A^2I: Abstract^2 Interpretation

PATRICK COUSOT, New York University, USA
ROBERTO GIACOBAZZI, University of Verona, Italy and IMDEA Software Institute, Spain
FRANCESCO RANZATO, University of Padova, Italy

The fundamental idea of Abstract^2 Interpretation (A^2I), also called meta-abstract interpretation, is to apply abstract interpretation to abstract interpretation-based static program analyses. A^2I is generally meant to use abstract interpretation to analyse properties of program analysers. A^2I can be either offline or online. Offline A^2I is performed either before the program analysis, such as variable packing used by the Astrée program analyser, or after the program analysis, such as in alarm diagnosis. Online A^2I is performed during the program analysis, such as Venet's cofibred domains or Halbwachs et al.'s and Singh et al.'s variable partitioning techniques for fast polyhedra/numerical abstract domains. We formalize offline and online meta-abstract interpretation and illustrate this notion with the design of widenings and the decomposition of relational abstract domains to speed-up program analyses. This shows how novel static analyses can be extracted as meta-abstract interpretations to design efficient and precise program analysis algorithms.

CCS Concepts: • **Software and its engineering → Automated static analysis**; **Formal software verification**; *Semantics*; • **Theory of computation → Program analysis**.

Additional Key Words and Phrases: Abstract interpretation, program analysis, meta-abstract interpretation

ACM Reference Format:
Patrick Cousot, Roberto Giacobazzi, and Francesco Ranzato. 2019. A^2I: Abstract^2 Interpretation. *Proc. ACM Program. Lang.* 3, POPL, Article 42 (January 2019), 31 pages. https://doi.org/10.1145/3290355

1 INTRODUCTION

1.1 Vision

Static program analysis consists of a range of well established and widely used techniques for automatically extracting properties concerning the dynamic behavior of programs. Abstract interpretation generalizes most existing methodologies for static program analysis into a unique sound-by-construction framework which is based on a simple but striking idea: *extracting properties of programs is approximating their semantics* [Cousot and Cousot 1977a]. Although this basic notion plays a fundamental role in most existing tools for sound (or even unsound) analysis of programs, we face new challenges coming from the nature of code which evolves. Software keeps growing in size and complexity and the idea of designing one reliable and comprehensive program analysis for one critical and well structured program is nowadays outmoded, unless very specific application constraints are considered (*e.g.*, in safety critical applications). A modern static program analysis tool is an extremely complex piece of software. It includes sophisticated abstractions, convoluted

Authors' addresses: Patrick Cousot, Courant Institute of Mathematical Sciences, New York University, USA, pcousot@cs.nyu.edu; Roberto Giacobazzi, Dipartimento di Informatica, University of Verona, Italy , IMDEA Software Institute, Spain, roberto.giacobazzi@univr.it; Francesco Ranzato, Dipartimento di Matematica, University of Padova, Italy, ranzato@math.unipd.it.

2475-1421/2019/1-ART42
https://doi.org/10.1145/3290355

optimizations and resource handling strategies in order to be applicable to a wide range of programs, to be maintained and adapted to the evolution of the code to be analysed. Our vision is that all this requires for program analysis the same optimizations and analyses that we require for the software we use program analysis for, that is, we advocate the *analysis of program analysis.*

1.2 Related Work

The idea of analysing program analyses roots back to the works by Halbwachs et al. [2006, 2003], who introduced a dynamic factorization of a polyhedra analysis, and has been made explicit by Giacobazzi et al. [2015] for analysing the precision of static analyses. However, no general framework is known in order to systematically design an analysis of analyses. Abstract^2 interpretation (A^{2}I) is our solution to this problem which puts forward a notion of *meta-analysis*, that is, a parametric domain-agnostic analysis of program analyses, designed through abstract interpretation.

A meta-analysis can be offline (or static), *i.e.*, before the underlying program analysis has been done so that the output of meta-analysis can be exploited by the program analysis. An example of offline meta-analysis is the packing of program variables for the octagon abstraction performed by the Astrée analyser [Bertrane et al. 2015]. Here, variables found likely unrelated by this meta-analysis are put in different packs at each program point. The abstract domain resulting from this offline meta-analysis is the map of program points to packs of variables to octagons. A further example of offline meta-analysis has been introduced by Giacobazzi et al. [2015] who define a static type inference system for abstract interpretation-based program analyses capable of inferring complete abstract interpretations, namely program analyses with no false alarms. An offline meta-analysis can also be done after a program analysis to provide a diagnosis on the output of the analysis. This pattern of meta-analysis has been followed by Cadar and Donaldson [2016] for analysing the absence of false negatives in the results of unsound program analyses and by Lee et al. [2017] for designing a methodology for clustering alarms detected by a program analysis according to their sound dependencies.

The basic idea of online (or dynamic) meta-analysis dates back to the extrapolation by widening and interpolation by narrowing in abstract interpretation: in fact, these operators analyse the evolution of the fixpoint iterations of a program analysis to calculate the next iterate. This was made explicit in cofibred domains by Venet [1996], generalizing [Bourdoncle 1992], and stating that: “Widening operators have originally been used in abstract interpretation in order to cope with infinite domains on which abstract iteration sequences were not necessarily computable. In fact, the notion of widening is much more powerful since it allows the definition of abstract interpretations with very few hypotheses on their structure by *dynamically* constructing the abstract domain.” [Venet 1996, Section 1]. Cofibred domains were the first clear example of an online analysis of the program analyser so as to use the past fixpoint iterates to affect how the next iterates will be computed. A more recent example is the online version of the above-mentioned offline variable packing for octagons. The variable partitioning for the polyhedra abstract domain [Halbwachs et al. 2006, 2003; Singh et al. 2017] has been applied to octagons [Singh et al. 2015] and later to linear numerical abstract domains [Singh et al. 2018]. The relational numerical abstract properties are decomposed into blocks where variables in different blocks are unrelated. The partition of program variables into blocks changes during program analysis and the next variable partition is computed by analysing the underlying program analysis. The “silhouettes” of Li et al. [2017]’s shape analysis can be also viewed as an online meta-analysis to decide when and how to merge case splits and to replace them with summaries in unions and widenings.

1.3 Contributions

We show how the standard methods of abstract interpretation can be naturally lifted to reason about abstract interpreters, therefore providing a systematic method for designing analyses of program analyses by A^{2}I. A simple running example of A^{2}I will be given in Section 2, where we describe a constant propagation-based analysis of interval program analysis.

After introducing some basic notation in Section 3, we start by defining a very general abstract interpreter in Section 4 which will be then the subject of meta-analysis. Several instances of the generic abstract interpreter are given in order to illustrate its wide spectrum of application.

We define the semantics of this generic abstract interpreter of programs in Section 5 and prove its soundness in Section 6. This semantics is defined to be the sequence of fixpoint iterates of the abstract interpreter. Keeping the sequence of iterates (*i.e.*, traces) instead of the set of reachable abstract values, as commonly done for standard program analysis, is essential here, since the meta-abstract interpreter must take into account the full evolution of the iterates of the analyser while performing its analysis, not only the properties of the abstract values incrementally computed by the abstract interpreter. This corresponds to the most general version of a widening operator based on the full past history of computations, as investigated by Cousot [2015]. The prefix/maximal/pointwise trace semantics of the abstract interpreter are given in fixpoint form using a prefix order between traces as calculational order. The equivalence of these semantics is shown by calculational design, a general proof technique advocated by Cousot [1999]. This therefore provides a meta-collecting semantics, which is defined in a systematic way from the standard collecting semantics of a programming language.

Offline meta-analyses are intended to extract properties of the analyser before its run (after its run is a simple extension which takes into account terminated executions of the analyser only). In Section 7, we specify properties of the abstract interpreter as sets of its traces. The strongest of such properties is defined to be the collecting semantics of the meta-abstract interpreter. It is given in fixpoint form for a calculational order (trace prefix ordering) which is different from logical implication (set inclusion): this also happens in strictness analysis [Cousot and Cousot 1993; Mycroft 1982], although here we deal with the additional technical difficulty deriving from the fact that we do not compute in a ω-cpo, since the collecting semantics of the meta-abstract interpreter must observe and keeps track of passing to the limit of infinite iterates of the abstract interpreter. The offline meta-abstract interpreter collects properties of the analyser represented as sets of its traces, as computed by its collecting semantics. This can be specified as an instance of the generic abstract interpreter that we define in Section 5.

Standard abstract interpretation is applied in Section 8 to derive by calculational design an abstract meta-abstract interpreter (thus defined by A^{2}I) from the collecting meta-abstract interpreter. In Section 9, this abstract meta-abstract interpreter is exploited for defining an interval widening from classical ones, including thresholds, therefore providing examples of a systematic (as opposed to a creative) method for designing widening operators (that can be also applied online). In Section 9.4, we mention examples hinting that offline meta-abstract interpretation can be also applied to design abstract domains.

Online meta-analyses are defined and studied in Sections 10-12. Online meta-analysis is intended to optimize an underlying program analyser. We show how the meta-interpreter, which is still an instance of the generic abstract interpreter, can be used to analyse the abstract interpreter by interspersing the meta-interpreter within runs of the program abstract interpreter. As an application of online meta-abstract interpretation, we consider a widening with dynamic thresholds based on slopes. We generalize the techniques for speeding-up numerical program analyses by Halbwachs et al. [2006, 2003] and Singh et al. [2015, 2017, 2018] to the decomposition of any relational domain as maps of blocks of related variables to relational properties of variables in each block. This cannot be implemented as, *e.g.*, a standard reduced product of abstract domains because the variable partition evolves during the program analysis and therefore must be calculated during the analysis by a suitable meta-analysis.

2 HOW A^{2}I WORKS

A crucial aspect of lifting abstract interpretation from programs to abstract interpreters is given by the nature of the properties we are interested in. Invariance and safety properties of the program analyser are less relevant in this context, while properties that hold along all partial executions of the analyser are particularly relevant. As significant examples, these properties include the detection of stable bounds of interval analysis of program variables for the systematic design of widening operators and the independence of program variables to dynamically weaken relational abstract domains. In a sentence, by rephrasing the well-known safety/liveness dualism [Alpern and Schneider 1987]: *The analysis of program analyses looks for properties that will potentially or eventually happen during an abstract interpretation.* We call this process abstract abstract interpretation or meta-abstract interpretation.

We show how meta-abstract interpretation works on the well-known interval analysis [Cousot and Cousot 1977a] of the following simple program decorated with program points ℓ_1 and ℓ_2:

$$\texttt{x=0; while}\ ^{\ell_1}\ \texttt{(true) \{x=x+2;}\ ^{\ell_2}\texttt{\}}$$

An abstract interpreter not using widening operators provides the pointwise $\sqsubseteq$-least solution in the interval domain $\langle Int, \sqsubseteq, \bot, \sqcup\rangle$ of the system of equations $(X_1, X_2) = F(X_1, X_2)$ defined by:

$$\begin{cases} X_1 &= F_1(X_1, X_2) &\triangleq [0,0] \sqcup X_2 \\ X_2 &= F_2(X_1, X_2) &\triangleq X_1 \oplus [2,2] \end{cases} \tag{1}$$

where $(X_1, X_2) \in Int^2$ and $\oplus$ denotes the standard binary interval addition operation. The Jacobi iterates for F in Int^2, extended with their limit which is the least fixpoint of (1), are as follows:

$$\begin{bmatrix}\bot\\\bot\end{bmatrix}, \begin{bmatrix}[0,0]\\\bot\end{bmatrix}, \begin{bmatrix}[0,0]\\[2,2]\end{bmatrix}, \begin{bmatrix}[0,2]\\[2,2]\end{bmatrix}, \ldots, \begin{bmatrix}[0,2n]\\[2,2n]\end{bmatrix}, \begin{bmatrix}[0,2n]\\[2,2(n+1)]\end{bmatrix}, \begin{bmatrix}[0,2(n+1)]\\[2,2(n+1)]\end{bmatrix}, \ldots, \begin{bmatrix}[0,\infty]\\[2,\infty]\end{bmatrix}$$

We have to define the collecting semantics of this abstract interpreter. In general, the collecting semantics is the strongest property of a given semantics S: by viewing properties as sets (of items having this property), the collecting semantics turns out to be the singleton $\{S\}$. Alternatively, the collecting semantics is sometimes understood as a first abstraction of $\{S\}$, which defines a class of properties of interest such as safety or reachability properties. For example, we could define a collecting semantics of the above abstract interpreter as the set of all iterates of the abstract interpreter. However, this invariant would be, in general, too imprecise. For example, this set would lose the information that the iterates start from $\langle\bot, \bot\rangle$ and are increasing. Thus, we consider a more refined collecting semantics of the interval abstract interpreter which collects sequences of iterates instead of sets of iterates. By analogy with program reachability analysis, we replace states which are reachable at some program point by prefix traces reaching that program point. The prefix trace collecting semantics is therefore given as a set of equations:

$$\begin{cases} \overline{X}_1 &= \overline{F}_1(\overline{X}_1, \overline{X}_2) &\triangleq \overline{X}_1 \bullet ([0,0] \sqcup \mathrm{lst}(\overline{X}_2)) \\ \overline{X}_2 &= \overline{F}_2(\overline{X}_1, \overline{X}_2) &\triangleq \overline{X}_2 \bullet (\mathrm{lst}(\overline{X}_1) \oplus [2,2]) \end{cases} \tag{2}$$

where $\langle\overline{X}_1, \overline{X}_2\rangle$ is a pair of denumerable and possibly infinite traces of intervals, $\bullet$ is trace concatenation (where $\tau \bullet \tau' = \tau$ when τ is an infinite trace), and $\mathrm{lst}(X_1 \cdots X_n) = X_n$ is the last element of a nonempty finite sequence $X_1 \cdots X_n$. Let us recall that denumerable possibly infinite traces, ordered by the prefix preorder $\preccurlyeq_{\mathrm{pf}}$, give rise to a cpo. The Jacobi iterates of (2) starting from the pair of interval traces $[\bot, \bot]$ are therefore as follows:

$$\begin{bmatrix}\bot\\ \bot\end{bmatrix}, \begin{bmatrix}\bot \cdot [0,0]\\ \bot \cdot \bot\end{bmatrix}, \begin{bmatrix}\bot \cdot [0,0] \cdot [0,0]\\ \bot \cdot \bot \cdot [2,2]\end{bmatrix}, \begin{bmatrix}\bot \cdot [0,0] \cdot [0,0] \cdot [0,2]\\ \bot \cdot \bot \cdot [2,2] \cdot [2,2]\end{bmatrix}, \begin{bmatrix}\bot \cdot [0,0] \cdot [0,0] \cdot [0,2] \cdot [0,2]\\ \bot \cdot \bot \cdot [2,2] \cdot [2,2] \cdot [2,4]\end{bmatrix}, \ldots$$

$$\begin{bmatrix}\bot \cdot [0,0] \cdot [0,0] \cdot [0,2] \cdot \cdots \cdot [0,2n]\\ \bot \cdot \bot \cdot [2,2] \cdot [2,2] \cdot \cdots \cdot [2,2(n+1)]\end{bmatrix}, \begin{bmatrix}\bot \cdot [0,0] \cdot [0,0] \cdot [0,2] \cdot \cdots \cdot [0,2n] \cdot [0,2(n+1)]\\ \bot \cdot \bot \cdot [2,2] \cdot [2,2] \cdot \cdots \cdot [2,2(n+1)] \cdot [2,2(n+1)]\end{bmatrix}, \ldots$$

The lub of this infinite chain of pairs of finite traces is a pair of infinite traces:

$$\begin{bmatrix}\bot \cdot [0,0] \cdot [0,0] \cdot [0,2] \cdot \cdots \cdot [0,2n] \cdot \cdots\\ \bot \cdot \bot \cdot [2,2] \cdot [2,2] \cdot \cdots \cdot [2,2n] \cdot \cdots\end{bmatrix}_{n \geq 1}$$

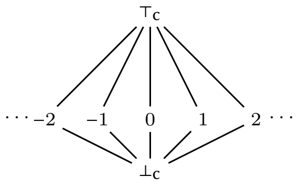
A^{2}I, *i.e.* the meta-abstract interpreter, has to effectively compute an abstraction of this prefix trace collecting semantics. Given that the prefix trace collecting semantics is expressed in fixpoint form (2), the fixpoint specification of this meta-abstract interpreter can be designed by fixpoint approximation as pioneered in [Cousot and Cousot 1979].

The traces $\overline{X}_1$ and $\overline{X}_2$ of all the iterates of (2), are of the form $\bot \cdot [\ell_1, h_1] \cdot [\ell_2, h_2] \cdot \cdots \cdot [\ell_n, h_n]$, with $n \geqslant 0$, which reduces to $\bot$ for $n = 0$. Consider the abstraction α_{c} (where c means constancy):

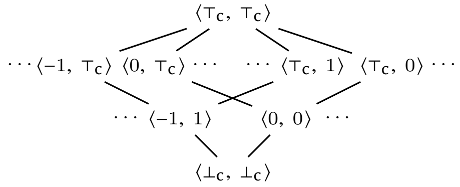
$$\begin{array}{rcl}\alpha^2_{\mathrm{c}}(\langle \overline{X}_1,\ \overline{X}_2\rangle) & \triangleq & \langle \alpha_{\mathrm{c}}(\overline{X}_1),\ \alpha_{\mathrm{c}}(\overline{X}_2)\rangle\\ \alpha_{\mathrm{c}}(\bot \cdot [\ell_1, h_1] \cdot [\ell_2, h_2] \cdot \cdots \cdot [\ell_n, h_n]) & \triangleq & \langle \bigsqcup_{\mathrm{c}\, i=1}^{n} \ell_i,\ \bigsqcup_{\mathrm{c}\, i=1}^{n} h_i\rangle\end{array}$$

Here, $\sqcup_{\mathrm{c}}$ denotes the lub in the well-known Kildall's constant propagation lattice [Kildall 1973] $\langle \mathcal{D}_{\mathrm{c}},\ \sqsubseteq_{\mathrm{c}},\ \bot_{\mathrm{c}},\ \top_{\mathrm{c}},\ \sqcup_{\mathrm{c}},\ \sqcap_{\mathrm{c}}\rangle$, as depicted below and whose abstraction $\langle \wp(\mathbb{Z}),\ \subseteq\rangle \xrightleftharpoons[\dot{\alpha}_{\mathrm{c}}]{\dot{\gamma}_{\mathrm{c}}} \langle \mathcal{D}_{\mathrm{c}},\ \sqsubseteq_{\mathrm{c}}\rangle$ is defined by: $\dot{\alpha}_{\mathrm{c}}(\emptyset) \triangleq \bot_{\mathrm{c}}$, $\forall z \in \mathbb{Z}\ .\ \dot{\alpha}_{\mathrm{c}}(\{z\}) \triangleq z$, and $\dot{\alpha}_{\mathrm{c}}(X) = \top_{\mathrm{c}}$ otherwise. The meta-abstract interpreter performs its computations componentwise in the product lattice $\mathcal{D}^2_{\mathrm{c}}$:

$\mathcal{D}_{\mathrm{c}} \triangleq$



$\mathcal{D}^2_{\mathrm{c}} \triangleq$



$\langle \top_{\mathrm{c}},\ \top_{\mathrm{c}}\rangle$
$\cdots\langle -1,\ \top_{\mathrm{c}}\rangle$
$\langle 0,\ \top_{\mathrm{c}}\rangle \cdots$
$\cdots\ \langle \top_{\mathrm{c}},\ 1\rangle$
$\langle \top_{\mathrm{c}},\ 0\rangle \cdots$
$\cdots\ \langle -1,\ 1\rangle$
$\langle 0,\ 0\rangle\ \cdots$
$\langle \bot_{\mathrm{c}},\ \bot_{\mathrm{c}}\rangle$

By calculational design which exploits fixpoint approximations of (2), we get the equations:

$$\begin{cases}\langle l_1,\ h_1\rangle = F^{\mathrm{c}}_1(\langle l_1,\ h_1\rangle, \langle l_2,\ h_2\rangle) \triangleq \langle l_1 \sqcup_{\mathrm{c}} 0 \sqcup_{\mathrm{c}} \min(0, l_2),\ h_1 \sqcup_{\mathrm{c}} 0 \sqcup_{\mathrm{c}} \max(0, h_2)\rangle\\ \langle l_2,\ h_2\rangle = F^{\mathrm{c}}_2(\langle l_1,\ h_1\rangle, \langle l_2,\ h_2\rangle) \triangleq \langle l_2 \sqcup_{\mathrm{c}} (l_1 \oplus^c 2),\ h_2 \sqcup_{\mathrm{c}} (h_1 \oplus^c 2)\rangle\end{cases} \tag{3}$$

where $\langle l_i,\ h_i\rangle \in \mathcal{D}^2_{\mathrm{c}}$ and $\bot_{\mathrm{c}}$ is absorbent for all the operators on $\mathcal{D}_{\mathrm{c}}$. Hence, the Jacobi iterates for the equations (3) are as follows:

$$\begin{bmatrix}\langle \bot_{\mathrm{c}},\ \bot_{\mathrm{c}}\rangle\\ \langle \bot_{\mathrm{c}},\ \bot_{\mathrm{c}}\rangle\end{bmatrix}, \begin{bmatrix}\langle 0,\ 0\rangle\\ \langle \bot_{\mathrm{c}},\ \bot_{\mathrm{c}}\rangle\end{bmatrix}, \begin{bmatrix}\langle 0,\ 0\rangle\\ \langle 2,\ 2\rangle\end{bmatrix}, \begin{bmatrix}\langle 0,\ \top_{\mathrm{c}}\rangle\\ \langle 2,\ 2\rangle\end{bmatrix}, \begin{bmatrix}\langle 0,\ \top_{\mathrm{c}}\rangle\\ \langle 2,\ \top_{\mathrm{c}}\rangle\end{bmatrix}.$$

This pointwise $\sqsubseteq_{\mathrm{c}}$-least fixpoint of (3) represents precisely the fact that only the lower interval bound is definitely stable at 0 for the abstract interpreter while no information on the upper interval bound can be obtained, meaning that the upper bound is possibly unstable. This very simple meta-analysis suggests a widening to $\langle [0, +\infty],\ [2, +\infty]\rangle$ for the underlying interval analysis.

3 BASIC NOTATION

$\mathbb{B} \triangleq \{\mathrm{tt}, \mathrm{ff}\}$ is the set of Booleans, $\mathbb{N}$ is the set of natural numbers, $\mathbb{N}_+$ is the set of positive naturals, $\mathbb{Z}$ is the set of integers. $\mathbb{O}$ is the class of all ordinals, where ω is the first infinite ordinal and ω+1 is its successor. The conditional choice is $\llbracket\, \mathrm{tt} \mathrel{?} a \mathrel{⸫} b \,\rrbracket = a$ and $\llbracket\, \mathrm{ff} \mathrel{?} a \mathrel{⸫} b \,\rrbracket = b$. $\wp(S)$ is the powerset of a set S while $\wp_{\mathrm{f}}(S)$ is the set of its finite subsets. Given a set D and $n \in \mathbb{N}_+$, both D^n and $\prod_1^n D$ denote the n-ary Cartesian product. The componentwise extension of an operation/relation f on D to a product D^n is denoted by $\dot{f}$. For example, $\dot{\sqsubseteq}$ denotes the componentwise extension of a partial order relation $\sqsubseteq$.

A poset $\langle D, \sqsubseteq\rangle$ is a set D endowed with a partial order $\sqsubseteq$. A poset is Noetherian when it contains no infinite chain (*i.e.*, it satisfies the ascending chain condition). $\langle D, \sqsubseteq, \bot, \sqcup\rangle$ is a complete partial order (cpo) if $\langle D, \sqsubseteq\rangle$ is a poset such that all its chains have a least upper bound (lub) $\sqcup$, including the empty chain whose lub is the infimum $\bot = \sqcup\emptyset$. For a ω-cpo, only denumerable chains (or ω-chains) are considered: hence, if $\langle x^i,\ i \in \mathbb{N}\rangle$ is a $\sqsubseteq$-increasing sequence in D then its lub $\sqcup_{i\in\mathbb{N}} x^i \in D$ exists. A complete lattice $\langle D, \sqsubseteq, \bot, \top, \sqcup, \sqcap\rangle$ is a poset $\langle D, \sqsubseteq\rangle$ such that any subset $X \subseteq D$ has a lub $\sqcup X$, and therefore a greatest lower bound (glb) $\sqcap X$, so that, in turn, D has the infimum $\bot = \sqcup\emptyset = \sqcap D$ and the supremum $\top = \sqcup D = \sqcap\emptyset$.

A (total) function is denoted by $f \in P \longrightarrow Q$ and for all subsets $X \subseteq P$, $f(X) \triangleq \{f(x) \in Q \mid x \in X\}$ denotes the image of X by f. If $\langle P, \sqsubseteq\rangle$ and $\langle Q, \preccurlyeq\rangle$ are posets then $f \in P \longrightarrow Q$ is monotone (or increasing), denoted by $f \in P \xrightarrow{\nearrow} Q$, when $\forall x, y \in P \,.\, x \sqsubseteq y \implies f(x) \preccurlyeq f(y)$. $f \in P \longrightarrow P$ is called extensive (reductive) when $\forall x \in P \,.\, x \sqsubseteq f(x)$ ($\forall x \in P \,.\, f(x) \sqsubseteq x$). f is idempotent when $f = f \circ f$, where $\circ$ is function composition. A upper (resp. ω-upper) continuous function $f \in P \xrightarrow{uc} P$ preserves existing lubs of chains (resp. ω-chains).

