Simple and Precise Widenings for *H*-Polyhedra

Axel Simon * and Liqian Chen **

* Informatik 2, Technische Universität München, 85748 Garching, Germany
** National Laboratory for Parallel and Distributed Processing, Changsha, China

Abstract. While the definition of the revised widening for polyhedra is defined in terms of inequalities, most implementations use the double description method as a means to an efficient implementation. We show how standard widening can be implemented in a simple and efficient way using a normalized H-representation (constraint-only) which has become popular in recent approximations to polyhedral analysis. We then detail a novel heuristic for this representation that is tuned to capture linear transformations of the state space while ensuring quick convergence for non-linear transformations for which no precise linear invariants exist.

1 Introduction

The lattice of convex polyhedra is a popular abstract domain [7] for inferring linear relations between variables. Implementations of the full domain [1] and its many approximations [14, 17, 21] were used to infer program properties (such as list sizes [16] and variable ranges) or to analyze models (such as hybrid automata [12]). The infinite ascending chains in the lattice of polyhedra require that the inferred states are widened to ensure that the fixpoint computation terminates. To this end, a widening operator extrapolates the changes between two (or more) iterates of a fixpoint computation to a state that likely includes all future iterates. Standard widening for polyhedra [10] is defined in terms of the so-called H-representation of polyhedra (a set of linear constraints), and its straightforward implementation requires a quadratic number of entailment checks [5]. In this paper, we introduce *normal widening* that implements standard widening using only simple syntactic checks, based on a normalized H-representation. It avoids the creation of redundant inequalities which, as we show, is inherent in the original definition [10]. These redundancies are avoided in classic polyhedra libraries that store a double description of a polyhedron consisting of constraints (equalities and inequalities) and generators (vertices, rays, and lines). Thus, our algorithm benefits novel implementations that only use constraints [5, 18, 19].

Since standard widening is often too imprecise, various heuristics have been proposed to improve the prediction of the state space growth. Most of these heuristics rely on the double description of polyhedra which makes them ill-suited to implementations using H-polyhedra. We therefore seek new heuristics that only require the H-representation and take advantage of the normal form.

* The work was supported by the DFG Emmy Noether Programme SI 1579/1.
** NSFC 60725206 and INRIA project "Abstraction" common to CNRS and ENS.

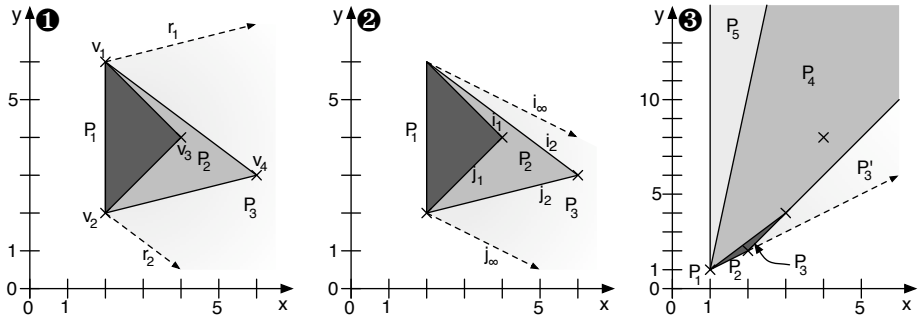

Fig. 1. Comparing the evolving-points heuristic with our method.

To this end, consider the first diagram of Fig. 1 which shows how the "evolving point" heuristic [1] widens the polyhedron P_1 with respect to the next iterate P_2: The vertices v_1, v_2, v_3 in P_1 are subtracted from vertices that are new in P_2 (here only v_4) and the resulting extreme rays $r_1 = v_4 - v_2$ and $r_2 = v_4 - v_1$ are added to P_2 thereby defining the resulting polyhedron P_3. The second diagram illustrates our heuristic that rotates inequalities i_1 in P_1 and i_2 in P_2 to i_∞ which has the same distant to the evolving vertices v_3 and v_4. Performing the analog rotation for j_1, j_2 to j_∞ yields the state P_3. This result is indeed an invariant for loops that contain the statement `if (y==5) {x=x+2;y=y-1;}`. Extrapolating with the assumption that transformations are linear means that our heuristic can widen rapidly when a non-linear transformation is observed. This is illustrated in the third diagram where the polyhedra P_n show the evaluation of the loop body `x=x+1;y=2*y;` with $P_1 = \{\langle 1, 1\rangle\}$. The exact values $\langle x, y\rangle$ are indicated by crosses. The Parma Polyhedra Library [1] delays widening[1] until both polyhedra have the same dimension. Then P_3 is widened with some heuristic to give P_4 which has to be widened to P_5 which is observed to be a fixpoint in the 6th iteration. In contrast, our heuristic is able to widen P_2, yielding $P'_3 = \{x \leq y, x \geq 1\}$ which is confirmed as a fixpoint in the 4th iteration. Thus, in case of non-linear transformations, polyhedra are unable to express a precise invariant (e.g. $2^x = y$) so that performing two additional iterations is likely to be a waste of time. In summary, this paper makes the following contributions:

- It presents an implementation of standard widening for H-polyhedra that requires only syntactic operations rather than expensive entailment checks.
- We present a heuristic that requires only the H-representation and which tries to guess linear transformations based on the observed changes.
- We show that our heuristic often terminates faster than classic heuristics.

The remainder of the paper is organized as follows. The next section introduces required notation. Section 3 presents the well-known standard widening and our implementation for normalized polyhedra. Section 4 details our novel heuristic which Sect. 5 evaluates. Section 6 presents related work and concludes.

[1] when given one token; without any tokens, the PPL library discards all inequalities.

2 Preliminaries

Let $\boldsymbol{x} = \langle x_1, \ldots x_n \rangle$ denote an ordered set of variables, let Ineq_n denote the set of linear inequalities $\boldsymbol{a} \cdot \boldsymbol{x} \leq c$ where $\boldsymbol{a} \in \mathbb{R}^n$ and $c \in \mathbb{R}$. Moreover, let e.g. $6x_3 \geq x_1 - 5$ abbreviate $\langle 1, 0, -6, 0, \ldots 0 \rangle \cdot \boldsymbol{x} \leq 5$. Define Eq_n to denote the set of equalities of the form $\boldsymbol{a} \cdot \boldsymbol{x} = c \in \mathit{Eq}_n$. Given an equality set $E \subseteq \mathit{Eq}_n$, we use $E^{\leq} := \{\boldsymbol{a} \cdot \boldsymbol{x} \leq c, \boldsymbol{a} \cdot \boldsymbol{x} \geq c \mid \boldsymbol{a} \cdot \boldsymbol{x} = c \in E\}$ to denote the corresponding set of inequalities. Each inequality $\boldsymbol{a} \cdot \boldsymbol{x} \leq c \in \mathit{Ineq}_n$ induces a half-space $[\![\boldsymbol{a} \cdot \boldsymbol{x} \leq c]\!] = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{a} \cdot \boldsymbol{x} \leq c\}$. Each finite set of inequalities $I = \{\iota_1, \ldots \iota_m\} \subseteq \mathit{Ineq}_n$ induces a closed, convex polyhedron $[\![I]\!] = \bigcap_{i=1}^{m} [\![\iota_i]\!]$. Let $\mathit{Poly}_n = \{[\![I]\!] \mid I \subseteq \mathit{Ineq}_n, |I| \in \mathbb{N}\}$ denote the set of all (finitely generated) polyhedra. The tuple $\langle \mathit{Poly}_n, \subseteq, \cap, \overline{\curlyvee} \rangle$ is a lattice with $P_1 \,\overline{\curlyvee}\, P_2 = \mathit{cl}(\mathit{hull}(P_1 \cup P_2))$ where cl denotes topological closure and *hull* is the convex hull operation on sets of points. This lattice can serve as abstract domain in a program analysis where a bifurcation with condition $c \in \mathit{Ineq}_n$ in the control flow graph is modeled using the *meet* operation $P \cap [\![c]\!]$ and a merge of control flow edges is modeled by the *join* $P_1 \,\overline{\curlyvee}\, P_2$ [7]. However, the lattice contains infinite chains $P_1 \subset P_2 \subset P_3 \ldots$ so that standard Kleene iteration may not converge onto a fixpoint in finite time. To guarantee convergence, a widening operator $\nabla : \mathit{Poly}_n \times \mathit{Poly}_n \to \mathit{Poly}_n$ is required, satisfying the following:

1. $\forall x, y \in \mathit{Poly}_n \,.\, x \subseteq x \nabla y$
2. $\forall x, y \in \mathit{Poly}_n \,.\, y \subseteq x \nabla y$
3. for all increasing chains $x_0 \subseteq x_1 \subseteq \ldots$, the increasing chain defined by $y_0 = x_0$ and $y_{i+1} = y_i \nabla x_{i+1}$ is ultimately stable.

