Decidability and Synthesis of Abstract Inductive Invariants

Francesco Ranzato
Dipartimento di Matematica, University of Padova, Italy
https://www.math.unipd.it/~ranzato
francesco.ranzato@unipd.it

Abstract

Decidability and synthesis of inductive invariants ranging in a given domain play an important role in software verification. We consider here inductive invariants belonging to an abstract domain A as defined in abstract interpretation, namely, ensuring the existence of the best approximation in A of any system property. In this setting, we study the decidability of the existence of abstract inductive invariants in A of transition systems and their corresponding algorithmic synthesis. Our model relies on some general results which relate the existence of abstract inductive invariants with least fixed points of best correct approximations in A of the transfer functions of transition systems and their completeness properties. This approach allows us to derive decidability and synthesis results for abstract inductive invariants which are applied to the well-known Karr's numerical abstract domain of affine equalities. Moreover, we show that a recent general algorithm for synthesizing inductive invariants in domains of logical formulae can be systematically derived from our results and generalized to a range of algorithms for computing abstract inductive invariants.

2012 ACM Subject Classification Theory of computation → Invariants; Theory of computation → Abstraction

Keywords and phrases Inductive invariant, program verification, abstract interpretation

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2020.30

Funding *Francesco Ranzato*: Partially funded by *University of Padova*, under the SID2018 project "Analysis of STatic Analyses (ASTA)"; *Italian Ministry of University and Research*, under the PRIN2017 project no. 201784YSZ5 "AnalysiS of PRogram Analyses (ASPRA)"; *Facebook Research*, under a "Probability and Programming Research Award".

Acknowledgements I thank Patrick Cousot for comments and suggestions on an earlier version of this paper.

1 Introduction

Proof and inference methods based on inductive invariants are widespread in automatic (or semi-automatic) program and system verification (see, *e.g.*, [2,5,8,9,10,11,18,19,22,25,28,32]). The inductive invariant proof method roots at the works of Floyd [13], Park [29,30], Naur [27] and Manna et al. [20]. Given a transition system $\mathcal{T} = \langle \Sigma, \tau, \Sigma_0 \rangle$, where τ is a transition relation on states ranging in Σ and $\Sigma_0 \subseteq \Sigma$ is a subset of initial states, together with a safety property $P \subseteq \Sigma$ to check, let us recall that a property $I \in \wp(\Sigma)$ is an *inductive invariant* for $\langle \mathcal{T}, P \rangle$ when: $\Sigma_0 \subseteq I$, *i.e.* the initial states satisfy I; $I \subseteq P$, *i.e.* I entails P; $\tau(I) \subseteq I$, *i.e.* I is inductive. The *inductive invariant principle* states that P holds for all the reachable states of $\mathcal{T}$ iff there exists an inductive invariant I for $\langle \mathcal{T}, P \rangle$. In such an explicit form this principle has been probably first formulated in 1982 by Cousot and Cousot [5, Section 5] and called "induction principle for invariance proofs". In most cases, verification and inference methods rely on inductive invariants I that range in some restricted domain $\mathcal{A} \subseteq \wp(\Sigma)$, such as a domain of logical formulae (*e.g.*, some separation logic or a fragment of first-order logic [28]) or a domain of abstract interpretation [3,4] (*e.g.*, numerical abstract domains of affine relations or convex polyhedra). In this context, if an inductive invariant I belongs to $\mathcal{A}$ then I is called an *abstract inductive invariant* (inductive $\mathcal{A}$-invariant in [32, Section 1]).

31st International Conference on Concurrency Theory (CONCUR 2020).
Editors: Igor Konnov and Laura Kovács; Article No. 30; pp. 30:1–30:21
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Main Contributions. Our primary goal was to investigate whether and how the inductive invariant principle can be adapted when inductive invariants are restricted to range in an abstract domain $\mathcal{A}$. We make the following working assumption: $\mathcal{A} \subseteq \wp(\Sigma)$ is an abstract domain as defined in abstract interpretation [3, 4]. This means that each state property $X \in \wp(\Sigma)$ has a *best* over-approximation (w.r.t. $\subseteq$) $\alpha_{\mathcal{A}}(X)$ in $\mathcal{A}$ and each state transition relation τ has a *best* correct approximation $\tau^{\mathcal{A}}$ on the abstract domain $\mathcal{A}$. Under these hypotheses, we prove an *abstract inductive invariant principle* stating that there exists an abstract inductive invariant in $\mathcal{A}$ proving a property P of a transition system $\mathcal{T}$ iff the *best abstraction* $\mathcal{T}^{\mathcal{A}}$ in $\mathcal{A}$ of the system $\mathcal{T}$ allows us to prove P. The decidability/undecidability question of the existence of abstract inductive invariants in some abstract domain $\mathcal{A}$ for some class of transition systems has been recently investigated in a few significant cases [12, 16, 24, 31, 32]. We show how the abstract inductive invariant principle allows us to derive a general decidability result on the existence of inductive invariants in some abstract domain $\mathcal{A}$ and to design a general algorithm for synthesizing the least (w.r.t. the order of $\mathcal{A}$) abstract inductive invariant in $\mathcal{A}$, when this exists, by a least fixpoint computation in $\mathcal{A}$.

We also show a related result which is of independent interest in abstract interpretation: the (concrete) inductive invariant principle for a system $\mathcal{T}$ is equivalent to the abstract inductive invariant principle for $\mathcal{T}$ on an abstract domain $\mathcal{A}$ iff *fixpoint completeness* of $\mathcal{T}$ on $\mathcal{A}$ holds, *i.e.*, the best abstraction in $\mathcal{A}$ of the reachable states of $\mathcal{T}$ coincides with the reachable states of the best abstraction $\mathcal{T}^{\mathcal{A}}$ of $\mathcal{T}$ on $\mathcal{A}$.

The decidability/synthesis of abstract inductive invariants in a domain $\mathcal{A}$ for some class $\mathcal{C}$ of systems essentially boils down to prove that the best correct approximation $\tau^{\mathcal{A}}$ in $\mathcal{A}$ of the transition relation τ of systems in $\mathcal{C}$ is algorithmically computable. As case study, we provide one such result for Karr's affine relationships [17], which is a well-known and widely used abstract domain in numerical program analysis [21]. As a second application, we design an inductive invariant synthesis algorithm which, by generalizing an algorithm by Padon et al. [28] tailored for logical invariants, outputs the most abstract (*i.e.*, weakest/greatest) inductive invariant in a domain $\mathcal{A}$ which satisfies some suitable hypotheses. In particular, we show that this synthesis algorithm is obtained by instantiating a concrete co-inductive greatest fixpoint checking algorithm by Cousot [1] to a domain $\mathcal{A}$ of abstract invariants which is *disjunctive*, *i.e.*, abstract least upper bounds of $\mathcal{A}$ do not lose precision. This generalization allows us to design further related co-inductive algorithms for synthesizing abstract inductive invariants.

Due to lack of space in the main body of the paper, the proofs are moved to Section A.2 in Appendix A.

2 Background

2.1 Order Theory

If X is a subset of some universe set U then $\neg X$ denotes the complement of X with respect to U when U is implicitly given by the context. If $f : X \to Y$ is a function between sets and $S \in \wp(X)$ then $f(S) \triangleq \{f(x) \in Y \mid x \in S\}$ denotes the image of f on S. If $\vec{x} \in X^n$ is a vector in a product domain, $j \in [1, n]$ and $y \in X$ then $\vec{x}[x_j/y]$ denotes the vector obtained from $\vec{x}$ by replacing its j-th component x_j with y. To keep the notation simple and compact, we use the same symbol for a function/relation and its componentwise (*i.e.* pointwise) extension on product domains, *e.g.*, if $\vec{S}, \vec{T} \in \wp(X)^n$ then $\vec{S} \subseteq \vec{T}$ denotes that for all $i \in [1, n]$, $\vec{S}_i \subseteq \vec{T}_i$. Sometimes, to emphasize a pointwise definition, a dotted notation can be used such as in $f \mathrel{\dot{\leq}} g$ for the pointwise ordering between functions.

A quasiordered set (or poset) $D_{\leq}$ satisfies the ascending (resp. descending) chain condition (ACC, resp. DCC) if D contains no countably infinite sequence of distinct elements $\{x_i\}_{i\in\mathbb{N}}$ such that, for all $i \in \mathbb{N}$, $x_i \leq x_{i+1}$ (resp. $x_{i+1} \leq x_i$). A poset is a directed-complete partial order (CPO) if it has the least upper bound (lub) of all its directed subsets. A complete lattice is a poset having the lub of all its arbitrary (possibly empty) subsets (and therefore having arbitrary glbs). In a complete lattice (or CPO), $\vee$ (or $\sqcup$) and $\wedge$ (or $\sqcap$) denote, resp., lub and glb, and $\bot$ and $\top$ denote, resp., least and greatest element.

Let $P_{\leq}$ be a poset and $f : P \to P$. Then, $\mathrm{Fix}(f) \triangleq \{x \in P \mid f(x) = x\}$, $\mathrm{Fix}^{\leq}(f) \triangleq \{x \in P \mid f(x) \leq x\}$, $\mathrm{Fix}^{\geq}(f) \triangleq \{x \in P \mid f(x) \geq x\}$, and $\mathsf{lfp}(f)$, $\mathsf{gfp}(f)$ denote, resp., the least and greatest fixpoint in $\mathrm{Fix}(f)$, when they exist. Let us recall Knaster-Tarski fixpoint theorem: if $\langle C, \leq, \vee, \wedge\rangle$ is a complete lattice and $f : C \to C$ is monotonic then $\langle \mathrm{Fix}(f), \leq\rangle$ is a complete lattice, $\mathsf{lfp}(f) = \wedge \mathrm{Fix}^{\leq}(f)$ and $\mathsf{gfp}(f) = \vee \mathrm{Fix}^{\geq}(f)$. Also, Knaster-Tarski-Kleene fixpoint theorem states that if $\langle C, \leq, \vee, \bot\rangle$ is a CPO with least element and $f : C \to C$ is Scott-continuous (*i.e.*, f preserves lubs of directed subsets) then $\mathsf{lfp}(f) = \vee_{i\in\mathbb{N}} f^i(\bot)$, where, for all $x \in C$ and $i \in \mathbb{N}$, $f^0(x) \triangleq x$ and $f^{i+1}(x) \triangleq f(f^i(x))$; dually, if $\langle C, \leq, \wedge, \top\rangle$ is a dual-CPO with greatest element and $f : C \to C$ is Scott-co-continuous then $\mathsf{gfp}(f) = \wedge_{i\in\mathbb{N}} f^i(\top)$. A function $f : C \to C$ on a complete lattice is additive when f preserves arbitrary lubs.

2.2 Abstract Domains

Let us recall some basic notions on closures and Galois connections which are commonly used in abstract interpretation [3, 4] to define abstract domains (see, *e.g.*, [21]). Closure operators and Galois connections are equivalent notions and are both used for defining the notion of approximation in abstract interpretation, where closure operators bring the advantage of defining abstract domains independently of a specific representation for abstract objects which is required by Galois connections.

An upper closure operator (uco), or simply upper closure, on a poset $C_{\leq}$ is a function $\mu : C \to C$ which is monotonic, idempotent and extensive (*i.e.*, $x \leq \mu(x)$ for all $x \in C$). Dually, a lower closure operator (lco) $\eta : C \to C$ is monotonic, idempotent and reductive (*i.e.*, $\eta(x) \leq x$ for all $x \in C$). The set of all upper/lower closures on $C_{\leq}$ is denoted by $\mathrm{uco}(C_{\leq})/\mathrm{lco}(C_{\leq})$. We write $c \in \mu(C)$, or simply $c \in \mu$, to denote that there exists $c' \in C$ such that $c = \mu(c')$, and we recall that this happens iff $\mu(c) = c$. In what follows, assume that $C_{\leq}$ is a complete lattice. Let us recall that $\langle \mu(C), \leq\rangle$ is closed under glb of arbitrary subsets and, conversely, $X \subseteq C$ is the image of some $\mu \in \mathrm{uco}(C)$ iff X is closed under glb of all its subsets, and in this case $\mu(c) = \wedge\{c' \in X \mid c \leq c'\}$ holds. Dually, $X \subseteq C$ is closed under arbitrary lub of its subsets iff X is the image a lower closure $\eta \in \mathrm{lco}(C)$, and in this case $\eta(c) = \vee\{c' \in X \mid c' \leq c\}$. In abstract interpretation, a closure $\mu \in \mathrm{uco}(C)$ on a concrete domain $C_{\leq}$ plays the role of an abstract domain having best approximations: $c \in C$ is (upper-)approximated by any $\mu(c')$ such that $c \leq \mu(c')$ and $\mu(c)$ is the best approximation of c in μ because $\mu(c) = \wedge\{\mu(c') \mid c' \in C, c \leq \mu(c')\}$.

A Galois Connection (GC, also called adjunction) between two posets $\langle C, \leq_C\rangle$, called concrete domain, and $\langle A, \leq_A\rangle$, called abstract domain, consists of two maps $\alpha : C \to A$ and $\gamma : A \to C$ such that $\alpha(c) \leq_A a \;\Leftrightarrow\; c \leq_C \gamma(a)$ holds. A GC is called Galois insertion (GI) when α is surjective or, equivalently, γ is injective. Any GC can be transformed into a GI simply by removing useless elements in $A \smallsetminus \alpha(C)$ from the abstract domain A. A GC/GI is denoted by $(C_{\leq_C}, \alpha, \gamma, A_{\leq_A})$. GCs and ucos are equivalent notions because any GC $\mathcal{G} = (C, \alpha, \gamma, A)$ induces a closure $\mu_{\mathcal{G}} \triangleq \gamma \circ \alpha \in \mathrm{uco}(C)$, any $\mu \in \mathrm{uco}(C)$ induces a GI $\mathcal{G}_\mu \triangleq (C, \mu, \lambda x.x, \mu(C))$, and these two transforms are inverse of each other.

2.3 Transition Systems

Let $\mathcal{T} = \langle \Sigma, \tau \rangle$ be a transition system where Σ is a set of states and $\tau \subseteq \Sigma \times \Sigma$ is a transition relation inducing the following transformers of type $\wp(\Sigma) \to \wp(\Sigma)$:

$$\mathsf{pre}(X) \triangleq \{s \in \Sigma \mid \exists s' \in X.(s,s') \in \tau\} \qquad \widetilde{\mathsf{pre}}(X) \triangleq \{s \in \Sigma \mid \forall s'.(s,s') \in \tau \Rightarrow s' \in X\}$$
$$\mathsf{post}(X) \triangleq \{s' \in \Sigma \mid \exists s \in X.(s,s') \in \tau\} \qquad \widetilde{\mathsf{post}}(X) \triangleq \{s' \in \Sigma \mid \forall s.(s,s') \in \tau \Rightarrow s \in X\}$$

We will equivalently specify a transition system by one of the above transformers (typically post) in place of the transition relation τ. Let us also recall (see *e.g.* [6]) that $(\wp(\Sigma)_\subseteq, \mathsf{pre}, \widetilde{\mathsf{post}}, \wp(\Sigma)_\subseteq)$ and $(\wp(\Sigma)_\subseteq, \mathsf{post}, \widetilde{\mathsf{pre}}, \wp(\Sigma)_\subseteq)$ are GCs. The set of reachable states of $\mathcal{T}$ from a set of initial states $\Sigma_0 \subseteq \Sigma$ is $\mathrm{Reach}[\mathcal{T}, \Sigma_0] \triangleq \mathsf{lfp}(\lambda X \in \wp(\Sigma).\, \Sigma_0 \cup \mathsf{post}(X))$, and $\mathcal{T}$ satisfies a safety property $P \subseteq \Sigma$ when $\mathrm{Reach}[\mathcal{T}, \Sigma_0] \subseteq P$ holds.

