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 \( \mathcal{A} \) as defined in abstract interpretation, namely, ensuring the existence of the best approximation in \( \mathcal{A} \) of any system property. In this setting, we study the decidability of the existence of abstract inductive invariants in \( \mathcal{A} \) 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 \( \mathcal{A} \) 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 SStatic Analyses (ASTA)”; Italian Ministry of University and Research, under the PRIN2017 project no. 201784YSZ3 “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 \( \mathcal{T} = (\Sigma, \tau, \Sigma_0) \), where \( \tau \) is a transition relation on states ranging in \( \Sigma \) 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 \( I \in \varphi(\Sigma) \) 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 \( \mathcal{A} \subseteq \varphi(\Sigma) \), 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]).
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 $A$. We make the following working assumption: $A \subseteq \wp(\Sigma)$ 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. $\subseteq$) $\alpha_A(X)$ in $A$ and each state transition relation $\tau$ has a best correct approximation $\tau^A$ on the abstract domain $A$. Under these hypotheses, we prove an abstract inductive invariant principle stating that there exists an abstract inductive invariant in $A$ proving a property $P$ of a transition system $T$ iff the best abstraction $T^A$ in $A$ of the system $T$ allows us to prove $P$. The decidability/undecidability question of the existence of abstract inductive invariants in some abstract domain $A$ 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 $A$ and to design a general algorithm for synthesizing the least (w.r.t. the order of $A$) abstract inductive invariant in $A$, when this exists, by a least fixpoint computation in $A$.

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

The decidability/synthesis of abstract inductive invariants in a domain $A$ for some class $\mathcal{C}$ of systems essentially boils down to prove that the best correct approximation $\tau^A$ in $A$ of the transition relation $\tau$ 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 $A$ 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 $A$ of abstract invariants which is disjunctive, i.e., abstract least upper bounds of $A$ 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 $f : X \rightarrow Y$ 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 $f \preceq g$ 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), \( \lor \) (or \( \sqcup \)) and \( \land \) (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, \( \text{Fix}(f) \triangleq \{ x \in P \mid f(x) = x \} \), \( \text{Fix}^\leq(f) \triangleq \{ x \in P \mid f(x) \leq x \} \), \( \text{Fix}^\geq(f) \triangleq \{ x \in P \mid f(x) \geq x \} \), and \( \text{lfp}(f) \), \( \text{gfp}(f) \) denote, resp., the least and greatest fixpoint in \( \text{Fix}(f) \), when they exist. Let us recall Knaster-Tarski fixpoint theorem: if \( (C, \leq, \lor, \land) \) is a complete lattice and \( f : C \to C \) is monotonic then \( \text{Fix}(f), \leq \) is a complete lattice, \( \text{lf}p(f) = \land \text{Fix}^\leq(f) \) and \( \text{gfp}(f) = \lor \text{Fix}^\geq(f) \). Also, Knaster-Tarski-Kleene fixpoint theorem states that if \( (C, \leq, \lor, \land) \) is a CPO with least element \( i \) and \( f : C \to C \) is Scott-continuous (i.e., \( f \) preserves lubs of directed subsets) then \( \text{lf}p(f) = \lor_{i \in \mathbb{N}} f^i(i) \), where, for all \( x \in C \) and \( i \in \mathbb{N} \), \( f^0(x) \leq x \) and \( f^{i+1}(x) = f(f^i(x)) \); dually, if \( (C, \leq, \land, \lor) \) is a dual-CPO with greatest element \( i \) and \( f : C \to C \) is Scott-co-continuous then \( \text{gfp}(f) = \land_{i \in \mathbb{N}} f^i(i) \).

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 \( \text{uco}(C \leq) \)/\( \text{lco}(C \leq) \). We write \( e \in \mu(C) \), or simply \( e \in \mu \), to denote that there exists \( e' \in C \) such that \( e = \mu(e') \), 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 \( \mu(C) \leq \) is closed under glb of arbitrary subsets and, conversely, \( X \subseteq C \) is the image of some \( \mu \in \text{uco}(C) \) iff \( X \) is closed under glb of all its subsets, and in this case \( \mu(c) = \land \{e' \in X \mid e \leq e' \} \) holds. Dually, \( X \subseteq C \) is closed under arbitrary lub of its subsets iff \( X \) is the image a lower closure \( \eta \in \text{lco}(C) \), and in this case \( \eta(e) = \lor \{e' \in X \mid e' \leq e \} \). In abstract interpretation, a closure \( \mu \in \text{uco}(C) \) on a concrete domain \( C \leq \) plays the role of an abstract domain having best approximations: \( e \in C \) is upper-approximated by any \( \mu(e') \) such that \( e \leq \mu(e') \) and \( \mu(c) \) is the best approximation of \( c \in \mu \) because \( \mu(c) = \land \{\mu(e') \mid e' \in C, e \leq \mu(e') \} \).

A Galois Connection (GC, also called adjunction) between two posets \( (C, \leq_C) \), called concrete domain, and \( (A, \leq_A) \), called abstract domain, consists of two maps \( \alpha : C \to A \) and \( \gamma : A \to C \) such that \( \alpha(c) \leq_A a \Rightarrow c \leq_C \gamma(a) \) holds. A GC is called Galois insertion (GI) when \( \alpha \) is surjective or, equivalently, \( \gamma \) is injective. Any GC can be transformed into a GI simply by removing useless elements in \( A \setminus \alpha(C) \) 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 \( \mu_{\mathcal{G}} \triangleq \gamma \circ \alpha \in \text{uco}(C) \), any \( \mu \in \text{uco}(C) \) 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 $\mathcal{T} = (\Sigma, \tau)$ be a transition system where $\Sigma$ is a set of states and $\tau \subseteq \Sigma \times \Sigma$ is a transition relation inducing the following transformers of type $\wp(\Sigma) \to \wp(\Sigma)$:

\[
\text{pre}(X) \triangleq \{ s \in \Sigma \mid \exists s' \in X. (s, s') \in \tau \} \quad \text{pre}(X) \triangleq \{ s \in \Sigma \mid \forall s'. (s, s') \in \tau \Rightarrow s' \in X \}
\]

\[
\text{post}(X) \triangleq \{ s' \in \Sigma \mid \exists s \in X. (s, s') \in \tau \} \quad \text{post}(X) \triangleq \{ s' \in \Sigma \mid \forall s. (s, s') \in \tau \Rightarrow s \in X \}
\]

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

2.4 Inductive Invariant Principle

Given a transition system $\mathcal{T} = (\Sigma, \tau)$, a set of states $I \in \wp(\Sigma)$ is an inductive invariant for $\mathcal{T}$ w.r.t. $(\Sigma_0, P) \in \wp(\Sigma)^2$ when: (i) $\Sigma_0 \subseteq I$; (ii) $\text{post}(I) \subseteq I$; (iii) $I \subseteq P$. An inductive invariant $I$ allows us to prove that $\mathcal{T}$ is safe, i.e. 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 $C_\subseteq$ is a complete lattice, $c', c \in C$ and $f : C \to C$ is monotonic then

\[
\text{lfp}(f) \leq c' \iff \exists i \in C. f(i) \leq i \land i \leq c'
\]

(1)

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

\[
\text{lfp}(\lambda x. c \lor_C f(x)) \leq c' \iff \exists i \in C. c \leq i \land f(i) \leq i \land i \leq c'
\]

(2)