All operations can be implemented by storing a set of constraints (equalities and inequalities) for each polyhedron: Suppose that $P_i = [\![I_i]\!]$, $i = 1, 2$, then $P_1 \cap P_2 = [\![I_1 \cup I_2]\!]$ and $P_1 \subseteq P_2$ iff for all $\boldsymbol{a} \cdot \boldsymbol{x} \leq c \in I_2$ it holds that $c \geq \mathit{max}(\boldsymbol{a} \cdot \boldsymbol{x}, I_1)$ where $\mathit{max} : \mathit{Lin}_n \times \mathcal{P}(\mathit{Ineq}_n) \to (\mathbb{Q} \cup \{\infty\})$ infers the maximum that $\boldsymbol{a} \cdot \boldsymbol{x}$ can take on in $[\![I_1]\!]$. Note that $\mathit{max}(\boldsymbol{a} \cdot \boldsymbol{x}, I)$ can be inferred using the Simplex algorithm for linear programming. Internally, this algorithm searches for a positive linear combination $\boldsymbol{\lambda} \in \mathbb{Q}^k$ of $k \leq n$ inequalities $\{\boldsymbol{a}_1 \cdot \boldsymbol{x} \leq c_1, \ldots \boldsymbol{a}_k \cdot \boldsymbol{x} \leq c_k\} \subseteq I$ such that $\boldsymbol{\lambda} \cdot (\boldsymbol{a}_1 \ldots \boldsymbol{a}_k) = \boldsymbol{a}$ and $c = \boldsymbol{\lambda} \cdot (c_1 \ldots c_k)$ maximizes $\boldsymbol{a} \cdot \boldsymbol{x}$ in $[\![I]\!]$. We use $\langle \boldsymbol{\lambda}, c \rangle = \mathit{maxExt}(\boldsymbol{a} \cdot \boldsymbol{x}, I)$ to calculate $\boldsymbol{\lambda}$. Linear programming has also been used to approximate the calculation of $P_1 \,\overline{\curlyvee}\, P_2$ [18, 19]. In this context, it is mainly used to remove redundant inequalities. An inequality $\iota \in I$ is redundant in I if $[\![I \setminus \iota]\!] \subseteq [\![I]\!]$. Throughout this paper, we assume that the representation I of a polyhedron $P = [\![I]\!]$ contains no redundant inequalities ("I is non-redundant").

The advent of approximate join operators that are solely based on constraints [18, 19] raises the question how other operations, e.g., the widening, can be implemented in a constraint-only representation. The following section presents a simple constraint-only based implementation of the standard widening operator before Sect. 4 addresses the question of additional heuristics that, so far, have only been implemented using the double description method which is a representation that uses both, constraints (equalities and inequalities) and generators (vertices, rays, and lines).

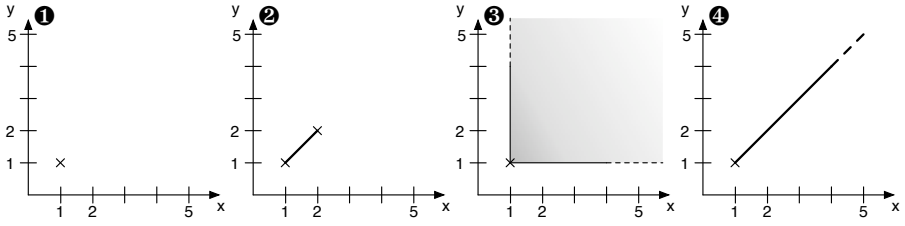

Fig. 2. Illustrating standard widening.

3 Widening for Polyhedra

The presence of infinite ascending chains requires a widening to accelerate and guarantee convergence of the fixpoint computation. The original widening operator on polyhedra proposed in [7] was intuitive in that all inequalities that were new are abandoned. Its improvement, presented in [10], refines this idea by making this process resilient to different representations of the same state space. The latter algorithm has specialized implementations based on the double description method and has become known as the *standard widening*. We restate the standard widening based on the constraint-only representation and present an efficient and simple implementation that does not need generators.

Definition 1 (Standard widening). *Given two polyhedra* $P_1 = [\![J_1]\!]$ *and* $P_2 = [\![J_2]\!]$ *satisfying* $P_1 \subseteq P_2$ *where* $J_1, J_2 \subseteq Ineq_n$ *are non-redundant, we define*

$$P_1 \triangledown P_2 \stackrel{\text{def}}{=} [\![\mathcal{C}_1 \cup \mathcal{C}_2]\!]$$

where

$$\mathcal{C}_1 = \{\, \iota_1 \in J_1 \mid [\![J_2]\!] \subseteq [\![\iota_1]\!] \,\},$$
$$\mathcal{C}_2 = \{\, \iota_2 \in J_2 \setminus J_1 \mid \exists\, \iota_1 \in J_1, [\![(J_1 \setminus \iota_1) \cup \iota_2]\!] = [\![J_1]\!] \,\}.$$

The first set $\mathcal{C}_1$ contains all inequalities of P_1 that are not violated by the larger P_2 and corresponds to the original widening. Standard widening adds $\mathcal{C}_2$, consisting of inequalities of J_2 that can be exchanged with an inequality of J_1 without changing the represented state. This set ensures that the result is independent of the representation of P_1 and P_2. In order to illustrate this, consider Fig. 2. The first two diagrams depict the two inputs to the widening operator. Suppose that the first polyhedron P_1 is represented by $I_1 = \{1 \leq x, x \leq 1, 1 \leq y, y \leq 1\}$ whereas P_2 is represented by $I_2 = \{1 \leq x, y \leq x, x \leq y, x \leq 2\}$. Only $\mathcal{C}_1 = \{1 \leq x, 1 \leq y\}$ is satisfied by P_2, thus the original widening returns the state depicted in the third diagram. Standard widening adds $\mathcal{C}_2 = \{y \leq x, x \leq y\}$ since these can be exchanged with $x \leq 1, y \leq 1 \in J_1$ without changing $[\![J_1]\!]$. However, each of the $|J_1||J_2|$ entailment checks [5] requires a Simplex query. Thus, we propose to store a polyhedron in a normal form and show how this can refine the problem of making widening resilient to the representation of polyhedra.

Listing 1 Inlining equalities into an inequality.

procedure *inline*$(E, \boldsymbol{a} \cdot \boldsymbol{x} \leq c)$ where $E \subseteq Eq_n, \boldsymbol{a} \cdot \boldsymbol{x} \leq c \in Ineq_n, \boldsymbol{a} \equiv \langle a_1, \ldots a_n \rangle$

1: **for** $\boldsymbol{a}^{=} \cdot \boldsymbol{x} = c^{=} \in E$ where $\boldsymbol{a}^{=} \equiv \langle a_1^{=}, \ldots a_n^{=} \rangle$ **do**
2: $\quad i \leftarrow \min\{i \mid a_i^{=} \neq 0\}$ /* *find the index of the first non-zero coefficient* */
3: $\quad$ **if** $a_i = 0$ **then continue;** /* *skip loop if inequality does not contain* x_i */
4: $\quad \boldsymbol{a} \cdot \boldsymbol{x} \leq c \leftarrow (a_i^{=}\boldsymbol{a} - a_i\boldsymbol{a}^{=}) \cdot \boldsymbol{x} \leq (a_i^{=}c - a_i c^{=})$
5: **end for**
6: **return** $\langle \boldsymbol{a} \cdot \boldsymbol{x} \leq c \rangle$

3.1 Normalizing the Constraint Representation

Given $J \subseteq Ineq_n$, we construct a canonical representation for $P = [\![J]\!]$ as follows:

1. We compute the affine space that P lies in by calculating $c' = -max(-\boldsymbol{a} \cdot \boldsymbol{x}, J)$ for each $\boldsymbol{a} \cdot \boldsymbol{x} \leq c \in J$. If $c' = c$ then $\boldsymbol{a} \cdot \boldsymbol{x} = c$ holds in P. By performing Gaussian elimination on these equalities, we obtain an equality system in reduced row echelon form, which we denote as E.
2. For each $\iota \in J$, we apply a function *inline*(E, ι), presented as Listing 1, that eliminates those variables in ι that are leading variables in E.
3. Finally, we normalize each inequality such that the leading coefficient is either 1 or -1 and remove constraints that are redundant. We get a new set of inequalities, which we denote as I.

