Termination Analysis of Integer Linear Loops*

Aaron R. Bradley, Zohar Manna, and Henny B. Sipma

Computer Science Department
Stanford University
Stanford, CA 94305-9045
{arbrad,zm,sipma}@theory.stanford.edu

Abstract. Usually, ranking function synthesis and invariant generation over a loop with integer variables involves abstracting the loop to have real variables. Integer division and modulo arithmetic must be soundly abstracted away so that the analysis over the abstracted loop is sound for the original loop. Consequently, the analysis loses precision. In contrast, we introduce a technique for handling loops over integer variables directly. The resulting analysis is more precise than previous analyses.

1 Introduction

Proving termination of program loops is necessary for ensuring the correct behavior of embedded systems and safety critical software. It is also required when proving general temporal properties of infinite state programs (*e.g.*, [9, 12, 14]). The traditional method for proving loop termination is by proving that some function of the program variables is well-founded within the loop. Such a function is called a ranking function.

Discovering ranking functions is thus one way of automating termination proofs. Colón and Sipma describe the synthesis of linear ranking functions over linear loops using polyhedra [3, 4]. In [13], Podelski and Rybalchenko specialize the technique to a restricted class of single-path imperative loops without initial conditions. Their method is complete for this class. The authors generalize these results in [1] to general linear loops: loops that contain multiple paths and that have a nontrivial initial condition. Ranking functions can be lexicographic and have supporting invariants simultaneously generated, yet the method is still complete. The primary feature linking these analyses is that loop variables are assumed to range over the reals, $\mathbb{R}$. On loops in which variables range over the integers, $\mathbb{Z}$, or the nonnegative integers, $\mathbb{Z}^*$, and which include integer division or modulo arithmetic, real-based analyses are weak. They must abstract away this arithmetic to maintain soundness. Yet loops may terminate precisely because of the behavior of integer arithmetic.

* This research was supported in part by NSF grants CCR-01-21403, CCR-02-20134, CCR-02-09237, CNS-0411363, and CCF-0430102, by ARO grant DAAD19-01-1-0723, and by NAVY/ONR contract N00014-03-1-0939. The first author was additionally supported by a Sang Samuel Wang Stanford Graduate Fellowship.

Our main contribution is a technique for synthesizing linear ranking functions with supporting linear invariants over *integer linear loops*, which allow integer division and modulo arithmetic. The technique treats the integer variables without abstraction, so that the resulting analysis is guaranteed to find a linear ranking function with a specified number of supporting linear (inductive) invariants, if one exists.

Linear inequality invariant generation is a related task. Abstract interpretation [7, 8] is the classical approach to invariant generation, while recently, the constraint-based approach [2] has been proposed. We show how to adapt the ranking function synthesis method to generate linear inequality invariants with a fixed number of conjuncts over integer linear loops. The analysis finds stronger invariants than a real-variable analysis, although at high computational cost.

Our technique is characteristically constraint-based: it solves directly for a set of expressions that satisfy the ranking function and invariant *verification conditions* — the constraints — generated by the loop. The focus of our paper, then, is the method for solving such constraint systems exactly when the loop has integer variables. While real analyses (*e.g.*, [2, 3, 13, 1, 6]) exploit the *dual* of a constraint system to solve for the ranking function or invariants, an integer analysis cannot be so formulated: an integer constraint system does not have a known dual. Our analysis is based on two observations. First, integer linear loops, which allow integer division and modulo arithmetic, may be converted to equivalent *Presburger loops*, in which all formulae are Presburger formulae. Second, one can in principle enumerate all linear functions of the program state with integer (or, equivalently, rational) coefficients. Determining if each such function is a ranking function is decidable, as the verification conditions can be encoded in Presburger arithmetic [15, 5]. A similar argument works for generating invariants. Thus, enumeration provides a complete, although prohibitively expensive, procedure.

Instead, we propose a *region-based* search. A *region* is a collection of intervals over which the *parameters* of the problem — the unknown coefficients of the linear function — may take their values. Thus, a region represents an infinite set of functions, one for each rational point that it contains. Regions are enumerated in such a way that examining the corners of regions is equivalent to naive enumeration. But a *feasibility check* can determine that no function in the region can be a ranking function (or an inductive invariant), pruning the region and all its functions in one step. For invariant generation, a *subsumption check* avoids regions that contain only assertions weaker than already discovered invariants. The feasibility and subsumption checks are motivated by ideas from the numerical constraint satisfaction community (see, *e.g.*, [11]). We demonstrate that they are powerful in practice, making an intractable problem tractable.

The rest of the paper is organized as follows. Section 2 introduces our loop abstraction and basic concepts. Section 3 describes our synthesis technique for termination analysis. Section 4 adapts the technique for invariant generation. Section 5 presents empirical evidence showing the effectiveness of the technique. Finally, Section 6 concludes.

$\mathtt{uint}\ x > 0$
while $2x > 1$ **do**
$\quad x := x - \frac{x}{2}$
done

(a)

$\mathtt{uint}\ x$
$\theta :\ x > 0$
$\tau_1 :\ 2x > 1 \wedge x' = x - \frac{x}{2}$

(b)

Fig. 1. Loop ZERO written in an **(a)** imperative form and **(b)** as an integer linear loop.

2 Preliminaries

This section introduces our loop abstraction and basic concepts.

Definition 1 (Integer Variable) A program variable x has type `int` if its domain is the integers $\mathbb{Z}$ or `uint` if its domain is the nonnegative integers $\mathbb{Z}^*$.

Definition 2 (Integer Linear Formula) An *integer linear term* is either c, cx, $c_1 \frac{E}{c_2}$ (division), or $c_1(E\%c_2)$ (modulus), for constants $c, c_1 \in \mathbb{Z}$, positive constant $c_2 \in \mathbb{Z}^+$, variable x of type `uint` or `int`, and *integer linear expression* E. An *integer linear expression* is the summation of *integer linear terms*.

