Closure Algorithms for Domains with Two Variables Per Inequality

City University London, Technical Report TR/2009/DOC/01

Jacob M. Howe[1] and Andy King[2]

[1] Department of Computing, City University, London EC1V OHB
[2] Portcullis Computer Security Limited, Pinner, HA5 2EX, UK***

Abstract. Weakly relational numeric domains express restricted classes of linear inequalities that strike a balance between what can be described and what can be efficiently computed. Such domains often restrict their attention of TVPI constraints which are systems of constraints where each constraint involves, at most, two variables. This technical report addresses the problem of deriving an incremental version of the closure operation. In this operation, a new constraint is added to a system that is already closed, and the computational problem is how to efficiently close the augmented system.

1 Introduction

One important thread of research in program analysis is finding the sweet-point in the design space for numeric domains. The research problem is to balance the expressiveness of a domain against the complexity and computational overhead of deploying it. The domain operations for general polyhedra [4] are of exponential complexity, irrespective of how they are computed: Chandru et al. [2] showed that eliminating variables from a system of inequalities can increase the number of inequalities exponentially; Benoy et al. [1] showed that polytopes (bounded polyhedra) exist whose convex hull is exponential in the number of inequalities defining the input polytopes. The class of so-called weakly-relational domains can be considered as a response to this. Octagons [7] possess cubic domain operations because the class of polyhedra they can represent are restricted to systems of inequalities of at most two variables where the coefficients are -1, 0 or 1. Other weakly-relational domains whose operations reside in low complexity classes are pentagons [6], two variable per inequality (TVPI) constraints [12], bounded differences [5] and octahedrons [3].

Another practical issue that is too frequently overlooked is the conceptual complexity, as opposed to the algorithmic complexity, of the domain operations. Flaws have been discovered in numeric domain operations that have been closely

*** Andy King is on secondment from the Computing Laboratory, University of Kent, Canterbury, CT2 7NF, UK scrutinised. This problem is exacerbated by the way domain operations are deployed in fixpoint calculations, and combined with other domains in reduced product constructions, which can make isolating the source of a bug difficult.

The most subtle domain operation for systems of TVPI constraints is incremental closure [10]. A system of TVPI constraints is closed if whenever a TVPI constraint c is entailed by a system I, then c is also entailed by the TVPI sub-system $\{c' \in I \mid vars(c') \subseteq vars(c)\}$. Given a closed TVPI system, incremental closure adds a new TVPI constraint and, if necessary other constraints as well, so as to ensure that the augmented system is closed. Closing a system I amounts to adding resultants [8]. For instance, if $x - y \leq 0 \in I$ and $y - z \leq 1 \in I$ then the two constraints can be combined to give the resultant $x - z \leq 1$. Since the resultant is entailed by I then if the sub-system $\{c \in I \mid vars(c) \subseteq \{x, z\}\}$ does not already entail it, then the resultant needs to be added to I. This process is iterated until no new resultants need to be added. The computational problem is putting is a bound on the number of iterations.

An incremental closure algorithm has been presented elsewhere [10], complete with a sketched correctness proof [10, page 159]. However, the proof makes unjustified assumptions about the effect of adding a new constraint. It is not that the algorithm is fundamentally flawed; rather that any correctness argument must consider the role of unary constraints. To illustrate why unary constraints are important consider the set of inequalities $I = \{-x+z \leq 0, -z+2x \leq 0\}$ and the problem of closing the augmented system $I_0 = I \cup \{x-y \leq 0\}$. Since $z-y \leq 0$ is entailed by I_0 and yet I_0 does not contain any constraints over y and z, it follows that I_0 must be extended with $z - y \leq 0$ to give $I_1 = I_0 \cup \{z - y \leq 0\}$. Now observe that I_1 entails $2x - y \leq 0$ yet $\{c \in I_1 \mid vars(c) \subseteq \{x, y\}\} = \{x - y \leq 0\}$ does not entail $2x - y \leq 0$. Thus I_1 is augmented to give $I_2 = I_1 \cup \{2x - y \leq 0\}$. But I_2 entails $2z - y \leq 0$ which is not entailed by $\{c \in I_2 \mid vars(c) \subseteq \{y, z\}\}$ so yet another constraint that must be added. I_0 can be extended ad infinitum in this way without achieving closure.

This problem stems from I itself since this system is, in fact, not closed even through it involves just two variables. In particular I entails the unary constraints $x \leq 0$ and $z \leq 0$ neither of which are entailed by their respective sub-systems of I. The constraints $2x - y \leq 0$ and $2z - y$ are actually redundant when the unary constraints are included in I. In fact, if $I' = I \cup \{x \leq 0, z \leq 0\}$ then the augmented system $I'_0 = I' \cup \{x - y \leq 0\}$ can be closed by merely adding the single resultant $z - y \leq 0$. The conclusion is that any argument addressing the correctness of an incremental closure algorithm must consider unary constraints. More generally, any correctness argument must consider how unary constraints (or binary constraints for that matter) combine with resultants to render unnecessary the derivation of further resultants.

To this end, this technical report presents a series of lemmata that lead to a result, a proposition, that stipulates exactly how a closed system of TVPI constraints needs to be updated when a new constraint added. This provides a rigorous foundation for an incremental closure algorithms for TVPI constraints, or even sub-classes of these constraints.

2 Preliminaries

The study commences with the class of two variable per inequality constraint [8, 12], denoted TVPI, that are defined over a given set of variables X:

Definition 1. $\mathsf{TVPI} = \{ax + by \leq d \mid x, y \in X \wedge a, b, d \in \mathbb{Q}\}$

This class of inequalities is more expressive than octagons [7], and possesses the property that each subset $S \subseteq \mathsf{TVPI}$ is closed under variable elimination. For example, the variable y can be eliminated from the system $S = \{x - 2y \leq 5, 3y + z \leq 7, 5y - u \leq 0\}$ by combining pairs of inequality with opposing signs for y. This yields the projection $\{3x+2z \leq 29, -2u+5x \leq 25\}$, which is indeed a system of TVPI inequalities. Since variable elimination can be used for deciding satisfiability, it is natural to ask whether there is a more tractable sub-class of TVPI that retains this property. One such class is given below:

Definition 2. $\mathsf{Log} = \{ax+by \leq d \mid x, y \in X \wedge a, b \in \{-2^n, 0, 2^n \mid n \in \mathbb{Z}\} \wedge d \in \mathbb{Q}\}$

Log is a strict subset of TVPI because (apart from 0) the absolute value of coefficients are restrained to be powers of two. It therefore suffices to represent the logarithm of an absolute value, rather than the value itself. This representational property is hinted at by the name *logahedra*, that we shall henceforth use to refer to these constraints. As with TVPI inequalities, a logahedral inequality can be binary, that is, involve two variables, when $a \neq 0$ and $b \neq 0$; or be unary, when either $a = 0$ or $b = 0$ (but not both); or be constant when $a = b = 0$. A dense representation for the binary case can be achieved by observing that $ax+by \leq d$ can be expressed as $x + b/ay \leq d/a$ if $a > 0$ and $-x - b/ay \leq -d/a$ otherwise. Therefore to represent $ax + by \leq d$ it is sufficient to distinguish the variables x and y, represent the signs of a and b (as two bits) and then either represent $\lg |b/a|$ and d/a or $\lg |b/a|$ and $-d/a$. A unary inequality such as $ax \leq d$ can be expressed as $x \leq d/a$ if $a > 0$ and $-x \leq -d/a$ otherwise, and therefore it is not even necessary to represent a logarithm.

