

# Numerical Domains for Software Analysis

EP/C520726/1 – Final Report

Patricia Hill, School of Computing, University of Leeds

## 1 Project Personnel

- Katy Dobson (Postgraduate Research Student),
- Pat Hill (Principal Investigator),
- Matthew Mundell (Computer Officer).

## 2 Summary

The aim of this small project was to study in more depth how to develop and apply a domain for capturing the distribution information about the values of numeric variable, clarify the underlying research issues. The intention was that this preliminary investigation will lead to the charting of the boundaries of, as yet, unexplored areas of research, but for which a fuller understanding is important for the development of automatic, practical tools for software analysis.

In summary, as a result of this year’s work, we have

- published a technical report [2] describing the grid domain and its possible applications in program analysis and verification;
- submitted two papers for presenting and publishing at international conferences based on this report;
- included as part of the latest release of the Parma Polyhedra Library (PPL) [6, 9], a well-documented and tested implementation of the grid domain.

## 3 Introduction

### Background

We distinguish between two kinds of numerical information about the values program variables can take: outer *limits* (or bounds within which the values must lie) and the pattern of *distribution* of these values. Both kinds of information have important applications:

in the field of automatic program verification, limit information is crucial to ensure that array accesses are within bounds, while distribution information is what is required to ensure that external memory accesses obey the alignment restriction imposed by the host architecture. In the field of program optimization, limit information can be used to compile out various kinds of run-time tests, whereas distribution information enables several transformations for efficient parallel execution as well as optimizations that enhance cache behavior.

Both limit and distribution information often come in a *relational* form; for instance, the outer limits or the pattern of possible values of one variable may depend on the values of one or more other variables. Domains that can capture relational information are generally much more complex than domains that do not have this capability; in exchange they usually offer significantly more precision, often important for the overall performance of the client application.

Relational limit information can be captured, among other possibilities, by means of *polyhedral domains*, that is, domains that represent regions of some  $n$ -dimensional vector space bounded by a finite set of hyperplanes [11]. While several polyhedral domains have been proposed and that of convex polyhedra, the most popular relational polyhedral domain, has been thoroughly researched and widely used, relational domains for representing the (linear) distribution of numerical values have been less well researched. In particular, proposed algorithms are unnecessarily complex and standard mathematical procedures have not been utilized to their best advantage. Moreover, as far as we know and at the time of writing, there is no available implementation providing all the basic operations needed by a relational abstract domain for distribution information. This is in spite of the fact that previous research has shown that a knowledge about the (discrete) distribution of numerical information, especially when combined with that of the limit information, can significantly improve the quality of the analysis results [1].

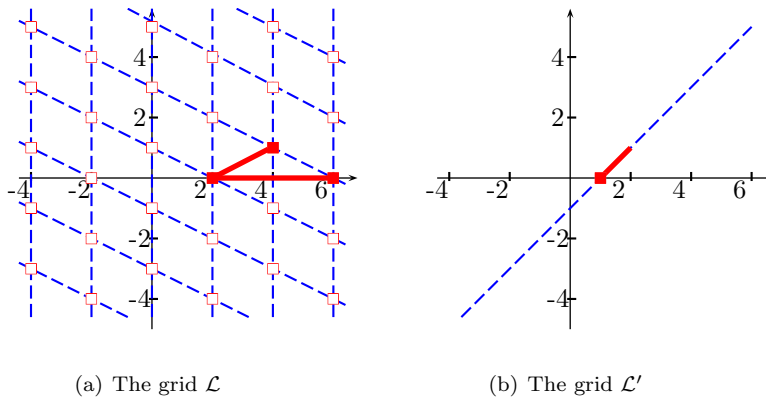


Figure 1: Congruence and generator systems representing two grids in  $\mathbb{R}^2$

### Grids in a Nutshell.

Figure 1 illustrates two ways of describing a grid; either by means of a finite set of congruence relations that all grid points must satisfy (given by the dashed lines) or by means of a finite set of generating vectors used for constructing the grid points and lines (given by the filled squares and thick lines).

The squares in Figure 1(a) illustrate the grid  $\mathcal{L}$  indicating possible values of integer variables  $x$  and  $y$  resulting from the execution of the program fragment in Figure 2 for any value of  $m$ .

