THÈSE

présentée à

l'ÉCOLE POLYTECHNIQUE

pour l'obtention du titre de

DOCTEUR DE L'ÉCOLE POLYTECHNIQUE
Spécialité Informatique

par

Khalil GHORBAL

28 Juillet 2011

Analyse Statique de Programmes Numériques: Ensembles Affines Contraints

Static Analysis of Numerical Programs: Constrained Affine Sets Abstract Domain

---

Président du jury:
Stéphane Gaubert, *Professeur, INRIA et École Polytechnique, Saclay.*

Rapporteurs:
Jean-Luc Lamotte, *Professeur, Université Pierre et Marie Curie, Paris.*
Sriram Sankaranarayanan, *Assistant Professor, University of Colorado Boulder, Colorado.*
Thao Dang, *Verimag CNRS, Grenoble.*

Examinateurs:
Bertrand Jeannet, *INRIA Rhône-Alpes, Montbonnot.*
Antoine Miné, *École Normale Supérieur, Paris.*

Directeurs de thése:
Eric Goubault, *Professeur, CEA List et École Polytechnique, Saclay.*
Sylvie Putot, *CEA List, Gif-sur-Yvette.*

École Polytechnique
Département d'Informatique

To my dear parents لطفي and سهام.

إلى خليلتي فاطمة.

Abstract

We aim at proving automatically the correctness of numerical behavior of a program by inferring invariants on numerical variables. More precisely, we over-approximate in a sound manner the set of reached values. We use Abstract Interpretation-based Static Analysis as a generic framework to define and approximate the semantics of a program in a unified manner. The semantics that describe the real behavior of the program (concrete semantics) is in general undecidable. Abstract interpretation offers a way to abstract this concrete semantics to obtain a decidable semantics involving machine-expressible objects. We introduce a new affine forms-based abstract domain, called *constrained affine sets*, which extends and generalizes an already existing abstract domain introduced by Eric Goubault and Sylvie Putot. The expressiveness of such new domain is enhanced thanks to its ability to encode and propagate linear constraints among variables. We have implemented our new domain to experiment the precision and the efficiency of our approach and compare our results to the already existing abstract domains. The theoretical work as well as the implementation and the experiments have been the subject of two publications [CAV 2009, CAV 2010].

Résumé

Nous nous plaçons dans le cadre de l'analyse statique de programmes, et nous nous intéressons aux propriétés numériques, c'est à dire celles qui concernent les valeurs numériques des variables de programmes. Nous essayons en particulier de déterminer une sur-approximation garantie de l'ensemble de valeurs possibles pour chaque variable numérique utilisée dans le programme à analyser. Cette analyse statique est faite dans le cadre de la théorie de l'interprétation abstraite, théorie présentant un compromis entre les limites théoriques d'indécidabilité et de calculabilité et la précision des résultats obtenus. Nous sommes partis des travaux d'Éric Goubault et Sylvie Putot, que nous avons étendus et généralisés. Notre nouveau domaine abstrait, appelé *ensembles affines contraints*, combine à la fois l'efficacité de calcul des domaines à base de formes affines et le pouvoir expressif des domaines relationnels classiques tels que les octogones ou les polyèdres. Le nouveau domaine a été implémenté pour mettre en évidence l'intérêt de cette combinaison, ses avantages, ses performances et ses limites par rapport aux autres domaines numériques déjà existants. Le formalisme ainsi que les résultats pratiques ont fait l'objet de plusieurs publications [CAV 2009, CAV 2010].

Remerciements

C'est à mes très chers parents que je dédie tout ce travail. Je vous suis et serai toujours reconnaissant • À ma complice, amie et compagne Fatma Safi, mon vocabulaire trouve ses limites en voulant te dire merci • Merci chers Stéphane Gaubert, Jean-Luc Lamotte, Thao Dang, Sriram Sankaranarayanan, Antoine Miné et Bertrand Jeannet, j'étais honoré par vous avoir dans mon jury. Merci pour votre temps, merci pour vos commentaires, cela a sans nul doute contribué à améliorer la qualité de ce travail • Un grand merci à mes deux directeurs de thèse Sylvie Putot et Éric Goubault. On ne peut-être plus gâté: pédagogie, rigueur, minutie, et fine intelligence, le tout avec le sourire et la bonne humeur. Sans oublier l'énorme quantité de chocolat que j'ai avalé durant ma thèse ainsi que l'initiation à l'escalade, *La vie au bout des doigts* et *Opéra vertical* faisaient partie de la bibliographie. Je garderai de très agréables souvenirs, merci du fond du coeur • Merci Assalé Adjé, j'ai vraiment aimé ta façon de voir les maths. J'ai eu la chance d'avoir Karim Tekkal à l'autre bout de l'open space. Ce fut très agréable de travailler avec toi, merci pour ton bon vivant. Merci Xavier Allamigeon, Olivier Bouissou et Alexandre Chapoutot, les moins jeunes thésards, désormais docteurs confirmés. Xavier, ton intérêt et tes questions m'ont permis de mieux cerner mon sujet. Merci Olivier pour tes explications et tes exposés, cela m'a beaucoup aidé. Merci Alexandre pour ton aide précieuse, tu avais toujours une solution simple à tout • Merci Emmanuel Haucourt pour le temps que tu m'avais consacré, tu as ce don de rendre accessible les notions les plus complexes. J'en profite pour remercier Sanjeevi Krishnan, qui était de passage au LMeASI, et qui m'a initié à la topologie algébrique. Merci Michel Hirschowitz, je me rappellerai toujours de nos longues et riches discussions métaphysiques. Merci pour ton humour et ton sens critique. Je ne manquerai pas finalement de remercier Franck Védrine, l'efficacité tranquille. Ce fut vraiment très agréable de travailler avec toi • Sans les encouragements et le soutient de mes amis et camarades, et toute ma famille ce travail n'aurait peut-être pas pu être accompli. Merci Béchir Zalila, Mehdi Frikha, Taoufik Hnia, Ala Ben Abbes, Mohamed Chakroun pour m'avoir toujours pousser à aller de l'avant. Je remercie aussi mes frères et soeur Youssef, Hamza et Sarra et toute ma belle famille, j'espère être à la hauteur de vos espérances • Un énorme merci à Fatma et Hasna Safi pour vos multiples relectures détaillées de ce manuscrit, cela a réduit considérablement les typos commises • Je remercie mon voisin et ami Sayed Mojabi, on a passé d'inoubliables années à Bures-Sur-Yvette • Je remercie finalement Audrey Lemarechal, Christine Ferret, et toute l'équipe de l'EDX. Vous faites un excellent travail.

❧ ❧ ❧

Contents

CHAPTER 1

Introduction

Ce qui importe, ce n'est pas d'arriver, mais d'aller vers.
ANTOINE DE SAINT-EXUPÉRY

A huge effort has been done during the last three decades in an attempt to ensure the correctness of software behavior while adding a reasonable overhead to the time-to-market of software dependant products. Nowadays, more than ever, such a goal is still one of the biggest challenges facing the computer science community. It is amazing to notice the importance and the impact of software in our everyday life. Cellphones, cars, planes, appliances, medias, networks, telecommunications, databases, power plants, factories ... software are everywhere, include a variety of heterogeneous services and monitor many critical and life dependant applications.

We focus on critical embedded control command software used for instance in airplanes or spacecrafts. This work is in line with the use of formal verification techniques to prove the correctness of a software with respect to its specification. We use the static analysis by abstract interpretation as a general theoretical framework. Static analysis means that we do not run the software under analysis (in contrast with dynamic methods, as tests for instance). Instead, the semantics of the program is extracted from the source code, then approximated in a sound manner by a (decidable) abstract semantics. The latter abstract semantics allows to synthesize invariants that the variables always verify. The level of abstraction reflects the expressiveness of the analysis and hence, the precision of the inferred invariants. Usually, a precise analysis is expensive in time, whereas cheap analysis gives imprecise results. Therefore, the main effort in the field consists in looking for a good precision-cost trade-offs.

Our main contributions are :

- The definition and formalization of a new precise and efficient abstract domain, that is a way to represent and compute efficiently the set of reached values of all numerical variables of the analyzed program.
- The improvement of the set-theoretic operations and mainly the join operation of the geometrical objects we use. Our approach makes our domain suitable for functional analysis, that is the study of the input/output relations of the program.
- An efficient implementation of our abstract domain called *Taylor1+*. This sophisticated prototype is freely available and distributed with the APRON Library widely used by the abstract interpretation community.

These contributions will be integrated in *Fluctuat*, a static analyzer that studies the discrepancy introduced by the use of finite-precision representation (such as floating-point numbers) instead of the use of real numbers.

Outline The first chapter motivates the need of the abstract domain we develop later in this work. Starting from an ESA-funded industrial case study, the formal verification of the source code of the ATV spacecraft using existing fully-fledged analyzers, has indeed shown an unwanted loss of precision. In chapter 3 we give a detailed overview of the numerical properties we would like to prove as well as the already existing abstract domains. We emphasize more particularly the numerical domains to which we compare our approach, namely the weakly relational abstract domains family (zones and octagons) and the linear template-based abstract domain. Our abstract domain is introduced in chapter 4, where we define its abstract objects and its lattice-like structure. The next two chapters formalizes the abstraction of the transfer functions. Chapter 5 focuses on assignments and tests, while chapter 6 is entirely dedicated to the join operation, one of the most challenging issue we have got to solve. Right before the conclusion, our experimental results are gathered in chapter 7, in which we detail the features of *Taylor1+*, the implementation of our abstract domain.

CHAPTER 2

Context and Motivations

2.1 Proving Numerical Properties

The need to estimate the computation errors due to the use of floating-point numbers is crucial in order to interpret how significant the returned results are. Rounding-off errors or overflows may cause a serious loss of precision leading to an unexpected behavior of the software. Although scientists and engineers are aware of these intrinsic issues [Gol91, Ste74] and despite the fact that there exists a norm which clarifies and normalizes the hardware implementation of floating-point arithmetic [IEE85, IEE08], it is hard to tell, given an implementation of an algorithm, if the numerical computations are safe, even for small programs.

Whenever, a loss of precision is detected, it is also interesting to point out the sources of this loss as a helpful feedback for the developer. Indeed, a minor local loss of precision may cause a wrong interpretation of a test, and hence lead to a wrong decision.

To make things clear, we consider the case of solving an ordinary differential equation (ODE) using a computer. The solution could be approximated numerically using for instance the Euler method. The Euler method gives a numerical approximation to the ideal solution of the ODE which can not be computed explicitly. The *method error* is defined by the difference between the Euler approximation and the ideal solution. Now, the Euler approximation is implemented as a computer program. This program uses finite-precision numbers (typically floating-point numbers) for all intern computations instead of real numbers, which introduces in turn what is called the *computation error*, or *round-off error*.

Although, the method errors could be theoretically estimated (with respect to the real numbers semantics), the computation errors are hard to estimate in general. The use of finite-precision numbers makes the estimation even harder. We give hereafter an overview of the known approaches used to estimate the computation errors.

The CESTAC Method

The CESTAC [1] method is useful to self-validate the implementation of the numerical approximations used in scientific computing to simulate a mathematical model (as a physical law for instance, usually defined as a set of ODE). It permits to measure the confidence one can have in the returned results by the software with respect to the use of finite precision numbers instead of real numbers. It was also successfully applied to find the best discrete step, that is the optimal step that minimizes the global error of the computation (the accumulation of the method error and the computation error), for a wide class of algorithms, such as the Runge-Kutta integration schemes or numerical derivation methods [Jea90].

The method was introduced by Vignes [Vig78] and uses the stochastic arithmetic [Vig93]. The idea is to inject a random perturbation by adding to the last bit of the mantissa of each resulting float-point number of each arithmetic operation either 0 or 1 with a probability $\frac{1}{2}$ for each [2]. The arithmetic operation is then performed according to all possible perturbed values of the operands. The final (returned) result is the arithmetic mean of all these possible partial results (samples). The method uses then the Student's t-test to estimate the number of significant digits of the final returned result, that is the digits common to all samples.

The CESTAC method was later proved efficient by Chesnaux [JM88] (actually only two or three executions are needed instead of all possible cases which may blow up the computations), and extended to synchronous CESTAC by Faye [JP89] and Flavigny [Fla88]. The synchronous CESTAC tests the significance of the result of each operation which permits to detect the origin of the loss of precision, and hence to emit an alert if the numerical algorithm is unstable (that is, if the loss of precision due to the use of floating-point numbers leads to a wrong interpretation of a test). The CESTAC method is implemented in the CADNA (Control of Accuracy and Debugging for Numerical Applications) library [JC08], it permits the

[1] French Acronym which stands for *Contrôl et Estimation STochastique des Arrondis de Calculs*, Stochastic Control and Estimation of Calculus Round-offs.

[2] depending on the rounding mode, one might add -1 or 1 with a probability $\frac{1}{4}$ and 0 with a probability $\frac{1}{2}$.

validation of the C, ADA, and FORTRAN programs. The CADNA Library uses a dynamic approach, it compiles and executes the program in order to estimate on-the-fly the round-off errors. The overhead of the method is estimated to 3 to 6 times the needed time to execute the algorithm.

Formal Specification of Floating-point Arithmetic

Recently, a specification language for the floating-point arithmetic for `C` programs was introduced in [BF07]. The program is annotated with formal pre- and post-conditions. Then, verification conditions (first order logic statements) are generated using Hoare logic [Hoa83]. Finally, these conditions are discharged, interactively, using a proof assistant.

The considered model to estimate the approximations errors for each number is a triplet : the floating point number, the idealized real number found if the computations were actually done with respect to real numbers semantics, and the real number that the algorithm is designed to compute. Therefore, in addition to the discrepancy related to the use of the floating-point numbers (computation error), the method error, that is the numerical error related to the used numerical algorithm, could be also estimated thanks to the third component.

In [BF07], the authors use the Caduceus tool [FMH] for the static verification of `C` program and the Coq proof assistant [coq] to discharge the generated proof obligations. The proof obligations could be also discharged using Gappa tool [dDLM06, Gap]. Gappa relies on interval arithmetic and formal specification of the floating-point number arithmetic to prove error bounds or the absence of overflows.

Abstract Interpretation-based Approach

This approach is completely automated like the CESTAC method, while giving sound estimation of both the computation and method errors like the formal specification method. It considers all possible executions in a sound manner and detects as well any possible loss of precision due to the use of finite precision numbers.

The method, introduced by Goubault [Gou01a, GMP02] is completely static. It relies on the semantics of the arithmetic operations and uses the Abstract Interpretation framework to approximate the floating-point computation semantics. It analyses the given source code with respect to real number semantics in one hand and floating-point semantics in the other hand. The loss of precision is then implied by the discrepancy observed between the results of the two semantics. The technique permits

also to infer numerical invariants that the program respects as well as an over-approximation (with respect to both used semantics) of all possibly reached value for all numerical variables used. Fluctuat tool [Flu] implements such abstract interpreter, using both relational and non-relational abstract domains.

Miné has also used a similar approach in [Min04a] using linear forms with interval coefficients to propagate the computation errors. Although this simple technique does not allow an estimation of the method errors, nor the source of the loss of precision as it accumulates the errors using interval arithmetic.

Goubault and Putot defined a rich model including the floating point representative, the ideal real number and the global error of the computation decomposed with respect to its origin (line of the program, number of iteration, etc.). As detailed in [GP11], the authors use a zonotopic relational domain to derive tight invariants for the real values of the variables, as well as the global error of the computation related to each variable.

This thesis focuses on this zonotopic relational domain. The domain uses the affine arithmetic (presented in the next section) to implicitly encode the relations between variables. The abstract domain is presented in detail with respect to the real number semantics all along the remaining chapters (abstract objects in Chapter 4, and abstract operations in Chapters 5 and 6). The improvements presented in this work could be then deployed to handle the abstract computation of the real values as well as the global errors.

2.2 Abstract Interpretation

Abstract Interpretation-based Static Analysis is an efficient way to statically and automatically prove the correctness of a program. It gives a generic framework to define and approximate the semantics of a program in a unified manner. The semantics that describe the real behavior of the program (concrete semantics) is in general undecidable. Abstract interpretation offers a way to abstract this concrete semantics (or any other semantics) to obtain a decidable semantics involving machine-expressible objects.

Throughout this thesis, new definitions are introduced by the symbol $\stackrel{\text{def}}{=}$, or $\stackrel{\text{def}}{\Longleftrightarrow}$. The set of real numbers is denoted by $\mathbb{R}$. Each vector e_i of the *canonical base* of $\mathbb{R}^n$, is defined by 1 in its ith position and 0 elsewhere. The transpose of a vector v (or a matrix M) is denoted using an upper star index, v^* (or M^*).

An *interval* is the set of real numbers $\{x \mid a \leq x \leq b\}$, where $a, b \in$

$\mathbb{R} \cup \{-\infty, +\infty\}$, such that $a \leq b$. It is denoted by $[a, b]$. a and b are the bounds of the interval.

The set of intervals is denoted by $\mathbb{I}$. As a convention, we use bold face fonts to denote the elements of $\mathbb{I}$.

For $\boldsymbol{i}$ an element of $\mathbb{I}$, we define:

- $\inf(\boldsymbol{i})$, or $\underline{i}$: the infimum bound of $\boldsymbol{i}$,
- $\sup(\boldsymbol{i})$, or $\overline{i}$: the supremum bound of $\boldsymbol{i}$,
- if $\boldsymbol{i}$ has a finite bounds, then $\mathrm{mid}(\boldsymbol{i}) \stackrel{\text{def}}{=} \frac{\sup(\boldsymbol{i})+\inf(\boldsymbol{i})}{2}$,
- if $\boldsymbol{i}$ has a finite bounds, then $\mathrm{dev}(\boldsymbol{i}) \stackrel{\text{def}}{=} \frac{\sup(\boldsymbol{i})-\inf(\boldsymbol{i})}{2}$.

We call a *hypercube*, or *box*, any subset of $\mathbb{R}^n$ of the form $\Pi_{i=1}^{n}[a_i, b_i]$. The symbol $\subseteq$ denotes the classical inclusion relation over $\mathbb{R}^n$

$$S_1 \subseteq S_2 \stackrel{\text{def}}{\iff} (x \in S_1 \implies x \in S_2) \ .$$

Let n be a positive integer, let λ be a real number, and let x and y be two vectors of $\mathbb{R}^n$. An application $\mathcal{N}\colon \mathbb{R}^n \to \mathbb{R}_+$, is a norm, if and only if the following properties hold

- $\mathcal{N}(x) = 0 \iff x = 0$,
- $\mathcal{N}(\lambda x) = |\lambda|\mathcal{N}(x)$ (positive homogeneity),
- $\mathcal{N}(x + y) \leq \mathcal{N}(x) + \mathcal{N}(y)$ (triangle inequality).

A seminorm (or equivalently a quasinorm) is a norm with the first requirement in the above list removed, that is $\mathcal{N}(x) = 0$ does not imply necessarily that x is the zero vector. The classical norms over $\mathbb{R}^n$ are:

- **Euclidean norm:** $\|x\|_2 \stackrel{\text{def}}{=} \left(\sum_{i=1}^{n} x_i^2\right)^{\frac{1}{2}}$.
- **Infinity (or uniform) norm:** $\|x\|_\infty \stackrel{\text{def}}{=} \max\{|x_1|, \ldots, |x_n|\}$.
- **Taxicab (or L_1, or Manhattan) norm:** $\|x\|_1 \stackrel{\text{def}}{=} \sum_{i=1}^{n} |x_i|$.

The *unit ball* of $\mathbb{R}^n$, with respect to the norm $\mathcal{N}$ is the set defined by

$$\mathcal{B}_{\mathcal{N}} \stackrel{\text{def}}{=} \{x \mid \mathcal{N}(x) \leq 1\} \subseteq \mathbb{R}^n \ .$$

We denote by $\mathfrak{B}$, the unit ball with respect to the infinity norm, the dimension should be clear from the context, otherwise specified.

We define the sign function sign over $\mathbb{R} \setminus \{0\}$ as follows:

2.2.1 Definition (The Sign Function)

$$\text{sign}(x) = \begin{cases} -1, & \text{if } x < 0, \\ 1, & \text{if } x > 0, \end{cases}$$

The sign of 0 *is undefined.*

Basic definitions and main concepts of abstract interpretation theory are presented briefly hereafter, following [Min04b].

Partially ordered set A partially ordered set or *poset* $(\mathcal{D}, \leq)$ is a set of elements $\mathcal{D}$ together with a partial order relation $\leq$, that is a binary relation which is reflexive ($\forall X \in \mathcal{D}, X \leq X$), transitive ($\forall X, Y, Z \in \mathcal{D}, X \leq Y \wedge Y \leq Z \implies X \leq Z$) and antisymmetric ($\forall X, Y \in \mathcal{D}, X \leq Y \wedge Y \leq X \implies X = Y$). Similarly, a partially pre-ordered set is a pair $(\mathcal{D}, \preceq)$, where $\preceq$ is a pre-order, that is a binary relation which is reflexive and transitive. Any partially pre-ordered set $(\mathcal{D}, \preceq)$ defines a poset $(\mathcal{D}/\sim, \leq)$, where the partial order $\leq$ is defined over the equivalence classes defined by $Y \sim X \stackrel{\text{def}}{\iff} \{Y \in \mathcal{D} \mid X \preceq Y \wedge Y \preceq X\}$. An *upper bound* of a subset D of a poset $(\mathcal{D}, \leq)$ is an element of $\mathcal{D}$ which is greater than or equal to all elements of D with respect to $\leq$. Similarly, a *lower bound* of a subset of a poset is an element which is less than or equal to all elements of that subset. A *least upper bound*, or lub, of a subset $D \subseteq \mathcal{D}$, denoted by $\cup D$ if it exists, is an upper bound of D which is less than or equal to all upper bounds of D. Dually, a *greatest lower bound*, or glb, of a subset $D \subseteq \mathcal{D}$, if it exists, is a lower bound of D which is greater than or equal to all upper bounds of D. A poset is *directed complete* if every increasing chain, that is $\{X_i\}_{i \in I}$, $i, j \in I, i \leq j \implies X_i \leq X_j$, admits a least upper bound (the set $I \subset \mathbb{N}$, may be infinite). A *complete* poset is a direct complete poset which admits a least element.

Lattice A lattice $(\mathcal{D}, \leq, \cup, \cap)$ is a poset $(\mathcal{D}, \leq)$, where every two elements X and Y of $\mathcal{D}$ admit a least upper bound, denoted by $X \cup Y$, and a greatest lower bound, denoted by $X \cap Y$. A lattice is *complete*, if every subset $D \subseteq \mathcal{D}$ admits a lub. A complete lattice is a complete poset. A lub of $\mathcal{D}$ is denoted by $\top$, and a glb of $\mathcal{D}$ is denoted by $\bot$.

Applications An application is a function from a poset $(\mathcal{D}_1, \leq_1)$ to another poset $(\mathcal{D}_2, \leq_2)$. It is called *operator* if it is defined over the same poset. An application $[\![\cdot]\!] : \mathcal{D}_1 \to \mathcal{D}_2$ that satisfies $\forall X, Y \in \mathcal{D}_1, X \leq_1 Y \implies$

$[\![\cdot]\!]X \leq_2 [\![\cdot]\!]Y$ is called a *monotonic* application. An application that preserves the limits of increasing chains, that is $[\![\cdot]\!]_1(\cup\{X_i\}_{i\in I}) = \cup\{[\![\cdot]\!]_1 X_i\}_{i\in I}$, whenever these limits exist, is said to be *continuous*. We call the ith iterate of an operator $[\![\cdot]\!]$, denoted by $[\![\cdot]\!]^i$ the operator defined by induction $[\![\cdot]\!]^i \stackrel{\text{def}}{=} [\![\cdot]\!]([\![\cdot]\!]^{i-1})$, for $i \in \mathbb{N}$, where, $[\![\cdot]\!]^0$ is the identity over $\mathcal{D}$. An application $[\![\cdot]\!] : \mathcal{D}_1 \to \mathcal{D}_2$ that preserves the lub (if it exists), that is $[\![\cdot]\!] \cup_1 D = \cup_2\{[\![\cdot]\!]X \mid X \in D\}$ for $D \subseteq \mathcal{D}_1$ is called a complete $\sqcup$-morphism.

Fixpoint A fixpoint of an operator $[\![\cdot]\!]$ defined over a poset $\mathcal{D}$ is an element $X \in \mathcal{D}$ such that $X = [\![\cdot]\!]X$. We denote by $\text{lfp}_X[\![\cdot]\!]$ the least fixpoint, if it exists, of the operator $[\![\cdot]\!]$, greater than or equal to X (with respect to the partial order of $\mathcal{D}$). Tarsky's Fixed Point Theorem [Tar55] proves the existence of the least fixpoint under some assumptions.

2.2.2 Theorem (Tarsky)
Let $f : \mathcal{D} \to \mathcal{D}$ be a monotonic operator on a complete lattice $\mathcal{D}$; then, f has at least one fixpoint. Furthermore the set of fixpoints of f is a complete sub lattice of $\mathcal{D}$, and as a consequence, it admits a least fixpoint, $\text{lfp}_\perp f$.

Galois connection A Galois connection is a pair of monotonic applications $\alpha : \mathcal{D}^\flat \to \mathcal{D}^\sharp$ and $\gamma : \mathcal{D}^\sharp \to \mathcal{D}^\flat$ between two posets $\mathcal{D}^\flat$, and $\mathcal{D}^\sharp$, such that $\forall X \in \mathcal{D}^\flat, \forall Y \in \mathcal{D}^\sharp$, $\alpha(X) \leq^\sharp Y \iff X \leq^\flat \gamma(Y)$. Thus, $\forall X \in \mathcal{D}^\flat$, $X \leq^\flat \gamma\circ\alpha X$. The latter property is known as the *soundness* of abstraction. The application α is called the *abstraction*, the application γ is called the *concretisation*.

Operator abstraction Operator abstraction stands for the transfer of a given operator $[\![\cdot]\!]^\flat$ defined over the poset $\mathcal{D}^\flat$, through a Galois connection (α, γ), to obtain an operator $[\![\cdot]\!]^\sharp$ defined over $\mathcal{D}^\sharp$. An abstract operator $[\![\cdot]\!]^\sharp$ is a *sound* operator if, for all $Y \in \mathcal{D}^\sharp$, $\alpha \circ [\![\cdot]\!]^\flat \circ \gamma Y \leq^\sharp [\![\cdot]\!]^\sharp Y$. For instance, $[\![\cdot]\!]^\sharp \stackrel{\text{def}}{=} \alpha \circ [\![\cdot]\!]^\flat \circ \gamma$ is a sound abstraction of the operator $[\![\cdot]\!]^\flat$. The soundness property is equivalent to $[\![\cdot]\!]^\flat \circ \gamma \leq^\flat \gamma \circ [\![\cdot]\!]^\sharp$, by definition of the Galois connection. The latter formulation is suitable to prove the soundness property whenever we do not have an explicit α which is usually the case in practice.

Fixpoint computation If two complete posets $\mathcal{D}^\flat$ and $\mathcal{D}^\sharp$ are linked by a Galois connection, where γ is continuous, and if $[\![\cdot]\!]^\sharp$ is a monotonic sound abstraction of a monotonic operator $[\![\cdot]\!]^\flat$; then, by [Cou02, Theorem 1],

$$\forall X \in \mathcal{D}^\sharp, \quad \mathrm{lfp}_{\gamma(X)}[\![\cdot]\!]^\flat \leq^\flat \gamma(\mathrm{lfp}_X[\![\cdot]\!]^\sharp) \ .$$

That is, the least fixpoint greater than or equal to $\gamma(X)$ in $\mathcal{D}^\flat$, for a given $X \in \mathcal{D}^\sharp$, is over-approximated by the concretisation of the least fixpoint greater than or equal to X in $\mathcal{D}^\sharp$. If $\mathcal{D}^\sharp$ is a complete poset; then, Kleene's Theorem gives an algorithm to compute such fixpoint iteratively:

2.2.3 Theorem (Kleene)
Let $F : \mathcal{D} \to \mathcal{D}$ be a monotonic operator over a complete poset $\mathcal{D}$. Then, the increasing chain F^i starting from $\bot \stackrel{\text{def}}{=} \cup\emptyset$, admits a limit F^ω, and $F^\omega = \mathrm{lfp}_\bot F$.

Starting from the bottom element, and applying successively the operator F, the computation converges towards the least fixpoint of F. However, if the complete poset $\mathcal{D}$ has an infinite strictly increasing chain; then, the procedure may take an infinite time. In general, we apply convergence acceleration.

Convergence acceleration A convergence acceleration is an operator used to reach in finite steps a post-fixpoint. A post-fixpoint of a monotonic operator $[\![\cdot]\!]^\sharp$ defined over the poset $\mathcal{D}^\sharp$, is an element $X \in \mathcal{D}^\sharp$ that satisfies $[\![\cdot]\!]^\sharp X \leq^\sharp X$. Usually such operation is denoted by $\nabla^\sharp$ and called *widening*. We also find in the literature the dual operation to widening, called *narrowing* which aims at bringing closer (to the fixpoint) the post-fixpoint obtained after a widening.

In the sequel, we define then use a simple imperative language for the sake of clarity. Semantics of real imperative languages, such as `C`, can be extended easily. We suppose that the only possible type for numerical variables is the real number type. In our language `SimpleC`, a statement s has the following grammar:

2.2.4 Definition (SimpleC Grammar)

$$
\begin{array}{lcll}
s & ::= & v \leftarrow expr & \textit{(assignment)} \\
 & | & \textbf{\textit{if}}\ bexpr\ \textbf{\textit{then}}\ s & \textit{(conditional)} \\
 & | & \textbf{\textit{while}}\ bexpr\ \textbf{\textit{do}}\ s & \textit{(loop)} \\
expr & ::= & v \mid [a,b] \mid expr \diamond expr \mid \sqrt{expr}, & \\
bexpr & ::= & expr \leq 0 \mid expr = 0 & \\
 & & \neg bexpr \mid bexpr \wedge bexpr \mid bexpr \vee bexpr, & \\
\multicolumn{4}{l}{\text{where } v \in \mathcal{V}, a, b \in \mathbb{R} \cup \{-\infty, +\infty\}, \diamond \in \{+, -, \times, \div\}}
\end{array}
$$

The arithmetic operations are restricted to $\{+, -, \times, \div\}$. The language allows non-deterministic inputs, expressed by intervals. Arrays and aliases are not supported.

The concrete semantics of our simple language describes the mathematical behavior of the values of variables during the execution of the program. A program environment $\sigma \in \Sigma$ maps each variable to a set of values, that is $\Sigma \stackrel{\text{def}}{=} \mathcal{V} \to \wp\mathbb{R}$. The semantics $[\![e]\!]$ of an expression $e \in expr$ maps an environment to a set of values in $\wp(\mathbb{R})$.

$$
\begin{array}{rcl}
\forall e \in expr, [\![e]\!] : \Sigma \to \wp(\mathbb{R}) & & \\
[\![v]\!]\sigma & \stackrel{\text{def}}{=} & \{\sigma(v)\} \\
[\![[a,b]]\!]\sigma & \stackrel{\text{def}}{=} & \{x \in \mathbb{R} \mid a \leq x \leq b\} \\
[\![e_1 \diamond e_2]\!]\sigma & \stackrel{\text{def}}{=} & \{x_1 \diamond x_2 \mid x_1 \in [\![e_1]\!]\sigma, x_2 \in [\![e_2]\!]\sigma\} \\
\multicolumn{3}{r}{\text{where } \diamond \in \{+, -, \times\}} \\
[\![e_1 \div e_2]\!]\sigma & \stackrel{\text{def}}{=} & \begin{cases} \emptyset, \text{if } 0 \in [\![e_2]\!]\sigma \\ \{x_1 \diamond x_2 \mid x_1 \in [\![e_1]\!]\sigma, x_2 \in [\![e_2]\!]\sigma\}, \text{otherwise.} \end{cases} \\
[\![\sqrt{e}]\!]\sigma & \stackrel{\text{def}}{=} & \begin{cases} \emptyset, \text{if } \forall x \in [\![e]\!]\sigma, x < 0 \\ \{\sqrt{x} \mid x \in [\![e]\!]\sigma \cap [0, +\infty]\}, \text{otherwise.} \end{cases}
\end{array}
$$

We do not store the locations of any error (here the division per zero and the square root of a negative real number are the only numerical errors that may happen). In a real analyzer, the semantics of expressions is context sensitive, (for instance the location of the operation is recorded); thus, whenever an error occurs, its context is reported to the user without necessarily halting the analysis.

We define the concrete semantics as the complete $\sqcup$-morphism on $\mathcal{D}^\flat \stackrel{\text{def}}{=}$

$(\wp(\Sigma), \subseteq, \sqcup, \sqcap, \emptyset, \Sigma)$ as follows:

$$\begin{aligned}
[\![s]\!]^\flat &: \mathcal{D}^\flat \to \mathcal{D}^\flat \\
[\![v \leftarrow e]\!]^\flat \varsigma &\stackrel{\text{def}}{=} \cup_{\sigma\in\varsigma}\{\sigma[v \mapsto x] \mid x \in [\![e]\!]\sigma\} \\
[\![\textbf{if } b \textbf{ then } s]\!]^\flat \varsigma &\stackrel{\text{def}}{=} \{[\![s]\!]^\flat \circ [\![b]\!]^\flat\}\varsigma \cup [\![\neg b]\!]^\flat \varsigma \qquad (2.2.1) \\
[\![\textbf{while } b \textbf{ do } s]\!]^\flat \varsigma &\stackrel{\text{def}}{=} [\![\neg b]\!]^\flat(\mathrm{lfp}_\varsigma\, \lambda X.X \cup ([\![s]\!]^\flat \circ [\![b]\!]^\flat)X) \\
\text{where } [\![b]\!]^\flat \varsigma &\stackrel{\text{def}}{=} \cup_{\sigma\in\varsigma}\{\sigma \mid \exists\, x \in [\![e]\!]\sigma, b \text{ is true}\}\ .
\end{aligned}$$

The environment $\sigma[v \mapsto x]$ denotes the environment derived from σ that assigns the value of the variable v with the real number x and leaves all other variables unchanged. We use the lambda-calculus functional notation $\lambda X.F(X)$ to denote the application that maps X to $F(X)$.

The semantics of the conditional statements $[\![b]\!]^\flat$, applied to a set of concrete environments, filters out the concrete environments that do not satisfy the condition.

The concrete semantics $[\![\cdot]\!]^\flat$ is undecidable in general. The set of environments may need infinite memory, and computation of the set of locations infinite time. To address these issues, we abstract the concrete semantics to obtain an abstract semantics which is *i*) decidable and *ii*) whose abstract objects are machine-expressible, and *iii*) which is sound.

To define an abstract semantics, one needs to define an abstract lattice, that is a partial order $\subseteq^\sharp$, an abstract join operator $\cup^\sharp$ over abstract elements, and a monotone abstract operator for every concrete operator $[\![s]\!]^\flat$, and every statement s that defines the initial language, and a continuous concretisation function γ.

Figure 2.1 depicts the main idea of abstract interpretation, where $\mathcal{X}_i$ denotes the sets of reached values at the control point i of the program. The final invariants are over-approximations of the concrete sets of values.

2.3 Fluctuat

Fluctuat [Gou01b, GMP02, Mar02, GMP06, GP11] is a static analyzer by abstract interpretation developed at the Laboratory for the Modelling and Analysis of Interacting Systems (LMeASI) at CEA LIST. It is suited to the analysis of numerical programs; in particular it gives a tight over-approximation of the discrepancy introduced by the use of finite precision (floating-point or fixed-point) numbers instead of real numbers. It keeps track of the contribution of each statement to the global error. Division

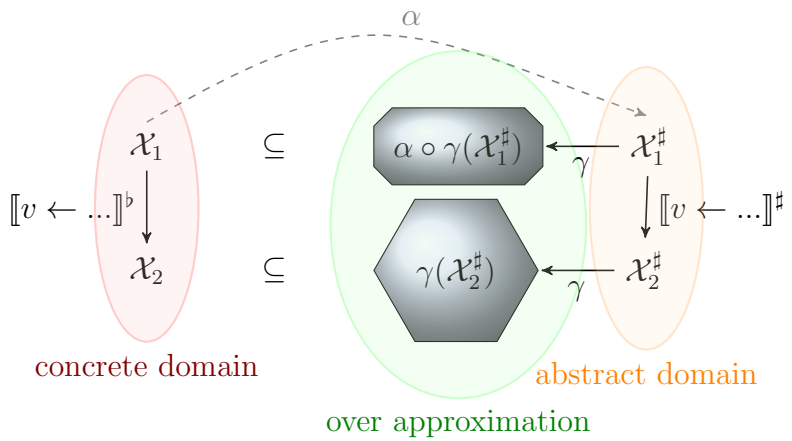

Figure 2.1: Abstract Interpretation.

by zero, overflows, unstable tests are also reported by the analyzer. Fluctuat was successfully applied in many case studies [GPBG07, DGP+09, BCC+09].

2.4 The ATV Case Study

The case study was an ESA funded project which main goal was the assessment of the abstract interpretation-based static analysis techniques on real life, automatically generated, industrial code. The source code, provided by Astrium Space Transportation, concerned the *Monitoring and Safing Unit* (MSU), the heart part of the Automated Transfer Vehicle (ATV). The spacecraft's mission was to supply, completely automatically, the International Space Station, ISS, with payloads (mainly fuel, and equipment for reactors) and to correct the spatial station orbit. To achieve successfully its mission, the ATV needs to dock into the ISS. In addition to the navigation, the MSU was in charge of the critical "take-away" phase triggered if any problem happens during the docking phase. Indeed, any failure of the docking phase can seriously damage the ISS as the engine operates too closely.

Two abstract interpretation-based static analyzers were involved: Fluctuat (see Section 2.3) and ASTRÉE. I was in charge of evaluating Fluctuat on this case study (during my master's internship).

ASTRÉE [Ast], *Analyses Statiques de Logiciels Temps RÉel Embarqués*, is a static analyzer by abstract interpretation developed at the *Laboratoire*

d'Informatique de lÉcole Normale Suprieure. It aims at automatically proving the absence of run-time error (RTE): division by zero, out of bounds array indexing, arithmetic overflow and user defined assertions given as input to the tool. ASTRÉE is avionic software oriented and was successfully applied in an industrial context [SD07, BCC+10].

The study has shown that the techniques in use are mature enough to be deployed as integrated tools within any project development cycle using C language [3] in that they substantially improve the reliability of the code with a reasonable time overhead. The main results are summarized in [BCC+09].

However, the case study has also shown that the current state-of-the-art of abstract domains are not fully suitable for space-like software because of the use of normalized quaternions (vectors of four dimensions). These quaternions are used to encode the space position of the spacecraft and express its possible motions (translation, space rotations) as linear transformations (matrix multiplications).

To have a good intuition about quaternions, the reader can think about the use of the complex numbers to encode the position of a point in the plan, together with the fact that all plan similitudes in the plan can be seen as 2×2 matrix multiplication.

Some (quaternion) operations implemented in the MSU determine only three components (the position of the spacecraft in the space) of the quaternion, the fourth remaining component is computed in a way for the final quaternion to have its Euclidean norm equal to 1. The other operations that determine all components, normalize the result. So that any quaternion given as input to any routine is always normalized.

These normalization operations were hard to abstract precisely using existing relational domain. Indeed, they involve four non-linear operations combined together: the square, the square root, the inverse and the multiplication.

Suppose we have the non-null quaternion (x_0, x_1, x_2, x_3), where x_i, are known real numbers; then, the normalized quaternion (y_0, y_1, y_2, y_3) is defined by

$$y_i \stackrel{\text{def}}{=} \frac{x_i}{\sqrt{x_0^2 + x_1^2 + x_2^2 + x_3^2}} \ . \tag{2.4.1}$$

In the abstract domains used, the evaluation of the expression of y_i is computed as a composition of basic arithmetic operations. Each operation introduces an approximation error term which in turn is propagated to the next operation. This makes it hard for instance to prove that the interval concretisations of y_i have to still within $[-1, 1]$, which should happen for

[3] The current version of Fluctuat supports Ada language.

a normalized quaternion. Since the normalization is repeated frequently, the intervals concretisations grow quickly leading to imprecise results and many false alarms.

One naive technique to address such problem is subdivision. However, subdividing 4 intervals into 10 smaller intervals gives 10^4 possible instances to analyze just for one normalization. The solution does not scale for the whole code.

Another technique could be the linearization of the whole expression once instead of the evaluation of the composition of basic operations. However, this method works only if the assignment is done once, as shown by equation (2.4.1) for instance. However, very often, in critical embedded systems, some operations such as the square root or the division are wrapped by safe guards, and thus the assignment is decomposed into more than one assignment using intermediate variables.

One other solution could be the transformation of the normalization function by an equivalent expression which computes the quotient of two independent terms; for y_0 for instance, the computation is done as follows:

$$y_0 = \frac{\text{sign}(x_0)}{\sqrt{1 + \frac{x_1^2+x_2^2+x_3^2}{x_0^2}}}$$

which proves that y_i, $1 \leq i \leq 4$ are within $[-1, 1]$ using interval arithmetic, since the numerator and denominator are no longer dependent.

However, in general, rewriting the code by the analyzer is not allowed, so for the purpose of the case study, we have proved the correct behavior of the normalization operation using an external prover [dDLM06]. We have then asserted that the intervals of the components of a normalized quaternion are always within $[-1, 1]$.

Another idea was to add a (quadratic) constraint which ensures that the vector y is normalized, that is $y_0^2 + y_1^2 + y_2^2 + y_3^2 = 1$ to an existing abstract domain.

In the Affine forms-based abstract domain, implemented in Fluctuat, such constraint could be expressed as a constraint on the noise symbols and propagated as such for future computations. This novel idea motivates our work as it also permits to precisely interpret tests, which were treated by a reduced product with intervals so far. The rest of this thesis develops and formalizes in detail this idea.

CHAPTER 3

Numerical Abstract Domains

One of the main drawbacks of static analysis by abstract interpretation is the number of false positives which are strongly related to the expressiveness of the underlying abstract domain. Indeed, the sound over-approximation usually adds unfeasible behaviors to the actual behavior of the program. The non-relational abstract domain [CC77] detailed earlier is for instance unsuitable to prove any dependency between variables. On the other hand, the abstraction is needed to overcome the decidability and computability limitations encountered when dealing with the concrete semantics of the program.

As an attempt to fill this gap, many abstract domains have been developed in the last three decades. In 1978, Cousot and Halbwachs [CH78, CMC08] presented a way to synthesize linear invariants using the double description of a set of linear constraints, generalizing Karr's special case of linear equalities [Kar76] presented in 1976. The exponential complexity of the analysis mainly due to the internal representation of a convex combination of a set of linear constraints (convex Polyhedron) motivated weakly linear relational abstract domains. In [Min01] Miné introduced the octagon abstract domain, which is restricted to invariants of the form $\pm X \pm Y \leq c$. The efficient internal representation gives a cubical complexity in the number of variables. Later, Simon et al. in [SKH03] extend the coefficients to be any real number instead of being constrained to ± 1. Sankaranarayanan and Manna in [SSM05] combine the two previous approaches in their guided polyhedra abstract domain: the linear constraints are restricted to a finite set of linear templates generated from the program to analyze. In addition to all these *explicit* relations-based abstract domain, Goubault and Putot [GP06] introduced an *implicit* relations-based on abstract domain,

using affine arithmetic as an extension to interval arithmetic to overcome its intrinsic dependency problem.

More recent and very promising abstract domains have targeted non-linear invariants. Allamigeon, Gaubert and Goubault, have used tropical algebra [SS04] to infer min-max invariants [AGG08], while in [AGG10] Adjé, Gaubert and Goubault, have used Lyapunov functions as non-linear templates together with SemiDefinite Programming relaxations for the fixpoint computation using policy iterations instead of value iteration (Kleene like techniques). The latter is a rising technique to cope with the limitations of Kleene iteration technique and was already formalized for other abstract domains [CGG+05].

Many abstract domains could be combined (in a manner to define) in order to increase the overall expressiveness of the analysis, and hence increase the precision of the final results. Cousot and Cousot pointed out in [CC77] a generic framework, called reduced product [CC79, Cou05], to combine two abstract domains. For instance Laviron and Logozzo exploit this combination in their sub-polyhedra abstract domain [LL09], which still suffers from the complexity of the polyhedra domain. In 2006, Tiwari and Gulwani have presented an elegant framework, called logical product, for combining abstract domains under some hypothesis [GT06]. The logical product allows a better and richer exchange of relations between the involved abstract domains than the reduced product which exchanges only the concretisations of the involved domains.

Contents In the remaining part of this chapter, we detail some abstract domains relevant to our work. In section 3.1, we recall the basics of the interval arithmetic first; then, we abstract the concrete semantics of our `SimpleC` language using intervals. Section 3.2 focuses on two (explicit) relational domains, namely the Polyhedra abstract domain 3.2, and the Linear Templates abstract domain 3.2. The affine arithmetic as well as most of its known extensions are covered by Section 3.3. The Perturbed Affine Sets abstract domain is briefly introduced in Section 3.3. The last section summarizes the main ideas behind the reduced product (Section 3.4) and the logical product (Section 3.4) of abstract domains.

3.1 Non Relational Abstract Domain

Interval Arithmetic

Interval Arithmetic [MY59] stands for rules that govern the computations using intervals instead of real numbers. The basic arithmetic operations over intervals are reduced to simple operations on operands' bounds. For any operation $\circ \in \{+, -, \times, \div\}$, we have:

$$[\underline{\boldsymbol{u}}, \overline{\boldsymbol{u}}] \circ [\underline{\boldsymbol{v}}, \overline{\boldsymbol{v}}] \stackrel{\text{def}}{=} [\min\{\underline{\boldsymbol{u}} \circ \underline{\boldsymbol{v}}, \underline{\boldsymbol{u}} \circ \overline{\boldsymbol{v}}, \overline{\boldsymbol{u}} \circ \underline{\boldsymbol{v}}, \overline{\boldsymbol{u}} \circ \overline{\boldsymbol{v}}\}, \max\{\underline{\boldsymbol{u}} \circ \underline{\boldsymbol{v}}, \underline{\boldsymbol{u}} \circ \overline{\boldsymbol{v}}, \overline{\boldsymbol{u}} \circ \underline{\boldsymbol{v}}, \overline{\boldsymbol{u}} \circ \overline{\boldsymbol{v}}\}] \ . \quad (3.1.1)$$

Computing all the combinations of bounds is not necessary for all operations listed above. For instance,

$$[\underline{\boldsymbol{u}}, \overline{\boldsymbol{u}}] + [\underline{\boldsymbol{v}}, \overline{\boldsymbol{v}}] \stackrel{\text{def}}{=} [\underline{\boldsymbol{u}} + \underline{\boldsymbol{v}}, \overline{\boldsymbol{u}} + \overline{\boldsymbol{v}}], \quad (3.1.2)$$

$$[\underline{\boldsymbol{u}}, \overline{\boldsymbol{u}}] - [\underline{\boldsymbol{v}}, \overline{\boldsymbol{v}}] \stackrel{\text{def}}{=} [\underline{\boldsymbol{u}} - \overline{\boldsymbol{v}}, \overline{\boldsymbol{u}} - \underline{\boldsymbol{v}}] \ . \quad (3.1.3)$$

For the division operation, whenever the denominator interval contains zero, the result is undefined, and the only possible result is the real line, that is $\mathbb{R}$.

Observe that, an interval does not have an additive inverse, and $\boldsymbol{i} - \boldsymbol{i}$ does not vanish to zero (e.g. $[1, 2] - [1, 2] = [-1, 1]$). Similarly, an interval does not have a multiplicative inverse, and $\boldsymbol{i} \div \boldsymbol{i}$, assuming that 0 is not within $\boldsymbol{i}$, is not equal to 1 (e.g. $[1, 2] \div [1, 2] = [0.5, 2]$). Furthermore, the multiplication operation ($\times$) does not distribute over the addition operation ($+$). We have instead a weaker notion of sub-distributivity (with respect to the relation order of inclusion over intervals).

$$\boldsymbol{w} \times (\boldsymbol{u} + \boldsymbol{v}) \subseteq \boldsymbol{w} \times \boldsymbol{u} + \boldsymbol{w} \times \boldsymbol{v}$$

For instance, $[1, 2] \times ([-3, -2] + [3, 4])$ is equal to $[0, 4]$ if the addition operation is evaluated first, and $[-3, 6]$ if we distribute the multiplication operation. Both intervals contain all true values, but the former is tighter (in the sense that $[0, 4] \subseteq [-3, 6]$). Also, the square of an interval, may contain non-positive values, e.g. $[-1, 2] \times [-1, 2] = [-2, 4]$.

The main reason behind these subtleties in arithmetic operations over intervals comes from the fact that substituting variables by their respective intervals loses all relations between the involved variables. As a matter of fact, if the variables x and y are within the same interval $[-1, 2]$, then evaluating x^2 or xy in interval arithmetic leads to the same interval operation, namely $[-1, 2] \times [-1, 2]$ $(= [-2, 4])$. Whereas, all values in $[-2, 4]$ are possible for xy, only the positive values, that is the interval $[0, 4]$, may occur for x^2.

Intervals Abstract Domain

We abstract our concrete semantics (see equation (2.2.1)) using the lattice of intervals. If $(\mathcal{D}, \leq, \cup, \cap, \bot, \top)$ is a complete lattice (resp. complete poset, poset), and V is a set, then $(V \to \mathcal{D}, \dot{\leq}, \dot{\cup}, \dot{\cap}, \dot{\bot}, \dot{\top})$ is a complete lattice (resp. complete poset, poset) called the *structural lifting* of $\mathcal{D}$. The pointwise lifting operations are defined as follows

$$\begin{aligned} X \dot{\leq} Y &\overset{\text{def}}{\iff} \forall v \in V, X(v) \leq Y(v) \\ \dot{\cup}\mathcal{X}(v) &\overset{\text{def}}{=} \cup\{X(v) \mid X \in \mathcal{X}\} \\ \dot{\cap}\mathcal{X}(v) &\overset{\text{def}}{=} \cap\{X(v) \mid X \in \mathcal{X}\} \\ \dot{\bot}(v) &\overset{\text{def}}{=} \bot \\ \dot{\top}(v) &\overset{\text{def}}{=} \top \end{aligned}$$

The abstract program environment $\Sigma^\sharp : \mathcal{V} \to \mathbb{I}$ maps each variable to an interval. The abstract domain $\mathcal{D}^\sharp \overset{\text{def}}{=} (\Sigma^\sharp, \subseteq^\sharp, \cup^\sharp, \cap^\sharp, \bot^\sharp, \top^\sharp)$ is the structural lifting of the lattice of boxes $(\mathbb{I}, \subseteq_\iota, \cup_\iota, \cap_\iota, \emptyset, [-\infty, +\infty])$, defined as follows:

$$\begin{aligned} \boldsymbol{i} \subseteq_\iota \boldsymbol{j} &\overset{\text{def}}{\iff} \sup(\boldsymbol{i}) \leq \sup(\boldsymbol{j}) \wedge \inf(\boldsymbol{j}) \leq \inf(\boldsymbol{i}) \\ \boldsymbol{i} \cup_\iota \boldsymbol{j} &\overset{\text{def}}{\iff} [\min\{\inf(\boldsymbol{i}), \inf(\boldsymbol{j})\}, \max\{\sup(\boldsymbol{i}), \sup(\boldsymbol{j})\}] \\ \boldsymbol{i} \cap_\iota \boldsymbol{j} &\overset{\text{def}}{\iff} [\max\{\inf(\boldsymbol{i}), \inf(\boldsymbol{j})\}, \min\{\sup(\boldsymbol{i}), \sup(\boldsymbol{j})\}] \end{aligned}$$

The abstract evaluation of arithmetic expressions is then given by:

$$\begin{aligned} &\forall e \in expr, [\![e]\!]^\sharp : \Sigma^\sharp \to \mathbb{I} \\ &[\![v]\!]^\sharp \sigma^\sharp \overset{\text{def}}{=} \sigma^\sharp(v) \\ &[\![[a,b]]\!]^\sharp \sigma \overset{\text{def}}{=} [a,b] \\ &[\![e_1 \diamond e_2]\!]^\sharp \sigma^\sharp \overset{\text{def}}{=} [\![e_1]\!]^\sharp \sigma^\sharp \diamond_\iota [\![e_2]\!]^\sharp \sigma^\sharp \\ &\quad \text{where} \diamond_\iota \in \{+_\iota, -_\iota, \times_\iota\} \\ &[\![e_1 \div e_2]\!]^\sharp \sigma \overset{\text{def}}{=} \begin{cases} [-\infty, +\infty], \text{if } 0 \in [\![e_2]\!]^\sharp \sigma \\ [\![e_1]\!]\sigma \div_\iota [\![e_2]\!]^\sharp \sigma, \text{otherwise.} \end{cases} \end{aligned}$$

We now give a sound abstraction of the concrete operator [CC77]:

$$\begin{aligned}
[\![s]\!]^\sharp &: \mathcal{D}^\sharp \to \mathcal{D}^\sharp \\
[\![v \leftarrow e]\!]^\sharp\sigma^\sharp &\stackrel{\text{def}}{=} \sigma^\sharp[v \mapsto [\![e]\!]\sigma^\sharp] \\
[\![\textbf{if } b \textbf{ then } s]\!]^\sharp\sigma^\sharp &\stackrel{\text{def}}{=} [\![s]\!]^\sharp \circ [\![b]\!]^\sharp\sigma^\sharp \cup^\sharp [\![\neg b]\!]^\sharp\sigma^\sharp \\
[\![\textbf{while } b \textbf{ do } s]\!]^\sharp\sigma^\sharp &\stackrel{\text{def}}{=} [\![\neg b]\!]^\sharp(\mathrm{lfp}_{\sigma^\sharp}\, \lambda X.X \cup^\sharp ([\![s]\!]^\sharp \circ [\![b]\!]^\sharp)X) \\
\text{where } [\![b]\!]^\sharp\sigma^\sharp &\stackrel{\text{def}}{=} \cup^\sharp\{\phi^\sharp \mid b \text{ is true}\} \cap^\sharp \sigma^\sharp.
\end{aligned}$$

The abstract environment $\sigma^\sharp[v \mapsto [\![e]\!]\sigma^\sharp]$ maps each variable v to the interval given by $[\![e]\!]\sigma^\sharp$, that is the evaluation of the expression e when the variables are within $\sigma^\sharp(v_1) \times \ldots \times \sigma^\sharp(v_p)$. For instance, if $e = v - c$ where $v \in \mathcal{V}$, then

$$[\![e \leq 0?]\!]^\sharp\sigma^\sharp = \sigma^\sharp[v \mapsto \sigma^\sharp(v) \cap [-\infty, c]]$$

Indeed, $[\![v - c]\!]^\sharp\phi^\sharp = \phi^\sharp(v) - [c, c]$, and the least upper bound of the set of intervals $\{\boldsymbol{i} \mid \boldsymbol{i} - [c, c] \leq 0\}$ is equal to $[-\infty, c]$. Similarly, we have

$$\begin{aligned}
[\![v - c \geq 0?]\!]^\sharp\sigma^\sharp &= [\![-e \leq 0?]\!]^\sharp\sigma^\sharp \\
&= \sigma^\sharp[v \mapsto \sigma^\sharp(v) \cap -[-\infty, -c]] \\
&= \sigma^\sharp[v \mapsto \sigma^\sharp(v) \cap [c, +\infty]] \ .
\end{aligned}$$

3.1.1 Example (Abstraction using intervals)
We compute the least fixpoint of the classical loop program [CC77] given below.

```
x ← 0;
while (x ≤ 100) do
   x ← x + 1;
return x;
```

We are interested in the final value of variable x for any execution of the above simple loop. The behavior of the program can be then formalized by:

$$P^\flat \stackrel{\text{def}}{=} [\![\textbf{while } x \leq 100 \textbf{ do}(x \leftarrow x + 1)]\!]^\flat \circ [\![x \leftarrow 0]\!]^\flat\bot^\flat,$$

where $\bot^\flat$ denotes the environment that maps $\mathcal{V} \stackrel{\text{def}}{=} \{x\}$ to the empty set (the bottom element of the partition set $\wp(\mathbb{R})$). We compute the fixpoint with respect to the intervals abstract domain earlier defined. The concretisation of the abstract fixpoint gives an over-approximation of the fixpoint

we seek. We have:

$$\begin{aligned}
\bot^\sharp(x) &= \emptyset \\
[\![x \leftarrow 0]\!]^\sharp \bot^\sharp &= \bot^\sharp[x \mapsto [0,0]] \\
[\![x \leftarrow x+1]\!]^\sharp \sigma^\sharp &= \sigma^\sharp[x \mapsto \sigma^\sharp(x) + [1,1]] \\
[\![x - 100 \leq 0?]\!]^\sharp \sigma^\sharp &= \sigma^\sharp[x \mapsto \sigma^\sharp(x) \cap [-\infty, 100]] \\
[\![x - 100 \geq 0?]\!]^\sharp \sigma^\sharp &= \sigma^\sharp[x \mapsto \sigma^\sharp(x) \cap [100, +\infty]]
\end{aligned}$$

The semantics of the program in the abstract domain is given by

$$P^\sharp \stackrel{\text{def}}{=} [\![\textbf{while } x \leq 100 \textbf{ do}(x \leftarrow x+1)]\!]^\sharp \bot^\sharp[x \mapsto [0,0]] \ .$$

It involves a fixpoint computation:

$$\text{lfp}_{\bot^\sharp[x \mapsto [0,0]]} \lambda\sigma^\sharp . F^\sharp(\sigma^\sharp),$$

where $F^\sharp(\sigma^\sharp) \stackrel{\text{def}}{=} \sigma^\sharp \cup^\sharp (\phi^\sharp[x \mapsto \phi^\sharp(x) + [1,1]])$, $\phi^\sharp \stackrel{\text{def}}{=} \sigma^\sharp[x \mapsto \sigma^\sharp(x) \cap [-\infty, 100]]$.

