Recurrence with affine level mappings is P-time decidable for binary CLP($\mathbb{R}$) Technical note

FRED MESNARD
IREMIA, Université de la Réunion, France
(*e-mail:* `Frederic.Mesnard@univ-reunion.fr`)

ALEXANDER SEREBRENIK
Laboratory for Quality Software (LaQuSo), T.U. Eindhoven, The Netherlands
(*e-mail:* `A.Serebrenik@tue.nl`)

submitted April 30, 2006; revised 1 January 2003; accepted 1 January 2003

Abstract

Termination of binary constraint logic programs has recently become an important question in the termination analysis community. In this paper we introduce a class of binary constraint logic programs such that their termination can be proved by using affine level mappings. We show that membership to this class is decidable in polynomial time.

KEYWORDS: constraint logic programming – termination – decidability

1 Introduction

Termination is well-known to be one of the crucial properties of software verification. Logic programming, and more generally constraint logic programming (CLP), with their strong theoretical basis lend themselves easily to termination analysis as witnessed by a very intensive research in the area. Some of the recent approaches to termination (see, e.g., (Codish and Taboch 1999; Lagoon et al. 2003)) proceed in three steps. First, a logic program is abstracted to a CLP($\mathbb{N}$)-program, i.e., a logic program extended with constraint solving over the domain of natural numbers $\mathbb{N}$. Second, the CLP($\mathbb{N}$)-program is approximated by a *binary* CLP($\mathbb{N}$) program, i.e., a set of clauses of the form $p(\tilde{x}) \leftarrow c, q(\tilde{y})$, where c is a CLP-constraint and p, q are user-defined predicates. Finally, they conclude termination of the input program from termination of the abstracted binary program.

In this paper, which is a revised version of (Serebrenik and Mesnard 2004), we study decidability of termination for binary CLP($\mathbb{C}$) programs for a given constraint domain $\mathbb{C}$. In general, decidability depends on the constraint domain $\mathbb{C}$. On the one hand, Devienne et al. (1993) have established undecidability of termination for one-clause binary CLP($\mathbb{H}$) programs, where $\mathbb{H}$ is the domain of Herbrand terms. On the other hand, Datalog, i.e., logic programming with no function symbols, provides an example of a constraint programming language such that termination is decidable for it[1]. For constraint domains with the undecidable termination property, we are interested in subclasses of binary programs such that termination is decidable for these subclasses. A trivial example is the subclass of non-recursive binary programs.

[1] Decidability of a related problem of *boundedness* for Datalog queries has been studied, for instance, in (Afrati et al. 2005; Marcinkowski 1996).

We organise the paper as follows. After the preliminary remarks of Section 2, in Section 3 we present our main result. Section 4 reviews related results before our conclusion.

2 Preliminaries

For CLP-related definitions, we follow (Jaffar and Maher 1994; Jaffar et al. 1998). An extensive introduction to CLP can be found in (Marriott and Stuckey 1998). The key notions of CLP are those of an algebra and an associated constraint solver over a class of constraints, namely a set of first order formulae closed under conjunction and existential quantification. If c is a constraint, we write $\exists c$ for its existential closure. We consider *ideal* CLP($\mathbb{C}$), i.e., we require the existence of a constraint solver $solv_{\mathbb{C}}$ mapping in finite time each constraint to $\{true, false\}$ such that if $solv_{\mathbb{C}}(c) = false$ then the constraint $\exists c$ is *false* with respect to $\mathbb{C}$ and if $solv_{\mathbb{C}}(c) = true$ then the constraint $\exists c$ is *true* with respect to $\mathbb{C}$. The associated domain is denoted $D_{\mathbb{C}}$. The set of predicate symbols associated with $\mathbb{C}$ is denoted $\Pi_{\mathbb{C}}$. We are interested in the following domains and languages:

- $\mathbb{N}$. The predicate symbols are $=$ and $\geq$, the function symbols are 0, 1, and $+$.
- $\mathbb{Q}$ and $\mathbb{R}$. The predicate and function symbols are as above. $\mathbb{Q}^+$ and $\mathbb{R}^+$ restrict $\mathbb{Q}$ and $\mathbb{R}$ to non-negative numbers.

