Affine Invariant Generation via Matrix Algebra

Yucheng Ji, Hongfei Fu, Bin Fang

Affine Invariant Generation via Matrix Algebra

Yucheng Ji[1,2], Hongfei Fu[1], Bin Fang[2]

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

Abstract. Loop invariant generation, which automates the generation of assertions that always hold inside loops, has many important applications such as safety analysis, reachability analysis and complexity-bound analysis. However, automated invariant generation for arbitrary loops is an undecidable problem. In this paper, we target at an important category of loops, namely affine loops, which widely exist in many programs but still lack general approach to invariant generation. We, however, provide a sound and complete approach based on matrix algebra to automatically synthesizing inductive invariants in the form of an affine inequality. Specifically, for the situation when the loop guard is tautological (i.e., '**true**'), we show that the eigenvalues and their eigenvectors generate all meaningful affine inductive invariants. Moreover, when the loop guard consists of one or more affine inequalities, we solve the invariant generation problem by (i) first establishing through matrix inverse the relationship between the invariants and a key parameter in the application of Farkas' Lemma, then (ii) solving the feasible domain of the key parameter, and finally (iii) showing that the key parameter can be addressed by a finite set of values w.r.t a tightness condition on the constraints for the invariants. Experimental results using existing and new cases show that our approach can generate affine invariants over linear dynamic systems that inherently involve the non-trivial choices (e.g., eigenvalues, boundary points of the feasible domain, etc.) of the key parameter.

1 Introduction

Invariants. An assertion at a program location is called an *invariant* if it is always satisfied whenever the location is reached in the execution of the program. Invariants play a fundamental role in program analysis and verification as they act as over-approximation for reachable program states. Applications of invariants include safety analysis, reachability analysis, complexity analysis, etc. A detailed account for these applications is as follows.

Safety Analysis. Given a program and a set of safety assertions that are expected to hold at critical program locations, the task of safety analysis is to prove that the assertions indeed hold or report that they might be violated by the program. As invariants are over-approximation of the reachable program states, an effective way for safety analysis is to generate invariants and check if they imply the safety assertions, as adopted by many existing approaches (see e.g. [44, 50, 3]).

Reachability Analysis. Reachability is the most basic liveness property that investigates whether a program location is reachable from a given set of initial program states. For reachability to the termination program location, a principal approach is to synthesize a ranking function [27] which require adequate invariants as input (see

e.g. [18, 7, 4, 51, 15]). Beyond termination, the reachability to erroneous program locations, resulting in static detection of program errors, requires danger invariants [24, 5] (intuitively as invariants extended with ranking functions) as a formal witness. Recently, invariants have also been shown necessary in the reachability analysis of probabilistic programs [9, 62, 13, 63].

Complexity-Bound Analysis. Another basic problem is to automatically infer asymptotic complexity bounds on the runtime of a program. Current algorithms for tackling this problem, such as [11], rely heavily on adequate invariants.

Invariant Generation. Invariant generation is the classical problem that asks to automatically generate invariants for an input program, and has been studied for decades. Various approaches have been proposed, such as abstract interpretation [20, 22], constraint solving [38, 17, 12], recurrence analysis [42, 26, 37, 40], logical inference [32, 25, 57, 29, 45, 28], machine learning [30, 66, 34], dynamic analysis [23, 58, 47], etc. To guarantee that an assertion is indeed an invariant, the widely-adopted way is to generate a *inductive invariant*, which holds for the first visit to the location and is preserved under every cyclic execution path to and from the location, that strengthens it [17, 44].

Numerical Invariants. In this work, we consider an important subclass of invariants called *numerical invariants*. Briefly, numerical invariants are assertions over numerical values taken by the program variables and closely related to many common failures of programs (such as array out-of-bound, division by zero, etc). In detail, we consider *affine* inductive invariants in the form of an affine assertion over program variables. To resolve the automated generation of affine inductive invariants, we adopt the method of constraint solving as follows.

The Method of Constraint-Solving. Constraint-solving based approaches first establish a template with unknown parameters for the target invariants, then collect constraints from the inductive condition, and finally solve the unknown parameters to get the desired invariants. For affine invariant generation, Farkas' Lemma provides a complete characterization of the inductive condition which has been studied in [17, 56] and further solved by quantifier elimination [17] and several heuristics [56]. The StInG invariant generator [59] implements the approach in [56], and the INVGEN invariant generator [33] integrates abstraction interpretation and the approach in [56]. Besides, an approach based on eigenvalues and eigenvectors for a restricted class of invariants is proposed in [49]. Recently, probabilistic affine invariants have also been considered in probabilistic programs through Farkas' Lemma and Motzkin's Transposition Theorem [39, 14]. Compared with other methods (such as abstract interpretation, machine learning, etc.), constraint solving has the advantage of a theoretical guarantee on the accuracy of the generated invariants, but typically requires higher runtime complexity.

Our Contribution. We propose matrix-based approaches for generating affine inductive invariants of affine while loops without nested loops. Following [17, 56], we base our approach on Farkas' Lemma. However, we completely solve the constraints obtained from Farkas' Lemma by matrix methods which is beyond the scope of [17, 56]. For affine while loops with tautological guard, we prove that the invariants are determined by the eigenvectors (of the transpose) of the transition matrix; for affine while loops with non-tautological guard, we solve the invariants by adopting a formula involving matrix inverse with a key parameter in the constraints that leads to non-linearity, and determining the feasible domain of the key parameter as well as showing that it suffices to choose a finite number of values for the key parameter if one imposes a tightness condition on the constraints. In addition,

we generalize our results to affine loops with non-deterministic updates and to bidirectional affine invariants. Furthermore, we prove the continuity of invariants w.r.t. the key parameter and develop an algorithmic approach to approximate the eigenvectors in high dimensions for implementing our methods.

2 Preliminaries

In this section, we specify the class of affine while loops considered in this work, 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 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 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 the 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 inequalities 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 $\mathbf{v}'$.

The syntax of (unnested) affine while loops is as follows.

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

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

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 affine assertion $\tau_j := \mathbf{T}_j \cdot \mathbf{x} - \mathbf{T}'_j \cdot \mathbf{x}' + \mathbf{b}_j \leq 0$ 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_i satisfying $\mathbf{v} \models \psi_i$, and sets the next program state $\mathbf{v}'$ non-deterministically such that $\mathbf{v}, \mathbf{v}' \models \tau_i$; 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 invariants over affine while loops. Informally, an affine invariant is an affine inequality satisfing the initiation and consecution conditions which roughly mean that the inequality should hold at the start of the loop (initiation) and be preserved under every iteration of the loop body (consecution).

Affine Invariants. An *affine 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 $(\mathbf{v} \models G \wedge \mathbf{v} \models \Phi \wedge \mathbf{v}, \mathbf{v}' \models \tau_j) \Rightarrow \mathbf{v}' \models \Phi'$.

It can be observed from the definition above that every program state traversed (as a current state at the start or after one loop iteration) in some execution of the underlying affine while loop will satisfy the invariant.

Problem Statement. In this work, we study the problem of automatically generating affine invariants over affine while 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 [17, 56]. 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 [17, 56], and then point out the weakness of each of the approaches.

We first present Farkas' Lemma following the formulation in [17, 56].

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:

$$\begin{array}{r|l} \lambda_0 & -1 \leq 0 \\ \lambda_1 & a_{11}y_1 + ... + a_{1n}y_n + b_1 \leq 0 \quad (\\ \vdots & \vdots \qquad\qquad \vdots \qquad \text{S} \\ \lambda_k & a_{k1}y_1 + ... + a_{kn}y_n + b_k \leq 0 \quad) \\ \hline & c_1y_1 + ... + c_ny_n + d \leq 0 \ (\phi) \end{array} \tag{‡}$$

The intuition of the table form above is that one first multiplies the λ_i's on the left to their corresponding affine inequalities (at 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 refer to call the table form as *Farkas table*.

