Polynomial Precise Interval Analysis Revisited

Thomas Gawlitza, Jérôme Leroux, Jan Reineke, Helmut Seidl, Grégoire Sutre, Reinhard Wilhelm

Polynomial Precise Interval Analysis Revisited

Thomas Gawlitza[1], Jérôme Leroux[2], Jan Reineke[3], Helmut Seidl[1], Grégoire Sutre[2], and Reinhard Wilhelm[3]

[1] TU München, Institut für Informatik, I2
80333 München, Germany
{gawlitza, seidl}@in.tum.de
[2] LaBRI, Université de Bordeaux, CNRS
33405 Talence Cedex, France
{leroux, sutre}@labri.fr
[3] Universität des Saarlandes, Germany
{reineke, wilhelm}@cs.uni-sb.de

Abstract. We consider a class of arithmetic equations over the complete lattice of integers (extended with $-\infty$ and ∞) and provide a polynomial time algorithm for computing least solutions. For systems of equations with addition and least upper bounds, this algorithm is a smooth generalization of the Bellman-Ford algorithm for computing the single source shortest path in presence of positive and negative edge weights. The method then is extended to deal with more general forms of operations as well as minima with constants. For the latter, a controlled widening is applied at loops where unbounded increase occurs. We apply this algorithm to construct a cubic time algorithm for the class of interval equations using least upper bounds, addition, intersection with constant intervals as well as multiplication.

1 Introduction

Interval analysis tries to derive tight bounds for the run-time values of variables [2]. This basic information may be used for important optimizations such as safe removals of array bound checks or for proofs of absence of overflows [4]. Since the very beginning of abstract interpretation, interval analysis has been considered as an algorithmic challenge. The reason is that the lattice of intervals may have infinite ascending chains. Hence, ordinary fixpoint iteration will not result in terminating analysis algorithms. The only general technique applicable here is the widening and narrowing approach of Cousot and Cousot [3]. If precision is vital, also more expressive domains are considered [8, 9]. While often returning amazingly good results, widening and narrowing typically does not compute the least solution of a system of equations but only a safe over-approximation.

In [11], however, Su and Wagner identify a class of interval equations for which the respective least solutions can be computed precisely and in polynomial time. As operations on intervals, they consider least upper bound, addition, scaling with positive and negative constants and intersection with constant intervals. The exposition of their algorithms, though, is not very explicit. Due to the importance of the problem,

we present an alternative and, hopefully more transparent approach. In particular, our methods also show how to deal with arbitrary multiplications of intervals. Our algorithm demonstrates how well-known ideas need only to be slightly extended to provide a both simple and efficient solution.

We start by investigating equations over integers only (extended with $-\infty$ and ∞ as least and greatest elements of the lattice) using maximum, addition, scaling with positive constants and minimum with constants as operations. In absence of minima, computing the least solution of such a system of equations can be considered as a generalization of the single-source shortest path problem from graphs to grammars in presence of positive and negative edge weights. A corresponding generalization for positive weights has been considered by Knuth [6]. Negative edge weights, though, complicate the problem considerably. While Knuth's algorithm can be considered as a generalization of Dijkstra's algorithm, we propose a generalization of the Bellman-Ford algorithm.

More generally, we observe that the Bellman-Ford algorithm works for all systems of equations which use operators satisfying a particular semantic property which we call BF-property. Beyond addition and multiplication with positive constants, positive as well as negative multiplication satisfies this property. *Positive* multiplication returns the product only if both arguments are positive, while *negative* multiplication returns the negated product if both arguments are negative. In order to obtain a polynomial algorithm also in presence of minima with constants, we instrument the basic Bellman-Ford algorithm to identify loops along which values might increase unboundedly. Once we have short-circuited the possibly costly iteration of such a loop we restart the Bellman-Ford algorithm until no further increments are found.

In the next step, we consider systems of equations over intervals using least upper bound, addition, negation, multiplication with positive constants as well as intersections with constant intervals and arbitrary multiplication of intervals. We show that computing the least solution of such systems can be reduced to computing the least solution of corresponding systems of integer equations. This reduction is inspired by the methods from [5] for interval equations with unrestricted intersections and the ideas of Leroux and Sutre [7], who first proved that interval equations with intersections with constant intervals as well as full multiplication can be solved in cubic time.

The rest of the paper is organized as follows. In Section 2, we introduce basic notions and consider methods for general systems of equations over $\mathcal{Z}$. Then we consider two classes of systems of equations over $\mathcal{Z}$ where least solutions can be computed in polynomial time. In Section 3, we consider systems of integer equations without minimum. In Section 4, we extend these methods to systems of equations where right-hand sides are *Bellman-Ford* functions. These systems can be solved in quadratic time (if arithmetic operations are executed in constant time). In Section 5, we then present our cubic time procedure for computing least solutions of systems of integer equations which additionally use minima with constants. In Section 6, we apply these techniques to construct a cubic algorithm for the class of interval equations considered by Su and Wagner [11] — even if additionally arbitrary multiplication of interval expressions is allowed.

2 Notation and Basic Concepts

Assume we are given a finite set of variables $\mathbf{X}$. We are interested in solving systems of constraints over the complete lattice $\mathcal{Z} = \mathbb{Z} \cup \{-\infty, \infty\}$ equipped with the natural ordering:

$$-\infty < \ldots < -2 < -1 < 0 < 1 < 2 < \ldots < \infty$$

On $\mathcal{Z}$, we consider the operations addition, multiplication with nonnegative constants, minimum "$\wedge$" and maximum "$\vee$". All operators are commutative where minimum, addition, and multiplication also preserve $-\infty$. Moreover for every $x > -\infty$,

$$\begin{array}{lll} x + \infty = \infty & & 0 \cdot \infty = 0 \\ x \cdot \infty \;\; = \infty & \text{whenever } x > 0 & x \cdot \infty = -\infty \quad \text{whenever } x < 0 \end{array}$$

For a finite set $\mathbf{X}$ of variables, we consider systems of equations

$$\mathbf{x} \;=\; e\,, \quad \mathbf{x} \in \mathbf{X}$$

where the right-hand sides e are expressions built from constants and variables from $\mathbf{X}$ by means of maximum, addition, multiplication with positive constants and minimum with constants. Thus right-hand sides e are of the form

$$e \quad ::= \quad a \mid \mathbf{y} \mid e_1 \vee e_2 \mid e_1 + e_2 \mid b \cdot e \mid e_1 \wedge a$$

for variables $\mathbf{y} \in \mathbf{X}$ and $a, b \in \mathcal{Z}$ where $a > -\infty$ and $b > 0$. Note that we excluded general multiplication since multiplication with negative numbers is no longer monotonic. Similar systems of equations have been investigated in [10] where polynomial algorithms for computing least upper bounds are presented — but only when computing least solutions over nonnegative integers.

