# Loop Summarization with Rational Vector Addition Systems

Jake Silverman(✉) and Zachary Kincaid

Princeton University, Princeton, USA
{Jakers,ZKincaid}@CS.Princeton.edu

**Abstract.** This paper presents a technique for computing numerical loop summaries. The method synthesizes a rational vector addition system with resets ($\mathbb{Q}$-VASR) that simulates the action of an input loop, and then uses the reachability relation of that $\mathbb{Q}$-VASR to over-approximate the behavior of the loop. The key technical problem solved in this paper is to automatically synthesize a $\mathbb{Q}$-VASR that is a *best abstraction* of a given loop in the sense that (1) it simulates the loop and (2) it is simulated by any other $\mathbb{Q}$-VASR that simulates the loop. Since our loop summarization scheme is based on computing the *exact* reachability relation of a *best* abstraction of a loop, we can make theoretical guarantees about its behavior. Moreover, we show experimentally that the technique is precise and performant in practice.

## 1 Introduction

Modern software verification techniques employ a number of heuristics for reasoning about loops. While these heuristics are often effective, they are unpredictable. For example, an abstract interpreter may fail to find the most precise invariant expressible in the language of its abstract domain due to imprecise widening, or a software-model checker might fail to terminate because it generates interpolants that are insufficiently general. This paper presents a loop summarization technique that is capable of generating loop invariants in an expressive and decidable language and provides theoretical guarantees about invariant quality.

The key idea behind our technique is to leverage reachability results of vector addition systems (VAS) for invariant generation. Vector addition systems are a class of infinite-state transition systems with decidable reachability, classically used as a model of parallel systems [12]. We consider a variation of VAS, *rational VAS with resets (*$\mathbb{Q}$-VASR*)*, wherein there is a finite number of rational-typed variables and a finite set of transitions that simultaneously update each variable in the system by either adding a constant value or (re)setting the variable to a constant value. Our interest in $\mathbb{Q}$-VASRs stems from the fact that there is (polytime) procedure to compute a linear arithmetic formula that represents a $\mathbb{Q}$-VASR's reachability relation [8].

Since the reachability relation of a $\mathbb{Q}$-VASR is computable, the dynamics of $\mathbb{Q}$-VASR can be analyzed without relying on heuristic techniques. However,

I. Dillig and S. Tasiran (Eds.): CAV 2019, LNCS 11562, pp. 97–115, 2019.
https://doi.org/10.1007/978-3-030-25543-5_7

there is a gap between $\mathbb{Q}$-VASR and the loops that we are interested in summarizing. The latter typically use a rich set of operations (memory manipulation, conditionals, non-constant increments, non-linear arithmetic, etc) and cannot be analyzed precisely. We bridge the gap with a procedure that, for any loop, synthesizes a $\mathbb{Q}$-VASR that simulates it. The reachability relation of the $\mathbb{Q}$-VASR can then be used to over-approximate the behavior of the loop. Moreover, we prove that if a loop is expressed in linear rational arithmetic (LRA), then our procedure synthesizes a *best* $\mathbb{Q}$-VASR abstraction, in the sense that it simulates any other $\mathbb{Q}$-VASR that simulates the loop. That is, imprecision in the analysis is due to inherent limitations of the $\mathbb{Q}$-VASR model, rather heuristic algorithmic choices.

One limitation of the model is that $\mathbb{Q}$-VASRs over-approximate multi-path loops by treating the choice between paths as non-deterministic. We show that $\mathbb{Q}$-VASRS, $\mathbb{Q}$-VASR extended with control states, can be used to improve our invariant generation scheme by encoding control flow information and inter-path control dependencies that are lost in the $\mathbb{Q}$-VASR abstraction. We give an algorithm for synthesizing a $\mathbb{Q}$-VASRS abstraction of a given loop, which (like our $\mathbb{Q}$-VASR abstraction algorithm) synthesizes *best* abstractions under certain assumptions.

Finally, we note that our analysis techniques extend to complex control structures (such as nested loops) by employing summarization compositionally (i.e., "bottom-up"). For example, our analysis summarizes a nested loop by first summarizing its inner loops, and then uses the summaries to analyze the outer loop. As a result of compositionality, our analysis can be applied to partial programs, is easy to parallelize, and has the potential to scale to large code bases.

The main contributions of the paper are as follows:

- We present a procedure to synthesize $\mathbb{Q}$-VASR abstractions of transition formulas. For transition formulas in linear rational arithmetic, the synthesized $\mathbb{Q}$-VASR abstraction is a *best* abstraction.
- We present a technique for improving the precision of our analysis by using $\mathbb{Q}$-VASR with states to capture loop control structure.
- We implement the proposed loop summarization techniques and show that their ability to verify user assertions is comparable to software model checkers, while at the same time providing theoretical guarantees of termination and invariant quality.

## 1.1 Outline

This section illustrates the high-level structure of our invariant generation scheme. The goal is to compute a *transition formula* that summarizes the behavior of a given program. A transition formula is a formula over a set of program variables $\mathsf{Var}$ along with primed copies $\mathsf{Var}'$, representing the state of the program

```
procedure enqueue(elt):
  back := cons(elt,back)
  size := size + 1

procedure dequeue():
  if (front == nil) then
    // Reverse back, append to front
    while (back != nil) do
      front := cons(head(back),front)
      back := tail(back)
  result := head(front)
  front := tail(front)
  size := size - 1
  return result
```

(a) Persistent queue

```
procedure enqueue():
  back_len := back_len + 1
  mem_ops := mem_ops + 1
  size := size + 1
procedure dequeue():
  if (front_len == 0) then
    while (back_len != 0) do
      front_len := front_len + 1
      back_len := back_len - 1
      mem_ops := mem_ops + 3
  size := size - 1
  front_len := front_len - 1
  mem_ops := mem_ops + 2
procedure harness():
  nb_ops := 0
  while nondet() do
    nb_ops := nb_ops + 1
    if (size > 0 && nondet())
      enqueue()
    else
      dequeue()
```

(b) Integer model & harness

**Fig. 1.** A persistent queue and integer model. `back_len` and `front_len` models the lengths of the lists `front` and `back`; `mem_ops` counts the number of memory operations in the computation.

before and after executing a computation (respectively). For any given program $P$, a transition formula $\mathbf{TF}[\![P]\!]$ can be computed by recursion on syntax:[1]

$$\mathbf{TF}[\![\texttt{x := } e]\!] \triangleq \texttt{x}' = e \wedge \bigwedge_{\texttt{y} \neq \texttt{x} \in \mathsf{Var}} \texttt{y}' = \texttt{y}$$

$$\mathbf{TF}[\![\textbf{if } c \textbf{ then } P_1 \textbf{ else } P_2]\!] \triangleq (c \wedge \mathbf{TF}[\![P_1]\!]) \vee (\neg c \wedge \mathbf{TF}[\![P_2]\!])$$

$$\mathbf{TF}[\![P_1 ; P_2]\!] \triangleq \exists X \in \mathbb{Z}. \mathbf{TF}[\![P_1]\!][\mathsf{Var}' \mapsto X] \wedge \mathbf{TF}[\![P_2]\!][\mathsf{Var} \mapsto X]$$

$$\mathbf{TF}[\![\textbf{while } c \textbf{ do } P]\!] \triangleq (c \wedge \mathbf{TF}[\![P]\!])^\star \wedge (\neg c[\mathsf{Var} \mapsto \mathsf{Var}'])$$

where $(-)^\star$ is a function that computes an over-approximation of the transitive closure of a transition formula. The contribution of this paper is a method for computing this $(-)^\star$ operation, which is based on first over-approximating the input transition formula by a $\mathbb{Q}$-VASR, and then computing the (exact) reachability relation of the $\mathbb{Q}$-VASR.

[1] This style of analysis can be extended from a simple block-structured language to one with control flow and recursive procedures using the framework of algebraic program analysis [13, 23].

