An Efficient Abstract Domain for Not Necessarily Closed Polyhedra

Anna Becchi[1] and Enea Zaffanella[2]

[1] University of Udine, Italy
becchi.anna@spes.uniud.it
[2] University of Parma, Italy
enea.zaffanella@unipr.it

Abstract. We present a construction of the abstract domain of NNC (not necessarily topologically closed) polyhedra based on a recently introduced variant of the double description representation and conversion procedure. We describe the implementation of the operators needed to interface the new abstract domain with commonly available static analysis tools, highlighting the efficiency gains enabled by the new representation. We also reconsider the widening operator for NNC polyhedra, proposing a more appropriate specification based on the semantics of the domain elements, rather than their low level representation details. Finally, we provide an experimental evaluation comparing the efficiency of the new abstract domain with respect to more classical implementations.

1 Introduction

When developing or configuring a program analysis or verification tool based on Abstract Interpretation, the choice of the underlying abstract domain is a critical design decision. For numerical properties, many possible alternatives are available, each one characterized by a different tradeoff between the precision of the properties that can be expressed and the corresponding computational cost.

The abstract domain of convex polyhedra [24] is often positively considered as far as precision is concerned, but deemed unfeasible due to well-known results on its worst case exponential complexity. On the other hand, the domain of intervals [22] is definitely one of the most efficient choices, but quite often leads to insufficient precision. Many weakly-relational domains have been proposed (bounded differences [38,44], bounded logahedra [33], octagons [39], octahedra [19], parallelotopes [1], pentagons [37], subpolyhedra [36], template polyhedra [43], two variables per inequality [45], weighted hexagons [25], ...), each one providing its own contribution to a whole spectrum of options. In many cases, analysis tools are based on a suitable *combination* of several domains [15].

The recent years have witnessed significant progress in the implementation of some of these abstract domains. Sometimes, efficiency gains have been obtained by strictly algorithmic improvements: this is the case, for instance, for the implementation of the octagon domain optimized for dense representations [46], or for the adoption of more efficient adjacency tests [26,49] in the conversion procedures of convex polyhedra. In other cases, the progress resulted from the application of generic techniques that allow for a scalable use of the precise abstract domains, such as the careful adoption of variable packing, either computed statically [15] or dynamically [28,29,47,48]. As a result, analyses that were previously dismissed as unfeasible turn out to be affordable and surprisingly effective.

In this paper, building on a recent result [13] on the representation of NNC (not necessarily topologically closed) convex polyhedra in the DD (double description) framework, we describe the development of an alternative implementation for the corresponding abstract domain. We reconsider all of the operators that are needed for the definition of a classical static analysis based on Abstract Interpretation, stressing on the efficiency gains that are triggered by the adoption of the new representation. In particular, by exploiting the availability of an efficiently computable canonical representation for NNC polyhedra, we propose a more appropriate, semantics-based specification for the widening operator.

All the proposed algorithms have been implemented in the PPLite library, a new C++ library derived from the PPL (Parma Polyhedra Library, [8,9]). In order to compare the new domain with respect to some of the available alternatives, we first interface the PPLite library with Apron [34] and then use it in the static analyzer PAGAI [32]. This experimental evaluation allows for comparing both the precision and the efficiency of the analysis, showing significant speedups with respect to the more classical implementations of the domain of NNC polyhedra on a wide range of benchmarks.

The paper is structured as follows: Section 2 introduces the required notation and background concepts; Section 3 summarizes the new representation for NNC polyhedra [13]; Section 4 shows how to implement on the new representation the operators needed for the development of a static analysis tool; Section 5 summarizes the results of the experimental evaluation; we conclude in Section 6.

2 Preliminaries

We write $\mathbb{R}^n$ to denote the Euclidean topological space of dimension $n > 0$ and $\mathbb{R}_+$ for the set of non-negative reals; for $S \subseteq \mathbb{R}^n$, $\mathrm{cl}(S)$ and $\mathrm{relint}(S)$ denote the topological closure and the relative interior of S, respectively.

A not necessarily topologically closed convex polyhedron (for short, NNC polyhedron) is defined as the set of solutions of a finite system $\mathcal{C} = \langle \mathcal{C}_=, \mathcal{C}_\geq, \mathcal{C}_> \rangle$ of linear equality, non-strict inequality and strict inequality constraints, i.e.,

$$\mathcal{P} = \mathrm{con}(\mathcal{C}) \overset{\mathrm{def}}{=} \bigl\{ \boldsymbol{p} \in \mathbb{R}^n \bigm| \forall \beta = (\boldsymbol{a}^{\mathrm{T}} \boldsymbol{x} \bowtie b) \in \mathcal{C}, \bowtie \in \{=, \geq, >\} \mathrel{.} \boldsymbol{a}^{\mathrm{T}} \boldsymbol{p} \bowtie b \bigr\}.$$

The set $\mathbb{P}_n$ of all NNC polyhedra on the vector space $\mathbb{R}^n$, partially ordered by set inclusion, is a lattice[3] $\langle \mathbb{P}_n, \subseteq, \emptyset, \mathbb{R}^n, \cap, \uplus \rangle$, where the emptyset and $\mathbb{R}^n$ are the bottom and top elements, the binary meet operator is set intersection and the binary join operator '$\uplus$' is the convex polyhedral hull. We write $\mathbb{CP}_n$ to denote the lattice of closed polyhedra on $\mathbb{R}^n$, which is a sublattice of $\mathbb{P}_n$.

[3] We assume some familiarity with the basic notions of lattice theory [14].

A constraint $\beta = (\boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} \bowtie b)$ is said to be *valid* for $\mathcal{P} \in \mathbb{P}_n$ if all the points in $\mathcal{P}$ satisfy β. We write $\mathcal{H}_\beta$ to denote the hyperplane induced by β; the set $F = \mathcal{H}_\beta \cap \mathcal{P}$ is a *face* of $\mathcal{P}$. We write $\mathit{nncFaces}_{\mathcal{P}}$ to denote the finite set of faces of $\mathcal{P} \in \mathbb{P}_n$. Note that $\mathcal{P} = \bigcup\bigl\{ \operatorname{relint}(F) \bigm| F \in \mathit{nncFaces}_{\mathcal{P}} \bigr\}$.

A vector $\boldsymbol{r} \in \mathbb{R}^n$ such that $\boldsymbol{r} \neq \boldsymbol{0}$ is a *ray* of a non-empty polyhedron $\mathcal{P} \subseteq \mathbb{R}^n$ if, $\forall \boldsymbol{p} \in \mathcal{P}$ and $\forall \rho \in \mathbb{R}_+$, it holds $\boldsymbol{p} + \rho\boldsymbol{r} \in \mathcal{P}$. The empty polyhedron has no rays. If both $\boldsymbol{r}$ and $-\boldsymbol{r}$ are rays of $\mathcal{P}$, then $\boldsymbol{r}$ is a *line* of $\mathcal{P}$. A vector $\boldsymbol{c} \in \mathbb{R}^n$ is a *closure point* of a non-empty polyhedron $\mathcal{P} \subseteq \mathbb{R}^n$ if, $\forall \boldsymbol{p} \in \mathcal{P}$ and $\forall \lambda \in \mathbb{R}$ such that $0 < \lambda < 1$, it holds $\lambda\boldsymbol{p} + (1-\lambda)\boldsymbol{c} \in \mathcal{P}$. The set $\mathcal{P} \subseteq \mathbb{R}^n$ is an NNC polyhedron if there exist finite sets $L, R, C, P \subseteq \mathbb{R}^n$ such that $\boldsymbol{0} \notin (L \cup R)$ and $\mathcal{P} = \operatorname{gen}\bigl(\langle L, R, C, P\rangle\bigr)$, where

$$\operatorname{gen}\bigl(\langle L, R, C, P\rangle\bigr) \stackrel{\text{def}}{=} \left\{ L\boldsymbol{\lambda} + R\boldsymbol{\rho} + C\boldsymbol{\gamma} + P\boldsymbol{\pi} \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \boldsymbol{\lambda} \in \mathbb{R}^\ell, \boldsymbol{\rho} \in \mathbb{R}^r_+, \\ \boldsymbol{\gamma} \in \mathbb{R}^c_+, \boldsymbol{\pi} \in \mathbb{R}^p_+, \boldsymbol{\pi} \neq \boldsymbol{0}, \\ \sum_{i=1}^c \gamma_i + \sum_{i=1}^p \pi_i = 1 \end{array} \right\}.$$

We say that $\mathcal{P} \neq \emptyset$ is described by the *generator system* $\mathcal{G} = \langle L, R, C, P\rangle$ [6,11].

For a constraint β and a generator system $\mathcal{G}$, we write $\operatorname{sat}(\beta, \mathcal{G})$ to denote the generator system composed by those elements of $\mathcal{G}$ *saturating* β (i.e., satisfying the corresponding equality constraint). For a constraint system $\mathcal{C}$, we define $\operatorname{sat}(\mathcal{C}, \mathcal{G}) = \bigcap\bigl\{ \operatorname{sat}(\beta, \mathcal{G}) \bigm| \beta \in \mathcal{C} \bigr\}$.[4] We define $\operatorname{sat}(g, \mathcal{C})$ and $\operatorname{sat}(\mathcal{G}, \mathcal{C})$ similarly.

The DD method [41] combines the constraints and the generators of a polyhedron into a DD pair $(\mathcal{C}, \mathcal{G})$: we write $\mathcal{P} \equiv (\mathcal{C}, \mathcal{G})$ when $\mathcal{P} = \operatorname{con}(\mathcal{C}) = \operatorname{gen}(\mathcal{G})$.