A function $\mu : \mathbf{X} \to \mathcal{Z}$ is called a *variable assignment*. Every expression e defines a function $[\![e]\!] : (\mathbf{X} \to \mathcal{Z}) \to \mathcal{Z}$ that maps variable assignments to values, i.e.:

$$\begin{array}{lll} [\![a]\!]\mu & = a & [\![\mathbf{x}]\!]\mu & = \mu(\mathbf{x}) \\ [\![e_1 \vee e_2]\!]\mu & = [\![e_1]\!]\mu \vee [\![e_2]\!]\mu & [\![e_1 + e_2]\!]\mu & = [\![e_1]\!]\mu + [\![e_2]\!]\mu \\ [\![b \cdot e]\!]\mu & = b \cdot [\![e]\!]\mu & [\![e \wedge a]\!]\mu & = [\![e]\!]\mu \wedge a \end{array}$$

for $a \in \mathcal{Z}$, $\mathbf{x} \in \mathbf{X}$, $b > 0$ and expressions e, e_1, e_2. For a system $\mathcal{E}$ of equations, we also denote the function $[\![e]\!]$ by $f_{\mathbf{x}}$, if $\mathbf{x} = e$ is the equation in $\mathcal{E}$ for $\mathbf{x}$. A variable assignment μ is called a *solution* of $\mathcal{E}$ iff it satisfies all equations in $\mathcal{E}$, i.e. $\mu(\mathbf{x}) = f_{\mathbf{x}}\mu$ for all $\mathbf{x} \in \mathbf{X}$. Likewise, μ is a *pre-solution* iff $\mu(\mathbf{x}) \leq f_{\mathbf{x}}\mu$ for all $\mathbf{x} \in \mathbf{X}$. Since the mappings $f_{\mathbf{x}}$ are monotonic, every $\mathcal{E}$ has a unique least solution. In the following, we denote by $|\mathcal{E}|$ the sum of expression sizes of right-hand sides of the equations in $\mathcal{E}$. The following fact states bounds on the sizes of occurring values of variables:

Proposition 1. *Assume that $\mathcal{E}$ is a system of integer equations with least solution μ^*. Then we have:*

1. *If $\mu^*(\mathbf{x}) \in \mathbb{Z}$ for a variable $\mathbf{x}$, then:*

$$-(B \vee 2)^{|\mathcal{E}|} \cdot A \leq \mu^*(\mathbf{x}) \leq (B \vee 2)^{|\mathcal{E}|} \cdot A$$

where A and B bound the absolute values of constants $a \in \mathbb{Z}$ and constant multipliers $b \in \mathbb{N}$, respectively, which occur in $\mathcal{E}$.

2. *If $\mathcal{E}$ does not contain multiplication or addition of variables, the bounds under 1 can be improved to:*

$$\Sigma^- \leq \mu^*(\mathbf{x}) \leq \Sigma^+$$

where Σ^- and Σ^+ are the sums of occurrences of negative and positive numbers, respectively, in $\mathcal{E}$. □

In order to prove these bounds, we observe that they hold for systems of constraints without minimum operators. Then we find that for every $\mathcal{E}$, we can construct a system of equations $\mathcal{E}'$ without minimum operators by appropriately replacing every minimum expression by one of its arguments in such a way that $\mathcal{E}'$ has the same least solution as $\mathcal{E}$.

Due to Proposition 1, the least solutions of systems of equations over $\mathcal{Z}$ are computable by performing ordinary fixpoint iteration over the finite lattice

$$\mathcal{Z}_{a,b} = \{-\infty < a < \ldots < b < \infty\}$$

for suitable bounds $a < b$. This results in practical algorithms if a reasonably small difference $b-a$ can be revealed. In the following, we consider algorithms whose runtime does not depend on the particular sizes of occurring numbers – given that operations and tests on integers take time $\mathcal{O}(1)$.

3 Integer Equations without Minimum

We first consider systems of integer equations without minimum. Let us call these systems *disjunctive*. Note that we obtain the equational formulation of the single-source *longest* path problem for positive and negative edge weights if we restrict systems of disjunctive equations further by excluding multiplication and addition of variables in right-hand sides. By replacing all weights a with $-a$, the latter problem is a reformulation of the single-source *shortest* path problem (see, e.g., [1]).

In [6], Knuth considers a generalization of the single-source shortest path problem with nonnegative edge weights to grammars. In a similar sense, computing least solutions of systems of disjunctive constraints can be considered as a generalization of the single-source shortest path problem with positive and negative edge weights. For the latter problem, only quadratic algorithms are known [1]. Here, we observe that quadratic time is also enough for systems of disjunctive constraints:

Theorem 1. *The least solution of a disjunctive system $\mathcal{E}$ of equations with n variables can be computed in time $\mathcal{O}(n \cdot |\mathcal{E}|)$.*

Proof. As a generalization of the Bellman-Ford algorithm [1] we propose alg. 1 for computing the least solution of the system $\mathcal{E}$. The algorithm consists of two nested loops l_1, l_2 where the first one corresponds to n rounds of round robin fixpoint iteration, and the second one differs from the first in widening the value of a variable to ∞ whenever a further increase is observed. Let μ^* denote the least solution of $\mathcal{E}$. For a formal proof, let us define $F : (\mathbf{X} \to \mathcal{Z}) \to (\mathbf{X} \to \mathcal{Z})$ by

$$F(\mu)(\mathbf{x}) = [\![e]\!]\mu \qquad \text{if } (\mathbf{x} = e) \in \mathcal{E}$$

Algorithm 1

forall $(\mathbf{x} \in \mathbf{X})\ \mu(\mathbf{x}) = -\infty$;
for $(i = 0; i < n; i{+}{+})$
  forall $((\mathbf{x} = e) \in \mathcal{E})$
    $\mu(\mathbf{x}) = \mu(\mathbf{x}) \vee [\![e]\!]\mu$;
for $(i = 0; i < n; i{+}{+})$
  forall $((\mathbf{x} = e) \in \mathcal{E})$
    if $(\mu(\mathbf{x}) \not\geq [\![e]\!]\mu)\ \mu(\mathbf{x}) = \infty$;
return μ;

for $\mu : \mathbf{X} \to \mathcal{Z}$. Additionally we define the variable assignments μ_i for $i \in \mathbb{N}_0$ by

$$\begin{array}{ll} \mu_0(\mathbf{x}) = -\infty & \text{for } \mathbf{x} \in \mathbf{X} \\ \mu_i \quad = F^i(\mu_0) & \text{for } i \in \mathbb{N}. \end{array}$$

Thus $\bigvee_{i \in \mathbb{N}_0} \mu_i = \mu^*$ and in particular $\mu_i \leq \mu^*$ for all $i \in \mathbb{N}$. In order to prepare us for the proof, we introduce the following notion. Variable $\mathbf{x}$ *μ-depends on* $\mathbf{x}'$ iff

$$F(\mu \oplus \{\mathbf{x}' \mapsto \mu(\mathbf{x}') + \delta\})(\mathbf{x}) \geq F(\mu)(\mathbf{x}) + \delta$$

for all $\delta \geq 0$. Here, $\oplus$ denotes the update operator for variable assignments. We claim:

Claim 1: Let $k \geq 1$. Assume that $\mu_{k+1}(\mathbf{x}) > \mu_k(\mathbf{x})$. There exists a $\mathbf{y}$ s.t. $\mathbf{x}$ μ_k-depends on $\mathbf{y}$ with $\mu_k(\mathbf{y}) > \mu_{k-1}(\mathbf{y})$. □

The key observation is stated in the following Claim.

Claim 2: $\mu_n(\mathbf{x}) = \mu^*(\mathbf{x})$ whenever $\mu^*(\mathbf{x}) < \infty$.

