Abstract Interpretation Meets Convex Optimization *

Thomas Martin Gawlitza[1], Helmut Seidl[2], Assalé Adjé[3], Stephane Gaubert[4], and Eric Goubault[5]

[1] CNRS/VERIMAG, France Thomas.Gawlitza@imag.fr
[2] Technische Universität München, Germany seidl@in.tum.de
[3] CEA, LIST and LIX, Ecole Polytechnique (MeASI) Assale.Adje@cea.fr
[4] INRIA Saclay and CMAP, Ecole Polytechnique, F-91128 Palaiseau Cedex, France Stephane.Gaubert@inria.fr **
[5] CEA, LIST (MeASI), F-91191 Gif-sur-Yvette Cedex, France Eric.Goubault@cea.fr

Abstract. Numerical static program analyses by abstract interpretation, e.g., the problem of inferring bounds for the values of numerical program variables, are faced with the problem that the abstract domains often contain infinite ascending chains. In oder to nevertheless enforce termination one traditionally applies a widening/narrowing approach that buys the guarantee for termination for loss of precision. However, recently, several interesting alternative approaches for computing numerical invariants by abstract interpretation were developed that aim at higher precision. One interesting research direction in this context is the study of *strategy improvement algorithms*. Such algorithms are successfully applied for solving two-players zero-sum games. In the present paper we discuss and compare max-strategy and min-strategy improvement algorithms that in particular can be utilized for computing numerical invariants by abstract interpretation. Our goal is to provide the intuitions behind these approaches by focussing on a particular application, namely template-based numerical analysis.

1 Introduction

Mathematical optimization aims at finding a value within an area of *feasible* values which maximizes (resp. minimizes) a given objective function. Quite efficient techniques have been developed for particular cases that are important in practice, e.g., when the objective function is linear and the area of feasible values a convex polytope (*linear programming*, see e.g. Schrijver [21]) or even an intersection of a convex polytope with the positive semi-definite cone (*semi-definite programming*, see e.g. Todd [22]) or a convex set that is defined through convex constraints (*convex optimization*, see e.g. Boyd and Vandenberghe [5], Nemirovski [16]). In a certain sense, also numerical static program analysis based on abstract interpretation can often be cast as an optimization problem as follows: Assume that we are given a complete lattice of potential program invariants at program points, i.e., an abstract domain. Then, the abstract semantics of each control-flow edge from a program point u to a program point v induces constraints

* This work was partially funded by the ANR project ASOPT.
** VERIMAG is a joint laboratory of CNRS, Université Joseph Fourier and Grenoble INP.

on the invariants for u and v. These constraints describe the feasible area. The objective of the analysis is to minimize all invariants for the program points.

In general, it is not clear how this insight may lead to better algorithms. In this paper, however, we show that in case of *template-based* analysis of relational numerical properties, techniques from mathematical optimization allow to construct novel program analysis algorithms. The *templates* we consider are (multivariate) polynomials in the program variables such as $2x_1^2 + 3x_2^2 + 2x_1x_2$, where x_1 an x_2 are program variables. The goal of the analysis is to determine, for every program point v, a safe upper bound to each template when reaching that program point v. In order to be as precise as possible, this upper bound should be as small as possible. Different templates may serve different purposes. If the analysis is only meant to infer (decently small) *intervals* for the values of the program variables $x_1, \ldots, x_n$, templates of the form x_i and $-x_i$ suffice. If the analysis additionally should infer bounds on the *differences* between certain variables, templates of the form $x_i - x_j$ should be used.

Templates consisting of arbitrary *linear* combinations have been introduced and studied by Sankaranarayanan et al. [19]. In some cases, e.g., when trying to prove that certain linear filters do not lead to floating-point overflows, linear templates are not sufficient (see e.g. Feron and Alegre [8]). However, these cases can be treated by using *quadratic* templates (see e.g. Adjé et al. [2], Feron and Alegre [7]).

Instead of directly performing the template based analysis, we reduce the analysis problem to the problem of computing least solutions for systems of in-equations of the form

$$\mathbf{x}_i \geq f(\mathbf{x}_1, \ldots, \mathbf{x}_n) \tag{1}$$

where the unknowns of the system now may take values in $\overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty, \infty\}$ and the right-hand sides f are *monotone and concave* operators on $\overline{\mathbb{R}}$. The unknowns are the upper bounds to the templates at the different program points. For the problem of solving system (1), we present two *strategy improvement approaches*:

The Min-Strategy Iteration Approach The min-strategy iteration approach as advocated by Adjé et al. [2] is, for the particular case we are studying in the present paper, similar to Newton's method. It starts with some solution of (1) and constructs a decreasing sequence of solutions. For any solution x, the next solution is constructed by over-approximating each concave right-hand side f in (1) with a linear function $T_{f,x}$ satisfying $T_{f,x}(x) = f(x)$. The improved next solution x' of (1) then is obtained as the least solution of the resulting linear constraint system

$$\mathbf{x}_i \geq T_{f,x}(\mathbf{x}_1, \ldots, \mathbf{x}_n) \tag{2}$$

which can be computed by means of linear programming. The crucial step here is to determine, for a monotone and concave function f and a given vector x, the linear function $T_{f,x}$ with $T_{f,x}(x) = f(x)$. If f is a point-wise minimum of finitely many affine functions, then $T_{f,x}$ can be chosen as one of the affine functions occurring in the minimum. In this paper, we are in particular interested in the more challenging cases where f is not simply the point-wise minimum of finitely many affine functions. In the cases of interest f is defined by means of a *parametrized*

convex optimization problem. Then, $T_{f,x}$ can be computed by solving a convex optimization problem that basically consists of the *duals* of the parametrized convex optimization problems. If the parametrized convex optimization problem that describes f is a linear programming problem, then we can use linear programming for determining $T_{f,x}$. If the parametrized convex optimization problem that describes f is a semi-definite programming problem, then we can use semi-definite programming for determining $T_{f,x}$.

The Max-Strategy Iteration Approach The max-strategy iteration approach of Gawlitza and Seidl [13] considers the constraint system (1) as a system of equations

$$\mathbf{x}_i = f_{i1}(\mathbf{x}_1, \ldots, \mathbf{x}_n) \vee \ldots \vee f_{im_i}(\mathbf{x}_1, \ldots, \mathbf{x}_n) \ , \ i = 1, \ldots, n \tag{3}$$

where the right-hand side of each unknown $\mathbf{x}_i$ is the finite maximum of concave functions f_{ij} (here, $\vee$ denotes the maximum operator, i.e., $x \vee y = \max\{x, y\}$). In order to compute the least solution of (3), Gawlitza and Seidl suggest *max-strategy iteration*. A max-strategy can be considered as a function that selects, for each unknown $\mathbf{x}_i$, one monotone and concave f_{ij} from the right-hand side of $\mathbf{x}_i$. Gawlitza and Seidl present an algorithm which *precisely* computes the least solution of (1) by iterating over max-strategies. In order to evaluate and improve a max-strategy, this algorithm requires a black box algorithm for computing *greatest finite solutions* of systems of in-equations of the form $\mathbf{x}_i \leq f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ where f is monotone and concave. How this black box algorithm can be realized depends on the class of operators the operators occurring in the right-hand sides of (1) are from.

If the operators occurring in (1) can be implemented through *parametrized* linear programs, the whole max-strategy improvement step can be implemented by linear programming. Likewise, if the operators can be implemented through parametrized semi-definite programs, the max-strategy improvement step can be implemented by semi-definite programming.

In the linear case, the above strategy iteration techniques can be applied to compute invariants for template domains where all templates are linear combinations of program variables. In the simple case of intervals, these approaches allow to perform interval analysis without widening [6, 11]. In case of more complex linear combinations, arbitrary *template polyhedra domains* [19] such as the *octagon domain* [15] can be handled [9, 10]. For quadratic templates, the above strategy iteration approaches can be utilized for computing (resp. approximating) a *semi-definite relaxation* of the abstract semantics (cf. Adjé et al. [2], Gawlitza and Seidl [13]).

The present paper is structured as follows: In Section 2 we discuss a simple example, where one is not able to infer non-trivial invariants through an analysis that is based on liner templates. However, non-trivial invariants can be obtained with quadratic templates and a semi-definite relaxation of the resulting abstract semantics. In Section 3 we discuss how we should relax an abstract semantics such that the resulting *relaxed semantic equations* fit in our framework, i.e., can be translated into a system of in-equations of the form $\mathbf{x} \geq f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, where f is a monotone and concave operator on $\overline{\mathbb{R}}$. After introducing some notations in Section 4, we explain the min-strategy approach and the max-strategy approach in Sections 5 and 6, respectively. Here, we don't

aim at completeness. Instead, we omit most of the things that are not directly connected to our application. Section 7 is dedicated for a comparison of the two approaches and a conclusion.

2 Motivation and Running Example

In this subsection we are going to have a look at a small example: the harmonic oscillator example of Adjé et al. [2]. The program consists only of the following simple loop:

```
float x_1, x_2, tmp;
x_1 = ran();
x_2 = ran();
while ( TRUE ) {
  printf("%f, %f\n", x_1, x_2);
  tmp = 1. * x_1 + 0.01 * x_2;
  x_2 = -0.01 * x_1 + 0.99 * x_2;
  x_1 = tmp;
}
```

Here, we assume that ran() returns a random float value between 0 and 1, where 0 and 1 are included. Figure 1 shows the control-flow graph of the program. The program

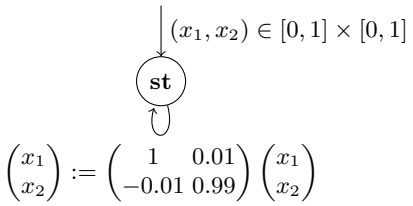
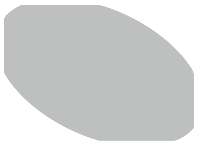


Fig. 1. The Harmonic Oscillator

implements an Euler explicit scheme with a small step $h = 0.01$, i.e., it simulates the linear system

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leftarrow \begin{pmatrix} 1 & h \\ -h & 1-h \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}.$$

The invariant found with our strategy improvement methods (we are going to explain these methods in Section 5 and Section 6) is shown in Figure 2. For finding this invariant, we aimed at computing upper bounds $b_1, \ldots, b_5 \in \overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$ that are



$$-1.8708 \leq x_1 \leq 1.8708 \text{ and } -1.5275 \leq x_2 \leq 1.5275 \text{ and } 2x_1^2 + 3x_2^2 + 2x_1x_2 \leq 7$$

Fig. 2. Invariants for the Harmonic Oscillator

as small as possible and fulfill the following inequations for all possible values of the program variables x_1 and x_2 at program point $\mathbf{st}$:

$$-x_1 \leq b_1 \qquad x_1 \leq b_2 \qquad -x_2 \leq b_3 \qquad x_2 \leq b_4 \qquad 2x_1^2 + 3x_2^2 + 2x_1x_2 \leq b_5$$

This means, we consider a domain where we are looking for upper bounds for the linear polynomials $-x_1, x_1, -x_2, x_2$ (i.e., intervals for the values of the program variables) and the *non-linear* polynomial $2x_1^2 + 3x_2^2 + 2x_1x_2$. The last polynomial comes from the Lyapunov function that the designer of the algorithm may have considered to prove the stability of his scheme, *before it has been implemented*. In view of proving the implementation correct, one is naturally led to considering such polynomial templates[6]. Last but not least, it is to be noted that the loop invariant obtained when using intervals, zones, octagons or even polyhedra (hence with any set of linear templates) is the very disappointing invariant $\top$ (the value of the program variables x_1 and x_2 cannot be bounded). However, the main interest of our methods is to carry over to the non-linear setting. The benchmarks of Adjé et al. [2] and Gawlitza and Seidl [13], for instance, include a computation of invariants (of the same quality) for an implementation of the Arrow-Hurwicz algorithm, which is essentially an harmonic oscillator limited by a non-linear saturation term (a projection on the positive cone). They also include a symplectic integration scheme, wich is a highly degenerated example for which alternative methods fail due to the absence of stability margins.

3 Abstract Interpretation and Monotone Fixpoint Equations

In this section we reduce template based numerical static analysis by abstract interpretation to solving systems of in-equations of the form $\mathbf{x} \geq e$ over $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, \infty\}$, where the right-hand sides e are *monotonic* and *concave*.

3.1 Notations

The set of real numbers (resp. the set of rational numbers) is denoted by $\mathbb{R}$ (resp. $\mathbb{Q}$). The complete linear ordered set $\mathbb{R} \cup \{-\infty, \infty\}$ is denoted by $\overline{\mathbb{R}}$. Additionally, we set $\overline{\mathbb{Q}} := \mathbb{Q} \cup \{-\infty, \infty\}$. For $f : X \to \overline{\mathbb{R}}^m$ with $X \subseteq \overline{\mathbb{R}}^n$, we set

$$\mathrm{dom}(f) := \{x \in X \mid f(x) \in \mathbb{R}^m\} \text{ and } \mathrm{fdom}(f) := \mathrm{dom}(f) \cap \mathbb{R}^n.$$

[6] Of course, as for the linear templates of Sankaranarayanan et al. [19, 20], we can be interested in automatically finding or refining the set of polynomial templates considered to achieve good precision of the abstract analysis. However, this is outside the scope of the present article.

We denote the i-th row (resp. j-th column) of a matrix A 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. We also use this notation 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 $i \in \{1, \ldots, k\}$.

For $x, y \in \overline{\mathbb{R}}^n$, we write $x \leq y$ iff $x_{i\cdot} \leq y_{i\cdot}$ for all $i \in \{1, \ldots, n\}$. $\overline{\mathbb{R}}^n$ is partially ordered by $\leq$. We write $x < y$ iff $x \leq y$ and $x \neq y$. Finally, we write $x \lhd y$ iff $x_{i\cdot} < y_{i\cdot}$ for all $i \in \{1, \ldots, n\}$. x and y are called *comparable* iff $x \leq y$ or $y \leq x$.

Let $\mathbb{D}$ be a partially ordered set. We denote the *least upper bound* and the *greatest lower bound* of a set $X \subseteq \mathbb{D}$ by $\bigvee X$ and $\bigwedge X$, respectively, provided that they exist. The existence is in particular guaranteed if $\mathbb{D}$ is a *complete lattice*. The least element $\bigvee \emptyset$ (resp. the greatest element $\bigwedge \emptyset$) is denoted by $\bot$ (resp. $\top$), provided that it exists. Accordingly, we define the binary operators $\vee$ and $\wedge$ by

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

for all $x, y \in \mathbb{D}$, respectively. If $\mathbb{D}$ is a *linearly ordered set* (for instance $\mathbb{R}$ or $\overline{\mathbb{R}}$), then $\vee$ is the *maximum* operator and $\wedge$ the *minimum* operator. 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 and commutative.

A function $f : \mathbb{D}_1 \to \mathbb{D}_2$, where $\mathbb{D}_1$ and $\mathbb{D}_2$ are partially ordered sets, is called *monotone* iff $x \leq y \implies f(x) \leq f(y)$ for all $x, y \in \mathbb{D}_1$.

