

Precise Fixpoint Computation Through Strategy Iteration

Thomas Gawlitza¹ and Helmut Seidl¹

TU München, Institut für Informatik, I2
85748 München, Germany
{gawlitza, seidl}@in.tum.de

Abstract. We present a practical algorithm for computing least solutions of systems of equations over the integers with addition, multiplication with positive constants, maximum and minimum. The algorithm is based on strategy iteration. Its run-time (w.r.t. the uniform cost measure) is independent of the sizes of occurring numbers. We apply our technique to solve systems of interval equations. In particular, we show how arbitrary intersections as well as full interval multiplication in interval equations can be dealt with precisely.

1 Introduction

In this paper we are interested in computing the precise least solutions of systems of interval equations using addition, multiplication, intersection and union [4, 11, 12]. Instead of doing so directly, we first consider the simpler problem of solving systems of equations over the integers using the operations addition, multiplication with positive constants, minimum and maximum. In fact, this computational problem can be considered as “one half” of precisely solving systems of equations over the interval domain: we simply may represent the value a by the interval $[-\infty, a]$. Thus, every method for computing precise least solutions of interval equations can be used to determine least solutions of integer equations. At least in absence of full multiplication, there is also a reduction in the opposite direction: solving interval equations precisely can be reduced to solving systems of integer equations as well.

Precise interval analysis has recently been considered by Su and Wagner [16] who propose a polynomial-time algorithm in case that one argument of every multiplication and intersection is constant. A clarified and improved version of this algorithm is presented in [6]. Since the linear ordering of integers has infinite ascending chains, ordinary fixpoint iteration will not result in terminating algorithms. For the lucky case where all numbers are non-negative, polynomial fixpoint algorithms are provided in [15]. In presence of general minima as well as negative numbers, no practical precise methods have been suggested so far. Clearly, we could apply general techniques such as the widening and narrowing approach of Cousot and Cousot [5]. While often returning amazingly good results, widening and narrowing is not guaranteed to compute the *least* solution of an equation system. Recently, *strategy iteration* has been proposed as an alternative method for approximative abstract interpretation in [3] where also conditions are derived under which least solutions can be obtained. Strategy iteration has been introduced by Howard for solving stochastic control problems [8, 14] and is also applied to

zero-sum two player games [7, 13, 17] or fixpoints of min-max-plus systems [2]. In general, strategy iteration will find *some* fixpoint which in the context of program analysis thus provides a safe over-approximation. For *expanding systems* the returned solution, though, is not always guaranteed to be the least possible [3]. Given that strategy iteration finds some fixpoint, we can be sure to have reached the *least* one if fixpoints are *unique*. One instance of this principle are fixpoint equations over Banach spaces where the transformation induced by right-hand sides is *contracting*. This is the reason why strategy iteration is nicely applicable to *discounted* mean-payoff games. Discounting, however, relies on exact arithmetic on potentially large numbers [18].

Here, we propose an approach based on an *instrumentation* of the underlying lattice. We first consider the simpler case of equation systems over the complete lattice of integers (extended with $\pm\infty$) where right-hand sides use addition, multiplication with positive constants as well as minimum and maximum. The instrumentation is meant to count the number of accesses to variables during fixpoint iteration. By itself, this instrumentation is not sufficient to guarantee uniqueness of fixpoints. It is sufficient, though, to guarantee for systems without maximum operators to admit at most one solution which maps all variables to values exceeding $-\infty$. This observation allows us to apply the generalization of the Bellman-Ford algorithm from [6] to compute this unique solution efficiently. Together with a suitable strategy iteration, we thus obtain an exact method for solving integer equations. This method vastly generalizes the results from [9, 15] which are only applicable to systems of equations without negative numbers. Along the lines of [6], our technique for systems of integer equations provides us with a *precise* algorithm for interval equations. This basic approach, however, can only handle equations where multiplication is always with constant intervals. Beyond that, we also provide a technically non-trivial extension resulting in a precise method also for systems of interval equations where arbitrary multiplication is allowed.

All our algorithms are *uniform*, i.e., their numbers of arithmetic operations do not depend on the numbers occurring in the systems. Also, they return *precise* answers and thus do not rely on widening or narrowing. Our implementation also indicates that the algorithm is decently efficient even on rather large systems of interval equations. The rest of the paper is organized as follows. In section 2, we introduce basic notions for systems of equations over the integers. In section 3, we present our instrumentation technique for integers and show how to construct a strategy iteration algorithm based on max strategies to compute the precise least solution of a system of integer equations containing multiplication with positive constants, addition, minimum and maximum. In section 4, we apply and generalize these methods to obtain algorithms for computing precise least solutions of interval equations. This section presents our novel techniques for precisely dealing with full multiplication in interval equations.

2 Notation and Basic Concepts

In the beginning, we are interested in solving systems of equations over the complete lattice of integers $\mathcal{Z} = \mathbb{Z} \cup \{-\infty, \infty\}$ equipped with the natural ordering

$$-\infty < \dots < -2 < -1 < 0 < 1 < 2 < \dots < \infty.$$

On \mathcal{Z} , we consider the operations addition, multiplication with positive constants, minimum “ \wedge ” and maximum “ \vee ” extended to operands “ $-\infty$ ” and “ ∞ ”. As usual, addition and multiplication are extended as follows:

$$\begin{array}{llll}
x + (-\infty) = (-\infty) + x = -\infty, & 0 \cdot x = x \cdot 0 = 0 & \text{for all } x & \\
x \cdot (-\infty) = (-\infty) \cdot x = -\infty, & x \cdot \infty = \infty \cdot x = \infty & \text{for all } x > 0 & \\
x \cdot (-\infty) = (-\infty) \cdot x = \infty, & x \cdot \infty = \infty \cdot x = -\infty & \text{for all } x < 0 & \\
x + \infty = \infty + x = \infty & & \text{for all } x \neq -\infty &
\end{array}$$

A system of integer equations is a sequence of equations $\mathbf{x}_i = e_i$ for $i = 1, \dots, n$, where the variables \mathbf{x}_i on the left-hand sides are pairwise distinct and the right-hand sides e_i are expressions e built up from constants and variables by means of our operations, i.e., adhere to the following grammar:

$$e ::= a \mid \mathbf{x}_i \mid e'_1 + e'_2 \mid b \cdot e \mid e'_1 \vee e'_2 \mid e'_1 \wedge e'_2$$

where $a \in \mathcal{Z}$ and $b > 0$. The set of variables of the system under consideration will be denoted by \mathbf{X} in the following. In [15] polynomial algorithms for computing least solutions are presented for similar systems — but only when computing least solutions over nonnegative integers. In [6], also negative integers are allowed. Minima, however, are only considered when one argument is constant. Here, we lift the latter restriction. For a *variable assignment* $\mu : \mathbf{X} \rightarrow \mathcal{Z}$ an expression e is mapped to a value $\llbracket e \rrbracket \mu \in \mathcal{Z}$:

$$\begin{array}{ll}
\llbracket a \rrbracket \mu & = a \\
\llbracket e'_1 + e'_2 \rrbracket \mu & = \llbracket e'_1 \rrbracket \mu + \llbracket e'_2 \rrbracket \mu \\
\llbracket e'_1 \wedge e'_2 \rrbracket \mu & = \llbracket e'_1 \rrbracket \mu \wedge \llbracket e'_2 \rrbracket \mu \\
\llbracket \mathbf{x}_j \rrbracket \mu & = \mu(\mathbf{x}_j) \\
\llbracket b \cdot e'_2 \rrbracket \mu & = b \cdot \llbracket e'_2 \rrbracket \mu
\end{array}$$

where $a \in \mathcal{Z}$ and $b > 0$. As usual, a *solution* is a variable assignment μ which satisfies all equations of a system \mathcal{E} , i.e. $\mu(\mathbf{x}_i) = \llbracket e_i \rrbracket \mu$ for all i . Since every right-hand side e_i induces a monotonic function $\llbracket e_i \rrbracket$, every system \mathcal{E} has a unique least solution.

