Constraint-Based Linear-Relations Analysis

Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna ⋆

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

Abstract. Linear-relations analysis of transition systems discovers linear invariant relationships among the variables of the system. These relationships help establish important safety and liveness properties. Efficient techniques for the analysis of systems using polyhedra have been explored, leading to the development of successful tools like HyTech. However, existing techniques rely on the use of approximations such as widening and extrapolation in order to ensure termination. In an earlier paper, we demonstrated the use of Farkas Lemma to provide a translation from the linear-relations analysis problem into a system of constraints on the unknown coefficients of a candidate invariant. However, since the constraints in question are non-linear, a naive application of the method does not scale. In this paper, we show that by some efficient simplifications and approximations to the quantifier elimination procedure, not only does the method scale to higher dimensions, but also enjoys performance advantages for some larger examples.

1 Introduction

Linear-relations analysis discovers linear relationships among the variables of a program, that hold in all the reachable program states. Such relationships are called *linear invariants*. Invariants are useful in the verification of both safety and liveness properties. Many existing techniques rely on the presence of these invariants to prove properties of interest. Some types of analysis, e.g., *variable-bounds* analysis, can be viewed as specializations of linear-relations analysis. Traditionally, this analysis is framed as an abstract interpretation in the domain of polyhedra [7, 8]. The analysis is carried out using a *propagation-based* technique, wherein polyhedral iterates that converge towards the final result, are computed. This convergence is ensured through the use of *widening*, or *extrapolation*, operators. Such techniques are popular in the domains of discrete and hybrid programs, motivating tools like HyTech [12] and improved widening operators over polyhedra [11, 1].

⋆ This research was supported in part by NSF grants CCR-01-21403, CCR-02-20134 and CCR-02-09237, by ARO grant DAAD19-01-1-0723, by ARPA/AF contracts F33615-00-C-1693 and F33615-99-C-3014, and by NAVY/ONR contract N00014-03-1-0939.

Alternatively, the fixpoint equations arising from abstract interpretation may be posed explicitly, and solved without relying directly on iteration or widening. This is achieved through applications of *Farkas Lemma* in our earlier work [6]. Given a *template inequality* with unknown coefficients, our technique computes constraints on the values of the coefficients, such that substituting any solution back into the template yields a valid invariant relationship. However, the constraints themselves are non-linear with existentially quantified parameters. Nevertheless, an exact elimination is possible in theory through quantifier elimination techniques for the theory of reals [16, 5, 17]. In practice, however, the technique using exact quantifier elimination does not scale to systems with more than five variables.

Fortunately, the constraints obtained in this process, though non-linear, exhibit many structural properties that can be exploited to simplify and solve them. In many cases, a series of simplifications resolves the constraints into a linear system. For instance, whenever the underlying transition system is a Petri net, the system of constraints resolves into a linear system [14]. This has led us to verify transition systems derived from Petri Nets with as many as 40 dimensions and 50 transitions. The use of quantifier elimination is clearly inefficient in such situations. In this paper, we provide a set of exact and heuristic rules for simplifying and solving the constraints for general linear transition systems. Most of our rules are exact, but their application may not resolve the constraints. Therefore, some heuristics are used instead of an exact elimination as a last resort.

At lower dimensions, our technique performs poorly in terms of time and space, relative to the propagation-based approach. When the dimension is increased, our technique not only scales but in some cases, outperforms the propagation-based techniques. Furthermore, our technique enjoys several advantages over related approaches that are very useful for analyzing larger systems, as presented in Section 4. The remainder of this paper consists of Section 2 on preliminaries, Section 3 on the constraint structure and solving rules, and Section 4 on some experimental results.

2 Preliminaries

We recount some standard results on polyhedra, and then define linear transition systems, followed by a description of *propagation-based* analysis techniques. We then demonstrate an alternative approach called *constraint-based* analysis.

2.1 Linear Assertions

Through this discussion, let $\{x_1, \ldots, x_n\}$ be a set of real-valued variables. Constant reals are denoted by a, b with subscripts, and unknown coefficients by c, d with subscripts. Further details about linear assertions can be obtained from standard texts [15].

Definition 1 (Linear Assertions) A *linear expression* is of the form $a_1x_1 + \cdots + a_nx_n + b$. The expression is *homogeneous* iff $b = 0$, or else it is *inhomogeneous.* A *linear inequality* is of the form $\alpha \bowtie 0$, where $\bowtie \in \{\geq, =, >\}$. The inequality is *strict* if $\bowtie \in \{>\}$. A *linear assertion* is a finite conjunction of linear inequalities. Linear assertions can be homogeneous or otherwise, depending on the underlying linear expressions. The set of points in $\mathcal{R}^n$ satisfying a linear assertion (homogeneous assertion) is called a *polyhedron* (*polyhedral cone*).

We shall assume that linear assertions do not contain any strict inequalities. It is well-known that any polyhedron is representable by a set of constraints (as a linear assertion), or by its *vertices*, and *rays* (infinite directions), collectively called its *generators.* The problem of computing the generators, given the assertion, and vice-versa have been well-studied with efficient algorithms [9]. However, the number of generators of a polyhedron can be worst-case exponential in the number of constraints (the n-dimensional hypercube is an example). Basic operations on these assertions are computed thus:

Intersection Combine the inequalities in both the polyhedra.
Convex Union Combine the generators of the two polyhedra.
Projection Project the generators of the polyhedron.
Containment Test every generator of φ_1 for subsumption by φ_2.
Emptiness A polyhedron is empty iff it has no generators.

We now state *Farkas Lemma*, which describes the linear consequences of a linear assertion. A proof is available from the standard references [15].