3.2 Convex and Concave Functions

A set $X \subseteq \mathbb{R}^n$ is called *convex* iff $\lambda x + (1 - \lambda)y \in X$ holds for all $x, y \in X$ and all $\lambda \in [0, 1]$. A mapping $f : X \to \mathbb{R}^m$ with $X \subseteq \mathbb{R}^n$ convex is called *convex* (resp. *concave*) iff

$$f(\lambda x + (1 - \lambda)y) \leq (\text{resp. } \geq)\ \lambda f(x) + (1 - \lambda) f(y)$$

holds for all $x, y \in X$ and all $\lambda \in [0, 1]$ (cf. e.g. Ortega and Rheinboldt [18]). Note that f is concave iff $-f$ is convex. Note also that f is convex (resp. concave) iff $f_{i\cdot}$ is convex (resp. concave) for all $i = 1, \ldots, m$.

We extend the notion of convexity/concavity from $\mathbb{R}^n \to \mathbb{R}^m$ to $\overline{\mathbb{R}}^n \to \overline{\mathbb{R}}^m$ as follows: Let $f : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}^m$, and $I : \{1, \ldots, n\} \to \{-\infty, \mathsf{id}, \infty\}$. Here, $-\infty$ denotes the function that assigns $-\infty$ to every argument, id denotes the identity function, and ∞ denotes the function that assigns ∞ to every argument. We define the mapping $f^{(I)} : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}^m$ by $f^{(I)}(x_1, \ldots, x_n) := f(I(1)(x_1), \ldots, I(n)(x_n))$ for all $x_1, \ldots, x_n \in \overline{\mathbb{R}}$. A mapping $f : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}^m$ is called *concave* iff $f_{i\cdot}$ is continuous on $\{x \in \overline{\mathbb{R}}^n \mid f_{i\cdot}(x) > -\infty\}$ for all $i \in \{1, \ldots, m\}$, and the following conditions are fulfilled for all $I : \{1, \ldots, n\} \to \{-\infty, \mathsf{id}, \infty\}$:

1. $\mathrm{fdom}(f^{(I)})$ is convex.
2. $f^{(I)}|_{\mathrm{fdom}(f^{(I)})}$ is concave.
3. For all $i \in \{1, \ldots, m\}$ the following holds: If there exists some $y \in \mathbb{R}^n$ such that $f_{i\cdot}^{(I)}(y) \in \mathbb{R}$, then $f_{i\cdot}^{(I)}(x) < \infty$ for all $x \in \mathbb{R}^n$.

A mapping $f : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}^m$ is called *convex* iff $-f$ is concave. In the following we are only concerned with mappings $f : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}^m$ that are monotone and concave. Figure 3 shows the graph of a function $f : \overline{\mathbb{R}}^2 \to \overline{\mathbb{R}}$ that is monotone and concave.

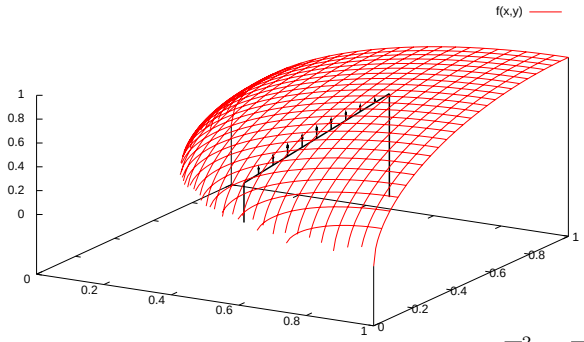
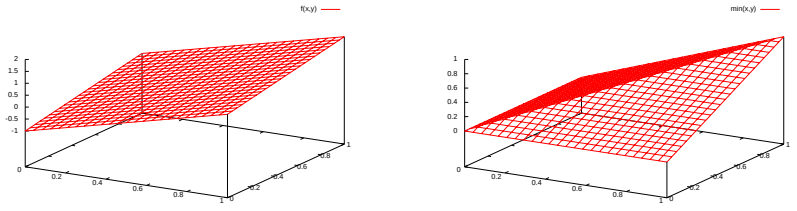


Fig. 3. Plot of a monotone and concave function $f : \overline{\mathbb{R}}^2 \to \overline{\mathbb{R}}$.



An affine function The minimum operator

Fig. 4. Examples of concave functions

Lemma 1. *Every affine function*[7] *is concave and convex. The operator* $\vee$ *is convex, but not concave. The operator* $\wedge$ *is concave, but not convex (see Figure 4).* □

3.3 Collecting Semantics

In our programming model, we consider statements of the form

$$g(x) \leq 0; x := p(x)$$

[7] A function $f : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}^m$ is called affine iff there exist some $A \in \mathbb{R}^{m \times n}$ and some $b \in \overline{\mathbb{R}}^m$ such that $f(x) = Ax + b$ for all $x \in \overline{\mathbb{R}}^n$. Here, we use the convention that $-\infty + \infty = -\infty$.

where $x = (x_1, \ldots, x_n)^\top \in \mathbb{R}^n$ denotes the vector of program variables, and $g \in \mathbb{R}^k[x_1, \ldots, x_n]$ and $p \in \mathbb{R}^n[x_1, \ldots, x_n]$ are multivariate polynomials with coefficients from $\mathbb{R}^k$ and $\mathbb{R}^n$, respectively. Here, 0 also denotes the zero vector. An example is

$$x_1^2 + x_2^2 - 16 \leq 0; \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} := \frac{5}{4} \begin{pmatrix} x_2 \\ x_1 \end{pmatrix}.$$

It assigns $\frac{5}{4}$ of the value of the program variable x_i to the program variable x_{3-i} for $i = 1, 2$, provided that $x_1^2 + x_2^2 - 16 \leq 0$ holds. A statement combines a guard followed by an assignment. The set of statements is denoted by $\mathbf{Stmt}$. Statements of the form $g(x) \leq 0$, i.e., p is the identity function, are called *guards*. Statements of the form $x := p(x)$, i.e., $k = 0$, are called *assignments*. A statement $g(x) \leq 0; x := p(x)$ is called *affine* (resp. *quadratic*, resp. *of order* d) iff the functions g and p are affine (resp. quadratic, resp. of order d).

As usual in static program analysis by abstract interpretation we refer to the program's collecting semantics, which safely over-approximates the concrete semantics. The *collecting semantics* $[\![s]\!] : 2^{\mathbb{R}^n} \to 2^{\mathbb{R}^n}$ of a statement $s \in \mathbf{Stmt}$ assigns a set $[\![s]\!]X$ of states after the execution of s to each set X of states before the execution of s. Here, a state of a program is modeled as a vector $x = (x_1, \ldots, x_n)^\top \in \mathbb{R}^n$. The collecting semantics of statements is defined by

$$[\![g(x) \leq 0; x := p(x)]\!]\, X := \{p(x) \mid x \in X,\ g(x) \leq 0\} \qquad \text{for all } X \subseteq \mathbb{R}^n.$$

We represent programs by their control-flow graphs, i.e., a program G is a triple $(N, E, \mathbf{st})$, where N is a finite set of *program points*, $E \subseteq N \times \mathbf{Stmt} \times N$ is a finite set of control-flow edges, and $\mathbf{st} \in N$ is the start program point. As usual, the *collecting semantics* V of a program $G = (N, E, \mathbf{st})$ w.r.t. a set $I \subseteq \mathbb{R}^n$ of *initial states* is the least solution of the following constraint system:

$$\mathbf{V}[\mathbf{st}] \supseteq I \qquad \mathbf{V}[v] \supseteq [\![s]\!](\mathbf{V}[u]) \quad \text{for all } (u, s, v) \in E$$

Here, the variables $\mathbf{V}[v]$, $v \in N$ take values in $2^{\mathbb{R}^n}$. The components of the collecting semantics V are denoted by $V[v]$ for all $v \in N$.

3.4 Polynomial Templates

We are now going to define the abstract domain we are going to use thru-out the present paper. Following the lines of Adjé et al. [2], we assume that we have given a fixed set

$$P \subseteq \mathbb{R}[x_1, \ldots, x_n]$$

of *polynomial templates* with coefficients from $\mathbb{R}$. P is called *linear* (resp. *quadratic*, resp. *of order* d) iff all polynomials $p \in P$ are linear (resp. quadratic, resp. of order d). Usually, P will consist of finitely many templates, only.

Example 1 (Adjé et al. [2]). The set $P = \{p_1, p_2, p_3, p_4, p_5\}$ with

$$p_1(x_1, x_2) = -x_1 \qquad\qquad p_2(x_1, x_2) = x_1$$

$$p_3(x_1, x_2) = -x_2 \qquad p_4(x_1, x_2) = x_2$$
$$p_5(x_1, x_2) = 2x_1^2 + 3x_2^2 + 2x_1x_2$$

is a *set of polynomial templates*. More precisely, it is a finite set of *quadratic templates*. This set of quadratic templates is used for analyzing the harmonic oscillator introduced in Section 2. □

Within the present paper, an abstract value is a mapping $v : P \to \overline{\mathbb{R}}$ that assigns an upper bound $v(q)$ to every polynomial $q \in P$. The abstract value v represents the set of all program states $x \in \mathbb{R}^n$ such that $q(x) \leq v(q)$ holds for all $q \in P$. This means, we define a Galois-connection that consists of the *abstraction* $\alpha : 2^{\mathbb{R}^n} \to P \to \overline{\mathbb{R}}$ and the *concretization* $\gamma : (P \to \overline{\mathbb{R}}) \to 2^{\mathbb{R}^n}$ as follows:

$$\gamma(v) := \{x \in \mathbb{R}^n \mid \forall p \in P \,.\, p(x) \leq v(p)\} \qquad \text{for all } v : P \to \overline{\mathbb{R}}$$
$$\alpha(X) := \bigwedge \{v : P \to \overline{\mathbb{R}} \mid \gamma(v) \supseteq X\} \qquad \text{for all } X \subseteq \mathbb{R}^n$$

As shown by Adjé et al. [2], α and γ form a Galois-connection. The elements from $\gamma(P \to \overline{\mathbb{R}})$ and the elements from $\alpha(2^{\mathbb{R}^n})$ are called *closed*. $\alpha(\gamma(v))$ is called the *closure* of the abstract value $v : P \to \mathbb{R}$. Accordingly, $\gamma(\alpha(X))$ is called the *closure* of the set $X \subseteq \mathbb{R}^n$ of states. It is the minimal set of states that subsumes X and can be represented by an abstract value v.

Before we go further, we discuss some aspects of the closure operation $\alpha \circ \gamma$. For all abstract values $v : P \to \overline{\mathbb{R}}$ and all polynomial templates $r \in P$, we have

$$\begin{aligned} \alpha(\gamma(v))(r) &= \sup \{r(x) \mid x \in \gamma(v)\} \\ &= \sup \{r(x) \mid x \in \mathbb{R}^n \text{ and } \forall q \in P \,.\, q(x) \leq v(q)\} \\ &= \inf \{-r(x) \mid x \in \mathbb{R}^n \text{ and } \forall q \in P \,.\, q(x) \leq v(q)\} \end{aligned} \tag{4}$$

The above equalities (cf. Adjé et al. [2]) lead to the following remarks:

Remark 1. If P is finite and all polynomial templates $p' \in P$ with $v(p') < \infty$ (i.e., all polynomial templates that are bounded) are linear and r is quadratic (not necessarily concave), then $\alpha(\gamma(v))(r)$ can be computed by solving a *quadratic optimization problem* (cf. (4)). Solving quadratic optimization problems is $\mathbf{NP}$-complete (see e.g. Vavasis [23]). Vice versa, solving quadratic optimization problems is polynomial-time reducible to computing closures. Thus, computing closures is $\mathbf{NP}$-hard.

Remark 2. If P is finite and linear, then closures can be computed by solving linear programming problems, i.e., in polynomial time.

Remark 3. If P is finite and all polynomial templates $p' \in P$ with $v(p') < \infty$ (i.e., all polynomial templates that are bounded) are convex and r is concave (i.e. $-r$ is convex), then $\alpha(\gamma(v))(r)$ can be computed by solving a *convex optimization problem* (cf. (4)). If all polynomial templates $p' \in P$ with $v(p') < \infty$ and p are additionally quadratic, then $\alpha(\gamma(v))(r)$ can be computed by solving a *convex quadratic optimization problem*. Convex quadratic optimization problems can be computed through semi-definite programming (see e.g. Todd [22]).

Example 2 (Adjé et al. [2]). We continue Example 1. Let

$$v = \{p_1 \mapsto 0,\ p_2 \mapsto 1,\ p_3 \mapsto 0,\ p_4 \mapsto 1,\ p_5 \mapsto \infty\}.$$

Then $\gamma(v) = [0,1] \times [0,1]$. The closure of v is

$$\alpha(\gamma(v)) = \{p_1 \mapsto 0,\ p_2 \mapsto 1,\ p_3 \mapsto 0,\ p_4 \mapsto 1,\ p_5 \mapsto 7\},$$

because $\alpha(\gamma(v))(p_5) = \sup\{p_5(x_1, x_2) \mid (x_1, x_2)^\top \in \gamma(v)\} = 7$. □

3.5 Abstract Semantics

As usual in static analysis by abstract interpretation, we abstract the collecting semantics using the abstraction introduced in Subsection 3.4: The *abstract semantics* $[\![s]\!]^\sharp : (P \to \overline{\mathbb{R}}) \to P \to \overline{\mathbb{R}}$ of a statement s is defined by $[\![s]\!]^\sharp := \alpha \circ [\![s]\!] \circ \gamma$. As usual, the *abstract semantics* $V^\sharp$ of a program $G = (N, E, \mathbf{st})$ w.r.t. to a set $I \subseteq \mathbb{R}^n$ of initial states is the least solution of the following constraint system:

$$\mathbf{V}^\sharp[\mathbf{st}] \geq \alpha(I) \qquad \mathbf{V}^\sharp[v] \geq [\![s]\!]^\sharp(\mathbf{V}^\sharp[u]) \quad \text{for all } (u, s, v) \in E$$

Here, the variables $\mathbf{V}^\sharp[v], v \in N$ take values in $P \to \overline{\mathbb{R}}$. The components of the abstract semantics $V^\sharp$ are denoted by $V^\sharp[v]$ for all $v \in N$. The abstract semantics safely over-approximates the collecting semantics:

Lemma 2. *$V[v] \subseteq \gamma(V^\sharp[v])$ and $\alpha(V[v]) \leq V^\sharp[v]$ hold for all program points v.* □

In the present paper we aim at using convex optimization for approximating abstract semantics as precisely as possible. However, for that, as we will see later, it would be preferable when $[\![s]\!]^\sharp$ was concave (i.e., $-[\![s]\!]^\sharp$ was convex) for every statement s. Unfortunately, this property is not always fulfilled as the following example shows:

Example 3. Assume that $P = \{p_1, p_2, p_3\} \subseteq \mathbb{R}[x_1]$ with

$$p_1(x_1) = x_1 \qquad p_2(x_1) = -x_1 \qquad p_3(x_1) = x_1^2$$

for all $x_1 \in \mathbb{R}$. We consider the statement $s = x_1 := x_1$, i.e., the statement s does not modify the state. Then, for all $v_x = \{p_1 \mapsto x, p_2 \mapsto 0, p_3 \mapsto \infty\}$ with $x \in \overline{\mathbb{R}}$, we have

$$\begin{aligned}([\![s]\!]^\sharp v_x)(p_3) &= \sup\{p_3(x_1) \mid x_1 \in \gamma(v_x)\} \\ &= \sup\{x_1^2 \mid x_1 \in \mathbb{R} \text{ and } 0 \leq x_1 \leq x\} \\ &= \begin{cases} x^2 & \text{if } x \geq 0 \\ -\infty & \text{if } x < 0 \end{cases}\end{aligned}$$

Hence, we get

$$\begin{aligned}([\![s]\!]^\sharp(\frac{1}{2}v_0 + \frac{1}{2}v_2))(p_3) &= ([\![s]\!]^\sharp(v_1)(p_3) = 1 \\ &\not\geq 2 = \frac{1}{2}0 + \frac{1}{2}4 = \frac{1}{2}([\![s]\!]^\sharp v_0)(p_3) + \frac{1}{2}([\![s]\!]^\sharp v_2)(p_3)\end{aligned}$$

This implies that $[\![s]\!]^\sharp$ is not concave. □

However, there are cases, where $[\![s]\!]^\sharp$ is indeed concave. One important case is when all polynomials $q \in P$ are affine, and the statement s is affine. This case is studied by Costan et al. [6] and by Gawlitza and Seidl [10].