Unlike TVPI, that can be alternatively be defined with $a, b, d \in \mathbb{Z}$, the constant d is required to be rational. To see this, observe that a variable y can be eliminated from a logahedral system S by again combining pairs of inequalities whose signs oppose on their y coefficients. Thus if $S = \{x - 2y \leq 5, 4y + z \leq 7, y - u \leq 5\}$ then the projection is $\{2x + z \leq 17, x - 2u \leq 15\}$ which is logahedral. Rational constants arise when eliminating y from a system such as $\{2x - 4y \leq 3, 4y + x \leq -1\}$ which yields the single inequality $3x \leq 2$. This, in itself, is not logahedral but the constraint can be equivalently expressed as $x \leq 2/3$ which is logahedral.

This leads to the concept of semantic equivalence. We write $3x \leq 2 \equiv x \leq 2/3$ to express that one constraint is merely a multiple of the other. More generally, equivalence is formulated in terms of the entailment relation. Given two systems $S, S' \subseteq \mathsf{TVPI}$, S entails S', denoted $S \models S'$, if any assignment that satisfies S is also satisfies S'. For instance, $S \models S'$ where $S = \{x + 2y \leq 7, y \leq 2\}$ and $S' = \{x \leq 4\}$ since every assignment to x and y that satisfies S also satisfies S'.

The converse is not true since the assignment $\{x \mapsto 4, y \mapsto 3\}$ satisfies S' but not S. Equivalence is defined $S \equiv S'$ iff $S \models S'$ and $S' \models S$. Thus $\equiv$ is symmetric even though $\models$ is not.

The action of combining inequalities, or computing resultants to use the terminology of Nelson [8], is formalised below:

Definition 3. If $c = ax + by \leq d$, $c' = a'x + b'z \leq d'$ and $a \cdot a' < 0$ then

$$\mathsf{result}(c, c', x) = |a'|by + |a|b'z \leq |a'|d + |a|d'$$

otherwise $\mathsf{result}(c, c', x) = \bot$.

Note that it is necessary to stipulate which variable is eliminated because a single pair of inequalities may possess two resultants, as is illustrated by the pair $c = x + y \leq 1$ and $c' = -2x - 3y \leq 1$ for which $\mathsf{result}(c, c', x) = -y \leq 3$ and $\mathsf{result}(c, c', y) = x \leq 4$. The resultant operator lifts to sets of inequalities by:

Definition 4. If $C_1, C_2 \subseteq \mathsf{TVPI}$ then

$$\mathsf{result}(C_1, C_2) = \left\{ c \;\middle|\; \begin{array}{l} c_1 \in C_1 \wedge c_2 \in C_2 \ \ \wedge x \in vars(c_1) \cap vars(c_2) \wedge \\ c = \mathsf{result}(c_1, c_2, x) \wedge c \neq \bot \end{array} \right\}$$

For brevity, we define $\mathsf{result}(c, C) = \mathsf{result}(\{c\}, C)$ and $\mathsf{result}(C, c) = \mathsf{result}(C, \{c\})$ where $c \in \mathsf{TVPI}$ and $C \subseteq \mathsf{TVPI}$.

Another fundamental operator is syntactic projection, denoted π_Y for some $Y \subseteq X$, which is defined $\pi_Y(S) = \{c \in S \mid vars(c) \subseteq Y\}$ where $S \subseteq \mathsf{TVPI}$ and $vars(c)$ is the set of variables occurring in c. (In a non-closed system syntactic projection will possibly loose information.) The set $vars(c)$ contains either 2, 1 or 0 variables depending on whether c is binary, unary or constant. If $Y = \{x, y\}$ then the syntactic projection $\pi_Y(S)$ yields a planar system. A planar system over Y can be filtered to remove any redundant inequalities. To this end, we assume the existence of an operator $\mathsf{filter}_Y(S) = S'$ which satisfies the three conditions that $S' \subseteq S$, $S' \equiv S$ and $S'' \not\equiv S$ for all $S'' \subset S'$. Such an operator can be constructed surprisingly straightforwardly [10], but rather than reproduce the algorithm we illustrate the underlying ideas with an example:

Example 1. Suppose $Y = \{x, y\}$ and the set S consists of the eight inequalities that are illustrated in figure 1. Suppose that the inequalities are sorted relatively by angle (which inequality is first does not matter since the filtering technique is cyclic). One consistent ordering is denoted by the labels $A, \ldots, G$. Henceforth the labels will be used to refer to inequalities themselves, thus $\langle A, B, \ldots, G\rangle$ is interpreted as a sequence of inequalities. The filtering algorithm proceeds by checking whether the first inequality is entailed by the second and last inequalities in the sequence; such a check resides in $O(1)$. If entailed, then the first inequality is removed and the remaining sub-sequence is rotated left. Otherwise, the whole sequence is rotated left. Filtering stops when the number of successive steps that do not remove an inequality reaches m where m is the number of inequalities remaining in the sequence. The filtering steps required for the eight inequalities