Theorem 1 (Farkas Lemma). *Consider a linear assertion S over real-valued variables $x_1, \ldots, x_n$,*

$$S : \begin{array}{rrr} a_{11}x_1 + \cdots + & a_{1n}x_n + & b_1 \geq 0 \\ \vdots & \vdots & \vdots \\ a_{m1}x_1 + \cdots + & a_{mn}x_n + & b_m \geq 0 \end{array}$$

When S is satisfiable, it implies a given linear inequality $\psi : \; a_1x_1 + \cdots + a_nx_n + b \geq 0$, i.e, $S \models \psi$, if and only if there exist non-negative real numbers $\lambda_0, \lambda_1, \ldots, \lambda_m$ such that $a_p = \sum_{i=1}^{m} \lambda_i a_{ip}, \quad p \in [1, n], \quad$ and *$b = (\sum_{i=1}^{m} \lambda_i b_i) + \lambda_0$ Furthermore, S is unsatisfiable if and only if the inequality $-1 \geq 0$ can be derived as shown above.*

In the rest of the paper we represent applications of this lemma by a table as shown below:

$$\begin{array}{c|l}
\lambda_0 & \hfill 1 \geq 0 \\
\begin{array}{c}\lambda_1 \\ \vdots \\ \lambda_m\end{array} & \left.\begin{array}{rrr} a_{11}x_1 + \cdots + & a_{1n}x_n + & b_1 \geq 0 \\ \vdots & \vdots & \vdots \\ a_{m1}x_1 + \cdots + & a_{mn}x_n + & b_m \geq 0 \end{array}\right\} S \\
\hline
 & c_1x_1 + \cdots + \; c_nx_n + \; d \geq 0 \leftarrow \; \psi, \; \textit{or} \\
 & \hfill -1 \geq 0 \leftarrow \; \textit{false}
\end{array}$$

The table shows the antecedents above the line and the consequences below. For each column, the sum of the column entries above the line, with appropriate multipliers, must be equal to the entry below the line. If a row corresponds to an equality rather than an inequality, we drop the requirement that the multiplier corresponding to it be non-negative.

2.2 Transition Systems and Invariants

In this section, we define linear transition systems and linear invariants. Our presentation concentrates only on linear systems. The reader is referred to standard textbooks for a more general presentation [13].

Definition 2 (Linear Transition Systems) Let $V = \{x_1, \ldots, x_n\}$ be a set of *system variables*. A *linear transition system* over V is a tuple $\langle L, \mathcal{T}, \ell_0, \Theta \rangle$, where L is a set of *locations*, $\mathcal{T}$ is a set of *transitions*, each transition $\tau \in \mathcal{T}$ is a tuple $\langle \ell_i, \ell_j, \rho_\tau \rangle$, such that $\ell_i, \ell_j \in L$ are the *pre-* and the *post-* locations, respectively, and ρ_τ is a linear assertion over $V \cup V'$, where V denotes the current-state variables, and V' the next-state variables. Location $\ell_0 \in L$ is the *initial location*, and Θ is a linear assertion over V specifying the initial condition.

Example 1. Let $V = \{x, y\}$ and $L = \{\ell_0\}$. Consider the transition system shown below. Each transition models a concurrent process, that updates the variables x, y atomically.

$$\begin{aligned}
&\Theta = (x = 0 \ \wedge \ y = 0) \\
&\mathcal{T} = \{\tau_1, \tau_2\} \\
&\tau_1 = \left\langle \ell_0, \ell_0, \left[x' = x + 2y \ \wedge \ y' = 1 - y \right] \right\rangle \\
&\tau_2 = \left\langle \ell_0, \ell_0, \left[x' = x + 1 \ \wedge \ y' = y + 2 \right] \right\rangle
\end{aligned}$$

A given linear assertion ψ is a *linear invariant* of a linear transition system (LTS) at a location ℓ iff it is satisfied by every state reaching ℓ. An *assertion map* maps each location of a LTS to a linear assertion. An assertion map η is an invariant map if $\eta(\ell)$ is an invariant at ℓ, for each $\ell \in L$. In order to prove a given assertion map invariant, we use the theory of inductive assertions due to Floyd and Hoare [13].

Definition 3 (Inductive Assertion Maps) An assertion map η is inductive iff it satisfies the following conditions:

Initiation: $\Theta \models \eta(\ell_0)$,
Consecution: For each transition $\tau : \langle \ell_i, \ell_j, \rho_\tau \rangle$, $\eta(\ell_i) \ \wedge \ \rho_\tau \ \models \eta(\ell_j)'$.

It can be shown by mathematical induction that any inductive assertion map is also an invariant map. It is well known that the converse need not be true in general. The standard technique for proving an assertion invariant is to find an inductive assertion that strengthens it. For example, the assertion $x + y \geq 0$ is an invariant for the LTS in Example 1.

Propagation-based Analysis These techniques are based on the *abstract-interpretation* framework formalized by Cousot and Cousot [7], and specialized for linear relations by Cousot and Halbwachs [8]. The technique starts from an inital assertion map, and weakens it iteratively using the *Post* and the *Widening* operators. When the iteration converges, the resulting map is guaranteed to be inductive, and hence invariant. Termination is guaranteed by the design of the widening operator. Often widening is not used, or replaced by an *Extrapolation* operator, and the termination guarantee is traded-off against accuracy.

Definition 4 (Post-condition and Widening Operators) The *post-condition* operator takes an assertion φ, and a transition relation ρ_τ.

$$post(\varphi, \tau) = (\exists V_0)(\varphi(V_0) \ \wedge \ \rho_\tau(V_0, V))$$

Intersection, followed by quantifier elimination using projection computes post. However, more efficient strategies for computing post exist when ρ_τ has a special structure.

Given assertions $\varphi_{\{1,2\}}$ such that $\varphi_1 \models \varphi_2$, the *standard widening* $\varphi_1 \nabla \varphi_2$ is an assertion φ that contains (roughly) all the inequalities in φ_1 that are satisfied by φ_2. The details along with key mathematical properties of widening are described in [8, 7], and enhanced versions appear in [11, 4, 1].