2.4 Inductive Invariant Principle

Given a transition system $\mathcal{T} = \langle \Sigma, \tau \rangle$, a set of states $I \in \wp(\Sigma)$ is an *inductive invariant* for $\mathcal{T}$ w.r.t. $\langle \Sigma_0, P \rangle \in \wp(\Sigma)^2$ when: (i) $\Sigma_0 \subseteq I$; (ii) $\mathsf{post}(I) \subseteq I$; (iii) $I \subseteq P$. An inductive invariant I allows us to prove that $\mathcal{T}$ is safe, *i.e.* $\mathrm{Reach}[\mathcal{T}, \Sigma_0] \subseteq P$, by the *inductive invariant principle* (a.k.a. fixpoint induction principle), a consequence of Knaster-Tarski fixpoint theorem: If $C_\leq$ is a complete lattice, $c' \in C$ and $f : C \to C$ is monotonic then

$$\mathsf{lfp}(f) \leq c' \Leftrightarrow \exists i \in C.\, f(i) \leq i \wedge i \leq c' \tag{1}$$

In particular, given $c, c' \in C$, since $c \vee_C f(i) \leq i$ iff $c \leq i \wedge f(i) \leq i$, it turns out that:

$$\mathsf{lfp}(\lambda x. c \vee_C f(x)) \leq c' \Leftrightarrow \exists i \in C.\, c \leq i \wedge f(i) \leq i \wedge i \leq c' \tag{2}$$

One such $i \in C$ such that $c \leq i \wedge f(i) \leq i \wedge i \leq c$ is called an inductive invariant of f for $\langle c, c' \rangle$. Hence, (2) is applied to the function $\lambda X.\Sigma_0 \cup \mathsf{post}(X) : \wp(\Sigma) \to \wp(\Sigma)$, which is monotonic on $\wp(\Sigma)_\subseteq$, so that $\mathsf{lfp}(\lambda X.\Sigma_0 \cup \mathsf{post}(X)) \subseteq P$ holds iff there exists an inductive invariant I for $\mathcal{T}$ w.r.t. $\langle \Sigma_0, P \rangle$. In most contexts for defining transition systems, the decision problem of the existence of a (concrete) inductive invariant for a class of transition systems w.r.t. a set of initial states and some safety property turns out to be undecidable.

3 Abstract Inductive Invariants

An array of recent works, [12,16,24,28,31,32] among the others, consider a notion of abstract inductive invariant and study the corresponding decidability/undecidability and synthesis problems. The common approach of these works consists in restricting the range of inductive invariants from a concrete domain C to some abstraction A_C of C, which, in a general setting, is simply a subset of C. Let us formalize abstract inductive invariants in order-theoretic terms. Given a class $\mathcal{C}$ of complete lattices and, for all $C \in \mathcal{C}$, a class of functions $\mathcal{F}_C \subseteq C \to C$, a set of initial properties $\mathrm{Init}_C \subseteq C$, a set of safety properties $\mathrm{Safe}_C \subseteq C$, and an abstract domain $A_C \subseteq C$, a first problem is the decidability of the following decision question:

$$\forall C \in \mathcal{C}. \forall f \in \mathcal{F}_C. \forall c \in \mathrm{Init}_C\, . \forall c' \in \mathrm{Safe}_C\, . \exists i \in^? A_C.\, c \leq i \wedge f(i) \leq i \wedge i \leq c' \tag{3}$$

where one such $i \in A_C$ is called an *abstract inductive invariant* for f and $\langle c, c' \rangle \in C^2$. Thakur et al. [32, Section 1] use the terminology "inductive A_C-invariant" when for some transition system $\langle \Sigma, \tau \rangle$, $f = \mathsf{post}_\tau$, $A_C \subseteq \wp(\Sigma)$ and $c' = \Sigma$.

The corresponding synthesis problem consists in designing algorithms which output abstract inductive invariants in A_C or notify that no inductive invariant in A_C exists.

Given $\mathcal{T} = \langle \Sigma, \tau \rangle$ whose successor transformer is post, the problem (3) is instantiated to $C_\leq = \wp(\Sigma)_\subseteq$, $f = \mathsf{post}(X)$, $c = \Sigma_0 \in \wp(\Sigma)$ set of initial states and $c' = P \in \wp(\Sigma)$ safety property. When $\mathcal{T}$ is the control flow graph generated by some program, Σ_0 are the states of some initial control node and P is a safety property given by the states which are not in some bad control node, abstract inductive invariants are called *separating invariants* and the decision problem (3) is called *Monniaux problem* by Fijalkow et al. [12], because this problem was first formulated by Monniaux [23, 24].

3.1 Abstract Inductive Invariant Principle

Our working assumption is that in problem (3) the invariants i range in an abstract domain A as dictated by abstract interpretation [3, 4].

▶ **Assumption 3.1.** $\langle A, \leq_A \rangle$ is an abstract domain of the complete lattice $\langle C, \leq_C \rangle$ which has best approximations, *i.e.*, one of these two equivalent assumptions is satisfied:
- **(i)** $(C_{\leq_C}, \alpha, \gamma, A_{\leq_A})$ is a Galois insertion;
- **(ii)** $\langle A, \leq_A \rangle = \langle \mu(C), \leq_C \rangle$ for some upper closure $\mu \in \mathrm{uco}(C_{\leq_C})$. ⌟

Under Assumption 3.1, let us recall that if $f : C \to C$ is a concrete monotonic function then the mappings $\alpha f \gamma : A \to A$, for the case of GIs, and $\mu f : \mu(C) \to \mu(C)$, for the case of ucos, are called *best correct approximation* (bca) in A of f. This is justified by the observation that an abstract function $f^\sharp : A \to A$ (or $f^\sharp : \mu(C) \to \mu(C)$ for ucos) is a correct (or sound) approximation of f when $\alpha f \gamma \mathrel{\dot{\leq}_A} f^\sharp$ (or $\mu f \mathrel{\dot{\leq}_C} f^\sharp$ for ucos) holds. Our first result is an *abstract* inductive invariant principle which restricts the invariants of f in (1) to those ranging in an abstract domain A: when the abstract domain A is specified by a GI, this means that $a \in A$ is an abstract invariant of f when $f\gamma(a) \leq_C \gamma(a)$ holds; when the abstract domain is a closure $\mu \in \mathrm{uco}(C)$, this means that $a \in \mu \subseteq C$ is an abstract invariant of f when $fa \leq_C a$ holds.

▶ **Lemma 3.2 (Abstract Inductive Invariant Principle).** *Let $(C_{\leq_C}, \alpha, \gamma, A_{\leq_A})$ be a GI. For all $c' \in C$ and $a' \in A$:*
- **(a)** $\gamma(\mathsf{lfp}(\alpha f \gamma)) \leq_C c' \Leftrightarrow \exists a \in A.\, f\gamma(a) \leq_C \gamma(a) \wedge \gamma(a) \leq_C c'$*;*
- **(b)** $\mathsf{lfp}(\alpha f \gamma) \leq_A a' \Leftrightarrow \exists a \in A.\, f\gamma(a) \leq_C \gamma(a) \wedge \gamma(a) \leq_C \gamma(a')$*.*

It is worth stating Lemma 3.2 (a) in an equivalent form for an abstract domain represented by a closure $\mu \in \mathrm{uco}(C)$: $\mathsf{lfp}(\mu f) \leq_C c' \Leftrightarrow \exists a \in \mu.\, fa \leq_C a \wedge a \leq_C c'$.

Let us observe that point (b) is an easy consequence of point (a), because, by surjectivity of α in GIs, for all $a' \in A$, there exists some $c' \in C$ such that $a' = \alpha(c')$, and $\gamma(\mathsf{lfp}(\alpha f \gamma)) \leq_C \gamma(\alpha(c')) \Leftrightarrow \mathsf{lfp}(\alpha f \gamma) \leq_A \alpha(c')$ holds. Moreover, point (b) easily follows from the inductive invariant principle (1) for the bca $\alpha f \gamma : A \to A$. On the other hand, it is worth remarking that point (a) cannot be obtained from (b), *i.e.* (a) is *strictly stronger* than (b), because (a) allows us to prove concrete properties $c' \in C$ which are not exactly represented by A (*i.e.*, $c' \notin \gamma(A)$) by abstract inductive invariants in A, as shown by the following tiny example.

▶ **Example 3.3.** Consider a 4-points chain $C = \{1 < 2 < 3 < 4\}$, the function $f : C \to C$ defined by $\{1 \mapsto 1;\, 2 \mapsto 2;\, 3 \mapsto 4;\, 4 \mapsto 4\}$, and the abstraction $A = \{2, 4\}$ with $\gamma = \mathrm{id}$ and $\alpha = \{1 \mapsto 2;\, 2 \mapsto 2;\, 3 \mapsto 4;\, 4 \mapsto 4\}$. Here, we have that $\alpha f \gamma = \{2 \mapsto 2;\, 4 \mapsto 4\}$ and $\mathsf{lfp}(\alpha f \gamma) = 2$. In this case, Lemma 3.2 (b) allows us to prove all the abstract properties $a' \in A$ by abstract inductive invariants, while Lemma 3.2 (a) allows us to prove an additional concrete property $3 \in C \smallsetminus \gamma(A)$, which is not exactly represented by A, by an abstract inductive invariant, and this would not be possible by resorting to Lemma 3.2 (b). Also, $\gamma(\mathsf{lfp}(\alpha f\gamma)) \not\leq 1$ holds, thus, by Lemma 3.2 (a), the concrete property $1 \in C \smallsetminus \gamma(A)$ cannot be proved by an abstract inductive invariant in A, whereas Lemma 3.2 (b) does not allow us to infer this. ⌟

Lemma 3.2 (b) tells us that the existence of an abstract inductive invariant of f proving an abstract property $a' \in A$ is equivalent to the fact that the least fixpoint of the bca $\alpha f\gamma$ entails a'. This formalizes for an abstract domain satisfying Assumption 3.1 an observation in [12, Section 1] stating (in our terminology) that "*the existence of some abstract inductive invariant for $\alpha f\gamma$ proving a' is equivalent to whether the strongest abstract invariant* $\mathsf{lfp}(\alpha f\gamma)$ *entails a'*", *i.e.* is inductive, and generalizes [32, Observation 1] stating (in our terminology) that "$\mathsf{lfp}(\alpha f\gamma)$ *is the strongest abstract inductive invariant*". If, instead, we aim at proving *any* concrete property $c' \in C$, possibly not in $\gamma(A)$, by an abstract inductive invariant then Lemma 3.2 (a) states that this is equivalent to the strictly stronger condition $\gamma(\mathsf{lfp}(\alpha f\gamma)) \leq_C c'$.

As a consequence of Lemma 3.2 (a) we derive the following characterization of the problem (3).

▶ **Corollary 3.4.** *Let $\mathcal{F} \subseteq C \to C$ and* $\mathrm{Init}, \mathrm{Safe} \subseteq C$. *The Monniaux decision problem* $\forall f \in \mathcal{F}. \forall c \in \mathrm{Init}\,. \forall c' \in \mathrm{Safe}\,. \exists a \in^? A.\ c \leq_C \gamma(a) \wedge f\gamma(a) \leq_C \gamma(a) \wedge \gamma(a) \leq_C c'$ *is decidable iff the decision problem* $\forall f \in \mathcal{F}. \forall c \in \mathrm{Init}\,. \forall c' \in \mathrm{Safe}\,. \gamma(\mathsf{lfp}(\lambda x \in A. \alpha(c) \vee_A \alpha f\gamma(x))) \leq_C^? c'$ *is decidable.*

Moreover, as a consequence of Lemma 3.2 (b) we obtain the following abstract invariant synthesis algorithm.

▶ **Corollary 3.5.** *Assume that the lub $\vee_A : A \times A \to A$ and the bca $\alpha f\gamma : A \to A$ are finitely computable, the partial order $\leq_A$ is decidable and A is an ACC CPO with least element. For all $c \in C$ such that $\alpha(c)$ is finitely computable and $a' \in A$, the following procedure:*

$$
\begin{aligned}
\mathrm{AI{\small NV}}(f, A, c, a') \;\triangleq\;\; & i := \alpha(c);\\
& \textbf{while } i \leq_A a' \textbf{ do } \{\textbf{if } \alpha f\gamma(i) \leq_A i \textbf{ return } i;\ \textbf{else } i := \alpha f\gamma(i);\}\\
& \textbf{return } \textit{no abstract inductive invariant for } f \textit{ and } \langle c, \gamma(a')\rangle;
\end{aligned}
$$

is a terminating algorithm which outputs the least abstract inductive invariant for f and $\langle c, \gamma(a')\rangle$, when one such abstract inductive invariant exists, otherwise outputs "no abstract inductive invariant".

Under the same hypotheses for the abstract domain A, Thakur et al. [32, Observation 2] state (in our terminology) that the problem of computing the least abstract inductive invariant in A for some successor transformer post_τ reduces to the problem of computing the best correct approximation $\alpha\, \mathsf{post}_\tau\, \gamma$.

