Approximating the Transitive Closure of a Boolean-Affine Relation

Paul Feautrier
ENS de Lyon, LIP, Compsys, INRIA

ABSTRACT

Boolean affine relations, which combine affine inequalities by boolean connectives are ubiquitous in all kind of static program analyzes. One of the crucial operations on such relations is transitive closure, which is closely related to the construction of loop inductive invariants. I present here a new over-approximation algorithm, which has the interesting property of being extendible for increased precision.

Keywords

Boolean-affine relation, non-convex polyhedra, transitive closure, static program analysis

1. MOTIVATION

In static program analysis, one has to construct, modify and inspect various kind of sets: sets of operations, data sets, dependences, parallel fronts ... These sets are usually enormous, are often dependent on large parameters, or even infinite. For example, a petaflop supercomputer generates 10^{15} floating points operations per second, and a digital TV processor may run for days and months without interruption. It follows that the relevant sets cannot be represented in extension – by listing their members – but in intention, by presenting them as the set of solutions of some system of constraints. Depending on the selected representation, the necessary operations, for instance intersection or emptiness test, may be easy or difficult, or even unfeasible. One such possibility is to use regular or context-free languages, but, since Post's correspondence problem is lurking not far away, one is soon faced with undecidable questions.

One of the most useful representation is by systems of affine inequalities, which define convex polyhedra; it has proved highly efficient, both for static analysis [3] and for automatic parallelization [4, 5]. However, recent research has generated a need for more flexible representations. A possible approach is to lift the convexity constraint, thus allowing the use of arbitrary boolean connectives, like disjunction and negation. The associated sets are *non-convex polyhedra*.

In many cases, one has to deal with relations, i.e. subsets of the artesian product $E \times E$ of a given universe, E. When the universe is $\mathbb{Z}^d$ or $\mathbb{Q}^d$, one can use boolean affine formulas to define boolean affine relations. One simply has to distinguish input and output variables. In program analysis, such a relation may formalize, for instance, the effect of an action on the program memory, input variables being associated to the state of memory *before* the action, and output variables to the state *after* the action. In this context, one does not need explicit parameters, which are simply variables which have equal input and output values. A common convention here is to use the same symbol for an input and the corresponding output variable, with a prime on the output.

Boolean affine formulas admit quantifier elimination, hence boolean affine relations are closed under composition. Closure under union, commutativity and distributivity are obvious. In fact boolean affine relations are an instance of the semi-ring structure [2]. The only missing operation is the transitive closure or Kleene star. Among other uses, this operation is necessary for constructing loop invariants. As another example, consider the problem of finding programs with communication-free parallelism. A possible solution is to construct the transitive closure of the dependence relation. Each connected component of the transitive closure is a communication-free process.

The difficulty is that the definition of the transitive closure involves quantification over sets, and hence cannot be expressed in first order logic. In other words, the transitive closure of a boolean affine relation may not be boolean affine. A trivial example is the relation

$$y' = y + x \wedge x' = x \wedge i' = i + 1$$

which defines multiplication. As a consequence, if one wants to stay in the boolean affine realm, one may have to be content with approximate results, which may be over- or under-approximations. In most cases, over-approximations are needed, and this paper concentrates on them. They can be used, for instance, for program verification, by showing that an error state is unreachable even when the reachable set is overestimated, or, in automatic parallelization, by showing that two statements are independent even when the dependence relation is overestimated. However, in other cases, under-approximation may be useful. For instance, when constructing communication-free processes, it is often

IMPACT 2012
Second International Workshop on Polyhedral Compilation Techniques
Jan 23, 2012, Paris, France
In conjunction with HiPEAC 2012.

http://impact.gforge.inria.fr/impact2012

the case that one obtains only one connected component. One must in that case use an under-approximation, the missing dependences being implemented as residual communications. This generalizes a proposal in [6, 7] and will be the subject of future work.

2. DEFINITIONS AND ELEMENTARY PROPERTIES

A base set E being given, a relation is a subset of the artesian product $E \times E$. The composition of two relations R and S is:

$$R \circ S = \lambda x, x'.\exists z \in E : R(x; z) \wedge S(z; x').$$

Composition is associative, monotone in both its arguments, distributive with respect to union, and has the identity relation, noted $I \equiv \lambda x, x'.x = x'$, as its unit.