As mentioned earlier the analysis begins with an initial assertion map defined by $\eta_0(\ell_0) = \Theta$, and $\eta_0(\ell) = \emptyset$ for $\ell \neq \ell_0$. At each *step*, the map η_i is updated to map η_{i+1} as follows:

$$\eta_{i+1}(\ell) = \eta_i(\ell) \ \langle \text{OP} \rangle \left[\eta_i(\ell) \bigsqcup_{\tau_j \equiv \langle \ell_j, \ell, \rho \rangle} (post(\eta_i(\ell_j), \tau_j)) \right]$$

where OP is the convex hull ($\sqcup$) operator for a propagation step, and the widening (∇) operator for a widening step. The overall algorithm requires a predefined *iteration strategy.* A typical strategy carries out a predefined sequence of initial propagation steps, followed by widening steps until termination. The choice of a strategy is of the utmost importance for minimizing the number of propagation and widening steps, in general.

The method described above was applied to the LTS in Example 1. Using the standard widening [8], we obtain the sequence of iterates shown in Figure 1. The result does not change even when the number of initial propagation steps k, is increased to 4. Using the widening operator by Bagnara et al., implemented in the PPL library [1], and $k = 4$ does not change the result, even if the number of propagation steps is increased. Surprisingly, when the number of initial propagation steps is reduced to $k = 3$, it yields the invariant $11x+10y \geq 0,\ 11x+12y \geq 0$, for $k = 2$, the invariant $7x + 6y \geq 0,\ 7x + 8y \geq 0$. $k = 0, 1$ also produce the trivial invariant (*true*). This demonstrates that an increase in the number of initial propagation steps does not necessarily increase the accuracy of the result.

Iteration #	$\eta(\ell_0)$	Iteration Type
0	$x = y = 0$	Init
1	$y - 2x \geq 0,\ x - y + 1 \geq 0,\ x \geq 0$	Propagation
2	$\begin{bmatrix} 5x + 3y \leq 22,\ x - y + 2 \geq 0,\ x \geq 0 \\ x + 5y \geq 0,\ 2x - y + 1 \geq 0 \end{bmatrix}$	Propagation
3	$x \geq 0,\ 2x - y + 1 \geq 0$	Widening
4	*true*	Widening

Fig. 1. Sequence of Propagation and Widening Steps for LTS in Example 1.

Constraint-based Analysis The framework of abstract interpretation [7] shows that any semantic analysis can be expressed as a fixpoint equation in an abstract domain. Consequently, linear-relations analysis is a fixed point computation in the domain of polyhedra. This computation is done by iteration in the propagation-based analysis of Cousot and Halbwachs [8]. We propose to use Farkas Lemma to generate constraints from the LTS description, directly describing the relevant fixed-point. The resulting constraints are solved using non-linear quantifier elimination.

Let C be a set of template variables. A *template* is an inequality of the form $c_1x_1 + \cdots + c_nx_n + d \geq 0$, where $c_{1,\ldots,n}, d \in C$. A *template map*, associates each location with a template. We shall use $\eta(\ell)$ to denote both the inequality, and the template expression at ℓ, disambiguated by context. We reduce the inductive assertion generation problem to one of computing those variables for which a given template map η is inductive. The answer consists of encoding initiation and consecution using Farkas Lemma.

Initiation: The implication $\Theta \models \eta(\ell_0)$ is encoded.

Consecution: For each $\tau : \langle \ell_i, \ell_j, \rho_\tau \rangle$, the implication $\eta(\ell_i) \ \wedge \ \rho_\tau \models \eta(\ell_j)'$ is encoded. We shall explore the structure of the resulting constraints in detail through the remainder of the paper.

The definition of consecution can be *relaxed* into two stronger forms:

Local Consecution: For transition $\tau : \langle \ell_i, \ell_j, \rho \rangle$, $\rho \ \models \eta(\ell_j)' \geq 0$,

Increasing Value: For transition $\tau : \langle \ell_i, \ell_j, \rho \rangle$, $\rho \ \models \eta(\ell_j)' \geq \eta(\ell_i)$.

Both these conditions imply consecution. Any map in which some transitions satisfy these stronger conditions continues to remain an inductive assertion map.

Example 2. Consider the LTS in Example 1. We fix a template map $\eta(\ell_0) = c_1x + c_2y + d$, $C = \{c_1, c_2, d\}$ being unknown quantities. Initiation is encoded using Farkas Lemma,

$$\begin{array}{c|l}
\lambda_1 & \left.\begin{array}{lcl} x & & = 0 \\ & y & = 0 \end{array}\right\} \ \Theta \\
\lambda_2 & \\
\hline
 & c_1x + c_2y + d \geq 0 \leftarrow \eta(\ell_0)
\end{array}$$

resulting in the constraints

$$(\exists \lambda_1, \lambda_2)\, [c_1 = \lambda_1 \ \wedge \ c_2 = \lambda_2 \ \wedge \ d \geq 0]$$

After eliminating the multipliers, we obtain $d \geq 0$ for the initiation constraint. Consecution is encoded using Farkas Lemma as

$$\begin{array}{r|l}
\mu_1 & c_1x + c_2y \qquad\qquad\qquad + d \geq 0 \leftarrow \eta(\ell_0) \\
\lambda_1 & \left.\begin{array}{l} \ \ x + \ 2y - \quad x' \qquad\qquad = 0 \\ \qquad\quad y \qquad + \quad y' - 1 = 0 \end{array}\right\} \rho_{\tau_1} \\
\lambda_2 & \\ \hline
 & \qquad\qquad c_1x' + c_2y' + d \geq 0 \leftarrow \eta'(\ell_0)
\end{array}$$

which produces the constraints

$$(\exists \mu_1)\, [\mu_1 c_1 - c_1 = 0 \ \wedge \ \mu_1 c_2 + c_2 - 2c_1 = 0 \ \wedge \ \mu_1 d - d - c_2 \leq 0]$$

After eliminating $\lambda_1, \lambda_2, \mu_1$, the resulting constraint simplifies to $c_2 = c_1 \geq 0$. Similarly, the constraint obtained for τ_2 simplifies to $c_1 + 2c_2 \geq 0$. The overall constraint is the conjunction of the initiation and consecution constraints, which reduces to $c_1 = c_2 \geq 0,\ d \geq 0$. Solutions are generated by $c_1 = 1,\ c_2 = 1,\ d = 0$, corresponding to the inductive assertion $x + y \geq 0$ at ℓ_0.