An *integer linear atom* is the comparison $E_1 \bowtie E_2$ of two integer linear expressions, for $\bowtie \in \{<, \leq, =, \neq, \geq, >\}$. An *integer linear formula* is a Boolean combination of *integer linear atoms*.

Definition 3 (Integer Linear Loop) An *integer linear loop* $L : \langle \mathcal{V}_{\mathbb{Z}}, \mathcal{V}_{\mathbb{Z}^*}, \theta, \mathcal{T} \rangle$ consists of variables $\mathcal{V}_{\mathbb{Z}}$ of type `int`, variables $\mathcal{V}_{\mathbb{Z}^*}$ of type `uint`, *initial condition* θ, and set of *transitions* $\mathcal{T}$. We refer to $\mathcal{V}_{\mathbb{Z}} \cup \mathcal{V}_{\mathbb{Z}^*}$ as $\mathcal{V}$.

θ is an integer linear formula over $\mathcal{V}$ expressing what is true before entering the loop. Each transition $\tau \in \mathcal{T}$ is an integer linear formula over $\mathcal{V} \cup \mathcal{V}'$, where the primed versions of variables indicate their values in the next state.

Integer linear loops allow modeling nondeterminism, both in the update of variables (*e.g.*, via inequality constraints or no constraints on the next state value of a variable) and in the execution of transitions (*e.g.*, when guards are not disjoint).

Example 1. Figure 1 shows the loop ZERO as an imperative loop and as an integer linear loop. If $x = 1$, then $\frac{x}{2} = 0$ so that x does not decrease; thus, ZERO does not terminate.

If x is a variable ranging over $\mathbb{R}$, the loop terminates: the transition simplifies to $x' = \frac{x}{2}$ so that x eventually reaches $\frac{1}{2}$ or below. Thus, abstracting this loop to the reals results in an unsound termination analysis.

Our synthesis technique is based on deciding validity of Presburger arithmetic formulae; hence, we need to transform away division and modulo operators.

Definition 4 (Presburger Formula) A *Presburger formula* is an integer linear formula that does not involve division or modulo arithmetic. Given an integer

$$\mathcal{P}(\mathcal{A}[c_1\tfrac{E}{c_2}]) \;=\; \bigvee_{b\in[0..c_2-1]} (\exists a)\left(\begin{array}{l}[c_2a+b=E\wedge E\geq 0\wedge\mathcal{P}(\mathcal{A}[c_1a])]\\ \vee\,[c_2a-b=E\wedge E\leq 0\wedge\mathcal{P}(\mathcal{A}[c_1a])]\end{array}\right)$$

$$\mathcal{P}(\mathcal{A}[c_1(E\%c_2)]) \;=\; \bigvee_{b\in[0..c_2-1]} (\exists a)\left(\begin{array}{l}[c_2a+b=E\wedge E\geq 0\wedge\mathcal{P}(\mathcal{A}[c_1b])]\\ \vee\,[c_2a-b=E\wedge E\leq 0\wedge\mathcal{P}(\mathcal{A}[-c_1b])]\end{array}\right)$$

Fig. 2. $\mathcal{P}(\mathcal{A})$ is recursively applied to remove division and modulo operators from $\mathcal{A}$.

linear formula $\mathcal{A}$, $\mathcal{P}(\mathcal{A})$ is an equivalent Presburger formula. $\mathcal{P}(\mathcal{A})$ is defined recursively in Figure 2, where $\mathcal{P}(\mathcal{A})=\mathcal{A}$ if neither rule applies.

Definition 5 (Presburger Loop) A loop L is a *Presburger loop* if all of its formulae are Presburger formulae.

Example 2. Consider the formula defining τ_1 of ZERO.

$$\mathcal{P}(\tau_1) \;=\; \mathcal{P}(2x>1\wedge x'=x-\frac{x}{2}) \;=\; \begin{array}{l}(\exists a)[2a=x\wedge x'=x-a]\\ \vee\,(\exists a)[2a+1=x\wedge x'=x-a]\end{array}$$

The formula is simplified according to the `uint` type of x.

Definition 6 (Linear Inductive Invariant) A *linear inductive invariant* φ for loop $L:\langle\mathcal{V}_{\mathbb{Z}},\mathcal{V}_{\mathbb{Z}^*},\theta,\mathcal{T}\rangle$ is a finite conjunction of affine formulae $\bigwedge_j(c_{j,1}x_1+\cdots+c_{j,n}x_n+c_{j,n+1}\geq 0)$ over $\mathcal{V}=\{x_1,\ldots,x_n\}$ with integer coefficients $c_{j,i}\in\mathbb{Z}$ that satisfies the following verification conditions:

Initiation $(\forall\mathcal{V})[\theta\rightarrow\varphi]$
Consecution $(\forall\tau\in\mathcal{T})(\forall\mathcal{V},\mathcal{V}')[(\varphi\wedge\tau)\rightarrow\varphi']$

φ' is the formula in which each variable is primed.

Definition 7 (Linear Ranking Function) A *linear ranking function* δ with *supporting linear invariant* φ for loop $L:\langle\mathcal{V}_{\mathbb{Z}},\mathcal{V}_{\mathbb{Z}^*},\theta,\mathcal{T}\rangle$ is an affine expression $c_1x_1+\cdots+c_nx_n+c_{n+1}$ over $\mathcal{V}=\{x_1,\ldots,x_n\}$ with integer coefficients $c_i\in\mathbb{Z}$ that satisfies the following verification conditions:

Bounded $(\forall\tau\in\mathcal{T})(\forall\mathcal{V},\mathcal{V}')[(\varphi\wedge\tau)\rightarrow\delta\geq 0]$
Ranking $(\forall\tau\in\mathcal{T})(\forall\mathcal{V},\mathcal{V}')[(\varphi\wedge\tau)\rightarrow\delta'<\delta]$

Example 3. Consider loop ONE in Figure 3**(a)**. Figure 3**(b)** presents the loop as a Presburger loop. Writing the existentially quantified variables as actual loop variables reveals that a Presburger loop may be written without quantification, as in Figure 3**(c)**. The type of a new variable depends on whether the replaced expression ranges over $\mathbb{Z}$, $\mathbb{Z}^*$, or $-\mathbb{Z}^*$.