The congruence relations  $x = 0 \pmod{2}$  and  $x + 2y = 2 \pmod{4}$  are represented by the vertical dashed lines and sloping lines, respectively. The set of congruence relations  $\mathcal{C} = \{x = 0 \pmod{2}, x + 2y = 2 \pmod{4}\}$ , called a *congruence system*,

```

x := 2; y := 0; (P1)
for i := 1 to m (P2)
  if ... then
    x := x + 4 (P3)
  else
    x := x + 2;
    y := y + 1 (P4)
  endif (P5)
endfor

```

Figure 2: Fragment based on an example in [11]

is said to *describe*  $\mathcal{L}$ . The filled squares mark the points  $\vec{p}_1 = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$ ,  $\vec{p}_2 = \begin{pmatrix} 6 \\ 0 \end{pmatrix}$  and  $\vec{p}_3 = \begin{pmatrix} 4 \\ 1 \end{pmatrix}$  while the squares (both filled and unfilled) mark points  $\vec{v} = \pi_1\vec{p}_1 + \pi_2\vec{p}_2 + \pi_3\vec{p}_3$ , where  $\pi_1, \pi_2, \pi_3 \in \mathbb{Z}$  and  $\pi_1 + \pi_2 + \pi_3 = 1$ . The set of *points*  $P = \{\vec{p}_1, \vec{p}_2, \vec{p}_3\}$  is said to *generate*  $\mathcal{L}$ . Some of these generating points can be replaced by *parameters* that give the gradient and distance between neighboring points. Specifically, by subtracting the point  $\vec{p}_1$  from each of the other two generating points  $\vec{p}_2$  and  $\vec{p}_3$ , we obtain the parameters  $\vec{q}_2 = \begin{pmatrix} 4 \\ 0 \end{pmatrix}$  and  $\vec{q}_3 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$  for  $\mathcal{L}$  that are marked by the thick lines between points  $\vec{p}_1$  and  $\vec{p}_2$  and points  $\vec{p}_1$  and  $\vec{p}_3$ . It follows that each point  $\vec{v} \in \mathcal{L}$  can be written as  $\vec{v} = \vec{p}_1 + \pi_2\vec{q}_2 + \pi_3\vec{q}_3$  for some  $\pi_2, \pi_3 \in \mathbb{Z}$ .

The dashed line in Figure 1(b) illustrates the grid  $\mathcal{L}'$  defining the line  $x = y + 1$  and marks the vectors of values of the real variables  $x$  and  $y$  after an assignment  $x := y + 1$ , assuming that nothing is known about the value of  $y$ . As equalities are congruences modulo 0, the set  $\mathcal{C}' = \{x - y = 1\}$  is also called a congruence system and describes  $\mathcal{L}'$ . Observe that the grid  $\mathcal{L}'$  consists of all points that can be obtained as  $\lambda\vec{\ell} + \vec{p}'$ , for any  $\lambda \in \mathbb{R}$ , where  $\vec{\ell} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$  and  $\vec{p}' = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ ; the vector  $\vec{\ell}$ , called a *line*, defines a gradient and the vector  $\vec{p}'$  is a generating point marking a position for the line (illustrated in Figure 1(b) by the thick line and the filled square, respectively).

From what we have just seen, any grid can be represented both by a congruence system and by a *generator system*. The latter may consist of three components: a set of lines, a set of parameters and a set of points. For instance, the triples  $\mathcal{G}_1 = (\emptyset, \emptyset, P)$  and  $\mathcal{G}_2 = (\emptyset, \{\vec{q}_2, \vec{q}_3\}, \{\vec{p}_1\})$  are both generator systems for  $\mathcal{L}$  while the triple  $\mathcal{G}' = (\{\vec{\ell}\}, \emptyset, \{\vec{p}'\})$  is a generator system for  $\mathcal{L}'$ .

## 4 Contributions.

In this project, we have explored the abstract domain of *grids*, a domain for representing sets of equally spaced points and hyperplanes over an  $n$ -dimensional vector space. A summarised account of this work on the domain of *grids* describing the domain and basic operations needed for its use within the theoretical framework of abstract interpretation has been published [3].

### The Grid Domain

For program analysis, an abstract domain needs to be a lattice (in the set-theoretic sense). The grid domain is no exception and in this project we have clarified and improved over previous proposals. In particular,

we have done the following.

**Survey of previous work.** We have provided a survey of several independent research threads concerning domains similar to the grid domain that have never before been compared or discussed together.

