Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 267 (2010) 127–138

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Speeding up Polyhedral Analysis by Identifying Common Constraints

Axel Simon[1,2]

Lehrstuhl 2 für Informatik, Technical University Munich, 85478 Garching, Germany

Abstract

Sets of linear inequalities are an expressive reasoning tool for approximating the reachable states of a program. However, the most precise way to join two states is to calculate the convex hull of the two polyhedra that are represented by the inequality sets, an operation that is exponential in the dimension of the polyhedra. We investigate how similarities in the two input polyhedra can be exploited to improve the performance of this costly operation. In particular, we discuss how common equalities and certain inequalities can be omitted from the calculation without affecting the result. We expose a maximum of common equalities and inequalities by converting the polyhedra into a normal form and give experimental evidence of the merit of our method.

Keywords: Abstract interpretation, polyhedra analysis, convex hull, factoring.

1 Introduction

Numeric invariants in programs are important for optimization and verification. In this context, one of the most interesting abstract domains to infer these invariants is that of convex polyhedra (polyhedra for short) which is able to infer linear relationships between variables [5]. Linear relationships make it possible to express symbolic bounds that are sufficient to prove, for example, the absence of buffer overflows in C programs, even when the terminating zero character is not fixed [14,13]. Furthermore, it is possible to express input/output relationships of a function [7]. However, the domain of convex polyhedra suffers from the poor scalability of its

dra whose

al output

is rather uncommon, the bottleneck is the exponential intermediate representation

[1] This work was supported by the Emmy Noether grant and the INRIA project “Abstraction” of CNRS and ENS.

[2] Email: Axel.Simon@in.tum.de

doi:10.1016/j.entcs.2010.09.011

$$
\begin{array}{r|cccccccc|rl}
\hline
 & x_1 & & & +x_3 & -x_4 & & +x_7 & & = & -1 \\
 & & 4x_2 & & +x_3 & -x_4 & & & & = & 4 \\
 & & & x_5 & & & & & & = & 0 \\
\hline
\text{equalities} & & & & & & x_6 & +4x_7 & & = & 0 \\
\text{partition 1} & & & & x_3 & -x_4 & & & -x_8 & \leq & 8 \\
 & & & & -x_3 & +x_4 & & & +x_8 & \leq & -4 \\
 & & & & & & & & x_8 & \leq & 1 \\
 & & & & & & & & -x_8 & \leq & 0 \\
\hline
\text{partition 2} & & & & & & & x_7 & & \leq & 1 \\
 & & & & & & & -x_7 & & \leq & 0 \\
\hline
\end{array}
$$

Fig. 1. Normalized form of a set of constraints

that is employed to calculate the convex hull [15]: Given two n-dimensional polyhedra as input, the double description method calculates a set of generators (vertices, rays, and lines) from the input constraints that is usually exponential in n. Similarly, calculating the convex hull via projection needs $n+1$ Fourier-Motzkin variable elimination steps, incurring a quadratic growth of inequalities in each step. Most of these inequalities are redundant and must be removed. As the output is often of manageable size anyway, we propose to omit constraints that are common to the two input polyhedra, which can reduce the size of the intermediate representation considerably.

We find a maximum of common constraints by storing a polyhedron in a normal form [10] which is obtained by substituting each equality that holds in a polyhedron in all remaining constraints, leading to system such as the one in Fig. 1. It is well known that equalities that are common to two polyhedra can be omitted when calculating their join [4]. We show that omitting common equalities yields indeed a speedup. This is contrary to the observation in [8] who only identify and remove common equalities on the exponential intermediate representation. We also show that sets of common inequalities that do not share variables with other inequalities can be omitted. Consider joining the normalized polyhedron in Fig. 1 with one where only the inequalities on x_7 are different: Firstly, the common equality $x_1 + x_3 - x_4 + x_7 = -1$ is omitted, thereby disentangling the variable x_7 from x_3, x_4, x_8. Secondly, the common set of inequalities denoted as "partition 1" shares no variables with "partition 2" and can thus be omitted, reducing the polyhedral join to a single variable.

In summary, our work makes the following three contributions: It proves that common equalities and stripes (constraints of the form $\boldsymbol{a} \cdot \boldsymbol{x} = [l, u]$) can be omitted; it proves that common, non-overlapping inequality sets can be omitted; it gives experimental evidence of the effectiveness of this factoring.

After introducing the normal form, Sect. 3 defines the convex hull using projection which is used in Sect. 4 to prove the factorings correct. The performance improvements are presented in Sect. 5 before Sect. 6 concludes.

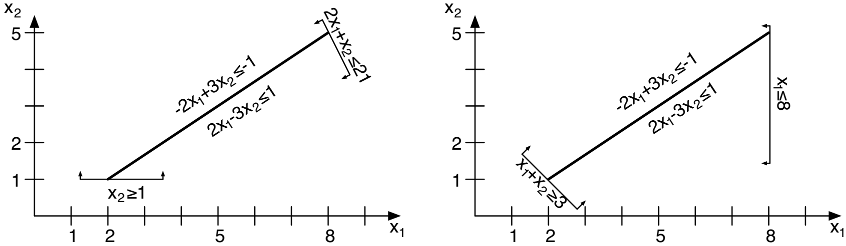

Fig. 2. The same convex polyhedron may have several representations

2 A Normal Form for Polyhedra

In order to identify common constraints in the input of the join, we first define a normal form for a polyhedron when represented as a set of equalities and inequalities. With respect to the notation, we write a constraint as $\boldsymbol{a}\cdot\boldsymbol{x}\odot c \in I$ where $\odot \in \{\leq, =\}$ and $[\![\boldsymbol{a}\cdot\boldsymbol{x}\odot c]\!] \in P \subseteq \mathbb{Q}^{|\boldsymbol{x}|}$ denotes the set of points that satisfy the constraint. We lift $[\![\cdot]\!]$ to sets with $[\![I]\!] = \bigcap\{[\![\iota]\!] \mid \iota \in I\}$. Finally, we use $Poly_n$ to denote the set of all polyhedra $P \in Poly_n$ over n variables. Let $\sqcup : Poly_n \times Poly_n \to Poly_n$ denote the join of two polyhedra of n dimensions.