In the kind of investigations I have in mind, E usually is $\mathbb{Z}^d$ or $\mathbb{Q}^d$ for some dimension d. An affine atom is an inequality $\sum_{i=1}^{d} a_i x_i + a_0 \geq 0$ or $\sum_{i=1}^{d} a_i x_i + a_0 > 0$ where the a_i are integers and the x_i are integers or rationals. A boolean affine formula is a combination of affine atoms by the usual boolean connectives $\wedge$, $\vee$, $\neg$.

A relation R is reflexive iff $\forall x \in E : R(x; x)$ and is transitive iff $\forall x, y, z \in E : R(x; y) \wedge R(y; z) \Rightarrow R(x; z)$. The transitive closure of R, noted R^*, is the smallest reflexive and transitive relation that includes R. It is easy to prove that:

$$R^* = I \cup R^1 \cup R^2 \cup \ldots \cup R^k \cup \ldots,$$

where $R^1 = R$, and $R^{k+1} = R \circ R^k$. It is clear that this formula is usually not an effective method for computing R^*. However, there are cases where, for some n, $R^n = \emptyset$ or $R^n = R^{2n}$, in which cases the summation above is finite.

Another interesting relation is

$$R^+ = R^1 \cup R^2 \cup \ldots \cup R^k \cup \ldots, \tag{1}$$

the strict transitive closure of R. R^+ is a useful intermediate result. For instance, let $\mathcal{D}(R)$ and $\mathcal{C}(R)$ be the domain and codomain, respectively, of R. It is easy to prove, by induction on k, that $R^k \subseteq \mathcal{D}(R) \times \mathcal{C}(R)$, and, as a consequence, that $R^+ \subseteq \mathcal{D}(R) \times \mathcal{C}(R))$. It follows that if R' is an over-approximation to R^+, then $R' \cap \mathcal{D}(R) \times \mathcal{C}(R))$ may be a more precise one. Such a strengthening will be tacitly applied to all subsequent results in this paper. On the other hand, since the domain and codomain of the identity are equal to the base set E, this construction is useless for a reflexive relation.

The construction of the transitive closure is much more difficult when the variables are integral instead of rational: for instance, quantifier elimination on integers is more complex than on rationals, and the application of Farkas lemma to integer sets necessitates the construction of the integer hull of the antecedent, which may be of exponential complexity. It is therefore usual to over-approximate the integral closure by the rational closure. Again, this approximation can be improved by the following device. Let us assume that the relation on integers is included in a lattice Λ (a subgroup of $\mathbb{Z}^d$, eventually $\mathbb{Z}^d$ itself). The relation associated to Λ is $L_\Lambda \equiv \lambda x, x'.x' - x \in \Lambda$. L_Λ is obviously reflexive and transitive, and hence is its own transitive closure. Now, transitive closure is monotone:

$$R \subseteq S \Rightarrow R^* \subseteq S^* \tag{2}$$

from which follows:

$$(R \cap S)^* \subseteq R^* \cap S^*. \tag{3}$$

If $R \subseteq L_\Lambda$ and if R' is a rational over-approximation of R^* then:

$$R^* \equiv (R \cap L_\Lambda)^* \subseteq R^* \cap L_\Lambda^* \subseteq R' \cap L_\Lambda.$$

The construction of Λ from R is a well known problem, hence the above result allows one to first ignore integrality constraints and reinstate them at the end of the algorithm.

The approximation (3) may be used when one wants to split the subject formula into several parts to which different algorithms may be applied. A case in point is the separation of equations and inequations.

3. THE BASIC ALGORITHM

A boolean affine relation R being given, the problem is to find another reflexive and transitive boolean affine relation R' which includes R and is as small as possible. One may start by attempting to characterize reflexive and transitive relations – quasi-orders – in the abstract. The following results are classical, and their proofs are left to the reader:

LEMMA 1. *If R is reflexive and transitive, then*

$$\approx_R = \lambda x, y.R(x; y) \wedge R(y; x)$$

is an equivalence relation.