This least solution can be computed by performing ordinary fixpoint iteration over the finite lattice $\{-\infty < a < \dots < b < \infty\}$ for suitable bounds $a \leq b$. This results in practical algorithms only if reasonably small bounds a, b to the values of variables can be revealed. Here, our goal is to exhibit practical algorithms whose run-time¹ is *independent* of the sizes of involved numbers. We call such algorithms *uniform*. The uniform run-time therefore only depends on structural properties of the equation system. In particular, we define the *size* $|\mathcal{E}|$ of \mathcal{E} as the number of variables plus the sum of expression sizes of right-hand sides.

3 Computing Least Solutions

An interesting approach for constructing uniform fixpoint algorithms is *strategy iteration* as proposed by Costan et al. [3]. Rephrased for a system \mathcal{E} of integer equations, they let a min strategy π select one of the e_i in every occurring minimum expression

¹ w.r.t. a uniform cost measure which counts every arithmetic operation as well as comparisons on integers as $\mathcal{O}(1)$.

$e_1 \wedge e_2$. The least solution μ_π of the resulting system then is guaranteed to be an *upper* bound to the least solution of \mathcal{E} . If μ_π is not a solution of \mathcal{E} , the strategy can be *improved*. If on the other hand, μ_π is a solution of \mathcal{E} , the iteration terminates.

Example 1 (From [3]). Consider the system consisting of $\mathbf{x} = \mathbf{y} \wedge 1$ and $\mathbf{y} = 2\mathbf{x} \vee -1$. A min strategy might select the expression 1 in the first equation resulting in:

$$\mathbf{x} = 1 \qquad \mathbf{y} = 2\mathbf{x} \vee -1$$

whose least solution maps \mathbf{x} to 1 and \mathbf{y} to 2, which is a solution of the original system. Note, however, that the *least* solution of the original system maps \mathbf{x} and \mathbf{y} to -1 . \square

As indicated by example 1, strategy iteration based on min strategies may not necessarily result in least solutions. Our idea therefore is to rely on *max* strategies instead which select one of the arguments of every maximum expression in the equation system. This alone, however, is not a meaningful approach since least solutions of equation systems with minimum alone will often have just trivial least solutions.

Example 2. Consider the system: $\mathbf{x} = 0 \vee \mathbf{x} + 1$. A (max) strategy might either select the expression 0 or the expression $\mathbf{x} + 1$ in the right-hand side. In the first case, the least solution of the resulting system maps \mathbf{x} to 0, whereas in the second case, the least solution maps \mathbf{x} to $-\infty$. The least solution of the original system, however, is given by $\mu^* = \{\mathbf{x} \mapsto \infty\}$. \square

Our extra idea here therefore is to *instrument* the underlying lattice in such a way that we can rely on *particular* solutions of conjunctive systems for approximating the least solution of the original system, namely those which do not map variables to $-\infty$. Due to our instrumentation, these solutions happen to be *unique* and thus are computable by *greatest* fixpoint iteration. The idea of the instrumentation is to provide an extra component which, besides the reached value in \mathbb{Z} , additionally records the minimal (nonnegative) depth of recursive descents into variables to produce this value. Accordingly, the instrumented domain is given by $\mathcal{D} = \mathbb{D} \cup \{-\infty, \infty\}$ where $\mathbb{D} = \mathbb{Z} \times \mathbb{N}$ is the set of *finite* elements of \mathcal{D} and $-\infty$ and ∞ are the least and greatest elements, respectively. The ordering on \mathbb{D} is given by:

$$(a_1, j_1) \leq (a_2, j_2) \quad \text{iff} \quad a_1 < a_2 \vee (a_1 = a_2 \wedge j_1 \geq j_2)$$

This ordering is again linear. The operators “+” and “ \cdot ” over \mathcal{D} behave similar to the corresponding operators over \mathcal{Z} when applied to $-\infty$ or ∞ . For finite elements, we define:

$$(a_1, j_1) + (a_2, j_2) = (a_1 + a_2, j_1 \vee j_2)$$

$$b \cdot (a, j) = \begin{cases} -\infty & \text{if } b = \infty \text{ and } a < 0 \\ (a, j) & \text{if } b = \infty \text{ and } a = 0 \\ \infty & \text{if } b = \infty \text{ and } a > 0 \\ (b \cdot a, j) & \text{if } \infty > b > 0 \end{cases}$$

Furthermore we introduce a function inc defined by $\text{inc}(-\infty) = -\infty$, $\text{inc}(\infty) = \infty$ and $\text{inc}((a, j)) = (a, j + 1)$. The function inc distributes over $+$, \cdot , \wedge and \vee , i.e.:

$$\begin{aligned} \text{inc}(x + y) &= \text{inc}(x) + \text{inc}(y) & \text{inc}(b \cdot x) &= b \cdot \text{inc}(y) \\ \text{inc}(x \wedge y) &= \text{inc}(x) \wedge \text{inc}(y) & \text{inc}(x \vee y) &= \text{inc}(x) \vee \text{inc}(y) \end{aligned}$$

Algorithm 1 Generalized Bellman-Ford algorithm

```
for  $i = 1$  to  $n$  do  $\mu(\mathbf{x}_i) \leftarrow \infty$ ;  
for  $j = 1$  to  $n$  do  
  for  $i = 1$  to  $n$  do  $\mu(\mathbf{x}_i) \leftarrow \llbracket e_i \rrbracket^\sharp \mu$ ;  
for  $j = 1$  to  $n$  do  
  for  $i = 1$  to  $n$  do if  $\llbracket e_i \rrbracket^\sharp \mu < \mu(\mathbf{x}_i)$  then  $\mu(\mathbf{x}_i) \leftarrow -\infty$ ;  
return  $\mu$ ;
```

The evaluation of an expression e (possibly containing applications of inc) over \mathcal{D} will be denoted by $\llbracket e \rrbracket^\sharp$. In order to instrument the equation system \mathcal{E} to additionally record accesses to variables, we define a lifting operation as follows. For every expression e over \mathcal{Z} , the corresponding *lifted* expression $[e]^\sharp$ is obtained from e by replacing every constant $a \in \mathbb{Z}$ with $(a, 0)$ and every variable \mathbf{x}_j with $\text{inc}(\mathbf{x}_j)$. Let \mathcal{E}^\sharp denote the corresponding lifted system over \mathcal{D} where every equation $\mathbf{x}_i = e_i$ is replaced with $\mathbf{x}_i = [e_i]^\sharp$ for $i = 1, \dots, n$. Thus, in a lifted system, every occurrence of a variable in a right-hand side is guarded by a call to the function inc . We verify:

Theorem 1. *Assume that \mathcal{E} is a system of integer equations with least solution μ^* . Let μ^\sharp denote the least solution of the corresponding lifted system \mathcal{E}^\sharp . Then for every variable \mathbf{x}_i the following holds:*

1. $\mu^*(\mathbf{x}_i) = \mu^\sharp(\mathbf{x}_i)$ whenever $\mu^*(\mathbf{x}_i) \in \{-\infty, \infty\}$;
2. $\mu^\sharp(\mathbf{x}_i) = (\mu^*(\mathbf{x}_i), j)$ for some $j \in \mathbb{N}$ whenever $\mu^*(\mathbf{x}_i) \in \mathbb{Z}$. □

Given the above theorem, our goal is to compute the least solution of the lifted equation system \mathcal{E}^\sharp over the instrumented lattice \mathcal{D} . Thereby, we first consider the case, in which no right-hand side of \mathcal{E}^\sharp contains a maximum. We call such systems *conjunctive*. The *greatest* solution of a conjunctive equation system turns out to be easily computable.