Then we illustrate the application of Farkas' Lemma in [17, 56]. Given an affine while loop in the form of (†), they first establish a template $\Phi : c_1x_1 + ... + c_nx_n + d \leq 0$ 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 by the Farkas table (‡) with $S := \theta$ and $\phi := \Phi$:

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

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 left column of (‡).

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

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

Note that the Farkas table above contains quadratic constraints as we multiply an unknown non-negative parameter μ to the unknown invariant Φ in the table. The Farkas tables for all the conditional branches are grouped conjunctively together to represent the whole consecution condition.

The weakness of the approaches [17, 56] lies at the treatment of the quadratic constraints from the consecution condition. The approach in [17] addresses the quadratic constraints by quantifier elimination that guarantees the theoretical completeness but typically has high runtime complexity. The approach in [56] solves the quadratic constraints by several heuristics that guesses possible values for the key parameter μ in ($*$) that causes non-linearity, hence losing completeness. Our approach considers to address the key parameter μ through matrix-based methods (e.g., eigenvalues, matrix inverse, etc.), which is capable of efficiently generating affine invariants (as compared with quantifier elimination [17]) while still ensuring theoretical completeness (as compared with the heuristics in [56]).

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 while loops where there is only one conditional branch in the loop body and the update of the next-value vector $\mathbf{x}'$ is deterministic.

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

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

For the loop above, we aim at *non-trivial* affine invariants, i.e., $\mathbf{c} \neq \mathbf{0}$. We summarize our results below.

1. When the loop guard is '**true**', there are only finitely many non-trivial invariants $\mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} + d \leq 0$ such that $\mathbf{c}$ is an eigenvector of the transpose of the transition matrix $\mathbf{T}^{\mathrm{T}}$.
2. When the loop guard is not a tautology, there can be infinitely many non-trivial invariants $\mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} + d \leq 0$ with $\mathbf{c}$ given by a direct formula of μ (see (6) and (6′)); in this case we derive the *feasible domain* of μ (Proposition 4 and Proposition 7) and select finitely many optimal ones (which we call *tight choices*) among them (Proposition 5 and Proposition 8).

In Section 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 Section 4.2 and the single-constraint loop guard case in Section 4.3. Finally we generalize the results to the multi-constraint loop guard case in Section 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 in (#). 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:

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

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

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

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

μ	$\mathbf{c}^{\mathrm{T}} \cdot \mathbf{x}$		$+\ d \leq 0$	(Φ)
λ_0^{C}			$-\ 1 \leq 0$	
$\boldsymbol{\xi}$	$\mathbf{P} \cdot \mathbf{x}$		$+\ \mathbf{q} \leq 0$	(G)
$\boldsymbol{\eta}$	$\mathbf{T} \cdot \mathbf{x}$	$-\ \mathbf{x}'$	$+\ \mathbf{b} = \mathbf{0}$	(τ)
		$\mathbf{c}^{\mathrm{T}} \cdot \mathbf{x}'$	$+\ d \leq 0$	(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, parameters $\eta_1, ..., \eta_n$ do not have to 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

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

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**':

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

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 $\mathbf{c}$ up to some certain freedom;
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 '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 tautology, we take parameters $\boldsymbol{\xi}$ 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 sequence). Consider 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:

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

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:*

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

If the above conditions have empty solution set for d, then no 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:

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

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

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

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

Example 2 (Fibonacci sequence, Part 2). We continue Example 1. Recall that

$$\mu = \frac{1+\sqrt{5}}{2},\ \mathbf{c} = \begin{bmatrix} c_1 \\ \frac{1+\sqrt{5}}{2} c_1 \end{bmatrix},\ \mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},\ \mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ \mathbf{f} = \begin{bmatrix} -1 \\ -1 \end{bmatrix};$$

thus (3)(5) read $-\frac{3+\sqrt{5}}{2} c_1 \ge d$ and $\frac{-1+\sqrt{5}}{2} d \ge 0$, hence yield $0 \le d \le -\frac{3+\sqrt{5}}{2} c_1$. The free variable c_1 must be negative here, so we choose $c_1 = -2$ and then

$$\mathbf{c} = \begin{bmatrix} -2 \\ -1-\sqrt{5} \end{bmatrix}, \quad 0 \le d \le 3 + \sqrt{5};$$

the two boundary values $d = 0$ and $d = 3 + \sqrt{5}$ yield invariants

$$-2x_1 - (1+\sqrt{5})x_2 \le 0; \quad -2x_1 - (1+\sqrt{5})x_2 + 3 + \sqrt{5} \le 0.$$

Obviously the latter is stronger; so we finally determine that $d = 3 + \sqrt{5}$:

$$\mu = (1+\sqrt{5})/2 :\ -2x_1 - (1+\sqrt{5})x_2 + 3 + \sqrt{5} \le 0.$$ □

4.3 Loops with Guard: Single-Constraint Case

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

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

$$\begin{array}{l}\textbf{initial condition } \theta : \mathbf{R}\cdot\mathbf{x}+\mathbf{f}\le\mathbf{0}\\ \textbf{while } \mathbf{p}^{\mathrm{T}}\cdot\mathbf{x}+q\le 0 \textbf{ do } \mathbf{x}'=\mathbf{T}\cdot\mathbf{x}+\mathbf{b}.\end{array} \tag{$\diamond'$}$$

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

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

1. we derive a formula to compute $\mathbf{c}$ in terms of μ; so for any 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 do 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:

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

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

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.

In contrast to Section 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 $\mathbf{c}^{\mathrm{T}}\cdot\mathbf{x}+d\le 0$ of the loop ($\diamond'$), we have that $\mathbf{c}$ is given by*

$$\mathbf{c}=\xi\cdot(\mathbf{T}^{\mathrm{T}}-\mu\cdot\mathbf{I})^{-1}\cdot\mathbf{p} \tag{6}$$

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 sequence, Part 3). We add a loop guard to Example 1:

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

and search for more invariants. The equation (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 (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 ($\diamond'$), 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 multiple $(\mu - 1)$ 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′) won't 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 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′). □

Example 4 (Fibonacci sequence, Part 4). Let us find out the feasible domain of μ for the Fibonacci sequence with loop guard $x_1 \leq 10$. Inequality (5′) is $(\mu - 1)d \geq 10\xi$; inequality (3″) is

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

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

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

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 select finitely many optimal ones among them: the ideal case is that there exists a basis consisting of finitely many invariants, such that all invariants are non-negative linear combinations

of the basis; however, this idea doesn't work out, where the reasons will be explained thoroughly in Appendix A.1 and Appendix A.2. Instead, we impose a weaker form of optimality called 'tightness' coming from the equality cases of constraints (3)(5′):

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

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

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

The tight choices are characterized by the following proposition:

Proposition 5. *For the loop ($\diamond'$), the tight choices of μ consist of 0 and the positive roots of the following rational equation:*

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

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 $\lambda_0^{\mathrm{I}} = \lambda_0^{\mathrm{C}} = 0$:

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

(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. If μ is a root to (8), then d is uniquely determined by:*

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

otherwise, the optimal value of d is determined by one of the two choices below:

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

We omit the detailed proof and put it in Appendix A.3.

Example 5 (Fibonacci sequence, 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 $\mu = \frac{5}{3}$. Thus by Proposition 5 and Proposition 6, we have two new 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:

$$\begin{aligned} &\textbf{initial condition } \ \theta : \mathbf{R} \cdot \mathbf{x} + \mathbf{f} \leq \mathbf{0} \\ &\textbf{while } \ \mathbf{P} \cdot \mathbf{x} + \mathbf{q} \leq \mathbf{0} \ \textbf{ do } \ \mathbf{x}' = \mathbf{T} \cdot \mathbf{x} + \mathbf{b}. \end{aligned} \tag{$\diamond''$}$$

where the loop guard $\mathbf{P} \cdot \mathbf{x} + \mathbf{q} \leq \mathbf{0}$ contains m affine inequalities (m could be larger than, equal to or less than n).

