on both sides of (3) and get

$$(\mu - 1)\mathbf{f}^{\mathrm{T}} \cdot (\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1} \cdot \mathbf{c} \leq (\mu - 1)d \quad \text{when} \quad 0 \leq \mu < 1 \tag{3′}$$
$$(\mu - 1)\mathbf{f}^{\mathrm{T}} \cdot (\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1} \cdot \mathbf{c} \geq (\mu - 1)d \quad \text{when} \quad \mu \geq 1 \tag{3″}$$

compare them with (5′), we see: (3′) (5′) would not conflict each other because they are both about $(\mu - 1)d$ being 'larger' than something. However, (3″) (5′) are two inequalities of opposite directions, they together must satisfy

$$\mathbf{b}^{\mathrm{T}} \cdot \mathbf{c} - \xi \cdot q \leq (\mu - 1)d \leq (\mu - 1)\mathbf{f}^{\mathrm{T}} \cdot (\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1} \cdot \mathbf{c}$$

to be compatible. Substitute $\mathbf{c}$ by (6) in the above inequality and cancel out $\xi > 0$, we obtain the desired inequality:

$$\mathbf{b}^{\mathrm{T}} \cdot (\mathbf{T}^{\mathrm{T}} - \mu \cdot \mathrm{I})^{-1} \cdot \mathbf{p} - q \leq (\mu - 1)\mathbf{f}^{\mathrm{T}} \cdot (\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1} (\mathbf{T}^{\mathrm{T}} - \mu \cdot \mathbf{I})^{-1} \cdot \mathbf{p}.$$

Every μ from $[0, 1)$ and $K \cap [1, +\infty)$ would lead to non-trivial invariant satisfying all constraints (1) (2) (3) (4′) (5′). $\square$

Example 4 (Fibonacci, Part 4). Let us compute the feasible domain of μ for Example 3. Inequality (5′) is $(\mu-1)d \geq 10\xi$; inequality (3″) is

$$(\mu-1)[-1,-1]\cdot\begin{bmatrix}1&0\\0&1\end{bmatrix}\cdot\mathbf{c}=\frac{\xi(\mu-1)\mu}{\mu^2-\mu-1}\geq(\mu-1)d\quad(\text{when }\mu\geq 1).$$

We combine them to form the compatibility condition (7) as

$$10\leq\frac{(\mu-1)\mu}{\mu^2-\mu-1}\;\Rightarrow\;0\leq-\frac{9(\mu-\frac{5}{3})(\mu+\frac{2}{3})}{(\mu-\frac{1-\sqrt{5}}{2})(\mu-\frac{1+\sqrt{5}}{2})}\quad(\text{when }\mu\geq 1).$$

The solution domain of it is $(\frac{1+\sqrt{5}}{2},\frac{5}{3}]$. Thus by Proposition 4, the feasible domain of μ is $[0,1)\cup(\frac{1+\sqrt{5}}{2},\frac{5}{3}]$. □

Step 3. Proposition 4 provides us with a continuum of candidates for μ, thus produces infinitely many legitimate invariants. We want to find a basis consisting of finitely many invariants, such that all invariants are non-negative linear combinations of the basis; however, this idea does not work out, where the reason is explained thoroughly in the full version of this paper [25, Appendix A.1 and A.2]. Instead, we impose a weaker form of optimality called *tightness* coming from the equality cases of constraints (3) (5′):

$$\mathbf{f}^{\mathrm{T}}\cdot(\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1}\cdot\mathbf{c}-d=\lambda_0^{\mathrm{I}}=0$$
$$(\mu-1)d-\mathbf{b}^{\mathrm{T}}\cdot\mathbf{c}+\xi\cdot q=\lambda_0^{\mathrm{C}}=0$$

we call an invariant *tight* and the corresponding μ as *tight choice* when both equalities are achieved:

- $\lambda_0^{\mathrm{I}}=0$: The invariant is tight at the initial state, i.e., the invariant reaches equality at the initial state;
- $\lambda_0^{\mathrm{C}}=0$: The invariant stays as close to being tight as much at later iterations.

The non-tight choices could be kept as back-up for invariant generation. The tight choices are characterized by the following proposition:

Proposition 5. *For the loop (⋄′), the tight choices of μ consist of 0 and the positive real roots of the following rational equation:*

$$\mathbf{b}^{\mathrm{T}}\cdot(\mathbf{T}^{\mathrm{T}}-\mu\cdot\mathbf{I})^{-1}\cdot\mathbf{p}-q=(\mu-1)\mathbf{f}^{\mathrm{T}}\cdot(\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1}(\mathbf{T}^{\mathrm{T}}-\mu\cdot\mathbf{I})^{-1}\cdot\mathbf{p}.\qquad(8)$$

Note that these roots are also the boundary points of the intervals in K defined in Proposition 4.

Proof. Recall Proposition 2, constraints (3) (5) form the two boundaries of the domain of d, which can not be achieved simultaneously in the case of loops with tautological guard. Nevertheless, in the case of loops with guard, we have an extra freedom on μ which allows us to set $\lambda_0^{\mathrm{I}}=\lambda_0^{\mathrm{C}}=0$:

$$\begin{aligned}&\mathbf{f}^{\mathrm{T}}\cdot(\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1}\cdot\mathbf{c}=d\;\wedge\;(\mu-1)d=\mathbf{b}^{\mathrm{T}}\cdot\mathbf{c}-\xi\cdot q\\ \Rightarrow\quad&\mathbf{b}^{\mathrm{T}}\cdot(\mathbf{T}^{\mathrm{T}}-\mu\cdot\mathbf{I})^{-1}\cdot\mathbf{p}-q=(\mu-1)\mathbf{f}^{\mathrm{T}}\cdot(\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1}(\mathbf{T}^{\mathrm{T}}-\mu\cdot\mathbf{I})^{-1}\cdot\mathbf{p}.\end{aligned}$$

Equation (8) is just the case that (7) achieves the equality, hence is a rational equation of μ with finite number of roots. These roots are also the boundary points of K since K is the solution domain to (7). Besides the roots of (8), $\mu = 0$ is also a boundary point of the feasible domain; its corresponding invariant reflects the feature of the loop guard itself. Thus we add it into the list of tight choices. □

With μ determined and $\mathbf{c}$ fixed up to a scaling factor, the last thing remains is to determine the optimal d. The strategy here is similar to Proposition 2:

Proposition 6. *Suppose μ is from the feasible domain and $\mathbf{c}$ is given by Proposition 3. Then the optimal value of d is determined by one of the two choices below:*

$$\mathbf{b}^{\mathrm{T}} \cdot \mathbf{c} - \xi \cdot q = (\mu - 1)d \quad \textit{or} \quad \mathbf{f}^{\mathrm{T}} \cdot (\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1} \cdot \mathbf{c} = d.$$

The proof is omitted here and can be found in our full version [25].