A Galois connection (GC) $\langle P, \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle Q, \preccurlyeq\rangle$ is a pair of functions $\alpha \in P \longrightarrow Q$ and $\gamma \in Q \longrightarrow P$ between posets $\langle P, \sqsubseteq\rangle$ and $\langle Q, \preccurlyeq\rangle$ such that $\forall x \in P \,.\, \forall y \in Q \,.\, \alpha(x) \preccurlyeq y \iff x \sqsubseteq \gamma(y)$. A GC is called Galois insertion (or retraction), denoted by $\mathrel{\substack{\longleftarrow\\ \twoheadrightarrow}}$, when α is onto (or, equivalently, γ is injective), while a GC is called a Galois isomorphism, denoted by $\mathrel{\substack{\twoheadleftarrow\\ \twoheadrightarrow}}$, when α is bijective (or, equivalently, γ is bijective). An upper closure operator ρ on a poset $\langle P, \sqsubseteq\rangle$ is a monotone, extensive and idempotent function $\rho \in P \longrightarrow P$. It turns out that ρ is an upper closure operator on $\langle P, \sqsubseteq\rangle$ if and only if $\langle P, \sqsubseteq\rangle \underset{\rho}{\overset{\mathbb{1}_P}{\mathrel{\substack{\longleftarrow\\ \twoheadrightarrow}}}} \langle \rho(P), \sqsubseteq\rangle$, where $\mathbb{1}_P$ is the identity map on P.

Given a poset $\langle D, \sqsubseteq\rangle$, $\mathsf{lfp}^{\sqsubseteq} f$ denotes the $\sqsubseteq$-least fixpoint of $f \in D \longrightarrow D$, if this exists. Given $a \in D$, $\mathsf{lfp}_a^{\sqsubseteq} f$ denotes the least fixpoint of f greater than or equal to a, if it exists. The transfinite iterates $f^\delta(a)$, for all $\delta \in \mathbb{O}$, of $f \in D \longrightarrow D$ starting from $a \in D$ ($\bot \in D$ by default) are defined by transfinite induction: $f^0(a) \triangleq a$; $f^{\delta+1}(a) \triangleq f(f^\delta(a))$ for successor ordinals $\delta + 1$; $f^\lambda(a) \triangleq \sqcup_{\delta<\lambda} f^\delta(a)$ for limit ordinals λ, if this lub in D exists. The transfinite iterates of f from a are increasing when for all $\delta, \delta' \in \mathbb{O}$, if $\delta \leqslant \delta'$ and $f^\delta(a), f^{\delta'}(a)$ are defined then $f^\delta(a) \sqsubseteq f^{\delta'}(a)$. Consider a system of fixpoint equations $\{x_i = f_i(x_1, \ldots, x_n)\}_{i=1}^n$, for $f_i \in D^n \longrightarrow D$. Given an initial vector $(x_1^0, \ldots, x_n^0) \in D^n$, the corresponding Jacobi and Gauss-Seidel iterates are inductively defined as follows: for all $k \geqslant 0$, $i = 1, \ldots, n$,

$$\begin{array}{rcll} x_i^{k+1} & \triangleq & f_i(x_1^k, \ldots, x_n^k) & \text{Jacobi iterates} \\ x_i^{k+1} & \triangleq & f_i(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x_i^k, x_{i+1}^k, \ldots, x_n^k) & \text{Gauss-Seidel iterates} \end{array}$$

4 STATIC ANALYSIS BY A GENERIC ABSTRACT INTERPRETER

4.1 Iterates of a Generic Abstract Interpreter

Generalizing from fixpoint equation solving to cope, *e.g.*, with extrapolation and interpolation operators, a static analysis of a program P [Cousot and Cousot 1977a, 1979] consists in computing a sequence of iterates $\langle X^k,\ k \in [0, \ell[\rangle$ of an abstract interpreter $\mathbf{A}[\![\mathsf{P}]\!]$, which is either finite for some $\ell \in \mathbb{N}_+$, in case the abstract interpreter does terminate, or infinite with $\ell = \omega$, in case the abstract interpreter does not terminate. Hence, this contemplates possibly nonterminating program analyses, *e.g.*, analyses not using widening operators in non-Noetherian abstract domains.

In the most general setting, an abstract interpreter $\mathbf{A}[\![\mathsf{P}]\!]$ of a program P is based on an infinite sequence of semantic entities $\langle D[\![\mathsf{P}]\!]^k, C[\![\mathsf{P}]\!]^k, F[\![\mathsf{P}]\!]^{k+1}, k \in \mathbb{N}\rangle$, where, for all $k \in \mathbb{N}$: $D[\![\mathsf{P}]\!]^k$ is an abstract domain, $C[\![\mathsf{P}]\!]^k \in D[\![\mathsf{P}]\!]^k \longrightarrow \mathbb{B}$ is a convergence/termination criterion and finally $F[\![\mathsf{P}]\!]^{k+1} \in D[\![\mathsf{P}]\!]^k \longrightarrow D[\![\mathsf{P}]\!]^{k+1}$ is an abstract value transformer. For the sake of conciseness, in the following we will often omit the qualifier $[\![\mathsf{P}]\!]$ in these notations.

The possibly infinite sequence of iterates $\langle X^k,\ k \in [0, \ell[\rangle$ of $\mathbf{A}[\![\mathsf{P}]\!]$, where X^0 is some initial abstract value ranging in D^0, is computed as follows:

$$\begin{array}{ll}
\text{initial iterate } k = 0: & X^0 \in D^0 \\
\text{next iterate } k+1 \text{ if } \neg C^k(X^k): & X^{k+1} = F^{k+1}(X^k) \in D^{k+1} \\
\text{termination if } \exists k.\, C^k(X^k): & \ell = k+1 \\
\text{nontermination if } \forall k.\, \neg C^k(X^k): & \ell = \omega
\end{array} \qquad (4)$$

Hence, ℓ is set to ω for a nonterminating analysis such that $\bigwedge_{k\in\mathbb{N}} \neg C^k(X^k)$ holds, while for a terminating analysis we have that $\ell \in \mathbb{N}_+$ and $\big(\bigwedge_{k=0}^{\ell-2} \neg C^k(X^k)\big) \wedge C^{\ell-1}(X^{\ell-1})$ holds.

We don't distinguish between terminating and nonterminating executions. Hence, in Section 5, we will give a semantics of the abstract interpreter where for any terminating execution of $\mathbf{A}[\![\mathsf{P}]\!]$, its finite sequence of iterates $\langle X^k,\ k \in [0, \ell[\rangle$, with $\ell \in \mathbb{N}_+$, will be extended to an infinite sequence by repeating the final iterate $X^{\ell-1}$ infinitely often. This defines an infinite sequence of iterates which is ultimately stationary after $\ell - 1$, *i.e.*, $\forall k \geqslant \ell - 1, X^k = X^{\ell-1}$ and $C^{\ell-1}(X^{\ell-1})$ holds. We will also assume that for a nonterminating execution of $\mathbf{A}[\![\mathsf{P}]\!]$, its infinite sequence of iterates $\langle X^k,\ k \in \mathbb{N}\rangle$ has a mathematical limit which is denoted by X^ω and, of course, will be never reached by a real static analysis. This limit X^ω belongs to some abstract domain D^ω and is provided by some function F^ω defined on infinite sequences of iterates, namely, $X^\omega = F^\omega(\langle X^k,\ k \in \mathbb{N}\rangle) \in D^\omega$.

It is worth remarking that, in this framework, the abstract domains $\langle D^k,\ k \in \mathbb{N} \cup \{\omega\}\rangle$, the convergence conditions $\langle C^k,\ k \in \mathbb{N}\rangle$ and the transformers $\langle F^k,\ k \in \mathbb{N}_+ \cup \{\omega\}\rangle$ are determined as a function of the program P, as opposed to using the same semantic entities (in particular a unique semantic domain) for all programs. As a simple example of this phenomenon, consider standard constant propagation [Kildall 1973], which maps program variables to the constancy lattice $\mathscr{D}_\mathrm{c}$ already recalled in Section 2. A map from all the variables to $\mathscr{D}_\mathrm{c}$ would not satisfy the ascending chain condition since there are infinitely many variables in the language, as opposed to finitely many variables occurring in a program[1]. Moreover, in our general setting, D^k, C^k and F^k may change at each fixpoint iteration of the abstract interpreter. Besides, we require that limits of infinite computations, provided by F^ω, are defined for infinite sequences of fixpoint iterates only, that is, not necessarily for all infinite sequences of abstract values, as it happens, *e.g.*, in a cpo which has the lub of any increasing chain of abstract values.

The iterates are generated by the following generic abstract interpreter $\mathbf{A}$:

```
A[[P]]($X^0$)  ≜  $X$ := $X^0$; k := 0;
                 while ($\neg C^k(X)$)                              (5)
                   { $X$ := $F^{k+1}(X)$; k := k + 1; }
```

[1] Let us remark that recursion may introduce infinitely many instances of the parameter/local variables and a widening may be needed [Cousot and Cousot 1977b].

Hence, for this abstract interpreter $\mathbf{A}[\![\mathsf{P}]\!]$, and as later made clear in (20), the abstract domains $\langle D^k,\ k \in \mathbb{N}\rangle$, convergence criteria $\langle C^k,\ k \in \mathbb{N}\rangle$ and transformers $\langle F^{k+1},\ k \in \mathbb{N}\rangle$ can be dynamically determined by the meta-abstract interpreter at each iteration step $k \in \mathbb{N}$ of the program analysis. The limit domain D^ω and the limit transformer F^ω are of mere mathematical nature, namely, not used by the generic abstract interpreter $\mathbf{A}[\![\mathsf{P}]\!]$ (interpreter) but exploited for modeling its possibly nonterminating behaviour through a suitable collecting semantics. For example, a widening must provide a sound approximation of this limit of an infinite sequence of iterates.

4.1.1 Fixpoint Abstract Interpreter. A standard example of abstract interpreter is provided by a single abstract domain $\langle D,\ \bot,\ \sqsubseteq,\ \sqcup\rangle$ which is a ω-cpo and by a ω-upper continuous transformer $F \in D \xrightarrow{uc} D$, both implicitly defined as a function of the program under analysis. The analysis computes the least fixpoint $\mathsf{lfp}^{\sqsubseteq} F$ of F which is the lub $\sqcup$ of the chain of iterates $X^0 \triangleq \bot$ and $X^{k+1} \triangleq F(X^k)$, with convergence criterion $C(X) \triangleq (F(X) \sqsubseteq X)$. Since the iterates form an increasing chain, they finitely converge when D is Noetherian, otherwise they may diverge to the limit $X^\omega = \bigsqcup_{k\in\mathbb{N}} X^k \triangleq \mathsf{lfp}^{\sqsubseteq} F$, which is well-defined in a ω-cpo. In this context, we can also use the abstract interpreter (5) to specify a concrete program semantics, for instance a denotational semantics (*e.g.*, [Jones and Nielson 1995; Scott 1972]).

4.1.2 Chaotic Abstract Interpreter. Continuing Section 4.1.1, the fixpoint equation $X = F(X)$ is often given by a system of equations $\{X_i = F_i(X_1, \ldots, X_n)\}_{i=1}^n$ on a product abstract domain $\langle D^n,\ \dot{\sqsubseteq},\ \dot{\bot},\ \dot{\sqcup}\rangle$, for some $n > 0$. In this case, one can use chaotic iterations [Cousot 1977; Cousot and Cousot 1977a] as defined by some set $\Delta^k \subseteq \{1, \ldots, n\}$ of components that evolve at each iteration step $k \in \mathbb{N}$. Hence, a different function F^{k+1} is applied at each iteration step k, namely, $(F^{k+1}(X))_i \triangleq$ if $i \in \Delta^{k+1}$ then $F_i(X_1^k, \ldots, X_n^k)$ else X_i^k. This includes as instances Gauss-Seidel iterations and the classical worklist algorithm of data-flow analysis [Kildall 1973].

4.1.3 Widening Abstract Interpreter. Continuing Section 4.1.2, the abstract domain D could be non-Noetherian. In this case, a terminating widening operator is applied on loop and recursion entries only. Thus, again, a different function F^{k+1} is applied at each iteration step k and this function, in general, is not monotone (recall that a terminating widening cannot be monotone [Cousot 2015]). Where, when and how the terminating widening should be applied can be determined by a meta-abstract interpreter.

4.2 Interval Abstract Interpreter

Let us illustrate how a meta-abstract interpreter can be fruitfully exploited for designing a widening for an abstract interpreter which performs program analysis with the interval abstract domain.

4.2.1 Interval Domain. Let $\mathbb{I}$ be either the set $\mathbb{Z}$ of mathematical integers, $\mathbb{M} = [\mathrm{min\_int}, \mathrm{max\_int}]$ of machine integers, $\mathbb{Q}$ of rationals, $\mathbb{F}$ of floating point numbers, or $\mathbb{R}$ of reals. For machine integers, $\mathrm{min\_int}$ and $\mathrm{max\_int}$ are, resp., the smallest and largest representable integers, and overflow is considered to be an error. For modular integer arithmetics we refer to [Gange et al. 2014]. Let the abstract domain of numerical intervals [Cousot and Cousot 1976, 1977a] be the complete lattice $\langle \mathit{Int},\ \sqsubseteq,\ \bot,\ \top,\ \sqcup,\ \sqcap\rangle$ where

$$\mathit{Int} \triangleq \{\bot\} \cup \{[\underline{x}, \overline{x}] \mid \underline{x} \in \mathbb{I} \wedge \underline{x} \leqslant \overline{x} \wedge \overline{x} \in \mathbb{I}\} \cup \{[-\infty, \overline{x}] \mid \overline{x} \in \mathbb{I}\} \cup \{[\underline{x}, \infty] \mid \underline{x} \in \mathbb{I}\} \cup \{[-\infty, \infty]\}.$$

The empty interval $\bot = \emptyset$ can be encoded by any $[\underline{x}, \overline{x}]$ with $\overline{x} < \underline{x}$ (*e.g.* normalized to $[\infty, -\infty]$) and $\top = [-\infty, \infty]$. The intervals $[-\infty, -\infty] \notin \mathit{Int}$ and $[\infty, \infty] \notin \mathit{Int}$ are excluded. The partial order $\sqsubseteq$ on Int is interval inclusion: $\bot \sqsubseteq \bot \sqsubset [\underline{x}, \overline{x}] \sqsubseteq [\underline{y}, \overline{y}]$ if and only if $\underline{y} \leqslant \underline{x} \leqslant \overline{x} \leqslant \overline{y}$.

We consider the standard Galois insertion $\langle \wp(\mathbb{I}), \subseteq\rangle \underset{\alpha_i}{\overset{\gamma_i}{\leftrightarrows}} \langle \mathit{Int}, \sqsubseteq\rangle$ where α_i abstracts a possibly unbounded set of numerical values by their minimum and maximal values. We define $\min \mathbb{I} \triangleq -\infty$, $\max \mathbb{I} \triangleq \infty$, $\min \emptyset \triangleq \infty$, $\max \emptyset \triangleq -\infty$, where $-\infty = \mathrm{min\_int}$ and $\infty = \mathrm{max\_int}$ for machine integers. Thus, we have that $\alpha_i(S) \triangleq [\min S, \max S]$ and $\gamma_i([\underline{x}, \overline{x}]) \triangleq \{z \in \mathbb{I} \mid \underline{x} \leqslant z \leqslant \overline{x}\}$.

4.2.2 Example. We illustrate the meta-analysis of the abstract interpreter $\mathbf{A}[\![P]\!]$ defined in (5) for the following program P on the interval domain *Int*, which is the same program already used in Section 2 with additional program points:

$$^{\ell_1}\ \texttt{x=0; while}\ ^{\ell_2}\ \texttt{(true) \{x=x+2;}\ ^{\ell_3}\texttt{\}}^{\ell_4} \tag{6}$$

We consider $D^k \triangleq \mathit{Int}^4$, for all $k \in \mathbb{N} \cup \{\omega\}$, and $X^0 \triangleq \langle \top,\ \bot,\ \bot,\ \bot\rangle \in \mathit{Int}^4$. The abstract transformers F^{k+1}, for $k \in \mathbb{N}$, are the chaotic iterations of the following function:

$$F(\langle I_1,\ I_2,\ I_3,\ I_4\rangle) \triangleq \langle I_1,\ [0,0] \sqcup I_3,\ (I_2 \sqcap \top) \oplus [2,2],\ I_2 \sqcap \bot\rangle.$$

The abstract convergence criterion C^k, for all $k \in \mathbb{N}$, is $C^k(\langle I_1,\ I_2,\ I_3,\ I_4\rangle) \triangleq ([0,0] \sqcup I_3 \sqsubseteq I_2)$, and the limit of an infinite sequence of iterates is given by the lub, that is, $F^\omega(\langle X^k,\ k \in \mathbb{N}\rangle) \triangleq \bigsqcup_{k\in\mathbb{N}} X^k$.

The iterates of the interval abstract interpreter $\mathbf{A}[\![P]\!]$ are assumed to be the Gauss-Seidel chaotic iterations for the abstract transformer F. Thus, it turns out that:

$$X^0 = \langle \top,\ \bot,\ \bot,\ \bot\rangle,\ X^1 = \langle \top,\ [0,0],\ [2,2],\ \bot\rangle,\ X^2 = \langle \top,\ [0,2],\ [2,4],\ \bot\rangle,\ \ldots$$
$$X^{n+1} = \langle \top,\ [0, 2*n],\ [2, 2*(n+1)],\ \bot\rangle,\ \ldots,\ X^\omega = \langle \top,\ [0,\infty],\ [2,\infty],\ \bot\rangle.$$

5 TRACE SEMANTICS OF THE ABSTRACT INTERPRETER

We define the prefix trace semantics of the abstract interpreter and then derive by calculational design its maximal trace and projection semantics.

5.1 Sequences

Let $\langle D^k,\ k \in \mathbb{N} \cup \{\omega\}\rangle$, $\langle C^k,\ k \in \mathbb{N}\rangle$ and $\langle F^k,\ k \in \mathbb{N}_+ \cup \{\omega\}\rangle$ specify an abstract interpreter $\mathbf{A}[\![\mathrm{P}]\!]$ as defined in Section 4. The disjoint union of all the abstract domains is denoted by:

$$\mathbf{D} \quad \triangleq \quad \biguplus_{k\in\mathbb{N}\cup\{\omega\}} D^k$$

We write $X^k \in \mathbf{D}$ to mean that $X^k \in D^k$. Given a nonempty set of indices $I \subseteq \mathbb{N} \cup \{\omega\}$, $X \in I \longrightarrow \mathbf{D}$ is a I-indexed sequence of elements in $\mathbf{D}$, where we typically use $X^i = X(i)$ to denote the element of X with index $i \in I$. We define the following sets of sequences, where $\ell \in \mathbb{N}$:

$$\begin{array}{rcll} \mathbf{D}^\ell & \triangleq & [0,\ell] \longrightarrow \mathbf{D} & \text{nonempty finite sequences of length } \ell + 1 \\ \mathbf{D}^\omega & \triangleq & \mathbb{N} \longrightarrow \mathbf{D} & \omega\text{-infinite sequences} \\ \mathbf{D}^{\omega+1} & \triangleq & \mathbb{N} \cup \{\omega\} \longrightarrow \mathbf{D} & \omega\text{+1-transfinite sequences} \end{array}$$

Concatenation of finite sequences is denoted by $\bullet$ and it is pointwise extended to sets of sequences. Sequences are also denoted with the following explicit notations:

$$\langle X^i,\ i \in [0,\ell]\rangle = X^0 \bullet \cdots \bullet X^i \bullet \cdots \bullet X^\ell \in \mathbf{D}^\ell$$
$$\langle X^i,\ i \in \mathbb{N}\rangle = X^0 \bullet \cdots \bullet X^i \bullet X^{i+1} \bullet \cdots \in \mathbf{D}^\omega$$
$$\langle X^i,\ i \in \mathbb{N} \cup \{\omega\}\rangle = X^0 \bullet \cdots \bullet X^i \bullet \cdots \bullet X^\omega \in \mathbf{D}^{\omega+1}$$

Sequence concatenation $\bullet$ is extended to concatenate a ω-infinite sequence in $\mathbf{D}^\omega$ with an element in $\mathbf{D}$ thus giving a ω+1-transfinite sequence as follows:

$$(X^0 \bullet \cdots \bullet X^i \bullet \cdots) \bullet X^\omega \triangleq X^0 \bullet \cdots \bullet X^i \bullet \cdots \bullet X^\omega \in \mathbf{D}^{\omega+1}$$

We also define the following sets:

$$\begin{array}{rcll}
\mathbf{D}^{+} & \triangleq & \bigcup_{\ell\in\mathbb{N}}\mathbf{D}^{\ell} & \text{nonempty finite sequences}\\
\mathbf{D}^{+\omega} & \triangleq & \mathbf{D}^{+}\cup\mathbf{D}^{\omega} & \text{nonempty finite or infinite sequences}\\
\mathbf{D}^{+\omega+1} & \triangleq & \mathbf{D}^{+\omega}\cup\mathbf{D}^{\omega+1} & \text{nonempty finite, infinite or transfinite sequences}\\
\mathbf{D}^{*\omega+1} & \triangleq & \mathbf{D}^{+\omega+1}\cup\{\ni\} & \text{empty, finite, infinite or transfinite sequences}
\end{array}$$

For any exponent κ, if $X^0 \in \mathbf{D}$ then $\mathbf{D}^{\kappa}(X^0) \triangleq \{\sigma \in \mathbf{D}^{\kappa} \mid \sigma(0) = X^0\}$. For a sequence $X \in \mathbf{D}^{*\omega+1}$, let $|X| \in \mathbb{N}\cup\{\omega, \omega{+}1\}$ denotes its length, which is ω when X is infinite and $\omega{+}1$ when X is transfinite. For any sequence $X \in \mathbf{D}^{*\omega+1}$ and $k \in \mathbb{N}$, the notation $X^0 \bullet X^1 \bullet \cdots \bullet X^k \in \mathbf{D}^{+}$ denotes a nonempty finite prefix of X, meaning that this prefix is well-defined, *i.e.*, $|X| \geq k+1$.

5.1.1 Well-Behaved Sequences. We define a Boolean predicate on nonempty sequences in $\mathbf{D}^{+\omega}$ which models the notion of being a well-behaved sequence for the termination conditions $\langle C^k, k \in \mathbb{N}\rangle$ of an interpreter $\mathbf{A}[\![\mathsf{P}]\!]$. For any sequence $X \in \mathbf{D}^{+\omega}$, we define:

$$\begin{array}{lrcl}
\text{iterating sequence:} & \mathsf{it}(X) & \triangleq & \forall i \in [0, |X|[\,.\ \neg C^i(X^i) \qquad\qquad (7)\\
\text{stationary sequence:} & \mathsf{st}(X) & \triangleq & \exists \ell \in [0, |X|[\,.\ \big(C^{\ell}(X^{\ell})\ \wedge\\
 & & & \forall i \in [0, \ell[\,.\ \neg C^i(X^i)\ \wedge\ \forall j \in [\ell, |X|[\,.\ X^j = X^{\ell}\big)\\
\text{well-behaved sequence:} & \mathsf{wb}(X) & \triangleq & \mathsf{it}(X)\ \vee\ \mathsf{st}(X)
\end{array}$$

Thus, $\mathsf{wb}(X)$ holds when either a termination condition C^i is never met in X, *i.e.*, $\mathsf{it}(X)$ holds, or if a termination condition is first met at some state X^{ℓ} of X then, from that state X^{ℓ} on, the sequence X is (finitely or infinitely) stationary at X^{ℓ}, *i.e.*, $\mathsf{st}(X)$ holds. We will require that well-behaved infinite sequences have a limit, *i.e.*, $F^{\omega} \in \{X \in \mathbf{D}^{\omega} \mid \mathsf{wb}(X)\} \longrightarrow D^{\omega}$. The notion of being well-behaved is extended to ω+1-transfinite sequences $X \bullet X^{\omega} \in \mathbf{D}^{\omega+1}$ simply by requiring that $\mathsf{wb}(X) \wedge X^{\omega} = F^{\omega}(X)$ holds.