**Minimizing representations.** Assuming the grid is represented by a congruence and generator system in an  $n$ -dimensional vector space consisting of  $m$  congruences or generators, then we have described algorithms for minimizing the representation (based on the Hermite normal form algorithm [28, 33]) that have worst-case complexity  $O(n^2m)$ . Note that previous proposals for minimization such as those in [20, 27] have worse complexity bounds.

**Converting representations.** The congruence and generator representations described informally above form the two components of a double description method for the grid domain very similar to that for convex polyhedra [26]. For a double description method, conversion algorithms between the two systems are needed; we have shown how conversion can be implemented using any matrix inversion algorithm, inheriting the corresponding worst-case complexity. For instance, the complexity is  $O(n^3)$  when adopting the standard Gaussian elimination method; since matrix inversion has the same worst-case complexity as matrix multiplication, better theoretical complexity bounds apply [10]. Previous proposals for congruence to generator conversion have complexity no better than  $O(n^4)$  [21].

**Grid operations.** For static analysis, it is useful to provide all the set-theoretic lattice operations for grids (assuming the usual subset ordering) such as comparison, join and meet. We have shown that these operations are straightforward given the availability of the appropriate representation(s) in minimal form; and hence show that some have complexities strictly better than that of previous proposals [20]. We also have provided a (new) grid difference operator. Observe that this operator is useful in the specification of the certificate-based widening for the grid powerset domain [7].

## A Practical tool for Program Analysis.

We have shown that the domain of grids can be made into a really practical analysis tool. In particular, we have done the following.

**Affine image and preimage operators.** Affine image and preimage operators can be used to capture the effect of assignment statements in a program when the expression is linear. We have specified, for the domain of grids, the affine image and preimage operators for a *single update* where only one dimension is modified. We have illustrated how these can capture the behavior of assignments in (possibly recursive) procedures, comparing our technique with the interprocedural analysis in [27]. Moreover, we have described generalized forms of these operators which, though still having the necessary closure property for grids, map individual points to (possibly infinite) sets of points and shown how they can capture the result of loops containing simple assignment statements.

## Using Rectilinear Grids to Approximate Grids.

In order to dynamically manage the complexity/precision trade-off of program analysis, the analyzer should be able to dynamically switch (in response to timeouts or driven by heuristics) to a simpler analysis domain, with a more compact representation and more efficient operations. The first proposal for a numerical domain for representing distribution information was for a domain (we call here *rectilinear*), that can be described by arithmetic congruence relations [19] over the integers. This is less precise than *rational* grids (that is, grids described by the relational form), but most of its operations have complexity  $O(n)$ . Thus, by providing algorithms for conversion from a grid to its *rectilinear* approximation and vice-versa, we have made dynamic switching between relational and non-relational domains a practical option. We also have shown how rectilinear grids can be represented using the standard domain of intervals and thereby able to exploit all the advantages of this domain.

**Widening Grids.** It was observed by Granger in [21], that rational grids do not satisfy the ascending chain condition; so, to guarantee termination of the analysis, a widening operation is required. In [21, Proposition 10], a widening is given for rectilinear rational grids that returns a line parallel to an axis whenever the modulus for that dimension changes. It is then proposed that a generalized form of this could be used as a widening for relational grids; however, exactly how this is to be done is unclear. In this project, we have defined two possible generalizations which come with simple syntactic checks that have efficient implementations.

**Applications.** We have shown, by means of an example, how *rational* grids can be used to model the effect of (possibly recursive) procedures and hence be used for interprocedural program analysis where the input parameters are not known. We have observed that, previously, examples of interprocedural analysis for distribution information [27] assume that the input values are known in advance of the analysis.

**Implementation.** The grid domain as described here and in [2] is implemented in the PPL [6, 9]; the source code is publicly available in the PPL’s CVS repository (see the PPL web site at <http://www.cs.unipr.it/pp1/> for more information). Apart from the PPL, the only other freely available library providing numerical domains for applications in static analysis and computer-aided verification including a domain similar to that for grids, is PolyLib [24]. This domain is based on the integer lattice domain proposed by Ancoart [1] (which, together with polyhedra, forms the compound domain of  $\mathbb{Z}$ -polyhedra) and lacks sufficient features for non-trivial program analysis over this domain.

## 5 Dissemination of results

The work has been fully described in a technical report [2] and in the papers [3].

The latest release of the PPL [6, 9] includes the grid domain; the source code is publicly available in the PPL’s CVS repository (see the PPL web site at <http://www.cs.unipr.it/pp1/> for more information).