We can easily generalize the results of Section 4.3 to this case. First of all, we generalize Proposition 3: modify the formula (6) for $\mathbf{c}$ in terms of μ into

$$\mathbf{c} = (\mathbf{T}^{\mathrm{T}} - \mu \cdot \mathbf{I})^{-1} \mathbf{P}^{\mathrm{T}} \cdot \boldsymbol{\xi}. \tag{6'}$$

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

Proposition 7. *For the loop ($\diamond''$), 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 'generalized compatibility condition':*

$$\begin{aligned} &\mathbf{u}(\mu) \coloneqq \mathbf{b}^{\mathrm{T}} \cdot (\mathbf{T}^{\mathrm{T}} - \mu \cdot \mathbf{I})^{-1} \mathbf{P}^{\mathrm{T}} - \mathbf{q}^{\mathrm{T}} \\ \leq \ &\mathbf{w}(\mu) \coloneqq (\mu - 1) \mathbf{f}^{\mathrm{T}} \cdot (\mathbf{R}^{\mathrm{T}})_{\mathrm{L}}^{-1} (\mathbf{T}^{\mathrm{T}} - \mu \cdot \mathbf{I})^{-1} \mathbf{P}^{\mathrm{T}} \end{aligned} \tag{7'}$$

where $\mathbf{u}(\mu), \mathbf{w}(\mu)$ are two m-dimensional vector functions of μ. (7') means that the j-th component of $\mathbf{u}(\mu)$ being no larger than the j-th component of $\mathbf{w}(\mu)$ for all $1 \leq j \leq m$; when $m = 1$, it goes back to (7).

Last but not least, we consider the tight choices of μ. The first idea comes up to mind is repeating Proposition 5: set the generalized compatibility condition to achieve 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 consists of 0 and the (positive) boundary points of K; so we generalize this to the multi-constraint case:

Proposition 8. *For the loop ($\diamond''$), the tight choices of μ consist of 0 and the (positive) boundary points of $\overline{K}$.*

Example 6. We consider the loop:

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

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

$$x_1 + x_2 - 2 \le 0,\ x_1 + x_2 - 2 \ge 0;\ -x_1 + x_2 \le 0.$$

Next we find out the other invariants from tight μ's. In this case $(7')$ read

$$(\text{when } \mu > 1:)\quad \frac{11-10\mu}{1-\mu} \le 1 \ \wedge\ \frac{6-5\mu}{1-\mu} \le -1.$$

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 $0, \frac{10}{9}$:

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

5 Generalizations

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

5.1 Loops with Non-deterministic Updates

In Section 4, we handled the loops with deterministic updates; here we generalize the results to the ones with non-deterministic updates in the form of (†). We still focus on the single-branch loops (for multi-branch ones, we shall solve out invariants for each branch separately and simply take the intersection):

$$\begin{aligned}&\textbf{initial condition } \ \theta : \mathbf{R}\cdot\mathbf{x}+\mathbf{f} \le \mathbf{0}\\&\textbf{while } \ G : \mathbf{P}\cdot\mathbf{x}+\mathbf{q} \le \mathbf{0} \ \textbf{ do } \ \mathbf{T}\cdot\mathbf{x}-\mathbf{T}'\cdot\mathbf{x}'+\mathbf{b} \le 0\,.\end{aligned}\qquad (\dagger')$$

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

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

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

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

where (10) provides the relationship of $\mathbf{c}$ and $\boldsymbol{\eta}$; plugging it into (9) yield

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

Hence for any non-trivial invariant $\mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} + d \leq 0$ of this loop ($\dagger'$), 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 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 null vector of the matrix $\mathbf{T}^{\mathrm{T}} - \mu \cdot (\mathbf{T}')^{\mathrm{T}}$.
2. $\mathbf{T}$ and $\mathbf{T}'$ are square matrices and there is loop guard. In this case, μ must 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 time 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 to solve (9′). By linear algebra, the solution $\boldsymbol{\eta}(\mu)$ would contain 'homogeneous term' (which doesn't involve ξ_i's) and 'non-homogeneous terms' (which contain linear terms of ξ_i's). Thus $\boldsymbol{\eta}(\mu)$ could be written as $\overline{\boldsymbol{\eta}}(\mu) + \sum_i \xi_i \cdot \widetilde{\boldsymbol{\eta}}_i(\mu)$.

For Case 2 and Case 3, we have a continuum of candidates for μ. The feasible domain of μ is given by $\Big([0,1) \cup \left(\widetilde{K} \cap [1,+\infty)\right)\Big) \cap J$, where $\widetilde{K}$ is the solution set to the following compatibility condition (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 $\boldsymbol{\eta}(\mu) \geq \mathbf{0}$. The tight choices of μ consists of 0 and the positive boundary points of $\widetilde{K} \cap J$.

5.2 An Extension to Bidirectional Affine Invariants

Here we restrict us 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 9. *For any bidirectional invariant $d_1 \leq \mathbf{c}^{\mathrm{T}} \cdot \mathbf{x} \leq d_2$ of the loop ($\diamond$), 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 sets of 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 Section 4.2. Thus we choose the second one: make the two inequalities induct each

other. Hence the Farkas tables are

μ	$-\mathbf{c}^{\mathrm{T}}\cdot\mathbf{x}$	$+\,d_1\le 0\ (\Phi_1)$	
λ_0^{C}		$-\ 1\le 0$	
$-\mathbf{c}$	$\mathbf{T}\cdot\mathbf{x}\ -\ \mathbf{x}'$	$+\,\mathbf{b}=\mathbf{0}\ \ (\tau)$	
	$\mathbf{c}^{\mathrm{T}}\cdot\mathbf{x}'-d_2\le 0\ (\Phi_2')$		

$\widetilde{\mu}$	$\mathbf{c}^{\mathrm{T}}\cdot\mathbf{x}$	$-\,d_2\le 0\ (\Phi_2)$	
$\widetilde{\lambda}_0^{\mathrm{C}}$		$-\ 1\le 0$	
$\mathbf{c}$	$\mathbf{T}\cdot\mathbf{x}\ -\ \mathbf{x}'$	$+\,\mathbf{b}=\mathbf{0}\ \ (\tau)$	
	$-\,\mathbf{c}^{\mathrm{T}}\cdot\mathbf{x}'+d_1\le 0\ (\Phi_1')$		

We write out the constraints of consecution:

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

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

Example 7 (Fibonacci sequence, 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:

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

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

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

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

6 Continuity of Invariants w.r.t. μ

In Section 4, we have shown that for any invariant $\mathbf{c}^{\mathrm{T}}\cdot\mathbf{x}+d\le 0$ of single-branch affine loop with deterministic updates, $\mathbf{c}$ is given in two ways:

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

with $\mathbf{z}=\mathbf{P}^{\mathrm{T}}\cdot\boldsymbol{\xi}$. We express $\mathbf{c}(\mu)$ differently at eigenvalues than at other points. A natural question is thus brought up to us: is $\mathbf{c}(\mu)$ continuous at eigenvalues w.r.t. μ? The following proposition answers this question:

Proposition 10. *Suppose λ is an 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; in other words, we can use $\{\mathbf{c}(\lambda_i)\}$ to approximate $\mathbf{c}(\lambda)$ in this case.*

In order to prove it, we first introduce the following linear algebra theorem:

Theorem 2. *We denote the adjoint matrix of $\mathbf{T}^{\mathrm{T}}-\mu\cdot\mathbf{I}$ by $\mathbf{Ad}(\mu)$. Suppose λ is an eigenvalue of $n\times n$ matrix $\mathbf{T}$. Then for any n-vector $\mathbf{x}$,*

1. *$\mathbf{Ad}(\lambda)\cdot\mathbf{x}$ is an eigenvector of $\mathbf{T}^{\mathrm{T}}$ with eigenvalue λ; moreover, there exists $\mathbf{x}$ such that $\mathbf{Ad}(\lambda)\cdot\mathbf{x}\neq\mathbf{0}$ if and only if λ has geometric multiplicity 1;*
2. *If $\{\lambda_i\}$ is a sequence convergent to λ, then $\{\mathbf{Ad}(\lambda_i)\cdot\mathbf{x}\}$ also converges to $\mathbf{Ad}(\lambda)\cdot\mathbf{x}$.*