Example 5 (Fibonacci, Part 5). Remember that

$$\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \frac{\xi}{\mu^2 - \mu - 1} \begin{bmatrix} 1 - \mu \\ -1 \end{bmatrix} \text{ and the feasible domain of } \mu \text{ is } [0, 1) \cup (\frac{1 + \sqrt{5}}{2}, \frac{5}{3}].$$

We compute the tight choices of μ and tight invariants. The equation (8) here is

$$0 = \frac{-9\mu^2 + 9\mu + 10}{\mu^2 - \mu - 1} = -\frac{9(\mu - \frac{5}{3})(\mu + \frac{2}{3})}{(\mu - \frac{1-\sqrt{5}}{2})(\mu - \frac{1+\sqrt{5}}{2})}$$

which has only one positive root $\mu = \frac{5}{3}$. By Proposition 5 and Proposition 6, We get two invariants:

$$\begin{aligned} \mu = 0 &: \ -x_1 + x_2 - 10 \leq 0; \\ \mu = 5/3 &: \ -2x_1 - 3x_2 + 5 \leq 0. \end{aligned}$$

□

4.4 Loops with Guard: Multi-constraint Case

After settling the single-constraint loop guard case, we consider the more general loop guard which contains the conjunction of multiple affine constraints:

$$\begin{array}{l} \textbf{initial condition } \ \theta : \mathbf{R} \cdot \mathbf{x} + \mathbf{f} \leq \mathbf{0} \\ \textbf{while } \ \mathbf{P} \cdot \mathbf{x} + \mathbf{q} \leq \mathbf{0} \ \textbf{ do } \ \mathbf{x}' = \mathbf{T} \cdot \mathbf{x} + \mathbf{b}; \ \textbf{ end} \end{array} \qquad (\diamond'')$$

where the loop guard $\mathbf{P} \cdot \mathbf{x} + \mathbf{q} \leq \mathbf{0}$ contains m affine inequalities.

We can easily generalize the results of Sect. 4.3 to this case. First of all, we generalize Proposition 3: one simply needs to modify the formula (6) into

$$\mathbf{c} = (\mathbf{T}^{\mathrm{T}} - \mu \cdot \mathbf{I})^{-1} \mathbf{P}^{\mathrm{T}} \cdot \boldsymbol{\xi} \quad \text{with} \quad \boldsymbol{\xi} \geq \mathbf{0} \qquad (6')$$

here $\boldsymbol{\xi}$ is a free non-negative m-dimensional vector parameter. With a fixed μ, we take $\boldsymbol{\xi}$ to traverse all vectors in the standard basis $\{\mathbf{e}_1, ..., \mathbf{e}_m\}$ to get m conjunctive invariants.

Next, we generalize Proposition 4 which describes the feasible domain of μ:

Proposition 7. *For the loop (◇″), the feasible domain of* μ *is* $[0,1) \cup \big(\overline{K} \cap [1,+\infty)\big)$*, where* $\overline{K}$ *is the solution set to the following generalized compatibility condition:*

$$\mathbf{b}^{\mathrm{T}} \cdot \mathbf{c} - \mathbf{q}^{\mathrm{T}} \cdot \boldsymbol{\xi} \leq (\mu - 1)d \leq (\mu - 1)\mathbf{f}^{\mathrm{T}} \cdot (\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1} \cdot \mathbf{c}$$

substitute $\mathbf{c}$ *by (6′) and take* $\boldsymbol{\xi}$ *to traverse all vectors in the standard basis (in order for all constraints in the loop guard to be satisfied by the invariant), we have the above condition completely decoded as* m *conjunctive inequalities:*

$$\begin{aligned} &\mathbf{u}(\mu) \coloneqq \mathbf{b}^{\mathrm{T}} \cdot (\mathbf{T}^{\mathrm{T}} - \mu \cdot \mathbf{I})^{-1}\mathbf{P}^{\mathrm{T}} - \mathbf{q}^{\mathrm{T}} \\ \leq\ &\mathbf{w}(\mu) \coloneqq (\mu - 1)\mathbf{f}^{\mathrm{T}} \cdot (\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1}(\mathbf{T}^{\mathrm{T}} - \mu \cdot \mathbf{I})^{-1}\mathbf{P}^{\mathrm{T}} \end{aligned} \tag{7′}$$

where $\mathbf{u}(\mu), \mathbf{w}(\mu)$ *are two* m*-dimensional vector functions in* μ*. The meaning of (7′) is that the* i*-th component of* $\mathbf{u}(\mu)$ *is no larger than the* i*-th component of* $\mathbf{w}(\mu)$ *for all* $1 \leq i \leq m$*; when* $m = 1$*, it goes back to (7).*

At last, we consider the tight choices of μ. The first idea comes up to mind is to repeat Proposition 5: setting $\lambda_0^{\mathrm{I}} = \lambda_0^{\mathrm{C}} = 0$ for arbitary $\boldsymbol{\xi}$ such that the generalized compatibility condition achieves equality, i.e., $\mathbf{u}(\mu) = \mathbf{w}(\mu)$; however, this is the conjunction of m rational equations and probably contains no solution.