We show that the function $\delta(x,y)=x+y$ is a ranking function with supporting invariant $x\geq 1$. The verification conditions have been trivially simplified for

$$
\begin{array}{l}
\texttt{uint}\ x, y \\
\theta :\ x > 0 \wedge x\%2 = 0 \\
\tau_1 :\ x\%2 = 0 \wedge x' = x - \frac{x}{2} \wedge y' = y \\
\tau_2 :\ x\%3 = 0 \wedge x' = x - 2 \wedge y' = y \\
\tau_3 :\ y > x \wedge x' = x \wedge y' = y - x
\end{array}
$$

(a)

$$
\begin{array}{l}
\texttt{uint}\ x, y \\
\theta :\ (\exists a)[x > 0 \wedge 2a = x] \\
\tau_1 :\ (\exists a)[2a = x \wedge x' = x - a \wedge y' = y] \\
\tau_2 :\ (\exists a)[3a = x \wedge x' = x - 2 \wedge y' = y] \\
\tau_3 :\ y > x \wedge x' = x \wedge y' = y - x
\end{array}
$$

(b)

$$
\begin{array}{l}
\texttt{uint}\ x, y, a \\
\theta :\ x > 0 \wedge 2a = x \\
\tau_1 :\ 2a = x \wedge x' = x - a \wedge y' = y \\
\tau_2 :\ 3a = x \wedge x' = x - 2 \wedge y' = y \\
\tau_3 :\ y > x \wedge x' = x \wedge y' = y - x
\end{array}
$$

(c)

Fig. 3. **(a)** Loop ONE and **(b)**, **(c)** two ways of writing ONE as a Presburger loop.

ease of presentation. Note that $\mathcal{V} \cup \mathcal{V}'$ are universally quantified and that they range over $\mathbb{Z}^*$.

$$
\begin{array}{ll}
x > 0 \wedge 2a = x \ \rightarrow\ x \geq 1 & \}\ \text{Initiation} \\
\left.\begin{array}{l}
\tau_1 :\ x \geq 1 \wedge 2a = x \ \rightarrow\ x - a \geq 1 \\
\tau_2 :\ x \geq 1 \wedge 3a = x \ \rightarrow\ x - 2 \geq 1 \\
\tau_3 :\ x \geq 1 \ \rightarrow\ x \geq 1
\end{array}\right\} & \text{Consecution} \\
\left.\begin{array}{l}
\tau_1 :\ x \geq 1 \wedge 2a = x \ \rightarrow\ x + y \geq 0 \\
\tau_2 :\ x \geq 1 \wedge 3a = x \ \rightarrow\ x + y \geq 0 \\
\tau_3 :\ x \geq 1 \wedge y > x \ \rightarrow\ x + y \geq 0
\end{array}\right\} & \text{Bounded} \\
\left.\begin{array}{l}
\tau_1 :\ x \geq 1 \wedge 2a = x \ \rightarrow\ (x - a) + y < x + y \\
\tau_2 :\ x \geq 1 \wedge 3a = x \ \rightarrow\ (x - 2) + y < x + y \\
\tau_3 :\ x \geq 1 \wedge y > x \ \rightarrow\ x + (y - x) < x + y
\end{array}\right\} & \text{Ranking}
\end{array}
$$

The verification conditions are all valid, so ONE terminates on all input.

3 Termination

Existence of a ranking function for a loop L proves that L terminates. In principle, to find a linear ranking function with supporting linear invariant of some size, one can enumerate pairs of functions and invariants with integer coefficients, check whether each satisfies Definitions 6 and 7, and stop when one pair is found. This enumeration is sufficient to find any linear ranking function with supporting linear invariant in which all coefficients are rational. Of course, if no such function exists, then the enumeration does not terminate. It trivially allows synthesis over linear loops, as all formulae can be expressed in Presburger arithmetic. However, it is impractical.

We describe an alternate version of enumeration in which an infinite sets of pairs of functions and assertions, none of which is a ranking function with supporting invariant, can be pruned in a single step. The technique is based on searching for a valid instantiation of a *ranking function synthesis template*.

Definition 8 (Template Assertion) A *template expression* over $\mathcal{V} = \{x_1, \ldots, x_n\}$ is an expression $c_1x_1 + \cdots + c_nx_n + c_{n+1}$ with unknown coefficients c_i. Letting $\mathbf{x} = (x_1, \ldots, x_n, 1)^{\mathrm{T}}$ be a homogeneous vector, we write $\mathbf{c}^{\mathrm{T}}\mathbf{x} = (c_1, \ldots, c_n, c_{n+1})(x_1, \ldots, x_n, 1)^{\mathrm{T}}$, with unknown coefficients $\mathbf{c}$. A *template assertion* is a conjunctive assertion $\bigwedge_i \mathbf{c_i}^{\mathrm{T}}\mathbf{x} \geq 0$, or $\mathbf{Cx} \geq 0$, with unknown coefficient matrix $\mathbf{C}$.