We illustrate the analysis on an integer model of a persistent queue data structure, pictured in Fig. 1. The example consists of two operations (`enqueue` and `dequeue`), as well as a test harness (`harness`) that non-deterministically executes `enqueue` and `dequeue` operations. The queue achieves $O(1)$ amortized memory operations (`mem_ops`) in `enqueue` and `queue` by implementing the queue as two lists, `front` and `back` (whose lengths are modeled as `front_len` and `back_len`, respectively): the sequence of elements in the queue is the `front` list followed by the reverse of the `back` list. We will show that the queue functions use $O(1)$ amortized memory operations by finding a summary for `harness` that implies a linear bound on `mem_ops` (the number of memory operations in the computation) in terms of `nb_ops` (the total number of `enqueue`/`dequeue` operations executed in some sequence of operations).

We analyze the queue compositionally, in "bottom-up" fashion (i.e., starting from deeply-nested code and working our way back up to a summary for `harness`). There are two loops of interest, one in `dequeue` and one in `harness`. Since the `dequeue` loop is nested inside the `harness` loop, `dequeue` is analyzed first. We start by computing a transition formula that represents one execution of the body of the `dequeue` loop:

$$\mathit{Body}_{\texttt{deq}} = \texttt{back\_len} > 0 \land \begin{pmatrix} \quad\texttt{front\_len}' = \texttt{front\_len} + 1 \\ \land\,\texttt{back\_len}' = \texttt{back\_len} - 1 \\ \land\,\texttt{mem\_ops}' = \texttt{mem\_ops} + 3 \\ \land\,\texttt{size}' = \texttt{size} \end{pmatrix}$$

Observe that each variable in the loop is incremented by a constant value. As a result, the loop update can be captured faithfully by a vector addition system. In particular, we see that this loop body formula is simulated by the $\mathbb{Q}$-VASR $V_{\texttt{deq}}$ (below), where the correspondence between the state-space of $\mathit{Body}_{\texttt{deq}}$ and $V_{\texttt{deq}}$ is given by the identity transformation (i.e., each dimension of $V_{\texttt{deq}}$ simply represents one of the variables of $\mathit{Body}_{\texttt{deq}}$).

$$\begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{bmatrix} \begin{bmatrix} \texttt{front\_len} \\ \texttt{back\_len} \\ \texttt{mem\_ops} \\ \texttt{size} \end{bmatrix} ; \quad V_{\texttt{deq}} = \left\{ \begin{bmatrix} w \\ x \\ y \\ z \end{bmatrix} \to \begin{bmatrix} w+1 \\ x-1 \\ y+3 \\ z \end{bmatrix} \right\}.$$

A formula representing the reachability relation of a vector addition system can be computed in polytime. For the case of $V_{\texttt{deq}}$, a formula representing $k$ steps of the $\mathbb{Q}$-VASR is simply

$$w' = w + k \land x' = x - k \land y' = y + 3k \land z' = z. \tag{†}$$

To capture information about the pre-condition of the loop, we can project the primed variables to obtain $\texttt{back\_len} > 0$; similarly, for the post-condition, we can project the unprimed variables to obtain $\texttt{back\_len}' \geq 0$. Finally, combining (†)

(translated back into the vocabulary of the program) and the pre/post-condition, we form the following approximation of the `dequeue` loop's behavior:

$$\exists k . k \geq 0 \wedge \left( \begin{array}{l} \texttt{front\_len}' = \texttt{front\_len} + k \\ \wedge \texttt{back\_len}' = \texttt{back\_len} - k \\ \wedge \texttt{mem\_ops}' = \texttt{mem\_ops} + 3k \\ \wedge \texttt{size}' = \texttt{size} \end{array} \right) \wedge \left( k > 0 \Rightarrow \left( \begin{array}{c} \texttt{back\_len} > 0 \\ \wedge \texttt{back\_len}' \geq 0) \end{array} \right) \right).$$

Using this summary for the `dequeue` loop, we proceed to compute a transition formula for the body of the `harness` loop (omitted for brevity). Just as with the `dequeue` loop, we analyze the `harness` loop by synthesizing a $\mathbb{Q}$-VASR that simulates it, $V_{\texttt{har}}$ (below), where the correspondence between the state space of the `harness` loop and $V_{\texttt{har}}$ is given by the transformation $S_{\texttt{har}}$:

$$\begin{bmatrix} v \\ w \\ x \\ y \\ z \end{bmatrix} = \underbrace{\begin{bmatrix} 0\,0\,0\,1\,0 \\ 0\,1\,0\,0\,0 \\ 0\,3\,1\,0\,0 \\ 1\,1\,0\,0\,0 \\ 0\,0\,0\,0\,1 \end{bmatrix}}_{S_{\text{har}}} \begin{bmatrix} \texttt{front\_len} \\ \texttt{back\_len} \\ \texttt{mem\_ops} \\ \texttt{size} \\ \texttt{nb\_ops} \end{bmatrix} ; i.e., \left( \begin{array}{l} \quad \texttt{size} = v \\ \wedge\ \texttt{back\_len} = w \\ \wedge\ \texttt{mem\_ops} + 3\texttt{back\_len} = x \\ \wedge\ \texttt{back\_len} + \texttt{front\_len} = y \\ \wedge\ \texttt{nb\_ops} = z \end{array} \right).$$

$$V_{\texttt{har}} = \left\{ \underbrace{\begin{bmatrix} v \\ w \\ x \\ y \\ z \end{bmatrix} \to \begin{bmatrix} v+1 \\ w+1 \\ x+4 \\ y+1 \\ z+1 \end{bmatrix}}_{\texttt{enqueue}}, \underbrace{\begin{bmatrix} v \\ w \\ x \\ y \\ z \end{bmatrix} \to \begin{bmatrix} v-1 \\ w \\ x+2 \\ y-1 \\ z+1 \end{bmatrix}}_{\texttt{dequeue}\text{ fast}}, \underbrace{\begin{bmatrix} v \\ w \\ x \\ y \\ z \end{bmatrix} \to \begin{bmatrix} v-1 \\ 0 \\ x+2 \\ y-1 \\ z+1 \end{bmatrix}}_{\texttt{dequeue}\text{ slow}} \right\}$$

Unlike the `dequeue` loop, we do not get an exact characterization of the dynamics of each changed variable. In particular, in the slow `dequeue` path through the loop, the value of `front_len`, `back_len`, and `mem_ops` change by a variable amount. Since `back_len` is set to 0, its behavior can be captured by a reset. The dynamics of `front_len` and `mem_ops` cannot be captured by a $\mathbb{Q}$-VASR, but (using our `dequeue` summary) we can observe that the sum of `front_len` + `back_len` is decremented by 1, and the sum of `mem_ops` + 3`back_len` is incremented by 2.

We compute the following formula that captures the reachability relation of $V_{\texttt{har}}$ (taking $k_1$ steps of `enqueue`, $k_2$ steps of `dequeue` fast, and $k_3$ steps of `dequeue` slow) under the inverse image of the state correspondence $S_{\texttt{har}}$:

$$\left( \begin{array}{l} \texttt{size}' = \texttt{size} + k_1 - k_2 - k_3 \\ \wedge ((k_3 = 0 \wedge \texttt{back\_len}' = \texttt{back\_len} + k_1) \vee (k_3 > 0 \wedge 0 \leq \texttt{back\_len}' \leq k_1)) \\ \wedge \texttt{mem\_ops}' + 3\texttt{back\_len}' = \texttt{mem\_ops} + 3\texttt{back\_len} + 4k_1 + 2k_2 + 2k_3 \\ \wedge \texttt{front\_len}' + \texttt{back\_len}' = \texttt{front\_len} + \texttt{back\_len} + k_1 - k_2 - k_3 \\ \wedge \texttt{nb\_ops}' = \texttt{nb\_ops} + k_1 + k_2 + k_3 \end{array} \right)$$

From the above formula (along with pre/post-condition formulas), we obtain a summary for the `harness` loop (omitted for brevity). Using this summary we can prove (supposing that we start in a state where all variables are zero) that `mem_ops` is at most 4 times `nb_ops` (i.e., `enqueue` and `dequeue` use O(1) amortized memory operations).

## 2 Background

The syntax of ∃LIRA, the existential fragment of linear integer/rational arithmetic, is given by the following grammar:

$$\begin{aligned} s,t \in \mathsf{Term} &::= c \mid x \mid s+t \mid c\cdot t \\ F,G \in \mathsf{Formula} &::= s<t \mid s=t \mid F\wedge G \mid F\vee G \mid \exists x\in\mathbb{Q}.F \mid \exists x\in\mathbb{Z}.F \end{aligned}$$

where $x$ is a (rational sorted) variable symbol and $c$ is a rational constant. Observe that (without loss of generality) formulas are free of negation. ∃LRA (linear rational arithmetic) refers to the fragment of ∃LIRA that omits quantification over the integer sort.