Thus we use a different idea: recall that in the single-constraint case, the tight choices are also the (positive) boundary points of K along with 0; so we adopt this property as the definition in the multi-constraint case:

Definition 1. *For the loop (◇″), the tight choices of* μ *consist of* 0 *and the (positive) boundary points of the domain* $\overline{K}$ *defined in Proposition 7.*

The generalized compatibility condition (7′) contains m inequalities; at each (positive) boundary point of $\overline{K}$, at least one inequality achieves equality and all other inequalities are satisfied (equivalently, $\lambda_0^{\mathrm{I}} = \lambda_0^{\mathrm{C}} = 0$ is achieved for at least one non-trivial evaluation of the free vector parameter $\boldsymbol{\xi}$). This is indeed a natural generalization of Proposition 5.

Example 6. We consider the loop:

$$\begin{aligned} &\textbf{initial condition } \theta : \mathbf{R} \cdot \mathbf{x} + \mathbf{f} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \mathbf{0} \\ &\textbf{while } \mathbf{P} \cdot \mathbf{x} + \mathbf{q} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -10 \\ -5 \end{bmatrix} \leq \mathbf{0} \textbf{ do} \\ &\quad \begin{bmatrix} x_1' \\ x_2' \end{bmatrix} = \mathbf{T} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \mathbf{b} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \end{bmatrix}; \textbf{ end} \end{aligned}$$

There is one eigenvalue $\mu = 1$ with geometric multiplicity 2; we solve three independent invariants from it:

$$x_1 + x_2 - 2 \leq 0,\ x_1 + x_2 - 2 \geq 0;\ -x_1 + x_2 \leq 0.$$

Next we find out the other invariants from tight μ's. In this case (7′) read $\frac{11-10\mu}{1-\mu} \leq 1 \ \wedge \ \frac{6-5\mu}{1-\mu} \leq -1 \quad (\text{when } \mu > 1)$. Then $\overline{K} = (1, \frac{10}{9}] \cap (1, \frac{7}{6}] = (1, \frac{10}{9}]$ and the feasible domain of μ is $[0, 1) \cup (1, \frac{10}{9}]$. The tight choices are 0, $\frac{10}{9}$ (taking $\boldsymbol{\xi}$ to be $[1, 0]^{\mathrm{T}}, [0, 1]^{\mathrm{T}}$ respectively yields the two conjunctive invariants for each μ):

$$\begin{aligned} \mu = 0 &: x_1 - 10 \leq 0 \ \wedge \ -x_2 - 5 \leq 0; \\ \mu = 10/9 &: -x_1 + 1 \leq 0 \ \wedge \ x_2 - 1 \leq 0. \end{aligned}$$ □

5 Generalizations

In this section, we extend our theory developed in Sect. 4 in two directions. For one direction, we consider the invariants $\mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} + d \leq 0$ for the affine loops in the general form (†): we will derive the relationship of μ and $\mathbf{c}$, as well as the feasible domain and tight choices of μ. For the other direction, we stick to the single-branch affine loops with deterministic updates and tautological guard (⋄), yet generalize the invariants to bidirectional-inequality form $d_1 \leq \mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} \leq d_2$; we will apply eigenvalue method to this case for solving the invariants. At the end of the section, we also give a brief discussion on some other possible generalizations.

5.1 Affine Loops with Non-deterministic Updates

In Sect. 4, we handled the loops with deterministic updates; here we generalize the results to the non-deterministic case in the form of (†). We focus on the single-branch loops here, because the multi-branch ones can be handled similarly by taking the conjunction of all branches, as illustrated in the full version of this paper [25, Appendix A.3].

$$\begin{array}{l} \textbf{initial condition} \ \ \theta : \mathbf{R} \cdot \mathbf{x} + \mathbf{f} \leq \mathbf{0} \\ \textbf{while} \ \ \mathbf{P} \cdot \mathbf{x} + \mathbf{q} \leq \mathbf{0} \ \ \textbf{do} \ \ \mathbf{T} \cdot \mathbf{x} - \mathbf{T}' \cdot \mathbf{x}' + \mathbf{b} \leq \mathbf{0}; \ \ \textbf{end} \end{array} \qquad (\dagger')$$

For this general form, the initiation constraints are still (1) (2) (3), while the consecution constraints from Farkas table (∗) are

$$\mu \cdot \mathbf{c} + \mathbf{P}^{\mathrm{T}} \cdot \boldsymbol{\xi} + \mathbf{T}^{\mathrm{T}} \cdot \boldsymbol{\eta} = \mathbf{0} \qquad (9)$$

$$-(\mathbf{T}')^{\mathrm{T}} \cdot \boldsymbol{\eta} = \mathbf{c} \qquad (10)$$

$$(\mu - 1)d + \mathbf{q}^{\mathrm{T}} \cdot \boldsymbol{\xi} + \mathbf{b}^{\mathrm{T}} \cdot \boldsymbol{\eta} = \lambda_0^{\mathrm{C}} \geq 0 \qquad (11)$$

with $\boldsymbol{\xi}, \boldsymbol{\eta} \geq \mathbf{0}$. The relationship of $\mathbf{c}$ and $\boldsymbol{\eta}$ is given by (10); plugging it into (9) yield

$$\left(\mathbf{T}^{\mathrm{T}} - \mu \cdot (\mathbf{T}')^{\mathrm{T}}\right) \cdot \boldsymbol{\eta} + \mathbf{P}^{\mathrm{T}} \cdot \boldsymbol{\xi} = \mathbf{0}. \qquad (9')$$

Hence for any non-trivial invariant $\mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} + d \leq 0$ of this loop (†′), we have $\mathbf{c} = -(\mathbf{T}')^{\mathrm{T}} \cdot \boldsymbol{\eta}$, where $\boldsymbol{\eta}$ is characterized differently in the following three cases:

1. $\mathbf{T}$ and $\mathbf{T}'$ are square matrices and the loop guard is '**true**'. In this case, we take $\boldsymbol{\xi} = \mathbf{0}$ in (9′) and easily see that μ must be a root of $\det\left(\mathbf{T}^{\mathrm{T}} - \mu \cdot (\mathbf{T}')^{\mathrm{T}}\right) = 0$ and $\boldsymbol{\eta}$ is a kernel vector of the matrix $\mathbf{T}^{\mathrm{T}} - \mu \cdot (\mathbf{T}')^{\mathrm{T}}$.
2. $\mathbf{T}$ and $\mathbf{T}'$ are square matrices and the loop guard is non-tautological. In this case, we set μ to be values other than the roots of $\det\left(\mathbf{T}^{\mathrm{T}} - \mu \cdot (\mathbf{T}')^{\mathrm{T}}\right) = 0$, thus the inverse matrix $\left(\mathbf{T}^{\mathrm{T}} - \mu \cdot (\mathbf{T}')^{\mathrm{T}}\right)^{-1}$ exists; we multiply it on (9′) and get that $\boldsymbol{\eta}(\mu) = -\left(\mathbf{T}^{\mathrm{T}} - \mu \cdot (\mathbf{T}')^{\mathrm{T}}\right)^{-1}\mathbf{P}^{\mathrm{T}} \cdot \boldsymbol{\xi}$.
3. Neither $\mathbf{T}$ nor $\mathbf{T}'$ is square matrix. In this case, we need to use Gaussian elimination method (with parameters) to solve (9′). By linear algebra, the solution $\boldsymbol{\eta}(\mu)$ would contain 'homogeneous term' (which does not involve $\boldsymbol{\xi}$ but possibly some free variables $\overline{\boldsymbol{\eta}} = [\eta_1, ..., \eta_l]^{\mathrm{T}}$) and 'non-homogeneous term' (which contains $\boldsymbol{\xi}$ linearly). Thus $\boldsymbol{\eta}(\mu)$ could be written in parametric vector form as $\mathbf{M}(\mu) \cdot \overline{\boldsymbol{\eta}} + \mathbf{N}(\mu) \cdot \boldsymbol{\xi}$, where $\mathbf{M}(\mu), \mathbf{N}(\mu)$ are matrix functions only in μ.

For Case 2 and Case 3, we have a continuum of candidates for μ. The feasible domain of μ is given by $\left([0, 1) \cup \left(\widetilde{K} \cap [1, +\infty)\right)\right) \cap J$, where $\widetilde{K}$ is the solution set to the following compatibility condition (obtained by combining constraints (3″) (11)):

$$\mathbf{b}^{\mathrm{T}} \cdot \boldsymbol{\eta}(\mu) + \mathbf{q}^{\mathrm{T}} \cdot \boldsymbol{\xi} \geq (\mu - 1)\mathbf{f}^{\mathrm{T}} \cdot (\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1}(\mathbf{T}')^{\mathrm{T}} \cdot \boldsymbol{\eta}(\mu)$$

and J is the solution set to constraints $\boldsymbol{\eta}(\mu) \geq \mathbf{0}$. Here both $\overline{\boldsymbol{\eta}}$ and $\boldsymbol{\xi}$ as free non-negative vector parameters are taken to traverse all standard basis vectors, just in the same way as Proposition 7. The tight choices of μ consists of 0 and the positive boundary points of $\widetilde{K} \cap J$, in the same sense as Definition 1.

5.2 An Extension to Bidirectional Affine Invariants

Here we restrict ourselves to single-branch affine loops with deterministic updates and tautological loop guard (⋄), but aim for the invariants of bidirectional-inequality form $d_1 \leq \mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} \leq d_2$. This is actually the conjunction of two affine inequalities: $\Phi_1 : -\mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} + d_1 \leq 0 \wedge \Phi_2 : \mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} - d_2 \leq 0$. We have the following proposition:

Proposition 8. *For any bidirectional invariant $d_1 \leq \mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} \leq d_2$ of the loop (⋄), we have that $\mathbf{c}$ must be an eigenvector of $\mathbf{T}^{\mathrm{T}}$ with a negative eigenvalue.*

Proof. We can easily write down the initiation condition: $\theta \models (\Phi_1 \wedge \Phi_2)$ and the corresponding constraints (with $\boldsymbol{\lambda}$, $\widetilde{\boldsymbol{\lambda}}$ being two different vector parameters):

$$\mathbf{R}^{\mathrm{T}} \cdot \boldsymbol{\lambda} = \mathbf{c}, \quad \mathbf{f}^{\mathrm{T}} \cdot \boldsymbol{\lambda} + d_2 = \lambda_0^{\mathrm{I}} \geq 0; \quad \mathbf{R}^{\mathrm{T}} \cdot \widetilde{\boldsymbol{\lambda}} = -\mathbf{c}, \quad \mathbf{f}^{\mathrm{T}} \cdot \widetilde{\boldsymbol{\lambda}} - d_1 = \widetilde{\lambda}_0^{\mathrm{I}} \geq 0.$$

However, there are two possible ways to propose the consecution condition:

$$(\Phi_1 \wedge \tau \models \Phi_1' \text{ and } \Phi_2 \wedge \tau \models \Phi_2') \quad \textbf{or} \quad (\Phi_1 \wedge \tau \models \Phi_2' \text{ and } \Phi_2 \wedge \tau \models \Phi_1')$$

If we choose the first one, there will be nothing different from the things we did in Sect. 4.2. Thus we choose the second one: making the two inequalities induct each other. Hence the Farkas tables are

$$\begin{array}{c|l} \mu & -\mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} \qquad\qquad + d_1 \leq 0 \ (\Phi_1) \\ \lambda_0^{\mathrm{C}} & \qquad\qquad\qquad\quad - \ 1 \leq 0 \\ -\mathbf{c} & \mathbf{T} \cdot \mathbf{x} - \mathbf{x}' + \mathbf{b} = \mathbf{0} \ \ (\tau) \\ \hline & \mathbf{c}^{\mathrm{T}} \cdot \mathbf{x}' - d_2 \leq 0 \ (\Phi_2') \end{array} \qquad \begin{array}{c|l} \widetilde{\mu} & \mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} \qquad\qquad - d_2 \leq 0 \ (\Phi_2) \\ \widetilde{\lambda}_0^{\mathrm{C}} & \qquad\qquad\qquad\quad - \ 1 \leq 0 \\ \mathbf{c} & \mathbf{T} \cdot \mathbf{x} - \mathbf{x}' + \mathbf{b} = \mathbf{0} \ \ (\tau) \\ \hline & -\mathbf{c}^{\mathrm{T}} \cdot \mathbf{x}' + d_1 \leq 0 \ (\Phi_1') \end{array}$$