For *topologically closed* polyhedra (i.e., those polyhedra that can be described by a constraint system where $\mathcal{C}_> = \emptyset$ and a generator system where $C = \emptyset$), there exist *conversion procedures* [17] that can compute each description starting from the other one. When converting from constraints to generators,[5] the procedure starts from a DD pair $(\mathcal{C}_0, \mathcal{G}_0)$ representing the whole vector space and adds, one at a time, the elements $\beta_0, \ldots, \beta_m$ of the input constraint system producing a sequence of DD pairs $\bigl\{(\mathcal{C}_k, \mathcal{G}_k)\bigr\}_{0 \leq k \leq m+1}$ representing the polyhedra

$$\mathbb{R}^n = \mathcal{P}_0 \xrightarrow{\beta_0} \ldots \xrightarrow{\beta_{k-1}} \mathcal{P}_k \xrightarrow{\beta_k} \mathcal{P}_{k+1} \xrightarrow{\beta_{k+1}} \ldots \xrightarrow{\beta_m} \mathcal{P}_{m+1} = \mathcal{P}.$$

When adding the constraint β_k to polyhedron $\mathcal{P}_k = \operatorname{gen}(\mathcal{G}_k)$, the generator system $\mathcal{G}_k$ is partitioned into the three components $\mathcal{G}_k^+$, $\mathcal{G}_k^0$, $\mathcal{G}_k^-$, according to the sign of the scalar products of the generators with β_k (those in $\mathcal{G}_k^0$ are the saturators of β_k); the new generator system for polyhedron $\mathcal{P}_{k+1}$ is computed as $\mathcal{G}_{k+1} \stackrel{\text{def}}{=} \mathcal{G}_k^+ \cup \mathcal{G}_k^0 \cup \mathcal{G}_k^\star$, where $\mathcal{G}_k^\star = \operatorname{comb\_adj}_{\beta_k}(\mathcal{G}_k^+, \mathcal{G}_k^-)$ and

$$\operatorname{comb\_adj}_{\beta_k}(\mathcal{G}_k^+, \mathcal{G}_k^-) \stackrel{\text{def}}{=} \bigl\{ \operatorname{comb}_{\beta_k}(g^+, g^-) \bigm| g^+ \in \mathcal{G}_k^+, g^- \in \mathcal{G}_k^-, \operatorname{adj}_{\mathcal{P}_k}(g^+, g^-) \bigr\}.$$

Function '$\operatorname{comb}_{\beta_k}$' computes a linear combination of its arguments, yielding a generator that saturates the constraint β_k; predicate '$\operatorname{adj}_{\mathcal{P}_k}$' is used to select only

[4] Note that we abuse notation by adopting the usual set operator and relation symbols to denote the corresponding component-wise extensions on systems.

[5] The opposite conversion works in the same way, exploiting duality.

those pairs of generators that are *adjacent* in $\mathcal{P}_k$. The conversion procedure can also simplify systems, putting them in *minimal form*: $\mathcal{C}$ (resp., $\mathcal{G}$) is in minimal form if it contains a maximal set of equalities (resp., lines) and no redundancies.[6]

The classical approach for the extension of the DD method to the case of NNC polyhedra, put forward in [30,31] and studied in more detail in [6,11], is the one adopted in both the `NNC_Polyhedra` domain of the PPL [9] and the NewPolka domain of the Apron library [34]. It is based on an *indirect representation*, whereby each NNC polyhedron $\mathcal{P} \in \mathbb{P}_n$ is mapped into a closed polyhedron $\mathcal{R} \in \mathbb{CP}_{n+1}$. The mapping encodes the strict inequality constraints by means of an additional space dimension (playing the role of a *slack variable*) usually denoted as ϵ, which needs to be non-negative and bounded from above. While allowing for reusing the same conversion procedures implemented for closed polyhedra, this approach is known to suffer from a few issues (which have been described in full detail in [13]), leading to avoidable inefficiencies.

3 The New Representation for NNC Polyhedra

To the best of our knowledge, the algorithms described in [13] are the first proposals of conversion procedures working on a *direct* DD representation for NNC polyhedra. In this section we summarize the results presented in [13], which lay the foundations for the development of a new implementation of the abstract domain of NNC polyhedra.

The new representation of [13] stems from the observation that some of the constraints and generators describing an NNC polyhedron need not be provided with a fully geometrical description. This can be seen for the polyhedron shown[7] on the left hand side of Figure 1: there is no need to know the exact slope of the strict inequality constraint β_1, as it can be replaced by any other strict inequality satisfied by all of the points of the polyhedron and saturated by closure point c_0; similarly, there is no need to know the precise position of point p_1, which can be replaced by any other point on the open segment (c_0, c_1).

Hence, the new representation distinguishes between the *skeleton* component, which is described geometrically, and the *non-skeleton* component, which is instead provided with a combinatorial description. For constraints, the non-skeleton component is a collection of strict inequality constraints behaving as *face cutters*: they remove some of the faces from the polyhedron described by the skeleton constraint system. For generators, the non-skeleton consists of generating points behaving as *face fillers*: they add to the polyhedron described by the skeleton generator system (the relative interior of) some of its faces.

For exposition purposes and without loss of generality, in the following we focus on the constraint representation. As usual, by duality arguments, all definitions and results can be extended to the case of generators [13].

[6] $\beta \in \mathcal{C}$ is *redundant* in $\mathcal{C}$ if $\mathrm{con}(\mathcal{C}) = \mathrm{con}(\mathcal{C} \setminus \{\beta\})$; similarly for generators.

[7] In the figures, the (strict) inequality constraints are denoted by (dashed) lines and the (closure) points are denoted by (unfilled) circles.

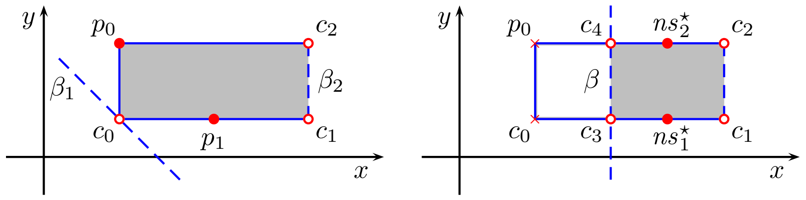

Fig. 1. On the left hand side, an NNC polyhedron having no "canonical" geometric representations; on the right hand side, the same polyhedron, after the strict inequality β has been (incrementally) processed by procedure SKEL-CONV.

A non-empty face can be uniquely identified by the set of skeleton constraints that it saturates: thus, a cutter for such a face can be represented in a combinatorial way using the same set, called its *support*.

Definition 1 (Skeleton and non-skeleton of a constraint system). *Let $\mathcal{P} = \mathrm{con}(\mathcal{C}) \in \mathbb{P}_n$ and $\mathcal{Q} = \mathrm{cl}(\mathcal{P})$, where $\mathcal{C} = \langle \mathcal{C}_=, \mathcal{C}_\geq, \mathcal{C}_> \rangle$ is a constraint system in minimal form; let $\mathcal{SC} \subseteq \mathcal{C}_>$ be the set of strict inequalities $\beta_>$ whose non-strict version $\beta_\geq$ cannot be obtained by a combination of the other constraints in $\mathcal{C}$. The* skeleton *of $\mathcal{C}$ is $\mathcal{SK} = \mathrm{skel}(\mathcal{C}) \stackrel{\text{def}}{=} \langle \mathcal{C}_=, \mathcal{C}_\geq \cup \mathcal{SC}, \emptyset \rangle$. The* support *of a face F of $\mathcal{Q}$ is $\mathcal{SK}_F \stackrel{\text{def}}{=} \{ \beta \in \mathcal{SK} \mid F \subseteq \mathcal{H}_\beta \}$. The* non-skeleton *of $\mathcal{C}$ is the set $\mathit{NS} \stackrel{\text{def}}{=} \uparrow\{ \mathcal{SK}_F \mid \exists \beta \in \mathcal{C}_> \,.\, F = \mathcal{H}_\beta \cap \mathcal{Q} \}$.*[8]

Note that the skeleton has no strict inequalities, so that $\mathrm{con}(\mathcal{SK}) = \mathrm{cl}(\mathcal{P})$. If $F' \subseteq F$ are two faces of $\mathcal{Q} = \mathrm{cl}(\mathcal{P})$, then $\mathcal{SK}_F \subseteq \mathcal{SK}_{F'}$; also, the set of faces of $\mathcal{Q}$ that are cut (i.e., not included) in $\mathcal{P}$ is downward closed, hence the non-skeleton NS is upward closed. Thus, polyhedron $\mathcal{P}$ can be obtained by removing from its topological closure those faces encoded by the non-skeleton: namely, $\mathcal{P} = \mathrm{con}(\langle \mathcal{SK}, \mathit{NS} \rangle) \stackrel{\text{def}}{=} \mathrm{con}(\mathcal{SK}) \setminus \bigcup\{ F \mid \mathcal{SK}_F \in \mathit{NS} \}$. Given a support $\mathit{ns} = \mathcal{SK}_F \in \mathit{NS}$, we write $\mathit{ns} \equiv \beta_>$ to denote that $\beta_>$ is a *materialization* of ns, i.e., a geometric cutter for face F, obtained by combining the constraints in ns.

Several optimizations can be applied at the implementation level. Since every $\mathit{ns} \in \mathit{NS}$ always includes all the equalities in $\mathcal{C}_=$, these can be left implicit, i.e., removed from the support. When this is done, the supports that happen to be singletons correspond to the combinatorial encoding of the constraints in $\mathcal{SC}$ (see Definition 1). Since their geometric position is uniquely identified, these can be *promoted*, i.e., removed from the non-skeleton component NS and directly included as strict inequalities in $\mathcal{SK}$; namely, the skeleton $\mathcal{SK} = \langle \mathcal{C}_=, \mathcal{C}_\geq \cup \mathcal{SC}, \emptyset \rangle$ is actually represented as $\mathcal{SK} = \langle \mathcal{C}_=, \mathcal{C}_\geq, \mathcal{SC} \rangle$. Finally, the upward closed set NS is represented by encoding only its minimal elements; a support $\mathit{ns} \in \mathit{NS}$ can be identified as redundant (and removed) when $\mathit{ns} \cap \mathcal{SC} \neq \emptyset$ or there exists $\mathit{ns}' \in \mathit{NS}$ such that $\mathit{ns}' \subset \mathit{ns}$. In the rest of the paper, when referring to a pair $\langle \mathcal{SK}, \mathit{NS} \rangle$, it is assumed that these optimizations are applied.

[8] $\uparrow S$ denotes the smallest upward closed set containing S.