The pathbreaking idea of Adjé et al. [2] is to use a convex relaxation of $-[\![s]\!]^\sharp$ (resp. concave relaxation of $[\![s]\!]^\sharp$) instead of $-[\![s]\!]^\sharp$ (resp. $[\![s]\!]^\sharp$). By doing so, an intractable NP-hard problem is approximated by a convex optimization problem. This is a big advantage, since a wide class of convex optimization problems can be solved efficiently. In the remainder of the present paper we will be in particular faced with semi-definite programming problems which are special convex optimization problems for which efficient interior point methods exist.

3.6 Relaxed Abstract Semantics

Adjé et al. [2] proposed to use *convex relaxation schemas* in order to approximate the abstract semantics. They approximate the abstract semantics $[\![s]\!]^\sharp$ of a statement s by a *relaxed abstract semantics* $[\![s]\!]^{\mathcal{R}}$ that fulfills the following properties:

1. $[\![s]\!]^{\mathcal{R}} \geq [\![s]\!]^\sharp$, i.e., the relaxed abstract semantics $[\![s]\!]^{\mathcal{R}}$ of s safely over-approximates the abstract semantics $[\![s]\!]^\sharp$ of s.
2. $[\![s]\!]^{\mathcal{R}}$ is monotone and concave.

The *relaxed abstract semantics* $V^{\mathcal{R}}$ of a program $G = (N, E, \mathbf{st})$ with initial states I is then defined as the least solution of the following constraint system over $P \to \overline{\mathbb{R}}$:

$$\mathbf{V}^{\mathcal{R}}[\mathbf{st}] \geq \alpha(I) \qquad \mathbf{V}^{\mathcal{R}}[v] \geq [\![s]\!]^{\mathcal{R}}(\mathbf{V}^{\mathcal{R}}[u]) \quad \text{for all } (u, s, v) \in E \tag{5}$$

Here, the variables $\mathbf{V}^{\mathcal{R}}[v]$, $v \in N$ take values in $P \to \overline{\mathbb{R}}$. The components of the relaxed abstract semantics $V^{\mathcal{R}}$ are denoted by $V^{\mathcal{R}}[v]$ for all $v \in N$. The relaxed abstract semantics safely over-approximates the abstract semantics, and thus finally the collecting semantics and the concrete semantics:

Lemma 3. *$V^\sharp[v] \leq V^{\mathcal{R}}[v]$ for all program points v.* □

We want to emphasize that the set of all solutions of the constraints system (5) which defines the relaxed abstract semantics $V^{\mathcal{R}}$ is *not* always convex, although the relaxed abstract semantics $[\![s]\!]^{\mathcal{R}}$ is concave for each statement s. In consequence it is not possible to compute $V^{\mathcal{R}}$ through convex optimization.

3.7 Obtaining a Relaxed Abstract Semantics through Semi-definite Relaxation

In this subsection we briefly discuss the relaxed abstract semantics introduced by Adjé et al. [2]. This relaxed abstract semantics is based on Shor's semi-definite relaxation schema. This subsection is slightly more technical then the rest of the present paper. However, it is not essential for the understanding of the rest of the paper. The purpose of this subsection is to demonstrate that a non-trivial relaxed abstract semantics exists that fulfills the requirements mentioned in Subsection 3.6.

Semi-Definite Programming $S\mathbb{R}^{n\times n}$ (resp. $S\mathbb{R}^{n\times n}_{+}$, resp. $S\mathbb{R}^{n\times n}_{++}$) denotes the set of symmetric matrices (resp. the set of positive semi-definite matrices, resp. the set of positive definite matrices). $\preceq$ denotes the Löwner ordering of symmetric matrices, i.e., $A \preceq B$ iff $B - A \in S\mathbb{R}^{n\times n}_{+}$. We write $A \prec B$ iff $B - A \in S\mathbb{R}^{n\times n}_{++}$. $\mathrm{Tr}(A)$ denotes the trace of a square matrix $A \in \mathbb{R}^{n\times n}$, i.e., $\mathrm{Tr}(A) = \sum_{i=1}^{n} A_{i\cdot i}$. The inner product of two matrices A and B is denoted by $A \bullet B$, i.e., $A \bullet B = \mathrm{Tr}(A^\top B)$. For $\mathcal{A} = (A_1, \ldots, A_m)$ with $A_i \in \mathbb{R}^{n\times n}$ for all $i = 1, \ldots, m$, we denote the vector $(A_1 \bullet X, \ldots, A_m \bullet X)^\top$ by $\mathcal{A}(X)$. We consider semi-definite programming problems (SDP problems for short) of the form

$$z^* = \sup \{C \bullet X \mid X \in S\mathbb{R}^{n\times n}_{+}, \mathcal{A}(X) = a, \mathcal{B}(X) \leq b\},$$

where $\mathcal{A} = (A_1, \ldots, A_m)$, $a \in \mathbb{R}^m$, $A_1, \ldots, A_m \in S\mathbb{R}^{n\times n}$, $\mathcal{B} = (B_1, \ldots, B_k)$, $B_1, \ldots, B_k \in S\mathbb{R}^{n\times n}$, $b \in \mathbb{R}^k$, and $C \in S\mathbb{R}^{n\times n}$. The value $z^* \in \overline{\mathbb{R}}$ is called the *optimal value*. The set

$$\{X \in S\mathbb{R}^{n\times n}_{+} \mid \mathcal{A}(X) = a, \mathcal{B}(X) \leq b\}$$

is called the *feasible space*. An element of the feasible space, is called *feasible solution*. The problem is called *infeasible* iff the feasible space is empty, i.e., $z^* = -\infty$. It is called *unbounded* iff $z^* = \infty$. A feasible solution X^* is called *optimal solution* iff $C \bullet X^* = z^*$.

The Relaxation For the remainder of this subsection we assume that P is finite, all templates $p \in P$ are quadratic (but not necessarily convex), and all statements are of the form $g(x) \leq 0; x := p(x)$, where g is quadratic and p is affine. The goal is to define a relaxed abstract semantics which satisfies the properties described in Subsection 3.6. For that, we use Shor's semi-definite relaxation schema.

Let $s = g(x) \leq 0; x := p(x)$ be a statement. Recall that the abstract semantics $[\![s]\!]^\sharp$ of s is given by

$$([\![s]\!]^\sharp v)(r) = \sup \{r(p(x)) \mid x \in \mathbb{R}^n \text{ and } g(x) \leq 0 \text{ and } \forall q \in P \,.\, q(x) \leq v(q)\}$$

for all abstract values $v : P \to \overline{\mathbb{R}}$ and all templates $r \in P$. Because g is quadratic and p is affine, we had to solve a non-linear optimization problem for computing $([\![s]\!]^\sharp v)(r)$. Unfortunately, this non-linear optimization problem is not necessarily convex. Using (the dual version of) Shor's semi-definite relaxation schema, we relax the abstract semantics $[\![s]\!]^\sharp$ of s as follows. W.l.o.g., we assume:

1. For every polynomial $q \in \mathbb{R}[x_1, \ldots, x_n]$ with coefficients from $\mathbb{R}$, there are some $A_q \in S\mathbb{R}^{n\times n}$, some $b_q \in \mathbb{R}^n$, and some $c_q \in \mathbb{R}$ such that

$$q(x) = x^\top A_q x + 2{b_q}^\top x + c_q.$$

2. $p(x) = Ax + b$ with $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$.

For all $v : P \to \mathbb{R}$, and all $r \in P$, we then get

$$
\begin{aligned}
(\llbracket s \rrbracket^\sharp v)(r) &= \sup \{ r(p(x)) \mid x \in \mathbb{R}^n, g(x) \leq 0,\ \forall q \in P \,.\, q(x) \leq v(q) \} \\
&= \sup \{ r(Ax+b) \mid x \in \mathbb{R}^n, \\
&\quad \forall i \in \{1,\ldots,k\} \,.\, x^\top A_{g_{i\cdot}} x + 2 b_{g_{i\cdot}}^\top x + c_{g_{i\cdot}} \leq 0, \\
&\quad \forall q \in P \,.\, x^\top A_q x + 2 b_q^\top x + c_q \leq v(q) \} \\
&= \sup \{ x^\top A^\top A_r A x + 2 b^\top A_r A x + 2 b_r^\top A x + b^\top A_r b + 2 b_r^\top b + c_r \mid \\
&\quad x \in \mathbb{R}^n, \\
&\quad \forall i \in \{1,\ldots,k\} \,.\, x^\top A_{g_{i\cdot}} x + 2 b_{g_{i\cdot}}^\top x + c_{g_{i\cdot}} \leq 0, \\
&\quad \forall q \in P \,.\, x^\top A_q x + 2 b_q^\top x + c_q \leq v(q) \} \\
&= \sup \{ (1, x^\top) \begin{pmatrix} b^\top A_r b + 2 b_r^\top b + c_r & b^\top A_r A + b_r^\top A \\ (b^\top A_r A + b_r^\top A)^\top & A^\top A_r A \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix} \mid \\
&\quad x \in \mathbb{R}^n, \\
&\quad \forall i \in \{1,\ldots,k\} \,.\, (1, x^\top) \begin{pmatrix} c_{g_{i\cdot}} & b_{g_{i\cdot}}^\top \\ b_{g_{i\cdot}} & A_{g_{i\cdot}} \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix} \leq 0, \\
&\quad \forall q \in P \,.\, (1, x^\top) \begin{pmatrix} c_q & b_q{}^\top \\ b_q & A_q \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix} \leq v(q) \} \\
&= \sup \{ \begin{pmatrix} b^\top A_r b + 2 b_r^\top b + c_r & b^\top A_r A + b_r^\top A \\ (b^\top A_r A + b_r^\top A)^\top & A^\top A_r A \end{pmatrix} \bullet \begin{pmatrix} 1 \\ x \end{pmatrix} (1, x^\top) \mid \\
&\quad x \in \mathbb{R}^n, \\
&\quad \forall i \in \{1,\ldots,k\} \,.\, \begin{pmatrix} c_{g_{i\cdot}} & b_{g_{i\cdot}}^\top \\ b_{g_{i\cdot}} & A_{g_{i\cdot}} \end{pmatrix} \bullet \begin{pmatrix} 1 \\ x \end{pmatrix} (1, x^\top) \leq 0, \\
&\quad \forall q \in P \,.\, \begin{pmatrix} c_q & b_q{}^\top \\ b_q & A_q \end{pmatrix} \bullet \begin{pmatrix} 1 \\ x \end{pmatrix} (1, x^\top) \leq v(q) \} \\
&\leq \sup \{ \begin{pmatrix} b^\top A_r b + 2 b_r^\top b + c_r & b^\top A_r A + b_r^\top A \\ (b^\top A_r A + b_r^\top A)^\top & A^\top A_r A \end{pmatrix} \bullet X \mid \\
&\quad X \succeq 0,\ X_{1\cdot 1} = 1 \\
&\quad \forall i \in \{1,\ldots,k\} \,.\, \begin{pmatrix} c_{g_{i\cdot}} & b_{g_{i\cdot}}^\top \\ b_{g_{i\cdot}} & A_{g_{i\cdot}} \end{pmatrix} \bullet X \leq 0, \\
&\quad \forall q \in P \,.\, \begin{pmatrix} c_q & b_q{}^\top \\ b_q & A_q \end{pmatrix} \bullet X \leq v(q) \}.
\end{aligned}
$$

The last inequality holds, because $X \succeq 0$ and $X_{1\cdot 1} = 1$ hold for all X and all x with

$$X = \begin{pmatrix} 1 \\ x \end{pmatrix} (1, x^\top).$$

Because of the above inequality, we define the relaxed abstract semantics $\llbracket s \rrbracket^{\mathcal{R}}$ of s by

$$(\llbracket s \rrbracket^{\mathcal{R}} v)(r) := \sup \{ \begin{pmatrix} b^\top A_r b + 2 b_r^\top b + c_r & b^\top A_r A + b_r^\top A \\ (b^\top A_r A + b_r^\top A)^\top & A^\top A_r A \end{pmatrix} \bullet X \mid$$

$$X \succeq 0,\ X_{1\cdot 1} = 1 \tag{6}$$
$$\forall i \in \{1,\ldots,k\}\ .\ \begin{pmatrix} c_{g_i\cdot} & b_{g_i\cdot}^\top \\ b_{g_i\cdot} & A_{g_i\cdot} \end{pmatrix} \bullet X \leq 0,$$
$$\forall q \in P\ .\ \begin{pmatrix} c_q & b_q^\top \\ b_q & A_q \end{pmatrix} \bullet X \leq v(q)\}.$$

We collect the important properties of the relaxed abstract semantics $[\![s]\!]^{\mathcal{R}}$ in the following lemma:

Lemma 4 (Adjé et al. [2], Gawlitza and Seidl [13]). *Let $s = g(x) \leq 0; x := p(x)$ be a statement, where g is quadratic and p is affine. Assume that P is finite and all $q \in P$ are quadratic. For the relaxed abstract semantics $[\![s]\!]^{\mathcal{R}}$ of s as defined in* (6) *we have:*

1. $[\![s]\!]^{\mathcal{R}} \geq [\![s]\!]^\sharp$.
2. $[\![s]\!]^{\mathcal{R}}$ *is monotone and concave.*
3. $([\![s]\!]^{\mathcal{R}} v)(r) = ([\![s]\!]^\sharp v)(r)$*, whenever r is concave, g is convex, and all polynomial templates $q \in P$ with $v(q) < \infty$ are convex. This in particular implies that $[\![s]\!]^{\mathcal{R}} = [\![s]\!]^\sharp v$, whenever s is affine and all polynomial templates $q \in P$ are affine.* □

The methods we are developing in the present paper are, because of the last statement of the above lemma, a generalization of the methods developed by Gaubert et al. [9] and Gawlitza and Seidl [10].

3.8 Systems of Inequations over $\overline{\mathbb{R}}$

We want to reduce the problem of computing the relaxed abstract semantics $V^{\mathcal{R}}$ of a program G w.r.t. a set $I \subseteq \mathbb{R}^n$ of initial states to solving a system $\mathcal{C}(G, I)$ of inequations of the form $\mathbf{x} \geq e$ over $\overline{\mathbb{R}}$, where each right-hand side e is monotone and concave. We set up this system $\mathcal{C}(G, I)$ as follows:

$$\begin{array}{ll} \mathbf{x}_{\mathbf{st},p} \geq \alpha(I)(p) & \text{for all } p \in P \\ \mathbf{x}_{v,p} \geq ([\![s]\!]^{\mathcal{R}}\{q \mapsto \mathbf{x}_{u,q} \mid q \in P\})(p) & \text{for all } (u,s,v) \in E, \text{and all } p \in P \end{array}$$

The system $\mathcal{C}(G, I)$ of inequations uses the set of variables

$$\mathbf{X} = \{\mathbf{x}_{v,p} \mid v \in N \text{ and } p \in P\}.$$

The variable $\mathbf{x}_{v,p}$ receives the value for the upper bound for $p \in P$ at program point v. The relaxed abstract semantics of G w.r.t. to the set I of initial states can finally be read off the least solution of the system $\mathcal{C}(G, I)$ of inequations over $\overline{\mathbb{R}}$:

Lemma 5. *Let $\rho^* : \mathbf{X} \to \overline{\mathbb{R}}$ denote the least solution of $\mathcal{C}(G, I)$. Then*

$$V^{\mathcal{R}}[v](p) = \rho^*(\mathbf{x}_{v,p})$$

for all $v \in N$ and all $p \in P$. □

Because of the above lemma, it remains to develop methods for approximating or computing the least solution of $\mathcal{C}(G, I)$. This will be the topic of the next sections.