4 Fixpoint Completeness and Abstract Inductive Invariants

4.1 Completeness in Abstract Interpretation

Soundness in abstract interpretation (or, more in general, in static analysis) is a mandatory requirement stating that no false negative can occur: if $f : C \to C$ and $f^\sharp : A \to A$ are the concrete and abstract monotonic transformers then *fixpoint soundness* means that $\alpha(\mathsf{lfp}(f)) \leq_A \mathsf{lfp}(f^\sharp)$ holds, so that a positive abstract proof $\mathsf{lfp}(f^\sharp) \leq_A a'$ entails that $\gamma(a')$ concretely holds, *i.e.*, $\mathsf{lfp}(f) \leq_C \gamma(a')$. Fixpoint soundness is usually proved as a consequence of *pointwise soundness*: if $f^\sharp$ is a pointwise correct approximation of f, *i.e.* $\alpha f \mathrel{\dot{\leq}_A} f^\sharp\alpha$, then $\alpha(\mathsf{lfp}(f)) \leq_A \mathsf{lfp}(f^\sharp)$ holds. While soundness is indispensable, completeness in abstract interpretation encodes an ideal situation where no false positives (also called false alarms) arise: *fixpoint completeness* means that $\alpha(\mathsf{lfp}(f)) = \mathsf{lfp}(f^\sharp)$ holds, so that $\mathsf{lfp}(f^\sharp) \not\leq_A a'$ entails $\mathsf{lfp}(f) \not\leq_C \gamma(a')$. One can also consider a *strong fixpoint completeness* requiring that $\mathsf{lfp}(f) = \gamma(\mathsf{lfp}(f^\sharp))$, so that $\mathsf{lfp}(f^\sharp) \not\leq_A \alpha(c')$ entails $\mathsf{lfp}(f) \not\leq_C c'$. However, it should be remarked that $\mathsf{lfp}(f) = \gamma(\mathsf{lfp}(f^\sharp))$ is much stronger than $\alpha(\mathsf{lfp}(f)) = \mathsf{lfp}(f^\sharp)$ since it means that the concrete lfp is precisely represented by the abstract lfp.

It is important to remark that if $f^\sharp$ is a pointwise correct approximation of f and fixpoint completeness for $f^\sharp$ holds then since $\alpha(\mathsf{lfp}(f)) \leq_A \mathsf{lfp}(\alpha f\gamma) \leq_A \mathsf{lfp}(f^\sharp)$ always holds, one obtains that $\alpha(\mathsf{lfp}(f)) = \mathsf{lfp}(\alpha f\gamma) = \mathsf{lfp}(f^\sharp)$ holds, namely, the bca $\alpha f\gamma$ is fixpoint complete as well. This means that the possibility (and therefore impossibility) of defining an approximate transformer $f^\sharp : A \to A$ on A which is fixpoint complete does not depend on the specific definition of $f^\sharp$ but is instead an intrisic *property of the abstract domain* A w.r.t. the concrete transformer f, as formalized by the equation $\alpha(\mathsf{lfp}(f)) = \mathsf{lfp}(\alpha f\gamma)$. Moreover, fixpoint completeness is typically proved as a by-product of *pointwise completeness* $\alpha f = f^\sharp\alpha$, and if $f^\sharp$ is pointwise complete then it turns out that $f^\sharp = \alpha f\gamma$, that is, $f^\sharp$ actually is the bca of f. This justifies why, without loss of generality, we can consider fixpoint and pointwise completeness of bca's $\alpha f\gamma$ only, *i.e.*, as properties of abstract domains [14, 15].

4.2 Characterizing Fixpoint Completeness by Abstract Inductive Invariants

We show that the abstract inductive invariant principle is closely related to fixpoint completeness. More precisely, we provide an answer to the following question: in the abstract inductive invariant principle as stated by Lemma 3.2, can we replace $\mathsf{lfp}(\alpha f\gamma)$ with $\alpha(\mathsf{lfp}(f))$? This question is settled by the following result.

▶ **Theorem 4.1.** *Let* $(C_{\leq_C}, \alpha, \gamma, A_{\leq_A})$ *be a GI.*

(a) $\mathsf{lfp}(f) = \gamma(\mathsf{lfp}(\alpha f\gamma))$ *iff* $\forall c' \in C.\big(\mathsf{lfp}(f) \leq_C c' \Leftrightarrow \exists a \in A.\ f\gamma(a) \leq_C \gamma(a) \wedge \gamma(a) \leq_C c'\big)$*;*

(b) $\alpha(\mathsf{lfp}(f)) = \mathsf{lfp}(\alpha f\gamma)$ *iff* $\forall a' \in A.\big(\mathsf{lfp}(f) \leq_C \gamma(a') \Leftrightarrow \exists a \in A.\ f\gamma(a) \leq_C \gamma(a) \wedge \gamma(a) \leq_C \gamma(a')\big)$*.*

Theorem 4.1 (b) can be stated by means of ucos as follows: if $\mu \in \mathrm{uco}(C)$ then $\mu(\mathsf{lfp}(f)) = \mathsf{lfp}(\mu f)$ iff $\forall a' \in \mu.(\mathsf{lfp}(f) \leq_C a' \Leftrightarrow \exists a \in \mu.\ fa \leq_C a \wedge a \leq_C a')$.

The above result can be read as follows. Since, by the inductive invariant principle (1), $\mathsf{lfp}(f) \leq_C c'$ iff there exists a concrete inductive invariant proving c', it turns out that Theorem 4.1 (a) states that the existence of an *abstract* inductive invariant proving c' is equivalent to the existence of *any* inductive invariant proving c' iff fixpoint completeness holds. In other terms, the (concrete) inductive invariant principle is equivalent to the abstract inductive invariant principle iff fixpoint completeness holds. This result is of independent interest in abstract interpretation, since it provides a new characterization of the key property of fixpoint completeness of abstract domains.

A further interesting characterization of fixpoint completeness is as follows.

▶ **Lemma 4.2.** $\alpha(\mathsf{lfp}(f)) = \mathsf{lfp}(\alpha f\gamma) \Leftrightarrow \exists a \in A. f\gamma(a) \leq_C \gamma(a) \wedge \gamma(a) \leq_C \gamma\alpha(\mathsf{lfp}(f))$.

As a consequence, fixpoint completeness for f does not hold in A iff the abstract property $\alpha(\mathsf{lfp}(f)) \in A$ cannot be proved by an abstract inductive invariant in A

▶ **Example 4.3.** Consider a 3-points chain $C = \{1 < 2 < 3\}$ and the monotonic concrete function $f : C \to C$ defined by $f = \{1 \mapsto 1;\ 2 \mapsto 3;\ 3 \mapsto 3\}$.

Consider the uco $\mu = \{2, 3\}$, *i.e.*, $\mu = \{1 \mapsto 2;\ 2 \mapsto 2;\ 3 \mapsto 3\}$, so that $\mu f = \{1 \mapsto 2;\ 2 \mapsto 3;\ 3 \mapsto 3\}$. Fixpoint completeness does not hold because $\mu(\mathsf{lfp}(f)) = \mu(1) = 2 < 3 = \mathsf{lfp}(\mu f)$. Thus, in accordance with Lemma 4.2, it turns out that $\mu(\mathsf{lfp}(f)) = 2$ cannot be inductively proved in the abstraction μ. In fact, $f(2) \not\leq 2$, while $f(3) \leq 3$ but $3 \not\leq \mu(\mathsf{lfp}(f))$.
Consider now the uco $\eta = \{1, 3\}$, *i.e.*, $\eta = \{1 \mapsto 1;\ 2 \mapsto 3;\ 3 \mapsto 3\}$, so that $\eta f = \{1 \mapsto 1;\ 2 \mapsto 3;\ 3 \mapsto 3\}$. Here, $\eta(\mathsf{lfp}(f)) = \eta(1) = 1 = \mathsf{lfp}(\eta f)$, therefore fixpoint completeness holds. Thus, by the uco version of Theorem 4.1 (b), any valid abstract invariant of f can be inductively proved: in fact, $1, 3 \in \eta$ are valid abstract invariants of f and are both inductive. ⌟

Due to lack of space, we moved to Appendix A.1 an application of Theorem 4.1 which provides a model showing how the "Safety $=^?$ Abstract Invariance" problem is related to fixpoint completeness in abstract interpretation, as informally hinted by Padon et al. [28].

5 Abstract Inductive Invariants of Nondeterministic Programs

We consider transition systems as represented by a control flow graph (CFG) of a possibly nondeterministic imperative program. A program is a tuple $\mathcal{P} = \langle Q, n, \mathbb{V}, \mathsf{T}, \rightarrow\rangle$ where Q is a finite set of control nodes (or program points), $n \in \mathbb{N}$ is the number of program variables of type $\mathbb{V}$ (*e.g.*, $\mathbb{V} = \mathbb{Z}, \mathbb{Q}, \mathbb{R}$), T is a finite set of (possibly nondeterministic) transfer functions of type $\mathbb{V}^n \to \wp(\mathbb{V}^n)$, $\rightarrow \subseteq Q \times \mathsf{T} \times Q$ is a (possibly nondeterministic) control flow relation, where $q \xrightarrow{t} q'$ denotes a flow transition with transfer function $t \in \mathsf{T}$. A program $\mathcal{P}$ therefore defines a transition system $\mathcal{T}_{\mathcal{P}} = \langle \Sigma, \tau\rangle$ where $\Sigma \triangleq Q \times \mathbb{V}^n$ is the set of states and the transition relation $\tau \subseteq \Sigma \times \Sigma$ is defined by $\langle (q, \vec{v}), (q', \vec{v}')\rangle \in \tau \overset{\triangle}{\Leftrightarrow} \exists t \in \mathsf{T}.\, q \xrightarrow{t} q' \wedge \vec{v}' \in t(\vec{v})$. The transfer functions in T include assignments and Boolean guards, where if $b \in \wp(\mathbb{V}^n)$ is a deterministic Boolean predicate (such as $x_1 + 2x_2 - 1 = 0$) then the corresponding transfer function $t_b : \mathbb{V}^n \to \wp(\mathbb{V}^n)$ is $t_b(\vec{v}) \triangleq \mathbf{if}\ \vec{v} \in b\ \mathbf{then}\ \{\vec{v}\}\ \mathbf{else}\ \varnothing$. Examples of transfer functions include: affine, polynomial, nondeterministic assignments and affine equalities guards. The next value transformer $\mathsf{post}_{\langle q,q'\rangle} : \wp(\mathbb{V}^n) \to \wp(\mathbb{V}^n)$ for a pair $\langle q, q'\rangle \in Q \times Q$ of control nodes is $\mathsf{post}_{\langle q,q'\rangle}(X) \triangleq \cup\{t(X) \in \wp(\mathbb{V}^n) \mid \exists t \in \mathsf{T}.q \xrightarrow{t} q'\}$. The complete lattice $\langle \wp(\Sigma), \subseteq\rangle$ of sets of states can be equivalently represented by the Q-indexed product lattice $\langle \wp(\mathbb{V}^n)^{|Q|}, \dot{\subseteq}\rangle$. Hence, the successor transformer $\mathsf{post}_{\mathcal{P}} : \wp(\mathbb{V}^n)^{|Q|} \to \wp(\mathbb{V}^n)^{|Q|}$ and the set of reachable states from $\Sigma_0 \in \wp(\mathbb{V}^n)^{|Q|}$ are defined as follows:

$$\mathsf{post}_{\mathcal{P}}(\langle X_q\rangle_{q\in Q}) \triangleq \langle \cup_{q\in Q}\, \mathsf{post}_{\langle q,q'\rangle}(X_q)\rangle_{q'\in Q} \qquad \mathrm{Reach}[\mathcal{P}, \Sigma_0] \triangleq \mathsf{lfp}(\lambda \vec{X}.\Sigma_0 \cup \mathsf{post}_{\mathcal{P}}(\vec{X}))$$

For all control nodes $q \in Q$ and vectors $\vec{X} \in \wp(\mathbb{V}^n)^{|Q|}$, we will also use $\pi_q(\vec{X})$ and $\vec{X}_q$ to denote the q-indexed component of $\vec{X}$, *e.g.*, $\mathrm{Reach}[\mathcal{P}, \Sigma_0]_q \in \wp(\mathbb{V}^n)$ will be the set of reachable values at control node q.

We are interested in decidability and synthesis of abstract inductive invariants ranging in an abstract domain A as specified by a GI $(\wp(\mathbb{V}^n)_{\subseteq}, \alpha, \gamma, A_{\leq_A})$ parametric on $n \in \mathbb{N}$. By Corollary 3.4, for a given class $\mathcal{C}$ of programs, a class Init of sets of initial states and a class Safe of sets of safety properties, the Monniaux problem (3) is decidable iff for all $\mathcal{P} = \langle Q, n, \mathbb{V}, \mathsf{T}, \rightarrow\rangle \in \mathcal{C}$, $\Sigma_0 \in$ Init and $P \in$ Safe,

$$\dot{\gamma}(\mathsf{lfp}(\lambda \vec{a} \in A^{|Q|}.\, \dot{\alpha}(\Sigma_0) \mathbin{\dot{\vee}_A} \dot{\alpha}(\mathsf{post}_{\mathcal{P}}(\dot{\gamma}(\vec{a}))))) \mathrel{\dot{\subseteq}^?} P \tag{4}$$

is decidable. Moreover, Corollary 3.5 provides an abstract inductive invariant synthesis algorithm AInv for safety properties represented by A (*i.e.*, $P \in \gamma(A)$) when A, $\mathcal{C}$, Init and Safe satisfy the hypotheses of Corollary 3.5.

5.1 Karr's Affine Equalities Domain

Program analysis on the domain of affine equalities has been introduced in 1976 by Karr [17] who designed some algorithms computing for each program point some correct affine equalities between numerical variables. This abstract domain, here denoted by $\textsc{Aff}$, is relatively simple and widely used in numerical program analysis (see, *e.g.*, [21]). Müller-Olm and Seidl [26] put forward simpler and more efficient algorithms for $\textsc{Aff}$ based on a different representation of affine sets and proved that $\textsc{Aff}$ is fixpoint complete for unguarded nondeterministic affine programs, while for linearly guarded nondeterministic affine programs it is undecidable whether a given affine equality holds at a given program point or not.