A **transition system** is a pair $(S, \rightarrow)$ where $S$ is a (potentially infinite) set of states and $\rightarrow \subseteq S \times S$ is a transition relation. For a transition relation $\rightarrow$, we use $\rightarrow^*$ to denote its reflexive, transitive closure.

A **transition formula** is a formula $F(\mathbf{x}, \mathbf{x}')$ whose free variables range over $\mathbf{x} = x_1, ..., x_n$ and $\mathbf{x}' = x'_1, ..., x'_n$ (we refer to the number $n$ as the *dimension* of $F$); these variables designate the state before and after a transition. In the following, we assume that transition formulas are defined over ∃LIRA. For a transition formula $F(\mathbf{x}, \mathbf{x}')$ and vectors of terms $\mathbf{s}$ and $\mathbf{t}$, we use $F(\mathbf{s}, \mathbf{t})$ to denote the formula $F$ with each $x_i$ replaced by $s_i$ and each $x'_i$ replaced by $t_i$. A transition formula $F(\mathbf{x}, \mathbf{x}')$ defines a transition system $(S_F, \rightarrow_F)$, where the state space $S_F$ is $\mathbb{Q}^n$ and which can transition $\mathbf{u} \rightarrow_F \mathbf{v}$ iff $F(\mathbf{u}, \mathbf{v})$ is valid.

For two rational vectors $\mathbf{a}$ and $\mathbf{b}$ of the same dimension $d$, we use $\mathbf{a}\cdot\mathbf{b}$ to denote the inner product $\mathbf{a}\cdot\mathbf{b} = \sum_{i=1}^{d} a_i b_i$ and $\mathbf{a}*\mathbf{b}$ to denote the pointwise (aka Hadamard) product $(\mathbf{a}*\mathbf{b})_i = a_i b_i$. For any natural number $i$, we use $\mathbf{e}_i$ to denote the standard basis vector in the $i$th direction (i.e., the vector consisting of all zeros except the $i$th entry, which is 1), where the dimension of $\mathbf{e}_i$ is understood from context. We use $I_n$ to denote the $n \times n$ identity matrix.

**Definition 1.** *A* ***rational vector addition system with resets*** *(*$\mathbb{Q}$*-****VASR****) of dimension $d$ is a finite set $V \subseteq \{0,1\}^d \times \mathbb{Q}^d$ of transformers. Each transformer $(\mathbf{r}, \mathbf{a}) \in V$ consists of a binary reset vector $\mathbf{r}$, and a rational addition vector $\mathbf{a}$, both of dimension $d$. $V$ defines a transition system $(S_V, \rightarrow_V)$, where the state space $S_V$ is $\mathbb{Q}^d$ and which can transition $\mathbf{u} \rightarrow_V \mathbf{v}$ iff $\mathbf{v} = \mathbf{r} * \mathbf{u} + \mathbf{a}$ for some $(\mathbf{r}, \mathbf{a}) \in V$.*

**Definition 2.** *A* ***rational vector addition system with resets and states*** *($\mathbb{Q}$-VASRS) of dimension $d$ is a pair $\mathcal{V} = (Q, E)$, where $Q$ is a finite set of control states, and $E \subseteq Q \times \{0,1\}^d \times \mathbb{Q}^d \times Q$ is a finite set of edges labeled by (d-dimensional) transformers. $\mathcal{V}$ defines a transition system $(S_{\mathcal{V}}, \rightarrow_{\mathcal{V}})$, where the state space $S_{\mathcal{V}}$ is $Q \times \mathbb{Q}^n$ and which can transition $(q_1, \mathbf{u}) \rightarrow_{\mathcal{V}} (q_2, \mathbf{v})$ iff there is some edge $(q_1, (\mathbf{r}, \mathbf{a}), q_2) \in E$ such that $\mathbf{v} = \mathbf{r} * \mathbf{u} + \mathbf{a}$.*

Our invariant generation scheme is based on the following result, which is a simple consequence of the work of Haase and Halfon:

**Theorem 1** **([8]).** *There is a polytime algorithm which, given a $d$-dimensional $\mathbb{Q}$-VASRS $\mathcal{V} = (Q, E)$, computes an $\exists$LIRA transition formula reach$(\mathcal{V})$ such that for all $\mathbf{u}, \mathbf{v} \in \mathbb{Q}^d$, we have $(p, \mathbf{u}) \to_{\mathcal{V}}^* (q, \mathbf{v})$ for some control states $p, q \in Q$ if and only if $\mathbf{u} \to_{reach(\mathcal{V})} \mathbf{v}$.*

Note that $\mathbb{Q}$-VASR can be realized as $\mathbb{Q}$-VASRS with a single control state, so this theorem also applies to $\mathbb{Q}$-VASR.

## 3 Approximating Loops with Vector Addition Systems

In this section, we describe a method for over-approximating the transitive closure of a transition formula using a $\mathbb{Q}$-VASR. This procedure immediately extends to computing summaries for programs (including programs with nested loops) using the method outlined in Sect. 1.1.

The core algorithmic problem that we answer in this section is: *given a transition formula, how can we synthesize a (best) abstraction of that formula's dynamics as a* $\mathbb{Q}$-VASR*?* We begin by formalizing the problem: in particular, we define what it means for a $\mathbb{Q}$-VASR to simulate a transition formula and what it means for an abstraction to be "best."

**Definition 3.** *Let $A = (\mathbb{Q}^n, \to_A)$ and $B = (\mathbb{Q}^m, \to_B)$ be transition systems operating over rational vector spaces. A **linear simulation** from $A$ to $B$ is a linear transformation $S : \mathbb{Q}^{m \times n}$ such that for all $\mathbf{u}, \mathbf{v} \in \mathbb{Q}^n$ for which $\mathbf{u} \to_A \mathbf{v}$, we have $S\mathbf{u} \to_B S\mathbf{v}$. We use $A \Vdash_S B$ to denote that $S$ is a linear simulation from $A$ to $B$.*

Suppose that $F(\mathbf{x}, \mathbf{x}')$ is an $n$-dimensional transition formula, $V$ is a $d$-dimensional $\mathbb{Q}$-VASR, and $S : \mathbb{Q}^{d \times n}$ is linear transformation. The key property of simulations that underlies our loop summarization scheme is that if $F \Vdash_S V$, then *reach*$(V)(S\mathbf{x}, S\mathbf{x}')$ (i.e., the reachability relation of $V$ under the inverse image of $S$) over-approximates the transitive closure of $F$. Finally, we observe that simulation $F \Vdash_S V$ can equivalently be defined by the validity of the entailment $F \models \gamma(S, V)$, where

$$\gamma(S, V) \triangleq \bigvee_{(\mathbf{r}, \mathbf{a}) \in V} S\mathbf{x}' = \mathbf{r} * S\mathbf{x} + \mathbf{a}$$

is a transition formula that represents the transitions that $V$ simulates under transformation $S$.

Our task is to synthesize a linear transformation $S$ and a $\mathbb{Q}$-VASR $V$ such that $F \Vdash_S V$. We call a pair $(S, V)$, consisting of a rational matrix $S \in \mathbb{Q}^{d \times n}$ and a $d$-dimensional $\mathbb{Q}$-VASR $V$, a $\mathbb{Q}$-**VASR abstraction**. We say that $n$ is the *concrete dimension* of $(S, V)$ and $d$ is the *abstract dimension*. If $F \Vdash_S V$, then we say that $(S, V)$ is a $\mathbb{Q}$-**VASR abstraction of** $F$. A transition formula may have many $\mathbb{Q}$-VASR abstractions; we are interested in computing a $\mathbb{Q}$-VASR abstraction $(S, V)$ that results in the most precise over-approximation of the transitive closure of $F$. Towards this end, we define a preorder $\preceq$ on $\mathbb{Q}$-VASR abstractions, where $(S^1, V^1) \preceq (S^2, V^2)$ iff there exists a linear transformation $T \in \mathbb{Q}^{e \times d}$ such that $V^1 \Vdash_T V^2$ and $TS^1 = S^2$ (where $d$ and $e$ are the abstract dimensions of $(S^1, V^1)$ and $(S^2, V^2)$, respectively). Observe that if $(S^1, V^1) \preceq (S^2, V^2)$, then $\mathit{reach}(V^1)(S^1\mathbf{x}, S^1\mathbf{x}') \models \mathit{reach}(V^2)(S^2\mathbf{x}, S^2\mathbf{x}')$.