LEMMA 2. *If R is reflexive and transitive, then the quotient relation $R/\approx_R$ is a partial order.*

PROOF. Let $[x]$ be the equivalence class of x. The quotient relation is $[x] \prec [y] \equiv R(x; y)$. $\prec$ is well defined: if x' and y' are other members of $[x]$ and $[y]$, we have:

$$x' \approx_R x, R(x; y), y \approx_R y' \Rightarrow R(x'; y')$$

hence $[x'] \prec [y']$ by transitivity. Reflexivity and transitivity are obvious. It remain to prove antisymmetry. If $[x] \prec [y]$ and $[y] \prec [x]$, then $R(x; y)$ and $R(y; x)$, hence $x \approx_R y$ and $[x] = [y]$. □

THEOREM 3. *Any reflexive and transitive relation can be expressed in the form $R = \lambda x, y.f_R(x) \prec f_R(y)$, where f_R is the function that maps E on the equivalence classes of $\approx_R$, and $\prec$ is the partial order $R/\approx_R$.*

These results are well known when R is the path relation of a finite graph. The equivalence classes are the strongly connected components of the graph, and $\prec$ is the path relation of the quotient graph.

These facts can be exploited in the following way:

- Select a set U to label the equivalence classes of $\approx_R$, and an order on U. One may take for instance $U = \mathbb{Q}$ with $\prec$ as the ordinary order $\leq$.

- Select the shape of f. One may take for instance f a linear function from $\mathbb{Q}^d \to \mathbb{Q}$: $f(x) = \sum_{i=1}^{d} \alpha_i x_i$.
- Solve the constraint:

$$\forall x, y : R(x; y) \Rightarrow f(x) \leq f(y). \quad (4)$$

Since f is linear and R is boolean affine, this can be solved by a trivial extension of Farkas lemma [13].

Aside: a short introduction to Farkas lemma. Farkas lemma gives a necessary and sufficient condition for an affine function to be non negative inside a polyhedron. It can be seen as a special case of quantifier elimination, and also as a linearization tool. The presentation I give here is non standard.

THEOREM 4.

$(\forall x : Ax+b \geq 0 \Rightarrow c.x+d \geq 0) \equiv \exists \Lambda \geq 0 : \Lambda A = c \wedge d \geq \Lambda.b$

PROOF. The proof of sufficiency is trivial and is left to the reader. To prove necessity, observe first that there is nothing to prove if $Ax+b \geq 0$ is unfeasible. If not, the hypothesis is equivalent to the assertion that the system $Ax+b \geq 0 \wedge -cx-d > 0$ is unfeasible. This can be checked by the Fourier-Motzkin algorithm, which constructs a contradiction in the form of a positive vector Λ and a positive scalar λ such that $\Lambda A - \lambda c = 0$ and $\Lambda b - \lambda d \leq 0$. Furthermore, λ cannot be null as that would imply that $Ax + b \geq 0$ be unfeasible. Hence, one can set $\lambda = 1$ and the result follows. The positive components of vector Λ are called Farkas multipliers, $Ax + b \geq 0$ is the *antecedent* and $c.x+d \geq 0$ is the *consequent*. □

To solve (4), express R in disjunctive normal form (DNF), and apply Farkas to each disjunct. Let $Ax + A'x' + a \geq 0$ be one of the disjuncts of R, and write $f.x$ for $f(x)$. Application of Farkas lemma gives the system:

$$\Lambda A = -f,\ \Lambda A' = f,\ \Lambda a \leq 0, \quad (5)$$

where Λ is the vector of the Farkas multipliers. One collects one such system per disjunct, eliminates the Farkas multipliers, and obtains a system of linear constraints for the coefficients of f , $\alpha_i, i = 1, d$. It is easy to see that $f = 0$ is a solution, and that if f_1 and f_2 are solutions and λ is positive, then both λf_1 and $f_1 + f_2$ are solutions. Hence, considered as the components of a vector in $\mathbb{Q}^d$, the αs belong to a cone, the *F-cone*. In fact, it is enough to consider the set of rays of the F-cone, and the corresponding functions $f_k, k = 1, n$. All other vectors in the F-cone generate redundant inequalities. For instance, if f_1 and f_2 satisfy (4), it is clear that $f_1 + f_2$ also does. Since the Cartesian product of several orders is an order, one can take as a better approximation to R^* the relation $R' = \lambda x, x'. \bigwedge_{k=1}^{n} f_k(x) \leq f_k(x')$. If the F-cone contains a line (i.e., two opposite rays) then in the preceding formula, the comparator $\leq$ can be replaced by $=$. Here, U is $\mathbb{Q}^n$, and $\prec$ is $\leq^n$, the component-wise partial order on $\mathbb{Q}^n$.

The method has been implemented using my own Farkas library [8] and the PolyLib implementation of Chernikova's algorithm[1] for finding the rays of the F-cone.