We use the Kleene iteration technique to compute such fixpoint. The operator $F^\sharp$ is by construction monotonic.

We start the iteration with $\bot^\sharp[x \mapsto [0,0]]$, at each iteration we evaluate $\phi^\sharp$ then $F^\sharp$:

$$\begin{aligned}
\phi^\sharp &= \bot^\sharp[x \mapsto [0,0] \cap [-\infty, 100]] \\
&= \bot^\sharp[x \mapsto [0,0]] \ .
\end{aligned}$$

The first iteration $(F^\sharp)^1$ gives:

$$\begin{aligned}
(F^\sharp)^1 \bot^\sharp[x \mapsto [0,0]] &= \bot^\sharp[x \mapsto [0,0]] \cup^\sharp \phi^\sharp[x \mapsto \phi^\sharp(x) + [1,1]] \\
&= \bot^\sharp[x \mapsto [0,0]] \cup^\sharp \bot^\sharp[x \mapsto [1,1]] \\
&= \bot^\sharp[x \mapsto [0,1]]
\end{aligned}$$

After 101 iterations, we obtain $(F^\sharp)^{101} \bot^\sharp[x \mapsto [0,0]] = \bot^\sharp[x \mapsto [0,101]]$, and for the iteration 102,

$$\begin{aligned}
\phi^\sharp &= \bot^\sharp[x \mapsto [0,101] \cap [-\infty, 100]] \\
&= \bot^\sharp[x \mapsto [0,100]] \ .
\end{aligned}$$

and

$$\begin{aligned}(F^\sharp)^{102}\bot^\sharp[x \mapsto [0,0]] &= F^\sharp(\bot^\sharp[x \mapsto [0,101]]) \\ &= \bot^\sharp[x \mapsto [0,101]] \cup^\sharp \phi^\sharp[x \mapsto \phi^\sharp(x) + [1,1]] \\ &= \bot^\sharp[x \mapsto [0,101]] \cup^\sharp \bot^\sharp[x \mapsto [0,100] + [1,1]] \\ &= \bot^\sharp[x \mapsto [0,101]] \cup^\sharp \bot^\sharp[x \mapsto [0,101]] \\ &= \bot^\sharp[x \mapsto [0,101]] \\ &= (F^\sharp)^{101}\bot^\sharp[x \mapsto [0,0]] \ .\end{aligned}$$

The increasing chain stabilizes at the iteration 102, its limit $\bot^\sharp[x \mapsto [0,101]]$ is the least fixpoint of the operator $F^\sharp$. We finally obtain a sound superset of the values of the variable x:

$$P^\flat \leq^\flat \gamma(\bot^\sharp[x \mapsto [100,101]]) = [100,101] \ .$$

3.2 Explicit Relational Abstract Domains

Polyhedra Abstract Domains

The polyhedra abstract domain [CH78] catches and propagates explicit linear relations between variables such that $b_1x_1 + b_2x_2 \leq \beta$, where x and y are two numerical variables and α, β, and δ are real numbers.

The internal abstract object has two dual representations: an *external representation* and an *internal representation.*

external representation: a polyhedron is defined as the intersection of a finite set of affine subspaces of $\mathbb{R}^p$, where p is the number of numerical variables abstracted. Each affine subspace is in fact represented by $\langle x, \alpha_i \rangle \leq \beta_i$. Equalities, $\langle x, \alpha_i \rangle = \beta_i$, are represented by two inequalities, $\langle x, \alpha_i \rangle \leq \beta_i$ and $\langle x, -\alpha_i \rangle \leq -\beta_i$. A polyhedron is then by definition a convex subset of $\mathbb{R}^p$.

internal representation: a polyhedron is *generated* by a finite set of vertices $\{v_i, 1 \leq i \leq k\}$and a finite set of rays $\{r_i, 1 \leq j \leq m\}$. The polyhedron consists of all vectors of the form

$$\{\sum_{i=1}^{k} \lambda_i v_i + \sum_{j=1}^{m} \beta_j r_j \mid \lambda_i \geq 0, \beta_j \geq 0, \sum_{i=1}^{k} \lambda_i = 1\} \ .$$

The duality of both representations is a classical result (see for instance [Roc70, Theorem 19.1]).

The intersection of two polyhedra is exactly represented by a polyhedron, the join of two polyhedra is their convex hull which is also a polyhedron. Using the convex hull as a join and the geometrical inclusion as a partial order, the set of polyhedra defines a lattice. The lattice is not complete, as we can extract an infinitely increasing chain which converges toward a circle (regular polygons inside a circle).

The set-theoretic operations (the meet and the join) are easier to define using the external representation: the intersection is for instance the concatenation of both lists of inequalities of the given polyhedra. The assignment operations are more convenient to define using the internal representation, for instance any linear expression can be immediately computed (linear time) using the generators representation. Therefore, the transformation from one representation to the other is a central operation to define an abstract domain using polyhedra.

The Chernikova's algorithm enhanced by Le Verge [Le 92] permits such transformation. The transformation minimizes the number of constraints (or generators) in the resulting polyhedron. Its worst-time complexity is exponential in time and in memory cost (function of number of variables p). The minimal number of generators needed to represent a given polyhedra in its minimal inequalities representation can be exponential.

The algorithm is implemented in two recent libraries, the Parma Polyhedra Library [Pro] and the Polka Library [Ja]. In practice the exponential complexity can be observed even for simple programs with 5 variables but which alternate set-theoretic operations and assignments. The polyhedra abstract domains in APRON Library are based on these two implementations.

Linear Templates

Polyhedra Templates abstract domain [SSM05] overcomes the complexity problem observed in classical polyhedra by fixing the directions of the polyhedra used during the analysis. The directions are fixed using a template constraint matrix T and only inequalities of the form $T(x_1, \ldots, x_p) + c \geq 0$, are allowed, where T is an $m \times p$ matrix, and $c \in \mathbb{R}^m$. Such an approach limits the expressiveness of the domain compared to the polyhedra domain; however, it improves the worst-case complexity from exponential (in the the classical polyhedra case) to polynomial with respect to the number of program variables.

As the internal representation of Templates is fixed, there is no more need to switch between the external and the internal representations of the polyhedron. The abstraction of linear assignment involves m linear

problems to solve: as any linear assignment can be expressed as $Ax+b \geq 0$, (where A is an $n \times p$ matrix and $b \in \mathbb{R}^n$) and the templates matrix T is fixed, it suffices to find the smallest c_i, $1 \leq i \leq m$, such that $Tx + c \geq 0$ satisfies $Ax+b \geq 0$. If we apply Farka's Lemma [Roc70, Theorem 22.3], such problem is transformed into m linear programs, involving each n variables and p constraints.

The geometrical inclusion is no more a partial order because of the fixed shape of the polyhedra, it is instead a pre-order, which is quotiented by an equivalence class to define a partial order. The intersection and the join operators are defined piecewisely using respectively the minimum and the maximum operators over the vectors "c" of the operands.

The synthesis of matrix T which fixes the directions of the analysis may benefit from the initial constraints, guards or properties one needs to prove. Authors in [SSM05] introduce the notion of *support vector* which gives additional constraints one can include in T. The support vector of a coefficient vector a is defined with respect to a (linear) transition system $x \leftarrow Ax + b$, by A^*a (where * denotes the transpose operator). In fact, if $\langle a, x\rangle \geq 0$, and we substitute x by $Ax+b$, then the new coefficient vector of x is A^*a (using the classical duality of the scalar product). Of course, these heuristics may not in general be optimal to catch exactly the invariants (even linear) of a given problem.

Discussion Observe that linear templates generalize the octagons abstract domain as one can easily define matrix T to catch exactly the inequalities of the form $\pm x_i \pm x_j \leq c_i$. Thus, the domain is at least as expressive as octagons and strictly less expressive than classical polyhedra. However, the native octagons abstract domain is more efficient than this generic approach. The internal representation using Differential Bound Matrices (DBM) is more efficient (cubical complexity). Moreover, for a sound and efficient implementation of the templates abstract domain, one needs to use a guaranteed LP solver, such as [Kei05] [1], in order to use safely floating-point numbers, which is expensive as it solves many times the same problem to come up with safe bounds.

[1] A recent and interesting survey of software packages for verified linear programming can be found in [Kei08].

3.3 Implicit Relational Abstract Domains

Affine Arithmetic

Affine arithmetic has been successfully used in many fields from the self-validation of numerical algorithms [CS93] where it was firstly introduced, to reliable computing to come up with tight enclosing intervals [Kol01, Kol04, Miy00, MK04a], and algebraic surface plotting [SLMW06].

In what follows, we motivate the use of affine forms by the dependency problems observed in both interval arithmetic, and an extension of IA, called *generalized intervals*.

Generalized Intervals

To overcome this dependency problem observed in interval arithmetic, Eldon R. Hansen proposed in 1975 an extension to intervals, called *Generalized Intervals* [Han75]:

3.3.1 Definition (Generalized Intervals)
A generalized interval, also known as Hansen's form, $\tilde{x}$ is defined by

$$\tilde{x} \stackrel{\text{def}}{=} \boldsymbol{c_0^x} + \boldsymbol{c_1^x}\zeta_1 + \cdots + \boldsymbol{c_n^x}\zeta_n = \boldsymbol{c_0^x} + \sum_{i=1}^{n} \boldsymbol{c_i^x}\zeta_i$$

where $\{\boldsymbol{c_i^x}\}_{0\leq i\leq n}$ are intervals and $\{\zeta_i\}_{1\leq i\leq n}$ are symbolic variables known to be in centred intervals (of the form $[-r_i, r_i]$). The parameter n, number of ζ_i is fixed for all generalized intervals.

The symbolic variables $\{\zeta_i\}_{1\leq i\leq n}$ express the dependency between variables. Conversions to and from classical intervals are defined as follow:

3.3.2 Definition (Conversion from interval)
Let $\boldsymbol{x}$ be an element of $\mathbb{I}$. If $\boldsymbol{x}$ has an infinite bound, then $\boldsymbol{c_0^x} \stackrel{\text{def}}{=} \boldsymbol{x}$, and $\boldsymbol{c_i^x} = [0, 0]$, for all i in $1 \ldots n$. Else,

$$\tilde{x} \stackrel{\text{def}}{=} [\text{mid}(\boldsymbol{x}), \text{mid}(\boldsymbol{x})] + [1, 1]\zeta_1,$$

where ζ_1 is unknown but has its values within $[-\text{dev}(\boldsymbol{x}), \text{dev}(\boldsymbol{x})]$.

3.3.3 Definition (Conversion to interval)
The interval related to $\tilde{x} = \boldsymbol{c_0^x} + \sum_{i=1}^{n} \boldsymbol{c_i^x}\zeta_i$ is the result of the evaluation in interval arithmetic of the following expression:

$$\boldsymbol{x} \stackrel{\text{def}}{=} \boldsymbol{c_0^x} + \sum_{i=1}^{n} \boldsymbol{c_i^x}[-r_i, r_i] \ .$$

Let $\tilde{x}$ and $\tilde{y}$ be two generalized intervals. We define the addition, subtraction and scalar multiplication by

$$\tilde{x} + \tilde{y} \stackrel{\text{def}}{=} (\boldsymbol{c_0^x} + \boldsymbol{c_0^y}) + \sum_{i=1}^{n} (\boldsymbol{c_i^x} + \boldsymbol{c_i^y})\zeta_i \tag{3.3.1}$$

$$\tilde{x} - \tilde{y} \stackrel{\text{def}}{=} (\boldsymbol{c_0^x} - \boldsymbol{c_0^y}) + \sum_{i=1}^{n} (\boldsymbol{c_i^x} - \boldsymbol{c_i^y})\zeta_i \tag{3.3.2}$$

$$\lambda\tilde{x} \stackrel{\text{def}}{=} (\lambda\boldsymbol{c_0^x}) + \sum_{i=1}^{n} (\lambda\boldsymbol{c_i^x})\zeta_i \tag{3.3.3}$$

The operations on coefficients, $\boldsymbol{c_i^x}$ and $\boldsymbol{c_i^y}$, are computed w.r.t. interval arithmetic introduced above (Equation 3.1.1).

Using generalized intervals and their related arithmetic instead of interval arithmetic is more expensive in time and memory since one has to record n intervals instead of just one interval. Nevertheless, it is possible to track relations between variables using the shared ζ_i between variables. For example, suppose that we have to compute an enclosure of the direct image of the function $f\colon [0,1] \to \mathbb{R}$ defined by $f(x) = 2x - x$. The exact interval is $[0,1]$ for $f(x)$ since $f(x) = x$. Using interval arithmetic, we obtain $f(x) \in [2,2] \times [0,1] - [0,1] = [0,2] - [0,1] = [-1,2]$ [2]. This catastrophic result can be improved using generalized intervals: we first convert $[0,1]$ into its related generalized interval, $\tilde{x} = [0.5, 0.5] + [1,1]\zeta_1$, where ζ_1 is within $[-0.5, 0.5]$. Then the generalized interval of $f(x)$ is $2 \times \tilde{x} - \tilde{x} = [0.5, 0.5] + [0.5, 0.5]\zeta_1$. Converted to an interval, the generalized interval of $f(x)$ gives the exact result, that is $[0,1]$. Moreover, we have the relation $f(x) = x$, encoded implicitly, since x and $f(x)$ have the same generalized interval.

Coefficients of generalized intervals are intervals. The lack of precision observed in interval arithmetic may also happen for these coefficients, which prevents the needed cancellation to occur. The next section presents *Affine Forms*, a special case of Hansen's forms where the coefficients are real numbers instead of intervals.

Affine Forms

Affine Forms [CS93] were introduced in 1993 by João L. D. Comba and Jorge Stolfi. They are defined as follows:

[2] For this simple example, one could use symbolic enhancement which addresses locally such problem; however, we could easily imagine that $f(x)$ is computed using intermediate variables. Therefore, the symbolic computation is no more immediate.

3.3.4 Definition (Affine forms)
An affine form $\hat{a}$ is defined by

$$\hat{a} \stackrel{\text{def}}{=} \alpha_0^a + \alpha_1^a \epsilon_1 + \cdots + \alpha_n^a \epsilon_n = \alpha_0^a + \sum_{i=1}^{n} \alpha_i^a \epsilon_i,$$

where $\alpha_0^a, \ldots, \alpha_n^a$ are real coefficients, called partial deviations, and $\epsilon_1, \ldots, \epsilon_n$ are symbolic variables, called noise symbols, known to be within $[-1, 1]$. The number of noise symbols is not a priori fixed.

The set of affine forms is denoted by $\mathbb{A}$.

In Definition 3.3.4 coefficients, or partial deviations, $\{\alpha_i^a\}_{0 \leq i \leq n}$ are real numbers, and not intervals as seen in Hansen's forms. Affine forms can be seen as special generalized intervals where all coefficients are points intervals, and each symbol ζ_i is reduced to $\text{dev}(\boldsymbol{\zeta_i})\epsilon_i$.

3.3.5 Definition (Conversion to intervals)
The range of an affine form $\hat{a}$, denoted by the bold face notation $\hat{\boldsymbol{a}}$, is

$$\hat{\boldsymbol{a}} \stackrel{\text{def}}{=} [\alpha_0^a - \sum_{i=1}^{n} |\alpha_i^a|, \alpha_0^a + \sum_{i=1}^{n} |\alpha_i^a|] \ .$$

3.3.6 Definition (Conversion from intervals)
If $\boldsymbol{i}$ is a bounded interval in $\mathbb{I}$, then

$$\hat{i} \stackrel{\text{def}}{=} \text{mid}(\boldsymbol{i}) + \text{dev}(\boldsymbol{i})\epsilon_f \ .$$

The noise symbol ϵ_f is a fresh noise symbol not used elsewhere. Unbounded intervals and the empty set can not be converted to affine forms with respect to Definition 3.3.4, since all the coefficients of affine forms are finite real numbers. The convention uses intervals without any further transformation to handle unbounded intervals.

The *joint range* of a set of affine forms $A \stackrel{\text{def}}{=} \{\hat{a}_1, \ldots, \hat{a}_p\}$ for $p \geq 2$, is the set of all possible values taken by $(\hat{a}_1, \ldots, \hat{a}_p)$ whenever the vector $(\epsilon_1, \ldots, \epsilon_n)$ ranges over $\mathcal{B}^n$. Formally, the joint range is the image of $\{1\} \times \mathcal{B}^n$ under the linear transformation defined by matrix $C^A \in \mathcal{M}(p, n+1)$

$$\begin{pmatrix} \hat{a}_1 \\ \vdots \\ \hat{a}_p \end{pmatrix} = \underbrace{\begin{pmatrix} \alpha_0^{a_1} & \cdots & \alpha_n^{a_1} \\ \vdots & & \vdots \\ \alpha_0^{a_p} & \cdots & \alpha_n^{a_p} \end{pmatrix}}_{C^A} \times \begin{pmatrix} 1 \\ \epsilon_1 \\ \vdots \\ \epsilon_n \end{pmatrix}$$

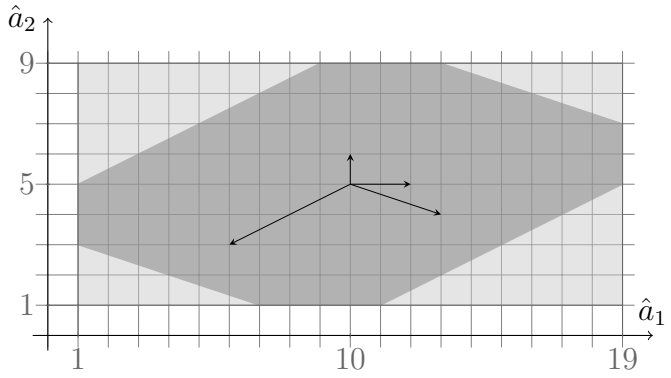

Figure 3.1: The joint range (the center symmetric polygon in dark gray) of two affine forms. This zonotope is spanned by the generators drawn in its center. The box $\hat{\boldsymbol{a}}_1 \times \hat{\boldsymbol{a}}_2$ (light gray) encloses the zonotope.

$$\gamma(A) \stackrel{\text{def}}{=} \{C^A \varepsilon \mid \varepsilon \in \{1\} \times \mathcal{B}^n\} \subset \mathbb{R}^p \ .$$

The geometrical concretisation $\gamma(A)$ is a center symmetric polytope called *zonotope*. The center is the vector given by the first column of matrix C^A. The other vectors of C^A, that is, $(\alpha_i^{a_1}, \ldots, \alpha_i^{a_p})^*$, for $1 \leq i \leq n$, are the generators of the zonotope (see Figure 3.1).

3.3.7 Example (Joint range of two affine forms)

Let A be the set of affine forms defined by

$$A = \begin{pmatrix} \hat{a}_1 \\ \hat{a}_2 \end{pmatrix} = \begin{pmatrix} 10 & -4\epsilon_1 & & +1\epsilon_3 & +3\epsilon_4 \\ 5 & -2\epsilon_1 & +1\epsilon_2 & & -1\epsilon_4 \end{pmatrix} \ .$$

The joint range (the zonotope), $\gamma(A)$, together with the box $\hat{\boldsymbol{a}}_1 \times \hat{\boldsymbol{a}}_2$ are shown in Figure 3.1. Shared noise symbols ϵ_1 and ϵ_4 give extra information about the relative correlations between variables $\hat{a}_1$ and $\hat{a}_2$.

The linear operations over affine forms are straightforward:

3.3.8 Definition (Linear operations)
Let $\hat{a}$ and $\hat{b}$ be two affine forms, let λ, ζ be two finite real numbers, then

$$\hat{a} \pm \hat{b} \stackrel{\text{def}}{=} (\alpha_0^a \pm \alpha_0^b) + \sum_{i=1}^{n} (\alpha_i^a \pm \alpha_i^b)\epsilon_i \tag{3.3.4}$$

$$\lambda\hat{a} \stackrel{\text{def}}{=} \lambda\alpha_0^a + \sum_{i=1}^{n} (\lambda\alpha_i^a)\epsilon_i \tag{3.3.5}$$

$$\hat{a} + \zeta \stackrel{\text{def}}{=} (\alpha_0^a + \zeta) + \sum_{i=1}^{n} \alpha_i^a \epsilon_i \tag{3.3.6}$$

Non affine operations over affine forms have to be linearized. This is achieved by over-approximating the error introduced by the linearization, then adding a *fresh noise symbol* that is a noise symbol which is not used elsewhere for any affine forms.

The multiplication operation motivates many extensions, detailed hereafter, of affine forms in order to reduce the range of the final result.

The multiplication operation Let $\hat{a}$ and $\hat{b}$ be two affine forms, then

$$\hat{a} \times \hat{b} = \left(\alpha_0^a + \sum_{i=1}^{n} \alpha_i^a \epsilon_i\right)\left(\alpha_0^b + \sum_{j=1}^{n} \alpha_j^b \epsilon_j\right) \tag{3.3.7}$$

$$= \alpha_0^a\alpha_0^b + \sum_{i=1}^{n} (\alpha_0^a\alpha_i^b + \alpha_0^b\alpha_i^a)\epsilon_i + \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i^a\alpha_j^b\epsilon_i\epsilon_j \tag{3.3.8}$$

The non-linear term (actually quadratic) term in Equation 3.3.7 is linearized in several ways. All of them bound first the non-linear term, then convert it to an affine form (using Definition 3.3.6).

- **centered form [dFS97]:** uses a generous range for $\alpha_i^a\alpha_i^b\epsilon_i\epsilon_j$, that is $|\alpha_i^a\alpha_i^b|$, then

$$\hat{a} \times \hat{b} \stackrel{\text{def}}{=} \alpha_0^a\alpha_0^b + \sum_{i=1}^{n} (\alpha_0^a\alpha_i^b + \alpha_0^b\alpha_i^a)\epsilon_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} |\alpha_i^a\alpha_j^b|\epsilon_{n+1} \ .$$

- **decentered form [Miy00]:** uses the fact that for $i = j$, the values of $\epsilon_i\epsilon_j$ are within $[0, 1]$; thus,

$$\begin{aligned} \hat{a} \times \hat{b} \stackrel{\text{def}}{=} & \left(\alpha_0^a\alpha_0^b + \frac{1}{2}\sum_{i=1}^{n} \alpha_i^a\alpha_i^b\right) + \sum_{i=1}^{n} (\alpha_0^a\alpha_i^b + \alpha_0^b\alpha_i^a)\epsilon_i \\ & + \left(\frac{1}{2}\sum_{i=1}^{n} |\alpha_i^a\alpha_i^b| + \sum_{1\le i<j\le n} |\alpha_i^a\alpha_j^b + \alpha_j^a\alpha_i^b|\right)\epsilon_{n+1} \ . \end{aligned} \tag{3.3.9}$$

- **extended form [Mes02]:** Messine introduces two extensions of affine forms to increase the accuracy of even power of affine forms.

3.3.9 Definition (Extended Form)
A Messine affine form is defined by

$$\breve{a} \stackrel{\text{def}}{=} \alpha_0^a + \sum_{i=1}^{n} \alpha_i^a \epsilon_i + \alpha_{n+1}^a [-1,1] + \alpha_{n+2}^a [0,1] + \alpha_{n+3}^a [-1,0]$$

where, $\alpha_0^a, \ldots, \alpha_n^a$ are real numbers, and α_{n+1}^a, α_{n+2}^a, α_{n+3}^a are positive real numbers.

3.3.10 Definition (Quadratic Form)
A quadratic form adds n new non-negative noise symbols to represent exactly square noise symbols, ϵ_i^2 $(= \epsilon_{i+n})$:

$$\breve{\breve{a}} \stackrel{\text{def}}{=} \alpha_0^a + \sum_{i=1}^{n} \alpha_i^a \epsilon_i + \alpha_{i+n}^a \epsilon_{i+n} + \alpha_{2n+1}^a [-1,1] + \alpha_{2n+2}^a [0,1] + \alpha_{2n+3}^a [-1,0]$$

where, $\alpha_0^a, \ldots, \alpha_{2n}^a$ are real numbers, and α_{2n+1}^a, α_{2n+2}^a, α_{2n+3}^a are positive real numbers. The noise symbols $\{\epsilon_{i+n}\}_{1 \le i \le n}$ are constrained to have their values within $[0,1]$.

The extra information recorded improves in general multiplication results but increases significantly the complexity of operations. The multiplication of two extended forms is detailed below. The multiplication of two quadratic forms is similar, details can be found in [Mes02].

$$\breve{a} \times \breve{b} \stackrel{\text{def}}{=} \alpha_0^a \alpha_0^b + \sum_{i=1}^{n} (\alpha_0^a \alpha_i^b + \alpha_0^b \alpha_i^a) \epsilon_i + K_1 [-1,1] + K_2 [0,1] + K_3 [-1,0],$$

where

$$K_1 = |\alpha_0^a| \alpha_{n+1}^b + |\alpha_0^b| \alpha_{n+1}^a + \alpha_{n+1}^a \alpha_{n+1}^b + \sum_{\substack{1 \le i,j \le n+3 \\ i \ne j}} |\alpha_i^a \alpha_j^b|,$$

$$K_2 = K_2^0 + \alpha_{n+2}^a \alpha_{n+2}^b + \alpha_{n+3}^a \alpha_{n+3}^b + \sum_{\substack{1 \le i \le n \\ \alpha_i^a \alpha_i^b \ge 0}} \alpha_i^a \alpha_i^b,$$

$$K_3 = K_3^0 + \sum_{\substack{1 \le i \le n \\ \alpha_i^a \alpha_i^b \le 0}} |\alpha_i^a \alpha_i^b|,$$

where, in turn, K_2^0 and K_3^0 are defined as follows

$$K_2^0 = \begin{cases} \alpha_0^a \alpha_{n+2}^b + \alpha_0^b \alpha_{n+2}^a & \text{if } \alpha_0^a > 0 \text{ and } \alpha_0^b > 0 \\ \alpha_0^a \alpha_{n+2}^b - \alpha_0^b \alpha_{n+3}^a & \text{if } \alpha_0^a > 0 \text{ and } \alpha_0^b < 0 \\ -\alpha_0^a \alpha_{n+3}^b + \alpha_0^b \alpha_{n+2}^a & \text{if } \alpha_0^a < 0 \text{ and } \alpha_0^b > 0 \\ -\alpha_0^a \alpha_{n+3}^b - \alpha_0^b \alpha_{n+3}^a & \text{if } \alpha_0^a < 0 \text{ and } \alpha_0^b < 0 \end{cases},$$

$$K_3^0 = \begin{cases} \alpha_0^a \alpha_{n+3}^b + \alpha_0^b \alpha_{n+3}^a & \text{if } \alpha_0^a > 0 \text{ and } \alpha_0^b > 0 \\ \alpha_0^a \alpha_{n+3}^b - \alpha_0^b \alpha_{n+2}^a & \text{if } \alpha_0^a > 0 \text{ and } \alpha_0^b < 0 \\ -\alpha_0^a \alpha_{n+2}^b + \alpha_0^b \alpha_{n+3}^a & \text{if } \alpha_0^a < 0 \text{ and } \alpha_0^b > 0 \\ -\alpha_0^a \alpha_{n+2}^b - \alpha_0^b \alpha_{n+2}^a & \text{if } \alpha_0^a < 0 \text{ and } \alpha_0^b < 0 \end{cases}.$$

In the sequel, we discuss three different techniques of linearization: the *minimax* approximation, the *min-range* approximation and a Taylor-series based technique, called *Taylor1+*.

Approximation Techniques

The standard affine arithmetic is non-closed under non-linear operations. Therefore, we seek an optimal (in a sense to define) affine form which approximates the non-linear result. The issue here is different from the problem of finding the narrower interval that encloses a given non-linear explicit real function defined over $\mathbb{R}^n$. Although we can use the techniques from the latter rich field and adapt them to our purpose as very often, linearizing before computing an enclosing interval gives better results. Methods such as the interval slope arithmetic [ZW90, Kol97], or interval derivative arithmetic [Kag86] together with the classical Taylor expansion are known in the literature of reliable computing to give tight enclosure. These methods have been already applied to affine arithmetic to improve the range of a given function [MK04a, MK04b]. We do not expand here these methods as our purpose is different from finding an interval enclosing a real valued function. We later give (Chapter 5) an example where the use of these techniques may lead to unsound results in the abstract interpretation framework.

The linear approximation of a function f over an interval $\boldsymbol{i}$ is the function $f^l(x) \stackrel{\text{def}}{=} \zeta + \alpha x + e(x)$, where $e(x)$ is the linearization error term. The error term is a non-linear function of x. As we are targeting a linear approximant, we consider in the sequel that it is independent of x and replace the function $e(x)$ by its image, which is an interval, denoted by $\boldsymbol{e}$.

The linearization can be optimized such that the error interval $\boldsymbol{e}$ is minimal or such that the width of the image of f is minimal. The former is called the minimax approximation, also known as Chebyshev [3] approximation.

[3] or Tchebycheff or Tshebyshev.

The latter is called the min-range approximation (as defined in [dFS97]). In general, there is no unique linear optimum form that achieves at the same time the minimax and min-range approximations (see the reciprocal function example below). In general, minimizing the error or the range of the linearized form is closely related to the underlying problem we are addressing.

The Taylor approximation of first order is in general not optimal with respect to both directions given above. However, by construction, it catches the main linear component of the function, which makes it convenient for the evaluation of the future transformations of the function. Moreover, its computation is straightforward and can be easily automated. We exemplify these techniques through the reciprocal function.

Let f denote the reciprocal function:

$$\begin{aligned} f\colon [a,b] &\to \mathbb{R} \\ x &\mapsto \frac{1}{x} \end{aligned}$$

where a and b are two real numbers such that $a > 0$. The linear approximation of f is f^l where

$$f^l(x) = \zeta + \alpha x + [-\beta, \beta] = \zeta + \alpha x + \beta\epsilon_f,$$

where the noise symbol ϵ_f is a fresh noise symbol used to encode the symmetric interval $[-1, 1]$. It suffices now to substitute x by its related affine form to obtain an affine form that approximates the function $f(x)$.

Min-range approximation To compute the min-range approximation, we need to compute ζ, α and β such that the interval

$$\zeta + \alpha[a,b] + [-\beta, \beta]$$

is equal to $[\frac{1}{b}, \frac{1}{a}]$. Since, $a > 0$, then $f(x)$ is a decreasing function. Therefore $\alpha \leq 0$. Thus

$$\zeta + \alpha[a,b] + [-\beta, \beta] = [\zeta + \alpha b - \beta, \zeta + \alpha a + \beta] = [\frac{1}{b}, \frac{1}{a}] \ .$$

The case $\alpha = 0$ is equivalent to IA: $f(x)$ is over-approximated by the box $[\frac{1}{b}, \frac{1}{a}]$. Observe that here, we have infinitely many linear functions that achieve the min-range approximation. All of them verify:

$$\begin{aligned} \zeta + \alpha b - \beta &= \frac{1}{b} \\ \zeta + \alpha a + \beta &= \frac{1}{a}, \end{aligned}$$

we pick up the one that tightly over-approximates $f(x)$, that is such as α is equal to the derivative of $f(x)$ on b (see Figure 3.2), which gives

$$\begin{aligned}\alpha &= \frac{-1}{b^2}\\ \zeta &= \frac{a}{2b^2} + \frac{1}{2a} + \frac{1}{b}\\ \beta &= \frac{a}{2b^2} + \frac{1}{2a} - \frac{1}{b}\end{aligned}$$

Taylor approximation The Taylor approximation of first order of a non-linear continuously differentiable function defined over a bounded interval is defined by: (a) its first order Taylor series computed in a point of the domain of the function (this gives the linear part), plus (b) an interval that over-approximates the difference of the function itself and its Taylor development (this gives the error part). That is, if f is defined over the bounded interval $[a, b]$, and c is an element of $[a, b]$, then

$$f^l(x) = f(c) + f'(c)x + \boldsymbol{e}, \text{ where } \forall x \in [a,b], f(x) - (f(c) + f'(c)x) \in \boldsymbol{e} \ .$$

The linear function obtained is a centered form (centered in c). It is sound, that is $f([a, b]) \subseteq f^l([a, b])$. Moreover, such centered form has a quadratic order of approximation [Han69, CM72], that is

$$w(f([a,b])) - w(f^l([a,b])) = O((b-a)^2),$$

where the function $w([a, b]) \stackrel{\text{def}}{=} b - a$ denotes the width of the interval $[a, b]$.

In general, the centered form obtained if c is the midpoint of the interval $[a, b]$ does not lead to a minimal interval for f^l. In fact, the center c that leads to the optimum upper bound may be different from the one leading to the optimum lower bound and both different from the midpoint [Bau88]. However, the choice of the midpoint eases the computations and gives a good heuristic.

For the reciprocal function f, we obtain:

$$\begin{aligned}\alpha &= \frac{-4}{(a+b)^2}\\ \beta &= -\frac{2b}{(a+b)^2} + \frac{1}{2a} = \frac{1}{2a} - \frac{1}{\frac{a^2}{2b} + \frac{b}{2} + a}\\ \zeta &= \frac{4}{a+b} + \beta = \frac{1}{2a} + \frac{2}{a+b} + \frac{1}{\frac{b^2}{2a} + \frac{a}{2} + b}\end{aligned}$$

This approach is more precise than just considering the first order Taylor approximation of the reciprocal function, and then over-approximating the error term using Cauchy's estimate [4] We call it *Taylor1+ approximation* as it is more precise than the first order approximation and less precise than the second order approximation.

Minimax approximation The computation of the minimax approximation is ruled by the *Chebyshev Alternation Theorem* [5].

3.3.11 Theorem
Chebyshev Alternation Theorem [1854] A polynomial p of degree n is the best approximant to $f \in C[a,b]$ (the set of continuous functions defined over the interval $[a,b]$) if and only if there exist $(n+2)$ points $a \leq t_1 \leq \ldots \leq t_{n+2} \leq b$ such that

$$f(t_i) - p(t_i) = (-1)^i \delta, \quad |\delta| = \sup_{x \in [a,b]} |f(x) - p(x)|,$$

i.e., if and only if the difference $f(x) - p(x)$ takes consecutively its maximal value with alternating signs at least $(n+2)$ times.

Theorem 3.3.11 is a sufficient condition for a given polynomial to be optimum with respect to minimizing the maximum error introduced by the linearization. There exists an iterative algorithm, called Remez (or Remes) algorithm [Rem34] that computes an optimum polynomial starting from an initial set of points (usually Chebyshev nodes, that is $x_i = \cos(\frac{(2i-1)\pi}{2n})$, $i = 1 \ldots n+2$ for a set of points in $[-1,1]$, linearly transformed if needed into any interval $[a,b]$). For instance, we can use the efficient implementation [Boo] of the Remez algorithm to find first degree polynomials (seen as functions of the Chebyshev space) that approximate f. This approximation is the best with respect to the uniform norm.

In our simple case, we can establish geometrically the polynomial of degree 1 defined by $\zeta + \alpha x$, that interpolates the two points $(a, \frac{1}{a})$ and

[4] Here, the Cauchy's estimate, or uniform estimate, is $M_1 \frac{(b-a)^2}{8}$, where M_1 dominates the absolute value of the second derivative of the reciprocal function, that is $|\frac{1}{x}^{(2)}|$.

[5] detailed proof and further details on the use of Chebyshev approximation theory can be found for instance in [E.W66, §6].

$(b, \frac{1}{b})$, and which wraps closely the graph of the reciprocal function. Thus:

$$\alpha = \frac{\frac{1}{a} - \frac{1}{b}}{a - b}$$
$$\zeta = \frac{1}{2}(\frac{1}{a} - \alpha a - 2\sqrt{-\alpha})$$

We then compute the point x where the tangent has the same direction of $\zeta + \alpha x$, we obtain $\sqrt{ab}$. We can verify that the hypothesis of Theorem 3.3.11 holds for $x = a$, $x = \sqrt{ab}$, and $x = b$, and that

$$\beta = |\delta| = 2\sqrt{-\alpha} + \zeta$$

3.3.12 Example (Numerical example)
Let $x \in [1, 4]$, and $\hat{x} = 2.5 + 1.5\epsilon_0$; then, for min-range approximation we have $\alpha = -0.0625$, $\beta = 0.28125$, $\zeta = 0.78125$, and

$$\begin{aligned} f^l_{min\ range} &= 0.78125 - 0.0625(2.5 + 1.5\epsilon_0) + 0.28125\epsilon_f \\ &= 0.625 - 0.09375\epsilon_0 + 0.28125\epsilon_f \\ &\in [0.25, 1] \end{aligned}$$

For Taylor approximation, we have $\alpha = -0.16$, $\zeta = 0.98$, $\beta = 0.18$, and

$$\begin{aligned} f^l_{Taylor} &= 0.98 - 0.16(2.5 + 1.5\epsilon_0) + 0.18\epsilon_f \\ &= 0.58 - 0.24\epsilon_0 + 0.18\epsilon_f \\ &\in [0.16, 1] \end{aligned}$$

For Chebyshev approximation, we have $\alpha = -0.25$, $\zeta = 1.125$, $\beta = 0.125$, and

$$\begin{aligned} f^l_{minimax} &= 1.125 - 0.25(2.5 + 1.5\epsilon_0) + 0.125\epsilon_f \\ &= 0.5 - 0.375\epsilon_0 + 0.125\epsilon_f \\ &\in [0, 1] \end{aligned}$$

Figure 3.2 depicts these approximations. Observe that the joint range of $(\hat{x}, f^l_{\text{minimax}})$ wraps closely the graph $(x, f(x))$. However, the interval of f^l_{minimax}, that is $[0, 1]$ is not optimal. The optimal interval, $[0.25, 1]$, is given by $f^l_{\text{min-range}}$, but the joint range of $(\hat{x}, f^l_{\text{min-range}})$ wraps loosely the graph $(x, f(x))$.

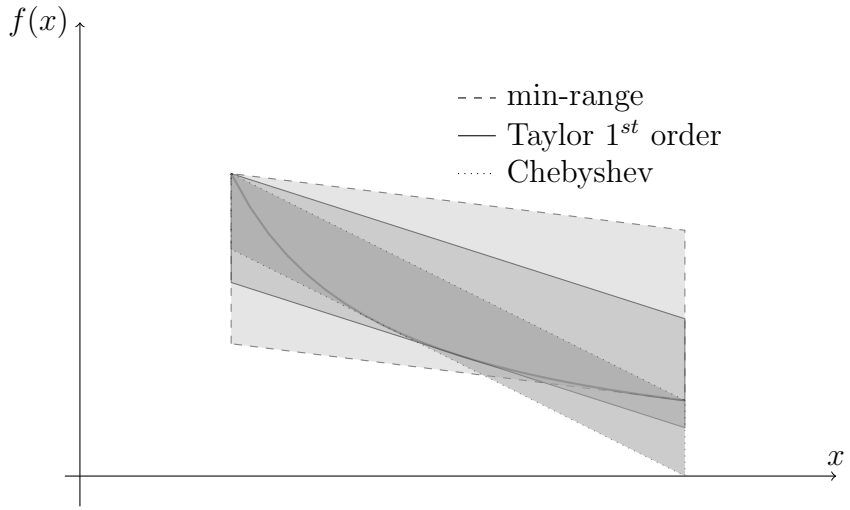

Figure 3.2: Affine form approximations of the reciprocal function.

Perturbed Affine Sets

The novelty of this domain introduced in [GP06], resides in the way it uses to encode relations between variables. While all prior attempts such as Polyhedra-like or Karr's affine equalities abstract domains, keep and propagate explicit relations between variables, the affine forms-based abstract domain keeps implicit relations between variables encoded by the shared noise symbols (see Section 3.3).

The geometrical concretisations of such domain are zonotopes, which are central symmetric polytopes. Thus, the domain is strictly more expressive than weakly relation domains such as octagons, it encodes perfectly linear equalities between variables, but is strictly less expressive than Polyhedra domain in general because of the symmetry of zonotopes. Thus, in term of expressiveness, it fills the gap between weakly relational domains and fully linear relational domain as the linear templates polyhedra do. However, the affine sets domain is definitely more precise and more efficient than all other domains whenever the program to analyze uses non-linear operations.

Zonotopes have an efficient memory representation (set of vectors). The complexity of arithmetic computations is almost linear with respect to the number of noise symbols used. Non linear operations can be precisely and efficiently linearized as detailed in Section 3.3.

For the purpose of abstract interpretation, one needs to define two set theoretic operations, the join and the meet, over affine forms. Since the convex hull of two zonotopes is not a zonotope, and so is the intersection

of two zonotopes, we have to compute a zonotope that encloses two given zonotopes and the zonotope that encloses the intersection of two zonotopes. For this purpose, Goubault and Putot [GP08, GP09] have extended affine forms.

We briefly describe in the sequel the affine forms-based abstract object, arithmetic operations over such abstract domain and finally set theoretic operations.

A set of numerical variables $\mathcal{V} = \{v_1, \ldots, v_p\}$ is abstract as follows:

3.3.13 Definition
A perturbed affine set $\hat{X}$ is defined by

$$\hat{X} \stackrel{\text{def}}{=} (C^X, P^X),$$

where C^X is matrix with p lines and $n+1$ columns and P^X is a matrix with p lines and m columns. Elements of C^X and P^X are real numbers.

Each variable v_l in $\mathcal{V}$ is abstract by an affine form:

$$\hat{X}_l \stackrel{\text{def}}{=} \underbrace{\sum_{i=0}^{n} C^X_{l,i}\epsilon_i}_{\text{central part}} + \underbrace{\sum_{j=1}^{m} P^X_{l,j}\eta^X_j}_{\textit{deviation}},$$

All noise symbols are unknown but within $[-1, 1]$ except ϵ_0 which is equal to 1. The coefficients of the affine form are the elements of the lth line of C^X and P^X. The first element of the lth line of C^X encodes the constant of the affine form. Indeed the noise symbol ϵ_0 is equal to 1. The sub affine form composed by the coefficients of the lth line of C^X is called the central part of the abstraction of the variable v_i. The sub affine form composed by the coefficient of the lth line of P^X is called the perturbation part.

The concretisation function γ is defined as follows:

3.3.14 Definition
Given a perturbed affine set $\hat{X}$, its geometrical concretisation is the set defined by

$$\gamma(\hat{X}) \stackrel{\text{def}}{=} \{C^X\epsilon + P^X\eta^X \mid (\epsilon, \eta^X) \in \{1\} \times [-1, 1]^{n+m}\},$$

The geometrical concretisation is exactly the joint range of all numerical variables. It can be seen as the Minkowski sum of two zonotopes, namely the one having as generators the columns of C^X and the one having as

generators the columns of P^X. We recall that the Minkowski sum of two sets A and B is the set defined by

$$A + B = \{a + b \mid a \in A, b \in B\} \ .$$

The noise symbols $(\epsilon_1, \ldots, \epsilon_n)$ have a strict meaning, they encode the non-deterministic input variables. They are strongly linked to these variables. If one substitutes these noise symbols by any other dummy symbol between $[-1, 1]$, the local joint range (zonotope) remains unchanged but the relations to those input variables get lost. Moreover, the final invariant (which is encoded as a function of these central noise symbols) gives the input/output relations of the program (functional analysis).

On the other hand, the perturbation noise symbols $(\eta_1^X, \ldots, \eta_m^X)$ are indexed by the X to enforce the fact that these noise symbols are not shared between all abstract objects. They are used to define an order over perturbed affine sets.

3.3.15 Definition (Order over Perturbed Affine Sets)
Given two Perturbed Affine Sets $\hat{X}$ and $\hat{Y}$, we say that $\hat{X}$ is lesser than or equal to $\hat{Y}$, if and only if

$$\forall t \in \mathbb{R}^p, \quad \|(C^X - C^Y)t\|_1 \leq \|P^Y t\|_1 - \|P^X t\|_1 \ .$$

The order defined above is stronger than the geometrical order. The geometrical order, $\gamma(X) \subseteq \gamma(Y)$, does not respect the semantics of the central noise symbols. Indeed, computing a zonotope that encloses the zonotopes $\gamma(X)$ and $\gamma(Y)$, without any other consideration, loses definitely the relations to central noise symbols. These relations that we need precisely to keep.

As established in [GP09], the order over Perturbed Affine Sets implies the geometrical order.

3.3.16 Proposition
$\hat{X} \leq \hat{Y} \implies \gamma(X) \subseteq \gamma(Y)$.

So far this affine forms based abstract domain ignore tests, our work is an attempt to address this limitation.

Arithmetic operations over perturbed affine sets rely on affine arithmetic (see Section 3.3). The definitions are exposed in detail in Chapter 5.

3.4 Combining Abstract Domains

The direct product of two or more abstract domains is equivalent to performing the analysis separately with each abstract domain. This approach does not combine the expressiveness power of the abstract domains in use, and hence does not improve the final result.

We briefly recall hereafter two different generic approaches designed to improve the precision of the analysis by sharing the information found by one domain in order to improve the result given by the other domain. The exchange of information is done dynamically during the analysis. Theoretically, these interleaves are strictly more expressive than each domain taken alone (as in direct product), nevertheless, in practice these approaches are either limited theoretically or need an extra effort to handle the exchange of information.

Reduced Product

If $D^\flat$ is a concrete domain abstracted by two abstract domains $D_1^\sharp$ and $D_2^\sharp$, then the concretisation function $\gamma_{1\times 2} : (D_1^\sharp, D_2^\sharp) \to D^\flat$, of the reduced product $(D_1^\sharp, D_2^\sharp)$ is defined by the meet in $D^\flat$ of the concretisations of $D_1^\sharp$ and $D_2^\sharp$:

$$\gamma_{1\times 2}(X_1^\sharp, X_2^\sharp) \stackrel{\text{def}}{=} \gamma_1(X_1^\sharp) \cap^\flat \gamma_2(X_2^\sharp) \ .$$

To propagate the conjunction of properties in the concrete domain given by $\gamma_{1\times 2}$, we need to abstract again the object $\gamma_{1\times 2}(X_1^\sharp, X_2^\sharp)$, using the abstraction functions α_1 and α_2 of the abstract domains $D_1^\sharp$ and $D_2^\sharp$ respectively :

$$\alpha_{1\times 2}(X^\flat) = (\alpha_1(X^\flat), \alpha_2(X^\flat)) \ .$$

The way used to share the information between both abstract domains relies on the concretisation of the abstract objects. It is formalized using a so called *reduction operator* $\rho : (D_1^\sharp, D_2^\sharp) \to (D_1^\sharp, D_2^\sharp)$, defined by the combination of $\gamma_{1\times 2}$ and $\alpha_{1\times 2}$:

$$\rho = \alpha_{1\times 2} \circ \gamma_{1\times 2} \ .$$

The reduction operator relies then on the abstraction functions of the abstract domains $D_1^\sharp$ and $D_2^\sharp$. These functions are in practice seldom available which prevents the immediate use of the generic previous definition of ρ. Instead, we may define local partial reduction specific to the transfer function: for instance, the evaluation of a non-linear expression relies very often on the ranges of variables. We could use the intervals lattice to record

such information, then use those ranges for a better linearization of the expression.

Logical Product

The logical product of abstract domains [GT06] combines the abstract domains in a way strictly more expressive than what we have seen in reduced product. The approach is based on the classic Nelson-Oppen methodology [NO79] for combining decision procedure.

Unlike the reduced product which requires in practice to define partial reductions specific to the abstract domains and even to transfer functions, the logical product can be build upon the native operators of the underlying abstract domains. Nevertheless, for an efficient combination (in a polynomial time), hard restrictions have to be verified by the *theories* upon which we build the underlying abstract domains. Namely these theories have to be *convex*, *stably infinite* and *disjoint*.

A theory consists of a *signature*, which is a set of symbols (predicates and functions), and a set of *axioms* which defines the semantics of the signature of that theory. An *atomic fact* of a theory is the simplest possible predicate over that theory, that is, one can no more decompose it into a conjunction of predicates of the same theory. A *logical lattice* is derived from a theory if and only if, its objects are all finite conjunctions of atomic facts and its partial order is the *logic implication* relationship in that theory. For instance, the signature of the theory of sign is defined by $\{=, positive, negative, +, -, 0, 1\}$, where *positive* and *negative* are unary predicates, $+$ and $-$ are binary functions, and 0 and 1 are constants; the signature of the linear arithmetic theory is $\{=, \leq, +, -, 0, 1\}$. The set of axioms of the linear arithmetic theory includes all the known rules such as $x \leq y \wedge y = z \implies x \leq z$.

Any abstract domain can be seen as a logical lattice. For instance, the polyhedra abstract domain is the logical lattice based upon the theory of linear arithmetic, whereas the affine sets abstract domain is the logical lattice based on the theory of linear arithmetic with only the equality symbol.

A theory is said to be convex if any conjunction of equalities implies necessarily that one equality holds. A theory is stably infinite if any quantified free property satisfied in that theory is also satisfied in any infinite model of that theory. Two theories are disjoint if and only if their signatures are disjoint except for the equality symbol.

Combining two theories with respect to Nelson-Oppen method requires the definition of three generic procedures over both theories. Using the terminology of [GT06], these procedures are as follows:

- a procedure that recognizes the terms that combine atomic facts from both theories, often called *alien terms* as they are not pure terms of one theory,
- a procedure that purifies a given alien terms: this is usually done by adding new variables,
- and finally a saturation procedure that takes two conjunctions of pure terms of both theories, then keeps exchanging all equalities between these conjunctions until no more new equality can be derived.

The saturation procedure is actually the one that permits the exchange of information between both theories. Since an expression can contain alien terms, the two other procedures are used to extract and purify these alien terms. Now that we have saturated pure terms, we can use the native abstract transfer operators relative to the underlying abstract domain.

The restrictions on theories, that is the convexity, the stable infiniteness and the disjointness, limit in practice the use of the generic approach. Nevertheless, one can always define a logical product of two logical lattices, of course without using the straightforward use of Nelson-Oppen methodology.

In this thesis, we introduce and formalize a logical product of the perturbed affine sets domain and any other (convex) abstract domain. We keep the terminology "logical product" even if the underlying theories are not disjoint.

CHAPTER 4

Constrained Affine Sets

We introduce a new affine sets-based abstract domain which extends and generalizes an already existing affine sets-based abstract domain [GP06, GP08, GP09], $(\mathcal{A}_1, \bot_1, \top_1, \leq_1, \cup_1)$. The expressiveness of such new domain is enhanced thanks to its ability to encode and propagate relations among the noise symbols. These relations, or constraints, over noise symbols encode the domain where the symbols range. The domain is abstracted using an abstract domain $(\mathcal{A}_2, \bot_2, \top_2, \leq_2, \cup_2, \cap_2)$.

We define a special logical product-like combination of the abstract domains $\mathcal{A}_1$ and $\mathcal{A}_2$, denoted by $\mathcal{A}_{1\times 2}$. The variables abstracted in $\mathcal{A}_1$ are the numerical variables of the program to analyze, whereas the variables abstracted in $\mathcal{A}_2$ are the noise symbols used to keep implicit linear relations between the program variables. Thus, we do not use two different abstract domains to abstract the same set of variables, as in reduced product of abstract domains [CC79] or logical product of abstract domains [GT06]. Moreover, the information shared between $\mathcal{A}_1$ and $\mathcal{A}_2$ is as expressive as the information shared in the logical product of abstract domains, for instance the equality constraints are propagated between the two domains in an intricate manner (this is not just a reduction operation).

4.1 Introduction

Despite their ability of keeping (linear and implicit) relations between variables, and their simple memory representation, the affine forms are not closed under set theoretic operations: the intersection and the join. Affine sets are not closed under set theoretic operations. Indeed, the join and the intersection of two zonotopes is not a zonotope in general.