Thus, our problem can be stated as follows: given a transition formula $F$, synthesize a $\mathbb{Q}$-VASR abstraction $(S, V)$ of $F$ such that $(S, V)$ is *best* in the sense that we have $(S, V) \preceq (\widetilde{S}, \widetilde{V})$ for any $\mathbb{Q}$-VASR abstraction $(\widetilde{S}, \widetilde{V})$ of $F$. A solution to this problem is given in Algorithm 1.

**Algorithm 1.** `abstract-VASR`(F)

```
   input  : Transition formula $F$ of dimension $n$
   output: $\mathbb{Q}$-VASR abstraction of $F$; Best $\mathbb{Q}$-VASR abstraction if $F$ in $\exists$LRA
1  Skolemize existentials of $F$;
2  $(S, V) \leftarrow (I_n, \emptyset)$;                    // $(I_n, \emptyset)$ is least in $\preceq$ order
3  $\Gamma \leftarrow F$;
4  while $\Gamma$ is satisfiable do
5      Let $M$ be a model of $\Gamma$;
6      $C \leftarrow$ cube of the DNF of $F$ with $M \models C$;
7      $(S, V) \leftarrow (S, V) \sqcup \hat{\alpha}(C)$;
8      $\Gamma \leftarrow \Gamma \land \neg\gamma(S, V)$
9  return $(S, V)$
```

Algorithm 1 follows the familiar pattern of an AllSat-style loop. The algorithm takes as input a transition formula $F$. It maintains a $\mathbb{Q}$-VASR abstraction $(S, V)$ and a formula $\Gamma$, whose models correspond to the transitions of $F$ that are *not* simulated by $(S, V)$. The idea is to build $(S, V)$ iteratively by sampling transitions from $\Gamma$, augmenting $(S, V)$ to simulate the sample transition, and then updating $\Gamma$ accordingly. We initialize $(S, V)$ to be $(I_n, \emptyset)$, the canonical least $\mathbb{Q}$-VASR abstraction in $\preceq$ order, and $\Gamma$ to be $F$ (i.e., $(I_n, \emptyset)$ does not simulate any transitions of $F$). Each loop iteration proceeds as follows. First, we sample a model $M$ of $\Gamma$ (i.e., a transition that is allowed by $F$ but not simulated by $(S, V)$). We then generalize that transition to a set of transitions by using $M$ to select a cube $C$ of the DNF of $F$ that contains $M$. Next, we use the procedure described in Sect. 3.1 to compute a $\mathbb{Q}$-VASR abstraction $\hat{\alpha}(C)$ that simulates the transitions of $C$. We then update the $\mathbb{Q}$-VASR abstraction $(S, V)$ to be the least upper bound of $(S, V)$ and $\hat{\alpha}(C)$ (w.r.t. $\preceq$ order) using the procedure described in Sect. 3.2 (line 7). Finally, we block any transition simulated by the least upper bound (including every transition in $C$) from being sampled again by conjoining $\neg\gamma(S, V)$ to $\Gamma$. The loop terminates when $\Gamma$ is unsatisfiable, in which case we have that $F \Vdash_S V$. Theorem 2 gives the correctness statement for this algorithm.

**Theorem 2.** *Given a transition formula $F$, Algorithm 1 computes a simulation $S$ and $\mathbb{Q}$-VASR $V$ such that $F \Vdash_S V$. Moreover, if $F$ is in $\exists$LRA, Algorithm 1 computes a* best *$\mathbb{Q}$-VASR abstraction of $F$.*

The proof of this theorem as well as the proofs to all subsequent theorems, lemmas, and propositions are in the extended version of this paper [20].

## 3.1 Abstracting Conjunctive Transition Formulas

This section shows how to compute a $\mathbb{Q}$-VASR abstraction for a consistent *conjunctive* formula. When the input formula is in $\exists$LRA, the computed $\mathbb{Q}$-VASR abstraction will be a best $\mathbb{Q}$-VASR abstraction of the input formula. The intuition is that, since $\exists$LRA is a convex theory, a best $\mathbb{Q}$-VASR abstraction consists of a single transition. For $\exists$LIRA formulas, our procedure produces a $\mathbb{Q}$-VASR abstract that is not guaranteed to be best, precisely because $\exists$LIRA is not convex.

Let $C$ be consistent, conjunctive transition formula. Observe that the set $Res_C \triangleq \{\langle \mathbf{s}, a\rangle : C \models \mathbf{s} \cdot \mathbf{x}' = a\}$, which represents linear combinations of variables that are *reset* across $C$, forms a vector space. Similarly, the set $Inc_C = \{\langle \mathbf{s}, a\rangle : C \models \mathbf{s} \cdot \mathbf{x}' = \mathbf{s} \cdot \mathbf{x} + a\}$, which represents linear combinations of variables that are *incremented* across $C$, forms a vector space. We compute bases for both $Res_C$ and $Inc_C$, say $\{\langle \mathbf{s}_1, a_1\rangle, ..., \langle \mathbf{s}_m, a_m\rangle\}$ and $\{\langle \mathbf{s}_{m+1}, a_{m+1}\rangle, ..., \langle \mathbf{s}_d, a_d\rangle\}$, respectively. We define $\hat{\alpha}(C)$ to be the $\mathbb{Q}$-VASR abstraction $\hat{\alpha}(C) \triangleq (S, \{(\mathbf{r}, \mathbf{a})\})$, where

$$S \triangleq \begin{bmatrix} \mathbf{s}_1 \\ \vdots \\ \mathbf{s}_d \end{bmatrix} \quad \mathbf{r} \triangleq [\underbrace{0 \cdots 0}_{m \text{ times}} \ \overbrace{1 \cdots 1}^{(d-m) \text{ times}}] \quad \mathbf{a} \triangleq \begin{bmatrix} a_1 \\ \vdots \\ a_d \end{bmatrix}.$$

*Example 1.* Let $C$ be the formula $x' = x + y \wedge y' = 2y \wedge w' = w \wedge w = w + 1 \wedge z' = w$. The vector space of resets has basis $\{\langle \begin{bmatrix} 0 & 0 & -1 & 1 \end{bmatrix}, 0\rangle\}$ (representing that $z - w$ is reset to 0). The vector space of increments has basis $\{\langle \begin{bmatrix} 1 & -1 & 0 & 0 \end{bmatrix}, 0\rangle, \langle \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}, 0\rangle, \langle \begin{bmatrix} 0 & 0 & -1 & 1 \end{bmatrix}, 1\rangle\}$ (representing that the difference $x - y$ does not change, the difference $z - w$ increases by 1, and the variable $w$ does not change). A best abstraction of $C$ is thus the four-dimensional $\mathbb{Q}$-VASR

$$V = \left\{ \left( \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right) \right\}, S = \begin{bmatrix} 0 & 0 & -1 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}.$$

In particular, notice that since the term $z - w$ is both incremented and reset, it is represented by two different dimensions in $\hat{\alpha}(C)$.

**Proposition 1.** *For any consistent, conjunctive transition formula $C$, $\hat{\alpha}(C)$ is a $\mathbb{Q}$-VASR abstraction of $C$. If $C$ is expressed in $\exists$LRA, then $\hat{\alpha}(C)$ is best.*

## 3.2 Computing Least Upper Bounds

This section shows how to compute least upper bounds w.r.t. the $\preceq$ order.

By definition of the $\preceq$ order, if $(S, V)$ is an upper bound of $(S^1, V^1)$ and $(S^2, V^2)$, then there must exist matrices $T^1$ and $T^2$ such that $T^1S^1 = S = T^2S^2$, $V^1 \Vdash_{T^1} V$, and $V^2 \Vdash_{T^2} V$. As we shall see, if $(S, V)$ is a *least* upper bound, then it is completely determined by the matrices $T^1$ and $T^2$. Thus, we shift our attention to computing simulation matrices $T^1$ and $T^2$ that induce a least upper bound.