Let us briefly recall the definition of the abstract domain $\textsc{Aff}_n$ for n program variables ranging in $\mathrm{Var}_n \triangleq \{x_1, ..., x_n\}$ and assuming rational values[1], that is, $\mathbb{V} = \mathbb{Q}$. The logical abstract invariants represented by $\textsc{Aff}_n$ are finite (possibly empty) conjunctions of affine equalities between variables, namely, $\bigwedge_{j=1}^{k}(\sum_{i=1}^{n} m_{i,j}x_i + b_j = 0)$, with $m_{i,j}, b_j \in \mathbb{Q}$. Any conjunction of affine equalities defines an affine subset of $\mathbb{Q}^n$, and each subset $X \in \wp(\mathbb{Q}^n)$ is approximated by the least (w.r.t. $\subseteq$) affine subset containing X, which is:

$$\mathsf{aff}(X) \triangleq \{\textstyle\sum_{j=0}^{m}\lambda_j\vec{v}_j \in \mathbb{Q}^n \mid m \in \mathbb{N},\ \lambda_j \in \mathbb{Q},\ \vec{v}_j \in X,\ \sum_{j=0}^{m}\lambda_j = 1\}.$$

This map $\mathsf{aff} : \wp(\mathbb{Q}^n) \to \wp(\mathbb{Q}^n)$ is an upper closure on $\langle\wp(\mathbb{Q}^n), \subseteq\rangle$ whose fixpoints are precisely the affine subsets of $\mathbb{Q}^n$ and therefore may be used to define the affine equalities domain $\textsc{Aff}_n \triangleq \langle\mathsf{aff}(\wp(\mathbb{Q}^n)), \subseteq\rangle$ independently of a specific representation for its elements. Karr [17] represents affine sets by kernels of affine transformations stored as matrix-vector pairs, while Müller-Olm and Seidl [26] employ an affine basis of indepedent vectors called generators. One can switch from one representation to the other by solving equations and using Gaussian elimination. Here, we do not need to choose a specific representation of affine sets so that the upper closure aff is meant to act as abstraction map $\alpha_{\textsc{Aff}} : \wp(\mathbb{Q}^n) \to \textsc{Aff}_n$ and correspondingly the concretization $\gamma_{\textsc{Aff}} : \textsc{Aff}_n \to \wp(\mathbb{Q}^n)$ is the identity. For the sake of clarity, in our examples we will use logical affine equalities for representing affine sets. $\textsc{Aff}_n$ is a complete lattice of finite height $n+1$, because if $a, a' \in \textsc{Aff}_n$ and $a \subsetneq a'$ then $\dim(a) < \dim(a')$, where $\dim(\varnothing) = -1$ and $\dim(\mathbb{Q}^n) = n$. The domain $\textsc{Aff}_n$ is not closed under arbitrary unions, *i.e.*, aff is not an additive uco, so that the lub of $\mathcal{X} \subseteq \textsc{Aff}_n$ is given by $\sqcup_{\textsc{Aff}_n}\mathcal{X} \triangleq \mathsf{aff}(\cup_{X\in\mathcal{X}}X)$. A matrix-based algorithm for computing a binary lub $a \sqcup_{\textsc{Aff}} a'$ of two affine sets represented by affine transformations is given by Karr [17, Section 5.2] (a simpler and more efficient version is in [21, Section 5.2.2]), while a binary lub can be easily computed for the generators-based representation in [26, Section 3].

By Corollary 3.4, the existence of abstract inductive invariants in $\textsc{Aff}$ for a given class $\mathcal{C}$ of programs, Init of sets of initial states and Safe of sets of safety properties, is a decidable problem iff for all $\mathcal{P} \in \mathcal{C}$ with n variables and control nodes in Q, for all $\Sigma_0 \in \mathrm{Init}_{\mathcal{P}} \subseteq \wp(\mathbb{V}^n)^{|Q|}$ and for all $P \in \mathrm{Safe}_{\mathcal{P}} \subseteq \wp(\mathbb{V}^n)^{|Q|}$,

$$\dot{\gamma}_{\textsc{Const}}(\mathsf{lfp}(\lambda\vec{a} \in \textsc{Aff}_n^{|Q|}\,.\,\dot{\alpha}_{\textsc{Aff}}(\Sigma_0)\,\dot{\sqcup}_{\textsc{Aff}_n}\,\dot{\alpha}_{\textsc{Aff}}(\mathsf{post}_{\mathcal{P}}(\dot{\gamma}_{\textsc{Aff}}(\vec{a}))))) \mathrel{\dot{\subseteq}^{?}} P \tag{5}$$

is decidable. Therefore, when $\Sigma_0, P \in \textsc{Aff}_n$ and since $\textsc{Aff}_n^{|Q|}$ has finite height $|Q|(n+1)$, a sufficient condition for the decidability of the problem (5) is that the bca $\dot{\alpha}_{\textsc{Aff}} \circ \mathsf{post}_{\mathcal{P}} \circ \dot{\gamma}_{\textsc{Aff}} : \textsc{Aff}_n^{|Q|} \to \textsc{Aff}_n^{|Q|}$ is computable, where for all $\vec{a} \in \textsc{Aff}_n^{|Q|}$:

[1] Values range in $\mathbb{Q}$ because the representation of affine subspaces and the transfer functions rely on algorithms working on fields rather than rings such as $\mathbb{Z}$.

$$\alpha_{\dot{\mathrm{AFF}}}(\mathsf{post}_{\mathcal{P}}(\dot{\gamma}_{\mathrm{AFF}}(\vec{a}))) = \langle \sqcup_{\mathrm{AFF}_n}\{\alpha_{\mathrm{AFF}}(t(\pi_q(\dot{\gamma}_{\mathrm{AFF}}(\vec{a})))) \mid \exists q \in Q, \exists t \in \mathsf{T}_{\mathcal{P}}.\, q \xrightarrow{t} q'\}\rangle_{q' \in Q} \quad (6)$$

Because in (6) we have a finite lub, it is enough that for all the transfer functions $t \in \mathsf{T}_{\mathcal{P}}$ of $\mathcal{P}$, the bca $\alpha_{\mathrm{AFF}} \circ t \circ \gamma_{\mathrm{AFF}} : \mathrm{AFF}_n \to \mathrm{AFF}_n$ is algorithmically computable.

We consider single affine assignments $t_a^j \stackrel{\triangle}{\equiv} x_j := \sum_{i=1}^n m_i x_i + b$ and linear Boolean guards $t_b \stackrel{\triangle}{\equiv} \sum_{i=1}^n m_i x_i + b \bowtie 0$, where $m_i, b \in \mathbb{Q}$ and $\bowtie \in \{=, \neq, <, \leq, >, \geq\}$, whose corresponding transfer functions are as follows: for all $Y \in \wp(\mathbb{Q}^n)$,

$$t_a^j(Y) \triangleq \{\vec{v}[\vec{v}_j/v'] \in \mathbb{Q}^n \mid \vec{v} \in Y, v' = \textstyle\sum_{i=1}^n m_i \vec{v}_i + b\}, \quad t_b(Y) \triangleq \{\vec{v} \in Y \mid \textstyle\sum_{i=1}^n m_i \vec{v}_i + b \bowtie 0\}.$$

These transfer functions can be extended to include parallel affine assignments $\vec{x} := M\vec{x} + \vec{b}$, where $M \in \mathbb{Q}^{n \times n}$ is a $n \times n$ matrix and $\vec{b} \in \mathbb{Q}^n$, which performs n parallel single affine assignments, and conjunctive (disjunctive) linear Boolean guards $M\vec{x} + \vec{b} \bowtie 0$, which holds iff, for all (there exists) $j \in [1, n]$, $\sum_{i=1}^n M_{ji} x_i + b_j \bowtie 0$ holds.

Karr gave already in [17, Section 4.2] an algorithm for computing the bca of an affine assignment t_a^j for affine sets represented by kernels of affine transformations. Müller-Olm and Seidl [26] put forward a more efficient algorithm for their representation based on generators. It is also worth remarking that Müller-Olm and Seidl [26, Lemma 2] observe that the bca of t_a^j turns out to be pointwise complete, namely $\mathsf{aff} \circ t_a^j \circ \mathsf{aff} = \mathsf{aff} \circ t_a^j$ holds. Hence, in turn, computability of parallel affine assignments $\vec{x} := M\vec{x} + \vec{b}$ easily follows. [26, Section 4] also shows that the bca of a nondeterministic assignment $t_{a?}^j \stackrel{\triangle}{\equiv} x_j :=?$ is computable, where the corresponding transfer function is defined by: $t_{x_j:=?}(Y) \triangleq \{\vec{v}[\vec{v}_j/v'] \in \mathbb{Q}^n \mid \vec{v} \in Y,\, v' \in \mathbb{Q}\}$. In fact, one can observe [26, Lemma 4] that $\mathsf{aff}(t_{x_j:=?}(\mathsf{aff}(Y))) = \mathsf{aff}(t_{x_j:=0}(\mathsf{aff}(Y))) \sqcup_{\mathrm{AFF}} \mathsf{aff}(t_{x_j:=1}(\mathsf{aff}(Y)))$, so that computing the bca $\mathsf{aff}(t_{x_j:=?}(\mathsf{aff}(Y)))$ is reduced to the lub of the bca's of the transfer functions of the affine assignments $x_j := 0$ and $x_j := 1$.

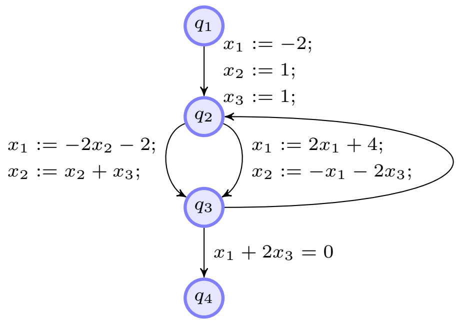
As observed by Karr [17, Section 4.1] (see also [21, Section 5.2.3] for a modern approach), bca's of affine equalities Boolean guards of the shape $t_{b^=} \stackrel{\triangle}{\equiv} \sum_{i=1}^n m_i x_i + b = 0$ are algorithmically computable through the glb of AFF_n, *i.e.*, for all $a \in \mathrm{AFF}_n$, $\alpha_{\mathrm{AFF}}(t_{b^=}(\gamma_{\mathrm{AFF}}(a))) = a \sqcap_{\mathrm{AFF}_n} a_{b^=}$, where $a_{b^=} \in \mathrm{AFF}_n$ denotes the affine set representing the affine equality $\sum_{i=1}^n m_i x_i + b = 0$. For affine inequalities Boolean guards $t_{b^{\neq}} \stackrel{\triangle}{\equiv} \sum_{i=1}^n m_i x_i + b \neq 0$, Karr [17, Section 4.1] defines the following abstract function: $t^\sharp_{b^{\neq}}(a) \triangleq \textbf{if } a \subseteq a_{b^=} \textbf{ then } \bot_{\mathrm{AFF}_n} \textbf{ else } a$, and states that "we must be content with a on the "otherwise" case...a general study of how best to handle decision nodes which are not of the simple form $t_{b^=}$ is *in preparation*", but this document never appeared. Nevertheless, we notice that the above definition of $t^\sharp_{b^{\neq}}$ actually is the bca $\alpha_{\mathrm{AFF}} \circ t_{b^{\neq}} \circ \gamma_{\mathrm{AFF}} = \lambda a.\, \mathsf{aff}(a \cap \neg a_{b^=})$. In fact, let us observe the following fact $(*)$: if $a, a' \in \mathrm{AFF}_n$ then $a' \subsetneq a \Rightarrow \mathsf{aff}(a \cap \neg a') = a$. In fact, we have that $\dim(a') < \dim(a)$ and, in turn, $\dim(\mathsf{aff}(a \cap \neg a')) = \dim(\mathsf{aff}(a \smallsetminus a')) = \dim(a)$ hold, therefore entailing that $\mathsf{aff}(a \cap \neg a') = a$. Thus: (a) if $a_{b^=} \subsetneq a$ then, by $(*)$, $\mathsf{aff}(a \cap \neg a_{b^=}) = a$; (b) if $a_{b^=}$ and a are incomparable then $a \cap a_{b^=} \subsetneq a$, so that, by $(*)$, $\mathsf{aff}(a \cap \neg a_{b^=}) = \mathsf{aff}(a \cap \neg(a \cap a_{b^=})) = a$.

Summing up, as a consequence of Corollary 3.4, the above analysis of bca's in the abstract domain AFF gives us the following result for the class $\mathcal{C}_{\mathrm{AFF}}$ of nondeterministic programs with (possibly parallel) affine assignments, (possibly parallel) nondeterministic assignments and (conjunctive or disjunctive) affine equalities/inequalities guards.

▶ **Theorem 5.1** (Decidability and Synthesis of Inductive Invariants in AFF). *The Monniaux problem* (4) *on* AFF *for programs in* $\mathcal{C}_{\mathrm{AFF}}$*, affine sets of initial states and affine sets of state properties is decidable. Moreover, the algorithm* AINV *of Corollary 3.5 instantiated to* $\mathsf{post}_{\mathcal{P}}$ *for* $\mathcal{P} \in \mathcal{C}_{\mathrm{AFF}}$*, synthesizes the least inductive invariant of* $\mathcal{P}$ *in* AFF*, when this exists.*

To the best of our knowledge, the literature provides no algorithm for computing the bca of further linear Boolean guards $\sum_{i=1}^{n} m_i x_i + b \bowtie 0$, with $\bowtie \in \{<, \leq, \}$. We conjecture that at least some of these bca's are algorithmically computable.

▶ **Example 5.2.** Consider the following nondeterministic program $\mathcal{R}$ with $\Sigma_0 = \{q_1\} \times \mathbb{Q}^3 \in \mathrm{A{\scriptsize FF}}^{|Q|}$ and the property $P^\sharp = \big\langle \langle q_1, \top\rangle, \langle q_2, \top\rangle, \langle q_3, \top\rangle, \langle q_4, x_1 + x_2 + 1 = 0\rangle \big\rangle \in \mathrm{A{\scriptsize FF}}^{|Q|}$.



The algorithm AI{\scriptsize NV} of Corollary 3.5 yields the following sequence of $I^j \in \mathrm{A{\scriptsize FF}}^{|Q|}$:

$$\begin{aligned}
I^0 &= \dot{\alpha}_{\mathrm{A{\scriptsize FF}}}(\Sigma_0) = \big\langle \langle q_1, \top\rangle, \langle q_2, \bot\rangle, \langle q_3, \bot\rangle, \langle q_4, \bot\rangle \big\rangle \\
I^1 &= \big\langle \langle q_1, \top\rangle, \langle q_2, x_1 + 2 = 0 \wedge x_2 - 1 = 0 \wedge x_3 - 1 = 0\rangle, \langle q_3, \bot\rangle, \langle q_4, \bot\rangle \big\rangle \\
I^2 &= \big\langle \langle q_1, \top\rangle, \langle q_2, x_1 + 2 = 0 \wedge x_2 - 1 = 0 \wedge x_3 - 1 = 0\rangle, \\
&\qquad\qquad \langle q_3, x_1 + 2x_2 = 0 \wedge x_3 = 1\rangle, \langle q_4, \bot\rangle \big\rangle \\
I^3 &= \big\langle \langle q_1, \top\rangle, \langle q_2, x_1 + 2x_2 = 0 \wedge x_3 = 1\rangle, \langle q_3, x_1 + 2x_2 = 0 \wedge x_3 = 1\rangle, \\
&\qquad\qquad \langle q_4, x_1 + x_2 + 1 = 0 \wedge x_3 = 1, \bot\rangle \big\rangle = \dot{\alpha}_{\mathrm{A{\scriptsize FF}}}(\mathsf{post}_{\mathcal{R}}(\dot{\gamma}_{\mathrm{A{\scriptsize FF}}}(I^3))) \dot{\leq}_{\mathrm{A{\scriptsize FF}}} P^\sharp
\end{aligned}$$

The output I^3 is the analysis of $\mathcal{R}$ with the bca's of its transfer functions in A{\scriptsize FF}, *i.e.*, it is the least inductive invariant in A{\scriptsize FF} which allows us to prove that $P^\sharp$ holds. ⌟