Proof. Assume $\mu^*(\mathbf{x}) > \mu_n(\mathbf{x})$. Thus there exists an index $k \geq n$ s.t. $\mu_{k+1}(\mathbf{x}) > \mu_k(\mathbf{x})$. Claim 1 implies that there exist variables

$$\mathbf{x}_{k+1}, \mathbf{x}_k, \ldots, \mathbf{x}_1$$

where $\mathbf{x}_{k+1} = \mathbf{x}$ and $\mathbf{x}_{i+1}$ μ_i-depends on $\mathbf{x}_i$ for $i = 1, \ldots, k$. Since there are at least $n + 1$ elements in the sequence $\mathbf{x}_{k+1}, \ldots, \mathbf{x}_1$, the pigeon-hole principle implies that there must be a variable $\mathbf{x}'$ which occurs twice. W.l.o.g., let $j_1 < j_2$ s.t. $\mathbf{x}' = \mathbf{x}_{j_1} = \mathbf{x}_{j_2}$. Furthermore by assumption $\mu_{j_2}(\mathbf{x}') > \mu_{j_1}(\mathbf{x}')$.
By a straight forward induction it follows that

$$F^{j_2 - j_1}(\mu_{j_1} \oplus \{\mathbf{x}' \mapsto \mu_{j_1}(\mathbf{x}') + \delta\})(\mathbf{x}') \geq \mu_{j_2}(\mathbf{x}') + \delta \tag{1}$$

Let $\delta := \mu_{j_2}(\mathbf{x}') - \mu_{j_1}(\mathbf{x}') > 0$. Then

$$\begin{array}{lll} \mu^*(\mathbf{x}') \geq F^{i(j_2-j_1)}(\mu_{j_2})(\mathbf{x}') & & \\ \quad \geq F^{i(j_2-j_1)}(\mu_{j_1} \oplus \{\mathbf{x}' \mapsto \mu_{j_1}(\mathbf{x}') + \delta\})(\mathbf{x}') & & \text{(monotonicity)} \\ \quad \geq F^{(i-1)(j_2-j_1)}(\mu_{j_1} \oplus \{\mathbf{x}' \mapsto \mu_{j_2}(\mathbf{x}') + \delta\})(\mathbf{x}') & & \text{(1)} \\ \quad = F^{(i-1)(j_2-j_1)}(\mu_{j_1} \oplus \{\mathbf{x}' \mapsto \mu_{j_1}(\mathbf{x}') + 2\delta\})(\mathbf{x}') & & \text{(def. } \delta\text{)} \\ \quad \geq \cdots \geq \mu_{j_2}(\mathbf{x}') + i\delta & & \end{array}$$

for every $i \in \mathbb{N}$. Since $\mathbf{x}$ depends on $\mathbf{x}'$, we conclude that $\mu^*(\mathbf{x}) = \infty$. This proves claim 2. □

Let $\hat{\mu}_i$ denote the value of the program variable μ after execution of the i-th nested loop. By construction $\mu_n \leq \hat{\mu}_1 \leq \mu^*$. Whenever a further increase in the second nested loop can be observed, we know that $\mu \leq \mu^*$ and by claim 2, that after the modification $\mu \leq \mu^*$ still holds. Thus, $\hat{\mu}_2 \leq \mu^*$.

To show that $\hat{\mu}_2 = \mu^*$ recall that there are n variables. Therefore, at most n variables can be set to ∞ — implying that the least fixpoint is reached after at most n rounds. □

4 Extension with Positive and Negative Multiplications

Algorithm 1 can be generalized also to systems of equations which utilize a wider range of operators. We observe:

Proposition 2. *For any monotonic function $f : (\mathbf{X} \to \mathcal{Z}) \to \mathcal{Z}$, the two following conditions are equivalent:*

- (i) *for any $\mu : (\mathbf{X} \to \mathcal{Z})$ and any $\mathbf{Y} \subseteq \mathbf{X}$, if $f(\mu \oplus \{\mathbf{y} \mapsto -\infty \mid \mathbf{y} \in \mathbf{Y}\}) < f(\mu)$ then there is $\mathbf{y} \in \mathbf{Y}$ such that $f(\mu \oplus \{\mathbf{y} \mapsto \mu(\mathbf{y}) + i\}) \geq f(\mu) + i$ for all $i \geq 0$.*
- (ii) *for any $\mu : (\mathbf{X} \to \mathcal{Z})$ and any $\mathbf{x} \in \mathbf{X}$, if $f(\mu \oplus \{\mathbf{x} \mapsto -\infty\}) < f(\mu)$ then $f(\mu \oplus \{\mathbf{x} \mapsto \mu(\mathbf{x}) + i\}) \geq f(\mu) + i$ for all $i \geq 0$.*

Proof. $(i) \Rightarrow (ii)$ is trivial. For any $\mu : (\mathbf{X} \to \mathcal{Z})$ and any subset $\mathbf{Y} \subseteq \mathbf{X}$, we will write $\mu_{\mathbf{Y}}$ for $\mu_{\mathbf{Y}} = \mu \oplus \{\mathbf{y} \mapsto -\infty \mid \mathbf{y} \in \mathbf{Y}\}$. Assume that (ii) holds, and let us prove by induction on $|\mathbf{Y}|$ that (i) holds. The case of $\mathbf{Y} = \emptyset$ is trivial and the basis $|\mathbf{Y}| = 1$ follows from (ii). To prove the induction step, let $\mathbf{Y} \subseteq \mathbf{X}$ with $|\mathbf{Y}| > 1$ and assume that $f(\mu_{\mathbf{Y}}) < f(\mu)$. Pick some $\mathbf{y} \in \mathbf{Y}$ and let $\mathbf{Z} = \mathbf{Y} \setminus \{\mathbf{y}\}$. If $f(\mu_{\mathbf{Z}}) < f(\mu)$ then we derive from the induction hypothesis that there is $\mathbf{z} \in \mathbf{Z} \subseteq \mathbf{Y}$ such that $f(\mu \oplus \{\mathbf{z} \mapsto \mu(\mathbf{z}) + i\}) \geq f(\mu) + i$ for all $i \geq 0$. Otherwise, $f(\mu_{\mathbf{Z}} \oplus \{\mathbf{y} \mapsto -\infty\}) = f(\mu_{\mathbf{Y}}) < f(\mu) = f(\mu_{\mathbf{Z}})$, and we deduce from (ii) that $f(\mu_{\mathbf{Z}} \oplus \{\mathbf{y} \mapsto \mu(\mathbf{y}) + i\}) \geq f(\mu_{\mathbf{Z}}) + i$ for all $i \geq 0$. We come to $f(\mu \oplus \{\mathbf{y} \mapsto \mu(\mathbf{y}) + i\}) \geq f(\mu) + i$ for all $i \geq 0$ since $\mu \geq \mu_{\mathbf{Z}}$ and $f(\mu) = f(\mu_{\mathbf{Z}})$. We have thus shown that (i) holds for all $\mathbf{Y} \subseteq \mathbf{X}$. □

We call a function $f : (\mathbf{X} \to \mathcal{Z}) \to \mathcal{Z}$ *Bellman-Ford* function (short: BF-function) when it is monotonic and it satisfies any (or equivalently all) of the above conditions.