Example 1. Consider the polyhedron on the left hand side of Figure 1, defined by constraint system $\mathcal{C} = \{2 \leq x < 7, 1 \leq y \leq 3, x + y > 3\}$. The constraint system is split into the skeleton component

$$\mathcal{SK}^{\mathrm{c}} = \bigl\langle \emptyset, \{2 \leq x, 1 \leq y \leq 3\}, \{x < 7\} \bigr\rangle$$

and the non-skeleton component $\mathit{NS}^{\mathrm{c}} = \{ns^c\}$, where $ns^c = \{2 \leq x, 1 \leq y\}$. Note that $\beta_1 = (x + y > 3)$ is one of the materializations of the support ns^c and the strict inequality $\beta_2 = (x < 7)$ has been promoted into the skeleton component. Similarly, the generator system is split into $\mathcal{SK}^{\mathrm{g}} = \bigl\langle \emptyset, \emptyset, \{c_0, c_1, c_2\}, \{p_0\} \bigr\rangle$ and $\mathit{NS}^{\mathrm{g}} = \{ns^g\}$, where $ns^g = \{c_0, c_1\}$. Point p_0 has been promoted into the skeleton component and point p_1 is a materialization of ns^g.

Conversion algorithm. Consider the conversion from constraints to generators, which incrementally adds a set of geometric constraints $\mathcal{SK}^{\mathrm{c}}_{in}$ to a DD pair $(\mathcal{C}_{dst}, \mathcal{G}_{dst})$. The new procedure [13], recalled in Pseudocode 1, handles each $\beta \in \mathcal{SK}^{\mathrm{c}}_{in}$ in two phases, one for each of the components of $\mathcal{G}_{dst} = \langle \mathcal{SK}, \mathit{NS} \rangle$.

The skeleton phase follows the same pattern of the conversion procedure for closed polyhedra. Namely, $\mathcal{SK}$ is partitioned in $\mathcal{SK}^+$, $\mathcal{SK}^0$, $\mathcal{SK}^-$ according to the sign of the scalar products with β, and the set $\mathcal{SK}^\star$ is computed by combining the generators in $\mathcal{SK}^+$ with those in $\mathcal{SK}^-$. Being restricted to the skeleton, the combination can safely apply the adjacency tests, which are crucial for efficiency.

The non-skeleton phase is where the new algorithm really differs from the classical one. Due to space constraints, we only provide here a high level, intuitive view of this phase, which is described in full detail in [13]. The NS component is partitioned in NS^+, NS^0, NS^- and $\mathit{NS}^\pm$, according to the already computed partition for the skeleton. These sets are then processed to produce $\mathit{NS}^\star$ by helper procedures MOVE-NS and CREATE-NS. Each support $ns \in \mathit{NS}^\pm$ (i.e., those whose materializations lie on both sides of $\mathcal{H}_\beta$) is updated by MOVE-NS so as to saturate (resp., satisfy) the non-strict (resp., strict) inequality constraint β. Each other support (i.e., those whose materializations lie on only one of the half-spaces induced by β) is processed by CREATE-NS, which "combines" it with the supports on the other side of β. This combination step includes a partial enumeration of the face lattice (ENUMERATE-FACES). In lines 8 to 12 $\mathit{NS}^\star$ is non-redundantly merged (using operator '$\oplus$') with the remaining supports. It should be stressed that the non-skeleton phase of the algorithm only performs set-theoretic operations on supports, i.e., no further linear combinations need to be computed. It uses two basic helper functions, to compute support closure [35] and project it on the correct side of constraint β:

$$\begin{aligned}
\mathrm{supp.cl}(ns) &\stackrel{\mathrm{def}}{=} \mathrm{sat}\bigl(\mathrm{sat}(ns, \mathcal{SK}^{\mathrm{c}}), \mathcal{SK}^{\mathrm{g}}\bigr) \setminus L, \\
\mathrm{proj}^\beta(ns) &\stackrel{\mathrm{def}}{=} \begin{cases} ns \setminus \mathcal{SK}^-, & \text{if } \beta \text{ is a strict inequality;} \\ ns \cap \mathcal{SK}^0, & \text{otherwise.} \end{cases}
\end{aligned}$$

Pseudocode 1 Conversion from geometric constraints to generators.

```
function SKEL-CONVERSION($\mathcal{SK}^{c}_{in}$, $\langle \mathcal{SK}, NS \rangle$)
    for all $\beta \in \mathcal{SK}^{c}_{in}$ do
        skel_partition($\beta$, $\mathcal{SK}$);
        nonskel_partition($\langle \mathcal{SK}, NS \rangle$);
        $\mathcal{SK}^{\star} \leftarrow$ comb_adj$_{\beta}(\mathcal{SK}^{+}, \mathcal{SK}^{-})$; $\mathcal{SK}^{0} \leftarrow \mathcal{SK}^{0} \cup \mathcal{SK}^{\star}$;
        $NS^{\star} \leftarrow$ MOVE-NS($\beta$, $\langle \mathcal{SK}, NS \rangle$);
        $NS^{\star} \leftarrow NS^{\star} \cup$ CREATE-NS($\beta$, $\langle \mathcal{SK}, NS \rangle$);
        if is_equality($\beta$) then $\langle \mathcal{SK}, NS \rangle \leftarrow \langle \mathcal{SK}^{0}, NS^{0} \oplus NS^{\star} \rangle$;
        else if is_strict_ineq($\beta$) then
            $\mathcal{SK}^{0} \leftarrow$ points_become_closure_points($\mathcal{SK}^{0}$);
            $\langle \mathcal{SK}, NS \rangle \leftarrow \langle \mathcal{SK}^{+} \cup \mathcal{SK}^{0}, NS^{+} \oplus NS^{\star} \rangle$;
        else $\langle \mathcal{SK}, NS \rangle \leftarrow \langle \mathcal{SK}^{+} \cup \mathcal{SK}^{0}, (NS^{+} \cup NS^{0}) \oplus NS^{\star} \rangle$;
        PROMOTE-SINGLETONS($\langle \mathcal{SK}, NS \rangle$);
    return $\langle \mathcal{SK}, NS \rangle$;
function MOVE-NS($\beta$, $\langle \mathcal{SK}, NS \rangle$)
    $NS^{\star} \leftarrow \emptyset$;
    for all $ns \in NS^{\pm}$ do $NS^{\star} \leftarrow NS^{\star} \cup \{\text{proj}^{\beta}(\text{supp.cl}(ns))\}$;
    return $NS^{\star}$;
function CREATE-NS($\beta$, $\langle \mathcal{SK}, NS \rangle$)
    $NS^{\star} \leftarrow \emptyset$;
    let $\mathcal{SK} = \langle L, R, C, SP \rangle$;
    for all $ns \in NS^{-} \cup \{\{\boldsymbol{p}\} \mid \boldsymbol{p} \in SP^{-}\}$ do
        $NS^{\star} \leftarrow NS^{\star} \cup$ ENUMERATE-FACES($\beta$, $ns$, $\mathcal{SK}^{+}$, $\mathcal{SK}$);
    if is_strict_ineq($\beta$) then
        for all $ns \in NS^{0} \cup \{\{\boldsymbol{p}\} \mid \boldsymbol{p} \in SP^{0}\}$ do
            $NS^{\star} \leftarrow NS^{\star} \cup$ ENUMERATE-FACES($\beta$, $ns$, $\mathcal{SK}^{+}$, $\mathcal{SK}$);
    else
        for all $ns \in NS^{+} \cup \{\{\boldsymbol{p}\} \mid \boldsymbol{p} \in SP^{+}\}$ do
            $NS^{\star} \leftarrow NS^{\star} \cup$ ENUMERATE-FACES($\beta$, $ns$, $\mathcal{SK}^{-}$, $\mathcal{SK}$);
    return $NS^{\star}$;
function ENUMERATE-FACES($\beta$, $ns$, $\mathcal{SK}'$, $\mathcal{SK}$)
    $NS^{\star} \leftarrow \emptyset$; let $\mathcal{SK}' = \langle L', R', C', SP' \rangle$;
    for all $g \in (R' \cup C')$ do $NS^{\star} \leftarrow NS^{\star} \cup \{\text{proj}^{\beta}(\text{supp.cl}(ns \cup \{g\}))\}$;
    return $NS^{\star}$;
procedure PROMOTE-SINGLETONS($\langle \mathcal{SK}, NS \rangle$)
    let $\mathcal{SK} = \langle L, R, C, SP \rangle$;
    for all $ns \in NS$ such that $ns = \langle \emptyset, \emptyset, \{\boldsymbol{c}\}, \emptyset \rangle$ do
        $NS \leftarrow NS \setminus \{ns\}$; $C \leftarrow C \setminus \{\boldsymbol{c}\}$; $SP \leftarrow SP \cup \{\boldsymbol{c}\}$;
```

Example 2. Consider again the polyhedron $\mathcal{P}$ shown on the left hand side of Figure 1, already described in Example 1. On the right hand side of the figure, we show the effect of adding to $\mathcal{P}$ the strict inequality $\beta = (4 < x)$.

The skeleton component $\mathcal{SK}^{\mathrm{g}}$ is partitioned in $\mathcal{SK}^{+} = \{c_1, c_2\}$, $\mathcal{SK}^{0} = \emptyset$ and $\mathcal{SK}^{-} = \{p_0, c_0\}$; thus, $NS^{+} = NS^{0} = NS^{-} = \emptyset$ and $NS^{\pm} = NS = \{ns^{g}\}$.[9] In the skeleton phase, the set $\mathcal{SK}^{\star} = \{c_3, c_4\}$ is obtained combining *adjacent* skeleton generators (c_0 with c_1 and p_0 with c_2, respectively). In the non-skeleton phase, MOVE-NS processes $ns^{g} \in NS^{\pm}$, obtaining

$$ns_1^{\star} = \mathrm{proj}^{\beta}(\mathrm{supp.cl}(ns^{g})) = \{c_0, c_1, c_3\} \setminus \{p_0, c_0\} = \{c_1, c_3\}.$$

We have intuitively *moved* the materializations for ns^{g} to the correct side of β. Function CREATE-NS processes the skeleton point p_0, which is a filler for (the relative interior of) point p_0 itself, the segments $[p_0, c_2]$, $[p_0, c_0]$ and the whole polyhedron $\mathcal{P}$: ENUMERATE-FACES explores this set of faces and projects them in NS^{+}, obtaining the new supports $ns_2^{\star} = \{c_2, c_4\}$ and $ns_3^{\star} = \{c_1, c_2, c_3, c_4\}$. After removing redundancies (i.e., dropping the non-minimal support $ns_3^{\star}$), we obtain

$$\mathcal{SK}^{\mathrm{g}} = \big\langle \emptyset, \emptyset, \{c_1, c_2, c_3, c_4\}, \emptyset \big\rangle, \quad NS^{\mathrm{g}} = \big\{ ns_1^{\star}, ns_2^{\star} \big\}.$$

4 Operators on the New Representation

In principle, when adopting the new representation recalled in the previous section, each operator on the abstract domain of NNC polyhedra could be implemented indirectly, by first *materializing* the non-skeleton elements and then applying the operator on the fully geometrical descriptions obtained. In this section we show that such a materialization step (and its computational overhead) is not really needed: all of the classical operators required for static analysis can be *directly* computed on the new representation, by distinguishing their effects on the geometrical and the combinatorial components, also exploiting this division to simplify some of the procedures.