Given a program P, we define Π_P as the set of user-defined predicate symbols appearing in P. We restrict our attention to *binary* programs. A binary program is a finite set of binary clauses in a *flat* form. So each clause is of form:

- $p(\tilde{x}) \leftarrow c$ where $\tilde{x}$ denotes a tuple of variables and c is a constraint, or
- $p(\tilde{x}) \leftarrow c, q(\tilde{y})$ where $\tilde{x}$ and $\tilde{y}$ denote tuples of *distinct* variables, $\tilde{x} \cap \tilde{y} = \varnothing$, and the set of variables of the constraint c is included in $\tilde{x} \cup \tilde{y}$.

Flat queries are defined accordingly. An *atomic query* is a flat query of the form $c, q(\tilde{y})$ where $q \in \Pi_P$. In this paper we restrict our attention to atomic queries.

Given a CLP($\mathbb{C}$)-program P, the $\mathbb{C}$-base $B_P^{\mathbb{C}}$ is defined as $\{p(d_1,\ldots,d_n) \mid p \in \Pi_P, (d_1,\ldots,d_n) \in (D_{\mathbb{C}})^n\}$. For a query Q of the form (c, A), the set of ground instances of Q, denoted $ground_{\mathbb{C}}(Q)$, is the set of terms obtained from A by replacing all variables by elements of $D_{\mathbb{C}}$ such that c holds. The notion of groundedness is extended for flat clauses and binary programs: $ground_{\mathbb{C}}(p(\tilde{x}) \leftarrow c)$ is the set of atoms obtained by replacing all variables in $p(\tilde{x})$ by elements of $D_{\mathbb{C}}$ such that c holds. Similarly, $ground_{\mathbb{C}}(p(\tilde{x}) \leftarrow c, q(\tilde{y})$ is the set of binary clauses $p(\tilde{d}) \leftarrow q(\tilde{e})$ obtained by replacing all variables in $p(\tilde{x})$ and $q(\tilde{y})$ by the corresponding elements of $D_{\mathbb{C}}$ such that c holds.

Example 1
Consider the following CLP($\mathbb{Q}$) program P:

$$\begin{array}{llll} r_1 & p(x) & \leftarrow & x = 2. \\ r_2 & p(x) & \leftarrow & 0 = 1. \\ r_3 & p(x) & \leftarrow & 72 \geq x, y = x + 1, p(y). \end{array}$$

Then, $ground_{\mathbb{Q}}(r_1)$ is $\{p(2)\}$, $ground_{\mathbb{Q}}(r_2)$ is $\varnothing$, and $ground_{\mathbb{Q}}(r_3)$ is an infinite set that contains, among others, $p(72) \leftarrow p(73)$ and $p(1/2) \leftarrow p(3/2)$. Ground instances do not contain any constraint.

We say that a binary CLP($\mathbb{C}$) program P and a query Q *terminate* if every derivation of Q with respect to P is finite, under the usual operational semantics (Jaffar and Maher 1994). P is called *terminating* if for any ground query Q, P and Q terminate. To characterize termination, we use the notion of *level mapping*. A *level mapping* for a constraint domain $\mathbb{C}$ is a function $|\cdot| : B_P^{\mathbb{C}} \to \mathbb{R}$. We adapt the idea of recurrence, originally introduced in (Bezem 1993), to CLP:

Definition 1
Let P be a binary CLP($\mathbb{C}$) program, and $|\cdot| : \mathbb{C}$-*base* $\to \mathbb{R}$ be a level mapping. P is called *recurrent* with respect to $|\cdot|$ if there exists a real number $\varepsilon > 0$ such that, for every $A \leftarrow B \in ground_{\mathbb{C}}(P)$,

$|A| \in \mathbb{R}^+$, $|B| \in \mathbb{R}^+$, and $|A| \geq |B| + \varepsilon$. We say that P is *recurrent* if there exists a level-mapping such that P is recurrent with respect to it.

Observe that clauses of the form $p(\tilde{x}) \leftarrow c$ are not taken into account by the definition above. Indeed, these clauses cannot lead to infinite computation. Moreover, without loss of generality, we may fix ε to 1: if P is recurrent in this narrow sense, P is trivially recurrent with respect to Definition 1. Conversely, since $\varepsilon > 0$, we can safely multiply the values of the level mapping by $1/\varepsilon$.

