Synthesis of Linear Ranking Functions

Michael A. Colón and Henny B. Sipma *

Computer Science Department
Stanford University
Stanford, CA. 94305-9045
{colon,sipma}@cs.stanford.edu

Abstract. Deductive verification of progress properties relies on finding ranking functions to prove termination of program cycles. We present an algorithm to synthesize linear ranking functions that can establish such termination. Fundamental to our approach is the representation of systems of linear inequalities and sets of linear expressions as polyhedral cones. This representation allows us to reduce the search for linear ranking functions to the computation of polars, intersections and projections of polyhedral cones, problems which have well-known solutions.

1 Introduction

Deductive verification of reactive systems relies on finding invariants and ranking functions. While automatic generation of invariants has received much attention [GW75,KM76,BLS96,BBM97], automatic generation of ranking functions has only recently started to emerge [DGG00].

Proofs of progress properties of systems (that is, properties that the system will achieve a certain goal) involve showing that cycles on the path to the goal terminate. The classical method for establishing such termination is the use of well-founded domains together with so-called ranking functions that assign a value from these domains to each program state. Progress is then shown by demonstrating that each step in the cycle reduces the measure assigned by the ranking function. As there can be no infinite descending chain of elements of a well-founded domain, the cycle must eventually terminate. Clearly the existence of such a ranking function implies termination. Conversely, it has been proven that if a cycle terminates, a ranking function exists.

Recent years have seen great progress in automating deductive verification by improvements in decision procedures, invariant generation and automatic abstraction. However, the synthesis of ranking functions remains largely a manual task. Some heuristics have been proposed in [DGG00], but these are limited to functions that appear as expressions in the program text.

* This research was supported in part by the National Science Foundation grant CCR-99-00984-001, by ARO grant DAAG55-98-1-0471, by ARO/MURI grant DAAH04-96-1-0341, by ARPA/Army contract DABT63-96-C-0096, and by ARPA/AirForce contracts F33615-00-C-1693 and F33615-99-C-3014.

T. Margaria and W. Yi (Eds.): TACAS 2001, LNCS 2031, pp. 67–81, 2001.

In this paper we propose an algorithm that generates ranking functions for a program cycle that are linear in the program variables. The algorithm consists of three steps. First, it derives a set of linear expressions that are bounded inside the cycle from some cycle invariant. Second, it derives a set of linear expressions that decrease discretely around the cycle from the cycle's transition relation. The third step then computes the intersection of these two sets. Any expression in the intersection serves as a ranking function, and thus nonemptiness of the intersection proves termination of the cycle.

The remainder of the paper is organized as follows. Section 2 presents our computational model of transition systems, it gives some background on well-founded domains, and it introduces the running example. In Sec. 3 we introduce polyhedral cones and demonstrate that problems involving systems of linear inequalities can be reduced to problems over cones. Our algorithm is presented in Sec. 4, and its application is illustrated on the example program. In Sec. 5 we discuss a more complex application of the algorithm, and in Sec. 6 we conclude with some discussion and limitations of our approach.

2 Preliminaries

The computational model used to describe programs is that of a *transition system* [MP95] (FTS), $\mathcal{S} = \langle V, \Theta, \mathcal{T} \rangle$, where V is a finite set of variables, Θ is an initial condition, and $\mathcal{T}$ is a finite set of transitions. A *state* s is an interpretation of V. Each transition $\tau \in \mathcal{T}$ is represented by a *transition relation* ρ_τ, an assertion that expresses the relation between the values of V in some state s and the values of V (referred to by V') in any state s' to which the system can transition by taking τ. A *run* of $\mathcal{S}$ is a sequence of states such that the first state satisfies Θ and any two consecutive states satisfy ρ_τ for some $\tau \in \mathcal{T}$. A state s is *accessible* if s appears in some run of $\mathcal{S}$. The set of all accessible states is Σ.

A *relational domain* (or just *domain*) $\langle \mathcal{D}, \succ \rangle$ is a set $\mathcal{D}$ paired with a binary relation $\succ$ on $\mathcal{D}$. A domain is said to be *well-founded* if there are no infinite sequences of elements of $\mathcal{D}$ which decrease under $\succ$. A function f is said to *map* $\langle \mathcal{D}_1, \succ_1 \rangle$ *into* $\langle \mathcal{D}_2, \succ_2 \rangle$ if f maps $\mathcal{D}_1$ into $\mathcal{D}_2$ and is monotone, that is, $f(d_1) \succ_2 f(d_2)$ for all $d_1, d_2 \in \mathcal{D}_1$ such that $d_1 \succ_1 d_2$. Notice that f maps infinite decreasing sequences in $\langle \mathcal{D}_1, \succ_1 \rangle$ to infinite decreasing sequences in $\langle \mathcal{D}_2, \succ_2 \rangle$. A *ranking function* for a domain is any function that maps it into some domain that is known to be well-founded, such as the non-negative integers with the greater-than relation. Notice that any domain for which a ranking function exists is well-founded.

Ranking functions can also be used to establish the termination of transition systems. Let $\mathcal{R} = \bigcup \{\rho_\tau | \tau \in \mathcal{T}\}$ be the combined transition relation of $\mathcal{S}$. The decreasing sequences of $\langle \Sigma, \mathcal{R} \rangle$ are precisely the suffixes of runs of $\mathcal{S}$. Thus $\mathcal{S}$ has an infinite run iff the domain $\langle \Sigma, \mathcal{R} \rangle$ has an infinite decreasing sequence. Therefore the termination of $\mathcal{S}$ is equivalent to the well-foundedness of $\langle \Sigma, \mathcal{R} \rangle$, and a ranking function for $\langle \Sigma, \mathcal{R} \rangle$ certifies that $\mathcal{S}$ terminates.