We remark that the class of Bellman-Ford functions is incomparable to the class of *bounded-increasing* functions as considered in [7]. Bounded-increasing functions are monotonic functions $f : (\mathbf{X} \to \mathcal{Z}) \to \mathcal{Z}$ such that $f(\mu_1) < f(\mu_2)$ for all $\mu_1, \mu_2 : \mathbf{X} \to \mathcal{Z}$ with $\mu_1 < \mu_2$, $f(\lambda \mathbf{x}. -\infty) < f(\mu_1)$ and $f(\mu_2) < f(\lambda \mathbf{x}.\infty)$. However, for any bounded-increasing function $f : (\mathbf{X} \to \mathcal{Z}) \to \mathcal{Z}$, if (1) f is continuous (i.e. $f(\bigvee_k \mu_k) = \bigvee_k f(\mu_k)$ for every ascending chain $\mu_0 \leq \mu_1 \leq \cdots$) and (2) $f(\lambda \mathbf{x}. -\infty) = -\infty$ and $f(\lambda \mathbf{x}.\infty) = \infty$, then f is a Bellman-Ford function.

Let us call a k-ary operator $\Box$ a BF-operator, if the function $f_\Box(\mu) = \Box(\mu(\mathbf{x}_1), \ldots, \mu(\mathbf{x}_k))$ (for distinct variables $\mathbf{x}_i$) is a BF-function.

Clearly, addition itself is a BF-operator as well as the least upper bound operation and the multiplication with constants. For simulating multiplication of intervals, we further rely on the following two approximative versions of multiplication:

$$x \cdot^{+} y = \begin{cases} x \cdot y \text{ if } x, y > 0 \\ -\infty \text{ otherwise} \end{cases} \qquad x \cdot^{-} y = \begin{cases} -(x \cdot y) \text{ if } x, y < 0 \\ \infty \quad\quad \text{otherwise} \end{cases}$$

We call these *positive* and *negative* multiplication, respectively. Note that, in contrast to full multiplication over the integers, both versions of multiplication are monotonic. Additionally, they satisfy the conditions for BF-functions and therefore are BF-operators. By induction on the structure of expressions, we find:

Lemma 1. *Assume e is an expression built up from variables and constants by means of application of BF-operators. Then the evaluation function $[\![e]\!]$ for e is a BF-function.*

Let us call an equation $\mathbf{x} = e$ BF-equation, if $[\![e]\!]$ is a BF-function. Our key observation is that the Bellman-Ford algorithm can be applied not only to disjunctive systems of equations but even to systems of BF-equations. In order to adapt the proof of theorem 1, we in particular adapt the proof of claim 1. We use the same notations from that proof. Let $k \geq 1$ and assume that $\mu_{k+1}(\mathbf{x}) > \mu_k(\mathbf{x})$. Then $(\mathbf{x} = e) \in \mathcal{E}$ for some expression e. The monotonic function $f_{\mathbf{x}} = [\![e]\!]$ is a Bellman-Ford function where $\mu_{k+1}(\mathbf{x}) = f_{\mathbf{x}}(\mu_k)$ and $\mu_k(\mathbf{x}) = f_{\mathbf{x}}(\mu_{k-1})$, and recall that $\mu_k \geq \mu_{k-1}$.

Let $\mathbf{Y} = \{\mathbf{y} \in \mathbf{X} \mid \mu_k(\mathbf{y}) > \mu_{k-1}(\mathbf{y})\}$. Since $f_{\mathbf{x}}(\mu_k \oplus \{\mathbf{y} \mapsto -\infty \mid \mathbf{y} \in \mathbf{Y}\}) \leq f_{\mathbf{x}}(\mu_{k-1}) < f_{\mathbf{x}}(\mu_k)$, we get from Proposition 2 that there is some $\mathbf{y} \in \mathbf{Y}$ such that $f(\mu_k \oplus \{\mathbf{y} \mapsto \mu_k(\mathbf{y}) + i\}) \geq f(\mu_k) + i$ for all $i \geq 0$. Hence, $\mathbf{x}$ μ_k-depends on $\mathbf{y}$, and moreover, $\mu_k(\mathbf{y}) > \mu_{k-1}(\mathbf{y})$ as $\mathbf{y} \in \mathbf{Y}$. This completes the proof of this claim.

Altogether, we obtain:

Theorem 2. *The least solution of a system $\mathcal{E}$ of BF-equations with n variables can be computed in time $\mathcal{O}(n \cdot |\mathcal{E}|)$.*

It is important here to recall that we consider a uniform cost measure where each operator can be evaluated in time $\mathcal{O}(1)$. If besides addition, also positive and negative multiplication is allowed, then the sizes of occurring numbers may not only be single exponential, but even double in the occurring numbers. More precisely, assume that μ^* is the least solution of $\mathcal{E}$ $\mathbf{x}$ is a variable of $\mathcal{E}$ with $\mu^*(\mathbf{x}) \in \mathbb{Z}$. Then

$$(A \vee 2)^{|\mathcal{E}|^n} \leq \mu^*(\mathbf{x}) \leq (A \vee 2)^{|\mathcal{E}|^n}$$

where A is an upper bound to the absolute values of constants $c \in \mathbb{Z}$ occurring in $\mathcal{E}$, and n is the number of variables.

5 Integer Equations with Minimum

In this section, we extend the results in the previous section by additionally allowing minima with constants. For convenience, let us assume that all right-hand sides r in the system $\mathcal{E}$ of equations either are of the following simple forms:

$$r ::= a \mid \mathbf{y} \mid \square(\mathbf{y}_1, \ldots, \mathbf{y}_k) \mid \mathbf{y} \wedge a$$

for constants $a \in \mathbb{Z}$, variables $\mathbf{y}$ and BF-operators $\Box$. Note that now the size $|\mathcal{E}|$ of $\mathcal{E}$ is proportional to the number of variables of $\mathcal{E}$. Our main result for systems of such equations is:

Theorem 3. *The least solution of a system $\mathcal{E}$ of integer equations using BF-operators and minima with constants can be computed in time $\mathcal{O}(|\mathcal{E}|^3)$.*

Proof. Let μ^* denote the least solution of $\mathcal{E}$. We introduce the following notions. We call a sequence $P = (\mathbf{y}_1, \ldots, \mathbf{y}_{k+1}) \in \mathbf{X}^*$ a path if for every $i = 1, \ldots, k$, variable $\mathbf{y}_{i+1}$ occurs in the right-hand side of the equation for $\mathbf{y}_i$ in $\mathcal{E}$. Thus, given a variable assignment μ, the path p represents the transformation $[\![p]\!]\mu : \mathcal{Z} \mapsto \mathcal{Z}$ defined by

$$[\![p]\!]\mu(z) = [\![e_1]\!](\mu \oplus \{\mathbf{y}_2 \mapsto [\![e_2]\!](\ldots [\![e_k]\!](\mu \oplus \{\mathbf{y}_{k+1} \mapsto z\})\ldots)\})$$

where $\mathbf{y}_i = e_i$ is the equation for $\mathbf{y}_i$ in $\mathcal{E}$.

The path p is called a *cycle* iff $\mathbf{y}_{k+1} = \mathbf{y}_1$. The cycle p is called *simple* if the variables $\mathbf{y}_1, \ldots, \mathbf{y}_k$ are pairwise distinct.