The above steps calculate a canonical representation $\langle E, I \rangle$ of any $P \in Poly_n$, see e.g. [15]. For the remainder of the paper, we assume that polyhedra are represented as normalized sets of equalities and inequalities.

3.2 Standard Widening on Normalized Constraints

Given that the input constraints $[\![E_i, I_i]\!] = P_i$, $i = 1, 2$ of the widening operator are normalized, the pre-condition $P_1 \subseteq P_2$ implies that $[\![E_1]\!] \subseteq [\![E_2]\!]$. Since the affine space common to E_1 and E_2 is by definition stable, it suffices to widen the two systems $E^{\leq} \cup I_1$ and I_2 where $E \subseteq Eq_n$ describes the affine space of P_1 without that of P_2. For example, if $E_1 = \{x = 0, y = 0\}$ and $E_2 = \{x = y\}$ then $E_C = \{x = y\}$ is the common affine space and $E = \{y = 0\}$. In general, after normalization and omission of E_C, the constraint sets E_1, I_1, and I_2 can be written as follows:

$$\begin{array}{ll} E_1\colon & \mathbb{I}\boldsymbol{v} + A_E\boldsymbol{w} = \boldsymbol{c}_E \\ I_1\colon & A_1\boldsymbol{w} \leq \boldsymbol{c}_1 \\ I_2\colon & A_2^v\boldsymbol{v} + A_2^w\boldsymbol{w} \leq \boldsymbol{c}_2 \end{array}$$

Here, $\mathbb{I}$ denotes the identity matrix and $\langle \boldsymbol{v}, \boldsymbol{w} \rangle$ span the variable set $\boldsymbol{x}$. Restricting the input constraints to the systems above not only improves efficiency of the widening operator but also allows its implementation to be much simpler than in the definition of standard widening presented in Sect. 3. Before we show this, we observe that an inequality can be modified by inlining equalities without changing the polyhedron. The following refers to the *inline* function in Listing 1.

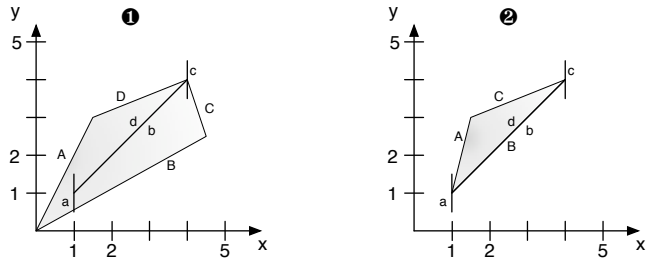

Fig. 3. Non-trivial applications of standard widening. Each letter is written on the infeasible side of the facet it denotes (i.e. the outside of the polyhedron). Lower case letters denote the constraints of the first polyhedron, capital letters those of the second.

Lemma 1. *Let* $\iota' = \mathit{inline}(E, \iota)$ *then* $[\![I \cup \{\iota\} \cup E^{\leq}]\!] = [\![I \cup \{\iota'\} \cup E^{\leq}]\!]$.

Proof. In Listing 1, at line 4, we have $a_i^{=} x_i + \Sigma_{k>i} a_k^{=} x_k = c^{=}$, which is equivalent to $x_i = (c^{=} - \Sigma_{k>i} a_k^{=} x_k)/a_i^{=}$. Thus $\boldsymbol{a} \cdot \boldsymbol{x} \leq c$, that is, $a_i x_i + \Sigma_{k \neq i} a_k x_k \leq c$, is equivalent to $a_i(c^{=} - \Sigma_{k>i} a_k^{=} x_k)/a_i^{=} + \Sigma_{k \neq i} a_k x_k \leq c$. Hence, $\boldsymbol{a} \cdot \boldsymbol{x} \leq c$ is equivalent to $(a_i^{=} \boldsymbol{a} - a_i \boldsymbol{a}^{=}) \cdot \boldsymbol{x} \leq (a_i^{=} c - a_i c^{=})$ w.r.t. the affine space E. □

We now show how standard widening can be implemented using the normal representation. To this end, suppose that the two inputs to the widening are $J_1 = E_1^{\leq} \cup I_1$ and $J_2 = I_2$ with I_1, E_1, I_2 in normal form as described above.

First, we calculate $\mathcal{C}_2$ (defined in Definition 1) syntactically and call this set I_C^S. The principle is depicted on the left of Fig. 3. Here the second polyhedron $I_2 = \{A, B, C, D\}$ is widened with respect to $I_1 = \{a, c\}, E_1^{\leq} = \{b, d\}$. The resulting set according to Definition 1 is $C_1 = \emptyset$ and $C_2 = \{C, D\}$ since $[\![I_1 \setminus \{c\} \cup \{C\}]\!] = [\![I_1]\!]$ and $[\![I_1 \setminus \{c\} \cup \{D\}]\!] = [\![I_1]\!]$. The set $I_C^S \subseteq I_2$ is defined to be all $\iota \in I_2$ such that $\mathit{inline}(E, \iota)$ is in I_1 or a tautology:

Lemma 2. *Given* $J_1 = E_1^{\leq} \cup I_1$ *and* $J_2 = I_2$ *with* E_1 *in reduced row echelon form,* $I_1 = \{\mathit{inline}(E_1, \iota) \mid \iota \in I_1\}$ *and* I_1, I_2 *normalized, then* $\mathcal{C}_2 = I_C^S$ *where*

$$I_C^S = \{\iota_2 \in J_2 \setminus J_1 \mid \mathit{inline}(E_1, \iota_2) \in I_1 \vee \mathit{inline}(E_1, \iota_2) \equiv \boldsymbol{0} \cdot \boldsymbol{x} \leq 0\}.$$

Proof. For any $\iota_2 \in J_2 \setminus J_1$ we show that $\exists\, \iota_1 \in J_1, [\![(J_1 \setminus \iota_1) \cup \{\iota_2\}]\!] = [\![J_1]\!]$ if and only if $\mathit{inline}(E_1, \iota_2) \in I_1 \vee \mathit{inline}(E_1, \iota_2) \equiv \boldsymbol{0} \cdot \boldsymbol{x} \leq 0$. We specialize this property into two separate cases depending on whether $\iota_1 \in E_1^{\leq}$ or $\iota_1 \in I_1$:

$\iota_1 \in E_1^{\leq}$**:** Show that $[\![I_1 \cup (E_1^{\leq} \setminus \iota_1) \cup \{\iota_2\}]\!] = [\![J_1]\!]$ iff $\mathit{inline}(E_1, \iota_2) = \boldsymbol{0} \cdot \boldsymbol{x} \leq 0$: Since $\iota_1 \in E_1^{\leq}$, we have $[\![I_1 \cup (E_1^{\leq} \setminus \iota_1) \cup \{\iota_2\}]\!] = [\![J_1]\!]$ iff $[\![(E_1^{\leq} \setminus \iota_1) \cup \{\iota_2\}]\!] = [\![E_1^{\leq}]\!]$. We will show next that $[\![(E_1^{\leq} \setminus \iota_1) \cup \{\iota_2\}]\!] = [\![E_1^{\leq}]\!]$ iff $\mathit{inline}(E_1, \iota_2) \equiv \boldsymbol{0} \cdot \boldsymbol{x} \leq 0$. Given any ι_2 with $\mathit{inline}(E_1, \iota_2) \equiv \boldsymbol{0} \cdot \boldsymbol{x} \leq 0$ then $\iota_2 + \lambda_1 e_1 + \ldots \lambda_k e_k \equiv \boldsymbol{0} \cdot \boldsymbol{x} \leq 0$ for some $\lambda_1, \ldots \lambda_k > 0$ where $\{e_1, \ldots e_k\} \in E_1^{\leq}$ but $\bar{e}_i \notin \{e_1, \ldots e_k\}$ for all $i \in [1, k]$ where $\bar{e}_i$ denotes the inequality in $E_1^{\leq}$ that opposes e_i. In other words, $\iota_2 = \lambda_1 \bar{e}_1 + \ldots \lambda_k \bar{e}_k$. On the other hand, choose $i \in [1, k]$ such that $\iota_1 = \bar{e}_i$. Then $[\![\iota_1]\!] = [\![\lambda_i \bar{e}_i]\!] = [\![\iota_2 + \lambda_1 e_1 + \ldots \lambda_{i-1} e_{i-1} + \lambda_{i+1} e_{i+1} + \ldots \lambda_k e_k]\!]$. It follows that $[\![(E_1^{\leq} \setminus \iota_1) \cup \{\iota_2\}]\!] = [\![E_1]\!]$.

