Abstract Interpretation over Zones without Widening

Thomas Martin Gawlitza
The University of Sydney, Australia
gawlitza@it.usyd.edu.au and Helmut Seidl
Technische Universität München, Germany
seidl@in.tum.de

Abstract

We present a practical algorithm for computing least solutions of systems of (fixpoint-)equations over the integers with, besides other monotone operators, addition, multiplication by positive constants, maximum, and minimum. The algorithm is based on max-strategy iteration. Its worst-case running-time (w.r.t. a uniform cost measure) is independent of the sizes of occurring numbers. We apply this algorithm to compute the abstract semantics of programs over integer intervals as well as over integer zones.

1 Introduction

Tight bounds on the possible values of integer variables are crucial when memory errors at array indexing are to be excluded. Just to name one further prominent application, such bounds are also crucial for inferring worst-case execution times [29]. Various extensions of interval analysis have been considered which also infer nontrivial relationships between integer variables. Beyond upper and lower bounds on the values of variables, the abstract domain of *zones* allows to express bounds on the *differences* between the values of two variables. This domain has been introduced by Dor et al. [11] for the analysis of string manipulating C functions and has later been refined by Miné [22]. Bounds on variable differences, however, have earlier been widely used by the model-checking community [21, 31]. A dedicated representation for zones, called *difference bound matrices*, as well as various operators on these for model-checking timed automata were introduced by Larsen et al. [21], Yovine [31]. Later, this domain has been generalized to *octagons* which additionally track bounds on the *sums* of the values of two variables [23]. All of these domains are special instances of *template polyhedra* as studied by Sankaranarayanan et al. [26].

The approach of inferring invariants by means of abstract interpretation first translates the program into a set of constraints over an abstract lattice of possible invariants. The least solution of this constraint system represents the abstract semantics of the program, i.e., the most precise invariants which the analysis can infer. The most immediate idea for determining this least solution is to use some variant of Kleene fixpoint iteration which, starting from the least value of the abstract domain, repeatedly re-evaluates constraints until no further contribution to the values of unknowns is found. Already interval analysis, however, relies on lattices with *infinite* strictly ascending chains. For such lattices, a Kleene fixpoint iteration may not terminate and thus fail to determine the least solution. In order to enforce termination, Cousot and Cousot propose a *widening* iteration which returns *some*, perhaps very imprecise solution, which subsequently is improved by a *narrowing* iteration [10]. For the specific case of zones, widening and narrowing operators have been introduced by Miné [22]. Since widening trades termination of the fixpoint iteration against precision, widening/narrowing-based techniques may not succeed in computing the least solution.

This was more or less state of the art until Costan et al. [7] proposed an alternative technique for computing solutions to constraint systems over integer intervals. They considered the

problem of computing least solutions to constraint systems of integer intervals as a two-players zero-sum game where one player (the maximizer) aims at maximizing and his opponent (the minimizer) aims at minimizing the values of the variables. Then they apply (min-)strategy iteration to find a small solution to the constraints, i.e., they try to improve the strategy for the minimizer step by step. While this approach still may not result in the least solution, it at least avoids the crude overapproximation through widening.

Strategy iteration in itself is not new. Strategy iteration was introduced by Howard [18] for solving one-player games. Hoffman and Karp [17] used the idea for solving certain *two-player games*. A strategy improvement algorithm performs two basic steps that are to be alternated: Each strategy has to be *evaluated* and based on the result of this evaluation it has to be *improved*, if possible. These steps are repeated until stabilization, i.e., until no further improvement is possible. In the context of program analysis through abstract interpretation, Gaubert et al. [13] generalized the idea of Costan et al. [7] and applied it to *zones*, *octagons* or, more generally, arbitrary *template polyhedra*. In contrast to the interval domain, these domains are *relational*, i.e., they are able to express relations between the values of different variables. Their method applies linear programming to perform a strategy improvement step, i.e., to evaluate and improve a given strategy. It returns quite precise solutions which, however, are not guaranteed to be minimal.

In this article, we elaborate the techniques for analyzing values of integer variables which we have developed in [15, 16]. In these articles, we have proposed a max-strategy iteration which is guaranteed to terminate with the least solution. Our approach differs from the approach proposed by Costan et al. [7], Gaubert et al. [13] in that we improve the strategy for the maximizer instead of the strategy for the minimizer. Each strategy for the maximizer is evaluated using a generalization of the Bellman-Ford algorithm. Our approach finally allows to compute the abstract semantics of programs over integer intervals. We later extended this approach to computing the abstract semantics over zones, octagons or template polyhedra [14]. The latter methods, however, use exact rational arithmetic and apply linear programming to evaluate the strategies for the maximizer.

Although fast implementations of linear programming are available, no *strongly* polynomial algorithm is known. Therefore, it is not clear how well algorithms that rely on linear programming may scale to larger inputs — in particular, when exact rational arithmetic is applied. In this article, we show that if we are only interested in integer zones, then full-fletched linear programming is not necessary to evaluate a strategy for the maximizer. Instead, an algorithm can be constructed which is based on the Bellman-Ford algorithm extended with subroutine calls to solve minimum cost flow problems [25]. For the latter problem various strongly polynomial algorithms have been devised (see e.g. Ahuja et al. [1] for a recent overview).

This article is organized as follows. In Section 2, we introduce *systems of integer equations* and discuss elementary properties. An adaption of the Bellman-Ford algorithm that can be used to evaluate strategies for the maximizer is presented in Section 4. Our strategy improvement algorithm is presented in Section 5. We slightly extend the applicability of our approach in Section 6, where we introduce systems of *extended* integer equations. In Section 7 we finally apply our techniques for computing the abstract semantics of programs over intervals and zones. We conclude with section 8.

2 Preliminaries

2.1 The max-strategy improvement approach by example

Before presenting the technical machinery, we introduce the key ideas of our max-strategy improvement algorithm by an example. The goal is to compute the least solution of a system of integer equations. Such a system consists of equations of the form $\mathbf{x} = e$, where e is an expression over $\overline{\mathbb{Z}} = \mathbb{Z} \cup \{-\infty, \infty\}$ which uses, beside other monotone operators, addition, multiplication by positive constants, minimum, and maximum. As an example, consider the system:

$$\begin{aligned} \mathbf{x}_1 &= 0 \vee (-1 + \mathbf{x}_1 \wedge \mathbf{x}_2) \\ \mathbf{x}_2 &= 0 \vee 5 + \mathbf{x}_1 \vee \mathbf{x}_1 \\ \mathbf{x}_3 &= 0 \vee 1 + \mathbf{x}_3 \vee 0 + \mathbf{x}_1 \end{aligned} \tag{1}$$

Here, $\wedge$ denotes the minimum and $\vee$ the maximum operator. The least solution of equation system (1) is given by $\mathbf{x}_1 = 0$, $\mathbf{x}_2 = 5$, and $\mathbf{x}_3 = \infty$.

Our max-strategy improvement algorithm considers the computation of the least solution of a fixpoint-equation system with monotone right-hand sides as a competition between a *maximizer* and a *minimizer*. The maximizer aims at maximizing the values of the variables, whereas the minimizer aims at minimizing it. A state of a play is a variable assignment and the play starts at the state $\mathbf{x}_1 = \mathbf{x}_2 = \mathbf{x}_3 = -\infty$. The maximizer is allowed to control the game at $\vee$-operators, whereas the minimizer is allowed to control the game at $\wedge$-operators. We focus on the maximizer and assume that the minimizer locally always choses the best strategy.

The strategy for the maximizer is improved successively. In the above example, the maximizer may choose to start the fixpoint iteration with a strategy that corresponds to the equation system

$$\mathbf{x}_1 = 0 \qquad \mathbf{x}_2 = 0 \qquad \mathbf{x}_3 = 0 \tag{2}$$

That is, the maximizer selects the left-most argument for every $\vee$-expression. Conceptually, a fixpoint iteration is now perfromed according to the strategy the maximizer has chosen. In consequence, the next states of the play are determined by evaluating the right-hand sides iteratively. The evaluation of the right-hand sides corresponds to a minimizer who always choses the locally best alternative. In this example the play reaches the state $\mathbf{x}_1 = \mathbf{x}_2 = \mathbf{x}_3 = 0$. In this state, the maximizer may choose to change his strategy to a strategy that corresponds to the equation system

$$\mathbf{x}_1 = 0 \qquad \mathbf{x}_2 = 5 + \mathbf{x}_1 \qquad \mathbf{x}_3 = 1 + \mathbf{x}_3 \tag{3}$$

Starting from the state $\mathbf{x}_1 = \mathbf{x}_2 = \mathbf{x}_3 = 0$, the fixpoint iteration now converges to the least solution of the original equation system, which is $\mathbf{x}_1 = 0$, $\mathbf{x}_2 = 5$, and $\mathbf{x}_3 = \infty$. In other words, $\mathbf{x}_1 = 0$, $\mathbf{x}_2 = 5$, and $\mathbf{x}_3 = \infty$ is the least solution that is greater than or equal to $\mathbf{x}_1 = \mathbf{x}_2 = \mathbf{x}_3 = 0$.

Observe that $\mathbf{x}_1 = 0$, $\mathbf{x}_2 = 5$, and $\mathbf{x}_3 = \infty$ is also the greatest solution of the equation system (3). This is not by accident. We will see that this is always the case — provided that we follow some rules when switching from one strategy to another. This greatest solution can be computed through the adaption of the Bellman-Ford algorithm that we present in Section 4.

2.2 Notations

As usual, $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{R}$ denote the set of natural numbers, the set of integers, and the set of real numbers, respectively. We assume $0 \in \mathbb{N}$ and we denote $\mathbb{N} \setminus \{0\}$ by $\mathbb{N}_{>0}$. For $\mathbb{D} \in \{\mathbb{Z}, \mathbb{R}\}$, we set $\overline{\mathbb{D}} := \mathbb{D} \cup \{-\infty, \infty\}$. $\overline{\mathbb{Z}}$ and $\overline{\mathbb{R}}$ are complete linearly ordered sets. For two functions $f : X \to Y$ and $g : X' \to Y$, the function $f \oplus g : (X \cup X') \to Y$ is defined by

$$(f \oplus g)(x) = \begin{cases} g(x) & \text{if } x \in X' \\ f(x) & \text{if } x \in X \setminus X' \end{cases} \qquad \text{for all } x \in X \cup X'. \tag{4}$$

We use a uniform cost measure where we count arithmetic operations and memory accesses for $\mathcal{O}(1)$. The size of a data structure S in the uniform cost measure will be denoted by $\|S\|$. An algorithm is called *uniform* iff its running time w.r.t. the uniform cost measure only depends on the size of the input in the uniform cost measure, i.e., the running time does not depend on the sizes of the occurring numbers.

2.3 Monotone self-maps on complete lattices

Let $\mathbb{D}$ be a partially ordered set (partially ordered by $\leq$). For $x \in \mathbb{D}$, we set $\mathbb{D}_{\geq x} := \{y \in \mathbb{D} \mid y \geq x\}$ and $\mathbb{D}_{\leq x} := \{y \in \mathbb{D} \mid y \leq x\}$. As usual, for $x, y \in \mathbb{D}$, we write $x < y$ iff $x \leq y$ and $x \neq y$. We denote the *least upper bound* and the *greatest lower bound* of a set X by $\bigvee X$ and $\bigwedge X$, respectively. $\mathbb{D}$ is called a *complete lattice* iff $\bigvee X$ and $\bigwedge X$ exist for all $X \subseteq \mathbb{D}$. The least element $\bigvee \emptyset$ (resp. the greatest element $\bigwedge \emptyset$) is denoted by $\bot$ (resp. $\top$). We define the binary operators $\vee$ and $\wedge$ by

$$x \vee y := \bigvee \{x, y\}, \text{and} \qquad\qquad x \wedge y := \bigwedge \{x, y\} \tag{5}$$

for all $x, y \in \mathbb{D}$, respectively. For $\square \in \{\vee, \wedge\}$, we will also consider $x_1 \square \cdots \square x_k$ as the application of a k-ary operator. This will cause no problems, since the binary operators $\vee$ and $\wedge$ are associative. In order to simplify notations, we assume that $\wedge$ binds tighter than $\vee$, i.e., $x \wedge y \vee z = (x \wedge y) \vee z$ holds for all $x, y, z \in \mathbb{D}$.

Let $f : \mathbb{D} \to \mathbb{D}$ be a self-map. An element $x \in \mathbb{D}$ is called a fixpoint (resp. pre-fixpoint, resp. post-fixpoint) iff $x = f(x)$ (resp. $x \leq f(x)$, resp. $x \geq f(x)$). The set of all fixpoints (resp. pre-fixpoints, resp. post-fixpoints) of f is denoted by $\mathbf{Fix}(f)$ (resp. $\mathbf{PreFix}(f)$, resp. $\mathbf{PostFix}(f)$). The least fixpoint (resp. the greatest fixpoint) of f is denoted by μf (resp. νf), provided that it exists. For $x \in \mathbb{D}$, $\mu_{\geq x} f$ (resp. $\nu_{\leq x} f$) denotes the least (resp. greatest) element from $\mathbb{D}_{\geq x} \cap \mathbf{Fix}(f)$ (resp. $\mathbb{D}_{\leq x} \cap \mathbf{Fix}(f)$), provided that it exists.

A map f is called monotone iff $x \leq y$ implies $f(x) \leq f(y)$ for all x, y. The existence of least and greatest fixpoints of monotone self-maps on complete lattices is ensured by the Knaster-Tarski fixpoint theorem [28]: Every monotone self-map f on a complete lattice has a least fixpoint μf and a greatest fixpoint νf. Furthermore, $\mu f = \bigwedge \mathbf{PostFix}(f)$, and $\nu f = \bigvee \mathbf{PreFix}(f)$.

Let f be a monotone self-map on a complete lattice $\mathbb{D}$. Let $x \in \mathbf{PreFix}(f)$. Then $\mathbb{D}_{\geq x}$ is also a complete lattice and the restriction of f to $\mathbb{D}_{\geq x}$ is a self-map on $\mathbb{D}_{\geq x}$. Thus, the Knaster-Tarski fixpoint theorem implies that $\mu_{\geq x} f$ exists. Dually, if $x \in \mathbf{PostFix}(f)$, then $\nu_{\leq x} f$ exists.

Let $\mathbf{X}$ be a set of variables. The set $\mathbf{X} \to \mathbb{D}$ of all *variable assignments* is partially ordered by the point-wise extension of $\leq$ which we, for simplicity, again denote by $\leq$. If $\mathbb{D}$ is a complete lattice, then $\mathbf{X} \to \mathbb{D}$ is also a complete lattice. For $d \in \mathbb{D}$, $\underline{d}$ denotes the variable assignment $\{\mathbf{x} \mapsto d \mid \mathbf{x} \in \mathbf{X}\}$.

2.4 Systems of monotone equations

Assume that a fixed set $\mathbf{X}$ of variables and a partially ordered set $\mathbb{D}$ (for instance $\overline{\mathbb{Z}}$) are given. We consider (fixpoint-)equations of the form $\mathbf{x} = e$, where $\mathbf{x} \in \mathbf{X}$ is a variable that takes values in $\mathbb{D}$ and e is an expression over $\mathbb{D}$. A *system* $\mathcal{E}$ of (fixpoint-)equations is a finite set $\{\mathbf{x}_1 = e_1, \ldots, \mathbf{x}_n = e_n\}$ of equations, where $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are pairwise distinct variables. We denote the set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of variables occurring in $\mathcal{E}$ by $\mathbf{X}_\mathcal{E}$. We drop the subscript, whenever it is clear from the context. The set of subexpressions occurring in the right-hand sides of $\mathcal{E}$ is denoted by $\mathcal{S}(\mathcal{E})$. The set of variables occurring in the right-hand sides of $\mathcal{E}$ is denoted by $\mathbf{Vars}(\mathcal{E})$. For $\mathbf{X}' \subseteq \mathbf{X}$, we set $\mathcal{E} \oplus \{\mathbf{x} = e_\mathbf{x} \mid \mathbf{x} \in \mathbf{X}'\} := \{\mathbf{x} = e \in \mathcal{E} \mid \mathbf{x} \notin \mathbf{X}'\} \cup \{\mathbf{x} = e_\mathbf{x} \mid \mathbf{x} \in \mathbf{X}'\}$.

For an expression e, we write $e[e_\mathbf{x}/\mathbf{x}]_{\mathbf{x} \in \mathbf{X}'}$ for the expression which is obtained from e by simultaneously substituting all occurrences of the variable $\mathbf{x}$ with the expression $e_\mathbf{x}$ for all variables $\mathbf{x} \in \mathbf{X}'$. We set $\mathcal{E}[e_\mathbf{x}/\mathbf{x}]_{\mathbf{x} \in \mathbf{X}'} := \{\mathbf{x} = e[e_\mathbf{x}/\mathbf{x}]_{\mathbf{x} \in \mathbf{X}'} \mid \mathbf{x} = e \in \mathcal{E}\}$.

For a variable assignment $\rho : \mathbf{X} \to \mathbb{D}$, e is mapped to a value $[\![e]\!]\rho$ by

$$[\![\mathbf{x}]\!]\rho := \rho(\mathbf{x}), \quad \text{and} \qquad [\![f(e_1, \ldots, e_k)]\!]\rho := f([\![e_1]\!]\rho, \ldots, [\![e_k]\!]\rho), \tag{6}$$

where $\mathbf{x} \in \mathbf{X}$, f is a k-ary operator, for instance $+$, and $e_1, \ldots, e_k$ are expressions. Let $\mathcal{E}$ be a system of (fixpoint-)equations. We define the unary operator $[\![\mathcal{E}]\!]$ on $\mathbf{X} \to \mathbb{D}$ by setting

$$([\![\mathcal{E}]\!]\rho)(\mathbf{x}) := [\![e]\!]\rho \qquad \text{for all equations } \mathbf{x} = e \text{ of } \mathcal{E}. \tag{7}$$

A solution of $\mathcal{E}$ is a variable assignment ρ such that $\rho = [\![\mathcal{E}]\!]\rho$. The set of solutions is denoted by $\mathbf{Sol}(\mathcal{E})$. A pre-solution (resp. post-solution) of $\mathcal{E}$ is a variable assignment ρ such that $\rho \leq [\![\mathcal{E}]\!]\rho$ (resp. $\rho \geq [\![\mathcal{E}]\!]\rho$). The set of pre-solutions (resp. the set of post-solutions) is denoted by $\mathbf{PreSol}(\mathcal{E})$ (resp. $\mathbf{PostSol}(\mathcal{E})$). The least solution (resp. the greatest solution) of a system $\mathcal{E}$ of equations is denoted by $\mu[\![\mathcal{E}]\!]$ (resp. $\nu[\![\mathcal{E}]\!]$), provided that it exists. For a pre-solution ρ (resp. for a post-solution ρ), $\mu_{\geq\rho}[\![\mathcal{E}]\!]$ (resp. $\nu_{\leq\rho}[\![\mathcal{E}]\!]$) denotes the least solution that is greater than or equal to ρ (resp. the greatest solution that is less than or equal to ρ).