Figure 2 shows two sets of inequalities that represent the same polyhedron. Although $P \sqcup P = P$, the join of the two depicted polyhedra has to be calculated explicitly since it is not clear from the inequality representation that the two polyhedra are, in fact, equal. Note that both inequality sets are non-redundant, that is, none of the inequalities can be omitted without changing the polyhedron. Thus, in order to make the representation of the two polyhedra canonical, it is necessary to transform the inequality set.

The first step to a normal form for polyhedra is to express constraints (equalities and inequalities) in a canonical way. We therefore assume that the coefficients $\boldsymbol{a} \in \mathbb{Z}^n$ and the constant $c \in \mathbb{Z}$ are in their lowest form. However, the inequalities in Fig. 2 are already canonical. The ambivalent representation is still possible because P is embedded in the affine space $[\![\{2x_1 = 3x_2 + 1\}]\!]$ which can be delimited by different inequalities. We eliminate the equality from the constraint set by substituting $\frac{3x_2+1}{2}$ for each occurrence of x_1 and appropriately scaling the inequalities, yielding $\{1 \leq x_2 \leq 5\}$ in the example. We say that x_1 is factored out in P. Repeated factoring of the smallest variable amongst the equalities eventually yields a polyhedron $P \in Poly_m$ that is fully dimensional, i.e. there exists m linearly independent points $c_1, \ldots c_m \in P$. The resulting system consists of a set of equalities whose matrix is in triangular form, as shown in Fig. 1. Indeed, Lassez et al. [10] proved that the transformation above leads to a normal form. In order to calculate the join of two polyhedra with different equality set, each equality $\boldsymbol{a}\cdot\boldsymbol{x} = c$ that is only present in one polyhedron is converted to two opposing inequalities $\boldsymbol{a}\cdot\boldsymbol{x} \leq c$ and $-\boldsymbol{a}\cdot\boldsymbol{x} \leq -c$ before the join is calculated on the inequality sets.

Substituting all equalities that hold in a polyhedron $P \in Poly_n$ requires finding all equalities first. Many equalities can be found by identifying pairs of inequalities of the form $\boldsymbol{a}\cdot\boldsymbol{x} \leq c$ and $-\boldsymbol{a}\cdot\boldsymbol{x} \leq -c$. These form naturally an equality and can be removed cheaply. A system such as $\{x + y \leq 0, -x + y \leq 0, -y \leq 0\}$ has *implicit* equalities which can be detected using Simplex. In general, given a set of inequalities I, Simplex can be used to test if an inequality $\boldsymbol{a} \cdot \boldsymbol{x} \leq c \in I$ also holds as an equality: Whenever minimizing $\boldsymbol{a} \cdot \boldsymbol{x}$ in P yields as minimum c, $\boldsymbol{a} \cdot \boldsymbol{x} = c$ holds in P and can be substituted. The task of testing all inequalities can be refined by appropriate implementation techniques [12].

The next section introduces an implementation of the join operation on polyhedra, namely the calculation of the convex hull of two polyhedra using Fourier-Motzkin elimination. We introduce this method here since it provides us with a way to argue about common constraints in the two input systems.

3 Calculating the Convex Hull via Projection

This section briefly reviews the principle of calculating the convex hull of two polyhedra using Fourier-Motzkin projection [3] which has been proposed as an alternative to the classic double-description method that is used in current implementations [2]. It forms the basis for factoring out common constraints.

Consider the calculation of $P_{12} = P_1 \sqcup P_2$ which can be defined as taking a convex combination of each point $\boldsymbol{x}_1 \in P_1$ with each point $\boldsymbol{x}_2 \in P_2$:

$$P_{12} = cl(\{\boldsymbol{x} \in \mathbb{Q}^n \mid \boldsymbol{x} = (1 - \lambda)\boldsymbol{x}_1 + \lambda \boldsymbol{x}_2 \wedge 0 \leq \lambda \leq 1 \wedge \boldsymbol{x}_1 \in P_1 \wedge \boldsymbol{x}_2 \in P_2\})$$

Here, the function cl denotes the topological closure of the result which is necessary since the convex combination of closed but infinite spaces may not be closed. See [14] for an example.