5.2 Prefix Trace Semantics of the Abstract Interpreter

5.2.1 Prefixes of Sequences. Given a set $S \in \wp(\mathbf{D}^{+\omega+1})$ of nonempty (finite, infinite or transfinite) sequences, its nonempty prefix closure $\mathsf{pref}(S)$ is defined as follows:

$$\begin{array}{rcl}
\mathsf{pref} & \in & \wp(\mathbf{D}^{+\omega+1}) \longrightarrow \wp(\mathbf{D}^{+\omega+1})\\
\mathsf{pref}(S) & \triangleq & \{\sigma \in \mathbf{D}^{+\omega+1} \mid \exists \sigma' \in \mathbf{D}^{*\omega+1}\,.\ \sigma \bullet \sigma' \in S\}
\end{array}$$

Observe that in this definition $\sigma' \in \mathbf{D}^{*\omega+1}$ may be the empty sequence, and therefore $S \subseteq \mathsf{pref}(S)$ always holds, or a singleton sequence, and therefore $\mathsf{pref}(S)$ may include ω-infinite prefixes of ω+1-transfinite sequences. For example, we have that

$$\begin{array}{l}
\mathsf{pref}(\{X^0 \bullet X^1 \bullet X^2 \bullet \cdots \bullet X^k \bullet \cdots \bullet X^{\omega}\}) = \{X^0 \bullet X^1 \bullet X^2 \bullet \cdots \bullet X^k \mid k \geqslant 0\}\\
\qquad\qquad\qquad\qquad \cup \{X^0 \bullet X^1 \bullet X^2 \bullet \cdots \bullet X^k \bullet \cdots,\ X^0 \bullet X^1 \bullet X^2 \bullet \cdots \bullet X^k \bullet \cdots \bullet X^{\omega}\}.
\end{array}$$

It turns out that pref is an upper closure operator on $\langle \wp(\mathbf{D}^{+\omega+1}), \subseteq\rangle$.

5.2.2 Prefix Trace Semantics. If we choose the semantics of $\mathbf{A}[\![\mathsf{P}]\!]$ defined in (5) to be either the final result (if any) or the full computation towards this result, as recorded by the sequence of iterates, then information is lost on how this result was obtained. Thus, our semantics for $\mathbf{A}[\![\mathsf{P}]\!]$ will be given by the set of all the partial sequences of iterates computed by $\mathbf{A}[\![\mathsf{P}]\!]$. For example, for a chaotic abstract interpreter as described in Section 4.1.2, this semantics will be the set of all the partial sequences of chaotic iterations for the specific iteration strategy chosen by the abstract interpreter. This semantics $\mathcal{S}_{\mathsf{pf}}[\![\mathbf{A}]\!]\,(X^0)$, for an initial abstract value $X^0 \in D^0$, is as follows:

$$\begin{array}{lll}
\mathcal{S}_{\mathsf{pf}}[\![\mathbf{A}]\!]\,(X^0) & & \\
\triangleq \{X^0, & \text{where} & X^0 \in D^0 \\
\quad \ldots & & \ldots \\
\quad X^0 \bullet X^1 \bullet X^2 \bullet \cdots \bullet X^{k+1}, & & X^{k+1} \triangleq \begin{cases} F^{k+1}(X^k) \in D^{k+1} & \text{if } \bigwedge_{i=0}^{k} \neg C^i(X^i) \\ X^\ell \in D^\ell & \text{if } \exists \ell \leqslant k.\ \bigwedge_{i=0}^{\ell-1} \neg C^i(X^i) \wedge C^\ell(X^\ell) \end{cases} \\
\quad \ldots & & \ldots \\
\quad X^0 \bullet X^1 \bullet \cdots \bullet X^k \bullet \cdots \bullet X^\omega\} & & X^\omega \triangleq F^\omega(\langle X^k,\ k \in \mathbb{N}\rangle) \in D^\omega
\end{array} \tag{8}$$

The semantics $\mathcal{S}_{\mathsf{pf}}[\![\mathbf{A}]\!]\,(X^0)$ of $\mathbf{A}$ is given by the infinite set of nonempty prefixes of its (possibly ultimately stationary) ω+1-transfinite sequence of iterates starting from X^0. Clearly, this set of sequences is closed by prefixes and is therefore called the prefix trace semantics of $\mathbf{A}$ starting from X^0. It is worth noting that this prefix semantics is a refinement of the standard invariance collecting semantics which for each reachable state X^k records the history of the computation $X^0 \bullet X^1 \bullet X^2 \bullet \cdots \bullet X^k$ leading to that state X^k, thus observing the past evolution of computations.

The prefix trace semantics $\mathcal{S}_{\mathsf{pf}}[\![\mathbf{A}]\!]\,(X^0)$ defined in (8) can be easily characterized as a $\subseteq$-least fixpoint of an operator $\mathcal{F}_{\mathsf{pf}}(X^0)$ on sets of sequences in $\wp(\mathbf{D}^{+\omega+1})$ defined as follows:

$$\begin{array}{rcl}
\mathcal{F}_{\mathsf{pf}}(X^0) & \in & \wp(\mathbf{D}^{+\omega+1}) \xrightarrow{\nearrow} \wp(\mathbf{D}^{+\omega+1}) \\
\mathcal{F}_{\mathsf{pf}}(X^0)\ \mathcal{X} & \triangleq & \{X^0\} \\
& & \cup\, \{X \bullet F^{k+1}(X^k) \mid \exists k \in \mathbb{N}.\, X \in \mathbf{D}^k \wedge X \in \mathcal{X} \wedge \mathsf{it}(X)\} \\
& & \cup\, \{X \bullet X^k \mid \exists k \in \mathbb{N}.\, X \in \mathbf{D}^k \wedge X \in \mathcal{X} \wedge \mathsf{st}(X)\} \\
& & \cup\, \{X \bullet X^\omega \mid X \in \mathbf{D}^\omega \wedge \mathsf{pref}(X) \subseteq \mathcal{X} \wedge \mathsf{wb}(X) \wedge X^\omega = F^\omega(X)\}
\end{array} \tag{9}$$

Proposition 5.1. *The transfinite iterates $\langle \mathcal{F}_{\mathsf{pf}}(X^0)^k\ \emptyset,\ k \in \mathbb{N} \cup \{\omega, \omega{+}1\}\rangle$ of $\mathcal{F}_{\mathsf{pf}}(X^0)$ from $\emptyset$ (or $\langle \mathcal{F}_{\mathsf{pf}}(X^0)^k\ X^0, k \in \mathbb{N} \cup \{\omega, \omega{+}1\}\rangle$ from X^0) in the complete lattice $\langle \wp(\mathbf{D}^{+\omega+1}), \subseteq, \emptyset, \cup\rangle$ are increasing and converge to the least fixpoint*

$$\mathcal{S}_{\mathsf{pf}}[\![\mathbf{A}]\!]\,(X^0) \;=\; \mathsf{lfp}^{\subseteq}\, \mathcal{F}_{\mathsf{pf}}(X^0) \tag{10}$$

Let $\mathcal{D}_{\mathsf{pf}}(X^0) \triangleq \{\mathsf{pref}(\{\sigma\}) \mid \sigma \in \mathbf{D}^{+\omega+1}(X^0)\}$ denote the set of nonempty prefixes of a nonempty sequence starting from X^0. Then, it turns out that, for all k, $\mathcal{F}_{\mathsf{pf}}(X^0)^k\ \emptyset \in \mathcal{D}_{\mathsf{pf}}(X^0)$, so that $\mathcal{S}_{\mathsf{pf}}[\![\mathbf{A}]\!]\, X^0 = \mathsf{lfp}^{\subseteq}\, \mathcal{F}_{\mathsf{pf}}(X^0) \in \mathcal{D}_{\mathsf{pf}}(X^0)$.

5.2.3 *Prefix Semantics of the Interval Abstract Interpreter.*

The meta-iterates $\langle \mathcal{X}^k, k \in \mathbb{N} \cup \{\omega, \omega{+}1\}\rangle$ of Proposition 5.1 providing the prefix trace semantics of the interval abstract interpreter for the Gauss-Seidel chaotic iterates of the program (6) in Section 4.2.2 are as follows:

$$\mathcal{X}^0 = \emptyset,\ \mathcal{X}^1 = \left\{\begin{bmatrix}\top\\\bot\\\bot\\\bot\end{bmatrix}\right\},\ \mathcal{X}^2 = \left\{\begin{bmatrix}\top\\\bot\\\bot\\\bot\end{bmatrix}, \begin{bmatrix}\top\\\bot\\\bot\\\bot\end{bmatrix} \bullet \begin{bmatrix}\top\\ [0,0]\\ [2,2]\\\bot\end{bmatrix}\right\},\ \mathcal{X}^3 = \left\{\begin{bmatrix}\top\\\bot\\\bot\\\bot\end{bmatrix}, \begin{bmatrix}\top\\\bot\\\bot\\\bot\end{bmatrix} \bullet \begin{bmatrix}\top\\ [0,0]\\ [2,2]\\\bot\end{bmatrix}, \begin{bmatrix}\top\\\bot\\\bot\\\bot\end{bmatrix} \bullet \begin{bmatrix}\top\\ [0,0]\\ [2,2]\\\bot\end{bmatrix} \bullet \begin{bmatrix}\top\\ [0,2]\\ [2,4]\\\bot\end{bmatrix}\right\}, \ldots,$$

$$\mathcal{X}^{k+1} = \left\{\begin{bmatrix}\top\\\bot\\\bot\\\bot\end{bmatrix}, \begin{bmatrix}\top\\\bot\\\bot\\\bot\end{bmatrix} \bullet \begin{bmatrix}\top\\ [0,0]\\ [2,2]\\\bot\end{bmatrix}, \ldots, \begin{bmatrix}\top\\\bot\\\bot\\\bot\end{bmatrix} \bullet \begin{bmatrix}\top\\ [0,0]\\ [2,2]\\\bot\end{bmatrix} \bullet \begin{bmatrix}\top\\ [0,2]\\ [2,4]\\\bot\end{bmatrix} \bullet \cdots \bullet \begin{bmatrix}\top\\ [0,2*k]\\ [2,2*(k+1)]\\\bot\end{bmatrix}\right\}, \ldots,$$

$$\mathcal{X}^{\omega} = \left\{\begin{bmatrix}\top\\\bot\\\bot\\\bot\end{bmatrix}, \begin{bmatrix}\top\\\bot\\\bot\\\bot\end{bmatrix} \bullet \begin{bmatrix}\top\\ [0,0]\\ [2,2]\\\bot\end{bmatrix}, \ldots, \begin{bmatrix}\top\\\bot\\\bot\\\bot\end{bmatrix} \bullet \begin{bmatrix}\top\\ [0,0]\\ [2,2]\\\bot\end{bmatrix} \bullet \begin{bmatrix}\top\\ [0,2]\\ [2,4]\\\bot\end{bmatrix} \bullet \cdots \bullet \begin{bmatrix}\top\\ [0,2*k]\\ [2,2*(k+1)]\\\bot\end{bmatrix} \bullet \begin{bmatrix}\top\\ [0,2*(k+1)]\\ [2,2*(k+2)]\\\bot\end{bmatrix} \bullet \cdots\right\},$$

$$\mathcal{X}^{\omega+1} = \left\{\begin{bmatrix}\top\\\bot\\\bot\\\bot\end{bmatrix}, \ldots, \begin{bmatrix}\top\\\bot\\\bot\\\bot\end{bmatrix} \bullet \cdots \bullet \begin{bmatrix}\top\\ [0, 2*k]\\ [2, 2*(k+1)]\\ \bot\end{bmatrix}, \ldots, \begin{bmatrix}\top\\\bot\\\bot\\\bot\end{bmatrix} \bullet \cdots \bullet \begin{bmatrix}\top\\ [0, 2*k]\\ [2, 2*(k+1)]\\ \bot\end{bmatrix} \bullet \cdots, \right.$$

$$\left. \begin{bmatrix}\top\\\bot\\\bot\\\bot\end{bmatrix} \bullet \begin{bmatrix}\top\\ [0,0]\\ [2,2]\\ \bot\end{bmatrix} \bullet \begin{bmatrix}\top\\ [0,2]\\ [2,4]\\ \bot\end{bmatrix} \bullet \cdots \bullet \begin{bmatrix}\top\\ [0, 2*k]\\ [2, 2*(k+1)]\\ \bot\end{bmatrix} \bullet \cdots \bullet \begin{bmatrix}\top\\ [0,\infty]\\ [2,\infty]\\ \bot\end{bmatrix}\right\}.$$

5.3 Maximal Trace Semantics of the Abstract Interpreter

Because sets of the semantic domain $\mathcal{D}_{\mathsf{pf}}(X^0)$ are prefix closed, an equivalent representation of some $\mathcal{X} \in \mathcal{D}_{\mathsf{pf}}(X^0)$ could be given by the longest sequence in $\mathcal{X}$, if this exists. Indeed, the limit meta-iterate $\mathcal{F}_{\mathsf{pf}}(X^0)^\omega\,\emptyset = \{X^0 \bullet X^1 \bullet \cdots \bullet X^k \mid k \geqslant 0\}$ is prefix closed and has no maximal sequence: in this case, the ω-infinite sequence $X^0 \bullet X^1 \bullet \cdots \bullet X^k \bullet \cdots$ is taken as its representation.

5.3.1 Maximal Trace Isomorphic Abstraction. Let us define $\lim \in \wp(\mathbf{D}^{+\omega+1}) \longrightarrow \wp(\mathbf{D}^{+\omega+1})$ which adds the ω-infinite limits of sets of sequences as follows:

$$\lim(S) \quad \triangleq \quad S \cup \{X \in \mathbf{D}^\omega \mid \mathsf{pref}(\{X\}) \subseteq S\}$$

Note that lim is an upper closure operator on $\langle \wp(\mathbf{D}^{+\omega+1}), \subseteq\rangle$ and $\langle \mathbf{D}^{+\omega+1}(X^0), \preccurlyeq_{\mathsf{pf}}, X_0, \curlyvee\rangle$ is a cpo, where $\preccurlyeq_{\mathsf{pf}}$ is the prefix partial order between sequences: $\sigma \preccurlyeq_{\mathsf{pf}} \sigma' \triangleq \exists \sigma'' \in \mathbf{D}^{*\omega+1} \,.\, \sigma \bullet \sigma'' = \sigma'$. In particular X^0 is the least element and that if $\sigma, \sigma' \in \mathbf{D}^{+\omega+1}(X^0)$ are both ω-infinite or ω+1-transfinite then σ and σ' are uncomparable unless they are equal. The lub $\curlyvee C$ of a nonempty chain C in $\mathbf{D}^{+\omega+1}(X^0)$ is the longest (with length in $\mathbb{N}_+ \cup \{\omega, \omega{+}1\}$) sequence in $\lim(C)$.

Recall that $\mathcal{D}_{\mathsf{pf}}(X^0) \triangleq \{\mathsf{pref}(\{\sigma\}) \mid \sigma \in \mathbf{D}^{+\omega+1}(X^0)\}$. It turns out that we have a Galois isomorphism $\langle \mathcal{D}_{\mathsf{pf}}(X^0), \subseteq\rangle \overset{\gamma_{\mathsf{m}}}{\underset{\alpha_{\mathsf{m}}}{\leftrightarrows}} \langle \mathbf{D}^{+\omega+1}(X^0), \preccurlyeq_{\mathsf{pf}}\rangle$ called maximal trace isomorphic abstraction, where, for all $\sigma \in \mathbf{D}^{+\omega+1}(X^0)$, $\alpha_{\mathsf{m}}(\mathsf{pref}(\{\sigma\})) \triangleq \curlyvee \mathsf{pref}(\{\sigma\})$ and $\gamma_{\mathsf{m}}(\sigma) \triangleq \mathsf{pref}(\{\sigma\})$.

5.3.2 Maximal Trace Semantics of the Abstract Interpreter. The maximal trace semantics $\mathcal{S}_{\mathsf{m}}[\![\mathbf{A}]\!]\,X^0$ of the abstract interpreter $\mathbf{A}$ is defined as maximal trace abstraction of its prefix trace semantics:

$$\mathcal{S}_{\mathsf{m}}[\![\mathbf{A}]\!]\,X^0 \;\triangleq\; \alpha_{\mathsf{m}}(\mathcal{S}_{\mathsf{pf}}[\![\mathbf{A}]\!]\,X^0) \;\in\; \mathcal{D}_{\mathsf{m}}(X^0) \;\triangleq\; \mathbf{D}^{+\omega+1}(X^0)$$

This maximal trace semantics can be characterized as limit of the transfinite iterates of the following function $\mathcal{F}_{\mathsf{m}}(X^0)$:

$$\mathcal{F}_{\mathsf{m}}(X^0) \in \mathcal{D}_{\mathsf{m}}(X^0) \longrightarrow \mathcal{D}_{\mathsf{m}}(X^0) \tag{11}$$

$$\mathcal{F}_{\mathsf{m}}(X^0)\; X \;\triangleq\; \begin{cases} X \bullet F^{k+1}(X^k) & \text{if } \exists k \in \mathbb{N} \,.\, X \in \mathbf{D}^k \wedge \mathsf{it}(X)\\ X \bullet X^k & \text{if } \exists k \in \mathbb{N} \,.\, X \in \mathbf{D}^k \wedge \mathsf{st}(X)\\ X \bullet X^\omega & \text{if } X \in \mathbf{D}^\omega \wedge \mathsf{wb}(X) \wedge X^\omega = F^\omega(X)\\ X^0 & \text{otherwise}\end{cases}$$

Proposition 5.2. *The transfinite iterates* $\langle \mathcal{F}_{\mathsf{m}}(X^0)^k\,X^0,\; k \in \mathbb{N} \cup \{\omega, \omega{+}1\}\rangle$ *of* $\mathcal{F}_{\mathsf{m}}(X^0)$ *from* X^0 *in the cpo* $\langle \mathcal{D}_{\mathsf{m}}(X^0), \preccurlyeq_{\mathsf{pf}}, X_0, \curlyvee\rangle$ *are* $\preccurlyeq_{\mathsf{pf}}$*-increasing and converge to the limit*

$$\mathcal{S}_{\mathsf{m}}[\![\mathbf{A}]\!]\,X^0 \;=\; \curlyvee_{k \in \mathbb{N} \cup \{\omega, \omega+1\}} \mathcal{F}_{\mathsf{m}}(X^0)^k X^0 \in \mathcal{D}_{\mathsf{m}}(X^0) \tag{12}$$

which is a minimal fixpoint of $\mathcal{F}_{\mathsf{m}}(X^0)$ (*but not necessarily the least one which may not exist*).

It turns out that if $\langle \mathcal{X}^k,\; k \in \mathbb{N}_+ \cup \{\omega, \omega{+}1\}\rangle$ are the iterates of $\mathcal{F}_{\mathsf{pf}}(X^0)$ defining $\mathcal{S}_{\mathsf{pf}}[\![\mathbf{A}]\!]\,X^0$ then the iterates of $\mathcal{F}_{\mathsf{m}}(X^0)$ converging to $\mathcal{S}_{\mathsf{m}}[\![\mathbf{A}]\!]\,X^0$ are exactly $\langle \alpha_{\mathsf{m}}(\mathcal{X}^k),\; k \in \mathbb{N} \cup \{\omega, \omega{+}1\}\rangle$. It is worth remarking that the absence of the least fixpoint for $\mathcal{F}_{\mathsf{m}}(X^0)$ in (12) is a consequence of the fact that the transformer $\mathcal{F}_{\mathsf{m}}(X^0)$ can also extend invalid sequences, *i.e.*, sequences in $\mathcal{D}_{\mathsf{m}}(X^0)$

which do not correspond to executions of the abstract interpreter and might be incomparable with $\mathcal{S}_{\mathsf{m}}[\![\mathbf{A}]\!]\,X^0$. Clearly, $\mathcal{S}_{\mathsf{m}}[\![\mathbf{A}]\!]\,X^0$ is the least fixpoint of $\mathcal{F}_{\mathsf{m}}(X^0)$ when $\mathcal{F}_{\mathsf{m}}(X^0)$ is restricted to its iterates starting from X^0.

5.3.3 Maximal Trace Semantics of the Interval Abstract Interpreter. $\langle X^i, i \in \mathbb{N} \cup \{\omega, \omega+1\}\rangle$ are the iterates of $\mathcal{F}_{\mathsf{m}}(X^0)$ of Proposition 5.2, providing the maximal trace semantics of the interval abstract interpreter for the Gauss-Seidel chaotic iterates of the program (6) in Section 4.2.2:

$$X^0 = \begin{bmatrix}\top\\ \bot\\ \bot\\ \bot\end{bmatrix},\; X^1 = \begin{bmatrix}\top\\ \bot\\ \bot\\ \bot\end{bmatrix} \bullet \begin{bmatrix}\top\\ [0,0]\\ [2,2]\\ \bot\end{bmatrix},\; X^2 = \begin{bmatrix}\top\\ \bot\\ \bot\\ \bot\end{bmatrix} \bullet \begin{bmatrix}\top\\ [0,0]\\ [2,2]\\ \bot\end{bmatrix} \bullet \begin{bmatrix}\top\\ [0,2]\\ [2,4]\\ \bot\end{bmatrix}, \ldots,$$

$$X^{k+1} = \begin{bmatrix}\top\\ \bot\\ \bot\\ \bot\end{bmatrix} \bullet \begin{bmatrix}\top\\ [0,0]\\ [2,2]\\ \bot\end{bmatrix} \bullet \cdots \bullet \begin{bmatrix}\top\\ [0,2*k]\\ [2,2*(k+1)]\\ \bot\end{bmatrix}, \ldots, X^{\omega} = \begin{bmatrix}\top\\ \bot\\ \bot\\ \bot\end{bmatrix} \bullet \begin{bmatrix}\top\\ [0,0]\\ [2,2]\\ \bot\end{bmatrix} \bullet \cdots \bullet \begin{bmatrix}\top\\ [0,2*k]\\ [2,2*(k+1)]\\ \bot\end{bmatrix} \bullet \cdots$$

$$X^{\omega+1} = \begin{bmatrix}\top\\ \bot\\ \bot\\ \bot\end{bmatrix} \bullet \begin{bmatrix}\top\\ [0,0]\\ [2,2]\\ \bot\end{bmatrix} \bullet \cdots \bullet \begin{bmatrix}\top\\ [0,2*k]\\ [2,2*(k+1)]\\ \bot\end{bmatrix} \bullet \cdots \bullet \begin{bmatrix}\top\\ [0,\infty]\\ [2,\infty]\\ \bot\end{bmatrix}.$$

5.4 Projection Semantics of the Abstract Interpreter

For the fixpoint and chaotic abstract interpreters described in Sections 4.1.1 and 4.1.2, the abstract domain at iteration k is a Cartesian product $\prod_{i=1}^{n} D_i^k$, with $n \geq 1$, so that $\mathbf{D}$ consists of n-ary vectors. We assume that the same domain is used at each program point, that is, for all $i \in [1,n]$, $D_i^k = D^k$. The abstract interpreter at iteration k attaches an abstract information ranging in D^k to each program point $i \in [1,n]$ of the program it analyses. A meta-analyser may want to analyse this local (to a program point) abstract information rather than the global abstract information provided by the maximal trace semantics $\mathcal{S}_{\mathsf{m}}[\![\mathbf{A}]\!]\,X^0$, as we did in the introductory example of Section 2. This can be done by relying on a projection abstraction.

5.4.1 Projection Isomorphic Abstraction. Let $\mathbf{D} = \biguplus_{k\in\mathbb{N}\cup\{\omega\}} \prod_{i=1}^{n} D^k$ and $\mathbf{D}_i \triangleq \biguplus_{k\in\mathbb{N}\cup\{\omega\}} D^k$, with $i \in [1,n]$, where the subscript hints that $\mathbf{D}_i$ refers to the program point i. In each iterate of the abstract interpreter, we abstract by α_{pr} a sequence of vectors in $\mathbf{D}$ to a n-ary vector of sequences in $(\mathbf{D}_i)^{+\omega+1}$ simply by taking the n sequences of abstract values in $\mathbf{D}$, one for each program point:

$$\alpha_{\mathsf{pr}}\left(\begin{bmatrix}v_1^0\\ \vdots\\ v_n^0\end{bmatrix} \bullet \begin{bmatrix}v_1^1\\ \vdots\\ v_n^1\end{bmatrix} \bullet \cdots \bullet \begin{bmatrix}v_1^k\\ \vdots\\ v_n^k\end{bmatrix} \bullet \cdots\right) \;\triangleq\; \begin{bmatrix}v_1^0 \bullet v_1^1 \bullet \cdots \bullet v_1^k \bullet \cdots\\ \vdots\\ v_n^0 \bullet v_n^1 \bullet \cdots \bullet v_n^k \bullet \cdots\end{bmatrix}$$

This abstraction gives rise to a Galois isomorphism $\langle \mathbf{D}^{0,+\omega+1}, \preccurlyeq_{\mathsf{pf}}\rangle \xtwoheadleftarrow[\alpha_{\mathsf{pr}}]{\gamma_{\mathsf{pr}}} \langle \prod_{i=1}^{n} (\mathbf{D}_i)^{+\omega+1}, \dot{\preccurlyeq}_{\mathsf{pf}}\rangle$.