In order to enhance alg. 1 for systems with minima, assume that an increase of the value of the variable $\mathbf{x}$ can be observed within the first iteration of the second nested loop. Then there exists a simple cycle $c = (\mathbf{y}_1, \ldots, \mathbf{y}_k, \mathbf{y}_1)$ that can be repeated until either all variables $\mathbf{y}_i$ receive values ∞ or the value of the argument e' in some minimum expression $\mathbf{y} \wedge a$ occurring along the cycle exceeds a. In order to deal with this, we provide the following modified version of the Bellman-Ford algorithm:

1. We initialize the variable assignment μ s.t. every variable is mapped to $-\infty$ and execute the first phase of alg. 1 which consists of n Round-Robin iterations.
2. Then we perform the second phase. If no increment in the second phase can be detected, we have reached the least solution and return μ as result.
3. Whenever an increment in the second phase under a current variable assignment μ is detected, we try to extract a simple cycle $c = (\mathbf{y}_1, \ldots, \mathbf{y}_k, \mathbf{y}_1)$ s.t. $f'_{c,\mu}(v) > v$ for some $v < \mu(\mathbf{y}_1)$. If this is possible, then we do an *accelerated* fixpoint computation on the cycle c to determine new values for the variables $\mathbf{y}_1, \ldots, \mathbf{y}_k$. We then update the variables with the new values and restart the procedure with step 2.

This gives us alg. 2. Extra effort is necessary in order to extract cycles in the second phase which can be repeated. For that, the algorithm records in the variable $time$, the number of equations evaluated so far. Moreover for every variable $\mathbf{x}$, it records in $\mathit{modified}(\mathbf{x})$ the last time when the variable $\mathbf{x}$ has received a new value, and in $\mathit{evaluated}(\mathbf{x})$ the last time when the equation for $\mathbf{x}$ has been evaluated. Also, it records for every variable $\mathbf{x}$ in $\mathit{pred}(\mathbf{x})$ a variable n the right-hand side of $\mathbf{x}$ which may have caused the increase and can give rise to an increase in the future. If no such occurrence exists, then $\mathit{pred}(\mathbf{x})$ is set to $\perp$. This is implemented by the function $\mathbf{pred}(\mathbf{x})$. Let μ denote the current variable assignment, and assume that the right-hand side of $\mathbf{x}$ is r. Furthermore, let $\mathbf{Y}$ denote the set of variables $\mathbf{y}$ occurring in r which have been modified *after* the last evaluation of $\mathbf{x}$, i.e., $\mathit{modified}(\mathbf{y}) \geq \mathit{evaluated}(\mathbf{x})$. Since the value of $\mathbf{x}$ has increased, $\mathbf{Y}$ is non empty and, in particular, r cannot be equal to a constant. If $r = \mathbf{y}$, then $\mathbf{pred}(\mathbf{x}) = \mathbf{y}$. If $r = \mathbf{y} \wedge c$, then $\mathbf{pred}(\mathbf{x}) = \perp$

Algorithm 2

forall $(\mathbf{x} \in \mathbf{X})\ \mu(\mathbf{x}) = -\infty$;
do {
  $done = $ **true** ; $time = 0$;
  forall $(\mathbf{x} \in \mathbf{X})$ $\{modified(\mathbf{x}) = 0; pred(\mathbf{x}) = \bot; evaluated(\mathbf{x}) = 0;$
  }
  for $(i = 0; i < n; i{+}{+})$
    forall $((\mathbf{x} = e) \in \mathcal{E})$ {
      $time{+}{+}$;
      if $([\![e]\!]\mu > \mu(\mathbf{x}))$ {
        $pred(\mathbf{x}) = \mathbf{pred}(\mathbf{x}); \mu(\mathbf{x}) = [\![e]\!]\mu; modified(\mathbf{x}) = time;$
      }
      $evaluated(\mathbf{x}) = time;$
    }
  forall $((\mathbf{x} = e) \in \mathcal{E})$
    if $([\![e]\!]\mu > \mu(\mathbf{x}))$ {
      $\mu(\mathbf{x}) = [\![e]\!]\mu;$
      if $(\mu(\mathbf{x}) < \infty)$ {
        $\mathbf{widen}(\mathbf{x}); done = $ **false** ; **break**;
      } ;
    }
} **until** $(done)$;
return μ;

if $\mu(\mathbf{y}) \geq c$ and $\mathbf{pred}(\mathbf{x}) = \mathbf{y}$ otherwise. Finally, assume $r = \Box(\mathbf{y}_1, \ldots, \mathbf{y}_k)$ and let $v_j = \mu(\mathbf{y}_j)$ for all j. Furthermore, let $v'_j = v_j$ if $\mathbf{y}_j \notin \mathbf{Y}$, i.e., has not been changed since the last evaluation of $\mathbf{x}$, and $v'_j = -\infty$ otherwise. Then $\Box(v'_1, \ldots, v'_k) < \Box(v_1, \ldots, v_k)$. Since $\Box$ is a BF-operator, we thus can retrieve an index j such that $\Box(v_1, \ldots, v_{j-1}, v_j + d, v_{j+1}, \ldots, v_k) \geq \Box(v_1, \ldots, v_k) + d$ for all $d \geq 0$. Accordingly, we set $\mathbf{pred}(\mathbf{x}) = \mathbf{y}_j$. The following observation shows that $\mathbf{pred}$ can be computed in time $\mathcal{O}(1)$ if the maximal arity of the BF-operators is considered as a constant.

Lemma 2. *Consider a BF-function $f : (\mathbf{X} \to \mathcal{Z}) \to \mathcal{Z}$ and a variable assignment $\mu : \mathbf{X} \to \mathcal{Z}$ and a variable $\mathbf{x} \in \mathbf{X}$. We have $f(\mu \oplus \{\mathbf{x} \mapsto \mu(\mathbf{x}) + 1\}) \geq f(\mu) + 1$ iff $f(\mu \oplus \{\mathbf{x} \mapsto \mu(\mathbf{x}) + i\}) \geq f(\mu) + i$ for all $i \geq 0$.*

Example 1. Let $\mu := \{\mathbf{x} \mapsto -10, \mathbf{y} \mapsto 0\}$ and consider the equation $\mathbf{z} = \mathbf{x} \vee \mathbf{y}$. Then for $\mathbf{Y} = \{\mathbf{x}, \mathbf{y}\}$, the function call $\mathbf{pred}(\mathbf{z})$ returns the variable $\mathbf{y}$.

Now we consider the second phase of alg. 2. Whenever a finite increase of the value of a variable $\mathbf{x}$ is detected, $\mathbf{widen}(\mathbf{x})$ is called (see alg. 3).