Theorem 2. *Let \mathcal{E}^\sharp denote a conjunctive lifted system of integer equations with n variables and greatest solution μ^\sharp . Then*

1. μ^\sharp can be computed in time $\mathcal{O}(n \cdot |\mathcal{E}^\sharp|)$;
2. If $\mu^\sharp(\mathbf{x}_i) \in \mathbb{D}$, then $\mu^\sharp(\mathbf{x}_i) = (a, j)$ for some $0 \leq j \leq n$.

Proof. Here, we rely on alg. 1, which is an adaption of the Bellman-Ford algorithm. In particular, if the greatest solution μ^\sharp does not map variables to $-\infty$, then just n rounds of Round Robin iterations suffice to determine μ^\sharp . For a correctness proof, we refer to [6] where a similar result is shown for least solutions of *disjunctive* systems, i.e., systems having no occurrences of minimum operators. The use of Gaussian elimination to determine μ^\sharp , also reveals the second statement. □

We call a variable assignment μ *feasible* iff $\mu(\mathbf{x}_i) > -\infty$ for all variables \mathbf{x}_i . Our key result for conjunctive lifted systems is:

Theorem 3. *Let \mathcal{E}^\sharp denote a conjunctive lifted system of integer equations with greatest solution μ^\sharp . If \mathcal{E}^\sharp has a feasible solution μ , then $\mu = \mu^\sharp$.*

Proof. Obviously, if there exists a feasible solution μ , then the greatest solution μ^\sharp is also feasible. To show, that μ^\sharp is *the only* feasible solution, we first consider a system \mathcal{E}^\sharp which consists of a single equation:

$$\mathbf{x}_1 = a_0 \wedge b_1 \cdot \text{inc}^{k_1}(\mathbf{x}_1) + a_1 \wedge \dots \wedge b_r \cdot \text{inc}^{k_r}(\mathbf{x}_1) + a_r$$

where $b_j > 0$, $k_j > 0$ and $a_0, a_j \in \mathcal{Z}$. Note that a_0 can also equal ∞ . If $b_1 \cdot \text{inc}^{k_1}(a_0) + a_1 \wedge \dots \wedge b_r \cdot \text{inc}^{k_r}(a_0) + a_r \geq a_0$, then a_0 is the greatest solution and the only one exceeding $-\infty$. For a contradiction, assume $z > -\infty$ were another solution, i.e., $-\infty < z < a_0$. Then $b_1 \cdot \text{inc}^{k_1}(z) + a_1 \wedge \dots \wedge b_r \cdot \text{inc}^{k_r}(z) + a_r < a_0$. Since the second component of $b_1 \cdot \text{inc}^{k_1}(z) + a_1 \wedge \dots \wedge b_r \cdot \text{inc}^{k_r}(z) + a_r$ exceeds the second component of z , z cannot be a solution. If on the other hand, $b_1 \cdot \text{inc}^{k_1}(a_0) + a_1 \wedge \dots \wedge b_r \cdot \text{inc}^{k_r}(a_0) + a_r < a_0$, the equation has $-\infty$ as only solution.

Now consider an arbitrary conjunctive equation system \mathcal{E}^\sharp with feasible solutions. We proceed by induction on the number n of variables in right-hand sides. If $n = 0$, the statement trivially holds. So let $n > 0$, and let \mathbf{x}_i be a variable that occurs in a right-hand side of \mathcal{E}^\sharp . Consider the equation $\mathbf{x}_i = e_i$ of \mathcal{E}^\sharp . Our goal is to construct an expression e' without occurrences of \mathbf{x}_i which is equivalent to e_i . Then we replace all occurrences of \mathbf{x}_i in right-hand sides with e' . By using distributivity, we have:

$$e_i = e'_0 \wedge b_1 \cdot \text{inc}^{k_1}(\mathbf{x}_i) + e'_1 \wedge \dots \wedge b_r \cdot \text{inc}^{k_r}(\mathbf{x}_i) + e'_r$$

for suitable constants $b_j > 0$, $k_j > 0$ and expressions e'_0, e'_j not containing \mathbf{x}_i . Then we choose e' as e'_0 . For an arbitrary feasible solution μ' of \mathcal{E}^\sharp , let ρ denote the substitution $\rho(\mathbf{x}_i) = \mathbf{x}_i$ and $\rho(\mathbf{x}_j) = \mu'(\mathbf{x}_j)$ for $j \neq i$. According to the single equation case, the single equation $\mathbf{x}_i = e_i \rho$ has a unique feasible solution which is given by $\mathbf{x}_i = e' \rho$. Thus, we can substitute every occurrence of \mathbf{x}_i in right-hand sides of \mathcal{E}^\sharp with e' to obtain a system of equations which has a superset of feasible solutions of \mathcal{E}^\sharp — but one variable less in right-hand sides. Then the assertion follows with the induction hypothesis. \square

Since conjunctive lifted systems with a feasible solution have exactly one feasible solution which thus is equal to the greatest solution², we can apply alg. 1 to compute it. Next, we show how conjunctive systems with feasible solutions can be used to determine the least solution of a lifted system \mathcal{E}^\sharp . For that, let $M(\mathcal{E}^\sharp)$ denote the set of all maximum expressions in \mathcal{E}^\sharp . A (max) *strategy* π is a function mapping every expression $e_1 \vee e_2$ in $M(\mathcal{E}^\sharp)$ to one of the subexpressions e_1, e_2 . Let $\mathcal{E}^\sharp(\pi)$ denote the conjunctive system obtained from \mathcal{E}^\sharp by recursively replacing every maximum expression with the respective subexpression selected by π .