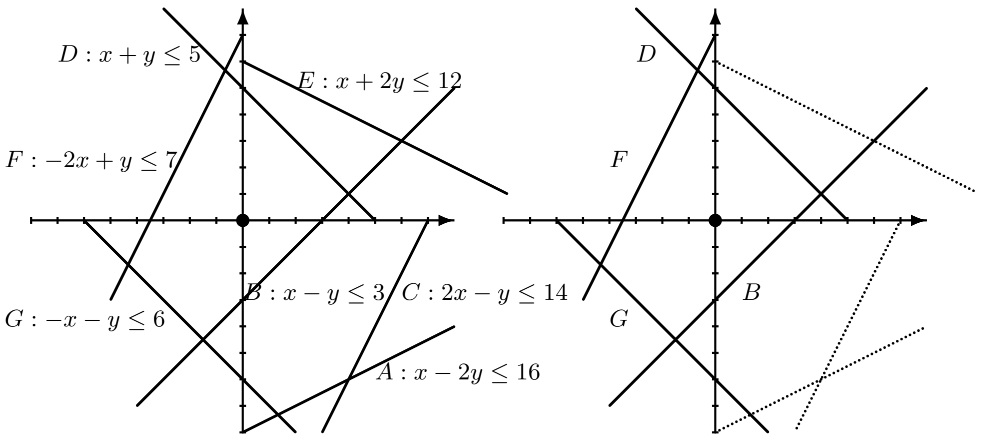

Fig. 1. Filtering out (removing) redundant planar inequalities

sequence	*check*	*action*
$\langle A, B, C, D, E, F, G\rangle$	$B \wedge G \models A\surd$	remove (first) then rotate (left)
$\langle G, B, C, D, E, F\rangle$	$B \wedge F \models G\times$	rotate
$\langle B, C, D, E, F, G\rangle$	$C \wedge G \models B\times$	rotate
$\langle C, D, E, F, G, B\rangle$	$D \wedge B \models C\surd$	remove then rotate
$\langle B, D, E, F, G\rangle$	$D \wedge G \models B\times$	rotate
$\langle D, E, F, G, B\rangle$	$E \wedge B \models D\times$	rotate
$\langle E, F, G, B, D\rangle$	$F \wedge D \models E\surd$	remove then rotate
$\langle D, F, G, B\rangle$	$F \wedge B \models D\times$	rotate
$\langle F, G, B, D\rangle$	$G \wedge D \models F\times$	rotate
$\langle G, B, D, F\rangle$	$B \wedge F \models G\times$	rotate
$\langle B, D, F, G\rangle$	$D \wedge G \models B\times$	rotate
$\langle D, F, G, B\rangle$		stop since 4 steps since removal

Fig. 2. Filtering eight inequalities

given in the table in figure 2. If $|S| = n$, then ordering the inequalities by angle is $O(n \lg n)$. Although the stopping criteria can actually be relaxed, the filtering algorithm as it stands requires just $O(n)$ steps, and thus the overall complexity is $O(n \lg n)$.

Finally, with filter_Y in place for any $|Y| = 2$, it is possible to filter a complete system $I \subseteq \mathsf{TVPI}$ by computing $\mathsf{filter}(I) = \cup\{\mathsf{filter}_Y(\pi_Y(I)) \mid Y \subseteq X \wedge |Y| = 2\}$.

3 Closure

Closing a set of inequalities $I \subseteq \mathsf{TVPI}$ [12] amounts to repeatedly augmenting I with inequalities $\mathsf{result}(I, I)$ until an I' is obtained such that no further (non-redundant) inequalities can be added to $\pi_Y(I)$ for any $|Y| = 2$. Nelson [8] used closure as a way of deciding whether a given $I \subseteq \mathsf{TVPI}$ is satisfiable. In the context of program analysis, where a domain must come equipped with a merge and entailment (fixpoint) check operation, closure provides a way of lifting planar merge (convex hull) and entailment check operations to systems of two variables constraints. For instance, to compute the merge of $I_1, I_2 \subseteq \mathsf{TVPI}$, denoted $I_1 \sqcup I_2$, it is sufficient to close I_1 and I_2 to obtain I'_1 and I'_2 respectively, and then put $I_1 \sqcup I_2 = \cup\{\pi_Y(I'_1) \sqcup \pi_Y(I'_2) \mid |Y| = 2\}$ [12]. It is not difficult to adapt this technique to logahedra by relaxing each planar system $M = \pi_Y(I'_1) \sqcup \pi_Y(I'_2)$ to a planar logahedron (which can be achieved in $O(|M|)$ time).

Since the closure operator for Log is essentially the same as that for TVPI one would expect that implementing a logahedral library would merely be an engineering task. However, it turns out that closure for Log is cubic, whereas TVPI does not even come with a (sound) polynomial guarantee. Moreover, reimplementing incremental closure for Log uncovered some overly strong assumptions on how a closed system I can be updated by $\mathsf{result}(I, I)$ [10]. Repairing this problem led to a thorough examination of incremental closure and the novel correctness arguments reported in the following sections.

3.1 Full closure

Definition 5. The (full closure) operator $\mathsf{close} : \wp(\mathsf{TVPI}) \to \wp(\mathsf{TVPI})$ is defined:

$$\mathsf{close}(I) = \cup_{i=0} I_i \text{ where } I_0 = I \text{ and } I_{i+1} = \mathsf{filter}(I_i \cup \mathsf{result}(I_i, I_i))$$

Nelson [8] used a divide and conquer argument to bound the number of iterations that need be computed before stability is achieved:

Lemma 1. $\mathsf{close}(I) = I_m$ where $m = \lceil \lg(|X|) \rceil$ and I_m is defined as above.

This result becomes more intruiguing when the domain is that of logahedra with bounded coefficients. Closure can be calculated in a semi-naive fashion by defining $I_0 = I$ and $\delta_0 = I$ and computing $I_{i+1} = \mathsf{filter}(I_i \cup \mathsf{result}(\delta_i, I_i))$ and $\delta_{i+1} = I_{i+1} \setminus I_i$ for $i \in [0, m-1]$. Since $|\cup_{i=0}^{m} \delta_i| = O(|X|^2)$ it follows that the cumulative running time of $\mathsf{result}(\delta_i, I_i)$ is $O(|X|^3)$. Since each invocation of filter resides in $O(|X|^2)$ and it is called $m-1$ times, it follows that the running time of closure is $O(|X|^3)$. Hence, like octagons, logahedra (with bounded coefficients) come with a (full) closure operation that is cubic.

3.2 Incremental closure

This section presents a rigorous derivation of an incremental algorithm. To this end, we present new results that show how a resultant, that is derived through a cascade of resultant derivations, can be derived in a less convoluted way.