$\iota_1 \in I_1$**:** Show that $[\![(I_1 \setminus \iota_1) \cup E_1^{\leq} \cup \{\iota_2\}]\!] = [\![J_1]\!]$ iff $\mathit{inline}(E_1, \iota_2) = \iota_1$: Again, we partition $\boldsymbol{x}$ into $\boldsymbol{v}, \boldsymbol{w}$ as described previously. Then let $\iota_2 \equiv \boldsymbol{a}^v \boldsymbol{v} + \boldsymbol{a}^w \boldsymbol{w} \leq c$. Suppose that $\boldsymbol{a}^v = 0$ and thus $\mathit{inline}(E_1, \iota_2) = \iota_2$. Due to normalization $[\![\iota_2]\!] = [\![\iota_1]\!]$ iff $\iota_2 = \iota_1$ and thus $[\![(I_1 \setminus \iota_1) \cup E_1^{\leq} \cup \{\iota\}]\!] = [\![J_1]\!]$ holds iff $\iota \equiv \iota_1$. Now suppose that $\boldsymbol{a}^v \neq 0$ and that $\mathit{inline}(E_1, \iota) = \iota'$ where $\iota' \equiv \boldsymbol{a'}^v \boldsymbol{v} + \boldsymbol{a'}^w \boldsymbol{w} \leq c'$. Then $\boldsymbol{a'}^v = 0$. Using Lemma 1, we obtain the equivalent $[\![(I_1 \setminus \iota_1) \cup E_1^{\leq} \cup \{\iota'\}]\!] = [\![J_1]\!]$ and, since $\boldsymbol{a'}^v = 0$ this is equivalent to $[\![(I_1 \setminus \iota_1) \cup \{\iota'\}]\!] = [\![I_1]\!]$ which, due to normalization, holds iff $\iota' \equiv \iota_1$. □

As far as we know, it is not widely known that the definition of standard widening generates redundant constraints although it has been shown that these have to be removed for the correctness of future widening applications [1]. The second diagram of Fig. 3 presents an example where redundancies are produced. Here, the set $I_2 = \{A, B, C\}$ is widened with respect to $I_1 = \{a, c\}, E_1^{\leq} = \{b, d\}$. Since $\mathcal{C}_1 = \{a, b, c\}$ and $\mathcal{C}_2 = \{A, C\}$, the inequalities a and c are redundant in $\mathcal{C}_1 \cup \mathcal{C}_2$. We will now show that the common constraints $I_C = (I_1 \cup E_1^{\leq}) \cap I_2$ corresponds to $\mathcal{C}_1$ without such redundant inequalities, in other words, we show that $[\![I_C \cup I_C^S]\!] = [\![\mathcal{C}_1 \cup \mathcal{C}_2]\!]$. We first characterize every inequality $\iota \in C_1 \setminus I_C$ with respect to I_C^S:

Lemma 3. *For any $\iota \in \mathcal{C}_1 \setminus I_C$, $[\![I_C \cup I_C^S]\!] \subseteq [\![\iota]\!]$.*

Proof. By definition of $\mathcal{C}_1$ and the fact that $[\![J_1]\!] \subseteq [\![J_2]\!]$, there exist $\lambda_j > 0$ with $\iota = \lambda_1 \iota_1 + \ldots \lambda_n \iota_n$ where $\iota_j \in I_2$, $j = 1, \ldots n$. Note that $I_C^S \subseteq J_2 \setminus J_1 = I_2 \setminus (I_1 \cup E_1^{\leq})$ and hence $I_C^S \cap I_C = \emptyset$. Hence, without loss of generality, let $\iota_1, \ldots \iota_k \in I_2 \setminus I_C$ and $\iota_{k+1}, \ldots \iota_n \in I_C$. We show that $\iota_i \in I_C^S$ for all $i \leq k$. Let $\iota \equiv \boldsymbol{a} \cdot \boldsymbol{x} \leq c$. Consider $\iota_i' = \mathit{inline}(E_1, \iota_i)$. Note that ι_i is not only entailed by J_1 but also touches J_1 and so does ι_i'. Hence, if $\iota_i' \equiv \boldsymbol{0} \cdot \boldsymbol{x} \leq c'$ for some $c' \in \mathbb{R}$ then $c' = 0$ and $\iota_i' \in I_C^S$. Now assume $\iota_i' \equiv \boldsymbol{a'} \cdot \boldsymbol{x} \leq c'$ with $\boldsymbol{a'} \neq \boldsymbol{0}$ but $\boldsymbol{a'} \neq \boldsymbol{a}$. Observe that $\mathit{inline}(E_1, \iota) = \iota$ can be defined as a positive linear combination of some $\mathit{inline}(E_1, \iota_j)$, $j = 1, \ldots n$. This positive linear combination involves ι_i' and therefore cannot define ι, since J_1 entails ι_i' which would mean that $\iota \in J_1$ is redundant. Hence $\boldsymbol{a'} = \boldsymbol{a}$. Since ι_i touches J_1, $c' = c$ and thus $\iota_i \in I_C^S$. Hence, for all $i \leq k$, $\iota_i \in I_C^S$ and thus ι is redundant in $[\![I_C \cup I_C^S]\!]$. □

Based on Lemma 2 and 3, standard widening can be implemented as follows:

Theorem 1. *Let I_1, E_1 and I_2 be in normal form. Then $[\![\mathcal{C}_1 \cup \mathcal{C}_2]\!] = [\![I_C \cup I_C^S]\!]$.*

Proof. By Lemma 2, $[\![\mathcal{C}_1 \cup \mathcal{C}_2]\!] = [\![\mathcal{C}_1 \cup I_C^S]\!]$. By Lemma 3, any $\iota \in \mathcal{C}_1 \setminus I_C$ is redundant with some inequalities in $I_C \cup I_C^S$. Thus, with $\mathcal{C}_1 = I_C \cup (\mathcal{C}_1 \setminus I_C)$, it follows that $[\![\mathcal{C}_1 \cup I_C^S]\!] = [\![I_C \cup (\mathcal{C}_1 \setminus I_C) \cup I_C^S]\!] = [\![I_C \cup I_C^S]\!]$. □

The above exercise of expressing standard widening on a normalized H-representation, which we call "normal widening", is beneficial not only to simplify an actual implementation: Standard widening still leads to certain imprecisions that are illustrated in the next section. The normalized H-representation forms the basis for heuristics that improve upon standard widening without requiring the generators of a polyhedron.

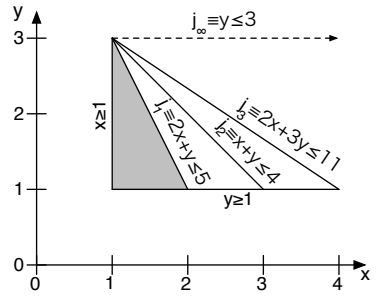

Fig. 4. Iterates with a single changing facet.

4 Improving Precision Through Additional Heuristics

The standard widening algorithm is very precise when the state changes in a way that adds new inequalities. In certain situations, however, the number of inequalities does not change, but their slopes change in that they rotate via some vertex of the polyhedron. In these cases standard widening removes the inequality, which may often cause loss of bounds on the variables in that direction. One technique is to check which upper and lower bounds of variables are the same between two iterations and to add these stable bounds back in if widening makes the polyhedron unbounded. In this section we are concerned with more sophisticated techniques that anticipate the way inequalities rotate. As far as we know, our proposal is the first heuristic that does not require the double description of a polyhedron. Furthermore, in contrast to previous heuristics [1], we are interested in extrapolating to a space that can possibly be a precise linear invariant. If this likely invariant turns out to an incorrect guess, we aim to widen quickly while retaining stable upper and lower bounds on variables.