5.1.1 Relationship with Müller-Olm and Seidl [26]

Müller-Olm and Seidl [26] implicitly show that the transfer functions of affine assignments t_a and of nondeterministic assignments $t_{x_j:=?}$ are pointwise complete. In fact, [26, Lemma 2] shows that for all $X \in \wp(\mathbb{Q}^n)$, $t_a(\mathsf{aff}(X)) = \mathsf{aff}(t_a(X))$, from which we easily obtain: $\mathsf{aff}(t_a(X)) = \mathsf{aff}(t_a(\mathsf{aff}(X)))$ and

$$\begin{aligned}
&\mathsf{aff}(t_{x_j:=?}(X)) = \mathsf{aff}(\cup_{z\in\mathbb{Q}} t_{x_j:=z}(X)) = \mathsf{aff}(\cup_{z\in\mathbb{Q}}\, \mathsf{aff}(t_{x_j:=z}(X))) = \\
&\qquad = \mathsf{aff}(\cup_{z\in\mathbb{Q}}\, \mathsf{aff}(t_{x_j:=z}(\mathsf{aff}(X))) = \mathsf{aff}(\cup_{z\in\mathbb{Q}} t_{x_j:=z}(\mathsf{aff}(X)) = \mathsf{aff}(t_{x_j:=?}(\mathsf{aff}(X)))
\end{aligned}$$

Thus, since pointwise completeness entails fixpoint completeness (cf. Section 4.1), for all unguarded programs $\mathcal{P}$ with affine and nondeterministic assignments, $\dot{\alpha}_{\mathrm{A{\scriptsize FF}}}(\mathsf{lfp}(\lambda\vec{X}.\Sigma_0 \cup \mathsf{post}_{\mathcal{P}}(\vec{X}))) = \mathsf{lfp}(\lambda\vec{a}.\dot{\alpha}_{\mathrm{A{\scriptsize FF}}}(\Sigma_0)\ \dot{\sqcup}_{\mathrm{A{\scriptsize FF}}}\ \dot{\alpha}_{\mathrm{A{\scriptsize FF}}}(\mathsf{post}_{\mathcal{P}}(\dot{\gamma}_{\mathrm{A{\scriptsize FF}}}(\vec{a}))))$ holds, which is the reason why "Karr's algorithm is precise for affine programs, *i.e.*, computes not just some but all valid affine relations" [26, Section 1]. However, fixpoint completeness is lost as soon as affine equality or inequality guards are included, although these programs still belong to $\mathcal{C}_{\mathrm{A{\scriptsize FF}}}$, because these guards are not pointwise complete: for example, we have that $\mathsf{aff}(t_{x_1=0}(\{(1,0),(-1,0)\})) = \mathsf{aff}(\varnothing) = \varnothing$, whereas $\mathsf{aff}(t_{x_1=0}(\mathsf{aff}(\{(1,0),(-1,0)\}))) = \mathsf{aff}(t_{x_1=0}(x_2 = 0)) = \mathsf{aff}(\{(0,0)\}) = \{(0,0)\}$. Müller-Olm and Seidl [26, Section 7] also prove that as soon as affine equality guards are added to nondeterministic affine programs it becomes undecidable whether a given affine equality holds in some program point or not. It is therefore worth remarking that this undecidability does not prevent the decidability result of Theorem 5.1 on the existence (and synthesis) of *inductive* affine equalities proving a given affine equality, since these two decision problems are different. In fact, Müller-Olm and Seidl [26, Section 7] prove that $\forall f \in \mathcal{C}_{\mathrm{AFF}}.\forall a \in \mathrm{AFF}\,.\, \alpha_{\mathrm{AFF}}(\mathsf{lfp}(f)) \leq^?_{\mathrm{AFF}} a$ is an undecidable problem, while Theorem 5.1 shows that $\forall f \in \mathcal{C}_{\mathrm{AFF}}.\forall a \in \mathrm{AFF}\,.\, \mathsf{lfp}(\alpha_{\mathrm{AFF}} f \gamma_{\mathrm{AFF}}) \leq^?_{\mathrm{AFF}} a$ is decidable, and, by Theorem 4.1 (b), these are equivalent problems iff (where "if" is obvious) fixpoint completeness $\forall f \in \mathcal{C}_{\mathrm{AFF}}.\alpha_{\mathrm{AFF}}(\mathsf{lfp}(f)) = \mathsf{lfp}(\alpha_{\mathrm{AFF}} f \gamma_{\mathrm{AFF}})$ holds.

▶ **Example 5.3.** Consider the following deterministic program $\mathcal{P} \in \mathcal{C}_{\mathrm{AFF}}$:

$$\mathcal{P} \;\equiv\; x_1 := 0;\; x_2 := 3;\; \textbf{while } (x_1 \neq 3) \textbf{ do } \{x_1 := x_1 + 2;\; x_2 := x_2 - 2\}$$

where q_i and q_o denote, resp., its entry and exit program points and $\Sigma_0 = \{q_i\} \times \mathbb{Q}^2$ is the set of initial states. Then, we have that the affine abstraction of the reachable states at the exit point q_0 is

$$\begin{aligned}\alpha_{\mathrm{AFF}}(\pi_{q_o}(\mathsf{lfp}(\lambda \vec{X}.\Sigma_0 \cup \mathsf{post}_{\mathcal{P}}(\vec{X})))) &= \alpha_{\mathrm{AFF}}(\{(0+2n, 3-2n) \mid n \in \mathbb{N}\} \cap \langle x_1 = 3\rangle)\\ &= \alpha_{\mathrm{AFF}}(\varnothing) = \bot_{\mathrm{AFF}}\end{aligned}$$

while the algorithm AInv of Corollary 3.5 using the best correct approximations of the transfer functions of $b^{=} \equiv (x_1 = 3)$ and $b^{\neq} \equiv (x_1 \neq 3)$ gives:

$$\begin{aligned}\pi_{q_o}(\mathsf{lfp}(\lambda \vec{a} \in \mathrm{AFF}^{|Q|}.\, \dot{\alpha_{\mathrm{AFF}}}(\Sigma_0) \,\dot{\sqcup}_{\mathrm{AFF}}\, \dot{\alpha_{\mathrm{AFF}}}(\mathsf{post}_{\mathcal{P}}(\dot{\gamma_{\mathrm{AFF}}}(\vec{a}))))) &= \langle x_1 + x_2 = 3\rangle \sqcap_{\mathrm{AFF}} \langle x_1 = 3\rangle\\ &= \langle x_1 = 3 \wedge x_2 = 0\rangle \not\leq_{\mathrm{AFF}} \bot_{\mathrm{AFF}}\end{aligned}$$

The least inductive invariant $\langle x_1 = 3 \wedge x_2 = 0\rangle$ in Aff does not entail $\bot_{\mathrm{AFF}}$, namely, it cannot prove that q_o is unreachable, therefore showing that the two aforementioned decision problems are not equivalent for programs ranging in $\mathcal{C}_{\mathrm{AFF}}$. ⌟

6 Co-Inductive Synthesis of Abstract Inductive Invariants

In this section we design a synthesis algorithm which, by generalizing an algorithm by Padon et al. [28], outputs the *greatest* abstract inductive invariant ranging in some abstract domain, when this exists. This algorithm is obtained by dualizing the procedure AInv in Corollary 3.5 to a co-inductive greatest fixpoint computation and requires that the abstract domain is equipped with a suitable well-quasiorder relation. Let us recall that a quasiordered set $D_{\leq}$ is a well-quasiordered set (wqoset), and $\leq$ is called well-quasiorder (wqo), when for every countably infinite sequence of elements $\{x_i\}_{i\in\mathbb{N}}$ in D there exist $i, j \in \mathbb{N}$ such that $i < j$ and $x_i \leq x_j$. Equivalently, D is a wqoset iff D is DCC (also called well-founded) and D has no infinite antichain (*i.e.*, a subset whose distinct elements are pairwise incomparable).

In the following, we will leverage on the closure operator approach for defining abstract domains, which, as recalled in Section 2.2, is completely equivalent to Galois connections and particularly suitable for reasoning on abstract domains independently from their representation. Let $\mathcal{T} = \langle \Sigma, \tau\rangle$ be a transition system whose successor transformer is denoted by post. Padon et al. [28] consider abstract invariants ranging in a set (of semantics of logical formulae) $L \subseteq \wp(\Sigma)$ and assume (in [28, Theorem 4.2]) that $\langle L, \subseteq\rangle$ is closed under finite intersections (*i.e.*, logical conjunctions). Accordingly to Assumption 3.1, we ask that $\langle L, \subseteq\rangle$ satisfies the requirement of being an abstract domain of $\langle \wp(\Sigma), \subseteq\rangle$, which corresponds to ask that $\langle L, \subseteq\rangle$ is closed under arbitrary, rather than finite, intersections. Thus, L is the image of an upper closure $\check{\mu}_L \in \mathrm{uco}(\wp(\Sigma)_\subseteq)$ defined by: $\check{\mu}_L(X) \triangleq \cap\{\phi \in L \mid X \subseteq \phi\}$. The three key definitions and related assumptions of the synthesis algorithm defined in [28, Theorem 4.2] concern a quasiorder $\sqsubseteq_L \subseteq \Sigma \times \Sigma$ between states, a function $Av_L : \Sigma \to \wp(\Sigma)$ called Avoid, and an abstract transition relation $\tau^L \subseteq \Sigma \times \Sigma$, and are as follows:

(1) $s \sqsubseteq_L s' \overset{\Delta}{\Leftrightarrow} \forall \phi \in L.\, s' \in \phi \Rightarrow s \in \phi$ Assumption (A_1): $\langle \Sigma, \sqsubseteq_L \rangle$ is a wqoset

(2) $Av_L(s) \triangleq \cup\{\phi \in L \mid \phi \subseteq \neg\{s\}\}$ Assumption (A_2): $\forall s \in \Sigma.\ Av_L(s) \in L$

(3) $(s, s') \in \tau^L \overset{\Delta}{\Leftrightarrow} (s, s') \in \tau \vee s' \sqsubseteq_L s$ Abstract Transition System $\mathcal{T}^L \triangleq \langle \Sigma, \tau^L \rangle$

Correspondingly, we define the down-closure $\delta_L : \wp(\Sigma) \to \wp(\Sigma)$ of the quasiorder $\sqsubseteq_L$, we lift $Av_L : \wp(\Sigma) \to \wp(\Sigma)$ to sets of states and we define the successor transformer $\mathsf{post}^L : \wp(\Sigma) \to \wp(\Sigma)$ of $\mathcal{T}^L$ as follows:

(1) $\delta_L(X) \triangleq \{s \in \Sigma \mid \exists s' \in X.\, s \sqsubseteq_L s'\}$ Down-closure of $\sqsubseteq_L$

(2) $Av_L(X) \triangleq \cup\{\phi \in L \mid \phi \subseteq \neg X\}$ Av_L acts on any set X of states

(3) $\mathsf{post}^L(X) \triangleq \mathsf{post}(X) \cup \delta_L(X)$ Successor transformer of $\mathcal{T}^L$

▶ **Lemma 6.1.** *The following properties hold:*
(a) $s \sqsubseteq_L s'$ *iff* $\check{\mu}_L(\{s\}) \subseteq \check{\mu}_L(\{s'\})$.
(b) (A_1) *holds iff* $\langle L, \subseteq \rangle$ *is a well-quasi order.*
(c) (A_2) *holds iff* L *is closed under arbitrary unions.*
(d) *If* (A_2) *holds then* $\delta_L = \check{\mu}_L$ *and both are additive ucos (and* $\delta_L(\phi) = \phi \Leftrightarrow \phi \in L$*).*
(e) *For all* $\Sigma_0 \in \wp(\Sigma)$, $\mathsf{lfp}(\lambda X.\Sigma_0 \cup \mathsf{post}^L(X)) = \mathsf{lfp}(\lambda X.\check{\mu}_L(\Sigma_0 \cup \mathsf{post}(X)))$.

In particular, Lemma 6.1 (e) states that the reachable states in $\mathcal{T}^L$ coincide with the reachable states of the abstract transition system $\mathcal{T}^{\check{\mu}_L} \triangleq \langle \Sigma, \check{\mu}_L \circ \mathsf{post} \rangle$ obtained by considering the best correct approximation of post in the abstract domain $\check{\mu}_L$. As a direct consequence of the abstract inductive invariant principle Lemma 3.2 (a), we obtain the following characterization of the abstract inductive invariants ranging in L of the transition system $\mathcal{T}$ which relies on the reachable states of its best abstraction $\mathcal{T}^{\check{\mu}_L}$ in the domain $\check{\mu}_L$.

▶ **Corollary 6.2.** *Let* $\Sigma_0, P \in \wp(\Sigma)$*. Then,* $\exists \phi \in L.\, \Sigma_0 \subseteq \phi \wedge \mathsf{post}(\phi) \subseteq \phi \wedge \phi \subseteq P$ *iff* $\mathrm{Reach}[\mathcal{T}^{\check{\mu}_L}, \Sigma_0] \subseteq P$*.*

6.1 Co-Inductive Invariants

Following [28], in the following we make assumption (A_2), that is, by Lemma 6.1 (c), we assume that $L \subseteq \wp(\Sigma)$ is closed under arbitrary unions. This means that $\check{\mu}_L$ is an additive uco on $\wp(\Sigma)_\subseteq$, *i.e.*, in abstract interpretation terminology, $\check{\mu}_L$ is a *disjunctive* abstract domain whose abstract lub does not lose precision (see, *e.g.*, [21, Section 6.3]). Furthermore, we also have that L is the image of a co-additive (*i.e.*, preserving arbitrary intersections) lower closure $\hat{\mu}_L : \wp(\Sigma) \to \wp(\Sigma)$ defined by $\hat{\mu}_L(X) \triangleq \cup\{\phi \in L \mid \phi \subseteq X\}$. It turns out that the uco $\check{\mu}_L$ is adjoint to the lco $\hat{\mu}_L$ namely, $\check{\mu}_L(X) \subseteq Y \Leftrightarrow X \subseteq \hat{\mu}_L(Y)$ holds. In fact, if $\check{\mu}_L(X) \subseteq Y$ then, by applying $\hat{\mu}_L$, $X \subseteq \check{\mu}_L(X) = \hat{\mu}_L(\check{\mu}_L(X)) \subseteq \hat{\mu}_L(Y)$; the converse is dual.