An expression e (resp. an equation $\mathbf{x} = e$) is called *monotone* iff all operators occurring in e are monotone. An expression e or an equation $\mathbf{x} = e$ is called *disjunctive* (resp. *conjunctive*) iff e does not contain $\wedge$-operators (resp. $\vee$-operators). An expression e or an equation $\mathbf{x} = e$ is called *basic* iff e does neither contain $\wedge$- nor $\vee$-operators.

In our setting, the Knaster-Tarski fixpoint theorem can be stated as follows: Every system $\mathcal{E}$ of monotone equations over a complete lattice has a least solution $\mu[\![\mathcal{E}]\!]$ and a greatest solution $\nu[\![\mathcal{E}]\!]$. Furthermore, $\mu[\![\mathcal{E}]\!] = \bigwedge \mathbf{PostSol}(\mathcal{E})$, and $\nu[\![\mathcal{E}]\!] = \bigvee \mathbf{PreSol}(\mathcal{E})$.

3 Systems of Integer Equations

In this section we introduce the class of fixpoint equation systems we will focus on in the remainder of this article. We moreover study elementary properties of these equation systems and the operators that are allowed in right-hand sides.

3.1 Operators on $\overline{\mathbb{Z}}$

On $\overline{\mathbb{Z}}$ we consider two additions, $+^{-\infty}$ and $+^{\infty}$, which are dual to each other. Both coincide on $\mathbb{Z}$ with the usual addition. The operator $+^{-\infty}$ preserves $-\infty$, whereas the operator $+^{\infty}$ preserves ∞:

$$x +^{-\infty} y := \begin{cases} -\infty & \text{if } -\infty \in \{x, y\} \\ \infty & \text{if } -\infty \notin \{x, y\} \text{ and } \infty \in \{x, y\} \\ x + y & \text{if } x, y \in \mathbb{Z} \end{cases} \qquad \text{for all } x, y \in \overline{\mathbb{Z}} \tag{8}$$

$$x+^{\infty}y := \begin{cases} \infty & \text{if } \infty \in \{x,y\} \\ -\infty & \text{if } \infty \notin \{x,y\} \text{ and } -\infty \in \{x,y\} \\ x+y & \text{if } x,y \in \mathbb{Z} \end{cases} \qquad \text{for all } x,y \in \overline{\mathbb{Z}}. \tag{9}$$

We have $x+^{-\infty}y = x+^{\infty}y$, whenever $(x,y) \notin \{(-\infty,\infty),(\infty,-\infty)\}$. In particular, both operators $+^{-\infty}$ and $+^{\infty}$ behave equal, if one argument is from $\mathbb{Z}$. In our applications, we mostly use the operator $+^{-\infty}$. Therefore, in the following we also denote the operator $+^{-\infty}$ simply by $+$. We extend the multiplication on $\mathbb{Z}$ as usual, i.e.,

$$x \cdot \infty = \infty \cdot x = \infty, \quad x \cdot -\infty = -\infty \cdot x = -\infty \qquad \text{for all } x > 0 \tag{10}$$

$$x \cdot \infty = \infty \cdot x = -\infty, \quad x \cdot -\infty = -\infty \cdot x = \infty \qquad \text{for all } x < 0 \tag{11}$$

$$0 \cdot \infty = \infty \cdot 0 = 0 \cdot -\infty = -\infty \cdot 0 = 0. \tag{12}$$

Observe that the multiplication $\cdot$ on $\overline{\mathbb{Z}}$ is not monotone, since for instance $(-1,0) \leq (-1,1)$ and $-1 \cdot 0 = 0 \not\leq -1 = -1 \cdot 1$. For $c \in \overline{\mathbb{Z}}$, we denote the unary operator that assigns $c+x$ (resp. $c \cdot x$) to each $x \in \overline{\mathbb{Z}}$ by $c+$ (resp. $c\cdot$). The operator $c\cdot$ is monotone, whenever c is positive. For simplicity, we make the following agreements: The operator $c\cdot$ binds tighter than the operators $+^{-\infty}$ and $+^{\infty}$. These operators bind tighter than $\wedge$, which binds tighter than $\vee$. All operators are left-associative.

3.2 Expansivity

In this article, we are in particular interested in expressions e for which the evaluation function $[\![e]\!]$ is *expansive* (see below) in all variables occurring in e. Informally, this means that the value of an expression e increases (resp. decreases) by at least δ, whenever the value of a variable increases (resp. decreases) by δ.

Let X be a set and $f : (X \to \overline{\mathbb{Z}}) \to \overline{\mathbb{Z}}$ be a mapping. The mapping f is called *upward-expansive in* $X' \subseteq X$ iff

$$f(\rho \oplus \{x \mapsto \rho(x) + \delta\}) \geq f(\rho) + \delta \qquad \text{for all } x \in X',\ \rho : X \to \overline{\mathbb{Z}},\ \delta \in \mathbb{N}. \tag{13}$$

Dually, f is called *downward-expansive in* $X' \subseteq X$ iff

$$f(\rho \oplus \{x \mapsto \rho(x) - \delta\}) \leq f(\rho) - \delta \qquad \text{for all } x \in X',\ \rho : X \to \overline{\mathbb{Z}},\ \delta \in \mathbb{N}. \tag{14}$$

The mapping f is called *expansive in* $X' \subseteq X$ iff it is upward- and downward-expansive in $X' \subseteq X$. It is simply called *upward-expansive* (resp. *downward-expansive*, resp. *expansive*) iff it is upward-expansive in X (resp. downward-expansive in X, resp. expansive in X). Since $\overline{\mathbb{Z}}^n$ can be identified with the set $\overline{\mathbb{Z}}^{\{1,\ldots,n\}} = \{1,\ldots,n\} \to \overline{\mathbb{Z}}$ of all functions from $\{1,\ldots,n\}$ to $\overline{\mathbb{Z}}$, the above definitions also apply to operators.

Lemma 1. *If all operators that occur in an expression e are expansive (respectively upward-expansive, respectively downward-expansive), then the evaluation function $[\![e]\!]$ of e is expansive (respectively upward-expansive, respectively downward-expansive) in* $\mathbf{Vars}(e)$. □

In the remainder of this article we are in particular interested in operators that are monotone and expansive. The operators $+^{-\infty}$,$+^{\infty}$, $c+$ ($c \in \mathbb{Z}$), and $c\cdot$ ($c \in \mathbb{N}_{>0}$), are important examples for operators that are monotone and expansive.

3.3 Systems of integer equations

Let $\mathbf{X}$ be a set of variables and $\mathcal{F}$ be a set of operators. We denote the set of all expressions that can be built up using variables from $\mathbf{X}$ and operators from $\mathcal{F}$ by $\mathbf{E}(\mathcal{F}, \mathbf{X})$. Moreover, we identify constants with nullary operators. For instance, we have $\mathbf{x} + 2 \in \mathbf{E}(\overline{\mathbb{Z}} \cup \{+\}, \{\mathbf{x}\})$. An expression from $\mathbf{E}(\mathcal{F}, \mathbf{X})$ is called a $\mathbf{E}(\mathcal{F}, \mathbf{X})$-expression. An equation $\mathbf{x} = e$ is called a $\mathbf{E}(\mathcal{F}, \mathbf{X})$-equation iff $\mathbf{x} \in \mathbf{X}$ and $e \in \mathbf{E}(\mathcal{F}, \mathbf{X})$. We define the sets $\mathcal{F}$, $\mathcal{F}^l$ and $\mathcal{F}^s$ of operators as follows:

$$\mathcal{F} := \{\wedge, \vee\} \cup \{f : \overline{\mathbb{Z}}^k \to \overline{\mathbb{Z}} \mid f \text{ is monotone and expansive}\} \tag{15}$$

$$\mathcal{F}^l := \{\wedge, \vee, +^{-\infty}, +^{\infty}\} \cup \{c\cdot \mid c \in \mathbb{N}_{>0}\} \tag{16}$$

$$\mathcal{F}^s := \{\wedge, \vee\} \cup \{c+ \mid c \in \mathbb{Z}\} \tag{17}$$

Let $\mathbf{X}$ be a set of variables. An $\mathbf{E}(\overline{\mathbb{Z}} \cup \mathcal{F}, \mathbf{X})$-expression is called *integer expression*, an $\mathbf{E}(\overline{\mathbb{Z}} \cup \mathcal{F}, \mathbf{X})$-equation is called *integer equation*, and an $\mathbf{E}(\overline{\mathbb{Z}} \cup \mathcal{F}^l, \mathbf{X})$-expression is called *linear integer equation*. Accordingly, an $\mathbf{E}(\overline{\mathbb{Z}} \cup \mathcal{F}^l, \mathbf{X})$-equation is called *linear integer equation*, an $\mathbf{E}(\overline{\mathbb{Z}} \cup \mathcal{F}^s, \mathbf{X})$-expression is called *simple integer expression*, and an $\mathbf{E}(\overline{\mathbb{Z}} \cup \mathcal{F}^s, \mathbf{X})$-equation is called *simple integer equation*.

Every system of *simple* integer equations can be identified with a system of *linear* integer equations. The Knaster-Tarski fixpoint theorem implies that every system of integer equations has a least and a greatest solution.

Example 1. *The system* $\mathcal{E} = \{\mathbf{x}_1 = (\mathbf{x}_2 \vee \mathbf{x}_1 + 1) \wedge 100,\ \mathbf{x}_2 = 0\}$ *is a system of simple integer equations. The least solution is* $\mu[\![\mathcal{E}]\!] = \{\mathbf{x}_1 \mapsto 100,\ \mathbf{x}_2 \mapsto 0\}$. □

As the following example shows, Kleene fixpoint iteration is not an effective method for computing least solutions of systems of integer equations.

Example 2 (Kleene Fixpoint Iteration)**.** *The least solution of the system* $\mathcal{E} = \{\mathbf{x} = 1 \vee \mathbf{x} + 1\}$ *of simple integer equations is* $\mu[\![\mathcal{E}]\!] = \{\mathbf{x} \mapsto \infty\}$. *For* $i \geq 1$*, the* i*-th Kleene approximate is* $[\![\mathcal{E}]\!]^i(\underline{-\infty}) = \{\mathbf{x} \mapsto i\} < \{\mathbf{x} \mapsto \infty\}$. *Thus, there does not exists an* $i \in \mathbb{N}$ *such that* $[\![\mathcal{E}]\!]^i(\underline{-\infty}) = \mu[\![\mathcal{E}]\!]$. □

For every system $\mathcal{E}$ of *linear* integer equations, we can compute bounds for the finite values of the least solution. That is, we can (even in polynomial time) compute some $B(\mathcal{E}) \in \mathbb{N}$ such that $|\mu[\![\mathcal{E}]\!](\mathbf{x})| \leq B(\mathcal{E})$ for every variable $\mathbf{x} \in \mathbf{X}$ with $\mu[\![\mathcal{E}]\!](\mathbf{x}) \in \mathbb{Z}$. We can then compute the least solution using a modified Kleene fixpoint iteration that sets the value of a variable to ∞ as soon as it is known to be greater then $B(\mathcal{E})$. However, the number of iterations then depends on the sizes of the constants that occur in the equation system, i.e., this method is *not* uniform.

3.4 Duality

Since $-(-x) = x$, $-(x \vee y) = -x \wedge -y$ and $-(x \wedge y) = -x \vee -y$ for all $x, y \in \overline{\mathbb{Z}}$, the unary minus $- : \overline{\mathbb{Z}} \to \overline{\mathbb{Z}}$ is a negation. The *dual* f^δ of a k-ary operator f on $\overline{\mathbb{Z}}$ is defined by $f^\delta(x_1, \ldots, x_k) := -f(-x_1, \ldots, -x_k)$ for all $x_1, \ldots, x_k \in \overline{\mathbb{Z}}$. The operators $\vee$ and $\wedge$ are dual to each other, $+^{-\infty}$ and $+^{\infty}$ are dual to each other, and $c\cdot$ ($c \in \mathbb{N}_{>0}$) is self-dual, i.e., $c\cdot$ is the dual of $c\cdot$. The dual e^δ of an integer expression e is inductively defined by $\mathbf{x}^\delta := \mathbf{x}$, and $f(e_1, \ldots, e_k) := f^\delta(e_1^\delta, \ldots, e_k^\delta)$, where $\mathbf{x}$ is a variable, f is a k-ary operator, and $e_1, \ldots, e_k$ are integer expressions. Finally, the dual $\mathcal{E}^\delta$ of a system $\mathcal{E}$ of integer equations is defined by $\mathcal{E}^\delta := \{\mathbf{x} = e^\delta \mid \mathbf{x} = e \in \mathcal{E}\}$.

The class of systems of integer equations (resp. linear integer equations, resp. simple integer equations) is closed under dualization. Moreover, we have $\nu[\![\mathcal{E}]\!] = -\mu[\![\mathcal{E}^\delta]\!]$ for every system $\mathcal{E}$ of integer equations.

Example 3. *The dual $\mathcal{E}^\delta$ of the system $\mathcal{E} = \{\mathbf{x}_1 = (2 \cdot \mathbf{x}_1 \wedge 100) \vee 1\}$ of linear integer equations is $\mathcal{E}^\delta = \{\mathbf{x}_1 = (2 \cdot \mathbf{x}_1 \vee -100) \wedge -1\}$. Indeed, we have*

$$\mu[\![\mathcal{E}]\!] = \{\mathbf{x}_1 \mapsto 100\} = -\{\mathbf{x}_1 \mapsto -100\} = -\nu[\![\mathcal{E}^\delta]\!]. \qquad \square \tag{18}$$

3.5 Strategies

A *$\vee$-strategy* σ (resp. *$\wedge$-strategy* π) for a system $\mathcal{E}$ of equations is a function that maps every expression $e_1 \vee \cdots \vee e_k$ (resp. $e_1 \wedge \cdots \wedge e_k$) occurring in $\mathcal{E}$ to one of the immediate subexpressions e_j, $j \in \{1, \ldots, k\}$. We denote the set of all $\vee$-strategies (resp. $\wedge$-strategies) for $\mathcal{E}$ by $\Sigma_\mathcal{E}$ (resp. $\Pi_\mathcal{E}$). We drop subscripts, whenever they are clear from the context. For all $\vee$-strategies $\sigma \in \Sigma$, the expression $e\sigma$ is inductively defined by

$$(e_1 \vee \cdots \vee e_k)\sigma := (\sigma(e_1 \vee \cdots \vee e_k))\sigma, \tag{19}$$

$$(f(e_1, \ldots, e_k))\sigma := f(e_1\sigma, \ldots, e_k\sigma), \tag{20}$$

where $f \neq \vee$ is some operator. Finally, we set

$$\mathcal{E}(\sigma) := \{\mathbf{x} = e\sigma \mid \mathbf{x} = e \in \mathcal{E}\}. \tag{21}$$

The definitions of $e\pi$ and $\mathcal{E}(\pi)$ for a $\wedge$-strategy π are dual.

There exists a $\wedge$-strategy π for every system $\mathcal{E}$ of integer equations such that $\mu[\![\mathcal{E}(\pi)]\!] = \mu[\![\mathcal{E}]\!]$. However, an analogous statement does not hold for $\vee$-strategies, i.e., there does not always exist a $\vee$-strategy σ for $\mathcal{E}$ such that $\mu[\![\mathcal{E}(\sigma)]\!] = \mu[\![\mathcal{E}]\!]$ — even if we restrict ourselves to systems of *simple* integer equations.

Example 4. *We consider the system $\mathcal{E} = \{\mathbf{x}_1 = \mathbf{x}_1 + 1 \vee 0\}$ of simple integer equations. Let $\sigma_1 := \{\mathbf{x}_1 + 1 \vee 0 \mapsto \mathbf{x}_1 + 1\}$ and $\sigma_2 := \{\mathbf{x}_1 + 1 \vee 0 \mapsto 0\}$ be the $\vee$-strategies for $\mathcal{E}$, i.e., we have $\Sigma = \{\sigma_1, \sigma_2\}$. We have*

$$\mu[\![\mathcal{E}(\sigma_1)]\!] = \{\mathbf{x}_1 \mapsto -\infty\} \neq \mu[\![\mathcal{E}]\!] = \{\mathbf{x}_1 \mapsto \infty\} \neq \{\mathbf{x}_1 \mapsto 0\} = \mu[\![\mathcal{E}(\sigma_2)]\!]. \tag{22}$$

Thus, there does not exist a $\vee$-strategy σ for $\mathcal{E}$ such that $\mu[\![\mathcal{E}(\sigma)]\!] = \mu[\![\mathcal{E}]\!]$. $\square$

Because of duality, the existence of a $\wedge$-strategy π for $\mathcal{E}$ such that $\nu[\![\mathcal{E}(\pi)]\!] = \nu[\![\mathcal{E}]\!]$ is also not ensured in general.

4 Adaption of the Bellman-Ford Algorithm

In this section we present an adaption of the Bellman-Ford algorithm. Later, we will use this algorithm for computing *least solutions* of systems of *disjunctive* integer equations, or dually for computing *greatest solutions* of systems of *conjunctive* integer equations. Recall that an equation $\mathbf{x} = e$ is called disjunctive (resp. conjunctive) iff the right-hand side e does not contain the operator $\wedge$ (resp. the operator $\vee$). The Bellman-Ford algorithm is a graph algorithm for solving the *single source shortest path problem* for edge-weighted directed graphs (see e.g. Cormen et al. [6]). An edge-weighted directed graph together with a source can be represented

by a system of *conjunctive* simple integer equations whose greatest solution corresponds to the solution of the *single source shortest path problem*.

The adaption of the Bellman-Ford algorithm we present in this section can be applied to compute least solutions of systems of disjunctive integer equations. However, in order to simplify the argumentations, we consider a more general class of equations, which we will call *BF-eqautions*.