Definition 9 (Ranking Function Synthesis Template) Given loop $L : \langle \mathcal{V}_{\mathbb{Z}}, \mathcal{V}_{\mathbb{Z}^*}, \theta, \mathcal{T} \rangle$ and the desired size of the supporting invariant *isz*, the *ranking function synthesis template* is

$$\varphi : \begin{array}{ll} \theta \rightarrow \mathbf{Ix} \geq 0 & \text{(initiation)} \\ \wedge \bigwedge_{\tau \in \mathcal{T}} [(\mathbf{Ix} \geq 0 \wedge \tau) \rightarrow \mathbf{Ix}' \geq 0] & \text{(consecution)} \\ \wedge \bigwedge_{\tau \in \mathcal{T}} [(\mathbf{Ix} \geq 0 \wedge \tau) \rightarrow \mathbf{r}^{\mathrm{T}}\mathbf{x} \geq 0] & \text{(bounded)} \\ \wedge \bigwedge_{\tau \in \mathcal{T}} [(\mathbf{Ix} \geq 0 \wedge \tau) \rightarrow \mathbf{r}^{\mathrm{T}}\mathbf{x}' < \mathbf{r}^{\mathrm{T}}\mathbf{x}] & \text{(ranking)} \end{array}$$

where $\mathbf{I}$ is an $(n+1) \times isz$ matrix representing the unknown coefficients of the supporting invariants, and $\mathbf{r}$ is an $(n+1)$-vector representing the unknown coefficients of the ranking function. The elements of $\mathbf{I}$ and $\mathbf{r}$ are the *parameters* of the synthesis template, collectively referred to as P. A synthesis template with parameters P is written $\varphi[P]$.

To prove the existence of a ranking function supported by invariants for a loop L, it suffices to prove the validity of $(\exists P)(\forall \mathbf{x}, \mathbf{x}')\varphi[P]$. This assertion is not a Presburger formula, as it involves multiplication of elements of P and $\mathbf{x}$. We propose a search for a solution $\widetilde{P}$ such that $(\forall \mathbf{x}, \mathbf{x}')\varphi[\widetilde{P}]$, which is a Presburger formula if L is a Presburger loop, is valid. Later, we show that it is sometimes useful to split the parameters P into two sets P_1 and P_2, where the parameters in P_2 are not involved in any multiplication with $\mathbf{x}$, and search for the solution $\widetilde{P_1}$ to P_1 such that $(\exists P_2)(\forall \mathbf{x}, \mathbf{x}')\varphi[\widetilde{P_1}, P_2]$ is valid.

Example 4. Consider loop ONE. $i_1x + i_2y + i_3 \geq 0$ is a 1-conjunct invariant template, and $r_1x + r_2y + r_3$ is a ranking function template. The conjunction of the verification conditions of Example 3 is the instantiation of the synthesis template for ONE in which $i_1 = 1,\ i_2 = 0,\ i_3 = -1,\ r_1 = 1,\ r_2 = 1,\ r_3 = 0$.

Definition 10 (Parameter Region and Corner) For parameter set P, a *parameter region* R is a hyper-rectangle of dimension $|P|$ assigning a closed interval from $\mathbb{R}$ to each parameter in P. A *corner* of a region is an extreme point: each parameter is assigned either the lower or upper bound of its interval. The *lower corner* is the extreme point in which each parameter is assigned its lower bound.

A parameter region R represents an infinite number of possible instantiations of a synthesis template $\varphi[P]$. Each rational point $r \in R$ corresponds to an integer

```
let TERMINATES L isz =
  let φ[P] = TEMPLATE L isz in
  let queue = {(1, [−1, 1]^|P|)} in
  while |queue| > 0 do
    let d, R = CHOOSE queue in
    if FEASIBLE φ R then begin
      if CORNER_SOLUTION φ R then raise Terminates;
      if d ≤ D · |P| then
        let l, r = BISECT R in
        ADD {(d + 1, l), (d + 1, r)} queue
    end
  done;
  raise Unknown
```

Fig. 4. The function TERMINATES returns **Terminates** if a ranking function with a supporting invariant is discovered.

instantiation of P, $\widetilde{P}$. Specifically, r corresponds to the integer point $\widetilde{P}$ such that the GCD of the coordinates of $\widetilde{P}$ is 1 and $\widetilde{P}$ is a scalar multiple of r. We write that $\widetilde{P} \in R$ if there is some rational point $r \in R$ such that r corresponds to $\widetilde{P}$.

Definition 11 (Feasible Region) Given synthesis template $\varphi[P]$, region R for P is *feasible* if it may contain a solution point.

Figure 4 presents the outline of the method. First, a synthesis template with supporting invariant template of size *isz* is constructed. P is the set of parameters. A search queue is initialized to contain a tuple $(1, [-1, 1]^{|P|})$ expressing that the search is at depth 1 and the *region* under consideration is $[-1, 1]^{|P|}$. While this queue is not empty, some pair is chosen. The selected region is checked for *feasibility*; if it is infeasible, it is pruned. If it is feasible, a *corner* of the region is checked by instantiating the template and checking validity. If it is not a solution and the maximum depth, given by parameter D, has not been reached, the region is bisected along some dimension, and each half is added to the queue. We now present the details.

We start with the *feasibility* check. While a region R represents an infinite number of possible instantiations of a synthesis template $\varphi[P]$, we want to determine the feasibility of R by checking the validity of only a finite number of Presburger formulae. We proceed by forming $2^{|\mathcal{V}_\mathbb{Z} \cup \mathcal{V}'_\mathbb{Z}|}$ *quadrant completions* of φ, $\widehat{\varphi}$, each of which forces each variable of $\mathcal{V}_\mathbb{Z} \cup \mathcal{V}'_\mathbb{Z}$ to range over either $\mathbb{Z}^*$ or $-\mathbb{Z}^*$. For each completion $\widehat{\varphi}$, we form a *relaxation* of $\widehat{\varphi}$ over R, $\widehat{\varphi}_R$. If any one of these relaxed instantiations is invalid, then R does not contain a solution.

Definition 12 (Quadrant Completion) Given loop $L : \langle \mathcal{V}_\mathbb{Z}, \mathcal{V}_{\mathbb{Z}^*}, \theta, \mathcal{T} \rangle$ and synthesis template φ, a *quadrant completion* $\widehat{\varphi}$ of φ has the form

$$\left(\bigwedge_{x \in \mathcal{V}_\mathbb{Z}} (x \bowtie_x 0 \ \wedge \ x' \bowtie_{x'} 0) \right) \ \rightarrow \ \varphi,$$

where each $\bowtie \in \{\leq, \geq\}$; *i.e.*, each $x \in \mathcal{V}_{\mathbb{Z}}$ and $x' \in \mathcal{V}'_{\mathbb{Z}}$ is forced to be either nonnegative or nonpositive.