5.4.2 Projection Abstraction of the Interval Abstract Interpreter. Continuing Example 5.3.3, we get the following transfinite sequence of vectors of sequences:

$$\dot{X}^0 = \begin{bmatrix}\top\\ \bot\\ \bot\\ \bot\end{bmatrix},\; \dot{X}^1 = \begin{bmatrix}\top \bullet \top\\ \bot \bullet [0,0]\\ \bot \bullet [2,2]\\ \bot \bullet \bot\end{bmatrix},\; \dot{X}^2 = \begin{bmatrix}\top \bullet \top \bullet \top\\ \bot \bullet [0,0] \bullet [0,2]\\ \bot \bullet [2,2] \bullet [2,4]\\ \bot \bullet \bot \bullet \bot\end{bmatrix}, \ldots,\; \dot{X}^{k+1} = \begin{bmatrix}\top \bullet \top \bullet \cdots \bullet \top\\ \bot \bullet [0,0] \bullet \cdots \bullet [0,2*k]\\ \bot \bullet [2,2] \bullet \cdots \bullet [2,2*(k+1)]\\ \bot \bullet \bot \bullet \cdots \bullet \bot\end{bmatrix}, \ldots,$$

$$\dot{X}^{\omega} = \begin{bmatrix}\top \bullet \top \bullet \cdots \bullet \top \bullet \cdots\\ \bot \bullet [0,0] \bullet \cdots \bullet [0,2*k] \bullet \cdots\\ \bot \bullet [2,2] \bullet \cdots \bullet [2,2*(k+1)] \bullet \cdots\\ \bot \bullet \bot \bullet \cdots \bullet \bot \bullet \cdots\end{bmatrix},\; \dot{X}^{\omega+1} = \begin{bmatrix}\top \bullet \top \bullet \cdots \bullet \top \bullet \cdots \bullet \top\\ \bot \bullet [0,0] \bullet \cdots \bullet [0,2*k] \bullet \cdots \bullet [0,\infty]\\ \bot \bullet [2,2] \bullet \cdots \bullet [2,2*(k+1)] \bullet \cdots \bullet [2,\infty]\\ \bot \bullet \bot \bullet \cdots \bullet \bot \bullet \cdots \bullet \bot\end{bmatrix}.$$

5.4.3 Projection Semantics of the Abstract Interpreter. The projection semantics $\mathcal{S}_{\mathsf{pr}}[\![\mathbf{A}]\!]X^0$ of the abstract interpreter $\mathbf{A}$, for an initial n-dimensional vector $X^0 \in \prod_{i=1}^{n} D^0$, is defined as projection abstraction of the maximal trace semantics of **A**:

$$\mathcal{S}_{\mathsf{pr}}\llbracket \mathbf{A} \rrbracket X^0 \ \triangleq\ \alpha_{\mathsf{pr}}(\mathcal{S}_{\mathsf{m}}\llbracket \mathbf{A} \rrbracket X^0) = \alpha_{\mathsf{pr}}(\alpha_{\mathsf{m}}(\mathcal{S}_{\mathsf{pf}}\llbracket \mathbf{A} \rrbracket X^0)) \ \in\ \mathcal{D}_{\mathsf{pr}}(X^0) \ \triangleq\ (\textstyle\prod_{i=1}^{n}(\mathbf{D}_i)^{+\omega+1})(X^0)$$

This projection semantics can be characterized as limit of the transfinite iterates of the following function $\mathcal{F}_{\mathsf{pr}}(X^0)$:

$$\mathcal{F}_{\mathsf{pr}}(X^0) \in \mathcal{D}_{\mathsf{pr}}(X^0) \longrightarrow \mathcal{D}_{\mathsf{pr}}(X^0) \tag{13}$$

$$\mathcal{F}_{\mathsf{pr}}(X^0)\ \dot{X} \triangleq \begin{cases} \dot{X} \cdot F^{k+1}(\dot{X}^k) & \text{if } \exists k \in \mathbb{N} \,.\, \dot{X} \in \prod_{i=1}^{n}(\mathbf{D}_i)^k \wedge \mathsf{it}_{\mathsf{pr}}(\dot{X}) \\ \dot{X} \cdot \dot{X}^k & \text{if } \exists k \in \mathbb{N} \,.\, \dot{X} \in \prod_{i=1}^{n}(\mathbf{D}_i)^k \wedge \mathsf{st}_{\mathsf{pr}}(\dot{X}) \\ \dot{X} \cdot \dot{X}^\omega & \text{if } \dot{X} \in \prod_{i=1}^{n}(\mathbf{D}_i)^\omega \wedge \mathsf{wb}_{\mathsf{pr}}(\dot{X}) \wedge \dot{X}^\omega = F^\omega(\dot{X}) \\ X^0 & \text{otherwise} \end{cases}$$

where $\mathsf{it}_{\mathsf{pr}}(\dot{X}) \triangleq \mathsf{it}(\gamma_{\mathsf{pr}}(\dot{X}))$ and similarly for $\mathsf{st}_{\mathsf{pr}}$ and $\mathsf{wb}_{\mathsf{pr}}$.

Proposition 5.3. *The transfinite iterates* $\langle \mathcal{F}_{\mathsf{pr}}(X^0)^k\, X^0,\ k \in \mathbb{N} \cup \{\omega, \omega{+}1\}\rangle$ *of* $\mathcal{F}_{\mathsf{pr}}(X^0)$ *from* X^0 *in the cpo* $\langle \mathcal{D}_{\mathsf{pr}}(X^0),\ \dot{\preccurlyeq}_{\mathsf{pf}},\ X^0,\ \dot{\curlyvee}\rangle$ *are* $\dot{\preccurlyeq}_{\mathsf{pf}}$*-increasing and converge to the limit*

$$\mathcal{S}_{\mathsf{pr}}\llbracket \mathbf{A} \rrbracket\, X^0 \quad = \quad \dot{\curlyvee}_{k \,\in\, \mathbb{N} \cup \{\omega, \omega+1\}} \quad \mathcal{F}_{\mathsf{pr}}(X^0)^k\, X^0 \quad \in \quad \mathcal{D}_{\mathsf{pr}}(X^0) \tag{14}$$

which is a minimal fixpoint of $\mathcal{F}_{\mathsf{pr}}(X^0)$ *(but not necessarily the least one which may not exist).*

6 SOUNDNESS OF THE ABSTRACT INTERPRETER

Assume that $\mathbf{A} = \langle D^k, C^k, F^k, X^0\rangle$ (where this simplified notation is meant to include D^ω and F^ω) is a "concrete" instance of the abstract interpreter (5) while $\overline{\mathbf{A}} = \langle \overline{D}^k, \overline{C}^k, \overline{F}^k, \overline{X}^0\rangle$ is a second "abstract" instance. Recall that $\mathbf{D} = \biguplus_{k \in \mathbf{N} \cup \{\omega\}} D^k$ and $\overline{\mathbf{D}} = \biguplus_{k \in \mathbf{N} \cup \{\omega\}} \overline{D}^k$. Let $\sqsubseteq$ be a partial order on $\mathbf{D}$, modeling logical implication of concrete properties, and $\gamma \in \overline{\mathbf{D}} \longrightarrow \mathbf{D}$ be a concretization function providing a notion of sound approximation for abstract values: $\overline{d} \in \overline{\mathbf{D}}$ is a sound approximation of $d \in \mathbf{D}$ when $d \sqsubseteq \gamma(\overline{d})$ holds. Assume that $=/\sqsubseteq$ consistently denotes either $=$ or $\sqsubseteq$. Let $\dot{=}/\dot{\sqsubseteq}\ \in\ \wp(\mathbf{D}^{+\omega+1}(X^0) \times \mathbf{D}^{+\omega+1}(X^0))$ and $\dot{\gamma}\ \in\ \overline{\mathbf{D}}^{+\omega+1}(X^0)\ \longrightarrow\ \mathbf{D}^{+\omega+1}(X^0)$ be the pointwise extensions of $=/\sqsubseteq\ \in \wp(\mathbf{D} \times \mathbf{D})$ and $\gamma \in \overline{\mathbf{D}} \longrightarrow \mathbf{D}$ to sequences in $\mathbf{D}^{+\omega+1}(X^0)$.

Let $\langle X^k,\ k \in \mathbb{N} \cup \{\omega, \omega+1\}\rangle$ be a well-behaved sequence of iterates for **A** and, analogously, $\langle \overline{X}^k,\ k \in \mathbb{N} \cup \{\omega, \omega+1\}\rangle$ for $\overline{\mathbf{A}}$. We define the following conditions relating these sequences through γ:

(6.a) $X^0 \ =/\sqsubseteq\ \gamma(\overline{X}^0)$;
(6.b) $\forall k \in \mathbb{N} \,.\, (\neg C^k(X^k) \wedge \neg\overline{C}^k(\overline{X}^k) \wedge X^k \ =/\sqsubseteq\ \gamma(\overline{X}^k)) \implies F^{k+1}(X^k) \ =/\sqsubseteq\ \gamma(\overline{F}^{k+1}(\overline{X}^k))$;
(6.c) $\forall k, \ell \in \mathbb{N} \,.\, (\ell < k \wedge \neg C^k(X^k) \wedge \overline{C}^\ell(\overline{X}^\ell) \wedge X^k \ =/\sqsubseteq\ \gamma(\overline{X}^k)) \implies (F^{k+1}(X^k) \ =/\sqsubseteq\ \gamma(\overline{X}^k))$;
(6.d) $\forall k, \ell \in \mathbb{N} \,.\, (\ell < k \wedge C^\ell(X^\ell) \wedge \neg\overline{C}^k(\overline{X}^k) \wedge X^k \ =/\sqsubseteq\ \gamma(\overline{X}^k)) \implies (X^k \ =/\sqsubseteq\ \gamma(\overline{F}^{k+1}(\overline{X}^k))$;
(6.e) $(\forall k \in \mathbb{N} \,.\, X^k \ =/\sqsubseteq\ \gamma(\overline{X}^k)) \implies (F^\omega(\langle X^k,\ k \in \mathbb{N}\rangle) \ =/\sqsubseteq\ \gamma(\overline{F}^\omega(\langle \overline{X}^k,\ k \in \mathbb{N}\rangle))))$.

Proposition 6.1. *Under hypotheses (6.a)–(6.e),* $\mathcal{S}_{\mathsf{m}}\llbracket \mathbf{A} \rrbracket\, X^0 \ \dot{=}/\dot{\sqsubseteq}\ \dot{\gamma}(\mathcal{S}_{\mathsf{m}}\llbracket \overline{\mathbf{A}} \rrbracket\ \overline{X}^0)$ *holds.*

Proposition 6.1 establishes a very general soundness of the abstract interpreter $\overline{\mathbf{A}}$ with respect to **A** for the maximal trace semantics. Since the maximal trace semantics of the concrete interpreter is the transfinite sequence $X = \mathcal{S}_{\mathsf{m}}\llbracket \mathbf{A} \rrbracket\, X^0 = X^0 \cdot X^1 \cdot X^2 \cdot\ \cdots\ \cdot X^k \cdot\ \cdots\ \cdot X^\omega$ and similarly for the abstract interpreter $\overline{X} = \mathcal{S}_{\mathsf{m}}\llbracket \overline{\mathbf{A}} \rrbracket\ \overline{X}^0 = \overline{X}^0 \cdot \overline{X}^1 \cdot \overline{X}^2 \cdot\ \cdots\ \cdot \overline{X}^\ell \cdot\ \cdots\ \cdot \overline{X}^\omega$, it is worth observing that $X \ \dot{=}/\dot{\sqsubseteq}\ \dot{\gamma}(\overline{X})$ implies $X^\omega \ =/\sqsubseteq\ \gamma(\overline{X}^\omega)$, *i.e.*, the soundness of the limit result $\overline{X}^\omega$ of the abstract interpreter $\overline{\mathbf{A}}$ with respect to the limit result X^ω of **A**.

Example 6.2 (Reachability Semantics). A classical example is the reachability semantics $\mathsf{post}[t^*]I$ of a transition system $\langle S,\ I,\ t\rangle$ where S is a set of states, $I \subseteq S$ are initial states, $t \in \wp(S \times S)$ is a transition relation, $\mathrm{post}[t]X \triangleq \{s' \in S \mid \exists s \in X \,.\, \langle s,\ s'\rangle \in t\}$ is the standard post-image of $X \subseteq S$ through the relation t, and t^* is the reflexive-transitive closure of t. This reachability semantics is an instance $\mathbf{A}$ of (5) where: for all $k \in \mathbb{N}$, $D^k = \wp(S)$, $F^{k+1}(X) = I \cup \mathrm{post}[t]X$, $C^k(X) = (F^k(X) \subseteq X)$, $D^\omega = \wp(S)$, $F^\omega(\langle X^k,\ k \in \mathbb{N}\rangle) = \bigcup_{k\in\mathbb{N}} X^k$, $X^0 = I$. The partial order $\sqsubseteq$ on $\mathbf{D} = \wp(S)$ is logical implication $\subseteq$. The soundness of an abstract interpreter $\overline{\mathbf{A}} = \langle \overline{D}^k, \overline{C}^k, \overline{F}^k, \overline{X}^0\rangle$ is often based on the following hypotheses: $\forall k \in \mathbb{N} \,.\, I \subseteq \gamma(\overline{X}^k)$, $\forall k \in \mathbb{N} . \forall \overline{X} \in \overline{D}^k \,.\, \mathrm{post}[t]\gamma(\overline{X}) \subseteq \gamma(\overline{F}^{k+1}(\overline{X}))$ (semi-commutation), $\overline{C}^k(\overline{X}) \implies (\gamma(\overline{F}^{k+1}(\overline{X})) \subseteq \gamma(\overline{X}))$ (convergence), and $\bigcup_{k\in\mathbb{N}} \gamma(\overline{X}^k) \subseteq \overline{F}^\omega(\langle \overline{X}^k,\ k \in \mathbb{N}\rangle)$. These satisfy (6.a)–(6.e), so that, by Proposition 6.1, we obtain a generalization of the standard soundness result $\mathrm{post}[t^*]I \subseteq \gamma((\mathcal{S}_{\mathrm{m}}[\![\overline{\mathbf{A}}]\!]\, \overline{X}^0)^\omega)$. □

7 COLLECTING SEMANTICS FOR OFFLINE META-ANALYSIS

Offline meta-analysis consists in performing an analysis of the abstract interpreter $\mathbf{A}$ which analyses a given program P before performing the analysis of P, that is, before executing $\mathbf{A}$ for that program P. Being P fixed so are $\langle D^k, C^k, F^k\rangle$, which are a function of P. Thus, the whole theory of abstract interpretation can be used for designing a static analysis of the abstract interpreter $\mathbf{A}$. To design such a meta-abstract interpreter, we first select a semantics of the abstract interpreter $\mathbf{A}$ among those given in Section 5, then we define its collecting semantics, next the abstraction of this collecting semantics, and finally we design the meta-abstract interpreter as an instance of (5).

7.1 From Semantics to Static Analyses

Semantics $S \in \mathcal{D}$ are defined on a computational domain $\mathcal{D}$ as limits of iterates $\langle X^k, k \in \Delta\rangle$ ranging in $\mathcal{D}$ and partially ordered by $\sqsubseteq$ (which is a total order on the iterates). Static analyses operate on properties of the semantics in $\wp(\mathcal{D})$ and are defined as limits of iterates of semantic properties $\langle \overline{X}^k, k \in \overline{\Delta}\rangle$. In order to connect semantics and static analyses, the (strongest) collecting semantics $\mathrm{C} \triangleq \{S\}$ is defined as the strongest property of S, so that collecting semantics and static analyses are both properties in $\wp(\mathcal{D})$. C is the limit of the "singleton" iterates $\langle\{X^k\}, k \in \Delta\rangle$. Although the order for the "singleton" iterates $\langle\{X^k\}, k \in \Delta\rangle$ could be the straightforward lifting of $\sqsubseteq$ from $\mathcal{D}$ to singletons of $\wp(\mathcal{D})$, we aim at defining an order on $\wp(\mathcal{D})$ that also works for static analyses, *i.e.*, which is not restricted to singletons of $\wp(\mathcal{D})$ only but is defined for arbitrary properties in $\wp(\mathcal{D})$. Thus, up to some abstraction, this order should be the one which provides the ordering of the abstract iterations $\langle \overline{X}^k, k \in \overline{\Delta}\rangle$ whose limit is the result of the static analysis.

As usual, we represent extensionally properties by the set of elements having this property. Thus, let the semantics $\mathcal{S}[\![\mathbf{A}]\!]\, X^0 \in \mathcal{D}$ of the abstract interpreter be any one defined in Section 5 where $\mathcal{D}$ is the underlying cpo. The collecting semantics is the strongest property of the abstract interpreter, that is $\mathcal{C}[\![\mathbf{A}]\!]\, X^0 \triangleq \{\mathcal{S}[\![\mathbf{A}]\!]\, X^0\} \in \wp(\mathcal{D})$. $\mathbf{A}$ has a property $P \in \wp(\mathcal{D})$ if and only if it is implied by the strongest one, *i.e.*, $\mathcal{C}[\![\mathbf{A}]\!]\, X^0 \subseteq P$, or, equivalently, $\mathcal{S}[\![\mathbf{A}]\!]\, X^0 \in P$. For some static analyses, a weaker collecting semantics is chosen (abstracting the strongest one) such as the reachable collecting semantics described in Example 6.2 as in classical abstract interpretation [Cousot and Cousot 1977a] for invariant generation.

7.2 Collecting Semantics of the Abstract Interpreter

The semantics $\mathcal{S}[\![\mathbf{A}]\!]\, X^0 \in \mathcal{D}$ is defined as transfinite limit of a $\sqsubseteq$-increasing and convergent sequence $\langle X^k,\ k \in \mathbb{N} \cup \{\omega\}\rangle$ where each X^k is in $\mathcal{D}$. Because $\mathcal{C}[\![\mathbf{A}]\!]\, X^0 = \{\mathcal{S}[\![\mathbf{A}]\!]\, X^0\} \in \wp(\mathcal{D})$, we would like the collecting semantics $\mathcal{C}[\![\mathbf{A}]\!]\, X^0$ to be defined as a limit of an increasing and convergent sequence $\langle\{X^k\},\ k \in \mathbb{N} \cup \{\omega\}\rangle$ of iterates in $\wp(\mathcal{D})$. The problem is: for which order on $\wp(\mathcal{D})$? Of course, the approximation ordering $\subseteq$ does not work. Therefore, we define the following computational preorder relation $\widetilde{\sqsubseteq}$ on $\wp(\mathcal{D})$:

$$P \mathrel{\widetilde{\sqsubseteq}} Q \overset{\triangle}{\iff} \forall \sigma \in P \,.\, \exists \sigma' \in Q \,.\, \sigma \sqsubseteq \sigma'$$

It is standard to consider $\widetilde{\sqsubseteq}$ as partial order on the quotient $\wp(\mathcal{D})|_{\widetilde{\cong}}$ for the equivalence relation $P \mathrel{\widetilde{\cong}} Q \Leftrightarrow P \mathrel{\widetilde{\sqsubseteq}} Q \wedge Q \mathrel{\widetilde{\sqsubseteq}} P$. However, we do not need to consider this quotienting since we use $\widetilde{\sqsubseteq}$ only to compare iterates and $\widetilde{\sqsubseteq}$ is a partial (indeed total) order along these iterates. Moreover, as in the classical case of strictness analysis for functional programs [Cousot and Cousot 1993; Mycroft 1982], we have a computational preorder $\widetilde{\sqsubseteq}$ which is different from the approximation order $\subseteq$ (*i.e.*, logical implication).[2]

7.3 Projection Collecting Semantics of the Abstract Interpreter

In the following, we consider as semantics of the abstract interpreter the projection semantics defined in Section 5.4. All the semantics in Section 5 can be used similarly.

The collecting semantics of the projection semantics $\mathcal{S}_{\mathrm{pr}}[\![\mathbf{A}]\!]\, X^0$, for an initial n-dimensional vector $X^0 \in \prod_{i=1}^{n} D_i^0$, can be defined as follows.

$$\begin{array}{lclclcl}
\mathcal{P}_{\mathrm{pr}}(X^0) & \triangleq & \biguplus_{k=1}^{\omega+1} \wp\big((\prod_{i=1}^{n}(\mathbf{D}_i)^k)(X^0)\big) & \qquad & \mathcal{C}_{\mathrm{pr}}[\![\mathbf{A}]\!]\, X^0 & \triangleq & \{\mathcal{S}_{\mathrm{pr}}[\![\mathbf{A}]\!]\, X^0\} \\
\mathcal{T}_{\mathrm{pr}}(X^0) & \in & \mathcal{P}_{\mathrm{pr}}(X^0) \longrightarrow \mathcal{P}_{\mathrm{pr}}(X^0) & & \mathcal{T}_{\mathrm{pr}}(X^0)\, \widehat{X} & \triangleq & \{\mathcal{F}_{\mathrm{pr}}(X^0)\, \dot{X} \mid \dot{X} \in \widehat{X}\}
\end{array} \tag{15}$$

Let $\langle \mathcal{P}_{\mathrm{pr}}(X^0),\ \widetilde{\preccurlyeq}_{\mathrm{pr}}\rangle$ be the pre-order defined by $P \mathrel{\widetilde{\preccurlyeq}_{\mathrm{pr}}} Q \iff \forall \sigma \in P \,.\, \exists \sigma' \in Q \,.\, \sigma \mathrel{\dot{\preccurlyeq}_{\mathrm{pf}}} \sigma'$. Let $\varsigma : \mathcal{D}_{\mathrm{pr}}(X^0) \longrightarrow \wp(\mathcal{D}_{\mathrm{pr}}(X^0))$ denote the singleton embedding, *i.e.*, $\varsigma(\dot{X}) \triangleq \{\dot{X}\}$. Hence, we have that $\langle \mathcal{D}_{\mathrm{pr}}(X^0),\ \dot{\preccurlyeq}_{\mathrm{pf}}\rangle \xleftarrow[\varsigma]{\varsigma^{-1}} \langle \varsigma(\mathcal{D}_{\mathrm{pr}}(X^0)),\ \widetilde{\preccurlyeq}_{\mathrm{pr}}\rangle$ and $\langle \varsigma(\mathcal{D}_{\mathrm{pr}}(X^0)),\ \widetilde{\preccurlyeq}_{\mathrm{pr}},\ \dot{\sqcup}_{\mathrm{pr}}\rangle$ is a cpo.

Proposition 7.1. *The transfinite iterates* $\langle \mathcal{T}_{\mathrm{pr}}(X^0)^k\, \{X^0\},\ k \in \mathbb{N} \cup \{\omega, \omega{+}1\}\rangle$ *of* $\mathcal{T}_{\mathrm{pr}}(X^0)$ *from* $\{X^0\}$ *in the cpo* $\langle \varsigma(\mathcal{D}_{\mathrm{pr}}(X^0)),\ \widetilde{\preccurlyeq}_{\mathrm{pr}},\ \dot{\sqcup}_{\mathrm{pr}}\rangle$ *are* $\widetilde{\preccurlyeq}_{\mathrm{pr}}$*-increasing, ultimately stationary, and converge to the limit*

$$\mathcal{C}_{\mathrm{pr}}[\![\mathbf{A}]\!]\, X^0 \quad = \quad \widetilde{\dot{\bigsqcup}}_{\mathrm{pr}\ k \,\in\, \mathbb{N} \cup \{\omega, \omega+1\}} \quad \mathcal{T}_{\mathrm{pr}}(X^0)^k\, \{X^0\} \quad \in \quad \mathcal{P}_{\mathrm{pr}}(X^0) \tag{16}$$

which is a minimal fixpoint of $\mathcal{T}_{\mathrm{pr}}(X^0)$ *(but not necessarily the least one which may not exist).*

A result similar to (16) holds for the prefix and maximal trace semantics defined in Section 5.