In order to illustrate a possible precision loss of standard widening, consider Fig. 4 which shows the polyhedra $P_i = [\![\{x \geq 1, y \geq 1, j_i\}]\!]$ that present consecutive iterations of a fixpoint computation. Standard widening identifies the inequalities j_i as unstable and thus return $P' = [\![\{x \geq 1, y \geq 1\}$ as fixpoint. This example is *reasonable* since the states are the result of $P_i = \bigcup_i F^i(P_0)$ where $P_0 = [\![\{x = 1, 1 \leq y \leq 3\}]\!]$ and the transfer function $F(P) = (P \cap \{y = 1\})[x/x - 1]$ where F implements the program fragment `if (y==1) x:=x+1`. Given that the crucial statement does not change any values in the area where $y = 3$, the loss of the upper bound $y \leq 3$ is unexpected and often unacceptable. While re-adding stable bounds would fix this particular case, it cannot help when the state space development is rotated slightly which would be the case for `if (x==y) { x:=x+1; y:=y+1 }`. It is tempting to "guess" that j_i may evolve until it is anti-parallel to $y \geq 1$, however, as the next section will demonstrate, a reasonable opposing inequality does not always exist nor is it evident that a loop in a program should lead to opposing inequalities. The next section therefore considers the more general case of two evolving inequalities.

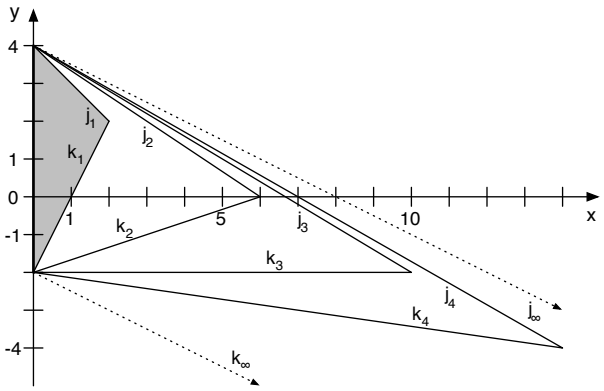

Fig. 5. An evolving sequence of polyhedra in which several facets change.

4.1 Widening in the Presence of Several Changing Inequalities

Defining a heuristic based on rotating inequalities is a tempting approach since these inequalities rotate because they connect a stable part of the state space with the evolving part. However, it is difficult to know how to capture rotation and how far to rotate. A rotation is defined as a multiplication with a matrix whose determinant is one which is impossible to implement with exact rational arithmetic. Instead, the difference between coefficients could be tracked which is demonstrated in Fig. 5. Here, the initial state $P_0 = [\![\{y \geq -2, y \leq 4, x = 0\}]\!]$ is repeatedly transformed by $F(P) = P \mathbin{\overline{\curlyvee}} ((P \cap [\![\{x{+}2y = 6\}]\!])[x/(x{+}4), y/(y{+}2)])$ which implements the program fragment `if (x+2y==6) { x:=x-4; y:=y-2; }`.

The polyhedra $P_1, \ldots P_3$ can be described by the inequality sets $\{y \geq -2, y \leq 4, x \geq 0, j_i, k_i\}$. The inequalities j_i and k_i are as follows:

$$
\begin{array}{cc}
\multicolumn{2}{c}{j_0 \equiv k_0 \equiv x \leq 0} \\
j_1 \equiv x + y \leq 4 & k_1 \equiv x - \frac{1}{2}y \leq 1 \\
j_2 \equiv x + \frac{3}{2}y \leq 6 & k_2 \equiv x - 3y \leq 6 \\
j_3 \equiv x + \frac{5}{3}y \leq \frac{20}{3} & k_3 \equiv -y \leq 2 \\
j_4 \equiv x + \frac{7}{4}y \leq 7 & k_4 \equiv -x - 7y \leq 14 \\
j_\infty \equiv x + 2y \leq 8 & k_\infty \equiv -x - 2y \leq 4
\end{array}
$$

While certain sequences of inequalities have a constant difference between their fractions of corresponding variables (e.g., the fractions of x and y in j_1, j_2, j_3, j_4), this difference is occasionally disrupted due to normalization (c.f. k_3). Furthermore, it is not clear how to infer j_∞ and k_∞. Thus, rather than anticipating how inequalities change that connect the evolving to the stable parts of the polyhedron, we try to identify the trajectory of the evolving parts and calculate new inequalities that do not obstruct this trajectory. To this end, we first partition the inequalities of the two polyhedra using *split* in Listing 2 that, unlike the standard widening, partitions the inequality sets into *stable* inequalities I_C and *unstable* inequalities I_i^Δ of P_i for $i = 1, 2$. The set I_2^Δ is furthermore reduced to those inequalities $I_{2,t}^\Delta \subseteq I_2^\Delta$ that have changed but which still touch

Listing 2 Separating inequalities according to stability

procedure $split(E_1, I_1, I_2)$ where $E_1 \subseteq Eq_n, I_1 \subseteq Ineq_n, I_2 \subseteq Ineq_n$
  $I_C \leftarrow (I_1 \cup E_1^{\leq}) \cap I_2$
  $I_1^{\Delta} \leftarrow (I_1 \cup E_1^{\leq}) \setminus I_C$
  $I_2^{\Delta} \leftarrow I_2 \setminus I_C$
  $I_{2,t}^{\Delta} \leftarrow \{\iota_2 \in I_2 \setminus (I_1 \cup E_1^{\leq}) \mid inline(E_1, \iota_2) \in I_1 \vee inline(E_1, \iota_2) \equiv \mathbf{0} \cdot \boldsymbol{x} \leq 0\}$
  for $\boldsymbol{a} \cdot \boldsymbol{x} \leq c \in I_2^{\Delta}$ **do**
    $c' \leftarrow max(\boldsymbol{a} \cdot \boldsymbol{x}, E_1 \cup I_1)$
    if $c' = c$ **then**
      $I_{2,t}^{\Delta} \leftarrow I_{2,t}^{\Delta} \cup \{\boldsymbol{a} \cdot \boldsymbol{x} \leq c\}$
    end if
  end for
  return $\langle I_C, I_1^{\Delta}, I_2^{\Delta}, I_{2,t}^{\Delta} \rangle$

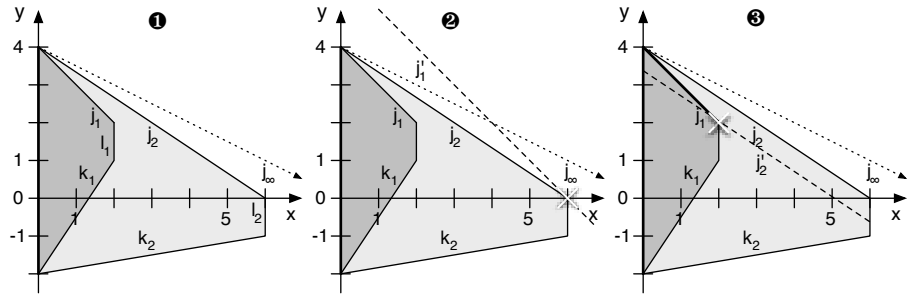

Fig. 6. Extrapolating the evolving area of the polyhedron using linear programming.

the polyhedron P_1. The idea is that inequalities in $I_{2,t}^{\Delta}$ connect the stable with the evolving part of the polyhedron whereas $I_2^{\Delta} \setminus I_{2,t}^{\Delta}$ only touch the evolving part and can thus not be used to reason about how the P_i will change.

Given these sets, the idea of our heuristic is illustrated in Fig. 6 which shows a modified version of the previous example. Here, $I_C = \{x \geq 0\}$ is the only common inequality and $I_i^{\Delta} = \{j_i, k_i, l_i\}$. In particular, $I_{2,t}^{\Delta} = \{j_2, k_2\}$ contains the set of inequalities that we can use to reason about the change in state. We commence by maximizing j_2 in P_1. Although we know that j_2 touches P_1 (that is, we know that the maximum is the right-hand side of j_2), the Simplex solver returns a set of inequalities whose combination is j_2. From the first diagram in Fig. 6 it can be seen that j_2 is a linear combination of $x \geq 0 \in I_C$ and $j_1 \in I_1^{\Delta}$. Hence, we know (or assume) that j_2 has evolved from j_1 as both are unstable and touch the stable part of the state space at (at least) one vertex.