We address the proof of this theorem in Appendix A.4. Now we are ready to prove Proposition 10:

Proof. By the definition of inverse matrix and adjoint matrix, $\mathbf{c}(\lambda_i) = (\mathbf{T}^{\mathrm{T}} - \lambda_i \cdot \mathbf{I})^{-1} \cdot \mathbf{z} = (\mathbf{Ad}(\lambda_i) \cdot \mathbf{z}) / \det(\mathbf{T}^{\mathrm{T}} - \lambda_i \cdot \mathbf{I})$ where $\det(\mathbf{T}^{\mathrm{T}} - \lambda_i \cdot \mathbf{I})$ is a (non-zero) number, so we can absorb it into $\mathbf{z}$ and write $\mathbf{c}(\lambda_i) = \mathbf{Ad}(\lambda_i) \cdot \mathbf{z}$. By Theorem 2.2, the limit of $\{\mathbf{c}(\lambda_i)\}$ exists, which is denoted by $\mathbf{Ad}(\lambda) \cdot \mathbf{z}$. There are two possible outcomes:

1. $\mathbf{Ad}(\lambda) \cdot \mathbf{z} = \mathbf{0}$ for any vector $\mathbf{z}$. By Theorem 2.1, we know the geometric multiplicity of λ is > 1 and no continuity is available;
2. There exists $\mathbf{z}$ such that $\mathbf{Ad}(\lambda) \cdot \mathbf{z} \neq \mathbf{0}$. By Theorem 2.1, we conclude that $\mathbf{Ad}(\lambda) \cdot \mathbf{z}$ is the non-zero eigenvector $\mathbf{c}(\lambda)$ of $\mathbf{T}^{\mathrm{T}}$ with eigenvalue λ. □

As a byproduct, Proposition 10 can be used to implement our eigenvalue method in dimensions higher than 4:

An Algorithmic Approach to Eigenvalue Method. In Section 4.2 and Section 5.2, we need to solve the characteristic polynomial to get the eigenvalues; while general polynomials with degree ≥ 5 do not have algebraic solution formula due to Abel-Ruffini-Galois theorem. We can develop a number sequence $\{\lambda_i\}$ to approach the true eigenvalue λ; however, we can't approximate the eigenvector of λ by solving the kernel of $\mathbf{T}^{\mathrm{T}} - \lambda_i \cdot \mathbf{I}$ because it has trivial kernel.

Nonetheless, if λ has geometric multiplicity 1, we can compute $(\mathbf{T}^{\mathrm{T}} - \lambda_i \cdot \mathbf{I})^{-1} \cdot \mathbf{z}$ (in the case of tautological loop guard, we just replace $\mathbf{z}$ by any non-zero n-vector) to approximate the eigenvector $\mathbf{c}(\lambda)$ due to Proposition 10. However, in the case that λ has geometric multiplicity > 1, no such approach is available.

7 Experimental Results

Implementation. We implemented our automatic invariants generation algorithm in Python 3.8 and used Sage [61] for matrix manipulation. All results were obtained on an Intel Core i7(2.00 GHz) machine with 64 GB memory, running Ubuntu 18.04. All μ values were obtained by explicitly solving the roots of characteristic polynomial of the transition matrix and the polynomial (8).

Benchmarks. Our benchmarks are affine while loops chosen from some original linear transition systems benchmarks (which we rewrite as programs) from the StInG invariant generator [59] that implements the most related approach in [56], some linear dynamical systems in [41], some loop programs in [60] and some other linear dynamical systems resulting from well-known linear recurrences such as Fibonacci numbers, Lucas numbers, etc.

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 in the benchmark, 'μ' specifies the values through eigenvalues of the transition matrix or boundary points of the intervals in the feasible domain, 'Invariants' lists the generated affine invariant from our approach, 'C(om)p(a)r(ison)' compares our approach with the existing cited work where '=' means the generated invariants are identical, '>' means our generated invariants are tighter, '/' means the invariants can only be generated in this work, and 'Time' records the amount of runtime for our approach measured in seconds.

Analysis. The results in Table 1 show that our approach can derive affine invariants with irrational coefficients which result from the calculated irrational μ's from eigenvalues of the transition matrix and boundary points of the feasible domain, which cannot be achieved by the invariant generator StInG. This demonstrates the thorough coverage for the μ value endowed from our approach.

Table 1. Affine Invariant Generation Experimental Results

Loop	Dim	μ	Invariants	Cpr	Time (s)
Fibonacci numbers	2	$(1-\sqrt{5})/2$ $(1+\sqrt{5})/2$	$2x_1+(1-\sqrt{5})x_2+2\sqrt{5}-4\geq 0$ $-2x_1+(\sqrt{5}-1)x_2-\sqrt{5}+3\geq 0$ $2x_1+(\sqrt{5}+1)x_2-\sqrt{5}-3\geq 0$ $2x_1+(\sqrt{5}+1)x_2\geq 0$	/	1.78
See-Saw [59]	2	1	$-x_1+2x_2\geq 0$ $3x_1-x_2\geq 0$	=	1.04
Example 6.2 [41]	4	$1-\sqrt{2}$ $1+\sqrt{2}$	$-w+y-(\sqrt{2}-1)x+(\sqrt{2}-1)z\geq 0$ $-w+y+(\sqrt{2}+1)x-(\sqrt{2}+1)z\geq 0$	>	1.73
css200 [60]	2	0, 1	$i-k+1\geq 0, i-j\geq 0$ $-2k+1\geq 0$	>	1.93
afnp2014 [60]	2	0, 1, 1000/999	$-x+999y+1\geq 0$ $y\geq 0, -y+999\geq 0$	>	1.93
gsv2008 [60]	2	0, 1, 8/7	$x-y+50\geq 0, x+7y+50\geq 0$ $-x+y-2\geq 0$	>	2.07
cggmp2005 [60]	2	0, 1, 10/9	$i-j+9\geq 0, i+2j-21=0$	>	1.84
Lucas numbers	2	$(1-\sqrt{5})/2$ $(1+\sqrt{5})/2$	$-2x_1+(\sqrt{5}-1)x_2+5-\sqrt{5}\geq 0$ $2x_1-(\sqrt{5}-1)x_2+3\sqrt{5}-5\geq 0$ $2x_1+(\sqrt{5}+1)x_2-5-\sqrt{5}\geq 0$ $2x_1+(\sqrt{5}+1)x_2\geq 0$	/	2.15
Pell numbers	2	$1-\sqrt{2}$ $1+\sqrt{2}$	$x_1-(\sqrt{2}-1)x_2+5\sqrt{2}-7\geq 0$ $-x_1+(\sqrt{2}-1)x_2+3-2\sqrt{2}\geq 0$ $x_1+(\sqrt{2}+1)x_2-3-2\sqrt{2}\geq 0$ $x_1+(\sqrt{2}+1)x_2\geq 0$	/	2.19
Pell-Lucas numbers	2	$1-\sqrt{2}$ $1+\sqrt{2}$	$x_1-(\sqrt{2}-1)x_2+6\sqrt{2}-8\geq 0$ $-x_1+(\sqrt{2}-1)x_2+4-2\sqrt{2}\geq 0$ $x_1+(\sqrt{2}+1)x_2-4-2\sqrt{2}\geq 0$ $x_1+(\sqrt{2}+1)x_2\geq 0$	/	2.17
Mersenne numbers	2	1, 2	$-x_1+x_2\geq 0$ $2x_1-x_2-1\geq 0$	/	1.79
Tribonacci numbers	3	$\Delta=(3\sqrt{33}+19)^{1/3}$ $\mu=(5\Delta+1)/3$	$a=\frac{1}{3}(\Delta+\frac{4}{\Delta}+1), b=1/a+1$ $x_1+bx_2+ax_3\geq b+a$	/	2.62
Perrin numbers	3	$\Delta=(\frac{\sqrt{69}+9}{18})^{1/3}$ $\mu=\frac{4}{3}\Delta$	$a=\frac{3\Delta+1/\Delta}{3}, b=1/a+1$ $x_1+bx_2+ax_3\geq \frac{2}{3\Delta}+2\Delta+3$	/	2.50
Jacobsthal numbers	2	$-1, 2$	$-2x_1+x_2+1\geq 0, -2x_1+x_2-1\leq 0$	/	1.93
Jacobsthal-Lucas numbers	2	$-1, 2$	$2x_1-x_2-3\leq 0, 2x_1+x_2+1\geq 0$	/	1.94

8 Related Works

Below we compare our approach with existing approaches from the literature in more detail.