Now assume that we are given a strategy π such that the greatest solution μ_π of $\mathcal{E}^\sharp(\pi)$ is feasible and a lower bound to the least solution of \mathcal{E}^\sharp , i.e., $\mu_\pi \leq \mu^\sharp$. As a consequence μ^\sharp must also be feasible. If μ_π is a solution of \mathcal{E}^\sharp , then we have already found the least solution of \mathcal{E}^\sharp and are done. Otherwise, some expression $e_1 \vee e_2$ in $M(\mathcal{E}^\sharp)$ exists where π does not select the expression e_i with $\llbracket e_i \rrbracket^\sharp \mu_\pi > \llbracket e_{3-i} \rrbracket^\sharp \mu_\pi$. In order to *improve* the strategy, we may, e.g., pursue the policy to modify π at all such expressions simultaneously. Thus, we define the improved strategy $P(\mu_\pi)$ by:

$$P(\mu_\pi)(e_1 \vee e_2) = \begin{cases} e_1 & \text{if } \llbracket e_1 \rrbracket^\sharp \mu_\pi \geq \llbracket e_2 \rrbracket^\sharp \mu_\pi \\ e_2 & \text{if } \llbracket e_1 \rrbracket^\sharp \mu_\pi < \llbracket e_2 \rrbracket^\sharp \mu_\pi \end{cases}$$

Proposition 1. *Let μ^\sharp denote the least solution of the system \mathcal{E}^\sharp , and assume that $\mu < \mu^\sharp$ is a feasible variable assignment. Let π be the strategy $\pi = P(\mu)$ and μ' denote the greatest solution of $\mathcal{E}^\sharp(\pi)$. Then $\mu < \mu' \leq \mu^\sharp$. \square*

² their least solution still might not be feasible.

Algorithm 2 Strategy Improvement Algorithm

$\mu \leftarrow \mu_0$;
while (μ not solution of \mathcal{E}^\sharp) {
 $\pi \leftarrow P(\mu)$; $\mu \leftarrow$ greatest solution of $\mathcal{E}^\sharp(\pi)$;
}
return μ ;

Proof. Let μ_1 denote the variable assignment defined by $\mu_1(\mathbf{x}_i) = \llbracket e_i \rrbracket^\sharp \mu$ for every equation $\mathbf{x}_i = e_i$ of \mathcal{E}^\sharp . By construction, $\mu < \mu_1 \leq \mu^\sharp$. We claim that $\mu_1 \leq \mu'$. By monotonicity, we have for every equation $\mathbf{x}_i = e'_i$ of $\mathcal{E}^\sharp(\pi)$, $\llbracket e'_i \rrbracket^\sharp \mu_1 \geq \llbracket e'_i \rrbracket^\sharp \mu = \llbracket e_i \rrbracket^\sharp \mu = \mu_1(\mathbf{x}_i)$. Thus, μ_1 is a pre-fixpoint of $\mathcal{E}^\sharp(\pi)$. Since, by the fixpoint theorem of Knaster-Tarski, the greatest solution of a system is an upper bound for all pre-fixpoints, $\mu_1 \leq \mu'$ which is the claim above. Since $\mathcal{E}^\sharp(\pi)$ has just one feasible solution, μ' is also the least solution of $\mathcal{E}^\sharp(\pi)$ exceeding μ . Therefore, μ' is bounded by the least solution of \mathcal{E}^\sharp exceeding μ . Since μ is bounded by μ^\sharp , the latter equals μ^\sharp . Thus $\mu' \leq \mu^\sharp$. \square

Assume that \mathcal{E}^\sharp is an equation system for which we are given an initial feasible variable assignment $\mu_0 \leq \mu^\sharp$. We propose strategy improvement algorithm 2, that, given \mathcal{E}^\sharp and μ_0 returns the least solution of \mathcal{E}^\sharp . By proposition 1, the sequence of variable assignments μ constructed by alg. 2 forms a strictly increasing chain. Since every variable assignment in this strictly increasing chain is the greatest solution of $\mathcal{E}^\sharp(\pi)$ for some strategy π , every strategy occurs at most once. Since algorithm 2 terminates with a solution of the system, proposition 1 also implies that it returns the least solution.

Our approach is remarkable in that it does not rely on discounting as, e.g., the related algorithms in [13, 2] for computing game values of mean-payoff games. Instead, we use ordinary arithmetic on numbers of length $\mathcal{O}(n \cdot \log(B))$ where B is the maximal absolute value of a finite constant occurring in \mathcal{E} .

So far we have assumed that the least solution of \mathcal{E}^\sharp is feasible and that we have an initial feasible variable assignment $\mu_0 \leq \mu^\sharp$ at hand. We have not yet revealed how to arrive at such a variable assignment. Note that we cannot ignore this problem and start with any strategy instead.

Example 3. Consider the lifted system $\mathbf{x} = (\text{inc}(\mathbf{x}) \wedge (0, 0)) \vee (0, 0)$. The strategy π which replaces the maximum-expression with $\text{inc}(\mathbf{x}) \wedge (0, 0)$ leads to the conjunctive system $\mathbf{x} = \text{inc}(\mathbf{x}) \wedge (0, 0)$ with a unique solution which maps \mathbf{x} to $-\infty$. \square

Our problem therefore is to come up with a first lower approximation to the least solution which is feasible. For that, consider again a lifted system \mathcal{E}^\sharp with n variables and least solution μ^\sharp . Our solution is to initially perform n rounds of Round-Robin iteration. A related idea seems also implicit in section 5 of [3] in order to speed up strategy iteration in general. Let μ_0 denote the variable assignment resulting from the initial iteration. By construction, $\mu_0 \leq \mu^\sharp$. A closer look also reveals that $\mu_0(\mathbf{x}_i) = -\infty$ iff $\mu^\sharp(\mathbf{x}_i) = -\infty$. Thus, we can use the variable assignment μ_0 to remove all variables from \mathcal{E}^\sharp that are mapped to $-\infty$ by the least solution. This means, that we replace every expression e where $\llbracket e \rrbracket \mu_0 = -\infty$ with $-\infty$ and remove every equation $\mathbf{x}_i = e_i$ where $\mu_0(\mathbf{x}_i) = -\infty$. For the resulting system we then can use μ_0 as an initial feasible variable assignment. We remark that we also can use a work-list-based approach to determine an initial variable assignment $\mu_0 \leq \mu^\sharp$ s.t. $\mu_0(\mathbf{x}_i) = -\infty$ iff $\mu^\sharp(\mathbf{x}_i) = -\infty$.

Example 4. Consider the lifted system

$$\mathbf{x} = (\text{inc}(\mathbf{x}) + \text{inc}(\mathbf{y})) \vee (0, 0) \quad \mathbf{y} = (\text{inc}(\mathbf{x}) + (1, 0)) \wedge (10, 0).$$

Two rounds of Round-Robin iteration results in the feasible variable assignment μ_0 that maps \mathbf{x} to $(2, 2)$ and \mathbf{y} to $(3, 3)$. Also, μ_0 results in a strategy which selects the first argument expression of the max expression. This results in the conjunctive system

$$\mathbf{x} = \text{inc}(\mathbf{x}) + \text{inc}(\mathbf{y}) \quad \mathbf{y} = \text{inc}(\mathbf{x}) + (1, 0) \wedge (10, 0)$$

with greatest solution μ_1 that maps \mathbf{x} to ∞ and \mathbf{y} to $(10, 0)$: which corresponds to the least solution of the original system. \square

For a precise characterization of the run-time, let $II(m)$ denote the maximal number of updates of strategies necessary for systems with m maximum expressions. We have:

Theorem 4. *The least solution of a system \mathcal{E} of integer equations with n variables and m maximum expressions can be computed uniformly in time $\mathcal{O}(n \cdot |\mathcal{E}| \cdot II(m))$. \square*

The factor $n \cdot |\mathcal{E}|$ accounts for computing greatest solutions of conjunctive systems through n rounds of Round Robin iteration. Practical implementations, though, might use variants of work-list-based fixpoint iteration instead which at least practically will terminate much earlier. Finally, there is the factor $II(m)$. At every maximum which has at least one constant argument, the strategy can be improved at most once. At general maximum subexpressions, the situation is less clear. The preliminary experience with our implementation as well as all practical experiments with strategy iteration we know of seem to indicate that the number of strategy improvements $II(m)$ (at least practically) grows quite slowly in the number m of maxima. The systems up to 100.000 variables, e.g., which we tried used less than 20 iterations! Interestingly, though, it is still open whether (or: under which circumstances) the trivial upper bound of 2^m for $II(m)$ can be significantly improved [17, 1]. Concerning the complexity, strategy iteration algorithms thus can be compared with the simplex method for linear programming: many known variants of the latter method work very well in practice even for large scale applications with thousands of variables. We are better off, though, with strategy iteration: while for many variants of the simplex algorithm, inputs are known on which the algorithm needs exponentially many pivot operations (see, e.g., [10] for a nice overview), no inputs are known for strategy iteration using more than a *linear* number of iterations.