The PPL is being used on several applications in the field of verification of hardware and software systems. It has been applied to the verification of properties of oscillator circuits [17]; to verify the soundness of *batch workflow networks* (a kind of Petri nets used in workflow management) [35]; in the field of safety analysis of continuous and hybrid systems to overapproximate the systems of linear differential equations expressing the dynamics of hybrid automata [12, 14, 16, 32] and, in particular, the PPL is used in PHAVer, an innovative tool for the verification of such systems [15]. The PPL is also used: in a version of *TVLA* (3-Valued Logic Analysis Engine, <http://www.cs.tau.ac.il/~tvla/>) a system for the verification of properties of arrays and heap-allocated data [18]; in *iCSSV* (interprocedural C String Static Verifier), a tool for verifying the safety of string operations in C programs [13]; and in a static analyzer for *gated data dependence graphs*, an intermediate representation for optimizing compilation [22]. This analyzer employs, in particular, the precise widen-

ing operator and the widening with tokens technique introduced in [4, 5]. In [29] the PPL is used to derive invariant linear equalities and inequalities for a subset of the C language; it is used in *StInG* [30] and *LPIInv* [31], two systems for the analysis of transition systems; it is used for the model-checking of reconfigurable hybrid systems [34]; it is used in a static analysis tool for x86 binaries that automatically identifies instructions that can be used to redirect control flow, thus constituting vulnerabilities that can be exploited in order to bypass intrusion detection systems [23]; it is also used to automatically derive the *argument size relations* that are needed for termination analysis of Prolog programs [25].

In conclusion, since the PPL is free software and distributed under the terms of the GNU General Public License (GPL), and due to the presence of extensive documentation, the library can already be regarded as an important contribution secured to the community.

## 6 Educational Value

This project meant that the research student, Katy Dobson had to hone her theoretical ideas into truly practical techniques suitable for implementation. Moreover Matthew, who was employed on this project to do the implementation work, developed his skills in software development, particularly as part of a larger project; not only with respect to the technical collaboration but also in the broader experience of team working and communication of ideas. Both Katy and Matthew are co-authors on the technical report and the submitted papers derived from this and therefore learned about preparing papers for peer reviewing and publication. Regular project meetings meant the Katy, Matthew and the other students in the group were able to join in the discussions about the work of others as well as present their own research.

The travel budget (enhanced by a small amount that was surplus to the consumables budget) enabled Matthew to visit our colleagues in Italy and, vice versa, for them to visit us in Leeds. It also enabled all the group to attend an event in London incorporating, the Static Analysis Symposium, which is particularly relevant to the topic of this project.

## 7 Acknowledgments

We are grateful to our colleagues in Parma, namely Professor Roberto Bagnara and Dr Enea Zaffanella for their involvement and many contributions to the work of this project.

We would like to thank all those who have contributed in any way to the development of the PPL. The PPL project, started by Bagnara and Zaffanella in January 2001 in the University of Parma, was joined by us in the School of Computing in the University of Leeds in 2002. Without the PPL, whose development has been carefully shepherded over the last five years by our colleagues in Parma and whose construction has relied on the work of many students and other researchers, we could not have achieved what we have in this one year EPSRC project.