Constraint Solving. Our approach falls in the category of constraint solving. We target affine invariants, thus our approach is incomparable with the ones for polynomial invariant generation [38, 65, 12, 35, 53, 19, 1, 43, 16, 48, 36, 55]. Below we compare our result with the existing results for affine invariant generation. [17] uses quantifier elimination, while our approach is more efficient since it only involves the calculation of eigenvalues/eigenvectors and matrix inverse. Compared with [56] that uses several heuristics to guess the value of μ, our approach provides the characterization for the domain of all feasible μ values, thus is complete and more accurate. [49] considers the use of eigenvalues and eigenvectors but restricted to the subclasses of equality and convergent invariants, while our approach targets general affine invariants over affine while loops and provides a comprehensive treatment of them. Another related work [41] considers to have a decidable logic for simple affine loops without loop guard by avoiding the computation of eigenvalues. Our result handles general affine loops with possibly loop guard and targets invariant generation, thus they target different objectives.

Abstract Interpretation. A mainstream technique to find inductive invariants is *abstract interpretation* [52, 2, 6, 21, 46, 54, 10], which is the oldest and most classical approach to invariant generation. Roughly speaking, the method of abstract interpretation first establishes an abstract domain for the specific form of invariants to

be generated, and then perform forward/backward propagation to reach a fixed point. Compared with constraint solving, abstract interpretation does not provide any guarantee on the accuracy of the generated invariants, except for rare special cases [31]. This point is experimental observed in [56], where constraint solving can produce much tighter invariants. In contrast, our approach ensures the tightness of the generated invariants by a thorough analysis on the choice of the key parameter in the application of Farkas' Lemma.

Recurrence Analysis. Another closely-related technique is *recurrence analysis* [42, 26, 37, 40, 8]. It usually transforms the invariant-generation problem into a recurrence relation, and then solve the invariants through the analysis of recurrence relations. The disadvantage of recurrence-based approaches is that they are only applicable to the situations where the underlying recurrence relation has a closed form solution. In our situation, we consider the general case of affine invariants over affine while loops, for which a closed form solution is not possible.

Logical Inference. Invariants could also be obtained through logical inference, such as abductive inference [25], Craig interpolation [28, 45], ICE learning [29, 64], predicate abstraction [32], random search [57], etc. Compared with our approach, those approaches cannot provide any theoretical guarantee on the tightness of the generated invariants, which is though ensured by our approach.

Machine Learning. The method of machine learning [30, 66, 34] solves the invariant-generation problem as follows: first, one establishes a (typically large) training set of programs whose invariants are given; then, training approaches (such as neural networks) are applied to the training set to train an invariant generator; finally, the trained invariant generator is used to generate invariants for programs outside the training set. Compared with constraint solving, machine-learning based approaches require a large training set to cover as most typical loops as possible, and still cannot ensure a theoretical guarantee on the tightness of the generated invariants. In particular, these approaches cannot produce specific numerical values (e.g. eigenvalues) needed to handle our examples.

Dynamic Analysis. Dynamic analysis [23, 58, 47] first runs the program of concern for multiple times to collect its execution data, and then guess the invariants based on these data. Compared with constraint solving that statically generates the invariants with correctness and precision guarantee, dynamic analysis needs to run the program multiple times for a potentially large amount of execution data, and still cannot ensure the tightness of the generated invariants.

9 Conclusion and Future Work

In this work, we had a thorough investigation on the affine invariant generation over unnested affine while loop. Our approach is based on the previous approaches [17, 56] through Farkas' Lemma and completely addresses the main issue of quadratic constraints (from the application of Farkas' Lemma) via matrix methods. Experimental results show that our approach can efficiently automatic generate affine invariants from eigenvalues/eigenvectors and matrix inverse of the relevant matrices derived from the loop. In particular, our approach is able to generate affine invariants with irrational coefficients resulting from irrational eigenvectors, which to our best knowledge is not possible from the previous approaches in the literature. Future work needs to concentrate on the complex examples.

References

1. Adjé, A., Garoche, P., Magron, V.: Property-based polynomial invariant generation using sums-of-squares optimization. In: SAS. LNCS, vol. 9291, pp. 235–251. Springer (2015). https://doi.org/10.1007/978-3-662-48288-9_14, https://doi.org/10.1007/978-3-662-48288-9_14
2. 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). https://doi.org/10.2168/LMCS-8(1:1)2012, https://doi.org/10.2168/LMCS-8(1:1)2012
3. Albarghouthi, A., Li, Y., Gurfinkel, A., Chechik, M.: Ufo: A framework for abstraction- and interpolation-based software verification. In: CAV. LNCS, vol. 7358, pp. 672–678. Springer (2012). https://doi.org/10.1007/978-3-642-31424-7_48, https://doi.org/10.1007/978-3-642-31424-7_48
4. Alias, C., Darte, A., Feautrier, P., Gonnord, L.: Multi-dimensional rankings, program termination, and complexity bounds of flowchart programs. In: SAS. LNCS, vol. 6337, pp. 117–133. Springer (2010). https://doi.org/10.1007/978-3-642-15769-1_8, https://doi.org/10.1007/978-3-642-15769-1_8
5. Asadi, A., Chatterjee, K., Fu, H., Goharshady, A.K., Mahdavi, M.: Polynomial reachability witnesses via stellensätze. In: PLDI. pp. 772–787. ACM (2021). https://doi.org/10.1145/3453483.3454076, https://doi.org/10.1145/3453483.3454076
6. Bagnara, R., Rodríguez-Carbonell, E., Zaffanella, E.: Generation of basic semi-algebraic invariants using convex polyhedra. In: SAS. LNCS, vol. 3672, pp. 19–34. Springer (2005). https://doi.org/10.1007/11547662_4, https://doi.org/10.1007/11547662_4
7. Bradley, A.R., Manna, Z., Sipma, H.B.: Linear ranking with reachability. In: CAV. LNCS, vol. 3576, pp. 491–504. Springer (2005). https://doi.org/10.1007/11513988_48, https://doi.org/10.1007/11513988_48
8. Breck, J., Cyphert, J., Kincaid, Z., Reps, T.W.: Templates and recurrences: better together. In: PLDI. pp. 688–702. ACM (2020). https://doi.org/10.1145/3385412.3386035, https://doi.org/10.1145/3385412.3386035
9. Chakarov, A., Sankaranarayanan, S.: Probabilistic program analysis with martingales. In: CAV. LNCS, vol. 8044, pp. 511–526. Springer (2013). https://doi.org/10.1007/978-3-642-39799-8_34, https://doi.org/10.1007/978-3-642-39799-8_34
10. Chakarov, A., Sankaranarayanan, S.: Expectation invariants for probabilistic program loops as fixed points. In: Müller-Olm, M., Seidl, H. (eds.) SAS. LNCS, vol. 8723, pp. 85–100. Springer (2014). https://doi.org/10.1007/978-3-319-10936-7_6, https://doi.org/10.1007/978-3-319-10936-7_6
11. 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). https://doi.org/10.1145/3339984, https://doi.org/10.1145/3339984
12. 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). https://doi.org/10.1145/3385412.3385969, https://doi.org/10.1145/3385412.3385969
13. Chatterjee, K., Fu, H., Novotný, P., Hasheminezhad, R.: Algorithmic analysis of qualitative and quantitative termination problems for affine probabilistic programs. ACM Trans. Program. Lang. Syst. **40**(2), 7:1–7:45 (2018). https://doi.org/10.1145/3174800, https://doi.org/10.1145/3174800
14. Chatterjee, K., Novotný, P., Zikelic, D.: Stochastic invariants for probabilistic termination. In: Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017. pp. 145–160. ACM (2017). https://doi.org/10.1145/3009837.3009873, https://doi.org/10.1145/3009837.3009873
15. Chen, Y., Xia, B., Yang, L., Zhan, N., Zhou, C.: Discovering non-linear ranking functions by solving semi-algebraic systems. In: ICTAC. LNCS, vol. 4711, pp. 34–49. Springer (2007). https://doi.org/10.1007/978-3-540-75292-9_3, https://doi.org/10.1007/978-3-540-75292-9_3