In view of the desired equation $T^1S^1 = S = T^2S^2$, let us consider the constraint $T^1S^1 = T^2S^2$ on two *unknown* matrices $T^1$ and $T^2$. Clearly, we have $T^1S^1 = T^2S^2$ iff each $(T^1_i, T^2_i)$ belongs to the set $\mathcal{T} \triangleq \{(\mathbf{t}^1, \mathbf{t}^2) : \mathbf{t}^1S^1 = \mathbf{t}^2S^2\}$. Observe that $\mathcal{T}$ is a vector space, so there is a *best* solution to the constraint $T^1S^1 = T^2S^2$: choose $T^1$ and $T^2$ so that the set of all row pairs $(T^1_i, T^2_i)$ forms a basis for $\mathcal{T}$. In the following, we use *pushout*$(S^1, S^2)$ to denote a function that computes such a *best* $(T^1, T^2)$.

While *pushout* gives a *best* solution to the equation $T^1S^1 = T^2S^2$, it is not sufficient for the purpose of computing least upper bounds for $\mathbb{Q}$-VASR abstractions, because $T^1$ and $T^2$ may not respect the structure of the $\mathbb{Q}$-VASR $V^1$ and $V^2$ (i.e., there may be no $\mathbb{Q}$-VASR $V$ such that $V^1 \Vdash_{T^1} V$ and $V^2 \Vdash_{T^2} V$). Thus, we must further constrain our problem by requiring that $T^1$ and $T^2$ are *coherent* with respect to $V^1$ and $V^2$ (respectively).

**Definition 4.** *Let $V$ be a $d$-dimensional $\mathbb{Q}$-VASR. We say that $i, j \in \{1, ..., d\}$ are* ***coherent dimensions*** *of $V$ if for all transitions $(\mathbf{r}, \mathbf{a}) \in V$ we have $r_i = r_j$ (i.e., every transition of $V$ that resets $i$ also resets $j$ and vice versa). We denote that $i$ and $j$ are coherent dimensions of $V$ by writing $i \equiv_V j$, and observe that $\equiv_V$ forms an equivalence relation on $\{1, ..., d\}$. We refer to the equivalence classes of $\equiv_V$ as the* ***coherence classes*** *of $V$.*

*A matrix $T \in \mathbb{Q}^{e \times d}$* ***is coherent with respect to*** *$V$ if and only if each of its rows have non-zero values only in the dimensions corresponding to a single coherence class of $V$.*

For any $d$-dimensional $\mathbb{Q}$-VASR $V$ and coherence class $C = \{c_1, ..., c_k\}$ of $V$, define $\Pi_C$ to be the $k \times d$ dimensional matrix whose rows are $\mathbf{e}_{c_1}, ..., \mathbf{e}_{c_k}$. Intuitively, $\Pi_C$ is a projection onto the set of dimensions in $C$.

Coherence is a necessary and sufficient condition for linear simulations between $\mathbb{Q}$-VASR in a sense described in Lemmas 1 and 2.

**Lemma 1.** *Let $V^1$ and $V^2$ be $\mathbb{Q}$-VASR (of dimension $d$ and $e$, respectively), and let $T \in \mathbb{Q}^{e \times d}$ be a matrix such that $V^1 \Vdash_T V^2$. Then $T$ must be coherent with respect to $V^1$.*

Let $V$ be a $d$-dimensional $\mathbb{Q}$-VASR and let $T \in \mathbb{Q}^{e \times d}$ be a matrix that is coherent with respect to $V$ and has no zero rows. Then there is a (unique) $e$-dimensional $\mathbb{Q}$-VASR *image*$(V, T)$ such that its transition relation $\rightarrow_{image(V,T)}$

**Algorithm 2.** $(S^1, V^1) \sqcup (S^2, V^2)$

```
input : Normal $\mathbb{Q}$-VASR abstractions $(S^1, V^1)$ and $(S^2, V^2)$ of equal concrete
        dimension
output: Least upper bound (w.r.t. $\preceq$) of $(S^1, V^2)$ and $(S^1, V^2)$
1 $S, T^1, T^2 \leftarrow$ empty matrices;
2 foreach coherence class $C^1$ of $V^1$ do
3     foreach coherence class $C^2$ of $V^2$ do
4         $(U^1, U^2) \leftarrow pushout(\Pi_{C^1} S^1, \Pi_{C^2} S^2)$;
5         $S \leftarrow \begin{bmatrix} S \\ U^1 \Pi_{C^1} S^1 \end{bmatrix}$; $T^1 \leftarrow \begin{bmatrix} T^1 \\ U^1 \Pi_{C^1} \end{bmatrix}$; $T^2 \leftarrow \begin{bmatrix} T^2 \\ U^2 \Pi_{C^2} \end{bmatrix}$;
6 $V \leftarrow image(V^1, T^1) \cup image(V^2, T^2)$;
7 return $(S, V)$
```

is equal to $\{(T\mathbf{u}, T\mathbf{v}) : \mathbf{u} \rightarrow_V \mathbf{v}\}$ (the image of $V$'s transition relation under $T$). This $\mathbb{Q}$-VASR can be defined by:

$$image(V, T) \triangleq \{(T \boxtimes \mathbf{r}, T\mathbf{a}) : (\mathbf{r}, \mathbf{a}) \in V\}$$

where $T \boxtimes \mathbf{r}$ is the reset vector $\mathbf{r}$ translated along $T$ (i.e., $(T \boxtimes \mathbf{r})_i = r_j$ where $j$ is an arbitrary choice among dimensions for which $T_{ij}$ is non-zero—at least one such $j$ exists because the row $T_i$ is non-zero by assumption, and the choice of $j$ is arbitrary because all such $j$ belong to the same coherence class by the assumption that $T$ is coherent with respect to $V$).

**Lemma 2.** *Let $V$ be a $d$-dimensional $\mathbb{Q}$-VASR and let $T \in \mathbb{Q}^{e \times d}$ be a matrix that is coherent with respect to $V$ and has no zero rows. Then the transition relation of $image(V, T)$ is the image of $V$'s transition relation under $T$ (i.e., $\rightarrow_{image(V,T)}$ is equal to $\{(T\mathbf{u}, T\mathbf{v}) : \mathbf{u} \rightarrow_V \mathbf{v}\}$).*

Finally, prior to describing our least upper bound algorithm, we must define a technical condition that is both assumed and preserved by the procedure:

**Definition 5.** *A $\mathbb{Q}$-VASR abstraction $(S, V)$ is **normal** if there is no non-zero vector $\mathbf{z}$ that is coherent with respect to $V$ such that $\mathbf{z}S = 0$ (i.e., the rows of $S$ that correspond to any coherence class of $V$ are linearly independent).*

Intuitively, a $\mathbb{Q}$-VASR abstraction that is *not* normal contains information that is either inconsistent or redundant.

We now present a strategy for computing least upper bounds of $\mathbb{Q}$-VASR abstractions. Fix (normal) $\mathbb{Q}$-VASR abstractions $(S^1, V^1)$ and $(S^2, V^2)$. Lemmas 1 and 2 together show that a pair of matrices $\widetilde{T}^1$ and $\widetilde{T}^2$ induce an upper bound (not necessarily *least*) on $(S^1, V^1)$ and $(S^2, V^2)$ exactly when the following conditions hold: (1) $\widetilde{T}^1 S^1 = \widetilde{T}^2 S^2$, (2) $\widetilde{T}^1$ is coherent w.r.t. $V^1$, (3) $\widetilde{T}^2$ is coherent w.r.t. $V^2$, and (4) neither $\widetilde{T}^1$ nor $\widetilde{T}^2$ contain zero rows. The upper bound induced by $\widetilde{T}^1$ and $\widetilde{T}^2$ is given by

$$ub(\widetilde{T}^1, \widetilde{T}^2) \triangleq (\widetilde{T}^1 S^1, image(V^1, \widetilde{T}^1) \cup image(V^2, T^2)).$$