Algorithm 3 $\mathbf{widen}(\mathbf{x})$

$c = (\mathbf{y}_1, \ldots, \mathbf{y}_k, \mathbf{y}_1) = \mathbf{extract\_cycle}(\mathbf{x})$;
$\mu(\mathbf{y}_1) = \mu(\mathbf{y}_1) \vee \mathbf{eval\_cycle}(c)$;
for $(i = k;\ i \geq 2;\ i{-}{-})$
  $\mu(\mathbf{y}_i) = \mu(\mathbf{y}_i) \vee f_{\mathbf{y}_i}(\mu)$;

Within the procedure $\mathbf{widen}()$, the function $\mathbf{extract\_cycle}()$ is used to extract a cycle which has caused the increase and possibly causes further increases in the future. It works as follows. The call $\mathbf{extract\_cycle}(\mathbf{x})$ for a variable $\mathbf{x}$ looks up the value of $pred(\mathbf{x})$. If $pred(\mathbf{x}) \neq \bot$ a variable $\mathbf{x}_1$ in the right-hand side for $\mathbf{x}$ is returned. Then the procedure records $(\mathbf{x}_1)$ and proceeds with the value stored in $pred(\mathbf{x}_1)$ and so on. Thus, it successively visits a path according to the information stored in $pred$ until it either reaches $\bot$ or visits a variable for the second time. In the latter case we obtain a simple cycle $(\mathbf{y}_1, \ldots, \mathbf{y}_k, \mathbf{y}_1)$. With a cyclic permutation $modified(\mathbf{y}_1)$ is assumed maximal. In the former case, the empty sequence will be returned.

The procedure $\mathbf{eval\_cycle}()$ does the accelerated fixpoint computation on a given cycle. The function $\mathbf{eval\_cycle}()$ takes a simple cycle $c = (\mathbf{y}_1, \ldots, \mathbf{y}_k, \mathbf{y}_1)$. Let $f := [\![c]\!]\mu$ and assume that $f(v) > v$ for some $v \leq \mu(\mathbf{y}_1)$. Then $\mathbf{eval\_cycle}()$ computes $\bigvee_{i\in\mathbb{N}} f^i(v)$. As monotonic functions over a linear order are distributive over $\wedge$, note that $f(z)$ can be written as

$$f(z) = f'(z) \wedge b'$$

for some unary BF-function f' and $b' \in \mathcal{Z}$. Since $f(v) > v, b' \geq f'(v) > v$. Therefore,

$$\bigvee_{i\in\mathbb{N}} f^i(v) = b' = f(\infty)$$

We conclude that $\bigvee_{i\in\mathbb{N}} f^i(v)$ can be computed in time linear to the size of the simple cycle c. Furthermore, $\bigvee_{i\in\mathbb{N}} f^i(v) \leq \mu^*(\mathbf{y}_1)$ by construction. Thus, we have shown the following claim:

Claim 1: Assume that c is a simple cycle which starts with the variable $\mathbf{y}_1$. Assume that $\mu \leq \mu^*$ and $v \leq \mu(\mathbf{y}_1)$ are s.t. $f(v) := [\![c]\!]\mu(v) > v$. Then $v' := \bigvee_{i\in\mathbb{N}} f^i(v) \leq \mu^*(\mathbf{y}_1)$ and v' can be computed in time linear to the size of c. □

For a formal proof of correctness of the algorithm, let μ_i denote the variable assignment μ before the i-th extraction of a simple cycle and c_i the value of c after the i-th extraction of a simple cycle. Thereby c_i can be $\bot$. Let furthermore μ'_i denote the value of μ after the i-th call of the procedure $\mathbf{widen}()$.

First, we show that the widening is correct, i.e., $\mu'_i \leq \mu^*$ for all i. For that, we only need to consider the case in which the i-th extraction leads to a simple cycle c_i and not to $\bot$. Thanks to Claim 1, we only need to show that the assertions of Claim 1 are fulfilled for every call of the procedure $\mathbf{widen}()$ in which $\mathbf{extract\_cycle}$ extracts a simple cycle. Thus we must show:

Claim 2: Assume that $c_i \neq \bot$ is a simple cycle which starts with the variable $\mathbf{y}_1$. Then $([\![c_i]\!]\mu_i)(v) > v$ for some value $v < \mu_i(\mathbf{y}_1)$.

Proof. Assume that $c_i = (\mathbf{y}_1, \ldots, \mathbf{y}_k, \mathbf{y}_1)$ where $\mathbf{y}_j = e_j$ is the equation for $\mathbf{y}_j$. Observe that the algorithm always records an occurrence of a variable which possibly has caused the increase. Therefore, by monotonicity, $[\![e_j]\!]\mu_i$ is at least the current value $\mu_i(\mathbf{y}_j)$ of the left-hand side $\mathbf{y}_j$. This means for the cycle c_i that

$$\mu_i(\mathbf{y}_1) \leq [\![e_1[\mu_i(\mathbf{y}_2)/\mathbf{y}_2]]\!]\mu_i \wedge \ldots \wedge \mu_i(\mathbf{y}_{k-1}) \leq [\![e_{k-1}[\mu_i(\mathbf{y}_k)/\mathbf{y}_k]]\!]\mu_i$$

as well as

$$\mu_i(\mathbf{y}_k) \leq [\![e_k[v/\mathbf{y}_1]]\!]\mu_i$$

where v is the value of the variable $\mathbf{y}_1$ at the last point in time where the evaluation of the equation $\mathbf{y}_k = e_k$ lead to an increase. As $\mathit{modified}(\mathbf{y}_1)$ is maximal, we get $v < \mu_i(\mathbf{y}_1)$. Since by construction, $([\![c_i]\!]\mu_i)(v) \geq \mu_i(\mathbf{y}_1) > v$, the assertion follows. □

Assume again that $c_i = (\mathbf{y}_1, \ldots, \mathbf{y}_k, \mathbf{y}_1)$ is a simple cycle and assume as induction hypothesis, that the variable assignment μ'_{i-1} after the $(i-1)$-th widening is less than or equal to the least solution μ^* of the system $\mathcal{E}$. Since the variable assignment μ_i before the extraction of the cycle c_i is computed by fixpoint iteration, it follows that $\mu_i \leq \mu^*$. Let v' denote the values returned from the i-th call of $\mathbf{eval\_cycle}()$. By Claim 1, $v' \leq \mu^*(\mathbf{y})$. Since the rest of procedure $\mathbf{widen}()$ consists in ordinary fixpoint iteration, we obtain $\mu' \leq \mu^*$.

Thus by construction, alg. 2 returns μ^* — whenever it terminates. In order to prove termination, let $M(\mathcal{E})$ denote the set of minimum expressions occurring in $\mathcal{E}$. We show the following claims which imply that a progress occurs at each increase of a variable's value in the second phase, i.e., either one further variable receives the value ∞ or another minimum can (conceptually) be replaced by its constant argument.

Claim 3: Assume that $c_i = \bot$. Then
- either there exists a variable $\mathbf{x}$ such that $\mu'_{i-1}(\mathbf{x}) < \infty$ and $\mu_i(\mathbf{x}) = \infty$;
- or there exists a subexpression $\mathbf{y} \wedge a$ from $M(\mathcal{E})$ s.t. $\mu'_{i-1}(\mathbf{y}) < a$ and $\mu_i(\mathbf{y}) \geq a$.