As observed by Cousot [1, Theorem 4], the inductive invariant principle (2) can be dualized when f admits right-adjoint $\widetilde{f} : C \to C$ (this happens iff f is additive): in this case,

$$\mathsf{lfp}(\lambda x.c \vee f(x)) \leq c' \Leftrightarrow c \leq \mathsf{gfp}(\lambda x.c' \wedge \widetilde{f}(x)) \tag{7}$$

holds and one obtains a *co-inductive* invariant principle:

$$c \leq \mathsf{gfp}(\lambda x.\widetilde{f}(x) \wedge c') \Leftrightarrow \exists j \in C.\, c \leq j \wedge j \leq \widetilde{f}(j) \wedge j \leq c' \tag{8}$$

One such $j \in C$ is therefore called a *co-inductive invariant* of f for $\langle c, c' \rangle$. The co-inductive invariant proof method (8) can be applied to safety verification of any transition system $\mathcal{T}$ because post is additive and therefore it always admits right adjoint $\widetilde{\mathsf{pre}}$ (cf. Section 2.3). Hence, we obtain that $\mathsf{lfp}(\lambda X.\Sigma_0 \cup \mathsf{post}(X)) \subseteq P$ iff $\Sigma_0 \subseteq \mathsf{gfp}(\lambda X.\, \widetilde{\mathsf{pre}}(X) \cap P)$ iff there exists a co-inductive invariant for $\widetilde{\mathsf{pre}}$ for $\langle \Sigma_0, P \rangle$. By (2) and (8), it turns out that I is an inductive invariant of post for $\langle \Sigma_0, P \rangle$ iff I is a co-inductive invariant of $\widetilde{\mathsf{pre}}$ for $\langle \Sigma_0, P \rangle$. Also, while $\mathsf{lfp}(\lambda X.\Sigma_0 \cup \mathsf{post}(X))$ is the least, *i.e.* logically strongest, inductive invariant, we have that $\mathsf{gfp}(\lambda X.\, \widetilde{\mathsf{pre}}(X) \cap P)$ is the greatest, *i.e.* logically weakest, inductive invariant [1, Theorem 6].

We show how the co-inductive invariant principle (8) applied to the best abstract transition system $\mathcal{T}^{\check{\mu}_L} = (\Sigma, \check{\mu}_L \circ \mathsf{post}_{\mathcal{T}})$ provides exactly the synthesis algorithm by Padon et al. [28, Algorithm 1]. In order to do this, we first give the following alternative characterization of the reachable states of $\mathcal{T}^{\check{\mu}_L}$.

▶ **Lemma 6.3.** $\mathsf{lfp}(\lambda X.\Sigma_0 \cup \check{\mu}_L(\mathsf{post}(X))) = \mathsf{lfp}(\lambda X.\Sigma_0 \cup \mathsf{post}(\check{\mu}_L(X)) \cup \check{\mu}_L(X))$.

Consequently, $\mathsf{lfp}(\lambda X.\Sigma_0 \cup \check{\mu}_L(\mathsf{post}(X))) \subseteq P \Leftrightarrow \mathsf{lfp}(\lambda X.\Sigma_0 \cup \mathsf{post}(\check{\mu}_L(X)) \cup \check{\mu}_L(X)) \subseteq P$ holds. Since $\lambda X.\, \mathsf{post}(\check{\mu}_L(X)) \cup \check{\mu}_L(X)$ is additive, we can apply the co-inductive invariant principle (8) by considering its adjoint function, which is as follows:

$$\begin{aligned} &\mathsf{post}(\check{\mu}_L(X)) \cup \check{\mu}_L(X) \subseteq Y \Leftrightarrow \mathsf{post}(\check{\mu}_L(X)) \subseteq Y \wedge \check{\mu}_L(X) \subseteq Y \Leftrightarrow \\ X \subseteq \hat{\mu}_L(&\widetilde{\mathsf{pre}}(Y)) \wedge X \subseteq \hat{\mu}_L(Y) \Leftrightarrow X \subseteq \hat{\mu}_L(\widetilde{\mathsf{pre}}(Y)) \cap \hat{\mu}_L(Y). \end{aligned}$$

Thus, by Lemma 6.3 and (7), we obtain: $\mathsf{lfp}(\lambda X.\check{\mu}_L(\Sigma_0) \cup \check{\mu}_L(\mathsf{post}(X))) \subseteq P$ iff $\check{\mu}_L(\Sigma_0) \subseteq \mathsf{gfp}(\lambda X.\hat{\mu}_L(\widetilde{\mathsf{pre}}(X) \cap X \cap P))$ iff $\Sigma_0 \subseteq \mathsf{gfp}(\lambda X.\hat{\mu}_L(\widetilde{\mathsf{pre}}(X) \cap X \cap P))$, and, in turn, by the abstract inductive invariant principle (Lemma 3.2 (a) for ucos) applied to $\check{\mu}_L \circ (\lambda X.\Sigma_0 \cup \mathsf{post}(X)) = \lambda X.\check{\mu}_L(\Sigma_0) \cup \check{\mu}_L(\mathsf{post}(X))$ we get:

$$\exists \phi \in L.\Sigma_0 \subseteq \phi \wedge \mathsf{post}(\phi) \subseteq \phi \wedge \phi \subseteq P \Leftrightarrow \Sigma_0 \subseteq \mathsf{gfp}(\lambda X.\hat{\mu}_L(\widetilde{\mathsf{pre}}(X) \cap X \cap P)).$$

This leads us to use the algorithm introduced by Cousot [1, Algorithm 2] which synthesizes an inductive invariant by applying Knaster-Tarski theorem to compute the iterates of the greatest fixpoint of $\lambda X.\hat{\mu}_L(\widetilde{\mathsf{pre}}(X) \cap X \cap P)$ as long as the current iterate I_1 contains Σ_0:

Algorithm 1 Co-inductive backward synthesis of abstract inductive invariants.

$I_1 := \Sigma$;
while $\Sigma_0 \subseteq I_1$ **do** // Loop invariant: $I_1 \in L$
  if $(I_1 = \hat{\mu}_L(\widetilde{\mathsf{pre}}(I_1) \cap I_1 \cap P))$ **then return** I_1 *is an inductive invariant in* L;
  $I_1 := I_1 \cap \hat{\mu}_L(\widetilde{\mathsf{pre}}(I_1) \cap I_1 \cap P)$;
return *no inductive invariant in* L;

Since, $\widetilde{\mathsf{pre}}$ is computable and, by Lemma 6.1 (b), $\langle \hat{\mu}_L, \supseteq \rangle = \langle L, \supseteq \rangle$ is a wqo, we immediately obtain that Algorithm 1 is correct and terminating. Furthermore, if Algorithm 1 outputs an inductive invariant I_1 proving the property P then I_1 is the *greatest* inductive invariant in L proving P. It turns out that Algorithm 1 exactly coincides with the synthesis algorithm by Padon et al. [28], which is replicated here as Algorithm 2.

▶ **Theorem 6.4.** *Algorithm* 1 = *Algorithm* 2.

This shows that Algorithm 2 in [28] for a disjunctive GC-based abstract domain A amounts to a backward (*i.e.*, propagating $\widetilde{\mathsf{pre}}$) static analysis using the best correct approximations in A of the transfer functions, as long as the ordering of A guarantees its termination, *e.g.*, because A is well-founded.

Algorithm 2 Inductive invariant algorithm by [28].

$I_2 := \Sigma$;
while I_2 *is not an inductive invariant* **do** // Loop invariant: $I_2 \in L$
  if $\Sigma_0 \not\subseteq I_2$ **then return** *no inductive invariant in* L;
  choose $s \in \Sigma$ *as a counterexample to inductiveness of* I_2;
  $I_2 := I_2 \cap Av_L(s)$;
return I_2 *is an inductive invariant in* L;

6.2 Backward and Forward Algorithms

Algorithm 1 is backward because it applies $\widetilde{\mathsf{pre}}$, for termination it requires that the abstract domain $\langle \check{\mu}_L, \subseteq\rangle$ is DCC and it turns out to be the dual of the forward algorithm AInv provided by Corollary 3.5 for post and requiring that $\langle \check{\mu}_L, \subseteq\rangle$ is ACC. A different gfp-based *forward* algorithm can be designed by observing (as in [6, Section 3]) that $\mathsf{lfp}(\lambda X.\Sigma_0 \cup \mathsf{post}(X)) \subseteq P$ iff $\mathsf{lfp}(\lambda X.\neg P \cup \mathsf{pre}(X)) \subseteq \neg\Sigma_0$. Here, the dualization provided by the equivalence (7) yields:

$$\mathsf{lfp}(\lambda X.\neg P \cup \check{\mu}_L(\mathsf{pre}(X))) \subseteq \neg\Sigma_0 \Leftrightarrow \neg P \subseteq \mathsf{gfp}(\lambda X.\hat{\mu}_L(\widetilde{\mathsf{post}}(X) \cap X \cap \neg\Sigma_0))$$

and induces the following *co-inductive forward algorithm* which relies on the state transformer $\widetilde{\mathsf{post}}$ and is terminating when $\langle \check{\mu}_L, \subseteq\rangle$ is assumed to be DCC:

Algorithm 3 Co-inductive forward synthesis of abstract inductive invariants.

$I := \Sigma$;
while $\neg P \subseteq I$ **do** // Loop invariant: $I \in L$
  if $(I = \hat{\mu}_L(\widetilde{\mathsf{post}}(I) \cap I \cap \neg\Sigma_0))$ **then return** I *is an inductive invariant in* L;
  $I := I \cap \hat{\mu}_L(\widetilde{\mathsf{post}}(I) \cap I \cap \neg\Sigma_0)$;
return *no inductive invariant in* L;

Furthermore, by dualizing the technique described by Cousot and Cousot [6, Section 4.3] for post and pre, one could also design a more efficient combined forward/backward synthesis algorithm which simultaneously make backward, by $\widetilde{\mathsf{pre}}$, and forward, by $\widetilde{\mathsf{post}}$, steps.

7 Future Work

As hinted by Monniaux [24], results of undecidability to the question (3) for some abstract domain A display a foundational trait since they "vindicate" (often years of intense) research on precise and efficient algorithms for *approximate* static program analysis on A. To the best of our knowledge, few undecidability results are available: an undecidability result by Monniaux [24, Theorem 1] for convex polyhedra [7] and by Fijalkow et al. [12, Theorem 1] for semilinear sets, *i.e.* finite unions of convex polyhedra. However, convex polyhedra and semilinear sets cannot be defined by a Galois connection and therefore do not satisfy our Assumption 3.1. As future work we plan to investigate whether the abstract inductive invariant principle could be exploited to provide a reduction of the undecidability of the question (3) for abstract domains which satisfy Assumption 3.1 and, in view of the characterization of fixpoint completeness in Section 4.2, for transfer functions which are not fixpoint complete.

We also plan to study whether complete abstractions can play a role in the decidability result by Hrushovski et al. [16] on the computation of the strongest polynomial invariant of an affine program. This hard result relies on the Zariski closure, which is continuous for affine functions and is pointwise complete for the transfer functions of affine programs.

Thus, fixpoint completeness for affine programs holds, and one could investigate whether the algorithm in [16] may be viewed as a least fixpoint computation of a best correct approximation on the Zariski abstraction.

References

1. Patrick Cousot. Partial completeness of abstract fixpoint checking. In *Proceedings of the 4th International Symposium on Abstraction, Reformulation, and Approximation (SARA'02)*, volume 1864 of *Lecture Notes in Computer Science*, pages 1–25. Springer-Verlag, 2000. doi:10.1007/3-540-44914-0_1.
2. Patrick Cousot. On fixpoint/iteration/variant induction principles for proving total correctness of programs with denotational semantics. In *Proc. 29th International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR 2019)*, volume 12042 of *LNCS*, pages 3–18. Springer, 2019. doi:10.1007/978-3-030-45260-5_1.
3. Patrick Cousot and Radhia Cousot. 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)*, pages 238–252. ACM Press, 1977. doi:10.1145/512950.512973.
4. Patrick Cousot and Radhia Cousot. Systematic design of program analysis frameworks. In *Proceedings of the 6th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL'79)*, pages 269–282, New York, NY, USA, 1979. ACM. doi:10.1145/567752.567778.
5. Patrick Cousot and Radhia Cousot. Induction principles for proving invariance properties of programs. In D. Néel, editor, *Tools & Notions for Program Construction: an Advanced Course*, pages 75–119. Cambridge University Press, Cambridge, UK, August 1982.
6. Patrick Cousot and Radhia Cousot. Refining model checking by abstract interpretation. *Automated Software Engineering*, 6(1):69–95, 1999. doi:10.1023/A:1008649901864.
7. Patrick Cousot and Nicolas Halbwachs. 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)*, pages 84–96. ACM, 1978. doi:10.1145/512760.512770.
8. Isil Dillig, Thomas Dillig, Boyang Li, and Kenneth L. McMillan. Inductive invariant generation via abductive inference. In *Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications, (OOPSLA'13)*, pages 443–456. ACM, 2013. doi:10.1145/2509136.2509511.
9. Yotam M. Y. Feldman, Neil Immerman, Mooly Sagiv, and Sharon Shoham. Complexity and information in invariant inference. *Proc. ACM Program. Lang.*, 4(POPL), 2019. doi:10.1145/3371073.
10. Yotam M. Y. Feldman, Oded Padon, Neil Immerman, Mooly Sagiv, and Sharon Shoham. Bounded Quantifier Instantiation for Checking Inductive Invariants. *Logical Methods in Computer Science*, Volume 15, Issue 3, 2019. doi:10.23638/LMCS-15(3:18)2019.
11. Yotam M. Y. Feldman, James R. Wilcox, Sharon Shoham, and Mooly Sagiv. Inferring inductive invariants from phase structures. In *Proc. 31st International Conference on Computer Aided Verification (CAV'19)*, volume 11562 of *Lecture Notes in Computer Science*, pages 405–425. Springer, 2019. doi:10.1007/978-3-030-25543-5_23.
12. Nathanaël Fijalkow, Engel Lefaucheux, Pierre Ohlmann, Joël Ouaknine, Amaury Pouly, and James Worrell. On the Monniaux problem in abstract interpretation. In *Proc. 26th International Static Analysis Symposium (SAS'19)*, volume 11822 of *Lecture Notes in Computer Science*, pages 162–180. Springer, 2019. doi:10.1007/978-3-030-32304-2_9.
13. Robert W. Floyd. Assigning meanings to programs. *Proceedings of Symposium on Applied Mathematics*, 19:19–32, 1967.
14. Roberto Giacobazzi, Francesco Ranzato, and Francesca Scozzari. Complete abstract interpretations made constructive. In *Proceedings of the 23rd International Symposium on Mathematical Foundations of Computer Science (MFCS'98)*, volume 1450 of *Lecture Notes in Computer Science*, pages 366–377. Springer, 1998. doi:10.1007/BFb0055786.

15 Roberto Giacobazzi, Francesco Ranzato, and Francesca Scozzari. Making abstract interpretations complete. *J. ACM*, 47(2):361–416, 2000. doi:10.1145/333979.333989.

16 Ehud Hrushovski, Joël Ouaknine, Amaury Pouly, and James Worrell. Polynomial invariants for affine programs. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, (LICS'18)*, pages 530–539. ACM, 2018. doi:10.1145/3209108.3209142.

17 Michael Karr. Affine relationships among variables of a program. *Acta Inf.*, 6:133–151, 1976.

18 Zachary Kincaid, John Cyphert, Jason Breck, and Thomas Reps. Non-linear reasoning for invariant synthesis. *Proc. ACM Program. Lang.*, 2(POPL), 2018. doi:10.1145/3158142.