The second diagram shows how we now infer an evolving vertex that lies in j_2. Specifically, we maximize the inequality $j_1 \equiv \boldsymbol{a}_1 \cdot \boldsymbol{x} \leq c_1$ in P_2 which yields a new constant c_1' at which the displaced inequality j_1' touches the evolving vertex. This vertex is shown as white cross in the second diagram. We store this displacement as $\delta_1 = c_1' - c_1$. Diagram three now shows how we find a vertex in P_1 from which this vertex originated. In particular, we reduce P_1 to the boundary of j_1 by adding $\boldsymbol{a}_1 \cdot \boldsymbol{x} = c_1$ to its inequality set. The resulting space

Listing 3 Directed widening.

procedure $widen(I_C, I_1^\Delta, I_2^\Delta, I_{2,t}^\Delta)$ where $I_C, I_1^\Delta, I_2^\Delta \subseteq Ineq_n, I_{2,t}^\Delta \subseteq I_2^\Delta$

1: $R \leftarrow \emptyset$
2: $I_D \leftarrow I_C$
3: $A_1\boldsymbol{x} \leq \boldsymbol{c}_1 \leftarrow I_C \cup I_1^\Delta$
4: $A_2\boldsymbol{x} \leq \boldsymbol{c}_2 \leftarrow I_C \cup I_2^\Delta$
5: **for** $\boldsymbol{a}_2 \cdot \boldsymbol{x} \leq c_2 \in I_{2,t}^\Delta$ **do**
6: $\quad\langle \boldsymbol{\lambda}_1, c_2 \rangle \leftarrow maxExt(\boldsymbol{a}_2 \cdot \boldsymbol{x}, A_1\boldsymbol{x} \leq \boldsymbol{c}_1)$
7: $\quad\boldsymbol{\lambda}_1^\Delta \leftarrow \langle f_1, \ldots f_m \rangle$ where $f_i = \begin{cases} \kappa_i \cdot \boldsymbol{\lambda}_1 & \text{if } (\kappa_i A) \cdot \boldsymbol{x} \leq (\kappa_i \boldsymbol{c}) \in I_1^\Delta \\ 0 & \text{otherwise} \end{cases}$
8: $\quad\boldsymbol{a}_1 \cdot \boldsymbol{x} \leq c_1 \leftarrow (\boldsymbol{\lambda}_1^\Delta A_1) \cdot \boldsymbol{x} \leq (\boldsymbol{\lambda}_1^\Delta \cdot \boldsymbol{c}_1)$
9: $\quad c_2' \leftarrow -max(-\boldsymbol{a}_2 \cdot \boldsymbol{x}, I_C \cup I_1^\Delta \cup \{\boldsymbol{a}_1 \cdot \boldsymbol{x} = c_1\})$
10: $\quad c_1' \leftarrow max(\boldsymbol{a}_1 \cdot \boldsymbol{x}, I_C \cup I_2^\Delta \cup \{\boldsymbol{a}_2 \cdot \boldsymbol{x} = c_2\})$
11: $\quad$**if** $c_1' < \infty \wedge c_2' > -\infty \wedge c_1' > c_1 \wedge c_2 > c_2'$ **then**
12: $\quad\quad\delta_1 \leftarrow c_1' - c_1$
13: $\quad\quad\delta_2 \leftarrow c_2 - c_2'$
14: $\quad\quad\boldsymbol{a}_3 \leftarrow \delta_2\boldsymbol{a}_1 - \delta_1\boldsymbol{a}_2$
15: $\quad\quad c_3 \leftarrow max(\boldsymbol{a}_3 \cdot \boldsymbol{x}, I_C \cup I_2^\Delta)$
16: $\quad\quad I_D \leftarrow I_D \cup \{\boldsymbol{a}_3 \cdot \boldsymbol{x} \leq c_3\}$
17: $\quad$**else**
18: $\quad\quad\boldsymbol{r}_1 \leftarrow calcRay(\boldsymbol{a}_1, \boldsymbol{a}_2)$ /* *Calculate ray with* $\boldsymbol{a}_1 \cdot \boldsymbol{r} = 0$ *and* $\boldsymbol{a}_2 \cdot \boldsymbol{r} \leq 0$. */
19: $\quad\quad\boldsymbol{r}_2 \leftarrow calcRay(-\boldsymbol{a}_2, -\boldsymbol{a}_1)$
20: $\quad\quad R \leftarrow R \cup \{evolveRay(\boldsymbol{r}_1, \boldsymbol{r}_2)\}$
21: $\quad$**end if**
22: **end for**
23: **for** $\boldsymbol{r} \in R$ **do**
24: $\quad I_D \leftarrow addRay(I_D, \boldsymbol{r})$
25: **end for**
26: **return** I_D

$P_1 \cap [\![\boldsymbol{a}_1 \cdot \boldsymbol{x} = c_1]\!]$ is indicated as thick line in the third diagram. We then minimize $j_2 \equiv \boldsymbol{a}_2 \cdot \boldsymbol{x} \leq c_2$ in this space, yielding the white vertex in the third diagram. From the inferred minimum c_2', we calculate the displacement $\delta_2 = c_2 - c_2'$. The inequality $(\delta_2\boldsymbol{a}_1 - \delta_1\boldsymbol{a}_2) \cdot \boldsymbol{x} \leq \delta_2 c_1 - \delta_1 c_2$ is now the sought after inequality j_∞ that is rotated around the intersection of j_1 and j_2 and has a slope that could connect the two white crosses. A similar calculation can be performed to find k_∞. Based on this general idea, the next section details the general algorithm that also works for possibly unbounded polyhedra in higher dimensions.

4.2 Implementation

Algorithm 3 implements our heuristic which we call "directed widening". The shown function takes the constraint sets describing P_1 and P_2 after they have been partitioned by Alg. 2. In terms of general structure, it calculates a set of rays R and a set of output constraints I_D. By default, line 23 will add the rays R to $[\![I_D]\!]$, and this process is detailed in Listing 5 for self-containedness.

Listing 4 Calculating a ray that is normal to $\boldsymbol{v}_1$.

procedure *calcRay*$(\boldsymbol{v}_1, \boldsymbol{v}_2)$ where $\boldsymbol{v}_1, \boldsymbol{v}_2 \in \mathbb{Q}^n$
1: **return** $(\boldsymbol{v}_2 \cdot \boldsymbol{v}_1)\boldsymbol{v}_1 - (\boldsymbol{v}_1 \cdot \boldsymbol{v}_1)\boldsymbol{v}_2$

Listing 5 Adding a ray $\boldsymbol{r}$ to an H-polyhedron.

procedure *addRay*$(I, \boldsymbol{r})$ where $I \subseteq Ineq_n$, $\boldsymbol{r} \in \mathbb{Q}^n$
1: $I' \leftarrow \{\kappa_{n+1} \cdot \boldsymbol{x}' \leq 0\}$ /* *Let* $\boldsymbol{x}'$ *be an* $n+1$*-dimensional vector of variables.* */
2: **for** $\boldsymbol{a} \cdot \boldsymbol{x} \leq c \in I$ **do**
3: $\quad \boldsymbol{a}' \leftarrow \langle a_1, \ldots a_n, \boldsymbol{a} \cdot \boldsymbol{r} \rangle$ where $\langle a_1, \ldots a_n \rangle = \boldsymbol{a}$
4: $\quad I' \leftarrow I' \cup \{\boldsymbol{a}' \cdot \boldsymbol{x}' \leq c\}$
5: **end for**
6: **return** $\exists_{x_{n+1}}(I')$ /* $\exists_x(\cdot)$ *is a function that projects out* x */

The algorithm itself converts the constraint system representing P_1 into matrix format (line 3) which is necessary to extract a facet of P_1 that an inequality $\boldsymbol{a}_2 \cdot \boldsymbol{x} \leq c_2 \in I^{\Delta}_{2,t}$ has evolved from. In particular, the facet of P_1 that $\boldsymbol{a}_2 \cdot \boldsymbol{x} \leq c_2$ has evolved from might be a linear combination of the unstable inequalities in I^{Δ}_1 rather than a single inequality. In order to find this linear combination, line 6 runs an extended linear program. The maximum c_2 in the result is ignored and only the vector of multipliers $\boldsymbol{\lambda}_1$ is kept which obeys $\boldsymbol{\lambda}_1 A_1 = \boldsymbol{a}_2$. However, $\boldsymbol{\lambda}_1$ combines inequalities from I^{Δ}_1 and I_C whereas we are only interested in a linear combination of the unstable inequalities I^{Δ}_1. To this end, line 7 sets all coefficients of $\boldsymbol{\lambda}^{\Delta}_1$ to zero that correspond to a stable inequality in I_C. Here, κ_i denotes a vector that contains a one in the ith position and is zero otherwise.