In this chapter, we introduce informally the way we interpret the meet operations on zonotopes that arise from the if-then-else statements. Where the previously defined Perturbation Affine Sets abstract domain proposes a join operation, it ignores tests (which is sound). We come up with an elegant way to express and propagate these tests, then extend the join operators of [GP08, GP09] to our newly defined domain.

Consider the simple code below extracted from an implementation of quadratic interpolation function.

```
begin
x = [-1,1]; ❶
 if (x <= 0)❷ then
  y = x*x + x; ❸
 endif ❻
end
```

At control point ❶, we first convert the interval $[-1, 1]$ into an affine form, denoted $\hat{a}$, by adding a new noise symbol $\epsilon_{❶}$ known to be within $[-1, 1]$:

$$x \in [-1, 1] \overset{\text{becomes}}{\longrightarrow} \hat{a} = \epsilon_{❶}, \quad \epsilon_{❶} \in [-1, 1],$$

If one ignores the test (`x <= 0`), at control point ❸ we obtain for y the affine form $0.5 + \epsilon_{❶} + 0.5\epsilon_{❷}$, where $\epsilon_{❷}$ is a new noise symbol introduced to linearize the non-linear expression $x^2 + x$ (see Chapter 5 for the abstract evaluation of expressions). Thus we conclude that, in the `if` branch, the variable `y` is within the interval $[-1, 2]$. In this example, the reduced product with boxes, to interpret the test `x <= 0`, improves slightly this result. Indeed, the interval analysis gives for `y`, $[-1, 0] \times [-1, 0] + [-1, 0]$, and hence the final interval is $[-1, 1]$.

However, if we use the information $\epsilon_{❶} \leq 0$, implied by the test, the linearization of x^2 using centered forms (discussed and formalized in Section 5.1), gives $-0.125 - \epsilon_{❶} + 0.125\epsilon_{❷}$, and thus the affine form related to y is $-0.125 + 0.125\epsilon_{❷}$. This gives the box $[-0.25, 0]$, which is exactly the image of the interval $[-1, 0]$ through the non-linear function $x^2 + x$.

Our main idea is to transfer the constraint from the variables' world to the noise symbols' world by substituting each variable by its corresponding affine form. Such a constraint is then kept and used in all incoming non-linear computations including the join of two affine sets. For instance, in the above example, the test `x <= 0`, where `x` is abstracted by $\hat{a}$, is interpreted by the constraint $\epsilon_{❶} \leq 0$. Observe that the affine form $\hat{a}$ is left unchanged, which permits the normal use of affine arithmetic.

The abstract object of interest in our domain is then a conjunction of

constraints, either linear or non-linear, expressed in the abstract domain of the noise symbols and a set of affine forms, one per variable. Typically, in our example, the abstract value at control point ❸ is

$$\begin{cases} -1 \leq \epsilon_{❶} \leq 0 \wedge -1 \leq \epsilon_{❷} \leq 1 \\ \hat{a} = \epsilon_{❶} \\ \hat{b} = -0.125 - \epsilon_{❶} + 0.125\epsilon_{❷} \end{cases}$$

where $\hat{b}$ is the affine form that abstracts the variable y.

In the next section, we formalize these abstract objects.

4.2 Constrained Affine Sets

Let $\mathcal{V} \stackrel{\text{def}}{=} \{v_1, \ldots, v_p\}$ denotes the finite set of numerical variables of the program to analyze.

Representation

4.2.1 Definition

A constrained affine set $\hat{X}$ is represented by a tuple

$$\hat{X} \stackrel{\text{def}}{=} \left(C^X, P^X, \Phi^X\right),$$

where C^X is a real matrix with p lines and $n+1$ columns, P^X is a real matrix with p lines and m columns, and Φ^X is an abstract element of $\mathcal{A}_2$; n and m are finite integers. The set of constrained affine sets is denoted by $\mathcal{A}_{1\times 2}$.

The dimension $p < +\infty$ is the cardinality of $\mathcal{V}$, the set of numerical variables. The object $\hat{X}$ represents the abstraction of these numerical variables at a control point of the program. Each numerical variable, is abstracted by the affine form given by the lth line ($1 \leq l \leq p$) of C^X and P^X, as follows:

$$\hat{x}_l \stackrel{\text{def}}{=} \underbrace{\sum_{i=0}^{n} C^X_{l,i}\epsilon_i}_{\text{central part}} + \underbrace{\sum_{j=1}^{m} P^X_{l,j}\eta^X_j}_{\textit{deviation}},$$

where $C^X_{l,i}$, $1 \leq l \leq p$, $0 \leq i \leq n$, denotes the (l,i) coefficient of matrix C^X. Notice that we use zero for the first index of the columns of matrix C^X. Likewise, $P^X_{l,i}$, $1 \leq l \leq p$, $1 \leq i \leq m$, denotes the (l,i) coefficient of matrix P^X.

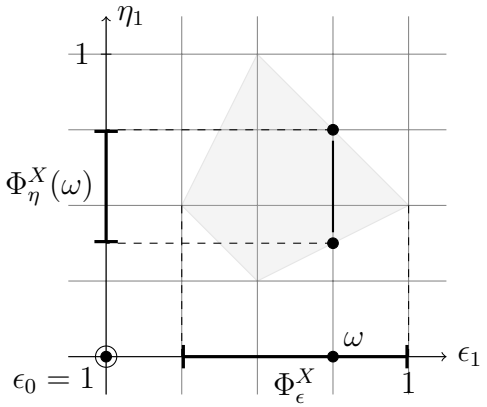

Figure 4.1: The sets Φ^X_ϵ and $\Phi^X_\eta(\omega)$, where the polygon represents the concretisation of Φ^X ($n = 1$,$m = 1$).

The symbol ϵ_0, equal to one, encodes constants [1]. The vector of central noise symbols is denoted by $\epsilon \stackrel{\text{def}}{=} (\epsilon_0, \epsilon_1, \ldots, \epsilon_n)^*$, these symbols have a strict semantics, they are related to the inputs of the program. One cannot substitute them by other symbols because they are not free variables. They encode the non-determinism of the inputs. The vector of deviation noise symbols $(\eta^X_1, \ldots, \eta^X_m)^*$ is denoted by η^X. We add explicitly the upper index ^{X} on η to stress the fact that these noise symbols are local noise symbols related to the abstract object $\hat{X}$ and are not shared with other abstract objects. They do not have any particular meaning as ϵ_i, $0 \leq i \leq n$, do. They are dummy symbols (free variables) used to encode the generators of the deviation part.

The vector ϵ augmented by the vector η^X ranges over the concretisation of Φ^X, that is $\gamma_2(\Phi^X)$, subset of $1 \times \mathbb{R}^{n+m}$, where γ_2 denotes the concretisation function of the abstract domain $\mathcal{A}_2$. We denote by $\Phi^X_\epsilon \subseteq 1 \times \mathbb{R}^n$ the projection of $\gamma_2(\Phi^X)$ over the $n+1$ first coordinates of $1 \times \mathbb{R}^{n+m}$.

For a fixed $\omega \in \Phi^X_\epsilon$, $\Phi^X_\eta(\omega) \subseteq \mathbb{R}^m$, denotes the section obtained by the intersection of the hyperplane $\epsilon = \omega$ and $\gamma_2(\Phi^X)$:

$$\Phi^X_\eta(\omega) \stackrel{\text{def}}{=} \{\eta^X \mid (\omega, \eta^X) \in \gamma_2(\Phi^X)\} \ .$$

Figure 4.1 depicts Φ^X_ϵ and a section $\Phi^X_\eta(\omega)$ for an arbitrary $\omega \in \Phi^X_\epsilon$.

[1] equivalent to the variable v_0 in the octagons abstract domain [Min06a]

Concretisation Function

4.2.2 Definition

The concretisation function of the abstract domain $\mathcal{A}_{1\times 2}$ is defined by,

$$\gamma_{1\times 2}(\hat{X}) \stackrel{\text{def}}{=} \{C^X\epsilon + P^X\eta^X \mid (\epsilon, \eta^X) \in \gamma_2(\Phi^X)\} \ .$$

In general, as in the example 4.2.3, the concretisation of a constrained affine set is not a zonotope (which is the case of any perturbed affine set).

4.2.3 Example

Suppose that $\mathcal{V} = \{v_1, v_2\}$. Let $\mathcal{A}_2$ be the polyhedra abstract domain, and let $p = n = m = 2$. The figure 4.2 depicts the concretisation of $\hat{X}$, where $\hat{X}$ is defined as follows

$$\hat{X} \stackrel{\text{def}}{=} \Big(\underbrace{\begin{pmatrix} 1 & 1 \\ -1 & 2 \end{pmatrix}}_{C^X}, \underbrace{\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}}_{P^X}, \Phi^X\Big),$$

and Φ^X is the conjunction of the following constraints

$$\epsilon_0 = 1 \wedge -1 + \frac{4}{3}\epsilon_1 + \frac{4}{3}\eta_1^X \geq 0 \wedge 2\epsilon_1 - 1\eta_1^X \geq 0 \wedge \frac{3}{2} - \epsilon_1 - \eta_1^X \geq 0$$
$$\wedge -\epsilon_1 + 2\eta_1^X \geq 0 \wedge \frac{1}{2} + \eta_2^X \geq 0 \wedge \frac{1}{2} - \eta_2^X \geq 0 \ .$$

The polytope given by the conjunction of the first four constraints is the polytope depicted in Figure 4.1 (which has four facets), the conjunction of the last two constraints involves only η_2^X and is simply the interval $[-\frac{1}{2}, \frac{1}{2}]$ for η_2^X.

Intervals Conversions

From a constrained affine set to a box The interval concretisation of the lth numerical variable of $\hat{X}$, $1 \leq l \leq p$, is the projection of $\gamma_{1\times 2}(\hat{X})$ on its lth dimension. The final box is the product of all these projections.

From a box to a constrained affine set Usually, intervals are useful at the beginning of the analysis, non-deterministic inputs of the program (for instance environment related variables) are often unknown but within intervals. The conversion given here is related to the input variables and thus updates only the central part C^X because of the particular semantics

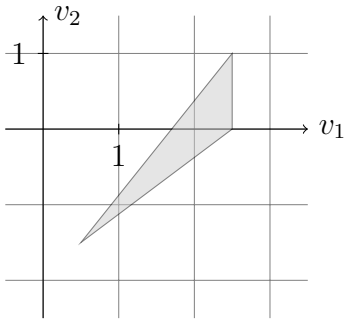

Figure 4.2: The concretisation of the constrained affine set given in Example 4.2.3.

of noise symbols, the perturbation matrix P^X is set to zero. After the conversion we have as many noise symbols as non-deterministic numerical variables.

Consider a bounded interval $[d_1, d_2]$, $d_1 \leq d_2 < +\infty$, then its related affine form is

$$\mathrm{mid}([d_1, d_2]) + \mathrm{dev}([d_1, d_2])\epsilon_f,$$

where ϵ_f is a fresh input noise symbol known to be within $[-1, 1]$. These constraints on ϵ_f are added to the abstract object Φ^X by computing the image of Φ^X through $[\![-\epsilon_l \leq 1 \wedge \epsilon_l \leq 1]\!]_2^\sharp$, that is the abstraction of the test transfer function in the abstract domain $\mathcal{A}_2$.

The affine form related to an unbounded interval is simply ϵ_f, without any constraint on ϵ_f. Indeed, we identify the variable to a fresh input noise symbol ϵ_f initialized to $\top_2$, and keep track of the finite bound of the interval as a constraint over the noise symbol freshly added. Therefore, for a left-bounded interval $[a, +\infty]$, $a < +\infty$, we compute the image of Φ^X through $[\![-\epsilon_f \leq -a]\!]_2^\sharp$. Similar transformation is done if the interval is right-bounded. Obviously, for the interval $[-\infty, +\infty]$, the freshly added noise symbol is unconstrained and remains equal to its initial value, that is $\top_2$.

We summarize the five possible cases in the following example.

4.2.4 Example
Let $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$ be the set of numerical variables known to be respectively within

$$[-\infty, +\infty] \times [a, +\infty] \times [-\infty, b] \times [d_1, d_2] \times [c, c] \ .$$

The constrained affine set related is then:

$$C^X = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \text{mid}([d_1, d_2]) & 0 & 0 & 0 & \text{dev}([d_1, d_2]) & 0 \\ c & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$P^X = 0$$

$$\Phi^X = [\![\epsilon_0 = 1 \wedge -\epsilon_2 \leq -a \wedge \epsilon_3 \leq b \wedge -1 \leq \epsilon_4 \leq 1 \wedge -1 \leq \epsilon_5 \leq 1]\!]_2^\sharp \top_2 \ .$$

Order over Constrained Affine Sets

We provide the constrained affine sets with an order relation. Whenever it is satisfied, the order relation should preserve all the information kept in one constrained affine set, so that using the "bigger" one guarantees the safety of all future computations.

4.2.5 Definition

Let $\hat{X} = (C^X, P^X, \Phi^X)$ and $\hat{Y} = (C^Y, P^Y, \Phi^Y)$ be two constrained affine sets. We say that $\hat{X}$ is less than or equal to $\hat{Y}$, denoted by $\hat{X} \leq_{1\times 2} \hat{Y}$, if and only if $\Phi^X_\epsilon \subseteq \Phi^Y_\epsilon$, and

$$\forall \omega \in \Phi^X_\epsilon, \forall \zeta^X \in \Phi^X_\eta(\omega), \exists \zeta^Y \in \Phi^Y_\eta(\omega) \quad C^X\omega + P^X\zeta^X = C^Y\omega + P^Y\zeta^Y.$$

What the definition says is that for every possible input, encoded by ω, we are able to recover each reached value in $\hat{X}$, that is $C^X\omega + P^X\zeta^X$, using the same input ω and a different possible perturbation ζ^Y. Observe that the order requires only the inclusion of the input noise symbols $\Phi^X_\epsilon \subseteq \Phi^Y_\epsilon$. This restriction denotes the fact that the input noise symbols are shared between abstract objects (which is not the case of perturbation noise symbols). The order respects this semantics and ensures that the set of values taken by these symbols in $\hat{Y}$, that is Φ^Y_ϵ, contains the one of $\hat{X}$, that is Φ^X_ϵ. The binary relation order $\leq_{1\times 2}$ is a pre-order over $\mathcal{A}_{1\times 2}$.

4.2.6 Proposition

The binary relation $\leq_{1\times 2}$ in Definition 4.2.5 is a pre-order. The equivalence relation $\sim$ ($\hat{X} \sim \hat{Y}$ if and only if $\hat{X} \leq_{1\times 2} \hat{Y}$ and $\hat{Y} \leq_{1\times 2} \hat{X}$) is characterized by $\Phi^X_\epsilon = \Phi^Y_\epsilon$ and $C^X\omega + P^X\Phi^X_\eta(\omega) = C^Y\omega + P^Y\Phi^Y_\eta(\omega)$ for all ω in Φ^X_ϵ (sets equality). For the sake of simplicity, we also denote by $\leq_{1\times 2}$ the pre-order $\leq_{1\times 2}$ quotiented by its equivalence relation $\sim$.

Proof. *Reflexivity.* $\hat{X} \leq_{1\times 2} \hat{X}$. *Indeed* $\Phi^X_\epsilon \subseteq \Phi^X_\epsilon$, *and, for all* $\omega \in \Phi^X_\epsilon$, *for all* $\zeta^X \in \Phi^X_\eta(\omega)$, *we have* $C^X\omega + P^X\zeta^X = C^X\omega + P^X\zeta^X$.

Transitivity. $\hat{X} \leq_{1\times 2} \hat{Y}$ *and* $\hat{Y} \leq_{1\times 2} \hat{Z}$ *imply* $\hat{X} \leq_{1\times 2} \hat{Z}$. *Indeed,* $\Phi^X_\epsilon \subseteq \Phi^Y_\epsilon$ *and* $\Phi^Y_\epsilon \subseteq \Phi^Z_\epsilon$ *ample* $\Phi^X_\epsilon \subseteq \Phi^Z_\epsilon$. *Moreover, for all* $\omega \in \Phi^X_\epsilon$, *for all* $\zeta^X \in \Phi^X_\eta(\omega)$, *there exists* $\zeta^Y \in \Phi^Y_\eta(\omega)$ *such that* $C^X\omega + P^X\zeta^X = C^Y\omega + P^Y\zeta^Y$. *For that* ζ^Y, *there exists,* $\zeta^Z \in \Phi^Z_\eta(\omega)$, *such that* $C^Y\omega + P^Y\zeta^Y = C^Z\omega + P^Z\zeta^Z$, *which makes* $C^X\omega + P^X\zeta^X = C^Z\omega + P^Z\zeta^Z$. ∎

The partial order $\leq_{1\times 2}$ is not equivalent to the geometrical order, since our ϵ noise symbols have a strict semantics that should be respected by the order. The geometrical order is however a necessary (but not sufficient) condition of this order.

4.2.7 Proposition

The concretisation function $\gamma_{1\times 2}$ *is a monotonic operator: given two constrained affine sets* $\hat{X} = (C^X, P^X, \Phi^X)$ *and* $\hat{Y} = (C^Y, P^Y, \Phi^Y)$, *we have*

$$\hat{X} \leq_{1\times 2} \hat{Y} \implies \gamma_{1\times 2}(\hat{X}) \leq \gamma_{1\times 2}(\hat{Y}) \ .$$

Proof. *Let* x *be an element of* $\gamma_{1\times 2}(\hat{X})$, *we prove that* $x \in \gamma_{1\times 2}(\hat{Y})$ *under the hypothesis* $\hat{X} \leq_{1\times 2} \hat{Y}$. *Since* $x \in \gamma_{1\times 2}(\hat{X})$, *then there exists* $\omega \in \Phi^X_\epsilon$ *and* $\zeta^X \in \Phi^X_\eta[\epsilon = \omega]$, *such that* $x = C^X\omega + P^X\zeta^X$. *Therefore, there exists* $\zeta^Y \in \Phi^Y_\eta[\epsilon = \omega]$, *such that* $x = C^X\omega + P^X\zeta^X = C^Y\omega + P^Y\zeta^Y \in \gamma_{1\times 2}(\hat{Y})$. ∎

For instance, if $\mathcal{A}_2$ is the lattice of intervals, then $\hat{X} \leq_{1\times 2} \hat{Y}$ if and only if $\Phi^X_\epsilon \subseteq \Phi^Y_\epsilon$ and $(C^X - C^Y)\Phi^X_\epsilon + P^X\Phi^X_\eta \subseteq P^Y\Phi^Y_\eta$.

The geometrical order does not imply in general the order $\leq_{1\times 2}$. Example 4.2.8 gives a counter example, where $\hat{X} \not\leq_{1\times 2} \hat{Y}$ and $\gamma_{1\times 2}(\hat{X}) \leq \gamma_{1\times 2}(\hat{Y})$.

4.2.8 Example

$$X = \left(\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} -0.25 & 1 \\ 0.25 & 0 \end{pmatrix}, 1 \times [-1,1]^3 \right)$$
$$Y = \left(\begin{pmatrix} 1 & -0.25 \\ 1 & 0.25 \end{pmatrix}, \begin{pmatrix} 0.5 & 0.5 \\ -0.5 & 0.5 \end{pmatrix}, 1 \times [-1,1]^3 \right)$$

Figure 4.3 depicts the zonotopes $\gamma_{1\times 2}(\hat{X})$ *and* $\gamma_{1\times 2}(\hat{Y})$, *one can see that the inclusion holds. Figure 4.4 depicts the zonotopes* $(C^X - C^Y)\Phi^X_\epsilon + P^X\Phi^X_\eta$

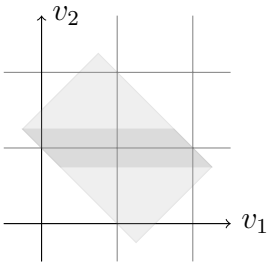

Figure 4.3: $\gamma_{1\times2}(\hat{X}) \subseteq \gamma_{1\times2}(\hat{Y})$.

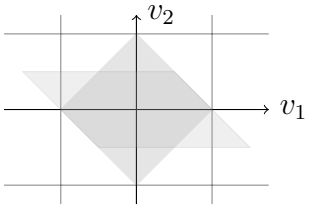

Figure 4.4: $(C^X - C^Y)\Phi_\epsilon^X + P^X\Phi_\eta^X \not\subseteq P^Y\Phi_\eta^Y$ (lozenge).

and $P^Y\Phi_\eta^Y$, the former is not included in the latter. That is, there exists $\omega \in 1 \times [-1,1]$ and $\zeta^X \in [-1,1]^2$, such that for all $\zeta^Y \in [-1,1]^2$, $C^X\omega + P^X\zeta^X \neq C^Y\omega + P^Y\zeta^Y$. Indeed, every pair (ω, ζ^X) that leads to a point outside the lozenge, for instance $\omega = (1,1)$ and $\zeta^X = (-1,1)$, violates the needed equality. Hence according to definition 4.2.5, $\hat{X} \not\leq_{1\times2} \hat{Y}$.

So far we have defined a poset $(\mathcal{A}_{1\times2}, \leq_{1\times2})$, elements of which are given in Definition 4.2.1. The next section focuses on a special case: the abstraction of noise symbols with intervals. This particular case is useful for the join operation over constrained affine sets.

4.3 Special Case: Non Relational Constraints

All over this section, $\mathcal{A}_2$ is the lattice of intervals. In Section 4.3 we reformulate the partial order (Definition 4.2.5) using the support function (recalled hereafter). Then, we give a special representative of the equivalence classes defined by the $\sim$ relation, called symmetric representative (Section 4.3). Section 4.3 discusses the decidability of $\leq_{1\times2}$. These definitions and reformulations are the basic ingredients needed to define and compute join operators over constrained affine sets (CAS).

Partial Order and Support Function

We first reformulate the partial order $\leq_{1\times2}$ over CAS (Definition 4.2.5) transforming the sets inclusion into a function comparison, namely comparison of the support function of convex sets (see hereafter Definition 4.3.1).

This classical convex function, offers a suitable and powerful tool for the upcoming computations.

4.3.1 Definition (The support function)
Let C be a non-empty convex set of $\mathbb{R}^n$. Let t be an element of $\mathbb{R}^n$ then $\delta : \mathbb{R}^n \to \mathbb{R}$ is defined by:

$$\delta(t \mid C) \stackrel{\text{def}}{=} \sup\{\langle t, x\rangle \mid x \in C\},$$

where $\langle \cdot, \cdot \rangle$ denotes the usual scalar product over $\mathbb{R}^n$.

The inclusion of two convex sets is equivalent to the comparison of the support functions related to these convex sets [Roc70, Corollary 13.1.1].

4.3.2 Proposition
For closed convex sets C_1 and C_2 in $\mathbb{R}^n$, one has $C_1 \subseteq C_2$ if and only if $\delta(\cdot \mid C_1) \leq \delta(\cdot \mid C_2)$.

In general, given two CAS, the cardinality of the perturbation symbols is not necessarily the same. Nevertheless, we can suppose that it is always the case, without loss of generality, by completing either P^X or P^Y by null columns. Let m denote such cardinality.

For a given CAS, $\hat{X}$, the sets Φ_ϵ^X and Φ_η^X are now two hypercubes. Moreover, the set $\Phi_\eta^X(\omega)$, for a given ω in Φ_ϵ^X, is independent from ω and is equal to Φ_η^X for all ω. Thus, the concretisation function $\gamma_{1\times 2}$ (see Definition 4.2.2) of a CAS $\hat{X}$ can be seen as the Minkowski sum of two sets, namely $C^X\Phi_\epsilon^X$ and $P^X\Phi_\eta^X$.

The order $\leq_{1\times 2}$ can be stated with respect to the support function.

4.3.3 Lemma
Given two CAS $\hat{X}$ and $\hat{Y}$, we have $\hat{X} \leq_{1\times 2} \hat{Y}$ if and only if $\Phi_\epsilon^X \subseteq \Phi_\epsilon^Y$ and

$$\forall t \in \mathbb{R}^p, \quad \delta(t \mid (C^X - C^Y)\Phi_\epsilon^X) \leq \delta(t \mid P^Y\Phi_\eta^Y) - \delta(t \mid P^X\Phi_\eta^X) \ .$$

Proof. *All we need to prove is the fact that the inequality written using the support function is equivalent to the sets inclusion*

$$\forall \omega \in \Phi_\epsilon^X, \quad C^X\omega + P^X\Phi_\eta^X \subseteq C^Y\omega + P^Y\Phi_\eta^Y,$$

given by the definition of the order (Definition 4.2.5). Using Proposition 4.3.2, for a fixed $\omega \in \Phi_\epsilon^X$, the sets inclusion is equivalent to

$$\forall t \in \mathbb{R}^p, \quad \delta(t \mid C^X\omega + P^X\Phi_\eta^X) \leq \delta(t \mid C^Y\omega + P^Y\Phi_\eta^Y) \ .$$

However, the support function of the Minkowski sum of two sets is equal to the sum of the support functions over each set (see Remark 2 of the Appendix), thus

$$\delta(t \mid C^X\omega + P^X\Phi_\eta^X) = \delta(t \mid C^X\omega) + \delta(t \mid P^X\Phi_\eta^X) = \langle t, C^X\omega\rangle + \delta(t \mid P^X\Phi_\eta^X) \ .$$

This makes

$$\forall t \in \mathbb{R}^p, \quad \langle t, C^X\omega\rangle + \delta(t \mid P^X\Phi_\eta^X) \leq \langle t, C^Y\omega\rangle + \delta(t \mid P^Y\Phi_\eta^Y)$$

or equivalently

$$\forall t \in \mathbb{R}^p, \quad \langle t, (C^X - C^Y)\omega\rangle \leq \delta(t \mid P^Y\Phi_\eta^Y) - \delta(t \mid P^X\Phi_\eta^X) \ .$$

The above inequality is satisfied for all $\omega \in \Phi_\epsilon^X$, then by definition of the support function

$$\forall t \in \mathbb{R}^p, \quad \delta(t \mid (C^X - C^Y)\Phi_\epsilon^X) \leq \delta(t \mid P^Y\Phi_\eta^Y) - \delta(t \mid P^X\Phi_\eta^X) \ ,$$

which ends the proof. ∎

Symmetric Representative

The formulation of the order using the support function allows a characterization of a particular representation of the equivalence class of a given constrained affine set.

We bind each perturbation noise symbol to a coordinate and consider, as said earlier, that all CAS have the same number of perturbation noise symbols m. Let $\hat{X} = (C^X, P^X, \Phi^X)$, and $\hat{Y} = (C^Y, P^Y, \Phi^Y)$ be two CAS, and let (A^X, b^X) (resp. (A^Y, b^Y)) be the affine map that transforms the unit ball with respect to the uniform norm of dimension m, $\mathfrak{B}$, into Φ_η^X (resp. Φ_η^Y). Such a map is in fact unique up to the permutation of the columns of matrices A^X and A^Y.

4.3.4 Definition (Symmetric Representative)
Let $\hat{X} = (C^X, P^X, \Phi^X)$ be a CAS. The symmetric representative of $\hat{X}$ is defined by

$$(C, P, \Phi) \stackrel{\text{def}}{=} (((b^X + C^X_{(\cdot,0)})C^X_{(\cdot,1)} \dots C^X_{(\cdot,n)}), P^X A^X, \Phi_\epsilon^X \times \mathfrak{B}) \ .$$

The first column of matrix C is the sum of vector b^X and the first column of matrix C^X. The remaining columns of C are equal to the ones of C^X. Matrix P is the product of matrices P^X and A^X. The intervals of the perturbation noise symbols are all set to $[-1, 1]$.

The proposition below characterizes the equivalence classes related to the binary relation $\sim$.

4.3.5 Proposition
Let $\hat{X} = (C^X, P^X, \Phi^X)$, and $\hat{Y} = (C^Y, P^Y, \Phi^Y)$ be two CAS, and let (A^X, b^X) (resp. (A^Y, b^Y)) be the affine map that transforms the unit ball with respect to the uniform norm, $\mathfrak{B}$, into Φ^X_η (resp. Φ^Y_η). If:

$$\begin{aligned}
\Phi^X_\epsilon &= \Phi^Y_\epsilon, && \textit{(sets equality)}\\
C^X_{i,j} &= C^Y_{i,j}, 1 \leq i \leq p, 1 \leq j \leq n\\
b^X + L_0^{C^X} &= b^Y + L_0^{C^Y},\\
P^X A^X \mathfrak{B} &= P^Y A^Y \mathfrak{B}, && \textit{(sets equality)}
\end{aligned}$$

then $\hat{X} \sim \hat{Y}$.

Proof. *We prove that $\hat{X} \leq_{1\times 2} \hat{Y}$ and $\hat{Y} \leq_{1\times 2} \hat{X}$. Let $t \in \mathbb{R}^p$. By definition of C^X and C^Y, matrix $C^X - C^Y$ is null everywhere except its first column which is equal to $P^Y b^Y - P^X b^X$. Therefore, $(C^X - C^Y)\Phi^X_\epsilon = \{P^Y b^Y - P^X b^X\}$ (recall that $\epsilon_0 = 1$). We then have*

$$\delta(t \mid (C^X - C^Y)\Phi^X_\epsilon) = \langle P^Y b^Y - P^X b^X, t\rangle \ .$$

On the other hand, by hypothesis, (A^X, b^X) transforms the unit ball $\mathfrak{B}$, into Φ^X_η, which gives $\Phi^X_\eta = b^X + A^X\mathfrak{B}$. Similarly, $\Phi^Y_\eta = b^Y + A^Y\mathfrak{B}$. Therefore,

$$\begin{aligned}
\delta(t \mid P^X\Phi^X_\eta) &= \langle P^X b^X, t\rangle + \delta(t \mid P^X A^X \mathfrak{B}),\\
\delta(t \mid P^Y\Phi^Y_\eta) &= \langle P^Y b^Y, t\rangle + \delta(t \mid P^Y A^Y \mathfrak{B}), \ .
\end{aligned}$$

We have $P^X A^X \mathfrak{B} = P^Y A^Y \mathfrak{B}$, thus $\delta(t \mid P^X A^X \mathfrak{B}) = \delta(t \mid P^Y A^Y \mathfrak{B})$. Now,

$$\begin{aligned}
\delta(t \mid (C^X - C^Y)\Phi_\epsilon) &= \langle P^Y b^Y - P^X b^X, t\rangle\\
&= \langle P^Y b^Y, t\rangle - \langle P^X b^X, t\rangle\\
&= \delta(t \mid P^Y\Phi^Y_\eta) - \delta(t \mid P^X\Phi^X_\eta) \ .
\end{aligned}$$

The equality

$$\delta(t \mid (C^X - C^Y)\Phi_\epsilon) = \delta(t \mid P^Y\Phi^Y_\eta) - \delta(t \mid P^X\Phi^X_\eta),$$

together with $\Phi^X_\epsilon = \Phi^Y_\epsilon$, makes $\hat{X} \leq_{1\times 2} \hat{Y}$ and $\hat{Y} \leq_{1\times 2} \hat{X}$. Thus, $\hat{X} \sim \hat{Y}$. ∎

Using this equivalence, we prove the equivalence of a given constrained affine set $\hat{X}$ and its particular representative introduced in Definition 4.3.4. This representative is convenient as its perturbation set is a symmetric convex set ($C = -C$).

4.3.6 Corollary
Let $\hat{X} = (C^X, P^X, \Phi^X)$ be a CAS, let $\hat{Y}$ denote its symmetric representative as defined in Definition 4.3.4, then $\hat{X} \sim \hat{Y}$.

Proof. *We check that the CAS $\hat{X}$ and its symmetric representative $\hat{Y}$ satisfy proposition 4.3.5. Indeed, $\Phi^X_\epsilon = \Phi^Y_\epsilon$, and by definition of $\hat{Y}$,*

$$P^Y = A^X P^X \tag{4.3.1}$$
$$\Phi^Y_\eta = \mathfrak{B} \tag{4.3.2}$$
$$L_0^{C^Y} = b^X + L_0^{C^X} \tag{4.3.3}$$
$$C^X_{i,j} = C^Y_{i,j}, 1 \leq i \leq p, 1 \leq j \leq n \ . \tag{4.3.4}$$

It remains to check that ı) $b^Y + L_0^{C^Y} = b^X + L_0^{C^X}$ and ıı) $P^X A^X \mathfrak{B} = P^Y A^Y \mathfrak{B}$. We start with ı). Equation 4.3.2 makes $b^Y = 0$ and A^Y equal to the identity matrix. Since $b^Y = 0$, and using equation (4.3.3), we obtain

$$b^Y + L_0^{C^Y} = 0 + b^X + L_0^{C^X} = b^X + L_0^{C^X} \ .$$

We now check ıı). By equation 4.3.1, and using again the fact that $b^Y = 0$ and A^Y is the identity matrix, we obtain

$$P^X A^X \mathfrak{B} = P^Y \mathfrak{B} = P^Y A^Y \mathfrak{B} \ . \qquad \blacksquare$$

If not mentioned otherwise, any CAS is represented by its related symmetric representative. As the boxes of all perturbation noise symbols are equal to $[-1, 1]$, we use Φ^X_ϵ instead of Φ^X.

Notice that using the formulation of the order with symmetric representatives, both $C^X - C^Y$ or $C^Y - C^X$ can be used indifferently. Moreover, the set Φ^X_ϵ can be extended, without loss of generality, to the convex combination of boxes Φ^X_ϵ and $-\Phi^X_\epsilon$, which is a symmetric convex set, namely an origin-centered zonotope.

We introduce the primitive $\text{bound}_2 : \mathcal{A}_1 \times \mathcal{A}_2 \to \mathbb{I}$, which bounds the expression (given in its first argument) with respect to the abstract object Φ (given in its second argument). Such a primitive is usually provided in numerical abstract domains, at least whenever the expression to be bounded is linear.

4.3.7 Proposition
Let $\hat{X} = (C^X, P^X, \Phi_\epsilon^X)$ and $\hat{Y} = (C^Y, P^Y, \Phi_\epsilon^Y)$ be two CAS (in their symmetric representations). Let n denote the number of input noise symbols, and $\boldsymbol{\epsilon_j^X} = \mathrm{bound}_2(\epsilon_i^X, \Phi_\epsilon^X)$, then $\hat{X} \leq_{1\times 2} \hat{Y}$ if and only if $\Phi_\epsilon^X \subseteq \Phi_\epsilon^Y$ and,

$$\forall t \in \mathbb{R}^p, \quad \delta(t \mid (C^X - C^Y)M^{X^*}\mathfrak{B}) \leq \delta(t \mid P^Y\mathfrak{B}) - \delta(t \mid P^X\mathfrak{B}),$$

where the square matrix M^X of dimension $(n+1)\times(n+1)$ is defined by

$$\forall i,j \in \{1,\ldots,n+1\},$$
$$M_{ij}^X = \begin{cases} 1 & \text{if } i=j=1, \\ \mathrm{mid}(\boldsymbol{\epsilon_j^X}) & \text{if } i=1 \text{ and } 1<j\leq n, \\ \mathrm{dev}(\boldsymbol{\epsilon_j^X}) & \text{if } 1<i,j\leq n \text{ and } i=j \text{ and } \mathrm{dev}(\boldsymbol{\epsilon_j^X})\neq 0, \\ 0 & \text{otherwise.} \end{cases}$$

In words, the upper left corner of M^X is 1, the centers of the intervals $\boldsymbol{\epsilon_j^X}$ are on the first line, and their deviations on the diagonal of M^X.

Proof. *Let $\hat{X} = (C^X, P^X, \Phi_\epsilon^X)$ and $\hat{Y} = (C^Y, P^Y, \Phi_\epsilon^Y)$ be two CAS such that $\hat{X} \leq_{1\times 2} \hat{Y}$. According to the definition of the order, and the definition of the symmetric representative, one has for all $t \in \mathbb{R}^p$*

$$\delta(t \mid (C^X - C^Y)\Phi_\epsilon^X) \leq \delta(t \mid P^Y\mathfrak{B}) - \delta(t \mid P^X\mathfrak{B}),$$

which gives for $-t$

$$\delta(-t \mid (C^X - C^Y)\Phi_\epsilon^X) \leq \delta(-t \mid P^Y\mathfrak{B}) - \delta(-t \mid P^X\mathfrak{B}),$$

or equivalently

$$\delta(t \mid (C^Y - C^X)\Phi_\epsilon^X) \leq \delta(t \mid -P^Y\mathfrak{B}) - \delta(t \mid -P^X\mathfrak{B}) \ .$$

The sets $P^X\mathfrak{B}$ and $P^Y\mathfrak{B}$ are symmetric, thus $-P^X\mathfrak{B} = P^X\mathfrak{B}$ and similarly $-P^Y\mathfrak{B} = P^Y\mathfrak{B}$. Therefore,

$$\delta(t \mid (C^Y - C^X)\Phi_\epsilon^X) \leq \delta(t \mid P^Y\mathfrak{B}) - \delta(t \mid P^X\mathfrak{B}) \ .$$

Now, if $(\epsilon_0,\ldots,\epsilon_n) \in \Phi_\epsilon^X \subset 1\times\mathbb{R}^n$, we want to prove that the inequality

$$\delta(t \mid (C^X - C^Y)\Phi_\epsilon^X) \leq \delta(t \mid P^Y\mathfrak{B}) - \delta(t \mid P^X\mathfrak{B}),$$

remains valid if ϵ_0 lies within $[-1,1]$ instead of being constrained to be equal to 1.

Suppose that the set obtained when we consider ϵ_0 *to be within* $[-1,1]$ *is exactly the convex combination of* $(C^X - C^Y)\Phi_\epsilon^X$ *and* $(C^Y - C^X)\Phi_\epsilon^X$ *(by definition, the convex combination of two sets* C_1 *and* C_2 *is the set* $\lambda C_1 + (1-\lambda)C_2$*, where* λ *ranges over* $[0,1]$*.), then the result is immediate by summing up the two inequalities*

$$\delta(t \mid \lambda(C^X - C^Y)\Phi_\epsilon^X) \leq \lambda(\delta(t \mid P^Y\mathfrak{B}) - \delta(t \mid P^X\mathfrak{B}))$$
$$\delta(t \mid (1-\lambda)(C^Y - C^X)\Phi_\epsilon^X) \leq (1-\lambda)(\delta(t \mid P^Y\mathfrak{B}) - \delta(t \mid P^X\mathfrak{B}))$$

Now let's prove that the convex combination of $(C^X - C^Y)\Phi_\epsilon^X$ *and* $(C^Y - C^X)\Phi_\epsilon^X$ *is the set obtained when we consider* ϵ_0 *to be within* $[-1,1]$*.*

By definition the convex combination of $(C^X - C^Y)\Phi_\epsilon^X$ *and* $(C^Y - C^X)\Phi_\epsilon^X$ *is equal to* $\lambda(C^X - C^Y)\Phi_\epsilon^X + (1-\lambda)((C^Y - C^X)\Phi_\epsilon^X)$*, where* λ *ranges over* $[0,1]$*. Let* g_0 *and* $(g_1, \ldots, g_n)$ *be respectively the center and the generators' list of the zonotope* $(C^X - C^Y)\Phi_\epsilon^X$*:*

$$(C^X - C^Y)\Phi_\epsilon^X = \{g_0 + \sum_{i=1}^{n} g_i\beta_i \mid \forall i, \beta_i \in [-1,1]\}$$

Therefore, $-g_0$ *and* $(g_1, \ldots, g_n)$ *are respectively the center and the generators' list of the zonotope* $(C^Y - C^X)\Phi_\epsilon^X$*.*

We then deduce that 0 *and* $(g_0, g_1, \ldots, g_n)$ *are respectively the center and the generators' list of the convex combination of* $(C^X - C^Y)\Phi_\epsilon^X$ *and* $(C^Y - C^X)\Phi_\epsilon^X$*. Indeed, any element of such convex combination can be written as* $\lambda u + (1-\lambda)v$*, where* $\lambda \in [0,1]$*,* $u = g_0 + \sum_{i=1}^n g_i\beta_i^u$ *and* $v = -g_0 + \sum_{i=1}^n g_i\beta_i^v$*,* β_i^u*,* $\beta_i^v \in [-1,1]$*. Thus,* $\lambda u + (1-\lambda)v = (-1+2\lambda)g_0 + \sum_{i=1}^n g_i(\lambda\beta_i^u + (1-\lambda)\beta_i^v)$*, where* $(-1+2\lambda) \in [-1,1]$ *and for all* i*,* $(\lambda\beta_i^u + (1-\lambda)\beta_i^v) \in [-1,1]$*.*

The zonotope spanned by $(g_0, g_1, \ldots, g_n)$ *(which is a symmetric convex set) is the one obtained by* $M^{X*}\mathfrak{B}$*. Indeed, the first generator* g_0 *is defined by* $(1, \mathrm{dev}(\boldsymbol{\epsilon_1^X}), \ldots, \mathrm{dev}(\boldsymbol{\epsilon_n^X}))^*$*, and the other generators* g_i*,* $1 \leq i \leq n$*, are defined by* $(0, \ldots, \mathrm{dev}(\boldsymbol{\epsilon_i^X}), \ldots)^*$*,* $1 \leq i \leq n$*. Therefore,*

$$\delta(t \mid (C^X - C^Y)M^{X*}\mathfrak{B}) \leq \delta(t \mid P^Y\mathfrak{B}) - \delta(t \mid P^X\mathfrak{B}),$$

The final step, that is moving matrix M^X *from "right to left" in the support function by applying the transpose operator (one has* $M^{X**} = M^X$*) is a direct consequence of Proposition A.0.5.* ∎

4.3.8 Example

If $\Phi_\epsilon^X = 1 \times [-1,0] \times [0,0.5]$ *(*$n = 2$*), then*

$$M^X = \begin{pmatrix} 1 & -0.5 & 0.25 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.25 \end{pmatrix}$$

The convex combination of Φ_ϵ^X *and* $-\Phi_\epsilon^X$ *is indeed equal to* $M^{X^*}\mathfrak{B}$ *(of dimension* 3*).*

Decidability

To decide whether $\hat{X}$ is less than or equal to $\hat{Y}$, using Lemma 4.3.3, it is necessary and sufficient to check the inclusion of Φ_ϵ^X and Φ_ϵ^Y, as well as, the set inclusion of two sets, namely $(C^X - C^Y)\Phi_\epsilon^X + P^X\Phi_\eta^X$ and $P^Y\Phi_\eta^Y$. The former inclusion is straightforward, it consists in checking the inclusion of n intervals, where n denotes the number of input noise symbols in use.

The latter involves the inclusion of two zonotopes since the Minkowski sum of two zonotopes is a zonotope. The generators of the resulting zonotope are simply the union of the generators of its two operands.

4.3.9 Theorem
The partial order $\leq_{1\times 2}$ *is decidable, with a complexity bounded by* $O(2^{n+2m}C)$*, where* C *denotes the complexity of solving a linear program of* p *variables and* $(n+2m)$ *constraints (each LP can be solved in* $O(p^{3.5}L)$ *using interior point methods [Kar84], where* L *denotes the bit length of the input data* [2]*).*

Proof. *We would like to decide the inclusion of two zonotopes, namely* $(C^X - C^Y)\Phi_\epsilon^X + P^X\Phi_\eta^X$*, and* $P^Y\Phi_\eta^Y$*. Let us denote by* $\mathcal{Z}_1$ *the former zonotope, and by* $\mathcal{Z}_2$ *the latter one. The zonotopes* $\mathcal{Z}_1$ *and* $\mathcal{Z}_2$ *are spanned by* $n_1 = n + m$ *and* $n_2 = m$ *generators respectively. The problem of the inclusion* $\mathcal{Z}_1 \subseteq \mathcal{Z}_2$ *can be stated using the support function as follows*

$$\mathcal{Z}_1 \subseteq \mathcal{Z}_2 \iff \forall t \in \mathbb{R}^p, \delta(t \mid \mathcal{Z}_1) \leq \delta(t \mid \mathcal{Z}_2) \ .$$

Let (A_1, b_1) *(resp.* (A_2, b_2)*) be the affine map that transforms the unit ball* $\mathfrak{B}$ *into* $\mathcal{Z}_1$ *(resp.* $\mathcal{Z}_2$*). We have then to decide, for all* t*, whether*

$$\langle b_1, t\rangle + \delta(A_1^* t \mid \mathfrak{B}) \leq \langle b_2, t\rangle + \delta(A_2^* t \mid \mathfrak{B}),$$

or equivalently

$$\langle b_1 - b_2, t\rangle + \|A_1^* t\|_1 - \|A_2^* t\|_1 \leq 0 \ .$$

Each line of A_1^* *(resp.* A_2^**), which is a generator of* $\mathcal{Z}_1$ *(resp.* $\mathcal{Z}_2$*), defines a hyperplane that contains the origin. We then have a partition of the space* $\mathbb{R}^p$ *with* $n_1 + n_2 = n + 2m$ *hyperplanes containing the origin. (The*

generators of the two zonotopes $\mathcal{Z}_1$ and $\mathcal{Z}_2$ are exactly the normal vectors to these hyperplanes.) The worst number of cells in such arrangement is $2^{n_1+n_2}$. Each cell is a polyhedron P_i, $1 \leq i \leq 2^{n_1+n_2}$, where the function $\langle b_1 - b_2, t\rangle + \|A_1^* t\|_1 - \|A_2^* t\|_1$ is perfectly linear. We solve the following LP :

$$\begin{aligned} \max \quad & \langle b_1 - b_2, t\rangle + \|A_1^* t\|_1 - \|A_2^* t\|_1 \\ s.t. \quad & t \in P_i \end{aligned}$$

If for each LP the objective value is less than or equal to zero, then the inclusion holds. Else, the procedure terminates immediately returning "false". ■

The best-case complexity is the one of solving exactly one LP. The worst-case complexity is the one given in Theorem 4.3.9, this bound is reached whenever the inclusion holds.

In static analysis by abstract interpretation, whenever we have a loop, for each iteration in the abstract domain, we have to compare two abstract objects. Hopefully, the worst case complexity holds only once, that is when a fixpoint is reached (which ends the analysis of the loop.).

CHAPTER 5

Assignment and Interpretation of Tests

We define the arithmetic over constrained affine forms and explain how we retrieve the available information encoded by the noise symbols abstract object in order to improve the evaluation of the expressions, both linear and non-linear. On the other hand we formalize the interpretation of tests, a key feature of our abstract domain : tests are projected on the noise symbols world as constraints, then interpreted using the abstract transfer functions of the noise symbols domain.

Contents Section 5.1 focuses on abstract assignment over constrained affine sets. Section 5.2 handles the abstraction of the tests statements: equality tests, section 5.2, and inequality tests, section 5.2.

5.1 Abstract Assignment

We extend the set of variables $\mathcal{V}$ with the special variable $v_0 = 1$ to encode constants. The abstract set of environment $\Sigma^\sharp : \wp(\mathcal{V}) \to \mathcal{A}_{1\times 2}$ maps a subset of variables to $\hat{X} \in \mathcal{A}_{1\times 2}$. For the sake of simplicity, we always abstract the set of all variables. Thus, any $\sigma^\sharp \in \Sigma^\sharp$ maps $\mathcal{V}$ to an object of $\mathcal{A}_{1\times 2}$. So, one can simply consider $\mathcal{A}_{1\times 2}$ instead of its lifted abstract domain $\Sigma^\sharp : \mathcal{V} \to \mathcal{A}_{1\times 2}$.

We denote by L_i^M the ith line of matrix M. The semantics of the

assignment function is defined by:

$$\begin{aligned}
[\![v_k \leftarrow e]\!]^\sharp(C,P,\Phi) &\stackrel{\text{def}}{=} (C',P',\Phi'),\\
&\text{where } \forall i \neq k, L_i^{C'} = L_i^C, L_i^{P'} = L_i^P,\\
&\quad \forall i, 0 \leq i \leq n, C'_{k,i} = \alpha_i^e, \forall j, 1 \leq j \leq m, P'_{k,j} = \beta_j^e,\\
&\quad (\sum_{i=0}^{n} \alpha_i^e \epsilon_i + \sum_{j=1}^{m} \beta_j^e \eta_j, \Phi') \stackrel{\text{def}}{=} [\![e]\!]^\sharp(C,P,\Phi).
\end{aligned}$$

Only the kth line of matrices C and P is updated after the assignment of the expression e to the variable v_k. The new coefficients of the kth line, that is α_i^e, $0 \leq i \leq n$, and β_j^e, $1 \leq j \leq m$, come from the evaluation of the expression e with respect to the abstract object (C,P,Φ). The semantics of the evaluation of an expression $e \in expr$ is given by:

$$\begin{aligned}
&\forall e \in expr, [\![e]\!]^\sharp : \mathcal{A}_{1\times 2} \to \mathcal{A}_1 \times \mathcal{A}_2\\
[\![v_k]\!]^\sharp(C,P,\Phi) &\stackrel{\text{def}}{=} (\sum_{i=0}^{n} C_{k,i}\epsilon_i + \sum_{j=1}^{m} P_{k,j}\eta_j, \Phi)\\
[\![[a,b]]\!]^\sharp(C,P,\Phi) &\stackrel{\text{def}}{=} \begin{cases} (\frac{a+b}{2} + \frac{b-a}{2}\epsilon_f, [\![-1 \leq \epsilon_f \leq 1]\!]_2^\sharp [\![\text{add}\,\epsilon_f]\!]_2^\sharp \Phi), \\ \quad \text{if } -\infty < a \leq b < +\infty, \\ (\epsilon_f, [\![\epsilon_f \leq b]\!]_2^\sharp [\![\text{add}\,\epsilon_f]\!]_2^\sharp \Phi), \quad \text{if } -\infty = a \\ (\epsilon_f, [\![a \leq \epsilon_f]\!]_2^\sharp [\![\text{add}\,\epsilon_f]\!]_2^\sharp \Phi), \quad \text{if } +\infty = b \end{cases}\\
[\![e_1 \diamond e_2]\!]^\sharp(C,P,\Phi) &\stackrel{\text{def}}{=} [\![e_1]\!]^\sharp(C,P,\Phi) \diamond [\![e_2]\!]^\sharp(C,P,\Phi)\\
&\quad \text{where } \diamond \in \{+_{1\times 2}, -_{1\times 2}, \times_{1\times 2}, \div_{1\times 2}\}\\
[\![\sqrt{e}]\!]^\sharp(C,P,\Phi) &\stackrel{\text{def}}{=} \surd_{1\times 2} [\![e]\!]^\sharp(C,P,\Phi)
\end{aligned}$$

Notice that the evaluation of an expression is by definition with respect to the same abstract object (C,P,Φ). Therefore, all noise symbols (input and perturbation) are shared between the involved operands.

The abstract operator $[\![\text{add}\,\epsilon_f]\!]_2^\sharp : \mathcal{A}_2 \to \mathcal{A}_2$ formalizes the add of a fresh noise symbol ϵ_f to the abstract object Φ. Here we add a new input noise symbol ϵ_f, however, the operator could be used to add a fresh perturbation noise symbol η_f as well. (From a point of view of $\mathcal{A}_2$, all noise symbols are variables and there is no more difference between them.)

Let $\hat{x} = \sum_{i=0}^{n} \alpha_i^x \epsilon_i + \sum_{j=1}^{m} \beta_j^x \eta_j$, and $\hat{y} = \sum_{i=0}^{n} \alpha_i^y \epsilon_i + \sum_{j=1}^{m} \beta_j^y \eta_j$ be two elements of $\mathcal{A}_1$, and Φ an element of $\mathcal{A}_2$. The linear operations $\{+_{1\times 2}, -_{1\times 2}\}$ are defined by their related operations in affine arithmetic. The abstract

element Φ is unused and remains unchanged:

$$(\hat{x}, \Phi) +_{1\times 2} (\hat{y}, \Phi) \stackrel{\text{def}}{=} (\sum_{i=0}^{n}(\alpha_i^x + \alpha_i^y)\epsilon_i + \sum_{j=1}^{m}(\beta_j^x + \beta_j^y)\eta_j, \Phi),$$
$$(\hat{x}, \Phi) -_{1\times 2} (\hat{y}, \Phi) \stackrel{\text{def}}{=} (\sum_{i=0}^{n}(\alpha_i^x - \alpha_i^y)\epsilon_i + \sum_{j=1}^{m}(\beta_j^x - \beta_j^y)\eta_j, \Phi) \ .$$

The scalar multiplication operation is defined by

$$\lambda._{1\times 2}(\hat{x}, \Phi) \stackrel{\text{def}}{=} (\sum_{i=0}^{n}(\lambda\alpha_i^x)\epsilon_i + \sum_{j=1}^{m}(\lambda\beta_j^x)\eta_j, \Phi) \ .$$

5.1.1 Proposition
The assignment of linear expression is monotonic.

Proof. *Given two constrained affine sets $\hat{X}$ and $\hat{Y}$ such that $\hat{X} \leq_{1\times 2} \hat{Y}$. We prove that $[\![v_k \leftarrow v_i + v_j]\!]^\sharp \hat{X} \leq_{1\times 2} [\![v_k \leftarrow v_i + v_j]\!]^\sharp \hat{Y}$. Let $\hat{A} = [\![v_k \leftarrow v_i + v_j]\!]^\sharp \hat{X}$ and $\hat{B} = [\![v_k \leftarrow v_i + v_j]\!]^\sharp \hat{Y}$. The abstract objects Φ^X and Φ^Y are unchanged, thus $\Phi^A = \Phi^X$ and $\Phi^B = \Phi^Y$, and the condition $\Phi_\epsilon^A \subseteq \Phi_\epsilon^B$ holds. Let $\omega \in \Phi_\epsilon^X$ and let $\eta_\omega^X \in \Phi_\eta^X(\omega)$. Since $\hat{X} \leq_{1\times 2} \hat{Y}$, there exists $\eta_\omega^Y \in \Phi_\eta^Y(\omega)$ such that*

$$C^X\omega + P^X\eta_\omega^X = C^Y\omega + P^Y\eta_\omega^Y,$$

thus

$$\langle L_i^{C^X}, \omega\rangle + \langle L_i^{P^X}, \eta_\omega^X\rangle = \langle L_i^{C^Y}, \omega\rangle + \langle L_i^{P^Y}, \eta_\omega^Y\rangle,$$
$$\langle L_j^{C^X}, \omega\rangle + \langle L_j^{P^X}, \eta_\omega^X\rangle = \langle L_j^{C^Y}, \omega\rangle + \langle L_j^{P^Y}, \eta_\omega^Y\rangle \ .$$

We prove that $C^A\omega + P^A\eta_\omega^X = C^B\omega + P^B\eta_\omega^Y$. Matrix C^A (resp. P^A) is equal to matrix C^X (resp. P^X) except for the kth line. Likewise the matrices C^B and P^B are equal everywhere to C^Y and P^Y respectively except for the kth line. For the kth line we have

$$L_k^{C^A} = L_i^{C^X} + L_j^{C^X}, \qquad L_k^{P^A} = L_i^{P^X} + L_j^{P^X},$$
$$L_k^{C^B} = L_i^{C^Y} + L_j^{C^Y}, \qquad L_k^{P^A} = L_i^{P^Y} + L_j^{P^Y} \ .$$

Therefore,

$$\langle L_k^{C^A}, \omega \rangle + \langle L_k^{P^A}, \eta_\omega^X \rangle = \langle L_k^{C^B}, \omega \rangle + \langle L_k^{P^B}, \eta_\omega^Y \rangle \ .$$

The operations $-_{1\times 2}$ and $._{1\times 2}$ can be proved monotonic in a similar manner. Any linear expression can be seen as a composition of these basic operations and the composition of monotonic operations is monotonic. ∎

Non linear binary operations $\{\times_{1\times 2}, \div_{1\times 2}\}$ and the unary operation $\{\sqrt{}_{1\times 2}\}$ benefit from both abstract domains $\mathcal{A}_1$ and $\mathcal{A}_2$ for a better precision.