19 K. Rustan M. Leino. Dafny: An automatic program verifier for functional correctness. In *Proc. International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-16), Revised Selected Papers*, volume 6355 of *Lecture Notes in Computer Science*, pages 348–370. Springer, 2010. doi:10.1007/978-3-642-17511-4_20.

20 Zohar Manna, Stephen Nes, and Jean Vuillemin. Inductive methods for proving properties of programs. *Commun. ACM*, 16(8):491–502, 1973. doi:10.1145/355609.362336.

21 Antoine Miné. Tutorial on static inference of numeric invariants by abstract interpretation. *Foundations and Trends in Programming Languages*, 4(3-4):120–372, 2017. doi:10.1561/2500000034.

22 Antoine Miné, Jason Breck, and Thomas W. Reps. An algorithm inspired by constraint solvers to infer inductive invariants in numeric programs. In *Proc. 25th European Symposium on Programming (ESOP'16)*, volume 9632 of *Lecture Notes in Computer Science*, pages 560–588. Springer, 2016. doi:10.1007/978-3-662-49498-1_22.

23 David Monniaux. On the decidability of the existence of polyhedral invariants in transition systems. *arXiv CoRR*, abs/1709.04382, 2017. arXiv:1709.04382.

24 David Monniaux. On the decidability of the existence of polyhedral invariants in transition systems. *Acta Inf.*, 56(4):385–389, 2019. doi:10.1007/s00236-018-0324-y.

25 Christian Müller, Helmut Seidl, and Eugen Zalinescu. Inductive invariants for noninterference in multi-agent workflows. In *Proc. 31st IEEE Computer Security Foundations Symposium (CSF'18)*, pages 247–261. IEEE Computer Society, 2018. doi:10.1109/CSF.2018.00025.

26 Markus Müller-Olm and Helmut Seidl. A note on Karr's algorithm. In *Proceedings 31st International Colloquium on Automata, Languages and Programming (ICALP'04)*, volume 3142 of *Lecture Notes in Computer Science*, pages 1016–1028. Springer, 2004. doi:10.1007/978-3-540-27836-8_85.

27 Peter Naur. Proof of algorithms by general snapshots. *BIT Numerical Mathematics*, 6(4):310–316, 1966. doi:10.1007/BF01966091.

28 Oded Padon, Neil Immerman, Sharon Shoham, Aleksandr Karbyshev, and Mooly Sagiv. Decidability of inferring inductive invariants. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'16)*, pages 217–231. ACM, 2016. doi:10.1145/2837614.2837640.

29 David Park. Fixpoint induction and proofs of program properties. *Machine Intelligence*, 5, 1969.

30 David Park. On the semantics of fair parallelism. In Dines Bjørner, editor, *Abstract Software Specifications*, pages 504–526. Springer Berlin Heidelberg, 1980.

31 Sharon Shoham. Undecidability of inferring linear integer invariants. *arXiv CoRR*, abs/1812.01069, 2018. arXiv:1812.01069.

32 A. Thakur, A. Lal, J. Lim, and T. Reps. PostHat and all that: Automating abstract interpretation. *Electronic Notes in Theoretical Computer Science*, 311:15–32, 2015. Fourth Workshop on Tools for Automatic Program Analysis (TAPAS 2013). doi:10.1016/j.entcs.2015.02.003.

A Appendix

A.1 When Safety = Abstract Invariance?

Padon et al. [28, Section 9] in their investigation on the decidability of inferring inductive invariants state that "*Usually completeness for abstract interpretation means that the abstract domain is precise enough to prove all interesting safety properties, e.g.,* [15]*. In our terms, this means that* SAFE = INV*, that is, that all safe programs have an inductive invariant expressible in the abstract domain.*" As a by-product of the results in Section 4.2, we are able to give a formal justification and statement of this informal characterization of completeness.

Let $\mathcal{F} \subseteq C \to C$ be a class of monotonic functions, $\mathcal{S} \subseteq C$ be some set of safety properties and $\mathcal{A} \subseteq C$ be an abstract domain of program properties. Let us define:

$$\textsc{safe}[\mathcal{F},\mathcal{S}] \triangleq \{\langle f,s\rangle \in \mathcal{F}\times\mathcal{S} \mid \mathsf{lfp}(f) \leq_C s\}$$
$$\textsc{inv}[\mathcal{F},\mathcal{S},\mathcal{A}] \triangleq \{\langle f,s\rangle \in \mathcal{F}\times\mathcal{S} \mid \exists a \in \mathcal{A}.\, fa \leq_C a \wedge a \leq_C s\}$$

so that in our model $\textsc{safe}[\mathcal{F},\mathcal{S}]$ and $\textsc{inv}[\mathcal{F},\mathcal{S},\mathcal{A}]$ play the role of, resp., "safe programs" and "programs having an inductive invariant expressible in $\mathcal{A}$". As a consequence of Theorem 4.1, we derive the following characterization.

▶ **Corollary A.1.** *Assume that $\mathcal{A}$ satisfies Assumption 3.1 for some GI $\langle C,\alpha,\gamma,\mathcal{A}\rangle$.*
(a) *Assume that $\mathcal{S} \subseteq \mathcal{A}$. Then, $\textsc{safe}[\mathcal{F},\mathcal{S}] = \textsc{inv}[\mathcal{F},\mathcal{S},\mathcal{A}]$ iff $\forall f \in \mathcal{F}.\alpha(\mathsf{lfp}(f)) = \mathsf{lfp}(\alpha f\gamma)$.*
(b) *$\textsc{safe}[\mathcal{F},\mathcal{S}] = \textsc{inv}[\mathcal{F},\mathcal{S},\mathcal{A}]$ iff $\forall f \in \mathcal{F}.\, \mathsf{lfp}(f) = \gamma(\mathsf{lfp}(\alpha f\gamma))$.*

Proof. Point (a) follows by Theorem 4.1 (a), since $\mathcal{S} \subseteq \mathcal{A}$ is assumed to hold. Point (b) follows by Theorem 4.1 (b). ◀

Corollary A.1 therefore provides a precise equivalence of the $\textsc{safe} \overset{?}{=} \textsc{inv}$ problem, as stated by Padon et al. [28], with fixpoint completeness (strong fixpoint completeness, in case (b)) in abstract interpretation.

A.2 Proofs

Proof of Lemma 3.2. Let us first recall that in a GI, for all $a, a' \in A$, $a \leq_A a' \Leftrightarrow \gamma(a) \leq_C \gamma(a')$ holds.
(a) ($\Leftarrow$) We have that:

$$\begin{aligned}
\exists a \in A.\, f\gamma(a) \leq_C \gamma(a) \wedge \gamma(a) \leq_C c' \Leftrightarrow \quad & [\text{by GC}]\\
\exists a \in A.\, \alpha f\gamma(a) \leq_A a \wedge \gamma(a) \leq_C c' \Rightarrow \quad & [\text{by } Fx \leq x \Rightarrow \mathsf{lfp}(F) \leq x]\\
\exists a \in A.\, \mathsf{lfp}(\alpha f\gamma) \leq_A a \wedge \gamma(a) \leq_C c' \Leftrightarrow \quad & [\text{by GI}]\\
\exists a \in A.\, \gamma(\mathsf{lfp}(\alpha f\gamma)) \leq_C \gamma(a) \wedge \gamma(a) \leq_C c' \Rightarrow \quad & [\text{by transitivity}]\\
\gamma(\mathsf{lfp}(\alpha f\gamma)) \leq_C c' \quad &
\end{aligned}$$

($\Rightarrow$) Define $a \triangleq \mathsf{lfp}(\alpha f\gamma) \in A$. It turns out that $\alpha f\gamma(a) \leq_A a$ so that, by GC, $f\gamma(a) \leq_C \gamma(a)$, and, by hypothesis, $\gamma(a) \leq_C c'$.
(b) It turns out that:

$$\begin{aligned}
\exists a \in A.\, f\gamma(a) \leq_C \gamma(a) \wedge \gamma(a) \leq_C \gamma(a') \Leftrightarrow \quad & [\text{By Lemma 3.2 (a)}]\\
\gamma(\mathsf{lfp}(\alpha f\gamma)) \leq_C \gamma(a') \Leftrightarrow \quad & [\text{by GI}]\\
\mathsf{lfp}(\alpha f\gamma) \leq_A a' \quad &
\end{aligned}$$

◀

Proof of Corollary 3.4. By Lemma 3.2 (a), because $\lambda x \in A.\alpha(c) \vee_A \alpha f\gamma(x) = \lambda x \in A.\alpha(c \vee_C f\gamma(x))$ is the best correct approximation of $\lambda x \in C.c \vee_C f(x)$. ◀

Proof of Corollary 3.5. The hypotheses guarantee that the procedure AInv is a terminating algorithm, in particular because the sequence of computed iterates i is an ascending chain in A. If the algorithm AInv outputs i then $i = \mathsf{lfp}(\lambda a.\alpha(c) \vee_A \alpha f\gamma(a)) \leq_A a'$, so that $i = \wedge\{a \in A \mid \alpha(c) \leq_A i, \alpha f\gamma(i) \leq_A i, i \leq_A a'\}$, that is, i is the least inductive invariant in A for f and $\langle c, \gamma(a')\rangle$. If the algorithm AInv outputs "no abstract inductive invariant for f and $\langle c, \gamma(a')\rangle$" then there exists $j \in \mathbb{N}$ such that $(\lambda a.\alpha(c) \vee_A \alpha f\gamma(a))^j(\perp_A) \not\leq_A a'$, so that $\mathsf{lfp}(\lambda a.\alpha(c) \vee_A \alpha f\gamma(a)) \not\leq_A a'$, that is, there exists no inductive invariant in A for f and $\langle c, \gamma(a')\rangle$. ◀

Proof of Theorem 4.1.

(a) $(\Rightarrow)$: By Lemma 3.2 (a).
$(\Leftarrow)$: Since $\mathsf{lfp}(f) \leq_C \mathsf{lfp}(f)$ holds, we have that $\exists a \in A.\ f\gamma(a) \leq_C \gamma(a) \wedge \gamma(a) \leq_C \mathsf{lfp}(f)$. Thus, by Lemma 3.2 (a), $\gamma(\mathsf{lfp}(\alpha f\gamma)) \leq_C \mathsf{lfp}(f)$ follows. On the other hand, $\mathsf{lfp}(f) \leq \gamma(\mathsf{lfp}(\alpha f\gamma))$ always holds because the pointwise correctness of $\alpha f\gamma$ implies $\alpha(\mathsf{lfp}(f)) \leq_A \mathsf{lfp}(\alpha f\gamma)$, hence, by GC, $\mathsf{lfp}(f) \leq \gamma(\mathsf{lfp}(\alpha f\gamma))$ follows.

(b) $(\Rightarrow)$: By Lemma 3.2 (b) because $\mathsf{lfp}(\alpha f\gamma) \leq_A a' \Leftrightarrow \alpha(\mathsf{lfp}(f)) \leq a' \Leftrightarrow \mathsf{lfp}(f) \leq_C \gamma(a')$.
$(\Leftarrow)$: We consider $a' \triangleq \alpha(\mathsf{lfp}(f))$, so that, $\mathsf{lfp}(f) \leq_C \gamma(a')$ holds and by the equivalence of the hypothesis, $\exists a \in A.\ f\gamma(a) \leq_C \gamma(a) \wedge \gamma(a) \leq_C \gamma\alpha(\mathsf{lfp}(f))$ holds. This implies, by GI, that $\exists a \in A.\ \alpha f\gamma(a) \leq_C a \wedge a \leq_A \alpha(\mathsf{lfp}(f))$. By the inductive invariant principle (1), this implies that (actually, is equivalent to) $\mathsf{lfp}(\alpha f\gamma) \leq \alpha(\mathsf{lfp}(f))$. Furthermore, $\alpha(\mathsf{lfp}(f)) \leq_A \mathsf{lfp}(\alpha f\gamma)$ always holds, therefore proving that $\alpha(\mathsf{lfp}(f)) = \mathsf{lfp}(\alpha f\gamma)$. ◀

Proof of Lemma 4.2. We have that:

$$\begin{aligned}
\exists a \in A.f\gamma(a) \leq_C \gamma(a) \wedge \gamma(a) \leq_C \gamma\alpha(\mathsf{lfp}(f)) \Leftrightarrow\ & [\text{by GI}]\\
\exists a \in A.\alpha f\gamma(a) \leq_A a \wedge a \leq_A \alpha(\mathsf{lfp}(f)) \Leftrightarrow\ & [\text{by (1) for } \alpha f\gamma]\\
\mathsf{lfp}(\alpha f\gamma) \leq_A \alpha(\mathsf{lfp}(f)) \Leftrightarrow\ & [\text{as } \alpha(\mathsf{lfp}(f)) \leq_A \mathsf{lfp}(\alpha f\gamma)]\\
\alpha(\mathsf{lfp}(f)) = \mathsf{lfp}(\alpha f\gamma)\ &
\end{aligned}$$
◀

Proof of Lemma 6.1.

(a) If $s \sqsubseteq_L s'$ and $t \in \check{\mu}_L(\{s\})$ then $t \in \check{\mu}_L(\{s'\}) = \cap\{\phi \in L \mid s' \in \phi\}$: if $\phi \in L$ and $s' \in \phi$ then $s \in \phi$, so that, since $t \in \check{\mu}_L(\{s\})$, $t \in \phi$. Conversely, if $\check{\mu}_L(\{s\}) \subseteq \check{\mu}_L(\{s'\})$, $\phi \in L$ and $s' \in \phi$, then, since $s \in \check{\mu}_L(\{s'\})$, $s \in \phi$.

(b) By (a), we equivalently prove that $\langle\{\check{\mu}_L(\{s\}) \mid s \in \Sigma\}, \subseteq\rangle$ is a wqo iff $\langle L, \subseteq\rangle$ is a wqo.

$(\Rightarrow)$: [28, Lemma 4.6] proves $\langle L, \subseteq\rangle$ is well-founded, we additionally show that it does not contain infinite antichains. By contradiction, assume that $\{\phi_i\}_{i\in\mathbb{N}}$ is an infinite antichain in $\langle L, \subseteq\rangle$. Thus, for all $i \neq j$, $\phi_i \not\subseteq \phi_j$ and $\phi_j \not\subseteq \phi_i$, so that there exist $s_{i,j} \in \phi_i \smallsetminus \phi_j$ and $s_{j,i} \in \phi_j \smallsetminus \phi_i$. From $s_{j,i} \in \phi_j$ we obtain that $\check{\mu}_L(\{s_{j,i}\}) \subseteq \check{\mu}_L(\phi_j) = \phi_j$. From $s_{i,j} \notin \phi_j$, we obtain that $s_{i,j} \in \check{\mu}_L(\{s_{i,j}\}) \not\subseteq \phi_{j,i}$. It turns out that $\check{\mu}_L(\{s_{i,j}\}) \not\subseteq \check{\mu}_L(\{s_{j,i}\})$, otherwise from $s_{i,j} \in \check{\mu}_L(\{s_{i,j}\}) \subseteq \check{\mu}_L(\{s_{j,i}\}) \subseteq \phi_j$ we would obtain the contradiction $s_{i,j} \in \phi_j$. Dually, $\check{\mu}_L(\{s_{j,i}\}) \not\subseteq \check{\mu}_L(\{s_{i,j}\})$ holds. Thus, for any $i \in \mathbb{N}$, $\{\check{\mu}_L(\{s_{i,j}\}) \mid j \in \mathbb{N}, j \neq i\}$ is an infinite antichain in $\langle\{\check{\mu}_L(\{s\}) \mid s \in \Sigma\}, \subseteq\rangle$, which is a contradiction.