7.3.1 *Projection Collecting Semantics of the Interval Abstract Interpreter.*

Continuing Example 5.4.2, we obtain

$$\widehat{X}^0 = \left\{\begin{bmatrix} \top \\ \bot \\ \bot \\ \bot \end{bmatrix}\right\} \quad \widetilde{\preccurlyeq}_{\mathrm{pr}} \quad \widehat{X}^1 = \left\{\begin{bmatrix} \top & \bullet & \top \\ \bot & \bullet & [0,0] \\ \bot & \bullet & [2,2] \\ \bot & \bullet & \bot \end{bmatrix}\right\} \quad \widetilde{\preccurlyeq}_{\mathrm{pr}} \quad \widehat{X}^2 = \left\{\begin{bmatrix} \top & \bullet & \top & \bullet & \top \\ \bot & \bullet & [0,0] & \bullet & [0,2] \\ \bot & \bullet & [2,2] & \bullet & [2,4] \\ \bot & \bullet & \bot & \bullet & \bot \end{bmatrix}\right\} \quad \widetilde{\preccurlyeq}_{\mathrm{pr}} \quad \ldots \quad \widetilde{\preccurlyeq}_{\mathrm{pr}}$$

$$\widehat{X}^{k+1} = \left\{\begin{bmatrix} \top & \bullet & \top & \bullet\bullet\bullet\bullet\bullet & \top \\ \bot & \bullet & [0,0] & \bullet\bullet\bullet\bullet\bullet & [0,2*k] \\ \bot & \bullet & [2,2] & \bullet\bullet\bullet\bullet\bullet & [2,2*(k+1)] \\ \bot & \bullet & \bot & \bullet\bullet\bullet\bullet\bullet & \bot \end{bmatrix}\right\} \widetilde{\preccurlyeq}_{\mathrm{pr}} \ldots \widetilde{\preccurlyeq}_{\mathrm{pr}}\ \widehat{X}^{\omega} = \left\{\begin{bmatrix} \top & \bullet & \top & \bullet\bullet\bullet\bullet\bullet & \top & \bullet\bullet\bullet\bullet \\ \bot & \bullet & [0,0] & \bullet\bullet\bullet\bullet\bullet & [0,2*k] & \bullet\bullet\bullet\bullet \\ \bot & \bullet & [2,2] & \bullet\bullet\bullet\bullet\bullet & [2,2*(k+1)] & \bullet\bullet\bullet\bullet \\ \bot & \bullet & \bot & \bullet\bullet\bullet\bullet\bullet & \bot & \bullet\bullet\bullet\bullet \end{bmatrix}\right\} \widetilde{\preccurlyeq}_{\mathrm{pr}}$$

$$\widehat{X}^{\omega+1} = \left\{\begin{bmatrix} \top & \bullet & \top & \bullet\bullet\bullet\bullet\bullet & \top & \bullet\bullet\bullet\bullet\bullet & \top \\ \bot & \bullet & [0,0] & \bullet\bullet\bullet\bullet\bullet & [0,2*k] & \bullet\bullet\bullet\bullet\bullet & [0,\infty] \\ \bot & \bullet & [2,2] & \bullet\bullet\bullet\bullet\bullet & [2,2*(k+1)] & \bullet\bullet\bullet\bullet\bullet & [2,\infty] \\ \bot & \bullet & \bot & \bullet\bullet\bullet\bullet\bullet & \bot & \bullet\bullet\bullet\bullet\bullet & \bot \end{bmatrix}\right\}.$$

[2] It is also standard to consider the lower/Hoare powerdomain [Abramsky and Jung 1994], that is the set of Scott down-closed subsets of $\mathcal{D}$ for the preorder $\sqsubseteq$ (that is, subsets $X \in \wp(\mathcal{D})$ which are down-closed, *i.e.* $X = \downarrow X$, and contain the lub of each of their $\sqsubseteq$-increasing ω-chains). The resulting ω-cpo ensures that continuous functions have a ω-limit of their iterates which is their least fixpoint. This is not applicable here because, as observed in Example 5.4.2, our functions may be noncontinuous and require iterates beyond ω.

8 OFFLINE A^{2}I

The meta-abstract interpreter for offline analysis of the abstract interpreter $\mathbf{A}$ is calculationally derived by an abstraction $\alpha(\mathcal{C}_{\mathrm{pr}}[\![\mathbf{A}]\!]\,X^0)$ of the projection collecting semantics defined in (16) of $\mathbf{A}$.

As illustrated by Example 7.3.1 for interval analysis, the abstract projection collecting semantics $\mathcal{C}_{\mathrm{pr}}[\![\mathbf{A}]\!]\,X^0$ defined in (16) collects at each program point i a singleton containing the sequence of abstract properties iteratively calculated at point i by the abstract interpreter (5) for the program under analysis (program (6) in our examples). In order to define the meta-abstract interpreter, we have therefore to provide an abstraction of (finite or infinite) sequences of abstract properties computed by the abstract interpreter.

8.1 Parametric Abstract Domain

To design the meta-abstract interpreter, we need an abstract domain $\langle\mathscr{D}_{\mathrm{pa}},\ \overline{\sqsubseteq},\ \overline{\bot},\ \overline{\sqcup}\rangle$ with arbitrary lubs $\overline{\sqcup}$ and an abstraction $\alpha_{\mathrm{pa}}^{k}\ \in\ D^k\ \longrightarrow\ \mathscr{D}_{\mathrm{pa}}$ for each abstract domain D^k, with $k\ \in\ \mathbb{N}\cup\{\omega\}$. This will be extended to an abstraction map $\dot{\alpha}_{\mathrm{pa}}\in\mathscr{P}_{\mathrm{pr}}(X^0)\longrightarrow\dot{\mathscr{D}}_{\mathrm{pa}}$, where $\mathscr{P}_{\mathrm{pr}}(X^0)=\biguplus_{k=1}^{\omega+1}\wp\big((\prod_{i=1}^{n}(\mathbf{D}_i)^k)(X^0)\big)$ is defined in (15) and $\dot{\mathscr{D}}_{\mathrm{pa}}\triangleq\prod_{i=1}^{n}\mathscr{D}_{\mathrm{pa}}$ so that $\langle\dot{\mathscr{D}}_{\mathrm{pa}},\ \dot{\overline{\sqsubseteq}},\ \dot{\overline{\bot}},\ \dot{\overline{\sqcup}}\rangle$ is the n-ary componentwise extension of the abstract domain $\mathscr{D}_{\mathrm{pa}}$. If $\dot{X}\in\prod_{i=1}^{n}(\mathbf{D}_i)^{+\omega+1}(X^0)$ then we write $\dot{X}_i^k\in D^k$ for the k-th element of the i-th component sequence $\dot{X}_i\in\mathbf{D}^{+\omega+1}(X^0)$ of $\dot{X}$. The abstraction $\alpha_{\mathrm{pa}}^{k}\in D^k\longrightarrow\mathscr{D}_{\mathrm{pa}}$ is extended to sequences by taking the lub along the whole sequence as follows:

$$\begin{array}{rcl}\alpha_{\mathrm{pa}} & \in & \mathbf{D}^{+\omega+1}(X^0)\longrightarrow\mathscr{D}_{\mathrm{pa}}\\ \alpha_{\mathrm{pa}}(X) & \triangleq & \displaystyle\overline{\bigsqcup_{k\in[0,|X|[}}\ \alpha_{\mathrm{pa}}^{k}(X^k)\end{array}$$

This abstraction map is then extended pointwise and homomorphically to sets of vectors of sequences by the function $\dot{\alpha}_{\mathrm{pa}}\in\mathscr{P}_{\mathrm{pr}}(X^0)\longrightarrow\dot{\mathscr{D}}_{\mathrm{pa}}$ as follows:

$$\dot{\alpha}_{\mathrm{pa}}(\widehat{P})\ \triangleq\ \textstyle\prod_{i=1}^{n}\overline{\bigsqcup}\big\{\alpha_{\mathrm{pa}}(\dot{X}_i)\ \big|\ \dot{X}\in\widehat{P}\big\}\ =\ \prod_{i=1}^{n}\overline{\bigsqcup}\big\{\displaystyle\overline{\bigsqcup_{k\in[0,|\dot{X}_i|[}}\ \alpha_{\mathrm{pa}}^{k}(\dot{X}_i^k)\ \big|\ \dot{X}\in\widehat{P}\big\}$$

Therefore, since we assume the existence of lubs $\overline{\sqcup}$ in $\mathscr{D}_{\mathrm{pa}}$, we have a Galois connection

$$\langle\mathscr{P}_{\mathrm{pr}}(X^0),\ \subseteq\rangle\ \underset{\dot{\alpha}_{\mathrm{pa}}}{\overset{\dot{\gamma}_{\mathrm{pa}}}{\leftrightarrows}}\ \langle\dot{\mathscr{D}}_{\mathrm{pa}},\ \dot{\overline{\sqsubseteq}}\rangle \tag{17}$$

8.2 Parametric Abstract Semantics

Consider meta-abstract functions $\overline{F}_i\in\mathscr{D}_{\mathrm{pa}}\longrightarrow\mathscr{D}_{\mathrm{pa}}$ for each program point $i\in[1,n]$, which are assumed to be monotone and such that:

$$\begin{array}{l}\forall k\in\mathbb{N}.\forall X\in\prod_{i=1}^{n}D^k.\ \neg C^k(X)\ \Longrightarrow\ \forall i\in[1,n].\ \alpha_{\mathrm{pa}}^{k+1}((F^{k+1}(X))_i)\ \overline{\sqsubseteq}\ \overline{F}_i(\alpha_{\mathrm{pa}}^{k}(X_i))\\ \forall\dot{X}\in\prod_{i=1}^{n}\mathbf{D}^{\omega}.\ \mathrm{wb}_{\mathrm{pr}}(\dot{X})\ \Longrightarrow\ \forall i\in[1,n].\ \alpha_{\mathrm{pa}}^{\omega}(F^{\omega}(\dot{X}_i))\ \overline{\sqsubseteq}\ \displaystyle\overline{\bigsqcup_{k\in\mathbb{N}}}\ \alpha_{\mathrm{pa}}^{k}(\dot{X}_i^k)\end{array}$$

The functions $\overline{F}_i$, $i\in[1,n]$, induce pointwise a meta-abstract function $\overline{F}\in\dot{\mathscr{D}}_{\mathrm{pa}}\longrightarrow\dot{\mathscr{D}}_{\mathrm{pa}}$ simply by defining $\overline{F}(\dot{X})\triangleq\prod_{i=1}^{n}\overline{F}_i(\dot{X}_i)$.

An over-approximation of $\mathcal{C}_{\mathrm{pa}}[\![\mathbf{A}]\!]\,X^0\ \triangleq\ \dot{\alpha}_{\mathrm{pa}}(\mathcal{C}_{\mathrm{pr}}[\![\mathbf{A}]\!]\,X^0)\ =\ \dot{\alpha}_{\mathrm{pa}}(\{\mathcal{S}_{\mathrm{pr}}[\![\mathbf{A}]\!]\,X^0\})$ can be obtained by iterating the meta-abstract transformer $\mathscr{T}_{\mathrm{pa}}\in\dot{\mathscr{D}}_{\mathrm{pa}}\longrightarrow\dot{\mathscr{D}}_{\mathrm{pa}}$ induced by $\overline{F}$ and defined as follows:

$$\mathscr{T}_{\mathrm{pa}}(\dot{X})\ \triangleq\ \dot{X}\ \dot{\overline{\sqcup}}\ \overline{F}(\dot{X})$$

Proposition 8.1. *The least fixpoint of $\mathscr{T}_{\mathrm{pa}}$ above $\dot{X}^0 \triangleq \prod_{i=1}^{n} \alpha_{\mathrm{pa}}^{0}(X_i^0) \in \dot{\mathscr{D}}_{\mathrm{pa}}$ is such that*

$$\mathscr{C}_{\mathrm{pa}}[\![\mathbf{A}]\!]\, X^0 \ \dot{\sqsubseteq}\ \mathsf{lfp}^{\dot{\sqsubseteq}}_{\dot{X}^0}\, \mathscr{T}_{\mathrm{pa}} \tag{18}$$

The iterates of the semantic transformer $\mathscr{T}_{\mathrm{pa}}$ in (18) can be calculated by an instance of the generic abstract interpreter **A** in (5) where the abstract domain is fixed to $\langle \dot{\mathscr{D}}_{\mathrm{pa}}, \dot{\sqsubseteq}, \dot{\bot}\rangle$, the abstract transformer is fixed to $\mathscr{T}_{\mathrm{pa}} \in \dot{\mathscr{D}}_{\mathrm{pa}} \longrightarrow \dot{\mathscr{D}}_{\mathrm{pa}}$, the termination condition is fixed to $\boldsymbol{\lambda} \dot{X} \bullet \mathscr{T}_{\mathrm{pa}}(\dot{X}) \ \dot{\sqsubseteq}\ \dot{X}$ and the initial abstract value is $\dot{\bot}$. This instance of **A** is called offline meta-abstract interpreter. Several possible choices for the abstract domain $\mathscr{D}_{\mathrm{pa}}$ are discussed below for designing widening operators for the interval abstract domain.

8.2.1 Example. Continuing Example 7.3.1, we obtain

$$\dot{X}^0 = \begin{bmatrix} \alpha_{\mathrm{pa}}^0(\top) \\ \alpha_{\mathrm{pa}}^0(\bot) \\ \alpha_{\mathrm{pa}}^0(\bot) \\ \alpha_{\mathrm{pa}}^0(\bot) \end{bmatrix} \quad \dot{\sqsubseteq} \quad \ldots \quad \dot{\sqsubseteq} \quad \dot{X}^{k+1} = \begin{bmatrix} \overline{\bigsqcup}\{\alpha_{\mathrm{pa}}^0(\top), \alpha_{\mathrm{pa}}^1(\top) \quad, \ldots, \alpha_{\mathrm{pa}}^{k+1}(\top)\} \\ \overline{\bigsqcup}\{\alpha_{\mathrm{pa}}^0(\bot), \alpha_{\mathrm{pa}}^1([0,0]), \ldots, \alpha_{\mathrm{pa}}^{k+1}([0, 2*k])\} \\ \overline{\bigsqcup}\{\alpha_{\mathrm{pa}}^0(\bot), \alpha_{\mathrm{pa}}^1([2,2]), \ldots, \alpha_{\mathrm{pa}}^{k+1}([2, 2*(k+1)])\} \\ \overline{\bigsqcup}\{\alpha_{\mathrm{pa}}^0(\bot), \alpha_{\mathrm{pa}}^1(\bot) \quad, \ldots, \alpha_{\mathrm{pa}}^{k+1}(\bot)\} \end{bmatrix} \quad \dot{\sqsubseteq} \quad \ldots \quad \dot{\sqsubseteq}$$

$$\dot{X}^{\omega+1} = \begin{bmatrix} \overline{\bigsqcup}\{\alpha_{\mathrm{pa}}^0(\top), \alpha_{\mathrm{pa}}^1(\top) \quad, \ldots, \alpha_{\mathrm{pa}}^{k+1}(\top) \quad, \ldots, \alpha_{\mathrm{pa}}^{\omega}(\top)\} \\ \overline{\bigsqcup}\{\alpha_{\mathrm{pa}}^0(\bot), \alpha_{\mathrm{pa}}^1([0,0]), \ldots, \alpha_{\mathrm{pa}}^{k+1}([0, 2*k]) \quad, \ldots, \alpha_{\mathrm{pa}}^{\omega}([0,\infty])\} \\ \overline{\bigsqcup}\{\alpha_{\mathrm{pa}}^0(\bot), \alpha_{\mathrm{pa}}^1([2,2]), \ldots, \alpha_{\mathrm{pa}}^{k+1}([2, 2*(k+1)]), \ldots, \alpha_{\mathrm{pa}}^{\omega}([2,\infty])\} \\ \overline{\bigsqcup}\{\alpha_{\mathrm{pa}}^0(\bot), \alpha_{\mathrm{pa}}^1(\bot) \quad, \ldots, \alpha_{\mathrm{pa}}^{k+1}(\bot) \quad, \cdots, \alpha_{\mathrm{pa}}^{\omega}(\bot)\} \end{bmatrix}.$$

9 DESIGN OF WIDENINGS AND ABSTRACT DOMAINS BY OFFLINE A^{2}I

A general guideline for designing a widening operator in an abstract domain whose values can be represented as a conjunction of constraints is to drop constraints which become unstable during abstract fixpoint computations. Applications of this principle can be found in widenings for intervals [Cousot and Cousot 1977a], octagons [Miné 2006] and polyhedra [Cousot and Halbwachs 1978]. In the context of A^{2}I, the general idea is to enforce convergence of a program analysis by detecting its instabilities through a meta-analysis.

9.1 Constancy Abstraction

The introductory example of Section 2 can be formulated as an offline meta-abstract interpretation. Instabilities in a sequence of intervals come from unstable lower or upper bounds of these intervals. This instability can be detected by exploiting Kildall's [1973] constancy abstract domain $\langle \mathscr{D}_{\mathrm{c}}, \sqsubseteq_{\mathrm{c}}, \bot_{\mathrm{c}}, \top_{\mathrm{c}}, \sqcup_{\mathrm{c}}, \sqcap_{\mathrm{c}}\rangle$, already recalled in Section 2. Here, the role of meta-abstract domain $\mathscr{D}_{\mathrm{pa}}$ of Section 8.1 is played by the product domain $\langle \mathscr{D}_{\mathrm{c}}^2, \overline{\sqsubseteq}\rangle$, whose elements are denoted by $\langle \ell, h\rangle$, while the meta-abstraction $\alpha_{\mathrm{pa}} \in \mathit{Int} \longrightarrow \mathscr{D}_{\mathrm{c}}^2$ (notice that since interval analysis always uses *Int* at each iteration k, the abstraction α_{pa} does not depend on an index k) is defined as follows:

$$\alpha_{\mathrm{pa}}(\bot) \triangleq \langle \bot,\ \bot\rangle, \qquad \alpha_{\mathrm{pa}}([\ell, h]) \triangleq \langle \ell,\ h\rangle, \qquad \alpha_{\mathrm{pa}}(\top) = \langle \top,\ \top\rangle.$$

The transformer $\overline{F}$ of Section 8.2 is defined to be the pairwise application of the standard constancy abstraction transformer of [Kildall 1973]. Therefore, an interval (lower or upper) bound which is not stable in the iterates of an interval analysis is meta-abstracted to $\top \in \mathscr{D}_{\mathrm{c}}$. Because the meta-abstract domain $\mathscr{D}_{\mathrm{c}}^2$ is of finite height, the meta-analysis always finitely converges.

Example 9.1. Given $\langle \ell,\ h\rangle \in \mathscr{D}_{\mathrm{c}}^2$ define $\langle \ell,\ h\rangle.\mathbf{1} = \ell$ and $\langle \ell,\ h\rangle.\mathbf{2} = h$. Continuing Example 5.4.2, the transformer $\overline{F}$ for the program P is calculationally designed (calculations are omitted due to lack of space) and provides the following system of equations for $\langle X_1, X_2, X_3, X_4\rangle \in (\mathscr{D}_{\mathrm{c}}^2)^4$:

$$\left\{\begin{array}{lcl} X_1 & = & \langle \top,\ \top \rangle \\ X_2 & = & \langle 0 \sqcup_{\mathrm{c}} \min(0, X_3.\mathbf{1}),\ 0 \sqcup_{\mathrm{c}} \max(0, X_3.\mathbf{2}) \rangle \\ X_3 & = & \langle (X_2.\mathbf{1} \sqcap_{\mathrm{c}} \top) \oplus^c 2,\ (X_2.\mathbf{2} \sqcap_{\mathrm{c}} \top) \oplus^c 2 \rangle \\ X_4 & = & \langle X_2.\mathbf{1} \sqcap_{\mathrm{c}} \bot,\ X_2.\mathbf{2} \sqcap_{\mathrm{c}} \bot \rangle \end{array}\right. \tag{19}$$

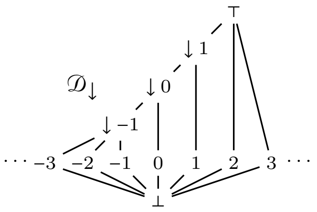
where $\top \oplus^c 2 = \top$, $\bot \oplus^c 2 = \bot$, $\min(0, \bot) = \max(0, \bot) = \bot$, and $\min(0, \top) = \max(0, \top) = \top$. The resolution of this system (19) provides the following finite sequence of Jacobi iterates $X^i \in (\mathcal{D}_{\mathrm{c}}^2)^4$:

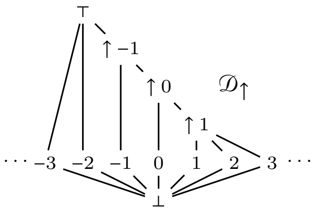
$$X^0 = \begin{bmatrix} \langle \bot,\ \bot \rangle \\ \langle \bot,\ \bot \rangle \\ \langle \bot,\ \bot \rangle \\ \langle \bot,\ \bot \rangle \end{bmatrix},\ X^1 = \begin{bmatrix} \langle \top,\ \top \rangle \\ \langle 0,\ 0 \rangle \\ \langle \bot,\ \bot \rangle \\ \langle \bot,\ \bot \rangle \end{bmatrix},\ X^2 = \begin{bmatrix} \langle \top,\ \top \rangle \\ \langle 0,\ 0 \rangle \\ \langle 2,\ 2 \rangle \\ \langle \bot,\ \bot \rangle \end{bmatrix},\ X^3 = \begin{bmatrix} \langle \top,\ \top \rangle \\ \langle 0,\ \top \rangle \\ \langle 2,\ 2 \rangle \\ \langle \bot,\ \bot \rangle \end{bmatrix},\ X^4 = \begin{bmatrix} \langle \top,\ \top \rangle \\ \langle 0,\ \top \rangle \\ \langle 2,\ \top \rangle \\ \langle \bot,\ \bot \rangle \end{bmatrix}$$

This tells us that in the interval analysis of the program P the interval upper bound of the variable x is possibly unstable at program points ℓ_2 and ℓ_3, while the lower bound is definitely stable (0 at program point ℓ_2 and 2 at program point ℓ_3). Hence, this suggests a widening of the interval upper bound to $+\infty$ for the variable x at program points ℓ_2 and ℓ_3 in the interval analysis of P by the analyser **A**. □

9.2 Threshold Abstraction

Widenings with thresholds (or "up to") [Cousot and Cousot 1992; Halbwachs et al. 1997] do not directly widen to infinity but to successive thresholds a priori determined. For example, these thresholds may be powers of 2 or 10, physical constants, constants syntactically found in the program, etc. To improve the meta-analysis described in Section 9.1, let us consider a widening that guarantees that interval analysis will always produce better results than sign analysis, *i.e.*, a widening with thresholds in $\{-1, 0, 1\}$. This widening can be designed through a meta-analysis which relies on the following lattices $\mathcal{D}_\downarrow$ and $\mathcal{D}_\uparrow$ for lower and upper bounds of intervals.





The meaning of abstract values are given by two concretization maps $\gamma_\downarrow \in \mathcal{D}_\downarrow \longrightarrow \wp(\mathbb{I})$ and $\gamma_\uparrow \in \mathcal{D}_\uparrow \longrightarrow \wp(\mathbb{I})$ defined as follows: $\gamma_\downarrow(\bot) \triangleq \emptyset$, $\gamma_\downarrow(z) \triangleq \{z\}$, $\gamma_\downarrow(\downarrow z) \triangleq \{z' \in \mathbb{I} \mid z' \leqslant z\}$ and $\gamma_\downarrow(\top) \triangleq \mathbb{I}$; $\gamma_\uparrow$ is dually defined, in particular $\gamma_\uparrow(\uparrow z) \triangleq \{z' \in \mathbb{I} \mid z' \geqslant z\}$.

The meta-abstract domain $\mathcal{D}_{\mathrm{pa}}$ here is the product $\langle \mathcal{D}_\downarrow \times \mathcal{D}_\uparrow, \overline{\sqsubseteq}, \overline{\bot}, \overline{\sqcup} \rangle$ while the abstraction $\alpha_{\mathrm{pa}} \in \mathit{Int} \longrightarrow \mathcal{D}_\downarrow \times \mathcal{D}_\uparrow$ is the same of Section 9.1: $\alpha_{\mathrm{pa}}(\bot) \triangleq \langle \bot,\ \bot \rangle$, $\alpha_{\mathrm{pa}}([\ell, h]) = \langle \ell,\ h \rangle$, and $\alpha_{\mathrm{pa}}(\top) = \langle \top,\ \top \rangle$. The transformer $\overline{F}$ of Section 8.2 is the pairwise application of the transformers for each abstract domain $\mathcal{D}_\downarrow$ and $\mathcal{D}_\uparrow$. These are calculationally designed in a similar way to the calculational design of the transformers for the interval abstract domain (cf. [Cousot 1999]). To give an example, the transformer $X \oplus^{\downarrow} 2 = \alpha_\downarrow(\{z + 2 \mid z \in \gamma_\downarrow(X)\})$ is designed by cases as follows:

- $\bot \oplus^{\downarrow} 2 = \alpha_\downarrow(\{z + 2 \mid z \in \gamma_\downarrow(\bot)\}) = \alpha_\downarrow(\emptyset) = \bot$;
- If $z \in \mathbb{I}$ then $z \oplus^{\downarrow} 2 = \alpha_\downarrow(\{z' + 2 \mid z' \in \gamma_\downarrow(z)\}) = \alpha_\downarrow(\{z + 2\}) = z + 2$;
- $\downarrow -1 \oplus^{\downarrow} 2 = \alpha_\downarrow(\{z + 2 \mid z \in \gamma_\downarrow(\downarrow -1)\}) = \alpha_\downarrow(\{z + 2 \mid z \leqslant -1\}) = \alpha_\downarrow(\{z \mid z \leqslant 1\}) = \downarrow 1$;

- If $i \in \{0, 1\}$ then $\downarrow i \oplus^{\downarrow} 2 = \alpha_{\downarrow}(\{z + 2 \mid z \in \gamma_{\downarrow}(\downarrow i)\}) = \alpha_{\downarrow}(\{z + 2 \mid z \leqslant i\}) = \alpha_{\downarrow}(\{z \mid z \leqslant i + 2\}) = \top$, since $i + 2 > 1$;
- $\top \oplus^{\downarrow} 2 = \alpha_{\downarrow}(\{z + 2 \mid z \in \gamma_{\downarrow}(\top)\}) = \alpha_{\downarrow}(\mathbb{N}) = \top$.

Because the threshold meta-abstract domain $\mathcal{D}_{\downarrow} \times \mathcal{D}_{\uparrow}$ has finite height, the meta-analysis always finitely converges. Therefore, an interval lower (resp. upper) bound which is not stable in the iterates of an interval analysis is meta-abstracted in the threshold abstract domain $\mathcal{D}_{\downarrow}$ (resp. $\mathcal{D}_{\uparrow}$) to the largest (resp. smallest) possible threshold.