Example 4. We continue our running example from Section 2. The set $P = \{p_1, \ldots, p_5\} \subseteq \mathbb{R}[x_1, x_2]$ of quadratic templates we consider for this example is given by

$$p_1(x_1, x_2) = -x_1 \qquad p_2(x_1, x_2) = x_1 \qquad p_3(x_1, x_2) = -x_2$$
$$p_4(x_1, x_2) = x_2 \qquad p_5(x_1, x_2) = 2x_1^2 + 3x_2^2 + 2x_1x_2$$

for all $x_1, x_2 \in \mathbb{R}$. By Lemma 5, the relaxed abstract semantics is given by the least solution of the following system of inequations:

$$\begin{aligned}
\mathbf{x}_{\mathbf{st},p_1} &\geq 0 & \mathbf{x}_{\mathbf{st},p_1} &\geq (\llbracket s \rrbracket^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_1)\\
\mathbf{x}_{\mathbf{st},p_2} &\geq 1 & \mathbf{x}_{\mathbf{st},p_2} &\geq (\llbracket s \rrbracket^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_2)\\
\mathbf{x}_{\mathbf{st},p_3} &\geq 0 & \mathbf{x}_{\mathbf{st},p_3} &\geq (\llbracket s \rrbracket^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_3)\\
\mathbf{x}_{\mathbf{st},p_4} &\geq 1 & \mathbf{x}_{\mathbf{st},p_4} &\geq (\llbracket s \rrbracket^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_4)\\
\mathbf{x}_{\mathbf{st},p_5} &\geq 7 & \mathbf{x}_{\mathbf{st},p_5} &\geq (\llbracket s \rrbracket^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_5)
\end{aligned}$$

Here,

$$s \;=\; \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} := \begin{pmatrix} 1 & 0.01 \\ -0.01 & 0.99 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

and, according to equality (6),

$$\begin{aligned}
&(\llbracket s \rrbracket^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_i)\\
=&\sup\{C_i \bullet X \mid X \succeq 0, X_{1\cdot 1} = 1, B_1 \bullet X \leq \mathbf{x}_{\mathbf{st},p_1}, \cdots, B_5 \bullet X \leq \mathbf{x}_{\mathbf{st},p_5}\}
\end{aligned}$$

for all $i \in \{1, \ldots, 5\}$, where

$$B_1 = \begin{pmatrix} 0 & -0.5 & 0 \\ -0.5 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad B_2 = \begin{pmatrix} 0 & 0.5 & 0 \\ 0.5 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$
$$B_3 = \begin{pmatrix} 0 & 0 & -0.5 \\ 0 & 0 & 0 \\ -0.5 & 0 & 0 \end{pmatrix} \qquad B_4 = \begin{pmatrix} 0 & 0 & 0.5 \\ 0 & 0 & 0 \\ 0.5 & 0 & 0 \end{pmatrix}$$
$$B_5 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 3 \end{pmatrix}$$

$$C_1 = \begin{pmatrix} 0 & -0.5 & -0.005 \\ -0.5 & 0 & 0 \\ -0.005 & 0 & 0 \end{pmatrix} \qquad C_2 = \begin{pmatrix} 0 & 0.5 & 0.005 \\ 0.5 & 0 & 0 \\ 0.005 & 0 & 0 \end{pmatrix}$$
$$C_3 = \begin{pmatrix} 0 & 0.005 & -0.495 \\ 0.005 & 0 & 0 \\ -0.495 & 0 & 0 \end{pmatrix} \qquad C_4 = \begin{pmatrix} 0 & -0.005 & 0.495 \\ -0.005 & 0 & 0 \\ 0.495 & 0 & 0 \end{pmatrix}$$

$$C_5 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1.9803 & 0.9802 \\ 0 & 0.9802 & 2.9603 \end{pmatrix}$$

The above system of inequations has the same least solution as the following system of equations:

$$\begin{aligned}
\mathbf{x}_{\mathbf{st},p_1} &= 0 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_1) \\
\mathbf{x}_{\mathbf{st},p_2} &= 1 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_2) \\
\mathbf{x}_{\mathbf{st},p_3} &= 0 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_3) \\
\mathbf{x}_{\mathbf{st},p_4} &= 1 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_4) \\
\mathbf{x}_{\mathbf{st},p_5} &= 7 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_5)
\end{aligned} \tag{7}$$

In Section 5 we will explain how to solve the above equation system through the $\wedge$-strategy iteration approach of Adjé et al. [1, 2], Costan et al. [6], Gaubert et al. [9]. In Section 6 we will do this using the $\vee$-strategy iteration approach of Gawlitza and Seidl [10, 11, 13, 14]. □

4 Systems of Concave Equations

This section is dedicated to introduce the main object of our studies, namely *systems of concave equations*. How these equation systems can be solved through strategy iteration will be explained in the following sections.

Assume that a fixed set $\mathbf{X}$ of variables and a domain $\mathbb{D}$ is given. We consider equations of the form $\mathbf{x} = e$, where $\mathbf{x} \in \mathbf{X}$ is a variable and e is an expression over $\mathbb{D}$. A *system* $\mathcal{E}$ of equations is a finite set

$$\mathcal{E} = \{\mathbf{x}_1 = e_1, \ldots, \mathbf{x}_n = e_n\}$$

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.

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

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

where $\mathbf{x} \in \mathbf{X}$, f is a k-ary operator ($k = 0$ is possible; then f is a constant), for instance $+$, and $e_1, \ldots, e_k$ are expressions. Let $\mathcal{E}$ be a system of equations. We define the unary operator $[\![\mathcal{E}]\!]$ on $\mathbf{X} \to \mathbb{D}$ by setting $([\![\mathcal{E}]\!]\rho)(\mathbf{x}) := [\![e]\!]\rho$ for all $\mathbf{x} = e \in \mathcal{E}$. A solution is a variable assignment ρ such that $\rho = [\![\mathcal{E}]\!]\rho$ holds. The set of solutions is denoted by $\mathbf{Sol}(\mathcal{E})$.

Assume in the following that $\mathbb{D}$ is a complete lattice. An expression e (resp. an equation $\mathbf{x} = e$) is called *monotone* iff all operators occurring in e are monotone.

The set $\mathbf{X} \to \mathbb{D}$ of all *variable assignments* is a complete lattice. For $\rho, \rho' : \mathbf{X} \to \mathbb{D}$, we write $\rho \lhd \rho'$ (resp. $\rho \rhd \rho'$) iff $\rho(\mathbf{x}) < \rho'(\mathbf{x})$ (resp. $\rho(\mathbf{x}) > \rho'(\mathbf{x})$) holds for all $\mathbf{x} \in \mathbf{X}$.

For $d \in \mathbb{D}$, $\underline{d}$ denotes the variable assignment $\{\mathbf{x} \mapsto d \mid \mathbf{x} \in \mathbf{X}\}$. A variable assignment ρ with $\underline{\bot} \lhd \rho \lhd \underline{\top}$ is called *finite*. A pre-solution (resp. post-solution) is a variable assignment ρ such that $\rho \leq [\![\mathcal{E}]\!]\rho$ (resp. $\rho \geq [\![\mathcal{E}]\!]\rho$) holds. 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 fixpoint (resp. the greatest fixpoint) of an operator $f : \mathbb{D} \to \mathbb{D}$ is denoted by μf (resp. νf), provided that it exists. Thus, 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). In our setting, Knaster-Tarski's 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 $\nu[\![\mathcal{E}]\!] = \bigvee \mathbf{PreSol}(\mathcal{E})$.

Definition 1 (Concave Equations). *An expression e (resp. equation $\mathbf{x} = e$) over $\overline{\mathbb{R}}$ is called* basic concave expression *(resp.* basic concave equation*) iff $[\![e]\!]$ is monotone and concave. An expression e (resp. equation $\mathbf{x} = e$) over $\overline{\mathbb{R}}$ is called* concave *iff $e = e_1 \vee \cdots \vee e_k$, where $e_1, \ldots, e_k$ are basic concave expressions.* □

Note that by Lemma 1 the class of systems of concave equations strictly subsumes the class of *systems of rational equations* and even the class of *systems of rational LP-equations* as introduced by Gawlitza and Seidl [10, 14].

Example 5. The operator $\sqrt{\cdot} : \overline{\mathbb{R}} \to \overline{\mathbb{R}}$ (defined by $\sqrt{x} = \sup\{y \in \mathbb{R} \mid y^2 \leq x\}$ for all $x \in \overline{\mathbb{R}}$) is monotone and concave. The least solution of the system $\mathcal{E} = \{\mathbf{x} = \frac{1}{2} \vee \sqrt{\mathbf{x}}\}$ of concave equations is $\mu[\![\mathcal{E}]\!] = 1$. □

5 The Min-Strategy Iteration Approach

In this section we briefly present the $\wedge$-strategy iteration approach of Costan et al. [6]. The general framework will be explained in Subsection 5.1. After that we start specializing the general $\wedge$-strategy iteration algorithm to an algorithm for solving systems of concave equations as introduced in Section 4. For that, we first show how to compute least solutions of systems of in-equations of the form $\mathbf{x}_k \geq f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, where f is an affine operator on $\overline{\mathbb{R}}$. This algorithm will later be used for evaluating $\wedge$-strategies. Then, in Subsection 5.3, we answer the question how the set of $\wedge$-strategies that is defined by a system of concave equations looks like. In Subsection 5.4 we finally show how convex optimization can be utilized for computing an improvement of a $\wedge$-strategy. In Subsection 5.5 we apply the developed $\wedge$-strategy improvement algorithm to the harmonic oscillator introduced in Section 2.

5.1 The General Framework

Costan et al. [6] introduced a $\wedge$-strategy iteration approach for finding small solutions of monotone fixpoint equation systems over complete lattices. Monotone fixpoint equation systems for instance arise when performing static analysis by abstract interpretation (cf.

Section 3). However, before we specialize the $\wedge$-strategy iteration approach of Costan et al. [6] to an algorithm that can be applied to the case described in Section 3, we briefly describe the framework in an abstract way. For details and proofs we refer to Adjé et al. [2], Costan et al. [6], Gaubert et al. [9].

Let $\mathbb{D}$ be a complete lattice. We are interested in computing a small fixpoint of a monotone self-map $f : \mathbb{D} \to \mathbb{D}$, where we assume that $f(x) = \min \{\pi(x) \mid \pi \in \Pi\}$ for all $x \in \mathbb{D}$, where Π is a family of "simpler" self-maps on $\mathbb{D}$. This in particular means that the family Π *admits lower selections* (cf. e.g. [1]): Observe that, for all $x \in \mathbb{D}$, $f(x) = \min \{\pi(x) \mid \pi \in \Pi\}$ iff $f(x) = \bigwedge\{\pi(x) \mid \pi \in \Pi\}$ and there exists a $\pi \in \Pi$ such that $f(x) = \pi(x)$. The latter π is the lower selection for x. The term "simpler" in practice means that we assume that we are capable of computing the least fixpoint $\mu\pi$ of π for any $\pi \in \Pi$ efficiently. The self-maps $\pi \in \Pi$ are the $\wedge$-*strategies* for f. $\vee$-strategies can be defined dually. Then we assume that $f(x) = \max \{\sigma(x) \mid \sigma \in \Sigma\}$ for all $x \in \mathbb{D}$, where Σ is a family of "simpler" self-maps on $\mathbb{D}$, i.e., we assume that Σ *admits upper selections*.

Example 6. Consider the following system of concave equations:

$$\mathbf{x} = 0 \vee \left(\frac{1}{2} \cdot \mathbf{x} + 1 \wedge 10\right)$$

Let $f(\mathbf{x})$ denote the right-hand side, i.e., $f : \overline{\mathbb{R}} \to \overline{\mathbb{R}}$ is defined by $f(x) = 0 \vee \left(\frac{1}{2} \cdot x + 1 \wedge 10\right)$ for all $x \in \overline{\mathbb{R}}$. Observe that $f(x) = \min \{\pi_1(x), \pi_2(x)\}$ for all $x \in \overline{\mathbb{R}}$, where

$$\pi_1(x) = 0 \vee \frac{1}{2} \cdot x + 1 \text{ and} \qquad \pi_2(x) = 0 \vee 10 = 10 \qquad \text{for all } x \in \overline{\mathbb{R}}.$$

Hence, π_1 and π_2 are the $\wedge$-strategies for f.

Moreover, observe that $f(x) = \max \{\sigma_1(x), \sigma_2(x)\}$ for all $x \in \overline{\mathbb{R}}$, where

$$\sigma_1(x) = 0 \text{ and} \qquad \sigma_2(x) = \frac{1}{2} \cdot x + 1 \wedge 10 \qquad \text{for all } x \in \overline{\mathbb{R}}.$$

Hence, σ_1 and σ_2 are the $\vee$-strategies for f. □

Since $f(x) = \min \{\pi(x) \mid \pi \in \Pi\}$ for all $x \in \mathbb{D}$, we get

$$\mu f = \min \{\mu\pi \mid \pi \in \Pi\} \tag{8}$$

using Knaster-Tarski's fixpoint theorem (recall that μf denotes the least fixpoint of f). This in particular implies that there exists a $\wedge$-strategy $\pi \in \Pi$ such that $\mu f = \mu\pi$.

However, if $f(x) = \max \{\sigma(x) \mid \sigma \in \Sigma\}$ for all $x \in \mathbb{D}$, then we can only conclude that $\nu f = \max \{\nu\sigma \mid \sigma \in \Sigma\}$. This is the dual of (8). We are not able to conclude $\mu f = \max \{\mu\sigma \mid \sigma \in \Sigma\}$, since this statement is simply not always true:

Example 7. We continue Example 6. We have

$$\mu f = 2 = \min \{2, 10\} = \min \{\mu\pi_1, \mu\pi_2\}$$

On the other hand we have

$$\mu f = 2 > 0 = \max\{0, -\infty\} = \max\{\mu\sigma_1, \mu\sigma_2\}$$

Hence, if we are able to compute the least fixpoints of the "simpler" self-maps π_1 and π_2, then we can compute the least fixpoint of the self-map f. However, the ability of computing the least fixpoints of the "simpler" self-maps σ_1 and σ_2 does not give us the ability of computing the least fixpoint of f. □

If we assume that Π is finite and we can compute $\mu\pi$ for every $\pi \in \Pi$, we immediately obtain a method for computing μf. However, this does not necessarily lead to a practical algorithm, since the cardinality of Π is usually large, for instance, exponential in the size of the input. For tackling this problem, the idea is to start with an arbitrary $\wedge$-strategy π_0 and improve this $\wedge$-strategy iteratively utilizing the assumption that $f(x) = \min\{\pi(x) \mid \pi \in \Pi\}$ holds for all x. This idea can be formalized as follows:

Algorithm 1 The $\wedge$-Strategy Improvement Algorithm

1. *Initialization.* Set $k = 0$ and select any $\wedge$-strategy $\pi^{(0)} \in \Pi$.
2. *Value determination.* Compute the least fixpoint $x^{(k)} := \mu\pi^{(k)}$ of $\pi^{(k)}$.
3. If $x^{(k)} = f(x^{(k)})$, then return $x^{(k)}$.
4. $\wedge$-*Strategy Improvement.* Take $\pi^{(k+1)} \in \Pi$ such that $f(x^{(k)}) = \pi^{(k+1)}(x^{(k)})$. Increment k by 1 and goto Step 2.