Suppose now that $P_i = [\![A_i \boldsymbol{x} \leq \boldsymbol{c}_i]\!]$ for $i = 1, 2$ where $A_i \boldsymbol{x} \leq \boldsymbol{c}_i$ is the matrix that represents the set of non-redundant inequalities defining P_i. We substitute this definition to obtain the result in terms of the inequality systems:

$$P_{12} = cl(\{\boldsymbol{x} \in \mathbb{Q}^n \mid \boldsymbol{x} = (1 - \lambda)\boldsymbol{x}_1 + \lambda \boldsymbol{x}_2 \wedge 0 \leq \lambda \leq 1 \wedge A_1 \boldsymbol{x}_1 \leq \boldsymbol{c}_1 \wedge A_2 \boldsymbol{x}_2 \leq \boldsymbol{c}_2\})$$

Substituting $\boldsymbol{z} := (1 - \lambda)\boldsymbol{x}_1$ and $\boldsymbol{y} := \lambda \boldsymbol{x}_2$ removes the multiplication, yielding:

$$P_{12} = cl(\{\boldsymbol{x} \in \mathbb{Q}^n \mid \boldsymbol{x} = \boldsymbol{z} + \boldsymbol{y} \wedge 0 \leq \lambda \leq 1 \wedge A_1 \frac{\boldsymbol{z}}{1 - \lambda} \leq \boldsymbol{c}_1 \wedge A_2 \frac{\boldsymbol{y}}{\lambda} \leq \boldsymbol{c}_2\})$$

Since $1 - \lambda$ and λ are positive, we can multiply the inequality systems by these factors without changing the $\leq$-relation. However, this step adds new solutions for $\lambda = 1$ and $\lambda = 0$ for which the division rendered the effect of the inequality systems in the previous expression undefined. Indeed, the resulting system contains only non-strict inequalities and is therefore topologically closed. Thus, we may omit the closure operation, giving:

$$P_{12} = \{\boldsymbol{x} \in \mathbb{Q}^n \mid \boldsymbol{x} = \boldsymbol{z} + \boldsymbol{y} \wedge 0 \leq \lambda \leq 1 \wedge A_1 \boldsymbol{z} \leq (1 - \lambda)\boldsymbol{c}_1 \wedge A_2 \boldsymbol{y} \leq \lambda \boldsymbol{c}_2\}$$

Now simplify by setting $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{y}$ and by bringing the variable λ to the left:

$$P_{12} = \{\boldsymbol{x} \in \mathbb{Q}^n \mid 0 \leq \lambda \leq 1 \wedge A_1\boldsymbol{x} - A_1\boldsymbol{y} + \boldsymbol{c}_1\lambda \leq \boldsymbol{c}_1 \wedge A_2\boldsymbol{y} - \boldsymbol{c}_2\lambda \leq 0\}$$

Thus, the points $\boldsymbol{x} \in \mathbb{Q}^n$ satisfied by the join $[\![A_1\boldsymbol{x} \leq \boldsymbol{c}_1]\!] \sqcup [\![A_2\boldsymbol{x} \leq \boldsymbol{c}_2]\!]$ are those that satisfy the following matrix for some $\boldsymbol{y} \in \mathbb{Q}^n$ and $\lambda \in \mathbb{Q}$:

$$\left(\begin{array}{c|c|c} A_1 & -A_1 & \boldsymbol{c}_1 \\ \hline \boldsymbol{0} & A_2 & -\boldsymbol{c}_2 \\ \hline 0\cdots0 & 0\cdots0 & -1 \\ 0\cdots0 & 0\cdots0 & 1 \end{array}\right) \left(\begin{array}{c} \boldsymbol{x} \\ \hline \boldsymbol{y} \\ \hline \lambda \end{array}\right) \leq \left(\begin{array}{c} \boldsymbol{c}_1 \\ \hline \boldsymbol{0} \\ \hline 0 \\ 1 \end{array}\right)$$

In order to calculate a set of inequalities over $\boldsymbol{x}$, the variables $\boldsymbol{y}$ and λ are projected out which constitutes the idea of calculating the convex hull via projection. Besides giving a constructive way to calculate the convex hull, the above characterisation furthermore exhibits some interesting properties. Eliminating $v \in \{\boldsymbol{y}, \lambda\}$ using Fourier-Motzkin elimination will calculate positive combinations of rows of the matrix such that the coefficient for v are zero. Thus, once all variables $\boldsymbol{y}, \lambda$ are eliminated, each inequality ι in the result is a positive linear combination of the rows of the matrix such that the terms over $\boldsymbol{y}$ and λ add up to zero. Thus, each ι is a positive linear combination of the inequalities $A_1\boldsymbol{x} \leq c_1$ plus some constant that is due to the -1 and 1 entries.

Some interesting properties can be derived from this representation of the convex hull. For instance, consider the fact that an inequality $\boldsymbol{a} \cdot \boldsymbol{x} \leq c$ that is present in both input polyhedra entails the convex hull (i.e. $P_1 \sqcup P_2 \subseteq [\![\boldsymbol{a} \cdot \boldsymbol{x} \leq c]\!]$). This can be shown directly by considering the following system where A_i' and c_i' denote the input systems without $\boldsymbol{a} \cdot \boldsymbol{x} \leq c$:

$$\left(\begin{array}{c|c|c} \begin{pmatrix} \boldsymbol{a} \\ A_1' \end{pmatrix} & \begin{pmatrix} -\boldsymbol{a} \\ -A_1' \end{pmatrix} & \begin{pmatrix} c \\ \boldsymbol{c}_1' \end{pmatrix} \\ \hline \boldsymbol{0} & \begin{pmatrix} \boldsymbol{a} \\ A_2' \end{pmatrix} & \begin{pmatrix} -c \\ -\boldsymbol{c}_2' \end{pmatrix} \\ \hline 0\cdots0 & 0\cdots0 & -1 \\ 0\cdots0 & 0\cdots0 & 1 \end{array}\right) \left(\begin{array}{c} \boldsymbol{x} \\ \hline \boldsymbol{y} \\ \hline \lambda \end{array}\right) \leq \left(\begin{array}{c} \begin{pmatrix} c \\ \boldsymbol{c}_1' \end{pmatrix} \\ \hline \begin{pmatrix} 0 \\ \boldsymbol{0} \end{pmatrix} \\ \hline 0 \\ 1 \end{array}\right)$$

Given the system above, it is easy to see that the first row of the top system can be added to the first row of the middle system, leading to $\boldsymbol{a} \cdot \boldsymbol{x} \leq c$ in the output system. Note that this inequality might be redundant.