Let X be a set. A monotone function $f : (X \to \overline{\mathbb{Z}}) \to \overline{\mathbb{Z}}$ is called *Bellman-Ford function* (*BF-function* for short) iff the following holds for all $\rho, \rho' : X \to \overline{\mathbb{Z}}$ with $\rho' \geq \rho$: If $f(\rho') > f(\rho)$, then there exists some $x \in X$ and some $\delta \in \overline{\mathbb{Z}} \setminus \{-\infty\}$ such that the following properties are fulfilled:

1. $\rho'(x) > \rho(x)$.
2. $f(\rho') = \rho'(x) + \delta$.
3. $f(\rho'') \geq \rho''(x) + \delta$ for all $\rho'' \geq \rho'$.

Such an x is called *relevant for the statement* $f(\rho') > f(\rho)$.

The above definition also makes sense in its dual form. Using the dual form it is possible to compute greatest solutions of systems of conjunctive integer equations directly. We will not discuss these aspects in detail. Instead we will compute greatest solutions of systems of conjunctive integer equations by using duality as discussed in Subsection 3.4.

The above definition can also be applied to k-ary operators, since the set $\overline{\mathbb{Z}}^k$ can be identified with the set $\overline{\mathbb{Z}}^{\{1,\ldots,k\}} = \{1,\ldots,k\} \to \overline{\mathbb{Z}}$ of all functions from $\{1,\ldots,k\}$ to $\overline{\mathbb{Z}}$. Every nullary operator is a BF-operator. Moreover, all operators that occur in systems of disjunctive integer equations are BF-operators:

Lemma 2. *The operator $\vee$ and all operators that are monotone and upward-expansive are BF-operators.* □

An expression e (resp. an equation $\mathbf{x} = e$) is called *BF-expression* (resp. *BF-equation*) iff all operators occurring in e are BF-operators. Since the set of BF-functions is closed under composition, $[\![e]\!]$ is a BF-function for all BF-expressions e.

The most important step for the adaption of the Bellman-Ford algorithm consists in showing that, for a system $\mathcal{E}$ of BF-equations with n variables, the value $\mu[\![\mathcal{E}]\!](\mathbf{x})$ for a variable $\mathbf{x}$, whose value changes after the n-th Kleene iteration, must be ∞:

Lemma 3. *Let $\mathcal{E}$ be a system of BF-equations with n variables. Let $\rho^{(i)} := [\![\mathcal{E}]\!]^i(\underline{-\infty})$ for all $i \in \mathbb{N}$. The following holds for every variable $\mathbf{x} \in \mathbf{X}$: If there exists some $k > n$ with $\rho^{(k)}(\mathbf{x}) > \rho^{(n)}(\mathbf{x})$, then $\mu[\![\mathcal{E}]\!](\mathbf{x}) = \infty$.*

Proof. See Appendix B. The proof is based on a pumping lemma argument. If there exists some $k > n$ with $\rho^{(k)}(\mathbf{x}) > \rho^{(n)}(\mathbf{x})$, then there exists some cyclic dependency with a positive weight that can be iterated. □

As a corollary of Lemma 3 we get the following theorem which enables us to use more sophisticated fixpoint iteration schemas, whenever the least solution does not contain the value ∞.

Theorem 1. *Let $\mathcal{E}$ be a system of BF-equations with n variables. Assume further that $\mu[\![\mathcal{E}]\!] \lhd \underline{\infty}$. Then $\mu[\![\mathcal{E}]\!] = [\![\mathcal{E}]\!]^n(\underline{-\infty})$.*

Proof. Let $\rho^{(i)} := [\![\mathcal{E}]\!]^i(\underline{-\infty})$ for all $i \in \mathbb{N}$ and $\rho^* := \mu[\![\mathcal{E}]\!]$. For the sake of contradiction assume $\rho^* \neq \rho^{(n)}$. Then $\rho^* > \rho^{(n)}$. In particular $\rho^{(n)} \notin \mathbf{Sol}(\mathcal{E})$. Thus, there exists a variable $\mathbf{x} \in \mathbf{X}$ with $\rho^{(n+1)}(\mathbf{x}) > \rho^{(n)}(\mathbf{x})$. Using Lemma 3 we get $\rho^*(\mathbf{x}) = \infty$ — contradiction to the assumption that $\rho^*(\mathbf{x}) < \infty$. □

In our applications, the pre-conditions of Theorem 1 are mostly fulfilled. Since then the Kleene fixpoint iteration always terminates, it is also possible to use fixpoint iteration schemes that perform their evaluations according to the variable dependencies, e.g. Charlier and Hentenryck [4, 5], Fecht and Seidl [12], Kildall [20]. Although these methods are not an improvement for the worst case, in practice theses methods are mostly vastly superior to the Kleene fixpoint iteration.

We are now prepared to present the main result of this section which states that the least solution of a system $\mathcal{E}$ of BF-equations can be computed using the following adaption of the Bellman-Ford algorithm:

Algorithm 1 Adaption of the Bellman-Ford Algorithm

Input: A system $\mathcal{E}$ of BF-equations with n variables

Output: The least solution $\mu[\![\mathcal{E}]\!]$ of $\mathcal{E}$

$\rho \leftarrow \underline{-\infty}$

for $i = 1$ **to** n **do** $\rho \leftarrow [\![\mathcal{E}]\!]\rho$

$\rho \leftarrow \rho'$ where $\rho'(\mathbf{x}) = \begin{cases} \rho(\mathbf{x}) & \text{if } ([\![\mathcal{E}]\!]\rho)(\mathbf{x}) \leq \rho(\mathbf{x}) \\ \infty & \text{if } ([\![\mathcal{E}]\!]\rho)(\mathbf{x}) > \rho(\mathbf{x}) \end{cases}$ for all $\mathbf{x} \in \mathbf{X}$

for $i = 1$ **to** $n - 1$ **do** $\rho \leftarrow \rho \vee [\![\mathcal{E}]\!]\rho$

return ρ

The first loop of the algorithm performs n least fixpoint iteration steps. It then sets the values of variables that are not stable to ∞. The second loop propagates the value ∞.

Theorem 2 (Adaption of the Bellman-Ford Algorithm)**.** *Let $\mathcal{E}$ be a system of BF-equations (for instance a system of disjunctive integer equations) with n variables. The least solution $\mu[\![\mathcal{E}]\!]$ of $\mathcal{E}$ can be computed using the adaption of the Bellman-Ford algorithm (Algorithm 1). The algorithm performs $2n$ evaluations of the operator $[\![\mathcal{E}]\!]$.*

Proof. See Section B. □

Example 5. *We use Algorithm 1 to compute the least solution of the system*

$$\mathcal{E} = \{\mathbf{x} = 1,\ \mathbf{y} = \mathbf{y} + \mathbf{x} \vee -10,\ \mathbf{z} = \mathbf{x} \cdot^{+} \mathbf{y}\} \tag{23}$$

of disjunctive integer equations, where the monotone and expansive operator $\cdot^{+}$ is defined by

$$x \cdot^{+} y := \begin{cases} x \cdot y & \text{if } x, y > 0 \\ -\infty & \text{if } x \leq 0 \text{ or } y \leq 0 \end{cases} \qquad \text{for all } x, y \in \overline{\mathbb{Z}}. \tag{24}$$

After the execution of the first for-loop we obtain the pre-solution

$$\rho_0 := [\![\mathcal{E}]\!]^3(\underline{-\infty}) = [\![\mathcal{E}]\!]^2\ \{\mathbf{x} \mapsto 1,\ \mathbf{y} \mapsto -10,\ \mathbf{z} \mapsto -\infty\} \tag{25}$$

$$= [\![\mathcal{E}]\!]\ \{\mathbf{x} \mapsto 1,\ \mathbf{y} \mapsto -9,\ \mathbf{z} \mapsto -\infty\} = \{\mathbf{x} \mapsto 1,\ \mathbf{y} \mapsto -8,\ \mathbf{z} \mapsto -\infty\}. \tag{26}$$

Since $([\![\mathcal{E}]\!]\rho_0)(\mathbf{y}) = -7 > \rho_0(\mathbf{y}) = -8$ *holds, it follows* $\mu[\![\mathcal{E}]\!](\mathbf{y}) = \infty$. *The value of the variable* $\mathbf{y}$ *is thus set to* ∞, *i.e.,*

$$\rho_1 := \{\mathbf{x} \mapsto 1,\ \mathbf{y} \mapsto \infty,\ \mathbf{z} \mapsto -\infty\} \tag{27}$$

is the value of the program variable ρ *before the first execution of the body of the second for-loop. After the first execution of the body of the second for-loop we obtain*

$$\rho_2 := \rho_1 \vee [\![\mathcal{E}]\!]\rho_1 = \{\mathbf{x} \mapsto 1,\ \mathbf{y} \mapsto \infty,\ \mathbf{z} \mapsto \infty\}. \tag{28}$$

After the second and last execution of the body of the second for-loop we finally obtain

$$\mu[\![\mathcal{E}]\!] = \rho_3 := \rho_2 \vee [\![\mathcal{E}]\!]\rho_2 = \{\mathbf{x} \mapsto 1,\ \mathbf{y} \mapsto \infty,\ \mathbf{z} \mapsto \infty\}. \qquad \square \tag{29}$$

5 The Max-Strategy Improvement Algorithm

In this section, we present and discuss our max-strategy improvement algorithm. Our goal is to compute least solutions of systems of monotone equations over a complete *linearly* ordered set. We afterwards specialize the algorithm in order to compute least solutions of systems of integer equations.

5.1 The general framework

Our algorithm iterates over $\vee$-strategies. It maintains a current $\vee$-strategy and a current *approximate* to the least solution. A so called $\vee$*-strategy improvement operator* is used to determine a next, improved $\vee$-strategy. Whether or not a $\vee$-strategy represents an *improvement* may depend on the current approximate. It can indeed be the case that a switch from one $\vee$-strategy to another $\vee$-strategy is only then *profitable*, when it is known that the least solution is of a certain size. Hence, we talk about an *improvement* of a $\vee$-strategy w.r.t. an approximate:

Let $\mathcal{E}$ be a system of monotone equations over a complete linearly ordered set. Let σ be a $\vee$-strategy for $\mathcal{E}$ and ρ be a pre-solution of $\mathcal{E}(\sigma)$. The $\vee$-strategy σ' is called an *improvement of* σ *w.r.t.* ρ iff the following conditions are fulfilled:

1. If $\rho \notin \mathbf{Sol}(\mathcal{E})$, then $[\![\mathcal{E}(\sigma')]\!]\rho > \rho$.
2. For all $\vee$-expressions $e \in \mathcal{S}_\vee(\mathcal{E})$ the following holds: If $\sigma'(e) \neq \sigma(e)$, then $[\![e\sigma']\!]\rho > [\![e\sigma]\!]\rho$.

A function $P_\vee$ which assigns an improvement of σ w.r.t. ρ to every pair (σ, ρ) is called a $\vee$*-strategy improvement operator for* $\mathcal{E}$. An *improvement of a* $\wedge$*-strategy* π *w.r.t. a post-solution of* $\mathcal{E}(\pi)$ and a $\wedge$*-strategy improvement operator for* $\mathcal{E}$ are defined dually.

The first condition ensures a progress in the case that ρ is not yet a solution. The second condition ensures that the $\vee$-strategy σ may only modified in such a way that every modification ensures a local progress that is *strict*. This will later be important in order to ensure that the $\vee$-strategy improvement algorithm stays in a *feasible area*. We illustrate the definitions by an example:

Example 6 (Improvement of a $\vee$-Strategy)**.** *We consider the system*

$$\mathcal{E} = \{\mathbf{x} = (2 \cdot \mathbf{x} \wedge 10) \vee 1 \vee -\infty\}$$

of linear integer equations. The mapping $\sigma := \{(2 \cdot \mathbf{x} \wedge 10) \vee 1 \vee -\infty \mapsto 1\}$ *is a* $\vee$*-strategy for* $\mathcal{E}$ *and* $\rho := \{\mathbf{x} \mapsto 1\}$ *is a pre-solution (even a solution) of* $\mathcal{E}(\sigma) = \{\mathbf{x} = 1\}$*. However, the variable assignment* ρ *is not a solution of* $\mathcal{E}$*. An improvement of* σ *w.r.t.* ρ *is the* $\vee$*-strategy* $\sigma' = \{(2 \cdot \mathbf{x} \wedge 10) \vee 1 \vee -\infty \mapsto (2 \cdot \mathbf{x} \wedge 10)\}$*, since* $[\![\mathcal{E}(\sigma')]\!]\rho = \{\mathbf{x} \mapsto 2\} > \rho$ *and* $[\![((2 \cdot \mathbf{x} \wedge 10) \vee 1 \vee -\infty)\sigma']\!]\rho = 2 > 1 = [\![((2 \cdot \mathbf{x} \wedge 10) \vee 1 \vee -\infty)\sigma]\!]\rho$*. The* $\vee$*-strategy* σ' *is the only improvement of* σ *w.r.t.* ρ*.* □

An improvement σ' of a $\vee$-strategy σ w.r.t. a pre-solution ρ of $\mathcal{E}(\sigma)$ is, locally at the approximate ρ, at least as good as the $\vee$-strategy σ. That is, if σ' is an *improvement of* σ *w.r.t.* ρ, then $[\![\mathcal{E}(\sigma')]\!]\rho \geq [\![\mathcal{E}(\sigma)]\!]\rho$. We in particular have $\rho \in \mathbf{PreSol}(\mathcal{E}(\sigma'))$. A dual statement holds for $\wedge$-strategies.

In many cases, there exist several, different improvements of a $\vee$-strategy σ w.r.t. a pre-solution ρ of $\mathcal{E}(\sigma)$. Accordingly, there exist several, different possibilities for defining a $\vee$-strategy improvement operator. Under the assumption that the operator $\vee$ is only used in its binary version, one possibility is known as *all profitable switches* (see e.g. Björklund et al. [2], Bjorklund et al. [3]). Carried over to the case considered here, this means that the $\vee$-strategy σ will be modified at any $\vee$-expression $e_1 \vee e_2$ with $[\![e_1 \vee e_2]\!]\rho > [\![\sigma(e_1 \vee e_2)]\!]\rho$. According to the definition, the selection must be preserved at the other $\vee$-expressions. If $\vee$ is not only used in its binary version, we can think of $\sigma' = P_{\vee}^{\mathsf{eager}}(\sigma, \rho)$ as some arbitrary improvement of σ w.r.t. ρ that satisfies $[\![\mathcal{E}(\sigma')]\!]\rho = [\![\mathcal{E}]\!]\rho$. One consequence is that a $\vee$-strategy iteration based on the $\vee$-strategy improvement operator $P_{\vee}^{\mathsf{eager}}$ converges at least as fast as a Kleene fixpoint iteration. Note that $\sigma' = P_{\vee}^{\mathsf{eager}}(\sigma, \rho)$ is not necessarily uniquely determined.

Example 7 (The $\vee$-Strategy Improvement Operator $P_{\vee}^{\mathsf{eager}}$)**.** *The function*

$$\sigma = \{10 \vee \mathbf{x}_1 \mapsto \mathbf{x}_1,\ \mathbf{x}_2 + 1 \vee \mathbf{x}_1 \mapsto \mathbf{x}_2 + 1\} \tag{30}$$

is a $\vee$*-strategy for the system*

$$\mathcal{E} = \{\mathbf{x}_1 = 10 \vee \mathbf{x}_1,\ \mathbf{x}_2 = \mathbf{x}_2 + 1 \vee \mathbf{x}_1\} \tag{31}$$

of simple integer equations. The variable assignment $\rho = \{\mathbf{x}_1 \mapsto 0,\ \mathbf{x}_2 \mapsto -\infty\}$ *is a pre-solution of* $\mathcal{E}(\sigma) = \{\mathbf{x}_1 = \mathbf{x}_1, \mathbf{x}_1 = \mathbf{x}_2 + 1\}$*. We have*

$$\sigma' := P_{\vee}^{\mathsf{eager}}(\sigma, \rho) = \{10 \vee \mathbf{x}_1 \mapsto 10,\ \mathbf{x}_2 + 1 \vee \mathbf{x}_1 \mapsto \mathbf{x}_1\} \tag{32}$$

and thus $\mathcal{E}(\sigma') = \{\mathbf{x}_1 = 10,\ \mathbf{x}_2 = \mathbf{x}_1\}$*. In this example, the* $\vee$*-strategy* σ' *is not the only improvement of* σ *w.r.t.* ρ*. The* $\vee$*-strategies*

$$\sigma_1 = \{10 \vee \mathbf{x}_1 \mapsto 10,\ \mathbf{x}_2 + 1 \vee \mathbf{x}_1 \mapsto \mathbf{x}_2 + 1\},\ \textit{and} \tag{33}$$

$$\sigma_2 = \{10 \vee \mathbf{x}_1 \mapsto \mathbf{x}_1,\ \mathbf{x}_2 + 1 \vee \mathbf{x}_1 \mapsto \mathbf{x}_1\} \tag{34}$$

are also improvements of σ *w.r.t.* ρ*. However,*

$$[\![\mathcal{E}]\!]\rho = [\![\mathcal{E}(\sigma')]\!]\rho > [\![\mathcal{E}(\sigma_1)]\!]\rho, [\![\mathcal{E}(\sigma_2)]\!]\rho > [\![\mathcal{E}(\sigma)]\!]\rho. \tag{35}$$

Thus, locally at ρ*,* σ' *is the best possible improvement.* □

We can now formulate the $\vee$-strategy improvement algorithm for computing least solutions of systems of monotone equations over complete linearly ordered sets. This algorithm is parameterized with a $\vee$-strategy improvement operator $P_{\vee}$. The input is a system $\mathcal{E}$ of monotone equations over a complete linearly ordered set, a $\vee$-strategy σ_{init} for $\mathcal{E}$, and a pre-solution ρ_{init}

of $\mathcal{E}(\sigma_{\text{init}})$. In order to compute the *least* and not some *arbitrary* solution, we additionally require that $\rho_{\text{init}} \leq \mu[\![\mathcal{E}]\!]$ holds. The algorithm maintains a *current* $\vee$-strategy σ and a *current* approximate ρ to the least solution $\mu[\![\mathcal{E}]\!]$. During the computation, the current approximate ρ remains smaller than or equal to the least solution $\mu[\![\mathcal{E}]\!]$. Furthermore, ρ will grow in each iteration until the least solution of $\mathcal{E}$ is found. In each iteration we compute an improvement σ' of the current $\vee$-strategy σ w.r.t. ρ using the $\vee$-strategy improvement operator $P_\vee$, provided that ρ does not solve $\mathcal{E}$. The improvement σ' will be the new current $\vee$-strategy σ for the next iteration. Then, we will consider the system $\mathcal{E}(\sigma)$ of conjunctive monotone equations for which ρ is a pre-solution. The next current approximate ρ is then the least solution of $\mathcal{E}(\sigma)$ that is greater than or equal to ρ:

Algorithm 2 The $\vee$-Strategy Improvement Algorithm

Parameter : A $\vee$-strategy improvement operator $P_\vee$

Input : $\begin{cases} \text{A system } \mathcal{E} \text{ of monotone equations over a complete linearly ordered set} \\ \text{A } \vee\text{-strategy } \sigma_{\text{init}} \text{ for } \mathcal{E} \\ \text{A pre-solution } \rho_{\text{init}} \text{ of } \mathcal{E}(\sigma_{\text{init}}) \text{ with } \rho_{\text{init}} \leq \mu[\![\mathcal{E}]\!] \end{cases}$

Output : The least solution $\mu[\![\mathcal{E}]\!]$ of $\mathcal{E}$

$\sigma \leftarrow \sigma_{\text{init}}$
$\rho \leftarrow \rho_{\text{init}}$
while $(\rho \notin \mathbf{Sol}(\mathcal{E}))$ {
  $\sigma \leftarrow P_\vee(\sigma, \rho)$
  $\rho \leftarrow \mu_{\geq\rho}[\![\mathcal{E}(\sigma)]\!]$
}
return ρ

In order to execute the $\vee$-strategy improvement algorithm (Algorithm 2), we need a method for computing $\mu_{\geq\rho}[\![\mathcal{E}(\sigma)]\!]$ for the $\vee$-strategies σ and the approximates ρ that occur during the execution. Which method we have to use for that purpose depends on the class of systems of monotone equations under consideration. So far, we have:

Lemma 4. *Let $\mathcal{E}$ be a system of monotone equations over a complete linearly ordered set. For $i \in \mathbb{N}$, let ρ_i be the value of the program variable ρ and σ_i be the value of the program variable σ in the $\vee$-strategy improvement algorithm (Algorithm 2) after the i-th evaluation of the loop-body. The following statements hold for all $i \in \mathbb{N}$: (1) $\rho_i \leq \mu[\![\mathcal{E}]\!]$. (2) $\rho_i \in \mathbf{PreSol}(\mathcal{E}(\sigma_{i+1}))$. (3) If $\rho_i < \mu[\![\mathcal{E}]\!]$, then $\rho_{i+1} > \rho_i$. (4) If $\rho_i = \mu[\![\mathcal{E}]\!]$, then $\rho_{i+1} = \rho_i$.* □

As an immediate consequence of Lemma 4, we obtain:

Lemma 5. *Whenever the $\vee$-strategy improvement algorithm (Algorithm 2) terminates, it returns the least solution $\mu[\![\mathcal{E}]\!]$ of $\mathcal{E}$.* □

If we use the $\vee$-strategy improvement operator $P_\vee^{\mathsf{eager}}$, then the i-th approximate ρ_i is greater than or equal to the i-th Kleene approximate $[\![\mathcal{E}]\!]^i(\underline{\bot})$.

Example 8 (A Run of the $\vee$-Strategy Improvement Algorithm)**.** *Let us compute the least solution of the system*

$$\mathcal{E} = \{\mathbf{x}_1 = 0 \vee \mathbf{x}_1 + \mathbf{x}_2 - 4,\ \mathbf{x}_2 = -10 \vee ((\mathbf{x}_1 + 1 \vee 2 \cdot \mathbf{x}_2) \wedge 5)\} \tag{36}$$

of linear integer equations using our $\vee$-strategy improvement algorithm. We will use the $\vee$-strategy improvement operator $P_{\vee}^{\mathsf{eager}}$. Assume that the first $\vee$-strategy σ_{init} leads to the system

$$\mathcal{E}(\sigma_{\mathrm{init}}) = \{\mathbf{x}_1 = 0,\ \mathbf{x}_2 = -10\}. \tag{37}$$

The variable assignment $\rho_{\mathrm{init}} = \{\mathbf{x}_1 \mapsto 0,\ \mathbf{x}_2 \mapsto -10\}$ is a pre-solution (even a solution) of $\mathcal{E}(\sigma_{\mathrm{init}})$. The first application of $P_{\vee}^{\mathsf{eager}}$ leads to the $\vee$-strategy $\sigma_1 := P_{\vee}^{\mathsf{eager}}(\sigma_{\mathrm{init}}, \rho_{\mathrm{init}})$ with

$$\mathcal{E}(\sigma_1) = \{\mathbf{x}_1 = 0,\ \mathbf{x}_2 = \mathbf{x}_1 + 1 \wedge 5\}. \tag{38}$$

In the next step, we get $\rho_1 := \mu_{\geq \rho_{\mathrm{init}}}[\![\mathcal{E}(\sigma_1)]\!] = \{\mathbf{x}_1 \mapsto 0,\ \mathbf{x}_2 \mapsto 1\}$. Within the next iteration, we obtain the $\vee$-strategy $\sigma_2 := P_{\vee}^{\mathsf{eager}}(\sigma_1, \rho_1)$ with

$$\mathcal{E}(\sigma_2) = \{\mathbf{x}_1 = 0,\ \mathbf{x}_2 = 2 \cdot \mathbf{x}_2 \wedge 5\}. \tag{39}$$

Therefore, we obtain $\rho_2 := \mu_{\geq \rho_1}[\![\mathcal{E}(\sigma_2)]\!] = \{\mathbf{x}_1 \mapsto 0,\ \mathbf{x}_2 \mapsto 5\}$. In the last iteration, we finally obtain $\sigma_3 := P_{\vee}^{\mathsf{eager}}(\sigma_2, \rho_2)$, which leads to the system

$$\mathcal{E}(\sigma_3) = \{\mathbf{x}_1 = \mathbf{x}_1 + \mathbf{x}_2 - 4,\ \mathbf{x}_2 = 2 \cdot \mathbf{x}_2 \wedge 5\}. \tag{40}$$

Hence, $\rho_3 := \mu_{\geq \rho_2}[\![\mathcal{E}(\sigma_3)]\!] = \{\mathbf{x}_1 \mapsto \infty,\ \mathbf{x}_2 \mapsto 5\}$. Since ρ_3 is a solution of $\mathcal{E}$, the algorithm terminates and returns $\mu[\![\mathcal{E}]\!] = \rho_3$. □

It remains to explain how to compute $\mu_{\geq \rho}[\![\mathcal{E}(\sigma)]\!]$ for the $\vee$-strategies σ and the approximates ρ that occur during the execution of our $\vee$-strategy improvement algorithm. How this can be done depends on the properties of the systems of equations under consideration.

Similar to the well-known simplex algorithm for linear programming our $\vee$-strategy improvement algorithm must be started in a *feasible area*. It then stays in the feasible area.

5.2 Feasibility

In this subsection, we define our notion of *feasibility*. In order to do so, we first define *derived equations* for systems of basic integer equations as follows:

Let $\mathcal{E}$ be a system of basic integer equations and ρ be a pre-solution of $\mathcal{E}$. The set $\mathcal{D}_\rho(\mathcal{E})$ of all *w.r.t. ρ derived equations of $\mathcal{E}$* is the smallest set of basic integer equations such that the following statements hold:

1. If $\mathbf{x} = e \in \mathcal{E}$ with $-\infty < \rho(\mathbf{x}) = [\![e]\!]\rho < \infty$, then $\mathbf{x} = e \in \mathcal{D}_\rho(\mathcal{E})$.
2. If $\mathbf{x} = e[\mathbf{x}'] \in \mathcal{D}_\rho(\mathcal{E})$ and $\mathbf{x}' = e' \in \mathcal{D}_\rho(\mathcal{E})$, then $\mathbf{x} = e[e'] \in \mathcal{D}_\rho(\mathcal{E})$.

In the above definition $e[\mathbf{x}']$ denotes an expression which contains the variable $\mathbf{x}'$ at the position denoted by $e[]$. $e[]$ itself is a *single hole context*. The expression $e[e']$ finally denotes the expression which contains the expression e' instead of the variable $\mathbf{x}'$ at the position denoted by $e[]$.

Example 9 (Derived Equations)**.** *The set $\mathcal{D}_\rho(\mathcal{E})$ of all w.r.t. $\rho = \{\mathbf{x}_1 \mapsto 1\}$ derived equations of the system $\mathcal{E} = \{\mathbf{x}_1 = 2 \cdot \mathbf{x}_1\}$ of basic integer equations is $\mathcal{D}_\rho(\mathcal{E}) = \emptyset$, because $1 = \rho(\mathbf{x}_1) < [\![2 \cdot \mathbf{x}_1]\!]\rho = 2$.* □

We are now prepared to define our notion of *feasibility*: A pre-solution ρ of a system $\mathcal{E}$ of basic integer equations is called *($\mathcal{E}$-)feasible* iff the following statements are fulfilled:

1. There does not exist an equation $\mathbf{x} = e$ in $\mathcal{E}$ with $[\![e]\!]\rho = -\infty$ and $e \neq -\infty$.
2. There does not exist a derived equation $\mathbf{x} = e \in \mathcal{D}_\rho(\mathcal{E})$ with $\mathbf{x} \in \mathbf{Vars}(e)$.

A pre-solution ρ of a system $\mathcal{E}$ of conjunctive integer equations is called *($\mathcal{E}$-)feasible* iff it is $\mathcal{E}(\pi)$-feasible for all $\pi \in \Pi$. A system of conjunctive integer equations is called *feasible* iff it has a feasible pre-solution.

Example 10 (Feasibility)**.** *We consider the system $\mathcal{E} = \{\mathbf{x}_1 = 2 \cdot \mathbf{x}_1 \wedge 10\}$ of conjunctive linear integer equations. Let $\pi_1 := \{2 \cdot \mathbf{x}_1 \wedge 10 \mapsto 2 \cdot \mathbf{x}_1\}$ and $\pi_2 := \{2 \cdot \mathbf{x}_1 \wedge 10 \mapsto 10\}$ be the $\wedge$-strategies for $\mathcal{E}$. The solution $\rho_0 := \{\mathbf{x}_1 \mapsto 0\}$ of $\mathcal{E}$ is not feasible, since $\mathbf{x}_1 = 2 \cdot \mathbf{x}_1 \in \mathcal{D}_{\rho_0}(\mathcal{E}(\pi_1))$ and thus ρ_0 is not $\mathcal{E}(\pi_1)$-feasible. The pre-solution $\rho_1 := \{\mathbf{x}_1 \mapsto 1\}$ of $\mathcal{E}$ is $\mathcal{E}$-feasible, since $\mathcal{D}_{\rho_1}(\mathcal{E}(\pi_1)) = \mathcal{D}_{\rho_1}(\mathcal{E}(\pi_2)) = \emptyset$. The greatest solution $\rho_2 := \{\mathbf{x}_1 \mapsto 10\}$ of $\mathcal{E}$ is also $\mathcal{E}$-feasible, since $\mathcal{D}_{\rho_2}(\mathcal{E}(\pi_1)) = \emptyset$ and $\mathcal{D}_{\rho_2}(\mathcal{E}(\pi_2)) = \{\mathbf{x} = 10\}$.* □

The set of feasible pre-solutions is upward closed in the following sense:

Lemma 6. *Let $\mathcal{E}$ be a system of conjunctive integer equations and ρ be a feasible pre-solution of $\mathcal{E}$. Every pre-solution ρ' of $\mathcal{E}$ with $\rho' \geq \rho$ is feasible.*

Proof. See Section C. □

Since the greatest solution $\nu[\![\mathcal{E}]\!]$ of a system $\mathcal{E}$ of conjunctive integer equations (by the Knaster-Tarski fixpoint theorem) is greater than or equal to any pre-solution of $\mathcal{E}$, we can in particular conclude (using Lemma 6) that the greatest solution $\nu[\![\mathcal{E}]\!]$ is feasible, whenever $\mathcal{E}$ is feasible. In the next step, we show that every *feasible* system $\mathcal{E}$ of conjunctive integer equations has exactly one *feasible solution*. Thus, this solution must be the greatest solution $\nu[\![\mathcal{E}]\!]$ of $\mathcal{E}$.

Lemma 7. *Let $\mathcal{E}$ be a system of basic integer equations and ρ be a feasible solution of $\mathcal{E}$. Then $\rho = \nu[\![\mathcal{E}]\!]$.*

Proof. We do induction on $\mathbf{Vars}(\mathcal{E})$. If $\mathbf{Vars}(\mathcal{E}) = \emptyset$, then the statement is fulfilled, since $\mathcal{E}$ has exactly one solution and this solution is feasible. Thus, we assume that $\mathbf{Vars}(\mathcal{E}) \neq \emptyset$. Let $\mathbf{x} \in \mathbf{Vars}(\mathcal{E})$ and $\mathbf{x} = e \in \mathcal{E}$ be the equations for the variable $\mathbf{x}$.

Case 1: $\mathbf{x} \notin \mathbf{Vars}(e)$. Let $\mathcal{E}' := \mathcal{E}[e/\mathbf{x}]$. Since $\mathcal{D}_\rho(\mathcal{E}') \subseteq \mathcal{D}_\rho(\mathcal{E})$, ρ is a feasible solution of $\mathcal{E}'$. Since $\mathbf{Vars}(\mathcal{E}') \subset \mathbf{Vars}(\mathcal{E})$, it follows $\rho = \nu[\![\mathcal{E}']\!]$ using the induction hypothesis. In order to show $\rho = \nu[\![\mathcal{E}]\!]$, let $\rho' \in \mathbf{Sol}(\mathcal{E})$. Then $\rho' \in \mathbf{Sol}(\mathcal{E}')$. Hence, we obtain $\rho' \leq \nu[\![\mathcal{E}']\!] = \rho$.

Case 2: $\mathbf{x} \in \mathbf{Vars}(e)$. Since ρ is a feasible solution of $\mathcal{E}$, we have $\rho(\mathbf{x}) = \infty$. Let $\mathcal{E}' := (\mathcal{E} \oplus \{\mathbf{x} = \infty\})[\infty/\mathbf{x}]$. Since $\mathcal{D}_\rho(\mathcal{E}') \subseteq \mathcal{D}_\rho(\mathcal{E})$, ρ is a feasible solution of $\mathcal{E}'$. Since $\mathbf{Vars}(\mathcal{E}') \subset \mathbf{Vars}(\mathcal{E})$, we get $\rho = \nu[\![\mathcal{E}']\!]$ using the induction hypothesis. In order to show $\rho = \nu[\![\mathcal{E}]\!]$, let $\rho' \in \mathbf{Sol}(\mathcal{E})$. Thus, $\rho' \in \mathbf{PreSol}(\mathcal{E}')$. Hence, $\rho' \leq \nu[\![\mathcal{E}']\!] = \rho$.

□

We now generalize the statement of Lemma 7:

Theorem 3 (Uniqueness of Feasible Solutions)**.** *Let $\mathcal{E}$ be a system of conjunctive integer equations and ρ be a feasible solution of $\mathcal{E}$. Then $\rho = \nu[\![\mathcal{E}]\!]$.*

Proof. There exists a $\wedge$-strategy $\pi \in \Pi$ for $\mathcal{E}$ such that $[\![\mathcal{E}(\pi)]\!]\rho = [\![\mathcal{E}]\!]\rho = \rho$. Hence, $\rho \in \mathbf{Sol}(\mathcal{E}(\pi))$ and ρ is by definition a feasible solution of $\mathcal{E}(\pi)$. By Lemma 7, we get $\rho = \nu[\![\mathcal{E}(\pi)]\!]$. Since by the Knaster-Tarski fixpoint theorem $\nu[\![\mathcal{E}(\pi)]\!] \geq \nu[\![\mathcal{E}]\!]$ and ρ is a solution of $\mathcal{E}$, we get $\rho = \nu[\![\mathcal{E}]\!]$. □

By Theorem 3, *the* (uniquely determined) feasible solution of a feasible system $\mathcal{E}$ of conjunctive integer equations is the greatest solution $\nu[\![\mathcal{E}]\!]$ of $\mathcal{E}$. Thus, we can compute it through our adaption of the Bellman-Ford algorithm (see Theorem 2).

In order to show that our $\vee$-strategy improvement algorithm stays in the feasible area, it remains to show that every $\vee$-strategy improvement step preserves feasibility:

Lemma 8 ($\vee$-Strategy Improvement Steps Preserve Feasibility)**.** *Let $\mathcal{E}$ be a system of integer equations, $\sigma \in \Sigma$ a $\vee$-strategy for $\mathcal{E}$ and ρ be a feasible pre-solution of $\mathcal{E}(\sigma)$. Let σ' be an improvement of σ w.r.t. ρ. Then ρ is also a feasible pre-solution of $\mathcal{E}(\sigma')$.*

Proof. See Section C. □

It remains to show how we can benefit from the above result in order to compute $\mu_{\geq\rho}[\![\mathcal{E}(\sigma)]\!]$ within the $\vee$-strategy improvement algorithm. For that, let $\mathcal{E}$ be a system of integer equations. Assume that σ_{init} is a $\vee$-strategy for $\mathcal{E}$ and ρ_{init} is a *feasible* pre-solution of $\mathcal{E}(\sigma_{\text{init}})$ with $\rho_{\text{init}} \leq \mu[\![\mathcal{E}]\!]$. For $i \in \mathbb{N}$, let σ_i and ρ_i denote the values of the program variables σ and ρ after the i-th evaluation of the body of the loop in the $\vee$-strategy improvement algorithm (Algorithm 2). We getx:

Lemma 9. *For all $i \in \mathbb{N}$, ρ_i is a feasible pre-solution of $\mathcal{E}(\sigma_{i+1})$. Moreover, $\rho_{i+1} = \nu[\![\mathcal{E}(\sigma_{i+1})]\!]$.* □

It remains to show that the $\vee$-strategy improvement algorithm terminates at the latest after considering all $\vee$-strategies $\sigma \in \Sigma$ for $\mathcal{E}$. For the sake of contradiction assume that this is not the case, i.e., assume that $\rho_{|\Sigma|} \notin \mathbf{Sol}(\mathcal{E})$. By Lemma 4, we get

$$\rho_{i+1} > \rho_i \qquad \text{for all } i \in \{0, \ldots, |\Sigma|\}. \tag{41}$$

Using the pigeonhole principle, we get that there exists a $\vee$-strategy $\sigma \in \Sigma$ which occurs twice in the sequence $\sigma_1, \ldots, \sigma_{|\Sigma|+1}$, i.e., there exist $k_1, k_2 \in \{1, \ldots, |\Sigma|+1\}$ with $k_1 < k_2$ and $\sigma_{k_1} = \sigma_{k_2}$. Using Lemma 9 we finally get $\rho_{k_1} = \nu[\![\mathcal{E}(\sigma_{k_1})]\!] = \nu[\![\mathcal{E}(\sigma_{k_2})]\!] = \rho_{k_2}$. This is in contradiction to (41).