Proof. $c_i = \bot$ implies that the procedure $\mathbf{pred}()$ returned $\bot$ for one of the equations $\mathbf{x} = e$ in the *same* iteration of the main loop. This is because all values of $\mathit{pred}()$ reachable within n steps by $\mathbf{extract\_cycle}()$ have been modified during this iteration. Longer paths would imply finding a simple cycle. However, the procedure $\mathbf{pred}()$ only returns $\bot$ if some minimum a is reached which had not been reached before. □

From Claim 3 and the fact that the sequence (μ'_i) is increasing we conclude that for every i,

$$\{\mathbf{x} \in \mathbf{X} \mid \mu'_i(\mathbf{x}) = \infty\} \supsetneq \{\mathbf{x} \in \mathbf{X} \mid \mu'_{i-1}(\mathbf{x}) = \infty\}$$

or

$$\{\mathbf{x} \wedge a \in M(\mathcal{E}) \mid [\![\mathbf{x}]\!]\mu'_i \geq a\} \supsetneq \{\mathbf{x} \wedge a \in M(\mathcal{E}) \mid [\![\mathbf{x}]\!]\mu'_{i-1} \geq a\}.$$

Accordingly, the algorithm can perform at most $\mathcal{O}(|\mathcal{E}|)$ iterations of the outer **while**-loop. Since every iteration of the outer loop of the algorithm can be executed in time $\mathcal{O}(n \cdot |\mathcal{E}|)$, the assertion follows. □

Example 2. Consider the following system of equations:

$$\mathbf{x} = \mathbf{y} \wedge 5 \qquad \mathbf{y} = \mathbf{z} \wedge 3 \qquad \mathbf{z} = -17 \vee \mathbf{z} + 2$$

The first three rounds of Round-Robin iteration give us:

	0	1	2
$\mathbf{x}$	$-\infty$	$-\infty$	-15
$\mathbf{y}$	$-\infty$	-15	-13
$\mathbf{z}$	-15	-13	-11

Since the value of $\mathbf{x}$ still increases during the next round of evaluation, we call the function $\mathbf{widen}()$ with the variable $\mathbf{x}$. Within $\mathbf{widen}()$ the function $\mathbf{extract\_cycle}$ is called which returns the simple cycle $(\mathbf{z}, \mathbf{z})$. — giving us the new value ∞ for $\mathbf{z}$. Restarting the Round-Robin iteration for all variables, reveals the least solution:

$$\mu^*(\mathbf{x}) = 3 \qquad \mu^*(\mathbf{y}) = 3 \qquad \mu^*(\mathbf{z}) = \infty$$

6 Intervals

In this section, we consider systems of equations over the complete lattice of integer intervals. Let

$$\mathcal{I} = \{\emptyset\} \cup \{[z_1, z_2] \in \mathcal{Z}^2 \mid z_1 \leq z_2, z_1 < \infty, -\infty < z_2\}$$

denote the complete lattice of intervals partially ordered by the subset relation (here denoted by "$\sqsubseteq$"). The empty interval $\emptyset$ is also denoted by $[\infty, -\infty]$. It is the least element of the lattice while $[-\infty, \infty]$ is the greatest element, and the least upper bound "$\sqcup$" is defined by:

$$[a_1, a_2] \sqcup [b_1, b_2] = [a_1 \wedge b_1, a_2 \vee b_2]$$

Here, we consider systems of equations over $\mathcal{I}$ similar to the ones we have considered over $\mathcal{Z}$ with the restriction that at least one argument of every intersection is constant. Instead of multiplication with positive constants only, we now also support negation as well as *full* multiplication of interval expressions. For a fixed set $\mathbf{X}$ of variables, we consider expressions e of the form

$$e \quad ::= \quad a \mid \mathbf{y} \mid c \cdot e \mid -e \mid e_1 \sqcup e_2 \mid e_1 + e_2 \mid e \sqcap a \mid e_1 \cdot e_2$$

where $a \in \mathcal{I}$, $c > 0$ is a positive integer constant, and $\mathbf{y}$ is a variable from $\mathbf{X}$.

As for expressions over $\mathcal{Z}$, we rely on an evaluation function $[\![e]\!]$ for interval expressions e built up from variables and constants by means of applications of operators. The function $[\![e]\!]$ then maps variable assignments $\mu : \mathbf{X} \to \mathcal{I}$ to interval values. Note that (in contrast to the integer case) full multiplication of intervals still is monotonic. Therefore, every system of interval equations has a unique least solution.

Our goal is to reduce solving of systems of equations over intervals, to solving of systems equations over integers. For that, we define the functions $(\cdot)^+, (\cdot)^- : \mathcal{I} \to \mathcal{Z}$ which extract from an interval the upper and *negated* lower bound, respectively. These functions are defined by:

$$\emptyset^+ = \emptyset^- = -\infty \qquad [a, b]^+ = b \qquad [a, b]^- = -a$$

where $[a, b] \in \mathcal{I}$. Thus x^+ denotes the upper bound and x^- denotes the *negated* lower bound of $x \in \mathcal{I}$. In the following, we indicate how operations on intervals can be realized by means of integer operations on interval bounds.

Assume $x, y \in \mathcal{I}$ are intervals and $c > 0$. Then we have:

$(c \cdot x)^-$	$=$	$c \cdot x^-$
$(c \cdot x)^+$	$=$	$c \cdot x^+$
$(-x)^-$	$=$	x^+
$(-x)^+$	$=$	x^-
$(x \sqcup y)^-$	$=$	$x^- \vee y^-$
$(x \sqcup y)^+$	$=$	$x^+ \vee y^+$
$(x + y)^-$	$=$	$x^- + y^-$
$(x + y)^+$	$=$	$x^+ + y^+$
$(x \sqcap y)^-$	$=$	$(x^+ + y^-); (x^- + y^+); x^- \wedge y^-$
$(x \sqcap y)^+$	$=$	$(x^+ + y^-); (x^- + y^+); x^+ \wedge y^+$
$(x \cdot y)^-$	$=$	$-(x^- \cdot y^-) \vee -(x^+ \cdot y^+) \vee x^- \cdot y^+ \vee x^+ \cdot y^-$
	$=$	$(x^- \cdot^- y^- \wedge 0) \vee (x^+ \cdot^- y^+ \wedge 0) \vee x^- \cdot^+ y^+ \vee x^+ \cdot^+ y^-$
$(x \cdot y)^+$	$=$	$x^- \cdot y^- \vee x^+ \cdot y^+ \vee -(x^- \cdot y^+) \vee -(x^+ \cdot y^-)$
	$=$	$x^- \cdot^+ y^- \vee x^+ \cdot^+ y^+ \vee (x^- \cdot^- y^+ \wedge 0) \vee (x^+ \cdot^- y^- \wedge 0)$

Here, the operator $x; y$ returns $-\infty$ if $x < 0$ and y otherwise. This operator can be expressed by means of positive multiplication together with a minimum with 0:

$$x\,;\, y = (((x + 1) \cdot^+ 1) \wedge 0) + y$$

Additionally, we observe that w.r.t. the interval bounds, interval multiplication can be expressed through positive and negative multiplications together with minima with 0.

Every system $\mathcal{E}$ of interval equations gives rise to a system $\mathcal{E}^\pm$ of integer equations over $\mathcal{Z}$ for the upper and negated lower bounds for the interval values of the variables from $\mathcal{E}$. For every variable $\mathbf{x}$ of the interval system $\mathcal{E}$, we introduce the two integer variables $\mathbf{x}^-, \mathbf{x}^+$. The variable $\mathbf{x}^+$ is meant to receive the upper interval bound of $\mathbf{x}$ whereas the variable $\mathbf{x}^-$ is meant to receive the negated lower interval bound of $\mathbf{x}$.