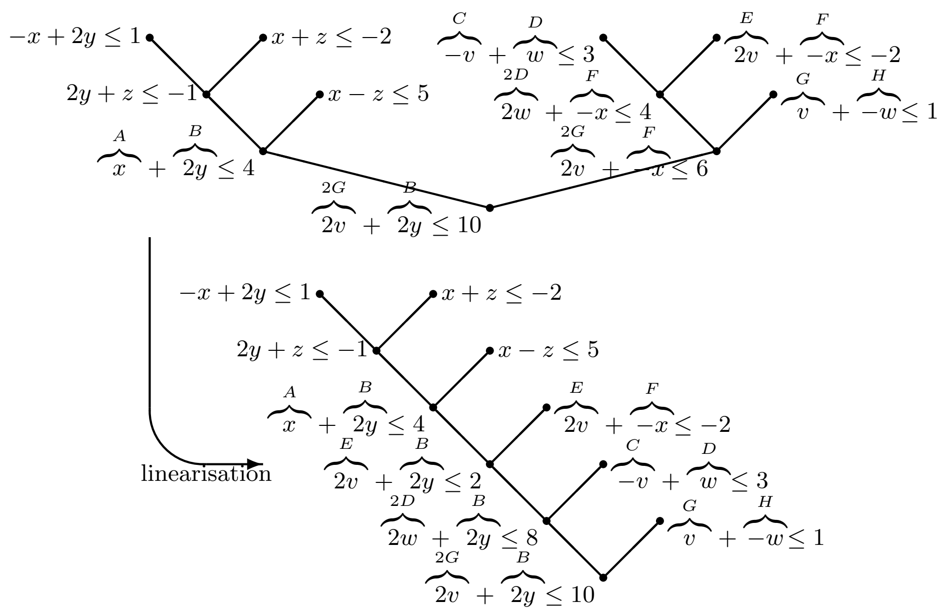
$-x + 2y \leq 1$
$x + z \leq -2$
$2y + z \leq -1$
$x - z \leq 5$
$\overbrace{x}^{A} + \overbrace{2y}^{B} \leq 4$
$\overbrace{-v}^{C} + \overbrace{w}^{D} \leq 3$
$\overbrace{2w}^{2D} + \overbrace{-x}^{F} \leq 4$
$\overbrace{2v}^{2G} + \overbrace{-x}^{F} \leq 6$
$\overbrace{2v}^{E} + \overbrace{-x}^{F} \leq -2$
$\overbrace{v}^{G} + \overbrace{-w}^{H} \leq 1$
$\overbrace{2v}^{2G} + \overbrace{2y}^{B} \leq 10$
linearisation
$\overbrace{2v}^{E} + \overbrace{2y}^{B} \leq 2$
$\overbrace{2w}^{2D} + \overbrace{2y}^{B} \leq 8$

Fig. 3. Linearisation of a tree composed of two chains into a single chain

3.3 Linearisation of a tree

Lemma 2. Suppose $c'_0, c_0, \ldots, c_n \in \mathsf{TVPI}$ and $d'_0, d_0, \ldots, d_m \in \mathsf{TVPI}$. Suppose too that $c'_{i+1} \in \mathsf{result}(c'_i, c_i)$ for $i \in [1, n]$ and $d'_{j+1} \in \mathsf{result}(d'_j, d_j)$ for $j \in [1, m]$. If $c \in \mathsf{result}(c'_{n+1}, d'_{m+1})$ then $\langle d'_0, d_0, \ldots, d_m \rangle$ can be reordered to obtain a sequence $\langle e'_0, e_0, \ldots, e_m \rangle$ such that $c = e'_{m+1}$ where $e''_0 \in \mathsf{result}(c'_{n+1}, e'_0)$ and $e''_{j+1} \in \mathsf{result}(e''_j, e_j)$ for $j \in [1, m]$.

The proof of this result centres on an algorithm that incrementally constructs $\langle e'_0, e_0, \ldots, e_m \rangle$ from $\langle d'_0, d_0, \ldots, d_m \rangle$. The construction is not difficult in itself, but the formal presentation is complicated by the need to label the individual terms that occur in the inequalities. Therefore, for clarity, the proof construction is illustrated with an example. The proof itself follows from this.

Example 2. Figure 3 illustrates a derivation tree whose two branches consist of chains. In this example, $n = 1$, $m = 1$ and

$$\begin{array}{llll} c'_0 = & -x + 2y \leq 1 & d'_0 = & -v + w \leq 3 \\ c_0 = & x + z \leq -2 & d_0 = & 2v - x \leq -2 \\ c_1 = & x - z \leq 5 & d_1 = & v - w \leq 1 \end{array}$$

This instance of the problem amounts to constructing $\langle e'_0, e_0, e_1 \rangle$ from $\langle d'_0, d_0, d_1 \rangle$. To this end, the terms in $c'_2 = x + 2y \leq 4$, d'_0, d_0 and d_1 are given unique labels as shown in the figure. A resultant that is derived from two labelled inequalities has its terms labelled with the labels inherited from the corresponding terms in the two inequalities. For example, the resultant $2w - x \leq 4$ adopts the label D for the $2w$ term and F for the $-x$ term. Iterating this scheme, all terms in all inequalities down the right branch of the tree can be labelled.

The construction now proceeds from the inequality $c'_2 = x + 2y \leq 4$ in the left branch, whose terms are labelled A and B. The construction in three steps:

– The A term of c'_2 does not occur in $c = 2v + 2y \leq 10$ and thus needs to be eliminated. In the right branch of the tree, A is eliminated by the F term of $2v - x \leq 6$ that first occurs in d_0. Therefore put $e'_0 = d_0$ whence $e''_0 = \mathsf{result}(e'_0, c'_2, x) = 2v + 2y \leq 2$ whose terms are labelled E and B.
– The E term of e'_1 does not occur in c and hence needs to be eliminated. In the right branch of the tree, E is eliminated by the C term in $-v + w \leq 3$ that first occurs in d'_0. Therefore put $e_0 = d'_0$ whence $e''_1 = \mathsf{result}(e''_0, e_0, v) = 2w + 2y \leq 8$ whose terms are labelled D and B.
– The D term of e'_2 does not occur in c and hence needs to be eliminated. In the right branch of the tree, D is eliminated by the H term in $v - w \leq 1$ that first occurs in d_1. Therefore put $e_1 = d_1$ whence $e''_2 = \mathsf{result}(e''_1, e_1, w) = 2v + 2y \leq 10$ whose terms are labelled G and B.

Then $\langle e'_0, e_0, e_1 \rangle = \langle d_0, d'_0, d_1 \rangle$ and observe that $c = e''_2$. More generally, it follows that $c = e''_{m+1}$ because, by the construction, a term is eliminated by the same inequality d_0, d'_0, d_1 in both derivations, which is also similarly scaled.

3.4 Compaction of a chain

The above result is of little value by itself. However, when strengthened with further lemmata that place constraints on what can be derived in a linear chain of derivations, the results together place strong constraints on how non-redundant inequalities can derived when a new inequality is added to a closed system. The first of these results that concern chains, explain the futility of combining the new inequality c'_0 with three inequalities drawn from the closed system:

Lemma 3. Let $c'_0 \in \mathsf{TVPI}$ and $c_0, c_1, c_2 \in I \subseteq \mathsf{TVPI}$ where $\mathsf{close}(I) = I$. Then there exists $d_0, d_1 \in I$ such that

$$\mathsf{result}(\mathsf{result}(\mathsf{result}(c'_0, c_0), c_1), c_2) \subseteq \mathsf{result}(\mathsf{result}(c'_0, d_0), d_1)$$