Consider for example the relation $R(x, y, z; x', y', z') \equiv (y' = z \wedge z' = x \wedge x' = y)$, and take $f(x, y, z) = \alpha.x + \beta.y + \gamma.z$. The instance of (4) is:

$$\begin{array}{rcl} (y' = z \wedge z' = x \wedge x' = y) & \Rightarrow & \alpha.x + \beta.y + \gamma.z \\ & \leq & \alpha.x' + \beta.y' + \gamma.z'. \end{array}$$

By Farkas lemma, there exists multipliers λ, μ, ν such that:

$$\begin{array}{rcl} \lambda = \beta & , & \mu = \gamma \\ \nu = \alpha & , & -\lambda = -\gamma \\ -\mu = -\alpha & , & -\nu = -\beta. \end{array}$$

After elimination of the multipliers – which, in that case, are not constrained to be positive – the F-cone is defined by the system $\alpha = \beta, \beta = \gamma, \gamma = \alpha$. The F-cone has one line, whose direction vector is $(1, 1, 1)^T$. The f function is therefore $x + y + z$, and the approximate transitive closure is $x + y + z = x' + y' + z'$.

This result is not the most precise one, since R is such that $R^3 = I$, hence $R^* = I \cup R \cup R^2$. It is nevertheless a useful invariant.

4. A PIECEWISE AFFINE EXTENSION

Application of the above algorithm usually gives good results when R is convex (i.e., when its DNF has only one disjunct). An extension is necessary to obtain more precise results in the non convex case. One possibility is to use a piecewise affine function for f:

$$f(x) = \textbf{if } \sigma(x) \geq 0 \textbf{ then } g(x) \textbf{ else } h(x),$$

where σ, g, and h are affine functions of x:

$$\begin{array}{rcl} \sigma(x) & = & \sigma.x + \sigma_0, \\ g(s) & = & g.x + g_0, \\ h(x) & = & h.x + h_0. \end{array}$$

The hyperplane $\sigma(x) = 0$ splits the domain of R in two subsets, where the f function may have different shapes. This choice enlarges the solution space, and hence increases the probability of success.

Consider first one of the disjuncts of R, written $Ax + A'x' + a \geq 0$ as above. The corresponding version of constraint (4) is:

$$\begin{array}{rcl} Ax + A'x' + a \geq 0 & \Rightarrow & (\textbf{if } \sigma(x) \geq 0 \textbf{ then } g(x) \textbf{ else } h(x)) \quad (6) \\ & \leq & (\textbf{if } \sigma(x') \geq 0 \textbf{ then } g(x') \textbf{ else } h(x')). \end{array}$$

This can be expanded into four simpler constraints according to the sign of $\sigma(x)$ and $\sigma(x')$. For instance, the first constraint is:

$$Ax + A'x' + a \geq 0 \wedge \sigma(x) \geq 0 \wedge \sigma(x') \geq 0 \Rightarrow g(x) \leq g(x').$$

The Farkas algorithm can be applied to all four constraints, giving four linear systems in positive unknowns:

[1] `http://icps.u-strasbg.fr/polylib`

$$\begin{array}{rcl|rcl} \Lambda^1 A + \lambda^1\sigma = -g & | & \Lambda^1 A' + \mu^1\sigma = g & (7)\\ \Lambda^1 a + (\lambda^1 + \mu^1)\sigma_0 & \leq & 0 & (8)\\ \hline \Lambda^2 A + \lambda^2\sigma = -g & | & \Lambda^2 A' - \mu^2\sigma = h & (9)\\ \Lambda^2 a + (\lambda^2 - \mu^2)\sigma_0 & \leq & h_0 - g_0 & (10)\\ \hline \Lambda^3 A - \lambda^3\sigma = -h & | & \Lambda^3 A' + \mu^3\sigma = g & (11)\\ \Lambda^3 a + (\mu^3 - \lambda^3)\sigma_0 & \leq & g_0 - h_0 & (12)\\ \hline \Lambda^4 A - \lambda^4\sigma = -h & | & \Lambda^4 A' - \mu^4\sigma = h & (13)\\ \Lambda^4 a - (\lambda^4 + \mu^4)\sigma_0 & \leq & 0 & (14) \end{array}$$

and the solution proceeds as above. The only problem is in the choice of σ.