Our algorithm generates ranking functions that map $\langle \Sigma, \mathcal{R} \rangle$ into the well-founded domain $\langle Rat_{\Lambda}, >_{\Delta} \rangle$ of rationals greater than some constant value Λ, with the *discretely-greater-than* relation $>_{\Delta}$, defined by

$$x >_{\Delta} y \quad \text{iff} \quad x \geq y + \Delta \ ,$$

where Δ is a positive constant.

Example

Consider the program TERMINATE, presented in Fig. 1. The expression $-i - j$ decreases by 1 with each iteration of the cycle $\{\ell_0, \ell_1, \ell_2\}$, and its value is bounded from below by $-100 - k_0$, where k_0 is the value of k upon entry into the loop. Therefore, $-i - j$ defines a ranking function for the cycle, and the program terminates.

```
local $i, j, k$ : integer
$\ell_0$ : while $i \leq 100 \land j \leq k$ do
    [ $\ell_1 : (i, j) := (j, i + 1)$
      $\ell_2 : k := k - 1$ ]
$\ell_3$ : halt
```

Fig. 1. Program TERMINATE

Notice that this system is not finite-state, so termination cannot be established by model checking. In addition, the expression $-i - j$ does not appear anywhere in the program, so analytic heuristics of the form proposed in [DGG00] are unlikely to discover it. Furthermore, the expression k, which seems the most promising analytic ranking function, has no obvious lower bound. In fact, it is bounded from below by $\min(i_0, j_0)$, but it is not clear that discovering this bound is any easier than finding the expression $-i - j$.

In Sec. 4 we will demonstrate that the ranking function $-i - j$ can be generated automatically.

3 Linear Inequalities and Polyhedral Cones

Our method is inherently deductive. It reduces the synthesis of linear ranking functions to the search for linear inequalities implied by systems of inequalities extracted from the program. Essential to the method, then, is the approach used to derive such consequences.

Consider any system of linear inequalities $\alpha_{i1} x_1 + \cdots + \alpha_{id} x_d \leq 0$. Recall that the inequality $\alpha_1 x_1 + \cdots + \alpha_d x_d \leq 0$ is a *consequence* of the system iff it is satisfied by every solution of the system. Two well-known rules for deducing consequences of such systems are that any inequality can be scaled by a non-negative factor, and that any pair of inequalities can be added. These two inference rules can be combined to yield a single rule which derives any non-negative linear combination of inequalities. It is this sound and complete inference rule which motivates the treatment of linear inequalities developed here.

A vector w is a *conic combination* of vectors $v_1, \ldots, v_n$ iff $w = \Sigma_i \lambda_i v_i$ for scalars $\lambda_i \geq 0$. A *cone* is any set of vectors closed under conic combinations. Thus a cone is a (vector) space in which the linear combinations are restricted to non-negative factors. Every space is a cone since spaces are closed under negation. As the vector 0 is the conic combination of the empty set, the least cone is $\{0\}$, not $\emptyset$. The greatest cone is the set containing all vectors (of a given dimension).

It is easy to see that the intersection of two cones is again a cone. However, the union of two cones need not form a cone. This observation motivates the introduction of the following two concepts. The *conic hull* of a set of vectors V, written $Con(V)$, is the set of conic combinations of V. The *conic union* of two cones C_1, C_2, written $C_1 \uplus C_2$, is the cone $\mathcal{C}on(C_1 \cup C_2)$. Thus the conic hull of a set is the least cone containing it, while the conic union of two cones is the least cone containing both.

A set of vectors R is called a *ray* (or *half-line*) if $R = \mathcal{C}on(r)$ for some vector $r \neq 0$. A set of vectors L is called a *line* if $L = \mathcal{L}in(l)$ for some vector $l \neq 0$, where the *linear hull* $\mathcal{L}in(V)$ is the set of linear combinations of V. Thus a ray is a unit cone, while a line is a unit space. A pair $G = \langle L, R \rangle$ of lines L and rays R is called a *generator* of the cone C iff $C = \mathcal{L}in(L) \uplus \mathcal{C}on(R)$. Lines are not essential components of generators. Since $\mathcal{L}in(l) = \mathcal{C}on(l) \uplus \mathcal{C}on(-l)$, every line can be replaced by a pair of rays in opposite directions without changing the generated cone. To simplify the theory, we assume all lines have been eliminated in this manner. In practice, however, maintaining an explicit representation of lines improves both the space and time complexity of algorithms on cones [Tel82,Wil93].

Notice that every cone has a generator, as C certainly generates itself. However, unlike spaces, some cones admit only infinite generators. Cones which do admit finite generators are said to be *polyhedral.*

Returning to linear inequalities, the inequality $\alpha_1 x_1 + \cdots + \alpha_d x_d \leq 0$ can be represented by the vector $(\alpha_1, \ldots, \alpha_d)$ of its coefficients, and the ray determined by this vector is the set of consequences of the inequality. A system of inequalities is represented by the set of its coefficient vectors, and the cone generated by these rays yields precisely the consequences of the system – a fact which we prove presently. Should the system also contain equalities, they can be represented either implicitly, as pairs of rays in opposite directions, or explicitly, as lines.

The *polar* C^* of a cone C is the set of vectors forming non-acute angles with every member of C, i.e., the set $\{u \mid u \cdot v \leq 0 \text{ for all } v \in C\}$, as illustrated in Fig. 2. The polar of a cone is itself a cone. In fact, the polar of any set of vectors is a cone, but our interest here lies in polars of cones. Polars of cones have the following properties

- $(C^*)^* \supseteq C$,
- $(C_1 \cap C_2)^* \supseteq C_1^* \uplus C_2^*$,
- $(C_1 \uplus C_2)^* = C_1^* \cap C_2^*$.

For arbitrary cones, the two inclusions cannot be strengthened to equalities. However, for polyhedral cones we have

- $(C^*)^* = C$,
- $(C_1 \cap C_2)^* = C_1^* \uplus C_2^*$.