## References

- [1] C. Ancourt. *Génération automatique de codes de transfert pour multiprocesseurs à mémoires locales*. PhD thesis, Université de Paris VI, Paris, France, March 1991.
- [2] R. Bagnara, K. Dobson, P. M. Hill, M. Mundell, and E. Zaffanella. A linear domain for analyzing the distribution of numerical values. Report 2005.06, School of Computing, University of Leeds, UK, 2005. Available at <http://www.comp.leeds.ac.uk/research/pubs/reports.shtml>.
- [3] R. Bagnara, K. Dobson, P. M. Hill, M. Mundell, and E. Zaffanella. Grids: A domain for analyzing the distribution of numerical values. In G. Puebla, editor, *Logic-based Program Synthesis and Transformation, 16th International Symposium*, volume 4407 of *Lecture Notes in Computer Science*, pages 219–235, Venice, Italy, 2007. Springer-Verlag, Berlin.
- [4] R. Bagnara, P. M. Hill, E. Ricci, and E. Zaffanella. Precise widening operators for convex polyhedra. In R. Cousot, editor, *Static Analysis: Proceedings of the 10th International Symposium*, volume 2694 of *Lecture Notes in Computer Science*, pages 337–354, San Diego, California, USA, 2003. Springer-Verlag, Berlin.
- [5] R. Bagnara, P. M. Hill, E. Ricci, and E. Zaffanella. Precise widening operators for convex polyhedra. *Science of Computer Programming*, 58(1–2):28–56, 2005.
- [6] R. Bagnara, P. M. Hill, and E. Zaffanella. *The Parma Polyhedra Library User's Manual*. Department of Mathematics, University of Parma, Parma, Italy, release 0.9 edition, March 2006. Available at <http://www.cs.unipr.it/ppl/>.
- [7] R. Bagnara, P. M. Hill, and E. Zaffanella. Widening operators for powerset domains. *Software Tools for Technology Transfer*, 8(4/5):449–466, 2006. In the printed version of this article, all the figures have been improperly printed (rendering them useless). See [8].
- [8] R. Bagnara, P. M. Hill, and E. Zaffanella. Widening operators for powerset domains. *Software Tools for Technology Transfer*, 9(3/4):413–414, 2007. Erratum to [7] containing all the figures properly printed.
- [9] R. Bagnara, E. Ricci, E. Zaffanella, and P. M. Hill. Possibly not closed convex polyhedra and the Parma Polyhedra Library. In M. V. Hermenegildo and G. Puebla, editors, *Static Analysis: Proceedings of the 9th International Symposium*, volume 2477 of *Lecture Notes in Computer Science*, pages 213–229, Madrid, Spain, 2002. Springer-Verlag, Berlin.
- [10] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990.
- [11] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages*, pages 84–96, Tucson, Arizona, 1978. ACM Press.
- [12] L. Doyen, T. A. Henzinger, and J.-F. Raskin. Automatic rectangular refinement of affine hybrid systems. In P. Pettersson and W. Yi, editors, *Proceedings of the 3rd International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS 2005)*, volume 3829 of *Lecture Notes in Computer Science*, pages 144–161, Uppsala, Sweden, 2005. Springer-Verlag, Berlin.
- [13] R. Ellenbogen. Fully automatic verification of absence of errors via interprocedural integer analysis. Master's thesis, School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel, December 2004.
- [14] G. Frehse. Compositional verification of hybrid systems with discrete interaction using simulation relations. In *Proceedings of the IEEE Conference on Computer Aided Control Systems Design (CACSD 2004)*, Taipei, Taiwan, 2004.
- [15] G. Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control: Proceedings of the 8th International Workshop (HSCC 2005)*, volume 3414 of *Lecture Notes in Computer Science*, pages 258–273, Zürich, Switzerland, 2005. Springer-Verlag, Berlin.
- [16] G. Frehse, Z. Han, and B. Krogh. Assume-guarantee reasoning for hybrid I/O-automata by overapproximation of continuous interaction. In *Proceedings of the 43rd IEEE Conference on Decision and Control (CDC 2004)*, Atlantis, Paradise Island, Bahamas, 2004.
- [17] G. Frehse, B. H. Krogh, and R. A. Rutenbar. Verifying analog oscillator circuits using forward/backward refinement. In *Proceedings of the 9th Conference on Design, Automation and Test in Europe (DATE 06)*, Munich, Germany, 2006. ACM SIGDA. CD-ROM publication.
- [18] D. Gopan, T. W. Reps, and M. Sagiv. A framework for numeric analysis of array operations. In *Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 338–350, Long Beach, California, USA, 2005.