$(\Leftarrow)$: $\langle\{\check{\mu}_L(\{s\}) \mid s \in \Sigma\}, \subseteq\rangle$ is trivially a wqo because $\{\check{\mu}_L(\{s\}) \mid s \in \Sigma\} \subseteq L$ and L is a wqo.

(c) Assume that for all $s \in \Sigma$, $Av_L(s) \in L$. Let us show that for all $S \in \wp(\Sigma)$, $\cap_{s\in S} Av_L(s) = Av_L(S)$.

($\supseteq$): Let $t \in \phi$ for some $\phi \in L$ such that $\phi \subseteq \neg S$. Then, for all $s \in S$, $\phi \subseteq Av_L(s)$, so that $t \in \cap_{s\in S} Av_L(s)$.

($\subseteq$): Let $t \in \cap_{s\in S} Av_L(s)$. For all $s \in S$, there exists $\phi_s \in L$ such that $\phi_s \subseteq \neg\{s\}$ and $t \in \phi_s$. Thus, $\cap_{s\in S}\phi_s \in L$ and $t \in \cap_{s\in S}\phi_s \subseteq \neg S$, meaning that $t \in Av_L(S)$.

Thus, since L is assumed to be closed under arbitrary intersections we obtain that $Av_L(S) = \cap_{s\in S} Av_L(s) \in L$. Consider now $\Phi \subseteq L$. Then, $Av_L(\neg(\cup\Phi)) = \cup\{\phi \in L \mid \phi \subseteq \neg\neg(\cup\Phi)\} = \cup\{\phi \in L \mid \phi \subseteq \cup\Phi\} = \cup\Phi$, so that $\cup\Phi \in L$.

Conversely, if L is closed under arbitrary unions then $Av_L(s) = \cup\{\phi \in L \mid \phi \subseteq \neg\{s\}\} \in L$.

(d) Since $\sqsubseteq_L$ is a quasi-order relation, its down-closure δ_L is an upper closure on $\langle\wp(\Sigma), \subseteq\rangle$. By (a), $\delta_L(X) = \{s \in \Sigma \mid \exists s' \in X. s \sqsubseteq_L s'\} = \{s \in \Sigma \mid \exists s' \in X. \check{\mu}_L(\{s\}) \subseteq \check{\mu}_L(\{s'\})\} = \{s \in \Sigma \mid \exists s' \in X. s \in \check{\mu}_L(\{s'\})\} = \cup_{s\in X}\check{\mu}_L(\{s\})$. By (c), since L is closed under arbitrary unions, the upper closure $\check{\mu}_L$ is additive, so that $\cup_{s\in X}\check{\mu}_L(\{s\}) = \check{\mu}_L(\cup_{s\in X}\{s\}) = \check{\mu}_L(X)$, consequently $\delta_L(X) = \check{\mu}_L(X)$. In particular, $\delta_L(\phi) = \phi \Leftrightarrow \check{\mu}_L(\phi) = \phi \Leftrightarrow \phi \in L$.

(e) By Lemma 6.1 (d), $\mathsf{lfp}(\lambda X. \mathsf{post}(X) \cup \delta_L(X) \cup \Sigma_0) = \mathsf{lfp}(\lambda X. \mathsf{post}(X) \cup \check{\mu}_L(X) \cup \Sigma_0)$. It turns out that

$$\begin{aligned}
\Sigma_0 \cup \mathsf{post}(X) \cup \delta_L(X) \subseteq X \Leftrightarrow \quad & [\text{by Lemma 6.1 (d)}]\\
\Sigma_0 \cup \mathsf{post}(X) \cup \check{\mu}_L(X) \subseteq X \Leftrightarrow \quad & [\text{by set theory}]\\
\Sigma_0 \subseteq X \wedge \mathsf{post}(X) \subseteq X \wedge \check{\mu}_L(X) \subseteq X \Leftrightarrow \quad & [\text{as } \check{\mu}_L \text{ is a uco}]\\
\Sigma_0 \subseteq X \wedge \mathsf{post}(X) \subseteq X \wedge \check{\mu}_L(X) = X \Leftrightarrow \quad & [\text{as } \check{\mu}_L(X) = X]\\
\Sigma_0 \subseteq X \wedge \mathsf{post}(\check{\mu}_L(X)) \subseteq X \wedge \check{\mu}_L(X) = X \Leftrightarrow \quad & [\text{by set theory}]\\
\Sigma_0 \cup \mathsf{post}(\check{\mu}_L(X)) \cup \check{\mu}_L(X) \subseteq X \quad &
\end{aligned}$$

Thus, by Knaster-Tarski theorem, $\mathsf{lfp}(\lambda X.\Sigma_0 \cup \mathsf{post}(\check{\mu}_L(X)) \cup \check{\mu}_L(X)) = \mathsf{lfp}(\lambda X.\Sigma_0 \cup \mathsf{post}(X) \cup \delta_L(X))$. We also have that:

$$\begin{aligned}
\Sigma_0 \cup \mathsf{post}(\check{\mu}_L(X)) \cup \check{\mu}_L(X) \subseteq X \Leftrightarrow \quad & [\text{by the equivalences above}]\\
\Sigma_0 \cup \mathsf{post}(\check{\mu}_L(X)) \cup \check{\mu}_L(X) \subseteq X = \check{\mu}_L(X) \Leftrightarrow \quad & [\text{by set theory}]\\
\Sigma_0 \cup \mathsf{post}(\check{\mu}_L(X)) \subseteq X = \check{\mu}_L(X) \Leftrightarrow \quad & [\text{as } \check{\mu}_L(X) = X]\\
\Sigma_0 \cup \mathsf{post}(X) \subseteq X = \check{\mu}_L(X) \Leftrightarrow \quad & [\text{as } \check{\mu}_L \text{ is a uco}]\\
\check{\mu}_L(\Sigma_0 \cup \mathsf{post}(X)) \subseteq X = \check{\mu}_L(X) \Leftrightarrow \quad & [\text{by set theory}]\\
\check{\mu}_L(\Sigma_0 \cup \mathsf{post}(X)) \subseteq X \quad &
\end{aligned}$$

Thus, by applying Knaster-Tarski theorem, $\mathsf{lfp}(\check{\mu}_L(\Sigma_0 \cup \mathsf{post}(X))) \subseteq \mathsf{lfp}(\lambda X.\Sigma_0 \cup \mathsf{post}(\check{\mu}_L(X)) \cup \check{\mu}_L(X))$ follows. Moreover, if $F \triangleq \mathsf{lfp}(\check{\mu}_L(\Sigma_0 \cup \mathsf{post}(X)))$, so that $F = \check{\mu}_L(F) = \Sigma_0 \cup \mathsf{post}(F)$, then $\Sigma_0 \cup \mathsf{post}(\check{\mu}_L(F)) \cup \check{\mu}_L(F) = \Sigma_0 \cup \mathsf{post}(F) \cup \check{\mu}_L(F) = F$, and this implies that $\mathsf{lfp}(\lambda X.\Sigma_0 \cup \mathsf{post}(\check{\mu}_L(X)) \cup \check{\mu}_L(X)) \subseteq \mathsf{lfp}(\check{\mu}_L(\Sigma_0 \cup \mathsf{post}(X)))$. Therefore, $\mathsf{lfp}(\lambda X.\Sigma_0 \cup \mathsf{post}(\check{\mu}_L(X)) \cup \check{\mu}_L(X)) = \mathsf{lfp}(\check{\mu}_L(\Sigma_0 \cup \mathsf{post}(X)))$. ◀

Proof of Corollary 6.2. It turns out that

$$\begin{aligned}
\mathsf{lfp}(\lambda X.\check{\mu}_L(\Sigma_0 \cup \mathsf{post}(X))) \subseteq P \Leftrightarrow \quad & [\text{by Lemma 3.2 (a) for ucos}]\\
\exists \phi \in \check{\mu}_L(\wp(\Sigma)).\Sigma_0 \cup \mathsf{post}(\phi) \subseteq \phi \wedge \phi \subseteq P \Leftrightarrow \quad & [\text{as } \check{\mu}_L(\wp(\Sigma)) = L]\\
\exists \phi \in L.\Sigma_0 \subseteq \phi \wedge \mathsf{post}(\phi) \subseteq \phi \wedge \phi \subseteq P \quad &
\end{aligned}$$
◀

Proof of Lemma 6.3. The proof of Lemma 6.1 (e) shows that $\mathsf{lfp}(\lambda X.\Sigma_0 \cup \mathsf{post}(\check{\mu}_L(X)) \cup \check{\mu}_L(X)) = \mathsf{lfp}(\check{\mu}_L(\Sigma_0 \cup \mathsf{post}(X)))$. Moreover, since $\check{\mu}_L$ is additive and $\check{\mu}_L(\Sigma_0) = \Sigma_0$, we also have that $\check{\mu}_L(\Sigma_0 \cup \mathsf{post}(X)) = \check{\mu}_L(\Sigma_0) \cup \check{\mu}_L(\mathsf{post}(X)) = \Sigma_0 \cup \check{\mu}_L(\mathsf{post}(X))$, and this allows us to conclude. ◀

▶ **Lemma A.2.** *Let $I \in L$ and $\Sigma_0 \subseteq I$.*

(a) *there exists a counterexample to inductiveness of I iff $I \not\subseteq \widetilde{\mathsf{pre}}(I) \cap P$ iff $I \neq \hat{\mu}_L(\widetilde{\mathsf{pre}}(I) \cap I \cap P)$.*

(b) *If $s \in \Sigma$ is a counterexample to inductiveness of I then $\hat{\mu}_L(\widetilde{\mathsf{pre}}(I) \cap I \cap P) \subseteq Av_L(s)$.*

Proof.

(a) Under the assumption that $\Sigma_0 \subseteq I$, s is a counterexample to inductiveness of I iff $(s \in I \wedge \mathsf{post}(s) \not\subseteq I) \vee s \in I \cap \neg P$. Observe that $\mathsf{post}(s) \not\subseteq I$ iff $s \notin \widetilde{\mathsf{pre}}(I)$, so that $\exists s \in \Sigma . s \in I \wedge \mathsf{post}(s) \not\subseteq I$ iff $I \not\subseteq \widetilde{\mathsf{pre}}(I)$. Hence, $\exists s \in \Sigma.(s \in I \wedge \mathsf{post}(s) \not\subseteq I) \vee s \in I \cap \neg P$ iff $I \not\subseteq \widetilde{\mathsf{pre}}(I) \cap P$. Also:

$$
\begin{aligned}
I = \hat{\mu}_L(\widetilde{\mathsf{pre}}(I) \cap I \cap P) \Leftrightarrow \quad & [\text{as } I \in L] \\
I = \hat{\mu}_L(I) = \hat{\mu}_L(\widetilde{\mathsf{pre}}(I) \cap I \cap P) \Leftrightarrow \quad & [\text{as } \widetilde{\mathsf{pre}}(I) \cap I \cap P \subseteq I] \\
I = \hat{\mu}_L(I) \subseteq \hat{\mu}_L(\widetilde{\mathsf{pre}}(I) \cap I \cap P) \Leftrightarrow \quad & [\text{as } \hat{\mu}_L \text{ is a lco}] \\
I = \hat{\mu}_L(I) \subseteq \widetilde{\mathsf{pre}}(I) \cap I \cap P \Leftrightarrow \quad & \\
I \subseteq \widetilde{\mathsf{pre}}(I) \cap P \quad &
\end{aligned}
$$

(b) The proof of point (a) shows that if $s \in \Sigma$ is a counterexample to inductiveness of I then $s \in I$ and $s \notin \widetilde{\mathsf{pre}}(I) \cap P$. Then, $\widetilde{\mathsf{pre}}(I) \cap P \subseteq \neg\{s\}$, so that, by monotonicity of $\hat{\mu}_L$, $\hat{\mu}_L(\widetilde{\mathsf{pre}}(I) \cap P) \subseteq \hat{\mu}_L(\neg\{s\})$ and, in turn, $\hat{\mu}_L(\widetilde{\mathsf{pre}}(I) \cap I \cap P) \subseteq \hat{\mu}_L(\neg\{s\}) = Av_L(s)$. ◀

Proof of Theorem 6.4. Consider the following variation of Algorithm 1:

Algorithm 4 A modification of Algorithm 1.

$I_4 := \Sigma$;
while $\Sigma_0 \subseteq I_4$ **do** // Invariant: $I_4 \in L$
  if $(I_4 \smallsetminus (\widetilde{\mathsf{pre}}(I_4) \cap P) = \varnothing)$ **then return** I_4 *is an inductive invariant in* L;
  choose $s \in I_4 \smallsetminus (\widetilde{\mathsf{pre}}(I_4) \cap P)$;
  $I_4 := I_4 \cap \hat{\mu}_L(\{s\})$;
return *no inductive invariant in* L;

Algorithm 1 returns $I_1 \in L$ iff $I_1 = \mathsf{gfp}(\lambda X.\hat{\mu}_L(\widetilde{\mathsf{pre}}(X) \cap X \cap P))$. In this case, by Lemma A.2 (a), since $\Sigma_0 \subseteq I_1$ holds, I_1 is an (actually, the greatest) inductive invariant in L. Otherwise, Algorithm 1 returns "no inductive invariant in L". By Lemma A.2 (a), Algorithm 4 returns $I_4 \in L$ iff $I_4 \subseteq \widetilde{\mathsf{pre}}(I_4) \cap P$ iff $I_4 = \hat{\mu}_L(\widetilde{\mathsf{pre}}(I_4) \cap I_4 \cap P)$, otherwise it returns "no inductive invariant in L". Algorithm 2 returns $I_2 \in L$ iff I_2 is an inductive invariant, otherwise it returns "no inductive invariant in L". Let I_k^n be the current candidate invariant of Algorithm $k \in \{1, 2, 4\}$ at its n-th iteration and I_k be the output invariant of Algorithm k. By Lemma A.2 (b), $I_1^n \subseteq I_4^n = I_2^n$, so that $I_1 \subseteq I_4 = I_2$. Since I_k are fixpoints of $\lambda X.\hat{\mu}_L(\widetilde{\mathsf{pre}}(X) \cap X \cap P)$ and I_1 is the greatest fixpoint, it turns out that $I_1 = I_4 = I_2$. ◀