Emptiness, inclusion and equality. $\mathcal{P} = \mathrm{gen}(\langle \mathcal{SK}^{\mathrm{g}}, NS^{\mathrm{g}} \rangle)$ is empty if and only if it has no point, i.e., $\mathcal{SK}^{\mathrm{g}}$ contains no point and $NS^{\mathrm{g}} = \emptyset$.

The inclusion $\mathcal{P}_1 \subseteq \mathcal{P}_2$ holds if and only if each generator of $\mathcal{P}_1$ satisfies all of the constraints of $\mathcal{P}_2$. Note that the lines, rays and closure points of $\mathcal{P}_1$ need to be checked only against the *skeleton* constraints of $\mathcal{P}_2$; only the points of $\mathcal{P}_1$ need to be checked against the non-skeleton strict inequalities of $\mathcal{P}_2$. Also, when checking a non-skeleton element, no additional scalar product needs to be computed: the result of the check is derived from the saturation information already computed (and cached) for skeleton elements. For instance, a skeleton point $\boldsymbol{p}_1 \in \mathcal{SK}_1^{\mathrm{g}}$ violates a non-skeleton constraint $ns_2 \in NS_2^{\mathrm{c}}$ when $\boldsymbol{p}_1 \in \mathrm{sat}(ns_2, \mathcal{SK}_1^{\mathrm{g}})$.

[9] Recall that $ns_g = \{c_0, c_1\}$ is the support describing the materialization p_1.

Equivalence $\mathcal{P}_1 = \mathcal{P}_2$ can be checked by performing two inclusion tests. Since the new representations satisfy a stronger form of normalization,[10] optimizations are possible: for instance, the test can be quickly answered negatively when the cardinalities of the minimized representations do not match.

Conditional and "forget". A conditional test checking an affine predicate on program variables is modeled by adding the corresponding constraint to the polyhedron defining the program state. Similarly, a non-deterministic (or non-linear) assignment can be modeled by "forgetting" all the constraints mentioning the variable assigned to, i.e., by adding the corresponding line as a generator. Hence, these two operators can be directly implemented by a call to the incremental conversion procedure of [13].

Meet and join. From a high level point of view, when a conversion procedure is available the computation of meets (i.e., set intersections) and joins (i.e., convex polyhedral hulls) on the domain of convex polyhedra is straightforward. Namely, if $\mathcal{P}_1 \equiv (\mathcal{C}_1, \mathcal{G}_1)$ and $\mathcal{P}_2 = (\mathcal{C}_2, \mathcal{G}_2)$, then the DD pair for $\mathcal{P} = \mathcal{P}_1 \cap \mathcal{P}_2$ is obtained by incrementally adding to $(\mathcal{C}_1, \mathcal{G}_1)$ the constraints in $\mathcal{C}_2$; similarly, the DD pair for $\mathcal{P} = \mathcal{P}_1 \uplus \mathcal{P}_2$ is obtained by adding to $(\mathcal{C}_1, \mathcal{G}_1)$ the generators in $\mathcal{G}_2$.

Without loss of generality, consider the case of set intersection. When incrementally adding the constraints in $\mathcal{C}_2 = \langle \mathcal{SK}_{in}^{\mathrm{c}}, \mathit{NS}_{in}^{\mathrm{c}} \rangle$ to the DD pair for $\mathcal{P}_1$, we first apply the algorithm in Pseudocode 1 to the skeleton constraints in $\mathcal{SK}_{in}^{\mathrm{c}}$; then, in order to avoid materializations, we process each non-skeleton constraint $ns \in \mathit{NS}_{in}^{\mathrm{c}}$ using the procedure shown in Pseudocode 2 (this extension of the conversion procedure was not considered in [13]).

Note that, in lines 3 to 4, the constraint ns always partitions the generators so that $\mathcal{SK}^- = \mathit{NS}^- = \mathit{NS}^\pm = \emptyset$. Hence, we can avoid all the scalar products that would have been computed in the case of a geometric input; also, saturation information is not affected, so that $\mathcal{SK}^0$ (and consequently NS^0) are easily computed by intersecting the generators saturating the support. As a consequence, the non-skeleton conversion procedure can directly call the helper STRICT-ON-EQ-POINTS(ns, $\mathcal{SK}$, NS) defined in [13], which is a tailored version of the CREATE-NS function, also including the final update of $\mathcal{SK}$ and NS.[11]

At the implementation level, a little additional care has to be taken when processing the skeleton component: if a geometric constraint $\beta \in \mathcal{SK}_{in}^{\mathrm{c}}$ is detected to be redundant, it cannot be eagerly dropped, because it might occur in a support $ns \in \mathit{NS}_{in}^{\mathrm{c}}$ and hence be needed to compute the corresponding partition; thus, the removal of β is delayed till completion of NONSKEL-CONVERSION.

[10] It is meant, with respect to those available for ϵ-representations.

[11] The first parameter β of STRICT-ON-EQ-POINTS seems to require a geometrical constraint, but this is not really the case; the parameter is only used to check the constraint kind (equality, non-strict inequality or strict inequality): in the special case of a non-skeleton element ns, we always have a strict inequality.

Pseudocode 2 Conversion from combinatorial constraints to generators.

```
    function NONSKEL-CONVERSION($NS^{c}_{in}$, $\langle \mathcal{SK}, NS \rangle$)
2:      for all $ns \in NS^{c}_{in}$ do
            $\mathcal{SK}^- = \emptyset$; $\mathcal{SK}^0 = \mathrm{sat}(ns, \mathcal{SK})$; $\mathcal{SK}^+ = \mathcal{SK} \setminus \mathcal{SK}^0$;
4:          nonskel_partition($\langle \mathcal{SK}, NS \rangle$);
            STRICT-ON-EQ-POINTS($ns$, $\langle \mathcal{SK}, NS \rangle$);
6:      return $\langle \mathcal{SK}, NS \rangle$;
    procedure STRICT-ON-EQ-POINTS($\beta$, $\langle \mathcal{SK}, NS \rangle$)
8:      $NS^\star \leftarrow \emptyset$; let $\mathcal{SK}^0 = \langle L^0, R^0, C^0, SP^0 \rangle$;
        for all $ns \in NS^0 \cup \{\{\boldsymbol{p}\} \mid \boldsymbol{p} \in SP^0\}$ do
10:         $NS^\star \leftarrow NS^\star \cup$ ENUMERATE-FACES($\beta$, $ns$, $\mathcal{SK}^+$, $\mathcal{SK}$);
        $\mathcal{SK}^0 \leftarrow$ points-become-closure-points($\mathcal{SK}^0$);
12:     $\langle \mathcal{SK}, NS \rangle \leftarrow \langle \mathcal{SK}^+ \cup \mathcal{SK}^0, NS^+ \oplus NS^\star \rangle$;
```

The extension of the conversion procedure that incrementally adds non-skeleton generators (used when computing joins) is similar and can be derived, as usual, by exploiting duality arguments.

Assignment of an affine expression. The assignment $x_i := \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} + b$ is modeled by computing the image the polyhedron under the affine map $f \colon \mathbb{R}^n \to \mathbb{R}^n$, where $\boldsymbol{q} = f(\boldsymbol{p})$ is such that $q_i = \boldsymbol{a}^{\mathrm{T}}\boldsymbol{p} + b$ and $q_j = p_j$, when $i \neq j$. If f is *invertible* (i.e., $a_i \neq 0$), then the image and its inverse f^{-1} can be easily applied to the skeleton components of the generator and constraint representations, respectively; the non-skeleton components are not affected at all. If f is not invertible (i.e., $a_i = 0$), then it is computed by first "forgetting" the constraints on x_i, adding the corresponding line, and then adding constraint $\beta = (x_i = \boldsymbol{a}^{\mathrm{T}}\boldsymbol{x} + b)$. In both cases, the minimal form of the input DD pair is *incrementally* maintained (i.e., there is no need to invoke the full conversion procedure).

4.1 A semantic widening for NNC polyhedra

The design of appropriate widening operators is considered both a key component and a main challenge in the development of abstract domains [5,7,20,21,23], in particular when targeting numerical properties [3,4,42], because their accuracy mostly depends on the particular context of application. For ease of exposition, in the following we will only consider the well known *standard widening* [27].[12]

Definition 2 (Standard widening on $\mathbb{CP}_n$). *Let $\mathcal{P}_1, \mathcal{P}_2 \in \mathbb{CP}_n$ be such that $\mathcal{P}_i = \mathrm{con}(\mathcal{C}_i)$, where $\mathcal{P}_1 \neq \emptyset$ and $\mathcal{C}_1$ is in minimal form. Let also $\mathcal{I}_i = \mathrm{ineqs}(\mathcal{C}_i)$;*[13]

[12] For all widenings '∇', we implicitly specify that $\emptyset \nabla \mathcal{P}_2 = \mathcal{P}_2$.