Using Knaster-Tarski's fixpoint theorem, we can prove the following statements:

1. $(x^{(i)})_{i\in\mathbb{N}}$ is a decreasing sequence of post-fixpoints of f (i.e., $x^{(i)} \geq f(x^{(i)})$ for all $i \in \mathbb{N}$) that is strictly decreasing until it is stable.
2. If it is stable, then we have found a solution, i.e., a fixpoint of f.
3. $x^{(i)}$ is greater than or equal to the least solution for all $i \in \mathbb{N}$, i.e., $x^{(i)} \geq \mu f$ for all $i \in \mathbb{N}$.
4. The sequence $(x^{(i)})_i$ is bounded from above by the sequence obtained by Kleene iteration, i.e., $x^{(i)} \leq f^i(x^{(0)})$ for all $i \in \mathbb{N}$.
5. If the set Π of all $\wedge$-strategies is finite, then termination is guaranteed after at most $|\Pi|$ steps.

Example 8. We again continue Example 6. Assume that $\pi^{(0)} = \pi_2$. Then we get $x^{(0)} = \mu\pi^{(0)} = \mu\pi_2 = 10$. We observe that $x^{(0)}$ is not a solution of f, because $x^{(0)} = 10 > 6 = f(10) = f(x^{(0)})$. Hence, we improve the current $\wedge$-strategy. For that we observe that

$$\pi_1(x^{(0)}) = \pi_1(10) = 6 = f(x^{(0)}) < 10 = \pi_2(10) = \pi_2(x^{(0)}).$$

Hence, the algorithm chooses $\pi^{(1)} = \pi_1$ as the next $\wedge$-strategy. Thus, we get $x^{(1)} = \mu\pi^{(1)} = \mu\pi_1 = 2$. As we will see in the following, in this case we can use linear programming in order to compute $\mu\pi_1$. Since $f(x^{(1)}) = f(2) = 2 = x^{(1)}$ holds, we have found a fixpoint of f. Hence, the algorithm terminates. In this case we have even found the least fixpoint μf of f. □

In the above example, we have found the least fixpoint μf of f. However, algorithm 1 stops whenever some fixpoint $x^{(k)}$ is reached, not necessarily the least one. We give a simple example for this phenomenon:

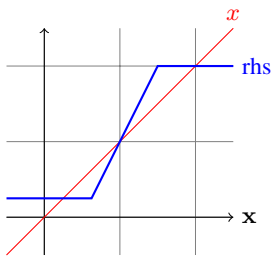


Fig. 5. The graphs of $\mathbf{x}$ and $(0.25 \vee 2 \cdot \mathbf{x} - 1) \wedge 2$

Example 9. Consider the following system of concave equations:

$$\mathbf{x} = (0.25 \vee 2 \cdot \mathbf{x} - 1) \wedge 2$$

The graph of the left hand-side $\mathbf{x}$ and the graph of the right-hand side $(0.25 \vee 2 \cdot \mathbf{x} - 1) \wedge 2$ are drawn in Figure 5. The least solution of the above system of concave equations is $\mathbf{x} = 0.25$. The set $\Pi = \{\pi_1, \pi_2\}$ of $\wedge$-strategies for the right-hand side is given by $\pi_1(x) = (0.25 \vee 2 \cdot x - 1)$ and $\pi_2(x) = 2$. If we initiliaze the $\wedge$-strategy iteration with the $\wedge$-strategy $\pi^{(0)} = \pi_2$, the algorithm returns 2, since the $\wedge$-strategy π_2 cannot be improved further, since $\pi_1(2) = 3 > 2 = \mu\pi_2$. Unfortunately, 2 is not the least solution. The problem here stems from the fact that the function π_1 is not non-expansive in the sup-norm, i.e., it does not hold $\|f(x) - f(y)\|_\infty \leq \|x - y\|_\infty$ for all $x, y \in \mathbb{R}$ (cf. Adjé et al. [1]). □

Although minimality of the obtained solution cannot be guaranteed in the general case, there are indeed important cases where minimality can be guaranteed by an enhanced $\wedge$-strategy improvement step. Adjé et al. [1] describe how to guarantee minimality for the case that all mappings are non-expansive.

One notable advantage of the $\wedge$-strategy method is that it can be stopped at anytime with a safe over-approximation of the least fixpoint. It thus can give us save results, even if the set of $\wedge$-strategies is infinite.

In the following we specialize our $\wedge$-strategy improvement algorithm to an algorithm for solving systems of concave equations as introduced in Section 4. The challenge here is to identify the set of $\wedge$-strategies.

5.2 Least Fixpoints for Max-Affine Self-Maps

In this subsection, we briefly explain how to compute the least solution of a system $\mathcal{C}$ of in-equations of the form $\mathbf{x}_i \geq f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, where $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are distinct variables

and $f : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}$ is monotone and affine. A function $f : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}$ is monotone and affine iff there exist $c_0 \in \overline{\mathbb{R}}$ and $c_1, \ldots, c_n \in \mathbb{R}_{\geq 0}$ such that $f(x_1, \ldots, x_n) = c_0 + c_1 \cdot x_1 + \cdots + c_n \cdot x_n$ for all $x_1, \ldots, x_n \in \overline{\mathbb{R}}$. Here, we use the convention $-\infty + \infty = -\infty$.

For simplicity we assume that the least solution ρ^* of $\mathcal{C}$ maps every variable to a value that is strictly greater than $-\infty$, i.e., $\rho^*(\mathbf{x}) > -\infty$ for all variable $\mathbf{x}$. We can do so w.o.l.g., since we can determine the variables that are $-\infty$ in the least solution by performing n Kleene iteration steps. Then we can remove these variables and the corresponding in-equations from the system of in-equations.

Let $\mathbf{X}$ denote the set of variables, and $G = (\mathbf{X}, \to)$ be the *variable dependency graph* of the system $\mathcal{C}$ of in-equations, i.e., the nodes of G are the variables of $\mathcal{E}$ and we write $\mathbf{x}_i \to \mathbf{x}_j$ iff $\mathbf{x}_i = \infty$ implies $\mathbf{x}_j = \infty$, i.e., if there exists an in-equation $\mathbf{x}_j \geq f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ with $f(x_1, \ldots, x_n) = c_0 + c_1 \cdot x_1 + \cdots + c_n \cdot x_n$, where $c_0 > -\infty$ and $c_i > 0$. If G is strongly connected, then the least solution of $\mathcal{E}$ can be determined by solving the following linear programming problem:

$$\min \sum_{i=1}^{n} \mathbf{x}_i \quad \mathbf{y} \geq f(\mathbf{x}_1, \ldots, \mathbf{x}_n) \quad \text{for all in-equations } \mathbf{y} \geq f(\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathcal{C}$$

The above linear program aims at minimizing the sum of all variables $\mathbf{x} \in \mathbf{X}$. The feasible space is simply the set of all solutions of the system $\mathcal{C}$ of in-equations. If this linear program is infeasible, then $\rho^*(\mathbf{x}) = \infty$ holds for all variables $\mathbf{x} \in \mathbf{X}$. If this linear program is feasible, then ρ^* is the uniquely determined optimal solution:

Lemma 6. *If the variable dependency graph of $\mathcal{C}$ is strongly connected, then the least solution of $\mathcal{C}$ can be computed by solving a linear programming problem that can be constructed in linear time.*

For computing the least solution in case that the variable dependency graph of $\mathcal{C}$ is not strongly connected we divide the system of in-equations into strongly connected components. We start with an arbitrary non-trivial strongly connected component without incomming edges. Thus, according to Lemma 6, we can compute the least solution of the induced system of in-equations by solving a linear programming problem that can be constructed in linear time. After we have determined the values for this strongly connected component, we can replace these variables with their values. We can repeat the above procedure until all strongly connected components are solved. Since the number of strongly connected components is bounded by the number of variables, we have shown the following:

Theorem 1 (Gaubert et al. [9]). *The least solution of a system of in-equations of the form $\mathbf{x}_i \geq f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, where f is a monotone and affine operator, can be computed by solving linearly many linear programming problems, each of which can be constructed in linear time. Thus, it can be computed in polynomial time.* □

Example 10. We consider the following system of in-equations:

$$\mathbf{x}_1 \geq -10 \quad \mathbf{x}_1 \geq \frac{1}{4} \cdot \mathbf{x}_2 + 1 \quad \mathbf{x}_2 \geq 2 \cdot \mathbf{x}_1 \quad \mathbf{x}_3 \geq \mathbf{x}_3 + \mathbf{x}_1 - 1 \quad \mathbf{x}_3 \geq 0$$

Our goal is to compute the least solution using the method presented in this subsection. The strongly connected components of the variable dependency graph are $\{\mathbf{x}_1, \mathbf{x}_2\}$ and $\{\mathbf{x}_3\}$. The variables $\mathbf{x}_1$ and $\mathbf{x}_2$ do not depend on the variable $\mathbf{x}_3$. Thus, in the first step we have to compute the uniquely determined optimal solution of the following linear programming problem:

$$\min \mathbf{x}_1 + \mathbf{x}_2 \qquad \mathbf{x}_1 \geq -10 \qquad \mathbf{x}_1 \geq \frac{1}{4} \cdot \mathbf{x}_2 + 1 \qquad \mathbf{x}_2 \geq 2 \cdot \mathbf{x}_1$$

The uniquely determined optimal solution gives us $\mathbf{x}_1 = 2$ and $\mathbf{x}_2 = 4$. After substituting the variables $\mathbf{x}_1$ and $\mathbf{x}_2$ with their values, it remains to compute the least solution of the following system of in-equations:

$$\mathbf{x}_3 \geq \mathbf{x}_3 + 2 - 1 \vee 0$$

Thus, we have to determine the uniquely determined optimal solution of the following linear programming problem:

$$\min \mathbf{x}_3 \qquad \mathbf{x}_3 \geq \mathbf{x}_3 + 1 \qquad \mathbf{x}_3 \geq 0$$

This linear programming problem is infeasible. Thus, we get $\mathbf{x}_3 = \infty$. Hence, the least solution of the original system of in-equations is $\mathbf{x}_1 = 2$, $\mathbf{x}_2 = 4$, and $\mathbf{x}_3 = \infty$. □

When we use interior point methods for solving the linear programming problems, we obtain a polynomial-time algorithm. However, the number of arithmetic operations and memory accesses then depends on the sizes of the occurring numbers. Thus, the algorithm is not uniform. A uniform polynomial-time algorithm is not known.

5.3 $\wedge$-Strategies for Systems of Concave Equations

We aim at specializing our $\wedge$-strategy improvement algorithm to an algorithm for solving systems of concave equations as introduced in Section 4. For that, let us consider the following system of concave equations:

$$\begin{aligned}
\mathbf{x}_1 &= f_{1,1}(\mathbf{x}_1, \ldots, \mathbf{x}_n) \vee \cdots \vee f_{1,k_1}(\mathbf{x}_1, \ldots, \mathbf{x}_n) \\
&\vdots \\
\mathbf{x}_n &= f_{n,1}(\mathbf{x}_1, \ldots, \mathbf{x}_n) \vee \cdots \vee f_{1,k_n}(\mathbf{x}_1, \ldots, \mathbf{x}_n)
\end{aligned}$$

Firstly, we have to define an adequate set Π of $\wedge$-strategies for the function

$$f = \begin{pmatrix} f_{1,1} \vee \cdots \vee f_{1,k_1} \\ \vdots \\ f_{n,1} \vee \cdots \vee f_{n,k_n} \end{pmatrix} : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}^n,$$

where we expect that we can compute the least fixpoint $\mu\pi$ for every $\wedge$-strategy $\pi \in \Pi$ efficiently. We proceed as follows: We define the set $\mathcal{T}_{k,1}$ as the smallest set that contains the following functions:

1. All monotone and affine functions $f : \overline{\mathbb{R}}^k \to \overline{\mathbb{R}}$, i.e. functions of the form $f(x) = a^\top x + b$ with $a \in \mathbb{R}^k_{\geq 0}$ and $b \in \overline{\mathbb{R}}$.
2. For all $y \in \overline{\mathbb{R}}^k$, the function $f_y : \overline{\mathbb{R}}^k \to \overline{\mathbb{R}}$ that is defined by

$$f_y(x) = \begin{cases} -\infty & \text{if } x \leq y \\ \infty & \text{if } x > y \end{cases} \qquad \text{for all } x \in \overline{\mathbb{R}}^k.$$

The set $\mathcal{T}^\vee_{k,1}$ is then defined by $\mathcal{T}^\vee_{k,1} := \{f_1 \vee \cdots \vee f_l \mid f_1, \ldots, f_l \in \mathcal{T}_{k,1}\}$. The set $\mathcal{T}^\vee_{k,m}$ is finally defined by $\mathcal{T}^\vee_{k,m} := \{(f_1, \ldots, f_m)^\top \mid f_1, \ldots, f_m \in \mathcal{T}^\vee_{k,1}\}$.

Let $f = (f_1, \ldots, f_n)^\top$, where $f_1, \ldots, f_n : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}$ are monotone and concave. The set $\mathit{\Pi}$ of $\wedge$-strategies for f is defined as follows:

$$\mathit{\Pi} = \{\pi : \mathcal{T}^\vee_{n,n} \mid \pi \geq f\}.$$

Each $\wedge$-strategy $\pi \in \mathit{\Pi}$ is a finite maximum of affine functions. It is thus in particular convex. As we have seen in Section 5.2, the least fixpoint $\mu\pi$ of a $\wedge$-strategy π can be computed through linear programming (cf. Gaubert et al. [9]). Because $f \leq \pi$ holds for all $\pi \in \mathit{\Pi}$, it follows $\mu f \leq \mu\pi$ for all $\pi \in \mathit{\Pi}$. Moreover, because each component of f is a maximum of finitely many concave functions, it follows that there indeed exists a π with $\mu\pi = \mu f$. Hence, $\mathit{\Pi}$ admits lower selections. We have $\mu f = \min\{\mu\pi \mid \pi \in \mathit{\Pi}\}$ as desired.

Note that $\mathit{\Pi}$ can be indeed infinite. However, in some cases there exists a finite subset $\mathit{\Pi}'$ of $\mathit{\Pi}$ such that $f(x) = \min\{\pi(x) \mid \pi \in \mathit{\Pi}'\}$ for all $x \in \overline{\mathbb{R}}^n$ (cf. Gaubert et al. [9]). Then we can restrict our considerations to this finite subset.

5.4 Improving $\wedge$-Strategies

In order to use Algorithm 1, it remains to explain how we can realize the $\wedge$-strategy improvement step (Step 4). We assume that we have given a post-fixpoint x of f, i.e., $x \geq f(x)$. Our goal is to compute a $\wedge$-strategy $\pi \in \mathit{\Pi}$ such that $\pi(x) = f(x)$, i.e., π is locally optimal at x. In other words: we have to find a lower selection.