Theorem 1
(Bezem 1993) P is recurrent iff P is terminating.

3 Alm-recurrent programs

Let us consider binary programs that can be analyzed by means of affine level mappings.

Definition 2
A level mapping $|\cdot|$ is called *affine* if for any n-ary predicate symbol $p \in \Pi_P$, there exist real numbers $\mu_{p,i}$, $0 \leq i \leq n$, such that for any atom $p(e_1,\ldots,e_n) \in B_P^{\mathbb{C}}$:

$$|p(e_1,\ldots,e_n)| = \mu_{p,0} + \sum_{i=1}^{n} \mu_{p,i} e_i$$

So for a given atom $p(\tilde{e})$, its affine level mapping is a linear combination of $\tilde{e}$ shifted by a constant. We can define the class of programs we are interested in:

Definition 3
Let P be a binary flat CLP($\mathbb{C}$) program. We say that P is *alm-recurrent* if there exists an affine level mapping $|\cdot|$ such that P is recurrent with respect to it.

Example 2
The CLP($\mathbb{Q}$) program P from Example 1 is alm-recurrent with respect to $|p(x)| = 73 - x$.

Clearly, if P is alm-recurrent, then P is recurrent thus terminating. The following result states that alm-recurrence can be efficiently decided.

Theorem 2
Alm-recurrence of a binary constraint logic program P over $\mathbb{Q}, \mathbb{Q}^+, \mathbb{R}$ and $\mathbb{R}^+$ is decidable in polynomial time with respect to the size of P.

Proof
The proof is constructive: we provide a decision procedure for alm-recurrence of binary constraint logic programs over $\mathbb{Q}, \mathbb{Q}^+, \mathbb{R}$ and $\mathbb{R}^+$. The decision procedure extends the algorithm proposed in (Sohn and Van Gelder 1991) for termination of Prolog programs (abstracted as CLP($\mathbb{N}$) programs) to binary CLP($\mathbb{C}$) where $\mathbb{C}$ is $\mathbb{Q}, \mathbb{Q}^+, \mathbb{R}$ or $\mathbb{R}^+$. The algorithm examines each user-defined predicate symbol p of a binary CLP program P in turn (the precise order does not matter) and tries to find an affine level mapping showing that P is alm-recurrent.

For every clause r, say $p(\tilde{x_p}) \leftarrow c, q(\tilde{x_q})$, we test the satisfiability of c. For the domains we consider, it can be done in polynomial time (Khachiyan 1979). If c is not satisfiable, we disregard this clause. Otherwise, let n_p and n_q be the arities of p and q. Recurrence requires:

$$\mathbb{C} \models c \rightarrow \left[|p(\tilde{x_p})| \geq 1 + |q(\tilde{x_q})| \wedge |q(\tilde{x_q})| \geq 0\right] \quad (1)$$

Strictly speaking, recurrence also requires $c \rightarrow |p(\tilde{x_p})| \geq 0$ but this condition can be omitted as it is implied by (1). Formula (1) is logically equivalent to $\mathbb{C} \models c \rightarrow |p(\tilde{x_p})| \geq 1 + |q(\tilde{x_q})|$ and $\mathbb{C} \models c \rightarrow |q(\tilde{x_q})| \geq 0$. Let $\tilde{x_p}$ be $(x_{p,1},\ldots,x_{p,n_p})$, $\tilde{x_q}$ be $(x_{q,1},\ldots,x_{q,n_q})$ and let $\mu_{p,0},\ldots,\mu_{p,n_p},\mu_{q,0},\ldots,\mu_{q,n_q} \in \mathbb{R}$ be such that for any atom $p(e_1,\ldots,e_{n_p}) \in B_P^{\mathbb{C}}$ and any atom $q(e_1,\ldots,e_{n_q}) \in B_P^{\mathbb{C}}$: $|p(e_1,\ldots,e_{n_p})| =$