Note that $\boldsymbol{\lambda}^{\Delta}_1$ is non-zero since otherwise $\boldsymbol{\lambda}_1$ would only combine inequalities in I_C and thus $[\![I_C]\!] \subseteq [\![\boldsymbol{a}_2 \cdot \boldsymbol{x} \leq c_2]\!]$. In this case, since the inequalities I_C also describe P_2, the constraint $\boldsymbol{a}_2 \cdot \boldsymbol{x} \leq c_2 \in I^{\Delta}_2$ would be redundant in $I_C \cup I^{\Delta}_2$ which contradicts our assumption of a non-redundant representation.

The resulting $\boldsymbol{\lambda}^{\Delta}_1$ is now used in line 8 to calculate a *virtual* inequality $\boldsymbol{a}_1 \cdot \boldsymbol{x} \leq c_1$ from I^{Δ}_1 which may be linear combination of several inequalities of I^{Δ}_1. As explained, we assume that $\iota_2 \equiv \boldsymbol{a}_2 \cdot \boldsymbol{x} \leq c_2$ has evolved from $\iota_1 \equiv \boldsymbol{a}_1 \cdot \boldsymbol{x} \leq c_1$. We now measure the distance that ι_2 can be moved inwards on the boundary of ι_1 in line 9. Analogously, we measure how much ι_1 can be moved outwards on the boundary of ι_2 in line 10. Note that, in order to ensure that we find a maximum on ι_2 and not on a different facet, also restrict the polyhedron P_2 to the boundary of ι_2. As an example for why this is necessary, consider adding an inequality m_2 to the system in Fig 6 whose slope is between j_2 and l_2. Simply maximizing j_1 in P_2 may find the maximum at the intersection of l_2 and m_2.

If both, ι_1 and ι_2 can be translated a finite amount within the other inequality, the two distances δ_1 and δ_2 are used to calculate the slope of the new inequality $\iota_\infty \equiv \boldsymbol{a}_3 \cdot \boldsymbol{x} \leq c_3$. However, rather than calculating c_3, line 15 infers the constant c_3 using a linear program. This is necessary, since ι_∞ has a slope that may cut off some state of the current state space. By maximizing constant of ι_∞ in $I_2 = I_C \cup I^{\Delta}_2$ we ensure that it entails the current state.

t	timed1_cl					timed2_cl					timed2					initializedRect				
	DW		PPL		FP	DW		PPL		FP	DW		PPL		FP	DW		PPL		FP
0	14	0.03	27	0.09	=	19	0.05	27	0.09	≠	14	0.02	20	0.05	⊒	26	0.07	33	0.11	⊑
1	14	0.03	43	0.15	=	19	0.05	43	0.16	≠	14	0.02	26	0.06	⊒	26	0.07	42	0.15	=
	=		=			≠		=			≠		=			=		=		
2	26	0.06	63	0.23	=	25	0.06	59	0.23	≠	19	0.03	29	0.08	=	34	0.10	51	0.19	=
	=		=			=		=			=		=			=		=		
3	42	0.11	80	0.29	=	41	0.11	73	0.32	≠	25	0.04	29	0.08	=	43	0.12	60	0.23	=
	=		=			=		=			=		=			=		=		
4	62	0.17	95	0.35	=	55	0.17	85	0.37	≠	28	0.05	29	0.08	=	52	0.15	69	0.27	=

t	multirate_cl					multirate					initializedSingular_cl					rectangular				
	DW		PPL		FP	DW		PPL		FP	DW		PPL		FP	DW		PPL		FP
0	19	0.04	27	0.08	⊒	14	0.02	20	0.04	⊒	26	0.16	51	0.34	⊒	14	0.07	14	0.07	⊑
1	19	0.04	43	0.12	⊒	14	0.02	26	0.06	⊒	26	0.16	83	0.55	⊒	14	0.07	26	1.33	⊒
	≠		≠			≠		≠			≠		≠			≠		≠		
2	25	0.05	63	0.19	=	19	0.03	32	0.07	=	50	0.28	123	0.81	=	26	1.30	42	5.50	=
	=		=			=		=			=		=			=		=		
3	41	0.08	87	0.26	=	25	0.03	38	0.07	=	82	0.45	171	1.15	=	42	5.45	58	10.75	=
	=		=			=		=			=		=			=		=		
4	61	0.12	115	0.34	=	31	0.04	44	0.09	=	122	0.65	227	1.50	=	58	10.67	73	15.81	=

Table 1. Counting the widening applications and comparing the precision of fixpoints.

In case ι_1 can be relaxed without ever touching a vertex in P_2, then P_2 contains a ray. The task, therefore, is to calculate a ray towards a different direction. Line 18 calculates a ray that is orthogonal to the normal vector $\boldsymbol{a}_1$ of inequality ι_1 and which lies on the feasible side of inequality ι_2. Analogously, line 19 infers a ray that is orthogonal to ι_2 but which lies on the infeasible side of inequality ι_1. Thus, $\boldsymbol{r}_1$ is contained in P_1 whereas P_2 contains both, $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$. A new ray is needed that anticipates the evolution of these two rays. This task is delegated to a heuristic *evolveRay* which checks in which elements the ray is changing (modulus scaling) and sets these indices to zero. For each index i that is set to zero, the corresponding variable x_i receives a lower or upper bound. Since this heuristic has already been presented in [1] we omit it here. The resulting ray is then added to the constraint set using projection as implemented by *addRay* in Fig 5. Projection on constraints can be implemented using Fourier-Motzkin variable elimination. We now proceed to evaluate our heuristic.

5 Evaluation

Implementing standard widening by our normal widening algorithm of Sect. 3.2 refines the quadratic number of entailment checks of [5] to a few syntactic checks. Each entailment check requires a different linear program to be run and is thus rather expensive. Replacing the costly entailment checks with normal widening reduces the total analysis time from 0.210s to 0.149s in one of our larger tests. Since the speed-up is only 40%, we conclude that the analysis time is dominated by the evaluation of the instructions in the loop body rather than by the widening

algorithm itself. Thus, in order to assess the merit of a widening, it is more informative to count the number of iterations that are required to find a fixpoint.

To this end, we compared directed widening of Sect. 4 against the BHRZ03 widening [1] of the Parma Polyhedra Library which is implemented based on the double description method and combines several heuristics. A direct comparison is hampered by the use of tokens. The idea is that the user assigns a number of tokens to a widening point which can be used up to perform heuristics that may not terminate. Once all tokens are used, only heuristics may be applied that eventually terminate, e.g. standard widening. Choosing the right number of tokens is often considered "black magic". Since our directed widening tries to find linear translations in the loop body, it needs to observe the effect of two translations, say between P_0, P_1 and P_1, P_2, in order to extrapolate the change between them. Thus, the right number of tokens for our algorithm is always two. Note that tokens do not directly relate to the number of iterations required to reach a fixpoint: tokens do not have to be used when applying a heuristic that eventually terminates. For instance, the number of equalities that hold in a polyhedron can only decrease, thus, one could perform any non-terminating extrapolation while the number of equalities decreases without using up tokens.

In order to assess how quickly the widenings enforce termination and how precise the obtained fixpoint is, we picked eight example systems from the timed/hybrid automata literature [11, 13], each containing several nested loops. Table 1 shows the number of tokens (column "t") that the widening was allowed to use at each loop. For each number of tokens, the double columns directed widening "DW" and BHRZ03 widening "PPL" show the number of calls to the widening operator required to reach a fixpoint and the total analysis time (measured in seconds). The running time of the analyses is roughly proportional to the number of calls to the widening operator which, in turn, corresponds to the number of times a loop body is evaluated. We thus address after how many iterations our directed widening obtains a fixpoint that is as precise as that of the PPL.