Definition 13 (Relaxation) Consider loop $L : \langle \mathcal{V}_{\mathbb{Z}}, \mathcal{V}_{\mathbb{Z}^*}, \theta, \mathcal{T} \rangle$, quadrant completion $\widehat{\varphi}[P]$ of $\varphi[P]$ expressed in negation normal form and with all minus signs eliminated by rearrangement, and region $R : [\mathbf{l}, \mathbf{u}]$ for P. The *relaxation* of $\widehat{\varphi}$ over R is the assertion $\widehat{\varphi}_R$ in which each inequality $\alpha \geq \beta$ of $\widehat{\varphi}[P]$ is replaced by $\overline{\alpha} \geq \underline{\beta}$. $\overline{\alpha}$ is obtained by replacing each term of the form p, where $p \in P$ and $\ell \leq p \leq u$ in R, with u and each term of the form px with ux if $\widehat{\varphi}[P]$ requires $x \geq 0$ and ℓx otherwise. Similarly $\underline{\beta}$ is obtained by replacing each term of the form p with ℓ and each term of the form px with ℓx if $\widehat{\varphi}[P]$ requires $x \geq 0$ and ux otherwise.

Lemma 1 (Feasible Region). *Consider synthesis template $\varphi[P]$ and region R for P. If for some quadrant completion $\widehat{\varphi}$ of φ, $(\forall \mathbf{x}, \mathbf{x}')\widehat{\varphi}_R$ is invalid, then R is infeasible.*

Definition 14 (Corner Solution) Given synthesis template $\varphi[P]$ and the corner $\widetilde{P} \in R$ for P, $\widetilde{P}$ is a *corner solution* if $\varphi[\widetilde{P}]$ is valid.

Theorem 1 (Sound and Complete). *Consider loop $L : \langle \mathcal{V}_{\mathbb{Z}}, \mathcal{V}_{\mathbb{Z}^*}, \theta, \mathcal{T} \rangle$, supporting invariant size isz, and maximum search depth parameter D. Suppose that* BISECT *always bisects one of the widest dimensions of a region and that* CORNER_SOLUTION *always chooses the lower corner of a region. Then L has a linear ranking function with supporting isz-conjunct linear invariant, expressed so that all coefficients are integers in $[-2^{D-1},\ 2^{D-1})$, if and only if* (TERMINATES L *isz*) *returns* **Terminates**.

Soundness is immediate: TERMINATES reports success only if it finds a solution, which by construction of the synthesis template obeys the verification conditions of Definitions 6 and 7.

Completeness (relative to the maximum search depth) is also fairly straightforward. First, every rational invariant and ranking function may be represented such that all coefficients lie in $[-1, 1]$ and the denominators of coefficients are powers of 2. Second, if BISECT and CORNER_SOLUTION satisfy the stated restrictions, then every combination of such coefficients, with coefficient denominators up to 2^{D-1} and numerators in $[-2^{D-1}, 2^{D-1})$, appears as a corner. Converting these rational coefficients to integers results in integer coefficients in $[-2^{D-1},\ 2^{D-1})$. Finally, Lemma 1 ensures that no region containing a solution is pruned.

It is easy to see that alternately running TERMINATES and incrementing the maximum depth results in a complete procedure: it is guaranteed to find a linear ranking function with its specified number of supporting invariants, in which coefficients are rational, if one exists. However, this procedure is not guaranteed to terminate when a solution does not exist, as the following example demonstrates.

Example 5. Consider applying the alternating procedure to ZERO. Consider checking the feasibility of a region R in which the coefficient of x and the

$$
\begin{array}{l}
\texttt{uint } x, y \\
\theta : \; x > 0 \wedge x\%2 = 0 \wedge y < 100 \\
\tau_1 : \; x\%2 = 0 \wedge x' = x - \frac{x}{2} \wedge y' = y \\
\tau_2 : \; x\%2 = 0 \wedge y < 100 - \frac{x}{2} \wedge x' = x \wedge y' = y + \frac{x}{2}
\end{array}
$$

Fig. 5. Loop with large constants.

constant offset are both positive intervals; that is, $R = [\ell_1, u_1] \times [\ell_2, u_2]$ for $\ell_1, u_1, \ell_2, u_2 > 0$. Clearly, any instance of $r_1 x + r_2$ in R is bounded from below, as $x \geq 0$. This region is also feasible: at worst, $x = 1$ so that $x' = x$, yet then $\ell_1 x' + \ell_2 = \ell_1 x + \ell_2 < u_1 x + u_2$, so that the relaxation $\widehat{\varphi}_R$ is valid. An infinite number of such regions is encountered during an unbounded search.

We conclude this section on termination by noting that one useful variation is possible. Constant parameters (*e.g.*, r_2 of $r_1 x + r_2$) that appear in ranking function synthesis templates may be existentially quantified. Consider loop $L : \langle \mathcal{V}_{\mathbb{Z}}, \mathcal{V}_{\mathbb{Z}^*}, \theta, \mathcal{T} \rangle$. Letting $\mathbf{x} = (x_1, \ldots, x_n)^{\mathrm{T}}$ for $x_i \in \mathcal{V}$, the ranking function synthesis template is then

$$
(\exists \mathbf{i}, r)(\forall \mathbf{x}, \mathbf{x}')
\begin{bmatrix}
\quad \theta \rightarrow \mathbf{I}\mathbf{x} + \mathbf{i} \geq 0 \\
\wedge \bigwedge_{\tau \in \mathcal{T}} [(\mathbf{I}\mathbf{x} + \mathbf{i} \geq 0 \wedge \tau) \rightarrow \mathbf{I}\mathbf{x}' + \mathbf{i} \geq 0] \\
\wedge \bigwedge_{\tau \in \mathcal{T}} [(\mathbf{I}\mathbf{x} + \mathbf{i} \geq 0 \wedge \tau) \rightarrow \mathbf{r}^{\mathrm{T}}\mathbf{x} + r \geq 0] \\
\wedge \bigwedge_{\tau \in \mathcal{T}} [(\mathbf{I}\mathbf{x} + \mathbf{i} \geq 0 \wedge \tau) \rightarrow \mathbf{r}^{\mathrm{T}}\mathbf{x}' < \mathbf{r}^{\mathrm{T}}\mathbf{x}]
\end{bmatrix}
$$

When $\mathbf{I}$ and $\mathbf{r}$ are instantiated, the formula is decidable, although the quantifier alternation increases the difficulty of the instance.