[13] We write $\mathrm{ineqs}(\mathcal{C})$ to denote the constraint system obtained by splitting each equality in $\mathcal{C}$ into a pair of non-strict inequalities.

then $\mathcal{P}_1 \nabla_{\mathrm{C}} \mathcal{P}_2 \overset{\text{def}}{=} \mathrm{con}(\mathcal{I}'_1 \cup \mathcal{I}'_2)$, *where*

$$\mathcal{I}'_1 = \left\{\, \beta_1 \in \mathcal{I}_1 \;\middle|\; \mathcal{P}_2 \subseteq \mathrm{con}(\{\beta_1\}) \,\right\},$$
$$\mathcal{I}'_2 = \left\{\, \beta_2 \in \mathcal{I}_2 \;\middle|\; \exists \beta_1 \in \mathcal{I}_1 \,.\, \mathcal{P}_1 = \mathrm{con}(\mathcal{I}_1 \setminus \{\beta_1\} \cup \{\beta_2\}) \,\right\}.$$

When $\mathcal{P}_1 \subseteq \mathcal{P}_2$, the following is an equivalent specification (see [3, Theorem 5]), more appropriate for implementations based on the DD method.

Definition 3. *Let* $\mathcal{P}_1, \mathcal{P}_2 \in \mathbb{CP}_n$ *be such that* $\mathcal{P}_1 \equiv (\mathcal{C}_1, \mathcal{G}_1)$, $\mathcal{P}_2 = \mathrm{con}(\mathcal{C}_2)$, $\emptyset \neq \mathcal{P}_1 \subseteq \mathcal{P}_2$ *and* $\mathcal{C}_1$ *is in minimal form. Then* $\mathcal{P}_1 \nabla_{\mathrm{C}} \mathcal{P}_2 \overset{\text{def}}{=} \mathrm{con}(\mathcal{C})$, *where*

$$\mathcal{C} = \left\{\, \beta_2 \in \mathcal{C}_2 \;\middle|\; \exists \beta_1 \in \mathcal{C}_1 \,.\, \mathrm{sat}(\beta_1, \mathcal{G}_1) = \mathrm{sat}(\beta_2, \mathcal{G}_1) \,\right\}.$$

More often than desired, the specification of widening operators relies on the "syntactic" representations of the abstract domain elements, rather than their "semantics"; thus, when a canonical representation is missing (or deemed too expensive to compute), the result of the widening depends on low level representation details. This was the case for the original proposal of widening on closed polyhedra [24], which was refined into a "semantic" widening in [27]. Similarly, the widenings defined in [40] for the graph-based representations of bounded differences and octagons were refined into semantic widenings in [2,10]. Available implementations of the domain of NNC polyhedra based on the DD method are affected by the same issue, because they compute the widening of the underlying ϵ-representations, which are not canonical. This happens for all of the widening variants defined on polyhedra, including the one proposed in [3,4], as well as the improved versions that can be obtained by applying generic techniques, such as the *widening up-to* [31].

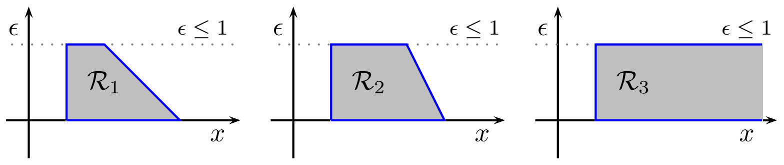

Fig. 2. Widening NNC polyhedra delegating to widening on ϵ-representations.

Example 3. Consider the ϵ-representation polyhedra in Figure 2. The two polyhedra $\mathcal{R}_1, \mathcal{R}_2 \in \mathbb{CP}_2$ on the left hand side and the middle of the figure are encoding the same NNC polyhedron $\mathcal{P} = \mathrm{con}(\mathcal{C}) \in \mathbb{P}_1$, where $\mathcal{C} = \{0.5 \leq x, x < 2\}$. Both representations encode no redundant constraint; they only differ in the slope of the facet representing the strict inequality constraint. As a consequence, when computing $\mathcal{R}_3 = \mathcal{R}_1 \nabla_{\mathrm{C}} \mathcal{R}_2$, shown on the right hand side of the figure, the standard widening operator on the ϵ-representations fails to detect the stability of the strict inequality constraint, which is dropped. $\mathcal{R}_3$ represents the NNC polyhedron $\mathcal{P}' = \mathrm{con}(\{0.5 \leq x\})$: even though correct from a theoretical point of view, the widening depends on the syntactic encoding of strict inequalities. As a side note, the user of the abstract domain might reasonably expect that a property such as $\mathcal{P} \nabla_{\mathrm{C}} \mathcal{P} = \mathcal{P}$ always holds, but this is not the case.

When implementing the widening on NNC polyhedra by delegating to the underlying widening on closed polyhedra, some precautions are required too:

- Definition 3 assumes that $\mathcal{P}_1 \subseteq \mathcal{P}_2$; note however that, for NNC polyhedra, $\mathcal{P}_1 \subseteq \mathcal{P}_2$ does not automatically imply that property $\mathcal{R}_1 \subseteq \mathcal{R}_2$ holds for the corresponding ϵ-representations;
- the implementation has to make sure that the result of the widening is still a valid ϵ-representation, i.e., the bounds for ϵ cannot be dropped;
- in order to ensure the finite convergence guarantee, the first argument $\mathcal{P}_1$ should be described by a constraint system encoding no redundant elements; however, a non-redundant description for the ϵ-representation $\mathcal{R}_1$ can still encode many redundant constraints; these have to be removed by applying the *strong minimization* procedures defined in [6,11].

As a consequence, the overall approach may also incur a significant overhead.

In contrast, when adopting the direct encoding of Section 3, we can adopt a variant of Definition 3 to obtain a *semantic* widening on NNC polyhedra, because all of the materializations of a non-skeleton strict inequality constraint share the same saturation information, no matter for the variation in their slopes.

Definition 4 (Widening on $\mathbb{P}_n$). *Let $\mathcal{P}_1, \mathcal{P}_2 \in \mathbb{P}_n$ be such that $\mathcal{P}_1 \equiv (\mathcal{C}_1, \mathcal{G}_1)$, $\mathcal{P}_2 = \mathrm{con}(\mathcal{C}_2)$, $\emptyset \neq \mathcal{P}_1 \subseteq \mathcal{P}_2$, each $\mathcal{C}_i = \langle \mathcal{SK}_i^{\mathrm{c}}, \mathit{NS}_i^{\mathrm{c}} \rangle$ is in minimal form and $\mathcal{G}_1 = \langle \mathcal{SK}_1^{\mathrm{g}}, \mathit{NS}_1^{\mathrm{g}} \rangle$. Then $\mathcal{P}_1 \nabla_{\mathrm{N}} \mathcal{P}_2 \overset{\mathrm{def}}{=} \mathrm{con}(\langle \mathcal{SK}^{\mathrm{c}}, \mathit{NS}^{\mathrm{c}} \rangle)$, where*

$$\mathcal{SK}^{\mathrm{c}} = \bigl\{\, \beta_2 \in \mathcal{SK}_2^{\mathrm{c}} \bigm| \exists \beta_1 \in \mathcal{SK}_1^{\mathrm{c}} \,.\, \mathrm{sat}(\beta_1, \mathcal{SK}_1^{\mathrm{g}}) = \mathrm{sat}(\beta_2, \mathcal{SK}_1^{\mathrm{g}}) \,\bigr\};$$
$$\mathit{NS}^{\mathrm{c}} = \{\, \mathit{ns}_2 \in \mathit{NS}_2^{\mathrm{c}} \mid \mathit{ns}_2 \subseteq \mathcal{SK}^{\mathrm{c}} \,\}.$$

The next two lemmas show that '∇_{N}' is a well-defined widening operator on $\mathbb{P}_n$.

Lemma 1. *Definition 4 specifies a binary operator on $\mathbb{P}_n$.*

Proof. We need to show that the result computed by '∇_{N}' is not affected by a change of representation for the two input arguments.

For $\mathcal{P}_1, \mathcal{P}_2 \in \mathbb{P}_n$, where $\mathcal{P}_1 \neq \emptyset$ and $\mathcal{P}_1 \subseteq \mathcal{P}_2$, let $\mathcal{P}_1 \nabla_{\mathrm{N}} \mathcal{P}_2$ be computed according to Definition 4; in particular, let $\mathcal{P}_1 \equiv (\mathcal{C}_1, \mathcal{G}_1)$ and $\mathcal{P}_2 = \mathrm{con}(\mathcal{C}_2)$, where $\mathcal{C}_i = \langle \mathcal{SK}_i^{\mathrm{c}}, \mathit{NS}_i^{\mathrm{c}} \rangle$ are arbitrary constraint representations for $\mathcal{P}_i$ satisfying the minimality hypothesis and $\mathcal{G}_1 = \langle \mathcal{SK}_1^{\mathrm{g}}, \mathit{NS}_1^{\mathrm{g}} \rangle$.

Note that, due to the inclusion hypothesis $\mathcal{P}_1 \subseteq \mathcal{P}_2$, all of the equality constraints in $\mathcal{SK}_2^{\mathrm{c}}$ are detected as stable. Let $\beta_1 \in \mathcal{SK}_1^{\mathrm{c}}$ be a skeleton (strict or non-strict) inequality constraint and $\beta_2 \in \mathcal{SK}_2^{\mathrm{c}}$ be a skeleton inequality constraint such that $\mathrm{sat}(\beta_1, \mathcal{SK}_1^{\mathrm{g}}) = \mathrm{sat}(\beta_2, \mathcal{SK}_1^{\mathrm{g}})$ holds; that is, $\beta_2 \in \mathcal{SK}_2^{\mathrm{c}}$ is detected to be stable due to $\beta_1 \in \mathcal{SK}_1^{\mathrm{c}}$. Being a skeleton constraint and due to the minimality assumption, β_1 identifies a *facet* F_1 of $\mathrm{cl}(\mathcal{P}_1)$; thus, any other constraint system representation for $\mathcal{P}_1$ will always contain a constraint β_1' (identifying the same facet F_1) such that $\mathrm{sat}(\beta_1, \mathcal{SK}_1^{\mathrm{g}}) = \mathrm{sat}(\beta_1', \mathcal{SK}_1^{\mathrm{g}})$. The same reasoning can be repeated for β_2 and F_2. Hence, the computed skeleton component $\mathcal{SK}^{\mathrm{c}}$ does not depend on the chosen representations for $\mathcal{P}_1$ and $\mathcal{P}_2$. As a side note, if $\mathcal{P}_1 = \mathrm{cl}(\mathcal{P}_1)$ then no strict inequality in $\mathcal{SK}_2^{\mathrm{c}}$ can be detected as stable. When working on closed polyhedra, Definition 4 becomes equivalent to Definition 3 and we have:

$$\mathrm{cl}(\mathcal{P}_1) \nabla_{\mathrm{N}} \mathrm{cl}(\mathcal{P}_2) = \mathrm{cl}(\mathcal{P}_1) \nabla_{\mathrm{C}} \mathrm{cl}(\mathcal{P}_2), \tag{1}$$

where '∇_{C}' is known to be well-defined on $\mathbb{CP}_n$ [3, Theorem 5].