3 The Constraint System and its Solution

In this section, we study the constraint structure arising from the encoding discussed briefly in Section 2, and in detail elsewhere [6].

We fix a linear transition system Π with variables $\{x_1, \ldots, x_n\}$, collectively referred to as $\boldsymbol{x}$. The system is assumed to have a single location ℓ to simplify the presentation. The template assertion at location ℓ, is $\alpha(\boldsymbol{c}) = c_1x_1 + \cdots + c_nx_n + d \geq 0$. The coefficient variables $\{c_1, \ldots, c_n, d\}$ are collectively referred to as $\boldsymbol{c}$. The system's transitions are $\{\tau_1, \ldots, \tau_m\}$, where $\tau_i : \langle \ell, \ell, \rho_i \rangle$. The initial condition is denoted by Θ. The system in Example 1 will be used as a running example to illustrate the presented ideas.

3.1 Deriving Constraints

We use Farkas Lemma (Theorem 1) in order to derive constraints for initiation and consecution, as shown in Example 2.

Initiation The case for initiation is relatively straightforward. We encode initiation by encoding $\Theta \models \alpha(\boldsymbol{c}) \geq 0$. The conditions on $\boldsymbol{c}$ are obtained from the application of Farkas Lemma after eliminating the multipliers. In practice, the constraints are derived using Farkas' Lemma. The result is a linear assertion over the unknowns $\boldsymbol{c}$, and the multipliers $\boldsymbol{\lambda}$. The multipliers are eliminated using polyhedral projection. Let $\Theta = \ (x = 0 \ \wedge \ y = 0)$, and $c_1x + c_2y + d \geq 0$. The initiation constraint, obtained by using Farkas Lemma is $d \geq 0$, as shown in Example 2.

Consecution Consecution for a transition τ_i encodes the assertion

$$(\alpha(\boldsymbol{c}) \geq 0) \ \wedge \ \rho_i \ \models \ (\alpha(\boldsymbol{c})' \geq 0)$$

Using Farkas Lemma, the constraints obtained are homogeneous, and involve an existentially quantified non-linear parameter μ_i. We shall term the class of constraints thus obtained *parametric linear assertions.*

Definition 5 (Parametric Linear Assertion) Let $\boldsymbol{c}$ be a set of variables and $\mu_1, \ldots, \mu_m$ be *parameters.* A *parametric linear expression* (PL expression) is of the form $\alpha_1 + \mu_i \alpha_2$, where $\alpha_{\{1,2\}}$ are (homogeneous) linear expressions over $\boldsymbol{c}$. A parametric linear (in)equality is of the form $\beta \bowtie 0$, β being a PL expression. A PL assertion is a finite conjunction of PL equalities and inequalities.

For a transition τ_i, and template $\alpha(\boldsymbol{c})$, the consecution constraints obtained through Farkas Lemma form a parametric linear assertion over a single parameter μ_i.

Example 3. We encode consecution for transition τ_2 from Example 1.

$$\begin{array}{r|lcccrl}
\mu_2 & c_1x + c_2y & & & & + d \geq 0 & \leftarrow \eta(\ell_0) \\
\lambda_1 & \ \ x & - & x' & & + 1 = 0 & \big\}\ \rho_{\tau_2} \\
\lambda_2 & & & y & - \ \ y' & + 2 = 0 & \big\} \\
\hline
 & & & & c_1x' + c_2y' & + d \geq 0 & \leftarrow \eta'(\ell_0)
\end{array}$$

which yields the constraints

$$\exists(\mu_2, \lambda_1, \lambda_2) \left[\begin{array}{r}
\mu_2 c_1 + \lambda_1 = 0 \\
\mu_2 c_2 + \lambda_2 = 0 \\
-\lambda_1 = c_1 \\
-\lambda_2 = c_2 \\
\mu_2 d - d + \lambda_1 + 2\lambda_2 \leq 0
\end{array} \right]$$

Eliminating λ_1, λ_2 yields $\mu_2 c_1 - c_1 = 0 \ \wedge \ \mu_2 c_2 - c_2 = 0 \ \wedge \ \mu_2 d - d - c_1 - 2c_2 \leq 0$.

These constraints are parametric linear. Local and increasing consecutions can be enforced by setting $\mu_2 = 0, 1$, respectively.

The Overall Constraint The overall constraint obtained is the conjunction of the constraints obtained from initiation and consecution for each transition. This constraint is a combination of several types of constraints. Initiation results in a linear assertion, whereas each consecution condition results in PL assertions over parameters $M = \{\mu_1, \ldots, \mu_m\}$, the parameter μ_i arising from τ_i. Each of these parameters is required to be nonnegative, and is existentially quantified. In order to compute the actual constraint over $\boldsymbol{c}$, the parameters in M need to be eliminated.

Example 4. The overall constraint for the system in Example 1 is now

$$\exists(\mu_1,\mu_2)\left(\underbrace{d \geq 0}_{\text{Initation}} \wedge \underbrace{\begin{bmatrix} \mu_1 c_1 - c_1 = 0 \\ \mu_1 c_2 + c_2 - 2c_1 = 0 \\ \mu_1 d - d - c_2 \leq 0 \\ \mu_1 \geq 0 \end{bmatrix}}_{\tau_1} \wedge \underbrace{\begin{bmatrix} \mu_2 c_1 - c_1 = 0 \\ \mu_2 c_2 - c_2 = 0 \\ \mu_2 d - d - c_1 - 2c_2 \leq 0 \\ \mu_2 \geq 0 \end{bmatrix}}_{\tau_2}\right)$$

3.2 Exact elimination

The constraints in Example 4 are non-linear and existentially quantified. However, the theory of non-linear assertions over reals admits computable quantifier elimination, as shown by Tarski [16]. Many others have improved the algorithm [5, 17]. Packages like REDLOG and QEPCAD can handle small/medium sized examples. In our earlier work, we used these techniques to handle the constraints derived from elimination. However, there are many drawbacks to using these tools.