We write out the constraints of consecution:

$$-\mu \cdot \mathbf{c} = \mathbf{T}^{\mathrm{T}} \cdot \mathbf{c} = -\widetilde{\mu} \cdot \mathbf{c} \tag{12}$$
$$\mu \cdot d_1 + d_2 - \mathbf{b}^{\mathrm{T}} \cdot \mathbf{c} = \lambda_0^{\mathrm{C}} \geq 0, \quad -\widetilde{\mu} \cdot d_2 - d_1 + \mathbf{b}^{\mathrm{T}} \cdot \mathbf{c} = \widetilde{\lambda}_0^{\mathrm{C}} \geq 0$$

the proposition is verified by (12) since $\mu, \widetilde{\mu} \geq 0$. □

Example 7 (Fibonacci, Part 6). Recall that in this example we have a negative eigenvalue $\frac{1-\sqrt{5}}{2}$. It yields the eigenvector $\mathbf{c} = [c_1, \frac{1-\sqrt{5}}{2}c_1]^{\mathrm{T}}$. The other constraints are computed as:

$$-(3-\sqrt{5})c_1/2 + d_2 = \lambda_0^{\mathrm{I}} \geq 0, \quad (3-\sqrt{5})c_1/2 - d_1 = \widetilde{\lambda}_0^{\mathrm{I}} \geq 0.$$
$$-(1-\sqrt{5})d_1/2 + d_2 = \lambda_0^{\mathrm{C}} \geq 0, \quad (1-\sqrt{5})d_2/2 - d_1 = \widetilde{\lambda}_0^{\mathrm{C}} \geq 0.$$

If we choose $c_1 = 2, \lambda_0^{\mathrm{I}} = 0 = \widetilde{\lambda}_0^{\mathrm{C}}$ (or $c_1 = -2, \widetilde{\lambda}_0^{\mathrm{I}} = 0 = \lambda_0^{\mathrm{C}}$), we get an invariant

$$\mu = |(1-\sqrt{5})/2| : \ 2(2-\sqrt{5}) \leq 2x_1 + (1-\sqrt{5})x_2 \leq 3-\sqrt{5}$$

which reflects the 'golden ratio' property of the Fibonacci numbers. □

Remark 1. The generalizations for bidirectional affine invariants to the loops with non-tautological guard or multiple branches are practicable but with some restrictions. The main restriction lies at the point that we need to assume the affine loop guard to also be bidirectional to make our approach for bidirectional affine invariants work. The issue of multiple branches is not critical as the bidirectional invariants can be derived in almost the same way as single-inequality invariants (illustrated in full version [25, Appendix A.3]), with the only difference at the adaption to bidirectional inequalities.

5.3 Other Possible Generalizations

Integer-valued Variables. One direction is to transfer some of the results for affine loops over real-valued variables to those over integer-valued variables. Our approach is based on Farkas' lemma which is dedicated to real-valued variables, thus can only provide a sound but not exact treatment for integer-valued variables. An exact treatment for integer-valued variables would require Presburger arithmetics [16], rather than Farkas' lemma.

Strict-inequality Invariants. We handle the non-strict-inequality affine invariants in this work. It's natural to consider the affine invariants of the strict-inequality form. For strict inequalities, we could utilize an extended version of Farkas' lemma in [6, Corollary 1], so that strict inequalities can be generated by either relaxing the non-strict ones obtained from our method or restricting the μ value to be positive. Since Motzkin transposition theorem is a standard theorem for handling strict inequalities, we believe that Motzkin transposition theorem can also achieve similar results, but may require more tedious manipulations.

6 Approximation of Eigenvectors through Continuity

In Sect. 4.2 and Sect. 5.2, we need to solve the characteristic polynomial of the transition matrix to get eigenvalues; while general polynomials with degree ≥ 5 do not have algebraic solution formula due to Abel-Ruffini theorem. We can develop a number sequence $\{\lambda_i\}$ to approximate the eigenvalue λ through root-finding algorithms; however, we cannot approximate the eigenvector of λ by solving the kernel of $\mathbf{T}^{\mathrm{T}} - \lambda_i \cdot \mathbf{I}$ since it has trivial kernel. In the case of dimensions ≥ 5, i.e., when an explicit formula for eigenvalues is unavailable, we introduce an approximation method of the eigenvectors through a continuity property of the invariants:

Continuity of Invariants w.r.t. μ. In Sect. 4, we have shown that for any invariant $\mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} + d \leq 0$ of single-branch affine loops with deterministic updates, the relationship of $\mathbf{c}$ and μ is given in two ways:

$$\mathbf{c} = \begin{cases} \text{kernel vector of } \mathbf{T}^{\mathrm{T}} - \mu \cdot \mathbf{I} & \text{when } \det(\mathbf{T}^{\mathrm{T}} - \mu \cdot \mathbf{I}) = 0 \\ (\mathbf{T}^{\mathrm{T}} - \mu \cdot \mathbf{I})^{-1} \cdot \mathbf{z} & \text{when } \det(\mathbf{T}^{\mathrm{T}} - \mu \cdot \mathbf{I}) \neq 0 \end{cases}$$

with $\mathbf{z} = \mathbf{P}^{\mathrm{T}} \cdot \boldsymbol{\xi}$. Thus $\mathbf{c} = \mathbf{c}(\mu)$ could be seemed as a vector function in μ expressed differently at eigenvalues from other points. $\mathbf{c}(\mu)$ is undoubtedly continuous at the points other than eigenvalues, while the following proposition illustrates the continuity property of $\mathbf{c}(\mu)$ at the eigenvalues:

Proposition 9. *Suppose λ is a real eigenvalue of $\mathbf{T}^{\mathrm{T}}$ with eigenvector $\mathbf{c}(\lambda)$; and $\{\lambda_i\}$ is a sequence lying in the feasible domain of μ which converges to λ. If λ has geometric multiplicity 1, then the sequence $\{\mathbf{c}(\lambda_i)\}$ converges to $\mathbf{c}(\lambda)$ as well; otherwise, $\{\mathbf{c}(\lambda_i)\}$ converges to $\mathbf{0}$.*

Due to the lack of space, the proof of Proposition 9 is omitted here and available in our full version [25].