Finally, consider the non-skeleton component and let $ns_2 \in \mathit{NS}^{\mathrm{c}}$, so that $ns_2 \in \mathit{NS}_2^{\mathrm{c}}$ and $ns_2 \subseteq \mathcal{SK}^{\mathrm{c}}$. Support ns_2 identifies a *face* (not a facet) F_2 of $\mathrm{cl}(\mathcal{P}_2)$ which is cut from $\mathcal{P}_2$, i.e., $F_2 \cap \mathcal{P}_2 = \emptyset$. Let $\mathcal{F} = \{ F_\beta \mid \beta \in ns_2 \}$ be the set of facets identified by the constraints in ns_2, so that $F_2 = \bigcap \mathcal{F}$. Note that, since ns_2 is non-redundant, all the facets in $\mathcal{F}$ have a non-empty intersection with $\mathcal{P}_2$ (i.e., they correspond to non-strict inequalities); moreover, all the facets in $\mathcal{F}$ are stable and, as observed in the previous paragraph, the set of stable facets does not depend on the chosen constraint representations. Therefore, in any other minimal representation for $\mathcal{P}_2$, there will be a set ns_2' (i.e., a support) of non-strict skeleton constraints that identifies the same set of stable facets $\mathcal{F}$; namely, ns_2' identifies the same cut face F_2 identified by ns_2. Hence, the computed non-skeleton component NS^{c} does not depend on the chosen representations for $\mathcal{P}_1$ and $\mathcal{P}_2$. $\square$

Lemma 2. $\nabla_{\mathrm{N}} \colon \mathbb{P}_n \times \mathbb{P}_n \to \mathbb{P}_n$ *is a widening operator.*

Proof. For $\mathcal{P}_1, \mathcal{P}_2 \in \mathbb{P}_n$, where $\mathcal{P}_1 \neq \emptyset$ and $\mathcal{P}_1 \subseteq \mathcal{P}_2$, let $\mathcal{P}' = \mathcal{P}_1 \nabla_{\mathrm{N}} \mathcal{P}_2$ be computed according to Definition 4.

First we show that '∇_{N}' is an upper bound operator, i.e., it satisfies both $\mathcal{P}_1 \subseteq \mathcal{P}'$ and $\mathcal{P}_2 \subseteq \mathcal{P}'$. By Definition 4, it can be seen that $\mathcal{P}' = \mathrm{con}(\mathcal{SK}^{\mathrm{c}}, \mathit{NS}^{\mathrm{c}})$, $\mathcal{P}_2 = \mathrm{con}(\mathcal{SK}_2^{\mathrm{c}}, \mathit{NS}_2^{\mathrm{c}})$ and both $\mathcal{SK}^{\mathrm{c}} \subseteq \mathcal{SK}_2^{\mathrm{c}}$ and $\mathit{NS}^{\mathrm{c}} \subseteq \mathit{NS}_2^{\mathrm{c}}$ hold; hence, the inclusion $\mathcal{P}_2 \subseteq \mathcal{P}'$ follows from the anti-monotonicity of function 'con'; the other inclusion $\mathcal{P}_1 \subseteq \mathcal{P}'$ follows from the hypothesis $\mathcal{P}_1 \subseteq \mathcal{P}_2$.

Next we show that the systematic application of '∇_{N}' forces the upward iteration sequence to stabilize after a finite number of iterates. To this end, we define a ranking function $\mathrm{rank} \colon \mathbb{P}_n \to \mathbb{N}^{2+n}$, mapping a polyhedron into the well-founded set $(\mathbb{N}^{2+n}, \ll)$, where '$\ll$' denotes the strict lexicographic ordering. For each $\mathcal{P} = \mathrm{con}(\mathcal{C}) \in \mathbb{P}_n$ such that $\mathcal{P} \neq \emptyset$ and $\mathcal{C} = \langle \mathcal{SK}^{\mathrm{c}}, \mathit{NS}^{\mathrm{c}} \rangle$ is in minimal form, we define $\mathrm{rank}(\mathcal{P}) \stackrel{\mathrm{def}}{=} (e, s, f_{n-1}, \ldots, f_j, \ldots, f_0)$, where e is the number of equality constraints in $\mathcal{SK}^{\mathrm{c}}$, s is the total number of constraints in $\mathcal{SK}^{\mathrm{c}}$ and, for each $j \in \{0, \ldots, n-1\}$, f_j is the number of strict inequality constraints in $\mathcal{C}$ cutting a face of $\mathrm{cl}(\mathcal{P})$ having affine dimension j.[14] Note that 'rank' is well-defined, because $\mathcal{C}$ is in minimal form.

[14] As an example, f_0 is the number of strict inequality constraints cutting only a vertex from the topological closure of the polyhedron.

To complete the proof we have to show that, whenever $\mathcal{P}_1 \subset \mathcal{P}' = \mathcal{P}_1 \nabla_{\mathrm{N}} \mathcal{P}_2$, i.e., when the increasing sequence has not stabilized yet, the ranking function is decreasing, i.e., $\operatorname{rank}(\mathcal{P}') \lll \operatorname{rank}(\mathcal{P}_1)$.

Let $\operatorname{rank}(\mathcal{P}_1) = (e, s, f_{n-1}, \dots, f_0)$ and $\operatorname{rank}(\mathcal{P}') = (e', s', f'_{n-1}, \dots, f'_0)$.

Since the constraint systems are in minimal form and '∇_{N}' is an upper bound operator on $\mathbb{P}_n$, for the equality constraints we always have $e' \leq e$. If $e' < e$, then the ranking function is decreasing; thus, in the rest of the proof, we assume that $e' = e$. Namely, we assume that $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}'$ all have the same affine dimension $k = n - e$.

Observe now that, by Definition 4, for the skeleton constraints we have $s' \leq s$. Namely, each skeleton (strict or non-strict) inequality constraint $\beta_2 \in \mathcal{SK}_2^{\mathrm{c}}$ that is selected to enter $\mathcal{SK}^{\mathrm{c}}$ has a unique corresponding skeleton constraint $\beta_1 \in \mathcal{SK}_1^{\mathrm{c}}$, which identifies the same facet of $\operatorname{cl}(\mathcal{P}')$ (recall that $\mathcal{P}_1$ and $\mathcal{P}'$ both have affine dimension k). Again, if $s' < s$, then the ranking function is decreasing; thus, in the rest of the proof, we assume both $e' = e$ and $s' = s$. Under such an assumption, by Definition 4, we obtain a one-to-one correspondence between the facets of $\operatorname{cl}(\mathcal{P}_1)$ and those of $\operatorname{cl}(\mathcal{P}')$: this implies $\operatorname{cl}(\mathcal{P}_1) = \operatorname{cl}(\mathcal{P}_2) = \operatorname{cl}(\mathcal{P}')$.

Consider now the tuples $t = (f_{k-1}, \dots, f_0)$ and $t' = (f'_{k-1}, \dots, f'_0)$, where as said above $k = n - e$ is the affine dimension of the polyhedra.[15] By hypothesis, $\operatorname{cl}(\mathcal{P}) = \operatorname{cl}(\mathcal{P}')$ but $\mathcal{P}_1 \subset \mathcal{P}'$; hence we obtain $t \neq t'$. Moreover, we cannot have $t \lll t'$, since this would mean that there exists a strict inequality in $\mathcal{P}'$ cutting a face which is not cut from $\mathcal{P}_1$, contradicting $\mathcal{P}_1 \subset \mathcal{P}'$. Therefore $t' \lll t$, which implies $\operatorname{rank}(\mathcal{P}') \lll \operatorname{rank}(\mathcal{P}_1)$. □

The new widening satisfies both $\mathcal{P} \nabla_{\mathrm{N}} \mathcal{P} = \mathcal{P}$ and $\mathcal{P} \nabla_{\mathrm{N}} \operatorname{cl}(\mathcal{P}) = \operatorname{cl}(\mathcal{P})$, which is not the case for the widening based on the ϵ-dimension approach. Also, equation (1) means that operator '∇_{N}' is indeed an *extension* on the domain $\mathbb{P}_n$ of the standard widening '∇_{C}' defined on $\mathbb{CP}_n$.

Example 4. Reconsider polyhedron $\mathcal{P} = \operatorname{con}(\{0.5 \leq x, x < 2\})$, for which a couple of possible ϵ-representations were shown in Figure 2. When directly encoding the strict inequalities and applying Definition 4, constraint $\beta = (x < 2)$ is detected to be stable, so that $\mathcal{P} \nabla_{\mathrm{N}} \mathcal{P} = \mathcal{P}$. Moreover, letting $\beta' = (x \leq 2)$, we also have $\mathcal{P} \nabla_{\mathrm{N}} \operatorname{cl}(\mathcal{P}) = \operatorname{cl}(\mathcal{P})$, because $\operatorname{sat}(\beta, \mathcal{SK}_1^{\mathrm{g}}) = \operatorname{sat}(\beta', \mathcal{SK}_1^{\mathrm{g}})$.

In Definition 4, the non-skeleton constraints and generators in NS_1^{c} and NS_1^{g} play no role in the computation of the widening, simplifying its implementation. As shown in Example 4, a *non-strict* inequality in $\beta_2 \in \mathcal{SK}_2^{\mathrm{c}}$ can be detected as stable (i.e., enter the result $\mathcal{SK}^{\mathrm{c}}$) even when it weakens a corresponding *strict* inequality in $\mathcal{SK}_1^{\mathrm{c}}$; this is not the case when blindly extending Definition 2. Also note that a stable non-skeleton constraint $ns \in NS^{\mathrm{c}}$ is only supported by stable skeleton constraints.