These equalities are implied by a fundamental result, due to Weyl and Minkowski, that a cone is polyhedral iff its polar is polyhedral. Another fundamental theorem concerning polars of polyhedral cones is the following.

Theorem 1 (Alternative) *Let $G = \{r_1, \ldots, r_n\}$ be a set of vectors and r be a vector. Either i) $r \in \mathcal{C}on(G)$ or ii) $v \cdot r > 0$ for some $v \in G^*$,* but not both*.*

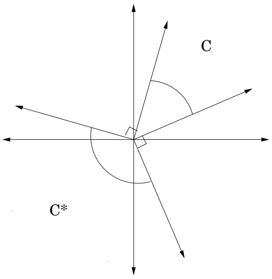

Fig. 2. A cone and its polar.

Our primary interest in polar cones is driven by the relationship they bear to solutions of systems of inequalities. A vector $(x_1, \ldots, x_d)$ is a solution of the inequality $\alpha_1 x_1 + \cdots + \alpha_d x_d \leq 0$ iff $(x_1, \ldots, x_d) \cdot (\alpha_1, \ldots, \alpha_d) \leq 0$. For a system of inequalities, the set of all solutions is precisely the polar of its set of coefficient vectors. With this observation, we are in a position to justify the soundness and completeness of the inference rule presented above.

Theorem 2 (Farkas' Lemma) *Let $\alpha_{i1} x_1 + \cdots + \alpha_{id} x_d \leq 0$ be a system of inequalities. Let $G = \{r_i\}$, where $r_i = (\alpha_{i1}, \ldots \alpha_{id})$. Then $\alpha_1 x_1 + \cdots + \alpha_d x_d \leq 0$ is a consequence of the system iff $(\alpha_1, \ldots, \alpha_d) \in \mathcal{C}on(G)$.*

This result is easily proved using the theorem of the alternative. In the case of soundness, assume $(\alpha_1, \ldots, \alpha_d) \in \mathcal{C}on(G)$. Then for all $(x_1, \ldots, x_d) \in G^*$, $(x_1, \ldots, x_d) \cdot (\alpha_1, \ldots, \alpha_d) \leq 0$. That is, every solution of the system satisfies the inequality. For completeness, assume $(\alpha_1, \ldots, \alpha_d) \notin \mathcal{C}on(G)$. Then for some

$(x_1, \ldots, x_d) \in G^*$, $(x_1, \ldots, x_d) \cdot (\alpha_1, \ldots, \alpha_d) > 0$. That is, some solution of the system fails to satisfy the inequality.

Thus far, we have demonstrated that the polar of a system of linear inequalities represents its solutions. Another perspective on these polars is that they too represent systems of inequalities. In this case, the inequalities are constraints not on solutions, but on the coefficients of consequences of the original system. That is, any solution $(\alpha_1, \ldots, \alpha_d)$ of the polar is in fact the coefficient vector of an inequality implied by the original system, as $(C^*)^* = C$. This perspective on polars of systems of linear inequalities is also integral to the method for synthesizing ranking functions presented here. By computing the polar of a system, adding additional inequalities, and computing the polar of the augmented polar, the algorithm presented in Sec. 4 derives those consequences of systems which satisfy syntactic criteria sufficient to guarantee the existence of ranking functions.

To compute polars, our method uses an algorithm, known as the Double Description Method, which is based on Motzkin's constructive proof of Minkowski's theorem [MRTT53,FP96]. This algorithm constructs the generator of the polar incrementally by successively intersecting the generator of the entire space with the polar of each ray in the generator of the cone.

The algorithm also serves as the basis for implementing additional operators on cones. For two polyhedral cones C_1, C_2 such that $C_i = \mathcal{C}on(G_i) = Con(H_i)^*$,

- $C_1 \uplus C_2 = \mathcal{C}on(G_1 \cup G_2)$,
- $C_1 \cap C_2 = \mathcal{C}on(H_1 \cup H_2)^*$, and
- $C_1 \subseteq C_2$ iff $r \cdot s \leq 0$ for every $r \in G_1$ and $s \in H_2$.

Another useful operation on polyhedral cones is projection onto a space. It is performed by intersecting the cone with the space and eliminating positions that are zero in every ray of the resulting generator. When the space has a basis consisting of canonical unit vectors, we can simply eliminate those positions in the generator of the cone that are zero in every line of the basis of the space.

For a more thorough discussion of the theory of polyhedral cones, the reader is referred to [Gal60,Sch86]. Those interested in the implementation of algorithms on cones should consult [Wil93].

Inhomogeneous Inequalities

Having demonstrated the equivalence of systems of homogeneous inequalities and polyhedral cones, we turn now to inhomogeneous systems and argue that they too can be represented as polyhedral cones.

Consider an inhomogeneous system $\alpha_{i1}x_1 + \cdots + \alpha_{id}x_d + \beta_i \leq 0$. The set of solutions of such a system is no longer a polyhedral cone, but rather a polyhedral convex set. However, by the addition of a single variable χ, the solutions of an inhomogeneous system can be embedded in a polyhedral cone. To see this, consider that the inhomogeneous system is equivalent to the homogeneous system $\alpha_{i1}x_1 + \cdots + \alpha_{id}x_d + \beta_i\chi \leq 0$, along with the single inhomogeneous side condition $\chi = 1$. That is, $(x_1, \ldots, x_d)$ is a solution of the inhomogeneous system iff $(x_1, \ldots, x_d, 1)$ is a solution of its homogeneous embedding.

While the set of solutions of an inhomogeneous system is a polyhedron, not a cone, its set of consequences remains a cone. Taking conic combinations of inequalities continues to be a sound inference rule when applied to inhomogeneous systems. Furthermore, with two minor modifications, it is also complete. First, the tautology $-1 \leq 0$ must be added to the system. The need for this apparently redundant inequality can be attributed to the side condition of the homogeneous embedding. Although $\chi = 1$ cannot be represented explicitly in the embedding, it has as a consequence the homogeneous inequality $-\chi \leq 0$, which is representable, but not derivable. Therefore, this inequality must be added explicitly.