Observe that one can add the two equations of (7), giving $\Lambda^1(A + A') + (\lambda^1 + \mu^1)\sigma = 0$. Hence, either $\lambda^1 = \mu^1 = 0$, or σ belongs to the opposite of the cone generated by the rows of $A + A'$ – the characteristic cone of $A + A'$. In the same way, from the two equations of (13), one deduces that either $\lambda^4 = \mu^4 = 0$ or σ belongs to the characteristic cone of $A+A'$. If the characteristic cone has lines, then σ may belong to both the cone and its opposite; otherwise, we have to choose one of the solutions, but since σ and $-\sigma$ generate the same system of constraints, the choice is unimportant. The same reasoning can be applied to each disjunct of R, hence σ must belong to the intersection of the several characteristic cones. If this intersection reduces to the origin, one may ignore some of the disjuncts until a non trivial σ is found.

It remains to explain how to choose σ_0. Once σ is chosen, one can decide whether $\lambda^1 + \mu^1$ and $\lambda^4 + \mu^4$ are null or not. If both are null, the method reduces to the basic algorithm. They can be both non-zero only of the characteristic cone has lines, and if not, one of them is null and the other is not. Whatever the situation, the system of constraints is homogeneous in the Farkas multipliers, hence the non-null terms can be set to one, and the system becomes linear. The remaining Farkas multipliers can be eliminated, giving a system of constraint for σ_0.

As an example, consider a relation from [11]

$$R(x; x') \equiv (x < 100 \wedge x' = x + 1) \vee (x \geq 100 \wedge x' = 0). \quad (15)$$

Application of the basic algorithm gives only $R'(x; x') \equiv x' \leq 100$. Here, since x is a scalar, the choice of σ is simple: $\sigma(x) = x + \sigma_0$. Notice nevertheless that for the first disjunct, $A + A' = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}$ while for the second disjunct, $A + A' = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$, hence σ is in both characteristic cones. Since the second cone has a line, one can set both $\lambda^1 + \mu^1 = 1$ and $\lambda^4 + \mu^4 = 1$ in the second system of equations. The resulting instances of (8) and (14) are $-100\Lambda^1_1 + \sigma_0 \leq 0$ and $-100\Lambda^4_1 - \sigma_0 \leq 0$, which clearly implies $\sigma_0 = 0$.

Taking $\sigma(x) = x$, one obtain the exact result,

$$R^*(x, x') \equiv (x = x') \vee ((x' < 101) \wedge (x \leq x' \vee 0 \leq x')).$$

In this case, the only non-trivial pair of functions is $g(x) = x$ and $h(x) = 0$, giving as the "central part" of the transitive closure:

$$(\textbf{ if } x \leq 0 \textbf{ then } x \textbf{ else } 0) \leq (\textbf{ if } x' \leq 0 \textbf{ then } x' \textbf{ else } 0),$$

which, when restricted to the codomain $x' < 101$, simplifies to the above formula.

Here again, the method has been implemented using the same tools as above, the choice of σ being still the responsibility of the user. One must remark that the raw results of the algorithm are much more complicated than shown above, and that the simplifier of [9] has been used to automatically reduce them to manageable form.

An interesting fact is that the function $f(x) =$ **if** $x \leq 0$ **then** x **else** 0 is equal to $\min(x, 0)$. Whether this observation may conduct to another closure algorithm is a question for future work.

5. RELATED WORKS

The transitive closure problem for relations defined by affine constraints has a long history, which began with the seminal paper of Kelly et. al. [10]. In this work, the concept of d-form constraints, i.e. constraints involving only the difference $x' - x$ was introduced for the first time. This idea was extended by Ancourt et. al., who introduced the distance polyhedron, and implemented it in the PIPS parallelizer long before publication in [1]. A similar approach is proposed in [14] and implemented in the `isl` library.

The method of Ancourt et. al. can be compared to the basic algorithm of Sect. 3 as follows. Consider the system (5). Instead of eliminating Λ to obtain constraints on f, one can eliminate f, giving:

$$\Lambda(A + A') = 0,\ \Lambda a \leq 0. \quad (16)$$

This implies that the valid Λs belong to a cone, $\mathcal{C}$. For each ray of $\mathcal{C}$, r, one obtain a function $f(x) = r.A'x$ and one constraint in the transitive closure, $r.A'(x' - x) \geq 0$.