16. Chen, Y., Hong, C., Wang, B., Zhang, L.: Counterexample-guided polynomial loop invariant generation by lagrange interpolation. In: CAV. LNCS, vol. 9206, pp. 658–674. Springer (2015). https://doi.org/10.1007/978-3-319-21690-4_44, https://doi.org/10.1007/978-3-319-21690-4_44
17. Colón, M., Sankaranarayanan, S., Sipma, H.: Linear invariant generation using non-linear constraint solving. In: CAV. LNCS, vol. 2725, pp. 420–432. Springer (2003). https://doi.org/10.1007/978-3-540-45069-6_39, https://doi.org/10.1007/978-3-540-45069-6_39
18. Colón, M., Sipma, H.: Synthesis of linear ranking functions. In: TACAS. LNCS, vol. 2031, pp. 67–81. Springer (2001). https://doi.org/10.1007/3-540-45319-9_6, https://doi.org/10.1007/3-540-45319-9_6
19. Cousot, P.: Proving program invariance and termination by parametric abstraction, lagrangian relaxation and semidefinite programming. In: VMCAI. LNCS, vol. 3385, pp. 1–24. Springer (2005). https://doi.org/10.1007/978-3-540-30579-8_1, https://doi.org/10.1007/978-3-540-30579-8_1
20. 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). https://doi.org/10.1145/512950.512973, https://doi.org/10.1145/512950.512973
21. Cousot, P., Cousot, R., Feret, J., Mauborgne, L., Miné, A., Monniaux, D., Rival, X.: The astreé analyzer. In: ESOP. LNCS, vol. 3444, pp. 21–30. Springer (2005). https://doi.org/10.1007/978-3-540-31987-0_3, https://doi.org/10.1007/978-3-540-31987-0_3
22. Cousot, P., Halbwachs, N.: Automatic discovery of linear restraints among variables of a program. In: POPL. pp. 84–96. ACM Press (1978). https://doi.org/10.1145/512760.512770, https://doi.org/10.1145/512760.512770
23. Csallner, C., Tillmann, N., Smaragdakis, Y.: Dysy: dynamic symbolic execution for invariant inference. In: ICSE. pp. 281–290. ACM (2008). https://doi.org/10.1145/1368088.1368127, https://doi.org/10.1145/1368088.1368127
24. David, C., Kesseli, P., Kroening, D., Lewis, M.: Danger invariants. In: FM. LNCS, vol. 9995, pp. 182–198 (2016). https://doi.org/10.1007/978-3-319-48989-6_12, https://doi.org/10.1007/978-3-319-48989-6_12
25. Dillig, I., Dillig, T., Li, B., McMillan, K.L.: Inductive invariant generation via abductive inference. In: OOPSLA. pp. 443–456. ACM (2013). https://doi.org/10.1145/2509136.2509511, https://doi.org/10.1145/2509136.2509511
26. Farzan, A., Kincaid, Z.: Compositional recurrence analysis. In: FMCAD. pp. 57–64. IEEE (2015)
27. Floyd, R.W.: Assigning meanings to programs. Mathematical Aspects of Computer Science **19**, 19–33 (1967)
28. Gan, T., Xia, B., Xue, B., Zhan, N., Dai, L.: Nonlinear craig interpolant generation. In: CAV. LNCS, vol. 12224, pp. 415–438. Springer (2020). https://doi.org/10.1007/978-3-030-53288-8_20, https://doi.org/10.1007/978-3-030-53288-8_20
29. Garg, P., Löding, C., Madhusudan, P., Neider, D.: ICE: A robust framework for learning invariants. In: CAV. LNCS, vol. 8559, pp. 69–87. Springer (2014). https://doi.org/10.1007/978-3-319-08867-9_5, https://doi.org/10.1007/978-3-319-08867-9_5
30. Garg, P., Neider, D., Madhusudan, P., Roth, D.: Learning invariants using decision trees and implication counterexamples. In: POPL. pp. 499–512. ACM (2016). https://doi.org/10.1145/2837614.2837664, https://doi.org/10.1145/2837614.2837664
31. Giacobazzi, R., Ranzato, F.: Completeness in abstract interpretation: A domain perspective. In: AMAST. LNCS, vol. 1349, pp. 231–245. Springer (1997). https://doi.org/10.1007/BFb0000474, https://doi.org/10.1007/BFb0000474
32. Gulwani, S., Srivastava, S., Venkatesan, R.: Constraint-based invariant inference over predicate abstraction. In: VMCAI. LNCS, vol. 5403, pp. 120–135. Springer (2009). https://doi.org/10.1007/978-3-540-93900-9_13, https://doi.org/10.1007/978-3-540-93900-9_13

33. Gupta, A., Rybalchenko, A.: Invgen: An efficient invariant generator. In: CAV. LNCS, vol. 5643, pp. 634–640. Springer (2009). https://doi.org/10.1007/978-3-642-02658-4_48, https://doi.org/10.1007/978-3-642-02658-4_48
34. He, J., Singh, G., Püschel, M., Vechev, M.T.: Learning fast and precise numerical analysis. In: PLDI. pp. 1112–1127. ACM (2020). https://doi.org/10.1145/3385412.3386016, https://doi.org/10.1145/3385412.3386016
35. Hrushovski, E., Ouaknine, J., Pouly, A., Worrell, J.: Polynomial invariants for affine programs. In: LICS. pp. 530–539. ACM (2018). https://doi.org/10.1145/3209108.3209142, https://doi.org/10.1145/3209108.3209142
36. Humenberger, A., Jaroschek, M., Kovács, L.: Automated generation of non-linear loop invariants utilizing hypergeometric sequences. In: ISSAC. pp. 221–228. ACM (2017). https://doi.org/10.1145/3087604.3087623, https://doi.org/10.1145/3087604.3087623
37. Humenberger, A., Kovács, L.: Algebra-based synthesis of loops and their invariants (invited paper). In: VMCAI. LNCS, vol. 12597, pp. 17–28. Springer (2021). https://doi.org/10.1007/978-3-030-67067-2_2, https://doi.org/10.1007/978-3-030-67067-2_2
38. 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), http://drops.dagstuhl.de/opus/volltexte/2006/511
39. Katoen, J., McIver, A., Meinicke, L., Morgan, C.C.: Linear-invariant generation for probabilistic programs: - automated support for proof-based methods. In: SAS. LNCS, vol. 6337, pp. 390–406. Springer (2010). https://doi.org/10.1007/978-3-642-15769-1_24, https://doi.org/10.1007/978-3-642-15769-1_24
40. Kincaid, Z., Breck, J., Boroujeni, A.F., Reps, T.W.: Compositional recurrence analysis revisited. In: PLDI. pp. 248–262. ACM (2017). https://doi.org/10.1145/3062341.3062373, https://doi.org/10.1145/3062341.3062373
41. Kincaid, Z., Breck, J., Cyphert, J., Reps, T.: Closed forms for numerical loops. Proceedings of the ACM on Programming Languages **3**(POPL), 1–29 (2019)
42. 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). https://doi.org/10.1145/3158142, https://doi.org/10.1145/3158142
43. Lin, W., Wu, M., Yang, Z., Zeng, Z.: Proving total correctness and generating preconditions for loop programs via symbolic-numeric computation methods. Frontiers Comput. Sci. **8**(2), 192–202 (2014)
44. Manna, Z., Pnueli, A.: Temporal verification of reactive systems - safety. Springer (1995)
45. McMillan, K.L.: Quantified invariant generation using an interpolating saturation prover. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS. LNCS, vol. 4963, pp. 413–427. Springer (2008). https://doi.org/10.1007/978-3-540-78800-3_31, https://doi.org/10.1007/978-3-540-78800-3_31
46. Müller-Olm, M., Seidl, H.: Computing polynomial program invariants. Inf. Process. Lett. **91**(5), 233–244 (2004). https://doi.org/10.1016/j.ipl.2004.05.004, https://doi.org/10.1016/j.ipl.2004.05.004
47. 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). https://doi.org/10.1109/ICSE.2012.6227149, https://doi.org/10.1109/ICSE.2012.6227149
48. de Oliveira, S., Bensalem, S., Prevosto, V.: Polynomial invariants by linear algebra. In: ATVA. LNCS, vol. 9938, pp. 479–494 (2016). https://doi.org/10.1007/978-3-319-46520-3_30, https://doi.org/10.1007/978-3-319-46520-3_30
49. de Oliveira, S., Bensalem, S., Prevosto, V.: Synthesizing invariants by solving solvable loops. In: ATVA. LNCS, vol. 10482, pp. 327–343. Springer (2017). https://doi.org/10.1007/978-3-319-68167-2_22, https://doi.org/10.1007/978-3-319-68167-2_22