The number $|\Sigma|$ of all $\vee$-strategies is exponential in the number of $\vee$-expressions. Whether or not there exist systems of linear integer equations, for which, when we use the $\vee$-strategy improvement operator $P_{\vee}^{\mathsf{eager}}$, we in fact have to do exponentially many $\vee$-strategy improvement steps for computing least solutions, is not known.

It remains to estimate the worst case running time of the loop-body w.r.t. the uniform cost measure. Since, by Lemma 9, $\rho_{i+1} = \nu[\![\mathcal{E}(\sigma_{i+1})]\!]$ for all $i \in \mathbb{N}$, we have to compute the greatest solution of a system of conjunctive integer equations. By Theorem 2, this can be done through our adaption of the Bellman-Ford algorithm. In practice we can compute ρ_{i+1} more efficiently. Since $\mathcal{E}(\sigma_{i+1})$ is feasible, the greatest solution $\rho_{i+1} = \nu[\![\mathcal{E}(\sigma_{i+1})]\!]$ of $\mathcal{E}(\sigma_{i+1})$ is, because of Theorem 1, the $|\mathbf{X}|$-th Kleene approximate. Therefore, we can compute it using an *arbitrary* generic fixpoint algorithm. Algorithms that take variable dependencies into account [4, 5, 12, 20] have proven themselves good. Summarizing, we have shown the following result:

Theorem 4. *Let $\mathcal{E}$ be a system of integer equations. Let σ_{init} be a $\vee$-strategy for $\mathcal{E}$ and ρ_{init} be a feasible pre-solution of $\mathcal{E}(\sigma_{\text{init}})$ with $\rho_{\text{init}} \leq \mu[\![\mathcal{E}]\!]$. The $\vee$-strategy improvement algorithm (Algorithm 2) computes the least solution $\mu[\![\mathcal{E}]\!]$ of $\mathcal{E}$. At most $|\Sigma|$ $\vee$-strategy improvement steps are performed. Each $\vee$-strategy improvement step can be carried out by performing a greatest fixpoint iteration that terminates after at most $|\mathbf{X}|$ steps.* □

5.3 Determining a feasible $\vee$-strategy

Until now, we have assumed that σ_{init} is a $\vee$-strategy for $\mathcal{E}$ and ρ_{init} is a *feasible* pre-solution of $\mathcal{E}(\sigma_{\text{init}})$ with $\rho_{\text{init}} \leq \mu[\![\mathcal{E}]\!]$. We are now going to explain how to abandon this precondition. For a system $\mathcal{E}$ of integer equations, we set

$$\mathcal{E} \vee -\infty := \{\mathbf{x} = e \vee -\infty \mid \mathbf{x} = e \in \mathcal{E}\}.$$

Obviously, $\mathcal{E}$ and $\mathcal{E} \vee -\infty$ have the same least solution, i.e., we have $\mu[\![\mathcal{E} \vee -\infty]\!] = \mu[\![\mathcal{E}]\!]$. Thus we can solve the system $\mathcal{E} \vee -\infty$ instead of the system $\mathcal{E}$. The advantage of considering $\mathcal{E} \vee -\infty$ instead of $\mathcal{E}$ is that $\underline{-\infty}$ is a feasible pre-solution of $(\mathcal{E} \vee -\infty)(\sigma_{-\infty})$. Here, the $\vee$-strategy $\sigma_{-\infty}$ is an arbitrary $\vee$-strategy for $\mathcal{E} \vee -\infty$ that assigns the expression $-\infty$ to every expression $e \vee -\infty$, i.e., $\sigma_{-\infty}(e \vee -\infty) = -\infty$ holds for all equations $\mathbf{x} = e$ of $\mathcal{E}$. Hence, Algorithm 2 can be started with $\sigma_{\text{init}} := \sigma_{-\infty}$ and $\rho_{\text{init}} := \underline{-\infty}$ for computing $\mu[\![\mathcal{E} \vee -\infty]\!] = \mu[\![\mathcal{E}]\!]$.

For the complexity estimation it is important to note that, since $\mathcal{E} \vee -\infty$ is considered instead of $\mathcal{E}$, the number of $\vee$-expressions increases by the number $|\mathbf{X}|$ of variables of $\mathcal{E}$. Thus, we have $|\Sigma_{\mathcal{E} \vee -\infty}| = 2^{|\mathbf{X}|} \cdot |\Sigma_{\mathcal{E}}|$. However, all right-hand sides of $\mathcal{E} \vee -\infty$ are of the form $e \vee -\infty$. It can be shown that $\sigma_j(e \vee -\infty) = e$ holds for all $j \geq i$, if $\sigma_i(e \vee -\infty) = e$ holds for $i \in \mathbb{N}$. In consequence, we need at most $|\mathbf{X}| \cdot |\Sigma_{\mathcal{E}}|$ $\vee$-strategy improvement steps. Summarizing, we have shown the following main result:

Theorem 5. *Let $\mathcal{E}$ be a system of integer equations. The least solution $\mu[\![\mathcal{E}]\!]$ of $\mathcal{E}$ can be computed using the $\vee$-strategy improvement algorithm (Algorithm 2). At most $|\mathbf{X}| \cdot |\Sigma|$ $\vee$-strategy improvement steps are performed. Each $\vee$-strategy improvement step can be carried out by performing a greatest fixpoint iteration that terminates after at most $|\mathbf{X}|$ steps.*

Proof. See Section C. □

6 Systems of Extended Integer Equations

In this section, we extend the applicability of our $\vee$-strategy improvement algorithm by allowing operators that are not expansive but at least equivalent to expansive operators on the regions where they evaluate to a value greater than $-\infty$.

An operator $f : (X \to \overline{\mathbb{Z}}) \to \overline{\mathbb{Z}}$ is called *quasi-expansive* iff there exists some $X' \subseteq X$ and some expansive operator $f^{>-\infty} : (X' \to \overline{\mathbb{Z}}) \to \overline{\mathbb{Z}}$ such that $f(\rho) = f^{>-\infty}(\rho|_{X'})$ for all $\rho : X \to \overline{\mathbb{Z}}$ with $f(\rho) > -\infty$.

The binary operators ; and $\geq z?$ (for $z \in \overline{\mathbb{Z}}$) that are defined by

$$x; y := \begin{cases} -\infty & \text{if } x = -\infty \\ y & \text{otherwise} \end{cases} \qquad \text{for all } x, y \in \overline{\mathbb{Z}}, \text{ and} \tag{42}$$

$$x \geq z?\ y := \begin{cases} -\infty & \text{if } x < z \\ y & \text{otherwise} \end{cases} \qquad \text{for all } x, y, z \in \overline{\mathbb{Z}} \tag{43}$$

are important instances of monotone and quasi-expansive. However, they are not expansive. Thus, they are not allowed to occur within systems of integer equations.

Every expansive function is quasi-expansive, but not vice-versa. We extend integer expressions (resp. integer equations) to *extended integer expressions* (resp. *extended integer equations*) by allowing monotone and quasi-expansive operators instead of monotone and expansive operators.

We now show that our $\vee$-strategy improvement algorithm is also capable of solving systems of *extended* integer equations. For that, we define the transformation $[\cdot]^{>-\infty}$ as follows:

$$[-\infty]^{>-\infty} := -\infty \tag{44}$$

$$[\mathbf{x}]^{>-\infty} := \mathbf{x} \tag{45}$$

$$[f(e_1,\ldots,e_k)]^{>-\infty} := f^{>-\infty}([e_1]^{>-\infty},\ldots,[e_k]^{>-\infty}) \tag{46}$$

$$[e_1 \wedge \cdots \wedge e_k)]^{>-\infty} := [e_1]^{>-\infty} \wedge \cdots \wedge [e_k]^{>-\infty} \tag{47}$$

Here, $\mathbf{x}$ is a variable, $f \notin \{-\infty, \wedge\}$ is a k-ary operator (recall that constants are nullary operators), and $e_1, e_2, \ldots, e_k$ are conjunctive extended integer expressions. If e is a conjunctive extended integer expression, then $[e]^{>-\infty}$ is a conjunctive integer expression. Finally, we set

$$[\mathcal{E}]^{>-\infty} := \{\mathbf{x} = [e]^{>-\infty} \mid \mathbf{x} = e \in \mathcal{E}\} \tag{48}$$

for every system $\mathcal{E}$ of conjunctive extended integer equations. $[\mathcal{E}]^{>-\infty}$ is then a system of conjunctive integer equations. Using the results of Section 5, we can straightforwardly prove the following statement:

Lemma 10. *Assume that each equation of the system $\mathcal{E}$ of extended integer equations is of the form $\mathbf{x} = -\infty \vee e$ and that the $\vee$-strategy σ_{init} maps every right-hand side to $-\infty$. For $i \in \mathbb{N}$, let σ_i and ρ_i denote the values of the program variables σ and ρ after the i-th evaluation of the body of the loop of the $\vee$-strategy improvement algorithm (Algorithm 2). For all $i \in \mathbb{N}$, $\mu_{\geq \rho_i}[\![\mathcal{E}(\sigma_{i+1})]\!] = \mu_{\geq \rho_i}[\![[\mathcal{E}(\sigma_{i+1})]^{>-\infty}]\!]$ and ρ_i is a feasible pre-solution of $[\mathcal{E}(\sigma_{i+1})]^{>-\infty}$. Thus,*

$$\rho_{i+1} = \mu_{\geq \rho_i}[\![\mathcal{E}(\sigma_{i+1})]\!] = \nu[\![[\mathcal{E}(\sigma_{i+1})]^{>-\infty}]\!] = \nu[\![\mathcal{E}(\sigma_{i+1})]\!] \tag{49}$$

for all $i \in \mathbb{N}$. □

Because of Lemma 10, we can, for all $i \in \mathbb{N}$, compute ρ_{i+1} by performing a greatest fixpoint computation on $[\![\mathcal{E}(\sigma_{i+1})]\!]$ which terminates at the latest after $|\mathbf{X}|$ fixpoint iteration steps, i.e., $\rho_{i+1} = \nu[\![\mathcal{E}(\sigma_{i+1})]\!] = [\![\mathcal{E}(\sigma_{i+1})]\!]^{|\mathbf{X}|}(\underline{\infty})$. Hence, we have shown the following theorem:

Theorem 6. *Let $\mathcal{E}$ be a system of extended integer equations. The least solution $\mu[\![\mathcal{E}]\!]$ of $\mathcal{E}$ can be computed through our $\vee$-strategy improvement algorithm (Algorithm 2). At most $|\mathbf{X}| \cdot |\Sigma|$ $\vee$-strategy improvement steps are performed. Each $\vee$-strategy improvement step can be carried out by performing a greatest fixpoint iteration that terminates after at most $|\mathbf{X}|$ steps.* □

Example 11. *Let us compute the least solution $\mu[\![\mathcal{E}]\!]$ of the system*

$$\mathcal{E} = \{\mathbf{x} = -\infty \vee 0 \vee \mathbf{x} + \mathbf{y},\ \mathbf{y} = -\infty \vee \mathbf{x}; 1\} \tag{50}$$

of extended integer equations. For $i \in \mathbb{N}$, let σ_i and ρ_i denote the values of the program variables σ and ρ after the i-th evaluation of the body of the loop in the $\vee$-strategy improvement algorithm (Algorithm 2). We get

$$\mathcal{E}(\sigma_0) = \{\mathbf{x} = -\infty,\ \mathbf{y} = -\infty\} \qquad \mathcal{E}(\sigma_1) = \{\mathbf{x} = 0,\ \mathbf{y} = -\infty\} \tag{51}$$

$$\mathcal{E}(\sigma_2) = \{\mathbf{x} = 0,\ \mathbf{y} = \mathbf{x}; 1\} \qquad \mathcal{E}(\sigma_3) = \{\mathbf{x} = \mathbf{x} + \mathbf{y},\ \mathbf{y} = \mathbf{x}; 1\} \tag{52}$$

Since $\rho_3 = \nu[\![\mathcal{E}(\sigma_3)]\!] = [\![\mathcal{E}(\sigma_3)]\!]^2\underline{\infty} = \{\mathbf{x} \mapsto \infty,\ \mathbf{y} \mapsto 1\}$ solves $\mathcal{E}$, we finally get $\mu[\![\mathcal{E}]\!] = \{\mathbf{x} \mapsto \infty,\ \mathbf{y} \mapsto 1\}$. □

7 Abstract Interpretation over Intervals and Zones

We now apply our techniques to perform static program analysis by abstract interpretation over integer intervals as studied by Cousot and Cousot [8, 9]. That is, for each program point, we aim at computing small upper bounds for expressions of the form x and $-x$, where x is a program variable. We then extend this to work over the abstract domain of integer zones (cf. Miné [22]). That is, we additionally aim at computing small upper bounds for expressions of the form $x - y$, where x and y are program variables.

7.1 Notations

The transpose of a matrix A is denoted by $A^\top$. The i-th row (resp. j-th column) of a matrix A is denoted by $A_{i\cdot}$ (resp. $A_{\cdot j}$). Accordingly, $A_{i\cdot j}$ denotes the component in the i-th row and the j-th column. This notation is also used for vectors and functions $f : X \to Y^k$, i.e., $f_{i\cdot}(x) = (f(x))_{i\cdot}$ for all $x \in X$ and all $i \in \{1, \dots, k\}$. For $x, y \in \overline{\mathbb{R}}^n$, we write $x \leq y$ iff $x_{i\cdot} \leq y_{i\cdot}$ for all $i \in \{1, \dots, n\}$. Partially ordered by $\leq$, $\overline{\mathbb{R}}^n$ is a complete lattice. Hence, the operators $\vee$ and $\wedge$ are well-defined on $\overline{\mathbb{R}}^n$.

7.2 Programs and their collecting semantics

In this article, a *program* is given by a *control flow graph* $G = (N, E, \mathbf{st})$ that consists of a finite set N of *control points*, a finite set $E \subseteq N \times \mathbf{Stmt} \times N$ of *(control flow) edges* and a special *start control point* $\mathbf{st} \in N$. $\mathbf{Stmt}$ is a set of statements. We assume that the program G uses $n \in \mathbb{N}_{>0}$ variables that take values from $\mathbb{Z}$. We fix a so-called *template constraint matrix* $T \in \{-1, 0, 1\}^{m \times n}$. Each row of the template constraint matrix represents a template (here: a linear function). We restrict ourselves to the case where we can only talk about upper and lower bounds for program variables and upper bounds for the differences of program variables. That is, we assume that each row of T contains at most one 1 and at most one -1. For simplicity, we further assume w.l.o.g. that each row of T contains at least one non-zero entry.

Example 12. *For $n = 2$ we might, for instance, choose*

$$T = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 1 \end{pmatrix}. \tag{53}$$

This template constraint matrix allows us to reason about upper bounds on the program variables $\mathbf{x}_1$ and $\mathbf{x}_2$, as well as upper bounds on the difference $\mathbf{x}_2 - \mathbf{x}_1$. □

We assume that each statement $s \in \mathbf{Stmt}$ is of the form

$$T\mathbf{x} \leq c; \mathbf{x} := A\mathbf{x} + b, \tag{54}$$

where $c \in \overline{\mathbb{Z}}^m$, $A \in \mathbb{Z}^{n \times n}$, and $b \in \mathbb{Z}^n$ (recall that T is the template constraint matrix we have fixed beforehand). Hence, a statement combines a guard ($T\mathbf{x} \leq c$) with an assignment ($\mathbf{x} := A\mathbf{x} + b$).

The collecting semantics $[\![T\mathbf{x} \leq c; \mathbf{x} := A\mathbf{x} + b]\!] : 2^{\mathbb{Z}^n} \to 2^{\mathbb{Z}^n}$ of the statement $T\mathbf{x} \leq c; \mathbf{x} := A\mathbf{x} + b$ is defined by

$$[\![T\mathbf{x} \leq c; \mathbf{x} := A\mathbf{x} + b]\!]X = \{Ax + b \mid x \in X,\ Tx \leq c\} \qquad \text{for all } X \subseteq \mathbb{Z}^n. \tag{55}$$

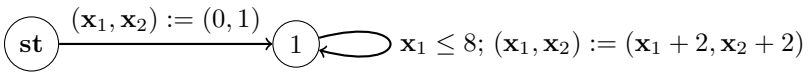

Figure 1: A simple program

In order to model user inputs we could additionally allow non-deterministic assignments, i.e., statements of the form $\mathbf{x}_i := ?$, where the collecting semantics $[\![\mathbf{x}_i := ?]\!] : 2^{\mathbb{Z}^n} \to 2^{\mathbb{Z}^n}$ of $\mathbf{x}_i := ?$ is defined by $[\![\mathbf{x}_i := ?]\!]X = \{(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n)^\top \mid (x_1, \ldots, x_n)^\top \in X,\ y \in \mathbb{Z}\}$ for all $X \subseteq \mathbb{Z}^n$. However, we abandon this, because these statements can be simulated by introducing new program variables or additional loops.

The collecting semantics V of a non-deterministic program G finally associates a set of vectors from $\mathbb{Z}^n$ to each control point $v \in N$. It is defined as the least solution of the following constraint system:

$$\mathbf{V}[\mathbf{st}] \supseteq \mathbb{Z}^n \tag{56}$$

$$\mathbf{V}[v] \supseteq [\![s]\!](\mathbf{V}[u]) \qquad \text{for each control-flow edge } (u, s, v) \in E \tag{57}$$

The unknowns $\mathbf{V}[v]$, $v \in N$ take values in $2^{\mathbb{Z}^n}$. We denote the components of the collecting semantics V by $V[v]$ for $v \in N$.