We now consider how to compute a *best* such $\widetilde{T}^1$ and $\widetilde{T}^2$. Observe that conditions (1), (2), and (3) hold exactly when for each row $i$, $(\widetilde{T}^1_i, \widetilde{T}^2_i)$ belongs to the set

$$\mathcal{T} \triangleq \{(\mathbf{t}^1, \mathbf{t}^2) : \mathbf{t}^1 S^1 = \mathbf{t}^2 S^2 \wedge \mathbf{t}^1 \textit{coherent w.r.t. } V^1 \wedge \mathbf{t}^1 \textit{coherent w.r.t. } V^2\}.$$

Since a row vector $\mathbf{t}^i$ is coherent w.r.t. $V^i$ iff its non-zero positions belong to the same coherence class of $V^i$ (equivalently, $\mathbf{t}^i = \mathbf{u}\Pi_{C^i}$ for some coherence class $C^i$ and vector $\mathbf{u}$), we have $\mathcal{T} = \bigcup_{C^1,C^2} \mathcal{T}(C^1, C^2)$, where the union is over all coherence classes $C^1$ of $V^1$ and $C^2$ of $V^2$, and

$$\mathcal{T}(C^1, C^2) \triangleq \{(\mathbf{u}^1 \Pi_{C^1}, \mathbf{u}^2 \Pi_{C^2}) : \mathbf{u}^1 \Pi_{C^1} S^1 = \mathbf{u}^2 \Pi_{C^2} S^2\}.$$

Observe that each $\mathcal{T}(C^1, C^2)$ is a vector space, so we can compute a pair of matrices $T^1$ and $T^2$ such that the rows $(T^1_i, T^2_i)$ collectively form a basis for each $\mathcal{T}(C^1, C^2)$. Since $(S^1, V^1)$ and $(S^2, V^2)$ are normal (by assumption), neither $T^1$ nor $T^2$ may contain zero rows (condition (4) is satisfied). Finally, we have that $ub(T^1, T^2)$ is the *least* upper bound of $(S^1, V^1)$ and $(S^2, V^2)$. Algorithm 2 is a straightforward realization of this strategy.

**Proposition 2.** *Let $(S^1, V^1)$ and $(S^2, V^2)$ be normal $\mathbb{Q}$-VASR abstractions of equal concrete dimension. Then the $\mathbb{Q}$-VASR abstraction $(S, V)$ computed by Algorithm 2 is normal and is a least upper bound of $(S^1, V^2)$ and $(S^2, V^2)$.*

## 4 Control Flow and $\mathbb{Q}$-VASRS

In this section, we give a method for improving the precision of our loop summarization technique by using $\mathbb{Q}$-VASRS; that is, $\mathbb{Q}$-VASR extended with control states. While $\mathbb{Q}$-VASRs over-approximate control flow using non-determinism, $\mathbb{Q}$-VASRSs allow us to analyze phenomena such as oscillating and multi-phase loops.

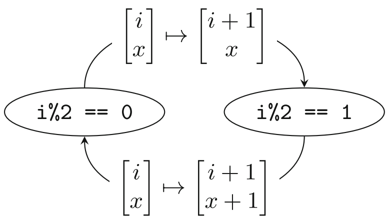
We begin with an example that demonstrates the precision gained by $\mathbb{Q}$-VASRS. The loop in Fig. 2a oscillates between (1) incrementing variable $i$ by 1 and (2) incrementing both variables $i$ and $x$ by 1. Suppose that we wish to prove

```
int x = 0; i = 1
while (*) do
  if i%2 == 0 then
    i := i + 1
  else
    i := i + 1
    x := x + 1
```

(a) Oscillating loop

$$\left\{ \begin{bmatrix} i \\ x \end{bmatrix} \mapsto \begin{bmatrix} i+1 \\ x+1 \end{bmatrix}, \begin{bmatrix} i \\ x \end{bmatrix} \mapsto \begin{bmatrix} i+1 \\ x \end{bmatrix} \right\}$$

(b) $\mathbb{Q}$-VASR abstraction.


i%2 == 0
i%2 == 1
$\begin{bmatrix} i \\ x \end{bmatrix} \mapsto \begin{bmatrix} i+1 \\ x \end{bmatrix}$
$\begin{bmatrix} i \\ x \end{bmatrix} \mapsto \begin{bmatrix} i+1 \\ x+1 \end{bmatrix}$

(c) $\mathbb{Q}$-VASRS abstraction.

**Fig. 2.** An oscillating loop and its representation as a $\mathbb{Q}$-VASR and $\mathbb{Q}$-VASRS.

that, starting with the configuration $x = 0 \land i = 1$, the loop maintains the invariant that $2x \leq i$. The (best) $\mathbb{Q}$-VASR abstraction of the loop, pictured in Fig. 2b, over-approximates the control flow of the loop by treating the conditional branch in the loop as a non-deterministic branch. This over-approximation may violate the invariant $2x \leq i$ by repeatedly executing the path where both variables are incremented. On the other hand, the $\mathbb{Q}$-VASRS abstraction of the loop pictured in Fig. 2c captures the understanding that the loop must oscillate between the two paths. The loop summary obtained from the reachability relation of this $\mathbb{Q}$-VASRS is powerful enough to prove the invariant $2x \leq i$ holds (under the precondition $x = 0 \land i = 1$).

## 4.1 Technical Details

In the following, we give a method for over-approximating the transitive closure of a transition formula $F(\mathbf{x}, \mathbf{x}')$ using a $\mathbb{Q}$-VASRS. We start by defining *predicate* $\mathbb{Q}$-VASRS, a variation of $\mathbb{Q}$-VASRS with control states that correspond to disjoint state predicates (where the states intuitively belong to the transition formula $F$ rather than the $\mathbb{Q}$-VASRS itself). We extend linear simulations and best abstractions to predicate $\mathbb{Q}$-VASRS, and give an algorithm for synthesizing best predicate $\mathbb{Q}$-VASRS abstractions (for a given set of predicates). Finally, we give an end-to-end algorithm for over-approximating the transitive closure of a transition formula.

**Definition 6.** *A* ***predicate*** $\mathbb{Q}$*-****VASRS*** *over* $\mathbf{x}$ *is a* $\mathbb{Q}$*-VASRS* $\mathcal{V} = (P, E)$*, such that each control state is a predicate over the variables* $\mathbf{x}$ *and the predicates in* $P$ *are pairwise inconsistent (for all* $p \neq q \in P$*,* $p \land q$ *is unsatisfiable).*

We extend linear simulations to predicate $\mathbb{Q}$-VASRS as follows:

- Let $F(\mathbf{x}, \mathbf{x}')$ be an $n$-dimensional transition formula and let $\mathcal{V} = (P, E)$ be an $m$-dimensional $\mathbb{Q}$-VASRS over $\mathbf{x}$. We say that a linear transformation $S : \mathbb{Q}^{m \times n}$ is a linear simulation from $F$ to $\mathcal{V}$ if for all $\mathbf{u}, \mathbf{v} \in \mathbb{Q}^n$ such that $\mathbf{u} \to_F \mathbf{v}$, (1) there is a (unique) $p \in P$ such that $p(\mathbf{u})$ is valid (2) there is a (unique) $q \in P$ such that $q(\mathbf{v})$ is valid, and (3) $(p, S\mathbf{u}) \to_{\mathcal{V}} (q, S\mathbf{v})$.
- Let $\mathcal{V}^1 = (P^1, E^1)$ and $\mathcal{V}^2 = (P^2, E^2)$ be predicate $\mathbb{Q}$-VASRSs over $\mathbf{x}$ (for some $\mathbf{x}$) of dimensions $d$ and $e$, respectively. We say that a linear transformation $S : \mathbb{Q}^{e \times d}$ is a linear simulation from $\mathcal{V}^1$ to $\mathcal{V}^2$ if for all $p^1, q^1 \in P^1$ and for all $\mathbf{u}, \mathbf{v} \in \mathbb{Q}^d$ such that $(p^1, \mathbf{u}) \to_{\mathcal{V}^1} (q^1, \mathbf{v})$, there exists (unique) $p^2, q^2 \in P^2$ such that (1) $(p^2, S\mathbf{u}) \to_{\mathcal{V}^2} (q^2, S\mathbf{v})$, (2) $p^1 \models p^2$, and (3) $q^1 \models q^2$.