50. Padon, O., McMillan, K.L., Panda, A., Sagiv, M., Shoham, S.: Ivy: safety verification by interactive generalization. In: PLDI. pp. 614–630. ACM (2016). https://doi.org/10.1145/2908080.2908118, https://doi.org/10.1145/2908080.2908118
51. Podelski, A., Rybalchenko, A.: A complete method for the synthesis of linear ranking functions. In: VMCAI. LNCS, vol. 2937, pp. 239–251. Springer (2004). https://doi.org/10.1007/978-3-540-24622-0_20, https://doi.org/10.1007/978-3-540-24622-0_20
52. Rodríguez-Carbonell, E., Kapur, D.: An abstract interpretation approach for automatic generation of polynomial invariants. In: SAS. LNCS, vol. 3148, pp. 280–295. Springer (2004). https://doi.org/10.1007/978-3-540-27864-1_21, https://doi.org/10.1007/978-3-540-27864-1_21
53. Rodríguez-Carbonell, E., Kapur, D.: Automatic Generation of Polynomial Loop Invariants: Algebraic Foundations. In: ISSAC. pp. 266–273. ACM (2004). https://doi.org/10.1145/1005285.1005324, https://doi.org/10.1145/1005285.1005324
54. Rodríguez-Carbonell, E., Kapur, D.: Automatic generation of polynomial invariants of bounded degree using abstract interpretation. Sci. Comput. Program. **64**(1), 54–75 (2007). https://doi.org/10.1016/j.scico.2006.03.003, https://doi.org/10.1016/j.scico.2006.03.003
55. Sankaranarayanan, S., Sipma, H., Manna, Z.: Non-linear loop invariant generation using gröbner bases. In: POPL. pp. 318–329. ACM (2004). https://doi.org/10.1145/964001.964028, https://doi.org/10.1145/964001.964028
56. Sankaranarayanan, S., Sipma, H.B., Manna, Z.: Constraint-based linear-relations analysis. In: SAS. LNCS, vol. 3148, pp. 53–68. Springer (2004). https://doi.org/10.1007/978-3-540-27864-1_7, https://doi.org/10.1007/978-3-540-27864-1_7
57. 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, https://doi.org/10.1007/s10703-016-0248-5
58. Sharma, R., Gupta, S., Hariharan, B., Aiken, A., Liang, P., Nori, A.V.: A data driven approach for algebraic loop invariants. In: ESOP. LNCS, vol. 7792, pp. 574–592. Springer (2013). https://doi.org/10.1007/978-3-642-37036-6_31, https://doi.org/10.1007/978-3-642-37036-6_31
59. Sting: Stanford invariant generator. http://theory.stanford.edu/ srirams/Software/sting.html (2004)
60. SV-COMP2021: 11th Competition on Software Verification (2021), `https://github.com/sosy-lab/sv-benchmarks`
61. The Sage Developers: SageMath, the Sage Mathematics Software System (Version 9.4) (2021), `https://www.sagemath.org`
62. Wang, J., Sun, Y., Fu, H., Chatterjee, K., Goharshady, A.K.: Quantitative analysis of assertion violations in probabilistic programs. In: PLDI. pp. 1171–1186. ACM (2021). https://doi.org/10.1145/3453483.3454102, https://doi.org/10.1145/3453483.3454102
63. Wang, P., Fu, H., Goharshady, A.K., Chatterjee, K., Qin, X., Shi, W.: Cost analysis of nondeterministic probabilistic programs. In: PLDI. pp. 204–220. ACM (2019). https://doi.org/10.1145/3314221.3314581, https://doi.org/10.1145/3314221.3314581
64. Xu, R., He, F., Wang, B.: Interval counterexamples for loop invariant learning. In: ESEC/FSE. pp. 111–122. ACM (2020). https://doi.org/10.1145/3368089.3409752, https://doi.org/10.1145/3368089.3409752
65. Yang, L., Zhou, C., Zhan, N., Xia, B.: Recent advances in program verification through computer algebra. Frontiers Comput. Sci. China **4**(1), 1–16 (2010). https://doi.org/10.1007/s11704-009-0074-7, https://doi.org/10.1007/s11704-009-0074-7
66. 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). https://doi.org/10.1145/3385412.3385986, https://doi.org/10.1145/3385412.3385986

A Appendix

A.1 Independence of Non-tight Invariants

By the following example, we show that the invariants produced by μ's from the interior of the feasible domain can not be written as the non-negative linear combinations of tight invariants:

Example 8. Consider the loop:

$$\begin{array}{l}\textbf{initial condition}\\ \quad \theta : \mathbf{R}\cdot\mathbf{x}+\mathbf{f} = \begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\cdot\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}+\begin{bmatrix}-1\\-2\\-3\end{bmatrix}=\mathbf{0}\\ \textbf{while } \mathbf{p}^{\mathrm{T}}\cdot\mathbf{x}+q=[-1,0,0]\cdot[x_1,x_2,x_3]^{\mathrm{T}}-10\le 0 \textbf{ do}\\ \begin{bmatrix}x_1'\\x_2'\\x_3'\end{bmatrix}=\mathbf{T}\cdot\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}+\mathbf{b}=\begin{bmatrix}2&1&-2\\1&0&0\\0&1&0\end{bmatrix}\cdot\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}+\mathbf{0}.\end{array}$$

The three eigenvalues of $\mathbf{T}$ are $1, 2, -1$, which yield the following invariants:

$$\begin{aligned}\mu=1: x_1-x_2-2x_3+7&\le 0;\\ x_1-x_2-2x_3+7&\ge 0;\\ \mu=2: x_1-x_3+2&\le 0;\\ \mu=-1: \ -1\le x_1-3x_2+2x_3&\le 1.\end{aligned}$$

(6) here is

$$\begin{aligned}\begin{bmatrix}c_1\\c_2\\c_3\end{bmatrix}&=\frac{\xi}{-\mu^3+2\mu^2+\mu-2}\begin{bmatrix}\mu^2 & \mu & 1\\ \mu-2 & \mu^2-2\mu & \mu-2\\ -2\mu & -2 & \mu^2-2\mu-1\end{bmatrix}\cdot\begin{bmatrix}-1\\0\\0\end{bmatrix}\\ &=\frac{\xi}{-\mu^3+2\mu^2+\mu-2}\begin{bmatrix}-\mu^2\\2-\mu\\2\mu\end{bmatrix}\end{aligned}$$

Inequality $(5')$ is

$$(\mu-1)d\ge 10\xi$$

Inequality $(3'')$ is

$$(\mu-1)[-1,-2,-3]\cdot\mathbf{c}=\frac{\xi(\mu-1)(\mu^2-4\mu-4)}{-\mu^3+2\mu^2+\mu-2}\ge(\mu-1)d \quad \text{when} \quad \mu\ge 1.$$