Example 13. *Figure 1 shows a program G with the two integer variables $\mathbf{x}_1$ and $\mathbf{x}_2$. The collecting semantics V of G is given by $V[\mathbf{st}] = \mathbb{Z}^2$, and $V[1] = \{(0,1), (2,3), (4,5), (6,7), (8,9), (10,11)\}$.* □

7.3 The program's abstract semantics

We use the abstract domain of integer zones as studied by Miné [22] to define the program's abstract semantics. In this article we restrict ourselves to the integer zones that can be expressed through the template constraint matrix T. That is, the abstract elements are vectors from $\overline{\mathbb{Z}}^m$ and the *concretization* $\gamma : \overline{\mathbb{Z}}^m \to 2^{\mathbb{Z}^n}$ and the *abstraction* $\alpha : 2^{\mathbb{Z}^n} \to \overline{\mathbb{Z}}^m$ are defined by

$$\gamma(d) := \{x \in \mathbb{Z}^n \mid Tx \leq d\} \qquad \text{for all } d \in \overline{\mathbb{Z}}^m, \text{ and} \tag{58}$$

$$\alpha(X) := \min\{d \in \overline{\mathbb{Z}}^m \mid \gamma(d) \supseteq X\} \qquad \text{for all } X \subseteq \mathbb{Z}^n. \tag{59}$$

Each abstract value $d \in \overline{\mathbb{Z}}^m$ represents the set $\gamma(d)$ of concrete values. The abstraction α and the concretization γ form a *Galois connection* (see e.g. Sankaranarayanan et al. [26]). Recall that the template constraint matrix T is an element of the set $\{-1, 0, 1\}^{m \times n}$ and each row contains at most one -1 and at most one 1.

Example 14. *Figure 2 shows $\gamma(d)$ for $d = (3, 2, 1)^\top$, where T is defined as in Example 12.* □

The *abstract semantics* $[\![s]\!]^\sharp : \overline{\mathbb{Z}}^m \to \overline{\mathbb{Z}}^m$ of a statement $s \in \mathbf{Stmt}$ is defined by $[\![s]\!]^\sharp := \alpha \circ [\![s]\!] \circ \gamma$. Hence, we are concerned with best abstract transformers (cf. Cousot and Cousot [8]). The abstract semantics $V^\sharp$ of a program G is then defined as the least solution of the following constraint system:

$$\mathbf{V}^\sharp[\mathbf{st}] \geq \alpha(\mathbb{Z}^n) \tag{60}$$

$$\mathbf{V}^\sharp[v] \geq [\![s]\!]^\sharp(\mathbf{V}^\sharp[u]) \qquad \text{for each control-flow edge } (u, s, v) \in E \tag{61}$$

Here, the variables $\mathbf{V}^\sharp[v]$, $v \in N$ take values in $\overline{\mathbb{Z}}^m$. We denote the components of the abstract semantics $V^\sharp$ by $V^\sharp[v]$ for all $v \in N$. The abstract semantics $V^\sharp$ safely over-approximates the collection semantics V, i.e., $V^\sharp[v] \supseteq V[v]$ for all $v \in N$.

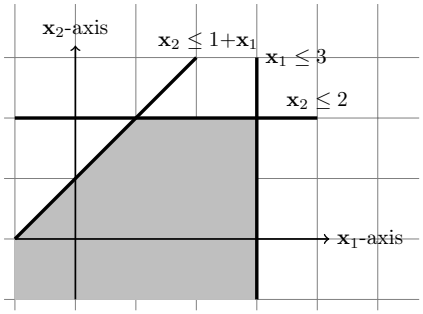

Figure 2: $\gamma(d)$

7.4 Intervals

Before dealing with the full domain of integer zones, we consider the simpler abstract domain of integer intervals. This case is obtained, if the matrix T provides rows with single non-zero entries 1 or -1 only. Let us for simplicity assume that $T_{i\cdot}$ contains a 1 in column i and $T_{n+i\cdot}$ contains a -1 in column i. Hence, $T \in \mathbb{Z}^{2n\times n}$. Assume that $d = (d_1, \ldots, d_{2n})^\top \in \overline{\mathbb{Z}}^{2n}$ is an abstract value. Then the entries d_i and d_{n+i} provide upper and *negated* lower bounds for the possible values of the program variable $\mathbf{x}_i$. The concretization $\gamma(d)$ of d is non-empty iff the interval $[-d_{n+i}, d_i]$ is non-empty for all i, i.e., $d_i + d_{n+i} \geq 0$ for all i.

Now consider a statement $s = (T\mathbf{x} \leq c; \mathbf{x}{:=}A\mathbf{x} + b)$ and let $d \in \overline{\mathbb{Z}}^{2n}$ denote an abstract value. If $\gamma(d \wedge c) = \emptyset$, then $[\![s]\!]^\sharp(d) = (-\infty, \ldots, -\infty)^\top$. In order to deal with the other cases, we define matrices $A^+, A^- \in \mathbb{Z}^{n\times n}$ by:

$$A^+_{i\cdot j} := A_{i\cdot j} \vee 0 \qquad\qquad A^-_{i\cdot j} := -A_{i\cdot j} \vee 0 \qquad\qquad \text{for all } i, j \in \{1, \ldots, n\} \tag{62}$$

Thus, the matrix A^+ collects the positive entries of A whereas the matrix A^- collects the negated negative entries of A. The other entries are set to 0. We in particular have $A = A^+ - A^-$. Finally, we set

$$A^\pm := \begin{pmatrix} A^+ & A^- \\ A^- & A^+ \end{pmatrix}, \text{ and} \qquad\qquad b^\pm := \begin{pmatrix} b \\ -b \end{pmatrix}. \tag{63}$$

Then

$$\alpha(\{Ax + b\}) = A^\pm \begin{pmatrix} x \\ -x \end{pmatrix} + b^\pm \qquad\qquad \text{for all } x \in \mathbb{Z}^n. \tag{64}$$

We now develop an explicit representation of the best abstract transformer. That is, we verify the following equality under the assumption that $\gamma(c \wedge d) \neq \emptyset$ holds:

$$\begin{aligned}
[\![s]\!]^\sharp(d) &= \alpha([\![s]\!](\gamma(d))) \\
&= \bigvee \{\alpha(\{Ax + b\}) \mid Tx \leq c,\ x \in \gamma(d)\} \\
&= \bigvee \left\{\alpha(\{Ax + b\}) \mid \begin{pmatrix} x \\ -x \end{pmatrix} \leq c,\ \begin{pmatrix} x \\ -x \end{pmatrix} \leq d\right\} \\
&= \bigvee \left\{A^\pm \begin{pmatrix} x \\ -x \end{pmatrix} + b^\pm \mid \begin{pmatrix} x \\ -x \end{pmatrix} \leq c \wedge d\right\} && \text{(because of (64))} \\
&= A^\pm (c \wedge d) + b^\pm && \text{(because } A^\pm \text{ is monotone)}
\end{aligned}$$

The last equation holds, since the matrix $A^{\pm}$ contains non-negative entries only, which implies that the linear operator $A^{\pm}$ is monotone.

In order to practically determine the abstract semantics w.r.t. the interval domain, we decompose the constraint system for the abstract semantics into a constraint system over the integers alone. For that, we introduce unknowns $\mathbf{x}_{v,i}$ for every program point $v \in N$ and $i \in \{1, \ldots, 2n\}$. The unknown $\mathbf{x}_{v,i}$ stands for the i-th component of the vector $\mathbf{V}^\sharp[v]$. Accordingly, each unknown takes values in $\overline{\mathbb{Z}}$. Each constraint $\mathbf{V}^\sharp[v] \geq [\![s]\!]^\sharp(\mathbf{V}^\sharp[u])$ and each $i \in \{1, \ldots, 2n\}$ gives rise to the following constraint:

$$\mathbf{x}_{v,i} \geq ((\mathbf{x}_{u,1} \wedge c_{1\cdot}) + (\mathbf{x}_{u,n+1} \wedge c_{n+1\cdot})) \geq 0? \tag{65}$$

$$\cdots \tag{66}$$

$$((\mathbf{x}_{u,n} \wedge c_{n\cdot}) + (\mathbf{x}_{u,n+n} \wedge c_{n+n\cdot})) \geq 0? \tag{67}$$

$$A^{\pm}_{i\cdot 1}(\mathbf{x}_{u,1} \wedge c_{1\cdot}) + \cdots + A^{\pm}_{i\cdot 2n}(\mathbf{x}_{u,2n} \wedge c_{2n\cdot}) + b^{\pm}_{i\cdot} \tag{68}$$

The tests at the beginning ensure that the right-hand sides only evaluate to a value greater than $-\infty$ if the guard $T\mathbf{x} \leq c$ can be fulfilled.

Let $\rho^\sharp$ denote the least solution of the resulting constraint system. By construction, $V^\sharp_{i\cdot}[v] = \rho^\sharp(\mathbf{x}_{v,i})$ for all program points $v \in N$ and all $i = \{1, \ldots, 2n\}$.

The resulting constraint system is a system of constraints over the integers using binary operators $+$, multiplication with non-negative constants and the binary operator $\geq 0?$. Hence, it can be formulated as a system of *extended* integer equations (cf. Section 6). Therefore, Theorem 6 can be applied, i.e., $\vee$-strategy iteration together with the generalized Bellman-Ford algorithm is applicable to determine its least solution. Accordingly, we obtain:

Theorem 7 (Interval Analysis)**.** *The abstract semantics $V^\sharp$ of G w.r.t. the interval domain can be computed using our $\vee$-strategy improvement algorithm. The number of $\vee$-strategy improvement steps is bounded by $2n \cdot |N| \cdot \prod_{v \in N}(\max\{1, \mathsf{indeg}(v)\})^{2n}$. Each $\vee$-strategy improvement step can be performed in strongly polynomial time. More precisely: the algorithm performs at most $\mathcal{O}(|N|^2 \cdot n^3)$ arithmetic operations for each $\vee$-strategy improvement step.* □

Instead of proving Theorem 7 for intervals, we turn to a treatment of the more general domain of integer zones. Analogously as for intervals, we aim at reducing the computation of the abstract semantics over integer zones to computing the least solution of a suitable system of extended integer constraints. In presence of bounds to variable differences, however, the abstract transformers for program statements are more involved. In Subsection 7.6, we show that these can be reduced to solving (uncapacited) minimum cost flow problems. This reduction allows us to use our $\vee$-strategy improvement algorithm developed in Sections 5 and 6 to compute the abstract semantics $V^\sharp$ of a program G over integer zones efficiently. The resulting algorithm is uniform, i.e., its running-time is independent of the sizes of involved numbers, since minimum cost flow problems can be solved in strongly polynomial time [25].

7.5 Minimum cost flow problems

A *(uncapacitated) minimum cost flow problem* is a linear programming problem of the form

$$z = \inf\{c^\top x \mid x \in \mathbb{R}^n_{\geq 0},\ Ax = b\} \tag{69}$$

where the following conditions are fulfilled: (1) $c \in \mathbb{Z}^n$. (2) $b \in \mathbb{Z}^m$ with $\sum_{i=1}^m b_{i\cdot} = 0$. (3) $A \in \{-1, 0, 1\}^{m \times n}$, where each column $A_{\cdot j}$ contains exactly one -1 and exactly one 1. All other entries of $A_{\cdot j}$ are 0.

The above linear programming problem represents a directed graph (the matrix A) together with a function that assigns supplies (the $b_{i\cdot}$'s) to nodes and a function that assigns costs (the $c_{j\cdot}$'s) to edges. Each $i \in \{1, \ldots, m\}$ represents a node that supplies $b_{i\cdot}$ packages (per time unit). A negative supply is a demand. Each $j \in \{1, \ldots, n\}$ represents an edge. A transmission of one package (per time unit) over this edge induces the cost $c_{j\cdot}$ (per time unit). The source (resp. target) of the edge j is k iff $A_{k\cdot j} = 1$ (resp. $A_{k\cdot j} = -1$). Note that each edge j has exactly one source and one target.

Each $x \in \mathbb{R}^n_{\geq 0}$ represents a *flow*, where $x_{j\cdot}$ is the number of packages (per time unit) transmitted over the edge j. A flow $x \in \mathbb{R}^n_{\geq 0}$ is called a *feasible flow* iff $Ax = b$, i.e., all supplies and all demands are fulfilled. The value $c^\top x$ is the cost (per time unit) of the flow x. The minimum cost flow problem is called *infeasible* iff there does not exists a feasible flow. If this is the case, then $z = \infty$. Another corner case is that the problem is unbounded, i.e., for every $z' \in \mathbb{Z}$, there exists some feasible flow x such that $c^\top x \leq z'$. If this is the case, then the problem is called *unbounded* and $z = -\infty$. If the problem is feasible and bounded, then $z \in \mathbb{Z}$ and moreover there exists a feasible flow $x \in \mathbb{R}^n_{\geq 0}$ such that $c^\top x = z$ and all entries of x are integral, i.e., natural numbers.

Since every minimum cost flow problem is a linear programming problem, it can be solved in polynomial time through interior point methods (see e.g. Karmarkar [19], Schrijver [27], Wright [30]). However, the worst-case running-times of these algorithms depend on the sizes of the numbers occurring in the input. In this context this is called a *weak* polynomial-time algorithm. Whether or not there exist *strongly* polynomial-time algorithms for linear programming is a long outstanding question. A strongly polynomial time algorithm is an algorithm where (1) the number of arithmetic operations is bounded by a polynomial that does not depend and the sizes of the occurring numbers (i.e., the polynomial only depends on the number of nodes m and the number of edges n) and (2) each arithmetic operation can be performed in polynomial time. For the special case of minimum cost flow problems there exist several algorithms that make use of the special structure of these linear programming problems. In contrast to the general case, there indeed exist strongly polynomial algorithms [25]. The enhanced capacity scaling algorithm of Orlin [25], for instance, requires $\mathcal{O}(m \cdot \log m \cdot (n + m \cdot \log m))$ arithmetic operations. For more information regarding minimum cost flow problems, we refer to Ahuja et al. [1]. In order to simplify notations, we denote the number of arithmetic operations required for solving a minimum cost flow problem with m nodes and n edges by $\mathsf{MCF}(m, n)$.

7.6 Computing $[\![s]\!]^\sharp(d)$

In this subsection, we aim at computing $[\![s]\!]^\sharp(d)$ for a given statement s and a given abstract value $d \in \overline{\mathbb{Z}}^m$. For $k \in \{1, \ldots, m\}$, we are hence interested in computing the k-th component of $[\![T\mathbf{x} \leq c; \mathbf{x} := A\mathbf{x} + b]\!]^\sharp(d)$, i.e., we aim at computing the value

$$z := [\![T\mathbf{x} \leq c; \mathbf{x} := A\mathbf{x} + b]\!]^\sharp_{k\cdot}(d) \tag{70}$$

We get:

$$\begin{aligned} z &= \sup \{T_{k\cdot}(Ax + b) \mid x \in \mathbb{Z}^n,\ Tx \leq (c \wedge d)\} && (71)\\ &= T_{k\cdot}b + \sup \{T_{k\cdot}Ax \mid x \in \mathbb{Z}^n,\ Tx \leq (c \wedge d)\} && (72)\\ &= T_{k\cdot}b + \sup \{T_{k\cdot}Ax \mid x \in \mathbb{R}^n,\ Tx \leq (c \wedge d)\} && (73) \end{aligned}$$

The last equality holds, because the matrix T is totally unimodular (this implies that all optimal solutions of the linear programming problem (73) are integral). Observe that $z > -\infty$

iff $\{x \in \mathbb{R}^n \mid Tx \leq (c \wedge d)\} \neq \emptyset$. Whether or not $\{x \in \mathbb{R}^n \mid Tx \leq (c \wedge d)\} \neq \emptyset$ holds can be determined in strongly polynomial time (using e.g. the Floyd–Warshall algorithm (cf. e.g. Miné [22])). Therefore, we from now on assume that $z > -\infty$, i.e., the linear programming problem is feasible. Hence, the strong duality theorem for linear programming can be applied. We get

$$z = T_{k\cdot} b + \inf \{(c \wedge d)^\top y \mid y \in \mathbb{R}^m_{\geq 0},\ T^\top y = (T_{k\cdot} A)^\top\}. \tag{74}$$

Our goal is to compute z by solving a minimum cost flow problem. Each row of T and thus each column of $T^\top$ has at most two non-zero entries. These entries are distinct and in the set $\{-1, 1\}$. Therefore, each column of the matrix

$$B := \begin{pmatrix} T^\top \\ -(1, \ldots, 1)\, T^\top \end{pmatrix} \tag{75}$$

contains exactly one -1 and exactly one 1. All other entries are 0. Further, for

$$g := \begin{pmatrix} (T_{k\cdot} A)^\top \\ -(1, ..., 1)\ (T_{k\cdot} A)^\top \end{pmatrix}, \tag{76}$$

we have $\sum_{i=1}^m g_{i\cdot} = 0$. If we now replace $T^\top$ by B and $(T_{k\cdot} A)^\top$ by g, then we do not modify the feasible space of the linear programming problem, i.e., $T^\top y = (T_{k\cdot} A)^\top$ iff $By = g$ for all $y \in \mathbb{R}^m_{\geq 0}$. Therefore, we are now faced with the problem of computing

$$z = T_{k\cdot} b + \inf \{(c \wedge d)^\top y \mid y \in \mathbb{R}^m_{\geq 0},\ By = g\}. \tag{77}$$

This is a minimum cost flow problem (cf. Subsection 7.5). Note that, for some edges j, the cost $(c \wedge d)_{j\cdot}$ might be ∞. If this is the case, then we remove these edges from the minimum cost flow problem, since these edges will never be used by an optimal flow. We get:

Lemma 11. *Let $d \in \overline{\mathbb{Z}}^m$ be an abstract value and $k \in \{1, \ldots, m\}$. Then:*

1. *$[\![T\mathbf{x} \leq c; \mathbf{x} := A\mathbf{x} + b]\!]^\sharp_{k\cdot}(d)$ can be computed in strongly polynomial time through a polynomial-time reduction to a minimum cost flow problem.*

2. *There exists some finite set $M \subseteq \mathbb{N}^m$ such that*

$$[\![T\mathbf{x} \leq c; \mathbf{x} := A\mathbf{x} + b]\!]^\sharp_{k\cdot}(d) = T_{k\cdot} b + \min\{a^\top (c \wedge d) \mid a \in M\} \tag{78}$$

for all $d \in \overline{\mathbb{Z}}^m$ with $[\![T\mathbf{x} \leq c; \mathbf{x} := A\mathbf{x} + b]\!]^\sharp_{k\cdot}(d) > -\infty$.

Proof. The first statement is already shown. The second statement follows from the fact that all edges of the convex polyhedron $\{y \in \mathbb{R}^m_{\geq 0} \mid By = g\}$ (with B and g as in (77)) are from $\mathbb{N}^m$. This is a consequence of the fact that B is totally unimodular. □

Part 2 of Lemma 11 states that the operator $[\![s]\!]^\sharp_{k\cdot}$ is a conjunctive extended integer expression. Hence, we can use it within our $\vee$-strategy improvement algorithm (cf. Section 6).