- [19] P. Granger. Static analysis of arithmetical congruences. *International Journal of Computer Mathematics*, 30:165–190, 1989.
- [20] P. Granger. Static analysis of linear congruence equalities among variables of a program. In S. Abramsky and T. S. E. Maibaum, editors, *TAPSOFT'91: Proceedings of the International Joint Conference on Theory and Practice of Software Development, Volume 1: Colloquium on Trees in Algebra and Programming (CAAP'91)*, volume 493 of *Lecture Notes in Computer Science*, pages 169–192, Brighton, UK, 1991. Springer-Verlag, Berlin.
- [21] P. Granger. Static analyses of congruence properties on rational numbers (extended abstract). In P. Van Hentenryck, editor, *Static Analysis: Proceedings of the 4th International Symposium*, volume 1302 of *Lecture Notes in Computer Science*, pages 278–292, Paris, France, 1997. Springer-Verlag, Berlin.
- [22] C. Hymans and E. Upton. Static analysis of gated data dependence graphs. In R. Giacobazzi, editor, *Static Analysis: Proceedings of the 11th International Symposium*, volume 3148 of *Lecture Notes in Computer Science*, pages 197–211, Verona, Italy, 2004. Springer-Verlag, Berlin.
- [23] C. Kruegel, E. Kirda, D. Mutz, W. Robertson, and G. Vigna. Automating mimicry attacks using static binary analysis. In *Proceedings of Security '05, the 14th USENIX Security Symposium*, pages 161–176, Baltimore, MD, USA, 2005.
- [24] V. Loechner. *PolyLib*: A library for manipulating parameterized polyhedra. Available at <http://icps.u-strasbg.fr/~loechner/polylib/>, March 1999. Declares itself to be a continuation of [36].
- [25] F. Mesnard and R. Bagnara. cTI: A constraint-based termination inference tool for ISO-Prolog. *Theory and Practice of Logic Programming*, 5(1&2):243–257, 2005.
- [26] T. S. Motzkin, H. Raiffa, G. L. Thompson, and R. M. Thrall. The double description method. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games – Volume II*, number 28 in *Annals of Mathematics Studies*, pages 51–73. Princeton University Press, Princeton, New Jersey, 1953.
- [27] M. Müller-Olm and H. Seidl. A generic framework for interprocedural analysis of numerical properties. In C. Hankin and I. Siveroni, editors, *Static Analysis: Proceedings of the 12th International Symposium*, volume 3672 of *Lecture Notes in Computer Science*, pages 235–250, London, UK, 2005. Springer-Verlag, Berlin.
- [28] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, 1988.
- [29] S. Sankaranarayanan, M. Colón, H. B. Sipma, and Z. Manna. Efficient strongly relational polyhedral analysis. In E. A. Emerson and K. S. Namjoshi, editors, *Verification, Model Checking and Abstract Interpretation: Proceedings of the 7th International Conference (VMCAI 2006)*, volume 3855 of *Lecture Notes in Computer Science*, pages 111–125, Charleston, SC, USA, 2006. Springer-Verlag, Berlin.
- [30] S. Sankaranarayanan, H. B. Sipma, and Z. Manna. Constraint-based linear-relations analysis. In R. Giacobazzi, editor, *Static Analysis: Proceedings of the 11th International Symposium*, volume 3148 of *Lecture Notes in Computer Science*, pages 53–68, Verona, Italy, 2004. Springer-Verlag, Berlin.
- [31] S. Sankaranarayanan, H. B. Sipma, and Z. Manna. Scalable analysis of linear systems using mathematical programming. In R. Cousot, editor, *Verification, Model Checking and Abstract Interpretation: Proceedings of the 6th International Conference (VMCAI 2005)*, volume 3385 of *Lecture Notes in Computer Science*, pages 25–41, Paris, France, 2005. Springer-Verlag, Berlin.
- [32] S. Sankaranarayanan, H. B. Sipma, and Z. Manna. Fixed point iteration for computing the time elapse operator. In J. Hespanha and A. Tiwari, editors, *Hybrid Systems: Computation and Control: Proceedings of the 9th International Workshop (HSCC 2006)*, volume 3927 of *Lecture Notes in Computer Science*, pages 537–551, Santa Barbara, CA, USA, 2006. Springer-Verlag, Berlin.
- [33] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, 1999.
- [34] H. Song, K. J. Compton, and W. C. Rounds. SPHIN: a model checker for reconfigurable hybrid systems based on SPIN. In R. Lazic and R. Nagarajan, editors, *Proceedings of the 5th International Workshop on Automated Verification of Critical Systems*, volume 145 of *Electronic Notes in Theoretical Computer Science*, pages 167–183, University of Warwick, UK, 2006.
- [35] K. van Hee, O. Oanea, N. Sidorova, and M. Voorhoeve. Verifying generalized soundness for workflow nets. In I. Virbitskaite and A. Voronkov, editors, *Perspectives of System Informatics: Proceedings of the Sixth International Andrei Ershov Memorial Conference*, volume 4378 of *Lecture Notes in Computer Science*, pages 231–244, Akademgorodok, Novosibirsk, Russia, 2006. Springer-Verlag, Berlin.
- [36] D. K. Wilde. A library for doing polyhedral operations. Master's thesis, Oregon State University, Corvallis, Oregon, December 1993. Also published as *IRISA Publication interne 785*, Rennes, France, 1993.