Multiplication

The multiplication operation benefits from the interval concretisation of every noise symbol. We detail the idea through a small example, then give the formal definition.

Let $\hat{x} \stackrel{\text{def}}{=} \epsilon_1$ and $\hat{y} = \epsilon_2$, and $\Phi = 1 \times [0.5, 1]^2$. Since ϵ_1 and ϵ_2 are independent variables, the exact range of the expression $\epsilon_1 \times \epsilon_2$ is given by interval arithmetic, that is $[0.5, 1] \times [0.5, 1] = [0.25, 1]$. The expression $\epsilon_1 \times \epsilon_2$ is non-linear. The naive solution which transforms the interval $[0.25, 1]$ into the affine form $\text{mid}([0.25, 1]) + \text{dev}([0.25, 1])\eta_f$, $\eta_f \in [-1, 1]$ definitely loses all relations with ϵ_1 and ϵ_2 while giving a perturbation deviation of $\frac{1-0.25}{2} = 0.375$. A better solution "extracts" first the linear component of $\epsilon_1 \times \epsilon_2$:

$$\begin{aligned}
\epsilon_1 \times \epsilon_2 &= (\epsilon_1 - 0.75 + 0.75) \times (\epsilon_2 - 0.75 + 0.75) \\
&= 0.75^2 + 0.75(\epsilon_1 - 0.75) + 0.75(\epsilon_2 - 0.75) + (\epsilon_1 - 0.75)(\epsilon_2 - 0.75) \\
&\in -0.75^2 + 0.75\epsilon_1 + 0.75\epsilon_2 + [-0.25, 0.25] \times [-0.25, 0.25] \\
&\in -0.5625 + 0.75\epsilon_1 + 0.75\epsilon_2 + [-0.0625, 0.0625]
\end{aligned}$$

The non-linear term considered now, $(\epsilon_1 - 0.75) \times (\epsilon_2 - 0.75)$ has a concretisation equal to $[-0.0625, 0.0625]$, which is 6 times less than $[-0.375, 0.375]$, obtained by the naive solution.

The multiplication operation is defined by:

$$(\hat{x},\Phi)\times(\hat{y},\Phi) \stackrel{\text{def}}{=} (\sum_{i=0}^{n+1}\alpha_i\epsilon_i + \sum_{j=1}^{m}\beta_j\eta_j + \beta_f\eta_f, [\![-1\le\eta_f\le 1]\!]_2^\sharp \circ [\![\operatorname{add}\eta_f]\!]_2^\sharp\Phi)$$

where

$$\begin{aligned}
\alpha_0 &= -\operatorname{mid}([\hat{x}])\operatorname{mid}([\hat{y}]) + \operatorname{mid}(\Delta)\\
\alpha_i &= \operatorname{mid}([\hat{x}])\alpha_i^y + \operatorname{mid}([\hat{y}])\alpha_i^x, \quad 1\le i\le n\\
\beta_j &= \operatorname{mid}([\hat{x}])\beta_i^y + \operatorname{mid}([\hat{y}])\beta_i^x, \quad 1\le j\le m\\
\beta_f &= \operatorname{dev}(\Delta)\\
\Delta &= \operatorname{bound}_2(\sum_{i=1}^{n}\alpha_i^x(\epsilon_i-\operatorname{mid}(\boldsymbol{\epsilon_i}))\sum_{i=1}^{n}\alpha_i^y(\epsilon_i-\operatorname{mid}(\boldsymbol{\epsilon_i}))\\
&\quad + \sum_{i=1}^{n}\alpha_i^x(\epsilon_i-\operatorname{mid}(\boldsymbol{\epsilon_i}))\sum_{j=1}^{m}\beta_i^y(\eta_j-\operatorname{mid}(\boldsymbol{\eta_j}))\\
&\quad + \sum_{j=1}^{m}\beta_i^x(\eta_j-\operatorname{mid}(\boldsymbol{\eta_j}))\sum_{i=1}^{n}\alpha_i^y(\epsilon_i-\operatorname{mid}(\boldsymbol{\epsilon_i}))\\
&\quad + \sum_{j=1}^{m}\beta_i^x(\eta_j-\operatorname{mid}(\boldsymbol{\eta_j}))\sum_{j=1}^{m}\beta_i^y(\eta_j-\operatorname{mid}(\boldsymbol{\eta_j})),\Phi)\\
[\hat{x}] &= \operatorname{bound}_2(\hat{x},\Phi),\\
[\hat{y}] &= \operatorname{bound}_2(\hat{y},\Phi)\ .
\end{aligned}$$

Recall that primitive $\operatorname{bound}_2 : \mathcal{A}_1\times\mathcal{A}_2\to\mathbb{I}$ bounds the expression given in its first argument with respect to the abstract object Φ given in its second argument. Computing Δ is not immediate. If $\mathcal{A}_2$ is a polyhedra-like abstract domain, then the computation needs to over-approximate a quadratic term over a polyhedron. We give hereafter two methods specifically tuned for the quadratic expressions.

Method 1 is based on interval arithmetic together with symbolic enhancement computations. The method distributes the multiplication, simplifies equal terms, then over-approximates each remaining term with an interval, and finally sums up these intervals. Simplification holds if $\alpha_i^x\alpha_j^y = -\alpha_j^x\alpha_i^y$ for some pair (i,j). The over-approximation step is smart enough to detect squares of intervals: $(\epsilon_i-\operatorname{mid}(\boldsymbol{\epsilon_i}))(\epsilon_i-\operatorname{mid}(\boldsymbol{\epsilon_i}))\in[0,\operatorname{dev}(\boldsymbol{\epsilon_i})^2]$ instead of $[-\operatorname{dev}(\boldsymbol{\epsilon_i})^2,\operatorname{dev}(\boldsymbol{\epsilon_i})^2]$ given by interval arithmetic. The final interval found is an over-approximation of the actual bounds reached by the expression.

Method 2 is more sophisticated and takes into account the dependency between noise symbols. It operates globally on the expression and uses SemiDefinite Programming (SDP) to compute tight bounds for Δ. The expression can be seen simply as $\sum_{i=1}^{n+m}\sum_{j=1}^{n+m}\zeta_i^x\zeta_j^y\delta_i\delta_j$ when all δ_i are within $[-1,1]$ where:

$$\begin{aligned}
\zeta_i^x &= \alpha_i^x \operatorname{dev}(\boldsymbol{\epsilon_i}),\ \zeta_i^y = \alpha_i^y \operatorname{dev}(\boldsymbol{\epsilon_i}) && 1 \le i \le n,\\
\zeta_i^x &= \beta_i^x \operatorname{dev}(\boldsymbol{\eta_i}),\ \zeta_i^y = \beta_i^y \operatorname{dev}(\boldsymbol{\eta_i}) && n+1 \le i \le n+m,\\
\delta_i &= \epsilon_i - \operatorname{mid}(\boldsymbol{\epsilon_i}), && 1 \le i \le n,\\
\delta_i &= \eta_i - \operatorname{mid}(\boldsymbol{\eta_i}), && n+1 \le i \le n+m,
\end{aligned}$$

and similarly for ζ_i^y and δ_i^y, $1 \le i \le n+m$ by substituting x by y. The following proposition bounds $\sup(\Delta)$ by a typical SDP program.

5.1.2 Proposition
The upper bound of Δ is bounded by:

$$\max_{|\delta_i|\le 1}\sum_{i=1}^{n+m}\sum_{j=1}^{n+m}\zeta_i^x\zeta_j^y\delta_i\delta_j = \max_{|\delta_i|\le 1}\delta^*.\Psi.\delta \le \inf_{\mu\in\mathbb{R}_+^{n+m}}\{\operatorname{trace}(\mu I_{n+m})|\Psi-\mu I_{n+m}\preceq 0\} \tag{S}$$

where $(\Psi_{i,j})_{1\le i,j\le n+m} = \frac{1}{2}(\zeta_i^x\zeta_j^y + \zeta_j^x\zeta_i^y)$ and $\Psi - \mu I_{n+m} \preceq 0$ denotes the fact that $\Psi - \mu I_{n+m}$ is negative SemiDefinite. The equality holds when matrix Ψ is negative SemiDefinite.

The infimum bound of Δ is computed similarly using the inequality

$$\min_{|\delta_i|\le 1}\sum_{i=1}^{n+m}\sum_{j=1}^{n+m}\zeta_i^x\zeta_j^y\delta_i\delta_j \le -\max_{|\delta_i|\le 1}\sum_{i=1}^{n+m}\sum_{j=1}^{n+m}(-\zeta_i^x)\zeta_j^y\delta_i\delta_j \ .$$

The first method is cost effective but gives coarse results for the non-linear term as it does not consider all dependencies between noise symbols. The second method gives tighter results but needs to solve two SDP programs with a polynomial complexity. Notice that the second method relies on the box that contains $\gamma_2(\Phi)$, and not $\gamma_2(\Phi)$ itself. It's hence an over-approximation of the actual bounds reached by the expression when the noise symbols lie within $\gamma_2(\Phi)$ unless $\mathcal{A}_2$ is the intervals abstract domain and Ψ is negative semidefinite.

5.1.3 Proposition
The abstract assignment operator $[\![v_k \leftarrow v_i \times v_j]\!]^\sharp$ is monotonic.

Proof. *The proof of the unconstrained case may be found [GP09].* ∎

Division

The division operation $\div_{1\times 2}$ is defined in two steps: we first compute the inverse then operate a multiplication:

$$(\hat{x}, \Phi) \div_{1\times 2} (\hat{y}, \Phi) \stackrel{\text{def}}{=} (1/\cdot_{1\times 2}(\hat{y}, \Phi)) \times_{1\times 2} (\hat{x}, \Phi) \ .$$

The inverse operation $1/\cdot_{1\times 2}$ is defined by:

$$\begin{aligned} 1/\cdot_{1\times 2}(\hat{x}, \Phi) &\stackrel{\text{def}}{=} (\zeta + \delta\hat{x} + \kappa\eta_f, [\![-1 \leq \eta_f \leq 1]\!]^\sharp [\![\text{add}\,\eta_f]\!]_2^\sharp \Phi), \\ &\text{where } [a, b] = \text{bound}_2(\hat{x}, \Phi) \text{ in} \\ &\qquad \delta = \frac{-4}{(a+b)^2} \\ &\qquad \kappa = -\frac{2b}{(a+b)^2} + \frac{1}{2a} \\ &\qquad \zeta = \frac{4}{a+b} + -\frac{2b}{(a+b)^2} + \frac{1}{2a} \ . \end{aligned}$$

The definition above supposes that $[a, b]$ does not contain zero and has finite bounds. The non-generic other cases are as follows:

$$1/\cdot_{1\times 2}(\hat{x}, \Phi) \stackrel{\text{def}}{=} \begin{cases} (\top_{1\times 2}, \Phi), & \text{if } 0 \in [a, b], \\ (\frac{1}{2a} + \frac{1}{2a}\eta_f, [\![-1 \leq \eta_f \leq 1]\!]^\sharp [\![\text{add}\,\eta_f]\!]_2^\sharp \Phi), & \\ \quad \text{if } b = +\infty & \\ (\frac{1}{2b} + \frac{-1}{2b}\eta_f, [\![-1 \leq \eta_f \leq 1]\!]^\sharp [\![\text{add}\,\eta_f]\!]_2^\sharp \Phi), & \\ \quad \text{if } a = -\infty & \end{cases}$$

Square Root

The square root is defined as follows:

$$\begin{aligned} \sqrt{}_{1\times 2}(\hat{x}, \Phi) &\stackrel{\text{def}}{=} (\zeta + \delta\hat{x} + \kappa\eta_f, [\![-1 \leq \eta_f \leq 1]\!]_2^\sharp [\![\text{add}\,\eta_f]\!]_2^\sharp \Phi), \\ &\text{where } [a, b] = [0, +\infty] \cap \text{bound}_2(\hat{x}, \Phi) \text{ in} \\ &\qquad \zeta = \frac{\sqrt{a+b}}{4\sqrt{2}} - \frac{1}{2\sqrt{2}\sqrt{a+b}} + \frac{\sqrt{a}}{2} \\ &\qquad \delta = \frac{1}{\sqrt{2(a+b)}} \\ &\qquad \kappa = \frac{\sqrt{a+b}}{4\sqrt{2}} + \frac{a}{2\sqrt{2}\sqrt{a+b}} - \frac{\sqrt{a}}{2} \ . \end{aligned}$$

Here, bound_2 gives the bounds of the affine expression $\hat{x}$ with respect to the abstract object Φ. Such a primitive is usually available in all numerical

abstract domains. If the underlying abstract domain does not implement such primitive, a relaxed version can be easily defined by taking the interval concretization of each variable in Φ, then use interval arithmetic. The real numbers ζ, δ and κ are computed using a special Taylor series development of the square root function, as detailed in Section 3.3. The definition above assumes that $[a, b]$ is not empty nor reduced to zero and its bounds are finite. All other cases are as follows:

$$\sqrt{}_{1\times 2}(\hat{x}, \Phi) \stackrel{\text{def}}{=} \begin{cases} (\bot_{1\times 2}, \Phi), & \text{if } [a, b] = \emptyset, \\ (0, \Phi), & \text{if } a = b = 0, \\ (\eta_f, [\![\sqrt{a} \leq \eta_f]\!]^\sharp [\![\text{add}\, \eta_f]\!]^\sharp_2 \Phi), & \text{if } b = +\infty \end{cases}$$

A typical flow-sensitive analyzer would emit a caveat and store the location if the interval $[a, b]$ admits negative values.

Soundness

Linearization of non-linear unary operations adds a new noise symbol η_f. For unary operations $f(x) \in \{\frac{1}{x}, \sqrt{x}\}$, where x is abstracted by $(\hat{x}, \Phi)$, we compute respectively the linear approximants $\{\sqrt{}_{1\times 2}(\hat{x}, \Phi), 1/\cdot_{1\times 2}(\hat{x}, \Phi)\}$. The joint range of $\hat{x}$ and $\diamond\hat{x}$, for $\diamond \in \{\sqrt{}_{1\times 2}, 1/\cdot_{1\times 2}\}$, has to enclose the graph of $(x, f(x))$, where the noise symbols range over $\gamma_2(\Phi)$. This condition is enforced by the soundness property that abstraction should respect: $X \leq^\flat \gamma \circ \alpha(X)$. In our case, X is given by the graph of the function $(x, f(x))$, its abstraction, $\alpha(X) = \hat{X}$, is the constrained affine set formed by the affine forms $\hat{x}$, $\diamond\hat{x}$ and the abstract element Φ:

$$\{(x_1, x_2) \mid x_2 = f(x_1)\} = X \subseteq \gamma_{1\times 2}(\hat{X}) \ .$$

If this condition is unsatisfied, then one might find a feasible configuration of noise symbols not represented by the linear approximant computed. Even if the interval range of the linear approximant contains the interval range of the approximated function, one can always come up with an "unsafe" future computation that exploits this unsound configuration.

We give hereafter an example of linearization which is unsound, but gives locally correct result if we consider only the interval range of the affine forms after the linearization. The example is picked up from the literature of reliable computing, it is given in [Kol07] to illustrate the so-called Kolev formula of multiplication over affine forms which yields no overestimation (under certain simple monotonicity conditions satisfied by the example).

5.1.4 Example

Consider $\hat{x} = 10 + 5\epsilon_1 + 3\epsilon_2$ *and* $\hat{y} = 10 - 2\epsilon_1 + \epsilon_3$. *All noise symbols are within* $[-1, 1]$. *The range of* $\hat{x}$ *is* $[2, 18]$, *and the range of* $\hat{y}$ *is* $[7, 13]$. *Kolev*

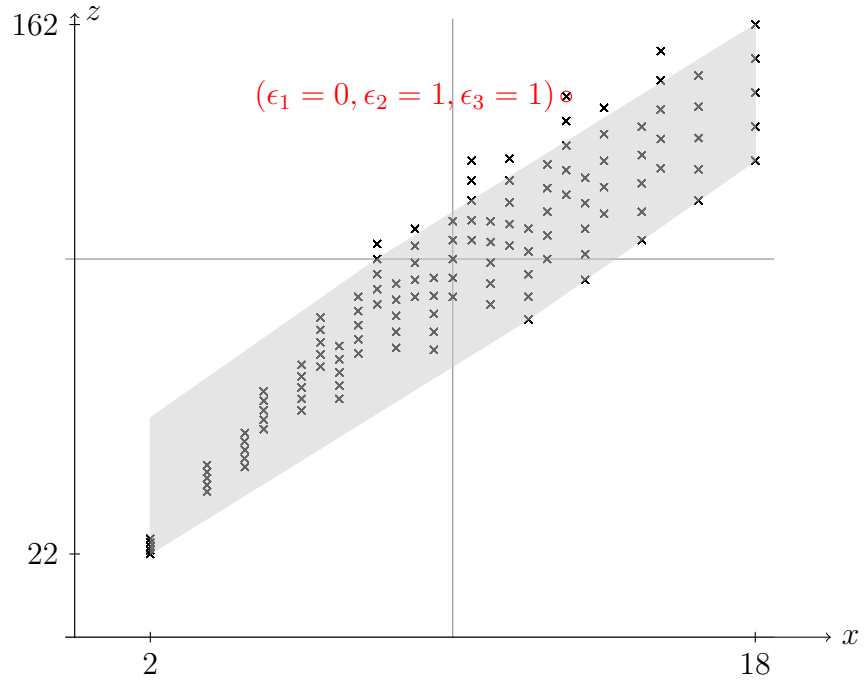

Figure 5.1: unsound multiplication: the concretisation of the abstraction does not over-approximate the concrete graph.

multiplication gives $\hat{z} = 92 + 31\epsilon_1 + 21\epsilon_2 + 2\epsilon_3 + 16\epsilon_4$. *The concretisation of* $\hat{z}$ *is* $[22, 162]$ *which is the exact range of* xy. *Consider now the (future) computation* $t = -4x + 0.8z - 79$. *Using the above* $\hat{z}$, *we find* $\hat{t} = -45.4 + 4.8\epsilon_1 + 4.8\epsilon_2 + 1.6\epsilon_3 + 12.8\epsilon_4$ *and conclude that* $t \in [-69.4, -21.4]$. *This is wrong: for* $\epsilon_1 = 0$ *and* $\epsilon_2 = 1$ *and* $\epsilon_3 = 1$, *we have* $x = 13$ *and* $y = 11$ *and* $z = 143$, *then* $t = -16.6$ *which is outside the concretisation of* $\hat{t}$ *found using this affine form for* z.

Figure 5.1 depicts the projection onto the (x, z) space of a cloud of vectors of the actual 3-dim graph $(x, y, z = xy)$ taken sparsely for $\epsilon_1, \epsilon_2, \epsilon_3 \in [-1 : 0.2 : 1]$, and the joint range (the gray zonotope) of the affine forms $\hat{x}$ and $\hat{z}$. Observe that some points are outside the zonotope even if their projection onto the z axes is within $[22, 162]$. For instance, the red point $(x, z) = (13, 143)$ obtained for $(\epsilon_1, \epsilon_2, \epsilon_3) = (0, 1, 1)$ is outside the zonotope and leads to the wrong over-approximation of t detailed above.

5.2 Interpretation of Tests

The scope of this section covers the novel idea behind interpretation of tests, informally discussed in the introduction (Section 4.1).

We study first (Section 5.2) the interpretation of the equality test $[\![e = 0]\!]^\sharp$ for an expression $e \in expr$. Then we formalize $[\![e \leq 0]\!]^\sharp$ as well as the interpretation of conjunction and disjunction of constraints (Section 5.2).

Equality Test

We start with a motivating example to help understanding the formal definition of the interpretation of an equality test over CAS.

5.2.1 Example
Let $\hat{X} = (C^X, P^X, \Phi^X)$ be a CAS abstracting three variables $\{v_1, v_2, v_3\}$, where $\mathcal{A}_2$ is the intervals lattice. Each variable v_i is abstracted by the affine form given by the ith line of matrices C^X and P^X.

$$\Phi^X := \{1\} \times [-1,1] \times [-1,1] \times [-1,1]$$
$$\hat{X}_1 := 4 + \epsilon_1 + \epsilon_2 + \eta_1, \qquad \text{bound}_2(\hat{X}_1, \Phi^X) = [1,7]$$
$$\hat{X}_2 := -\epsilon_1 + 3\epsilon_2, \qquad \text{bound}_2(\hat{X}_2, \Phi^X) = [-4,4]$$
$$\hat{X}_3 := -\epsilon_1 + 2\epsilon_2 + \eta_1, \qquad \text{bound}_2(\hat{X}_3, \Phi^X) = [-4,4]$$

The evaluation of expression $v_1 - v_2$ in our abstract domain gives

$$[\![v_1 - v_2]\!]^\sharp(C^X, P^X, \Phi^X) = (4 + 2\epsilon_1 - 2\epsilon_2 + \eta_1, \Phi),$$

Constraint $v_1 - v_2 = 0$, interpreted with affine forms, gives then

$$4 + 2\epsilon_1 - 2\epsilon_2 + \eta_1 = 0 \ . \tag{5.2.1}$$

This constraint, established using the noise symbols, is first used to enhance the interval concretisation of the noise symbols (in this example, domain $\mathcal{A}_2$ is the intervals lattice). This gives $\Phi^Y \stackrel{\text{def}}{=} \{1\} \times [-1,-0.5] \times [0.5,1] \times [-1,0]$. Moreover, for each variable v_i we inject equation (5.2.1) to seek the best trade-off one can have by substituting one of its noise symbols; "best" in the sense of minimizing the interval concretisation of the variable v_i. For instance for $\hat{X}_1 = 4 + \epsilon_1 + \epsilon_2 + \eta_1$, we have 3 choices

$$\hat{Y}_1 = 2 + 2\epsilon_2 + 0.5\eta_1, \quad \text{bound}_2(\hat{Y}_1, \Phi^Y) = [2.5,4] \quad \text{(by substituting } \epsilon_1\text{)}$$
$$\hat{Y}_1 = 6 + 2\epsilon_1 + 1.5\eta_1, \quad \text{bound}_2(\hat{Y}_1, \Phi^Y) = [2.5,5] \quad \text{(by substituting } \epsilon_2\text{)}$$
$$\hat{Y}_1 = -\epsilon_1 + 3\epsilon_2, \quad \text{bound}_2(\hat{Y}_1, \Phi^Y) = [2,4] \quad \text{(by substituting } \eta_1\text{)}$$

The best substitution, which minimizes the range of the variable is the one which substitutes ϵ_1 by its corresponding expression given by the equation (5.2.1).

Since variable v_2 is involved in constraint $v_1 = v_2$, its best affine form is the same one found for v_1. So there is no more computation needed here.

For the last variable v_3, we also choose the best possible substitution as we did with v_1, we have

$$\begin{aligned}
&\hat{Y}_3 = 2 + \epsilon_2 + 1.5\eta_1, && \text{bound}_2(\hat{Y}_3, \Phi^Y) = [1, 3] && \text{(by substituting } \epsilon_1\text{)}\\
&\hat{Y}_3 = 4 + \epsilon_1 + 2\eta_1, && \text{bound}_2(\hat{Y}_3, \Phi^Y) = [1, 3.5] && \text{(by substituting } \epsilon_2\text{)}\\
&\hat{Y}_3 = -4 - 3\epsilon_1 + 4\epsilon_2, && \text{bound}_2(\hat{Y}_3, \Phi^Y) = [-0.5, 3] && \text{(by substituting } \eta_1\text{)}
\end{aligned}$$

and it turns out that substituting ϵ_1 gives also the best affine form. The interval concretisation of this best form is tighter than $[0.5, 3]$, the one obtained with the original affine form of v_3, $-\epsilon_1 + 2\epsilon_2 + \eta_1$ with the new intervals of noise symbols Φ^Y. Of course, the choice of which noise symbol substitute depends on the original affine form of the variable as well as the constraint. For instance, if $\hat{X}_3$ was equal to $-\epsilon_1 + 0.5\epsilon_2 + \eta_1$, with respect to the same constraint given in (5.2.1), then ϵ_2 would be the best substitution.

Finally, the CAS $\hat{Y}$ obtained after the interpretation of the equality test is

$$\begin{aligned}
&\Phi^Y := \{1\} \times [-1, -0.5] \times [0.5, 1] \times [-1, 0]\\
&\hat{Y}_1 := 2 + 2\epsilon_2 + 0.5\eta_1, && \text{bound}_2(\hat{Y}_1, \Phi^Y) = [2.5, 4]\\
&\hat{Y}_2 := 2 + 2\epsilon_2 + 0.5\eta_1, && \text{bound}_2(\hat{Y}_2, \Phi^Y) = [2.5, 4]\\
&\hat{Y}_3 := 2 + \epsilon_2 + 1.5\eta_1, && \text{bound}_2(\hat{Y}_3, \Phi^Y) = [1, 3]
\end{aligned}$$

The so obtained interval concretizations are better than the ones obtained by the reduced product of affine sets and intervals, which gives, after the test, $[1, 7] \cap [-4, 4] = [1, 4]$ for $\hat{Y}_1$ and $\hat{Y}_2$, and $[-4, 4]$ for $\hat{Y}_3$. The substitution is injects in fact the exact constraint into the affine forms. Observe also that in the CAS $\hat{Y}$, the equality is algebraically satisfied as the affine forms $\hat{Y}_1$ and $\hat{Y}_2$ are equal.

The complexity of the straightforward method to compute the best substitution used in the example below (testing each noise symbol then comparing the concretisations) is $O((n+m)^2)$ for each numerical variable v_i, for $n + m$ noise symbols. We can reduce such complexity by transforming the problem to the following optimization problem:

$$\min_{\lambda \in \mathbb{R}} f(\lambda), \quad f(\lambda) \stackrel{\text{def}}{=} \sum_{i=1}^{n+m} |a_i - b_i\lambda| \qquad (*)$$

where a_i, b_i are known real numbers for all i, $1 \leq i \leq n+m$. We can suppose without loss of generality, that $b_i > 0$. Indeed if b_k is null for some k, then the term $|a_k|$ is not involved in the minimization problem as it is independent from λ. If $b_k < 0$, then we just multiply by -1.

5.2.2 Proposition
The best average complexity of solving the problem (*) is $O(n\log(n))$.

Proof. *The function f in (*) is convex: it is defined as a finite sum of convex functions, $|a_i - b_i\lambda|$. Let us denote by $R = \{r_1, \ldots, r_{n+m}\}$, the sorted set of roots of the functions $a_i - b_i\lambda$, $r_i \stackrel{\text{def}}{=} \frac{a_i}{b_i}$, in the increasing order (i.e. $r_i \leq r_{i+1}$) : Note that, the function f is a piecewise linear function with $n+m-1$ line segments, each defined over $[r_i, r_{i+1}]$, $1 \leq i < n+m$, and two half-lines (rays) for $\lambda \leq r_1$ and $r_{n+m} \leq \lambda$. By convexity of f, when λ varies over the real number line, the slopes of each line segment are ordered and vary from $-\sum_{i=1}^{n+m} b_i$ (the slope of the ray $\lambda \leq r_1$) to $\sum_{i=1}^{n+m} b_i$ (the slope of the ray $r_{n+m} \leq \lambda$). By hypothesis, $b_i > 0$, then there exists necessarily at least one point r_p, such that the slope of the line segment $[r_{p-1}, r_p]$ is non-positive (negative or null) and the slope of the line segment $[r_p, r_{p+1}]$ is positive. Then, a minimum of f is reached at r_p since f is decreasing before r_p and increasing after r_p. Such a local minimum is global by the convexity of f. Moreover, if the sign of the slopes in $[r_{p-1}, r_p]$ is null, then the global minimum is reached for all $\lambda \in [r_{p-1}, r_p]$.*

The best average complexity of sorting a list of $n+m$ elements is $O((n+m)log(n+m))$, using a divide and conquer strategy (Quicksort algorithm by Hoare, Merge algorithm by Von Neuman, etc.). We have then, in the worst case, to run through $n+m$ elements seeking the change of the sign of the slopes of the line segments. ∎

Our original problem of seeking the best substitution, can be easily translated to the form of (*). We exemplify this translation for variable $\hat{X}_1$, then formalize the general case.

$$\begin{aligned}\hat{X}_1 &= \hat{X}_1 + \lambda \times 0 \\ &= 4 + \epsilon_1 + \epsilon_2 + \eta_1 + \lambda(4 + 2\epsilon_1 - 2\epsilon_2 + \eta_1) \\ &= 4 + 4\lambda + (1+2\lambda)\epsilon_1 + (1-2\lambda)\epsilon_2 + (1+\lambda)\eta_1\end{aligned}$$

In the second equality, the zero was replaced by the constraint (5.2.1) deduced from the test. The deviation of the interval concretisation $\gamma(\hat{X}_1)$ is then:

$$0.25|(1+2\lambda)| + 0.25|1-2\lambda| + 0.5|1+\lambda| \ .$$

Minimizing this deviation ensures the minimal interval concretisation for the variable v_1.

Now, for any affine form $\hat{a} = \alpha_0 + \sum_{i=1}^{n} \alpha_i \epsilon_i + \sum_{i=j}^{m} \beta_j \eta_j$ and a constraint $\hat{c} = c_0 + \sum_{i=1}^{n} c_i \epsilon_i + \sum_{j=1}^{m} p_j \eta_j = 0$, the optimization problem to solve is

$$\min_{\lambda \in \mathbb{R}} \sum_{i=1}^{n} \mathrm{dev}(\boldsymbol{\epsilon_i}) |\alpha_i + \lambda c_i| + \sum_{j=1}^{m} \mathrm{dev}(\boldsymbol{\eta_j}) |\beta_j + \lambda p_j|, \qquad (**)$$

where $\boldsymbol{\epsilon_i}$ and $\boldsymbol{\eta_j}$ denote respectively the interval concretisations of the input noise symbols ϵ_i, $1 \leq i \leq n$, and the perturbation noise symbols η_j, $1 \leq j \leq m$, that is

$$\begin{aligned} \boldsymbol{\epsilon_i} &\stackrel{\text{def}}{=} \mathrm{bound}_2(\epsilon_i, \Phi^Y), \\ \boldsymbol{\eta_j} &\stackrel{\text{def}}{=} \mathrm{bound}_2(\eta_j, \Phi^Y) \ . \end{aligned}$$

The new affine form is derived from the objective solution of (**), $\bar{\lambda}$, and the current affine form.

The abstract operator $[\![v_k = 0]\!]^\sharp$ is formalized as follows:

5.2.3 Definition
Let $\hat{X}$ be a CAS abstracting p numerical variables. Then $\hat{Y} = [\![v_k = 0]\!]^\sharp \hat{X}$ is defined by

$$\begin{aligned} \Phi^Y &\stackrel{\text{def}}{=} [\![\hat{X}_k = 0]\!]_2^\sharp \Phi^X \\ \hat{Y}_i &\stackrel{\text{def}}{=} \hat{X}_i + \bar{\lambda}_i(\hat{X}_k), 1 \leq i \leq p, i \neq k \\ \hat{Y}_k &\stackrel{\text{def}}{=} 0 \end{aligned}$$

where $\bar{\lambda}_i$ is the optimal solution of the problem (**) solved for

$$\begin{aligned} \hat{a} &= \hat{X}_i, & \hat{c} &= \hat{X}_k \\ \boldsymbol{\epsilon_i} &= \mathrm{bound}_2(\epsilon_i, \Phi^Y), & \boldsymbol{\eta_j} &= \mathrm{bound}_2(\eta_j, \Phi^Y). \end{aligned}$$

The matrices C^Y and P^Y are computed (together) line by line: the central part of the affine form $\hat{Y}_i$ gives the coefficient of the ith line of C^Y and its perturbation part completes the ith line of matrix P^Y. The operator $[\![e = 0]\!]_2^\sharp$ denotes the abstract conditional operator of the noise symbol abstract domain $\mathcal{A}_2$. We recall that the primitive $\mathrm{bound}_2 : \mathcal{A}_1 \times \mathcal{A}_2 \to \mathbb{I}$ returns the range of an affine form with respect to an abstract object in $\mathcal{A}_2$.

5.2.4 Proposition
The average complexity of the abstract function $[\![v_k = 0]\!]^\sharp$ *is* $O((p-1)(n+m)log(n+m))$ *for* p *numerical variable,* n *input noise symbols and* m *perturbation noise symbols.*

Proof. *We solve* $p-1$ *times the problem* (**) *which has an average complexity of* $O((n+m)log(n+m))$ *by Proposition 5.2.2. We subtract* 1 *from* p *since the affine form of* v_k *is set to* 0. ∎

Interpretation of expressions equality If the test involves an expression rather than a variable, then the definition of the operator $[\![e = 0]\!]^\sharp$ has the same definition of $[\![v_k = 0]\!]^\sharp$ up to the evaluation of the expression e.

5.2.5 Definition
The abstraction of the test $e = 0$ *is defined by:*

$$[\![e = 0]\!]^\sharp \hat{X} \stackrel{\text{def}}{=} [\![v' = 0]\!]^\sharp \hat{X},$$

where v' *is a temporary numerical variable abstracted by* $[\![e]\!]^\sharp \hat{X}$.

5.2.6 Example
Consider $\hat{Y} = [\![x_1 + x_2 - x_3 = 0]\!]^\sharp \hat{X}$ *where*

$$\begin{aligned}
&\Phi^X := \{1\} \times [-1,1] \times [-1,1] \times [-1,1] \\
&\hat{X}_1 := 2 + \epsilon_1, && \text{bound}_2(\hat{X}_1, \Phi^X) = [1,3] \\
&\hat{X}_2 := 2 + \epsilon_2 + \eta_1, && \text{bound}_2(\hat{X}_2, \Phi^X) = [0,4] \\
&\hat{X}_3 := -\epsilon_1 + 3\epsilon_2, && \text{bound}_2(\hat{X}_3, \Phi^X) = [-4,4]
\end{aligned}$$

The evaluation of the expression $x_1 + x_2 - x_3$ *gives the same constraint of the example 5.2.1, that is*

$$4 + 2\epsilon_1 - 2\epsilon_2 + \eta_1 = 0,$$

thus, $\Phi^Y = 1 \times [-1,-0.5] \times [0.5,1] \times [-1,0]$. *The computation of the best substitutions replaces* ϵ_1 *by* $-2 + \epsilon_2 - 0.5\eta_1$ *in all affine forms:*

$$\begin{aligned}
&\hat{Y}_1 := \epsilon_2 - 0.5\eta_1, && \text{bound}_2(\hat{Y}_1, \Phi^Y) = [0.5, 1.5] \\
&\hat{Y}_2 := 2 + \epsilon_2 + \eta_1, && \text{bound}_2(\hat{Y}_2) = [1.5, 3] \\
&\hat{Y}_3 := 2 + 2\epsilon_2 + 0.5\eta_1, && \text{bound}_2(\hat{Y}_3) = [2.5, 4] \ .
\end{aligned}$$

Observe that, after the test, $\hat{Y}_1 + \hat{Y}_2 = \hat{Y}_3$.

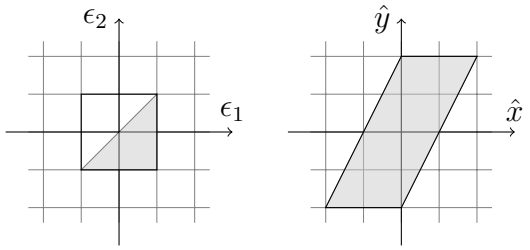

Figure 5.2: The constraints over variables are interpreted as constraints over the noise symbols

Inequality Tests

The inequality test supported by our language is $e \leq 0$. The test is propagated to the noise symbols abstract domain.

5.2.7 Definition
Let $\hat{X}$ be a CAS abstracting p numerical variables. Then $\hat{Y} = [\![x_k \leq 0]\!]^\sharp \hat{X}$ is defined by

$$\begin{aligned} \Phi^Y &\stackrel{\text{def}}{=} [\![e \leq 0]\!]_2^\sharp \Phi^X \\ C^Y &\stackrel{\text{def}}{=} C^X \\ P^Y &\stackrel{\text{def}}{=} P^X \quad . \end{aligned}$$

Only the abstract domain of the noise symbols is updated in the inequality test. The affine set (C^X, P^X) remains unchanged. For instance, consider the affine set

$$\begin{aligned} \hat{x} &= \epsilon_1 - \epsilon_2 \\ \hat{y} &= 2\epsilon_1 \quad . \end{aligned}$$

In figure 5.2, we depict the final object which we propagate after the test $\hat{x} \geq 0$. On one hand, the initial affine set given above is untouched, and so is its geometrical concretisation, that is the gray zonotope (right hand side). On the other hand, the values of noise symbols are constrained to the gray area (left hand side), instead of being independent within $[-1, 1]$ each.

The conjunction and disjunction of constraints are interpreted similarly, that is as constraints over the noise symbols. If the underlying domain of noise symbols does not handle disjunctions, the convex hull is considered instead, as it is done in classical convex abstract domain.

5.3 Related Work

On the Use of Zonotopes

The efficient encoding of affine forms (as lists of generators) and the accuracy of computations (for both linear and non-linear operations) have motivated other applications of these special polytopes or zonotopes. For instance, Girard in [Gir05] and Combastel [Com05] have used zonotopes for the computation of reachable sets of uncertain linear systems. Combastel [Com05] has also proposed rigorous bounds for uncertain non-linear continuous-time systems using zonotopes.

Years before, Kühn [Küh98] has used zonotopes for the purpose of numerical quality control: he used zonotopes to enclose the orbits of discrete dynamical systems; the higher order zonotopes permit to reduce the wrapping effect and hence lead to more accurate results. Zonotopes were also used as bounding volumes for collision detection [GNZ03].

Zonotope/Hyperplane Intersection

The use of zonotopes in reachability analysis of hybrid systems needs to first detect the collision of a zonotope with guards that govern the discrete transitions of the system, and second to be able to wrap the intersection with the active guard by a zonotope. Indeed, the intersection of a hyperplane (for linear guards) and zonotope is in general not a zonotope.

As seen in the previous section, the interpretation of tests while using zonotopes (or equivalently affine sets) as abstract objects rose a similar problem as one needs to compute a zonotopic approximation of the intersection which is in general a polytope.

The geometrical approaches proposed in [GLG08, LG09] and [ASB08] do not embed the noise symbols with a particular semantics. The zonotopes are encoded with unordered list of generators. In our case, the noise symbols are related to the inputs of the program, these symbols have a precise meaning and can not be substituted. Indeed, the order we define over our abstract objects is strictly stronger than the geometrical order. Therefore, a zonotope that over-approximates geometrically the intersection may not be sound in our context.

CHAPTER 6

Join over Constrained Affine Sets

This chapter is dedicated to the computation of the union of two constrained affine sets, as an optimal (in a sense to define) upper bound of two given constrained affine sets, with respect to the partial order $\leq_{1\times 2}$.

Contents Firstly, we define, in section 6.1 the general procedure we use to build a sound upper bound of two given CAS. The procedure is generic and not related to the noise symbols abstract domain $\mathcal{A}_2$. Section 6.2 characterizes the set of minimal upper bounds of two constrained affine forms (and not sets); Section 6.2 presents an algorithm to pick up one of these minimal upper bounds earlier characterized. This algorithm is extended in Section 16 to handle reduced intervals case, that is when the noise symbols are constants, or equivalently, within intervals of the form $[c, c]$, where c is a real number. Sections 16 and 16 are special cases of our algorithm: firstly, we consider the case of perturbed affine forms; secondly, we apply the efficient join operator defined for perturbed affine forms to the constrained affine forms using our characterization of minimal upper bounds, and hence our algorithm, to compute the minimal perturbation. Finally, the last part (Section 6.3) defines, piecewisely, an upper bound over constrained affine sets.

6.1 General Procedure

As we are seeking a good precision-time cost trade-off, our general procedure starts by relaxing the given abstract objects by forgetting all noise symbols relations, then computes an upper bound of the relaxed constrained affine sets. Here, relaxing means taking the smallest box containing the concretisation of Φ instead of Φ itself. Our approach is sound, since relaxing a CAS (C, P, Φ) gives an over-approximation of (C, P, Φ).

6.1.1 Proposition
Let $\hat{X} = (C, P, \Phi)$ be a CAS, then

$$(C, P, \Phi) = \hat{X} \leq_{1\times 2} \Box\hat{X} \stackrel{\text{def}}{=} (C, P, \Box\Phi)$$

where $\Box\Phi$ denotes the interval concretisation of Φ, that is, $\Pi_{i=0}^{n}\boldsymbol{\epsilon_i} \times \Pi_{j=1}^{m}\boldsymbol{\eta_j}$, where $\boldsymbol{\epsilon_i} = \text{bound}_2(\epsilon_i, \Phi)$ and $\boldsymbol{\eta_j} = \text{bound}_2(\eta_j, \Phi)$.

Proof. *Obviously the condition $\Phi_\epsilon \subseteq \Box\Phi_\epsilon$ holds as Φ_ϵ denotes the concretisation of the projection of Φ over the subspace defined by the input noise symbols, and $\Box\Phi_\epsilon$ is the smallest box containing this concretisation by construction. Similarly, we have $\Phi_\eta \subseteq \Box\Phi_\eta$. Moreover, for all $\omega \in \Phi_\epsilon$,*

$$(C - C)\omega + P\Phi_\eta(\omega) = P\Phi_\eta(\omega) \subseteq P\Box\Phi_\eta(\omega),$$

which ends the proof. ∎

The general procedure used to compute the join of two CAS $\hat{X}$ and $\hat{Y}$ starts by computing an upper bound of $\Box\hat{X}$ and $\Box\hat{Y}$, which is also an upper bound of $\hat{X}$ and $\hat{Y}$ thanks to Proposition 6.1.1. Therefore, computing the join of two CAS is $\mathcal{A}_2$-independent, since the problem is always brought back to the computation of an upper bound of CAS where the noise symbols range over intervals.

In the remaining sections, $\mathcal{A}_2$ is the intervals lattice. We characterize and compute upper bounds of two CAS with respect to the partial order $\leq_{1\times 2}$. Our computation is defined componentwisely over the variables' set. The result is constructed, line by line, by computing the minimal upper bound (mub) of two Constrained Affine Forms (CAF), which are exactly the CAF. The computation of such mub extends and generalizes the one of [GP08] which computes the mub of two perturbed (but unconstrained) affine forms.

It is important to notice that such approach does not consider the CAS globally while computing the join, it rather focuses on a fixed set of directions, namely those of the canonical base of $\mathbb{R}^p$. This means that the perturbation for each variable, taken alone, is optimized. However, the perturbation along any other random direction is not minimal in general.

This chapter is organized as follows. Section 6.2 details the way we compute the minimal upper bound of two constrained affine forms. In Section 6.3 we define our join operators.

6.2 Join over Constrained Affine Forms

In this section, the number of variables p is equal to one. Thus, matrices C^X and P^X for a CAS $\hat{X}$ are simply two lines; $\hat{X}$ is simply a constrained affine form, or CAF.

A least upper bound (lub) does not exist in general, this fact was established for perturbed affine forms in [GP08], which are a special CAF (the noise symbols are unconstrained). Instead, two given CAF may have infinitely many minimal upper bounds (mub). We first remind the definition of a mub. Then, we focus on a particular subset of mubs (the ones which minimize the perturbation) that we can characterize as the set of saddle-points of a function $L(\alpha, \lambda)$ defined over $\mathbb{R}^{n+1} \times [0, 1]$. Finally, we solve the saddle-point problem using standard tools from the subdifferential theory of convex functions.

As we have seen in Section 4.3, when using intervals to abstract noise symbols, the partial order $\leq_{1\times 2}$ is not sensitive to the domain of each perturbation symbol η_i, only the (symmetric) perturbation set considered globally is of interest. We use the symmetric representative (see Definition 4.3.4) of CAF. Therefore, only the box Φ_ϵ^X should be considered. One can then represent, without loss of generality, a CAF $\hat{X}$ by $(\alpha^X, \tau^X, \Phi_\epsilon^X)$, where $\alpha^X \in \mathbb{R}^{n+1}$, τ^X is a non-negative real number, deviation of the perturbation interval, and Φ_ϵ^X, which is an hypercube of dimension n, domain of the input noise symbols.

6.2.1 Definition
Let $\hat{X} = (\alpha^X, \tau^X, \Phi_\epsilon^X)$ and $\hat{Y} = (\alpha^Y, \tau^Y, \Phi_\epsilon^Y)$ be two CAF. We say that $\hat{Z} = (\alpha^Z, \tau^Z, \Phi_\epsilon^Z)$ is a minimal upper bound (mub) of $\hat{X}$ and $\hat{Y}$ if and only if

- *$\hat{Z}$ is an upper bound of $\hat{X}$ and $\hat{Y}$, that is $\hat{X} \leq_{1\times 2} \hat{Z}$ and $\hat{Y} \leq_{1\times 2} \hat{Z}$, and*
- *for all $\hat{W}$ upper bound of $\hat{X}$ and $\hat{Y}$, $\hat{W} \leq_{1\times 2} \hat{Z}$ implies $\hat{W} = \hat{Z}$.*

The proposition below establishes that if the deviation of the perturbation set of any upper bound is minimal, then this upper bound is a mub.

6.2.2 Proposition
If $\hat{Z} = (\bar{\alpha}, \bar{\tau}, \Phi_\epsilon^X \cup_2 \Phi_\epsilon^Y)$ is an upper bound of two given CAF, $\hat{X}$ and $\hat{Y}$, such that $\bar{\tau}$ (the deviation of its perturbation set) is minimal over all deviations τ of any upper bound of $\hat{X}$ and $\hat{Y}$, then $\hat{Z}$ is a mub of $\hat{X}$ and $\hat{Y}$.

Proof. *Suppose that $\hat{T} = (\alpha, \tau, \Phi_\epsilon)$ is an upper bound of $\hat{X}$ and $\hat{Y}$ such that $\hat{T} \leq_{1\times 2} \hat{Z}$, then by definition of the order, $(\imath)$ $\Phi_\epsilon \subseteq \Phi_\epsilon^X \cup \Phi_\epsilon^Y$, and $(\imath\imath)$:*

$$\delta(M(\alpha - \bar{\alpha}) \mid \mathfrak{B}) \leq \bar{\tau} - \tau \ .$$

where M is the matrix related to Φ_ϵ. Since $\hat{T}$ is an upper bound of $\hat{X}$ and $\hat{Y}$, we have $\Phi_\epsilon^X \subseteq \Phi_\epsilon$, and $\Phi_\epsilon^Y \subseteq \Phi_\epsilon$, thus $\Phi_\epsilon^X \cup \Phi_\epsilon^Y \subseteq \Phi_\epsilon$. By $(\imath)$ we obtain $\Phi_\epsilon = \Phi^X \cup \Phi^Y$. By hypothesis, $\bar{\tau}$ is minimal, therefore, $\bar{\tau} - \tau \leq 0$. However, by definition of the support function and $(\imath\imath)$, we have $\delta(M(\alpha - \bar{\alpha}) \mid \mathfrak{B}) = \|M(\alpha - \bar{\alpha})\|_1 \geq 0$. Therefore, $\|M(\alpha - \bar{\alpha})\|_1 = 0$, and $\alpha = \bar{\alpha}$. Finally $0 \leq \bar{\tau} - \tau$, and $\bar{\tau} = \tau$. ∎

Notice that the mubs that minimize the perturbation are not the only possible mubs. Indeed, Proposition 6.2.2 is only sufficient but not necessary. Example 6.2.3 gives a counter example of a minimal upper bound which does not have the minimal perturbation.

6.2.3 Example
Let $\hat{X} = ((1, -1, 2), 0, 1 \times [-1, 0] \times [0, 0.5])$, and $\hat{Y} = ((2, 1, 1), 0, 1 \times [-0.5, 0.5] \times [0, 1])$, then $\hat{T} = ((1.75, 0, 0.75), 0.75, 1 \times [-1, 0.5] \times [0, 1])$ is a mub that minimizes the interval concretisation, indeed $[\hat{T}] = [1, 3.5] = [1, 3] \cup [1.5, 3.5] = [\hat{X}] \cup [\hat{Y}]$. However, in Section 6.2, we have seen that $\hat{Z} = \hat{X} \sqcup \hat{Y} = ((1.7, 0.2, 1.6), 0.7, 1 \times [-1, 0.5] \times [0, 1])$. Observe that $[\hat{Z}] = [0.8, 4.1] \supseteq [1, 3.5] = [\hat{T}]$. Of course, $\hat{T}$ and $\hat{Z}$ are not comparable.

What Example 6.2.3 suggests is that enforcing the minimality of the concretization, then seeking the minimal perturbation among these upper bound with minimal concretization, may lead to a different subset of mubs.

We focus in the sequel on the computation of the subset of mubs $(\bar{\alpha}, \bar{\tau}, \Phi)$ such that the perturbation $\bar{\tau}$ is minimal.

By Proposition 4.3.7, for $\hat{T} = (\alpha, \tau, \Phi_\epsilon)$ to be an upper bound of $\hat{X}$ and $\hat{Y}$, it is necessary and sufficient that $\Phi_\epsilon^X \cup_2 \Phi_\epsilon^Y \leq_2 \Phi_\epsilon$ and that α and τ satisfy:

$$\delta(M^X(\alpha^X - \alpha) \mid \mathfrak{B}) + \tau^X \leq \tau,$$
$$\delta(M^Y(\alpha^Y - \alpha) \mid \mathfrak{B}) + \tau^Y \leq \tau,$$

or said differently:

$$\max\{\delta(M^X(\alpha^X - \alpha) \mid \mathfrak{B}) + \tau^X, \delta(M^Y(\alpha^Y - \alpha) \mid \mathfrak{B}) + \tau^Y\} \leq \tau \ .$$

We look for the set of $\alpha \in \mathbb{R}^{n+1}$ that minimizes the maximum above. If such set is not empty, by Proposition 6.2.2, it should contain all mubs of $\hat{X}$ and $\hat{Y}$. Formally, we define $\bar{\alpha}$ and $\bar{\tau}$ as being respectively the objective vector and the objective value of the following minimax problem

$$\bar{\tau} = \min_{\alpha \in \mathbb{R}^{n+1}} \max\{\delta(M^X(\alpha^X - \alpha) \mid \mathfrak{B}) + \tau^X, \delta(M^Y(\alpha^Y - \alpha) \mid \mathfrak{B}) + \tau^Y\} \ .$$

We can rewrite the maximum of two real numbers as a maximum of a linear real-valued function, using the lemma below:

6.2.4 Lemma
Let a and b be two real numbers. Then

$$\max\{a, b\} = \max_{\lambda \in [0,1]} \lambda a + (1 - \lambda)b \ .$$

Proof. *If $a \leq b$, then $\max\{a, b\} = b$. On the other hand $f(\lambda) \stackrel{\text{def}}{=} \lambda a + (1 - \lambda)b = \lambda(a - b) + b$ is an affine function with a negative slope $(a - b)$, thus it reaches its maximum for $\lambda = 0$, that is, $\max_{\lambda \in [0,1]} \lambda a + (1 - \lambda)b = b$. The case $b \leq a$ is similar (just interchange a and b).* ∎

6.2.5 Definition (Minimal perturbation of two CAF)
The minimal perturbation $\bar{\tau}$ is the objective value of the following problem

$$\bar{\tau} = \inf_{\alpha \in \mathbb{R}^{n+1}} \sup_{\lambda \in [0,1]} L(\alpha, \lambda) \ .$$

where $L : \mathbb{R}^{n+1} \times [0, 1] \to \mathbb{R}$ maps (α, λ) to

$$\lambda(\delta(M^X(\alpha^X - \alpha) \mid \mathfrak{B}) + \tau^X) + (1 - \lambda)(\delta(M^Y(\alpha^Y - \alpha) \mid \mathfrak{B}) + \tau^Y) \ . \quad (6.2.1)$$

Matrix M^X is a square matrix of dimension $(n+1)^2$, its determinant is equal to

$$\Pi_{i=1}^{n} \operatorname{dev}(\boldsymbol{\epsilon}_j^{\boldsymbol{X}}) \ .$$

If all the intervals $\boldsymbol{\epsilon}_j^{\boldsymbol{X}}$ are non-reduced to a point, the matrix is non-singular. For the sake of readability, the special case where these matrices are singular, that is where some noise symbols' intervals are reduced to points, is left to a separate section (see Section 16). From now on we suppose that matrices M^X and M^Y are non-singular.

The rest of this section details our approach to solve efficiently the minimax problem of Definition 6.2.5. We characterize the set of solutions (Section 6.2), then solve the system of equations found (Section 6.2). Through these different sections, each step is exemplified using the same following example:

6.2.6 Example
We would like to compute $(\bar{\alpha}, \bar{\tau})$ *such that* $\hat{Z} = (\bar{\alpha}, \bar{\tau}, 1 \times [-1, 0.5] \times [0, 1])$ *is a mub of* $\hat{X}$ *and* $\hat{Y}$*, defined by*

$$\hat{X} = ((1, -1, 2), 0, 1 \times [-1, 0] \times [0, 0.5])$$
$$\hat{Y} = ((2, 1, 1), 0, 1 \times [-0.5, 0.5] \times [0, 1]) \ .$$

The matrices M^X *and* M^Y*, related to* Φ_ϵ^X *and* Φ_ϵ^Y *respectively are*

$$M^X \stackrel{\text{def}}{=} \begin{pmatrix} 1 & -0.5 & 0.25 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.25 \end{pmatrix}, \quad M^Y \stackrel{\text{def}}{=} \begin{pmatrix} 1 & 0 & 0.5 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{pmatrix} \ .$$

Characterization of the Set of Mubs with Minimal Perturbation

The set of points $(\bar{\alpha}, \bar{\lambda})$, solution of the minimax problem of Definition 6.2.5 are known as the *saddle-points* of the function L defined over $\mathbb{R}^{n+1} \times [0, 1]$.