7.7 Computing the program's abstract semantics through $\vee$-strategy iteration

In order to compute the abstract semantics $V^\sharp$, we construct a system $\mathcal{C}$ of inequalities as follows: For each $k \in \{1, \ldots, m\}$, we add the inequality

$$\mathbf{x}_{\mathbf{st},k} \geq \infty \tag{79}$$

These inequalities correspond to inequality (60). For each control-flow edge $(u, s, v) \in E$ and each $k \in \{1, \ldots, m\}$, we add the inequality

$$\mathbf{x}_{v,k} \geq [\![s]\!]^\sharp_{k\cdot}(\mathbf{x}_{u,1}, \ldots, \mathbf{x}_{u,m}) \tag{80}$$

These inequalities correspond to inequality (61). The Knaster-Tarski fixpoint theorem implies that $\mathcal{E} := \{\mathbf{x} = \bigvee_{\mathbf{x} \geq e \text{ is an inequality from } \mathcal{C}} e \mid \mathbf{x} \text{ is a variable of } \mathcal{C}\}$ has the same least solution as $\mathcal{C}$. By construction, we get:

Lemma 12. *$(V^\sharp[v])_{i\cdot} = (\mu[\![\mathcal{E}]\!])(\mathbf{x}_{v,i})$ for all $v \in N$ and all $i \in \{1, \ldots, m\}$.* □

Since $\mathcal{E}$ is a system of extended integer equations, we can apply our $\vee$-strategy improvement algorithm. The equation system $\mathcal{E}$ has $|N| \cdot m$ variables. Each expression $[\![s]\!]^\sharp_{k\cdot}(\mathbf{x}_{u,1}, \ldots, \mathbf{x}_{u,m})$ can be evaluated in strongly polynomial time using $\mathsf{MCF}(n+1, m)$ operations. Therefore, we finally get the following main result:

Theorem 8. *The abstract semantics $V^\sharp$ of G w.r.t. the integer zone template constraint matrix $T \in \mathbb{Z}^{m \times n}$ can be computed through our $\vee$-strategy improvement algorithm. The number of $\vee$-strategy improvement steps is bounded by $m \cdot |N| \cdot \prod_{v \in N} (\max\ \{1, \mathsf{indeg}(v)\})^m$. Each $\vee$-strategy improvement step can be performed in strongly polynomial time. More precisely: the algorithm performs at most $\mathcal{O}(|N|^2 \cdot m^2 \cdot \mathsf{MCF}(n+1, m))$ arithmetic operations for each $\vee$-strategy improvement step.* □

Example 15. *We continue Examples 12 and 13. In order to determine the abstract semantics $V^\sharp$ of the program G, we have to compute the least solution of the following system of inequalities:*

$$\mathbf{x}_{\mathbf{st},i} \geq \infty \qquad i = 1, 2, 3 \tag{81}$$

$$\mathbf{x}_{1,i} \geq [\![(x_1, x_2) := (0, 1)]\!]^\sharp_{i\cdot}(\mathbf{x}_{\mathbf{st},i}, \mathbf{x}_{\mathbf{st},2}, \mathbf{x}_{\mathbf{st},3}) \qquad i = 1, 2, 3 \tag{82}$$

$$\mathbf{x}_{1,i} \geq [\![x_1 \leq 8;\ (x_1, x_2) := (x_1 + 2, x_2 + 2)]\!]^\sharp_{i\cdot}(\mathbf{x}_{1,1}, \mathbf{x}_{1,2}, \mathbf{x}_{1,3}) \qquad i = 1, 2, 3 \tag{83}$$

For simplicity, we replace the variables $\mathbf{x}_{\mathbf{st},i}$ $(i = 1, 2, 3)$ by their values. Since

$$[\![(x_1, x_2) := (0, 1)]\!]^\sharp_{1\cdot}(\infty, \infty, \infty) = 0, \qquad [\![(x_1, x_2) := (0, 1)]\!]^\sharp_{2\cdot}(\infty, \infty, \infty) = 1, \text{ and} \tag{84}$$

$$[\![(x_1, x_2) := (0, 1)]\!]^\sharp_{3\cdot}(\infty, \infty, \infty) = 1, \tag{85}$$

we have to compute the least solution of the following system $\mathcal{E}$ of extended integer equations:

$$\mathbf{x}_{1,1} = -\infty \vee 0 \vee [\![x_1 \leq 8;\ (x_1, x_2) := (x_1 + 2, x_2 + 2)]\!]^\sharp_{1\cdot}(\mathbf{x}_{1,1}, \mathbf{x}_{1,2}, \mathbf{x}_{1,3}) \tag{86}$$

$$\mathbf{x}_{1,2} = -\infty \vee 1 \vee [\![x_1 \leq 8;\ (x_1, x_2) := (x_1 + 2, x_2 + 2)]\!]^\sharp_{2\cdot}(\mathbf{x}_{1,1}, \mathbf{x}_{1,2}, \mathbf{x}_{1,3}) \tag{87}$$

$$\mathbf{x}_{1,3} = -\infty \vee 1 \vee [\![x_1 \leq 8;\ (x_1, x_2) := (x_1 + 2, x_2 + 2)]\!]^\sharp_{3\cdot}(\mathbf{x}_{1,1}, \mathbf{x}_{1,2}, \mathbf{x}_{1,3}) \tag{88}$$

Our $\vee$-strategy improvement algorithm starts with the $\vee$-strategy σ_0 that corresponds to the following system $\mathcal{E}(\sigma_0)$ of conjunctive extended integer equations:

$$\mathbf{x}_{1,1} = -\infty \qquad \mathbf{x}_{1,2} = -\infty \qquad \mathbf{x}_{1,3} = -\infty \tag{89}$$

The first $\vee$-strategy improvement step can, for instance, result in the $\vee$-strategy σ_1 that corresponds to the following system $\mathcal{E}(\sigma_1)$ of conjunctive extended integer equations:

$$\mathbf{x}_{1,1} = 0 \qquad \mathbf{x}_{1,2} = 1 \qquad \mathbf{x}_{1,3} = 1 \tag{90}$$

The second $\vee$-strategy improvement step can, for instance, result in the $\vee$-strategy σ_2 that corresponds to the following system $\mathcal{E}(\sigma_2)$ of conjunctive extended integer equations:

$$\mathbf{x}_{1,1} = [\![x_1 \leq 8;\, (x_1, x_2) := (x_1 + 2, x_2 + 2)]\!]^\sharp_{1.}(\mathbf{x}_{1,1}, \mathbf{x}_{1,2}, \mathbf{x}_{1,3}) \tag{91}$$

$$\mathbf{x}_{1,2} = [\![x_1 \leq 8;\, (x_1, x_2) := (x_1 + 2, x_2 + 2)]\!]^\sharp_{2.}(\mathbf{x}_{1,1}, \mathbf{x}_{1,2}, \mathbf{x}_{1,3}) \tag{92}$$

$$\mathbf{x}_{1,3} = 1 \tag{93}$$

We can switch to this $\vee$-strategy, since

$$[\![x_1 \leq 8;\, (x_1, x_2) := (x_1 + 2, x_2 + 2)]\!]^\sharp_{1.}(0, 1, 1) = 2 > 0 \tag{94}$$

$$[\![x_1 \leq 8;\, (x_1, x_2) := (x_1 + 2, x_2 + 2)]\!]^\sharp_{2.}(0, 1, 1) = 3 > 1 \tag{95}$$

$$[\![x_1 \leq 8;\, (x_1, x_2) := (x_1 + 2, x_2 + 2)]\!]^\sharp_{3.}(0, 1, 1) = 1 \tag{96}$$

We now have to compute the greatest solution $\nu[\![\mathcal{E}(\sigma_2)]\!]$ of $\mathcal{E}(\sigma_2)$. From the results of Section 5 and 6, we know that this can be done by a greatest fixpoint iteration that reaches the greatest fixpoint at the latest after 3 iterations, i.e., $[\![\mathcal{E}(\sigma_2)]\!]^3(\underline{\infty})$ is the greatest fixpoint. Because of Lemma 11, we can perform each iteration in strongly polynomial time through a reduction to minimum cost flow problems. The following table illustrates the greatest fixpoint iteration:

	0	1	2	3
$\mathbf{x}_{1,1}$	∞	10	10	10
$\mathbf{x}_{1,2}$	∞	∞	11	11
$\mathbf{x}_{1,3}$	∞	1	1	1

(97)

Observe that $\nu[\![\mathcal{E}(\sigma_2)]\!] = \{\mathbf{x}_{1,1} \mapsto 10, \mathbf{x}_{1,2} \mapsto 11, \mathbf{x}_{1,3} \mapsto 1\}$ is a solution of $\mathcal{E}$ and hence the least solution of $\mathcal{E}$. Therefore, $V^\sharp[\mathbf{st}] = (\infty, \infty, \infty)$, and $V^\sharp[1] = (10, 11, 1)$. That is, 10 is an upper bound on the value for the variable x_1 at program point 1, 11 is an upper bound on the value for the variable x_2 at program point 1, and 1 is an upper bound on the difference $x_2 - x_1$ at program point 1. Observe that the obtained result cannot be established if we use intervals as the abstract domain. □

8 Conclusion

We presented a practical algorithm for computing least solutions of systems of (extended) integer equations. This algorithm is based on iteration over $\vee$-strategies. While the required number of arithmetic operations is independent of the sizes of occurring numbers, the practical complexity crucially depends on the number of strategies encountered during iteration. We indicated how this algorithm can be applied to perform precise interval analysis. We further indicated how this algorithm can also be applied to perform precise program analysis based on integer zones. For that analysis, we provided that the basic step, the application of the best abstract transformer corresponding to guards and assignments, can be implemented by reduction to a minimum cost network flow problem, for which fast polynomial algorithms exist.

Acknowledgement We thank Riko Jakob from the Technical University of Munich for the reduction to minimum cost flow problems.

References

[1] Ravindra K. Ahuja, Thomas L. Magnati, and James B. Orlin. *Network Flows*. Prentice Hall, 1993.

[2] H. Björklund, S. Sandberg, and S. Vorobyov. Optimization on completely unimodal hypercubes. Technichal report 2002-18, Department of Information Technology, Uppsala University, 2002.

[3] Henrik Bjorklund, Sven Sandberg, and Sergei 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.

[4] Baudouin Le Charlier and Pascal Van Hentenryck. A Universal Top-Down Fixpoint Algorithm. Technical Report CS-92-25, Brown University, Providence, RI 02912, 1992.

[5] Baudouin Le Charlier and Pascal Van Hentenryck. Experimental Evaluation of a Generic Abstract Interpretation Algorithm for Prolog. *ACM Transactions of Programming Languages and Systems (TOPLAS)*, 16(1):35–101, 1994.

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. The MIT Press and McGraw-Hill Book Company, 2001. ISBN 0-262-03293-7, 0-07-013151-1.

[7] Alexandru Costan, Stephane Gaubert, Eric Goubault, Matthieu Martel, and Sylvie 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.

[8] P. Cousot and R. Cousot. Systematic Design of Program Analysis Frameworks. In *6th ACM Symp. on Principles of Programming Languages (POPL)*, pages 238–352, 1979.

[9] Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL*, pages 238–252, 1977.

[10] Patrick Cousot and Radhia Cousot. Comparison of the Galois Connection and Widening/-Narrowing Approaches to Abstract Interpretation. JTASPEFL '91, Bordeaux. *BIGRE*, 74: 107–110, October 1991.

[11] N. Dor, M. Rodeh, and M. Sagiv. Cleanness Checking of String Manipulations in C Programs via Integer Analysis. In *8th Int. Static Analysis Symposium (SAS'01)*, pages 194–212. LNCS 2126, Springer Verlag, 2001.

[12] C. Fecht and H. Seidl. A Faster Solver for General Systems of Equations. *Science of Computer Programming (SCP)*, 35(2):137–161, 1999.

[13] Stephane Gaubert, Eric Goubault, Ankur Taly, and Sarah Zennou. Static analysis by policy iteration on relational domains. In Nicola [24], pages 237–252. ISBN 978-3-540-71314-2.

[14] Thomas Gawlitza and Helmut Seidl. Precise relational invariants through strategy iteration. In Jacques Duparc and Thomas A. Henzinger, editors, *CSL*, volume 4646 of *Lecture Notes in Computer Science*, pages 23–40. Springer, 2007. ISBN 978-3-540-74914-1.

[15] Thomas Gawlitza and Helmut Seidl. Precise fixpoint computation through strategy iteration. In Nicola [24], pages 300–315. ISBN 978-3-540-71314-2.

[16] Thomas Gawlitza and Helmut Seidl. Precise interval analysis vs. parity games. In Jorge Cuéllar, T. S. E. Maibaum, and Kaisa Sere, editors, *FM*, volume 5014 of *Lecture Notes in Computer Science*, pages 342–357. Springer, 2008. ISBN 978-3-540-68235-6.

[17] A.J. Hoffman and R.M. Karp. On Nonterminating Stochastic Games. *Management Sci.*, 12:359–370, 1966.

[18] R. Howard. *Dynamic Programming and Markov Processes*. Wiley, New York, 1960.

[19] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–396, 1984.

[20] Gary A. Kildall. A unified approach to global program optimization. In *POPL*, pages 194–206, 1973.

[21] Kim Guldstrand Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. Efficient verification of real-time systems: compact data structure and state-space reduction. In *IEEE Real-Time Systems Symposium*, pages 14–24. IEEE Computer Society, 1997.

[22] Antoine Miné. A new numerical abstract domain based on difference-bound matrices. In Olivier Danvy and Andrzej Filinski, editors, *PADO*, volume 2053 of *Lecture Notes in Computer Science*, pages 155–172. Springer, 2001. ISBN 3-540-42068-1.

[23] Antoine Miné. The octagon abstract domain. In *WCRE*, pages 310–, 2001.

[24] Rocco De Nicola, editor. *Programming Languages and Systems, 16th European Symposium on Programming, ESOP 2007, Held as Part of the Joint European Conferences on Theory and Practics of Software, ETAPS 2007, Braga, Portugal, March 24 - April 1, 2007, Proceedings*, volume 4421 of *Lecture Notes in Computer Science*, 2007. Springer. ISBN 978-3-540-71314-2.

[25] James B. Orlin. A faster strongly polynominal minimum cost flow algorithm. In *STOC*, pages 377–387. ACM, 1988.

[26] Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna. Scalable analysis of linear systems using mathematical programming. In Radhia Cousot, editor, *VMCAI*, volume 3385 of *Lecture Notes in Computer Science*, pages 25–41. Springer, 2005. ISBN 3-540-24297-X.

[27] Alexander Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1986.

[28] Alfred Tarski. A lattice-theoretical fixpoint theorem and its appications. *Pac. J. Math.*, 5:285–309, 1955.

[29] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David B. Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, Frank Mueller, Isabelle Puaut, Peter P. Puschner, Jan Staschulat, and Per Stenström. The worst-case execution-time problem - overview of methods and survey of tools. *ACM Trans. Embedded Comput. Syst.*, 7(3), 2008.

[30] Stephen J. Wright. *Primal-Dual Interior-Point Methods*. SIAM, Philadelphia, PA, U.S.A., 1997.

[31] Sergio Yovine. Model checking timed automata. In Grzegorz Rozenberg and Frits W. Vaandrager, editors, *European Educational Forum: School on Embedded Systems*, volume 1494 of *Lecture Notes in Computer Science*, pages 114–152. Springer, 1996. ISBN 3-540-65193-4.

A Additional Lemmata

Lemma 13. *Let $f : (X \to \mathbb{Z}) \to \mathbb{Z}$ be a BF-function. For all $\rho : X \to \overline{\mathbb{Z}}$ and all $X' \subseteq X$ we have: $f(\rho \oplus \{x \mapsto \infty \mid x \in X'\}) > f(\rho)$ implies $f(\rho \oplus \{x \mapsto \infty \mid x \in X'\}) = \infty$.* □

B Omitted Proofs: Section 4

of Lemma 2. First of all we consider the operator $\vee$. Let $x = (x_1, x_2)$, $y = (y_1, y_2)$, $y \geq x$, and $y_1 \vee y_2 > x_1 \vee x_2$. We assume w.l.o.g. that $y_1 \geq y_2$ holds. Let $\delta := 0$. Then, we get (a) $y_1 > x_1$, (b) $y_1 \vee y_2 = y_1 + \delta$, and (c) $z_1 \vee z_2 \geq z_1 + \delta$ for all $(z_1, z_2) \geq y$.

We now consider an arbitrary monotone and upward-expansive operator $f : (X \to \overline{\mathbb{Z}}) \to \overline{\mathbb{Z}}$. Let $\rho' \geq \rho$, $f(\rho') > f(\rho)$, $x \in X$ with $\rho'(x) > \rho(x)$, and $\delta := f(\rho') - \rho'(x)$. Here, we assume that $\rho'(x) < \infty$ (otherwise it is trivial, since $\rho'(x) = \infty$ implies $f(\rho') = \infty$). Requirement (a) and (b) hold be definition. In order to prove requirement (c), let $\rho'' \geq \rho'$. We get:

$$\begin{aligned} f(\rho'') &\geq f(\rho' \oplus \{x \mapsto \rho''(x)\}) && \text{(Monotonicity)} \\ &\geq f(\rho') + \rho''(x) - \rho'(x) && \text{(Upward-expansivity)} \\ &= \rho''(x) + \delta \end{aligned}$$

This completes the proof. □

of Lemma 3. Let $e_{\mathbf{x}}$ denote the right hand-side of the equation for $\mathbf{x}$ in $\mathcal{E}$, i.e., $\mathbf{x} = e_{\mathbf{x}} \in \mathcal{E}$ holds for all $\mathbf{x} \in \mathbf{X}$. Within this proof, for some $i \geq 2$ and some variable $\mathbf{x}' \in \mathbf{X}$, a variable $\mathbf{x}$ is called *relevant for* $\rho^{(i)}(\mathbf{x}') > \rho^{(i-1)}(\mathbf{x}')$ iff $\mathbf{x}$ is relevant for $[\![e_{\mathbf{x}'}]\!]\rho^{(i-1)} > [\![e_{\mathbf{x}'}]\!]\rho^{(i-2)}$. Since $[\![e_{\mathbf{x}'}]\!]$ is a BF-function, there always exists some variable $\mathbf{x}$ that is relevant for $\rho^{(i)}(\mathbf{x}') > \rho^{(i-1)}(\mathbf{x}')$.

Note that $\rho^{(i)} \leq \rho^* := \mu[\![\mathcal{E}]\!]$ for all $i \in \mathbb{N}$, since $[\![\mathcal{E}]\!]$ is monotone. Let $\mathbf{x} \in \mathbf{X}$ and $k > n$ with $\rho^{(k)}(\mathbf{x}) > \rho^{(n)}(\mathbf{x})$. Furthermore, assume w.l.o.g. that $\rho^{(k)}(\mathbf{x}) > \rho^{(k-1)}(\mathbf{x})$. If $\rho^{(k)}(\mathbf{x}) = \infty$, then we directly obtain $\rho^*(\mathbf{x}) = \infty$. Therefore, we assume $\rho^{(k)}(\mathbf{x}) < \infty$.