One such $i \in C$ such that $c \leq i \land f(i) \leq i \land i \leq c$ is called an inductive invariant of $f$ for $(c, c')$. Hence, (2) is applied to the function $\lambda X. \Sigma_0 \cup \text{post}(X) : \wp(\Sigma) \to \wp(\Sigma)$, which is monotonic on $\wp(\Sigma)_\subseteq$, so that $\text{lfp}(\lambda X. \Sigma_0 \cup \text{post}(X)) \subseteq P$ holds iff there exists an inductive invariant $I$ for $\mathcal{T}$ w.r.t. $(\Sigma_0, P)$. 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 $C$ of complete lattices and, for all $C \in C$, a class of functions $F_C \subseteq C \to C$, a set of initial properties $\text{Init}_C \subseteq C$, a set of safety properties $\text{Safe}_C \subseteq C$, and an abstract domain $A_C \subseteq C$, a first problem is the decidability of the following decision question:

\[
\forall C \in C. \forall f \in F_C. \forall e \in \text{Init}_C. \forall e' \in \text{Safe}_C. \forall i \in A_C. c \leq i \land f(i) \leq i \land i \leq c'
\]

(3)

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

Decidability and Synthesis of Abstract Inductive Invariants
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 $T = (\Sigma, \tau)$ whose successor transformer is post, the problem (3) is instantiated to $C_L = \varphi(\Sigma)_L$, $f = \text{post}(X)$, $c = \Sigma_0 \subseteq \varphi(\Sigma)$ set of initial states and $c' = P \subseteq \varphi(\Sigma)$ safety property. When $T$ is the control flow graph generated by some program, $\Sigma_0$ 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) $\langle C_{\leq_C}, \alpha, \gamma, A_{\leq_A} \rangle$ is a Galois insertion;

(ii) $\langle A, \leq_A \rangle = \langle \mu(C), \leq_C \rangle$ for some upper closure $\mu \in \text{uco}(C_{\leq_C})$.

Under Assumption 3.1, let us recall that if $f : C \rightarrow C$ is a concrete monotonic function then the mappings $\alpha f \gamma : A \rightarrow A$, for the case of GIs, and $\mu f : \mu(C) \rightarrow \mu(C)$, 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 $f^\gamma : A \rightarrow A$ (or $f^\mu : \mu(C) \rightarrow \mu(C)$ for ucos) is a correct (or sound) approximation of $f$ when $\alpha f \gamma \leq_A f^\gamma$ (or $\mu f \leq_C f^\mu$ 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 $\mu \in \text{uco}(C)$, 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 $\langle C_{\leq_C}, \alpha, \gamma, A_{\leq_A} \rangle$ be a GI. For all $c \in C$ and $a' \in A$:

(a) $\gamma(\text{lfp}(\alpha f \gamma)) \leq_C c' \Leftrightarrow \exists a \in A. f \gamma(a) \leq_C \gamma(a) \land \gamma(a) \leq_C c'$;

(b) $\text{lfp}(\alpha f \gamma) \leq_A a' \Leftrightarrow \exists a \in A. f \gamma(a) \leq_C \gamma(a) \land \gamma(a) \leq_C \gamma(a')$.

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

Let us observe that point (b) is an easy consequence of point (a), because, by surjectivity of $\alpha$ in GIs, for all $a' \in A$, there exists some $c' \in C$ such that $a' = \alpha(c')$, and $\gamma(\text{lfp}(\alpha f \gamma)) \leq_C \gamma(\alpha(c')) \Leftrightarrow \text{lfp}(\alpha f \gamma) \leq_A \alpha(c')$ holds. Moreover, point (b) easily follows from the inductive invariant principle (1) for the bca $\alpha f \gamma : A \rightarrow A$. 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 $f : C \rightarrow C$ defined by $\{1 \mapsto 1; 2 \mapsto 2; 3 \mapsto 4; 4 \mapsto 4\}$, and the abstraction $A = \{2, 4\}$ with $\gamma = \text{id}$ 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 $\text{lfp}(\alpha f \gamma) = 2$. 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
Decidability and Synthesis of Abstract Inductive Invariants

30:6

concrete property \( 3 \in C \setminus \gamma(A) \), 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, \( \gamma(\text{lfp}(\alpha f \gamma)) \not\subseteq 1 \) holds, thus, by Lemma 3.2 (a), the concrete property \( 1 \in C \setminus \gamma(A) \) 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 bea \( \alpha f \gamma \) 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 \( a f \gamma \) proving \( a' \) is equivalent to whether the strongest abstract invariant \( \text{lfp}(\alpha f \gamma) \) entails \( a' \)”, i.e. is inductive, and generalizes [32, Observation 1] stating (in our terminology) that “\( \text{lfp}(\alpha f \gamma) \) 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 \( \gamma(\text{lfp}(\alpha f \gamma)) \subseteq C \ c' \).

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

\[ \text{Corollary 3.4.} \text{ Let } F \subseteq C \rightarrow C \text{ and } \text{Init, Safe} \subseteq C. \text{ The Monniaux decision problem } \forall f \in F. \forall c \in \text{Init}. \forall c' \in \text{Safe}. \exists a \in \ A. c \leq C \gamma(a) \land f(\gamma(a)) \leq C \gamma(a) \land \gamma(a) \leq C \ c' \text{ is decidable iff the decision problem } \forall f \in F. \forall c \in \text{Init}. \forall c' \in \text{Safe}. \gamma(\text{lfp}(\lambda x \in A. \alpha(c) \lor A f \gamma(x))) \leq C \ c' \text{ is decidable.} \]

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

\[ \text{Corollary 3.5.} \text{ Assume that the lub } \forall A : A \times A \rightarrow A \text{ and the bea } \alpha f \gamma : A \rightarrow A \text{ are finitely computable, the partial order } \leq A \text{ is decidable and } A \text{ is an ACC CPO with least element. For all } c \in C \text{ such that } \alpha(c) \text{ is finitely computable and } a' \in A, \text{ the following procedure:} \]

\[
\text{AInv}(f, A, c, a') \triangleq i \triangleq \alpha(c);
\text{ while } i \leq A a' \text{ do } \{ \text{ if } \alpha f \gamma(i) \leq A i \text{ return } i; \text{ else } i \triangleq \alpha f \gamma(i); \}
\text{ return no abstract inductive invariant for } f \text{ and } \langle c, \gamma(a') \rangle; \]

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 \( \text{post}_r \) reduces to the problem of computing the best correct approximation \( \alpha \text{ post}_r \gamma \).

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 \( f : C \rightarrow C \) and \( f^2 : A \rightarrow A \) are the concrete and abstract monotonic transformers then fixpoint soundness means that \( \alpha(\text{lfp}(f)) \leq A \text{lfp}(f^2) \) holds, so that a positive abstract proof \( \text{lfp}(f^2) \leq A a' \) entails that \( \gamma(a') \) concretely holds, i.e., \( \text{lfp}(f) \leq C \gamma(a') \). Fixpoint soundness is usually proved as a consequence
of pointwise soundness: if \( f^2 \) is a pointwise correct approximation of \( f \), i.e. \( \alpha f \leq_A f^2 \alpha \), then \( \alpha \lf(f) \leq_A \lf(f^2) \) 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 \( \alpha \lf(f) = \lf(f^2) \) holds, so that \( \lf(f^2) \not\leq_A \alpha' \) entails \( \lf(f) \not\leq_C \gamma'(a') \). One can also consider a strong fixpoint completeness requiring that \( \lf(f) = \gamma(\lf(f^2)) \), so that \( \lf(f^2) \not\leq_A \alpha(c') \) entails \( \lf(f) \not\leq_C c' \). However, it should be remarked that \( \lf(f) = \gamma(\lf(f^2)) \) is much stronger than \( \alpha \lf(f) = \lf(f^2) \) since it means that the concrete \( \lf \) is precisely represented by the abstract \( \lf \).