For that, we assume that for any monotone and concave function $f : \overline{\mathbb{R}}^k \to \overline{\mathbb{R}}$ and any $x \in (\mathbb{R} \cup \{\infty\})^k$ with $-\infty < f(x) < \infty$, $T_{f,x} : \overline{\mathbb{R}}^k \to \overline{\mathbb{R}}$ denotes a monotone and affine function such that

$$T_{f,x} \geq f, \text{ and} \qquad\qquad T_{f,x}(x) = f(x).$$

The function $T_{f,x}$ must exist due to the monotonicity and concavity of f. For the sake of simplicity of the presentation we omit the case that $x \notin (\mathbb{R} \cup \{\infty\})^k$. For all $x \in \overline{\mathbb{R}}^k$ with $f(x) = \infty$, we additionally set $T_{f,x}(y) = \infty$ for all $y \in \overline{\mathbb{R}}^k$, and for all $x \in \overline{\mathbb{R}}^k$ with $f(x) = -\infty$, we set

$$T_{f,x}(y) = \begin{cases} -\infty & \text{if } y \leq x \\ \infty & \text{if } y > x \end{cases}$$

For $f = f_1 \vee \cdots \vee f_k$ with $f_1, ..., f_k : \overline{\mathbb{R}}^k \to \overline{\mathbb{R}}$ monotone and concave, we set

$$T_{f,x} := T_{f_1,x} \vee \cdots \vee T_{f_k,x} \qquad \text{for all } x \in \overline{\mathbb{R}}^k.$$

Moreover, for $f = (f_1, \ldots, f_n)^T$ with $f_1, \ldots, f_n : \overline{\mathbb{R}}^k \to \overline{\mathbb{R}}$ we set

$$T_{f,x} = (T_{f_1,x}, \ldots, T_{f_n,x})^\top$$

Then, we have:

Lemma 7. *$T_{f,x} \geq f$ and $T_{f,x}(x) = f(x)$ for all $x \in \overline{\mathbb{R}}^n$.* □

Therefore, $T_{f,x}$ is a $\wedge$-strategy for f that is locally optimal at x. However, it remains to answer the question how $T_{f,x}$ can be computed.

Let $f : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}$ be concave, and $x \in \overline{\mathbb{R}}^n$ with $-\infty < f(x) < \infty$. For simplicity we assume that $x \in \mathbb{R}^n$. We can do so w.l.o.g., since we can remove the other components and treat them separately. Our goal is to compute an affine function $T_{f,x} : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}$ such that $T_{f,x}(x) = f(x)$ and $T_{f,x} \geq f$. Hence,

$$T_{f,x}(y) = f(x) + d^\top (y - x) \qquad \text{for all } y \in \mathbb{R}^n$$

Hence, we are searching for a $d \in \mathbb{R}^n$ such that

$$g(d) := \sup \{g(d, y) \mid y \in \mathbb{R}^n\} \leq f(x)$$

holds, where $g(d, y) := f(y) - d^\top (y - x)$. The function g is convex, since it is the supremum of a set of affine functions. For computing the value $g(d)$ for a given $d \in \mathbb{R}^n$, we have to solve an unconstraint convex optimization problem, because $g(d, \cdot)$ is concave and thus $-g(d, \cdot)$ is convex. However, this does not lead to an practical method for our application mentioned in Section 3.

For our application described in Section 3, we have to consider the case that $f : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}$ is given by

$$f(x) = \sup \{C \bullet X \mid X \succeq 0, \mathcal{A}(X) = a, \mathcal{B}(X) \leq x\},$$

i.e., $f(x)$ is given by the optimal value of the following SDP problem:

$$\max_X C \bullet X \qquad \mathcal{A}(X) = a \qquad \mathcal{B}(X) \leq x \qquad X \succeq 0$$

Note that f is monotone. We moreover assume that $-\infty < f(x) < \infty$ holds. For simplicity we again assume that $x \in \mathbb{R}^n$. We can deal with the other cases by removing constraints from the semi-definite programming problem. We aim at using the dual problem in order to compute $T_{f,x}$. The dual problem (see e.g. Todd [22]) is given by:

$$\min_{\lambda,\mu} x^\top \lambda + a^\top \mu \qquad \mathcal{B}\lambda + \mathcal{A}\mu \succeq C \qquad \lambda \geq 0$$

Here, the column vectors λ and μ are the variables. Let $d(x)$ denote the optimal value of the dual problem. Weak duality gives us $f(x) \leq d(x)$. By assumption we in particular have $-\infty < d(x)$.

Now we define a hyperplane $T_{f,x}$ as follows. If $d(x) = \infty$, i.e., if the dual is infeasible, then we set

$$T_{f,x}(z) = \infty \qquad \text{for all } z \in \overline{\mathbb{R}}^n$$

Finally, assume $-\infty < d(x) < \infty$. If the dual has an optimal solution (λ, μ), then we define the hyperplane $T_{f,x}$ by

$$T_{f,x}(z) = \lambda^\top z + \mu^\top a \qquad \text{for all } z \in \overline{\mathbb{R}}^n$$

If the dual has no optimal solution, we in practice simply choose a good feasible solution. Weak duality gives us $T_{f,x} \geq f$.

Observe that $T_{f,x}$ is monotone, because $\lambda \geq 0$. In order to conclude $T_{f,x}(x) = f(x)$, we however need stronger assumptions. For instance assumptions that imply strong duality. One sufficient criterion for strong duality and the existence of an optimal solution for the dual problem is that all components of $\mathcal{A}$ are linearly independent and $\{X \succ 0 \mid \mathcal{A}(X) = a,\ \mathcal{B}(X) \lhd x\} \neq \emptyset$ (cf. Todd [22]).

The result of the above discussion can be summarized as follows: we can compute a self-map $T_{f,x}$ that fulfills the statements of Lemma 7 (i.e., $T_{f,x} \geq f$ and $T_{f,x}(x) = f(x)$) through semi-definite programming, whenever the above sufficient condition for strong duality is fulfilled.

5.5 The Harmonic Oscillator

We continue Example 4 (page 15), i.e., in order to analyze the harmonic oscillator we aim at solving the following systems of equations:

$$\begin{aligned}
\mathbf{x}_{\mathbf{st},p_1} &= 0 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_1)\\
\mathbf{x}_{\mathbf{st},p_2} &= 1 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_2)\\
\mathbf{x}_{\mathbf{st},p_3} &= 0 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_3)\\
\mathbf{x}_{\mathbf{st},p_4} &= 1 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_4)\\
\mathbf{x}_{\mathbf{st},p_5} &= 7 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_5)
\end{aligned}$$

We emphasize that the right-hand sides are finite maximums of monotone and concave functions. It is easy to verify that $\mathbf{x}_{\mathbf{st},p_1} = \cdots = \mathbf{x}_{\mathbf{st},p_4} = \infty$, $\mathbf{x}_{\mathbf{st},p_5} = 7$ is a post-solution. In oder to simplify notations, let $c_1 = c_3 = 0$, $c_2 = c_4 = 1$, $c_5 = 7$, and, for all $i \in \{1, \ldots, 5\}$, $f_i : \overline{\mathbb{R}}^n \to \overline{\mathbb{R}}$ be defined by

$$f_i((x_1, \ldots, x_5)^\top) := c_i \vee ([\![s]\!]^{\mathcal{R}}\{p_i \mapsto x_i \mid i \in \{1, \ldots, 5\}\})(p_i)$$

Let moreover $f = (f_1, \ldots, f_5)^\top$, i.e., f denotes the right-hand side of the above system of equations. If we evaluate the right-hand sides, we get

$$f((\infty, \infty, \infty, \infty, 7)^\top) \simeq (2.0426, 2.0426, 1.6651, 1.6651, 7)^\top.$$

For evaluating the right-hand sides, we can use semi-definite programming. The usual implementations solve the primal and the dual problem at the same time. From a dual optimal solution, we obtain our first $\wedge$-strategy $\pi^{(0)}$ that is given as follows:

$$\pi^{(0)} := T_{f,(\infty,\infty,\infty,\infty,7)^\top}((x_1,\ldots,x_5)^\top) \simeq \begin{pmatrix} 0 \vee 0.14588 \cdot x_5 + 1.0214 \\ 1 \vee 0.14588 \cdot x_5 + 1.0214 \\ 0 \vee 0.11892 \cdot x_5 + 0.83263 \\ 1 \vee 0.11892 \cdot x_5 + 0.83263 \\ 7 \vee 0.99456 \cdot x_5 \end{pmatrix}$$

We are now going to explain how we have obtained the first component. According to the findings from Example 4, we have

$$\begin{aligned} &([\![s]\!]^{\mathcal{R}}\{p_1 \mapsto \infty, p_2 \mapsto \infty, p_3 \mapsto \infty, p_4 \mapsto \infty, p_5 \mapsto 7\})(p_1) \\ =&\sup\{C_1 \bullet X \mid X \succeq 0, X_{1\cdot 1} = 1, B_5 \bullet X \leq 7\} \\ =&\sup\left\{ \begin{pmatrix} 0 & -0.5 & -0.005 \\ -0.5 & 0 & 0 \\ -0.005 & 0 & 0 \end{pmatrix} \bullet X \mid X \succeq 0, X_{1\cdot 1} = 1, \begin{pmatrix} 0&0&0 \\ 0&2&1 \\ 0&1&3 \end{pmatrix} \bullet X \leq 7 \right\} \\ =&2.0426 \end{aligned}$$

We have seen in Subsection 5.4, that, in order to compute an affine over-approximation of $([\![s]\!]^{\mathcal{R}}\{p_i \mapsto x_i \mid i \in \{1,\ldots,5\}\})(p_i)$ that is exact at $x_1 = x_2 = x_3 = x_4 = \infty$ and $x_5 = 7$, we can solve the dual problem that is given as follows:

$$\begin{aligned} &\inf\left\{ 7\lambda + \mu \mid \lambda \geq 0,\ \mu \in \mathbb{R},\ \lambda B_5 + \mu \begin{pmatrix} 1&0&0 \\ 0&0&0 \\ 0&0&0 \end{pmatrix} \succeq C_1 \right\} \\ =&\inf\ \{7\lambda + \mu \mid \lambda \geq 0,\ \mu \in \mathbb{R}, \\ &\qquad\qquad \lambda \begin{pmatrix} 0&0&0 \\ 0&2&1 \\ 0&1&3 \end{pmatrix} + \mu \begin{pmatrix} 1&0&0 \\ 0&0&0 \\ 0&0&0 \end{pmatrix} \succeq \begin{pmatrix} 0 & -0.5 & -0.005 \\ -0.5 & 0 & 0 \\ -0.005 & 0 & 0 \end{pmatrix} \Bigg\} \end{aligned}$$

Running a solver may give us the result $\lambda \simeq 0.14588$ and $\mu \simeq 1.0214$. This gives us the first component of $\pi^{(0)}$. The remaining components can be computed in the same way.

As described in Subsection 5.2 we can compute the least fixpoint of $\pi^{(0)}$ through linear programming. We get

$$x^{(0)} := \mu\pi^{(0)} = (2.0426, 2.0426, 1.6651, 1.6651, 7)^\top.$$

Then, by again solving semi-definite and linear programming problems, we get

$$\pi^{(1)} := T_{f,x^{(0)}}((x_1,\ldots,x_5)^\top) \simeq \begin{pmatrix} 0 \vee 0.90541 \cdot x_1 + 0.01340 \cdot x_5 + 0.093820 \\ 1 \vee 0.90541 \cdot x_2 + 0.01340 \cdot x_5 + 0.093819 \\ 0 \vee 0.88297 \cdot x_3 + 0.01346 \cdot x_5 + 0.094205 \\ 1 \vee 0.88297 \cdot x_4 + 0.01346 \cdot x_5 + 0.094205 \\ 7 \vee 0.99456 \cdot x_5 \end{pmatrix}$$

and

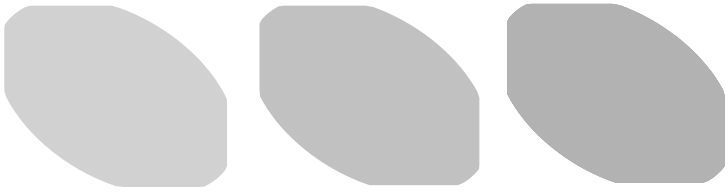
$$x^{(1)} := \mu\pi^{(1)} \simeq (1.9838, 1.9838, 1.6098, 1.6098, 7.0000)^\top.$$

Continuing this process, we find:

$$\begin{aligned}
x^{(2)} &:= \mu\pi^{(2)} \simeq (1.8971, 1.8971, 1.5434, 1.5434, 7.0000)^\top \\
x^{(3)} &:= \mu\pi^{(3)} \simeq (1.8718, 1.8718, 1.5280, 1.5280, 7.0000)^\top \\
x^{(4)} &:= \mu\pi^{(4)} \simeq (1.8708, 1.8708, 1.5275, 1.5275, 7.0000)^\top \\
x^{(5)} &:= \mu\pi^{(5)} \simeq (1.8708, 1.8708, 1.5275, 1.5275, 7.0000)^\top
\end{aligned}$$

$\{-2.0462 \leq x \leq 2.0426,\ -1.665 \leq v \leq 1.665,\ 2x^2 + 3v^2 + 2xv \leq 7\}$

$\{-1.9838 \leq x \leq 1.9838,\ -1.6097 \leq v \leq 1.6097,\ 2x^2 + 3v^2 + 2xv \leq 7\}$



$\{-1.8971 \leq x \leq 1.8971,\ -1.5435 \leq v \leq 1.5435,\ 2x^2 + 3v^2 + 2xv \leq 7\}$

$\{-1.8718 \leq x \leq 1.8718,\ -1.5275 \leq v \leq 1.5275,\ 2x^2 + 3v^2 + 2xv \leq 7\}$

$\{-1.8708 \leq x \leq 1.8708,\ -1.5275 \leq v \leq 1.5275,\ 2x^2 + 3v^2 + 2xv \leq 7\}$

Fig. 6. Visualization of a run of our $\wedge$-strategy iteration algorithm for the harmonic oscillator from Section 2

Our $\wedge$-strategy iteration stabilizes after a few iterations. The run of our $\wedge$-strategy improvement algorithm is visualized in Figure 6. As a result, we obtain

$$\mu f \leq (1.8708, 1.8708, 1.5275, 1.5275, 7.0000)^\top.$$

As a result we finally obtain that the following invariants hold at program point $\mathbf{st}$ of the harmonic oscillator (page 4):

$$\begin{aligned}
&-x_1 \leq 1.8708 && x_1 \leq 1.8708 && -x_2 \leq 1.5275 \\
&x_2 \leq 1.5275 && 2x_1^2 + 3x_2^2 + 2x_1x_2 \leq 7
\end{aligned}$$

6 The Max-Strategy Iteration Approach

Before giving a formal description of the max-strategy iteration approach in Subsection 6.2, we explain it by a simple example in Subsection 6.1. In Subsection 6.3 we finally apply the max-strategy iteration approach to the harmonic oscillator introduced in Section 2.