6.2.7 Definition (Saddle-point)
Let L *be a convex-concave function from* $C \times D$ *to* $[-\infty, +\infty]$*. A point* $(\bar{u}, \bar{v})$ *is a saddle-point of* L *with respect to minimizing over* C *and maximizing over* D *if* $(\bar{u}, \bar{v}) \in C \times D$ *and*

$$\forall u \in C, \quad \forall v \in D, \quad L(\bar{u}, v) \leq L(\bar{u}, \bar{v}) \leq L(u, \bar{v}) \ .$$

When we fix v to $\bar{v}$, the convex function L, seen as a function of u, achieves its minimum at $u = \bar{u}$. Likewise, when we fix u to $\bar{u}$, the concave function L, seen as function of v, achieves its maximum at $v = \bar{v}$ (see Figure 6.1 [wik]).

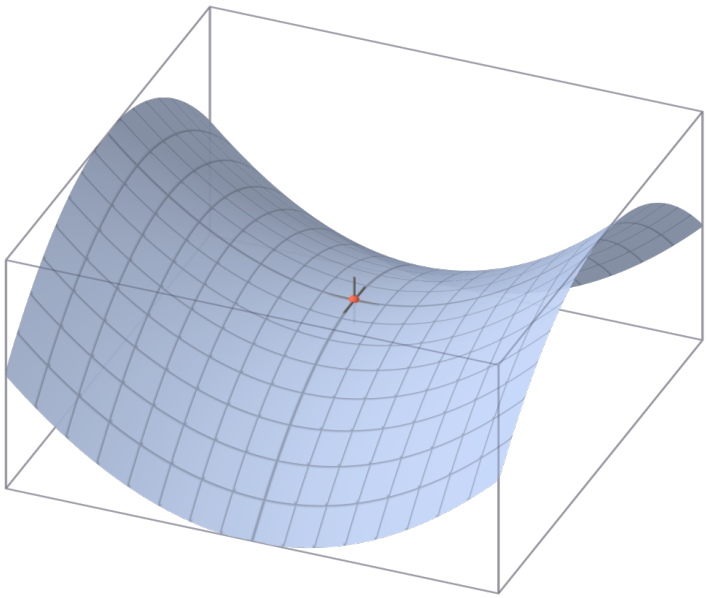

Figure 6.1: Function $f(x, y) = x^2 - y^2$, and its saddle-point $(0, 0)$, depicted in red.

The lemma below shows that, whenever a saddle-point exists, one has

$$\sup_{v \in D} \inf_{u \in C} L(u, v) = \inf_{u \in C} \sup_{v \in D} L(u, v) = L(\bar{u}, \bar{v}) \ .$$

The value $L(\bar{u}, \bar{v})$ is called the *saddle-value* of L.

6.2.8 Lemma
Let L be any convex-concave function from a non-empty product set $C \times D$ to $[-\infty, +\infty]$. A point $(\bar{u}, \bar{v})$ is a saddle-point of L with respect to minimizing over C and maximizing over D if and only if the supremum of the expression

$$\inf_{u \in C} L(u, v),$$

is reached at $\bar{v}$, the infimum in the expression

$$\sup_{v \in D} L(u, v),$$

is reached at $\bar{u}$, and these two extremes are equal. If $(\bar{u}, \bar{v})$ is a saddle-point, the saddle-value of L is by definition $L(\bar{u}, \bar{v})$.

One can start by fixing v, then computing the infimum of L (as function of u), and finally maximizing this infimum by varying v; or fixing u, then

computing the supremum of L (as function of v), and finally minimizing this supremum. Both cases need a characterization of the set of points that optimize (minimize or maximize) the function L, either with respect to u or v.

The differential theory offers a suitable toolset for characterizing such set of optimum points, whenever the objective function is a differentiable function (a function whose derivative exists at each point in its domain). In our case, the function L of equation (6.2.1) is not differentiable in the usual sense with respect to α. Indeed the function can be seen as a sum of absolute value functions, which are not differentiable in 0.

Instead, we use a weaker notion of differentiability, called subdifferential theory, which requires only the convexity of the function. We first remind the definition of a *subgradient*, then the *subdifferential* of a convex function from $\mathbb{R}^n$ to $\mathbb{R}$ at a point x of its domain.

6.2.9 Definition (Subgradient)
A vector t is said to be a subgradient of a convex function f at a point x if

$$\forall z, \quad f(z) \geq f(x) + \langle t, z - x \rangle \ .$$

If the function f is differentiable at x, then its subgradient is exactly its gradient, that is the vector whose components are the partial derivatives of the function f, usually denoted by ∇f:

$$\nabla f \stackrel{\text{def}}{=} (\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n})$$

evaluated at x.

The intuitive geometrical meaning of the *subgradient inequality* of Definition 6.2.9 for a convex function $f : \mathbb{R}^n \to \mathbb{R}$ at x, is the fact that the graph of the affine function $h(z) = f(x) + \langle t, z - x \rangle$ is a non-vertical supporting hyperplane to the epigraph (reminded hereafter) epi f at the point $(x, f(x))$.

6.2.10 Definition (Epigraph)
Let f be a function whose values are real or $\pm\infty$ and whose domain is a subset S of $\mathbb{R}^p$. The set

$$\{(x, \mu) \mid x \in S, \mu \in \mathbb{R}, \mu \geq f(x)\}$$

is called the epigraph of f and is denoted by epi f.

In Figure 6.2, we depict the epigraph of the function $f : \mathbb{R} \to \mathbb{R}$, where $f(x) = x^2$, which is a convex set of dimension 2 (the gray area). Since the

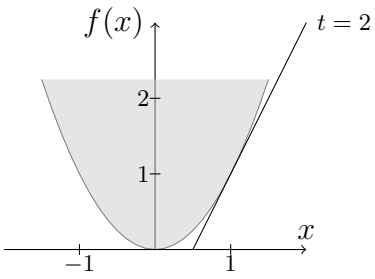

Figure 6.2: epigraph of $f(x) = x^2$ and the unique subgradient at $x = 1$.

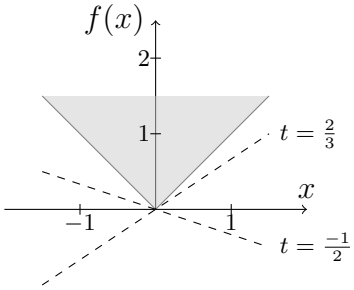

Figure 6.3: subgradients of $f(x) = |x|$ at $x = 0$.

function is differentiable, the hyperplane $h(z)$ is the tangent (hyperplanes of dimension 2 are lines) to the graph of the function at a point x; ∇f at x gives the slope of that tangent at x. Here, this slope is the subgradient t mentioned in Definition 6.2.9.

In Figure 6.3, the absolute value function is not differentiable at 0, however it has infinitely many subgradients at $x = 0$ (for instance $t = \frac{2}{3}$ and $t = \frac{-1}{2}$). The gray area shows the epigraph of the absolute value function. Observe that if t is outside $[-1, 1]$, the hyperplane does no more support the epigraph.

Since many subgradients may exist at a given point x, the set of these subgradients is called the *subdifferential* of f at x.

6.2.11 Definition (Subdifferential)
The set of all subgradients of f at x is called the subdifferential of f at x and is denoted by $\partial f(x)$. If the set $\partial f(x)$ is not empty, the function f is said to be subdifferentiable at x.

For instance, for the absolute value function, the subdifferential of f at 0 is the interval $[-1, 1]$; whereas, the subdifferential elsewhere is the singleton $\{1\}$ if x is non-negative and $\{-1\}$ if x is non-positive.

With respect to Definition 6.2.9 of a subgradient, given a convex function $f : \mathbb{R}^n \to \mathbb{R}$, if $0 \in \partial f(x)$, the subdifferential of f at x, then

$$\forall z \in \mathbb{R}^n, \quad f(z) \geq f(x),$$

thus, x is a global minimum of the function f. Reciprocally, if x is a global minimum of the function, by definition, the inequality above holds, which makes $0 \in \partial f(x)$.

Dually, if f is a concave function, then $-f : x \mapsto -f(x)$, is convex. Therefore, if $0 \in \partial(-f)(x)$, then

$$\forall z \in \mathbb{R}^n, \quad -f(z) \geq -f(x),$$

which makes x a global maximum of the function.

For our function L defined in equation (6.2.1), we first prove the existence of saddle-points.

6.2.12 Proposition
The convex-concave function L defined in equation (6.2.1) has a saddle-point.

Proof. *The proof is a direct application of [Roc70, Chapter 37, Theorem 37.6], which states that if* $\imath$*) the functions $-L_\alpha(\lambda) : \lambda \in \mathbb{R}^n \mapsto -L(\alpha, \lambda)$, defined over $]0,1[$ have no common* direction of recession*, and* $\imath\imath$*) the functions $L_\lambda(\alpha) : \alpha \in \mathbb{R}^n \mapsto L(\alpha, \lambda)$, defined over $\mathbb{R}^n$ have also no common direction of recession, then $L(\alpha, \lambda)$ has a saddle-point. The set of directions of recession of a convex function f, is defined by*

$$\{y \mid y \neq 0, \forall \lambda \geq 0, \forall x \text{ such that } f^+(x) \leq 0, f^+(x + \lambda y) \leq 0\},$$

where f^+ denotes the recession function *of f and can be defined by $f^+ \stackrel{\text{def}}{=} \lim_{\theta \to 0} \theta f(\theta^{-1} x)$ [1]. Since for all $\alpha \in \mathbb{R}^n$, the domain of $-L_\alpha(\lambda)$ is bounded (]0,1[), then condition ($\imath$) is fulfilled. We have*

$$\lim_{\theta \to 0} \theta L_\lambda^+(\theta^{-1}\alpha) = \lambda \delta(M^X(\alpha^X - \alpha) \mid \mathfrak{B}) + (1 - \lambda)\delta(M^Y(\alpha^Y - \alpha)),$$

then, for all $\lambda \in]0,1[$, the set of α such that $L_\lambda^+(\alpha) \leq 0$ is $\{0\}$, and ($\imath\imath$) is also satisfied. ∎

Let $(\bar{\alpha}, \bar{\lambda})$ denote a saddle-point of L. Then, by definition of saddle-points,

$$\forall \alpha \in \mathbb{R}^{n+1}, \quad \forall \lambda \in [0,1], \quad L(\bar{\alpha}, \lambda) \leq L(\bar{\alpha}, \bar{\lambda}) \leq L(\alpha, \bar{\lambda}),$$

which makes $\bar{\lambda}$ a global maximum of the (linear) function

$$L_{\bar{\alpha}}(\lambda) : \lambda \mapsto L(\bar{\alpha}, \lambda), \tag{$L_{\bar{\alpha}}$}$$

[1] This formula holds for our case because the hypothesis $0 \in \operatorname{dom} f$ is satisfied for both functions $-L_\alpha(\lambda)$ and $L(\alpha, \lambda)$.

and $\bar{\alpha}$ a global minimum of the non-linear convex function

$$L_{\bar{\lambda}}(\alpha) : \alpha \mapsto L(\alpha, \bar{\lambda}) \ . \tag{$L_{\bar{\lambda}}$}$$

Therefore, we can use the subdifferential characterization, namely

$$0 \in \partial - L_{\bar{\alpha}}(\bar{\lambda}) \quad \text{and} \quad 0 \in \partial L_{\bar{\lambda}}(\bar{\alpha}) \ .$$

In the sequel, we seek for $\bar{\alpha} \in \mathbb{R}^{n+1}$ and $\bar{\lambda} \in [0, 1]$ that satisfy these conditions. We start with the easier one, that is $\partial L_{\bar{\alpha}}(\lambda)$, and then focus on $\partial L_{\bar{\lambda}}(\alpha)$.

Computing the set $\partial L_{\bar{\alpha}}(\lambda)$, for $\lambda \in [0, 1]$ is immediate. Since the function $L_{\bar{\alpha}}(\lambda)$ is linear, it is differentiable, and its derivative is straightforward:

$$\begin{aligned} L_{\bar{\alpha}}(\lambda) &= \lambda(\delta(M^X(\alpha^X - \bar{\alpha}) \mid \mathfrak{B}) + \tau^X) + (1 - \lambda)(\delta(M^Y(\alpha^Y - \bar{\alpha}) \mid \mathfrak{B}) + \tau^Y) \\ &= a_{\bar{\alpha}}\lambda + b_{\bar{\alpha}}, \end{aligned}$$

where

$$\begin{aligned} a_{\bar{\alpha}} &= \delta(M^X(\alpha^X - \bar{\alpha}) \mid \mathfrak{B}) + \tau^X - \delta(M^Y(\alpha^Y - \bar{\alpha}) \mid \mathfrak{B}) - \tau^Y \\ b_{\bar{\alpha}} &= \delta(M^Y(\alpha^Y - \bar{\alpha}) \mid \mathfrak{B}) + \tau^Y \ . \end{aligned}$$

We have to be cautious with the subdifferential of the linear function at its borders, that is $\lambda = 0$ and $\lambda = 1$. Inbetween these borders, the subdifferential matches the differential of the function, that is its slope $\{a_{\bar{\alpha}}\}$.

Figure 6.4 depicts the epigraph of $-L_{\bar{\alpha}}(\lambda)$ (that is $-a_{\bar{\alpha}}\lambda - b_{\bar{\alpha}}$), and some of its subgradients at $\lambda = 0$ and $\lambda = 1$. Observe that the subgradient 0, drawn in horizontal red line, is an element of $\partial - L_{\bar{\alpha}}(1)$. Indeed, when the slope is non-positive ($-a_{\bar{\alpha}} < 0$), the global minimum of the function is reached at $\lambda = 1$. Dually, the maximum (which interests us) of $L_{\bar{\alpha}}(\lambda)$, that is $a_{\bar{\alpha}}\lambda + b_{\bar{\alpha}}$, when $-a_{\bar{\alpha}} < 0$ (or $a_{\bar{\alpha}} > 0$) is also reached at $\lambda = 1$.

Thus, we conclude with

$$\partial - L_{\bar{\alpha}}(\lambda) = \begin{cases} \{-a_{\bar{\alpha}}\} & \text{if } \lambda \in]0, 1[\\ (-\infty, -a_{\bar{\alpha}}] & \text{if } \lambda = 0 \\ [-a_{\bar{\alpha}}, +\infty) & \text{if } \lambda = 1 \end{cases} \tag{∂}$$

The proposition below summarizes the characterization of $\bar{\lambda}$. For the sake of clarity, we remind the expression of $a_{\bar{\alpha}}$ defined earlier to emphasize the linearity of $L_{\bar{\alpha}}(\lambda)$.

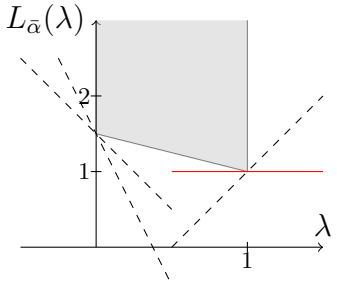

Figure 6.4: Subgradients of $L_{\bar{\alpha}}(\lambda)$ at $\lambda = 0$ and $\lambda = 1$.

6.2.13 Proposition
Let $a_{\bar{\alpha}} = \delta(M^X(\alpha^X - \bar{\alpha}) \mid \mathfrak{B}) + \tau^X - \delta(M^Y(\alpha^Y - \bar{\alpha}) \mid \mathfrak{B}) - \tau^Y$, *where* $\bar{\alpha}$ *is unknown vector of* $\mathbb{R}^{n+1}$, *then*

- *If* $a_{\bar{\alpha}} = 0$, *then* $\bar{\lambda}$ *may be any real number within* $[0, 1]$.
- *However, if* $a_{\bar{\alpha}} < 0$, *then necessarily* $\bar{\lambda} = 0$.
- *Finally, if* $a_{\bar{\alpha}} > 0$, *then necessarily* $\bar{\lambda} = 1$.

Proof. *The proposition is immediate from the computation of* $\partial - L_{\bar{\alpha}}(\bar{\lambda})$ *given in equation* (∂). ∎

The first case indicates that if the slope of the linear function is null, then every point of the domain of the function is a global maximum (or minimum). The two last cases formalize the intuition behind the fact that if $a_{\bar{\alpha}} < 0$ (resp. > 0), then the linear function $a_{\bar{\alpha}}\lambda + b_{\bar{\alpha}}$ is decreasing (res. increasing), then its global maximum is reached for $\lambda = 0$ (resp. $\lambda = 1$).

The proposition 6.2.13 gives the first relation between $\bar{\lambda}$ and $\bar{\alpha}$, the two components of the saddle-point we seek. The second and non-trivial relation is derived from $0 \in \partial L_{\bar{\lambda}}(\bar{\alpha})$. We detail hereafter, step by step, the way we derive it.

The remaining difficult part is the computation of $\partial L_{\bar{\lambda}}(\alpha)$, or more precisely, the characterization of $\bar{\alpha}$, such that the subdifferential $\partial L_{\bar{\lambda}}(\bar{\alpha})$ contains 0. To this aim, we use a central theorem which links the subgradients to the *Fenchel conjugate* (see Definition 6.2.14) of a convex function.

We define first the Fenchel conjugate of a given convex function, and then state, without proof [2], the theorem to be used.

6.2.14 Definition (Conjugate of a Convex Function)
Let f be a convex function on $\mathbb{R}^n$. We define the function f^* on $\mathbb{R}^n$, called the conjugate of f, by

$$f^*(t) \stackrel{\text{def}}{=} \sup\{\langle x, t\rangle - f(x) \mid x \in \mathbb{R}^p\} \ .$$

For instance, the support function δ we use for the order (see Definition 4.3.1 is the conjugate of the indicator function:

6.2.15 Definition (Indicator Function)
Let C be a set of $\mathbb{R}^n$, then

$$\delta^*(x \mid C) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } x \in C, \\ +\infty & \text{if } x \notin C. \end{cases}$$

Clearly, C is a convex set if and only if $\delta(x \mid C)$ is a convex function on $\mathbb{R}^p$.

Notice that we use the * notation to denote the conjugate of a given function; There is no operator meaning to the star notation when used with functions.

Theorem below shows that the conjugate f^* of a convex function f is at the heart of the characterization of the subdifferential of f.

6.2.16 Theorem (Duality and Subgradients)
For any proper convex function f and any vector x, the following four conditions on a vector t are equivalent to each other:

(a) $t \in \partial f(x)$;

(b) $\langle z, t\rangle - f(z)$ *achieves its supremum in* z *at* $z = x$;

(c) $f(x) + f^*(t) \leq \langle x, t\rangle$;

(d) $f(x) + f^*(t) = \langle x, t\rangle$.

[2] To be concise and focus on our computations, we made the choice to state some well-known theorem without proofs. Please, refer to Chapter 23, Theorem 23.5, in [Roc70] for instance, for detailed proofs.

A *proper* convex function means that the epigraph of that function does not contain a vertical line, that is a convex function f, where epi f is not empty, has at least one x such that $f(x) < +\infty$, and for every x, $f(x) > -\infty$. Simple improper convex functions are the functions $x \mapsto +\infty$ and $x \mapsto -\infty$.

The characterization of the set $\bar{\alpha}$ such that $0 \in \partial L_{\bar{\lambda}}(\bar{\alpha})$ is then a corollary of Theorem 6.2.16.

6.2.17 Corollary
We have $0 \in \partial L_{\bar{\lambda}}(\bar{\alpha})$ *if and only if*

$$L_{\bar{\lambda}}(\bar{\alpha}) + L_{\bar{\lambda}}{}^*(0) = 0 \ .$$

Proof. *Once we prove that the function $L_{\bar{\lambda}}$ is a proper convex function, the corollary is immediate from Theorem 6.2.16 using the equivalence between (a) and (d) for $t = 0$. The epigraph of $L_{\bar{\lambda}}$ is a non-empty subset of $\mathbb{R}^{n+1}$, the function is finite for at least one α, and by definition $L_{\bar{\lambda}} > -\infty$ for every $\alpha \in \mathbb{R}^n$.* ∎

6.2.18 Lemma
The conjugate of the function $L_{\bar{\lambda}}$ evaluated at 0, that is $L_{\bar{\lambda}}{}^*(0)$, is equal to

$$-\delta(\alpha^X - \alpha^Y \mid \bar{\lambda} M^{X*}\mathfrak{B} \cap (1-\bar{\lambda}) M^{Y*}\mathfrak{B}) - \bar{\lambda}\tau^X - (1-\bar{\lambda})\tau^Y \ .$$

Proof. *The proof is mainly an application of properties and operations of the Fenchel conjugate of convex functions. The detailed proof is given in appendix B.1.* ∎

Corollary 6.2.17, together with Lemma 6.2.18, give the second relation that puts together $\bar{\alpha}$ and $\bar{\lambda}$.

6.2.19 Proposition
Vector $\bar{\alpha}$ such that $\partial L_{\bar{\lambda}}(\bar{\alpha})$ contains 0 satisfies:

$$\bar{\lambda}\delta(M^X(\bar{\alpha} - \alpha^X) \mid \mathfrak{B}) + (1-\bar{\lambda})\delta(M^Y(\bar{\alpha} - \alpha^Y) \mid \mathfrak{B}) = \\ \delta(\alpha^X - \alpha^Y \mid \bar{\lambda} M^{X*}\mathfrak{B} \cap (1-\bar{\lambda}) M^{Y*}\mathfrak{B}) \ .$$

The following Theorem summarizes the two main propositions, that is Proposition 6.2.13 and Proposition 6.2.19, that establish the relations that $\bar{\alpha}$ and $\bar{\lambda}$ have to satisfy for $(\bar{\alpha}, \bar{\lambda})$ to be a saddle-point of $L(\alpha, \lambda)$.

6.2.20 Theorem

- *If* $\delta(M^Y(\alpha^Y - \alpha^X) \mid \mathfrak{B}) < \tau^Y - \tau^X$, *then* $(\alpha^Y, 0)$ *is the unique saddle-point of* L. *Its saddle-value is* τ^Y.
- *If* $\delta(M^Y(\alpha^Y - \alpha^X) \mid \mathfrak{B}) = \tau^Y - \tau^X$, *and* $\alpha^Y \neq \alpha^X$, *then* L *admits infinitely many saddle-points such that* $\bar{\alpha} = \alpha^Y$. *Its saddle-value is* τ^Y.
- *If* $\delta(M^X(\alpha^Y - \alpha^X) \mid \mathfrak{B}) < \tau^X - \tau^Y$, *then* $(\alpha^X, 1)$ *is the unique saddle-point of* L. *Its saddle-value is* τ^X.
- *If* $\delta(M^X(\alpha^Y - \alpha^X) \mid \mathfrak{B}) = \tau^X - \tau^Y$, *and* $\alpha^Y \neq \alpha^X$, *then* L *admits infinitely many saddle-points such that* $\bar{\alpha} = \alpha^X$. *Its saddle-value is* τ^X.
- *Otherwise,* $\delta(M^Y(\alpha^Y - \alpha^X) \mid \mathfrak{B}) > |\tau^Y - \tau^X|$ *and* $\delta(M^X(\alpha^Y - \alpha^X) \mid \mathfrak{B}) > |\tau^X - \tau^Y|$, *and* $(\bar{\alpha}, \bar{\lambda})$ *satisfies:*

$$\begin{aligned}
&\bar{\lambda} \in]0, 1[, \\
&\delta(M^X(\bar{\alpha} - \alpha^X) \mid \mathfrak{B}) + \tau^X = \delta(M^Y(\bar{\alpha} - \alpha^Y) \mid \mathfrak{B}) + \tau^Y, \\
&\bar{\lambda}\delta(M^X(\bar{\alpha} - \alpha^X) \mid \mathfrak{B}) + (1 - \bar{\lambda})\delta(M^Y(\bar{\alpha} - \alpha^Y) \mid \mathfrak{B}) = \\
&\qquad\qquad \delta(\alpha^X - \alpha^Y \mid \bar{\lambda} M^{X*}\mathfrak{B} \cap (1 - \bar{\lambda}) M^{Y*}\mathfrak{B}) \ .
\end{aligned}$$

Proof. *The proof is mainly a discussion about the combination of Propositions 6.2.19 and 6.2.13. The lengthy detailed proof can be found in appendix B.2.* ∎

Theorem 6.2.20 is not sufficient to compute automatically the saddle-points of L whenever $\delta(M^Y(\alpha^Y - \alpha^X) \mid \mathfrak{B}) > |\tau^Y - \tau^X|$ and $\delta(M^X(\alpha^Y - \alpha^X) \mid \mathfrak{B}) > |\tau^X - \tau^Y|$. It gives instead a system of equations that have to be satisfied by the saddle-points. Observe that in the last case of Theorem 6.2.20, $\bar{\lambda} \notin \{0, 1\}$, as this violates the hypothesis of that case. The next section focuses on solving these equations while presenting an algorithm which returns a saddle-point of L.

Computation of the Set of Mubs that Minimize the Perturbation

Let $\bar{\lambda}$ be a fixed element of $]0,1[$, and $u_{\bar{\lambda}}$ be a vector of $\bar{\lambda}M^{X^*}\mathfrak{B} \cap (1-\bar{\lambda})M^{Y^*}\mathfrak{B}$ such that

$$\langle \alpha^X - \alpha^Y, u_{\bar{\lambda}} \rangle = \delta(\alpha^X - \alpha^Y \mid \bar{\lambda}M^{X^*}\mathfrak{B} \cap (1-\bar{\lambda})M^{Y^*}\mathfrak{B}) \ .$$

The vector $u_{\bar{\lambda}}$ exists as the convex $\bar{\lambda}M^{X^*}\mathfrak{B} \cap (1-\bar{\lambda})M^{Y^*}\mathfrak{B}$ is not empty (contains at least $\{0\}$). We know, by definition of the saddle-point, that $\bar{\lambda}$ maximizes $L_{\bar{\alpha}}$ defined in equation $L_{\bar{\alpha}}$. So our approach is twofold: we first seek the pair $(\bar{\lambda},\ u_{\bar{\lambda}})$ that maximizes $L_{\bar{\alpha}}$; we then deduce $\bar{\alpha}$. Since the expression of $L_{\bar{\alpha}}$ depends on $\bar{\alpha}$, we need to overcome such dependency. This can be done by using the definition of $u_{\bar{\lambda}}$, as introduced above. Indeed $L_{\bar{\alpha}}(\bar{\lambda}) = L(\bar{\alpha}, \bar{\lambda})$, which gives

$$L_{\bar{\alpha}}(\bar{\lambda}) = \langle \alpha^X - \alpha^Y, u_{\bar{\lambda}} \rangle + \bar{\lambda}\tau^X + (1-\bar{\lambda})\tau^Y \ .$$

Therefore, $\bar{\lambda}$ is the optimal solution of the following non-linear optimization problem:

$$\begin{array}{rl} \max & \langle \alpha^X - \alpha^Y, u_\lambda \rangle + \lambda\tau^X + (1-\lambda)\tau^Y \\ \text{s.t.} & 0 < \lambda < 1 \\ & u_\lambda \in \lambda M^{X^*}\mathfrak{B} \cap (1-\lambda)M^{Y^*}\mathfrak{B} \end{array} \qquad \text{(P)}$$

Solving (P)

The dimension of (P) is $n+2$ since the unknown vector u_λ is an element of $\mathbb{R}^{n+1}$. To solve (P), we first reduce the dimension of the problem from $n+2$ to 2. We hence obtain a much simpler non-linear optimization problem defined then solved at the end of this section.

We start with our running example 6.2.6 to bring closer the idea we apply later to the general case. Let $u_\lambda = (u_0, u_1, u_2)^*$, the constraint $u_\lambda \in \lambda M^{X^*}\mathfrak{B} \cap (1-\lambda)M^{Y^*}\mathfrak{B}$ is equivalent to

$$\lambda^{-1}M^{X^{*-1}}u_\lambda \in \mathfrak{B} \quad \text{and} \quad (1-\lambda)^{-1}M^{Y^{*-1}}u_\lambda \in \mathfrak{B} \ .$$

The inverses of λ and $(1-\lambda)$ are finite since λ is within $]0,1[$. The matrices M^X and M^Y are non-singular as we are restricted to the case where all deviations of the interval concretisations of noise symbols are not null. We recall the matrices M^X and M^Y:

$$M^X = \begin{pmatrix} 1 & -0.5 & 0.25 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.25 \end{pmatrix}, \quad M^Y = \begin{pmatrix} 1 & 0 & 0.5 \\ 0 & 0.5 & 0 \\ 0 & 0 & 0.5 \end{pmatrix} \ .$$

which gives

$$M^{X^{*-1}} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 2 & 0 \\ -1 & 0 & 4 \end{pmatrix}, \quad M^{Y^{*-1}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ -1 & 0 & 2 \end{pmatrix},$$

and one has:

$$\lambda^{-1} M^{X^{*-1}} u_\lambda \in \mathfrak{B} \iff \begin{cases} -\lambda \leq u_0 \leq \lambda \\ -\lambda \leq u_0 + 2u_1 \leq \lambda \\ -\lambda \leq -u_0 + 4u_2 \leq \lambda \end{cases}$$

and

$$(1-\lambda)^{-1} M^{Y^{*-1}} u_\lambda \in \mathfrak{B} \iff \begin{cases} -(1-\lambda) \leq u_0 \leq (1-\lambda) \\ -(1-\lambda) \leq 2u_1 \leq (1-\lambda) \\ -(1-\lambda) \leq -u_0 + 2u_2 \leq (1-\lambda) \end{cases}$$

Combined together these constraints lead to

$$\begin{array}{l} \max\{-\lambda, -(1-\lambda)\} \leq u_0 \leq \min\{\lambda, (1-\lambda)\} \\ \max\{\frac{-\lambda-u_0}{2}, \frac{-(1-\lambda)}{2}\} \leq u_1 \leq \min\{\frac{\lambda-u_0}{2}, \frac{1-\lambda}{2}\} \\ \max\{\frac{-\lambda+u_0}{4}, \frac{-(1-\lambda)+u_0}{2}\} \leq u_2 \leq \min\{\frac{\lambda+u_0}{4}, \frac{(1-\lambda)+u_0}{2}\} \end{array}$$

or equivalently, using only the min operator ($\max\{a, b\} = -\min\{-a, -b\}$):

$$\begin{array}{l} -\min\{\lambda, (1-\lambda)\} \leq u_0 \leq \min\{\lambda, (1-\lambda)\} \\ -\min\{\frac{\lambda+u_0}{2}, \frac{(1-\lambda)}{2}\} \leq u_1 \leq \min\{\frac{\lambda-u_0}{2}, \frac{1-\lambda}{2}\} \\ -\min\{\frac{\lambda-u_0}{4}, \frac{(1-\lambda)-u_0}{2}\} \leq u_2 \leq \min\{\frac{\lambda+u_0}{4}, \frac{(1-\lambda)+u_0}{2}\} \ . \end{array}$$

We also know that $\alpha_0^X - \alpha_0^Y = 1 - 2 = -1$, $\alpha_1^X - \alpha_1^Y = -1 - 1 = -2$, and $\alpha_2^X - \alpha_2^Y = 2 - 1 = 1$ and that $\tau^X = \tau^Y = 0$. We can now explicit an upper bound of $\langle \alpha^X - \alpha^Y, u_\lambda \rangle + \lambda\tau^X + (1-\lambda)\tau^Y$ by substituting u_1 and u_2 by their respective upper bounds:

$$\begin{aligned} & \langle \alpha^X - \alpha^Y, u_\lambda \rangle + \lambda\tau^X + (1-\lambda)\tau^Y \\ = & -u_0 - 2u_1 + u_2 + 0 + 0 \\ \leq & -u_0 + 2\min\{\frac{\lambda+u_0}{2}, \frac{(1-\lambda)}{2}\} + \min\{\frac{\lambda+u_0}{4}, \frac{(1-\lambda)+u_0}{2}\} \ . \end{aligned}$$

This upper bound is reached whenever

$$u_1 = -\min\{\frac{\lambda+u_0}{2}, \frac{(1-\lambda)}{2}\} \quad , \quad u_2 = \min\{\frac{\lambda+u_0}{4}, \frac{(1-\lambda)+u_0}{2}\} \ .$$

Thus, instead of maximizing $\langle \alpha^X - \alpha^Y, u_\lambda \rangle + \lambda\tau^X + (1-\lambda)\tau^Y$, which is our original objective function, we maximize its (attained) upper bound. This ends the dimension reduction step: the original problem has a dimension of $n+2 = 2+2 = 4$, the variables involved are (u_0, u_1, u_2) and λ. Now the problem we obtain has only 2 unknowns, namely u_0 and λ, and the feasible region is defined by

$$0 < \lambda < 1 \quad , \quad -\min\{\lambda, (1-\lambda)\} \leq u_0 \leq \min\{\lambda, (1-\lambda)\} \ .$$

We formalize the general case of the dimension reduction step exemplified above. Then, we continue solving our running example before formalizing the general case of the newly reduced optimization problem.

6.2.21 Proposition (Dimension Reduction)
Let the pair $(\bar\lambda, u_{\bar\lambda})$ be a solution to the optimization problem (P). Let $\bar u_i$ denote the ith component of the vector $u_{\bar\lambda}$. Then $\bar u_i$, $1 \leq i \leq n$, satisfies

$$\bar u_i = \begin{cases} \min\{\bar\lambda \operatorname{dev}(\boldsymbol{\epsilon_i^X}) + \operatorname{mid}(\boldsymbol{\epsilon_i^X})\bar u_0, (1-\bar\lambda)\operatorname{dev}(\boldsymbol{\epsilon_i^Y}) + \operatorname{mid}(\boldsymbol{\epsilon_i^Y})\bar u_0\}, \\ \quad \textit{if } \operatorname{sign}(\alpha_i^X - \alpha_i^Y) = 1, \\ -\min\{\bar\lambda \operatorname{dev}(\boldsymbol{\epsilon_i^X}) - \operatorname{mid}(\boldsymbol{\epsilon_i^X})\bar u_0, (1-\bar\lambda)\operatorname{dev}(\boldsymbol{\epsilon_i^Y}) - \operatorname{mid}(\boldsymbol{\epsilon_i^Y})\bar u_0\}, \\ \quad \textit{if } \operatorname{sign}(\alpha_i^X - \alpha_i^Y) = -1, \\ \textit{any real number}\ , \\ \quad \textit{if } \alpha_i^X = \alpha_i^Y . \end{cases}$$

where the pair $(\bar\lambda, \bar u_0)$ is a solution of the following optimization problem

$$\begin{aligned} \max \quad & \tau^Y + (\alpha_0^X - \alpha_0^Y)u_0 + (\tau^X - \tau^Y)\lambda + \sum_{i=1}^{n} |\alpha_i^x - \alpha_i^y| u_i \\ s.t. \quad & 0 < \lambda < 1 \\ & -\min\{\lambda, (1-\lambda)\} \leq u_0 \leq \min\{\lambda, (1-\lambda)\} \end{aligned} \qquad (P_r)$$

The coordinates u_i, $1 \leq i \leq n$, are defined as $\bar u_i$ by substituting $\bar\lambda$ with λ and $\bar u_0$ by u_0.

Proof. *The proof is a generalization of the computations previously done. The inverse of matrix M^* is*

$$M^{*-1}_{(i,j)} = \begin{cases} 1 & \textit{if } i = 1 \textit{ and } j = 1, \\ \frac{-M_{(i,1)}}{M_{(i,i)}} & \textit{if } i \neq 1 \textit{ and } j = 1, \\ \frac{1}{M_{(i,i)}} & \textit{if } i \neq 1 \textit{ and } j = i, \\ 0 & \textit{otherwise} \end{cases}$$

Let u_i, $0 \leq i \leq n$, denote the ith component of the vector u_λ. From the constraints $\lambda^{-1}M^{X*-1}u_\lambda \in \mathfrak{B}$, and $(1-\lambda)^{-1}M^{Y*-1}u_\lambda \in \mathfrak{B}$ we deduce an (attained) upper bound for each component u_i, $i \geq 1$, function of u_0 (the first component) and λ, and an additional condition on u_0 and λ : $|u_0| \leq \min\{\lambda, (1-\lambda)\}$. Using these upper bounds, we over-approximate the objective function of (P), which involves in turn only two unknowns u_0 and λ. Therefore, the total dimension of the problem is reduced from $n+2$ initially to 2 in (P_r). ∎

The objective function of the optimization problem (P_r), denoted by $f(u_0, \lambda)$, has the form:

$$f(u_0,\lambda) \stackrel{\text{def}}{=} \tau^Y + \alpha_0 u_0 + (\tau^X - \tau^Y)\lambda + \sum_{i=1}^{n} \alpha_i \min\{a_i\lambda + b_i u_0, a'_i(1-\lambda) + b'_i u_0\}$$

where $\alpha_0 \stackrel{\text{def}}{=} \alpha_0^X - \alpha_0^Y \in \mathbb{R}$, $(\alpha_i)_{1\leq i\leq n}$ are positive real numbers, $0 < a_i, a'_i \leq 1$, and $-1 < b_i, b'_i \leq 1$. The feasible region is draw in Figure 6.5.

The function $f(u_0, \lambda)$ is concave: it is defined as a sum of concave functions, f_i,

$$f_i : (u_0,\lambda) \mapsto \alpha_i \min\{a_i\lambda + b_i u_0, a'_i(1-\lambda) + b'_i u_0\}$$

and a linear function, f_0,

$$f_0 : (u_0,\lambda) \mapsto \tau^Y + (\alpha_0^X - \alpha_0^Y)u_0 + (\tau^X - \tau^Y)\lambda \ .$$

Each concave function f_i, $1 \leq i \leq n$ reaches its maximum necessarily on the line L_i

$$L_i \stackrel{\text{def}}{=} \{(\lambda, u_0) \mid a_i\lambda + b_i u_0 = a'_i(1-\lambda) + b'_i u_0\}, \qquad (L_i)$$

that is, when the operands of the min operator are equal. Now, each line (L_i) that intersects the feasible region divides that feasible region into two regions. In each region, the function f_i is a linear function since the min operator is evaluated to one of each both operands. Therefore, the lines (L_i) that intersect the feasible region define a tiling of polygons; in each polygon the objective function f is an affine function. It is well-known that an affine function defined over a bounded polytope achieves its maximum at at least one vertex of that polytope. The function f reaches then its maximum at least at one of the vertices defined by the polygons' tiling itself defined by the set of (L_i) lines.

For our running example, using Proposition 6.2.21, the problem (P_r) is

$$\begin{array}{ll} \max & -u_0 + 2\min\{\frac{\lambda+u_0}{2}, \frac{1-\lambda}{2}\} + \min\{\frac{\lambda+u_0}{4}, \frac{1-\lambda+u_0}{2}\} \\ \text{s.t.} & 0 < \lambda < 1 \\ & |u_0| \leq \min\{\lambda, (1-\lambda)\} \end{array}$$

Figure 6.6 depicts the feasible region (gray diamond), as well as the polygons' tiling defined by the lines L_1 and L_2 (see equation L_i):

$$\begin{aligned} L_1 &\stackrel{\text{def}}{=} \{(\lambda, u_0) \mid \lambda + u_0 = (1-\lambda\} \\ L_2 &\stackrel{\text{def}}{=} \{(\lambda, u_0) \mid \lambda + u_0 = 2(1-\lambda+u_0)\} \end{aligned}$$

We obtain 6 feasible vertices, and denote between parentheses, for each vertex, the evaluation of the objective function of (P_r). In this example, the optimal value 0.7 is reached by a unique vertex, which is the optimal solution, $(\bar{\lambda} = 0.6, \bar{u}_0 = -0.2)$, defined by the intersection of lines L_1 and L_2. Since $\text{sign}(\alpha_1^X - \alpha_1^Y) = -1$ and $\text{sign}(\alpha_2^X - \alpha_2^Y) = 1$, we deduce $\bar{u}_1$ and $\bar{u}_2$ from Proposition 6.2.21

$$\begin{aligned} \bar{u}_1 &= -\min\{\frac{\bar{\lambda}+\bar{u}_0}{2}, \frac{1-\bar{\lambda}}{2}\} = -\min\{0.4, 0.2\} = -0.2 \\ \bar{u}_2 &= \min\{\frac{\bar{\lambda}+\bar{u}_0}{4}, \frac{1-\bar{\lambda}+\bar{u}_0}{2}\} = \min\{0.1, 0.1\} = 0.1 \ . \end{aligned}$$

Algorithm 1 solves problem (P_r) then gives an optimal solution to the problem (P), in the general case. The set $\{B_k\}_{1\leq k\leq 4}$ denotes the equations of the four borders of the feasible region:

$$\begin{aligned} B_1 &\stackrel{\text{def}}{=} \{(\lambda, u_0) \mid u_0 + \lambda = 0\} & B_2 &\stackrel{\text{def}}{=} \{(\lambda, u_0) \mid u_0 - (1-\lambda) = 0\} \\ B_3 &\stackrel{\text{def}}{=} \{(\lambda, u_0) \mid u_0 + (1-\lambda) = 0\} & B_4 &\stackrel{\text{def}}{=} \{(\lambda, u_0) \mid u_0 - \lambda = 0\} \ . \end{aligned}$$

The set V denotes the set of vertices of the polygons' tiling. It is initially set to $\{(\frac{1}{2}, \frac{1}{2}), (\frac{1}{2}, \frac{-1}{2})\}$, the two unique feasible vertices of the feasible region. The first for loop (line 3) goes through the given list of all L_i lines. If the intersection of L_i and the borders $\{B_k\}_{1\leq k\leq 4}$ (line 4) of the feasible region is feasible, we store the vertex in V. We then compute all the intersections of the line L_i with the other lines L_j such that $i \neq j$ (line 8). For each vertex $v = (v_0, v_1)$ in V (line 14), we evaluated the objective function f using an external routine `evalf` (lines 13 and 16). The algorithm updates the variable *objval* with the greatest value of f (line 18) and the temp

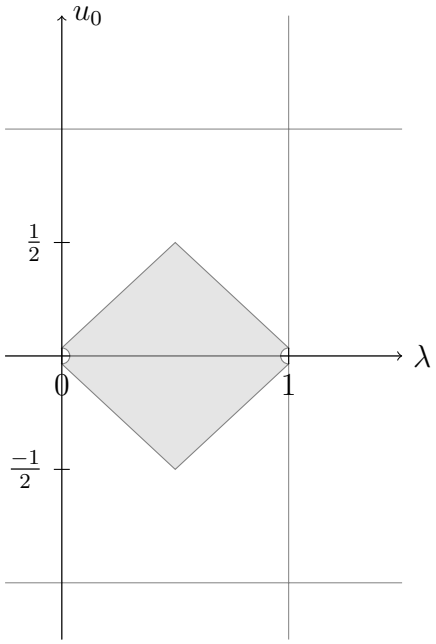

Figure 6.5: The feasible region of the optimization problem (P_r). The points $(0, 0)$ and $(1, 0)$ are not feasible.

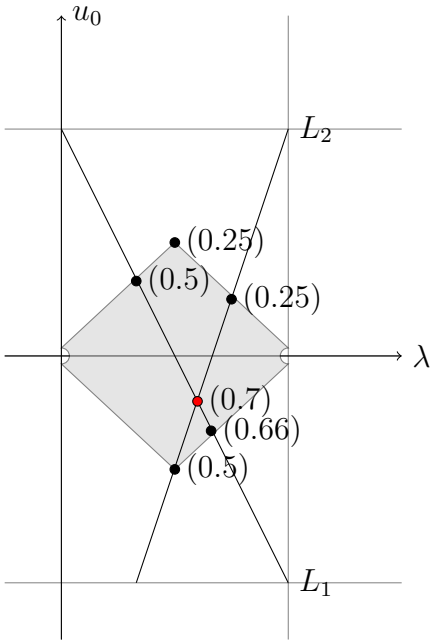

Figure 6.6: The evaluation of the objective function of (P_r) at each vertex.

variable t with the last vertex for which this objective value is reached (line 19). The final for loop (line 21) assigns the value of $\bar{u}_i$, $1 \leq i \leq n$ using Proposition 6.2.21. We finally return the objective value *objval*, as well as $(\bar{\lambda}, u_{\bar{\lambda}})$ an optimal solution to the problem (P).

6.2.22 Proposition

Algorithm 1 has a complexity of $\mathcal{O}(n^3)$ in the worst case, where n denotes the number of the L_i lines (or equivalently the number of noise symbols).

Proof. *The cardinal of the set of intersections of an arrangement of n lines is equal to $\frac{n(n-1)}{2}$ in the worst case. Thus, we have to evaluate the objective function at $(2n+2+\frac{n(n-1)}{2}) = \frac{n^2+3n+4}{2}$ vertices in the worst case: $2n$ for the intersections of the n lines with the borders of the feasible region, 2 for the vertices $(\frac{1}{2}, \frac{1}{2})$ and $(\frac{1}{2}, \frac{-1}{2})$, and $\frac{n(n-1)}{2}$ for the intersections of the L_i lines themselves. The evaluation of the objective function on each point needs in*

Algorithm 1: Solving (P)

input : A set of lines $\{L_i\}_{1\leq i\leq n}$, $\{B_k\}_{1\leq k\leq 4}$, $\{\alpha_i^X - \alpha_i^Y\}_{0\leq i\leq n}$, τ^X and τ^Y.

output: *optval*, the optimal value of (P), and $(\bar{\lambda}, u_{\bar{\lambda}})$ an optimal solution of (P).

$l \longleftarrow 0$;
$V \longleftarrow \{(\frac{1}{2}, \frac{1}{2}), (\frac{1}{2}, \frac{-1}{2})\}$;
for $i \leftarrow 1$ **to** n **do**
    for $k \leftarrow 1$ **to** 4 **do**
        if $(L_i \cap B_k)$ *is feasible* **then**
            $V \longleftarrow V \cup (L_i \cap B_k)$;
            $l \longleftarrow l + 1$;
    for $j \leftarrow i + 1$ **to** n **do**
        if $L_i \neq L_j$ ***and*** $(L_i \cap L_j)$ *is feasible* **then**
            $V \longleftarrow V \cup (L_i \cap L_j)$;
            $l \longleftarrow l + 1$;
$v \longleftarrow V[0]$;
$objval \longleftarrow$ `evalf`$(\{\alpha_i^X - \alpha_i^Y\}_{0\leq i\leq n}, \tau^X, \tau^Y, v_0, v_1)$;
for $i \leftarrow 1$**to** $l - 1$ **do**
    $v \longleftarrow V[i]$;
    $f \longleftarrow$ `evalf`$(\{\alpha_i^X - \alpha_i^Y\}_{0\leq i\leq n}, \tau^X, \tau^Y, v_0, v_1)$;
    if $f \geq objval$ **then**
        $objval \longleftarrow f$;
        $(t_0, t_1) \longleftarrow V[i]$;
$\bar{\lambda} \longleftarrow t_0$;
for $i \longleftarrow 1$**to** n **do**
    if $\alpha_i^X - \alpha_i^Y \geq 0$ **then**
        $\bar{u}_i \leftarrow$
        $\min\{\bar{\lambda}\,\mathrm{dev}(\boldsymbol{\epsilon_i^X}) + \mathrm{mid}(\boldsymbol{\epsilon_i^X})\bar{u}_0, (1-\bar{\lambda})\,\mathrm{dev}(\boldsymbol{\epsilon_i^Y}) + \mathrm{mid}(\boldsymbol{\epsilon_i^Y})\bar{u}_0\}$;
    else
        $\bar{u}_i \leftarrow$
        $-\min\{\bar{\lambda}\,\mathrm{dev}(\boldsymbol{\epsilon_i^X}) - \mathrm{mid}(\boldsymbol{\epsilon_i^X})\bar{u}_0, (1-\bar{\lambda})\,\mathrm{dev}(\boldsymbol{\epsilon_i^Y}) - \mathrm{mid}(\boldsymbol{\epsilon_i^Y})\bar{u}_0\}$;
$u_{\bar{\lambda}} \longleftarrow (t_1, \bar{u}_1, \ldots, \bar{u}_n)$;
return $objval, \bar{\lambda}, u_{\bar{\lambda}}$;

the worst case n operations. The total number of operations, in the worst case, is then $\frac{n^3+3n^2+4n}{2}$. ∎

Deducing $\bar{\alpha}$

This section details the second step towards computing automatically a saddle-point of the function (6.2.1) whenever the conditions $\delta(M^Y(\alpha^Y - \alpha^X) \mid \mathfrak{B}) > |\tau^Y - \tau^X|$ and $\delta(M^X(\alpha^Y - \alpha^X) \mid \mathfrak{B}) > |\tau^X - \tau^Y|$ hold. Theorem 6.2.20 characterizes the set of saddle-points as the set of solutions to the equations:

$$\begin{aligned} &\imath) \quad \delta(M^X(\bar{\alpha} - \alpha^X) \mid \mathfrak{B}) + \tau^X = \delta(M^Y(\bar{\alpha} - \alpha^Y) \mid \mathfrak{B}) + \tau^Y, \\ &\imath\imath) \quad \bar{\lambda}\delta(M^X(\bar{\alpha} - \alpha^X) \mid \mathfrak{B}) + (1-\bar{\lambda})\delta(M^Y(\bar{\alpha} - \alpha^Y) \mid \mathfrak{B}) = \\ &\qquad \delta(\alpha^X - \alpha^Y \mid \bar{\lambda}M^{X^*}\mathfrak{B} \cap (1-\bar{\lambda})M^{Y^*}\mathfrak{B}) \ . \end{aligned}$$

The previous section was dedicated to the computation of $\bar{\lambda}$ as well as $u_{\bar{\lambda}}$ such that $u_{\bar{\lambda}} \in \bar{\lambda}M^{X^*}\mathfrak{B} \cap (1-\bar{\lambda})M^{Y^*}\mathfrak{B}$ and

$$\langle \alpha^X - \alpha^Y \mid u_{\bar{\lambda}} \rangle = \delta(\alpha^X - \alpha^Y \mid \bar{\lambda}M^{X^*}\mathfrak{B} \cap (1-\bar{\lambda})M^{Y^*}\mathfrak{B}) \ .$$

In the sequel we deduce $\bar{\alpha}$ from $\bar{\lambda}$ and $u_{\bar{\lambda}}$.

6.2.23 Proposition
The vector $\bar{\alpha}$ *satisfies*

$$\begin{aligned} \delta(\bar{\alpha} - \alpha^X \mid \bar{\lambda}M^{X^*}\mathfrak{B}) &= \langle \alpha^X - \bar{\alpha}, u_{\bar{\lambda}} \rangle \\ \delta(\bar{\alpha} - \alpha^Y \mid (1-\bar{\lambda})M^{Y^*}\mathfrak{B}) &= \langle \bar{\alpha} - \alpha^Y, u_{\bar{\lambda}} \rangle \ . \end{aligned}$$

Proof. *Since,* $u_{\bar{\lambda}} \in \bar{\lambda}M^{x*}\mathcal{B}_\infty^X$, *and by definition of the support function, we have*

$$\langle \alpha^X - \bar{\alpha}, u_{\bar{\lambda}} \rangle \leq \delta(\alpha^X - \bar{\alpha} \mid \bar{\lambda}M^{X^*}\mathfrak{B}), \tag{6.2.2}$$

Similarly, we have $u_{\bar{\lambda}} \in (1-\bar{\lambda})M^{Y^*}\mathfrak{B}$, *which gives*

$$\langle \bar{\alpha} - \alpha^Y, u_{\bar{\lambda}} \rangle \leq \delta(\bar{\alpha} - \alpha^Y \mid (1-\bar{\lambda})M^{Y^*}\mathfrak{B}) \ . \tag{6.2.3}$$

The strict inequality in the equation 6.2.2 or the equation 6.2.3 leads, by summing the two inequalities, to the strict inequality

$$\langle \alpha^X - \bar{\alpha}, u_{\bar{\lambda}} \rangle + \langle \bar{\alpha} - \alpha^Y, u_{\bar{\lambda}} \rangle < \delta(\alpha^X - \bar{\alpha} \mid \bar{\lambda}M^{x*}\mathfrak{B}) + \delta(\bar{\alpha} - \alpha^Y \mid (1-\bar{\lambda})M^{Y^*}\mathfrak{B})$$

By the linearity of the scalar product, the right hand side of the inequality above is equal to $\langle \alpha^X - \alpha^Y, u_{\bar{\lambda}} \rangle$. *By definition of* $u_{\bar{\lambda}}$,

$$\langle \alpha^X - \alpha^Y \mid u_{\bar{\lambda}} \rangle = \delta(\alpha^X - \alpha^Y \mid \bar{\lambda} M^{X*}\mathfrak{B} \cap (1-\bar{\lambda}) M^{Y*}\mathfrak{B}) \ .$$

and by 6.2.20, ıı, we have

$$\bar{\lambda}\delta(M^X(\bar{\alpha} - \alpha^X) \mid \mathfrak{B}) + (1-\bar{\lambda})\delta(M^Y(\bar{\alpha} - \alpha^Y) \mid \mathfrak{B}) = \\ \delta(\alpha^X - \alpha^Y \mid \bar{\lambda} M^{X*}\mathfrak{B} \cap (1-\bar{\lambda}) M^{Y*}\mathfrak{B}) \ .$$

thus, the right hand side of the inequality above is also equal to $\langle \alpha^X - \alpha^Y, u_{\bar{\lambda}} \rangle$, *which is impossible. Therefore, the equalities*

$$\delta(\bar{\alpha} - \alpha^X \mid \bar{\lambda} M^{X*}\mathfrak{B}) = \langle \alpha^X - \bar{\alpha}, u_{\bar{\lambda}} \rangle$$
$$\delta(\bar{\alpha} - \alpha^Y \mid (1-\bar{\lambda}) M^{Y*}\mathfrak{B}) = \langle \bar{\alpha} - \alpha^Y, u_{\bar{\lambda}} \rangle$$

hold necessarily. ∎

These equalities together with the equality (ı) of Theorem 6.2.20 restrict the vector $\bar{\alpha}$ to the following hyperplane.

6.2.24 Proposition
The vector $\bar{\alpha}$ *lies in the hyperplane defined by*

$$\langle \bar{\alpha}, u_{\bar{\lambda}} \rangle = \langle (1-\bar{\lambda})\alpha^X + \bar{\lambda}\alpha^Y, u_{\bar{\lambda}} \rangle - \bar{\lambda}(1-\bar{\lambda})\tau^Y + \bar{\lambda}(1-\bar{\lambda})\tau^X \ .$$

Proof.