It is important to remark that if \( f^2 \) is a pointwise correct approximation of \( f \) and fixpoint completeness for \( f^2 \) holds then since by the inductive invariant principle \( \gamma \), the abstract transformer \( f^2 : A \rightarrow A \) which is fixpoint complete does not depend on the specific definition of \( f^2 \) but is instead an intrinsic property of the abstract domain \( A \) w.r.t. the concrete transformer \( f \), as formalized by the equation \( \alpha \lf(f) = \lf(f^2) \). Moreover, fixpoint completeness is typically proved as a by-product of pointwise completeness \( \alpha f = f^2 \alpha \), and if \( f^2 \) is pointwise complete then it turns out that \( f^2 = \alpha f \gamma \), that is, \( f^2 \) 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 \( \lf {\alpha f \gamma} \) with \( \alpha \lf(f) \)? This question is settled by the following result.

**Theorem 4.1.** Let \((C_{\leq C} , \alpha, \gamma, A_{\leq_A})\) be a GI.

(a) \( \lf(f) = \gamma(\lf \alpha f \gamma) \) if and only if \( \forall c' \in C. (\lf(f) \leq_C c' \Leftrightarrow \exists a \in A. . f \gamma(a) \leq_C \gamma(a) \wedge \gamma(a) \leq_C c') \);

(b) \( \alpha \lf(f) = \lf \alpha f \gamma \) if and only if \( \exists a' \in A. (\lf(f) \leq_C \gamma(a')) \Leftrightarrow \exists a \in A. f \gamma(a) \leq_C \gamma(a) \wedge \gamma(a) \leq_C \gamma(a'). \)

Theorem 4.1 (b) can be stated by means of ucos as follows: if \( \mu \in uco(C) \) then \( \mu \lf(f) = \lf \mu f \) if and only if \( \exists a' \in c. \lf(f) \leq_C a' \Leftrightarrow \exists a \in \mu. f a \leq_C a \wedge a \leq_C a' \).

The above result can be read as follows. Since, by the inductive invariant principle (1), \( \lf(f) \leq_C c' \) if and only if 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 \( c' \) if fixpoint completeness holds. In other terms, the (concrete) inductive invariant principle is equivalent to the abstract inductive invariant principle if 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.** \( \alpha \lf(f) = \lf \alpha f \gamma \Leftrightarrow \exists a \in A. f \gamma(a) \leq_C \gamma(a) \wedge \gamma(a) \leq_C \gamma \alpha \lf(f) \).

As a consequence, fixpoint completeness for \( f \) does not hold in \( A \) if and only if the abstract property \( \alpha \lf(f) \in A \) 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 \( f : C \rightarrow C \) 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 \( \mu(\text{lfp}(f)) = \mu(1) = 2 < 3 = \text{lfp}(\mu f) \).

Thus, in accordance with Lemma 4.2, it turns out that \( \mu(\text{lfp}(f)) = 2 \) cannot be inductively proved in the abstraction \( \mu \). In fact, \( f(2) \leq 2 \), while \( f(3) \leq 3 \) but \( 3 \not\leq \mu(\text{lfp}(f)) \).

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

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

For all control nodes \( q \in Q \) and vectors \( \vec{X} \in \wp(\forall^n)^{|Q|} \), we will also use \( \pi_q(\vec{X}) \) and \( \vec{X}_q \) to denote the \( q \)-indexed component of \( \vec{X} \), e.g., \( \text{Reach}[\mathcal{P}, \Sigma_0] \in \wp(\forall^n) \) 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 \( (\wp(\forall^n)_{\leq}, \alpha, \gamma, A_{\leq}, A) \) parametric on \( n \in \mathbb{N} \). By Corollary 3.4, for a given class \( C \) of programs, a class \( \text{Init} \) of sets of initial states and a class \( \text{Safe} \) of sets of safety properties, the Monniaux problem (3) is decidable iff for all \( \mathcal{P} = \langle Q, n, V, T, \rightarrow \rangle \in C, \Sigma_0 \in \text{Init} \) and \( P \in \text{Safe}, \)

\[
\dot{\gamma}(\text{lfp}(\lambda a \in A^{|Q|}. \dot{\alpha}(\Sigma_0) \vee_{A} \dot{\alpha}(\text{post}_{\mathcal{P}}(\dot{\gamma}(\dot{a}))))) \subseteq_{\gamma} P
\]

is decidable. Moreover, Corollary 3.5 provides an abstract inductive invariant synthesis algorithm \( \text{AINV} \) for safety properties represented by \( A \) (i.e., \( P \in \gamma(A) \)) when \( A, C, \text{Init} \) and \( \text{Safe} \) 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 \( \text{AFF} \), 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 \( \text{AFF} \) based on a different representation of affine sets and proved that \( \text{AFF} \) 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 \( \text{AFF}_n \) for \( n \) program variables ranging in \( \text{Var}_n \equiv \{x_1,\ldots,x_n\} \) and assuming rational values\(^1\), that is, \( \mathbb{V} = \mathbb{Q} \). The logical abstract invariants represented by \( \text{AFF}_n \) 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:

\[
\text{aff}(X) \equiv \{ \sum_{j=0}^m \alpha_j \vec{v}_j \in \mathbb{Q}^n \mid m \in \mathbb{N}, \alpha_j \in \mathbb{Q}, \vec{v}_j \in X, \sum_{j=0}^m \alpha_j = 1 \}.
\]

This map \( \text{aff} : \wp(\mathbb{Q}^n) \to \wp(\mathbb{Q}^n) \) is an upper closure on \( \wp(\mathbb{Q}^n), \subseteq \) whose fixpoints are precisely the affine subsets of \( \mathbb{Q}^n \) and therefore may be used to define the affine equalities domain \( \text{AFF}_n \equiv \{ \text{aff}(\wp(\mathbb{Q}^n)), \subseteq \} \) 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 independent 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 \( \text{aff} \) is meant to act as abstraction map \( \alpha_{\text{AFF}} : \wp(\mathbb{Q}^n) \to \text{AFF}_n \) and correspondingly the concretization \( \gamma_{\text{AFF}} : \text{AFF}_n \to \wp(\mathbb{Q}^n) \) is the identity. For the sake of clarity, in our examples we will use logical affine equalities for representing affine sets. \( \text{AFF}_n \) is a complete lattice of finite height \( n + 1 \), because if \( a, a' \in \text{AFF}_n \) and \( a \subseteq a' \) then \( \dim(a) < \dim(a') \), where \( \dim(\emptyset) = -1 \) and \( \dim(\mathbb{Q}^n) = n \). The domain \( \text{AFF}_n \) is not closed under arbitrary unions, i.e., \( \text{aff} \) is not an additive uco, so that the lub of \( X \subseteq \text{AFF}_n \) is given by \( \bigcup_{X \subseteq \text{AFF}_n} X \equiv \text{aff}(\bigcup_{X \subseteq \text{AFF}_n} X) \). A matrix-based algorithm for computing a binary lub \( a \sqcup_{\text{AFF}} a' \) 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 \( \text{AFF} \) for a given class \( C \) of programs, Init of sets of initial states and Safe of sets of safety properties, is a decidable problem iff for all \( P \in C \) with \( n \) variables and control nodes in \( Q \), for all \( \Sigma_0 \in \text{Init}_P \subseteq \wp(\mathbb{V}^n)^{|Q|} \) and for all \( P \in \text{Safe}_P \subseteq \wp(\mathbb{V}^n)^{|Q|} \),

\[
\hat{\gamma}_{\text{AFF}}(\text{aff}(\Sigma_0) \sqcup_{\text{AFF}} \text{aff}(\text{post}_P(\hat{\gamma}_{\text{AFF}}(\vec{a})))) \subseteq \gamma_{\text{AFF}}(P)
\]

is decidable. Therefore, when \( \Sigma_0, P \in \text{AFF}_n \) and since \( \text{AFF}_n^{|Q|} \) has finite height \( |Q|(n+1) \), a sufficient condition for the decidability of the problem (5) is that the bca \( \hat{\alpha}_{\text{AFF}} \circ \text{post}_P \circ \gamma_{\text{AFF}} : \text{AFF}_n^{|Q|} \to \text{AFF}_n^{|Q|} \) is computable, where for all \( \vec{a} \in \text{AFF}_n^{|Q|} \);