4 Interval Analysis

In this section, we explain how the fixpoint methods from section 3 can be used to compute precise least solutions of systems of interval equations. Let \mathcal{I} denote the complete lattice of intervals partially ordered by the subset relation (here denoted by “ \sqsubseteq ”). Thus,

$$\mathcal{I} = \{\emptyset\} \cup \{[l, u] \in (\mathbb{Z} \cup \{-\infty\}) \times (\mathbb{Z} \cup \{\infty\}) \mid l \leq u\}$$

where $[l, u]$ represents the interval $\{z \in \mathbb{Z} \mid l \leq z \leq u\}$. As usual, the empty interval \emptyset is the least element of \mathcal{I} , the greatest lower bound “ \sqcap ” of intervals is given by

their intersection while the least upper bound “ \sqcup ” for non-empty intervals is defined by $[l_1, u_1] \sqcup [l_2, u_2] = [l_1 \wedge l_2, u_1 \vee u_2]$. Addition and multiplication are given by:

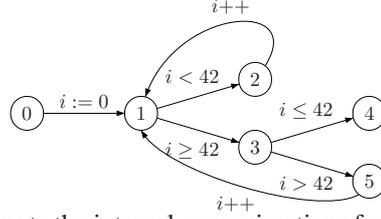
$$\begin{aligned} [l_1, u_1] + [l_2, u_2] &= [l_1 + l_2, u_1 + u_2] \\ [l_1, u_1] \cdot [l_2, u_2] &= [u_1 \cdot u_2 \wedge l_1 \cdot l_2 \wedge l_1 \cdot u_2 \wedge u_1 \cdot l_2, u_1 \cdot u_2 \vee l_1 \cdot l_2 \vee l_1 \cdot u_2 \vee u_1 \cdot l_2] \end{aligned}$$

We consider systems of equations $\mathbf{x}_i = e_i$ for $i = 1, \dots, n$ over intervals similar to the ones we have considered over \mathcal{Z} . Thus, we allow right-hand sides e_i of the form

$$e ::= I \mid \mathbf{x}_j \mid e'_1 \sqcup e'_2 \mid e'_1 \sqcap e'_2 \mid e'_1 + e'_2 \mid e'_1 \cdot e'_2$$

where $I \in \mathcal{I}$ denotes constant intervals. In [3], also postfix operators “ \uparrow ” and “ \downarrow ” are provided which preserve empty intervals and, when applied to a non-empty interval $[l, u]$ return $[l, u] \uparrow = [l, \infty]$ and $[l, u] \downarrow = [-\infty, u]$, respectively. We also could introduce an operator “ $;$ ” defined by $I_1 ; I_2 = \emptyset$ if $I_1 = \emptyset$ and $I_1 ; I_2 = I_2$ otherwise. This operator is useful for expressing reachability assumptions. We have omitted all three operators, since these can be defined by: $I \uparrow = I + [0, \infty]$, $I \downarrow = I + [-\infty, 0]$ and $I_1 ; I_2 = [0, 0] \cdot I_1 + I_2$. Since all right-hand sides of equations are monotonic, every system of interval equations has a unique least solution. Such systems can be used for determining safe ranges for the values of integer variables. Instead of formally introducing interval analysis, we illustrate this application by an example.

Example 5. Consider the following control-flow graph for which we want to infer the information, that program point 5 is unreachable:



Let i_k denote the interval approximations for the sets of values of variable i at program point $k = 0, \dots, 5$. This leads to the system :

$$\begin{aligned} i_1 &= [0, 0] \sqcup i_5 + [1, 1] \sqcup i_2 + [1, 1] & i_2 &= i_1 \sqcap [-\infty, 41] \\ i_3 &= i_1 \sqcap [42, \infty] & i_4 &= i_3 \sqcap [-\infty, 42] \\ i_5 &= i_3 \sqcap [43, \infty] \end{aligned}$$

It is not obvious how the usual widening and narrowing approach (with program point 1 as widening point) would identify program point 5 as unreachable. The least solution of the equation system, however, identifies program point 5 as unreachable. \square

Assume that \mathcal{E} is an equation system over \mathcal{I} . Our goal is to compute the least solution by decomposing \mathcal{E} into a system of integer equations for the upper and *negated* lower bounds. For every interval variable \mathbf{x}_i of \mathcal{E} , we therefore introduce two integer variables \mathbf{x}_i^+ , \mathbf{x}_i^- for the upper and *negated* lower bound of \mathbf{x}_i , respectively. Negating the lower bounds of intervals allows us to determine the values of the variables \mathbf{x}_i^+ as well as the values of the variables \mathbf{x}_i^- by means of *least* fixpoint iteration within the same

system of equations. Note, however, that upper and lower bounds of intervals are not independent. Interaction occurs at subexpressions which evaluate to the empty interval. Therefore, our construction will depend on information about the potential emptiness of the expressions in S where S is the set of all subexpressions of right-hand sides in \mathcal{E} . This information is specified by a valuation σ from S into the two-element lattice $D_2 = \{\perp, \top\}$ where $\perp < \top$. $\sigma(e) = \perp$ thereby indicates that the value of e is the empty interval, and $\sigma(e) = \top$ that the value of e is non-empty. The set of valuations form a complete lattice where the maximal length of a strictly ascending chain is bounded by $|S| \in \mathcal{O}(|\mathcal{E}|)$. Given a variable assignment μ , we can determine a valuation $\Sigma(\mu) : S \rightarrow D_2$ which is compatible with μ by:

$$\Sigma(\mu)(e) = \begin{cases} \perp & \text{if } \llbracket e \rrbracket \mu = \emptyset \\ \top & \text{if } \llbracket e \rrbracket \mu \neq \emptyset \end{cases}$$