The second modification concerns unsatisfiable systems. Consider the system consisting of the single inequality $1 \leq 0$. The system is unsatisfiable, so every inequality is a consequence of it. For example, $x_i \leq 0$ is a consequence for any i. However, $x_i \leq 0$ is not a conic combination of $1 \leq 0$ and $-1 \leq 0$, where the second inequality is added for the reasons previously given. For unsatisfiable systems, the taking of conic combinations is sound, but incomplete. Care must be taken, then, to detect unsatisfiablility and to replace unsatisfiable systems with an equivalent system which generates all inequalities.

The next theorem, which we state without proof, provides a procedure for detecting unsatisfiability and shows that inferring conic combinations is sound and complete for satisfiable systems.

Theorem 3 *Let $\alpha_{i1}x_1 + \cdots + \alpha_{id}x_d + \beta_i \leq 0$ be a system of inequalities. Let $r_0 = (0, \ldots, 0, -1)$, and let $r_i = (\alpha_{i1}, \ldots, \alpha_{id}, \beta_i)$. Let $G = \{r_i\}$. The system is unsatisfiable iff $(0, \ldots, 0, 1) \in \mathcal{C}on(G)$. When satisfiable, $\alpha_1 x_1 + \cdots + \alpha_d x_d + \beta \leq 0$ is a consequence iff $(\alpha_1, \ldots, \alpha_d, \beta) \in \mathcal{C}on(G)$.*

Strict Inequalities

Consider now the mixed inhomogeneous system $\alpha_{i1}x_1 + \cdots + \alpha_{id}x_d + \beta_i \; \{<, \leq\} \; 0$, containing both strict and weak inequalities. The solutions of this system are embedded in the cone of solutions of the weak homogeneous system $\alpha_{i1}x_1 + \cdots + \alpha_{id}x_d + \beta_i\chi + \delta_i\epsilon \leq 0$, where δ_i is positive or zero depending on whether the ith inequality is strict or weak, along with the side conditions $\chi = 1$ and $\epsilon > 0$. That is, $(x_1, \ldots, x_d)$ is a solution of the original system iff $(x_1, \ldots x_d, 1, \epsilon)$ is a solution of its embedding for some $\epsilon > 0$.

The consequences of the mixed inhomogeneous system are the members of the cone generated by its weak homogeneous embedding, provided the embedding is augmented with two additional inequalities. First, it is necessary to add $-\chi + \epsilon \leq 0$, which is the representation of the tautology $-1 < 0$. Second, $-\epsilon \leq 0$ must also be added, as it is a representable, but not derivable consequence of the side condition $\epsilon > 0$. The presence of this second inequality guarantees that the coefficient of ϵ in any consequence of the weak system can be driven to zero.

Thus, it plays the role of the well-known inference rule for mixed systems which allows any strict inequality to be weakened.

The following theorem, a variant of which appears to have been first proved by Kuhn [Kuh56], demonstrates the soundness and completeness of this approach to mixed inhomogeneous systems.

Theorem 4 *Let* $\alpha_{i1}x_1 + \cdots + \alpha_{id}x_d + \beta_i \ \{<, \leq\}\ 0$ *be a system of inequalities. Let* $r_{-1} = (0, \ldots, -1, 1)$, $r_0 = (0, \ldots, 0, -1)$ *and* $r_i = (\alpha_{i1}, \ldots, \alpha_{id}, \beta_i, \delta_i)$, *where* $\delta_i > 0$ *when strict and* $\delta_i = 0$ *when weak. Let* $G = \{r_i\}$. *The system is unsatisfiable iff* $(0, \ldots, 0, 1) \in \mathcal{C}on(G)$. *When satisfiable,* $\alpha_1 x_1 + \cdots + \alpha_d x_d + \beta \ \{<, \leq\}\ 0$ *is a consequence iff* $(\alpha_1, \ldots, \alpha_d, \beta, \delta) \in \mathcal{C}on(G)$ *for some appropriate* δ.

Note that if our interest lies in just the weak consequences of a mixed system, we can simply treat each strict inequality as if it were weak. However, there is no generator of only the strict consequences. In fact, that set is not a cone as it is not closed under scaling by zero.

4 Generating Linear Ranking Functions

Our objective is to show that a transition system $\mathcal{S} = \langle V, \Theta, \mathcal{T} \rangle$ terminates. We do so by attempting to find a ranking function for each cycle in $\mathcal{S}$.

Let $\langle \Sigma, \mathcal{R} \rangle$ be the domain associated with $\mathcal{S}$, as described in Sec. 2, and let $\langle \Sigma^{\mathcal{A}}, \mathcal{R}^{\mathcal{A}} \rangle$ be a finite abstraction of $\langle \Sigma, \mathcal{R} \rangle$ with abstraction function $\alpha : \Sigma \mapsto \Sigma^{\mathcal{A}}$ for some finite set $\Sigma^{\mathcal{A}}$. That is, $(s_1^{\mathcal{A}}, s_2^{\mathcal{A}}) \in \mathcal{R}^{\mathcal{A}}$ iff there exists $s_1, s_2 \in \Sigma$ such that $s_1^{\mathcal{A}} = \alpha(s_1)$ and $s_2^{\mathcal{A}} = \alpha(s_2)$ and $(s_1, s_2) \in \mathcal{R}$. Thus $\Sigma^{\mathcal{A}}$ induces a finite partition on Σ. When $\Sigma^{\mathcal{A}}$ is the partition induced by the control variables of the system, $\langle \Sigma^{\mathcal{A}}, \mathcal{R}^{\mathcal{A}} \rangle$ is a called the *control flow graph* of $\langle \Sigma, \mathcal{R} \rangle$. Let $\gamma : \Sigma^{\mathcal{A}} \mapsto 2^{\Sigma}$ be the function that maps each abstract state $s^{\mathcal{A}}$ to the set of concrete states it represents, i.e., $\{s \in \Sigma \mid \alpha(s) = s^{\mathcal{A}}\}$. To show that $\langle \Sigma, \mathcal{R} \rangle$ is well-founded it suffices to show that for each cycle in $\langle \Sigma^{\mathcal{A}}, \mathcal{R}^{\mathcal{A}} \rangle$, no infinite decreasing sequence of $\langle \Sigma, \mathcal{R} \rangle$ is mapped to that cycle. This is so because any infinite decreasing sequence in $\langle \Sigma, \mathcal{R} \rangle$ is mapped to an infinite decreasing sequence in $\langle \Sigma^{\mathcal{A}}, \mathcal{R}^{\mathcal{A}} \rangle$, which must end in a cycle, as $\Sigma^{\mathcal{A}}$ is finite.