Auxiliary Lemma 1. *There exist variables $\mathbf{x}_1, \ldots, \mathbf{x}_k \in \mathbf{X}$ such that the following conditions are fulfilled:*

1. $\mathbf{x}_k = \mathbf{x}$.
2. $\rho^{(i)}(\mathbf{x}_i) > \rho^{(i-1)}(\mathbf{x}_i)$ *for all* $i \in \{1, \ldots, k\}$.
3. $\mathbf{x}_{i-1}$ *is relevant for* $\rho^{(i)}(\mathbf{x}_i) > \rho^{(i-1)}(\mathbf{x}_i)$ *for all* $i \in \{2, \ldots, k\}$.

Proof. The statement can be shown by backward induction using the fact that $\mathcal{E}$ is a system of BF-equations. □

Auxiliary Lemma 2. *Let $i_1, i_2 \in \{1, \ldots, k\}$ with $i_1 \leq i_2$ and $\delta := \rho^{(i_2)}(\mathbf{x}_{i_2}) - \rho^{(i_1)}(\mathbf{x}_{i_1})$. Then $([\![\mathcal{E}]\!]^{i_2 - i_1}\rho)(\mathbf{x}_{i_2}) \geq \rho(\mathbf{x}_{i_1}) + \delta$ for all $\rho \geq \rho^{(i_1)}$.*

Proof. Let $\rho \geq \rho^{(i_1)}$. We do induction on $i_2 - i_1$. The statement is obviously fulfilled for $i_2 - i_1 = 0$. Therefore, assume $i_2 - i_1 > 0$. From the induction hypothesis we get $([\![\mathcal{E}]\!]^{i_2 - 1 - i_1}\rho)(\mathbf{x}_{i_2 - 1}) \geq \rho(\mathbf{x}_{i_1}) + \delta'$, where $\delta' := \rho^{(i_2 - 1)}(\mathbf{x}_{i_2 - 1}) - \rho^{(i_1)}(\mathbf{x}_{i_1})$. We have $\rho^{(i_2)}(\mathbf{x}_{i_2}) = [\![e_{\mathbf{x}_{i_2}}]\!]\rho^{(i_2 - 1)}$ and, by Auxiliary Lemma 1, $\mathbf{x}_{i_2 - 1}$ is relevant for $\rho^{(i_2)}(\mathbf{x}_{i_2}) > \rho^{(i_2 - 1)}(\mathbf{x}_{i_2})$. Let $\delta'' := \rho^{(i_2)}(\mathbf{x}_{i_2}) - \rho^{(i_2 - 1)}(\mathbf{x}_{i_2 - 1})$ and $\delta := \delta' + \delta'' = \rho^{(i_2)}(\mathbf{x}_{i_2}) - \rho^{(i_1)}(\mathbf{x}_{i_1})$. Thus, we get:

$$([\![\mathcal{E}]\!]^{i_2 - i_1}\rho)(\mathbf{x}_{i_2}) = ([\![\mathcal{E}]\!]([\![\mathcal{E}]\!]^{i_2 - 1 - i_1}\rho))(\mathbf{x}_{i_2}) = [\![e_{\mathbf{x}_{i_2}}]\!]([\![\mathcal{E}]\!]^{i_2 - 1 - i_1}\rho)$$

$$\geq (\llbracket\mathcal{E}\rrbracket^{i_2-1-i_1}\rho)(\mathbf{x}_{i_2-1}) + \delta'' \geq \rho(\mathbf{x}_{i_1}) + \delta' + \delta'' = \rho(\mathbf{x}_{i_1}) + \delta$$

This completes the proof of auxiliary lemma 2. □

Since $k > n$, by the pigeon-hole principle, there exist $k_1, k_2 \in \mathbb{N}$ with $1 \leq k_1 < k_2 \leq k$ such that $\mathbf{y} := \mathbf{x}_{k_2} = \mathbf{x}_{k_1}$. Let $\delta := \rho^{(k_2)}(\mathbf{y}) - \rho^{(k_1)}(\mathbf{y})$. Since $\rho^{(k_2)}(\mathbf{y}) > \rho^{(k_1)}(\mathbf{y})$, we have $\delta = \rho^{(k_2)}(\mathbf{y}) - \rho^{(k_1)}(\mathbf{y}) > 0$.

Next, we show $\rho^*(\mathbf{y}) = \infty$. For the sake of contradiction assume $\rho^*(\mathbf{y}) < \infty$. Since $\rho^* \geq \rho^{(k_1)}$ holds, $(\llbracket\mathcal{E}\rrbracket^{k_2-k_1}\rho^*)(\mathbf{y}) \geq \rho^*(\mathbf{y}) + \delta > \rho^*(\mathbf{y})$ holds by Auxiliary Lemma 2 — contradiction to $\rho^* \in \mathbf{Sol}(\mathcal{E})$. Thus, we have $\rho^*(\mathbf{y}) = \infty$. It remains to show $\rho^*(\mathbf{x}) = \infty$. Since $\rho^* \in \mathbf{Sol}(\mathcal{E})$, we have $\rho^*(\mathbf{x}) = (\llbracket\mathcal{E}\rrbracket^{k-k_2}\rho^*)(\mathbf{x}) \geq \rho^*(\mathbf{y}) + (\rho^{(k)}(\mathbf{x}) - \rho^{(k_2)}(\mathbf{y})) = \infty$, because of Auxiliary Lemma 2. □

of Theorem 2. Let ρ_0 be the value of the program variable ρ after the execution of the first for-loop, i.e., we have $\rho_0 = \llbracket\mathcal{E}\rrbracket^n(\underline{-\infty})$. We denote the value of the program variable ρ immediate before the execution of the second for-loop by ρ_1. For $i = 2, \ldots, n$, we denote the value of the program variable ρ after the $(i-1)$-th execution of the loop-body of the second for-loop by ρ_i.

Let $\rho^* \in \mathbf{PostSol}(\mathcal{E})$. Since $\llbracket\mathcal{E}\rrbracket$ is monotone, we have $\rho_0 \leq \rho^*$. Firstly, we show that $\rho_i \leq \rho^*$ holds for all $i = 1, \ldots, n$. By Lemma 3, $\rho^*(\mathbf{x}) = \infty$, whenever $(\llbracket\mathcal{E}\rrbracket\rho_0)(\mathbf{x}) > \rho_0(\mathbf{x})$. Hence, we have $\rho_1 \leq \rho^*$, i.e., the statement holds for $i = 1$. Assume that the statement holds for $i \in \{1, \ldots, n-1\}$, i.e., $\rho_i \leq \rho^*$. Since $\llbracket\mathcal{E}\rrbracket$ is monotone, we have $\rho_{i+1} = \rho_i \vee \llbracket\mathcal{E}\rrbracket\rho_i \leq \rho_i \vee \llbracket\mathcal{E}\rrbracket\rho^* \leq \rho^*$. Thus, we have shown that $\rho_i \leq \rho^*$ for all $i = 1, \ldots, n$.

Finally, we show that $\rho_n \in \mathbf{PostSol}(\mathcal{E})$. For $i = 0, \ldots, n$, let $\mathbf{X}_i^\infty := \{\mathbf{x} \in \mathbf{X} \mid \rho_i(\mathbf{x}) = \infty\}$. By construction we have $\rho_1 = \rho_0 \oplus \{\mathbf{x} \mapsto \infty \mid \mathbf{X}_1^\infty \setminus \mathbf{X}_0^\infty\}$. By induction we show that $\rho_i = \rho_{i-1} \oplus \{\mathbf{x} \mapsto \infty \mid \mathbf{x} \in \mathbf{X}_i^\infty \setminus \mathbf{X}_{i-1}^\infty\}$ for all $i \in \{2, \ldots, n\}$. Because of Lemma 13, the following holds for $i = 2$ and every equations $\mathbf{x} = e \in \mathcal{E}$:

$$\rho_i(\mathbf{x}) = \rho_2(\mathbf{x}) = \rho_1(\mathbf{x}) \vee \llbracket e\rrbracket\rho_1 = \rho_1(\mathbf{x}) \vee \begin{cases} \infty & \text{if } \llbracket e\rrbracket\rho_1 > \llbracket e\rrbracket\rho_0 \\ \llbracket e\rrbracket\rho_0 & \text{if } \llbracket e\rrbracket\rho_1 \leq \llbracket e\rrbracket\rho_0 \end{cases}$$
$$= \begin{cases} \infty & \text{if } \llbracket e\rrbracket\rho_1 > \llbracket e\rrbracket\rho_0 \\ \rho_1(\mathbf{x}) & \text{if } \llbracket e\rrbracket\rho_1 \leq \llbracket e\rrbracket\rho_0 \end{cases}$$

Let $i > 2$ and assume that the statement holds for $i-1$. Since $\llbracket e\rrbracket$ is a BF-function, the following holds for every equation $\mathbf{x} = e \in \mathcal{E}$:

$$\rho_i(\mathbf{x}) = \rho_{i-1}(\mathbf{x}) \vee \llbracket e\rrbracket\rho_{i-1} = \rho_{i-1}(\mathbf{x}) \vee \begin{cases} \infty & \text{if } \llbracket e\rrbracket\rho_{i-1} > \llbracket e\rrbracket\rho_{i-2} \\ \llbracket e\rrbracket\rho_{i-2} & \text{if } \llbracket e\rrbracket\rho_{i-1} \leq \llbracket e\rrbracket\rho_{i-2} \end{cases}$$
$$= \begin{cases} \infty & \text{if } \llbracket e\rrbracket\rho_{i-1} > \llbracket e\rrbracket\rho_{i-2} \\ \rho_{i-1}(\mathbf{x}) & \text{if } \llbracket e\rrbracket\rho_{i-1} \leq \llbracket e\rrbracket\rho_{i-2} \end{cases}$$

Thus, $\rho_i = \rho_{i-1} \oplus \{\mathbf{x} \mapsto \infty \mid \mathbf{x} \in \mathbf{X}_i^\infty \setminus \mathbf{X}_{i-1}^\infty\}$ for all $i \in \{1, \ldots, n\}$. Thus, for all $i = 1, \ldots, n$, either $\mathbf{X}_i^\infty \supset \mathbf{X}_{i-1}^\infty$ or $\rho_i = \rho_{i-1}$. If there exists some $i = 1, \ldots, n-1$ with $\rho_i = \rho_{i-1}$, then the statement is proven, since then $\llbracket\mathcal{E}\rrbracket\rho_{i-1} \leq \rho_{i-1}$ and thus $\rho_n = \rho_{i-1} \in \mathbf{PostSol}(\mathcal{E})$. Otherwise, i.e., if $\rho_i > \rho_{i-1}$ for all $i \in \{1, \ldots, n\}$, then $\mathbf{X}_i^\infty \supset \mathbf{X}_{i-1}^\infty$ for all $i \in \{1, \ldots, n\}$. Therefore, we get $\rho_n = \underline{\infty}$. Thus, we have shown, that $\rho_n \in \mathbf{PostSol}(\mathcal{E})$.

We have $\rho^* \geq \rho_n \in \mathbf{PostSol}(\mathcal{E})$ for all $\rho^* \in \mathbf{PostSol}(\mathcal{E})$, i.e., ρ_n is the least post-solution. Thus, ρ_n is also the least solution of $\mathcal{E}$. □

C Omitted Proofs: Section 5

of Lemma 6. We only have to consider the case that $\mathcal{E}$ is a system of basic integer equations. Thus, let $\mathcal{E}$ be a system of basic integer equations. For a pre-solution ρ'' of $\mathcal{E}$, let $M_{\rho''}$ be the smallest set of of equations of the form $\mathbf{x} = e$ with the following properties:

1. If $\mathbf{x} = e \in \mathcal{E}$ with $\rho''(\mathbf{x}) < [\![e]\!]\rho'' < \infty$, then $\mathbf{x} = e \in M_{\rho''}$.
2. If $(\mathbf{x} = e[\mathbf{x}'],\ \mathbf{x}' = e') \in M_{\rho''} \times M_{\rho''} \cup M_{\rho''} \times \mathcal{D}_{\rho''}(\mathcal{E}) \cup \mathcal{D}_{\rho''}(\mathcal{E}) \times M_{\rho''}$, then $\mathbf{x} = e[e'] \in M_{\rho''}$.

Because of expansivity $\rho''(\mathbf{x}) < [\![e]\!]\rho'' < \infty$ holds for all $\mathbf{x} = e \in M_{\rho''}$ and all pre-solutions ρ'' of $\mathcal{E}$. Furthermore, $M_\rho \cup \mathcal{D}_\rho(\mathcal{E}) \supseteq M_{\rho'} \cup \mathcal{D}_{\rho'}(\mathcal{E})$, because of the feasibility of ρ.

We show that ρ' is feasible. For the sake of contradiction we assume that ρ' is not feasible. Thus, there exists some $\mathbf{x} = e \in \mathcal{D}_{\rho'}(\mathcal{E})$ with $\mathbf{x} \in \mathbf{Vars}(e)$. By definition, we have $-\infty < \rho'(\mathbf{x}) = [\![e]\!]\rho' < \infty$. Since $[\![e]\!]$ is monotone and by Lemma 1 expansive, we have $\rho(\mathbf{x}) \geq [\![e]\!](\rho' \oplus \{\mathbf{x} \mapsto \rho(\mathbf{x})\}) \geq [\![e]\!]\rho$. Since $\mathbf{x} = e \in \mathcal{D}_{\rho'}(\mathcal{E}) \subseteq M_\rho \cup \mathcal{D}_\rho(\mathcal{E})$, we get $\rho(\mathbf{x}) \leq [\![e]\!]\rho$. We get $\rho(\mathbf{x}) = [\![e]\!]\rho$. Therefore, $\mathbf{x} = e \in \mathcal{D}_\rho(\mathcal{E})$. This contradicts the assumption that ρ is feasible. □

of Lemma 8. For the sake of contradiction assume that ρ is not $\mathcal{E}(\sigma')$-feasible. Hence, there exists a $\pi' \in \Pi_{\mathcal{E}(\sigma')}$ such that ρ is not $\mathcal{E}(\sigma')(\pi')$-feasible. Thus, there exists an equation $\bar{\mathbf{x}} = \bar{e} \in \mathcal{D}_\rho(\mathcal{E}(\sigma')(\pi'))$ with $\bar{\mathbf{x}} \in \mathbf{Vars}(\bar{e})$. For all expressions $e \in \mathcal{S}(\mathcal{E})$, the following holds: If $\infty > [\![e\sigma'\pi']\!]\rho = [\![e\sigma]\!]\rho > -\infty$ holds, then there exists a $\wedge$-strategy $\pi \in \Pi_{\mathcal{E}(\sigma)}$, such that $e\sigma'\pi' = e\sigma\pi$ holds. This holds, because, by Lemma 1, $[\![e]\!]$ is expansive in $\mathbf{Vars}(e)$ for every basic integer expression and because σ' is an improvement of σ w.r.t. ρ and thus we have $[\![e\sigma']\!]\rho > [\![e\sigma]\!]\rho$ for all $e \in \mathcal{S}_\vee(\mathcal{E})$ with $\sigma'(e) \neq \sigma(e)$. From this it in particular follows the validity of the following statement: If $\infty > [\![e\sigma'\pi']\!]\rho = [\![e\sigma]\!]\rho = \rho(\mathbf{x}) > -\infty$ for an equation $\mathbf{x} = e \in \mathcal{E}$, then there exists a $\wedge$-strategy π for $\mathcal{E}(\sigma)$ such that $e\sigma'\pi' = e\sigma\pi$. Thus, we can construct a $\wedge$-strategy π for $\mathcal{E}(\sigma)$ such that $e\sigma'\pi' = e\sigma\pi$ for all equations $\mathbf{x} = e \in \mathcal{E}$ with $\infty > [\![e\sigma'\pi']\!]\rho = \rho(\mathbf{x}) > -\infty$. Let π be such a $\wedge$-strategy for $\mathcal{E}(\sigma)$. We in particular also get $\bar{\mathbf{x}} = \bar{e} \in \mathcal{D}_\rho(\mathcal{E}(\sigma)(\pi))$. This is a contradiction, since ρ is $\mathcal{E}(\sigma)$-feasible and thus also $\mathcal{E}(\sigma)(\pi)$-feasible. □

of Theorem 5. We show that at most $|\mathbf{X}| \cdot |\Sigma|$ $\vee$-strategy improvement steps are preformed. As input we have the system $\mathcal{E} \vee -\infty$, a $\vee$-strategy σ_{init} with $\sigma_{\text{init}}(e \vee -\infty) = -\infty$ for all $\mathbf{x} = e \in \mathcal{E}$ and the variable assignment $\rho_{\text{init}} = \underline{-\infty}$. For $i \in \mathbb{N}$, let σ_i and ρ_i be the values of the program variables σ and ρ after the i-th execution of the loop-body, respectively.

For the sake of contradiction assume that $\rho_{|\mathbf{X}| \cdot |\Sigma|} \neq \mu[\![\mathcal{E}]\!]$ holds. Thus, we have $\rho_0 < \cdots < \rho_{|\mathbf{X}| \cdot |\Sigma| + 1}$. From $\sigma_i(e \vee -\infty) = e$ we get $\sigma_j(e \vee -\infty) = e$ for all $j \geq i \in \mathbb{N}$. Hence, at most $|\mathbf{X}| \cdot |\Sigma|$ different $\vee$-strategies occur in the sequence $\sigma_1, \ldots, \sigma_{|\mathbf{X}| \cdot |\Sigma| + 1}$. Because of the pigeonhole principle, there exists a $\vee$-strategy σ that occurs twice in the sequence $\sigma_1, \ldots, \sigma_{|\mathbf{X}| \cdot |\Sigma| + 1}$, i.e., there exist $i_1, i_2 \in \{1, \ldots, |\mathbf{X}| \cdot |\Sigma| + 1\}$ with $i_1 < i_2$ and $\sigma_{i_1} = \sigma_{i_2}$. Thus, we have $\rho_{i_1} = \nu[\![\mathcal{E}(\sigma_{i_1})]\!] = \nu[\![\mathcal{E}(\sigma_{i_2})]\!] = \rho_{i_2}$. This contradicts the fact that $\rho_{i_1} < \rho_{i_2}$ holds. □