An Algorithmic Approach to Eigenvalue Method in Dimensions ≥ 5. By Proposition 9, if λ has geometric multiplicity 1, we can compute $\mathbf{c}(\lambda_i) = (\mathbf{T}^{\mathrm{T}} - \lambda_i \cdot \mathbf{I})^{-1} \cdot \mathbf{z}$ (in the case of tautological loop guard, we just replace $\mathbf{z}$ by any non-zero n-dimensional real vector) to approximate the eigenvector $\mathbf{c}(\lambda)$. On the other hand, in the case that λ has geometric multiplicity > 1, one can adopt Least-squares approximation as presented in [5, Section 8.9]. Though the

Least-squares approximation applies to the cases of eigenvalues with arbitrary geometric multiplicity, our method is much easier to implement and has higher efficiency.

7 Experimental Results

Experiment. We implement our automatic invariant-generation algorithm of eigenvalues and tight choices in Python 3.8 and use Sage [42] for matrix manipulation. All results are obtained on an Intel Core i7 (2.00 GHz) machine with 64 GB memory, running Ubuntu 18.04. Our benchmarks are affine loops chosen from some benchmark in the StInG invariant generator [40], some linear dynamical system in [30], some loop programs in [41] and some other linear dynamical systems resulting from well-known linear recurrences such as Fibonacci numbers, Tribonacci numbers, etc.

Complexity. The main bottleneck of our algorithm lies at exactly solving or approximating real roots of univariate polynomials (for computing eigenvalues and boundary points in our algorithmic approach). The rest includes Gaussian elimination with a single parameter (the polynomial-time solvability of which is guaranteed by [26]), matrix inverse and solving eigenvectors with fixed eigenvalues, which can easily be done in polynomial time. The exact solution for degrees less than 5 can be done by directly applying the solution formulas. The approximation of real roots can be carried out through real root isolation and a further divide-and-conquer (or Newton's method) in each obtained interval, which can be completed in polynomial time (see e.g. [36] for the polynomial-time solvability of real root isolation). Thus, our approach runs in polynomial time and is much more efficient than quantifier elimination in [12].

Results. The experimental results are presented in Table 1. In the table, the column 'Loop' specifies the name of the benchmark, 'Dim(ension)' specifies the number of program variables, 'μ' specifies the values through eigenvalues of the transition matrices (which we marked with $\mathfrak{e}$) or boundary points of the intervals in the feasible domain, 'Invariants' lists the generated affine invariants from our approach. We compare our approach with the existing generators StInG [40] and InvGen [22], where '$=$', '$>$', '$\gg$' and '$\neq$' means the generated invariants are identical, more accurate, can only be generated in this work, and incomparable, respectively. Table 2 compares the amounts of runtime for our approach and StInG and InvGen respectively, measured in seconds. Note that the runtime of StInG and InvGen are obtained by executing their binary codes on our platform.

Analysis. StInG [40] implements constraint-solving method proposed in [12,37], InvGen [22] integrates both constraint-solving method and abstract interpretation, while our approach uses matrix algebra to refine and upgrade the constraint-solving method. Based on the results in Table 1 and Table 2, we conclude that:

Table 1. Experimental Results of Invariants

Loop	Dim	μ	Invariants	[40]	[22]
Fibonacci numbers	2	$\lvert(1-\sqrt{5})/2\rvert_\epsilon$ $(1+\sqrt{5})/2_\epsilon$	$2x_1+(1-\sqrt{5})x_2-3+\sqrt{5}\le 0$ $-2x_1-(1-\sqrt{5})x_2+4-2\sqrt{5}\le 0$ $-2x_1-(1+\sqrt{5})x_2+3+\sqrt{5}\le 0$ $-2x_1-(1+\sqrt{5})x_2\le 0$	$\gg$	$\gg$
See-Saw [40]	2	1_ϵ	$x_1-2x_2\le 0$ $-3x_1+x_2\le 0$	$=$	$>$
Example 6.2 [30]	4	$\lvert1-\sqrt{2}\rvert_\epsilon$ $1+\sqrt{2}_\epsilon$	$w-y-(1-\sqrt{2})x+(1-\sqrt{2})z\le 0$ $w-y-(1+\sqrt{2})x+(1+\sqrt{2})z\le 0$	$>$	$>$
css2003 [41]	3	$0,\ 1_\epsilon$	$i-L^1\ \le 0$ $-i+1\le 0, i+k-1=0$	$=$	$=$
afnp2014 [41]	2	$0,\ 1_\epsilon,\ 1000/999$	$y-999\le 0$ $-y\le 0, x-999y-1\le 0$	$=$	$>$
gsv2008 [41]	2	$0,\ 1_\epsilon,\ 8/7$	$x-y+2\le 0$ $-y\le 0, -x-7y-50\le 0$	$>$	$\neq$
cggmp2005 [41]	2	$0, 1_\epsilon, 4/3$	$i-j-3\le 0, -i+1\le 0, j-10\le 0$ $i+2j-21=0, -i+j-9\le 0$	$>$	$>$
Jacobsthal numbers	2	$\lvert-1\rvert_\epsilon, 2_\epsilon$	$2x_1-x_2-1\le 0, -2x_1+x_2-1\le 0$ $-x_1-x_2+2\le 0$	$\gg$	$>$
Pell numbers	2	$\lvert1-\sqrt{2}\rvert_\epsilon$ $1+\sqrt{2}_\epsilon$	$x_1+(1-\sqrt{2})x_2-3+2\sqrt{2}\le 0$ $-x_1-(1-\sqrt{2})x_2+7-5\sqrt{2}\le 0$ $-x_1-(1+\sqrt{2})x_2+3+2\sqrt{2}\le 0$ $-x_1-(1+\sqrt{2})x_2\le 0$	$\gg$	$\gg$
Perrin numbers	3	$\Delta=\sqrt[3]{\frac{\sqrt{69}+9}{18}}$ $\mu=\frac{4}{3}\Delta_\epsilon$	$a=\frac{3\Delta+1/\Delta}{3}, b=1/a+1$ $x_1+bx_2+ax_3\ge\frac{2}{3\Delta}+2\Delta+3$	$\gg$	$\gg$
Tribonacci numbers	3	$\Delta=\sqrt[3]{3\sqrt{33}+19}$ $\mu=(5\Delta+1)/3_\epsilon$	$a=\frac{1}{3}(\Delta+\frac{4}{\Delta}+1), b=1/a+1$ $x_1+bx_2+ax_3\ge b+a$	$\gg$	$\gg$