Example 6. Consider the loop in Figure 5. $x - y + 99$ is a ranking function supported by the invariants $x \geq 1$ and $y \leq 99$. Without quantifying the constant coefficients, the search must continue to depth 8; however, with quantification, the search finds a solution at depth 2.

4 Invariant Generation

In this section, we adapt the technique to invariant generation. The primary challenge in invariant generation, relative to ranking function synthesis, is to avoid discovering invariants weaker than those already known. We adapt the region-based technique to invariant generation by introducing a region-subsumption check, which may prune a region in which all assertions are subsumed by those already found.

Definition 15 (Invariant Synthesis Template) Given a loop $L : \langle \mathcal{V}_{\mathbb{Z}}, \mathcal{V}_{\mathbb{Z}^*}, \theta, \mathcal{T} \rangle$ and the desired size of the conjunctive invariant *isz*, the *invariant synthesis template* is

$$\varphi: \quad \begin{array}{ll} & \theta \rightarrow \mathbf{Ix} \geq 0 \\ \wedge & \bigwedge_{\tau \in \mathcal{T}} [(\mathbf{Ix} \geq 0 \wedge \tau) \rightarrow \mathbf{Ix}' \geq 0] \end{array} \qquad \begin{array}{r} \text{(initiation)} \\ \text{(consecution)} \end{array}$$

where $\mathbf{I}$ is an $(n+1) \times isz$ matrix representing the unknown coefficients of the invariant. The elements of $\mathbf{I}$ are the *parameters*, which we refer to collectively as P, of the template.

As with the termination analysis, solutions are integer instantiations $\widetilde{P}$ of P such that $(\forall \mathbf{x}, \mathbf{x}')\varphi[\widetilde{P}]$ is valid. However, generating invariants requires reporting all discovered solutions, rather than halting after finding a solution. Invariants are common, yet many are weak. The termination analysis prunes many regions containing invariants because they do not support a ranking function. No such criterion exists here. Instead, the invariant generation method may prune a region if all of its assertions are *subsumed by* already discovered invariants.

Definition 16 (Subsumed Region) Consider invariant template $\mathbf{Ix} \geq 0$, region R for $\mathbf{I}$, and the set of known invariants K. R is a *subsumed region* if every instantiation of $\mathbf{Ix} \geq 0$ from R is subsumed by $\bigwedge K$.

Definition 17 (Subsumption Template) Consider known invariants K and invariant template $\mathbf{Ix} \geq 0$. Then

$$\left(\bigwedge K\right) \rightarrow \mathbf{Ix} \geq 0$$

is the *subsumption template* for $\mathbf{Ix} \geq 0$.

Definition 18 (Strengthening) Consider loop $L : \langle \mathcal{V}_{\mathbb{Z}}, \mathcal{V}_{\mathbb{Z}^*}, \theta, \mathcal{T} \rangle$, quadrant completion $\widehat{\psi}[P]$ of subsumption template $\psi[P]$, and region $R : [\mathbf{l}, \mathbf{u}]$ for P. The *strengthening* of $\widehat{\psi}$ is the assertion $\widehat{\psi}^R$ in which each inequality $\alpha \geq 0$ of $\widehat{\psi}[P]$ containing parameterized terms is replaced by $\underline{\alpha} \geq 0$, and other literals are unchanged. $\underline{\alpha}$ is obtained by replacing each term of the form p, where $p \in P$ and $\ell \leq p \leq u$ in R, with ℓ and each term of the form px with ℓx if $\widehat{\psi}[P]$ requires $x \geq 0$ and ux otherwise.

Lemma 2 (Subsumed Region). *Consider known invariants K, subsumption template $\psi[P]$, and region R for P. If for all quadrant completions $\widehat{\psi}$ of ψ, $(\forall \mathbf{x})\widehat{\psi}^R$ is valid, then R is a subsumed region.*

Figure 6 shows the function INVARIANTS. It is similar in structure to TERMINATES, except that it returns the discovered invariants after it has searched the space defined by the maximum depth parameter D. The function CORNER_SOLUTION returns None if the corner of R is not a solution, and Some I otherwise. Discovered invariants are recorded in K. SUBSUMED implements the subsumption check.

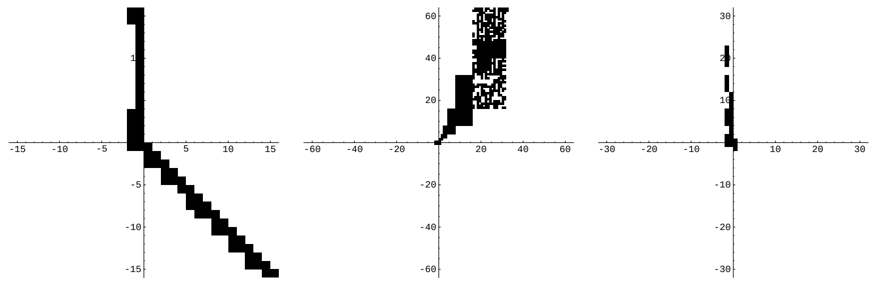
```
let INVARIANTS L isz =
  let φ[P] = TEMPLATE L isz in
  let K = {} in
  let queue = {(1, [−1, 1]^|P|)} in
  while |queue| > 0 do
    let d, R = CHOOSE queue in
    if (FEASIBLE φ R) ∧ ¬(SUBSUMED φ R K) then begin
      begin
        match CORNER_SOLUTION φ R with
        | Some I  →  ADD I K
        | None    →  ()
      end;
      if d ≤ D · |P| then
        let l, r = BISECT R in
        ADD {(d + 1, l), (d + 1, r)} queue
    end
  done;
  K
```

Fig. 6. The function INVARIANTS returns a set of invariants of L.