Example 3. Suppose $c'_0 = x + y \leq 1$ and $c_0, c_1, c_2 \in I \subseteq \mathsf{TVPI}$ where $\mathsf{close}(I) = I$ and

$$c_0 = -2u - x \leq 1 \quad c_1 = u - 5v \leq 1 \qquad c_2 = 3v - 2z \leq 0$$

Then

$$\begin{aligned} c'_1 &= -2u + y \leq 2 \in \mathsf{result}(c'_0, c_0) \\ c'_2 &= y - 10v \leq 4 \in \mathsf{result}(c'_1, c_1) \\ c'_3 &= 3y - 20z \leq 12 \in \mathsf{result}(c'_2, c_2) \end{aligned}$$

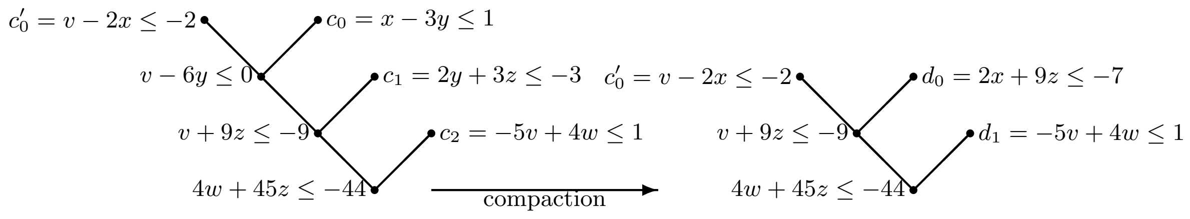

Fig. 4. Compaction of a single chain

Put $d_0 = c_0$ and $d_1 = \mathsf{result}(c_1, c_2, v) = 3u - 10z \le 3 \in I$ since $\mathsf{close}(I) = I$. Then observe

$$\begin{array}{c} c'_1 = d'_1 = -2u + y \le 2 \in \mathsf{result}(c'_0, d_0) \\ c'_3 = d'_2 = 3y - 20z \le 12 \in \mathsf{result}(c'_1, d_1) \end{array}$$

Example 4. Suppose $c'_0 = v - 2x \le -2$ and $c_0, c_1, c_2 \in I \subseteq \mathsf{TVPI}$ where $\mathsf{close}(I) = I$ and

$$c_0 = x - 3y \le 1 \qquad c_1 = 2y + 3z \le -3 \qquad c_2 = -5v + 4w \le 1$$

Then

$$\begin{array}{c} c'_1 = v - 6y \le 0 \in \mathsf{result}(c'_0, c_0) \\ c'_2 = v + 9z \le -9 \in \mathsf{result}(v - 6y \le 0, c_1) \\ c'_3 = 4w + 45z \le -44 \in \mathsf{result}(v + 9z \le -9, c_3) \end{array}$$

Put $d_0 = \mathsf{result}(c_0, c_1, y) = 2x + 9z \le -7$ whence $d_0 \in I$ and define $d_1 = c_2$. Then observe

$$\begin{array}{c} v + 9z \le -9 \in \mathsf{result}(c'_0, d_0) \\ 4w + 45z \le -44 \in \mathsf{result}(v + 9z \le -9, d_1) \end{array}$$

Note that the above lemma cannot be simplified to assert that if $c'_0 \in \mathsf{TVPI}$ and $c_0, c_1 \in I$ where $\mathsf{close}(I) = I$ then there exists $d_0 \in I$ such that

$$\mathsf{result}(\mathsf{result}(c'_0, c_0), c_1) \subseteq \mathsf{result}(c'_0, d_0)$$

The following example illustrates why this is not so:

Example 5. Let $I = \{c_0, c_1\}$ where

$$c'_0 = u + 2x \le 0 \qquad c_0 = -x + 4y \le 0 \qquad c_1 = -u + 8v \le 0$$

Observe that $\mathsf{close}(I) = I$ and

$$\mathsf{result}(c'_0, c_0) = \{u + 8y \le 0\} \qquad \mathsf{result}(c'_0, c_1) = \{x + 4v \le 0\}$$

But $v + y \le 0 \in \mathsf{result}(\mathsf{result}(c'_0, c_0), c_1)$ and therefore the simplified version of the lemma does not hold.

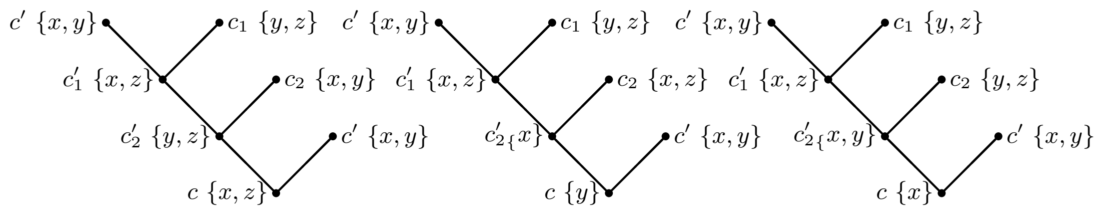

Fig. 5. Resultant combination sequences involving c', c_1, c_2

From the perspective of closing a system when a new constraint c' is added, it is also fruitless to recombine c' with any inequality that results from a chain of derivations that emanates from c'. By the previous lemma, it is sufficient to consider chains that start at c' with at most two intermediate inequalities; chains with more intermediates can be collapsed down by the previous result.

Lemma 4. Let $c', c_1 \in \mathsf{TVPI}$. If $c \in \mathsf{result}(\mathsf{result}(c', c_1), c')$ then there exists $c'' \in \mathsf{result}(c', c_1)$ such that $c \equiv c''$.

Lemma 5. Let $c', c_1, c_2 \in \mathsf{TVPI}$. If $c \in \mathsf{result}(\mathsf{result}(\mathsf{result}(c', c_1), c_2), c')$ then one of the following hold:

1. $\pi_{vars(c)}(\mathsf{result}(c', c_1) \cup \mathsf{result}(c', c_2)) \models c$
2. $c \in \mathsf{result}(\mathsf{result}(c', \mathsf{result}(c_1, c_2)), c')$
3. $c \in \mathsf{result}(c', \mathsf{result}(c_1, c_2))$

Example 6. The three cases of the lemma are illustrated in Figure 5. The cases differ in the variables that arise in the given inequalities c', c_1 and c_2; the intermediate inequalities $c'_1 \in \mathsf{result}(c', c_1)$ and $c'_2 \in \mathsf{result}(c'_1, c_2)$; and the final inequality $c \in \mathsf{result}(c'_2, c')$. The derivation sequence in the left of the figure is exemplified by:

$$c' = -x - 7y \leq 1 \qquad c_1 = 2y + z \leq 2 \qquad c_2 = 5x + 3y \leq -3$$

and

$$c'_1 = -2x + 7z \leq 16 \qquad c'_2 = 6y + 35z \leq 74 \qquad c = -6x + 245z \leq 524$$

But observe $c'_1 \in \mathsf{result}(c', c_1)$ and $32x \leq -18 = \mathsf{result}(c', c_2, y) \in \mathsf{result}(c', c_2)$. Scaling the inequalities c'_1 and $\mathsf{result}(c', c_2, y)$ by 35 and 64 respectively, and then summing, we deduce that $\{c'_1, \mathsf{result}(c', c_2, y)\} \models -6x + 245z \leq 524 = c$. Moreover, observe that $vars(c'_1) = \{x, z\}$ and $vars(\mathsf{result}(c', c_2, y)) = \{x\}$, therefore the first case of the lemma holds.