[1] L stands for the variable `LARGE_INT` in the original program [41]. Note that we modified the loop programs in [41] as affine loops before execution.

Table 2. Experimental Results of Execution Time (s)

Loop	StInG [40]	InvGen [22]	Our Approach
Fibonacci numbers	0.030	0.079	0.178
See-Saw [40]	0.024	0.104	0.104
Example 6.2 [30]	0.030	0.092	0.173
css2003 [41]	0.019	0.111	0.193
afnp2014 [41]	0.025	0.076	0.193
gsv2008 [41]	0.027	0.092	0.207
cggmp2005 [41]	0.026	0.111	0.184
Jacobsthal numbers	0.026	0.085	0.193
Pell numbers	0.023	0.102	0.219
Perrin numbers	0.031	0.129	0.250
Tribonacci numbers	0.029	0.115	0.262

– For the benchmarks with rather simple transition matrices (identity or diagonal matrices), our approach covers or outnumbers the invariants generated by StInG and InvGen.
– For the benchmarks with complicated transition matrices (which are the matrices far away from diagonal ones), especially the ones with irrational eigenvalues, our approach generates adequate accurate invariants while StInG and InvGen generate nothing or only trivial invariants.
– For all benchmarks, the runtime of StInG and InvGen are faster but comparable with our runtime, hence shows the efficiency of our approach.

Summarizing all above, the experimental results demonstrate the wider coverage for the μ value endowed from our approach, and show the generality and efficiency of our approach.

Acknowledgements. This research is partially funded by the National Natural Science Foundation of China (NSFC) under Grant No. 62172271. We sincerely thank the anonymous reviewers for their insightful comments, which helped improve this paper. We also thank Mr. Zhenxiang Huang and Dr. Xin Gao for their pioneering contributions in the experimental part of this work.

References

1. Adjé, A., Gaubert, S., Goubault, E.: Coupling policy iteration with semi-definite relaxation to compute accurate numerical invariants in static analysis. Log. Methods Comput. Sci. **8**(1) (2012)
2. Albarghouthi, A., Li, Y., Gurfinkel, A., Chechik, M.: Ufo: a framework for abstraction- and interpolation-based software verification. In: Madhusudan, P., Seshia, S.A. (eds.) CAV 2012. LNCS, vol. 7358, pp. 672–678. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31424-7_48
3. Alias, C., Darte, A., Feautrier, P., Gonnord, L.: Multi-dimensional rankings, program termination, and complexity bounds of flowchart programs. In: Cousot, R., Martel, M. (eds.) SAS 2010. LNCS, vol. 6337, pp. 117–133. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15769-1_8
4. Almagor, S., Karimov, T., Kelmendi, E., Ouaknine, J., Worrell, J.: Deciding ω-regular properties on linear recurrence sequences. Proc. ACM Program. Lang. **5**(POPL), 1–24 (2021)
5. Andrilli, S., Hecker, D.: Chapter 8 - Additional applications. In: Andrilli, S., Hecker, D. (eds.) Elementary Linear Algebra, 5th edn, pp. 513–605. Academic Press, Boston (2016)
6. Asadi, A., Chatterjee, K., Fu, H., Goharshady, A.K., Mahdavi, M.: Polynomial reachability witnesses via Stellensätze. In: PLDI, pp. 772–787. ACM (2021)
7. Bagnara, R., Rodríguez-Carbonell, E., Zaffanella, E.: Generation of basic semi-algebraic invariants using convex polyhedra. In: Hankin, C., Siveroni, I. (eds.) SAS 2005. LNCS, vol. 3672, pp. 19–34. Springer, Heidelberg (2005). https://doi.org/10.1007/11547662_4
8. Breck, J., Cyphert, J., Kincaid, Z., Reps, T.W.: Templates and recurrences: better together. In: PLDI, pp. 688–702. ACM (2020)
9. Chatterjee, K., Fu, H., Goharshady, A.K.: Non-polynomial worst-case analysis of recursive programs. ACM Trans. Program. Lang. Syst. **41**(4), 20:1–20:52 (2019)