On the other hand, the distance polyhedron of R, ΔR is defined as: $\Delta R \equiv \{d \mid \exists x : Ax + A'(x + d) + a \geq 0\}$. Eliminating x, for instance by the Fourier-Motzkin algorithm, generates a positive matrix L such that $L(A + A') = 0$, and $\Delta R = \{d \mid LA'd + La \geq 0\}$. The difference $x' - x$ when $R^+(x; x')$ is true is a sum of vectors $d \in \Delta R$, hence, if La is negative, $LA'(x' - x) \geq -La$. A vector r such that $r(A + A') = 0$ and $r.a \leq 0$ is clearly a solution of (16), but the converse is not true. In cases where the base vectors are the same, Ancourt et. al. result is slightly more precise than that of the basic algorithm.

Sriram Sankaranarayanan work [12] is similar to the present one in its use of Farkas lemma for quantifier elimination, even if its concern is invariant construction rather than transitive closure. Its starting point is the fix-point equation $R \cup R \circ R^+ = R^+$. Application of Farkas lemma gives a non-linear system of equations, since the unknown, R^+, occurs both in the antecedent and the consequent. The authors then propose either algebraic quantifier elimination, or a set of rewrite rules to bring the system in linear form. In con-

trast, the present method stays in the linear domain while giving comparable results.

> The relation $(x' = x + 2y \wedge y' = 1 - y) \vee (x' = x+1 \wedge y' = y+2)$ from Example 1 in [12] is found to have transitive closure $x' + y' \geq x + y$, which directly gives the invariant $x + y \geq 0$.

6. CONCLUSION

It is clear that the present proposal is still very preliminary, and that much work is needed for its transformation into a reliable algorithm. I think that its main interest is that it is open: one may want to explore other choices for the function f, as has been done in the preceding section, or for the $\prec$ order, an obvious candidate being the lexicographic order. One may want to have several σs, although the complexity of the algorithm will increase exponentially with their number. In this connection, in the present implementation, the expansion of (6) into four parts is "wired" into the code. This must be improved when refactoring future versions. Another point is that the transitive closure problem admits many special cases, in which special purpose algorithms may give better results. Examples are cases where the infinite sum (1) terminates, cases where disjuncts commutes, and many others. How to recognize these cases and integrate them in the basic algorithm is left for future work.

7. REFERENCES

[1] Corinne Ancourt, Fabien Coelho, and François Irigoin. A Modular Static Analysis Approach to Affine Loop Invariants Detection. In *NSAD: Numerical and Symbolic Abstract Domains*, ENCTS, Perpignan, France, September 2010. Elsevier.

[2] Jean Berstel and Christophe Retenauer. *Rational Series and their Languages*. Springer-Verlag, 1988.

[3] Patrick Cousot and Nicolas Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Proceedings of the ACM POPL'78*, 1978.

[4] Paul Feautrier. Some efficient solutions to the affine scheduling problem, I, one dimensional time. *Int. J. of Parallel Programming*, 21(5):313–348, October 1992.

[5] Paul Feautrier. Some efficient solutions to the affine scheduling problem, II, multidimensional time. *Int. J. of Parallel Programming*, 21(6):389–420, December 1992.

[6] Paul Feautrier. Toward automatic distribution. *Parallel Processing Letters*, 4(3):233–244, 1994.

[7] Paul Feautrier. Automatic distribution of data and computations. Technical Report 2000/3, PRiSM, 2000.

[8] Paul Feautrier. Scalable and structured scheduling. *Int. J. of Parallel Programming*, 34(5):459–487, May 2006.

[9] Paul Feautrier. Simplification of Boolean Affine Formulas. Technical Report RR-7689, INRIA, July 2011.

[10] Wayne Kelly, William Pugh, Evan Rosser, and Tatiana Shpeisman. Transitive closure of infinite graphs and its applications. In C.-H. Huang, P. Saddayapan, U. Banerjee, D. Gelernter, A. Nicolau, and D. Padua, editors, *8th Language and Compilers for Parallel Computing Workshop*, LNCS 1033, pages 126–140. Springer, 1995.

[11] David Monniaux and Laure Gonnord. Using bounded model checking to focus fixpoint iterations. In *Static analysis (SAS'2011)*, 2011. To appear.

[12] Sriram Sankaranarayanan, Henny Sipma, and Zohar Manna. Constraint-based linear relation analysis. In *Static Analysis Symposium, SAS'2004*, LNCS 3148, pages 53–68. Springer-Verlag, 2004.

[13] A. Schrijver. *Theory of linear and integer programming*. Wiley, NewYork, 1986.

[14] Sven Verdoolaege, Albert Cohen, and Anna Beletska. Transitive closures of affine integer tuple relations and their overapproximations. In *SAS*, pages 216–232, 2011.