Example 7. The derivation sequence in the middle diagram of the figure 5 is illustrated by:

$$c' = 9x - 3y \leq 2 \qquad c_1 = 5y + 7z \leq -3 \qquad c_2 = -4x - 2z \leq 11$$

Then $c'_1 \in \mathsf{result}(c', c_1)$ and $c'_2 \in \mathsf{result}(c'_1, c_2)$ where

$$c'_1 = 45x + 21z \leq 1 \qquad c'_2 = 6x \leq 233$$

This circumstance is addressed in the third case of the lemma since:

$$\begin{aligned} \mathsf{result}(c_1, c_2, z) &= -28x + 10y \leq 71 \\ \mathsf{result}(c', \mathsf{result}(c_1, c_2, z), y) &= 6x \leq 233 = c'_2 \end{aligned}$$

Therefore $c'_2 \in \mathsf{result}(c', \mathsf{result}(c_1, c_2))$ and hence $c \in \mathsf{result}(\mathsf{result}(c', \mathsf{result}(c_1, c_2)), c')$ as prescribed in the second case of the lemma.

Example 8. The derivation sequence in the right diagram of the figure 5 is illustrated thus:

$$c' = x - 5y \leq 7 \qquad c_1 = 3y + 4z \leq -2 \qquad c_2 = 2y - z \leq 1$$

and $c'_1 \in \mathsf{result}(c', c_1)$ and $c'_2 \in \mathsf{result}(c'_1, c_2)$ where

$$c'_1 = 3x + 20z \leq 11 \qquad c'_2 = 3x + 40y \leq 31 \qquad c = 11x \leq 87$$

Then

$$\begin{aligned} \mathsf{result}(c_1, c_2, z) &= 11y \leq 2 \\ \mathsf{result}(c', \mathsf{result}(c_1, c_2, z), y) &= 11x \leq 87 = c \end{aligned}$$

Therefore $c \in \mathsf{result}(c', \mathsf{result}(c_1, c_2))$ and the third case of the lemma applies.

The findings can now be summarised in a single statement. The result follows by using the lemmata to show that the result of any derivation tree, where the leaf inequalities are drawn from $I \cup \{c'\}$, can be collapsed into a chain that coincides with one of the three cases. The strength of the result is that *only* these simple chains need be considered when computing incremental closure.

Proposition 1. Suppose $c \in \mathsf{TVPI}$ and $I \subseteq \mathsf{TVPI}$ such that $\mathsf{close}(I) = I$. If $c \in \mathsf{close}(I \cup \{c'\})$ then one of the following hold:

- $c \in I \cup \{c'\}$
- $c \in \mathsf{result}(c', c_0)$ where $c_0 \in I$
- $c \in \mathsf{result}(\mathsf{result}(c', c_0), c_1)$ where $c_0, c_1 \in I$

4 Conclusion

This technical report has shown how the key domain operation over TVPI constraints — incremental closure — can been derived by a systematic examination of the shapes that derivation trees can that when a new inequality is added to a closed system.

Acknowledgements The authors thank Phil Charles, Tim Hopkins and Stefan Kahrs for useful discussions. We particularly thank Axel Simon, whose book [10] motived much of this study. The work was supported by EPSRC projects EP/E033105/1 and EP/E034519/1. Andy King's secondment to Portcullis is supported by a Royal Society Industrial Fellowship. Jacob Howe would like to thank the University of St Andrews for supporting him as a visiting scholar.

References

1. F. Benoy, A. King, and F. Mesnard. Computing Convex Hulls with a Linear Solver. *Theory and Practice of Logic Programming*, 5(1&2):259–271, 2005.
2. V. Chandru, C. Lassez, and J.-L. Lassez. Qualitative Theorem Proving in Linear Constraints. In *International Symposium on Artificial Intelligence and Mathematics*, 2000. `http://www.seas.upenn.edu/~chandru/amai.ps`.
3. R. Clarisó and J. Cortadella. The Octahedron Abstract Domain. *Science of Computer Programming*, 64:115–139, 2007.
4. P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Symposium on Principles of Programming Languages*, pages 84–97. ACM Press, 1978.
5. K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi. Efficient verification of real-time systems: compact data structure and state-space reduction. In *IEEE Real-Time Systems Symposium*, pages 14–24. IEEE Computer Society, 1997.
6. F. Logozzo and M. Fähndrich. Pentagons: a weakly relational abstract domain for the efficient validation of array accesses. In *ACM Symposium on Applied Computing*, pages 184–188. ACM Press, 2008.
7. A. Miné. The Octagon Abstract Domain. *Higher-Order and Symbolic Computation*, 19(1):31–100, 2006.
8. C. G. Nelson. An $n^{\lg n}$ Algorithm for the Two-Variable-Per-Constraint Linear Programming Satisfiability Problem. Technical Report STAN-CS-78-689, Stanford University, Computer Science Department, 1978.
9. H. Raynaud. Sur L'enveloppe Convexe des Nuages de Points Aléatoires dans $\mathbb{R}^n$. *Journal of Applied Probability*, 7(1):35–48, 1970.
10. A. Simon. *Value-Range Analysis of C Programs: Towards Proving the Absence of Buffer Overflow Vulnerabilities.* Springer-Verlag, 2008.
11. A. Simon and A. King. Exploiting Sparsity in Polyhedral Analysis. In C. Hankin, editor, *Static Analysis Symposium*, volume 3672 of *Lecture Notes in Computer Science*, pages 336–351. Springer-Verlag, 2005.
12. A. Simon, A. King, and J. M. Howe. Two Variables per Linear Inequality as an Abstract Domain. In M. Leuschel, editor, *Proceedings of Logic Based Program Development and Transformation*, volume 2664 of *Lecture Notes in Computer Science*, pages 71–89. Springer-Verlag, 2002.

A Proof Appendix

Proof (for lemma 3). Let $c'_{i+1} \in \mathsf{result}(c'_i, c_i)$ for $i \in [0, 2]$. Moreover let $c'_1 = \mathsf{result}(c'_0, c_0, x)$, $c'_2 = \mathsf{result}(c'_1, c_1, u)$ and $c'_3 = \mathsf{result}(c'_2, c_2, v)$.