$\mu_{p,0} + \sum_{i=1}^{n_p} \mu_{p,i} e_i$ and $|q(e_1, \ldots, e_{n_q})| = \mu_{q,0} + \sum_{i=1}^{n_q} \mu_{q,i} e_i$. Hence, c should imply $(\mu_{p,0} - \mu_{q,0}) + \sum_{i=1}^{n_p} \mu_{p,i} x_{p,i} + \sum_{i=1}^{n_q} (-\mu_{q,i}) x_{q,i} \geq 1$ and $\mu_{q,0} + \sum_{i=1}^{n_q} \mu_{q,i} x_{q,i} \geq 0$. For the sake of uniformity, we rewrite the second inequality as $\mu_{q,0} + \sum_{i=1}^{n_p} 0 x_{p,i} + \sum_{i=1}^{n_q} \mu_{q,i} x_{q,i} \geq 0$. Both inequalities can be presented using the scalar product notation as $\tilde{\mu}\tilde{x} \geq 1$ and $\tilde{\mu}'\tilde{x} \geq 0$, where:

$\tilde{x}$	$=$	$(x_0, x_{p,1}, \ldots, x_{p,n_p}, x_{q,1}, \ldots, x_{q,n_q})$
x_0		is a new variable fixed to 1 and used to obtain the free coefficient in the product
$\tilde{\mu}$	$=$	$(\mu_{p,0} - \mu_{q,0}, \mu_{p,1}, \ldots, \mu_{p,n_p}, -\mu_{q,1}, \ldots, -\mu_{q,n_q})$
$\tilde{\mu}'$	$=$	$(\mu_{q,0}, 0, \ldots, 0, \mu_{q,1}, \ldots, \mu_{q,n_q})$.

Hence, the binary clause r gives rise to the following two *pseudo* linear programming problems. The problems are *pseudo* linear rather than linear because *symbolic* parameters appear in the objective functions.

$$\textit{minimise}\ \ \theta = \tilde{\mu}\tilde{x}\ \ \textit{subject to}\ c \wedge x_0 = 1 \tag{2}$$

$$\textit{minimise}\ \ \delta = \tilde{\mu}'\tilde{x}\ \ \textit{subject to}\ c \wedge x_0 = 1 \tag{3}$$

We note that $c \wedge x_0 = 1$ is satisfiable as c is satisfiable and x_0 is a new variable, and we rewrite $c \wedge x_0 = 1$ as $A\tilde{x} \geq b$ in the standard way (Schrijver 1986). An affine level mapping $|\cdot|$ ensuring recurrence exists at least for this clause if and only if $\theta^* \geq 1$ and $\delta^* \geq 0$, where θ^* and δ^* denote the minima of the corresponding objective functions. Because of the symbolic constants $\mu_{p,i}$ and $\mu_{q,i}$, neither (2) nor (3) is a linear programming problem. Now, the idea is to consider the dual form:

$$\textit{maximise}\ \ \eta = b^T\tilde{y}\ \textit{subject to}\ A^T\tilde{y} = \tilde{\mu}^T \wedge \tilde{y} \geq 0 \tag{4}$$

$$\textit{maximise}\ \ \gamma = b^T\tilde{z}\ \textit{subject to}\ A^T\tilde{z} = \tilde{\mu}'^T \wedge \tilde{z} \geq 0 \tag{5}$$

where $\tilde{y}$ and $\tilde{z}$ are tuples of adequate length of new variables. By the duality theorem of linear programming which holds in $\mathbb{C}$ (see (Schrijver 1986) for instance), we have $\theta^* = \eta^*$ and $\delta^* = \gamma^*$. Furthermore, we observe that $\tilde{\mu}$ appears linearly in the dual problem (4). Hence the constraints of (4) can be rewritten, by adding $\eta \geq 1$ as a set of linear inequations denoted $S_r^{p\geq 1+q}$. Similarly, the constraints of (5) can be rewritten, by adding $\gamma \geq 0$ as a set of linear inequations, denoted $S_r^{q\geq 0}$. Let us define $\mathrm{defn}_P(p)$ as the set of binary clauses defining p in P, S_p as the conjunction $\bigwedge_{r\in \mathrm{defn}_P(p)} [S_r^{p\geq 1+q} \wedge S_r^{q\geq 0}]$, and S_P as the conjunction $\bigwedge_{p\in\Pi_P} S_p$. We have by construction S_P is satisfiable if and only if there exists a affine level mapping ensuring recurrence of P.