1. The technique does not scale to systems of more than five variables.
2. The technique yields large formulas with many non-linear constraints that cancel in the final result, leading to much redundant effort.
3. The structure in the constraints is not fully utilized. The constraints are of low degree, and exhibit a uniform structure. This is lost as soon as some of the parameters are eliminated.
4. In case the underlying LTS has some special structure, the use of elimination may be completely unnecessary, as demonstrated for the case of Petri Net transitions in [14]. The result can be extended to cases where a subset of the transitions have a *Petri-net like* structure, as is the case with many systems.

Of course, the *completeness* of quantifier elimination, and Farkas Lemma lead to theoretical claims of completeness (see [6] for details). We are not aware of any alternative exact procedure for solving these constraints precisely. Therefore, we shall concentrate on under-approximate elimination.

3.3 Under-approximate Elimination Technique

Any under-approximate elimination technique is sound.

Lemma 1. *Let $\psi(\boldsymbol{c},\boldsymbol{\mu})$ be the overall constraints obtained from encoding inductiveness. Let $\varphi(\boldsymbol{c})$ be an assertion such that*

$$\varphi(\boldsymbol{c}) \models (\exists\ \boldsymbol{\mu} \geq 0)\ \psi(\boldsymbol{c},\boldsymbol{\mu})$$

Any solution to φ is an inductive assertion.

Proof. Let $\boldsymbol{c} = \boldsymbol{a}$ be a solution to φ. Then, there exist positive parameters $\boldsymbol{\mu}$ such that $\psi(\boldsymbol{a},\boldsymbol{\mu})$ holds. The rest follows by the soundness of our constraint generation process. See [6] for a proof.

We first split the overall constraints $\psi(\boldsymbol{c}, \boldsymbol{\mu})$ into different groups: $\varphi_{\{eq,in\}}$, $\gamma_{\{eq,in\}}$, and ψ_μ:

- φ_{eq} and φ_{in} contain the equalities and inequalities, respectively, on $\boldsymbol{c}$. We assume $\varphi_{eq} \models \varphi_{in}$.
- γ_{eq} and γ_{in} contain the PL equalities and inequalities, respectively, over $\boldsymbol{c}$ and $\boldsymbol{\mu}$.
- ψ_μ contains the constraints on μ, conjunctions of linear inequalities, equalities, and disequalities, where the disequalities are produced by our constraint solving rules.

Example 5. The constraints from Example 4 are classified as follows:

$$\underbrace{d \geq 0}_{\varphi_{in}} \wedge \underbrace{\begin{bmatrix} \mu_1 c_1 - c_1 = 0 \\ \mu_1 c_2 + c_2 - 2c_1 = 0 \\ \mu_2 c_1 - c_1 = 0 \\ \mu_2 c_2 - c_2 = 0 \end{bmatrix}}_{\gamma_{eq}} \wedge \underbrace{\begin{bmatrix} \mu_1 d - d - c_2 \leq 0 \\ \mu_2 d - d - c_1 - 2c_2 \leq 0 \end{bmatrix}}_{\gamma_{in}} \wedge \underbrace{\begin{bmatrix} \mu_1 \geq 0 \\ \mu_2 \geq 0 \end{bmatrix}}_{\psi_\mu}$$

The *linear part* of a system of constraints is defined as the constraint $\varphi_{in} \wedge \varphi_{eq}$. The system is unsatisfiable if $\varphi_{\{in,eq\}}$ or ψ_μ are, and *trivial* if φ_{eq} is of the form $c_1 = \ldots = c_n = 0$. The only inductive assertion that a trivial system can possibly yield is $1 \geq 0$.

Constraint Simplification The simplifications involving equalities in φ_{eq}, ψ_μ, are the following:

1. Every equality expression in φ_{eq} of the form $a_1 c_1 + \cdots + a_n c_n + a_{n+1} d = 0$ forms a *rewrite rule* of the form $c_i \to -\frac{a_{i+1}}{a_i} c_{i+1} - \cdots - \frac{a_{n+1}}{a_i} d$, where i is the smallest index with $a_i \neq 0$.
2. Apply this rule to eliminate c_i over the linear and PL parts. Simplify, and repeat until all the equalities have been converted.

Similarly, a constraint of the form $\mu = a$ in ψ_μ is used to rewrite μ in $\gamma_{\{eq,in\}}$. The constraints added to $\varphi_{\{eq,in\}}$ can trigger further simplifications and similarly, constraints in γ_{eq} can be used as rewrite rules in order to simplify constraints in γ_{in}.

Factorization and Splitting A PL expression is *factorizable* iff it can be written in the form $(\mu - a)\alpha$, where α is a linear expression over $\boldsymbol{c}$. Deciding if an expression is factorizable is linear time in the expression size. A PL equality $(\mu - a)\alpha = 0$ factorizes into two factors $\mu - a = 0 \ \vee \ \alpha = 0$. Similarly a PL inequality $(\mu - a)\alpha \geq 0$ factorizes into $(\mu - a \geq 0 \wedge \alpha \geq 0) \vee (\mu - a \leq 0 \wedge \alpha \leq 0)$. Since our system of constraints is a conjunction of (in)equalities, factorization splits a constraint system into a *disjunction* of two systems. The following is a factorization strategy, for equalities:

1. Choose a factorizable expression $(\mu - a)\alpha = 0$, and remove it from the constraints,

2. Create two constraint systems, each containing all the remaining constraints. Add $\mu = a$ to one system, rewriting all occurrences of μ by a. Add $\alpha = 0 \ \wedge \ \mu \neq a$ to the other system, and simplify.