Theorem 2 (Sound and Complete). *Consider loop $L : \langle \mathcal{V}_{\mathbb{Z}}, \mathcal{V}_{\mathbb{Z}^*}, \theta, \mathcal{T} \rangle$, invariant size isz, and maximum search depth parameter D. Suppose that* BISECT *always bisects one of the widest dimensions of a region and that* CORNER_SOLUTION *always chooses the lower corner of a region. Then L has a linear isz-conjunct invariant I, expressed so that all coefficients are integers in $[-2^{D-1},\ 2^{D-1})$, if and only if the conjunction of the set returned by* (INVARIANTS L *isz*) *implies I.*

Proving soundness and completeness is again straightforward, given Lemma 2. As with TERMINATES, the procedure that alternately runs INVARIANTS and increases its maximum depth will find any linear inductive invariant with rational coefficients, unless a stronger one has been found. This procedure is not guaranteed to terminate.

Example 7. Consider ZERO again and any region $R = [\ell_1, u_1] \times [\ell_2, u_2]$ in which $\ell_1, u_1 \geq 0$, $\ell_2 = -u_1$, and $u_2 = -\ell_1$. The corner instantiation $\ell_1 x + u_2 \geq 0$ (*e.g.*, $4x - 4 \geq 0$) is an invariant of ZERO. It is also the strongest invariant in the region, yet it trivially subsumes $x \geq 1$, as they are equivalent. But in the strengthening $\widehat{\psi}^R$ in which $x \geq 1$ is known, $\ell_1 x + \ell_2 = \ell_1 x - u_1 = \ell_1 x - (\ell_1 + \epsilon)$, for some $\epsilon > 0$. At $x = 1$, $\ell_1 x - (\ell_1 + \epsilon) = \ell_1(1) - \ell_1 - \epsilon = -\epsilon \not\geq 0$, so that $\widehat{\psi}^R$ is invalid and R is not pruned as subsumed. An infinite number of such regions is encountered during an unbounded search.



(a) (b) (c)

Fig. 7. Feasible regions explored during search.

$\texttt{int}\ i, j, k$
$\theta :\ \top$
$\tau_1 :\ i + 17j + 33k > 0 \wedge i' = i + 16 \wedge j' = j - 1 \wedge k' = k$
$\tau_2 :\ i + 17j + 33k > 0 \wedge i' = i + 32 \wedge j' = j \wedge k' = k - 1$

Fig. 8. Loop with large coefficients.

5 Empirical Observations

In this section, we examine the behavior of TERMINATES and INVARIANTS on a set of loops. We prototyped TERMINATES and INVARIANTS in O'Caml, using the OMEGA TEST [16] to decide validity of Presburger formulae.

When INVARIANTS is applied to loop ZERO with a maximum search depth of 5, it can potentially search over two thousand regions. It actually searches approximately 8% of the regions. Figure 7**(a)** shows the feasible regions encountered during the search. To represent feasible regions pictorially, regions are normalized to have integer bounds. Regions farther from the origin occur later in the search. INVARIANTS focuses on the invariant *true*, represented by the vertical line of regions, and the invariant $x \geq 1$, indicated by the diagonal group.

The loop in Figure 8 has the obvious ranking function $i + 17j + 33k$. We analyze it to force a deep search: finding it requires searching to a depth of 7, resulting in over four million possible regions. TERMINATES finds the solution after searching under three thousand regions. Figure 7**(b)** shows the feasible regions encountered during the search. As the search focuses on a coefficient of 1 for i, the graph shows the projection for the coefficients of j and k.

Figure 9**(a)** presents a loop that does not have a linear ranking function. When the transitions are strengthened with the invariants discovered during a depth-two search (*e.g.*, $s \geq 1$), running TERMINATES to a depth of 6 reveals that the loop does not have a linear ranking function supported by that set of invariants. That is, no region is skipped because of depth, proving the nonexistence of a linear ranking function (relative to the discovered supporting invariants). In this search, about 1% of possible regions are examined. Figure 7**(c)** shows the examined feasible regions for the coefficients of x and s.

uint x, s
$\theta: \; s = 1$
$\tau_1: \; x > 100 \wedge s \neq 1 \wedge x' = x - 10 \wedge s' = s - 1$
$\tau_2: \; x \leq 100 \wedge x' = x + 11 \wedge s' = s + 1$

(a)

uint x, y
$\theta: \; x = 0 \wedge y > 10$
$\tau_1: \; x' = x + 1 \wedge y' = y + 7$
$\tau_2: \; x' = \frac{y}{7} \wedge y' = y$

(b)

Fig. 9. (a) Loop without linear ranking function. **(b)** Loop SEVEN.

Table 1. Summary of experiments. (T) indicates termination analysis; (I) indicates invariant generation. **Time** is in seconds. The **Regions** column indicates the percentage of the search space explored; for searches terminating at depth 2, this percentage is too variable to state.

Loop	**Time**	**Depth**	**Regions**	**Loop**	**Time**	**Depth**	**Regions**
ONE (T)	20s	2	—	Fig. 5 (T)	5s	2	—
ZERO (I)	1s	5	8%	Fig. 8 (T)	100s	7	.07%
Fig. 9**(a)** (T)	1s	6	1%	Fig. 9**(b)** (I)	45s	5	5%

Running INVARIANTS to a depth of 5 on the loop in Figure 9**(b)** reveals the two invariants $y \geq 11$ and $7x \leq y$. The search examines approximately 5% of the possible regions and produces about 15 invariants. Disabling the subsumption check results in examining about a third of the possible regions and producing over ten thousand invariants.