Consider an arbitrary cycle $\mathcal{C}^{\mathcal{A}} \subseteq \Sigma^{\mathcal{A}}$ of $\langle \Sigma^{\mathcal{A}}, \mathcal{R}^{\mathcal{A}} \rangle$ and let $c^{\mathcal{A}} \in \mathcal{C}^{\mathcal{A}}$ be any element of that cycle. Let $\mathcal{R}_{\mathcal{C}}^{\mathcal{A}}$ be the composition of the transition relations along the cycle from $c^{\mathcal{A}}$ back to $c^{\mathcal{A}}$. Any infinite decreasing sequence that ends in this cycle induces an infinite decreasing sequence in $\langle \{c^{\mathcal{A}}\}, \mathcal{R}_{\mathcal{C}}^{\mathcal{A}} \rangle$ and hence in $\langle \gamma(c^{\mathcal{A}}), \gamma(\mathcal{R}_{\mathcal{C}}^{\mathcal{A}}) \rangle$. Our approach to proving the well-foundedness of $\langle \Sigma, \mathcal{R} \rangle$ is to prove that for each cycle of $\langle \Sigma^{\mathcal{A}}, \mathcal{R}^{\mathcal{A}} \rangle$ there exists some element $c^{\mathcal{A}}$ such that $\langle \gamma(c^{\mathcal{A}}), \gamma(\mathcal{R}_{\mathcal{C}}^{\mathcal{A}}) \rangle$ is well-founded.

In the remainder of this section we will assume that the well-foundedness of $\langle \Sigma, \mathcal{R} \rangle$ is to be established, where Σ stands for $\gamma(c^{\mathcal{A}})$, the set of states accessible at the chosen element of the cycle, and $\mathcal{R}$ stands for $\gamma(\mathcal{R}^{\mathcal{A}})$, the transition relation around the cycle. To show that $\langle \Sigma, \mathcal{R} \rangle$ is well-founded, our algorithm attempts to generate all functions f of the form

$$f : \alpha_1 x_1 + \ldots + \alpha_d x_d,$$

that map $\langle \Sigma, \mathcal{R} \rangle$ into the well-founded domain $\langle Rat_{\Lambda}, >_{\Delta} \rangle$, for some constant Λ and positive constant Δ. The algorithm computes all functions definable as linear expressions over the rational program variables $x_1, \ldots, x_d$, which are bounded and discretely decreasing for $\langle \Sigma, \mathcal{R} \rangle$. It does so by computing approximations of the set of bounded expressions and the set of decreasing expressions and taking their intersection.

The principle behind the algorithm is the representation of these sets as polyhedral cones. Up to this point, we have demonstrated that polyhedral cones can conveniently represent systems of linear inequalities. But notice that the generator of all inequalities implied by the system is also the generator of all linear expressions that are non-positive in every solution of the system. Our algorithm exploits this observation to derive a generator of the bounded decreasing expressions for $\langle \Sigma, \mathcal{R} \rangle$ from two systems of linear inequalities – the first characterizing the set Σ and the second characterizing $\mathcal{R}$.

Computing the Cycle Invariant

The first step of the algorithm is to compute an invariant $\mathcal{I}$ characterizing Σ. For this step we assume the existence of an invariant generator that extracts an invariant for each control location from the description of the system. Furthermore we posit that the generated invariants are systems of linear inequalities, and that they are sufficiently strong. These assumptions are reasonable, given the success of the automatic invariant generation techniques proposed in [CH77].

In an effort to increase the utility of the generated invariant for computing the set of bounded expressions in the next step of the algorithm, we automatically augment the system with auxiliary variables. For each system variable, an auxiliary variable is added which assumes the value of the corresponding system variable upon entry into the cycle and which never changes in value while the computation remains within the cycle. Thus, these variables can be considered symbolic constants within the cycle.

To see the effect of such augmentation, consider again the program TERMINATE, shown in Fig. 1. An invariant for location ℓ_1 is

$$\mathcal{I}_{-} : \ i \leq 100 \ \wedge \ j - k \leq 0 \ .$$

This invariant bounds i from above by the constant 100, but neither j nor k is bounded. However, the augmented program, shown in Fig. 3, produces the invariant

$$\mathcal{I} : \ i \leq 100 \ \wedge \ j - k \leq 0 \ \wedge \ \ k - k_0 \leq 0 \ ,$$

in which i, j, and k are all bounded (since $j \leq k_0$ is a consequence of $j - k \leq 0$ and $k - k_0 \leq 0$).