In the next section, we shall use the above characterisation of the convex hull to deduce that certain inequalities can be omitted without affecting the result. Since the above representation is derived from the definition of the convex hull, the results below apply equally to the double description method.

4 Factoring Out Common Constraints

This section illustrates that certain common inequalities in the input of the join can be omitted by arguing based on the matrix characterisation of the convex hull calculation from the previous section.

4.1 Omitting Common Equalities and Stripes

By transforming a polyhedron into the normal form as described in Sect. 2, the affine space that the polyhedron is embedded in is made explicit in the set of its equalities. Furthermore, since the left-hand sides of the equality set is brought into diagonal form, it is easy to identify equalities that are common to two polyhedra. Given m common inequalities in two polyhedra $P_1, P_2 \in Poly_n$, it is possible to restrict the calculation of their convex hull to the $(n-m)$-dimensional subspace and to add the omitted common equalities back to the result without altering the outcome [4, p. 84]. The correctness of this approach can be shown by assuming that both polyhedra are normalized and contain the two opposing inequalities that correspond to the equality $\langle a_1, \ldots a_n\rangle \cdot \boldsymbol{x} = c$ where $a_1 \neq 0$, leading to the following system:

$$\left(\begin{array}{c|c|c} \begin{pmatrix} a_1 & \boldsymbol{a}' \\ -a_1 & -\boldsymbol{a}' \\ \boldsymbol{0} & A'_1 \end{pmatrix} & \begin{pmatrix} -a_1 & -\boldsymbol{a}' \\ a_1 & \boldsymbol{a}' \\ \boldsymbol{0} & -A'_1 \end{pmatrix} & \begin{pmatrix} c \\ -c \\ \boldsymbol{c}'_1 \end{pmatrix} \\ \hline \boldsymbol{0} & \begin{pmatrix} a_1 & \boldsymbol{a}' \\ -a_1 & -\boldsymbol{a}' \\ \boldsymbol{0} & A'_2 \end{pmatrix} & \begin{pmatrix} -c \\ c \\ -\boldsymbol{c}'_2 \end{pmatrix} \\ \hline 0\cdots0 & 0\cdots0 & -1 \\ 0\cdots0 & 0\cdots0 & 1 \end{array}\right) \begin{pmatrix} \boldsymbol{x} \\ \hline \boldsymbol{y} \\ \hline \lambda \end{pmatrix} \leq \left(\begin{array}{c} \begin{pmatrix} c \\ -c \\ \boldsymbol{c}'_1 \end{pmatrix} \\ \hline \boldsymbol{0} \\ \hline 0 \\ 1 \end{array}\right)$$

Here, $\boldsymbol{a}' = \langle a_2, \ldots a_n\rangle$ denotes the coefficients of the equality constraint without a_1 and A'_1, A'_2, c'_1, c'_2 denote the remaining coefficients of the input polyhedra. As before, the convex hull is defined by the inequalities that result from projecting out the variables $\boldsymbol{y}, \lambda$, i.e., the middle column. In particular, eliminating y_1 can only combine the first and second lines of each sub-system. As a result, the coefficients over $\boldsymbol{y}' = \langle y_2, \ldots y_n\rangle$ and λ cancel, yielding the following matrix:

$$\left(\begin{array}{c|c|c} \begin{pmatrix} a_1 & \boldsymbol{a}' \\ -a_1 & -\boldsymbol{a}' \end{pmatrix} & \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{pmatrix} & \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ \hline \boldsymbol{0} \quad A'_1 & -A'_1 & \boldsymbol{c}'_1 \\ \hline \boldsymbol{0} & A'_2 & -\boldsymbol{c}'_2 \\ \hline 0\cdots0 & 0\cdots0 & -1 \\ 0\cdots0 & 0\cdots0 & 1 \end{array}\right) \begin{pmatrix} \boldsymbol{x} \\ \hline \boldsymbol{y}' \\ \hline \lambda \end{pmatrix} \leq \left(\begin{array}{c} \begin{pmatrix} c \\ -c \end{pmatrix} \\ \hline \boldsymbol{c}'_1 \\ \hline \boldsymbol{0} \\ \hline 0 \\ 1 \end{array}\right)$$

Besides illustrating that x_1 in the final system is defined by the two opposing inequalities of the input polyhedra, the above system also shows that the elimination of the remaining variables $\boldsymbol{y}', \lambda$ cannot combine any other inequality with the two opposing inequalities since their coefficients over $\boldsymbol{y}'$ and λ are zero. Imbert showed that the result of Fourier-Motzkin projection is independent of the sequence in which variables are eliminated [9]. Thus, choosing to eliminate y_1 first does not affect the generality of the observation above. Hence, an equality can be omitted from the convex hull calculation without affecting the result. This observation lifts naturally to several equalities.