Example 5. Consider $\mathcal{P}_1 = \operatorname{con}(\{0 \leq x < 4, 0 \leq y, 0 < x + 4y \leq 8\})$ and $\mathcal{P}_2 = \operatorname{con}(\{0 \leq x \leq 4, 0 \leq y \leq 2, 0 < 2x + y\})$, shown on the left and middle of

[15] Note that for all $k \leq j \leq n - 1$, we have $f_j = f'_j = 0$.

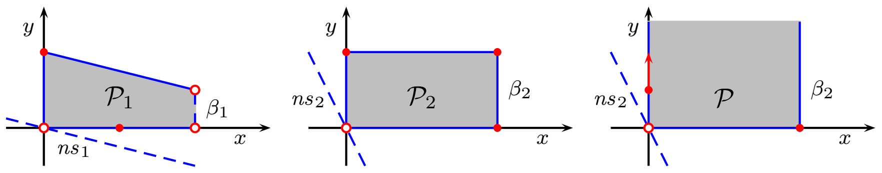

Fig. 3. From left to right: $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P} = \mathcal{P}_1 \nabla_{\mathrm{N}} \mathcal{P}_2$.

Figure 3, respectively. Constraint $\beta_2 = (x \leq 4)$ is stable, as it shares on $\mathcal{P}_1$ the same saturation information of $\beta_1 = (x < 4)$. Support $ns_2 = \{x \geq 0, y \geq 0\} \equiv (0 < 2x + y)$ is stable, no matter if the shown materialization differs from the one chosen for $ns_1 \equiv (0 < x + 4y)$, because the skeleton constraints defining it are both stable. Thus, $\mathcal{P}_1 \nabla_{\mathrm{N}} \mathcal{P}_2 = \mathrm{con}(\{0 \leq x \leq 4, 0 \leq y, 0 < 2x + y\})$, shown on the right hand side of the figure. An implementation based on Definition 2 would drop β_2 and, depending on the chosen materializations, maybe also ns_2.

In the next example we show that a blind extension of Definition 3 to the case of NNC polyhedra, where the non-skeleton component NS_2^{c} is treated the same of the skeleton component $\mathcal{SK}_2^{\mathrm{c}}$, would not result in a proper widening, since the finite convergence guarantee is compromised.

Example 6. For each $i \in \mathbb{N} \setminus \{0\}$, let $\beta_i = (x + iy \leq i)$, $\mathcal{C}_i = \{0 \leq x, 0 \leq y < 1, \beta_i\}$ and $\mathcal{P}_i = \mathrm{con}(\mathcal{C}_i)$; note that $\mathcal{P}_i \subset \mathcal{P}_{i+1}$. Polyhedra $\mathcal{P}_1$ and $\mathcal{P}_2$ are shown on the left hand side and middle of Figure 4. Note that $\mathcal{C}_i = \langle \mathcal{SK}_i^{\mathrm{c}}, NS_i^{\mathrm{c}} \rangle$, where $\mathcal{SK}_i^{\mathrm{c}} = \{0 \leq x, 0 \leq y, \beta_i\}$, $NS_i^{\mathrm{c}} = \{ns_i\}$ and $ns_i = \{0 \leq x, \beta_i\} \equiv (y < 1)$. By using Definition 3 as is, we would obtain $\mathcal{P}_i \nabla_{\mathrm{C}} \mathcal{P}_{i+1} = \mathcal{P}_{i+1}$; namely, the skeleton constraint β_{i+1} and the non-skeleton constraint ns_{i+1} are detected to be stable, since in $\mathcal{P}_i$ they share the same saturation information of ns_i (they are only saturated by closure point $(0, 1)$). Hence, $\{\mathcal{P}_i\}_{i \in \mathbb{N}}$ would form an infinite increasing chain. In contrast, when using Definition 4 to compute $\mathcal{P}_1 \nabla_{\mathrm{N}} \mathcal{P}_2$, shown on the right of the figure, constraints β_2 and ns_2 are both dropped.

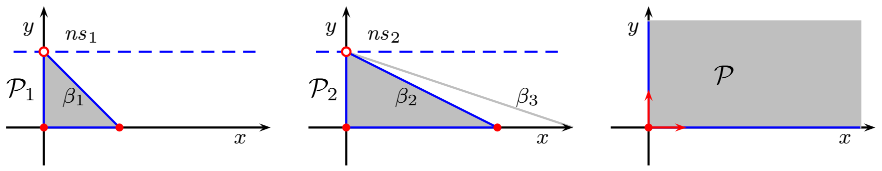

Fig. 4. An increasing chain in $\mathbb{P}_2$ where Definition 3 is not stabilizing; $\mathcal{P} = \mathcal{P}_1 \nabla_{\mathrm{N}} \mathcal{P}_2$ is the result obtained when using Definition 4.

The ranking function defined in the proof of Lemma 2 may be decreasing even though the number of non-redundant constraints is increasing.

Example 7. For $i = 1, 2$, consider $\mathcal{P}_i = \mathrm{con}(\mathcal{C}_i) \in \mathbb{P}_2$, where

$$\begin{aligned}
\mathcal{C}_1 &= \{0 < x < 1, 0 < y < 1\}, \\
\mathcal{C}_2 &= \{0 \leq x \leq 1, 0 \leq y \leq 1, 0 < x + y < 2, -1 < x - y < 1\},
\end{aligned}$$

so that $\mathcal{P}_1$ is a topologically open square and $\mathcal{P}_2$ (which is neither closed nor open) is obtained from $\mathrm{cl}(\mathcal{P}_1)$ by cutting away its four vertices. It is easy to observe that $\mathcal{P}_1 \nabla_{\mathrm{N}} \mathcal{P}_2 = \mathcal{P}_2$, because all of the skeleton constraints are stable. Note that both constraint systems are in minimal form and their cardinalities are increasing: $|\mathcal{C}_2| = 8 > 4 = |\mathcal{C}_1|$. Nonetheless, the ranking function is decreasing:

$$\mathrm{rank}(\mathcal{P}_2) = (e', s', f'_1, f'_0) = (0, 4, 0, 4) \ll (0, 4, 4, 0) = (e, s, f_1, f_0) = \mathrm{rank}(\mathcal{P}_1).$$

5 Experimental Evaluation

The new representation for NNC polyhedra, the conversion algorithms and the operators presented in the previous sections are implemented in PPLite, a new software library developed at the Department of Mathematical, Physical and Computer Sciences of the University of Parma. Derived from the Parma Polyhedra Library, PPLite is written in modern C++ and has a different goal: to provide a simpler framework for experimenting with new ideas and algorithms in the context of polyhedral computations, for both researchers and students. In particular, it is not aimed at implementing the full range of abstract domains (and operators) made available by the PPL. Other main characteristics are: (a) both closed and NNC rational polyhedra are supported; (b) arithmetic computations are based on FLINT (`http://www.flintlib.org/`); (c) encapsulation is not fully enforced, so that a knowledgeable user can directly change the contents of data structures, e.g., to experiment with alternative implementations of domain operators; (d) while performance and portability are deemed important, priority is given to ease of implementation and readability.

A preliminary experimental evaluation of the new representation and conversion algorithms for (closed and NNC) polyhedra was reported in [13], showing impressive efficiency gains with respect to the PPL. Those results were obtained inside the PPL framework, hence they were orthogonal with respect to many of the PPLite's implementation choices (e.g., the use of FLINT).[16]

In the following paragraphs we summarize the results of a more thorough experimental evaluation,[17] where the PPLite library is used in a program analysis based on Abstract Interpretation. To this end, we have interfaced the PPLite's NNC polyhedra domain to the Apron library [34], so as to make it available to PAGAI [32], a static analyzer for invariant generation built on top of the LLVM infrastructure. When using PAGAI, it is possible to choose between several abstract domains, including boxes (`box`), octagons (`oct`), the native Apron domain

[16] The efficiency gains have been confirmed when adopting the PPLite implementation.
[17] All experiments have been performed on a laptop with an Intel Core i7-3632QM CPU, 16 GB of RAM and running GNU/Linux 4.13.0-36.

for polyhedra (`pk`) and the Apron layer for the PPL's polyhedra (`ppl_poly`); we added support for the new domain `pplite_poly`. In Table 1 we report the time spent by PAGAI in calls to the operators of these abstract domains (column 'size' shows the size of the LLVM bitcode file) when analyzing some C source files distributed with PAGAI; most of these are variants of benchmarks taken from the SNU real-time benchmark suite for worst-case execution time analysis.[18]

	size	Apron's time				
test	KB	box	oct	pplite	ppl	pk
decompress	549	6.64	41.04	40.83	101.08	211.04
filter	15	1.08	5.77	19.02	88.02	82.32
adpcm	67	0.75	3.12	5.08	14.31	21.78
decompress-opt	71	0.59	9.97	3.02	7.85	13.42
nsichneu	527	0.51	0.49	1.55	3.06	2.33
cover	33	0.35	0.38	1.25	2.09	1.61
fft1	20	0.16	0.51	0.82	1.74	2.02
edn	57	0.17	0.32	0.73	1.57	1.71
compress	30	0.15	0.67	0.69	1.84	2.64
ndes	45	0.17	0.25	0.64	1.27	1.20
minver	30	0.15	0.24	0.52	1.02	1.10

Table 1. Efficiency comparison for PAGAI's domains.

The new domain performs significantly better than the other polyhedra domains, being also competitive with respect to the domain of octagons on the biggest benchmarks. It is worth stressing that these efficiency gains have been obtained even if PAGAI makes a quite limited use of strict inequalities, which are only used to model floating point values: among the tests reported in Table 1, only 'fft1' and 'minver' declare floating point variables. Moreover, the "classic" static analysis implemented in PAGAI applies no variable packing technique at all: hence, all the relational domains incur avoidable overheads [47,49], which are orthogonal with respect to the chosen implementation of NNC polyhedra.

In order to assess correctness, we also performed a different experimental evaluation where, after each and every invocation of an abstract operator, the result computed by the new domain `pplite_poly` is systematically compared with the result computed by `ppl_poly`: the only differences were recorded when computing widenings, where the semantic widening '∇_{N}' used by PPLite was sometimes more precise than the syntactic one used by PPL.