Fig. 4 shows the generator of the consequences of $\mathcal{I}$, where, as explained in Sec. 3, the first ray represents the tautology $-1 < 0$, the second ray allows strict inequalities to be weakened, and the remaining rays represent the three conjuncts of the invariant.

```
$i, j, k$ : integer
$i_0, j_0, k_0$ : integer

$\ell_{-1}$ : $(i_0, j_0, k_0) := (i, j, k)$
  $\ell_0$ : while $i \leq 100 \wedge j \leq k$ do
        $\left[\begin{array}{l} \ell_1 : (i, j) := (j, i+1) \\ \ell_2 : k := k - 1 \end{array}\right]$
  $\ell_3$ : halt
```

Fig. 3. Augmented version of TERMINATE

$$G_1 : \left\{ \begin{array}{rrrrrrrrrl} & i, & j, & k, & i_0, & j_0, & k_0, & \chi, & \epsilon & \\ r^1_{-1} : & (0, & 0, & 0, & 0, & 0, & 0, & -1, & 1) & -1 < 0 \\ r^1_0 : & (0, & 0, & 0, & 0, & 0, & 0, & 0, & -1) & <\to\leq \\ r^1_1 : & (1, & 0, & 0, & 0, & 0, & 0, & -100, & 0) & i - 100 \leq 0 \\ r^1_2 : & (0, & 1, & -1, & 0, & 0, & 0, & 0, & 0) & j - k \leq 0 \\ r^1_3 : & (0, & 0, & 1, & 0, & 0, & -1, & 0, & 0) & k - k_0 \leq 0 \end{array} \right\}$$

Fig. 4. Generator of the consequences of $\mathcal{I}$

Computing the Bounded Expressions

The second step of the algorithm is to compute the generator of bounded expressions. Recall that a function $f : \Sigma \mapsto Rat$ is bounded if there exists a constant Λ such that $f(s) \geq \Lambda$ for every $s \in \Sigma$. That is, f is bounded if $-f + \Lambda \leq 0$ is implied by $\mathcal{I}$, or equivalently, if $-f + \Lambda$ is in the cone generated by $\mathcal{I}$, for some constant expression Λ. The generator of negations of bounded expressions is computed by projecting $\mathcal{I}$ onto the system variables. In fact, we project $\mathcal{I}$ with the ray $(0, \ldots, 0, 1)$ added, since strictness is not relevant for establishing boundedness. We then negate this generator, using the following result.

Proposition 1 *Let $G = \{r_i\}$, where $r_i = (\alpha_{i1}, \ldots, \alpha_{id}, \delta_i)$, and $G' = \{r'_i\}$, with $r'_i = (-\alpha_{i1}, \ldots, -\alpha_{id}, \delta_i)$. Then $(\alpha_1, \ldots, \alpha_d, \delta) \in G$ iff $(-\alpha_1, \ldots, -\alpha_d, \delta) \in G'$.*

Fig. 5 presents the generator of bounded expressions for program TERMINATE.

Computing the Decreasing Expressions

The third step of the algorithm is to compute a generator of expressions that decrease discretely around the cycle. Recall that a function $f : \Sigma \mapsto Rat$ is discretely decreasing if there exists a positive constant Δ such that, for every $(s, s') \in \mathcal{R}$, $f(s) \geq f(s') + \Delta$. Thus, the discretely decreasing expressions are exactly those expressions f such that $f \geq f' + \Delta$ is implied by the transition

$$G_2 : \left\{ \begin{array}{lrrrl} & i, & j, & k, \epsilon & \\ l_1^2 : & (\; 0, & 0, & 0, \; 1) & \text{any strictness} \\ r_1^2 : & (-1, & 0, & 0, \; 0) & -i \text{ is bounded} \\ r_2^2 : & (\; 0, & -1, & 1, \; 0) & -j+k \text{ is bounded} \\ r_3^2 : & (\; 0, & 0, & -1, \; 0) & -k \text{ is bounded} \end{array} \right\}$$

Fig. 5. Generator of bounded expressions

relation $\mathcal{R}$, for some positive constant Δ. Alternatively, they are those f for which $-f + f' + \Delta$ is in the cone generated by $\mathcal{R}$, with $\Delta > 0$ implied by $\mathcal{I}$.

The generator of the decreasing expressions is computed incrementally: First we transform $\mathcal{I}$ into a generator of the positive constant expressions Δ. Then we restrict $\mathcal{R}$ to generate only those expressions of the form $-f + f' + \Delta$, with Δ a constant expression. This restricted generator is then further constrained to ensure that Δ is in fact positive. This result, when projected onto the coefficients of primed system variables, yields the set of decreasing expressions.

The positive constant expressions Δ cannot be represented directly, as they do not form a cone. For example, they are not closed under scaling by zero. Therefore, we adopt the technique introduced in Sec. 3 for representing strict inequalities, and compute a generator of the non-negative constant expressions along with an indication of strictness. Now, Δ is a non-negative constant expression iff $-\Delta$ is non-positive and the coefficients of the system variables, both primed and unprimed, are zero in Δ. That is, for Δ to be a constant expression, only the auxiliary variables can have non-zero coefficients.

Recall that $\mathcal{I}$ generates the set of all non-positive expressions. So the polar of $\mathcal{I}$ is a system of constraints on the coefficients of these expressions, and every solution of the polar is the coefficient vector of some non-positive expression. Adding the equalities $\alpha_1 = 0, \ldots, \alpha_d = 0$ to the polar yields the subset of non-positive expressions in which the system variables all have zero coefficients, assuming d system variables. Thus, the polar of the augmented polar is precisely the set of non-positive constant expressions. By negating the generator of this set, we arrive at a generator of non-negative constant expressions.

Applying this transformation to the invariant $\mathcal{I}$ of TERMINATE and eliminating the system variables yields the generator shown in Fig. 6. The only positive constant expression is 1.