(*by equality (ı) of Theorem 6.2.20.*)
$\delta(M^X(\bar{\alpha} - \alpha^X) \mid \mathfrak{B}) + \tau^X = \delta(M^Y(\bar{\alpha} - \alpha^Y) \mid \mathfrak{B}) + \tau^Y$
(*using Proposition 6.2.23.*)
$\iff \bar{\lambda}^{-1}\langle \alpha^X - \bar{\alpha}, u_{\bar{\lambda}} \rangle + \tau^X = (1-\bar{\lambda})^{-1}\langle \bar{\alpha} - \alpha^Y, u_{\bar{\lambda}} \rangle + \tau^Y$
(*multiply by* $\bar{\lambda}(1-\bar{\lambda})$.)
$\iff \langle (1-\bar{\lambda})(\alpha^X - \bar{\alpha}), u_{\bar{\lambda}} \rangle + \bar{\lambda}(1-\bar{\lambda})\tau^X = \langle \bar{\lambda}(\bar{\alpha} - \alpha^Y), u_{\bar{\lambda}} \rangle + \bar{\lambda}(1-\bar{\lambda})\tau^Y.$

The result is then deduced by linear properties of the inner product. ∎

Moreover, the equalities of Proposition 6.2.23 imply interesting constraints on the components of the vector $\bar{\alpha}$, namely they determine the sign vector of the vectors $M^X(\alpha^X - \bar{\alpha})$ and $M^Y(\bar{\alpha} - \alpha^Y)$. A sign vector of a vector $v \in \mathbb{R}^n$ is a vector of $\mathbb{R}^n$ defined by $(\text{sign}(v_1), \ldots, \text{sign}(v_n))$, the sign of 0 is undefined and can be any real number (which means that we may have more than one sign vector for a given vector v).

6.2.25 Proposition
The vector $\bar{\lambda}^{-1} M^{X^{*-1}} u_{\bar{\lambda}}$ is a sign vector of $M^X(\alpha^X - \bar{\alpha})$. That is:

$$\begin{aligned}
(\bar{\lambda}^{-1} M^{X^{*-1}} u_{\bar{\lambda}})_i = 1 &\implies (\bar{\lambda} M^X(\alpha^X - \bar{\alpha}))_i \geq 0 \\
(\bar{\lambda}^{-1} M^{X^{*-1}} u_{\bar{\lambda}})_i = -1 &\implies (\bar{\lambda} M^X(\alpha^X - \bar{\alpha}))_i \leq 0 \\
(\bar{\lambda}^{-1} M^{X^{*-1}} u_{\bar{\lambda}})_i \neq \{-1, 1\} &\implies (\bar{\lambda} M^X(\alpha^X - \bar{\alpha}))_i = 0
\end{aligned}$$

Similarly, the vector $((1-\bar{\lambda})^{-1} M^{Y^{*-1}} u_{\bar{\lambda}})_i$ is a sign vector of $M^Y(\bar{\alpha} - \alpha^Y)$, and

$$\begin{aligned}
((1-\bar{\lambda})^{-1} M^{Y^{*-1}} u_{\bar{\lambda}})_i = 1 &\implies ((1-\bar{\lambda}) M^Y(\bar{\alpha} - \alpha^Y))_i \geq 0 \\
((1-\bar{\lambda})^{-1} M^{Y^{*-1}} u_{\bar{\lambda}})_i = -1 &\implies ((1-\bar{\lambda}) M^Y(\bar{\alpha} - \alpha^Y))_i \leq 0 \\
((1-\bar{\lambda})^{-1} M^{Y^{*-1}} u_{\bar{\lambda}})_i \neq \{-1, 1\} &\implies ((1-\bar{\lambda}) M^Y(\bar{\alpha} - \alpha^Y))_i = 0 \ .
\end{aligned}$$

Proof. *We only detail the case of $M^X(\alpha^X - \bar{\alpha})$, the second one, concerning $M^Y(\bar{\alpha} - \alpha^Y)$, is similar by substituting X by Y and $\bar{\lambda}$ by $1 - \bar{\lambda}$.*

We know that the support function of a vector v over the ball $\mathfrak{B}$ is equal to the taxicab norm of the vector v, and that the latter, by definition, is equal to the sum of the terms $v_i \,\text{sign}(v_i)$:

$$\delta(v \mid \mathfrak{B}) = \|v\|_1 = \sum_{i=1}^{n} v_i \,\text{sign}(v_i) = \langle v, \text{sign}(v) \rangle \ .$$

The vector $\text{sign}(v)$ is unique up to the null component v_i of v, that is, if w is another sign vector of v, whenever the sign of v_i is well defined ($v_i \neq 0$), $w_i = \text{sign}(v_i)$. We use this property in what follows while taking $\bar{\lambda} M^X(\alpha^X - \bar{\alpha})$ as our vector "v". We have

$$\begin{aligned}
\langle \bar{\lambda} M^X(\alpha^X - \bar{\alpha}), \bar{\lambda}^{-1} M^{X^{*-1}} u_{\bar{\lambda}} \rangle &= \langle \alpha^X - \bar{\alpha}, u_{\bar{\lambda}} \rangle \quad \textit{(inner product properties)} \\
&= \delta(\alpha^X - \bar{\alpha} \mid \bar{\lambda} M^{X^*} \mathfrak{B}) \ \textit{(Proposition 6.2.23)} \\
&= \delta(\bar{\lambda} M^X(\alpha^X - \bar{\alpha}) \mid \mathfrak{B}) \ \textit{(Proposition A.0.5)}
\end{aligned}$$

Thus, $\bar{\lambda}^{-1} M^{X^{*-1}} u_{\bar{\lambda}}$ is a sign vector of $\bar{\lambda} M^X(\alpha^X - \bar{\alpha})$. ∎

Proposition 6.2.25 deduces from the equalities of Proposition 6.2.23 a set of constraints that the components of the vector $\bar{\alpha}$ have to satisfy. These constraints may be simplified using the fact that $\bar{\lambda}$ and $(1-\bar{\lambda})$ are positive real numbers and the sparse form of the matrices M^X and M^Y. Indeed, entries of these matrices are all zero except for the first line and the main diagonal. Moreover, all entries of the latter are positive real numbers (the deviations of the interval concretisations of noise symbols).

6.2.26 Proposition
If $\bowtie$ denotes an element of $\{\leq, =, \geq\}$, then $(\bar{\lambda}M^X(\alpha^X - \bar{\alpha}))_i \bowtie 0$ if and only if

$$\begin{cases} (\alpha_0^X - \bar{\alpha}_0) + \sum_{i=1}^n \mathrm{mid}(\boldsymbol{\epsilon_i^X})\bar{\alpha}_i \bowtie 0, & \text{if } i = 0, \\ \alpha_i^X - \bar{\alpha}_i \bowtie 0, & \text{if } 1 \leq i \leq n \end{cases} .$$

Similarly, $((1-\bar{\lambda})M^Y(\bar{\alpha} - \alpha^Y))_i \bowtie 0$ if and only if

$$\begin{cases} -(\alpha_0^Y - \bar{\alpha}_0) - \sum_{i=1}^n \mathrm{mid}(\boldsymbol{\epsilon_i^Y})\bar{\alpha}_i \bowtie 0, & \text{if } i = 0, \\ \bar{\alpha}_i - \alpha_i^Y \bowtie 0, & \text{if } 1 \leq i \leq n \end{cases} .$$

We apply Propositions 6.2.24 and 6.2.25 to our running example. We recall, from the previous section, that $\bar{\lambda} = 0.6$ and $u_{\bar{\lambda}} = (-0.2, -0.2, 0.1)$. Proposition 6.2.24 gives the following hyperplane

$$-0.2\bar{\alpha}_0 - 0.2\bar{\alpha}_1 + 0.1\bar{\alpha}_2 + 0.22 = 0 \ .$$

Proposition 6.2.25 and the equivalences of Proposition 6.2.26 give

$$\begin{cases} -1 < \bar{\alpha}_1 < 1 \\ 1 < \bar{\alpha}_2 < 2 \\ -\bar{\alpha}_0 + 0.5\bar{\alpha}_1 - 0.25\bar{\alpha}_2 + 2 = 0 \\ \bar{\alpha}_0 + 0.5\bar{\alpha}_2 - 2.5 = 0 \end{cases}$$

The unique vector $\bar{\alpha}$ that satisfies the above constraints is $(1.7, 0.2, 1.6)$, which in consequence gives $\tau^Z = L(\bar{\alpha}, \bar{\lambda}) = 0.7$ (the objective value of the optimization problem P). Thus

$$\hat{Z} = ((1.7, 0.2, 1.6), 0.7, 1 \times [-1, 0.5] \times [0, 1])$$

is a mub of $\hat{X}$ and $\hat{Y}$; the perturbation 0.7 is the least perturbation possible for any upper bounds of $\hat{X}$ and $\hat{Y}$.

So far, we have characterized $\bar{\alpha}$ by a system of liner equations, given by Propositions 6.2.26 and 6.2.24, that the components of $\bar{\alpha}$ have to satisfy. We can use any linear solver to pick up a solution (which exists, because a saddle-point exists). The pseudo-algorithm 2 summarizes the steps needed to compute a minimal upper bound of two given CAF. The routine `matrixOf` used in lines 1 and 2 computes the matrices M^X and M^Y related to Φ_ϵ^X and Φ_ϵ^Y respectively. Routines `init` and `solve` (lines 13 and 14 respectively) are used to initialize then solve the problem (P).

Algorithm 2: Computing a mub

input : Two CAF $\hat{X} = (\alpha^X, \tau^X, \Phi_\epsilon^X)$ and $\hat{Y} = (\alpha^Y, \tau^Y, \Phi_\epsilon^Y)$.
output: A CAF $\hat{Z}$ mub of $\hat{X}$ and $\hat{Y}$.

1 $M^X \longleftarrow$ `matrixOf`(Φ_ϵ^X);
2 $M^Y \longleftarrow$ `matrixOf`(Φ_ϵ^Y);
3 **if** $\|M^X(\alpha^X - \alpha^Y)\|_1 \leq \tau^Y - \tau^X$ **then**
4 $\quad \Phi_\epsilon^Z \longleftarrow \Phi_\epsilon^Y$;
5 $\quad \bar{\alpha} \longleftarrow \alpha^Y$;
6 $\quad objval \longleftarrow \tau^Y$;
7 **else if** $\|M^Y(\alpha^X - \alpha^Y)\|_1 \leq \tau^X - \tau^Y$ **then**
8 $\quad \Phi_\epsilon^Z \longleftarrow \Phi_\epsilon^X$;
9 $\quad \bar{\alpha} \longleftarrow \alpha^X$;
10 $\quad objval \longleftarrow \tau^X$;
11 **else**
12 $\quad \Phi_\epsilon^Z \longleftarrow \Phi_\epsilon^X \cup \Phi_\epsilon^Y$;
13 $\quad$ `init`$(\text{(P)}, \alpha^X, \alpha^Y, \tau^X, \tau^Y, M^X, M^Y)$;
14 $\quad (objval, \bar{\lambda}, u_{\bar{\lambda}}) \longleftarrow$ `solve` (P); `/* defined in algorithm 1 */`
15 $\quad \bar{\alpha} \longleftarrow LP((6.2.24), (6.2.26))$; `/* use a LP Solver */`
16 **return** $(\bar{\alpha}, objval, \Phi_\epsilon^Z)$;

Handling Reduced Intervals

If the interval concretisation of one noise symbol is reduced to a point, then the deviation of that interval is zero and matrix M^X is no longer non-singular. Thus we can not apply immediately our previously detailed mub computation. On the other hand, we may lose some relations if we replace the noise symbol by its unique value. We detail the latter remark in the following example, explain how we would like to handle these particular

cases, and finally clarify how our mub computation can be extended to these cases with almost no extra effort.

6.2.27 Example
Suppose we have two CAF, $\hat{X} = ((1,-1),0,1\times[1,1])$*, and* $\hat{Y} = ((2,1),0,1\times[-1,-1])$*. We would like to compute a minimal upper bound of* $\hat{X}$ *and* $\hat{Y}$*.*

Since ϵ_1 has a unique value in both affine forms, they can be simplified by considering the actual value of the noise symbol. This gives $\hat{X}' = ((0,0),0,1\times[1,1])$, and $\hat{Y}' = ((1,0),0,1\times[-1,-1])$. In fact, with respect to the order $\leq_{1\times 2}$, $\hat{X} \sim \hat{X}'$, and $\hat{Y} \sim \hat{Y}'$. The order is insensitive to such subtleties, it does not make difference between $\hat{X}$ and $\hat{X}'$, or $\hat{Y}$ and $\hat{Y}'$, or any similar case involving reduced intervals. However, considering these fixed noise symbols rather than replacing them by their respective values increases the accuracy of computations. The unique mub obtained using algorithm 2 for $\hat{X}'$ and $\hat{Y}'$ is $\hat{Z}' = (0.5, 0.5, 1\times[-1,1])$. The minimal perturbation for these affine forms is 0.5. Moreover, the relation with ϵ_1 is lost: the central part has only a center equal to 0.5 without any dependency to ϵ_1.

However, we could hope for a better result, which keeps the relation and at the same time reduces the perturbation. For instance, consider $\hat{Z} = ((0.5,-0.5),0,1\times[-1,1])$:

- $\hat{Z}$ is an upper bound. Indeed, with respect to $\leq_{1\times 2}$, $\hat{X} \leq \hat{Z}$: $1 \in [-1,1]$, and

$$\delta((1,-1)-(0.5,-0.5) \mid [-1,1]\times[1,1]) = \delta(\begin{pmatrix}1 & 1\\ 0 & 0\end{pmatrix}(0.5,-0.5) \mid \mathfrak{B})$$
$$= \delta(0|\mathfrak{B}) = 0 \leq 0 = \tau^Z - \tau^X \ .$$

  Similarly, $\hat{Y} \leq \hat{Z}$.

- $\hat{Z}$ is a minimal upper bound. Indeed if $\hat{T}$ is an upper bound such that $\hat{T} \leq_{1\times 2} \hat{Z}$, than necessarily, $\tau^Z = \tau^T = 0$ and $|0.5-\alpha_0^T|+|-0.5-\alpha_1^T| = 0$, thus $\hat{T} = \hat{Z}$.

- There is no new perturbation noise symbol, the perturbation of $\hat{Z}$ remains null.

- The affine form of $\hat{Z}$, that is $0.5 - 0.5\epsilon_1$, is exactly equal to the affine form of $\hat{X}$, $1-\epsilon_1$, when $\epsilon_1 = 1$ (both are equal to 0). Likewise, the affine form of $\hat{Z}$, is equal to the affine form of $\hat{Y}$, 1, when $\epsilon_1 = -1$ (both are equal to 1).

Observe also that $\hat{Z} \leq_{1\times 2} \hat{Z}'$. In fact, $\hat{Z}'$ is a mub of $\hat{X}'$ and $\hat{Y}'$, but is only an upper bound (it is not minimal) of $\hat{X}$ and $\hat{Y}$. Whereas $\hat{Z}$ is a mub of $\hat{X}$ and $\hat{Y}$. The affine set $\hat{X}'$ can be considered as an over-approximation of $\hat{X}$ (even if both are equivalent with respect to $\leq_{1\times 2}$) as the symbol ϵ_1 gets lost.

We present in the sequel how to handle these reduced intervals, without losing their respective noise symbols while keeping our specification and algorithm detailed in the previous section.

Matrix M related to a box Φ_ϵ, is defined as previously except that now, the deviation of a reduced interval $\boldsymbol{\epsilon_i}$ is set by convention to -1. Observe that this convention leaves matrix M non-singular. To absorb the effect of this "-1" during the computation, the ith component of the unit ball $\mathfrak{B}$ is set to 0 instead of $[-1, 1]$. Thus, to a CAS $\hat{X}$, we associate $\mathfrak{B}^X$. Given two CAS $\hat{X}$ and $\hat{Y}$, the set of indices such that the coordinates of $\mathfrak{B}^X$ are null and $\mathfrak{B}^Y$ are not null is denoted by I^X. The set of indices such that the coordinates of $\mathfrak{B}^Y$ are null and $\mathfrak{B}^X$ are not is denoted by I^Y. The set of indices of coordinates such that both $\mathfrak{B}^X$ and $\mathfrak{B}^Y$ are null is denoted by I^{XY}. Finally, the set of indices of coordinates such that both $\mathfrak{B}^X$ and $\mathfrak{B}^Y$ are not null is denoted by J. For instance, in example 6.2.27, matrix M^X related to $1 \times [1, 1]$, is $\left(\begin{smallmatrix} 1 & 1 \\ 0 & -1 \end{smallmatrix}\right)$, whereas $\mathfrak{B}^X = [-1, 1] \times 0$.

Saddle-points Theorem 6.2.20 remains unchanged as the matrices M^X and M^Y are non-singular. One just needs to replace $\mathfrak{B}$ by $\mathfrak{B}^X$ when $\hat{X}$ is involved and do the same thing for $\hat{Y}$. For instance, instead of $\lambda M^{X*}\mathfrak{B} \cap (1-\lambda)M^{Y*}\mathfrak{B}$, we obtain $\lambda M^{X*}\mathfrak{B}^X \cap (1-\lambda)M^{Y*}\mathfrak{B}^Y$.

Problem (P) needs to consider whether or not the coordinate of $\mathfrak{B}^X$ and $\mathfrak{B}^Y$ are null. So in Proposition 6.2.21: if the $i \in I^X$, then $\bar{u}_i = \bar{u}_0 \operatorname{mid}(\boldsymbol{\epsilon_i^X})$, similarly $\bar{u}_i$ is enforced to $\bar{u}_0 \operatorname{mid}(\boldsymbol{\epsilon_i^Y})$ if $i \in I^Y$. If $i \in I^{XY}$, then both conditions hold, which makes $\bar{u}_0$ null if $\operatorname{mid}(\boldsymbol{\epsilon_i^X}) \neq \operatorname{mid}(\boldsymbol{\epsilon_i^Y})$.

Problem (P_r) is in consequence also affected. It keeps the same generic formulation but with more constraints for u_0 and λ: $|u_0| \leq \beta\lambda$ and $|u_0| \leq \theta(1-\lambda)$, where,

$$\beta \stackrel{\text{def}}{=} \min\{1, \left(\frac{\operatorname{dev}(\boldsymbol{\epsilon_i^Y})}{|\operatorname{mid}(\boldsymbol{\epsilon_i^X}) - \operatorname{mid}(\boldsymbol{\epsilon_i^Y})|}\right)_{i\in I^x}\}(1-\lambda),$$

$$\theta \stackrel{\text{def}}{=} \min\{1, \left(\frac{\operatorname{dev}(\boldsymbol{\epsilon_i^X})}{|\operatorname{mid}(\boldsymbol{\epsilon_i^X}) - \operatorname{mid}(\boldsymbol{\epsilon_i^Y})|}\right)_{i\in I^Y}\}\lambda \ .$$

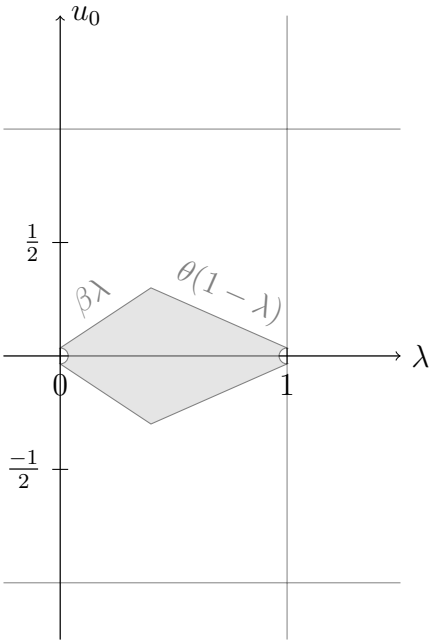

Figure 6.7: The feasible region of the extended optimization problem (P_r). The points $(0,0)$ and $(1,0)$ are not feasible.

Since for $i \in I^X \cup I^Y \cup I^{XY}$, $\bar{u}_i$ is well known (function of $\bar{u}_0$), the coefficient of u_0 in the objective function of (P_r) becomes

$$(\alpha_0^X - \alpha_0^Y) + \sum_{i \in I^X} (\alpha_i^X - \alpha_i^Y)\,\mathrm{mid}(\boldsymbol{\epsilon_i^X}) + \sum_{i \in I^Y} (\alpha_i^X - \alpha_i^Y)\,\mathrm{mid}(\boldsymbol{\epsilon_i^Y}) + \sum_{i \in I^{XY}} (\alpha_i^X - \alpha_i^Y)\,\mathrm{mid}(\boldsymbol{\epsilon_i^X}) \ .$$

The solving algorithm 1 remains unchanged. The feasible region is no longer a perfect diamond minus $(0,0)$ and $(1,0)$, but a diamond like shape (Figure 6.7). The slopes of the borders are β and θ.

Deducing $\bar{\alpha}$ Proposition 6.2.25 is valid for all coordinates $i \notin I^X$ for $M^X(\alpha^X - \bar{\alpha})$, and for all $i \notin I^Y$ for $M^Y(\bar{\alpha} - \alpha^Y)$. For $i \in I^X$ (resp. I^Y), the fact that the ith coordinate of $\mathfrak{B}^X$ (resp. $\mathfrak{B}^Y$) is null adds no constraint for $\bar{\alpha}$ to respect. Proposition 6.2.24 remains unchanged.

We apply the pseudo-algorithm 2 to our motivating example 6.2.27, with respect to the adds detailed above. We have $\mathfrak{B}^X = \mathfrak{B}^Y = [-1,1] \times 0$. By Theorem 6.2.20 we have to solve (P). We have $I^X = I^Y = J = \emptyset$,

$I^{XY} = \{1\}$. Since, $\text{mid}(\boldsymbol{\epsilon}_1^{\boldsymbol{X}}) = 1 \neq -1 = \text{mid}(\boldsymbol{\epsilon}_1^{\boldsymbol{Y}})$, then $\bar{u}_0 = \bar{u}_1 = 0$, the objective function of (P_r) is zero (so the perturbation of the mub is also null), and $\bar{\lambda} \in]0, 1[$. By Propositions 6.2.25, $\bar{\lambda}^{-1} M^{X*^{-1}} u_{\bar{\lambda}}$ is a sign vector of $M^X(\alpha^X - \bar{\alpha})$, since $u_{\bar{\lambda}}$ is null, the sign vector is null, for all $i \notin I^X$ the ith component of $M^X(\alpha^X - \bar{\alpha})$ is null. Similarly for $(1 - \bar{\lambda})^{-1} M^{Y*^{-1}} u_{\bar{\lambda}}$ which is also null. This gives the system:

$$\begin{cases} 1 - \alpha_0^Z + (-1 - \alpha_1^Z) & = 0 \\ \alpha_0^Z - 2 - (\alpha_1^Z - 1) & = 0 \end{cases}$$

Proposition 6.2.24 adds no more constraints for $\bar{\alpha}$ to respect. The system above gives $\alpha_0^Z = 0.5$ and $\alpha_1^Z = -0.5$, and $\hat{Z} = ((0.5, -0.5), 0, 1 \times [-1, 1])$, which is the mub announced earlier.

Application: Join over Perturbed Affine Forms

The Perturbed Affine Sets introduced by Goubault and Putot [GP08, GP09] are a particular case of Constrained Affine Sets, where noise symbols are abstracted by the unit box $\mathfrak{B}$, that is all noise symbols lie always within the interval $[-1, 1]$. The join operator defined over Perturbed Affine Forms [GP08] and Perturbed Affine Sets are revisited here as an application of our results established for CAS.

6.2.28 Definition
A Perturbed Affine Set, or PAS, is a CAS $\hat{X} = (C^X, P^X, \Phi^X)$, *where* $\Phi^X = \mathfrak{B}$.

Since Φ^X is always equal to $\mathfrak{B}$ and is independent from the abstract object $\hat{X}$, we denote a PAS by the pair of matrices (C^X, P^X). Moreover, matrix M^X related to $\Phi^X = \mathfrak{B}$ is the identity matrix I_{n+1}.

By Lemma 4.3.3, the order relation $\leq_{1\times 2}$ gives:

6.2.29 Proposition (Partial Order over PAS)
Given two PAS, $\hat{X} = (C^X, P^X)$ *and* $\hat{Y} = (C^Y, P^Y)$, *we have* $\hat{X} \leq_{1\times 2} \hat{Y}$ *if and only if*

$$\forall t \in \mathbb{R}^p, \quad \delta(t \mid (C^X - C^Y)\mathfrak{B}) \leq \delta(t \mid P^Y\mathfrak{B}) - \delta(t \mid P^X\mathfrak{B}) \ .$$

Proof. *Immediate application of Lemma 4.3.3 for* $\Phi^X = \Phi^Y = \mathfrak{B}$. ∎

As already mentioned, $\delta(x \mid \mathfrak{B}) = \|x\|_1$. The reformulation of Proposition 6.2.29 using this identity gives

$$\forall t \in \mathbb{R}^p, \quad \|(C^X - C^Y)^* t\|_1 \leq \|P^{Y^*} t\|_1 - \|P^{X^*} t\|_1 \ .$$

which is exactly the order defined in [GP09, Definition 2], up to the transpose operation of matrices $(C^X - C^Y)$, P^X and P^Y. The set of saddle-points in this particular case is deduced from Theorem 6.2.20.

6.2.30 Corollary

- *If* $\delta(\alpha^Y - \alpha^X \mid \mathfrak{B}) < \tau^Y - \tau^X$, *then* $(\alpha^Y, 0)$ *is the unique saddle-point of* L. *Its saddle-value is* τ^Y.
- *If* $\delta(\alpha^Y - \alpha^X \mid \mathfrak{B}) = \tau^Y - \tau^X$, *and* $\alpha^Y \neq \alpha^X$, *then* L *admits infinitely many saddle-points such that* $\bar{\alpha} = \alpha^y$. *Its saddle-value is* τ^Y.
- *If* $\delta(\alpha^Y - \alpha^X \mid \mathfrak{B}) < \tau^X - \tau^Y$, *then* $(\alpha^X, 1)$ *is the unique saddle-point of* L. *Its saddle-value is* τ^X.
- *If* $\delta(\alpha^Y - \alpha^X \mid \mathfrak{B}) = \tau^X - \tau^Y$, *and* $\alpha^Y \neq \alpha^X$, *then* L *admits infinitely many saddle-points such that* $\bar{\alpha} = \alpha^X$. *Its saddle-value is* τ^X.
- *Otherwise,* $\delta(\alpha^Y - \alpha^X \mid \mathfrak{B}) > |\tau^Y - \tau^X|$, *and* $(\bar{\alpha}, \bar{\lambda})$ *satisfies* $\bar{\lambda} \in]0, 1[$, *and*

  i) $\delta(\bar{\alpha} - \alpha^X \mid \mathfrak{B}) + \tau^X = \delta(\bar{\alpha} - \alpha^Y \mid \mathfrak{B}) + \tau^Y$,

  ii) $\bar{\lambda}\delta(\bar{\alpha} - \alpha^X \mid \mathfrak{B}) + (1 - \bar{\lambda})\delta(\bar{\alpha} - \alpha^Y \mid \mathfrak{B}) = \delta(\alpha^X - \alpha^Y \mid \bar{\lambda}\mathfrak{B} \cap (1 - \bar{\lambda})\mathfrak{B})$.

Proof. *Apply Theorem 6.2.20 for* $M^X = M^Y = I_{n+1}$. ∎

When $\delta(\alpha^Y - \alpha^X \mid \mathfrak{B}) > |\tau^Y - \tau^X|$, we establish that the value of $\bar{\lambda}$ is necessarily equal to $\frac{1}{2}$, which permits to compute the saddle-value.

6.2.31 Proposition

If $\delta(\alpha^Y - \alpha^X \mid \mathfrak{B}) > |\tau^Y - \tau^X|$, *then* $\bar{\lambda} = \frac{1}{2}$, *and* $\bar{\alpha}$ *satisfies:*

$$\begin{aligned} &i) \quad \delta(\bar{\alpha} - \alpha^X \mid \mathfrak{B}) + \tau^X = \delta(\bar{\alpha} - \alpha^Y \mid \mathfrak{B}) + \tau^Y, \\ &ii) \quad \delta(\bar{\alpha} - \alpha^X \mid \mathfrak{B}) + \delta(\bar{\alpha} - \alpha^Y \mid \mathfrak{B}) = \delta(\alpha^X - \alpha^Y \mid \mathfrak{B}) \ . \end{aligned}$$

The saddle-value of L *is equal to*

$$\frac{1}{2}(\delta(\alpha^X - \alpha^Y \mid \mathfrak{B}) + \tau^X + \tau^Y)$$

.

Proof. *Suppose that* $\bar{\lambda} = \frac{1}{2}$*, the system of equations satisfied by* $\bar{\alpha}$ *is the same of Corollary 6.2.30 for* $\bar{\lambda} = \frac{1}{2}$*. The saddle-value,* $L(\bar{\alpha}, \bar{\lambda})$*, is equal to (by Equation 6.2.1)):*

$$\begin{aligned} L(\bar{\alpha}, \bar{\lambda}) &= \frac{1}{2}(\delta(\bar{\alpha} - \alpha^X \mid \mathfrak{B}) + \tau^X) + \frac{1}{2}(\delta(\bar{\alpha} - \alpha^Y \mid \mathfrak{B}) + \tau^Y) \\ &= \frac{1}{2}(\delta(\bar{\alpha} - \alpha^X \mid \mathfrak{B}) + \delta(\bar{\alpha} - \alpha^Y \mid \mathfrak{B}) + \tau^Y + \tau^X) \text{ (Corol. 6.2.30, } \imath) \\ &= \frac{1}{2}(\delta(\alpha^X - \alpha^Y \mid \mathfrak{B}) + \tau^Y + \tau^X)) \qquad \text{(Corol. 6.2.30, } \imath\imath) \end{aligned}$$

It remains to prove that $\bar{\lambda} = \frac{1}{2}$*. The set* $\bar{\lambda}\mathfrak{B} \cap (1-\bar{\lambda})\mathfrak{B}$ *is equal to* $\min\{\bar{\lambda}, (1-\bar{\lambda})\}\mathfrak{B}$*. The constant* $\min\{\bar{\lambda}, (1-\bar{\lambda})\}$ *is positive, using the fact that if* $\lambda \geq 0$*,* $\delta(x \mid \lambda C) = \lambda\delta(x \mid C)$ *(see for instance [Roc70, Theorem 16.1.1]), we then have*

$$\delta(\alpha^X - \alpha^Y \mid \bar{\lambda}\mathfrak{B} \cap (1-\bar{\lambda})\mathfrak{B}) = \min\{\bar{\lambda}, (1-\bar{\lambda})\}\delta(\alpha^X - \alpha^Y \mid \mathfrak{B}),$$

we next use the triangle inequality of the support function:

$$\begin{aligned} &\min\{\bar{\lambda}, (1-\bar{\lambda})\}\delta(\alpha^X - \alpha^Y \mid \mathfrak{B}) \\ &\quad \leq \min\{\bar{\lambda}, (1-\bar{\lambda})\}\delta(\alpha^X - \bar{\alpha} \mid \mathfrak{B}) + \min\{\bar{\lambda}, (1-\bar{\lambda})\}\delta(\bar{\alpha} - \alpha^Y \mid \mathfrak{B}) \ . \end{aligned}$$

By Corollary 6.2.30, equation $\imath\imath$*),* $\delta(\alpha^X - \alpha^Y \mid \bar{\lambda}\mathfrak{B} \cap (1-\bar{\lambda})\mathfrak{B})$ *is equal to* $\bar{\lambda}(\delta(\alpha^X - \bar{\alpha} \mid \mathfrak{B})) + (1-\bar{\lambda})(\delta(\alpha^Y - \bar{\alpha} \mid \mathfrak{B}))$*, thus the inequality becomes:*

$$\begin{aligned} &(\bar{\lambda} - \min\{\bar{\lambda}, (1-\bar{\lambda})\})\delta(\alpha^X - \bar{\alpha} \mid \mathfrak{B}) \\ &\qquad + ((1-\bar{\lambda}) - \min\{\bar{\lambda}, (1-\bar{\lambda})\})\delta(\bar{\alpha} - \alpha^Y \mid \mathfrak{B}) \leq 0, \quad (6.2.4) \end{aligned}$$

which makes the sum of two positive terms non-positive, therefore each term is necessarily equal to zero:

$$(\bar{\lambda} - \min\{\bar{\lambda}, (1-\bar{\lambda})\})\delta(\alpha^X - \bar{\alpha} \mid \mathfrak{B}) = 0 \qquad (6.2.5)$$

$$((1-\bar{\lambda}) - \min\{\bar{\lambda}, (1-\bar{\lambda})\})\delta(\bar{\alpha} - \alpha^Y \mid \mathcal{B}^{1+n}) = 0 \qquad (6.2.6)$$

The equation (6.2.5) gives $\bar{\lambda} = \min\{\bar{\lambda}, (1-\bar{\lambda})\}$ *(that is* $\bar{\lambda} \leq \frac{1}{2}$*) or* $\bar{\alpha} = \alpha^X$*. If* $\bar{\alpha} = \alpha^X$*, then Corollary 6.2.30, equation* $\imath$*), contradicts the hypothesis* $\delta(\alpha^Y - \alpha^X \mid \mathfrak{B}) > |\tau^X - \tau^Y|$*, as it makes* $\delta(\alpha^Y - \alpha^X \mid \mathfrak{B}) = \tau^X - \tau^Y$*.*

Likewise, the equation (6.2.6) gives $(1-\bar{\lambda}) = \min\{\bar{\lambda}, (1-\bar{\lambda})\}$ *(that is* $\bar{\lambda} \geq \frac{1}{2}$*) or* $\bar{\alpha} = \alpha^Y$*. The latter also contradicts the hypothesis.*

Finally, $\bar{\lambda} \geq \frac{1}{2}$ *and* $\bar{\lambda} \leq \frac{1}{2}$*, that is* $\bar{\lambda} = \frac{1}{2}$*.* ∎

Instead of proving the result directly, we could also use Algorithm 1, which gives the values of $\bar{\lambda}$ in the general case. We show that it gives also this unique value for $\bar{\alpha}$. Here, all noise symbols are within $[-1, 1]$, so we have a unique line L_i (see Definition L_i):

$$L_i = \{(\bar{\lambda}, u_0) \mid \bar{\lambda} = 1 - \bar{\lambda}\} \ .$$

The set of points V is equal to $\{(\frac{1}{2}, \frac{1}{2}), (\frac{1}{2}, \frac{-1}{2})\}$. The algorithm ends with $\bar{\lambda} = \frac{1}{2}$, $\bar{u}_i = \frac{\text{sign}(\alpha_i^X - \alpha_i^Y)}{2}$, $0 \leq i \leq n$, and $\frac{1}{2}(\delta(\alpha^X - \alpha^Y \mid \mathfrak{B}) + \tau^X + \tau^Y)$ as objective value, which is equal to the saddle-value found in Proposition 6.2.31.

We can now deduce $\bar{\alpha}$ using Propositions 6.2.24 and 6.2.25.

6.2.32 Proposition
The vector $\bar{\alpha}$ satisfies,

$$\langle \bar{\alpha}, \bar{u}_{\bar{\lambda}} \rangle = \langle \frac{\alpha^Y + \alpha^X}{2}, \bar{u}_{\bar{\lambda}} \rangle + \frac{\tau^X - \tau^Y}{4},$$

where $\bar{u}_{\bar{\lambda}} = \frac{1}{2}\text{sign}(\alpha^X - \alpha^Y)$ (the sign vector of $\alpha^X - \alpha^Y$).

Its coordinates respect

$$\forall i, 0 \leq i \leq n, \min\{\alpha_i^X, \alpha_i^Y\} \leq \bar{\alpha}_i \leq \max\{\alpha_i^X, \alpha_i^Y\} \ .$$

Proof. *Proposition 6.2.24, with $\bar{\lambda} = \frac{1}{2}$, determines immediately the first equation. Moreover, since $M^X = M^Y = I_n$, and $\bar{u}_i = \frac{\text{sign}(\alpha_i^X - \alpha_i^Y)}{2}$, Proposition 6.2.25 gives:*

$$\begin{aligned} \text{sign}(\alpha_i^X - \alpha_i^Y) = 1 &\implies \alpha_i^X - \bar{\alpha}_i \geq 0 \wedge \bar{\alpha}_i - \alpha_i^Y \geq 0 \\ \text{sign}(\alpha_i^X - \alpha_i^Y) = -1 &\implies \alpha_i^X - \bar{\alpha}_i \leq 0 \wedge \bar{\alpha}_i - \alpha_i^Y \leq 0 \ . \end{aligned}$$

Moreover, there is always an $\bar{\alpha}$ which respects these constraints, as we have proved that a saddle-point exists. ∎

Propositions 6.2.31 and 6.2.32 are equivalent to [GP08, Proposition 20].

Among all possible solutions, Goubault and Putot in [GP08], picked up the one which minimizes the interval concretisation of the perturbed affine form. The minimal interval concretisation of any upper bound is the join of the interval concretisations of the involved operands. This gives two additional constraints that $\bar{\alpha}$ needs to respect:

$$\begin{aligned} \alpha_0^Z &= \text{mid}([\hat{X}] \cup [\hat{Y}]), \\ \sum_{i=1}^{n} |\bar{\alpha}_i| + L(\bar{\alpha}, \bar{\lambda}) &= \text{dev}([\hat{X}] \cup [\hat{Y}]) \ . \end{aligned}$$

Recall that $[\hat{X}]$ (resp. $[\hat{Y}]$) denotes the interval concretisation of $\hat{X}$ (resp. $\hat{Y}$), and that the interval concretisation of (α, τ) is given by

$$[\alpha_0 - \sum_{i=1}^{n} |\alpha_i| - \tau, \alpha_0 + \sum_{i=1}^{n} |\alpha_i| + \tau] \ .$$

The authors prove in [GP08] that such solution exists and is unique whenever the intervals $[\hat{X}]$ and $[\hat{Y}]$ are in generic position. Two intervals $\boldsymbol{i}$ and $\boldsymbol{j}$ are said to be in generic position if ($\boldsymbol{i} \subseteq \boldsymbol{j}$ or $\boldsymbol{j} \subseteq \boldsymbol{i}$) imply ($\sup(\boldsymbol{i}) = \sup(\boldsymbol{j})$ or $\inf(\boldsymbol{i}) = \inf(\boldsymbol{j})$).

When the intervals $[\hat{X}]$ and $[\hat{Y}]$ are in generic position, the solution is obtained by minimizing each term of the sum $\sum_{i=1}^{n} |\bar{\alpha}_i|$:

$$\forall i, 1 \leq i \leq n, \quad \bar{\alpha}_i = \operatorname{argmin}(\alpha_i^X, \alpha_i^Y),$$

where the argmin operator is defined as follows:

$$\operatorname{argmin}(\alpha_1, \alpha_2) := \{\alpha \in [\min(\alpha_1, \alpha_2), \max(\alpha_1, \alpha_2)], |\alpha| \text{ is minimal } \},$$

that is, if $\alpha_1 \alpha_2 \leq 0$, then $\operatorname{argmin}(\alpha_1, \alpha_2) = 0$, else if both are positives then $\operatorname{argmin}(\alpha_1, \alpha_2) = \min\{\alpha_1, \alpha_2\}$, else (both are negatives) $\operatorname{argmin}(\alpha_1, \alpha_2) = \max\{\alpha_1, \alpha_2\}$.

If, however, the interval concretisations are not in generic position, the uniqueness is no more guaranteed, as shown in the following example. Moreover, the argmin solution may be not admissible.

6.2.33 Example
Let $\hat{X} = ((1, 1, 2, 1), 0)$, *and* $\hat{Y} = ((-2, -6, 1, 2), 0)$. *We have* $[\hat{X}] = [-3, 5]$, *and* $[\hat{Y}] = [-11, 7]$, *thus the interval concretisations are not in generic position. The condition* $\delta(\alpha^X - \alpha^Y \mid \mathfrak{B}) \geq |\tau^Y - \tau^X|$ *is satisfied, since* $|\tau^Y - \tau^X| = 0$, *and* $\delta(\alpha^X - \alpha^Y \mid \mathfrak{B}) = 12$. *By Proposition 6.2.31,* $\bar{\tau} = L(\bar{\alpha}, \bar{\lambda}) = \frac{12}{2} = 6$, *and by Proposition 6.2.32,* $\bar{\alpha}_0 + \bar{\alpha}_1 + \bar{\alpha}_2 - \bar{\alpha}_3 = -3$, *and*

$$\bar{\alpha} \in [-2, 1] \times [-6, 1] \times [1, 2] \times [1, 2] \ .$$

The argmin *solution gives then* $\bar{\alpha}_1 = 0$, $\bar{\alpha}_2 = 1$, *and* $\bar{\alpha}_3 = 1$, *which makes* $\bar{\alpha}_0 = -3 \notin [-2, 1]$. *So here, the* argmin *operator is too strong to respect the constraints of Proposition 6.2.32. Still, any vector* $(-2, \bar{\alpha}_1, \bar{\alpha}_2, \bar{\alpha}_3)$ *such that* $|\bar{\alpha}_1| + \bar{\alpha}_2 + \bar{\alpha}_3 = 3$, $\bar{\alpha}_1 + \bar{\alpha}_2 - \bar{\alpha}_3 = -1$, $-6 \leq \bar{\alpha}_1 \leq 1$, $1 \leq \bar{\alpha}_2 \leq 2$, *and* $1 \leq \bar{\alpha}_3 \leq 2$ *is a mub of* $\hat{X}$ *and* $\hat{Y}$ *with a minimal interval concretisation.*

We define the operator $\vee$ over two perturbed affine forms (PAF), as follows:

6.2.34 Definition
Let $\hat{X} = (\alpha^X, \tau^X)$ *and* $\hat{Y} = (\alpha^Y, \tau^Y)$ *be two PAF.*

$$\hat{Z} = \hat{X} \vee \hat{Y} \overset{\text{def}}{\iff} \begin{cases} \alpha_0^Z = \text{mid}([\hat{X}] \cup [\hat{Y}]) \\ \forall i \geq 1, \alpha_i^Z = \text{argmin}(\alpha_i^X, \alpha_i^Y), \\ \tau^Z = \text{dev}([\hat{X}] \cup [\hat{Y}]) - \sum_{i=1}^n |\alpha_i^Z| \end{cases}$$

The operator $\vee$ has many advantages: in general it gives an upper bound of $\hat{X}$ and $\hat{Y}$; if $[\hat{X}]$ and $[\hat{Y}]$ are in generic position, then it returns exactly the unique mub of $\hat{X}$ and $\hat{Y}$; the interval concretisation of $\hat{X} \vee \hat{Y}$ is minimal, it is indeed equal to $[\hat{X}] \cup [\hat{Y}]$; the computation is a linear function of n, the number of noise symbols involved.

We would like to define a similar operator over CAF. We stress first a major difference between the unconstrained and the constrained cases.

For CAF, we have seen that, among all upper bounds, it is sufficient to have minimal perturbation to be a mub. This sufficient condition is also necessary in the unconstrained case [GP08, Lemma 18].

6.2.35 Proposition
Given two PAF, $\hat{X}$ *and* $\hat{Y}$*, then* $\hat{Z}$ *is a mub of* $\hat{X}$ *and* $\hat{Y}$ *if and only if it is an upper bound and* τ^Z *is minimal among all* τ^T*, perturbation of any upper bound* $\hat{T}$*.*

Thus, in the unconstrained case, we can always find a mub which minimizes the perturbation and has a minimal concretisation, that is its interval concretisation is the join of the interval concretisations of its operands. Whereas in the constrained case, being a mub is not equivalent to minimizing the perturbation. The mub that minimizes the perturbation does not have in general the minimal interval concretisation, and dually, the mub that has the minimal interval concretisation does not have the minimal perturbation. (See for instance Example 6.2.3.)

Thus, in general, in the constrained case, computing the set of mubs that minimize the perturbation, then enforcing the minimality of the interval concretisation may give an empty set of solutions.

Efficient Upper Bound Computation

As we have detailed earlier, the $\vee$ operator (Definition 6.2.34) which uses the argmin solution, in the unconstrained case, has several advantages. We define a similar upper bound operator over CAF.

6.2.36 Proposition
Let $\hat{X} = (\alpha^X, \tau^X, \Phi_\epsilon^X)$ and $\hat{Y} = (\alpha^Y, \tau^Y, \Phi_\epsilon^Y)$ be two CAF. Let $\hat{Z} = \hat{X} \vee \hat{Y}$, be defined by $\hat{X}$ (resp. $\hat{Y}$) if $\hat{Y} \leq_{1\times 2} \hat{X}$ (resp. $\hat{X} \leq_{1\times 2} \hat{Y}$), and else :

- $\forall i \geq 1, \quad \alpha_i^Z = \operatorname{argmin}(\alpha_i^X, \alpha_i^Y)$,
- *if* $|c^X - c^Y| \leq d^Y - d^X$, *then* $\alpha_0^Z = c^X$ *and* $\tau^Z = d^X$,
- *if* $|c^X - c^Y| \leq d^X - d^Y$, *then* $\alpha_0^Z = c^Y$ *and* $\tau^Z = d^Y$,
- *if* $|c^X - c^Y| > d^Y - d^X$, *then*

$$\alpha_0^Z = \frac{c^X + c^Y}{2} + \frac{d^X - d^Y}{2}\operatorname{sign}(c^X - c^Y)$$
$$\tau^Z = \frac{1}{2}(|c^X - c^Y| + d^X + d^Y)$$

where,

$$c^X \stackrel{\text{def}}{=} \alpha_0^X - \sum_{i=1}^{n}(\alpha_i^Z - \alpha_i^X)\operatorname{mid}(\boldsymbol{\epsilon_i^X}), \quad c^Y \stackrel{\text{def}}{=} \alpha_0^Y - \sum_{i=1}^{n}(\alpha_i^Z - \alpha_i^Y)\operatorname{mid}(\boldsymbol{\epsilon_i^Y}),$$
$$d^X \stackrel{\text{def}}{=} \sum_{i=1}^{n}|\alpha_i^Z - \alpha_i^X|\operatorname{dev}(\boldsymbol{\epsilon_i^X}) + \tau^X, \quad d^Y \stackrel{\text{def}}{=} \sum_{i=1}^{n}|\alpha_i^Z - \alpha_i^Y|\operatorname{dev}(\boldsymbol{\epsilon_i^Y}) + \tau^Y,$$
$$\boldsymbol{\epsilon_i^X} = \operatorname{bound}_2(\epsilon_i, \Phi_\epsilon^X),$$
$$\boldsymbol{\epsilon_i^Y} = \operatorname{bound}_2(\epsilon_i, \Phi_\epsilon^Y) \ .$$

Then, $\hat{Z}$ is an upper bound of $\hat{X}$ and $\hat{Y}$.

Proof. *By construction, $\hat{X} \vee \hat{Y}$ is an upper bound of $\hat{X}$ and $\hat{Y}$. This is obvious when $\hat{X}$ and $\hat{Y}$ are comparable. Otherwise, we set α_i^Z, $1 \leq i \leq n$ to $\operatorname{argmin}(\alpha_i^X, \alpha_i^Y)$, then compute α_0^Z and τ^Z such as $\hat{Z}$ is an upper bound of $\hat{X}$ and $\hat{Y}$ and τ^Z is minimal. For $\hat{Z}$ to be an upper bound, (α^Z, τ^Z) needs to satisfy (M^X and M^Y are the matrices related to Φ^X and Φ^Y):*

$$\|M^X(\alpha^Z - \alpha^X)\|_1 \leq \tau^Z - \tau^X$$
$$\|M^Y(\alpha^Z - \alpha^Y)\|_1 \leq \tau^Z - \tau^Y,$$

or equivalently,

$$|\alpha_0^Z - c^X| \leq \tau^Z - d^X$$
$$|\alpha_0^Z - c^Y| \leq \tau^Z - d^Y \ .$$

Now, we compute τ^Z and α_0^Z such that τ^Z is minimal, which is a special case ($n = 0$) of Corollary 6.2.30. Therefore, the definitions of τ^Z and α_0^Z are immediate from Propositions 6.2.31 and 6.2.32 respectively. ∎

Notice that in general the operator $\vee$ over CAF, does not give a mub, the interval concretisation of $\hat{X} \vee \hat{Y}$ is not minimal in general. Nevertheless, the procedure gives, in linear complexity, an upper bound.

6.3 Join over Constrained Affine Sets

We build a join operator componentwise using the minimal upper bound of each dimension. The domain $\mathcal{A}_2$ is the intervals lattice. We use the canonical representation of CAS.

6.3.1 Proposition
Let $\hat{X} = (C^X, P^X, \Phi_\epsilon^X)$ and $\hat{Y} = (C^Y, P^Y, \Phi_\epsilon^Y)$ be the canonical representatives of two CAS, then $\hat{Z}$ defined by

- $\Phi^Z = \Phi_\epsilon^X \cup_2 \Phi_\epsilon^Y$,
- *$(L_i^{C^Z}, P_{i,i}^Z)$ is a mub of $(L_i^{C^X}, \delta(P^X e_i \mid \mathfrak{B}))$ and $(L_i^{C^Y}, \delta(P^Y e_i \mid \mathfrak{B}))$, where L_i^M denotes the ith line of matrix M and $M_{i,i}$ is the (i,i) component of matrix M; the vector e_i is the ith vector of the canonical base of $\mathbb{R}^m$.*
- *All other components of P^Z are null.*

is an upper bound of $\hat{X}$ and $\hat{Y}$. We denote $\hat{Z}$ by $\hat{X} \sqcup \hat{Y}$.

Proof. *We prove that $\hat{X} \leq_{1\times 2} \hat{Z}$ using the equivalence of Proposition 4.3.7. Firstly, we have $\Phi_\epsilon^X \subseteq \Phi_\epsilon^X \cup \Phi_\epsilon^Y = \Phi_\epsilon^Z$. Secondly, for all $t \in \mathbb{R}^p$, we prove that the inequality*

$$\delta(t \mid (C^X - C^Z)M^{X*}\mathfrak{B}) \leq \delta(t \mid P^Z\mathfrak{B}) - \delta(t \mid P^X\mathfrak{B}),$$

holds. Equivalently, using that $\delta(t \mid \mathfrak{B}) = \|t\|_1$, we write

$$\|M^X(C^X - C^Y)^* t\|_1 \leq \|P^Z t\|_1 - \|P^{X*} t\|_1 \ .$$

An element $P_{(i,i)}^Z$, $1 \leq i \leq p$, by definition satisfies:

$$\delta(M^X(L_i^{C^Z} - L_i^{C^X}) \mid \mathfrak{B}) \leq P_{(i,i)}^Z - \delta(P^X e_i \mid \mathfrak{B}) \ .$$

Observing that $(L_i^{C^Z} - L_i^{C^X}) = (C^Z - C^X)^* e_i$, $P^Z_{(i,i)} = \|P^{Z^*} e_i\|_1$ *and using the norm notation, we obtain*

$$\|M^X (C^Z - C^X)^* e_i\|_1 \leq \|P^{Z^*} e_i\|_1 - \|P^{X^*} e_i\|_1 \ .$$

Let t_i, $1 \leq i \leq p$, *denote the coordinates of* t. *We have:*

$$\begin{aligned}\|M^X (C^X - C^Z)^* t\|_1 + \|P^{X^*} t\|_1 &\leq \sum_{i=1}^{p} |t_i| (\|M^X (C^X - C^Z)^* e_i\|_1 + \|P^{X^*} e_i\|_1) \\ &\leq \sum_{i=1}^{p} |t_i| \|P^{Z^*} e_i\|_1 \\ &= \|P^{Z^*} t\|_1\end{aligned}$$

The first inequality uses the triangle inequality. The last equality is due to the fact that P^Z *is a diagonal matrix. We prove similarly that* $\hat{Y} \leq_{1\times 2} \hat{Z}$, *by substituting* X *and* Y. *Therefore,* $\hat{Z}$ *is an upper bound of* $\hat{X}$ *and* $\hat{Y}$. ∎

The previous upper bound considers the enclosing boxes of the perturbation zonotopes, rather than the zonotopes themselves, then computes a minimal upper bound of the so obtained CAS. Indeed, the join operator defined in Proposition 6.3.1 gives a minimal upper bound of its operands whenever the perturbation zonotopes of $\hat{X}$ and $\hat{Y}$ are simple boxes.

6.3.2 Proposition
Let $\hat{X} = (C^X, P^X, \Phi_\epsilon^X)$ *and* $\hat{Y} = (C^Y, P^Y, \Phi_\epsilon^Y)$ *be the canonical representatives of two CAS such that the perturbation zonotopes* $P^X\mathfrak{B}$ *and* $P^Y\mathfrak{B}$, *are simple boxes (*P^X *and* P^Y *are diagonal matrices). Then* $\hat{X} \sqcup \hat{Y}$ *is the minimal upper bound of* $\hat{X}$ *and* $\hat{Y}$.

Proof. *By Proposition 6.3.1,* $\hat{Z}$ *is an upper bound of* $\hat{X}$ *and* $\hat{Y}$. *Let* $\hat{T}$ *be an upper bound such that* $\hat{T} \leq_{1\times 2} \hat{Z}$, *we prove that* $\hat{T}$ *is necessarily equal to* $\hat{Z}$. *Firstly,* $\Phi_\epsilon^T \subseteq \Phi_\epsilon^Z = \Phi_\epsilon^X \cup \Phi_\epsilon^Y$, *and since* $\hat{T}$ *is an upper bound,* $\Phi_\epsilon^X \cup \Phi_\epsilon^Y \subseteq \Phi_\epsilon^T$, *then necessarily* $\Phi_\epsilon^T = \Phi_\epsilon^Z$. *We next prove that the matrices* P^T *and* P^Z *are equal. For* $t = e_i$, *the* i*th vector of the canonical base of* $\mathbb{R}^p$, *we have* $\|P^{Z^*} e_i\|_1$ *is minimal by construction, and*

$$\|M^X (C^Z - C^T)^* e_i\|_1 \leq \|P^{Z^*} e_i\|_1 - \|P^{T^*} e_i\|_1,$$

thus $\|P^{Z^*} e_i\|_1 = \|P^{T^*} e_i\|_1$ *for all* i, *which makes* $P^Z = P^T$. *In turn this gives* $(C^Z - C^T)^* e_i = 0$ *(*M^X *is non-singular), for all* i. *Thus* $\mathbb{R}^p \subseteq$

$\ker((C^Z - C^T)^*)$. *Rank-nullity Theorem states that the sum of the rank of a matrix and the dimension of its kernel is equal to the number of its columns, that is* $\text{rank}((C^Z - C^T)^*) + \ker((C^Z - C^T)^*) = p$. *This makes* $\text{rank}((C^Z - C^T)^*) = 0$, *and* $C^Z = C^T$. ∎

This approximation loses all relations kept by the perturbed noise symbols in each CAS before evaluating the upper bound. The total number of the perturbation noise symbols after $\sqcup$ operation is p.

We propose a slightly different componentwise upper bound which improves the accuracy of the previous upper bound in that it keeps track to the relations between perturbation noise symbols.