We define a $\mathbb{Q}$-VASRS abstraction over $\mathbf{x} = x_1, ..., x_n$ to be a pair $(S, \mathcal{V})$ consisting of a rational matrix $S \in \mathbb{Q}^{d \times n}$ and a predicate $\mathbb{Q}$-VASRS of dimension $d$ over $\mathbf{x}$. We extend the simulation preorder $\preceq$ to $\mathbb{Q}$-VASRS abstractions in the natural way. Extending the definition of "best" abstractions requires more care, since we can always find a "better" $\mathbb{Q}$-VASRS abstraction (strictly smaller in $\preceq$ order) by using a finer set of predicates. However, if we consider only predicate

**Algorithm 3.** `abstract-VASRS`$(F, P)$

```
input : Transition formula $F(\mathbf{x}, \mathbf{x}')$, set of pairwise-disjoint predicates $P$ over
        $\mathbf{x}$ such that for all $\mathbf{u}, \mathbf{v}$ with $\mathbf{u} \to_F \mathbf{v}$, there exists $p, q \in P$ with $p(\mathbf{u})$
        and $q(\mathbf{v})$ both valid
output: Best $\mathbb{Q}$-VASRS abstraction of $F$ with control states $P$
1 For all $p, q \in P$, let $(S_{p,q}, V_{p,q}) \leftarrow$ abstract-VASR$(p(\mathbf{x}) \wedge F(\mathbf{x}, \mathbf{x}') \wedge q(\mathbf{x}'))$;
2 $(S, V) \leftarrow$ least upper bound of all $(S_{p,q}, V_{p,q})$;
3 For all $p, q \in P$, let $T_{p,q} \leftarrow$ the simulation matrix from $(S_{p,q}, V_{p,q})$ to $(S, V)$;
4 $E = \{(p, \mathbf{r}, \mathbf{a}, q) : p, q \in P, (\mathbf{r}, \mathbf{a}) \in image(V_{p,q}, T_{p,q})\}$;
5 return $(S, (P, E))$
```

$\mathbb{Q}$-VASRS that share the same set of control states, then best abstractions do exist and can be computed using Algorithm 3.

Algorithm 3 works as follows: first, for each pair of formulas $p, q \in P$, compute a best $\mathbb{Q}$-VASR abstraction of the formula $p(\mathbf{x}) \wedge F(\mathbf{x}, \mathbf{x}') \wedge q(\mathbf{x}')$ and call it $(S_{p,q}, V_{p,q})$. $(S_{p,q}, V_{p,q})$ over-approximates the transitions of $F$ that begin in a program state satisfying $p$ and end in a program state satisfying $q$. Second, we compute the least upper bound of all $\mathbb{Q}$-VASR abstractions $(S_{p,q}, V_{p,q})$ to get a $\mathbb{Q}$-VASR abstraction $(S, V)$ for $F$. As a side-effect of the least upper bound computation, we obtain a linear simulation $T_{p,q}$ from $(S_{p,q}, V_{p,q})$ to $(S, V)$ for each $p, q$. A best $\mathbb{Q}$-VASRS abstraction of $F(\mathbf{x}, \mathbf{x}')$ with control states $P$ has $S$ as its simulation matrix and has the image of $V_{p,q}$ under $T_{p,q}$ as the edges from $p$ to $q$.

**Proposition 3.** *Given an transition formula $F(\mathbf{x}, \mathbf{x}')$ and control states $P$ over $\mathbf{x}$, Algorithm 3 computes the best predicate $\mathbb{Q}$-VASRS abstraction of $F$ with control states $P$.*

We now describe `iter-VASRS` (Algorithm 4), which uses $\mathbb{Q}$-VASRS to over-approximate the transitive closure of transition formulas. Towards our goal of *predictable* program analysis, we desire the analysis to be *monotone* in the sense that if $F$ and $G$ are transition formulas such that $F$ entails $G$, then `iter-VASRS`$(F)$ entails `iter-VASRS`$(G)$. A sufficient condition to guarantee monotonicity of the overall analysis is to require that the set of control states that we compute for $F$ is at least as fine as the set of control states we compute for $G$. We can achieve this by making the set of control states $P$ of input transition formula $F(\mathbf{x}, \mathbf{x}')$ equal to the set of connected regions of the topological closure of $\exists \mathbf{x}'.F$ (lines 1–4). Note that this set of predicates may fail the contract of `abstract-VASRS`: there may exist a transition $\mathbf{u} \to_F \mathbf{v}$ such that $\mathbf{v} \not\models \bigvee P$ (this occurs when there is a state of $F$ with no outgoing transitions). As a result, $(S, \mathcal{V}) =$ `abstract-VASRS`$(F, P)$ does not necessarily approximate $F$; however, it *does* over-approximate $F \wedge \bigvee P(\mathbf{x}')$. An over-approximation of the transitive closure of $F$ can easily be obtained from $reach(\mathcal{V})(S\mathbf{x}, S\mathbf{x}')$ (the over-approximation of the transitive closure of $F \wedge \bigvee P(\mathbf{x}')$ obtained from the

$\mathbb{Q}$-VASRS abstraction $(S, \mathcal{V})$) by sequentially composing with the disjunction of $F$ and the identity relation (line 6).

**Algorithm 4.** `iter-VASRS`$(F)$

```
input  : Transition formula $F(\mathbf{x}, \mathbf{x}')$
output: Over-approximation of the transitive closure of $F$
1 $P \leftarrow$ topological closure of DNF of $\exists \mathbf{x}'.F$ (see [17]);
2 /* Compute connected regions                                  */
3 while $\exists p_1, p_2 \in P$ with $p_1 \wedge p_2$ satisfiable do
4     $P \leftarrow (P \setminus \{p_1, p_2\}) \cup \{p_1 \vee p_2\}$
5 $(S, \mathcal{V}) \leftarrow$ abstract-VASRS$(F, P)$;
6 return $reach(\mathcal{V})(S\mathbf{x}, S\mathbf{x}') \circ (\mathbf{x}' = \mathbf{x} \vee F)$
```

*Precision Improvement.* The `abstract-VASRS` algorithm uses predicates to infer the control structure of a $\mathbb{Q}$-VASRS, but after computing the $\mathbb{Q}$-VASRS abstraction, `iter-VASRS` makes no further use of the predicates (i.e., the predicates are irrelevant in the computation of $reach(\mathcal{V})$). Predicates can be used to improve `iter-VASRS` as follows: the reachability relation of a $\mathbb{Q}$-VASRS is expressed by a formula that uses auxiliary variables to represent the state at which the computation begins and ends [8]. These variables can be used to encode that the pre-state of the transitive closure must satisfy the predicate corresponding to the begin state and the post-state must satisfy the predicate corresponding to the end state. As an example, consider the Fig. 2 and suppose that we wish to prove the invariant $x \leq 2i$ under the pre-condition $i = 0 \wedge x = 0$. While this invariant holds, we cannot prove it because there is counter example if the computation begins at $i\%2 == 1$. By applying the above improvement, we can prove that the computation must begin at $i\%2 == 0$, and the invariant is verified.

## 5 Evaluation

The goals of our evaluation is the answer the following questions:

- Are $\mathbb{Q}$-VASR sufficiently expressive to be able to generate accurate loop summaries?
- Does the $\mathbb{Q}$-VASRS technique improve upon the precision of $\mathbb{Q}$-VASR?
- Are the $\mathbb{Q}$-VASR/$\mathbb{Q}$-VASRS loop summarization algorithms performant?

We implemented our loop summarization procedure and the compositional whole-program summarization technique described in Sect. 1.1. We ran on a suite of 165 benchmarks, drawn from the C4B [2] and HOLA [4] suites, as well as the safe, integer-only benchmarks in the loops category of SV-Comp 2019 [22]. We ran each benchmark with a time-out of 5 min, and recorded how many benchmarks were proved safe by our $\mathbb{Q}$-VASR-based technique and our $\mathbb{Q}$-VASRS-based technique. For context, we also compare with CRA [14] (a related loop summarization technique), as well as SeaHorn [7] and UltimateAutomizer [9] (state-of-the-art software model checkers). The results are shown in Fig. 3.

The number of assertions proved correct using $\mathbb{Q}$-VASR is comparable to both SeaHorn and UltimateAutomizer, demonstrating that $\mathbb{Q}$-VASR can indeed model interesting loop phenomena. $\mathbb{Q}$-VASRS-based summarization significantly improves precision, proving the correctness of 93% of assertions in the svcomp suite, and more than any other tool in total. Note that the most precise tool for each suite is not strictly better than each of the other tools; in particular, there is only a single program in the HOLA suite that neither $\mathbb{Q}$-VASRS nor CRA can prove safe.