An interesting generalisation of the result above is that it is equally possible to omit a dimension that is defined by two opposing inequalities whose constants are not equal; a so-called stripe. For example, if both input polyhedra contain

$5 \leq x_1 + x_2 \leq 10$ and no other inequality in the input polyhedra mentions x_1, the two opposing inequalities $x_1 + x_2 \leq 10$ and $-x_1 - x_2 \leq -5$ will trigger the same operations as the opposing inequality constraint $10 \leq x_1 + x_2 \leq 10$. By removing the upper bound completely, it follows that a single inequality $\boldsymbol{a} \cdot \boldsymbol{x} \leq c$ that is common to the two input polyhedra can also be omitted if it contains a variable not present in any other inequality.

How can factoring out equalities and stripes improve performance? Suppose that common equalities in the two input polyhedra are not removed but that polyhedra are nevertheless normalised, such that the smallest variable in each equality does not occur in any other constraint. By using a projection algorithm that chooses the smallest variable first, common equalities or stripes vanish before any other variables are projected out. Hence, equalities and stripes incur only a minor administrative overhead. In order to assess the benefits for the double-description method, let V_i denote the set of vertices of the polyhedron $P_i = [\![A_i\boldsymbol{x} \leq \boldsymbol{c}_i]\!]$, $i = 1, 2$. Let $x_1 = \boldsymbol{a}' \cdot \langle x_2, \ldots x_n\rangle + c$ hold in P_i from which it follows that $v_1 = \boldsymbol{a}' \cdot \langle v_2, \ldots v_n\rangle + c$ holds for each vertex $\langle v_1, \ldots v_n\rangle \in V_i$, $i = 1, 2$. Omitting the equality from the convex hull calculation will generate the vertices $V_i' = \{\langle v_2, \ldots v_n\rangle \mid \langle v_1, \ldots v_n\rangle \in V_i\}$. In particular, note that $|V_i'| = |V_i|$ and, thus, factoring out the common equality constraint does not decrease the number of vertices. Since the running time of the double description method is dominated by the (often exponential) number of vertices in the generator representation, factoring out a common equality over n dimensions will only reduce the execution time by $1/n$. However, when storing polyhedra in normal form, identifying common equalities is cheap and likely to be worth the constant reduction.

A more promising approach for speeding up the convex hull calculation is the omission of common inequalities which is the topic of the next section.

4.2 Omitting Groups of Inequalities

The domain of convex polyhedra may relate any number of variables and thereby create states with very complex linear relationships. In the context of program analysis, linear relationships usually arise from the analysis of loops. Once a loop is analysed, any linear invariant that holds after the loop is passed on to the analysis of the next loop. Calculating the loop invariant of the next loop therefore operates on inequalities from the previous loop invariant and inequalities that stem from the analysis of the current loop. As a result, the inequalities inferred in the first loop may slow down the convex hull calculations in the second loop. It would therefore be beneficial to factor out sets of inequalities $A\boldsymbol{x}_a \leq \boldsymbol{c}$ that do not change and whose variables do not overlap with inequalities $B_i\boldsymbol{x}_b \leq \boldsymbol{c}_i$, $i = 1, 2$ that do change. The convex hull of such a system can be derived by projecting out $\boldsymbol{y} = \langle \boldsymbol{y}_a | \boldsymbol{y}_b \rangle$ and λ from the following matrix:

$$\left(\begin{array}{c|c|c} \begin{pmatrix} A & \mathbf{0} \\ \mathbf{0} & B_1 \end{pmatrix} & \begin{pmatrix} -A & \mathbf{0} \\ \mathbf{0} & -B_1 \end{pmatrix} & \begin{pmatrix} \boldsymbol{c} \\ \boldsymbol{c}_1 \end{pmatrix} \\ \hline \mathbf{0} & \begin{pmatrix} A & \mathbf{0} \\ \mathbf{0} & B_2 \end{pmatrix} & \begin{pmatrix} -\boldsymbol{c} \\ -\boldsymbol{c}_2 \end{pmatrix} \\ \hline 0\cdots 0 & 0\cdots 0 & -1 \\ 0\cdots 0 & 0\cdots 0 & 1 \end{array}\right) \left(\begin{array}{c} \boldsymbol{x}_a \\ \boldsymbol{x}_b \\ \hline \boldsymbol{y}_a \\ \boldsymbol{y}_b \\ \hline \lambda \end{array}\right) \leq \left(\begin{array}{c} \begin{pmatrix} \boldsymbol{c} \\ \boldsymbol{c}_1 \end{pmatrix} \\ \hline \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix} \\ \hline 0 \\ 1 \end{array}\right)$$

Projecting out $\boldsymbol{y}_a$ will combine the inequalities $A\boldsymbol{x}_a - A\boldsymbol{y}_a + \boldsymbol{c}\lambda \leq \boldsymbol{c}$ with $A\boldsymbol{y}_a - \boldsymbol{c}\lambda \leq \mathbf{0}$ while ignoring inequalities with non-zero entries over $\boldsymbol{y}_b$. Similarly, projecting out the remaining variables $\boldsymbol{y}_b$ will not affect the result of projecting out $\boldsymbol{y}_a$ since all these inequalities have zero coefficients over $\boldsymbol{y}_b$. Thus, the convex hull calculation can be separated into two problems:

$$\left(\begin{array}{c|c|c} A & -A & \boldsymbol{c} \\ \hline \mathbf{0} & A & -\boldsymbol{c} \\ \hline 0\cdots 0 & 0\cdots 0 & -1 \\ 0\cdots 0 & 0\cdots 0 & 1 \end{array}\right) \left(\begin{array}{c} \boldsymbol{x}_a \\ \hline \boldsymbol{y}_a \\ \hline \lambda \end{array}\right) \leq \left(\begin{array}{c} \boldsymbol{c} \\ \hline \mathbf{0} \\ \hline 0 \\ 1 \end{array}\right) \text{ and } \left(\begin{array}{c|c|c} B_1 & -B_1 & \boldsymbol{c}_1 \\ \hline \mathbf{0} & B_2 & -\boldsymbol{c}_2 \\ \hline 0\cdots 0 & 0\cdots 0 & -1 \\ 0\cdots 0 & 0\cdots 0 & 1 \end{array}\right) \left(\begin{array}{c} \boldsymbol{x}_b \\ \hline \boldsymbol{y}_b \\ \hline \lambda \end{array}\right) \leq \left(\begin{array}{c} \boldsymbol{c}_1 \\ \hline \mathbf{0} \\ \hline 0 \\ 1 \end{array}\right)$$

Since $P \sqcup P = P$, projecting out $\boldsymbol{y}_a, \lambda$ from the left system will yield only exactly the system $A\boldsymbol{x} \leq \boldsymbol{c}$ and besides that only redundant inequalities. Avoiding the calculation of this system is beneficial since the redundant inequalities generated during projection have to be removed by running a linear program to test each inequality. Furthermore, the overhead of handling these inequalities while projecting out $\boldsymbol{y}_b$ is avoided. With respect to the double description method, the situation is exacerbated: Suppose that calculating the vertices of the systems $A\boldsymbol{x}_a \leq \boldsymbol{c}$ and $B_i\boldsymbol{x}_b \leq c_i$, $i = 1, 2$ results in the vertices V^A and V_i^B, $i = 1, 2$, respectively. Thus, when identifying the system $A\boldsymbol{x}_a \leq \boldsymbol{c}$ as being equal in both polyhedra, the number of vertices required to calculate the output constraints is $|V_1^B| + |V_2^B|$. In contrast, generating vertices for each argument including those for $A\boldsymbol{x}_a \leq \boldsymbol{c}$ results in $V_i^{AB} = \{\langle v^A | v_i^B \rangle \mid v^A \in V^A \wedge v^B \in V_i^B\}$ and, hence, the whole frame contains $|V_1^{AB}| + |V_2^{AB}| = |V^A|(|V_1^B| + |V_2^B|)$ vertices. Thus, the generator representation of the system $A\boldsymbol{x}_a \leq \boldsymbol{c}$ only needs to consist of two or more vertices to make the omission of the common inequalities worthwhile.

Slightly orthogonal to omitting common constraints is the observation that a set of inequalities $A\boldsymbol{x}_a \leq c$ that only occurs in one polyhedron can be omitted, provided that it does not share any variables with other inequalities. For simplicity of the argument, suppose that this inequality set is present in the second polyhedron. In this case, the matrix takes on the following form:

$$\left(\begin{array}{c|c|c} \mathbf{0}\;\; B_1 & \mathbf{0}\;\; -B_1 & \boldsymbol{c}_1 \\ \hline \mathbf{0} & \begin{pmatrix} A & \mathbf{0} \\ \mathbf{0} & B_2 \end{pmatrix} & \begin{pmatrix} -\boldsymbol{c} \\ -\boldsymbol{c}_2 \end{pmatrix} \\ \hline 0\cdots 0 & 0\cdots 0 & -1 \\ 0\cdots 0 & 0\cdots 0 & 1 \end{array}\right) \left(\begin{array}{c} \boldsymbol{x}_a \\ \boldsymbol{x}_b \\ \hline \boldsymbol{y}_a \\ \boldsymbol{y}_b \\ \hline \lambda \end{array}\right) \leq \left(\begin{array}{c} \boldsymbol{c}_1 \\ \hline \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix} \\ \hline 0 \\ 1 \end{array}\right)$$

Projecting out $\boldsymbol{y}_a$ removes all coefficients of A without combining any rows containing variables of $\boldsymbol{y}_b$. Thus, $A\boldsymbol{x}_a \leq \boldsymbol{c}$ has no effect on the final outcome and can therefore be omitted from the convex hull calculation. In case $A\boldsymbol{x}_a \leq \boldsymbol{c}$ is not omitted from the system, the overhead of calculating the convex hull via projection is equivalent to projecting out all variables from $A\boldsymbol{x}_a \leq \boldsymbol{c}$ which results in tautologous inequalities which are all discarded. The overhead in the double-description method is to compute the vertices for $A\boldsymbol{x}_a \leq \boldsymbol{c}, B_2\boldsymbol{x}_b \leq \boldsymbol{c}_2$ rather than for $B_2\boldsymbol{x}_b \leq \boldsymbol{c}_2$ which is similar to the case of common inequality sets. Hence, omitting $A\boldsymbol{x}_a \leq \boldsymbol{c}$ is always worthwhile.

The ability to exploit the benefits of a cheaper convex hull hinges on how quickly the above inequality sets can be found. Identifying sets of inequalities that do not share variables with other inequalities can be performed in near-linear time by using a union-find data structure [6]: For each inequality ι in either polyhedron, force all variables that occur in ι with non-zero coefficient to be in the same equivalence class. In a second, linear pass, partition the inequalities in each polyhedron according to the equivalence classes. Equivalence classes that have no inequalities associated with them in at least one of the polyhedra can be discarded. For all other equivalence classes, the inequalities of each polyhedron are sorted using some total order on variable indices and coefficients. Equality of two inequality sets can now be detected in linear time. Each equal set can be directly moved to the result set. The remaining inequalities are passed on to the convex hull algorithm.