6.3.3 Proposition
Let $\hat{X} = (C^X, P^X, \Phi^X)$ *and* $\hat{Y} = (C^Y, P^Y, \Phi^Y)$ *be the canonical representation of two CAS, then* $\hat{Z}$ *defined by*

- $\Phi^Z = \Phi^X \cup_2 \Phi^Y$,
- $((L_i^{C^Z}, L_i^{P^Z}), P^Z_{i,m+i})$ *is the mub of* $((L_i^{C^X}, L_i^{P^X}), 0)$ *and* $((L_i^{C^Y}, L_i^{P^Y}), 0)$, $1 \leq i \leq p$, *where* (L_i^M, L_i^Q) *denotes the vector formed by concatenating the* i*th line of both matrices* M *and* Q.
- *The components* $P^Z_{i,m+j}$, $1 \leq i, j \leq p$, $i \neq j$ *are null.*

is an upper bound of $\hat{X}$ *and* $\hat{Y}$. *We denote* $\hat{Z}$ *by* $\hat{X} \sqcup^+ \hat{Y}$.

Proof. *We prove that* $\hat{X} \leq_{1\times 2} \hat{Z}$ *using the equivalence of Proposition 4.3.7. Firstly, we have* $\Phi^X_\epsilon \subseteq \Phi^X_\epsilon \cup \Phi^Y_\epsilon = \Phi^Z_\epsilon$. *Secondly, for all* $t \in \mathbb{R}^p$, *we prove that the inequality*

$$\delta(t \mid (C^X - C^Z)M^{X*}\mathfrak{B}) \leq \delta(t \mid P^Z\mathfrak{B}) - \delta(t \mid P^X\mathfrak{B}),$$

holds. Equivalently, using that $\delta(t \mid \mathfrak{B}) = \|t\|_1$, *we write*

$$\|M^X(C^X - C^Y)^*t\|_1 \leq \|P^{Z*}t\|_1 - \|P^{X*}t\|_1 \ .$$

We decompose matrix P^Z *in two blocs* $(R^Z|D^Z)$. *Matrix* D^Z *is a diagonal matrix,* $D^Z_{(i,i)} = P^Z_{(i,m+i)}$, $1 \leq i \leq p$. *We denote by* M *the matrix defined by* 4 *blocs*

$$\begin{pmatrix} M^X & 0 \\ 0 & I_m \end{pmatrix}$$

Since the perturbation noise symbols are within $[-1,1]$*,* $M_{(1,i)}$*,* $i > n+1$*, are null (the center of* $[-1,1]$*), where* $M_{(i,i)}$*,* $i > n+1$*, is equal to* 1 *(the deviation of* $[-1,1]$*). By definition of* $P^Z_{(i,m+i)}$*,* $1 \leq i \leq p$*, we have:*

$$\delta(M((L_i^{C^Z}, L_i^{R^Z}) - (L_i^{C^X}, L_i^{P^X})) \mid \mathfrak{B}) \leq P^Z_{(i,m+i)},$$

or equivalently,

$$\|M((L_i^{C^Z}, L_i^{R^Z}) - (L_i^{C^X}, L_i^{P^X}))\|_1 \leq P^Z_{(i,m+i)} \ .$$

Given a vector $w = (u, v)$*, defined as a concatenation of two vectors* $u \in \mathbb{R}^{n+1}$ *and* $v \in \mathbb{R}^m$*, we have*

$$\|w\|_1 = \sum_{i=0}^{n+m} |w_i| = \sum_{i=0}^{n} |u_i| + \sum_{i=n+1}^{n+1+m} |v_i| = \|u\|_1 + \|v\|_1 \ .$$

Since $Mw = (M^X u, I_m v)$*, then* $\|Mw\|_1 = \|M^X u\|_1 + \|v\|_1$*. We use this property for* $\|M((L_i^{C^Z}, L_i^{R^Z}) - (L_i^{C^X}, L_i^{P^X}))\|_1$*, we obtain*

$$\begin{aligned}\|M((L_i^{C^Z}, L_i^{R^Z}) - (L_i^{C^X}, L_i^{P^X}))\|_1 &= \|M((L_i^{C^Z} - L_i^{C^X}, L_i^{R^Z} - L_i^{P^X}))\|_1 \\ &= \|M^X(L_i^{C^Z} - L_i^{C^X})\|_1 + \|L_i^{R^Z} - L_i^{P^X}\|_1\end{aligned}$$

Using the identities $(L_i^{C^Z} - L_i^{C^X}) = (C^Z - C^X)^* e_i$ *and* $L_i^{R^Z} - L_i^{P^X} = R^{Z^*} e_i - P^{X^*} e_i$*, we obtain:*

$$\|(C^Z - C^X)^* e_i\|_1 + \|R^{Z^*} e_i - P^{X^*} e_i\|_1 \leq P^Z_{(i,m+i)} \ .$$

Thus, for all $1 \leq i \leq p$*:*

$$\|(C^Z - C^X)^* e_i\|_1 + \|R^{Z^*} e_i - P^{X^*} e_i\|_1 \leq P^Z_{(i,m+i)} \ .$$

Let t_i*,* $1 \leq i \leq p$*, denote the coordinates of* t*. We have:*

$$\begin{aligned}&\|M^X(C^X - C^Z)^* t\|_1 + \|P^{X^*} t - R^{Z^*} t\|_1 \\ &\qquad \leq \sum_{i=1}^{p} |t_i| (\|M^X(C^X - C^Z)^* e_i\|_1 + \|R^{X^*} e_i - P^{X^*} e_i\|_1) \\ &\qquad \leq \sum_{i=1}^{p} |t_i| P^Z_{(i,m+i)} \\ &\qquad = \|D^{Z^*} t\|_1\end{aligned}$$

The first inequality uses the triangle inequality. The last equality is due to the fact that D^Z is a diagonal matrix. Now, again the triangle inequality gives

$$\begin{aligned}\|M^X(C^X - C^Z)^* t\|_1 + \|P^{X^*} t\|_1 - \|R^{Z^*} t\|_1 \\ \leq \|M^X(C^X - C^Z)^* t\|_1 + \|P^{X^*} t - R^{Z^*} t\|_1,\end{aligned}$$

which makes,

$$\begin{aligned}\|M^X(C^X - C^Z)^* t\|_1 + \|P^{X^*} t\|_1 &\leq \|R^{Z^*} t\|_1 + \|D^{Z^*} t\|_1 \\ &= \|P^{Z^*} t\|_1 \ .\end{aligned}$$

The last equality uses Proposition 2:

$$\begin{aligned}\|P^{Z^*} t\|_1 &= \delta(t \mid P^Z \mathfrak{B}) \\ &= \delta(t \mid R^Z \mathfrak{B} + D^Z \mathfrak{B}) \\ &= \delta(t \mid R^Z \mathfrak{B}) + \delta(t \mid D^Z \mathfrak{B}) \\ &= \|R^{Z^*} t\|_1 + \|D^{Z^*} t\|_1\end{aligned}$$

We prove similarly that $\hat{Y} \leq_{1\times 2} \hat{Z}$, by substituting X and Y. Therefore, $\hat{Z}$ is an upper bound of $\hat{X}$ and $\hat{Y}$. ∎

The computation of $\sqcup^+$ considers, only for the purpose of computation of the join, that the perturbation noise symbols η_i, $1 \leq i \leq m$ are shared between the operands. Observe that the operation $\sqcup^+$ adds p new perturbation noise symbols, whereas the number of noise symbols after $\sqcup$ is at most p.

The operators $\sqcup$ and $\sqcup^+$ are based on the computation of the minimal upper bound of the two perturbed affine forms related to each variable. Instead of computing the mub line by line, we could also compute the upper bound using the efficient (linear) operator $\vee$ (see Proposition 6.2.36). For instance, the use of $\sqcup^+$ together with $\vee$ over perturbed affine forms (see Definition 6.2.34) gives exactly [GP09, Lemma 10].

CHAPTER 7

Implementation and Experiments

We detail in this section our implementation of the Constrained Affine Sets abstract domain. Our domain is called *Taylor1+* [GGP09]. Taylor1+ is fully compliant with the APRON library [apr07]. The APRON project presents a set of numerical abstract domains, namely Boxes, Octagons and Polyhedra, with a common interface, allowing to switch from one domain to another without an extra effort. One can then easily compare results of different numerical domains, or combine them for a better precision. Taylor1+ takes as a parameter any other APRON abstract domain (the default choice is Boxes) to handle constraints on variables.

We use our domain to experiment the precision and the efficiency of our approach and compare our results to the already existing abstract domains.

We use floating-point numbers arithmetic (precisely double-precision) for the abstract computations. Our choice is motivated by the flexibility and the efficiency of computations of this binary representation. Dealing with floating-points needs a particular attention, firstly to avoid classical pitfalls, and secondly to ensure the soundness of the analysis.

7.1 Abstract Computations Using Floating-point Numbers

So far, all results and operations over affine sets, either constrained or not, are defined over real numbers semantics. The coefficients of the matrices C and P were considered as perfect real numbers, and operations assume that

operands are within the unbounded real numbers line. In practice, however, computers are unable to represent these real numbers, the set of expressible numbers is even finite. Two finite-precision representations of real numbers exist: the fixed-point representation and the floating-point representation.

Fixed-point representation Numbers are divided into an integer part and a fractional part. The number of bits used for each part determines the fixed-point type. Any base can be used to represent these parts. For instance, if we consider the human readable base 10, the decimal number 3.14 can be represented by 0314 multiplies the fraction $\frac{1}{100}$, if we consider 4 bits for the integer part and 3 bits for the fractional part. If the number does not fit into the representation, it is rounded or truncated. For instance we can not store more than 3 decimals of π using our previous type, as 3.1415 is truncated . Many directions could be considered when rounding. A possible fixed point representatives of 3.1415, could be either 3.14 or 3.15. The first fixed-point number is the nearest representative, whereas the second is the first bigger representative. In both cases, there is a loss of precision due to the restriction to fit the internal representation of numbers. The computation is always performed with respect to the same type. Thus, the type is chosen to improve the accuracy of computations and to avoid the overflows. The latter occurs when we would like to represent a number which is strictly greater than the biggest number that the internal representation allows. For instance, with respect to our convention, the number 100 overflows, as the biggest number we could represent is 99.99, represented by the integer part 9999 and the fraction $\frac{1}{100}$. Implementations usually rely on 2 instead of base 10 for efficiency reasons. There is non-built-in support in common processors for fixed-point computations. Classical high level languages such as `C` or `C++` do not offer neither a type nor a library for fixed-point computations. Nevertheless, this representation is used for special computations such as decoding the audio signal [tre] (for embedded lower consumption circuit), or financial accounting softwares [gnu] as the rounding error is more predictable than the floating-point representation.

Floating-point representation This finite-precision representation is by far the most commonly used during the last quarter century. Partly because of the existence of a standard (IEEE-754) since 1985 [IEE85] for floating-point arithmetic (with one major revision in 2008 [IEE08]), which was adopted by software editors as well as major semiconductor chip makers (Intel, AMD, IBM). Since early 1980, many manufactured CPU come with a special co-processor dedicated to the floating point computations, such as

the famous Motorola 68000 family. Intel $x86$ family comes even with a built-in floating-point unit. Floating-point representation is flexible, intuitive, and supported by both high-level languages. Moreover, the native hardware support allows a great performance of computations. We briefly introduce in the next section the standard IEEE 754 floating-point numbers.

Floating-point Numbers

We focus on the binary floating-point representation of real numbers. The base used is then the base 2.

Representation The floating-point representation is divided into three parts: one bit for the sign, m bits to encode the fraction part, called the *mantissa* and e bits to encode the exponent. The total number of used bits, that is $1 + e + m$, defines the precision of a representation, we denote it by p. The standard IEEE-754 defines 4 representations: single-precision ($p = 1 + 8 + 23$), double-precision ($p = 1 + 11 + 52$), single-extended-precision, and($p \geq 43$), and the double-extended-precision ($p \geq 79$). The single and double precision floating-point numbers are the most common precisions used, the last one is used for instance in Intel registers which store floating-point numbers with $p = 80$. The single-extended precision is seldom used. A typical (big-endian [1]) single-precision floating-point number (32 bits) is decomposed from most (left) to less (right) significant bits into three parts: the sign bit, 1 bit, has the position 31 counting from right to left (and starting from 0), the biased exponent (8 bits, from position 23 to 30), and the mantissa, 23 bits, for the rest. The sign bit determines the sign of the binary number: 1 means positive, and 0 means negative. The exponent is stored in biased format *bexp* without a sign bit: a constant is added to the bexp to find the actual exponent. This bias to add is defined by $2^{e-1} - 1$, where e denotes the number of bits allocated for the exponent. For instance, for single-precision representation, the $e = 8$, thus the bias is equal to $2^7 - 1 = 128 - 1 = 127$: to encode the exponent 1, bexp should contain $1+127 = 128$. The mantissa encodes the fraction part of the binary number, that is the sequence of bits on the right hand side of the binary point. The most significant bit of the mantissa is hidden. It is in fact encoded in the exponent as follows:

[1] Endianness refers to the bit order used to represent a sequence of bits. Big-endian starts from the most significant bit, whereas little-endian starts with the less significant bit.

- if bexp is zero and the fraction is non-zero (at least one bit is set to one), then the hidden bit is equal to 0. These numbers are called *denormalized* floating-point numbers.

- if the bexp lies within $[1, 2^e-2]$, then the hidden bit is considered equal to one. These numbers are called *normalized* floating-point numbers.

The remaining possible value of the exponent, that is $2^e - 1$, encodes the infinities and *NaN*s. NaNs stands for "Not a number", and is used to store undefined results such as the inverse of zero. The value of the mantissa is used to distinguish between infinities and NaNs. If it is equal to zero, then the number is either $-\infty$ or $+\infty$, depending on the sign. If at least one bit of the mantissa is equal to one, then we have NaNs (sign is useless). Observe that the floating-point representation makes a difference between -0 and $+0$, as the bit sign is independent from the value. The final form of a normalized number is

$$(-1)^s 1.\text{mantissa} \times 2^{\text{bexp}-\text{bias}} \ .$$

For instance, our above example gives the (normalized) binary number $(-1)^0 1.10010010000111111101 \times 2^{128-127}$ (equal to 3.1415901 in base 10). Given a precision, the number of floating-point numbers is finite and their values bounded. For instance, the double-precision numbers range from $-1.1111\ldots \times 2^{2046-1023}$ to $1.1111\ldots \times 2^{2046-1023}$, which gives in base ten:

$$\pm(1 + \frac{1}{2} + \ldots + \frac{1}{2^{52}}) \times 2^{1023} = \pm(1 - \frac{1}{2^{53}}) \times 2^{1024} \simeq \pm 1.7977 \times 10^{308} \ .$$

It is worth noting that the floating-point numbers are not uniformly distributed. The population is dense around zero (because of the denormalized representation), whereas the gap between two successive floating-point numbers increases as we recede from zero toward the bounds. In fact, the gap is scaled by the factor 2^{exp}, where exp denotes the exponent. For instance, in double-precision representation:

exp	bexp	range	gap
0	1023	$[1, 2 - \frac{1}{2^{52}}]$	$\frac{1}{2^{52}} \simeq 2.22 \times 10^{-16}$
2	1025	$[4, 8 - \frac{1}{2^{48}}]$	$\frac{2^2}{2^{52}} \simeq 8.88 \times 10^{-16}$
53	1076	$[2^5, 2^{54} - 2]$	$\frac{2^{53}}{2^{52}} = 2$

This behavior makes floating-point numbers unsuitable for computations that involve large real numbers, such as accounting softwares. As said earlier, the fixed-point computation offers in that case a more convenient way as the distribution is more regular than floating-point numbers.

Rounding directions Arithmetic over the floating-point numbers is not complete as the result may be non-floating-point number. A round off is then needed to *cast* the result into the needed precision. The IEEE-745 standard defines four possible rounding directions:

- *Round to Nearest* rounds the result to its nearest floating-point representative, if it falls midway, the representative with its least significant bit equal to zero is preferred. This is the default rounding mode. It is called Round to nearest, *ties to even* in the new revision which also precises other modes.
- *Round toward* 0 rounds the result to the first representative between the result and zero.
- *Round toward* $+\infty$ rounds the result to the closest bigger representative.
- *Round toward* $-\infty$ rounds the result to the closest smaller representative.

Given a real number r, we denote by $\text{float}_{p,r}(r)$ the floating-point representative of r with respect to the precision p, and the rounding direction r, where $r \in \{\mathfrak{n}, 0, -\infty, +\infty\}$, and $\mathfrak{n}$ denotes the rounding to the nearest with ties to even mode. We define $\text{float}_{p,r}(+\infty) = +\infty$, $\text{float}_{p,r}(-\infty) = -\infty$, and $\text{float}_{p,r}(0) = +0$, for all rounding directions and all precisions.

Invalid operations The undefined operations in real numbers arithmetic are also invalid in floating-point numbers arithmetic. For instance, $(-\infty)+(+\infty)$, $0 \times \infty$, $\frac{0}{0}$, $\frac{\infty}{\infty}$, the square root of non-positive floating-point number, returns a NaN. Any operation involving a NaN returns also a NaN.

Some pitfalls need to be aware of when using floating-point numbers. An excellent survey should be the Goldberg article [Gol91]. Floating-point numbers are not real numbers and should not be considered as such when reasoning. Classical arithmetic operations $\{+, -, \times, \div\}$ are not commutative, neither associative. Thus, depending on the order of evaluation, the result may be different. Two given floating-point numbers are comparable with respect to the order over the extended real numbers line. The order considers $-0 = +0$, $-\infty = -\infty$, $+\infty = +\infty$, and $x \neq NaN$, for any given floating-point number x including NaNs themselves. The decimal fractions in base ten, such as 0.1, 0.01, etc. do not have exact representatives independently from the precision in use. In fact, the binary representatives of these fractions come with infinite binary chain; for instance the "pure" binary representative of 0.1 is $1.100\mathbf{1100}11001100\ldots$.

Intervals domain

Many numerical abstract domains use floating-point numbers for their internal computations. The loss of precision due to such approximation is usually compensated by a tremendous gain of efficiency. However, their use for the purpose of abstract interpretation needs a particular attention to ensure the soundness of computations.

As mentioned in [Min04b] and implemented in [apr07], intervals arithmetic can be implemented in a sound manner using rounding toward $\pm\infty$. We denote by $\mathbb{I}_{\mathbb{F}}$ the set of intervals with floating-point bounds. A real interval $[r_1, r_2]$ is abstracted as follows:

$$\begin{aligned} \alpha_{\mathbb{F}} : \mathbb{I} &\to \mathbb{I}_{\mathbb{F}} \\ [r_1, r_2] &\mapsto [\mathrm{float}_{p,-\infty}(r_1), \mathrm{float}_{p,+\infty}(r_2)] \ . \end{aligned}$$

The concretisation, from $\mathbb{I}_{\mathbb{F}}$ to $\mathbb{I}$ is the restriction to $\mathbb{I}_{\mathbb{F}}$ of the identity over $\mathbb{I}$ since $\mathbb{I}_{\mathbb{F}} \subseteq \mathbb{I}$. The definition of $\alpha_{\mathbb{F}}$ respects the soundness property, that is $[r_1, r_2] \subseteq \alpha_{\mathbb{F}}([r_1, r_2])$. Operations over intervals of $\mathbb{I}_{\mathbb{F}}$ are defined as follows:

$$\begin{aligned}
[f_1, f_2] +_{\mathbb{F}} [f_1', f_2'] &\stackrel{\text{def}}{=} [\mathrm{float}_{p,-\infty}(f_1 + f_1'), \mathrm{float}_{p,+\infty}(f_2 + f_2')] \\
[f_1, f_2] -_{\mathbb{F}} [f_1', f_2'] &\stackrel{\text{def}}{=} [\mathrm{float}_{p,-\infty}(f_1 - f_2'), \mathrm{float}_{p,+\infty}(f_2 - f_1')] \\
[f_1, f_2] \times_{\mathbb{F}} [f_1', f_2'] &\stackrel{\text{def}}{=} [\min\{\mathrm{float}_{p,-\infty}(f_1 \times f_1'), \mathrm{float}_{p,-\infty}(f_2 \times f_1'), \\
&\qquad \mathrm{float}_{p,-\infty}(f_1 \times f_2'), \mathrm{float}_{p,-\infty}(f_2 \times f_2')\}, \\
&\qquad \max\{\mathrm{float}_{p,+\infty}(f_1 \times f_1'), \mathrm{float}_{p,+\infty}(f_2 \times f_1'), \\
&\qquad \mathrm{float}_{p,+\infty}(f_1 \times f_2'), \mathrm{float}_{p,+\infty}(f_2 \times f_2')\}] \\
[f_1, f_2] \div_{\mathbb{F}} [f_1', f_2'] &\stackrel{\text{def}}{=} [\min\{\mathrm{float}_{p,-\infty}(f_1 \div f_1'), \mathrm{float}_{p,-\infty}(f_2 \div f_1'), \\
&\qquad \mathrm{float}_{p,-\infty}(f_1 \div f_2'), \mathrm{float}_{p,-\infty}(f_2 \div f_2')\}, \\
&\qquad \max\{\mathrm{float}_{p,+\infty}(f_1 \div f_1'), \mathrm{float}_{p,+\infty}(f_2 \div f_1'), \\
&\qquad \mathrm{float}_{p,+\infty}(f_1 \div f_2'), \mathrm{float}_{p,+\infty}(f_2 \div f_2')\}] \\
\sqrt[\mathbb{F}]{[f_1, f_2]} &\stackrel{\text{def}}{=} [\mathrm{float}_{p,-\infty}(\sqrt{f_1}), \mathrm{float}_{p,+\infty}(\sqrt{f_2})]
\end{aligned}$$

Affine Forms Domain

We denote by $\mathcal{A}_1(\mathcal{F})$, the set of affine forms with floating-point numbers coefficients.

The definition of the affine arithmetic over $\mathcal{A}_1(\mathcal{F})$ is not immediate, as the computations need to be safe. As mentioned by Figueiredo and Stolfi

in [dFS97], there is no safe rounding direction that we could use to obtain a sound affine form [2]. The rounding alters the correlations between variables which may lead to unsafe affine representation. Consider for instance the CAS $\hat{X}$, defined over $\mathcal{A}_1$:

$$\hat{X} = \left(\begin{pmatrix} 0 & 0.5 \\ 0 & 0.1 \end{pmatrix}, 0, 1 \times [-1, 1] \right),$$

The four CAS defined over $\mathcal{A}_1(\mathcal{F})$ obtained by rounding 0.1 are not safe:

$$\hat{X}_{p,r} = \left(\begin{pmatrix} 0 & \text{float}_{p,r}(0.5) \\ 0 & \text{float}_{p,r}(0.1) \end{pmatrix}, 0, 1 \times [-1, 1] \right),$$

where the rounding mode r is within $\{\mathfrak{n}, 0, -\infty, +\infty\}$. Figure 7.1 illustrates the (degenerated) zonotope $\gamma_{1\times 2}(\hat{X})$ (gray segment), and the two zonotopes $\gamma_{1\times 2}(\hat{X}_{p,+\infty})$ and $\gamma_{1\times 2}(\hat{X}_{p,-\infty})$ (red segments). In this case, we have

$$\gamma_{1\times 2}(\hat{X}_{p,0}) = \gamma_{1\times 2}(\hat{X}_{p,-\infty}) \quad \text{and} \quad \gamma_{1\times 2}(\hat{X}_{p,\mathfrak{n}}) = \gamma_{1\times 2}(\hat{X}_{p,+\infty}),$$

since $\text{float}_{p,r}(0.5) = 0.1$ for all rounding directions (0.5 is exactly representable), and

$$\text{float}_{p,+\infty}(0.1) = \text{float}_{p,\mathfrak{n}}(0.1) \quad \text{and} \quad \text{float}_{p,-\infty}(0.1) = \text{float}_{p,0}(0.1),$$

for $p \in \{32, 64\}$. None of these zonotopes contains the zonotope $\gamma_{1\times 2}(\hat{X})$, which makes $\gamma_{1\times 2}(\hat{X}) \not\subseteq \gamma_{1\times 2}(\text{cast}_{p,r}(\hat{X}))$, for all $r \in \{n, 0, -\infty, +\infty\}$. Figueiredo and Stolfi proposed in [dFS97] to add a fresh noise symbol to compensate the rounding error. The coefficient of this newly added symbol is an over-approximation of the rounding error. For instance in our simple example, the affine form related to v_2 becomes

$$\text{float}_{p,\mathfrak{n}}(0.1)\epsilon_1 + \text{float}_{p,+\infty}(|0.1 - \text{float}_{p,\mathfrak{n}}(0.1)|)\eta_f \ .$$

The choice of round-offs is meant to minimize the coefficient of η_f: $|0.1 - \text{float}_{p,r}(0.1)|$ is minimal when $r = \mathfrak{n}$, and rounding the absolute value toward $+\infty$ gives a floating-point greater than (or equal to) the actual real number given by this absolute value. Figure 7.2 depicts the zonotope

$$\hat{X}_{\mathbb{F}} \stackrel{\text{def}}{=} \left(\begin{pmatrix} 0 & \text{float}_{p,\mathfrak{n}}(0.5) \\ 0 & \text{float}_{p,\mathfrak{n}}(0.1) \end{pmatrix}, \begin{pmatrix} 0 \\ \text{float}_{p,+\infty}(|0.1 - \text{float}_{p,\mathfrak{n}}(0.1)|) \end{pmatrix}, 1 \times [-1, 1]^2 \right).$$

Observe that it wraps closely the degenerated zonotope $\gamma_{1\times 2}(\hat{X})$ but leads to a loss of precision, in addition to a new perturbation noise symbol.

[2] In intervals domain for instance, it is sufficient to round toward $+\infty$ the upper bound computations and toward $-\infty$ the lower bound computations to obtain a safe interval that contains all possible real numbers.

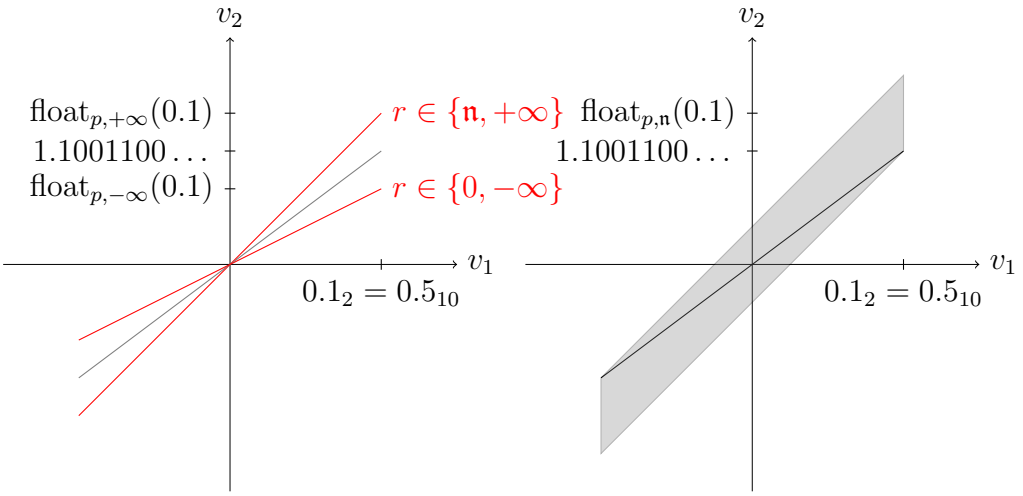

Figure 7.1: Unsafe approximation of affine sets: the correlation is falsified by the use of floating-point numbers.

Figure 7.2: Safe approximation: $\hat{X}_{\mathbb{F}}$ wraps $\hat{X}$.

Constrained Affine Sets Domain

The coefficients of the matrices C and P are now floating-point numbers. The domain $\mathcal{A}_1$ is substituted by its related domain which respects the floating-point semantics (see Section 7.1).

Related work Ideas we present here to overcome the use of floating-point numbers while remaining sound are well known in the literature. Figueiredo and Stolfi have presented in [dFS97] similar approaches to implement a library of affine forms. Moreover, the use of Taylor models, with floating-point numbers, as an approximation technique for linearization has been widely studied and proved guaranteed (see for instance [RMB05] for a proof of the respect of Taylor models, with floating-point coefficients, of the "containment property"– which is equivalent to the soundness in our context –). The only main difference with our implementation is the use of intervals, instead of simple floating-point numbers, to encode the coefficients of noise symbols.

The semantics of the evaluation of an expression $e \in expr$, in the ab-

stract domain $\mathcal{A}(\mathbb{F})_{1\times 2}$ is given by:

$$\forall e \in expr, [\![e]\!]^\sharp_{\mathbb{F}} : \mathcal{A}(\mathbb{F})_{1\times 2} \to \mathcal{A}(\mathbb{F})_1 \times \mathcal{A}_2(\mathbb{F})$$
$$[\![v_k]\!]^\sharp_{\mathbb{F}}(C,P,\Phi) \stackrel{\text{def}}{=} (\sum_{i=0}^{n} C_{(k,i)}\epsilon_i + \sum_{j=1}^{m} P_{(k,j)}\eta_j, \Phi)$$

The evaluation of an interval $[a,b]$ depends whether the interval is bounded or unbounded. If $-\infty < a \leq b < +\infty$, then $[\![[a,b]]\!]^\sharp_{\mathbb{F}}(C,P,\Phi)$ is defined by

$$\big(\text{mid}_{\mathbb{F}}(\alpha_{\mathbb{F}}([a,b])) + \text{dev}_{\mathbb{F}}(\alpha_{\mathbb{F}}([a,b]))\epsilon_f, [\![-1 \leq \epsilon_f \leq 1]\!]^\sharp_{2(\mathbb{F})}[\![\text{add}\,\epsilon_f]\!]^\sharp_{2(\mathbb{F})}\Phi\big),$$

Otherwise, the affine form is just a new noise symbol :

$$[\![[a,b]]\!]^\sharp_{\mathbb{F}}(C,P,\Phi) \stackrel{\text{def}}{=} \begin{cases} (\epsilon_f, [\![\epsilon_f \leq b]\!]^\sharp_{2(\mathbb{F})}[\![\text{add}\,\epsilon_f]\!]^\sharp_{2(\mathbb{F})}\Phi), & \text{if } -\infty = a \\ (\epsilon_f, [\![a \leq \epsilon_f]\!]^\sharp_{2(\mathbb{F})}[\![\text{add}\,\epsilon_f]\!]^\sharp_{2(\mathbb{F})}\Phi), & \text{if } +\infty = b \end{cases}$$

For arithmetic unary and binary operations, the semantics is given by:

$$[\![e_1 \diamond e_2]\!]^\sharp_{\mathbb{F}}(C,P,\Phi) \stackrel{\text{def}}{=} [\![e_1]\!]^\sharp_{\mathbb{F}}(C,P,\Phi) \diamond^{\mathbb{F}}_{1\times 2} [\![e_2]\!]^\sharp_{\mathbb{F}}(C,P,\Phi)$$
$$\text{where } \diamond^{\mathbb{F}}_{1\times 2} \in \{+^{\mathbb{F}}_{1\times 2}, -^{\mathbb{F}}_{1\times 2}, \times^{\mathbb{F}}_{1\times 2}, \div^{\mathbb{F}}_{1\times 2}\}$$
$$[\![\sqrt{e}]\!]^\sharp_{\mathbb{F}}(C,P,\Phi) \stackrel{\text{def}}{=} \sqrt{}^{\mathbb{F}}_{1\times 2}[\![e]\!]^\sharp_{\mathbb{F}}(C,P,\Phi)$$

The operators $[\![\cdot]\!]^\sharp_{2(\mathbb{F})}$ denote the abstract operators of the abstract domain $\mathcal{A}_2(\mathbb{F})$. The operations $\text{mid}_{\mathbb{F}}$ return the nearest floating-point number to the exact mid point of an interval:

$$\text{mid}_{\mathbb{F}} : \mathbb{I}_{\mathbb{F}} \to \mathbb{F}$$
$$[f_1, f_2] \mapsto \text{float}_{p,\mathfrak{n}}(\frac{f_1 + f_2}{2}) \ .$$

The function $\text{dev}_{\mathbb{F}}$, defined below, ensures an over-approximation to the exact radius of the interval, that is $\text{dev}(\boldsymbol{i}) \leq \text{dev}_F(\boldsymbol{i})$, for all $\boldsymbol{i} \in \mathbb{I}_{\mathbb{F}}$:

$$\text{dev}_{\mathbb{F}} : \mathbb{I}_{\mathbb{F}} \to \mathbb{F}$$
$$[f_1, f_2] \mapsto \max\{\text{float}_{p,+\infty}(f_2 - \text{mid}_{\mathbb{F}}([f_1, f_2])), \text{float}_{p,+\infty}(\text{mid}_{\mathbb{F}}([f_1, f_2]) - f_1)\}$$

Notice that, the definitions of $\text{mid}_{\mathbb{F}}$ and $\text{dev}_{\mathbb{F}}$ are such that the conversion from intervals to affine forms, then back to intervals encloses the original interval.

Let $\hat{x}_{\mathbb{F}} = \sum_{i=0}^{n} \alpha_i^x \epsilon_i + \sum_{j=1}^{m} \beta_j^x \eta_j$, and $\hat{y}_{\mathbb{F}} = \sum_{i=0}^{n} \alpha_i^y \epsilon_i + \sum_{j=1}^{m} \beta_j^y \eta_j$ be two elements of $\mathcal{A}_1(\mathbb{F})$, and Φ an element of $\mathcal{A}_2(\mathbb{F})$. The abstract element Φ is unused and remains unchanged. The addition $+_{1\times 2}^{\mathbb{F}}$ of two CAS $(\hat{x}_{\mathbb{F}}, \Phi)$ and $(\hat{y}_{\mathbb{F}}, \Phi)$ gives an affine form $\hat{z}_{\mathbb{F}}$ and a noise symbols' abstract element Φ^z, defined as follows:

$$\hat{z}_{\mathbb{F}} \stackrel{\text{def}}{=} \sum_{i=0}^{n} \text{float}_{p,\mathfrak{n}}(\alpha_i^x + \alpha_i^y)\epsilon_i + \sum_{j=1}^{m} \text{float}_{p,\mathfrak{n}}(\beta_j^x + \beta_j^y)\eta_j + \boxplus\eta_{m+1}$$
$$\Phi^z \stackrel{\text{def}}{=} [\![-1 \leq \eta_{m+1} \leq 1]\!]_{2(\mathbb{F})}^{\sharp} [\![\text{add}\,\eta_{m+1}]\!]_{2(\mathbb{F})}^{\sharp} \Phi \ .$$

The floating-point number $\boxplus$ accumulates all rounding errors:

$$\boxplus \stackrel{\text{def}}{=} \text{float}_{p,+\infty}\left(\frac{\text{ulp}_p}{2}\left(\sum_{i=0}^{n} 2^{\log_2(\text{float}_{p,+\infty}(\alpha_i^x+\alpha_i^y))} + \sum_{j=1}^{m} 2^{\log_2(\text{float}_{p,+\infty}(\beta_j^x+\beta_j^y))}\right)\right),$$

where ulp_p denotes the floating-point number which corresponds to the unit in the last place, that is the floating-point number obtained when only the least significant bit is set to 1. The function $\log_2$ extracts the exponent of the floating-point number of its operand, that is the biased exponent minus the bias. For instance, $\log_2(10.01) = 1$. Such a function is provided in the API of high languages, in `C` language for instance, the primitive `logb` defined in the header file `math.h` extracts the exponent of a floating-point number. As we round to the nearest,

$$\frac{\text{ulp}_p}{2} 2^{\log_2(\text{float}_{p,+\infty}(x))},$$

which is usually denoted by $\frac{\text{ulp}(x)}{2}$, gives a tight over-approximation of $|x - \text{float}_{p,\mathfrak{n}}(x)|$. Notice that the computation of $\boxplus$ requires to change the rounding mode, which is a relatively expensive operation (costs 6 floating-point additions, that is around a dozen of cycles on a typical FPU [SS98]). To avoid the repeated changes of the rounding direction, we may compute the $\log_2$ of the next representable neighbor of $\text{float}_{p,\mathfrak{n}}(x)$, instead of $\text{float}_{p,+\infty}(x)$, which is less precise but safe.

In our implementation the coefficients operations are handled by intervals. For instance, to compute the addition of two floating-point numbers $\alpha_i^x + \alpha_i^y$, we actually compute $\boldsymbol{\alpha_i^x} +_{\mathbb{F}} \boldsymbol{\alpha_i^y}$, where $\boldsymbol{\alpha_i^x} \stackrel{\text{def}}{=} [\alpha_i^x, \alpha_i^x]$, and $\boldsymbol{\alpha_i^y} \stackrel{\text{def}}{=} [\alpha_i^y, \alpha_i^y]$. The obtained interval contains the exact value of the addition. The operation $+_{\mathbb{F}}$ over intervals is sound as detailed in section 7.1. The new coefficient is assigned $\text{mid}_{\mathbb{F}}(\boldsymbol{\alpha_i^x} +_{\mathbb{F}} \boldsymbol{\alpha_i^y})$, while $\text{dev}_{\mathbb{F}}(\boldsymbol{\alpha_i^x} +_{\mathbb{F}} \boldsymbol{\alpha_i^y})$ is added to $\boxplus$. The method is less precise than the two previous methods. It

is also more expensive as it involves the addition of two intervals. We now have for the addition $+^{\mathbb{F}}_{1\times 2}$:

$$\begin{aligned}
\hat{z}_{\mathbb{F}} &= \sum_{i=0}^{n} \mathrm{mid}_{\mathbb{F}}(\boldsymbol{\alpha_i^x} +_{\mathbb{F}} \boldsymbol{\alpha_i^y})\epsilon_i + \sum_{j=1}^{m} \mathrm{mid}_{\mathbb{F}}(\boldsymbol{\beta_j^x} +_{\mathbb{F}} \boldsymbol{\beta_j^y})\eta_j + \boxplus\eta_{m+1},\\
\Phi^z &= [\![-1 \le \eta_{m+1} \le 1]\!]^\sharp_{2(\mathbb{F})}[\![\mathrm{add}\,\eta_{m+1}]\!]^\sharp_{2(\mathbb{F})}\Phi),\\
\boxplus &= \mathrm{float}_{p,+\infty}\Big(\sum_{i=0}^{n} \mathrm{dev}_{\mathbb{F}}(\boldsymbol{\alpha_i^x} +_{\mathbb{F}} \boldsymbol{\alpha_i^y}) + \sum_{j=1}^{m} \mathrm{dev}_{\mathbb{F}}(\boldsymbol{\beta_j^x} +_{\mathbb{F}} \boldsymbol{\beta_j^y})\Big)\ .
\end{aligned}$$

The operation may be seen as a composition of two operations. The first computes the sum of two affine forms as if the coefficients were intervals. The second applies a *reduction* to get back to a classical affine form by replacing intervals by their midpoint with respect to the operation $|_{\mathbb{F}}$, and by accumulating the rounding errors into $\boxplus$. We denote by $\mathcal{A}_1(\mathbb{I}_{\mathbb{F}})$ the set of affine forms with interval coefficients. We use the bold face notation to denote an element of $\mathcal{A}_1(\mathbb{I}_{\mathbb{F}})$. The reduction operator is defined as follows:

7.1.1 Definition (Reduction)

$$\begin{aligned}
&\mathrm{reduction} : \mathcal{A}_1(\mathbb{I}_{\mathbb{F}}) \times \mathcal{A}_2(\mathbb{F}) \to \mathcal{A}_1(\mathbb{F}) \times \mathcal{A}_2(\mathbb{F})\\
&\mathrm{reduction}(\hat{\boldsymbol{x}}_{\mathbb{F}}, \Phi^X) \stackrel{\mathrm{def}}{=} (\hat{y}_{\mathbb{F}}, [\![-1 \le \eta_f \le 1]\!]^\sharp_{2(\mathbb{F})}[\![\mathrm{add}\,\eta_f]\!]^\sharp_{2(\mathbb{F})}\Phi^X), \textit{ where}\\
&\qquad\qquad \alpha_0^y = \mathrm{mid}(\boldsymbol{\imath})\\
&1 \le i \le n, \quad \alpha_i^y = \mathrm{mid}(\boldsymbol{\alpha_i^x})\\
&1 \le j \le m, \quad \beta_j^y = \mathrm{mid}(\boldsymbol{\beta_j^x})\\
&\qquad\qquad \beta_f^y = \mathrm{dev}(\boldsymbol{\imath})\ .
\end{aligned}$$

The interval $\boldsymbol{\imath}$ *given by*

$$\boldsymbol{\imath} \stackrel{\mathrm{def}}{=} \boldsymbol{\alpha_0^x} + \mathrm{dev}(\boldsymbol{\alpha_0^x})[-1,1] + \sum_{i=1}^{n} \mathrm{dev}(\boldsymbol{\alpha_i^x})[-1,1]\boldsymbol{\epsilon_i^x} + \sum_{i=1}^{m} \mathrm{dev}(\boldsymbol{\beta_i^x})[-1,1]\boldsymbol{\eta_i^x},$$

needs the interval concretisation of the noise symbols, $\boldsymbol{\epsilon_i^x} = \mathrm{bound}_2(\epsilon_i^x, \Phi^X)$, *and* $\boldsymbol{\eta_j^x} = \mathrm{bound}_2(\eta_j^x, \Phi^X)$.

The abstract object of noise symbols is augmented with a new perturbation noise symbol η_f. The so obtained affine form is sound.

1 Remark
It is important to understand that we use intervals as a safe "receptacle" for local intermediate computations of floating-point numbers. The final

result of the operations is always an affine form with floating-point numbers coefficients. The use of intervals as coefficients is interesting in its own and is not covered in this work.

We authorize to scale an affine form $\hat{x}_{\mathbb{F}}$ by an interval $\boldsymbol{\imath}$, which gives an element of $\mathcal{A}_1(\mathbb{I}_{\mathbb{F}})$:

$$\boldsymbol{\imath}\hat{x}_{\mathbb{F}} \stackrel{\text{def}}{=} \sum_{i=0}^{n}(\alpha_i^x \boldsymbol{\imath})\epsilon_i + \sum_{j=1}^{m}(\beta_j^x \boldsymbol{\imath})\eta_j$$

Non linear binary operations $\{\times_{1\times 2}^{\mathbb{F}}, \div_{1\times 2}^{\mathbb{F}}\}$ and the unary operation $\{\sqrt{}_{1\times 2}^{\mathbb{F}}\}$ benefit from both abstract domains $\mathcal{A}_1(\mathbb{F})$ and $\mathcal{A}_2(\mathbb{F})$ for a better precision.

For the multiplication and division operations, we first compute with respect to interval arithmetic, then reduce using the reduction operation. For the multiplication, the SDP method needs a guaranteed solver such as in [JCK07]. However, its use is expensive as it solves the problem more than once to obtain rigorous bounds for the optimal solution.

The definition of $\sqrt{}_{1\times 2}^{\mathbb{F}}$ adds one new perturbation noise symbol: it is firstly created to encode the imprecision due to the linearization, and then used (implicitly in the reduction operation) to store the inaccuracy of computations using floating-point numbers. The special cases where $[a, b]$ is empty, or is equal to zero or has an infinity bound are defined as $\sqrt{}_{1\times 2}$ (the real number coefficients case, see Section 5.1).

The compositional evaluation of an expression may add many new noise symbols (1 per each atomic operation). To avoid this behavior, which can lead to too long affine forms, we can choose to reduce once at each assignment, that is all computations are performed as with interval arithmetic for the coefficients, then the final affine form is reduced, which adds at most one perturbation noise symbol per assignment. The main drawback of this approach is the local loss of precision during the evaluation of the expression due to interval arithmetic.

The join operator The join operator is defined componentwise as discussed in Chapter 6. As coefficients are intervals instead of real numbers, we can not use immediately the already established results that characterize and compute a mub of two affine forms. To overcome this issue, we first reduce the involved affine forms then compute the join. The newly added perturbation noise symbols due to the reduction operation will not survive after the join: if $\sqcup$ is used then all perturbation noise symbols are lost; if $\sqcup^+$ is used instead we accumulate the fresh perturbations due to reduction

into τ^X and τ^Y. Thus, the total number of perturbation noise symbols after the join operation (either $\sqcup/\sqcup^+$, or $\vee$) after reduction is exactly the same than the number of perturbation noise symbols when the join is computed over CAS with real number coefficients.

Algorithm 3, is an extended version of algorithm 2. The values of τ^X and τ^Y are updated respectively lines 3 and 4. By definition of the reduction, the intervals β^X_{m+1} and β^Y_{m+2} are positive real numbers (deviation of an interval), so τ^X and τ^Y remain non-negative.

Algorithm 3: Computing a mub of two Extended Constrained Affine Forms

input : Two extended CAF $\hat{X} = (\boldsymbol{\alpha^X}, \tau^X, \Phi^X_\epsilon)$ and $\hat{Y} = (\boldsymbol{\alpha^Y}, \tau^Y, \Phi^Y_\epsilon)$.
output: A CAF $\hat{Z}$ mub of $\hat{X}$ and $\hat{Y}$.

1 $(\alpha^{X\prime}, \Phi^{X\prime}) \longleftarrow \text{reduction}(\boldsymbol{\alpha^X}, \Phi^X)$;
2 $(\alpha^{Y\prime}, \Phi^{Y\prime}) \longleftarrow \text{reduction}(\boldsymbol{\alpha^Y}, \Phi^Y)$;
3 $\tau^X \longleftarrow \tau^X + \beta^X_{m+1}$;
4 $\tau^Y \longleftarrow \tau^Y + \beta^Y_{m+2}$;
5 `mubCAF`$((\alpha^{X\prime}, \tau^X, \Phi^X), (\alpha^{Y\prime}, \tau^Y, \Phi^Y))$; `/* algorithm 2 */`

7.2 Implementation

Taylor1+ is a `C` library. Its API (Application Programming Interface) is APRON compliant, which means that any analyzer linked to the APRON Library can use Taylor1+ without any extra effort. In addition to its C interface, the library offers an Ocaml interface. This interface is convenient since usually static analysis tools are written in Ocaml language.

The Library is under Lesser General Public Licence (LGPL) and is distributed for free together with the APRON Library.

Data structure Taylor1+ represents a CAS by an array of pointers of size p (the number of the numerical variables) and a generic abstract object for the noise symbols. Each pointer points to a special structure which encodes the affine form. The data structure of an affine form is a coefficient plus a simple chained list of terms. Each term contains a non-null coefficient and a pointer to a noise symbol. (terms with null coefficients are not stored.)

Context-sensitive noise symbols In Taylor1+, noise symbols are not context aware, that is we do not keep track of the control point context (input uncertainty, line, iteration, condition branch, etc.) that creates the noise symbol. We simply make a difference between input noise symbols and perturbation noise symbols. All these symbols are indexed by a global integer variable.

7.3 Experiments

In this section, we compare Taylor1+ with some other relational abstract domains in their APRON implementation, namely octagons and polyhedra. In Section 7.3 we show the accuracy of computations whether these computations are linear or non-linear. Indeed, the affine forms-based domain handles the non-linear operations in an efficient and precise manner. The improvements of expressiveness due to the interpretation of tests using constraints over noise symbols rather than a simple reduced product with intervals is clearly demonstrated in Section 7.3. We focus finally, in section 7.3, on the two join operators, $\vee$ and $\sqcup$, formally defined in Chapter 6. We compare the time cost of each operator as well as the "quality" of the final affine forms.

We used a laptop equipped with Intel(R) Core(TM)2 CPU (1.06GHz) and 2GB of RAM. All results are rounded to two significant digits for the sake of readability.

Efficiency of Computations

We present in this section two benchmarks. We show the efficiency and precision of Taylor1+ computations compared to box, octagons and polyhedra. To this aim, we unroll two recursive schemes, one linear and one non-linear. The first is a (linear) 2nd order filter, the second involves a 3rd order Householder scheme to compute the square root of a given value (usually used when the square root routine is not provided).

We unroll two simple iterative schemes and compare results with the other domains interfaced with APRON, namely boxes, octagons and polyhedra abstract domains. All numerical values are rounded to two significant decimal digits for the sake of readability.

Linear Iterative Schemes

Consider the following 2^{nd} order filter :

$$S_n = 0.7E_n - 1.3E_{n-1} + 1.1E_{n-2} + 1.4S_{n-1} - 0.7S_{n-2}$$

where E_n are independent inputs with unknown values in range $[0, 1]$, and S_n is the output of the filter at iteration n. Pôles are inside the unit circle (norm close to 0.84), so the output in real numbers is provably bounded, and can be tightly estimated by manual methods to $[-1.09, 2.75]$.

We fully unroll this 2^{nd} order filter scheme to compute the abstract value at each iteration. Figure 7.3 compares accuracy and performance of Taylor1+ with three domains, provided in APRON: Boxes (Interval Analysis), Octagons, Polyhedra (both PK [Ja] and PPL [Pro] implementations were tested). The current version of the octogons domain does not integrate any of the symbolic enhancement methods of [Min06b], which leads to inaccurate results. The Polyhedra domain with exact arithmetic (using GMP) gives the exact bounds for the filter output (the scheme is linear). One can see that Taylor1+ wraps very closely the exact range given by polyhedra (left figure) with great performance (right figure).

Non-linear Iterative Scheme

The non-linear scheme we are considering is based on a Householder method of order 3 that converges towards the inverse of the square root of an input A. It originates from an industrial code, used as a test case in [GPBG07]; The current estimate of the inverse of the square root is updated as follows:

$$x_{n+1} = x_n + x_n \left(\frac{1}{2} h_n + \frac{3}{8} h_n^2 \right)$$

where $h_n = 1 - Ax_n^2$, $A \in [16, 20]$ and $x_0 = 2^{-4}$.

We study the fully unrolled scheme for 5 iterations, and compare different implementations of the multiplication; results are shown in Table 7.3. We can see that the results are tight even for non-linear computations. The SDP solver is costly in time and does not seem to buy much more precision. However, for a larger range for input A, SDP gives tighter results than the standard multiplication. Moreover, the real advantage of SDP over subdividing is that the process of subdividing inputs might become intractable when several inputs would need subdividing. We tested here a non-guaranteed SDP solver [Bor99], but we plan in the future to use guaranteed SDP solver such as the one described in [JCK07].

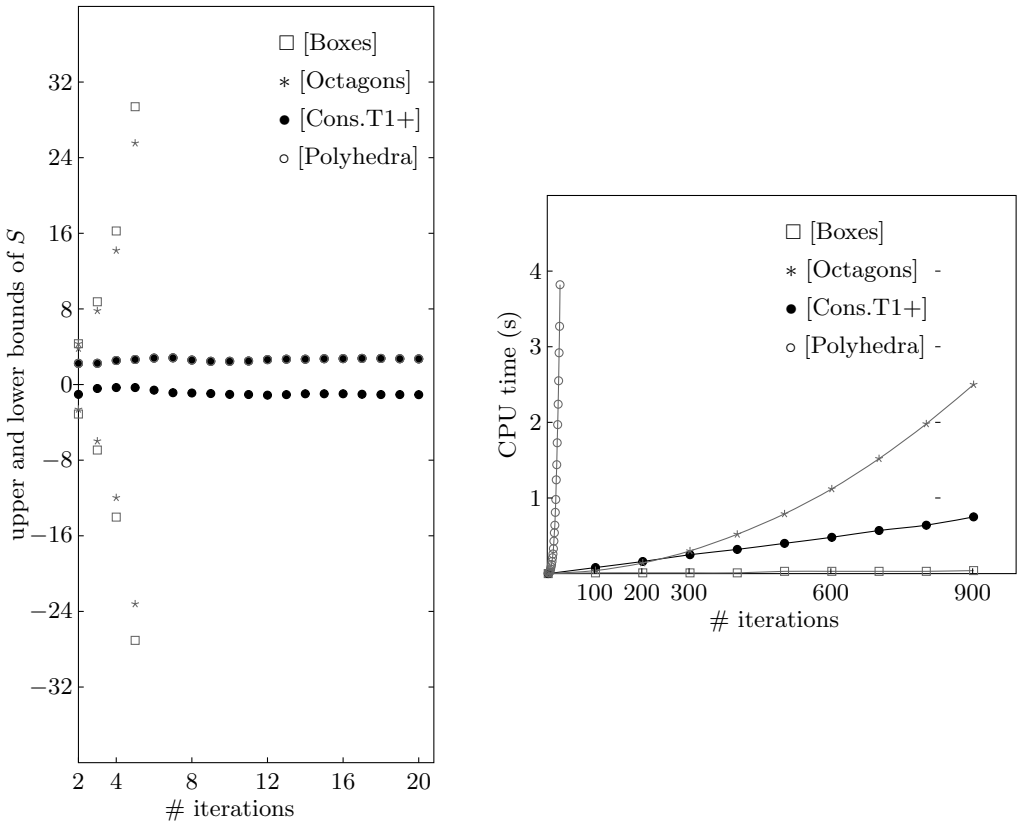

Figure 7.3: Unrolled scheme for the 2^{nd} order filter

Unrolling (5 It.)	$\sqrt{A} = Ax_n$	t(s)
Boxes	[0.51 , 8.44]	1×10^{-4}
Octagons	[0.51 , 7.91]	0.01
Polyhedra	[2.22 , 6.56]	310
T.1+ :	[3.97 , 4.51]	1×10^{-3}
• 10 subdivisions	[4.00 , 4.47]	0.02
• SDP	[3.97 , 4.51]	0.16

Table 7.1: Comparison of domains on Householder (o3) example

	Exact	Octagons	Polyhedra	Uncons. Taylor1+	Taylor1+ ($\vee$)
InterQ1	$[0, 1875]$	$[-3750, 6093]$	$[-2578, 4687]$	$[0, 2500]$	$[0, 1875]$
Cosine	$[-1, 1]$	$[-1.50, 1.0]$	$[-1.50, 1.0]$	$[-1.073, 1]$	$[-1, 1]$
SinCos	$\{1\}$	$[0.84, 1.15]$	$[0.91, 1.07]$	$[0.86, 1.15]$	$[0.99, 1.00]$
InterL2	$\{0.1\}$	$[-1, 1]$	$[0.1, 0.4]$	$[-1, 1]$	$[0.1, 1]$
InterQ2	$\{0.36\}$	$[-1, 1]$	$[-0.8, 1]$	$[-1, 1]$	$[-0.4, 1]$
InterQ2b	$[-0.1, 3]$	$[-3, 27]$	$[-3, 27]$	$[-0.1, 27]$	$[-0.1, 3.77]$

Table 7.2: Comparison of Constrained T1+ with APRON's abstract domains

Interpretation of tests

In this section, we compare the results[3] obtained with the implementation of our domain, with the octagons and polyhedra APRON domains and the unconstrained Taylor1+ [GGP10].

Table 7.2 shows the numerical range of a variable of interest of each test case and for each domain, after giving the exact range we hope to find. It can be noted that on these examples, Taylor1+ is always more accurate than octagons, and is also more accurate than polyhedra on non-affine problems.