Example 6. The constraint system in Example 4 has a factorizable equality $\mu_2 c_i - c_i = 0$, for $i \in \{1, 2\}$. We add $\mu_2 = 1$ to one child, and $c_1 = 0 \ \wedge \ c_2 = 0$ to the other, yielding

$$\left[\begin{array}{rl}
\left.\begin{array}{r} d \geq 0 \\ -c_1 - 2c_2 \leq 0 \end{array}\right\} & \varphi_{in} \\
\left.\begin{array}{r} \mu_1 c_1 - c_1 = 0 \\ \mu_1 c_2 + c_2 - 2c_1 = 0 \end{array}\right\} & \gamma_{eq} \\
\mu_1 d - d - c_2 \leq 0 \leftarrow & \gamma_{in} \\
\left.\begin{array}{r} \mu_1 \geq 0 \\ \mu_2 = 1 \end{array}\right\} & \psi_\mu
\end{array}\right] \vee \left[\begin{array}{rl}
\left.\begin{array}{r} c_1 = 0 \\ c_2 = 0 \end{array}\right\} & \varphi_{eq} \\
d \geq 0 \leftarrow & \varphi_{in} \\
\left.\begin{array}{r} \mu_1 d - d \leq 0 \\ \mu_2 d - d \leq 0 \end{array}\right\} & \gamma_{in} \\
\left.\begin{array}{r} \mu_1 \geq 0 \\ \mu_2 \neq 1 \end{array}\right\} & \psi_\mu
\end{array}\right]$$

The constraints on the right are *trivial.* The system on the left can be factorized using the equality $\mu_1 c_1 - c_1 = 0$. We obtain:

$$\left[\begin{array}{rl}
2c_2 - 2c_1 = 0 \leftarrow & \varphi_{eq} \\
\left.\begin{array}{r} d \geq 0 \\ -c_1 - 2c_2 \leq 0 \\ -c_2 \leq 0 \end{array}\right\} & \varphi_{in} \\
\left.\begin{array}{r} \mu_1 = 1 \\ \mu_2 = 1 \end{array}\right\} & \psi_\mu
\end{array}\right] \vee \left[\begin{array}{rl}
c_1 = 0 \leftarrow & \varphi_{eq} \\
\left.\begin{array}{r} d \geq 0 \\ -c_1 - 2c_2 \leq 0 \end{array}\right\} & \varphi_{in} \\
\mu_1 c_2 + c_2 = 0 \leftarrow & \gamma_{eq} \\
\mu_1 d - d - c_2 \leq 0 \leftarrow & \gamma_{in} \\
\left.\begin{array}{r} \mu_1 \geq 0 \\ \mu_2 = 1 \\ \mu_1 \neq 1 \end{array}\right\} & \psi_\mu
\end{array}\right]$$

The system on the left (unsimplified) has been completely linearized. The system on the right can be further factored using $\mu_1 c_2 + c_2 = 0$, yielding $\mu_1 = -1$ on one side, and $c_2 = 0$ on the other. Setting $\mu_1 = -1$ contradicts $\mu_1 \geq 0$, while setting $c_2 = 0$ makes the system trivial. Therefore, repeated factorization and simplification yields the linear assertion $c_1 = c_2 \ \wedge \ c_1 \geq 0 \ \wedge \ d \geq 0$, which is equivalent to the result of the exact elimination.

Simplification and factorization can be repeatedly applied to split the initial constraints into a tree of constraints, such that each leaf has no more rules applicable. Each node in the tree is equivalent to the disjunction of its children. Therefore, the root is equivalent to the disjunction of all the leaves. The leaves can be either completely resolved (linear), unsatisfiable, trivial, or *terminal.* A terminal leaf is satisfiable and non-trivial, but contains unresolved non-linearities, which cannot be further split or simplified.

Handling Terminal Constraints There are many ways of handling these constraints, some exact and some under-approximate.
Subsumption If a terminal (or even a non-terminal branch) has its linear part subsumed by another fully linear leaf, we can ignore it without loss of accuracy. Checking subsumption allows us to eliminate non-terminal nodes too. Even

#	name	description
1	Simplification	Substitute equalities into Expressions
2	Factorization	Choose a factorizable expression, and split disjuncts
3	Subsumption	Test containment of linear part w.r.t fully-resolved nodes
4	Split	Use lemmas 2,3 to split
5	Instantiate	Set $\mu_i = 0, 1$, split and proceed

Fig. 2. Constraint-simplification rules

though polyhedral containment is expensive for higher-dimensions, we find that a significant fraction of the nodes explored are eliminated this way.
Split In some special cases, it is possible to simplify a terminal system further. The following lemmas are inspired by our work on Petri Net Transitions [14].

Lemma 2. *Let α_1, α_2 be linear expressions, and μ be a parameter not occurring in $\alpha_{\{1,2\}}$. Then*

- $(\exists\, \mu \geq 0)\ (\alpha_1 + \mu\alpha_2 = 0) \ \equiv\ \left[\begin{array}{c} (\alpha_1 = \alpha_2 = 0)\ \vee \\ (\alpha_1 \leq 0\ \wedge\ \alpha_2\ > 0)\ \vee\ (\alpha_1 \geq 0\ \wedge\ \alpha_2 < 0) \end{array}\right]$,
- $(\exists\, \mu \geq 0)\ (\alpha_1 + \mu\alpha_2 \geq 0)\ \equiv\ (\alpha_1 \geq 0)\ \vee\ (\alpha_2 > 0)$.

These lemmas can be extended to systematically handle more complicated constraints on μ. They can also be modified to apply when more than one constraint exists, with loss of completeness.

Lemma 3. $(A\boldsymbol{c} \geq 0)\ \vee\ (B\boldsymbol{c} > 0)\ \models\ (\exists\, \mu \geq 0)\ A\boldsymbol{c} + \mu B\boldsymbol{c} \geq 0$

Instantiate Finally, instantiating some parameter μ to $\{0, 1\}$ lets us resolve it. Other values of μ are also possible. However, using $\{0, 1\}$ restricts the template assertion $\alpha \geq 0$ to satisfy local or increasing-value consecution, respectively, as defined in Section 2. The advantage of this strategy is that it is efficient and simple, especially if some invariants are to be generated in as short a time as possible.