\(^1\)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 \( \mathbb{Z} \).
Decidability and Synthesis of Abstract Inductive Invariants

\[ \alpha_{\text{AFF}}(\text{post}_P(\gamma_{\text{AFF}}(\bar{a}))) = (\bigcup_{\text{AFF}} \{ \alpha_{\text{AFF}}(t(\pi_q(\gamma_{\text{AFF}}(\bar{a})))) \mid \exists q \in Q, \exists t \in T_P.q \mapsto q' \})_{q' \in Q} \]  

(6)

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

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

\[ t_{a}^{b}(Y) \triangleq \{ \bar{v} | v \in Y, v' = \sum_{i=1}^{n} m_i \bar{v}_i + b \}, \quad t_{b}(Y) \triangleq \{ \bar{v} \in Y | \sum_{i=1}^{n} m_i \bar{v}_i + b \Delta 0 \}. \]

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

Karr gave already in [17, Section 4.2] an algorithm for computing the bca of an affine assignment \( t_{a}^{b} \) 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 \( t_{a}^{b} \) turns out to be pointwise complete, namely \( \text{aff} \circ t_{a}^{b} \circ \text{aff} = \text{aff} \circ t_{a}^{b} \) holds. Hence, in turn, computability of parallel affine assignments \( \bar{x} := M \bar{x} + \bar{b} \) easily follows. [26, Section 4] also shows that the bca of a nondeterministic assignment \( t_{a}^{b} \) is computable, where the corresponding transfer function is defined by:

\[ t_{a}^{b}(Y) \triangleq \{ \bar{v} | v \in Y, v' = \sum_{i=1}^{n} m_i \bar{v}_i + b \}. \]

In fact, one can observe [26, Lemma 4] that \( \text{aff}(t_{a}^{b}(\text{aff}(Y))) = \text{aff}(t_{a}^{b}(\text{aff}(Y))) \cap_{\text{AFF}} \text{aff}(t_{a}^{b}(\text{aff}(Y))) \), so that computing the bca of \( t_{a}^{b}(\text{aff}(Y)) \) 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 \( t_{a}^{b} := \sum_{i=1}^{n} m_i x_i + b = 0 \) are algorithmically computable through the glb of \( \text{AFF}_n \), i.e., for all \( a \in \text{AFF}_n \), \( \alpha_{\text{AFF}}(t_{a}^{b}(\gamma_{\text{AFF}}(a))) = a \cap_{\text{AFF}} a_{b, -} \), where \( a_{b, -} \in \text{AFF}_n \) denotes the affine set representing the affine equality \( \sum_{i=1}^{n} m_i x_i + b = 0 \). For affine inequalities Boolean guards \( t_{a}^{b} := \sum_{i=1}^{n} m_i x_i + b \neq 0 \), Karr [17, Section 4.1] defines the following abstract function:

\[ t_{a}^{b}(a) \triangleq \text{if } a \subseteq a_{b, -} \text{ then } \bot_{\text{AFF}} \text{ else } a, \]

and states that “we must be content with a \( a \) on the “otherwise” case...a general study of how best to handle decision nodes which are not of the simple form \( t_{a}^{b} \) is in preparation”, but this document never appeared. Nevertheless, we notice that the above definition of \( t_{a}^{b} \) actually is the bca \( \alpha_{\text{AFF}} \circ t_{a}^{b} \circ \gamma_{\text{AFF}} = \lambda a. \text{aff}(a \cap \neg a_{b, -}) \). In fact, let us observe the following fact (2): if \( a, a' \in \text{AFF}_n \), then \( a' \subseteq a \Rightarrow \text{aff}(a \cap \neg a_{b, -}) = a \). In fact, we have that \( \text{dim}(a') < \text{dim}(a) \), and, in turn, \( \text{dim}(\text{aff}(a \cap \neg a')) = \text{dim}(\text{aff}(a \cap \neg a')) = \text{dim}(a) \) hold, therefore entailing that \( \text{aff}(a \cap \neg a') = a \).

Thus: \( a \) if \( a_{b, -} \subseteq a \) then, by (2), \( \text{aff}(a \cap \neg a_{b, -}) = (b) \) if \( a_{b, -} \) and \( a \) are incomparable then \( a \cap a_{b, -} \subseteq a \), so that, by (2), \( \text{aff}(a \cap \neg a_{b, -}) = (b) \) if \( a_{b, -} \) and \( a \) are incomparable then \( a \cap a_{b, -} \subseteq a \), so that, by (2), \( \text{aff}(a \cap \neg a_{b, -}) = (b) \) if \( a_{b, -} \) and \( a \) are incomparable then \( a \cap a_{b, -} \subseteq a \), so that, by (2), \( \text{aff}(a \cap \neg a_{b, -}) = a \).

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 \( \mathcal{C}_{\text{AFF}} \) 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 \( \mathcal{C}_{\text{AFF}} \), affine sets of initial states and affine sets of state properties is decidable. Moreover, the algorithm \( \text{AINV} \) of Corollary 3.5 instantiated to \( \text{post}_P \) for \( P \in \mathcal{C}_{\text{AFF}} \), synthesizes the least inductive invariant of \( P \), when this exists.
To the best of our knowledge, the literature provides no algorithm for computing the bca of further linear Boolean guards \(\sum_{i=1}^{n} m_i x_i + b \leq 0\), with \(\leq \in \{<, \leq\}\). We conjecture that at least some of these bca’s are algorithmically computable.

**Example 5.2.** Consider the following nondeterministic program \(\mathcal{R}\) with \(\Sigma_0 = \{q_1\} \times \mathbb{Q}^3 \in \text{AFF}^{[\mathcal{Q}]}\) and the property \(P^f = \langle (q_1, \top), (q_2, \top), (q_3, \top), (q_4, x_1 + x_2 + 1 = 0) \rangle \in \text{AFF}^{[\mathcal{Q}]}\).

![Diagram of the example program](image)

The algorithm ALV of Corollary 3.5 yields the following sequence of \(I^j \in \text{AFF}^{[\mathcal{Q}]}\):

\[
I^0 = \hat{\alpha}_{\text{AFF}}(\Sigma_0) = \langle (q_1, \top), (q_2, \bot), (q_3, \bot), (q_4, \bot) \rangle
\]

\[
I^1 = \langle (q_1, \top), (q_2, x_1 + 2 = 0 \land x_2 - 1 = 0 \land x_3 - 1 = 0), (q_3, \bot), (q_4, \bot) \rangle
\]

\[
I^2 = \langle (q_1, \top), (q_2, x_1 + 2 = 0 \land x_2 - 1 = 0 \land x_3 - 1 = 0),
(q_3, x_1 + 2x_2 = 0 \land x_3 = 1), (q_4, \bot) \rangle
\]

\[
I^3 = \langle (q_1, \top), (q_2, x_1 + 2x_2 = 0 \land x_3 = 1), (q_3, x_1 + 2x_2 = 0 \land x_3 = 1),
(q_4, x_1 + x_2 + 1 = 0 \land x_3 = 1, \bot) \rangle = \hat{\alpha}_{\text{AFF}}(\text{post}_\mathcal{R}(\gamma_{\text{AFF}}(I^3))) \preceq_{\text{AFF}} P^f
\]

The output \(I^3\) is the analysis of \(\mathcal{R}\) with the bca’s of its transfer functions in \(\text{AFF}\), i.e., it is the least inductive invariant in \(\text{AFF}\) which allows us to prove that \(P^f\) 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 \(t_{x_1 := 0}\) are pointwise complete. In fact, [26, Lemma 2] shows that for all \(X \in \rho(\mathbb{Q}^n)\), \(t_a(\text{aff}(X)) = \text{aff}(t_a(X))\), from which we easily obtain:

\[
\text{aff}(t_a(X)) = \text{aff}(t_a(\text{aff}(X)))
\]