Moreover, as P is a finite set of binary clauses, computing S_P can be done in polynomial time with respect to the size of P and results in a constraint the size of which is also polynomial with respect to the size of P. Finally, testing satisfiability of S_P in $\mathbb{Q}$, $\mathbb{Q}^+$, $\mathbb{R}$, and $\mathbb{R}^+$ can be done in polynomial time (Khachiyan 1979). □

Note: As an immediate consequence of the result above, recurrency with affine level mappings is also P-time decidable for non-binary CLP($\mathbb{R}$) program with clauses which contain more than one atom in their bodies.

Example 3

Applying the algorithm to the example 1, we obtain the following two pseudo linear programming problems corresponding to (2) and (3), respectively:

$$\textit{minimise}\ \ \theta = \mu_{p,1}x_1 - \mu_{p,1}x_2\ \ \textit{subject to}\ \ 72 \geq x_1 \wedge x_2 = x_1 + 1 \wedge x_0 = 1$$

$$\textit{minimise}\ \ \delta = \mu_{p,0} + \mu_{p,1}x_2\ \ \textit{subject to}\ \ 72 \geq x_1 \wedge x_2 = x_1 + 1 \wedge x_0 = 1$$

Rewriting the system of constraints as $A\tilde{x} \geq b$ and switching to the dual form, we get the system S_P:

$$\left\{\begin{array}{l} \eta = y_1 - y_2 - 72 * y_3 + y_4 - y_5, \\ \eta \geq 1, \\ y_1 - y_2 = 0, -y_3 - y_4 + y_5 = \mu_{p,1}, \\ y_4 - y_5 = -\mu_{p,1}, \\ y_1 \geq 0, \\ y_2 \geq 0, \\ y_3 \geq 0, \\ y_4 \geq 0, \\ y_5 \geq 0 \end{array}\right\} \cup \left\{\begin{array}{l} \gamma = z_1 - z_2 - 72 * z_3 + z_4 - z_5, \\ \gamma \geq 0, \\ z_1 - z_2 = \mu_{p,0}, \\ -z_3 - z_4 + z_5 = 0, \\ z_4 - z_5 = \mu_{p,1}, \\ z_1 \geq 0, \\ z_2 \geq 0, \\ z_3 \geq 0, \\ z_4 \geq 0, \\ z_5 \geq 0 \end{array}\right\}$$

Since S_P is satisfiable, P is alm-recurrent. Note that projecting S_P onto the $\mu_{p,i}$'s gives $\{\mu_{p,0} + 73 * \mu_{p,1} \geq 0, \mu_{p,1} \leq -1\}$. Any solution to this last constraint is a level mapping ensuring alm-recurrence of P.

Although the technique above is not complete for binary programs over ℕ, it is a sound way to prove termination of programs over this domain: if the program terminates over ℚ it also terminates over ℕ. As we allow negative coefficients in the level mapping, we get a more powerful criterion than the one proposed in (Sohn and Van Gelder 1991). For instance, termination of Example 1 (considered as a CLP(ℕ) program) cannot be proved by (Sohn and Van Gelder 1991). Furthermore, the restriction to binary programs and dense domains as ℚ and ℝ expands the sufficient condition proposed by Sohn and Van Gelder to a correct, complete, and decidable proof technique.

For CLP(ℚ), the decision procedure described above has been *prototyped* in SICStus Prolog (SICS 2005) using the Simplex algorithm (Dantzig 1951) and a Fourier-based projection operator (Holzbaur 1995) to ease manual verification. Therefore, the complexity of the prototype is not polynomial. The implementation is available at `http://www.univ-reunion.fr/~gcc/soft/binterm4q.tgz`

4 Related Works

The basic idea of identifying decidable and undecidable subsets of logic programs goes back to (Devienne et al. 1993).