Note that the valuation $\Sigma(\mu)$ monotonically depends on μ . We introduce the system \mathcal{E}_σ^\pm which is obtained from \mathcal{E} by replacing every interval equation $\mathbf{x}_i = e_i$ with the equations $\mathbf{x}_i^+ = [e_i]_\sigma^+$ and $\mathbf{x}_i^- = [e_i]_\sigma^-$ over \mathcal{Z} . Thereby, $[e]_\sigma^+ = [e]_\sigma^- = -\infty$ whenever $\sigma(e) = \perp$. Otherwise:

$$\begin{array}{ll} [\emptyset]_\sigma^+ & = -\infty & [\emptyset]_\sigma^- & = -\infty \\ [[l, u]]_\sigma^+ & = u & [[l, u]]_\sigma^- & = -l \\ [e_1 + e_2]_\sigma^+ & = [e_1]_\sigma^+ + [e_2]_\sigma^+ & [e_1 + e_2]_\sigma^- & = [e_1]_\sigma^- + [e_2]_\sigma^- \\ [e_1 \sqcup e_2]_\sigma^+ & = [e_1]_\sigma^+ \vee [e_2]_\sigma^+ & [e_1 \sqcup e_2]_\sigma^- & = [e_1]_\sigma^- \vee [e_2]_\sigma^- \\ [e_1 \sqcap e_2]_\sigma^+ & = [e_1]_\sigma^+ \wedge [e_2]_\sigma^+ & [e_1 \sqcap e_2]_\sigma^- & = [e_1]_\sigma^- \wedge [e_2]_\sigma^- \end{array}$$

The rules of the transformation do not yet deal with multiplication. Multiplication will be considered subsequently. In absence of multiplications, the least solution μ_σ^\pm of \mathcal{E}_σ^\pm can be computed with our methods from section 3. Once we are given a variable assignment μ for \mathcal{E}_σ^\pm , we obtain a variable assignment $[\mu]$ for the original system \mathcal{E} by³:

$$[\mu](\mathbf{x}_i) = \begin{cases} \emptyset & \text{if } \mu(\mathbf{x}_i^+) = \mu(\mathbf{x}_i^-) = -\infty \\ [-\mu(\mathbf{x}_i^-), \mu(\mathbf{x}_i^+)] & \text{if } \mu(\mathbf{x}_i^+), \mu(\mathbf{x}_i^-) > -\infty \end{cases}$$

For the correctness of our algorithm the following proposition is fundamental.

Proposition 2. *Assume that \mathcal{E} is a system of interval equations without multiplication whose least solution is μ^* . Let $\sigma : S \rightarrow D_2$ be a valuation. Assume \mathcal{E}_σ^\pm is the corresponding system of integer equations for the upper and negated lower bounds with least solution μ_σ^\pm . Then:*

1. $[\mu_\sigma^\pm] \sqsubseteq \mu^*$ whenever $\sigma \leq \Sigma(\mu^*)$;
2. $[\mu_\sigma^\pm] = \mu^*$ whenever $\sigma = \Sigma(\mu^*)$;
3. $\Sigma([\mu_\sigma^\pm]) > \sigma$ whenever $\sigma < \Sigma(\mu^*)$. □

Assertion 1 of proposition 2 guarantees for a possibly too small valuation σ , that the integer system will return a lower approximation to the least solution μ^* . Assertion 2

³ We do not need to define $[\mu](\mathbf{x}_i)$ for any of the remaining cases, since these will not occur in the algorithms to be presented below.

Algorithm 3 Algorithm for interval equations

```
 $\sigma \leftarrow \perp;$   
do {  
   $\mu_\sigma^\pm \leftarrow$  least solution of  $\mathcal{E}_\sigma^\pm$ ;  $\mu \leftarrow [\mu_\sigma^\pm]$ ;  $\sigma_{old} \leftarrow \sigma$ ;  $\sigma \leftarrow \Sigma(\mu)$ ;  
} while ( $\sigma \neq \sigma_{old}$ )  
return  $\mu$ ;
```

guarantees for precise σ that the integer system in deed recovers μ^* . Finally, assertion 3 assures that, as long as $\Sigma(\mu^*)$ has not been reached, the new valuation will be strictly larger than the old one. In light of proposition 2, it is now clear how our algorithm for computing the least solution μ^* of a system \mathcal{E} of interval equations should work. It starts with a valuation $\sigma = \perp$ from $S \rightarrow D_2$ that maps every subexpression e to \perp . Given a current valuation $\sigma \leq \Sigma(\mu^*)$, it computes the least solution of the integer system \mathcal{E}_σ^\pm . According to proposition 2, this will either reveal the least solution of \mathcal{E} , if σ already equals $\Sigma(\mu^*)$, or some further expressions evaluating to non-empty intervals, in the other case. In the latter case, σ is updated and the algorithm repeats. Thus, alg. 3 computes the least solution of a system of interval equations \mathcal{E} after $\mathcal{O}(|\mathcal{E}|)$ iterations. Before we consider multiplication, we illustrate alg. 3 by an example.

Example 6. Consider the following system of interval equations:

$$\mathbf{x} = (\mathbf{x} + [1, 1] \sqcap [0, 42]) \sqcup [10, 10]$$

In the first step the algorithm considers \mathcal{E}_\perp^\pm given by:

$$\mathbf{x}^+ = -\infty \vee 10 \quad \mathbf{x}^- = -\infty \vee -10$$

Using the obvious least solution the algorithm computes a valuation σ that returns \top for all expressions. Thus, we obtain \mathcal{E}_σ^\pm as:

$$\mathbf{x}^+ = (\mathbf{x}^+ + 1 \wedge 42) \vee 10 \quad \mathbf{x}^- = (\mathbf{x}^- + (-1) \wedge 0) \vee -10$$

Solving \mathcal{E}_σ^\pm reveals the least solution which maps \mathbf{x}^+ to 42 and \mathbf{x}^- to -10 . This corresponds to the least solution of \mathcal{E} which maps \mathbf{x} to $[10, 42]$. \square

We now extend the setting by multiplications where at least one argument is a constant non-empty interval $I \in \mathcal{I}$, i.e., every multiplication subexpression in right-hand sides is of the form $I \cdot e$ for $I \in \mathcal{I} \setminus \{\emptyset\}$. Therefore, we enrich the transformations $[\cdot]_\sigma^+$ and $[\cdot]_\sigma^-$ by defining $[I \cdot e]_\sigma^+$ and $[I \cdot e]_\sigma^-$ for $I = [l, u]$ and e an expression s.t. $\sigma(e) \neq \perp$ by:

$$[I \cdot e]_\sigma^+ = \begin{cases} l \cdot [e]_\sigma^+ \vee u \cdot [e]_\sigma^+ & \text{if } l \geq 0 \\ -l \cdot [e]_\sigma^- \vee u \cdot [e]_\sigma^+ & \text{if } l < 0, u \geq 0 \\ -l \cdot [e]_\sigma^- \vee -u \cdot [e]_\sigma^- & \text{if } u < 0 \end{cases}$$
$$[I \cdot e]_\sigma^- = \begin{cases} l \cdot [e]_\sigma^- \vee u \cdot [e]_\sigma^- & \text{if } l \geq 0 \\ -l \cdot [e]_\sigma^+ \vee u \cdot [e]_\sigma^- & \text{if } l < 0, u \geq 0 \\ -l \cdot [e]_\sigma^+ \vee -u \cdot [e]_\sigma^+ & \text{if } u < 0 \end{cases}$$

If $\sigma(e) = \perp$, then $[I \cdot e]_\sigma^+$ and $[I \cdot e]_\sigma^-$ are given as $-\infty$. Thus, we obtain right-hand sides in which multiplications with *nonnegative* constants occur. By simplifying expressions $0 \cdot e$ to 0, we obtain a system of integer equations as considered in section 3.