Example 9.2. Continuing Example 9.1, the system of equations for $\langle X_1, X_2, X_3, X_4\rangle \in (\mathcal{D}_{\downarrow} \times \mathcal{D}_{\uparrow})^4$ has the same form of the equations in (19) but for the equation of X_3 which is replaced by:

$$X_3 = \langle (X_2.\mathbf{1} \sqcap \top) \oplus^{\downarrow} 2,\ (X_2.\mathbf{2} \sqcap \top) \oplus^{\uparrow} 2\rangle$$

The iterative resolution of this system provides the same finite sequence of Jacobi iterates of Example 9.1. This is coeherent with the fact that on the program P the thresholds $\{-1, 0, 1\}$ cannot be used for the widening. Instead, the threshold meta-analysis for the following program Q:

```
x=−3; while^ℓ (true) { if (x < 0) x=x+2; else x=0; }
```

would detect that in the interval analysis of Q the interval upper bound of the variable x is possibly unstable at program point ℓ but is bounded by 1. This meta-analysis suggests a widening to 1 instead of $+\infty$ for x at program point ℓ in the interval analysis of Q. □

9.3 On the Design of Widenings

Since there is hardly any systematic methodology to invent, in general, inductive arguments, there is hardly any systematic methodology to design widenings and the related extrapolation or interpolation operators [Cousot 2015]. Our approach of defining a meta-abstract interpreter which analyses the program abstract interpreter works necessarily per program, as shown in general by Cousot and Cousot [1992]. From our perspective, the design of a static program analysis is a conceptual inductive process which essentially consists in repeating this meta-analysis for all programs (*e.g.* to design a widening that is valid for all programs not only for a finite set of programs). Hankin and Hunt [1994] have proposed a method to design widenings by abstraction to a finite lattice (or more generally a Noetherian lattice). Cousot and Cousot [1992] showed that this is not as powerful as a general widening because it lacks the "ability to extrapolate to infinitely many distinct abstract values for all programs but to a finite number only for any given program". Instead, this is different for meta-abstract interpretation which is applied to a given program.

9.4 Offline A^{2}I for Designing Relational Abstract Domains

A relational abstract domain $\mathcal{D}_{\mathrm{r}}$ can be essentially viewed as an encoding of a subset of functions in $\mathcal{L} \longrightarrow \wp(\mathcal{R} \longrightarrow \mathcal{V})$ where $\mathcal{L}$ is a set of program labels/points, $\mathcal{R}$ is a set of references (*e.g.* program variables) and $\mathcal{V}$ is a set of values (more details are in Section 12.2). When the size of $\mathcal{R}$ (*e.g.* program variables) is very large, its polynomial (*e.g.* for octagons) or exponential (*e.g.* for polyhedra) cost may become prohibitive. A common solution is to consider a further abstraction to functions having a dependent type $\ell \in \mathcal{L} \longrightarrow \wp(\hat{\mathcal{R}}(\ell) \longrightarrow \mathcal{V})$ where $\hat{\mathcal{R}}(\ell) \subseteq \mathcal{R}$ is a subset, depending on the program point ℓ, of references for which one may assume that a relation between the variables in $\hat{\mathcal{R}}(\ell)$ at program point ℓ will be useful while a relation with the other variables in $\mathcal{R} \setminus \hat{\mathcal{R}}(\ell)$ is likely to be useless. How to define $\prod_{\ell \in \mathcal{L}} \hat{\mathcal{R}}(\ell)$ depends on the relational abstract domain $\mathcal{D}_{\mathrm{r}}$ and on the specific program, so that this process can be understood as an offline meta-analysis. An early example of this approach for designing relational domains is provided by the offline/static variable packing in the Astrée program analyser as described by [Blanchet et al. 2003, Section 7]. More refined offline meta-analyses for octagons are designed in [Heo et al. 2016, 2017; Lee et al. 2017; Oh et al. 2016] where $\hat{\mathscr{R}}(\ell)$ over-approximates the set of pairs of program variables x, y such that the octagon analysis is guaranteed to find a precise relation $\pm x \pm y \leqslant c$ with $c \neq +\infty$.

10 ONLINE A^2I

Offline abstract^2 interpretations are interesting but duplicate in part the work necessary to perform program analysis. But exactly the same idea can be applied online, that is, during the static analysis itself to determine some parameters which may improve the precision/cost ratio of that program analysis. This is called online (or dynamic) abstract^2 interpretation.

10.1 Online Abstract^2 Interpreter

The online abstract^2 interpreter $\mathbf{A^2}\llbracket\mathrm{P}\rrbracket$ of a program P is the generic abstract interpreter $\mathbf{A}\llbracket\mathrm{P}\rrbracket$ defined in (5) using the projection semantics of Section 5.4 and modified so that the next abstract domain D^{k+1}, the next transformer $F^{k+1} \in D^k \longrightarrow D^{k+1}$, and the next convergence criterion C^{k+1} for any iteration $k+1$, with $k \in \mathbb{N}$, are obtained through a meta-analysis $\mathbf{MA}\llbracket\mathrm{P}\rrbracket$ which takes as input the previous domain D^k, previous transformer F^k and previous convergence criterion C^k together with an abstraction $\overline{X}$ of the history of computation of $\mathbf{A^2}\llbracket\mathrm{P}\rrbracket$. The online abstract^2 interpreter $\mathbf{A^2}\llbracket\mathrm{P}\rrbracket$ depends on a parametric abstraction $\mathscr{D}_{\mathrm{pa}}$, defined through $\alpha_{\mathrm{pa}}/\gamma_{\mathrm{pa}}$ as in Section 8, of its history of computation: this abstraction $\overline{X}$ is initially $\alpha_{\mathrm{pa}}(X^0)$ and is then updated to $\alpha_{\mathrm{pa}}(\gamma_{\mathrm{pa}}(\overline{X}) \bullet F^{k+1}(X^k))$ at each iteration k. The standard abstract interpreter $\mathbf{A}\llbracket\mathrm{P}\rrbracket$ defined in (5) is obtained as a particular case of $\mathbf{A^2}\llbracket\mathrm{P}\rrbracket$ by making no meta-analysis and by skipping to record the abstraction $\overline{X}$ of its history of computation.

```
$\mathbf{MA}\llbracket\mathrm{P}\rrbracket(D, F, C, \gamma_{\mathrm{pa}}, \overline{X}) \triangleq$
  $\mathcal{X}$ := $\langle D,\ F,\ C,\ \gamma_{\mathrm{pa}}(\overline{X})\rangle$; k := 0;
  while ($\neg\mathcal{C}^{k}_{\mathrm{ma}}(\mathcal{X})$) {
    $\mathcal{X}$ := $\mathcal{F}^{k+1}_{\mathrm{ma}}(\mathcal{X})$; k := k+1;
  }
  let $\langle D,\ F,\ C,\ X\rangle = \mathcal{X}$ in
    return $\langle D,\ F,\ C\rangle$;
```

```
$\mathbf{A^2}\llbracket\mathrm{P}\rrbracket(D^0, D^1, F^1, C^0, X^0, \alpha_{\mathrm{pa}}, \gamma_{\mathrm{pa}}) \triangleq$                                   (20)
  $X$ := $X^0$; k := 0; $\overline{X}$ := $\alpha_{\mathrm{pa}}(X^0)$;
  while ($\neg C^k(X)$) {
    $X$ := $F^{k+1}(X)$; k := k + 1;
    $\overline{X}$ := $\alpha_{\mathrm{pa}}(\gamma_{\mathrm{pa}}(\overline{X}) \bullet X)$;
    $\langle D^{k+1},\ F^{k+1},\ C^{k}\rangle$ := $\mathbf{MA}\llbracket\mathrm{P}\rrbracket(D^k, F^k, C^{k-1}, \gamma_{\mathrm{pa}}, \overline{X})$;
  }
```

The soundness of the online abstract^2 interpreter $\mathbf{A^2}\llbracket\mathrm{P}\rrbracket$ follows directly from Proposition 6.1. The difference is that the hypotheses (6.a) to (6.e) on $\langle\langle D^k,\ F^k,\ C^k\rangle,\ k \in \mathbb{N}_+\rangle$ in Proposition 6.1, which are the assumptions for the soundness of the abstract interpreter $\mathbf{A}\llbracket\mathrm{P}\rrbracket$, must now be taken as a formal specification for the calls to the meta-abstract interpreter $\mathbf{MA}\llbracket\mathrm{P}\rrbracket(D^k, F^k, C^{k-1}, \gamma_{\mathrm{pa}}, \overline{X})$ defined in (20). It should be noted that in the pseudocode above this latter meta-abstract interpreter $\mathbf{MA}\llbracket\mathrm{P}\rrbracket$ is itself implemented as an instance of the generic interpreter $\mathbf{A}\llbracket\mathrm{P}\rrbracket$ where $\mathcal{C}^{k}_{\mathrm{ma}}$ are its convergence conditions and $\mathcal{F}^{k+1}_{\mathrm{ma}}$ its transformers. Thus, the proof of soundness of $\mathbf{MA}\llbracket\mathrm{P}\rrbracket$ essentially follows from the results of Section 6.

11 DESIGN OF WIDENING OPERATORS BY ONLINE A^2I

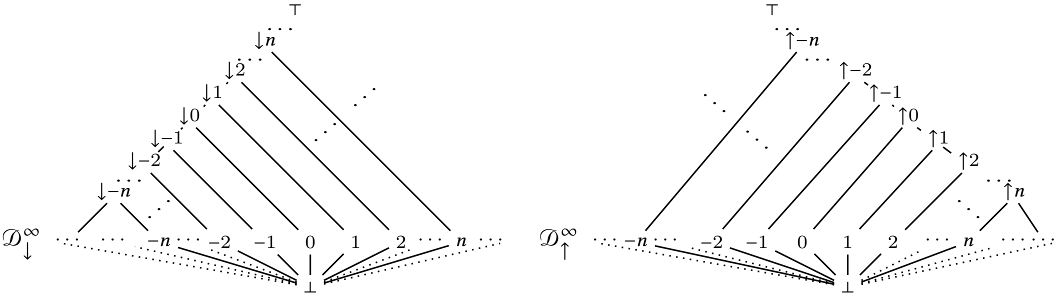
In general, a widening can be understood as an attempt to observe a program analysis in order to take a decision on how the analysis should go on. In our perspective, a widening can be therefore viewed and designed as instance of an online A^{2}I.

11.1 Widening Steps by Online A^2I

The constancy and threshold abstractions described in Sections 9.1 and 9.2 and used for designing a widening by an offline meta-analysis, can be also used by an online meta-analysis whose goal is to dynamically determine the behaviour of the widening steps. We assume this scenario: given an abstract domain D, transformer $F \in D \longrightarrow D$, and convergence criterion C, for any $k \in \mathbb{N}$, we have $D^k = D$, $C^k = C$ and $F^{k+1} = \boldsymbol{\lambda} X \bullet X \mathbin{\nabla} F(X)$ when an application of a widening ∇ is determined to be necessary by a meta-analysis, otherwise $F^{k+1} = F$. As noticed by Cousot [2015], in principle the widening can take all the previous iterates into account to make its extrapolation decision, *e.g.* a widening could depend on the increasing/decreasing rates of the previous iterates. It is worth noticing that an online choice of the widening steps will be more precise (as a widening operator) compared to an offline static determination of all the widening steps since the online approach allows to apply different widenings during the iterates of the abstract interpreter. In particular, termination enforcement is not required at the start of the iterates and can be postponed (*e.g.* within a given time bound) to improve the precision of the analysis.

11.2 Interval Widenings by Online Slope Abstraction

A problem with the threshold abstraction described in Section 9.2 is that it is not program dependent. An alternative is to consider a non-Noetherian meta-abstract domain given by the product of the following two non-Noetherian lattices $\mathcal{D}^\infty_\downarrow$ and $\mathcal{D}^\infty_\uparrow$ which extend, resp., $\mathcal{D}_\downarrow$ and $\mathcal{D}_\uparrow$ with an infinite set of "slopes" to determine the thresholds for the widening of the meta-analysis.


⊤
↓n
↓2
↓1
↓0
↓−1
↓−2
↓−n
$\mathcal{D}^\infty_\downarrow$
−n
−2
−1
0
1
2
n
⊥
⊤
↑−n
↑−2
↑−1
↑0
↑1
↑2
↑n
$\mathcal{D}^\infty_\uparrow$
−n
−2
−1
0
1
2
n
⊥

A slope function associates with each iteration step a corresponding threshold in $\mathcal{D}^\infty_\uparrow$ and $\mathcal{D}^\infty_\downarrow$ respectively for the lower and upper bound of intervals. Given a finite set of m natural numbers $0 \leq N_1 < \ldots < N_m$ which partitions $\mathbb{N}$ in $m+1$ intervals, and a corresponding slope function $t \in \mathbb{N}_+ \longrightarrow \mathbb{N}_+ \cup \{+\infty\}$ such that $i, j \in [1, m] \wedge i < j \implies t(i) < t(j)$ and $j \geq m \implies t(j) = +\infty$, the widening of an interval lower or upper bound x at iterate $k+1$ is obtained by an online meta-analysis as follows: $x \mathbin{\nabla^k} x' \triangleq x \pm t(k)|x' - x|$, where x' is the new value of the (lower for $-$, upper for $+$) bound x. Let us describe this online meta-analysis by a simple example.

Example 11.1. Consider the following program P:

$$\texttt{x=0; while}^{\ell}\ \texttt{(x <= 128) x=x+2;}$$

with slopes for the interval bounds defined by the function $t(n) \triangleq \llbracket n < 10 \mathrel{?} 2^n \mathrel{:} +\infty \rrbracket$. The interval analysis of P at program point ℓ is specified by the equation:

$$[a, b] = F([a, b]) \triangleq [0, 0] \sqcup (([a, b] \sqcap [-\infty, 128]) \oplus [2, 2]).$$

The widening ∇^k to be used between iterates $X^k = [a, b]$ and $X^{k+1} = F([a, b]) = [a', b']$ of the analysis at ℓ is determined by a meta-analysis of the iterates of the abstract interpreter as follows:

$$[a, b] \nabla^k [a', b'] \triangleq [a - t(k)|a' - a|, b + t(k)|b' - b|]$$

At any iteration of the analyser, the online meta-analysis determines the threshold to jump to in the analysis. The threshold is the result of a meta-analysis in the abstract domains $\mathcal{D}^\infty_\uparrow$ and $\mathcal{D}^\infty_\downarrow$.

The following table summarises the iterates (1st column) of the analyser (2nd column) and of the online meta-analysis (5th column) for P:

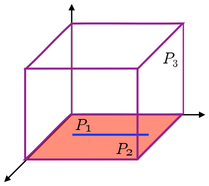
analysis iteration	interval analysis at ℓ : $[a^k, b^k] = F([a'^{k-1}, b'^{k-1}])$	widening to the result $[a'^k, b'^k]$ of the meta-analysis at ℓ	meta-analysis ∇^k of the lower bound a'^k at ℓ	meta-analysis ∇^k of the upper bound b'^k at ℓ
$k = 0$	$\bot$	$\bot$		
$k = 1$	$[0, 0]$	$[0, 0]$		
$k = 2$	$[0, 2]$	$[0, 8]$	$0 - t(2)(0 - 0) = 0$	$0 + t(2)(2 - 0) = 8$
$k = 3$	$[0, 10]$	$[0, 24]$	$0 - t(3)(0 - 0) = 0$	$8 + t(3)(10 - 8) = 24$
$k = 4$	$[0, 26]$	$[0, 56]$	$0 - t(4)(0 - 0) = 0$	$24 + t(4)(26 - 24) = 56$
$k = 5$	$[0, 58]$	$[0, 120]$	$0 - t(5)(0 - 0) = 0$	$56 + t(5)(58 - 56) = 120$
$k = 6$	$[0, 122]$	$[0, 248]$	$0 - t(6)(0 - 0) = 0$	$120 + t(6)(122 - 120) = 248$
$k = 7$	convergence of the interval analysis at ℓ to $[0, 250]$			

Instead of performing the widening ∇^k in the ranges N_k, the online meta-analysis allows to perform them less frequently at their frontiers only (which makes no difference for ranges of width 1 but is more precise in general). Lagrange, Newton-Gregory, Hermit, Taylor, Laurent, etc., polynomial extrapolation of the sequences of lower and upper bounds may also be useful. □

11.3 Improving the Precision of Polyhedra Widenings

The basic strategy for a polyhedra widening $P_1 \nabla P_2$ keeps all the constraints of P_1 satisfied by the system of generators of P_2 [Cousot and Halbwachs 1978]. There might be several different minimal systems of linear inequalities defining P_1 so that the widening depends upon the representation of P_1. The standard widening on polyhedra originally suggested by [Halbwachs 1979, Définition 5.3.3, p. 57] solves this problem by adding to P_1 the constraints of P_2 that can replace a constraint of P_1 without changing P_1.

Several improvements of the precision of this widening have been proposed based on suitable analyses of the iterates of a polyhedra analysis (see [Bagnara et al. 2005]) and can be viewed under the light of a meta-analysis. For example, the improvement suggested by Cousot and Cousot [1992] and adopted by Besson et al. [1999] is to use the lub of polyhedra instead of the widening when the iterates are along a chain of finite length. This always happens for a $\subseteq$-increasing chain of polyhedra whose dimensions are strictly increasing, as in the case of the polyhedra P_1, P_2 and P_3 depicted in the picture, whose dimensions are strictly increasing. Of course, dimensions are finite because they are bounded by the number of program variables, so that, using this strategy, the dimensions will ultimately stabilize and therefore a widening will become necessary.



12 DYNAMIC PARTITIONING OF RELATIONAL DOMAINS BY A^{2}I

In a series of papers, Halbwachs et al. [2006, 2003]; Singh et al. [2015, 2017, 2018] have shown that transformers of relational numerical domains, notably polyhedra, can be efficiently implemented by dynamically partitioning the program variables in disjoint blocks such that variables in different blocks of the partition are ensured to be unrelated. This dynamic decomposition of the analysis has been proved to be often much more efficient than the standard monolithic analysis despite the overhead cost due to maintaining the variable partition. We show that dynamic partitioning is an online meta-abstract interpretation in a more general context of (not necessarily numerical) relational abstract domains for static program analysis of invariance properties.

12.1 Collecting Semantics

Let $\mathcal{R}$ be a set of references (variables, memory addresses, array elements, program points, recursion stacks, etc.), $\mathcal{V}$ be a set of values and $\Sigma \triangleq \mathcal{R} \longrightarrow \mathcal{V}$ be the corresponding set of states, whose properties range in the complete lattice $\langle \mathcal{P} \triangleq \wp(\Sigma), \subseteq \rangle$. Let $\mathcal{F} \in \mathcal{P} \longrightarrow \mathcal{P}$ be a join preserving transformer for a given program P such that its collecting semantics of reachable states is $\mathsf{lfp}^{\subseteq} \mathcal{F}$. The collecting semantics $\mathsf{lfp}^{\subseteq} \mathcal{F}$ for P is formally calculated by the abstract interpreter $\mathbf{A}[\![\mathrm{P}]\!]$ in (5) with $D^k = \mathcal{P}$, $F^{k+1} = \mathcal{F}$ and $C^k(X) = \mathcal{F}(X) \subseteq X$, for all $k \in \mathbb{N}$, and $X^0 = \Sigma$. The semantics of this instance of the interpreter $\mathbf{A}[\![\mathrm{P}]\!]$ has $D^\omega = \mathcal{P}$ and F^ω is the lub $\bigcup$ of $\wp(\Sigma)$.

12.2 Relational Abstract Domains

Relational abstract domains $\mathcal{D}_{\mathsf{r}}$ rely on constraints ranging in a set $\mathcal{C}_{\mathsf{r}}$. Each constraint $C \in \mathcal{C}_{\mathsf{r}}$ refers to a finite number of references $\mathsf{ref}(C) \in \wp_{\mathsf{f}}(\mathcal{R})$ and typically provides some bounds to the possible values of these references. Hence, the concretization γ_{r} of a constraint $C \in \mathcal{C}_{\mathsf{r}}$ is a set of states for references ranging in $\mathsf{ref}(C)$, that is, $\gamma_{\mathsf{r}}(C) \in \wp(\mathsf{ref}(C) \longrightarrow \mathcal{V})$. We assume ref to be surjective, *i.e.*, $\forall R \in \wp_{\mathsf{f}}(\mathcal{R})\ .\ \exists C \in \mathcal{C}_{\mathsf{r}}\ .\ \mathsf{ref}(C) = R$.

A relational abstract property $P \in \mathcal{D}_{\mathsf{r}}$ is a finite set of constraints understood as a logical conjunction, so that $\mathcal{D}_{\mathsf{r}} \triangleq \wp_{\mathsf{f}}(\mathcal{C}_{\mathsf{r}})$ and $\gamma_{\mathsf{r}}(P) \triangleq \bigcap_{C \in P} \{\rho \in \mathcal{R} \longrightarrow \mathcal{V} \mid \rho \downarrow \mathsf{ref}(C) \in \gamma_{\mathsf{r}}(C)\} \in \mathcal{P}$, where $\rho \downarrow R$ denotes the restriction of the function ρ to a subset $R \subseteq \mathcal{R}$ of its domain and, by a slight abuse of notation, γ_{r} is used both for constraints and for abstract elements in $\mathcal{D}_{\mathsf{r}}$. Therefore, γ_{r} is an antitone function, meaning that the more constraints the less states satisfy them. For example, $\gamma_{\mathsf{r}}(w = x) = \{\rho \in \{w, x\} \longrightarrow \mathbb{R} \mid \rho(w) = \rho(x)\}$, $\gamma_{\mathsf{r}}(y \leqslant z) = \{\rho \in \{y, z\} \longrightarrow \mathbb{R} \mid \rho(y) \leqslant \rho(z)\}$, and $\gamma_{\mathsf{r}}(\{w = x, y \leqslant z\}) = \{\rho \in \{w, x, y, z\} \longrightarrow \mathbb{R} \mid \rho \downarrow \{w, x\} \in \gamma_{\mathsf{r}}(w = x),\ \rho \downarrow \{y, z\} \in \gamma_{\mathsf{r}}(y \leqslant z)\}$. Observe that $\langle \mathcal{D}_{\mathsf{r}}, \supseteq, \mathcal{C}_{\mathsf{r}}, \varnothing, \cap, \cup \rangle$ is a lattice but cannot be chosen as abstract domain since different predicates may have the same meaning, so that $\cap$ is not the abstract logical union. For example, $\gamma_{\mathsf{r}}(\{x \geqslant 0, x = y\}) = \gamma_{\mathsf{r}}(\{y \geqslant 0, x = y\})$ but $\gamma_{\mathsf{r}}(\{x \geqslant 0, x = y\} \cap \{y \geqslant 0, x = y\}) = \gamma_{\mathsf{r}}(\{x = y\})$ is a strict superset.

A relational abstract domain is a lattice $\langle \mathcal{D}_{\mathsf{r}}, =_{\mathsf{r}}, \sqsubseteq_{\mathsf{r}}, \bot_{\mathsf{r}}, \top_{\mathsf{r}}, \sqcup_{\mathsf{r}}, \sqcap_{\mathsf{r}} \rangle$, not necessarily complete (*e.g.* for polyhedra arbitrary lubs do not exist), where $=_{\mathsf{r}}$ is semantic equality, *i.e.* $P =_{\mathsf{r}} P'$ when $\gamma_{\mathsf{r}}(P) = \gamma_{\mathsf{r}}(P')$, and $\sqsubseteq_{\mathsf{r}}$ is abstract logical implication, *i.e.* $P \sqsubseteq_{\mathsf{r}} P'$ when $\gamma_{\mathsf{r}}(P) \subseteq \gamma_{\mathsf{r}}(P')$. In general, several different abstract predicates may have the same meaning so that $P \sqsubseteq_{\mathsf{r}} P' \wedge P' \sqsubseteq_{\mathsf{r}} P$ in general does not imply that $P = P'$. The concretization γ_{r} is assumed to preserve finite meets $\sqcap_{\mathsf{r}}$ but not necessarily infinite ones. Since constraints are interpreted conjunctively, we have that $P \supseteq P'$ implies $P \sqsubseteq_{\mathsf{r}} P'$ (but not necessarily the converse holds) and $\sqcap_{\mathsf{r}} = \cup$. We also assume that $\gamma_{\mathsf{r}}(\bot_{\mathsf{r}}) = \varnothing$, $\gamma_{\mathsf{r}}(\top_{\mathsf{r}}) = \Sigma$ (so $\top_{\mathsf{r}}$ can be represented by the empty set of constraints), and $\gamma_{\mathsf{r}}(P \cap P') \supseteq \gamma_{\mathsf{r}}(P \sqcup_{\mathsf{r}} P') \supseteq \gamma_{\mathsf{r}}(P) \cup \gamma_{\mathsf{r}}(P')$ so that $\sqcup_{\mathsf{r}}$ is an overapproximation of the union (*e.g.* convex hull for polyhedra).