- Suppose $vars(c'_0) = \{x, y\}$ where $x \neq y$.
  - Suppose $u \notin \{x, y\}$ and $v \notin \{u, y\}$. Since $x \neq y$ then $vars(c'_1) = \{u, y\}$. Since $v \in vars(c'_2)$ and $v \notin \{u, y\}$ then $vars(c_1) = \{u, v\}$. Since $v \in vars(c_2)$ then suppose $vars(c_2) \subseteq \{v, z\}$. Moreover, suppose

    $$c_1 = au + bv \leq d \qquad c'_1 = a'u + ey \leq g \qquad c_2 = b'v + fz \leq h$$

    Since $c'_2 \neq \perp$ it follows that $a.a' < 0$ and since $c'_3 \neq \perp$ it follows that $b.b' < 0$. Then $c'_3 = |b'||a|ey + |a'||b|fz \leq |b'||a'|d + |b'||a|g + |a'||b|h = \mathsf{result}(\mathsf{result}(c_1, c_2, v), c'_1, u)$.
  - Suppose $u \notin \{x, y\}$ and $v = u$. Impossible since $u \notin vars(c'_2)$.
  - Suppose $u \notin \{x, y\}$ and $v = y$. Since $x \neq y$ then $vars(c'_1) = \{u, y\}$. Since $u \notin \{x, y\}$ then $vars(c_0) = \{u, x\}$. Since $u \in vars(c_1)$ then suppose $vars(c_1) \subseteq \{u, z\}$. Moreover, suppose

    $$c_0 = au + bx \leq d \qquad c'_0 = b'x + ey \leq g \qquad c_1 = a'u + fz \leq h$$

    Then $a.a' < 0$ and $b.b' < 0$ hence $c'_1 = \mathsf{result}(\mathsf{result}(c_0, c_1, u), c'_0, x)$.
  - Suppose $u = x$. Impossible since $x \notin vars(c'_1)$.
  - Suppose $u = y$, $v \neq y$ and $v \in vars(c_1)$. Then $vars(c_1) = \{v, y\}$. Since $y \in vars(c'_1)$ then suppose $vars(c'_1) \subseteq \{y, z\}$. Likewise, since $v \in vars(c_2)$ then suppose $vars(c_2) \subseteq \{v, w\}$. Moreover, suppose

    $$c_1 = av + by \leq d \qquad c'_1 = b'y + ez \leq g \qquad c_2 = a'v + fw \leq h$$

    Then $a.a' < 0$ and $b.b' < 0$ hence $c'_3 = \mathsf{result}(\mathsf{result}(c_1, c_2, x), c'_1, y)$.
  - Suppose $u = y$, $v \neq y$ and $v \in vars(c'_1)$. Then $v \in vars(c_0)$. But since $x \notin vars(c'_1)$ then it follows that $v \neq x$ and $vars(c_0) = \{v, x\}$. Since $y \in vars(c_1)$ suppose $vars(c_1) \subseteq \{y, z\}$ and likewise because $v \in vars(c_2)$ suppose $vars(c_2) \subseteq \{v, w\}$. Moreover, suppose

    $$\begin{array}{ll} c_0 = ax + by \leq d & c'_0 = ev + a'x \leq g \\ c_1 = b'y + fz \leq m & c_2 = e'v + hw \leq n \end{array}$$

    Then $b.b' < 0$, $e.e' < 0$ and $a.a' < 0$ hence $c'_3 = \mathsf{result}(\mathsf{result}(c'_0, \mathsf{result}(c_0, c_1, y), x), c_2, v)$.
  - Suppose $u = y$, $v \neq y$ and $v \notin vars(c_1)$ and $v \notin vars(c'_1)$. Then $v \notin vars(c'_2)$ which is impossible.
  - Suppose $u = y$ and $v = y$. Impossible since $y \notin vars(c'_2)$.
- Suppose $vars(c'_0) = \{x\}$. Since $x \notin vars(c'_1)$ then $u \neq x$. Since $u \in vars(c'_1)$ then $vars(c_0) = \{u, x\}$ hence $vars(c'_1) = \{u\}$. Since $v \in vars(c'_2)$ then $v \in vars(c_1)$ hence $vars(c_1) = \{u, v\}$. Then $c'_2 = \mathsf{result}(\mathsf{result}(c_0, c_1, u), c'_0, x)$.

Proof (for lemma 4). Suppose $c'_1 = \mathsf{result}(c', c_1, x)$ and $c = \mathsf{result}(c'_1, c', u)$ such that $c'_1 \neq \bot$ and $c \neq \bot$. Suppose $vars(c') = \{x\}$. Since $x \notin vars(c'_1)$ and because $c \neq \bot$ it follows that $u \in vars(c')$ such that $x \neq u$ which is a contradiction. Thus, henceforth, suppose $vars(c') = \{x, y\}$ for $x \neq y$. Moreover, suppose $c' = ax + by \leq d$. Since $x \in vars(c_1)$ suppose $vars(c_1) \subseteq \{x, z\}$. Moreover, suppose $c_1 = a'x + b'z \leq e$. Since $c'_1 \neq \bot$ then $a.a' < 0$. Then $c'_1 = |a'|by + |a|b'z \leq |a'|d + |a|e$. If $y \neq z$ then $|a'|b.b > 0$ which contradicts $c \in \mathsf{result}(c'_1, c')$. Thus $z = y$ and $u = y$ and therefore $c'_1 = (|a|b' + |a'|b)y \leq |a|e + |a'|d$. Since $c \in \mathsf{result}(c'_1, c')$ then $(|a'|b + |a|b').b < 0$.

- Suppose $b > 0$. Then $|a'|b + |a|b' < 0$ and thus $b' < 0$.
  - Suppose $a > 0$. Then $c'_1 = (ab' - a'b)y \leq ae - a'd$ and $c = a(a'b - ab')x \leq d(a'b - ab') + b(ae - a'd) = abe - ab'd = a(be - b'd)$. Furthermore, since $b.b' < 0$ then $\mathsf{result}(c', c_1, y) = (a'b - ab')x \leq (be - b'd) \equiv c$ as $a > 0$.
  - Suppose $a < 0$. Then $c'_1 = (a'b - ab')y \leq a'd - ae$ and $c = a(ab' - a'b)x \leq d(ab' - a'b) + b(a'd - be) = a(b'd - be)$. Moreover $\mathsf{result}(c', c_1, y) = (a'b - ab')x \leq (be - b'd) \equiv c$ as $a < 0$.
- Suppose $b < 0$. Then $|a'|b + |a|b' > 0$ and thus $b' > 0$.
  - Suppose $a > 0$. Then $c'_1 = (ab' - a'b)y \leq ae - a'd$ and $c = a(ab' - a'b)x \leq d(ab' - a'b) - b(ae - a'd) = ab'd - abe = a(b'd - be)$. Moreover $\mathsf{result}(c', c_1, y) = (ab' - a'b)x \leq (b'd - be) \equiv c$ as $a > 0$.
  - Suppose $a < 0$. Then $c'_1 = (a'b - ab')y \leq a'd - ae$ and $c = a(a'b - ab')x \leq d(a'b - ab') - b(a'd - ae) = a(be - b'd)$. Moreover $\mathsf{result}(c', c_1, y) = (ab' - a'b)x \leq (b'd - be) \equiv c$ as $a < 0$.