Given an efficient preprocessing of inequalities, the next section empirically assesses the benefits of this preprocessing to the calculation of the convex hull.

5 Experimental Evaluation

We assess the impact of the proposed factoring by calculating the convex hull via the double-description method and via the projection method on two benchmarks suites. In particular, we generate one C++ and one Haskell program that calculates all convex hulls in each benchmark. The C++ program calls the Parma Polyhedra Library (PPL) [2], version 0.10, in order to calculate the convex hull via the double description method. The result is converted back into a minimal constraint representation. The Haskell program calculates the convex hull via the projection method as described in Sect. 3. The times reported do not include the parsing nor the translation into normal form. We do not report the time for normalization as we assume that the detection of equalities is done during the meet operation (i.e. when inequalities from conditionals of the program are added). In particular, when the meet operation performs a minimization of the constraint set, detecting equalities can be done on-the-fly [12].

The conversion from and to the constraint representation may seem unfair on the double-description method since no conversion has to take place if one or both inputs are already in generator form. However, as every loop body normally includes some form of linear test which requires the constraint representation, the double description method converts back and forth between the two representations at least once for every join point and, hence, we deem the assumption that the input and output are in constraint form to be realistic.

Benchmark: "PerfectClub"

factoring	double-descr.		double-descr.†		projection		dims		ineqs	
none	1.68s	100%	2.51s	100%	5.24s	100%	8.60	100%	10.45	100%
asym	1.42s	85%	2.18s	87%	4.94s	92%	7.25	84%	9.45	90%
equalities	1.06s	63%	1.50s	60%	4.63s	88%	5.63	65%	6.66	64%
inequalities	0.93s	55%	1.35s	54%	3.72s	71%	4.53	53%	5.48	52%
stripes	0.91s	54%	1.33s	53%	3.72s	71%	4.49	52%	5.43	52%

Benchmark: "Spec 95"

factoring	double-descr.		double-descr.†		projection		dims		ineqs	
none	1.01s	100%	1.37s	100%	5.31s	100%	9.12	100%	10.21	100%
asym	0.82s	81%	1.12s	82%	4.99s	94%	7.41	81%	9.04	89%
equalities	0.67s	66%	0.84s	61%	4.84s	91%	6.02	66%	6.68	65%
inequalities	0.54s	53%	0.71s	52%	3.98s	75%	4.40	48%	5.06	50%
stripes	0.53s	52%	0.70s	51%	3.94s	74%	4.36	48%	5.00	49%

†: each equality was passed to the library as a pair of opposing inequalities

Table 1
Running times of different factorings. The times were measured using single-threaded programs running on a 2.4 GHz Core 2 Duo computer under Fedora Linux.

The two benchmark suites were taken from the PIPS project [1] which analyses the relationships between loop indices in order to determine how a Fortran loop can be parallelized or vectorized automatically. While the measurements can only hint at how the factorings would speed up the convex hull calculation rather than the full analyser, the benchmarks are nevertheless representative as they present real program invariants from a wide variety of programs. The "PerfectClub" suite consists of 3867 convex hull problems ("inputs" for short) while the "Spec 95" suite consists of 2415 inputs. Of these, 4 and 10 samples were omitted from our benchmark since the projection method failed to terminate within 3 seconds due to the exact linear programming algorithm in the GLPK LP toolkit [11] entering an infinite loop. The remaining inputs require 4.28s ("PerfectClub") and 2.32s ("Spec 95") to be calculated using the double-description method. However, the benchmark suites contain 144 inputs which are empty after normalisation. Furthermore, when omitting asymmetric inequalities, that is, those that contain variables present only in one argument, the number of empty inputs rises to 715 in total. One possible explanation is that variables are added in the body of a loop so that the calculation of the convex hull at the loop head joins edges on which the set of live variables is different. When omitting empty constraint sets and asymmetric inequalities, the double-description method takes only 1.68s and 1.01s (row "none" in the "double-descr." column of Table 1). While this seems to be a strong argument in favor of normalization, some of these speed-ups could be obtained by identifying (and removing) dead program variables at the loop-head. These constraint sets represent the reference input to the convex hull algorithms.

The row labelled "asym" in Table 1 shows the running times when omitting inequalities that contain variables not present in the other system. The remaining rows show the running times when omitting more and more *common* constraints. We found that the Parma Polyhedra Library does not automatically combine op-

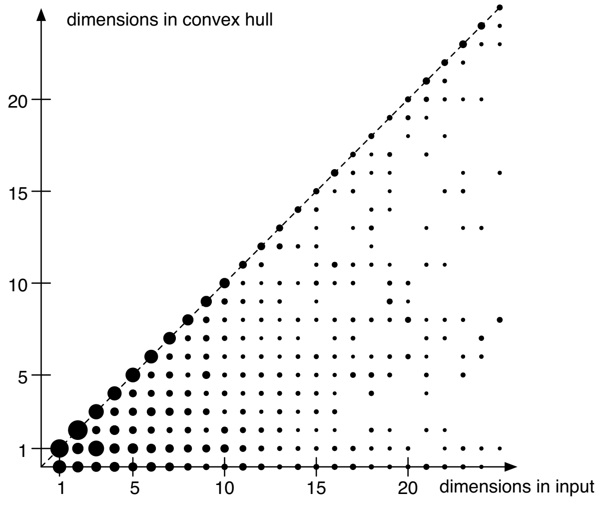

Fig. 3. Dimensions in the input without asymmetric inequalities vs. dimensions after factoring out common constraints. The area of each point is proportional to the number of samples. Not shown are 2.3% of all samples with dimensions between 26 and 88.