Every equation $\mathbf{x} = e$ of $\mathcal{E}$ then gives rise to the equations $\mathbf{x}^- = [e]^-$ and $\mathbf{x}^+ = [e]^+$ of $\mathcal{E}^\pm$ for the new integer variables corresponding to the left-hand side $\mathbf{x}$ where the new right-hand sides $[e]^-$ and $[e]^+$ are obtained by the following transformations:

$[[a_1, a_2]]^-$	$= -a_1$	$[[a_1, a_2]]^+$	$= a_2$
$[\mathbf{x}]^-$	$= \mathbf{x}^-$	$[\mathbf{x}]^+$	$= \mathbf{x}^+$
$[c \cdot e]^-$	$= c \cdot [e]^-$	$[c \cdot e]^+$	$= c \cdot [e]^+$
$[-e]^-$	$= [e]^+$	$[-e]^+$	$= [e]^-$
$[e_1 \sqcup e_2]^-$	$= [e_1]^- \vee [e_2]^-$	$[e_1 \sqcup e_2]^+$	$= [e_1]^+ \vee [e_2]^+$
$[e_1 + e_2]^-$	$= [e_1]^- + [e_2]^-$	$[e_1 + e_2]^+$	$= [e_1]^+ + [e_2]^+$

$[e \sqcap a]^-$	$= ([e]^+ + a^-); ([e]^- + a^+); [e]^- \wedge a^-$
$[e \sqcap a]^+$	$= ([e]^+ + a^-); ([e]^- + a^+); [e]^+ \wedge a^+$
$[e_1 \cdot e_2]^-$	$= ([e_1]^- \cdot^- [e_2]^- \wedge 0) \vee ([e_1]^+ \cdot^- [e_2]^+ \wedge 0) \vee [e_1]^- \cdot^+ [e_2]^+ \vee [e_1]^+ \cdot^+ [e_2]^-$
$[e_1 \cdot e_2]^+$	$= [e_1]^- \cdot^+ [e_2]^- \vee [e_1]^+ \cdot^+ [e_2]^+ \vee ([e_1]^- \cdot^- [e_2]^+ \wedge 0) \vee ([e_1]^+ \cdot^- [e_2]^- \wedge 0)$

We have:

Proposition 3. *Assume that $\mathcal{E}$ is a system of equations over the complete lattice of intervals, and $\mathcal{E}^{\pm}$ is the corresponding system for the negated lower and upper interval bounds of values for the variables of $\mathcal{E}$. Let μ and $\mu^{\pm}$ denote the least solutions of $\mathcal{E}$ and $\mathcal{E}^{\pm}$, respectively. Then for every variable $\mathbf{x}$ of $\mathcal{E}$, $(\mu(\mathbf{x}))^{-} = \mu^{\pm}(\mathbf{x}^{-})$ and $(\mu(\mathbf{x}))^{+} = \mu^{\pm}(\mathbf{x}^{+})$.* □

Proposition 3 follows by standard fixpoint induction. By Proposition 3, computing least solutions of systems of interval equations reduces to computing least solutions of systems of equations over $\mathcal{Z}$ using the BF operators maximum, addition, multiplication with positive constants, positive and negative multiplications together with minima with constants. Thus, theorem 3 is applicable, and we obtain:

Theorem 4. *The least solution of a system $\mathcal{E}$ of interval equations can be computed in time $\mathcal{O}(|\mathcal{E}|^3)$.*

Note that before application of theorem 3, we must instroduce auxiliary variables for simplifying complex interval expressions in right-hand sides of $\mathcal{E}$. Furthermore, the transformations $[.]^{-}$ and $[.]^{+}$ may produce composite expressions which we again decompose by means of auxiliary variables. The number of these fresh variables, however, is linear in the number of occurring multiplications and thus altogether bounded by $\mathcal{O}(|\mathcal{E}|)$.

7 Conclusion

We presented a cubic time algorithm for solving systems of integer equations where minimum is restricted to always have at least one constant argument. The methods relied on a subtle generalization of the Bellman-Ford algorithm for computing shortest paths in presence of positive and negative edge weights. We also observed that this algorithm is still applicable when right-hand sides of equations not only contain maxima, addition and multiplication with constants, but additionally use positive and negative multiplications.

In the second step, we showed how solving systems of interval equations with addition, full multiplication and intersection with constant intervals can be reduced to solving systems of integer equations. In particular, the restricted variants of multiplication allowed us to simulate full interval multiplication as well as to construct tests whether or not the intersection of an interval with a constant interval is empty. The one hand, our methods thus clarifies the upper complexity bound for solving systems of interval equations with intersection with constant intervals as presented by Su and Wagner [11]; on the other hand the approach generalizes the system of equations considered in [11] by additionally allowing full multiplication of intervals.

Our algorithms were designed to be *uniform*, i.e., have run-times independent of occurring numbers — given that arithmetic operations are counted as $\mathcal{O}(1)$. This is a reasonable assumption when multiplication is allowed with constants only. It is also

reasonable in presence of full multiplication for intervals — given that numbers are from a fixed finite range only.

In [5], the ideas presented here have been extended to work also for systems of interval equations with full multiplication as well as with arbitrary intersections.

References

1. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms. 2nd Edition*. MIT Press, Cambridge, MA, U.S.A., 2001.
2. P. Cousot and R. Cousot. Static Determination of Dynamic Properties of Programs. In *Second Int. Symp. on Programming*, pages 106–130. Dunod, Paris, France, 1976.
3. P. Cousot and R. Cousot. Comparison of the Galois Connection and Widening/Narrowing Approaches to Abstract Interpretation. JTASPEFL '91, Bordeaux. *BIGRE*, 74:107–110, Oct. 1991.
4. P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. The ASTRÉE Analyser. In *European Symposium on Programming (ESOP)*, volume 3444 of *LNCS*, pages 21–30. Springer, 2005.
5. T. Gawlitza and H. Seidl. Precise Fixpoint Computation Through Strategy Iteration. In *European Symposium on Programming (ESOP)*, pages 300–315. Springer Verlag, LNCS 4421, 2007.
6. D. E. Knuth. A Generalization of Dijkstra's algorithm. *Information Processing Letters (IPL)*, 6(1):1–5, 1977.
7. J. Leroux and G. Sutre. Accelerated Data-Flow Analysis. In *Static Analysis, 14th Int. Symp. (SAS)*, pages 184–199. LNCS 4634, Springer, 2007.
8. A. Miné. Relational Abstract Domains for the Detection of Floating-Point Run-Time Errors. In *European Symposium on Programming (ESOP)*, volume 2986 of *LNCS*, pages 3–17. Springer, 2004.
9. A. Miné. Symbolic Methods to Enhance the Precision of Numerical Abstract Domains. In *Verification, Model Checking, and Abstract Interpretation, 7th Int. Conf. (VMCAI)*, pages 348–363. LNCS 3855, Springer Verlag, 2006.
10. H. Seidl. Least and Greatest Solutions of Equations over $\mathcal{N}$. *Nordic Journal of Computing (NJC)*, 3(1):41–62, 1996.
11. Z. Su and D. Wagner. A Class of Polynomially Solvable Range Constraints for Interval Analysis Without Widenings. *Theor. Comput. Sci. (TCS)*, 345(1):122–138, 2005.