Thus, since pointwise completeness entails fixpoint completeness (cf. Section 4.1), for all unguarded programs \(\mathcal{P}\) with affine and nondeterministic assignments, \(\hat{\alpha}_{\text{AFF}}(\text{lfp}(\lambda \hat{\alpha}_{\text{AFF}}(\Sigma_0) \cup \text{post}_\mathcal{P}(X))) = \text{lfp}(\lambda \hat{\alpha}_{\text{AFF}}(\Sigma_0) \cup \hat{\alpha}_{\text{AFF}}(\text{post}_\mathcal{P}(\gamma_{\text{AFF}}(\overline{a}))))\) 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 \(\mathcal{C}_{\text{AFF}}\), because these guards are not pointwise complete: for example, we have that \(\text{aff}(t_{x_1 := 0}(\{(1, 0), (-1, 0)\})) = \text{aff}(\overline{a}) = \emptyset\), whereas \(\text{aff}(t_{x_1 := 0}(\text{aff}(\{(1, 0), (-1, 0)\}))) = \text{aff}(t_{x_1 := 0}(x_2 = 0)) = \text{aff}(\{(0, 0)\}) = \{(0, 0)\}\). 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
30:12 Decidability and Synthesis of Abstract Inductive Invariants

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 \( \forall f \in \mathcal{C}_{\text{AFF}}, \forall a \in \text{AFF}, \alpha_{\text{AFF}}(\mathsf{fp}(f)) \leq_{\text{AFF}} a \) is an undecidable problem, while Theorem 5.1 shows that \( \forall f \in \mathcal{C}_{\text{AFF}}, \forall a \in \text{AFF}, \mathsf{fp}(\alpha_{\text{AFF}} f \gamma_{\text{AFF}}) \leq_{\text{AFF}} a \) is decidable, and, by Theorem 4.1 (b), these are equivalent problems iff (where “iff” is obvious) fixpoint completeness \( \forall f \in \mathcal{C}_{\text{AFF}}, \alpha_{\text{AFF}}(\mathsf{fp}(f)) = \mathsf{fp}(\alpha_{\text{AFF}} f \gamma_{\text{AFF}}) \) holds.

Example 5.3. Consider the following deterministic program \( \mathcal{P} \in \mathcal{C}_{\text{AFF}} \):

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

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

\[
\alpha_{\text{AFF}}(\pi_{q_o}(\mathsf{fp}(\lambda \vec{x}, \Sigma_0 \cup \mathsf{post}_{\mathcal{P}}(\vec{x})))) = \alpha_{\text{AFF}}(\{(0 + 2n, 3 - 2n) \mid n \in \mathbb{N}\} \cap \langle x_1 = 3 \rangle)
\]

\[
= \alpha_{\text{AFF}}(\emptyset) = \bot_{\text{AFF}}
\]

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

\[
\pi_{q_o}(\mathsf{fp}(\lambda \vec{a} \in \text{AFF}[^Q], \alpha_{\text{AFF}}(\Sigma_0) \cup_{\text{AFF}} \alpha_{\text{AFF}}(\mathsf{post}_{\mathcal{P}}(\gamma_{\text{AFF}}(\vec{a})))))) = \langle x_1 + x_2 = 3 \rangle \cap_{\text{AFF}} \langle x_1 = 3 \rangle
\]

\[
= \langle x_1 = 3 \land x_2 = 0 \rangle \not\subset_{\text{AFF}} \bot_{\text{AFF}}
\]

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

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 \( \mathsf{AInv} \) 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 wqo 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 \( \mathsf{post} \). Padon et al. [28] consider abstract invariants ranging in a set (of semantics of logical formulae) \( \mathcal{L} \subseteq \varphi(\Sigma) \) and assume (in [28, Theorem 4.2]) that \( \langle \mathcal{L}, \subseteq \rangle \) is closed under finite intersections (i.e., logical conjunctions). Accordingly to Assumption 3.1, we ask that \( \langle \mathcal{L}, \subseteq \rangle \) satisfies the requirement of being an abstract domain of \( \langle \varphi(\Sigma), \subseteq \rangle \), which corresponds to ask that \( \langle \mathcal{L}, \subseteq \rangle \) is closed under arbitrary, rather than finite, intersections. Thus, \( \mathcal{L} \) is the image of
an upper closure \( \tilde{\mu}_L \in \text{uco}(\wp(\Sigma) \subseteq) \) defined by: \( \tilde{\mu}_L(X) \triangleq \cap \{ \phi \in L \mid X \subseteq \phi \} \). 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. \( s \sqsubseteq_L s' \iff \exists \phi \in L \mid s' \subseteq \phi \Rightarrow s \in \phi \) \quad Assumption (A1): \( (\Sigma, \sqsubseteq_L) \) is a wqoset
2. \( Av_L(s) \triangleq \cup \{ \phi \in L \mid \phi \subseteq \neg s \} \) \quad Assumption (A2): \( \forall s \in \Sigma. \ Av_L(s) \subseteq L \)
3. \( \mu^L \) is adjoint to the lco \( L \) acts on any set \( X \) of states

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 \( \text{post}^L : \wp(\Sigma) \to \wp(\Sigma) \) of \( T^L \) as follows:

1. \( \delta_L(X) \triangleq \{ s \in \Sigma \mid \exists s' \in X. s \sqsubseteq_L s' \} \) \quad Down-closure of \( \sqsubseteq_L \)
2. \( Av_L(X) \triangleq \cup \{ \phi \in L \mid \phi \subseteq \neg X \} \) \quad \( Av_L \) acts on any set \( X \) of states
3. \( \text{post}^L(X) \triangleq \text{post}(X) \cup \delta_L(X) \) \quad Successor transformer of \( T^L \)

\section{Co-Inductive Invariants}

Following [28], in the following we make assumption (A2), that is, by Lemma 6.1 (c), we assume that \( L \subseteq \wp(\Sigma) \) is closed under arbitrary unions. This means that \( \mu_L \) is an additive uco on \( \wp(\Sigma) \subseteq \), i.e., in abstract interpretation terminology, \( \mu_L \) 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 \( \mu_L : \wp(\Sigma) \to \wp(\Sigma) \) defined by \( \mu_L(X) \triangleq \cup \phi \in L \mid \phi \subseteq X \). It turns out that the uco \( \mu_L \) is adjoint to the lco \( \mu_L \), namely, \( \mu_L(X) \subseteq Y \iff X \subseteq \mu_L(Y) \) holds. In fact, if \( \mu_L(X) \subseteq Y \) then, by applying \( \mu_L, X \subseteq \mu_L(X) = \mu_L(\mu_L(X)) \subseteq \mu_L(Y) \); the converse is dual.