10. Chatterjee, K., Fu, H., Goharshady, A.K., Goharshady, E.K.: Polynomial invariant generation for non-deterministic recursive programs. In: PLDI, pp. 672–687. ACM (2020)
11. Chonev, V., Ouaknine, J., Worrell, J.: The polyhedron-hitting problem. In: Indyk, P. (ed.) Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, 4–6 January 2015, pp. 940–956. SIAM (2015)
12. Colón, M.A., Sankaranarayanan, S., Sipma, H.B.: Linear invariant generation using non-linear constraint solving. In: Hunt, W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 420–432. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45069-6_39
13. Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: POPL, pp. 238–252. ACM (1977)
14. Cousot, P., Halbwachs, N.: Automatic discovery of linear restraints among variables of a program. In: POPL, pp. 84–96. ACM Press (1978)
15. David, C., Kesseli, P., Kroening, D., Lewis, M.: Danger invariants. In: Fitzgerald, J., Heitmeyer, C., Gnesi, S., Philippou, A. (eds.) FM 2016. LNCS, vol. 9995, pp. 182–198. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48989-6_12
16. Dillig, I., Dillig, T., Li, B., McMillan, K.L.: Inductive invariant generation via abductive inference. In: OOPSLA, pp. 443–456. ACM (2013)
17. Farzan, A., Kincaid, Z.: Compositional recurrence analysis. In: 2015 Formal Methods in Computer-Aided Design (FMCAD), pp. 57–64 (2015)
18. Gan, T., Xia, B., Xue, B., Zhan, N., Dai, L.: Nonlinear craig interpolant generation. In: Lahiri, S.K., Wang, C. (eds.) CAV 2020. LNCS, vol. 12224, pp. 415–438. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-53288-8_20
19. Garg, P., Löding, C., Madhusudan, P., Neider, D.: ICE: a robust framework for learning invariants. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 69–87. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08867-9_5
20. Garg, P., Neider, D., Madhusudan, P., Roth, D.: Learning invariants using decision trees and implication counterexamples. In: POPL, pp. 499–512. ACM (2016)
21. Giacobazzi, R., Ranzato, F.: Completeness in abstract interpretation: a domain perspective. In: Johnson, M. (ed.) AMAST 1997. LNCS, vol. 1349, pp. 231–245. Springer, Heidelberg (1997). https://doi.org/10.1007/BFb0000474
22. Gupta, A., Rybalchenko, A.: InvGen: an efficient invariant generator. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 634–640. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02658-4_48
23. He, J., Singh, G., Püschel, M., Vechev, M.T.: Learning fast and precise numerical analysis. In: PLDI, pp. 1112–1127. ACM (2020)
24. Humenberger, A., Kovács, L.: Algebra-based synthesis of loops and their invariants (invited paper). In: Henglein, F., Shoham, S., Vizel, Y. (eds.) VMCAI 2021. LNCS, vol. 12597, pp. 17–28. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-67067-2_2
25. Ji, Y., Fu, H., Fang, B., Chen, H.: Affine Loop Invariant Generation via Matrix Algebra, May 2022. https://hal.archives-ouvertes.fr/hal-03494611, preprint
26. Kannan, R.: Solving systems of linear equations over polynomials. Theoret. Comput. Sci. **39**, 69–88 (1985)

27. Kapur, D.: Automatically generating loop invariants using quantifier elimination. In: Deduction and Applications. Dagstuhl Seminar Proceedings, vol. 05431. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany (2005)
28. Karimov, T., Lefaucheux, E., Ouaknine, J., Purser, D., Varonka, A., Whiteland, M.A., Worrell, J.: What's decidable about linear loops? Proc. ACM Program. Lang. **6**(POPL) (2022)
29. Kincaid, Z., Breck, J., Boroujeni, A.F., Reps, T.W.: Compositional recurrence analysis revisited. In: PLDI, pp. 248–262. ACM (2017)
30. Kincaid, Z., Breck, J., Cyphert, J., Reps, T.: Closed forms for numerical loops. Proc. ACM Program. Lang. **3**(POPL) (2019)
31. Kincaid, Z., Cyphert, J., Breck, J., Reps, T.W.: Non-linear reasoning for invariant synthesis. Proc. ACM Program. Lang. **2**(POPL), 54:1–54:33 (2018)
32. Manna, Z., Pnueli, A.: Temporal Verification of Reactive Systems: Safety. Springer, New York (2012). https://doi.org/10.1007/978-1-4612-4222-2
33. Nguyen, T., Kapur, D., Weimer, W., Forrest, S.: Using dynamic analysis to discover polynomial and array invariants. In: ICSE. pp. 683–693. IEEE Computer Society (2012)
34. de Oliveira, S., Bensalem, S., Prevosto, V.: Synthesizing invariants by solving solvable loops. In: D'Souza, D., Narayan Kumar, K. (eds.) ATVA 2017. LNCS, vol. 10482, pp. 327–343. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68167-2_22
35. Rodríguez-Carbonell, E., Kapur, D.: Automatic generation of polynomial invariants of bounded degree using abstract interpretation. Sci. Comput. Program. **64**(1), 54–75 (2007)
36. Sagraloff, M., Mehlhorn, K.: Computing real roots of real polynomials. J. Symb. Comput. **73**, 46–86 (2016)
37. Sankaranarayanan, S., Sipma, H.B., Manna, Z.: Constraint-based linear-relations analysis. In: Giacobazzi, R. (ed.) SAS 2004. LNCS, vol. 3148, pp. 53–68. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27864-1_7
38. Sharma, R., Aiken, A.: From invariant checking to invariant inference using randomized search. Formal Methods Syst. Des.' **48**(3), 235–256 (2016). https://doi.org/10.1007/s10703-016-0248-5
39. Sharma, R., Gupta, S., Hariharan, B., Aiken, A., Liang, P., Nori, A.V.: A data driven approach for algebraic loop invariants. In: Felleisen, M., Gardner, P. (eds.) ESOP 2013. LNCS, vol. 7792, pp. 574–592. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37036-6_31
40. Sting: Stanford invariant generator (2004). http://theory.stanford.edu/~srirams/Software/sting.html
41. SV-COMP2021: 11th Competition on Software Verification (2021). https://github.com/sosy-lab/sv-benchmarks
42. The Sage Developers: SageMath, the Sage Mathematics Software System (Version 9.4) (2021). https://www.sagemath.org
43. Xu, R., He, F., Wang, B.: Interval counterexamples for loop invariant learning. In: ESEC/FSE, pp. 111–122. ACM (2020)
44. Yao, J., Ryan, G., Wong, J., Jana, S., Gu, R.: Learning nonlinear loop invariants with gated continuous logic networks. In: PLDI, pp. 106–120. ACM (2020)