To this end, we decorated the table with $=, \neq, \sqsubseteq, \sqsupseteq$ to compare the precision of the obtained fixpoints. Specifically, the column "FP" contains $\sqsubseteq$ if the fixpoint was better in the directed widening, and $\sqsupseteq$ if it was better in the PPL. Fixpoints can also be equal $=$ or incomparable $\neq$. For comparisons between the fixpoints of the same algorithm running with different tokens, we use $\neq$ to indicate that the fixpoint changed. As predicted, our directed widening obtains its best fixpoint with two tokens, which is sufficient to identify linear translations. Interestingly, both heuristics obtain similar precision given enough tokens. However, the number of iterations needed to obtain this precision is always lower for our directed widening, thereby leading to a faster analysis. For instance, in the seventh table "initializedSingular", both algorithms obtain their best fixpoint with two tokens. However, the PPL requires 123 evaluations of loop bodies whereas our directed widening only requires 50, yielding a considerable speed-up in the overall analysis time. Thus, even if our heuristic cannot infer more precise invariants than the combined heuristics gathered in the BHRZ03 widening of the Parma Polyhedra Library, our directed widening performs better by finding the fixpoints faster.

6 Conclusion and Related work

We have presented a simple implementation of standard widening [10] and a precise heuristic that finds fixpoints quickly. Moreover, our heuristic operates on H-polyhedra which, to our knowledge, make it the first heuristic that does not rely on the double description. This makes our directed widening particularly interesting to implementations of polyhedra that only use constraints [18, 19].

The first widening operator for the polyhedra domain was proposed in [7] and corresponds to the set $\mathcal{C}_1$ as defined in Sect. 3. Halbwachs proposed the revised widening or standard widening [10] and already provided an efficient implementation based on the double description of polyhedra. Benoy [2] showed that the above two widenings coincide when the affine spaces of the two argument polyhedra are stable. Chen et al. [5] showed that standard widening can be implemented on constraints only by using linear programming.

A wider field is the area of defining heuristics to improve standard widening. Besson et al. [3] propose a heuristic based on the generator representation that terminates since it guarantees a decreasing number of vertices and an increasing number of extreme rays. In the context of the analysis of timed automata, several heuristics have been proposed [8, 12, 13]. Bagnara et al. [1] compile several heuristics, such as combining constraints, evolving points, evolving rays, etc. Their heuristics require the generator representation as well as constraints.

Mostly orthogonal to improving the widening operator directly are attempts to limit the state space after widening. Besides classic narrowing [7], an established technique is *widening with thresholds* [4] which uses a finite set of user-specified values (thresholds) on individual variables up to which the state space is extrapolated. Similar to the thresholds strategy, Halbwachs et al. [11, 13] propose *widening up-to* technique to improve the widening by adding additional constraints from a fixed and finite set of constraints. Chen et al. [6] lift the thresholds strategy to relational domains in order to guess the slope (i.e. the variable coefficients) to obtain possibly stable constraints. Simon et al. [20] propose *widening with landmarks* which refines widening with thresholds by collecting unsatisfiable inequalities (called landmarks) and extrapolating polyhedra to the closest landmark during widening. Gopan et al. [9] propose *lookahead widening*, which improve the precision by a tuple of polyhedra in which the first determines which branches of a program are enabled while the second polyhedron is widened and narrowed. The net effect of both methods is that no new branches are enabled as the result of widening.

References

1. R. Bagnara, P. M. Hill, E. Ricci, and E. Zaffanella. Precise Widening Operators for Convex Polyhedra. *Science of Computer Programming*, 58(1–2):28–56, 2005.
2. F. Benoy. *Polyhedral Domains for Abstract Interpretation in Logic Programming.* PhD thesis, Computing Lab., University of Kent, Canterbury, UK, January 2002.
3. F. Besson, T. P. Jensen, and J.-P. Talpin. Polyhedral Analysis for Synchronous Languages. In A. Cortesi and G. Filé, editors, *Static Analysis Symposium*, volume 1694 of *LNCS*, pages 51–68, Venice, Italy, September 1999. Springer.

4. B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. A Static Analyzer for Large Safety-Critical Software. In *Programming Language Design and Implementation*, San Diego, Calif., USA, June 2003. ACM.
5. L. Chen, A. Miné, and P. Cousot. A sound floating-point polyhedra abstract domain. In *APLAS'08*, volume 5356 of *LNCS*, pages 3–18. Springer, 2008.
6. L. Chen, A. Miné, J. Wang, and P. Cousot. An abstract domain to discover interval linear equalities. In *VMCAI'10*, volume 5944 of *LNCS*, pages 112–128. 2010.
7. P. Cousot and N. Halbwachs. Automatic Discovery of Linear Constraints among Variables of a Program. In *Principles of Programming Languages*, pages 84–97, Tucson, Arizona, USA, January 1978. ACM.
8. L. Gonnord and N. Halbwachs. Combining Widening and Acceleration in Linear Relation Analysis. In *Static Analysis Symposium*, volume 4134 of *LNCS*, pages 144–160. Springer, 2006.
9. D. Gopan and T. Reps. Lookahead Widening. In T. Ball and R. B. Jones, editors, *Computer-Aided Verification*, volume 4144 of *LNCS*, Seattle, Washington, USA, August 2006. Springer.
10. N. Halbwachs. *Détermination Automatique de Relations Linéaires Vérifiées par les Variables d'un Programme.* Thèse de 3ème cicle d'informatique, Université scientifique et médicale de Grenoble, Grenoble, France, March 1979.
11. N. Halbwachs. Delay analysis in synchronous programs. In *Computer Aided Verification*, volume 697 of *LNCS*, pages 333–346. Springer, 1993.
12. N. Halbwachs, Y.-E. Proy, and P. Raymond. Verification of Linear Hybrid Systems by Means of Convex Approximations. In B. Le Charlier, editor, *Static Analysis Symposium*, Namur, Belgium, September 1994. Springer.
13. N. Halbwachs, Y.-E. Proy, and P. Roumanoff. Verification of Real-Time Systems using Linear Relation Analysis. *Formal Methods in System Design*, 11(2):157–185, August 1997.
14. J. M. Howe and A. King. Logahedra: a New Weakly Relational Domain. In Z. Lu and A. P. Ravn, editors, *Automated Technology for Verification and Analysis*, LNCS. Springer, December 2009.
15. J. L. Imbert and P. Van Hentenryck. Redundancy Elimination with a Lexicographic Solved Form. Technical Report CS-95-02, Brown University, Providence, Rhode Island, USA, 1995.
16. F. Mesnard and R. Bagnara. cTI: a Constraint-Based Termination Inference Tool for ISO-Prolog. *Theory and Practice of Logic Programming*, 5(1-2):243–257, 2005.
17. A. Miné. The Octagon Abstract Domain. In *Conference on Reverse Engineering*, pages 310–319, Stuttgart, Germany, October 2001. IEEE Computer Society.
18. S. Sankaranarayanan, M. Colón, H. B. Sipma, and Z. Manna. Efficient Strongly Relational Polyhedral Analysis. In E. A. Emerson and K. S. Namjoshi, editors, *Verification, Model Checking and Abstract Interpretation*, volume 3855 of *LNCS*, pages 111–125, Charleston, South Carolina, USA, January 2006. Springer.
19. A. Simon and A. King. Exploiting Sparsity in Polyhedral Analysis. In C. Hankin and I. Siveroni, editors, *Static Analysis Symposium*, volume 3672 of *LNCS*, pages 336–351, London, UK, September 2005. Springer.
20. A. Simon and A. King. Widening Polyhedra with Landmarks. In N. Kobayashi, editor, *Asian Symposium on Programming Languages and Systems*, volume 4279 of *LNCS*, pages 166–182, Sydney, Australia, November 2006. Springer.
21. A. Simon, A. King, and J. M. Howe. Two Variables per Linear Inequality as an Abstract Domain. In M. Leuschel, editor, *Logic-Based Program Synthesis and Transformation*, volume 2664 of *LNCS*, pages 71–89, Madrid, Spain, 2003. Springer.