In Table 7.2, `InterQ1` combines linear tests with quadratic expressions, only constrained T1+ finds the right upper bound of the invariant. `Cosine` is a piecewise 3rd order polynomial interpolation of the cosine function: once again, only constrained T1+ finds the exact invariant. The program `SinCos` computes the sum of the squares of the sine and cosine functions (real result is 1). `InterL2` (resp. `InterQ2`) computes a piecewise affine (resp. quadratic) function of the input, then focuses on the inverse image of 1 by this function. In `InterQ2b`, which is the running example of [GGP10], we do not evaluate the inverse.

We now consider the computation of $g(g(x))$ on the range $x = [-2, 2]$, where

$$g(x) = \frac{\sqrt{x^2 - x + 0.5}}{\sqrt{x^2 + 0.5}}.$$

We parametrize the program that computes $g(g(x))$ by a number of tests that subdivide the domain of the input variable (see Figure 7.4 left for a parametrization by n subdivisions), in order to compare the relative costs and precisions of the different domains when the size of the program grows.

It can be noted (Figure 7.6 left) that our domain scales up well while giving here more accurate results (Figure 7.6 right) than the other domains. As a matter of fact, with an interval domain for the noise symbols,

[3] sources of the examples are available on line http://www.lix.polytechnique.fr/Labo/Khalil.Ghorbal/CAV2010

```
g(x) = sqrt(x*x-x+0.5)/sqrt(x*x+0.5);
x = [-2,2];
/* for n subdivisions */
h = 4/n;
if (-x<=h-2)
  y = g(x); z = g(y);
...
else if (-x<=i*h-2) /* 2 <= i <= n-1 */
  y = g(x); z = g(y);
...
else
  y = g(x); z = g(y);
```

Figure 7.4: Implementation of g(g(x)) for x in [-2,2]

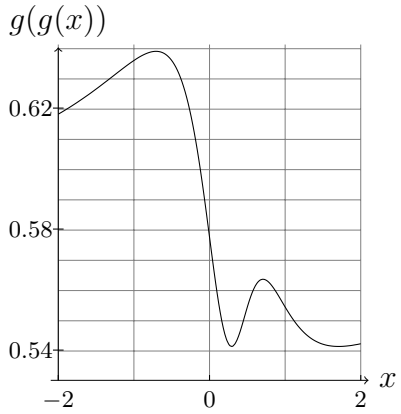

Figure 7.5: plot of g(g(x))

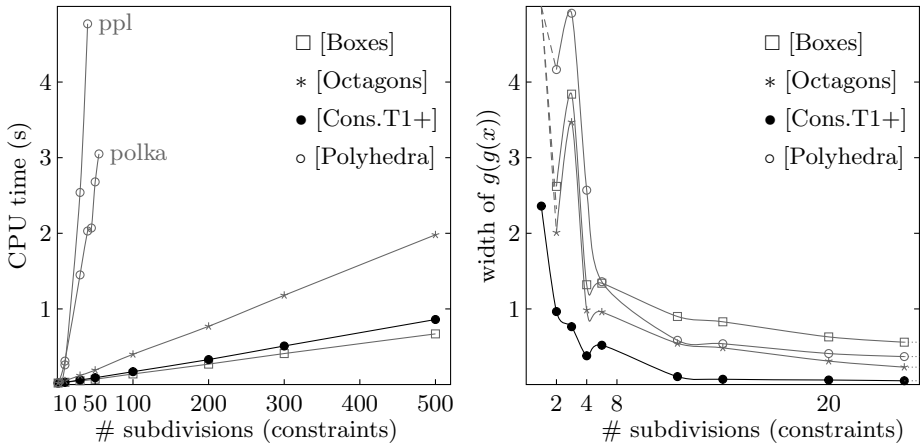

Figure 7.6: Comparing analysis time and results of the different APRON domains.

all abstract transfer functions are linear or at worst quadratic in the number of noise symbols appearing in the affine forms. Notice also that our implementation detects the squares of variables, which allows constrained T1+ to give $[0, 4.72]$ without subdivisions while all other domains end with $[-\infty, +\infty]$ (noted by the dotted lines on Figure 7.6 right). The fact that the results observed for 3 and 5 subdivisions (Figure 7.6 right) are less accurate respectively than those observed for 2 and 4 subdivisions, is related to the behavior of $g(g(x))$ on $[-2, 2]$ (see Figure 7.4 right): for example when a change of monotony appears near the center of a subdivision, the approximations will be less accurate than when it appears at the border.

Join operators Performance

We compare the join operator $\vee$ and $\sqcup$ defined over CAF. The perturbation computed by the first is optimal, while its complexity is cubical function of the number of noise symbols. The latter is much more efficient, its complexity is linear, but is less precise.

The test is performed over randomly generated affine forms of length $n + m$, with coefficients within $[-1, 1]$. The computation of the perturbation with respect to $\vee$ is linear, in our experiments, the needed time never exceeds $0.01s$ (gray line in the bottom of figure 7.7 left). On the other hand, the computation with respect to $\sqcup$ needs much more time, in fact it is cubical as expected theoretically. In the right hand side of figure 7.7, we observe that the optimal perturbation given by $\sqcup$ gives a more accurate

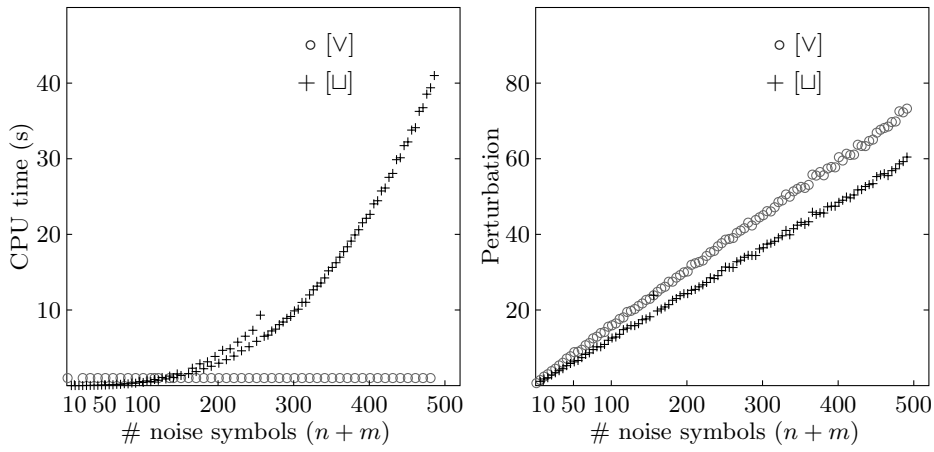

Figure 7.7: Comparison of the join operators : computation time and accuracy of the perturbation

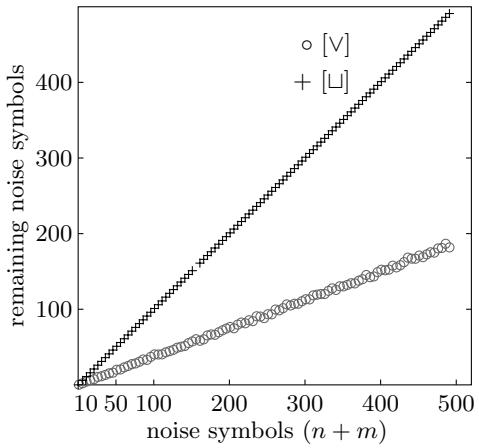

Figure 7.8: Comparison of the join operators : remaining noise symbols

perturbation even though the perturbation given by the operator $\vee$ stays close to the optimal one, for instance for $n+m=400$, the perturbation given by $\vee$ is 60.4, whereas the one given by $\sqcup$ is 48.4. In addition, we observe in Figure 7.8, that the operator $\sqcup$ leaves in general all the noise symbols alive, when $\vee$ cancels many of them (actually more than a half) by reducing their coefficient to zero (because of the argmin operator), which leads to a less precise but "lighter" affine forms.

Table 7.3 reconsiders the same examples seen before (see Section 7.3), but now with respect to $\sqcup$. Observe the clear improvement of the invariants

	Exact	Uncons. Taylor1+	Taylor1+ ($\vee$)	Taylor1+ ($\sqcup$)
InterQ1	$[0, 1875]$	$[0, 2500]$	$[0, 1875]$	$[0, 1875]$
Cosine	$[-1, 1]$	$[-1.073, 1]$	$[-1, 1]$	$[-1, 1]$
SinCos	$\{1\}$	$[0.86, 1.15]$	$[0.99, 1.00]$	$[0.99, 1.00]$
InterL2	$\{0.1\}$	$[-1, 1]$	$[0.1, 1]$	$[0.066, 0.4]$
InterQ2	$\{0.36\}$	$[-1, 1]$	$[-0.4, 1]$	$[-0.29, 0.52]$
InterQ2b	$[-0.1, 3]$	$[-0.1, 27]$	$[-0.1, 3.77]$	$[-0.1, 3.77]$

Table 7.3: $\vee$ vs $\sqcup$

of examples `InterL2` and `InterQ2` due to a more accurate affine form after the join using the $\sqcup$ operator. For instance, in `InterQ2`, we have to compute the join of $-1.25 + 10\epsilon_0 + 1.25\epsilon_2$ (the `if` branch) and $2.5 + 20\epsilon_0 + 2.5\epsilon_1$ (the `else` branch). The affine form obtained for after the join is as follows:

$$\begin{aligned} \vee &: -2.5 + 10\epsilon_0 + 7.5\eta_3 \\ \sqcup &: -0.625 + 13.75\epsilon_0 + 5.625\eta_3 \ . \end{aligned}$$

Indeed, $\sqcup$ makes a better repartition of the partial deviations in order to minimize the perturbation, and hence optimize the correlation between the input variables (encoded by the input noise symbols ϵ_i) and the final result. Therefore, when we compute the inverse of 1, the input noise symbol ϵ_0 is fixed, and the $\sqcup$ yields to a better result as it maximizes the contribution of that input to the final result.

Figure 7.9 depicts the two zonotopes found in the `if` and `else` branches of example `InterQ2`as well as the upper bound of these zonotopes given by the operator $\vee$. The projection on the variable y is $[-20, 15]$. It is slightly larger than the union of the projections on that variable, that is the union of $[-20, 5]$ and $[-2.5, 10]$, which gives $[-20, 10]$. In our implementation we use a reduced product with intervals to cancel such unnecessary over-approximation. In figure 7.10, we depict the final invariant of such reduced product, observe that the upper right corner of the original zonotope is truncated. The black shape of the same figure is the invariant given by the polyhedra abstract domain.

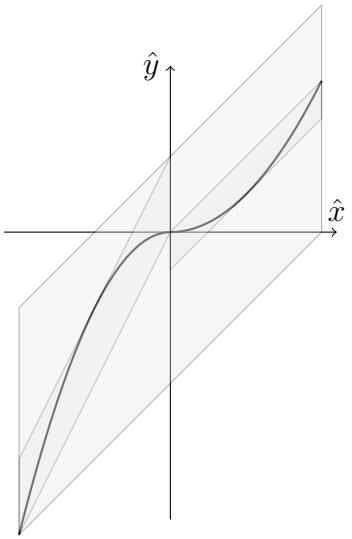

Figure 7.9: The if and else branches zonotopes and their join ($\vee$)

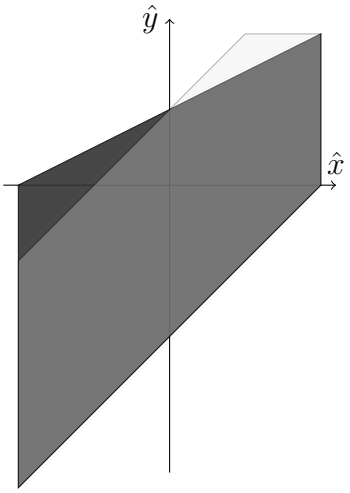

Figure 7.10: The final invariant of InterQ2: Reduced Product of T1+ and boxes (gray), Polyhedra (black)

CHAPTER 8

Conclusion

We have studied in this thesis the interesting combination of two different classes of abstract domains : the affine forms based abstract domain on one hand, and the polyhedra-like abstract domains on the other hand.

The affine sets abstract domain keeps and propagates implicitly (using noise symbols) the linear relations among variables. It shows a great efficiency of computation as well as accurate results for both linear and non-linear operations. The polyhedra-like family of abstract domains, includes polyhedra, zones, octagons and linear templates, is suitable to handle exactly the linear constraints among variables. We use this latter feature to address the interpretation of tests over affine sets, such as the intersection of a zonotope (the geometrical concretisation of affine sets) and a hyperplane. The exchange of information between the two abstract domains in use is formalized as a special logical product of these abstract domains. This particular combination, as shown in the experiments part, leads to finer invariants than the simple use of reduced product of affine sets and intervals.

Moreover, we have extended and generalized the componentwise join operators defined in the classical affine sets domains to the newly defined domain. We have characterized a particular set of minimal upper bounds which minimize the perturbation, and have presented an algorithm, with a cubical complexity in time, to compute these upper bounds. Another algorithm, with a linear complexity, was also defined following ideas from the classical perturbed affine sets domain. The latter algorithm trades the minimality of the perturbation with the cost of computations and therefore could be useful either as a first trial analysis or as a convergence accelerator.

The global approach for the join operators (instead of the component-

wise approach detailed in this work) is an important direction we would like to explore as a future work. First for the perturbed affine sets, following the work already done by Goubault and Putot, then for their constrained variant.

Another non-less interesting avenue for the future work could be the abstraction of the coefficients (the partial deviations) of affine sets, as already done for the noise symbols. Such an approach could be relevant to infer non-convex invariants. In fact, the generalized intervals (see definition 3.3.1 of Section 3.3) use intervals as coefficients. As mentioned by Stolfi, this constitutes a fundamental difference between generalized intervals and affine forms, not only because the latter uses real numbers as coefficients, but also because the joint-range of generalized intervals is not convex, whereas the joint-range of affine form is a special polyhedron (zonotope). This convexity property is crucial in affine forms as defined and used in our work. Nevertheless, the non-convexity property may also be attractive and desired as it permits to catch some non-convex invariants. This characteristic was for instance exploited in the recent work of interval polyhedra abstract domain of Chen and al. [CMWC09].

A

The Support Function

We gather in this appendix some properties (without proofs) of the support function and its dual (or conjugate), the indicator function. An introduction to these particular functions, as well as detailed proofs may be found for instance in [Roc70].

Recall the definition of the support function.

A.0.1 Definition
Let C be a non-empty convex set of $\mathbb{R}^n$, then

$$\delta(t \mid C) \stackrel{\text{def}}{=} \sup\big\{\langle t, x\rangle \mid x \in C\big\},$$

where $\langle \cdot, \cdot \rangle$ denotes the usual scalar product over $\mathbb{R}^n$.

The belonging of a vector to a closed convex set may be characterized using the support function.

A.0.2 Proposition
Let C be a closed convex set. Then $x \in C$ if and only if

$$\langle t, x\rangle \leq \delta(t \mid C),$$

for every vector $t \in \mathbb{R}^n$.

The support function over a symmetric convex set is symmetric itself. A convex set is said to be symmetric if and only if $x \in C \implies -x \in C$. If $-C \stackrel{\text{def}}{=} \{x \mid -x \in C\}$, then C is symmetric if and only if $C = -C$.

A.0.3 Proposition
Let $C \subset \mathbb{R}^n$ be a convex set, then

$$\delta(t \mid -C) = \delta(-t \mid C) \ .$$

Moreover, the support function of a symmetric convex set is symmetric, that is, if $-C = C$, we have $\delta(t \mid C) = \delta(-t \mid C)$ for all $t \in \mathbb{R}^n$.

Proof. *By definition, $-x \in C$ if and only if $x \in -C$. The result is then immediate from Proposition A.0.2.* ∎

Proposition A.0.3 makes it possible to "move" the minus sign from the variable to the convex set and vice versa; moreover, if the convex set is symmetric, the minus sign can be absorbed by the symmetric convex set.

2 Remark
The support function of the sum of two convex sets can be expanded to the sum of the support functions of each operand of the sum. Indeed, for two non-empty convex sets $C_1, C_2 \subset \mathbb{R}^n$, one has $\delta(t \mid C_1+C_2) = \delta(t \mid C_1)+\delta(t \mid C_2)$ for all $t \in \mathbb{R}^n$. The proof is straightforward by the linearity of the scalar product:

$$\begin{aligned}\delta(t \mid C_1 + C_2) &= \sup\{\langle t, x\rangle \mid x \in C_1 + C_2\} \\ &= \sup\{\langle t, x_1 + x_2\rangle \mid x_1 \in C_1, x_2 \in C_2\} \\ &= \sup\{\langle t, x_1\rangle \mid x_1 \in C_1\} + \sup\{\langle t, x_2\rangle \mid x_2 \in C_2\} \\ &= \delta(t \mid C_1) + \delta(t \mid C_2)\end{aligned}$$

The support function verifies the triangle inequality.

A.0.4 Proposition
The support function $\delta(t \mid C)$ verifies

$$\delta(t_1 + t_2 \mid C) \leq \delta(t_1 \mid C) + \delta(t_2 \mid C), \quad \forall t_1, \forall t_2$$

The last proposition evaluates the composition of the support function and a linear transformation A from $\mathbb{R}^m$ to $\mathbb{R}^n$.

A.0.5 Proposition
Let A be a linear transformation from $\mathbb{R}^m$ to $\mathbb{R}^n$. For any convex set $C \subset \mathbb{R}^n$, one has

$$\delta(At \mid C) = \delta(t \mid A^*C), \quad \forall t \in \mathbb{R}^m,$$

where A^* denotes the transpose matrix of the matrix A.

The support function respects the positive homogeneity property.

A.0.6 Proposition
For any non-empty convex set C, one has $\delta(x \mid \lambda C) = \lambda\delta(x \mid C)$, $0 \leq \lambda < +\infty$.

The remaining properties concern the dual operations.
The first property gives the conjugate of partial affine functions.

A.0.7 Proposition
Let h be a convex function on $\mathbb{R}^n$, and let

$$f(x) = h(A(x-a)) + \langle x, b\rangle + \alpha,$$

where A is a one-to-one linear transformation from $\mathbb{R}^n$ onto $\mathbb{R}^n$, a and b are vectors in $\mathbb{R}^n$, and $\alpha \in \mathbb{R}$. Then

$$f^*(t) = h^*(A^{*-1}(t-b)) + \langle t, a\rangle + \alpha^*,$$

where A^* is the adjoint of A and $\alpha^* = -\alpha - \langle a, b\rangle$.

The conjugate function of a sum is defined using the infimal convolution.

A.0.8 Proposition
Let $f_1, \ldots, f_m$ be proper convex functions on $\mathbb{R}^p$. Then

$$(f_1 \Diamond \cdots \Diamond f_m)^* = f_1^* + \cdots + f_m^*,$$
$$(\operatorname{cl} f_1 + \cdots + \operatorname{cl} f_m)^* = \operatorname{cl}(f_1^* \Diamond \cdots \Diamond f_m^*).$$

If the sets $\operatorname{ri}(\operatorname{dom} f_i)$, $i = 1, \ldots, m$ *have a point in common, the closure operation can be omitted from the second formula, and*

$$(f_1 + \cdots + f_m)^* = \inf\{f_1^*(x_1^*) + \cdots + f_m^*(x_m^*) \mid x_1^* + \cdots + x_m^* = x^*\},$$

where for each x^* the infimum is attained.

The dual of a positive scalar multiplication multiply the epigraph of the function (called, and denoted as, right multiplication).

A.0.9 Proposition
For any proper convex function f, one has $(\lambda f)^* = f^*\lambda$, and $(f\lambda)^* = \lambda f^*$, $0 \le \lambda \le +\infty$.

B

Lengthy Proofs

B.1 Lemma 6.2.18: Fenchel Conjugate of $L_{\bar{\lambda}}$

Proof. *of Lemma 6.2.18. We compute the conjugate of $L_{\bar{\lambda}}$ at 0, as the conjugate of the sum of*

$$L1_{\bar{\lambda}}(\alpha) \stackrel{\text{def}}{=} \bar{\lambda}(\delta(M^X(\alpha - \alpha^X) \mid \mathfrak{B}) + \tau^X)$$
$$L2_{\bar{\lambda}}(\alpha) \stackrel{\text{def}}{=} (1-\bar{\lambda})(\delta(M^Y(\alpha - \alpha^Y) \mid \mathfrak{B}) + \tau^Y)$$

The conjugate of a sum of convex functions is ruled by Proposition A.0.8. Functions $L1_{\bar{\lambda}}$ and $L2_{\bar{\lambda}}$ are proper convex functions, indeed, they are finite for a subset of $\mathbb{R}^n$, and $L1_{\bar{\lambda}}(\alpha) > -\infty$, $L2_{\bar{\lambda}}(\alpha) > -\infty$ for every α. Moreover, $\mathrm{ri}(\mathrm{dom}\, L1_{\bar{\lambda}}) = \mathrm{ri}(\mathrm{dom}\, L2_{\bar{\lambda}}) = \mathbb{R}^n$. Therefore, for all $\alpha \in \mathbb{R}^n$:

$$L1_{\bar{\lambda}} + L2_{\bar{\lambda}}{}^*(\alpha) = \inf\{L1^*_{\bar{\lambda}}(\alpha_1) + L2^*_{\bar{\lambda}}(\alpha_2) \mid \alpha_1 + \alpha_2 = \alpha\} \ .$$

To evaluate $L^*_{\bar{\lambda}}(0)$, we need to compute the conjugates of $L1_{\bar{\lambda}}$ and $L2_{\bar{\lambda}}$. Then apply the previous formula for $\alpha = 0$. We detail the computation of the conjugate of the convex function $L1_{\bar{\lambda}}$, the computation of $L2_{\bar{\lambda}}$ is similar. The function $L1_{\bar{\lambda}}$ is defined as a scalar multiplication of a composition of the support function and a linear transformation, the multiplication by M^X and a translation by $-\alpha^X$. By Proposition A.0.9:

$$L1^*_{\bar{\lambda}}(\alpha) = (\bar{\lambda}(\delta(M^X(\alpha - \alpha^X) \mid \mathfrak{B}) + \tau^X))^*(\alpha) \qquad \text{(B.1.1)}$$
$$= (((\delta(M^X(\alpha - \alpha^X) \mid \mathfrak{B}) + \tau^X))^* \bar{\lambda})(\alpha) \qquad \text{(B.1.2)}$$

Matrix M^X is non-singular by construction, so by Proposition A.0.7, with $h \leftrightarrow \delta$, $A \leftrightarrow M^X$, $a \leftrightarrow \alpha^X$, $b \leftrightarrow 0$, $\theta \leftrightarrow \tau^X$:

$$(\delta(M^X(\alpha - \alpha^X) \mid \mathfrak{B}) + \tau^X)^*(\alpha) = \delta^*(M^{X*-1}(\alpha) \mid \mathfrak{B}) + \langle \alpha, \alpha^X \rangle - \tau^X \ .$$

The function δ^* is the indicator function, conjugate of the support function δ (see Definition 6.2.15). The right multiplication by $\bar{\lambda}$ in (B.1.2) can now be evaluated:

$$((\delta(M^X(\alpha - \alpha^X) \mid \mathfrak{B}) + \tau^X)^* \bar{\lambda})(\alpha) \\ = \bar{\lambda}(\delta^*(M^{X^{*-1}}(\bar{\lambda}^{-1}\alpha) \mid \mathfrak{B}) + \langle \bar{\lambda}^{-1}\alpha, \alpha^X \rangle - \tau^X) \ .$$

We compute $L2^*_{\bar{\lambda}}$ in a similar manner by substituting X with Y and $\bar{\lambda}$ by $(1 - \bar{\lambda})$.

$$\begin{array}{ll} L1^*_{\bar{\lambda}}(\alpha) & = \bar{\lambda}(\delta^*(M^{X^{*-1}}(\bar{\lambda}^{-1}\alpha) \mid \mathfrak{B}) + \langle \bar{\lambda}^{-1}\alpha, \alpha^X \rangle - \tau^X) \\ L2^*_{\bar{\lambda}}(\alpha) & = (1-\bar{\lambda})(\delta^*(M^{Y^{*-1}}((1-\bar{\lambda})^{-1}\alpha) \mid \mathfrak{B}) + \langle (1-\bar{\lambda})^{-1}\alpha, \alpha^Y \rangle - \tau^Y) \end{array}$$

We proceed by evaluating $L^*_{\bar{\lambda}}(0)$.

$$\begin{aligned} L^*_{\bar{\lambda}}(0) &= \inf_{\alpha \in \mathbb{R}^n} \{L1^*_{\bar{\lambda}}(\alpha) + L2^*_{\bar{\lambda}}(-\alpha)\} \\ &= \inf_{\alpha \in \mathbb{R}^n} \{\bar{\lambda}(\delta^*(M^{X^{*-1}}(\bar{\lambda}^{-1}\alpha) \mid \mathfrak{B}) + \langle \bar{\lambda}^{-1}\alpha, \alpha^X \rangle - \tau^X) + \\ &\quad + (1-\bar{\lambda})(\delta^*(M^{Y^{*-1}}(-(1-\bar{\lambda})^{-1}\alpha) \mid \mathfrak{B}) + \langle -(1-\bar{\lambda})^{-1}\alpha, \alpha^Y \rangle - \tau^Y)\} \\ &= \inf_{\alpha \in \mathbb{R}^n} \{\bar{\lambda}(\delta^*(\alpha \mid \bar{\lambda} M^{X^*}\mathfrak{B}) + \langle \alpha, \alpha^X \rangle - \bar{\lambda}\tau^X + \\ &\quad + (1-\bar{\lambda})(\delta^*(\alpha \mid (1-\bar{\lambda})M^{Y^*}\mathfrak{B}) - \langle \alpha, \alpha^Y \rangle - (1-\bar{\lambda})\tau^Y\} \\ &= \inf_{\alpha \in \bar{\lambda}M^{X^*}\mathfrak{B} \cap (1-\bar{\lambda})M^{Y^*}\mathfrak{B}} \{\langle \alpha, \alpha^X - \alpha^Y \rangle - \bar{\lambda}\tau^X - (1-\bar{\lambda})\tau^Y\} \\ &= -\delta(\alpha^X - \alpha^Y \mid \bar{\lambda}M^{X^*}\mathfrak{B} \cap (1-\bar{\lambda})M^{Y^*}\mathfrak{B}) - \bar{\lambda}\tau^X - (1-\bar{\lambda})\tau^Y \ . \quad \blacksquare \end{aligned}$$

B.2 Theorem 6.2.20: Saddle-Point Characterization

Proof. *of Theorem 6.2.20. We remind that $L(\alpha, \lambda)$ is a linear function with respect to λ denoted by $a_\alpha \lambda + b_\alpha$ where:*

$$\begin{aligned} a_\alpha &= \delta(M^X(\alpha^X - \alpha) \mid \mathfrak{B}) + \tau^X - \delta(M^Y(\alpha^Y - \alpha) \mid \mathfrak{B}) - \tau^Y \\ b_\alpha &= \delta(M^Y(\alpha^Y - \alpha) \mid \mathfrak{B}) + \tau^Y \ . \end{aligned}$$

We detail the first cases, that is when $\delta(M^X(\alpha^Y - \alpha^X) \mid \mathfrak{B}) < \tau^Y - \tau^X$, and $\delta(M^X(\alpha^Y - \alpha^X) \mid \mathfrak{B}) = \tau^Y - \tau^X$. The third and fourth cases are similar. The last case is a combination of Propositions 6.2.13 and 6.2.19.

First case: $\delta(M^X(\alpha^Y - \alpha^X) \mid \mathfrak{B}) < \tau^Y - \tau^X$. We prove that $(\alpha^Y, 0)$ is a saddle-point of L. Then, we prove that it is the unique saddle-point. We know that, for all $\alpha \in \mathbb{R}^{n+1}$, $\delta(M^Y(\alpha - \alpha^Y) \mid \mathfrak{B})$ is non-negative, thus

$$\forall \alpha \in \mathbb{R}^{n+1}, \quad \tau^Y \leq \delta(M^Y(\alpha - \alpha^Y) \mid \mathfrak{B}) + \tau^Y \ . \tag{B.2.1}$$

On the other hand, by hypothesis,

$$\delta(M^X(\alpha^Y - \alpha^X) \mid \mathfrak{B}) + \tau^X < \tau^Y,$$

we multiply this inequality by λ, then add $(1-\lambda)\tau^Y$ in both sides of the inequality, we obtain

$$\forall \lambda \in [0,1], \quad \lambda(\delta(M^X(\alpha^Y - \alpha^X) \mid \mathfrak{B}) + \tau^X) + (1-\lambda)\tau^Y < \tau^Y \ . \tag{B.2.2}$$

We combine equations (B.2.1) and (B.2.2):

$$\forall \alpha \in \mathbb{R}^{n+1}, \forall \lambda \in [0,1],$$
$$\lambda(\delta(M^X(\alpha^Y - \alpha^X) \mid \mathfrak{B}) + \tau^X) + (1-\lambda)\tau^Y < \tau^Y \leq \delta(M^Y(\alpha - \alpha^Y) \mid \mathfrak{B}) + \tau^Y,$$

which is equivalent to

$$\forall \alpha \in \mathbb{R}^{n+1}, \forall \lambda \in [0,1], \quad L(\alpha^Y, \lambda) \leq L(\alpha^Y, 0) \leq L(\alpha, 0),$$

thus, $(\alpha^Y, 0)$ is a saddle-point of L. We next prove, by contradiction, that it is the unique saddle-point of L when the considered hypothesis is verified. Suppose that $(\bar{\alpha}, \bar{\lambda}) \neq (\alpha^Y, 0)$ is saddle-point of L, then $L(\bar{\alpha}, \bar{\lambda}) \leq L(\alpha^Y, \bar{\lambda})$. If $a_{\bar{\alpha}} > 0$, then by Proposition 6.2.13, $\bar{\lambda} = 1$. Therefore, $L(\bar{\alpha}, \bar{\lambda}) = \delta(M^X(\alpha^X - \bar{\alpha}) \mid \mathfrak{B}) + \tau^X$, and $L(\alpha^Y, \bar{\lambda}) = \delta(M^X(\alpha^X - \alpha^Y) \mid \mathfrak{B}) + \tau^X$. Thus

$$\delta(M^X(\alpha^X - \bar{\alpha}) \mid \mathfrak{B}) + \tau^X \leq \delta(M^X(\alpha^X - \alpha^Y) \mid \mathfrak{B}) + \tau^X,$$

However, $a_{\bar{\alpha}} > 0$ implies:

$$\delta(M^X(\alpha^X - \bar{\alpha}) \mid \mathfrak{B}) + \tau^X > \delta(M^Y(\alpha^Y - \bar{\alpha}) \mid \mathfrak{B}) + \tau^Y$$

Thus,

$$\delta(M^X(\alpha^X - \alpha^Y) \mid \mathfrak{B}) + \tau^X > \delta(M^Y(\alpha^Y - \bar{\alpha}) \mid \mathfrak{B}) + \tau^Y$$

which leads to the contradiction

$$\delta(M^Y(\alpha^Y - \bar{\alpha}) \mid \mathfrak{B}) < 0 \ .$$

Indeed, for all x, $\delta(x \mid \mathfrak{B}) = \|x\|_1 \geq 0$. *Now, if* $a_{\bar{\alpha}} = 0$, *then* $L(\bar{\alpha}, \bar{\lambda}) \leq L(\alpha^Y, \bar{\lambda})$ *gives*

$$\delta(M^Y(\alpha^Y - \bar{\alpha}) \mid \mathfrak{B}) + \tau^Y \leq \tau^Y,$$

which makes $\bar{\alpha} = \alpha^Y$, *and* $a_{\bar{\alpha}} = 0$ *contradicts the hypothesis, as it makes*

$$\delta(M^X(\alpha^X - \alpha^Y) \mid \mathfrak{B}) + \tau^X = \tau^Y \quad .$$

The last case, $a_{\bar{\alpha}} < 0$, *makes* $\bar{\lambda} = 0$ *by Proposition 6.2.13. Using Proposition 6.2.19, we obtain* $\bar{\alpha} = \alpha^Y$, *which also contradicts the hypothesis* $(\bar{\alpha}, \bar{\lambda}) \neq (\alpha^Y, 0)$. *Thus, the unique saddle-point is indeed* $(\alpha^Y, 0)$. *The saddle-value of* L *is then* $L(\alpha^Y, 0) = \tau^Y$.

Second case: $\delta(M^X(\alpha^Y - \alpha^X) \mid \mathfrak{B}) = \tau^Y - \tau^X$. *If* $\alpha^Y = \alpha^X$ *and* $\tau^Y = \tau^X$, *then it is obvious that* $\bar{\alpha} = \alpha^X = \alpha^Y$ *and* $\bar{\lambda}$ *can be any real number within* $[0, 1]$. *The saddle-value is then equal to* τ^X *(or* τ^Y*). If however,* $\alpha^Y = \alpha^X$ *and* $\tau^Y \neq \tau^X$, *the hypothesis is not satisfied. Similarly, if* $\alpha^Y \neq \alpha^X$ *and* $\tau^Y = \tau^X$. *Now, if* $\alpha^Y \neq \alpha^X$ *and* $\tau^Y \neq \tau^X$, *we prove that* L *admits infinitely many saddle-points, such that* $\bar{\alpha} = \alpha^Y$, *and that its saddle-value is equal to* τ^Y. *Seen as a linear function, observe that, by hypothesis,* $a_{\alpha^Y} = 0$. *Thus*

$$\forall \lambda \in [0,1], \forall \bar{\lambda} \in [0,1], \quad L(\alpha^Y, \lambda) = L(\alpha^Y, \bar{\lambda}) = b_{\alpha^Y} \quad .$$

On the other hand,

$$L(\alpha^Y, \bar{\lambda}) = \tau^Y \leq L(\alpha, \bar{\lambda}),$$

indeed $L(\alpha, \bar{\lambda})$ *can be written as* τ^Y *plus a positive term:*

$$L(\alpha, \bar{\lambda}) = \bar{\lambda}\delta(M^X(\alpha^X - \alpha) \mid \mathfrak{B}) + (1-\bar{\lambda})\delta(M^Y(\alpha^Y - \alpha) \mid \mathfrak{B}) + \bar{\lambda}(\tau^X - \tau^Y) + \tau^Y,$$

where the positiveness of $(\tau^X - \tau^Y)$ *is due to the equality* $\delta(M^X(\alpha^Y - \alpha^X) \mid \mathfrak{B}) = \tau^Y - \tau^X$ *and the positiveness of* $\delta(M^X(\alpha^Y - \alpha^X) \mid \mathfrak{B})$. *Thus,*

$$\forall \alpha \in \mathbb{R}^{n+1}, \forall \lambda \in [0,1], \forall \bar{\lambda} \in [0,1], \quad L(\alpha^Y, \lambda) \leq L(\alpha^Y, \bar{\lambda}) \leq L(\alpha, \bar{\lambda}),$$

which makes all the points $(\alpha^Y, \bar{\lambda})$, $\bar{\lambda} \in [0, 1]$, *saddle-points of* L. *The saddle-value related to all these saddle-points is* τ^Y.

Third and forth cases. The proof is very similar to the first and second cases respectively by exchanging X *by* Y, *and* $\bar{\lambda}$ *by* $(1 - \bar{\lambda})$.

Last case:

$$\delta(M^X(\alpha^Y - \alpha^X) \mid \mathfrak{B}) \geq |\tau^Y - \tau^X|$$
$$\delta(M^Y(\alpha^Y - \alpha^X) \mid \mathfrak{B}) \geq |\tau^Y - \tau^X| \quad .$$

We just combine Propositions 6.2.13 and 6.2.19. We prove in addition that $\bar{\lambda} \in]0, 1[$, *that is the values* 0 *and* 1 *are excluded when this hypothesis*

is satisfied. If $\bar{\lambda} = 1$*, then by Proposition 6.2.19,* $\bar{\alpha} = \alpha^X$*, which makes* $a_{\bar{\alpha}} = \tau^X - \tau^Y - \delta(M^Y(\alpha^Y - \alpha^X) \mid \mathfrak{B})$ *negative by hypothesis. By Proposition 6.2.13,* $a_{\bar{\lambda}} < 0$ *implies the contradiction* $\bar{\lambda} = 0$*, thus* $a_{\bar{\lambda}} = 0$*, and necessarily* $\tau^X - \tau^Y = \delta(M^Y(\alpha^Y - \alpha^X) \mid \mathfrak{B})$*, which also contradicts the strict inequality of the hypothesis. Therefore,* $\bar{\lambda} = 1$ *is impossible. Similarly, we prove that* $\bar{\lambda} = 0$ *is also impossible.* ∎

Bibliography

[AGG08] Xavier Allamigeon, Stéphane Gaubert, and Eric Goubault. Inferring Min and Max Invariants Using Max-plus Polyhedra. In María Alpuente and Germán Vidal, editors, *Proceedings of the 15th International Static Analysis Symposium (SAS'08)*, volume 5079 of *Lecture Notes in Computer Science*, pages 189–204, Valencia, Spain, July 2008. Springer Verlag.

[AGG10] Assalé Adjé, Stephane Gaubert, and Eric Goubault. Coupling policy iteration with semi-definite relaxation to compute accurate numerical invariants in static analysis. In *ESOP*, pages 23–42, 2010.

[apr07] Numerical abstract domain library, 2007. http://apron.cri.ensmp.fr.

[ASB08] Matthias Althoff, Olaf Stursberg, and Martin Buss. Verification of uncertain embedded systems by computing reachable sets based on zonotopes. In *IFAC World Congress*, pages 5125–5130, 2008.

[Ast] Astre. real-time embedded software static analyzer. http://www.astree.ens.fr.

[Bau88] Eckart Baumann. Optimal centered forms. *BIT Numerical Mathematics*, 28:80–87, 1988.

[BCC+09] O. Bouissou, E. Conquet, P. Cousot, R. Cousot, J. Feret, K. Ghorbal, E. Goubault, D. Lesens, L. Mauborgne, A. Miné, S. Putot, X. Rival, and M. Turin. Space software validation using abstract interpretation. In *Proc. of the Int. Space System Engineering Conf., Data Systems in Aerospace (DASIA 2009)*, volume SP-669, pages 1–7, Istambul, Turkey, May 2009. ESA. http://www.lix.polytechnique.fr/Labo/Khalil.Ghorbal/publi/bouissou-al-dasia09.pdf.

[BCC+10] J. Bertrane, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, and X. Rival. Static analysis and verification

of aerospace software by abstract interpretation. In *AIAA Infotech@Aerospace (I@A 2010)*, pages 1–38. AIAA (American Institute of Aeronautics and Astronautics), Apr. 2010. http://www.di.ens.fr/~mine/publi/bertrane-al-aiaa10.pdf.

[BF07] Sylvie Boldo and Jean-Christophe Filliatre. Formal verification of floating-point programs. *Computer Arithmetic, IEEE Symposium on*, 0:187–194, 2007.

[Boo] Boost. *BOOST C++ Libraries*. `http://www.boost.org`.

[Bor99] B. Borchers. A C library for Semidefinite Programming, 1999. https://projects.coin-or.org/Csdp.

[CC77] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 238–252, Los Angeles, California, 1977. ACM Press, New York, NY.

[CC79] Patrick Cousot and Radhia Cousot. Systematic design of program analysis frameworks. In *POPL*, pages 269–282, 1979.

[CGG+05] A. Costan, S. Gaubert, E. Goubault, M. Martel, and S. Putot. A policy iteration algorithm for computing fixed points in static analysis of programs. *Computer Aided Verification*, 2005.

[CH78] Patrick Cousot and Nicolas Halbwachs. Automatic discovery of linear restraints among variables of a program. In *POPL*, pages 84–96, 1978.

[CM72] William Chuba and Webb Miller. Quadratic convergence in interval arithmetic, part i. *BIT Numerical Mathematics*, 12:284–290, 1972.

[CMC08] L. Chen, A. Miné, and P. Cousot. A sound floating-point polyhedra abstract domain. In *Proc. of the Sixth Asian Symposium on Programming Languages and Systems (APLAS'08)*, volume 5356 of *LNCS*, pages 3–18, Bangalore, India, December 2008. Springer. http://www.di.ens.fr/~mine/publi/article-chen-al-aplas08.pdf.

[CMWC09] L. Chen, A. Miné, J. Wang, and P. Cousot. Interval polyhedra: An abstract domain to infer interval linear relationships. In *Proc. of the 16th Int. Static Analysis Symposium (SAS'09)*, volume 5673 of *LNCS*, pages 309–325, Los Angeles, CA, USA, August 2009. Springer. http://www.di.ens.fr/~mine/publi/article-chen-al-sas09.pdf.

[Com05] C. Combastel. A state bounding observer for uncertain nonlinear continuous-time systems based on zonotopes. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 7228 – 7234, dec. 2005.

[coq] The coq proof assistant. http://coq.inria.fr.

[Cou02] P. Cousot. Constructive design of a hierarchy of semantics of a transitio system by abstract interpretation. *Theoretical Computer Science*, 277(1–2):47–103, 2002.

[Cou05] Patrick Cousot. Iterative reduced product, 2005. http://web.mit.edu/16.399/www/.

[CS93] João L. D. Comba and Jorge Stolfi. Affine arithmetic and its applications to computer graphics. *SIBGRAPI'93*, 1993.

[dDLM06] Florent de Dinechin, Christoph Quirin Lauter, and Guillaume Melquiond. Assisted verification of elementary functions using gappa. In *Proceedings of the 2006 ACM symposium on Applied computing*, SAC '06, pages 1318–1322. ACM, 2006.

[dFS97] Luiz H. de Figueiredo and Jorge Stolfi. *Self-Validated Numerical Methods and Applications*. Brazilian Mathematics Colloquium monographs. IMPA/CNPq, Rio de Janeiro, Brazil, 1997.

[DGP+09] D. Delmas, E. Goubault, S. Putot, J. Souyris, K. Tekkal, and F. Védrine. Towards an industrial use of FLUCTUAT on safety-critical avionics software. In *Proceedings of Formal Methods in Industrial Critical Systems, LNCS 5825*, pages 53–69. Springer-Verlag, 2009.

[E.W66] Cheney E.W. *Introduction to approximation theory*. McGraw-Hill Book Co. (New York), 1966.

[Fla88] B. Flavigny. A new machine providing accuracy estimates of computation results. In *12th IMACS Congress, Paris*, 1988.

[Flu] Fluctuat. Static analysis for numerical precision. http://www-list.cea.fr/labos/gb/LSL/fluctuat/index.html.

[FMH] Jean-Christophe Fillitre, Claude March, and Thierry Hubert. The caduceus tool for the verification of c programs. http://caduceus.lri.fr.

[Gap] Gappa. Gnration automatique de preuves de proprits arithmtiques. http://gappa.gforge.inria.fr/.

[GGP09] K. Ghorbal, E. Goubault, and S. Putot. The zonotope abstract domain Taylor1+. In *Proc. of the 21th Int. Conf. on Computer Aided Verification (CAV 2009)*, volume 5643 of *Lecture Notes in Computer Science*, pages 627–633, Grenoble, France, June 2009. Springer. http://www.lix.polytechnique.fr/Labo/Khalil.Ghorbal/publi/ghorbal-cav09.pdf.

[GGP10] K. Ghorbal, E. Goubault, and S. Putot. A logical product approach to zonotope intersection. In *Proc. of the 22th Int. Conf. on Computer Aided Verification (CAV 2010)*, Lecture Notes in Computer Science, Edinburgh, UK, July 2010. Springer. http://www.lix.polytechnique.fr/Labo/Khalil.Ghorbal/publi/ghorbal-cav10.pdf.

[Gir05] Antoine Girard. Reachability of uncertain linear systems using zonotopes. In *Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 291–305. Springer Berlin / Heidelberg, 2005.

[GLG08] Antoine Girard and Colas Le Guernic. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In Magnus Egerstedt and Bud Mishra, editors, *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science, pages 215–228. Springer Berlin / Heidelberg, 2008.

[GMP02] Eric Goubault, Matthieu Martel, and Sylvie Putot. Asserting the precision of floating-point computations: A simple abstract interpreter. In *Proceedings of the 11th European Symposium on Programming Languages and Systems*, ESOP '02, pages 209–212, London, UK, UK, 2002. Springer-Verlag.

[GMP06] E. Goubault, M. Martel, and S. Putot. Some future challenges in the validation of control systems. In *European Congress on Embedded Real Time Software (ERTS)*, 2006.

[gnu] Gnucash, free accounting software. http://www.gnucash.org.

[GNZ03] Leonidas J. Guibas, An Nguyen, and Li Zhang. Zonotopes as bounding volumes. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '03, pages 803–812, 2003.

[Gol91] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23, No 1, March 1991.

[Gou01a] Eric Goubault. Static analyses of the precision of floating-point operations. In *SAS*, pages 234–259, 2001.

[Gou01b] Eric Goubault. Static analyses of the precision of floating-point operations. In Patrick Cousot, editor, *Static Analysis*, volume 2126 of *Lecture Notes in Computer Science*, pages 234–259. Springer Berlin / Heidelberg, 2001.

[GP06] Eric Goubault and Sylvie Putot. Static analysis of numerical algorithms. In *SAS*, pages 18–34, 2006.

[GP08] Eric Goubault and Sylvie Putot. Perturbed affine arithmetic for invariant computation in numerical program analysis. *CoRR*, abs/0807.2961, 2008.

[GP09] Eric Goubault and Sylvie Putot. A zonotopic framework for functional abstractions. *CoRR*, abs/0910.1763, 2009.

[GP11] Eric Goubault and Sylvie Putot. Static analysis of finite precision computations. In Ranjit Jhala and David Schmidt, editors, *Verification, Model Checking, and Abstract Interpretation*, volume 6538 of *Lecture Notes in Computer Science*, pages 232–247. Springer Berlin / Heidelberg, 2011.

[GPBG07] E. Goubault, S. Putot, P. Baufreton, and J. Gassino. Static analysis of the accuracy in control systems: Principles and experiments. In *Proceedings of Formal Methods in Industrial Critical Systems, LNCS 4916*. Springer-Verlag, 2007.

[GT06] Sumit Gulwani and Ashish Tiwari. Combining abstract interpreters. In *PLDI*, pages 376–386, 2006.

[Han69] E. R. Hansen. The centered form. *Topics in Interval Analysis*, pages 102–106, 1969.

[Han75] Eldon R. Hansen. A generalized interval arithmetic. In *Interval Mathematics*, volume 29 of *LNCS*, pages 7–18. Springer, 1975.

[Hoa83] C. A. R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 26:53–56, January 1983.

[IEE85] IEEE Standards Committee 754. *IEEE Standard for binary floating-point arithmetic, ANSI/IEEE Standard 754-1985*. Institute of Electrical and Electronics Engineers, New York, 1985. Reprinted in SIGPLAN Notices, 22(2):9-25, 1987.

[IEE08] IEEE. *IEEE Std. 754^{TM}-2008 Standard for Floating-Point Arithmtic*. IEEE, 3 Park Avenue, NY 10016-5997, USA, 2008.

[Ja] B. Jeannet and al. Newpolka library. http://www.inrialpes.fr/pop-art/people/bjeannet/newpolka.

[JC08] Fabienne Jzquel and Jean-Marie Chesneaux. Cadna: a library for estimating round-off error propagation. *Computer Physics Communications*, 178(12):933 – 955, 2008.

[JCK07] C. Jansson, D. Chaykin, and C. Keil. Rigorous error bounds for the optimal value in semidefinite programming. *SIAM J. Numer. Anal.*, 46(1):180–200, 2007.

[Jea90] Vignes Jean. Estimation de la prcision des rsultats de logiciels numriques, 1990.

[JM88] Chesneaux J.-M. Modelisation et conditions de validit de la mthode cestac, 1988.

[JP89] Faye J.-P. Implementation synchrone de cestac, 1989.

[Kag86] H Kagiwada. *Numerical derivatives and nonlinear analysis*. Plenum Press, New York, NY, USA, 1986.

[Kar76] Michael Karr. Affine relationships among variables of a program. *Acta Inf.*, 6:133–151, 1976.

[Kar84] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–396, 1984.

[Kei05] C. Keil. Lurupa - rigorous error bounds in linear programming. In *Algebraic and Numerical Algorithms and Computer-assisted Proofs, Dagstuhl Seminar 5391*, 2005.

[Kei08] C. Keil. A comparison of software packages for verified linear programming, 2008.

[Kol97] Lubomir Kolev. Use of interval slopes for the irrational part of factorable functions. *Reliable Computing*, 3:83–93, 1997.

[Kol01] Lubomir V. Kolev. Automatic computation of a linear interval enclosure. *Reliable Computing*, 7:17–28, 2001.

[Kol04] Lubomir V. Kolev. An improved interval linearization for solving nonlinear problems. *Numerical Algorithms*, 37:213–224, 2004.

[Kol07] Lubomir V. Kolev. Optimal multiplication of g-intervals. *Reliable Computing*, 13(5):399–408, 2007.

[Küh98] Wolfgang Kühn. Zonotope dynamics in numerical quality control. In Hans-Christian Hege and Konrad Polthier, editors, *Visualization and Mathematics*, pages 125–134. Springer Verlag, Heidelberg, 1998.

[Le 92] Harvey Le Verge. A note on Chernikova's algorithm. Technical Report 635, IRISA, Rennes, France, February 1992.

[LG09] C. Le Guernic. *Reachability analysis of hybrid systems with linear continuous dynamics.* PhD thesis, Université Grenoble 1 - Joseph Fourier, Grenoble, France, 2009.

[LL09] Vincent Laviron and Francesco Logozzo. Subpolyhedra: A (more) scalable approach to infer linear inequalities. In *VMCAI*, pages 229–244, 2009.

[Mar02] Matthieu Martel. Propagation of roundoff errors in finite precision computations: A semantics approach. In *Proceedings of the 11th European Symposium on Programming Languages and Systems*, ESOP '02, pages 194–208. Springer-Verlag, 2002.

[Mes02] Frédérique Messine. Extensions of affine arithmetic: Application to unconstrained global optimization. *Journal of Universal Computer Science*, 8:992–1015, 2002.

[Min01] A. Miné. The octagon abstract domain. In *Proc. of the Workshop on Analysis, Slicing, and Transformation (AST'01)*, IEEE, pages 310–319, Stuttgart, Gernamy, October

2001. IEEE CS Press. http://www.di.ens.fr/~mine/publi/article-mine-ast01.pdf.

[Min04a] A. Miné. Relational abstract domains for the detection of floating-point run-time errors. In *Proc. of the European Symposium on Programming (ESOP'04)*, volume 2986 of *Lecture Notes in Computer Science*, pages 3–17. Springer, 2004.

[Min04b] A. Miné. *Weakly Relational Numerical Abstract Domains*. PhD thesis, École Polytechnique, Palaiseau, France, December 2004.

[Min06a] A. Miné. The octagon abstract domain. *Higher-Order and Symbolic Computation*, 19(1):31–100, 2006. http://www.di.ens.fr/~mine/publi/article-mine-HOSC06.pdf.

[Min06b] A. Miné. Symbolic methods to enhance the precision of numerical abstract domains. In *VMCAI'06*, pages 348–363, 2006.

[Miy00] Shinya Miyajima. On the improvement of the division of the affine arithmetic. http://www.kashi.info.waseda.ac.jp/Non-linear/thesis-e.html, 2000.

[MK04a] Shinya Miyajima and Masahide Kashiwagi. A dividing method utilizing the best multiplication in affine arithmetic. *IEICE Electronic Express*, 1(7):176–181, 2004.

[MK04b] Shinya Miyajima and Masahide Kashiwagi. A method which finds the maxima and minima of a multivariable function applying affine arithmetic. In *NAA*, pages 424–431, 2004.

[MY59] Ramon E. Moore and C. T. Yang. Interval analysis I. Technical Report LMSD-285875, Lockheed Missiles and Space Division, Sunnyvale, CA, USA, 1959.

[NO79] Greg Nelson and Derek C. Oppen. Simplification by cooperating decision procedures. *ACM Trans. Program. Lang. Syst.*, 1:245–257, October 1979.

[Pro] PPL Project. The Parma Polyhedra Library. http://www.cs.unipr.it/ppl/.

[Rem34] E. Ya. Remez. Sur le calcul effectif des polynomes d'approximation de tchebichef, 1934.

[RMB05] N. Revol, K. Makino, and M. Berz. Taylor models and floating-point arithmetic: proof that arithmetic operations are validated in cosy. *The journal of Logic and Algebraic Programming*, pages 135–154, 2005.

[Roc70] R. Tyrrell Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[SD07] Jean Souyris and David Delmas. Experimental assessment of astre on safety-critical avionics software. In Francesca Saglietti and Norbert Oster, editors, *Computer Safety, Reliability, and Security*, volume 4680 of *Lecture Notes in Computer Science*, pages 479–490. Springer Berlin / Heidelberg, 2007.

[SKH03] Axel Simon, Andy King, and Jacob M. Howe. Two variables per linear inequality as an abstract domain. In *Proceedings of the 12th international conference on Logic based program synthesis and transformation*, LOPSTR'02, pages 71–89, Berlin, Heidelberg, 2003. Springer-Verlag.

[SLMW06] Huahao Shou, Hongwei Lin, Ralph R. Martin, and Guojin Wang. Modified affine arithmetic in tensor form for trivariate polynomial evaluation and algebraic surface plotting. *J. Comput. Appl. Math.*, 195:155–171, October 2006.

[SS98] Michael J. Schulte and James E. Stine. A combined interval and floating-point divider. *Signals, Systems and Computers*, 1:218–222, 1998.

[SS04] David Speyer and Bernd Sturmfels. Tropical mathematics. *Combinatorics*, 2004.

[SSM05] Sriram Sankaranarayanan, Henny B. Sipma, and Zohar Manna. Scalable analysis of linear systems using mathematical programming. In *VMCAI*, pages 25–41, 2005.

[Ste74] Pat H. Sterbenz. *Floating-point Computation*. Prentice-Hall series in automatic computation, 1974.

[Tar55] A. Tarsky. A lattice-theoretical fixpoint theorem and its applications. *Pacific J. Math.*, 5:285–309, 1955.

[tre] Decoder library for vorbis audio format. http://wiki.xiph.org/index.php/Tremor.

[Vig78] J. Vignes. New methods for evaluating the validity of the results of mathematical computations. *Mathematics and Computers in Simulation*, 20(4):227 – 249, 1978.

[Vig93] J. Vignes. A stochastic arithmetic for reliable scientific computation. *Mathematics and Computers in Simulation*, 35(3):233 – 261, 1993.

[wik] Wikipedia, the free encyclopedia. http://www.wikipedia.org.

[ZW90] Shen Zuhe and M. A. Wolfe. On interval enclosures using slope arithmetic. *Appl. Math. Comput.*, 39:89–105, September 1990.