6 Conclusion

By leveraging on a new DD representation and conversion algorithm, we have presented the corresponding implementation of the abstract domain of NNC

[18] We only show those tests where the time spent by `pplite_poly` is above 0.5 seconds.

polyhedra. In particular, we focused our work on the specification of the operators needed for defining a static analysis based on Abstract Interpretation, here included a semantics-based widening operator. The experimental evaluation conducted shows that the new domain systematically outperforms the more classical implementations. As future work, we plan to extend the abstract domain so as to also support operators needed in other contexts. For instance, in the analysis and verification of hybrid systems, strict inequalities usually play a more important role: we reasonably expect that the adoption of our new implementation for the domain of NNC polyhedra may result in even larger efficiency gains.

References

1. G. Amato and F. Scozzari. The abstract domain of parallelotopes. *Electr. Notes Theor. Comput. Sci.*, 287:17–28, 2012.
2. R. Bagnara, P. M. Hill, E. Mazzi, and E. Zaffanella. Widening operators for weakly-relational numeric abstractions. In *Static Analysis: Proceedings of the 12th International Symposium*, vol. 3672 of *LNCS*, pp. 3–18, London, UK, 2005.
3. R. Bagnara, P. M. Hill, E. Ricci, and E. Zaffanella. Precise widening operators for convex polyhedra. In *Static Analysis: Proceedings of the 10th International Symposium*, vol. 2694 of *LNCS*, pp. 337–354, San Diego, USA, 2003.
4. 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.
5. R. Bagnara, P. M. Hill, and E. Zaffanella. Widening operators for powerset domains. In *Verification, Model Checking and Abstract Interpretation: Proceedings of the 5th International Conference (VMCAI 2004)*, vol. 2937 of *LNCS*, pp. 135–148, Venice, Italy, 2003.
6. R. Bagnara, P. M. Hill, and E. Zaffanella. Not necessarily closed convex polyhedra and the double description method. *Formal Asp. Comput.*, 17(2):222–257, 2005.
7. R. Bagnara, P. M. Hill, and E. Zaffanella. Widening operators for powerset domains. *Software Tools for Technology Transfer*, 8(4/5):449–466, 2006.
8. R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 72(1–2):3–21, 2008.
9. R. Bagnara, P. M. Hill, and E. Zaffanella. Applications of polyhedral computations to the analysis and verification of hardware and software systems. *Theoretical Computer Science*, 410(46):4672–4691, 2009.
10. R. Bagnara, P. M. Hill, and E. Zaffanella. Weakly-relational shapes for numeric abstractions: Improved algorithms and proofs of correctness. *Formal Methods in System Design*, 35(3):279–323, 2009.
11. R. Bagnara, E. Ricci, E. Zaffanella, and P. M. Hill. Possibly not closed convex polyhedra and the Parma Polyhedra Library. In *Static Analysis: Proceedings of the 9th International Symposium*, vol. 2477 of *LNCS*, pp. 213–229, Madrid, 2002.
12. A. Becchi and E. Zaffanella. A conversion procedure for NNC polyhedra. *CoRR*, abs/1711.09593, 2017.
13. A. Becchi and E. Zaffanella. A direct encoding for NNC polyhedra. To appear in *Computer Aided Verification: Proceedings of the 30th International Conference (CAV 2018)*, 2018. An extended version with proofs is available as [12].
14. G. Birkhoff. *Lattice Theory*, volume XXV of *Colloquium Publications.* American Mathematical Society, Providence, USA, 1967.

15. 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 *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation (PLDI'03)*, pp. 196–207, San Diego, USA, 2003.
16. N. V. Chernikova. Algorithm for finding a general formula for the non-negative solutions of system of linear equations. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 4(4):151–158, 1964.
17. N. V. Chernikova. Algorithm for finding a general formula for the non-negative solutions of system of linear inequalities. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 5(2):228–233, 1965.
18. N. V. Chernikova. Algorithm for discovering the set of all solutions of a linear programming problem. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 8(6):282–293, 1968.
19. R. Clarisó and J. Cortadella. The octahedron abstract domain. *Science of Computer Programming*, 64(1):115–139, 2007.
20. A. Cortesi. Widening operators for abstract interpretation. In *Sixth IEEE International Conference on Software Engineering and Formal Methods (SEFM 2008)*, pp. 31–40, Cape Town, South Africa, 2008.
21. A. Cortesi and M. Zanioli. Widening and narrowing operators for abstract interpretation. *Computer Languages, Systems & Structures*, 37(1):24–42, 2011.
22. P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the Fourth Annual ACM Symposium on Principles of Programming Languages*, pp. 238–252, Los Angeles, USA, 1977.
23. P. Cousot and R. Cousot. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation. In *Proceedings of the 4th International Symposium on Programming Language Implementation and Logic Programming*, vol. 631 of *LNCS*, pp. 269–295, Leuven, Belgium, 1992.
24. P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages*, pp. 84–96, Tucson, USA, 1978.
25. J. Fulara, K. Durnoga, K. Jakubczyk, and A. Schubert. Relational abstract domain of weighted hexagons. *Electr. Notes Theor. Comput. Sci.*, 267(1):59–72, 2010.
26. B. Genov. *The Convex Hull Problem in Practice: Improving the Running Time of the Double Description Method.* PhD thesis, University of Bremen, Germany, 2014.
27. N. Halbwachs. *Détermination Automatique de Relations Linéaires Vérifiées par les Variables d'un Programme.* Thèse de 3$^{\text{ème}}$ cycle d'informatique, Université scientifique et médicale de Grenoble, Grenoble, France, 1979.
28. N. Halbwachs, D. Merchat, and L. Gonnord. Some ways to reduce the space dimension in polyhedra computations. *Formal Methods in System Design*, 29(1):79–95, 2006.
29. N. Halbwachs, D. Merchat, and C. Parent-Vigouroux. Cartesian factoring of polyhedra in linear relation analysis. In *Static Analysis: Proceedings of the 10th International Symposium*, vol. 2694 of *LNCS*, pp. 355–365, San Diego, USA, 2003.
30. N. Halbwachs, Y.-E. Proy, and P. Raymond. Verification of linear hybrid systems by means of convex approximations. In *Static Analysis: Proceedings of the 1st International Symposium*, vol. 864 of *LNCS*, pp. 223–237, Namur, Belgium, 1994.
31. 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, 1997.

32. J. Henry, D. Monniaux, and M. Moy. PAGAI: A path sensitive static analyser. *Electr. Notes Theor. Comput. Sci.*, 289:15–25, 2012.
33. J. M. Howe and A. King. Logahedra: A new weakly relational domain. In *Automated Technology for Verification and Analysis, 7th International Symposium (ATVA 2009)*, pp. 306–320, Macao, China, 2009.
34. B. Jeannet and A. Miné. Apron: A library of numerical abstract domains for static analysis. In *Computer Aided Verification, Proceedings of the 21st International Conference (CAV 2009)*, vol. 5643 of *LNCS*, pp. 661–667, Grenoble, France, 2009.
35. V. Kaibel and M. E. Pfetsch. Computing the face lattice of a polytope from its vertex-facet incidences. *Computational Geometry*, 23(3):281–290, 2002.
36. V. Laviron and F. Logozzo. Subpolyhedra: A (more) scalable approach to infer linear inequalities. In *Verification, Model Checking, and Abstract Interpretation: Proceedings of the 10th International Conference (VMCAI 2009)*, vol. 5403 of *LNCS*, pp. 229–244, Savannah, USA, 2009.
37. F. Logozzo and M. Fähndrich. Pentagons: A weakly relational abstract domain for the efficient validation of array accesses. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, pp. 184–188, Fortaleza, Brazil, 2008.
38. A. Miné. A new numerical abstract domain based on difference-bound matrices. In *Proceedings of the 2nd Symposium on Programs as Data Objects (PADO 2001)*, vol. 2053 of *LNCS*, pp. 155–172, Aarhus, Denmark, 2001.
39. A. Miné. The octagon abstract domain. In *Proceedings of the Eighth Working Conference on Reverse Engineering*, pp. 310–319, Stuttgart, Germany, 2001.
40. A. Miné. *Weakly Relational Numerical Abstract Domains.* PhD thesis, École Polytechnique, Paris, France, 2005.
41. T. S. Motzkin, H. Raiffa, G. L. Thompson, and R. M. Thrall. The double description method. In *Contributions to the Theory of Games – Volume II*, number 28 in Annals of Mathematics Studies, pp. 51–73. Princeton, USA, 1953.
42. V. Notani and R. Giacobazzi. Learning based widening. 8th Workshop on Tools for Automatic Program Analysis (TAPAS'17), New York, USA, 2017.
43. S. Sankaranarayanan, H. B. Sipma, and Z. Manna. Scalable analysis of linear systems using mathematical programming. In *Verification, Model Checking and Abstract Interpretation: Proceedings of the 6th International Conference (VMCAI 2005)*, vol. 3385 of *LNCS*, pp. 25–41, Paris, France, 2005.
44. R. Shaham, E. K. Kolodner, and S. Sagiv. Heap profiling for space-efficient java. In *Proceedings of the 2001 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pp. 104–113, Snowbird, USA, 2001.
45. A. Simon, A. King, and J. M. Howe. Two variables per linear inequality as an abstract domain. In *Logic Based Program Synthesis and Transformation, 12th International Workshop*, vol. 2664 of *LNCS*, pp. 71–89, Madrid, Spain, 2002.
46. G. Singh, M. Püschel, and M. T. Vechev. Making numerical program analysis fast. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 303–313, Portland, USA, 2015.
47. G. Singh, M. Püschel, and M. T. Vechev. Fast polyhedra abstract domain. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017*, pp. 46–59, Paris, France, 2017.
48. G. Singh, M. Püschel, and M. T. Vechev. A practical construction for decomposing numerical abstract domains. *PACMPL*, 2(POPL):55:1–55:28, 2018.
49. E. Zaffanella. On the efficiency of convex polyhedra. *Electr. Notes Theor. Comput. Sci.*, 334:31–44, 2018.