Table 1 summarizes these results.

Finally, we describe the application of our method to a simple hardware multiplication algorithm [10]. Figure 10 presents the algorithm for multiplying two n-bit nonnegative integers, m_1 and m_2, into $2n$-bit product register p. Actually, m_2 is a $2n$-bit register, but its left half is initially 0. As we would like to verify facts about multiplication, which we cannot model explicitly, we introduce tracking variables M and M_0. The variable M tracks the changes to $m_1 m_2$ as

```
uint i, n, m1, m2, p
θ : p = 0
for i = 1 to n do
   if m1%2 = 1 then
      p := p + m2
   m1 := m1/2
   m2 := 2 * m2
done
```

(a)

uint i, n, m_1, m_2, p
[uint M, M_0]
$\theta: \; p = 0 \wedge i = 1 \wedge [M = M_0]$

$$\tau_1: \left(\begin{array}{l} i \leq n \wedge m_1\%2 = 0 \wedge i' = i + 1 \wedge n' = n \\ \wedge \; m_1' = m_1/2 \wedge m_2' = 2m_2 \wedge p' = p \\ [\wedge \; M' = M \wedge M_0' = M_0] \end{array} \right)$$

$$\tau_2: \left(\begin{array}{l} i \leq n \wedge m_1\%2 = 1 \wedge i' = i + 1 \wedge n' = n \\ \wedge \; m_1' = m_1/2 \wedge m_2' = 2m_2 \wedge p' = p + m_2 \\ [\wedge \; M' = M - m_2 \wedge M_0' = M_0] \end{array} \right)$$

(b)

Fig. 10. (a) Multiplication of two n-bit binary numbers m_1 and m_2 into $2n$-bit product p, where multiplication and division by 2 and modulo arithmetic model bit manipulation. **(b)** Integer linear loop form, in which M and M_0 have been introduced to track $m_1 m_2$. Augmenting variables and assertions are in brackets.

the algorithm progresses, while M_0 records M's initial value. After transforming Figure 10**(a)** to an integer linear loop, we calculate

$$M' = m_1' m_2' = \tfrac{m_1}{2}(2m_2)$$
$$= \begin{cases} m_1 m_2 = M & \text{if } m_1\%2 = 0 \\ \tfrac{m_1-1}{2}(2m_2) = m_1 m_2 - m_2 = M - m_2 & \text{if } m_1\%2 = 1 \end{cases}$$

and augment the loop as in Figure 10**(b)**. Termination is trivial. Invariant generation of 1-conjunct invariants produces the two invariants $p + M \leq M_0$ and $p + M \geq M_0$, so that $p = M_0 - M$, proving correctness.

6 Conclusion

We presented a technique for synthesizing linear ranking functions with supporting linear invariants and generating linear invariants of loops with integer variables. Unlike previous work, the technique handles integer variables directly, yet remains complete. Thus, it is more precise than previous analyses.

Several parameters of TERMINATES and INVARIANTS are left open for heuristics, including the criterion for choosing the dimension to bisect and the method of choosing regions from the queue. Even with heuristics, analyses over integer variables cannot scale to the size of problems that real-variable analyses can handle, while many integer loops can be soundly and effectively analyzed by real-variable techniques. Thus, future work includes developing an analysis system that identifies when the stronger integer-based analyses are required. Handling mixed-variable loops, in which only some variables are integers, would also be of value.

Acknowledgments We thank Sriram Sankaranarayanan and the reviewers for their insightful comments.

References

1. Bradley, A. R., Manna, Z., and Sipma, H. B. Linear ranking with reachability. In *CAV* (2005). To appear.
2. Colón, M., Sankaranarayanan, S., and Sipma, H. Linear invariant generation using non-linear constraint solving. In *CAV* (2003), pp. 420–433.
3. Colón, M., and Sipma, H. Synthesis of linear ranking functions. In *TACAS* (2001), pp. 67–81.
4. Colón, M., and Sipma, H. Practical methods for proving program termination. In *CAV* (2002), pp. 442–454.
5. Cooper, D. C. Theorem proving in arithmetic without multiplication. *Machine Intelligence 7* (1972), 91–100.
6. Cousot, P. Proving program invariance and termination by parametric abstraction, lagrangian relaxation and semidefinite programming. In *VMCAI* (2005), pp. 1–24.
7. Cousot, P., and Cousot, R. Abstract Interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *ACM Principles of Programming Languages* (1977), pp. 238–252.

8. Cousot, P., and Halbwachs, N. Automatic discovery of linear restraints among the variables of a program. In *ACM Principles of Programming Languages* (Jan. 1978), pp. 84–97.
9. H. B. Sipma, T. E. Uribe, and Z. Manna. Deductive model checking. In *CAV* (1996), pp. 209–219.
10. Hennessy, J. L., and Patterson, D. A. *Computer organization and design (2nd ed.): the hardware/software interface*. Morgan Kaufmann Publishers Inc., 1998.
11. Hentenryck, P. V., Michel, L., and Benhamou, F. Newton - constraint programming over nonlinear constraints. *Sci. Comput. Program.* (1998), 83–118.
12. Manna, Z., Browne, A., Sipma, H., and Uribe, T. E. Visual abstractions for temporal verification. In *Algebraic Methodology and Software Technology* (1998), pp. 28–41.
13. Podelski, A., and Rybalchenko, A. A complete method for the synthesis of linear ranking functions. In *VMCAI* (2004), pp. 239–251.
14. Podelski, A., and Rybalchenko, A. Transition invariants. In *LICS* (2004), pp. 32–41.
15. Presburger, M. Ueber die vollstaendigkeit eines gewissen systems der arithmetik ganzer zahlen, in welchem die addition als einzige operation hervortritt. *Comptes Rendus du I congrs de Mathmaticiens des Pays Slaves* (1929), 92–101.
16. Pugh, W. The Omega test: a fast and practical integer programming algorithm for dependence analysis. *Communications of the ACM 35* (1992), 102–114.