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

\[
\text{ifp}(\lambda x. c \lor \tilde{f}(x)) \leq c' \iff c \leq \text{gfp}(\lambda x. c' \land \tilde{f}(x))
\]  

(7)

holds and one obtains a co-inductive invariant principle:

\[
c \leq \text{gfp}(\lambda x. \tilde{f}(x) \land c') \iff \exists j \in C. c \leq j \land j \leq \tilde{f}(j) \land j \leq c'
\]  

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

We show how the co-inductive invariant principle (8) applied to the best abstraction transition system $T^\mu$ = $(\Sigma, \mu_L \circ \text{post}_T)$ 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 $T^\mu$.

\begin{itemize}
\item \textbf{Lemma 6.3.} $\text{ifp}(\lambda X. \Sigma_0 \cup \mu_L(\text{post}(X))) = \text{ifp}(\lambda X. \Sigma_0 \cup \text{post}(\mu_L(\Sigma_0)) \cup \mu_L(X))$.
\end{itemize}

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

\begin{align*}
\text{post}(\mu_L(\Sigma_0)) \cup \mu_L(X) \subseteq Y & \Leftrightarrow \text{post}(\mu_L(\Sigma_0)) \subseteq Y \land \mu_L(X) \subseteq Y \Leftrightarrow \\
X \subseteq \mu_L(\pre(Y)) \land X \subseteq \mu_L(Y) & \Leftrightarrow X \subseteq \mu_L(\pre(Y)) \cap \mu_L(Y).
\end{align*}

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

$$\exists \phi \in L. \Sigma_0 \subseteq \phi \land \text{post}(\phi) \subseteq \phi \land \phi \subseteq P \Leftrightarrow \Sigma_0 \subseteq \text{gfp}(\lambda X. \mu_L(\pre(X)) \cap X \cap P))$$

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 $\lambda X. \mu_L(\pre(X)) \cap X \cap P)$ as long as the current iterate $I_1$ contains $\Sigma_0$:

\begin{algorithm}
\caption{Co-inductive backward synthesis of abstract inductive invariants.}
\begin{algorithmic}
\STATE $I_1 := \Sigma$;
\WHILE {$\Sigma_0 \subseteq I_1$} // Loop invariant: $I_1 \in L$
\IF {($I_1 = \mu_L(\pre(I_1)) \cap I_1 \cap P)$} \textbf{return} \textit{$I_1$ is an inductive invariant in $L$}; \ENDIF
\STATE $I_1 := I_1 \cap \mu_L(\pre(I_1)) \cap I_1 \cap P$;
\STATE \textbf{return} no inductive invariant in $L$;
\ENDWHILE
\end{algorithmic}
\end{algorithm}

Since, $\pre$ is computable and, by Lemma 6.1 (b), $\langle \mu_L, \subseteq \rangle = \langle L, \subseteq \rangle$ 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.

\begin{algorithm}
\caption{Algorithm 1 = Algorithm 2.}
\end{algorithm}

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

\begin{algorithm}
\begin{algorithmic}
\STATE \textbf{Algorithm 2} Inductive invariant algorithm by [28].
\STATE \hspace{1em} \( I_2 := \Sigma; \)
\WHILE {\( I_2 \) is not an inductive invariant} \hspace{1em} // Loop invariant: \( I_2 \in L \)
\STATE \hspace{2em} \textbf{if} \( \Sigma_0 \not\subseteq I_2 \) \textbf{then return} no inductive invariant in \( L \);
\STATE \hspace{2em} \textbf{choose} \( s \in \Sigma \) as a counterexample to inductiveness of \( I_2 \);
\STATE \hspace{2em} \( I_2 := I_2 \cap \text{Av}_L(s); \)
\RETURN \( I_2 \) is an inductive invariant in \( L \);
\ENDWHILE
\end{algorithmic}
\end{algorithm}

6.2 Backward and Forward Algorithms

Algorithm 1 is backward because it applies \( \overline{\text{pre}} \), for termination it requires that the abstract domain \( \langle \overline{\mu}_L, \subseteq \rangle \) is DCC and it turns out to be the dual of the forward algorithm \( A_{\text{Inv}} \) provided by Corollary 3.5 for \( \text{post} \) and requiring that \( \langle \overline{\mu}_L, \subseteq \rangle \) is ACC. A different gfp-based forward algorithm can be designed by observing (as in [6, Section 3]) that

\[ \text{lfp}(\lambda X. \Sigma_0 \cup \text{post}(X)) \subseteq P \iff \text{lfp}(\lambda X. \overline{\mu}_L(\overline{\text{post}}(X) \cap X \cap \neg \Sigma_0)) \]

and induces the following co-inductive forward algorithm which relies on the state transformer \( \overline{\text{post}} \) and is terminating when \( \langle \overline{\mu}_L, \subseteq \rangle \) is assumed to be DCC:

\begin{algorithm}
\begin{algorithmic}
\STATE \textbf{Algorithm 3} Co-inductive forward synthesis of abstract inductive invariants.
\STATE \hspace{1em} \( I := \Sigma; \)
\WHILE {\( \neg P \subseteq I \)} \hspace{1em} // Loop invariant: \( I \in L \)
\STATE \hspace{2em} \textbf{if} \( I = \overline{\mu}_L(\overline{\text{post}}(I) \cap I \cap \neg \Sigma_0) \) \textbf{then return} \( I \) is an inductive invariant in \( L \);
\STATE \hspace{2em} \( I := I \cap \overline{\mu}_L(\overline{\text{post}}(I) \cap I \cap \neg \Sigma_0); \)
\RETURN no inductive invariant in \( L \);
\ENDWHILE
\end{algorithmic}
\end{algorithm}

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


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 \( F \subseteq C \rightarrow C \) be a class of monotonic functions, \( S \subseteq C \) be some set of safety properties and \( A \subseteq C \) be an abstract domain of program properties. Let us define:

\[
\text{safe}[F,S] \triangleq \{(f,s) \in F \times S \mid \text{lfp}(f) \leq_C s\}
\]

\[
\text{inv}[F,S,A] \triangleq \{(f,s) \in F \times S \mid \exists a \in A. f a \leq_C a \land a \leq_C s\}
\]

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

\[\text{Corollary A.1. Assume that } A \text{ satisfies Assumption 3.1 for some GI } (C,\alpha,\gamma,A).\]

\[(a) \text{ Assume that } S \subseteq A. \text{ Then, } \text{safe}[F,S] = \text{inv}[F,S,A] \iff \forall f \in F. \alpha(\text{lfp}(f)) = \text{lfp}(\alpha f \gamma).\]

\[(b) \text{ safe}[F,S] = \text{inv}[F,S,A] \iff \forall f \in F. \text{lfp}(f) = \gamma(\text{lfp}(\alpha f \gamma)).\]

\[\text{Proof.}\] Point (a) follows by Theorem 4.1 (a), since \( S \subseteq A \) is assumed to hold. Point (b) follows by Theorem 4.1 (b). \(\square\)

Corollary A.1 therefore provides a precise equivalence of the \( \text{safe} \Rightarrow \text{inv} \) 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 \( a, a' \in A, a \leq_A a' \iff \gamma(a) \leq_C \gamma(a') \) holds.

\[(a) \text{ (}\Leftarrow\text{)} \text{ We have that:}\]

\[
\exists a \in A. f \gamma(a) \leq_C \gamma(a) \land \gamma(a) \leq_C c' \iff \text{[by GC]} \]

\[
\exists a \in A. \alpha f \gamma(a) \leq_A a \land \gamma(a) \leq_C c' \iff \text{[by GC]} \]

\[
\exists a \in A. \text{lfp}(\alpha f \gamma) \leq_A a \land \gamma(a) \leq_C c' \iff \text{[by GI]} \]

\[
\exists a \in A. \gamma(\text{lfp}(\alpha f \gamma)) \leq_C \gamma(a) \land \gamma(a) \leq_C c' \iff \text{[by transitivity]} \]

\[
\gamma(\text{lfp}(\alpha f \gamma)) \leq_C c'.
\]

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

\[(b) \text{ It turns out that:}\]

\[
\exists a \in A. f \gamma(a) \leq_C \gamma(a) \land \gamma(a) \leq_C \gamma(a') \iff \text{[By Lemma 3.2 (a)]} \]

\[
\gamma(\text{lfp}(\alpha f \gamma)) \leq_C \gamma(a') \iff \text{[by GI]} \]

\[
\text{lfp}(\alpha f \gamma) \leq_A a' \quad \square.
\]
Proof of Corollary 3.4. By Lemma 3.2 (a), because $\lambda x \in A.\alpha(c) \lor A.\alpha(f_{\gamma}(x)) = \lambda x \in A.\alpha(c \lor C f_{\gamma}(x))$ is the best correct approximation of $\lambda x \in C.c \lor C f(x)$.