12.3 Abstract Relational Semantics

Let $\mathcal{F}_{\mathsf{r}} \in \mathcal{D}_{\mathsf{r}} \longrightarrow \mathcal{D}_{\mathsf{r}}$ be a relational abstract transformer for a given program P, not necessarily monotone (*e.g.*, it can rely on nonmonotone widenings [Cousot 2015]), whose soundness condition is $\mathcal{F} \circ \gamma_{\mathsf{r}} \subseteq \gamma_{\mathsf{r}} \circ \mathcal{F}_{\mathsf{r}}$. The abstract relational semantics is formally calculated by the abstract interpreter $\mathbf{A}[\![\mathrm{P}]\!]$ in (5) with $D^k = \mathcal{D}_{\mathsf{r}}$, $F^{k+1} = \mathcal{F}_{\mathsf{r}}$, $C^k(X) = \mathcal{F}_{\mathsf{r}}(X) \sqsubseteq_{\mathsf{r}} X$, for all $k \in \mathbb{N}$, and $X^0 = \top_{\mathsf{r}}$. Here, limits of infinite sequences of abstract values might not exist (*e.g.* for an infinite sequence of polyhedra converging to a disk), hence the semantics of $\mathbf{A}[\![\mathrm{P}]\!]$ should simply record nontermination, for example with $D^\omega = \{\Omega\}$ and $F^\omega(\overline{X}) = \Omega$ for any infinite sequence $\overline{X}$ of abstract values which is not ultimately stationary. If the execution of the abstract interpreter $\mathbf{A}[\![\mathrm{P}]\!]$ terminates with some abstract value $P \in \mathcal{D}_{\mathsf{r}}$ then $\mathsf{lfp}^{\subseteq} \mathcal{F} \subseteq \gamma_{\mathsf{r}}(P)$ holds.

12.4 The Complete Lattice of Partitions

A partition of a set $\mathcal{R}$ is a set π of nonempty pairwise disjoint subsets of $\mathcal{R}$, called blocks, whose union is $\mathcal{R}$. Let $\mathrm{Part}(\mathcal{R})$ denotes the set of partitions of $\mathcal{R}$, where given $\pi \in \mathrm{Part}(\mathcal{R})$, for any $x \in \mathcal{R}$, $\pi(x)$ denotes the unique block of π containing x. Define $\pi \preceq \pi'$ when $\forall x \in \mathcal{R} \,.\, \pi(x) \subseteq \pi'(x)$ so that $\langle \mathrm{Part}(\mathcal{R}),\ \preceq,\ \{[x]\}_{x\in\mathcal{R}},\ \{\mathcal{R}\},\ \curlyvee,\ \curlywedge\rangle$ is a complete lattice where $\{[x]\}_{x\in\mathcal{R}}$ is the least (or finest) partition; $\{\mathcal{R}\}$ is the greatest (or coarsest) partition, and $\pi \curlywedge \pi'(x) = \pi(x) \cap \pi'(x)$ is the glb. The lub $\pi \curlyvee \pi'$ can be obtained by an iterative procedure (cf. [Grätzer 2011, Section V.4]).

12.5 Abstract Property Decomposition

In general, the abstract relational properties $\mathcal{D}_\mathrm{r}$ are costly to represent and compute with (*e.g*. exponential in the size of $\mathcal{R}$ for polyhedra). Halbwachs et al. [2006, 2003]; Singh et al. [2015, 2017, 2018] have shown that relational numerical domains can be efficiently implemented by decomposing relational properties according to a variable partition in such a way that variables in different blocks of the partition are unrelated. This can be extended to generic relational abstract domains.

Let $B \in \wp_\mathrm{f}(\mathcal{R})$ be a nonempty finite set of references and define $\mathcal{C}_\mathrm{r}{\downarrow}B \triangleq \{C \in \mathcal{C}_\mathrm{r} \mid \mathsf{ref}(C) \subseteq B\}$ to be the set of relational constraints with references ranging in B only, and, in turn, $\mathcal{D}_\mathrm{r}{\downarrow}B \triangleq \wp(\mathcal{C}_\mathrm{r}{\downarrow}B)$ to be the set of relational properties whose references are in B. $\langle \mathcal{D}_\mathrm{r} \downarrow B, =_\mathrm{r}, \sqsubseteq_\mathrm{r}, \bot_\mathrm{r}, \top_\mathrm{r}, \sqcup_\mathrm{r}, \sqcap_\mathrm{r}\rangle$ is a subset of $\langle \mathcal{D}_\mathrm{r},\ \sqsubseteq_\mathrm{r}\rangle$ and it is assumed to be a lattice. If necessary, the infimum $\bot_\mathrm{r}$ is added to $\mathcal{D}_\mathrm{r} \downarrow B$. Let us define the abstraction map $\alpha_B \in \mathcal{D}_\mathrm{r} \longrightarrow \mathcal{D}_\mathrm{r} \downarrow B$ by $\alpha_B(P) \triangleq P \cap (\mathcal{D}_\mathrm{r} \downarrow B)$, *i.e.*, $\alpha_B(P)$ is the subset of constraints of P whose references are all in B. This is extended componentwise for a given partition $\pi \in \mathrm{Part}(\mathcal{R})$ by defining $\alpha_\pi(P) \triangleq \prod_{B\in\pi} \alpha_B(P)$. Thus, $\alpha_\pi \in \mathcal{D}_\mathrm{r} \longrightarrow \mathcal{D}_\mathrm{r} \downarrow \pi$ where $\mathcal{D}_\mathrm{r}{\downarrow}\pi \triangleq \prod_{B\in\pi} \mathcal{D}_\mathrm{r}{\downarrow}B$. Hence, a π-partitioned set of constraints $\prod_{B\in\pi} P_B \in \mathcal{D}_\mathrm{r}{\downarrow}\pi$ is a tuple of sets of constraints which are indexed by blocks $B \in \pi$ and such that any set P_B of the tuple may contain constraints with variables ranging in the block B only. We consider $\sqsubseteq_\mathrm{r}^\pi$ to be the componentwise (*i.e.*, block by block) extension of $\sqsubseteq_\mathrm{r}$, so that $\langle \mathcal{D}_\mathrm{r} \downarrow \pi, =_\mathrm{r}^\pi, \sqsubseteq_\mathrm{r}^\pi, \bot_\mathrm{r}^\pi, \top_\mathrm{r}^\pi, \sqcup_\mathrm{r}^\pi, \sqcap_\mathrm{r}^\pi\rangle$ is a lattice.

The meaning of a partitioned abstract property in $\mathcal{D}_\mathrm{r} \downarrow \pi$ has to be understood conjunctively, namely, $\gamma_\pi \in \mathcal{D}_\mathrm{r} \downarrow \pi \longrightarrow \mathcal{D}_\mathrm{r}$ is defined by $\gamma_\pi(\prod_{B\in\pi} P_B) \triangleq \bigcup_{B\in\pi} P_B$. Hence, the concretization in $\wp(\mathcal{R} \longrightarrow \mathcal{V})$ of $\prod_{B\in\pi} P_B \in \mathcal{D}_\mathrm{r} \downarrow \pi$ is given by $\gamma_\mathrm{r}(\gamma_\pi(\prod_{B\in\pi} P_B)) = \gamma_\mathrm{r}(\bigcup_{B\in\pi} P_B) = \bigcap_{B\in\pi} \gamma_\mathrm{r}(P_B)$. This is an instance of a cofibred domain in the sense of Venet [1996], viewed as a map from blocks of a partition evolving during the analysis to constraints with references ranging in that block.

Observe that $\alpha_B \in \mathcal{D}_\mathrm{r} \longrightarrow \mathcal{D}_\mathrm{r} \downarrow B$ admits the adjoint map $\overline{\gamma}_B(P_B) \triangleq P_B \cup \neg(\mathcal{D}_\mathrm{r} \downarrow B)$, with $\neg(\mathcal{D}_\mathrm{r} \downarrow B) \triangleq \mathcal{D}_\mathrm{r} \setminus (\mathcal{D}_\mathrm{r} \downarrow B)$, hence we have a Galois insertion $\langle \mathcal{D}_\mathrm{r},\ \subseteq\rangle \underset{\alpha_B}{\overset{\overline{\gamma}_B}{\leftrightarrows}} \langle \mathcal{D}_\mathrm{r} \downarrow B,\ \subseteq\rangle$. In turn, this induces a Galois insertion $\langle \mathcal{D}_\mathrm{r},\ \subseteq\rangle \underset{\alpha_\pi}{\overset{\overline{\gamma}_\pi}{\leftrightarrows}} \langle \mathcal{D}_\mathrm{r} \downarrow \pi,\ \dot{\subseteq}\rangle$, where $\dot{\subseteq}$ is the blockwise inclusion. However, note that $\overline{\gamma}_\pi$ is not γ_π. For example, consider $\mathcal{R} = \{x, y\}$, $\pi = \{\{x\}, \{y\}\}$, $P = \{x = 0, x + y \leq 0\}$ and $\overline{P} = \langle\{x = 0\},\ \{y = 0\}\rangle \in \mathcal{D}_\mathrm{r} \downarrow \pi$. We have that $\alpha_\pi(P) = \langle\{x = 0\},\ \emptyset\rangle \mathrel{\dot{\subseteq}} \overline{P}$ whereas $P \not\subseteq \gamma_\pi(\overline{P}) = \{x = 0, y = 0\}$. On the other hand, $P \subseteq \gamma_\pi(\overline{P}) \implies \alpha_\pi(P) \mathrel{\dot{\subseteq}} \overline{P}$ always holds.

12.6 Exact Decompositions

Since $\alpha_\pi(P)$ removes from $P \in \mathcal{D}_\mathrm{r}$ the constraints whose variables are not all in some block of π, in general a partition π is not exact for P. For example, if $\pi = \{\{x, y\}, \{z, w\}\}$ then $\gamma_\pi(\alpha_\pi(\{x = w, x = y, y \leqslant z\})) = \gamma_\pi(\langle\{x = y\}, \emptyset\rangle) = \{x = y\} \subsetneq \{x = w, x = y, y \leqslant z\}$. We are interested in partitions π of the variables occurring in $P \in \mathcal{D}_\mathrm{r}$ such that $\gamma_\pi(\alpha_\pi(P)) = P$. These partitions are called *exact*[3], while π is called *approximate* for P when $\gamma_\pi(\alpha_\pi(P)) \subsetneq P$. For example,

[3] In [Singh et al. 2017] π is called *permissible* for P, while in [Halbwachs et al. 2006] P is called *factorizable* according to π.

$\pi = \{\{w,x\},\{y,z\}\}$ is exact for $P = \{w = x, y \leqslant z\}$ and $\alpha_\pi(P) = \langle \{w,x\} \mapsto \{w = x\}, \{y,z\} \mapsto \{y \leqslant z\}\rangle$. On the other hand, the partition $\pi' = \{\{w\},\{x,y,z\}\}$ is approximate for P since the decomposition yields $\alpha_{\pi'}(P) = \langle \{w\} \mapsto \varnothing, \{x,y,z\} \mapsto \{y \leqslant z\}\rangle$. In general, observe that the coarsest partition $\{\mathcal{R}\}$ is always exact for all $P \in \mathcal{D}_\mathsf{r}$. The decomposition $\alpha_\pi(P)$ of an abstract relational property $P \in \mathcal{D}_\mathsf{r}$ by means of an exact partition π of $\mathcal{R}$ does not lose any information with respect to P since $\gamma_\mathsf{r}(\gamma_\pi(\alpha_\pi(P))) = \gamma_\mathsf{r}(P)$. Otherwise, it should be remarked that $\alpha_\pi(P)$ may be an over-approximation of P since $P \supseteq \gamma_\pi(\alpha_\pi(P))$ so that $\gamma_\mathsf{r}(P) \subseteq \gamma_\mathsf{r}(\gamma_\pi(\alpha_\pi(P)))$ holds. We have the following characterization of an exact partition $\pi \in \mathrm{Part}(\mathcal{R})$ for a relational property $P \in \mathcal{D}_\mathsf{r}$: $\gamma_\pi(\alpha_\pi(P)) = P \iff \forall C \in P\,.\,\exists B \in \pi\,.\,\mathsf{ref}(C) \subseteq B$.

12.7 Partition Abstraction

A partition is a property of a set of constraints in $\mathcal{D}_\mathsf{r}$ so it abstracts properties in $\wp(\mathcal{D}_\mathsf{r})$. Let us first define $\alpha_{\mathrm{pt}} \in \mathcal{C}_\mathsf{r} \longrightarrow \mathrm{Part}(\mathcal{R})$ by $\alpha_{\mathrm{pt}}(C) \triangleq \{\mathsf{ref}(C)\} \cup \{\{x\} \mid x \in \mathcal{R} \setminus \mathsf{ref}(C)\}$, and in turn $\alpha_{\mathrm{pt}} \in \mathcal{D}_\mathsf{r} \longrightarrow \mathrm{Part}(\mathcal{R})$ by $\alpha_{\mathrm{pt}}(P) \triangleq \curlyvee\{\alpha_{\mathrm{pt}}(C) \mid C \in P\}$. It turns out that if $P \in \mathcal{D}_\mathsf{r}$ then $\forall C \in P\,.\,\exists B \in \alpha_{\mathrm{pt}}(P)\,.\,\mathsf{ref}(C) \subseteq B$ so that the partition $\alpha_{\mathrm{pt}}(P)$ is always exact for the relational property P and, furthermore, $\alpha_{\mathrm{pt}}(P)$ is the finest exact partition for P.[4]

12.8 Sound Transformer Decomposition

Assume that a concrete transformer $\mathcal{F} \in \mathcal{P} \longrightarrow \mathcal{P}$ is approximated by an abstract relational transformer $\mathcal{F}_\mathsf{r} \in \mathcal{D}_\mathsf{r} \longrightarrow \mathcal{D}_\mathsf{r}$ which is sound $\mathcal{F} \circ \gamma_\mathsf{r} \mathrel{\dot{\subseteq}} \gamma_\mathsf{r} \circ \mathcal{F}_\mathsf{r}$.

When calculating with partitioned relational properties $\prod_{B\in\pi} P_B$ for a current partition $\pi \in \mathrm{Part}(\mathcal{R})$, we also need a partitioned abstract transformer $[\![\mathcal{F}_\mathsf{r}]\!]_{\pi\longrightarrow\pi'} \in \mathcal{D}_\mathsf{r}\downarrow\pi \longrightarrow \mathcal{D}_\mathsf{r}\downarrow\pi'$, where π' is the new partition, which is a sound approximation of $\mathcal{F}_\mathsf{r}$, meaning that $\gamma_\mathsf{r} \circ \mathcal{F}_\mathsf{r} \circ \gamma_\pi \mathrel{\dot{\subseteq}} \gamma_\mathsf{r} \circ \gamma_{\pi'} \circ [\![\mathcal{F}_\mathsf{r}]\!]_{\pi\longrightarrow\pi'}$. By soundness of $\mathcal{F}_\mathsf{r}$, this implies soundness with respect to the concrete transformer $\mathcal{F}$, namely, $\mathcal{F} \circ \gamma_\mathsf{r} \circ \gamma_\pi \mathrel{\dot{\subseteq}} \gamma_\mathsf{r} \circ \gamma_{\pi'} \circ [\![\mathcal{F}_\mathsf{r}]\!]_{\pi\longrightarrow\pi'}$. Moreover, the new partition π' should be obtained as a function of the old one, namely $\pi' = \mathcal{F}_\mathsf{p}[\![\mathcal{F}_\mathsf{r}]\!]\pi$.

It is preferable to define $[\![\mathcal{F}_\mathsf{r}]\!]_{\pi\longrightarrow\pi'} \in \mathcal{D}_\mathsf{r}\downarrow\pi \longrightarrow \mathcal{D}_\mathsf{r}\downarrow\pi'$ "blockwise" as follows:

$$[\![\mathcal{F}_\mathsf{r}]\!]_{\pi\longrightarrow\pi'}(\textstyle\prod_{B\in\pi} P_B) \quad\triangleq\quad \prod_{B'\in\pi'} \bigsqcap_{\mathsf{r}\,B\in\pi} [\![\mathcal{F}_\mathsf{r}]\!]_{B\longrightarrow B'}(P_B) \tag{21}$$

for some suitable transformers $[\![\mathcal{F}_\mathsf{r}]\!]_{B\longrightarrow B'} \in \mathcal{D}_\mathsf{r}\downarrow B \longrightarrow \mathcal{D}_\mathsf{r}\downarrow B'$ depending on a pair $\langle B, B'\rangle \in \pi\times\pi'$ of blocks. If possible, $[\![\mathcal{F}_\mathsf{r}]\!]_{B\longrightarrow B'}$ should be defined by reusing $\mathcal{F}_\mathsf{r}$ on the blocks B and B' of the partitions π and π'. A sufficient condition for the soundness $\mathcal{F} \circ \gamma_\mathsf{r} \circ \gamma_\pi \mathrel{\dot{\subseteq}} \gamma_\mathsf{r} \circ \gamma_{\pi'} \circ [\![\mathcal{F}_\mathsf{r}]\!]_{\pi\longrightarrow\pi'}$ of $[\![\mathcal{F}_\mathsf{r}]\!]_{\pi\longrightarrow\pi'}$ with respect to the collecting semantics $\mathcal{F}$ is:

$$\forall B \in \pi\,.\,\mathcal{F}_\mathsf{r}(P_B) \subseteq \textstyle\bigcup_{B'\in\pi'} [\![\mathcal{F}_\mathsf{r}]\!]_{B\longrightarrow B'}(P_B) \tag{22}$$

Also, soundness can be strengthened to precision by replacing $\subseteq$ with $=$. If soundness (precision) holds then the computation X^0, ..., $X^{k+1} = \mathcal{F}_\mathsf{r}(X^k)$, ..., of the abstract interpreter for $\mathcal{F}_\mathsf{r}$ can be soundly (precisely) replaced by a decomposed computation $Y^k = \langle \pi^k,\ \overline{P}^k\rangle \in \mathrm{Part}(\mathcal{R}) \times \mathcal{D}_\mathsf{r}\downarrow\pi^k$ defined on pairs as follows:

$$\begin{aligned}
&Y^0 \triangleq \langle \pi^0 = \{\mathcal{R}\},\ \overline{P}^0 = \alpha_{\pi^0}(X^0)\rangle;\\
&Y^{k+1} \triangleq \langle \pi^{k+1} = \mathcal{F}_\mathsf{p}[\![\mathcal{F}_\mathsf{r}]\!]\pi^k,\ \overline{P}^{k+1} = [\![\mathcal{F}_\mathsf{r}]\!]_{\pi^k\longrightarrow\pi^{k+1}}(\overline{P}^k)\rangle
\end{aligned}$$

Let us remark that the next partition is computed as a function of the current abstract value(s) and of the current partition(s), as allowed by the general template of the meta-interpreter $\mathbf{MA}[\![\mathrm{P}]\!]$ given

[4] $\alpha_{\mathrm{pt}}(P)$ is called the finest permissible partition for P in [Singh et al. 2017] and the greatest common partition for P in [Halbwachs et al. 2006].

in (20). Of course, this general technique leaves the problem of computing the next partitions π^{k+1} and the corresponding blockwise transformers $\llbracket \mathcal{F}_\mathrm{r} \rrbracket_{B \longrightarrow B'}$ for these partitions. One naïve solution would be to choose the next partition to be $\pi^{k+1} = \alpha_\mathrm{pt} \circ \mathcal{F}_\mathrm{r} \circ \gamma_{\pi^k}(\overline{P}^k)$ but, of course, we do not want to use the transformer $\mathcal{F}_\mathrm{r}$ to compute the decomposition of this transformer. The solution is a meta-abstract interpretation.

12.9 Partition Meta-Analysis and Analysis

Given a pre-partition π which decomposes a pre- partitioned property $\prod_{B \in \pi} P_B$, the meta-analysis must compute a post-partition π' which approximates $\alpha_\mathrm{pt} \circ \mathcal{F}_\mathrm{r} \circ \gamma_\pi(\prod_{B \in \pi} P_B)$. We therefore design a sound transformer $\mathcal{F}_\mathrm{p}\llbracket \mathcal{F}_\mathrm{r} \rrbracket \mathrel{\dot{\succeq}} \alpha_\mathrm{pt} \circ \mathcal{F}_\mathrm{r} \circ \gamma_\pi$, that is, $\mathcal{F}_\mathrm{p}\llbracket \mathcal{F}_\mathrm{r} \rrbracket(\pi) \succeq \bigcurlyvee \{\alpha_\mathrm{pt}(\mathcal{F}_\mathrm{r}(P)) \mid P \sqsubseteq_\mathrm{r} \gamma_\pi(\prod_{B \in \pi} P_B)\}$, by calculational design. Since the concrete transformer $\mathcal{F}_\mathrm{r}$ is defined for primitive statements and then applied inductively, we also inductively define the transformers $\mathcal{F}_\mathrm{p}\llbracket \mathcal{F}_\mathrm{r} \rrbracket$ and $\llbracket \mathcal{F}_\mathrm{r} \rrbracket_{B \longrightarrow B'}$. Then, the partitioned relational analysis $\llbracket \mathcal{F}_\mathrm{r} \rrbracket_{\pi \longrightarrow \pi'}$ can be based on $\llbracket \mathcal{F}_\mathrm{r} \rrbracket_{B \longrightarrow B'}$, where $B \in \pi$ and $B' \in \pi' \triangleq \mathcal{F}_\mathrm{p}\llbracket \mathcal{F}_\mathrm{r} \rrbracket(\pi)$, as shown in (21).

12.9.1 Skip. Consider the concrete transformer $\mathcal{F}\llbracket \mathtt{skip} \rrbracket \in \mathscr{P} \longrightarrow \mathscr{P}$ such that $\mathcal{F}\llbracket \mathtt{skip} \rrbracket(X) = X$. Then, the relational transformer $\mathcal{F}_\mathrm{r}\llbracket \mathtt{skip} \rrbracket \in \mathcal{D}_\mathrm{r} \longrightarrow \mathcal{D}_\mathrm{r}$ is $\mathcal{F}_\mathrm{r}\llbracket \mathtt{skip} \rrbracket(P) \triangleq P$, which is obviously sound. In this case, no update to a partition π is needed, hence $\mathcal{F}_\mathrm{p}\llbracket \mathtt{skip} \rrbracket\pi \triangleq \pi$ and $\llbracket \mathtt{skip} \rrbracket_{\pi \longrightarrow \pi}(P) \triangleq P$.

For later use in Section 12.9.9, we can also define a partitioned transformer for a possibly coarser post-partition $\pi' \succeq \pi$ by defining the blockwise transformer $\llbracket \mathtt{skip} \rrbracket_{B \longrightarrow B'}(P_B) \triangleq \llparenthesis\, B \subseteq B' \mathrel{?} P_B \mathrel{:} \emptyset \,\rrparenthesis$ and the partitioned relational transformer $\llbracket \mathtt{skip} \rrbracket_{\pi \longrightarrow \pi'}$ as given by (21) in Section 12.8.

12.9.2 Inclusion and Emptiness Test. The relational domain $\mathcal{D}_\mathrm{r}$ implements the test $P \sqsubseteq_\mathrm{r} P'$. Let us consider two partitioned relations $\prod_{B \in \pi} P_B$ and $\prod_{B' \in \pi'} P'_{B'}$. We first compute the greatest common partition $\pi \curlyvee \pi'$ and then we compare blockwise as follows:

$$\prod_{B \in \pi} P_B \sqsubseteq_\mathrm{r} \prod_{B' \in \pi'} P'_{B'} \;\triangleq\; \bigwedge_{B \in \pi \curlyvee \pi'} \big(\bigcup_{B_1 \in \pi, B_1 \subseteq B} P_{B_1}\big) \sqsubseteq_\mathrm{r} \big(\bigcup_{B_2 \in \pi', B_2 \subseteq B} P'_{B_2}\big)$$

Domain-specific refinements are given in [Halbwachs et al. 2003; Singh et al. 2017, 2018] which may be generalizable depending on the specific domain of constraints $\mathcal{C}_\mathrm{r}$. The emptiness test $P \sqsubseteq_\mathrm{r} \bot_\mathrm{r}$ is handled analogously.

12.9.3 Meet. If π_1, π_2 are, resp., (finest) exact partitions for P_1 and P_2 then $\pi_1 \curlyvee \pi_2$ is the (finest) exact partition for their meet $P_1 \sqcap_\mathrm{r} P_2$ since $\sqcap_\mathrm{r}$ is the union of sets of constraints and α_p preserves unions.

$\sqcap_\mathrm{r}$ is a binary transformer, so that the abstract transformer is $\llbracket \sqcap_\mathrm{r} \rrbracket_{\pi_1 \times \pi_2 \longrightarrow \pi_1 \curlyvee \pi_2}$. This is given by the blockwise meet $\dot{\bigsqcap}_\mathrm{r} \in (\mathcal{D}_\mathrm{r} \downarrow \pi_1 \times \mathcal{D}_\mathrm{r} \downarrow \pi_2) \longrightarrow \mathcal{D}_\mathrm{r} \downarrow \pi_1 \curlyvee \pi_2$ defined as follows (where $\sqcap_\mathrm{r} \emptyset = \top_\mathrm{r}$):

$$\dot{\bigsqcap}_\mathrm{r}\big(\prod_{B_1 \in \pi_1} P_{B_1}, \prod_{B_2 \in \pi_2} P_{B_2}\big) \;\triangleq\; \prod_{B' \in \pi_1 \curlyvee \pi_2} \sqcap_\mathrm{r}\{P_{B_1} \sqcap_\mathrm{r} P_{B_2} \mid \langle B_1, B_2 \rangle \in \pi_1 \times \pi_2, B_1 \cup B_2 \subseteq B'\}$$

12.9.4 Conditional. Let "`if` C `then  stm`" be a branching statement with a relational condition $C \in \mathcal{D}_\mathrm{r}$. If P a relational pre-condition then the relational post-condition on the entry of the then branch is $\mathcal{F}_\mathrm{r}\llbracket \mathtt{if}\ C\ \mathtt{then} \rrbracket(P) \triangleq P \sqcap_\mathrm{r} C$.