Next we compute the generator of that subset of the expressions generated by $\mathcal{R}$ which have the form $-f + f' + \Delta$, for some non-negative constant expression Δ. Again, this result is achieved by taking the polar of an augmented polar. First, the ray $(0, \ldots, 0, 1)$ is added to $\mathcal{R}$, and the polar of the augmented generator is computed. The strictness of the non-negativity of expressions in $\mathcal{R}$ is not relevant, and adding the ray eliminates any constraints which $\mathcal{R}$ places on δ. Next, the equalities $\alpha_1 = -\alpha_{d+1}, \ldots, \alpha_d = -\alpha_{2d}$ are added to the polar,

$$G_3 : \left\{ \begin{array}{ll} \quad\; i_0 \; j_0 \; k_0 \;\; \chi \;\; \epsilon & \\ r_1^3 : (0,\ 0,\ \ 0,\ 1,\ 1) & \text{1 is positive} \\ r_2^3 : (0,\ 0,\ \ 0,\ 1,\ 0) & \text{1 is non-negative} \end{array} \right\}$$

Fig. 6. Generator of non-negative constant expressions.

thereby restricting its solutions to those expressions in which the coefficient of each unprimed system variable is the negation of the coefficient of the corresponding primed system variable. Finally, the system is augmented with all of the constraints on the coefficients of non-negative constant expressions. That is, we add all equalities and inequalities that result from taking the polar of the non-negative constant expressions computed earlier.

The resulting system is precisely the set of constraints satisfied by the coefficients of the expressions we seek. The vector $(\alpha_1, \ldots, \alpha_{3d}, \beta, \delta)$ is a solution of this system iff the corresponding expression has the form $-f + f' + \Delta$, where Δ is a non-negative constant expression. Furthermore, if $\delta > 0$, then Δ is positive. Taking the polar of this system and projecting the result onto the coefficients of the primed system variables and ϵ yields the generator of a set of expressions all of whose strict members are discretely decreasing.

Continuing with the program TERMINATE, the generator of the decreasing expressions is shown in Fig. 7.

$$G_4 : \left\{ \begin{array}{ll} \quad\; i,\ j,\ k,\ \epsilon & \\ l_1^4 : (1,\ 1,\ 1,\ 0) & i + j + k \text{ is invariant} \\ r_1^4 : (0,\ 0,\ 1,\ 1) & k \text{ decreases} \\ r_2^4 : (0,\ 0,\ 1,\ 0) & k \text{ does not increase} \end{array} \right\}$$

Fig. 7. Generator of decreasing expressions.

Computing the Ranking Functions

The final step of the algorithm intersects the bounded expressions with the decreasing expressions. Any strict member of the resulting cone is a ranking function.

The generator of the ranking functions for TERMINATE is shown in Fig. 8. Thus $-i - j + k$ is a ranking function. Notice that $-i - j$ is also a ranking function, since $\frac{1}{2}r_1^5 + \frac{1}{2}r_2^5 = (-1, -1, 0, 1)$ is a strict member of the generated cone.

$$G_5 : \left\{ \begin{array}{lcl} & \quad i, \quad j, \quad k, \ \epsilon & \\ r_1^5 : & (-1, -1, -1, 0) & -i-j-k \text{ weakly} \\ r_2^5 : & (-1, -1, \ \ 1, 2) & -i-j+k \text{ strictly} \\ r_3^5 : & (-1, -1, \ \ 1, 0) & -i-j+k \text{ weakly} \end{array} \right\}$$

Fig. 8. Generator of ranking functions.

5 Application

We applied our algorithm to a system modeling the biological mechanism of lateral inhibition in cells, brought to our attention by Ronojoy and Tomlin [RT00]. Lateral inhibition, a mechanism extensively studied in biology [CMML96], causes a group of initially equivalent cells to differentiate. It is based on an intra- and intercellular feedback mechanism, whereby a cell developing into one type inhibits its neighbors from developing into the same type. The result is a more or less regular pattern of cells of different types.

In collaboration with David Dill, we abstracted the (continuous) model of differentiation of skin cells described in [CMML96] and [RT00] into a discrete transition system. The system consists of a planar hexagonal configuration of cells. Cells can be black, white or gray, where black and white cells, if stable, lead to specialization into ciliated and unciliated cells, respectively. Cells transition based on their own color and the colors of their six immediate neighbors. Therefore we define the state of a cell by the two variables $color \in \{w, g, b\}$ and $ncolor \in \{W, G, B\}$, where the value of $ncolor$ is determined as follows:

$$\begin{array}{l} W : \forall i.(n_i = white \vee n_i = gray) \wedge \exists i.(n_i = white) \\ G : \ \forall i.(n_i = gray) \\ B : \ \exists i.(n_i = black) \end{array}$$

with n_i the neighbor cells. The transitions of a cell can then be described by

$$\begin{array}{lll} \tau_1 : w \wedge W \wedge g' & \tau_2 : g \wedge W \wedge b' & \tau_3 : g \wedge G \wedge (b' \vee w') \\ \tau_4 : b \wedge B \wedge g' & \tau_5 : g \wedge B \wedge w' & \end{array}$$

The objective is to prove that this system, like its biological counterpart, stabilizes for an arbitrary number of cells. To do so we attempt to find a ranking function F for the entire plane of cells $\mathcal{C}$. We assume F has the form

$$F = \Sigma_{c \in \mathcal{C}} f(c) \ ,$$

where $f(c)$ is the measure of a single cell. To show that F is a ranking function, it is sufficient to show that its value decreases whenever any cell c transitions. Let c be an arbitrary cell. We can write F as $F = G_c + H_c$ with

$$G_c = f(c) + \Sigma_{i=1}^{6} f(n_i(c)) \quad \text{and} \quad H_c = \Sigma_{d \in \mathcal{C} \setminus \{c, n_1(c) \ldots n_6(c)\}} f(d) \ .$$

To show that F is decreased by every transition of c, it is sufficient to show that G_c is decreased by every transition of c, as transitions of c can affect only the state of c and the state of c's neighbors, so H_c is unaffected. Thus it suffices to consider a group of seven cells (c and its six neighbors) and determine whether a function f exists such that G_c is a ranking function.