Proof (for lemma 5). Suppose $c'_1 = \mathsf{result}(c', c_1, y)$, $c'_2 = \mathsf{result}(c'_1, c_2, u)$ and $c = \mathsf{result}(c', c'_2, v)$. Suppose $vars(c') = \{y\}$. Then $vars(c'_1) = \{x, y\}$ for $x \neq y$. Thus $vars(c'_1) = \{x\}$. Since $vars(c') = \{y\}$, $y \in vars(c'_2)$. Thus $vars(c_1) = \{x, y\}$ whence $vars(c'_2) = \{y\}$. But $vars(c') = \{y\}$ thus $c = \bot$ which is a contradiction. Thus, henceforth, suppose that $vars(c') = \{x, y\}$ for $x \neq y$.

- Suppose $vars(c_1) = \{y, z\}$ and $vars(c_2) = \{x, y\}$ where $y \neq z$. Then $vars(c'_1) = \{x, z\}$, $vars(c'_2) = \{y, z\}$ and $vars(c) = \{x, z\}$. Suppose

  $$c' = ax + by \leq d \qquad c_1 = b'y + ez \leq f \qquad c_2 = a'x + b''y \leq g$$

  Suppose $a < 0$ and $b < 0$. Then $b' > 0$ and $ab' < 0$ whence $a' > 0$. Since $b < 0$ then $-ab'b'' > 0$ whence $b'' > 0$. Moreover $ab'' - a'b > 0$. Then

  $$\begin{aligned} c'_1 &= ab'x - bez \leq b'd - bf \\ c'_2 &= -ab'b''y - a'bez \leq a'b'd - a'bf - ab'g \\ c &= -a^2b'b''x + a'b^2ez \leq -a'bb'd + a'b^2f + abb'g - ab'b''d \\ \mathsf{result}(c', c_2, y) &= (ab'' - a'b)x \leq b''d - bg \equiv x \leq (b''d - bg)/(ab'' - a'b) \end{aligned}$$

  Scaling c'_1 by $-a'b > 0$ and $\mathsf{result}(c', c_2, y)$ by $-ab'(ab'' - a'b) > 0$, and then summing, we derive:

  $$-a^2b'b''x + a'b^2ez \leq ab'(bg - b''d) + a'b(bf - b'd) \equiv c$$

Thus $\{c'_1, \mathsf{result}(c', c_2, y)\} \models c$. Analogous arguments follow for $a > 0$ and $b < 0$; $a < 0$ and $b > 0$; and $a > 0$ and $b > 0$. Hence, in all four cases, case 1 of the lemma holds.

– Suppose $vars(c_1) = \{y, z\}$ and $vars(c_2) = \{x, z\}$ where $x \neq z$ and $y \neq z$. Then $vars(c'_1) = \{x, z\}$, $vars(c'_2) = \{x\}$ and $vars(c) = \{y\}$. Suppose

$$c' = ax + by \leq d \qquad c_1 = b'y + ez \leq f \qquad c_2 = a'x + e'z \leq g$$

Suppose $b < 0$ and $e' < 0$. Then $b' > 0$ and $-be > 0$ whence $e > 0$. Moreover $-b'e' > 0$. Then

$$\begin{aligned} c'_1 &= ab'x - bez \leq b'd - bf \\ c'_2 &= -(ab'e' + a'be)x \leq -beg - e'(b'd - bf) \\ \mathsf{result}(c_1, c_2, z) &= a'ex - b'e'y \leq eg - e'f \\ \mathsf{result}(c', \mathsf{result}(c_1, c_2, z), y) &= -(ab'e' + a'be)x \leq -b(eg - e'f) - b'e'd = c'_2 \end{aligned}$$

Analogous arguments follow for $b > 0$ and $e' < 0$; $b < 0$ and $e' > 0$; and $b > 0$ and $e' > 0$. Therefore, in all four cases, case 2 of the lemma holds.

– Suppose $vars(c_1) = \{y, z\}$ and $vars(c_2) = \{y, z\}$ where $y \neq z$. Then $vars(c'_1) = \{x, z\}$, $vars(c'_2) = \{x, y\}$ and $vars(c) = \{x\}$. Suppose

$$c' = ax + by \leq d \qquad c_1 = b'y + ez \leq f \qquad c_2 = b''y + e'z \leq g$$

Suppose $b < 0$ and $e' < 0$. Then $b' > 0$ and $-be > 0$ whence $e > 0$. Then $-bb''e > 0$ whence $b'' > 0$. Then

$$\begin{aligned} c'_1 &= ab'x - bez \leq b'd - bf \\ c'_2 &= -ab'e'x - bb''ey \leq -beg + e'(bf - b'd) \\ c &= a(b''e - b'e')x \leq -beg + be'f - b'de' + b''ed \\ \mathsf{result}(c_1, c_2, z) &= (b''e - b'e')y \leq eg - e'f \\ \mathsf{result}(c', \mathsf{result}(c_1, c_2, z), y) &= a(b''e - b'e')x \leq (b''e - b'e')d - b(eg - e'f) = c \end{aligned}$$

Analogous arguments follow for $b < 0$ and $e' > 0$; $b > 0$ and $e' < 0$; and $b > 0$ and $e' > 0$. Therefore, in all four cases, case 3 of the lemma holds.

– Suppose $vars(c_1) = \{y, z\}$ and $vars(c_2) = \{y, z\}$ where $y \neq z$. Then $vars(c'_1) = \{x, z\}$, $vars(c'_2) = \{x, y\}$ and $vars(c) = \{y\}$. Suppose

$$c' = ax + by \leq d \qquad c_1 = b'y + ez \leq f \qquad c_2 = b''y + e'z \leq g$$

Suppose $b < 0$ and $e' < 0$. Then $b' > 0$ and $-be > 0$ whence $e > 0$.

$$\begin{aligned} c'_1 &= ab'x - bez \leq b'd - bf \\ c'_2 &= -ab'e'x - bb''ey \leq -beg + e'(bf - b'd) \end{aligned}$$

But $a. - ab'e' > 0$ and thus $\mathsf{result}(c'_2, c', x) = \bot$ which is impossible. Analogous impossibility arguments follow for $b < 0$ and $e' > 0$; $b > 0$ and $e' < 0$; and $b > 0$ and $e' > 0$.