Recently, decidability of classes of imperative programs has been studied in (Cousot 2005; Podelski and Rybalchenko 2004; Tiwari 2004). Tiwari considers real-valued programs with no nested loops and no branching inside a loop (Tiwari 2004). Such programs correspond to one-binary-clause CLP(ℝ). The author provides decidability results for subclasses of these programs. Our approach does not restrict nesting of loops and it allows internal *branching*. While in general termination of such programs is undecidable (Tiwari 2004), we identified a subclass of programs with decidable termination property. Termination of the following CLP(ℝ) program and its imperative equivalent can be shown by our method but not by the one proposed in (Tiwari 2004).

Example 4

$$\begin{array}{lcl} q(x) & \leftarrow & -20 \leq x, x \leq 20, y + 5 = x, q(y). \\ q(x) & \leftarrow & 0 \leq x, x \leq 100, y + 1 = x, q(y). \end{array}$$

```
while $((-20 \leq x \leq 20)$ or $(0 \leq x \leq 100))$ do
    if $(-20 \leq x \leq 20)$ $x = x - 5$ fi
    if $(0 \leq x \leq 100)$ $x = x - 1$ fi
od
```

Similarly to (Tiwari 2004), Podelski and Rybalchenko (2004) have considered programs with no nested loops and no branching inside a loop. However, they focused on integer programs and provide a polynomial time decidability technique for a subclass of such programs. In case of general programs their technique can be applied to provide a sufficient condition for liveness.

In a recent paper, Cousot (2005) applied abstraction techniques and langrangian relaxation to prove termination. Extension of the basic technique should be able to analyse loops with disjunctions in their condition such as Example 4. However, complexity of the approach is not discussed and it is not clear whether the technique is complete for some class of programs.

One might like to investigate a more expressive language of constraints including polynomials. Recall that we require the constraints domain to be *ideal*, i.e., one needs a decision procedure for existentially closed conjunctions. Such a decision procedure exists, for instance, for real-closed fields such as $\mathbb{R}$ (Tarski 1931; Renegar 1992). For some domains such as $\mathbb{Q}$, existence of a decision procedure is still an open problem, although it seems to be unlikely (Pheidas 2000). If one restricts attention to real-closed fields, one might even consider polynomial level-mappings of a certain power rather than the affine ones. One can show that in this case proving recurrency is equivalent to determining satisfiability of the equivalent quantifier-free formula (Tarski 1931; Tarski 1951). Hence, recurrency is still decidable in this case. Although the known complexity bound of determining the equivalent quantifier-free formula given an existential formula is a double exponential (Basu et al. 1996; Collins 1975), to the best of our knowledge the complexity of the subclass of formulae which we obtain is an open question.

5 Conclusion

In this paper we have considered constraints solving over the rationals and the reals. For these domains we have identified a class of CLP programs such that an affine level mapping is sufficient to prove their termination. We have seen that membership to this class is decidable and presented a polynomial-time decision procedure. The decision procedure can also be used as a sound termination proof technique for binary constraint logic programs over the naturals and has been prototyped in SICStus Prolog.

Acknowledgements

We thank the referees for useful suggestions.

References

Afrati, F. N., Cosmadakis, S. S., and Foustoucos, E. 2005. Datalog programs and their persistency numbers. *ACM Transactions on Computational Logic (TOCL), 6,* 3, 481–518.

Basu, S., Pollack, R., and Roy, M.-F. 1996. On the combinatorial and algebraic complexity of quantifier elimination. *Journal of the ACM 43,* 6, 1002–1045.

Bezem, M. 1993. Strong termination of logic programs. *Journal of Logic Programming 15,* 1&2, 79–97.

Codish, M. and Taboch, C. 1999. A semantic basis for termination analysis of logic programs. *Journal of Logic Programming 41,* 1, 103–123.

Collins, G. E. 1975. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Second GI conference on Automata Theory and Formal Languages*. Lecture Notes in Computer Science, vol. 33. Springer, 134–183.

Cousot, P. 2005. Proving program invariance and termination by parametric abstraction, lagrangian relaxation and semidefinite programming. In *Verification, Model Checking, and Abstract Interpretation, 6th International Conference, VMCAI, Paris, France, January 17-19, 2005, Proceedings,* R. Cousot, Ed. Lecture Notes in Computer Science, vol. 3385. Springer, 1–24.