Capitalizing on the symmetry in the description, we take as variables the nine states in which a cell and its neighbors can be: $\mathcal{N}_{wW}$, $\mathcal{N}_{wG}$, $\mathcal{N}_{wB}$, $\mathcal{N}_{gW}$, $\mathcal{N}_{gG}$, $\mathcal{N}_{gB}$, $\mathcal{N}_{bW}$, $\mathcal{N}_{bG}$, $\mathcal{N}_{bB}$, with each variable denoting the number of cells among the seven with that configuration, also taking into consideration the colors of the neighbors of the neighbors. For example, if the set consists of a black center cell with six white neighbors, then $\mathcal{N}_{bW} = 1$, $\mathcal{N}_{wB} = 6$ and all other variables are zero[1].

Applying the algorithm to the transition system leads to the following ranking function:

$$24\mathcal{N}_{bB} + 6\mathcal{N}_{wW} + 4\mathcal{N}_{wG} + 5\left(\mathcal{N}_{gW} + \mathcal{N}_{gG} + \mathcal{N}_{gB}\right)$$

This ranking function was found earlier by Dill [Dil00] using an ILP solver. From this ranking function we can conclude that with

$$f(c) = \begin{cases} 24 & \text{if } color = b \text{ and } ncolor = B \\ 6 & \text{if } color = w \text{ and } ncolor = W \\ 4 & \text{if } color = w \text{ and } ncolor = G \\ 5 & \text{if } color = g \\ 0 & \text{otherwise} \end{cases}$$

the function F is a ranking function for the entire plane of cells.

6 Conclusions

We have implemented our algorithm using the polyhedral cone library in the invariant generator of the Stanford Temporal Prover [BBC$^+$95]. Our experience thus far is that simple systems are easily handled, but systems with complex transition relations often exhaust the available memory before a ranking function can be found. This is to be expected, as the library, based on the Double Description Method, represents each cone dually, i.e., by its generator and the generator of its polar. The generator of the polar, however, can be exponentially larger than the generator of the cone [McM70]. We are currently investigating an implementation of our method that avoids this explosion in space by maintaining parametric representations of cones, rather than computing polars explicitly.

Assuming a space-efficient implementation is possible, the method, as presented thus far, might still fail to find a ranking function when one exists. This incompleteness is due to the fact that the required bounds may not be linear expressions in the (auxiliary) variables. Future work includes finding a characterization of the class of systems for which our method is complete.

[1] In this configuration the values of the variables are independent of the colors of the neighbors of the neighbors. In general however, the variables are dependent on them.

References

BBC$^+$95. Nikolaj S. Bjørner, Anca Browne, Eddie S. Chang, Michael Colón, Arjun Kapur, Zohar Manna, Henny B. Sipma, and Tomás E. Uribe. STeP: The Stanford Temporal Prover, User's Manual. Technical Report STAN-CS-TR-95-1562, Computer Science Department, Stanford University, 1995.

BBM97. Nikolaj S. Bjørner, Anca Browne, and Zohar Manna. Automatic generation of invariants and intermediate assertions. *Theoretical Computer Science*, 173(1):49–87, 1997.

BLS96. Saddek Bensalem, Yassine Lakhnech, and Hassen Saidi. Powerful techniques for the automatic generation of invariants. In Rajeev Alur and Thomas A. Henzinger, editors, *CAV'96*, volume 1102 of *LNCS*, pages 323–335. Springer-Verlag, 1996.

CH77. Patrick Cousot and Nicolas Halbwachs. Automatic discovery of linear restraints among variables of a program. In *4th A.C.M. Symposium on Principles of Programming Languages*, 1977.

CMML96. Joanne R. Collier, Nicholas A.M. Monk, Philip K. Maini, and Julian H. Lewis. Pattern formation by lateral inhibition with feedback: a mathematical model of delta-notch intercellular signalling. *J. Theoretical. Biology*, 183:429–446, 1996.

DGG00. Dennis Dams, Rob Gerth, and Orna Grumberg. A heuristic for the automatic generation of ranking functions. In *Workshop on Advances in Verification (WAVe'00)*, pages 1–8, 2000.

Dil00. David Dill. personal communication. August 2000.

FP96. Komei Fukuda and Alain Prodon. Double description method revisited. In *Combinatorics and Computer Science*. 1996.

Gal60. David Gale. *The Theory of Linear Economic Models*. McGraw Hill, 1960.

GW75. Steven M. German and B. Wegbreit. A Synthesizer of Inductive Assertions. *IEEE transactions on Software Engineering*, 1(1):68–75, March 1975.

KM76. Shmuel Katz and Zohar Manna. Logical analysis of programs. *Communications of the ACM*, 19(4):188–206, April 1976.

Kuh56. H. W. Kuhn. Solvability and consistency for linear equations and inequalities. *American Mathematics Monthly*, 63, 1956.

McM70. P. McMullen. The maximum numbers of faces of a convex polytope. *Mathematika*, 17(4):179–184, 1970.

MP95. Zohar Manna and Amir Pnueli. *Temporal Verification of Reactive Systems: Safety*. Springer-Verlag, New York, 1995.

MRTT53. T.S. Motzkin, H. Raiffa, G. L. Thompson, and R. M. Thrall. The double description method. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games II*. Princeton University Press, 1953.

RT00. Gosh Ronojoy and Claire Tomlin. Lateral inhibition through delta-notch signaling. a piecewise affine hybrid model. Technical report, Stanford University, 2000.

Sch86. Alexander Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1986.

Tel82. J. Telgen. Minimal representation of convex polyhedral sets. *Journal of Optimization Theory and Application*, 38:1–24, 1982.

Wil93. Doran K. Wilde. A library for doing polyhedral operations. Technical report, IRISA, 1993.