posing inequalities to equalities, thereby probably generating twice as many vertices than a single equality would, for each equality. Inserting equalities as equality constraints resulted in a speedup of about 24% . While the gain of omitting common equalities is now smaller (from 1.42s to 1.06s for "PerfectClub" and from 0.82s to 0.67s for "Spec 95"), it is still significant which stands in contrast to Halbwachs et al. [8] who observed no significant speedup when identifying equalities on the generator representation, that is, after converting the constraint system to the (possibly exponential) double description method. The prediction that factoring out equalities is insignificant for the projection method is confirmed by an improvement of less than 7% for "PerfectClub" (from 4.94s to 4.63s) and 4% for "Spec 95" (from 4.99s to 4.84s). Omitting common inequality sets is worthwhile for both algorithms, although the double description method benefits more. The reason for this difference could lie in the exponential growth of the vertex set in the double description method whereas the projection method merely has to remove the additional redundancies generated from projecting out the common inequalities. Overall, the proposed factorings speed up the convex hull algorithms by up to 50%. The columns "dims" and "ineqs" show the average number of variables and inequalities, respectively, in the inputs.

Observe that all considered convex hull problems do not lead to an exponential blow up in either the projection or double-description method since these calculations were aborted by a timeout in the original analyser. This leads to a surprisingly linear correlation between the number of inequalities and dimensions on the one hand and the running time on the other hand.

Figure 3 depicts how many variables were factored between column "asym" and "stripes" for both benchmarks together. It shows that most convex hull calculations are performed on low-dimensional polyhedra. Furthermore, it seems that the effectiveness of factoring is widely varying.

6 Conclusion

We proposed to store polyhedra in normal form which allows us to identify common constraints that can be omitted during a join operation. We demonstrated that omitting these common constraints can speed up the join operation. Future work should address if and how other operations may benefit from the normalized presentation. For instance, widening reduces to an intersection of the constraint sets if the sets of equalities of the two normalized input polyhedra are identical [4].

The author wishes to thank Duong Nguyen Que for making the benchmark suite available; also Liqian Chen and Antoine Miné for useful discussions and Enea Zaffanella for his insightful comments on an earlier version of the paper.

References

[1] Ancourt, C., F. Coelho, F. Irigoin and R. Keryell, *A Linear Algebra Framework for Static High Performance Fortran Code Distribution*, Scientific Programming **6** (1997), pp. 3–27.

[2] Bagnara, R., P. M. Hill and E. Zaffanella, *Not Necessarily Closed Convex Polyhedra and the Double Description Method*, Formal Aspects of Computing **17** (2005), pp. 222–257.

[3] Balas, E., *Disjunctive Programming and a Hierarchy of Relaxations for Discrete Optimization Problems*, J. on Algebraic and Discrete Methods **6** (1985), pp. 466–486.

[4] Benoy, F., "Polyhedral Domains for Abstract Interpretation in Logic Programming," Ph.D. thesis, Computing Lab., University of Kent, Canterbury, UK (2002).

[5] Cousot, P. and N. Halbwachs, *Automatic Discovery of Linear Constraints among Variables of a Program*, in: *Principles of Programming Languages* (1978), pp. 84–97.

[6] Galil, Z. and G. F. Italiano, *Data structures and algorithms for disjoint set union problems*, ACM Comput. Surv. **23** (1991), pp. 319–344.

[7] Gulwani, S., T. Lev-Ami and M.Sagiv, *A Combination Framework for Tracking Partition Sizes*, in: *Principles of Programming Languages* (2009).

[8] Halbwachs, N., D. Merchat and L. Gonnord, *Some ways to reduce the space dimension in polyhedra computations*, Form. Methods Syst. Des. **29** (2006), pp. 79–95.

[9] Imbert, J.-L., *Fourier's Elimination: Which to Choose?*, in: *Principles and Practice of Constraint Programming*, 1993, pp. 117–129.

[10] Lassez, J.-L. and K. McAloon, *A Canonical Form for Generalized Linear Constraints*, Journal of Symbolic Computation **13** (1993), pp. 1–24.

[11] Makhorin, A., *GLPK (GNU Linear Programming Kit)* (2008), version 4.32.
URL http://www.gnu.org/software/glpk/

[12] Refalo, P., *Approaches to the Incremental Detection of Implicit Equalities with the Revised Simplex Method*, in: *Principles of Declarative Programming*, LNCS **1490** (1998), pp. 481–496.

[13] Simon, A., "Value-Range Analysis of C Programs," ISBN 978-1-84800-016-2, Springer, 2008.

[14] Simon, A. and A. King, *Analyzing String Buffers in C*, in: H. Kirchner and C. Ringeissen, editors, *Algebraic Methodology and Software Technology*, LNCS **2422** (2002), pp. 365–379.

[15] Simon, A. and A. King, *Exploiting Sparsity in Polyhedral Analysis*, in: C. Hankin and I. Siveroni, editors, *Static Analysis Symposium*, LNCS **3672** (2005), pp. 336–351.