CRA-based summarization is the most performant of all the compared techniques, followed by $\mathbb{Q}$-VASR and $\mathbb{Q}$-VASRS. SeaHorn and UltimateAutomizer employ abstraction-refinement loops, and so take significantly longer to run the test suite.

| | | $\mathbb{Q}$-VASR | | $\mathbb{Q}$-VASRS | | CRA | | SeaHorn | | UltAuto | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | #safe | time | #safe | time | #safe | time | #safe | time | #safe | time |
| C4B | 35 | 21 | 37.9 | **31** | 35.4 | 27 | **33.1** | 23 | 2434.4 | 25 | 3881.6 |
| HOLA | 46 | 32 | 57.2 | 39 | 73.0 | **40** | **56.0** | 35 | 2115.0 | 36 | 2995.9 |
| svcomp19-int | 84 | 68 | **86.9** | **78** | 184.5 | 76 | 91.9 | 62 | 3038.0 | 64 | 6923.5 |

**Fig. 3.** Experimental results.

## 6 Related Work

*Compositional Analysis.* Our analysis follows the same high-level structure as compositional recurrence analysis (CRA) [5,14]. Our analysis differs from CRA in the way that it summarizes loops: we compute loop summaries by over-approximating loops with vector addition systems and computing reachability relations, whereas CRA computes loop summaries by extracting recurrence relations and computing closed forms. The advantage of our approach is that is that we can use $\mathbb{Q}$-VASR to accurately model multi-path loops and can make theoretical guarantees about the precision of our analysis; the advantage of CRA is its ability to generate non-linear invariants.

*Vector Addition Systems.* Our invariant generation method draws upon Haase and Halfon's polytime procedure for computing the reachability relation of integer vector addition systems with states and resets [8]. Generalization from the integer case to the rational case is straightforward. Continuous Petri nets [3] are a related generalization of vector addition systems, where time is taken to be continuous ($\mathbb{Q}$-VASR, in contrast, have rational state spaces but discrete time). Reachability for continuous Petri nets is computable polytime [6] and definable in $\exists$LRA [1].

Sinn et al. present a technique for resource bound analysis that is based on modeling programs by lossy vector addition system with states [21]. Sinn et al. model programs using vector addition systems with states over the natural numbers, which enables them to use termination bounds for VASS to compute upper bounds on resource usage. In contrast, we use VASS with resets over the rationals, which (in contrast to VASS over $\mathbb{N}$) have a $\exists$LIRA-definable reachability relation, enabling us to summarize loops. Moreover, Sinn et al.'s method for extracting VASS models of programs is heuristic, whereas our method gives precision guarantees.

*Affine and Polynomial Programs.* The problem of *polynomial* invariant generation has been investigated for various program models that generalize $\mathbb{Q}$-VASR, including solvable polynomial loops [19], (extended) P-solvable loops [11,15], and affine programs [10]. Like ours, these techniques are *predictable* in the sense that they can make theoretical guarantees about invariant quality. The kinds invariants that can be produced using these techniques (conjunctions of polynomial equations) is incomparable with those generated by the method presented in this paper ($\exists$LIRA formulas).

*Symbolic Abstraction.* The main contribution of this paper is a technique for synthesizing the best abstraction of a transition formula expressible in the language of $\mathbb{Q}$-VASR (with or without states). This is closely related to the *symbolic abstraction* problem, which computes the best abstraction of a formula within an abstract domain. The problem of computing best abstractions has been undertaken for finite-height abstract domains [18], template constraint matrices (including intervals and octagons) [16], and polyhedra [5,24]. Our best abstraction result differs in that (1) it is for a disjunctive domain and (2) the notion of "best" is based on simulation rather than the typical order-theoretic framework.

## References

1. Blondin, M., Finkel, A., Haase, C., Haddad, S.: Approaching the coverability problem continuously. In: Chechik, M., Raskin, J.-F. (eds.) TACAS 2016. LNCS, vol. 9636, pp. 480–496. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49674-9_28
2. Carbonneaux, Q., Hoffmann, J., Shao, Z.: Compositional certified resource bounds. In: PLDI (2015)
3. David, R., Alla, H.: Continuous Petri nets. In: Proceedings of 8th European Workshop on Applications and Theory Petri Nets, pp. 275–294 (1987)
4. Dillig, I., Dillig, T., Li, B., McMillan, K.: Inductive invariant generation via abductive inference. In: OOPSLA (2013)
5. Farzan, A., Kincaid, Z.: Compositional recurrence analysis. In: FMCAD (2015)
6. Fraca, E., Haddad, S.: Complexity analysis of continuous Petri nets. Fundam. Inf. **137**(1), 1–28 (2015)
7. Gurfinkel, A., Kahsai, T., Komuravelli, A., Navas, J.A.: The SeaHorn verification framework. In: Kroening, D., Păsăreanu, C.S. (eds.) CAV 2015. LNCS, vol. 9206, pp. 343–361. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21690-4_20

8. Haase, C., Halfon, S.: Integer vector addition systems with states. In: Ouaknine, J., Potapov, I., Worrell, J. (eds.) RP 2014. LNCS, vol. 8762, pp. 112–124. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11439-2_9
9. Heizmann, M., et al.: Ultimate automizer and the search for perfect interpolants. In: Beyer, D., Huisman, M. (eds.) TACAS 2018. LNCS, vol. 10806, pp. 447–451. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-89963-3_30
10. Hrushovski, E., Ouaknine, J., Pouly, A., Worrell, J.: Polynomial invariants for affine programs. In: Logic in Computer Science, pp. 530–539 (2018)
11. Humenberger, A., Jaroschek, M., Kovács, L.: Invariant Generation for Multi-Path Loops with Polynomial Assignments. In: Verification, Model Checking, and Abstract Interpretation. LNCS, vol. 10747, pp. 226–246. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73721-8_11
12. Karp, R.M., Miller, R.E.: Parallel program schemata. J. Comput. Syst. Sci. **3**(2), 147–195 (1969)
13. Kincaid, Z., Breck, J., Forouhi Boroujeni, A., Reps, T.: Compositional recurrence analysis revisited. In: PLDI (2017)
14. Kincaid, Z., Cyphert, J., Breck, J., Reps, T.: Non-linear reasoning for invariant synthesis. PACMPL **2**(POPL), 1–33 (2018)
15. Kovács, L.: Reasoning algebraically about P-solvable loops. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 249–264. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78800-3_18
16. Li, Y., Albarghouthi, A., Kincaid, Z., Gurfinkel, A., Chechik, M.: Symbolic optimization with SMT solvers. In: POPL, pp. 607–618 (2014)
17. Monniaux, D.: A quantifier elimination algorithm for linear real arithmetic. In: Cervesato, I., Veith, H., Voronkov, A. (eds.) LPAR 2008. LNCS (LNAI), vol. 5330, pp. 243–257. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89439-1_18
18. Reps, T., Sagiv, M., Yorsh, G.: Symbolic implementation of the best transformer. In: Steffen, B., Levi, G. (eds.) VMCAI 2004. LNCS, vol. 2937, pp. 252–266. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24622-0_21
19. Rodríguez-Carbonell, E., Kapur, D.: Automatic generation of polynomial loop invariants: algebraic foundations. In: ISSAC, pp. 266–273 (2004)
20. Silverman, J., Kincaid, Z.: Loop summarization with rational vector addition systems (extended version). arXiv e-prints. arXiv:1905.06495, May 2019
21. Sinn, M., Zuleger, F., Veith, H.: A simple and scalable static analysis for bound analysis and amortized complexity analysis. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 745–761. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08867-9_50
22. 8th International Competition on Software Verification (SV-COMP 2019) (2019). https://sv-comp.sosy-lab.org/2019/
23. Tarjan, R.E.: A unified approach to path problems. J. ACM **28**(3), 577–593 (1981)
24. Thakur, A., Reps, T.: A method for symbolic computation of abstract operations. In: Madhusudan, P., Seshia, S.A. (eds.) CAV 2012. LNCS, vol. 7358, pp. 174–192. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31424-7_17