Let π' be an exact partition for the condition C, *i.e.*, $\gamma_{\pi'}(\alpha_{\pi'}(C)) = C$. The partition for the post-condition is then $\pi'' = \mathcal{F}_\mathrm{p}\llbracket \mathtt{if}\ C\ \mathtt{then} \rrbracket(\pi) \triangleq \pi \curlyvee \pi'$. The partitioned relational transformer is $\llbracket \mathtt{if}\ C\ \mathtt{then} \rrbracket_{\pi \longrightarrow \pi''}(\prod_{B \in \pi} P_B) \triangleq (\prod_{B \in \pi} P_B) \mathbin{\dot{\sqcap}_\mathrm{r}} \alpha_{\pi'}(C)$.

12.9.5 Substitution. If $P \in \mathscr{P}$ and $x, x' \in \mathscr{R}$ then $\mathscr{F}\llbracket[x := x']\rrbracket \in \mathscr{P} \longrightarrow \mathscr{P}$ denotes the substitution of a fresh reference x' for x in P, that is, $\mathscr{F}\llbracket[x := x']\rrbracket P \triangleq \{\rho \in (\mathscr{R} \setminus \{x\}) \cup \{x'\} \longrightarrow \mathscr{V} \mid \exists \rho' \in P \,.\, \forall y \in \mathscr{R} \setminus \{x\} \,.\, \rho(y) = \rho'(y) \wedge \rho(x') = \rho'(x)\}$.

Substitution in relational properties $P \in \mathscr{D}_{\mathsf{r}}$ is defined as $\mathscr{F}_{\mathsf{r}}\llbracket[x := x']\rrbracket(P) \triangleq P[x := x'] \triangleq \{C[x := x'] \mid C \in P\}$, where $C[x := x']$ is a given substitution primitive for constraints, which is assumed to be sound: $\gamma_{\mathsf{r}}(C[x := x']) \supseteq \mathscr{F}\llbracket[x := x']\rrbracket\gamma_{\mathsf{r}}(C)$. If $x \notin \mathsf{ref}(C)$ is not bound in C then its substitution is assumed to have no effect, that is, $C[x := x'] = C$. As a consequence, substitution for relational properties $P \in \mathscr{D}_{\mathsf{r}}$ is sound: $\gamma_{\mathsf{r}}(P[x := x']) \supseteq \mathscr{F}\llbracket[x := x']\rrbracket\gamma_{\mathsf{r}}(P)$.

If π is an exact partition for P then $\mathscr{F}_{\mathsf{p}}\llbracket[x := x']\rrbracket\pi \triangleq \{\llparenthesis\, x \in B \mathrel{⸮} (B \setminus \{x\}) \cup \{x'\} \mathrel{⁏} B \,\rrparenthesis \mid B \in \pi\}$ provides an exact partition for $P[x := x']$. Given $B \in \pi$ and $B' \in \mathscr{F}_{\mathsf{p}}\llbracket[x := x']\rrbracket\pi$, the block relational transformer is defined by $\llbracket[x := x']\rrbracket_{B \longrightarrow B'}(P_B) \triangleq \llparenthesis\, x \in B \mathrel{⸮} P_B[x := x'] \mathrel{⁏} P_B \,\rrparenthesis$. In turn, the partitioned relational transformer $\llbracket[x := x']\rrbracket_{\pi \longrightarrow \pi'}$ is given blockwise by (21) in Section 12.8.

12.9.6 Variable Elimination. If $P \in \mathscr{P}$ and $x \in \mathscr{R}$ then $\mathscr{F}\llbracket\exists x\rrbracket \in \mathscr{P} \longrightarrow \mathscr{P}$ is the elimination (or existential quantification) of x in P defined by $\mathscr{F}\llbracket\exists x\rrbracket P \triangleq \{\rho \in (\mathscr{R} \setminus \{x\}) \longrightarrow \mathscr{V} \mid \exists \rho' \in P \,.\, \forall y \in (\mathscr{R} \setminus \{x\}) \,.\, \rho(y) = \rho'(y)\}$.

We assume that $\mathscr{D}_{\mathsf{r}}$ provides a sound elimination for relational properties $\mathscr{F}_{\mathsf{r}}\llbracket\exists x\rrbracket(P) \triangleq \exists x.\, P$ (*e.g.* Fourier-Motzkin elimination for polyhedra), that is, $\gamma_{\mathsf{r}}(\exists x.\, P) \supseteq \mathscr{F}\llbracket\exists x\rrbracket\gamma_{\mathsf{r}}(P)$. In particular, we assume that if $x \notin \mathsf{ref}(P)$ then $\exists x.\, P = P$.

If π is an exact partition for P then $\mathscr{F}_{\mathsf{p}}\llbracket\exists x \,.\, P\rrbracket\pi \triangleq \{B \setminus \{x\} \mid B \in \pi\} \cup \{\{x\}\}$ is an exact partition for $\exists x.P$. Since x belongs to the block $\pi(x)$ the analysis uses the relational elimination for that block, leaves the other blocks unchanged, and the block $\{x\}$ is assigned the abstract property $\top_{\mathsf{r}}$. Thus, given $B \in \pi$ and $B' \in \mathscr{F}_{\mathsf{p}}\llbracket\exists x\rrbracket\pi$, the block relational transformer is $\llbracket\exists x\rrbracket_{B \longrightarrow B'}(P_B) \triangleq \llparenthesis\, B' = \{x\} \mathrel{⸮} \top_{\mathsf{r}} \mathrel{⁏} \exists x.\, P_B \,\rrparenthesis$. The partitioned relational transformer $\llbracket\exists x\rrbracket_{\pi \longrightarrow \pi'}$ is then given by (21) in Section 12.8.

12.9.7 Assignment. Let $\mathrm{x} := E$ be an assignment. We assume that there is a constraint $C\llbracket\mathrm{x} := E\rrbracket \in \mathscr{C}_{\mathsf{r}}$ with references in $\mathsf{ref}(E) \cup \{x', x\}$ relating the values of the references of E and the value x' of x before the assignment to the value x of x after the assignment. If $P \in \mathscr{D}_{\mathsf{r}}$ is a relational pre-condition then the relational post-condition is simply $\exists x'.P[x := x'] \sqcap_{\mathsf{r}} C\llbracket\mathrm{x} := E\rrbracket$. The partition transformer $\mathscr{F}_{\mathsf{p}}\llbracket\mathrm{x} := E\rrbracket$ and the block relational transformers $\llbracket\mathrm{x} := E\rrbracket_{B \longrightarrow B'}$ follow by composition of the previous primitives of substitution and variable elimination.

12.9.8 Widening. A simple solution for widening ∇_{r} (and narrowing) is the blockwise application for the greatest common partition $\pi_1 \curlyvee \pi_2$ which, similarly to the meet, is defined as follows:

$$\llbracket\nabla_{\mathsf{r}}\rrbracket_{\pi_1 \times \pi_2 \longrightarrow \pi_1 \curlyvee \pi_2}\left(\textstyle\prod_{B_1 \in \pi_1} P_{B_1}, \prod_{B_2 \in \pi_2} P_{B_2}\right) \triangleq \\ \prod_{B' \in \pi_1 \curlyvee \pi_2} \sqcap_{\mathsf{r}}\{P_{B_1} \nabla_{\mathsf{r}} P_{B_2} \mid \langle B_1, B_2\rangle \in \pi_1 \times \pi_2, B_1 \cup B_2 \subseteq B'\}$$

It must be checked that the widening is terminating (a coarser one would use $\bigsqcup_{\mathsf{r}}$). One can also apply the technique of not using a widening when the dimensions of the relational properties strictly increase (Section 11.3) as well as the widening for cofibred domains [Venet 1996].

12.9.9 Join. The concrete join $\mathscr{F}\llbracket\sqcup\rrbracket \in \mathscr{P} \times \mathscr{P} \longrightarrow \mathscr{P}$ is $\mathscr{F}\llbracket\sqcup\rrbracket(X_1, X_2) = P_1 \cup P_2$. We assume that the relational domain $\mathscr{D}_{\mathsf{r}}$ provides a sound abstract join $\sqcup_{\mathsf{r}} = \mathscr{F}_{\mathsf{r}}\llbracket\sqcup\rrbracket \in \mathscr{D}_{\mathsf{r}} \times \mathscr{D}_{\mathsf{r}} \longrightarrow \mathscr{D}_{\mathsf{r}}$, namely, $\forall P_1, P_2 \in \mathscr{D}_{\mathsf{r}}.\, \gamma_{\mathsf{r}}(P_1) \cup \gamma_{\mathsf{r}}(P_2) \subseteq \gamma_{\mathsf{r}}(P_1 \sqcup_{\mathsf{r}} P_2)$. We have to define a partitioned join $\llbracket\sqcup_{\mathsf{r}}\rrbracket_{\pi \times \pi' \longrightarrow \bar{\bar{\pi}}} \in (\mathscr{D}_{\mathsf{r}} \downarrow \pi \times \mathscr{D}_{\mathsf{r}} \downarrow \pi') \longrightarrow \mathscr{D}_{\mathsf{r}} \downarrow \bar{\bar{\pi}}$ which is sound. The blockwise join of $P = \prod_{B \in \pi} P_B$ and $P' = \prod_{B' \in \pi'} P'_{B'}$ for the greatest common partition $\bar{\bar{\pi}} = \pi \curlyvee \pi'$ is usually imprecise as shown by [Halbwachs et al. 2006, Section 5.2] for polyhedra. We therefore follow the technique in [Singh et al. 2017, 2018]. We first express the partitioned properties P and P' using the greatest common partition $\bar{\pi} \triangleq \pi \curlyvee \pi'$ by defining $\overline{P} \triangleq [\![\mathtt{skip}]\!]_{\pi\longrightarrow\bar{\pi}}(P)$ and $\overline{P}' \triangleq [\![\mathtt{skip}]\!]_{\pi'\longrightarrow\bar{\pi}}(P')$. Then, for $\overline{P} = \prod_{\bar{B}\in\bar{\pi}} \overline{P}_{\bar{B}}$ and $\overline{P}' = \prod_{\bar{B}\in\bar{\pi}} \overline{P}'_{\bar{B}}$, we consider the partition

$$\bar{\bar{\pi}} \quad \triangleq \quad \{\bar{B} \in \bar{\pi} \mid \overline{P}_{\bar{B}} =_{\mathsf{r}} \overline{P}'_{\bar{B}}\} \cup \{\bigcup\{\bar{B} \in \bar{\pi} \mid \overline{P}_{\bar{B}} \neq_{\mathsf{r}} \overline{P}'_{\bar{B}}\}\}.$$

Hence, $\bar{\bar{\pi}}$ preserves the blocks of semantically equal components and merges the blocks of semantically different components. $\overline{P}$ and $\overline{P}'$ are further decomposed according to the partition $\bar{\bar{\pi}}$ by defining $\overline{\overline{P}} \triangleq [\![\mathtt{skip}]\!]_{\bar{\pi}\longrightarrow\bar{\bar{\pi}}}(\overline{P})$ and $\overline{\overline{P}}' \triangleq [\![\mathtt{skip}]\!]_{\bar{\pi}\longrightarrow\bar{\bar{\pi}}}(\overline{P}')$. The block relational transformer is the join on the relational domain defined by $[\![\sqcup_{\mathsf{r}}]\!]_{\langle B,\, B\rangle\longrightarrow B}(\overline{\overline{P}}_B, \overline{\overline{P}}'_B) \triangleq \overline{\overline{P}}_B \sqcup_{\mathsf{r}} \overline{\overline{P}}'_B$ where $B \in \bar{\bar{\pi}}$. The partitioned relational property transformer $[\![\sqcup_{\mathsf{r}}]\!]_{\pi\times\pi'\longrightarrow\bar{\bar{\pi}}}$ is then defined componentwise:

$$[\![\sqcup_{\mathsf{r}}]\!]_{\pi\times\pi'\longrightarrow\bar{\bar{\pi}}}(P, P') \quad \triangleq \quad \prod_{B\in\bar{\bar{\pi}}} [\![\sqcup_{\mathsf{r}}]\!]_{\langle B,\, B\rangle\longrightarrow B}(\overline{\overline{P}}_B, \overline{\overline{P}}'_B).$$

12.9.10 Examples. [Halbwachs et al. 2006, 2003; Singh et al. 2015, 2017, 2018] are all instances of the dynamic partitioning of relational numerical domains by online A^{2}I. Of course, when considering a particular relational domain $\mathcal{D}_{\mathsf{r}}$, the partitioned transformers can be made more precise by taking specific features of $\mathcal{D}_{\mathsf{r}}$ into account (*e.g.* [Halbwachs et al. 2006, Proposition 1]).

13 CONCLUSION

We introduced the notion of abstract^2 interpretation, or meta-abstract interpretation, as a general method for lifting standard abstract interpretation from properties of a program to properties of a program analyser. In our view, analysing program analyses, or meta-analysis, corresponds precisely to analyse the collection of (partial) program properties that the analyser generates in order to supply an approximate program semantics as result. This can be modeled by abstract interpreting the analyser, *i.e.*, by abstracting the properties of the traces produced by a collecting semantics of the program analyser. We allow finite, infinite and transfinite (up to the ordinal ω+1) traces in the collecting semantics of the analyser in order to let the meta-analysis observe the full behaviour (possibly including nontermination) of the analyser. The whole framework of abstract interpretation can be lifted from program properties to properties of the analyser, thus providing the possibility of designing meta-abstract interpreters in a calculational way as it is already done for standard program analyses. We distinguish between offline and online meta-analyses. An offline meta-analysis extracts properties of the analyser before (or after) its execution on a given input program. As an example, we showed that offline meta-analysis allows us to design widening operators for the standard interval program analysis by detecting unstable bounds of the intervals computed by the analyser. On the other hand, an online meta-analysis is performed during program analysis in order to optimise its precision/cost ratio. This allows us to view the dynamic variable partitioning techniques for polyhedra by Halbwachs et al. [2003] and Singh et al. [2017] as an instance of an online meta-analysis.

ACKNOWLEDGMENTS

This work is supported by the NSF under Grants CNS-1446511 and CCF-1617717, by the ATEN Project by Fondazione Cariverona and by the Talento Grant 2016-T3/TIC-1995 of the Comunidad de Madrid, Fundación IMDEA Software & Universidad Politécnica de Madrid.

REFERENCES

Samson Abramsky and Achim Jung. 1994. Domain Theory. In *Handbook of Logic in Computer Science*, Samson Abramsky, Dov M. Gabbay, and Thomas Stephen Edward Maibaum Maibaum (Eds.). Vol. 3: Semantic Structures. Clarendon Press, Oxford, Chapter 1, 1–168.

Bowen Alpern and Fred B. Schneider. 1987. Recognizing Safety and Liveness. *Distributed Computing* 2, 3 (1987), 117–126. https://doi.org/10.1007/BF01782772

Roberto Bagnara, Patricia M. Hill, Elisa Ricci, and Enea Zaffanella. 2005. Precise widening operators for convex polyhedra. *Sci. Comput. Program.* 58, 1-2 (2005), 28–56. https://doi.org/10.1016/j.scico.2005.02.003

Julien Bertrane, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, and Xavier Rival. 2015. Static Analysis and Verification of Aerospace Software by Abstract Interpretation. *Foundations and Trends in Programming Languages* 2, 2-3 (2015), 71–190. https://doi.org/10.1561/2500000002

Frédéric Besson, Thomas P. Jensen, and Jean-Pierre Talpin. 1999. Polyhedral Analysis for Synchronous Languages. In *Proceedings of the 6th International Static Analysis Symposium (SAS'99), LNCS vol. 1694*. Springer, 51–68. https://doi.org/10.1007/3-540-48294-6_4

Bruno Blanchet, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, and Xavier Rival. 2003. A static analyzer for large safety-critical software. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'03)*. ACM, 196–207. https://doi.org/10.1145/781131.781153

François Bourdoncle. 1992. Abstract Interpretation by Dynamic Partitioning. *J. Funct. Program.* 2, 4 (1992), 407–423. https://doi.org/10.1017/S0956796800000496

Cristian Cadar and Alastair F. Donaldson. 2016. Analysing the Program Analyser. In *Proceedings of the 38th International Conference on Software Engineering Companion (ICSE'16)*. ACM, 765–768. https://doi.org/10.1145/2889160.2889206

Patrick Cousot. 1977. *Asynchronous iterative methods for solving a fixed point system of monotone equations in a complete lattice.* Res. rep. R.R. 88. Laboratoire IMAG, Université scientifique et médicale de Grenoble, Grenoble, France. 15 p.

Patrick Cousot. 1999. The Calculational Design of a Generic Abstract Interpreter. In *Calculational System Design*, Manfred Broy and Ralf Steinbrüggen (Eds.). NATO ASI Series F. IOS Press, Amsterdam.

Patrick Cousot. 2015. Abstracting Induction by Extrapolation and Interpolation. In *Proceedings 16th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'15), LNCS Vol. 8931*. Springer, 19–42. https://doi.org/10.1007/978-3-662-46081-8_2

Patrick Cousot and Radhia Cousot. 1976. Static determination of dynamic properties of programs. In *Proceedings of the Second International Symposium on Programming*. Dunod, Paris, France, 106–130.

Patrick Cousot and Radhia Cousot. 1977a. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL'77)*. ACM, 238–252. https://doi.org/10.1145/512950.512973

Patrick Cousot and Radhia Cousot. 1977b. Static determination of dynamic properties of recursive procedures. In *IFIP Conf. on Formal Description of Programming Concepts, St-Andrews, N.B., CA*, E.J. Neuhold (Ed.). North-Holland Pub. Co., 237–277.

Patrick Cousot and Radhia Cousot. 1979. Systematic Design of Program Analysis Frameworks. In *Proceedings of the 6th Annual ACM Symposium on Principles of Programming Languages (POPL'79)*. ACM, 269–282. https://doi.org/10.1145/567752.567778

Patrick Cousot and Radhia Cousot. 1992. Comparing the Galois Connection and Widening/Narrowing Approaches to Abstract Interpretation, invited paper. In *Proceedings of the International Workshop Programming Language Implementation and Logic Programming, LNCS Vol. 631 (PLILP'92)*. Springer, 269–295. https://doi.org/10.1007/3-540-55844-6_142

Patrick Cousot and Radhia Cousot. 1993. Galois Connection Based Abstract Interpretations for Strictness Analysis (Invited Paper). In *Proceedings of the International Conference on Formal Methods in Programming and Their Applications, LNCS Vol. 735*. Springer, 98–127. https://doi.org/10.1007/BFb0039703

Patrick Cousot and Nicolas Halbwachs. 1978. Automatic Discovery of Linear Restraints Among Variables of a Program. In *Proceedings of the 5th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL'78)*. ACM, 84–96. https://doi.org/10.1145/512760.512770

Graeme Gange, Jorge A. Navas, Peter Schachte, Harald Søndergaard, and Peter J. Stuckey. 2014. Interval Analysis and Machine Arithmetic: Why Signedness Ignorance Is Bliss. *ACM Trans. Program. Lang. Syst.* 37, 1 (2014), 1:1–1:35. https://doi.org/10.1145/2651360

Roberto Giacobazzi, Francesco Logozzo, and Francesco Ranzato. 2015. Analyzing Program Analyses. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, (POPL'15)*. ACM, 261–273. https://doi.org/10.1145/2676726.2676987

George Grätzer. 2011. *Lattice Theory: Foundation.* Birkhäuser, Basel. https://doi.org/10.1007/978-3-0348-0018-1

Nicolas Halbwachs. 1979. *Détermination automatique de relations linéaires vérifiées par les variables d'un programme (in French).* Thèse de 3$^{\text{ème}}$ cycle informatique. Université de Grenoble Alpes, Grenoble, France.

Nicolas Halbwachs, David Merchat, and Laure Gonnord. 2006. Some ways to reduce the space dimension in polyhedra computations. *Formal Methods in System Design* 29, 1 (2006), 79–95. https://doi.org/10.1007/s10703-006-0013-2

Nicolas Halbwachs, David Merchat, and Catherine Parent-Vigouroux. 2003. Cartesian Factoring of Polyhedra in Linear Relation Analysis. In *Proceedings of 10th International Static Analysis Symposium (SAS'03), LNCS Vol. 2694*. Springer, 355–365. https://doi.org/10.1007/3-540-44898-5_20

Nicolas Halbwachs, Yann-Erick Proy, and Patrick Roumanoff. 1997. Verification of Real-Time Systems using Linear Relation Analysis. *Formal Methods in System Design* 11, 2 (01 Aug 1997), 157–185. https://doi.org/10.1023/A:1008678014487

Chris Hankin and Sebastian Hunt. 1994. Approximate Fixed Points in Abstract Interpretation. *Sci. Comput. Program.* 22, 3 (1994), 283–306. https://doi.org/10.1016/0167-6423(94)00007-7

Kihong Heo, Hakjoo Oh, and Hongseok Yang. 2016. Learning a Variable-Clustering Strategy for Octagon from Labeled Data Generated by a Static Analysis. In *Proceedings of 23rd International Static Analysis Symposium (SAS'16), LNCS Vol. 9837.* Springer, 237–256. https://doi.org/10.1007/978-3-662-53413-7_12

Kihong Heo, Hakjoo Oh, and Kwangkeun Yi. 2017. Selective conjunction of context-sensitivity and octagon domain toward scalable and precise global static analysis. *Softw., Pract. Exper.* 47, 11 (2017), 1677–1705. https://doi.org/10.1002/spe.2493

Neil D. Jones and Flemming Nielson. 1995. Abstract Interpretation: A Semantics-based Tool for Program Analysis. In *Handbook of Logic in Computer Science (Vol. 4)*, S. Abramsky, Dov M. Gabbay, and T. S. E. Maibaum (Eds.). Oxford University Press, Inc., 527–636. http://dl.acm.org/citation.cfm?id=218623.218637

Gary A. Kildall. 1973. A Unified Approach to Global Program Optimization. In *Proceedings of the ACM Symposium on Principles of Programming Languages (POPL'73).* ACM, 194–206. https://doi.org/10.1145/512927.512945

Woosuk Lee, Wonchan Lee, Dongok Kang, Kihong Heo, Hakjoo Oh, and Kwangkeun Yi. 2017. Sound Non-Statistical Clustering of Static Analysis Alarms. *ACM Trans. Program. Lang. Syst.* 39, 4 (2017), 16:1–16:35. https://doi.org/10.1145/3095021

Huisong Li, Francois Berenger, Bor-Yuh Evan Chang, and Xavier Rival. 2017. Semantic-directed clumping of disjunctive abstract states. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017.* ACM, 32–45. https://doi.org/10.1145/3009837.3009881

Antoine Miné. 2006. The octagon abstract domain. *Higher-Order and Symbolic Computation* 19, 1 (2006), 31–100. https://doi.org/10.1007/s10990-006-8609-1

Alan Mycroft. 1982. *Abstract interpretation and optimising transformations for applicative programs.* Ph.D. Dissertation. University of Edinburgh, UK.

Hakjoo Oh, Wonchan Lee, Kihong Heo, Hongseok Yang, and Kwangkeun Yi. 2016. Selective X-Sensitive Analysis Guided by Impact Pre-Analysis. *ACM Trans. Program. Lang. Syst.* 38, 2 (2016), 6:1–6:45. https://doi.org/10.1145/2821504

Dana S. Scott. 1972. Mathematical concepts in programming language semantics. In *AFIPS Spring Joint Computing Conference (AFIPS Conference Proceedings)*, Vol. 40. AFIPS, 225–234. https://doi.org/10.1145/1478873.1478903

Gagandeep Singh, Markus Püschel, and Martin Vechev. 2015. Making Numerical Program Analysis Fast. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'15).* ACM, 303–313. https://doi.org/10.1145/2737924.2738000

Gagandeep Singh, Markus Püschel, and Martin T. Vechev. 2017. Fast polyhedra abstract domain. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, (POPL'17).* ACM, 46–59. https://doi.org/10.1145/3009837.3009885

Gagandeep Singh, Markus Püschel, and Martin T. Vechev. 2018. A practical construction for decomposing numerical abstract domains. *Proc. ACM Program. Lang.* 2, POPL (2018), 55:1–55:28. https://doi.org/10.1145/3158143

Arnaud Venet. 1996. Abstract Cofibered Domains: Application to the Alias Analysis of Untyped Programs. In *Proceedings of the 3rd International Static Analysis Symposium (SAS'96), LNCS Vol. 1145.* Springer, 366–382. https://doi.org/10.1007/3-540-61739-6_53