DANTZIG, G. B. 1951. Maximization of a linear function of variables subject to linear inequalities. In *Activity Analysis of Production and Allocation - Proceedings of a Conference*, T. Koopmans, Ed. Cowles Commission Monograph, vol. 13. Wiley, New York, 339–347.

DEVIENNE, P., LEBÈGUE, P., AND ROUTIER, J.-C. P. 1993. Halting problem of one binary horn clause is undecidable. In *STACS 93, 10th Annual Symposium on Theoretical Aspects of Computer Sc ience, Würzburg, Germany, February 25-27, 1993, Proceedings.*, P. Enjalbert, A. Finkel, and K. W. Wagner, Eds. Lecture Notes in Computer Science, vol. 665. Springer, 48–57.

HOLZBAUR, C. 1995. OFAI clp(Q,R) Manual. Tech. Rep. TR-95-09, Austrian Research Institute for Artificial Intelligence (ÖFAI), Schottengasse 3, A-1010 Vienna, Austria.

JAFFAR, J. AND MAHER, M. J. 1994. Constraint logic programming: A survey. *Journal of Logic Programming 19/20*, 503–582.

JAFFAR, J., MAHER, M. J., MARRIOTT, K., AND STUCKEY, P. J. 1998. The semantics of constraint logic programs. *Journal of Logic Programming 37*, 1-3, 1–46.

KHACHIYAN, L. 1979. A polynomial algorithm in linear programming. *Soviet Mathematics—Doklady 20*, 191–194.

LAGOON, V., MESNARD, F., AND STUCKEY, P. J. 2003. Termination analysis with types is more accurate. In *Logic Programming, 19th International Conference, ICLP 2003, Mumbai, India, December 9-13, 2003, Proceedings*, C. Palamidessi, Ed. Lecture Notes in Computer Science, vol. 2916. Springer, 254–268.

MARCINKOWSKI, J. 1996. DATALOG SIRUPs uniform boundedness is undecidable. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*. 13–24.

MARRIOTT, K. AND STUCKEY, P. J. 1998. *Programming with Constraints: An Introduction*. The MIT Press.

PHEIDAS, T. 2000. An effort to prove that the existential theory of is undecidable. *Contemporary Mathematics 270*, 237–252. Available at http://www.ams.org/mathscinet-getitem?mr=2001m:03085.

PODELSKI, A. AND RYBALCHENKO, A. 2004. A complete method for the synthesis of linear ranking functions. In *Verification, Model Checking, and Abstract Interpretation, 5th International Conference, Venice, January 11-13, 2004, Proceedings*, B. Steffen and G. Levi, Eds. Lecture Notes in Computer Science, vol. 2937. Springer, 239–251.

RENEGAR, J. 1992. On the computational complexity and geometry of the first-order theory of the reals. *Journal of Symbolic Computation 13*, 3, 255–352.

SCHRIJVER, A. 1986. *Theory of Linear and Integer Programming*. Wiley.

SEREBRENIK, A. AND MESNARD, F. 2004. On termination of binary CLP programs. In *Logic Based Program Synthesis and Transformation, 14th International Symposium, LOPSTR, Verona, Italy, August 26-28, 2004, Revised Selected Papers*, S. Etalle, Ed. Lecture Notes in Computer Science, vol. 3573. Springer, 231–244.

SICS. 2005. *SICStus User Manual. Version 3.12.3*. Swedish Institute of Computer Science.

SOHN, K. AND VAN GELDER, A. 1991. Termination detection in logic programs using argument sizes. In *Proceedings of the Tenth ACM SIGACT-SIGART-SIGMOD Symposium on Principles of Database Systems*. ACM Press, 216–226.

TARSKI, A. 1931. Sur les ensembles définissables des nombres réels. *Fundamenta Mathematicae 17*, 210–239.

TARSKI, A. 1951. *A Decision Method for Elementary Algebra and Geometry*, 2nd ed. University of California Press.

TIWARI, A. 2004. Termination of linear programs. In *Computer-Aided Verification, CAV*, R. Alur and D. Peled, Eds. Lecture Notes on Computer Science, vol. 3114. Springer, 70–82.