4 Experimental Results

We have implemented our method and evaluated it on several programs. Our prototype implementation uses the library PPL for manipulating polyhedra [2] supplemented with our own implementation of some of the rules in Figure 2, discussed below. We compared our method against forward propagation with two different widenings provided by PPL: the standard CH79 widening [8] and BHRZ03 widening [1]. The BHRZ03 operator is provably more accurate, but less efficient than the CH79 widening. Since we implemented the post-condition ourselves, we present separately the time spent computing post-conditions and the time spent on PPL-provided widening.

Experimenting with a few strategies, we converged on a strategy that scaled to larger examples. Some of the salient features of the strategy are the following:

- For multi-location systems, the transitions are classified as *intra*-location and *inter*-location. The constraints for the intra-location transitions at each location are resolved by a subset of the rules described previously. Specifically, factorization is performed only over equalities, and Lemmas 2 and 3 are not used. Handling factors over inequalities requires more polyhedral reasoning at every simplification while the use of the two lemmas requires sophisticated reasoning involving equalites, inequalities and disequalities. Our disequality constraint solver uses heuristic rules whose completeness remains unresolved.
- Local and increasing consecution are used for each inter-location transition. This strategy can be proven exact for many situations.
- The constraints for each location and the inter-location transitions are combined conjunctively. Converting this CNF expression to DNF is a significant bottleneck, requiring aggressive subsumption tests.
- Constraints are solved *depth-first* as much as possible, favouring branches that can be resolved faster. The collection of linear constraints from resolved branches enable aggressive subsumption testing. The CNF to DNF conversion is also performed in a depth-first fashion to enable invariants to be computed eagerly.

As an added benefit, the execution can be interrupted after a reasonable amount of time, and still yield many non-trivial invariants. In several cases our invariants are disjoint with the results of forward propagation, because propagation-based techniques can compute invariants $\varphi_1 \wedge \varphi_2$ that are *mutually inductive*, that is, neither φ_1 nor φ_2 are inductive by themselves, while our technique only discovers single inequalities that are inductive by themselves. However, repeating the procedure with the computed invariants added to the guards of the transition system usually provides the stronger invariants.

4.1 Low Dimensional Systems

Figure 3 shows the experimental results for some small to medium sized examples from the related work and some benchmarks from analysis tools such as FAST [3]. The number of variables for each program is shown in the second column. The table shows for each program the time (in seconds) of our (constraint-based) approach, and the time taken by the CH79 and BHRZ03 approach. All computation times were measured on an Intel Xeon 1.7 GHz CPU with 2 Gb RAM. The last two columns show the strength of the invariants computed by our method compared with those computed by CH79 and BHRZ03, respectively. A + indicates that our invariants are strictly stronger, or no invariants were obtained by the other method within 1 hour. The $+^3 =$ indicates that our invariants were stronger for three locations, while they were the same for the other locations. An $=$, $\neq$ and $-$ indicate that our invariants are equal, incomparable, and strictly weaker, respectively. The suffix N indicates that all the variables in the system were constrained to be positive to increase the number of invariants discovered in one run. The programs SWIM1, EFM1 were obtained by adding the previously computed invariants as guards to the transition relations.

Program		Constraint-based method			CH79 time (secs)			BHRZ03 time (secs)			C-B Invariants versus	
Name	vars	time	# br	# sub	total	post	widen	total	post	widen	CH79	BHRZ03
SEE-SAW	2	0.03	13	8	0	0	0	0	0	0	+	+
ROBOT	3	0.02	2	1	0.01	0	0.01	0.01	0.01	0	=	=
TRAIN-HPR97	3	0.86	25	5	0.02	0.02	0	0.02	0.02	0	$+^3 =$	$+^3 =$
RAND	4	0.02	3	2	0.01	0	0	0	0	0	=	=
BERKELEY	4	0.06	11	8	0.01	0	0.01	0.01	0	0.01	≠	≠
BERKELEY-N	4	0.04	9	4	0.01	0.01	0	0.01	0	0.01	+	+
HEAPSORT	5	0.1	21	12	0.02	0.01	0.01	0.02	0.02	0	≠	≠
TRAIN-RM03	6	1.16	193	99	0.06	0.05	0.01	0.07	0.05	0.02	+	=
EFM	6	0.36	57	23	0	0	0	0.01	0.01	0	−	−
EFM1	6	0.32	57	27	0	0	0	0.01	0.01	0	−	−
LIFO	7	0.88	58	51	0.29	0.27	0.02	0.32	0.29	0.03	≠	≠
LIFO-N	7	10.13	1191	593	0.27	0.25	0.02	0.32	0.27	0.04	+	+
CARS-MIDPT	7	0.1	17	8	32.8	5	27.8	> 3600			+	+
BARBER	8	1.68	125	84	0.18	0.17	0.01	20.41	0.18	20.23	+	+
SWIM	9	0.42	36	22	0.08	0.06	0.02	0.61	0.06	0.55	−	−
SWIM1	9	0.88	65	32	0.07	0.06	0.01	0.59	0.06	0.53	=	=

Fig. 3. Experimental results for some low-dimensional systems. #br is the number of branches, #sub is the number pruned by subsumption tests .

The figure shows that for the programs tested our invariants are mostly superior or comparable, but at a significant extra cost in computation time for the smaller dimensions. However, the situation changes when the dimensionality of the systems is increased beyond ten variables, as shown in the next section.

4.2 Higher-dimensional Systems

To evaluate our method for systems with more variables we compared its performance on instances of two parameterized systems.

Pre-emptive Scheduler: The first system is an n process pre-emptive scheduler inspired by the two-process example in Halbwachs et. al. [11]. Two arrivals of process p_i are separated by at least c_i time units, for a fixed c_i. Process p_i preempts process p_j for $j < i$. The system has n locations, where location ℓ_i denotes that process p_i is executing and that there are no waiting processes p_j for $j > i$.