6.1 A Simple Example

Our goal is to compute the least solution of the following equation system:

$$\mathbf{x} = 0.4 \vee \sqrt{\mathbf{x}} \vee 1 + \sqrt{\mathbf{x} - 1}$$

Here, $\sqrt{x} = \sup\,\{y \in \overline{\mathbb{R}} \mid y^2 \leq x\}$. Note that, for all $x < 0$, $\sqrt{x} = \sup \emptyset = -\infty$. The important property is that all right-hand sides are finite maximums of monotone and concave functions. The graph of the left-hand side $\mathbf{x}$ and the graph of the right-hand side $0.4 \vee \sqrt{\mathbf{x}} \vee 1 + \sqrt{\mathbf{x} - 1}$ are drawn in Figure 7.(a). The least solution is the least x-coordinate where the two graphs cross, i.e., it is given by 1.

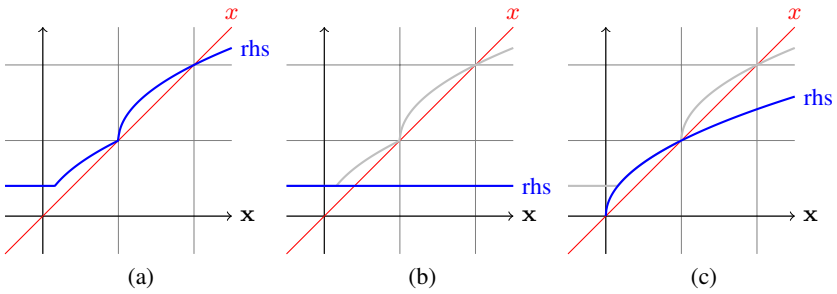


Fig. 7. A run of the $\vee$-strategy improvement algorithm

We now use the $\vee$-strategy improvement algorithm of Gawlitza and Seidl [10, 11, 13, 14] for finding the least solution. We consider the computation of the least solution as a competition between a maximizer and a minimizer. The maximizer aims at maximizing the least solution whereas the minimizer aims at minimizing it. The state of the game is the current approximate to the least solution. The play starts with the approximate that assigns $-\infty$ to every variable. At some approximate the maximizer is allowed to select an argument of the finite maximum $0.4 \vee \sqrt{\mathbf{x}} \vee 1 + \sqrt{\mathbf{x} - 1}$, for instance $\sqrt{\mathbf{x}}$. Such a selection is called a $\vee$-*strategy*.

The play starts at the approximate $-\infty$. This is the current state of the play at the beginning. At this point, the most profitable $\vee$-strategy is the argument 0.4, since $\sqrt{\mathbf{x}}$ and $1 + \sqrt{\mathbf{x} - 1}$ evaluate to $-\infty$. The play proceeds by performing a least fixpoint

iteration starting at $-\infty$ using the current $\vee$-strategy, i.e., the next approximate is the least solution of the equation system

$$\mathbf{x} = 0.4$$

that exceeds $-\infty$. Hence, the next approximate is 0.4 (cf. Figure 7.(b)). Note that 0.4 is not only the least solution of the equation system $\mathbf{x} = 0.4$ that exceeds 0.4, but it is also the greatest solution of the inequation $\mathbf{x} \leq 0.4$, i.e., the greatest point in the convex area that is above the graph of the left hand-side and below the graph of the concave right-handside (cf. Figure 7.(b)). This is not by accident. We will see that during a run of our $\vee$-strategy improvement algorithm, the next approximate is always the greatest point of such a convex area. This implies that it can be computed through algorithms for solving convex optimization problems.

Now, we try to improve the current $\vee$-strategy locally at 0.4. Since $\sqrt{0.4} > 0.4$ holds, we can improve the current $\vee$-strategy to the $\vee$-strategy that selects the argument $\sqrt{\mathbf{x}}$.[8] This gives us a *strict local improvement*. Thus, the next approximate is the least solution of the equation system

$$\mathbf{x} = \sqrt{\mathbf{x}}$$

that exceeds 0.4. Hence, the next approximate is 1 (cf. Figure 7.(c)). It is again the greatest solution of the inequation system $\mathbf{x} \leq \sqrt{\mathbf{x}}$. Thus, it is the uniquely determined optimal solution of the following convex optimization problem:

$$\max \mathbf{x} \qquad\qquad \mathbf{x}^2 - \mathbf{x} \leq 0$$

In this case the unique optimal solution can for instance be computed through semi-definite programming, because it is a *convex quadratic optimization problem*.

Now, our current approximate is 1 and our current $\vee$-strategy selects the argument $\sqrt{\mathbf{x}}$. We now again try to improve our current $\vee$-strategy, i.e., we search for a $\vee$-strategy that is strictly more profitable at our current approximate 1 than our current $\vee$-strategy. Since $0.4 < 1 = 1 + \sqrt{1-1} = 1 = \sqrt{1}$ holds, there is no such $\vee$-strategy. In other words: The current $\vee$-strategy cannot be improved at the current approximate (cf. Figure 7.(c)). This means: We have found a solution of the equation system

$$\mathbf{x} = 0.4 \vee \sqrt{\mathbf{x}} \vee 1 + \sqrt{\mathbf{x} - 1}.$$

Since the sequence of approximates is monotonically increasing and bounded by the least solution, we in fact have found the least solution. For short: The $\vee$-strategy improvement algorithm terminates and returns the least solution 1.

6.2 The Max-Strategy Improvement Algorithm

In this section we are going to compute least solutions of systems of concave equations through the $\vee$-strategy improvement algorithm of Gawlitza and Seidl [10, 11, 12, 14].

[8] Since $1 + \sqrt{0.4 - 1} = 1 + \sqrt{-0.6} = 1 + -\infty = -\infty$ holds, a switch to the $\vee$-strategy that selects the argument $1 + \sqrt{\mathbf{x} - 1}$ is not profitable at the approximate 0.4.

Systems of concave equations are in particular systems of monotone equations over the *complete linearly ordered set* $\overline{\mathbb{R}}$. For the sake of generality, we subsequently consider an arbitrary complete linearly ordered set.

A $\vee$-*strategy* σ for a system $\mathcal{E}$ of monotone equations over a complete linearly ordered set is a function that maps every expression $e_1 \vee \cdots \vee e_k$ occurring in $\mathcal{E}$ to one of the immediate sub-expressions e_j, $j \in \{1, \ldots, k\}$. We denote the set of all $\vee$-strategies for $\mathcal{E}$ by $\Sigma_{\mathcal{E}}$. We drop the subscript, whenever it is clear from the context. Finally, we set

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

Example 11. For the system $\mathcal{E} = \{\mathbf{x} = \frac{1}{2} \vee \sqrt{\mathbf{x}}\}$ of concave equations and the $\vee$-strategy $\sigma = \{\frac{1}{2} \vee \sqrt{\mathbf{x}} \mapsto \frac{1}{2}\}$, we have $\mathcal{E}(\sigma) = \{\mathbf{x} = \frac{1}{2}\}$. □

Our $\vee$-strategy improvement algorithm iterates over $\vee$-strategies. It maintains a current $\vee$-strategy and a current *approximate* to the least solution. In each step, if possible, the current $\vee$-strategy is improved w.r.t. the current approximate, and a new current approximate is computed w.r.t. the new current $\vee$-strategy and the current approximate:

Definition 2 (Improvements). *Let $\mathcal{E}$ be a system of monotone equations over a complete linearly ordered set. Let $\sigma, \sigma' \in \Sigma$ be $\vee$-strategies for $\mathcal{E}$ and ρ be a pre-solution of $\mathcal{E}(\sigma)$. The $\vee$-strategy σ' is called* 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 $e_1 \vee \cdots \vee e_k$ occurring in $\mathcal{E}$ the following holds:*
   If $\sigma'(e) \neq \sigma(e)$, then $[\![\sigma'(e)]\!]\rho > [\![\sigma(e)]\!]\rho$. □

In many cases there exist several, different improvements of a $\vee$-strategy σ w.r.t. a pre-solution ρ of $\mathcal{E}(\sigma)$. Under the assumption that the operator $\vee$ is only used in its binary version, one is known as *all profitable switches* (see e.g. Björklund et al. [3, 4]). 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 definition 2 the selection at the other $\vee$-expressions must be preserved.

We can now formulate the $\vee$-strategy improvement algorithm for computing least solutions of systems of monotone equations over complete linearly ordered sets. 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 just some solution, we additionally require that $\rho_{\text{init}} \leq \mu[\![\mathcal{E}]\!]$ holds:

Algorithm 2 The $\vee$-Strategy Improvement Algorithm

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 $(\rho \notin \mathbf{Sol}(\mathcal{E}))$ {
  $\sigma \leftarrow$ improvement of σ w.r.t. ρ;
  $\rho \leftarrow \mu_{\geq\rho}[\![\mathcal{E}(\sigma)]\!]$;
}
return ρ;

Lemma 8. *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. $\rho_{i+1} = \mu_{\geq\rho_i}[\![\mathcal{E}(\sigma_{i+1})]\!]$.
4. *If* $\rho_i < \mu[\![\mathcal{E}]\!]$, *then* $\rho_{i+1} > \rho_i$.
5. *If* $\rho_i = \mu[\![\mathcal{E}]\!]$, *then* $\rho_{i+1} = \rho_i$. □

An immediate consequence of Lemma 8 is the following lemma:

Lemma 9. *Whenever the $\vee$-strategy improvement algorithm terminates, it computes the least solution $\mu[\![\mathcal{E}]\!]$ of $\mathcal{E}$.* □

In the following we are interested in solving *systems of concave equations* through our $\vee$-strategy improvement algorithm. Hence, assume that $\mathcal{E}$ is a system of concave equations. In this case our $\vee$-strategy improvement algorithm terminates and returns the least solution at the latest after considering every $\vee$-strategy at most $|\mathbf{X}|$ times.

In order to start our $\vee$-strategy improvement algorithm in a *feasible area* (see Gawlitza and Seidl [13] for detailed explanations), we start the $\vee$-strategy improvement algorithm with the system

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

the $\vee$-strategy

$$\sigma_{\text{init}} = \{e \vee -\infty \mapsto -\infty \mid \mathbf{x} = e \in \mathcal{E}\},$$

and the pre-solution $\underline{-\infty}$ of $(\mathcal{E} \vee -\infty)(\sigma_{\text{init}})$. 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. For all $i \in \mathbb{N}$, the value $\rho_{i+1} = \mu_{\geq\rho_i}[\![\mathcal{E}(\sigma_{i+1})]\!]$ is determined as follows:

Lemma 10. *Let*

$$\begin{aligned}\mathbf{X}^{-\infty} &:= \{\mathbf{x} \in \mathbf{X} \mid \mathbf{x} = e \in \mathcal{E}(\sigma_{i+1}) \text{ and } [\![e]\!]\rho_i = -\infty\}\\ \mathbf{X}^{\infty} &:= \{\mathbf{x} \in \mathbf{X} \mid \mathbf{x} = e \in \mathcal{E}(\sigma_{i+1}) \text{ and } [\![e]\!]\rho_i = \infty\}\\ \mathbf{X}' &:= \mathbf{X} \setminus (\mathbf{X}^{-\infty} \cup \mathbf{X}^{\infty})\\ \mathcal{E}' &:= \{\mathbf{x} = e \in \mathcal{E}(\sigma_{i+1}) \mid \mathbf{x} \in \mathbf{X}'\}[-\infty/\mathbf{X}^{-\infty}][\infty/\mathbf{X}^{\infty}].\end{aligned}$$

Here, $\{\mathbf{x} = e \in \mathcal{E}(\sigma_{i+1}) \mid \mathbf{x} \in \mathbf{X}'\}[-\infty/\mathbf{X}^{-\infty}][\infty/\mathbf{X}^{\infty}]$ *denotes the system of basic concave equations that is obtained from* $\mathcal{E}(\sigma_{i+1})$ *by removing all equations* $\mathbf{x} = e$ *with* $\mathbf{x} \notin \mathbf{X}'$ *and then replacing all occurrences of variables from* $\mathbf{X}^{-\infty}$ *in the right-hand sides with the constant* $-\infty$ *and all occurrences of variables from* $\mathbf{X}^{\infty}$ *in the right-hand sides with the constant* ∞*. Then*

$$\begin{aligned}\rho_{i+1}(\mathbf{x}') &= \mu_{\geq\rho_i}[\![\mathcal{E}(\sigma_{i+1})]\!](\mathbf{x}')\\ &= \mu_{\geq\rho_i|_{\mathbf{X}'}}[\![\mathcal{E}']\!](\mathbf{x}')\\ &= \sup\{\rho(\mathbf{x}') \mid \rho : \mathbf{X}' \to \mathbb{R} \text{ and } \rho(\mathbf{x}) \leq [\![e]\!]\rho \text{ for all equations } \mathbf{x} = e \in \mathcal{E}'\}\end{aligned}$$

for all $\mathbf{x}' \in \mathbf{X}'$*. Moreover,* $\rho_{i+1}(\mathbf{x}^{-\infty}) = \mu_{\geq\rho_i}[\![\mathcal{E}(\sigma_{i+1})]\!](\mathbf{x}^{-\infty}) = -\infty$ *for all* $\mathbf{x}^{-\infty} \in \mathbf{X}^{-\infty}$*, and* $\rho_{i+1}(\mathbf{x}^{\infty}) = \mu_{\geq\rho_i}[\![\mathcal{E}(\sigma_{i+1})]\!](\mathbf{x}^{\infty}) = \infty$ *for all* $\mathbf{x}^{\infty} \in \mathbf{X}^{\infty}$*,*

Hence, provided that $\mathcal{E}$ *is a system of concave equations,* ρ_{i+1} *can be computed by solving* $|\mathbf{X}|$ *convex optimization problems. Moreover, it is important to note that* ρ_{i+1} *is uniquely determined through the system* $\mathcal{E}$*, the* $\vee$*-strategy* σ_{i+1} *and the set* $\mathbf{X}^{\infty}$ *of all variables that are already known to be* ∞*.* □

Since the sequence $((\rho_i, \{\mathbf{x} \in \mathbf{X} \mid \mathbf{x} = e \in \mathcal{E}(\sigma_{i+1}) \text{ and } [\![e]\!]\rho_i = \infty\}))_i$ is strictly increasing (ordered component-wise), Lemma 10 implies that the $\vee$-strategy improvement algorithm considers each $\vee$-strategy at most $|\mathbf{X}|$ times. Thus, we have shown the following theorem:

Theorem 2. *Let* $\mathcal{E}$ *be a system of concave equations. Our* $\vee$*-strategy improvement algorithm (Algorithm 2) computes the least solution* $\mu[\![\mathcal{E}]\!]$ *of* $\mathcal{E}$ *and performs at most* $(|\Sigma| + |\mathbf{X}|) \cdot |\mathbf{X}|$ $\vee$*-strategy improvement steps. If* $\mathcal{E}$ *is a system of concave equations, we have to solve* $|\mathbf{X}|$ *convex optimization problems for every* $\vee$*-strategy improvement step.* □

Example 12. We consider the system

$$\mathcal{E} = \left\{\mathbf{x} = -\infty \vee \tfrac{1}{2} \vee \sqrt{\mathbf{x}} \vee \tfrac{7}{8} + \sqrt{\mathbf{x} - \tfrac{47}{64}}\right\}$$

of concave equations. We start with the uniquely determined $\vee$-strategy σ_0 such that

$$\mathcal{E}(\sigma_0) = \{\mathbf{x} = -\infty\}$$

and with the solution $\rho_0 := \{\mathbf{x} \mapsto -\infty\}$ of $\mathcal{E}(\sigma_0)$. Since $\rho_0 \notin \mathbf{Sol}(\mathcal{E})$, we improve the $\vee$-strategy σ_0 w.r.t. ρ_0 to a $\vee$-strategy σ_1 . Necessarily, we get

$$\mathcal{E}(\sigma_1) = \left\{\mathbf{x} = \frac{1}{2}\right\}.$$

By Lemma 10, we get

$$\rho_1(\mathbf{x}) = \mu_{\geq\rho_0}[\![\mathcal{E}(\sigma_1)]\!](\mathbf{x}) = \sup\ \left\{\mathbf{x} \mid \mathbf{x} \leq \frac{1}{2}\right\}.$$