Example 7. Let \mathcal{E} be the system $\mathbf{x} = [-1, 0] \cdot \mathbf{x} \sqcup [2, 4]$. Assume that σ is the valuation which maps all expressions from \mathcal{E} to \top . Then the system \mathcal{E}_σ^\pm is given by:

$$\mathbf{x}^+ = 1 \cdot \mathbf{x}^- \vee 0 \cdot \mathbf{x}^+ \vee 4 \quad \mathbf{x}^- = 1 \cdot \mathbf{x}^+ \vee 0 \cdot \mathbf{x}^- \vee -2$$

The least solution of this system maps \mathbf{x}^+ and \mathbf{x}^- to 4. This corresponds to the least solution of \mathcal{E} which maps \mathbf{x} to $[-4, 4]$. \square

The resulting systems of integer equations for the upper and negated lower interval bounds still are of the form considered in section 3 and therefore can be solved by alg. 2. It turns out that proposition 2 still holds when multiplication with constant intervals is allowed. Thus, alg. 3 can be applied and we get the following important result:

Theorem 5. *Assume that \mathcal{E} is a system of interval equations with n variables, m occurrences of “ \sqcup ” and k multiplications in which at least one argument is a constant interval. The least solution of \mathcal{E} can be computed in time $\mathcal{O}(n \cdot |\mathcal{E}|^2 \cdot \Pi(2m + 2k))$. \square*

According to theorem 5, the complexity for solving interval equations consists of the complexity for iteratively solving integer systems until the number of variables receiving non-empty intervals remains stable. The result of theorem 5, though, is not yet completely satisfactory since it is not able to deal with *full multiplication* of intervals — meaning that an interval analysis based on theorem 5 is bound to treat general multiplication expressions conservatively. E.g. $x \cdot y$ have to be treated as $[-\infty, \infty]$. Dealing with full multiplication of intervals is non-trivial, though. Let $(x)^+$ and $(x)^-$ denote the upper and negated lower bound of an interval x . For non-empty intervals x, y the values $(xy)^+$ and $(xy)^-$ are given as:

$$\begin{aligned} (xy)^+ &= x^+y^+ \vee x^-y^- \vee -x^+y^- \vee -x^-y^+ \\ (xy)^- &= -x^+y^+ \vee -x^-y^- \vee x^+y^- \vee x^-y^+ \end{aligned}$$

Note that, in presence of positive and negative numbers, none of the individual products is monotonic. Fortunately, the necessary multiplications are *piecewise* distributive:

Proposition 3. *Assume $a_1, a_2, b \in \mathcal{Z}$. Then:*

1. $(a_1 \vee a_2)b = a_1b \vee a_2b$ as well as $(a_1 \wedge a_2)b = a_1b \wedge a_2b$ if $b \geq 0$;
2. $-(a_1 \vee a_2)b = -a_1b \vee -a_2b$ as well as $-(a_1 \wedge a_2)b = -a_1b \wedge -a_2b$ if $b \leq 0$. \square

Our key idea is to introduce a case distinction on whether x and y consist of negative numbers only, of positive numbers only or contain 0. Under these extra assumptions, the computations of $(xy)^+$ and $(xy)^-$ for non-empty intervals x and y can be significantly simplified as shown in figure 1. We observe that for computing $(xy)^+$ and $(xy)^-$ only two kinds of integer products occur:

- non-negative products : ab for $a, b \geq 0$, or
- negative products : $-ab$ for $a, b < 0$.

Proposition 3 shows that in either case, the result does not only monotonically depend on the arguments a, b but even distributively (both for “ \vee and “ \wedge ”). Therefore, we now

	$x^+ \geq 0, x^- \geq 0$ $y^+ \geq 0, y^- \geq 0$	$x^+ > 0, x^- < 0$ $y^+ \geq 0, y^- \geq 0$	$x^+ < 0, x^- > 0$ $y^+ \geq 0, y^- \geq 0$
$(xy)^+$	$x^+y^+ \vee x^-y^-$	x^+y^+	x^-y^-
$(xy)^-$	$x^+y^- \vee x^-y^+$	x^+y^-	x^-y^+
	$x^+ \geq 0, x^- \geq 0$ $y^+ > 0, y^- < 0$	$x^+ > 0, x^- < 0$ $y^+ > 0, y^- < 0$	$x^+ < 0, x^- > 0$ $y^+ > 0, y^- < 0$
$(xy)^+$	x^+y^+	x^+y^+	$-x^+y^-$
$(xy)^-$	x^-y^+	$-x^-y^-$	x^-y^+
	$x^+ > 0, x^- > 0$ $y^+ < 0, y^- > 0$	$x^+ > 0, x^- < 0$ $y^+ < 0, y^- > 0$	$x^+ < 0, x^- > 0$ $y^+ < 0, y^- > 0$
$(xy)^+$	x^-y^-	$-x^-y^+$	x^-y^-
$(xy)^-$	x^+y^-	x^+y^-	$-x^+y^+$

Fig. 1. Simplification of bounds

consider systems of integer equations where we additionally allow in right-hand sides subexpressions e of the form $(e_1 \vee 0) \cdot (e_2 \vee 0)$ and $-((e_1 \wedge -1) \cdot (e_2 \wedge -1))$. We call such equations *extended*. It turns out that the least solution of a system of extended integer equations can be computed by means of max strategy iteration over the instrumented lattice \mathcal{D} along the same lines as in section 3. To handle the occurring multiplications in the lifted systems we additionally define $(a_1, j_1) \cdot (a_2, j_2) = (a_1 \cdot a_2, j_1 \vee j_2)$ for $a_1, a_2 \neq 0$ and $(0, 0) \cdot z = (0, 0)$ for $z \in \mathcal{D}$. For a system \mathcal{E} of extended integer equations, we consider the corresponding lifted system \mathcal{E}^\sharp whose least solution is approximated by greatest *feasible* solutions of *feasible* max strategies π . For a conjunctive lifted system of extended integer equations⁴, a variable assignment μ is called *feasible* iff

1. $\mu(\mathbf{x}_i) > -\infty$ for every variable \mathbf{x}_i ;
2. For every subexpression $e_1 \cdot e_2$ and $i = 1, 2$: $\llbracket e_i \rrbracket^\sharp \mu > (0, 0)$ whenever $e_i \neq 0$

Assume that \mathcal{E}^\sharp denotes a lifted system of extended integer equations with least solution μ^\sharp and that π denotes a strategy. As in section 3, we verify that there exists at most one feasible solution of $\mathcal{E}^\sharp(\pi)$ and that this feasible solution can be computed by n rounds of Round-Robin iteration on $\mathcal{E}^\sharp(\pi)$ whenever it exists. As for ordinary systems of integer equations, $\mathcal{E}^\sharp(P(\mu'))$ has a feasible solution if $-\infty < \mu'(\mathbf{x}_i) \leq \mu^\sharp(\mathbf{x}_i)$ for all variables \mathbf{x}_i . Let μ_0 denote the variable assignment obtained by n rounds of Round-Robin iteration on \mathcal{E}^\sharp . By construction, $\mu_0 \leq \mu^\sharp$ and w.l.o.g., $\mu_0(\mathbf{x}_i) > -\infty$ for all variables \mathbf{x}_i . If μ_0 does not yet equal μ^\sharp , then successive strategy improvement will construct a strictly ascending chain of variable assignments approximating the least solution of \mathcal{E}^\sharp along the same lines as in section 3. We have:

Theorem 6. *Let \mathcal{E} be a extended integer system with n variables and m occurrences of “ \vee ”. The least solution of \mathcal{E} can be computed uniformly in time $\mathcal{O}(n \cdot |\mathcal{E}| \cdot II(m))$. \square*

The exact method for extended integer systems allows us to fully deal with multiplication in systems of interval equations. Let \mathcal{E} denote a system of interval equations possibly containing arbitrary multiplications. Now we consider valuations to be functions which map the set of subexpressions S into the *four*-element lattice $D_4 = \{\perp, -, +, \top\}$ where $\perp < -, + < \top$. Thus, $\sigma(e)$ indicates whether the value e is currently only known

⁴ In a conjunctive system, the multiplications are of the form $e_1 \cdot e_2$ and $-((e_1 \wedge -1) \cdot (e_2 \wedge -1))$.

to be empty, contained in the negative or positive numbers, respectively, or contains 0. Given a variable assignment μ , we now determine the valuation $\Sigma(\mu) : S \rightarrow D_4$ by:

$$\Sigma(\mu)(e) = \begin{cases} \perp & \text{if } \llbracket e \rrbracket \mu = \emptyset \\ - & \text{if } \llbracket e \rrbracket \mu \subseteq [-\infty, -1] \\ + & \text{if } \llbracket e \rrbracket \mu \subseteq [1, \infty] \\ \top & \text{if } \llbracket e \rrbracket \mu \ni 0 \end{cases}$$

The valuation $\Sigma(\mu)(e)$ monotonically depends on μ . The maximal length of a strictly ascending chain in the complete lattice of valuations over D_4 is bounded by $2 \cdot |S| \in \mathcal{O}(|\mathcal{E}|)$. Our goal is to construct a system \mathcal{E}_σ^\pm for upper and negated lower interval bounds in presence of full multiplication, relative to a valuation σ . For that, we enrich the transformations $[\cdot]_\sigma^+$ and $[\cdot]_\sigma^-$. So far, these transformations are only defined for valuations over D_2 and expressions which are not multiplications. The corresponding rules of the new transformation for these cases are syntactically identical. Therefore, it remains to explain how subexpressions $[e_1 \cdot e_2]_\sigma^+$ and $[e_1 \cdot e_2]_\sigma^-$ should be handled. If one argument e_i of the multiplication is mapped to \perp , i.e., currently evaluates to \emptyset , we define $[e_1 \cdot e_2]_\sigma^+$ and $[e_1 \cdot e_2]_\sigma^-$ as $-\infty$. If e.g. $\sigma(e_1) = \sigma(e_2) = -$, we define

$$[e_1 \cdot e_2]_\sigma^+ = ([e_1]_\sigma^- \vee 0) \cdot ([e_2]_\sigma^- \vee 0) \quad [e_1 \cdot e_2]_\sigma^- = -(([e_1]_\sigma^+ \wedge -1) \cdot ([e_2]_\sigma^+ \wedge -1))$$

which corresponds to the case in the lower right corner of the table in figure 1. The rules for the remaining cases are constructed analogously corresponding to figure 1. The resulting system \mathcal{E}_σ^\pm is extended integer. Thus, we can compute its least solution μ_σ^\pm through the strategy iteration algorithm from section 3. Also, we find that proposition 2 also holds for systems with full multiplication. We only need now to consider valuations $\sigma : S \rightarrow D_4$. Proposition 2 implies, that algo. 3 also works for systems with full multiplication — which proves our main result for interval analysis.

Theorem 7. *Assume that \mathcal{E} is a system of interval equations with arbitrary intersections, n variables, m occurrences of “ \sqcup ” and k arbitrary multiplications. The least solution of \mathcal{E} can be computed uniformly in time $\mathcal{O}(n \cdot |\mathcal{E}|^2 \cdot \Pi(2m + 6k))$. \square*

The complexity estimation is based on the corresponding estimation for extended integer equations. Additionally, we must take into account the number of updates to valuations. Note also that the number of occurrences of “ \vee ”-operators in the generated extended integer systems are now bounded only by $2m + 6k$.

5 Conclusion

We considered systems of integer equations. These are necessary for precisely solving equations over the interval domain. We used an instrumentation of the lattice \mathcal{Z} with one extra component to guarantee for conjunctive systems to admit at most one feasible solution. This uniqueness allowed us to construct a strategy iteration algorithm for computing least solutions of systems of integer equations. We extended this result to construct an algorithm for precisely solving systems of interval equations — even for systems using arbitrary multiplication. In the latter case we had to

take into account that multiplication of integers is not monotonic. The resulting algorithms are amazingly simple and natural. Implementations can be down-loaded from <http://www2.in.tum.de/~gawlitza/policy>. First experiments show that the efficiency is promising. It remains for future work to systematically evaluate the solvers for systems of integer and interval equations on real-world examples.

References

1. H. Bjorklund, S. Sandberg, and S. Vorobyov. Complexity of Model Checking by Iterative Improvement: the Pseudo-Boolean Framework . In *Proc. 5th Int. Andrei Ershov Memorial Conf. Perspectives of System Informatics*, pages 381–394. LNCS 2890, Springer, 2003.
2. J. Cochet-Terrasson, S. Gaubert, and J. Gunawardena. A Constructive Fixed Point Theorem for Min-Max Functions. *Dynamics and Stability of Systems*, 14(4):407–433, 1999.
3. A. Costan, S. Gaubert, E. Goubault, M. Martel, and S. Putot. A Policy Iteration Algorithm for Computing Fixed Points in Static Analysis of Programs. In *Computer Aided Verification, 17th Int. Conf. (CAV)*, pages 462–475. LNCS 3576, Springer Verlag, 2005.
4. 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.
5. 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.
6. T. Gawlitza, J. Reineke, H. Seidl, and R. Wilhelm. Polynomial Exact Interval Analysis Revisited. Technical report, TU München, 2006.
7. A. Hoffman and R. Karp. On Nonterminating Stochastic Games. *Management Sci.*, 12:359–370, 1966.
8. R. Howard. *Dynamic Programming and Markov Processes*. Wiley, New York, 1960.
9. D. E. Knuth. A Generalization of Dijkstra's algorithm. *Information Processing Letters (IPL)*, 6(1):1–5, 1977.
10. N. Megiddo. On the Complexity of Linear Programming. In T. Bewley, editor, *Advances in Economic Theory: 5th World Congress*, pages 225–268. Cambridge University Press, 1987.
11. 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.
12. 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.
13. A. Puri. *Theory of Hybrid and Discrete Systems*. PhD thesis, University of California, Berkeley, 1995.
14. M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York, 1994.
15. H. Seidl. Least and Greatest Solutions of Equations over \mathcal{N} . *Nordic Journal of Computing (NJC)*, 3(1):41–62, 1996.
16. 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.
17. J. Vöge and M. Jurdzinski. A Discrete Strategy Improvement Algorithm for Solving Parity Games. In *Computer Aided Verification, 12th Int. Conf. (CAV)*, pages 202–215. LNCS 1855, Springer, 2000.
18. U. Zwick and M. Paterson. The Complexity of Mean Payoff Games on Graphs. *Theoretical Computer Science (TCS)*, 158(1&2):343–359, 1996.