Convoy of Cars: The second system consists of n cars on a straight road whose accelerations are controlled (as in real life), determining their velocity. The lead car non-deterministically chooses an acceleration. The controller for each car detects when the lead car is too close or too far, and, after a bounded reaction

Program		Constraint-based method			CH79 time (secs)			BHRZ03 time (secs)			C-B Invariants versus	
Name	vars	time	# br	# sub	total	post	widen	total	post	widen	CH79	BHRZ03
SCHEDULER												
2 proc.	7	0.54	23	10	0.15	0.12	0.02	0.19	0.12	0.07	$\neq -$	$\neq -$
3 proc.	10	8.21	36	16	39.5	26.9	12.8	2232	27.5	2204	$+^3$	$\neq^3$
4 proc.	13	284	55	26	> 3600			> 3600			$+^4$	$+^4$
5 proc.	16	> 3600	81	41	> 3600			> 3600			?	?
CARS												
2 proc.	10	3.54	93	44	5.22	4.23	0.97	443	200	243	+	$\neq$
3 proc.	14	20.5	468	239	> 3600			> 3600			+	+
4 proc.	18	1006	3722	1897	> 3600			> 3600			+	+

Fig. 4. Performance Comparison on parameterized examples SCHEDULER and CARS

time, adjusts acceleration. Time was discretized in order to linearize the resulting transition system.

Figure 4 shows the performance comparison In all cases above 10 variables, our technique out-performs the other two techniques. The propagation-based techniques ran out of time for these systems. Our method ran out of time for the 5-process scheduler. It did so while converting a large CNF formula into a DNF formula. In fact, two different timeouts, 700s and 3600s, yielded the same (non-trivial) invariants. A total of 19 disjuncts in the normal form conversion were found to be relevant within the first 700 seconds, while all the 75791 disjuncts computed in the next 2900 seconds were found to be subsumed by the original 19. This suggests that a vast majority of disjuncts in the computed DNF form yield the same invariant, which was confirmed by other examples.

5 Conclusion

Linear programming, as a discipline has seen tremendous advances in the past century. Our research demonstrates that some ideas from linear programming can be used to provide alternative techniques for linear-relations analysis. Analysis carried out this way has some powerful advantages. It provides the ability to adjust the complexity and the accuracy in numerous ways. The constraint-based perspective for linear relations analysis can be powerful, both in theory and in practice.

Future work needs to concentrate on increasing the dimensionality and the complexity of the application examples for this analysis. Numerous mathematical tools remain to be explored for this domain. The use of numerical, and interval-numerical techniques for handling robustness in polyhedral computations has remained largely unexplored. Manipulation techniques for compressed representations along the lines of Halbwachs et al. [10] has also shown promise. Further investigations into the geometry of these constraints will yield a precise and faster analysis.

Acknowledgements: The authors are grateful to the people behind PPL [2] for making this study possible, and the anonymous reviewers for their detailed comments.

References

1. Bagnara, R., Hill, P. M., Ricci, E., and Zaffanella, E. Precise widening operators for convex polyhedra. In *Static Analysis Symposium* (2003), vol. 2694 of *LNCS*, Springer-Verlag, pp. 337–354.
2. Bagnara, R., Ricci, E., Zaffanella, E., and Hill, P. M. Possibly not closed convex polyhedra and the Parma Polyhedra Library. In *Static Analysis Symposium* (2002), vol. 2477 of *LNCS*, Springer-Verlag, pp. 213–229.
3. Bardin, S., Finkel, A., Leroux, J., and Petrucci, L. Fast: Fast accelereation of symbolic transition systems. In *Computer-aided Verification* (July 2003), vol. 2725 of *LNCS*, Springer-Verlag.
4. Besson, F., Jensen, T., and Talpin, J.-P. Polyhedral analysis of synchronous languages. In *Static Analysis Symposium* (1999), vol. 1694 of *LNCS*, pp. 51–69.
5. Collins, G. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Automata Theory and Formal Languages* (1975), H.Brakhage, Ed., vol. 33 of *LNCS*, Springer-Verlag, pp. 134–183.
6. Colón, M., Sankaranarayanan, S., and Sipma, H. Linear invariant generation using non-linear constraint solving. In *Computer Aided Verification* (July 2003), vol. 2725 of *LNCS*, Springer-Verlag, pp. 420–433.
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. Fukuda, K., and Prodon, A. Double description method revisited. In *Combinatorics and Computer Science*, vol. 1120 of *LNCS*. Springer-Verlag, 1996, pp. 91–111.
10. Halbwachs, N., Merchat, D., and Parent-Vigouroux, C. Cartesian factoring of polyhedra for linear relation analysis. In *Static Analysis Symposium* (2003), vol. 2694 of *LNCS*, Springer-Verlag, pp. 355–365.
11. Halbwachs, N., Proy, Y., and Roumanoff, P. Verification of real-time systems using linear relation analysis. *Formal Methods in System Design 11*, 2 (1997), 157–185.
12. Henzinger, T. A., and Ho, P. HyTech: The Cornell hybrid technology tool. In *Hybrid Systems II* (1995), vol. 999 of *LNCS*, Springer-Verlag, pp. 265–293.
13. Manna, Z., and Pnueli, A. *Temporal Verification of Reactive Systems: Safety.* Springer-Verlag, New York, 1995.
14. Sankaranarayanan, S., Sipma, H. B., and Manna, Z. Petri net analysis using invariant generation. In *Verification: Theory and Practice* (2003), vol. 2772 of *LNCS*, Springer-Verlag, pp. 682–701.
15. Schrijver, A. *Theory of Linear and Integer Programming.* Wiley, 1986.
16. Tarski, A. A decision method for elementary algebra and geometry. *Univ. of California Press, Berkeley 5* (1951).
17. Weispfenning, V. The complexity of linear problems in fields. *Journal of Symbolic Computation 5*, 1-2 (April 1988), 3–27.