Thus, $\rho_1 = \{\mathbf{x} \mapsto \frac{1}{2}\}$. Since $\sqrt{\frac{1}{2}} > \frac{1}{2}$ and $\frac{7}{8} + \sqrt{\frac{1}{2} - \frac{47}{64}} < \frac{1}{2}$ hold, we necessarily improve the $\vee$-strategy σ_1 w.r.t. ρ_1 to the uniquely determined $\vee$-strategy σ_2 such that

$$\mathcal{E}(\sigma_2) = \left\{\mathbf{x} = \sqrt{\mathbf{x}}\right\}.$$

Again by Lemma 10, we get

$$\rho_2(\mathbf{x}) = \mu_{\geq\rho_1}[\![\mathcal{E}(\sigma_2)]\!](\mathbf{x}) = \sup\ \left\{\mathbf{x} \mid \mathbf{x} \leq \sqrt{\mathbf{x}}\right\} = 1.$$

Thus, $\rho_2 = \{\mathbf{x} \mapsto 1\}$. Since

$$\frac{7}{8} + \sqrt{1 - \frac{47}{64}} > \frac{7}{8} + \sqrt{1 - \frac{60}{64}} = \frac{9}{8} > 1,$$

we get

$$\mathcal{E}(\sigma_3) = \left\{\mathbf{x} = \frac{7}{8} + \sqrt{\mathbf{x} - \frac{47}{64}}\right\}.$$

Again by Lemma 10, we get

$$\rho_3(\mathbf{x}) = \mu_{\geq\rho_2}[\![\mathcal{E}(\sigma_3)]\!](\mathbf{x}) = \sup\ \left\{\mathbf{x} \mid \mathbf{x} \leq \frac{7}{8} + \sqrt{\mathbf{x} - \frac{47}{64}}\right\} = 2.$$

Thus, we finally get $\rho_3 = \{\mathbf{x} \mapsto 2\}$. The algorithm terminates, because ρ_3 solves $\mathcal{E}$. Thus, $\rho_3 = \mu[\![\mathcal{E}]\!]$. We have found the least solution.

In each step we had to solve convex optimization problems that can be solved through semi-definite programming (c.f. Gawlitza and Seidl [13]). □

6.3 The Harmonic Oscillator

We continue Example 4 on page 15. After introducing $-\infty$ at the right-hand sides, we obtain the following system of concave equations:

$$\begin{aligned}
\mathbf{x}_{\mathbf{st},p_1} &= -\infty \vee 0 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_1)\\
\mathbf{x}_{\mathbf{st},p_2} &= -\infty \vee 1 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_2)\\
\mathbf{x}_{\mathbf{st},p_3} &= -\infty \vee 0 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_3)\\
\mathbf{x}_{\mathbf{st},p_4} &= -\infty \vee 1 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_4)\\
\mathbf{x}_{\mathbf{st},p_5} &= -\infty \vee 7 \vee ([\![s]\!]^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_5)
\end{aligned} \tag{9}$$

In this example we have $3^5 = 243$ different $\vee$-strategies. It is clear that the algorithm will switch to the $\vee$-strategy that is given by the finite constants in the first step. At each equation, it then can switch to the non-constant expression, but then, because it constructs a strictly increasing sequence, it can never return to the constant. Summarizing, because of the simple structure, it is clear that our $\vee$-strategy improvement algorithm will perform at most 6 $\vee$-strategy improvement steps. In fact our prototypical implementation performs 4 $\vee$-strategy improvement steps when solving this example. The last $\vee$-strategy that the algorithm considers leads to the system

$$\begin{aligned}
\mathbf{x}_{\mathbf{st},p_1} &= (\llbracket s \rrbracket^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_1)\\
\mathbf{x}_{\mathbf{st},p_2} &= (\llbracket s \rrbracket^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_2)\\
\mathbf{x}_{\mathbf{st},p_3} &= (\llbracket s \rrbracket^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_3)\\
\mathbf{x}_{\mathbf{st},p_4} &= (\llbracket s \rrbracket^{\mathcal{R}}\{p \mapsto \mathbf{x}_{\mathbf{st},p} \mid p \in P\})(p_4)\\
\mathbf{x}_{\mathbf{st},p_5} &= 7
\end{aligned}$$

of basic concave equations. The current approximate at this point of time does not assign ∞ to any variable. Because of Lemma 10, in order to determine the next value for the variable $\mathbf{x}_{\mathbf{st},p_k}$ (for $k \in \{1,\ldots,5\}$), we solve the following convex optimization problem

$$\begin{aligned}
&\sup\ \{\rho(\mathbf{x}_{\mathbf{st},p_k}) \mid \rho : \mathbf{X} \to \mathbb{R} \text{ and}\\
&\qquad \rho(\mathbf{x}_{\mathbf{st},p_i}) \leq (\llbracket s \rrbracket^{\mathcal{R}}\{q \mapsto \rho(\mathbf{x}_{\mathbf{st},q}) \mid q \in P\})(p_i), i \in \{1,\ldots,4\} \text{ and}\\
&\qquad \mathbf{x}_{\mathbf{st},p_5} \leq 7\}
\end{aligned}$$

For that, we solve the following semi-definite programming problem, that is obtained from the above convex optimization problem through unfolding the definition of the relaxed abstract semantics:

$$\begin{array}{lll}
\sup\ \mathbf{x}_{\mathbf{st},p_k} & & \\
\quad \mathbf{x}_{\mathbf{st},p_1} \leq C_k \bullet X^{(1)} & X^{(1)} \succeq 0 & \quad X^{(1)}_{1\cdot 1} = 1\\
B_1 \bullet X^{(1)} \leq \mathbf{x}_{\mathbf{st},p_1} & \cdots & \quad B_5 \bullet X^{(1)} \leq \mathbf{x}_{\mathbf{st},p_5}\\
\quad \mathbf{x}_{\mathbf{st},p_2} \leq C_k \bullet X^{(2)} & X^{(2)} \succeq 0 & \quad X^{(2)}_{1\cdot 1} = 1\\
B_1 \bullet X^{(2)} \leq \mathbf{x}_{\mathbf{st},p_1} & \cdots & \quad B_5 \bullet X^{(2)} \leq \mathbf{x}_{\mathbf{st},p_5}\\
\quad \mathbf{x}_{\mathbf{st},p_3} \leq C_k \bullet X^{(3)} & X^{(3)} \succeq 0 & \quad X^{(3)}_{1\cdot 1} = 1\\
B_1 \bullet X^{(3)} \leq \mathbf{x}_{\mathbf{st},p_1} & \cdots & \quad B_5 \bullet X^{(3)} \leq \mathbf{x}_{\mathbf{st},p_5}\\
\quad \mathbf{x}_{\mathbf{st},p_4} \leq C_k \bullet X^{(4)} & X^{(4)} \succeq 0 & \quad X^{(4)}_{1\cdot 1} = 1\\
B_1 \bullet X^{(4)} \leq \mathbf{x}_{\mathbf{st},p_1} & \cdots & \quad B_5 \bullet X^{(4)} \leq \mathbf{x}_{\mathbf{st},p_5}\\
\quad \mathbf{x}_{\mathbf{st},p_5} \leq 7 & &
\end{array}$$

The matrices $B_1,\ldots,B_5,C_1,\ldots,C_5$ are defined in Example 4 (page 15). Solving the above semi-definite programming problem gives us the final values for the variables

$\mathbf{x}_{\mathbf{st},p_1}, \ldots, \mathbf{x}_{\mathbf{st},p_5}$. We get

$$\begin{aligned}\mu[\![\mathcal{E}]\!] = \{&\mathbf{x}_{\mathbf{st},p_1} \mapsto 1.8708.., \mathbf{x}_{\mathbf{st},p_2} \mapsto 1.8708.., \\ &\mathbf{x}_{\mathbf{st},p_3} \mapsto 1.5275.., \mathbf{x}_{\mathbf{st},p_4} \mapsto 1.5275.., \mathbf{x}_{\mathbf{st},p_5} \mapsto 7\}\end{aligned}$$

Hence, the following invariants hold at program point $\mathbf{st}$ of the harmonic oscillator (see page 4):

$$\begin{array}{llll} -x_1 \leq 1.8708 & & x_1 \leq 1.8708 & -x_2 \leq 1.5275 \\ x_2 \leq 1.5275 & 2x_1^2 + 3x_2^2 + 2x_1x_2 \leq 7 & & \end{array}$$

For this example the $\vee$-strategy improvement algorithm presented in this section finds the same invariants as the $\wedge$-strategy improvement algorithm presented in section 5 (cf. Subsection 5.5).

7 Comparison and Conclusion

In this paper we discussed how we can use strategy iteration for solving systems of concave equations. This class is a natural and strict generalization of systems of rational equations (studied by Gawlitza and Seidl [10, 14]). In the context of static program analysis, such equation systems are useful for approximating the abstract semantics of programs w.r.t. quadratic templates through semi-definite relaxations. We discussed two different approaches that can be specialized for solving systems of concave equations:

For the studied case, the $\wedge$-strategy approach of Adjé et al. [2] successively approximates the given equation system by systems of affine in-equations which can be efficiently solved by linear programming. The resulting method works similar to Newton's method. For each approximate, an improved $\wedge$-strategy (a system of affine in-equations) can be efficiently determined through semi-definite programming.

As an alternative approach, we discussed the $\vee$-strategy improvement approach of Gawlitza and Seidl [10, 11]. From an algorithmic perspective these two approaches have quite distinct characteristics. $\vee$-strategy iteration, when applied to quadratic zones, in each iteration combines one constraint for each program point and program variable into a *global* semi-definite programming problem which is jointly solved by SDP.

The advantage of the $\vee$-strategy iteration approach is that (given an ideal SDP solver) the number of iterations is guaranteed to be finite and that it guarantees minimality of the obtained solution. The draw-back, however, is that only after termination, a safe invariant is found. Intermediate approximates to the least solution are not safe.

The $\wedge$-strategy iteration approach on the other hand, when applied to quadratic templates, relies on solving (dual) SDP problems *locally* for every constraint separately — each of which typically involves just few unknowns of the analysis problem. The *global* task of determining the next approximate for all program points and program variables then is delegated to LP solving. The disadvantage of the $\wedge$-strategy iteration approach is that the iteration is not guaranteed to *terminate* but only to *converge* to a solution. Moreover, this solution is not necessarily minimal. On the other hand (again assuming ideal solvers for SDP and LP), it produces a decreasing sequence of post-fixpoints. Thus,

the iteration may any time be terminated with a valid program invariant. Moreover, the speed of convergence is — as for Newton's method — usually quite good. Another advantage of the $\wedge$-strategy iteration approach is that LP solvers in general scale to larger problems than SDP solvers. Therefore, we expect the $\wedge$-strategy iteration approach to be applicable not just to small, but also to medium sized input programs. A detailed practical comparison w.r.t. efficiency and precision, however, remains for future work.

Bibliography

[1] A. Adjé, S. Gaubert, and E. Goubault. Computing the smallest fixed point of nonexpansive mappings arising in game theory and static analysis of programs. In *Proceedings of the Eighteenth International Symposium on Mathematical Theory of Networks and Systems (MTNS2008)*, 2008.

[2] A. Adjé, S. Gaubert, and E. Goubault. Coupling policy iteration with semi-definite relaxation to compute accurate numerical invariants in static analysis. In A. D. Gordon, editor, *ESOP*, volume 6012 of *LNCS*, pages 23–42. Springer, 2010. ISBN 978-3-642-11956-9.

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

[4] H. Björklund, S. Sandberg, and S. Vorobyov. Complexity of Model Checking by Iterative Improvement: the Pseudo-Boolean Framework . In *Proc. 5th Int. Andrei Ershov Memorial Conf. Perspectives of System Informatics*, pages 381–394. LNCS 2890, Springer, 2003.

[5] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[6] A. Costan, S. Gaubert, E. Goubault, M. Martel, and S. Putot. A Policy Iteration Algorithm for Computing Fixed Points in Static Analysis of Programs. In *Computer Aided Verification, 17th Int. Conf. (CAV)*, pages 462–475. LNCS 3576, Springer Verlag, 2005.

[7] E. Feron and F. Alegre. Control software analysis, part i open-loop properties. *CoRR*, abs/0809.4812, 2008.

[8] E. Feron and F. Alegre. Control software analysis, part ii: Closed-loop analysis. *CoRR*, abs/0812.1986, 2008.

[9] S. Gaubert, E. Goubault, A. Taly, and S. Zennou. Static analysis by policy iteration on relational domains. In Nicola [17], pages 237–252. ISBN 978-3-540-71314-2.

[10] T. Gawlitza and H. Seidl. Precise relational invariants through strategy iteration. In J. Duparc and T. A. Henzinger, editors, *CSL*, volume 4646 of *LNCS*, pages 23–40. Springer, 2007. ISBN 978-3-540-74914-1.

[11] T. Gawlitza and H. Seidl. Precise fixpoint computation through strategy iteration. In Nicola [17], pages 300–315. ISBN 978-3-540-71314-2.

[12] T. Gawlitza and H. Seidl. Precise interval analysis vs. parity games. In J. Cuéllar, T. S. E. Maibaum, and K. Sere, editors, *FM*, volume 5014 of *LNCS*, pages 342–357. Springer, 2008. ISBN 978-3-540-68235-6.

[13] T. M. Gawlitza and H. Seidl. Computing relaxed abstract semantics w.r.t. quadratic zones precisely. In R. Cousot and M. Martel, editors, *SAS*, volume 6337 of *Lecture Notes in Computer Science*, pages 271–286. Springer, 2010. ISBN 978-3-642-15768-4.

[14] T. M. Gawlitza and H. Seidl. Solving systems of rational equations through strategy iteration. *TOPLAS*, (accepted, to appear).

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

[16] A. Nemirovski. *Modern Convex Optimization*. Department ISYE, Georgia Institute of Technology, 2005.

[17] R. D. 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 *LNCS*, 2007. Springer. ISBN 978-3-540-71314-2.

[18] J. Ortega and W. Rheinboldt. *Iterative solution of nonlinear equations in several variables*. Academic Press, 1970.

[19] S. Sankaranarayanan, H. B. Sipma, and Z. Manna. Scalable analysis of linear systems using mathematical programming. In R. Cousot, editor, *VMCAI*, volume 3385 of *LNCS*, pages 25–41. Springer, 2005. ISBN 3-540-24297-X.

[20] S. Sankaranarayanan, M. Colón, H. B. Sipma, and Z. Manna. Efficient strongly relational polyhedral analysis. In E. A. Emerson and K. S. Namjoshi, editors, *VMCAI*, volume 3855 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2006. ISBN 3-540-31139-4.

[21] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1986.

[22] M. J. Todd. Semidefinite optimization. *Acta Numerica*, 10:515–560, 2001.

[23] S. A. Vavasis. Quadratic programming is in np. *Inf. Process. Lett.*, 36(2):73–77, 1990.