Proof of Corollary 3.5. The hypotheses guarantee that the procedure $\text{AINV}$ is a terminating algorithm, in particular because the sequence of computed iterates $i$ is an ascending chain in $A$. If the algorithm $\text{AINV}$ outputs $i$ then $i = \text{lfp}(\lambda a.\alpha(c) \lor A.\alpha(f_{\gamma}(a))) \leq A.\alpha'(a')$, so that $i = \wedge\{a \in A \mid \alpha(c) \leq A \alpha(f_{\gamma}(i)) \leq A i, i \leq A \alpha'(a')\}$, that is, $i$ is the least inductive invariant in $A$ for $f$ and $\langle c, \gamma(a') \rangle$. If the algorithm $\text{AINV}$ outputs “no abstract inductive invariant for $f$ and $\langle c, \gamma(a') \rangle$” then there exists $j \in \mathbb{N}$ such that $(\lambda a.\alpha(c) \lor A.\alpha(f_{\gamma}(a))) \leq A \alpha(a')$, so that $\text{lfp}(\lambda a.\alpha(c) \lor A.\alpha(f_{\gamma}(a))) \leq A \alpha(a')$, that is, there exists no inductive invariant in $A$ for $f$ and $\langle c, \gamma(a') \rangle$.

Proof of Theorem 4.1. (a) (⇒): By Lemma 3.2 (a).

(⇐): Since $\text{lfp}(f) \leq_C \text{lfp}(f)$ holds, we have that $\exists a \in A. f_{\gamma}(a) \leq_C \gamma(a) \land \gamma(a) \leq_C \text{lfp}(f)$. Thus, by Lemma 3.2 (a), $\gamma(\text{lfp}(\alpha(f_{\gamma}))) \leq_C \text{lfp}(f)$ follows. On the other hand, $\text{lfp}(f) \leq_C \gamma(\text{lfp}(\alpha(f_{\gamma})))$ always holds because the pointwise correctness of $\alpha f_{\gamma}$ implies $\alpha(\text{lfp}(f)) \leq C \text{lfp}(f)$, hence, by GC, $\text{lfp}(f) \leq C \gamma(\text{lfp}(\alpha(f_{\gamma})))$ follows.

(b) (⇒): By Lemma 3.2 (b) because $\text{lfp}(\alpha(f_{\gamma})) \leq C \alpha'(a') \Leftrightarrow \alpha(\text{lfp}(f)) \leq a' \lor \text{lfp}(f) \leq C \gamma(\alpha'(a'))$.

(⇐): We consider $a' \neq \alpha(\text{lfp}(f))$, so that, $\text{lfp}(f) \leq C \gamma(\alpha'(a'))$ holds and by the equivalence of the hypothesis, $\exists a \in A. f_{\gamma}(a) \leq C \gamma(a) \land \gamma(a) \leq C \alpha(\text{lfp}(f))$ holds. This implies, by GI, that $\exists a \in A. \alpha f_{\gamma}(a) \leq_C \alpha(a \land \alpha(\text{lfp}(f)))$. By the inductive invariant principle (1), this implies that (actually, is equivalent to) $\text{lfp}(\alpha(f_{\gamma})) \leq C \alpha(\text{lfp}(f))$. Furthermore, $\alpha(\text{lfp}(f)) \leq_C \text{lfp}(\alpha(f_{\gamma}))$ always holds, therefore proving that $\alpha(\text{lfp}(f)) = \text{lfp}(\alpha(f_{\gamma}))$.

Proof of Lemma 4.2. We have that:

$\exists a \in A. f_{\gamma}(a) \leq_C \gamma(a) \land \gamma(a) \leq_C \alpha(\text{lfp}(f)) \Leftrightarrow$ [by GI]

$\exists a \in A. \alpha f_{\gamma}(a) \leq a \land a \leq_C \alpha(\text{lfp}(f)) \Leftrightarrow$ [by (1) for $\alpha f_{\gamma}$]

$\text{lfp}(\alpha(f_{\gamma})) \leq_C \alpha(\text{lfp}(f)) \Leftrightarrow$ [as $\alpha(\text{lfp}(f)) \leq_C \text{lfp}(\alpha(f_{\gamma}))$]

$\alpha(\text{lfp}(f)) = \text{lfp}(\alpha(f_{\gamma}))$

Proof of Lemma 6.1.

(a) If $s \subseteq L^c$ and $t \in \bar{\mu}_L(s)$ then $t \in \bar{\mu}_L(s') = \emptyset$ where $\emptyset \subseteq \phi \in L$ and $s' \in \phi$ then $s \in \phi$, so that, since $t \in \bar{\mu}_L(s)$, $t \in \phi$. Conversely, if $\bar{\mu}_L(s) \subseteq \bar{\mu}_L(s')$, $\phi \in L$ and $s' \in \phi$, then, since $s \in \bar{\mu}_L(s')$, $s \in \phi$.

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

(⇒): [28, Lemma 4.6] proves $\langle L, \subseteq \rangle$ is well-founded, additionally we show 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 $s_{i,j} \in \phi_i \setminus \phi_j$ and $s_{j,i} \in \phi_j \setminus \phi_i$. From $s_{i,j} \in \phi_j$ we obtain that $\bar{\mu}_L(s_{i,j}) \subseteq \bar{\mu}_L(\phi_i) = \phi_i$. From $s_{i,j} \not\subseteq \phi_j$, we obtain that $s_{i,j} \not\subseteq \bar{\mu}_L(\phi_j)$ holds. It turns out that $\bar{\mu}_L(\{s_{i,j}\}) \subseteq \bar{\mu}_L(s_{i,j})$, otherwise from $s_{i,j} \in \bar{\mu}_L(\{s_{i,j}\}) \subseteq \bar{\mu}_L(\phi_j) \subseteq \phi_j$ we would obtain the contradiction $s_{i,j} \subseteq \phi_j$. Dually, $\bar{\mu}_L(\{s_{j,i}\}) \not\subseteq \bar{\mu}_L(s_{i,j})$ holds. Thus, for any $i \in \mathbb{N}$, $\{\bar{\mu}_L(s_{i,j}) \mid j \in \mathbb{N}, j \neq i\}$ is an infinite antichain in $\langle \{\bar{\mu}_L(s) \mid s \in S, \subseteq \}, \subseteq \rangle$, which is a contradiction.

(⇐): $\{\bar{\mu}_L(s) \mid s \in S, \subseteq \}$ is trivially a wqo because $\{\bar{\mu}_L(s) \mid s \in S, \subseteq \} \subseteq L$ and $L$ is a wqo.

(c) Assume that for all $s \in S$, $A\nu_L(s) \in L$. Let us show that for all $S \subseteq S, \cap_{s \in S} A\nu_L(s) = A\nu_L(S)$. 

CONCUR 2020
(3): Let \( t \in \phi \) for some \( \phi \in L \) such that \( \phi \subseteq \neg S \). Then, for all \( s \in S \), \( \phi \subseteq \text{Av}_L(s) \), so that \( t \in \cap_{s \in S} \text{Av}_L(s) \).

(\( \subseteq \)): Let \( t \in \cap_{s \in S} \text{Av}_L(s) \). 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 \( t \in \text{Av}_L(S) \).

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

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

(d) Since \( \subseteq \) is a quasi-order relation, its down-closure \( \delta_L \) is an upper closure on \( \langle \varphi(\Sigma), \subseteq \rangle \).