Compatibility condition (7) is

$$\begin{aligned}&\frac{(\mu-1)(\mu^2-4\mu-4)}{-\mu^3+2\mu^2+\mu-2}\ge 10 \ \Rightarrow\ \frac{11\mu^2-14\mu-24}{-\mu^2+\mu+2}\ge 0\\ \Rightarrow\ &\frac{(\mu-\frac{7-\sqrt{313}}{11})(\mu-\frac{7+\sqrt{313}}{11})}{(\mu+1)(\mu-2)}\le 0 \quad \text{when} \quad \mu\ge 1.\end{aligned}$$

The feasible domain is $[0,1) \cup (2, \frac{7+\sqrt{313}}{11}]$, the tight choices are:

$$\begin{aligned} \mu = 0: & \ -x_2 - 10 \leq 0; \\ \mu = \frac{7+\sqrt{313}}{11}: & \ \frac{362+14\sqrt{313}}{121}x_1 + \frac{\sqrt{313}-15}{11}x_2 \\ & - \frac{14+2\sqrt{313}}{11}x_3 + \frac{430+30\sqrt{313}}{121} \leq 0, \\ \text{or approximately } & 504x_1 + 24x_2 - 449x_3 + 794 \leq 0. \end{aligned}$$

We compute invariants from some other μ's:

$$\begin{aligned} \mu = 1/3: & \ x_1 - 15x_2 - 6x_3 - 200 \leq 0; \\ \mu = 2/3: & \ x_1 - 3x_3 - 3x_3 - 50 \leq 0; \\ \mu = 2.1: & \ 441x_1 + 10x_2 - 420x_3 + 799 \leq 0; \\ \mu = 2.2: & \ 484x_1 + 20x_2 - 440x_3 + 796 \leq 0. \end{aligned}$$

The reader can check that the invariants from $0 < \mu < 1$ can not be expressed by the non-negative linear combinations of the ones from $\mu = 0, 1$; the invariants from $2 < \mu < \frac{7+\sqrt{313}}{11}$ can not be expressed by the non-negative linear combinations of the ones from $\mu = 2, \frac{7+\sqrt{313}}{11}$ either. So from this example we conclude that though the μ's from the interior of the feasible domain are not tight, they still would provide independent invariants and thus could be kept as back-up choices. □

A.2 Basis of All Legitimate Invariants

We care about one question: is there a basis of invariants, such that all legitimate invariants are non-negative linear combinations of this basis? By Example 8, we already know that tight invariants can not form such a basis.

We have the following proposition to answer this question:

Proposition 11. *Consider a single-branch affine while loop with deterministic updates. There exists a basis consisting of $2n$ vectors (n is the number of variables), such that any $\mathbf{c}$ produced by some μ from the feasible domain is a non-negative linear combination of this basis. However, these basis vectors don't lead to invariants.*

Proof. Recall the formula $(6')$ for $\mathbf{c}$ produced by μ from the feasible domain (with $\mathbf{z} = \mathbf{P}^{\mathrm{T}} \cdot \boldsymbol{\xi}$):

$$\mathbf{c}(\mu) = (\mathbf{T}^{\mathrm{T}} - \mu \cdot \mathbf{I})^{-1} \cdot \mathbf{z} \textit{ is proportional to } \pm\, \mathbf{Ad}(\mu) \cdot \mathbf{z}$$

where the entries of $\mathbf{Ad}(\mu)$ are cofactors of $\mathbf{T}^{\mathrm{T}} - \mu \cdot \mathbf{I}$, thus are polynomials of μ with degree $\leq n-1$. If we do Taylor expansion for $\mathbf{Ad}(\mu)$ w.r.t μ:

$$\mathbf{Ad}(\mu) = \mathbf{A}_0 + \mu \cdot \mathbf{A}_1 + ... + \mu^{n-1} \cdot \mathbf{A}_{n-1}$$

then $\mathbf{c}$ can be written as a non-negative linear combination

$$\mathbf{c} = (\pm\mathbf{A}_0 \cdot \mathbf{z}) + \mu \cdot (\pm\mathbf{A}_1 \cdot \mathbf{z}) + ... + \mu^{n-1} \cdot (\pm\mathbf{A}_{n-1} \cdot \mathbf{z})$$

of the vectors $\{\pm\mathbf{A}_0 \cdot \mathbf{z}, ..., \pm\mathbf{A}_{n-1} \cdot \mathbf{z}\}$. Thus these $2n$ vectors form a desired basis. However, these vectors themselves are usually not invariant, because there does not always exist $\mu > 0$ such that $\mathbf{Ad}(\mu) = \mathbf{A}_i$ for $1 \leq i \leq n-1$. □

Furthermore, it's complicated to compute this basis for any specific loop. We need to calculate the coefficients $\mathbf{A}_0, \mathbf{A}_1, ..., \mathbf{A}_{n-1}$ of the Taylor expansion, which are given by the formula:

$$\mathbf{A}_i = \frac{d^i}{d\mu^i}\mathbf{Ad}(\mu)|_{\mu=0}.$$

Therefore, we can not use the basis as 'optimal' invariants.

A.3 Proof of Proposition 6

Proof (Proof of Proposition 6). If μ is a root to (8), then the two conditions (3)(5′) for d coincide:

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

otherwise, the feasible domain of d is an interval. We discuss two cases respectively:

- when $0 \leq \mu < 1$: constraints (3′)(5′) give us

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

- when $\mu \geq 1$: constraints (3″)(5′) give us

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

and the optimal value of d is one of the two boundaries of the interval. □

A.4 Proof of Theorem 2

Proof (Proof of Theorem 2).

1. By the definition of adjoint matrix, we have

$$(\mathbf{T}^{\mathrm{T}} - \mu \cdot \mathbf{I})\mathbf{Ad}(\mu) = \det(\mathbf{T}^{\mathrm{T}} - \mu \cdot \mathbf{I}) \cdot \mathbf{I}$$

for any μ. Since λ is an eigenvalue of $\mathbf{T}$, we have $\det(\mathbf{T}^{\mathrm{T}} - \lambda \cdot \mathbf{I}) = 0$, and

$$(\mathbf{T}^{\mathrm{T}} - \lambda \cdot \mathbf{I})\mathbf{Ad}(\lambda) = \det(\mathbf{T}^{\mathrm{T}} - \lambda \cdot \mathbf{I}) \cdot \mathbf{I} = [0]_{n\times n}$$

thus for any vector $\mathbf{x}$,

$$(\mathbf{T}^{\mathrm{T}} - \lambda \cdot \mathbf{I})\mathbf{Ad}(\lambda) \cdot \mathbf{x} = \mathbf{0};$$

so $\mathbf{Ad}(\lambda) \cdot \mathbf{x}$ is an eigenvector of $\mathbf{T}^{\mathrm{T}}$ with eigenvalue λ.
Using the condition that the geometric multiplicity of λ is 1, we know the rank of $\mathbf{T}^{\mathrm{T}} - \lambda \cdot \mathbf{I}$ is $n - 1$; it has at least one non-zero cofactor. Hence $\mathbf{Ad}(\lambda)$ is not zero matrix, and there exists $\mathbf{x}$ such that $\mathbf{Ad}(\lambda) \cdot \mathbf{x} \neq \mathbf{0}$.
Otherwise the geometric multiplicity of λ is larger than 1, then $\mathrm{rank}(\mathbf{T}^{\mathrm{T}} - \lambda \cdot \mathbf{I}) < n - 1$ and all its cofactors are 0. In this case $\mathbf{Ad}(\lambda) = [0]_{n\times n}$.
2. We know that every entry of $\mathbf{Ad}(\mu)$ is a cofactor of $\mathbf{T}^{\mathrm{T}} - \mu \cdot \mathbf{I}$, which is a polynomial of μ, hence continuous in μ. Thus for any $\{\lambda_i\} \to \lambda$, we have $\{\mathbf{Ad}(\lambda_i) \cdot \mathbf{x}\} \to \mathbf{Ad}(\lambda) \cdot \mathbf{x}$. □