By (a), \( \delta_L(X) = \{s \in \Sigma \mid \exists s' \in X.s \subseteq s'\} = \{s \in \Sigma \mid \exists s' \in X.\mu_L(\{s\}) \subseteq \mu_L(\{s\})\} = \{s \in \Sigma \mid \exists s' \in X.s \in \mu_L(\{s'\})\} \subseteq \cup_{s \in X} \mu_L(\{s\}) \).

By (c), since \( L \) is closed under arbitrary unions, the upper closure \( \mu_L \) is additive, so that \( \cup_{s \in X} \mu_L(\{s\}) = \mu_L(\cup_{s \in X} \{s\}) = \mu_L(X) \), consequently \( \delta_L(X) = \delta_L(X) \).

In particular, \( \delta_L(\phi) = \phi \iff \mu_L(\phi) = \phi \iff \phi \in L \).

(e) By Lemma 6.1 (d), \( \lfp(\lambda X. \text{post}(X) \cup \delta_L(X) \cup \Sigma_0) = \lfp(\lambda X. \text{post}(X) \cup \mu_L(X) \cup \Sigma_0) \).

It turns out that
\[
\Sigma_0 \cup \text{post}(X) \cup \delta_L(X) \subseteq X \iff \text{by Lemma 6.1 (d)}
\]
\[
\Sigma_0 \cup \text{post}(X) \cup \mu_L(X) \subseteq X \iff \text{by set theory}
\]
\[
\Sigma_0 \subseteq X \land \text{post}(X) \subseteq X \land \mu_L(X) \subseteq X \iff \text{as } \mu_L \text{ is a uco}
\]
\[
\Sigma_0 \subseteq X \land \text{post}(X) \subseteq X \land \mu_L(X) = X \iff \text{as } \mu_L(X) = X
\]
\[
\Sigma_0 \cup \text{post}(\mu_L(X)) \subseteq X \land \mu_L(X) \subseteq X \iff \text{by set theory}
\]
\[
\Sigma_0 \cup \text{post}(\mu_L(X)) \subseteq X \iff \text{by the equivalences above}
\]
\[
\Sigma_0 \cup \text{post}(\mu_L(X)) \cup \mu_L(X) \subseteq X \iff \text{by set theory}
\]
\[
\Sigma_0 \cup \text{post}(\mu_L(X)) \subseteq X = \mu_L(X) \iff \text{as } \mu_L(X) = X
\]
\[
\Sigma_0 \cup \text{post}(X) \subseteq X \iff \text{as } \mu_L \text{ is a uco}
\]
\[
\mu_L(\Sigma_0 \cup \text{post}(X)) \subseteq X \iff \text{by set theory}
\]
\[
\mu_L(\Sigma_0 \cup \text{post}(X)) \subseteq X \iff \text{by Knaster-Tarski theorem, } \lfp(\mu_L(\Sigma_0 \cup \text{post}(X))) \subseteq \lfp(\lambda X. \Sigma_0 \cup \text{post}(X)) \text{ follows. Moreover, if } F \triangleq \lfp(\mu_L(\Sigma_0 \cup \text{post}(X))), \text{ so that } F = \mu_L(F) = \Sigma_0 \cup \text{post}(F), \text{ then } \Sigma_0 \cup \text{post}(\mu_L(F)) \cup \mu_L(F) = \Sigma_0 \cup \text{post}(F) \cup \mu_L(F) = F, \text{ and this implies that } \lfp(\lambda X. \Sigma_0 \cup \text{post}(\mu_L(X))) \cup \mu_L(X) \subseteq \lfp(\mu_L(\Sigma_0 \cup \text{post}(X))) \text{. Therefore, } \lfp(\lambda X. \Sigma_0 \cup \text{post}(\mu_L(X))) \cup \mu_L(X) = \lfp(\mu_L(\Sigma_0 \cup \text{post}(X))) \text{.} \]

Proof of Corollary 6.2. It turns out that
\[
\lfp(\lambda X. \mu_L(\Sigma_0 \cup \text{post}(X))) \subseteq X \iff \text{by Lemma 3.2 (a) for ucos}
\]
\[
\exists \phi \in \mu_L(\varphi(\Sigma)). \Sigma_0 \cup \text{post}(\phi) \subseteq \phi \land \phi \subseteq X \iff \text{as } \mu_L(\varphi(\Sigma)) = L
\]
\[
\exists \phi \in L. \Sigma_0 \subseteq \phi \land \phi \subseteq X \iff \text{as } \mu_L \text{ is additive and } \Sigma_0 = \Sigma_0 \text{, we also have that } \mu_L(\Sigma_0 \cup \text{post}(X)) = \mu_L(\Sigma_0) \cup \mu_L(\text{post}(X)) = \Sigma_0 \cup \mu_L(\text{post}(X)), \text{ and this allows us to conclude.} \]
Lemma A.2. Let $I \in L$ and $\Sigma_0 \subseteq I$.

(a) there exists a counterexample to inductiveness of $I$ iff $I \not\subseteq \text{pre}(\Sigma) \cap P$ iff $I \neq \mu_L(\text{pre}(\Sigma) \cap I)$.

(b) If $s \in \Sigma$ is a counterexample to inductiveness of $I$ then $\mu_L(\text{pre}(\Sigma) \cap I \cap P) \subseteq \text{Av}_L(s)$.

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

\[
\begin{align*}
I &= \mu_L(\text{pre}(\Sigma) \cap I \cap P) \iff [\text{as } I \in L] \\
I &= \mu_L(I) = \mu_L(\text{pre}(\Sigma) \cap I \cap P) \iff [\text{as } \text{pre}(I) \cap I \subseteq I] \\
I &= \mu_L(I) \subseteq \mu_L(\text{pre}(\Sigma) \cap I \cap P) \iff [\text{as } \mu_L \text{ is a lco}] \\
I &= \mu_L(I) \subseteq \text{pre}(\Sigma) \cap I \cap P \iff \\
I &\subseteq \text{pre}(\Sigma) \cap I \cap P
\end{align*}
\]

(b) The proof of point (a) shows that if $s \in \Sigma$ is a counterexample to inductiveness of $I$ then $s \in I$ and $s \not\in \text{pre}(I) \cap P$. Then, $\text{pre}(I) \cap P \subseteq \neg\{s\}$, so that, by monotonicity of $\mu_L$, $\mu_L(\text{pre}(I) \cap P) \subseteq \mu_L(\neg\{s\})$ and, in turn, $\mu_L(\text{pre}(I) \cap I \cap P) \subseteq \mu_L(\neg\{s\}) = \text{Av}_L(s)$.

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

\[
\begin{align*}
I_4 &:= \Sigma; \\
\text{while } \Sigma_0 \subseteq I_4 \text{ do } \text{// Invariant: } I_4 \in L \\
&\quad \text{if } (I_4 \setminus (\text{pre}(I_4) \cap P) = \emptyset) \text{ then return } I_4 \text{ is an inductive invariant in } L; \\
&\quad \text{choose } s \in I_4 \setminus (\text{pre}(I_4) \cap P); \\
&\quad I_4 := I_4 \cap \mu_L(\{s\}); \text{ return no inductive invariant in } L;
\end{align*}
\]

Algorithm 4 returns $I_4 \in L$ iff $I_4 = \text{gfp}(\lambda X. \mu_L(\text{pre}(X) \cap X \cap P))$. In this case, by Lemma A.2 (a), since $\Sigma_0 \subseteq I_4$ holds, $I_4$ 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 $I_4 \subseteq \text{pre}(I_4) \cap P$ iff $I_4 = \mu_L(\text{pre}(I_4) \cap I_4 \cap P)$, 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), $I_k^n \subseteq I_k^0 = I_k^1$, so that $I_1 \subseteq I_4 = I_2$. Since $I_k$ are fixpoints of $\lambda X. \mu_L(\text{pre}(X) \cap X \cap P)$ and $I_1$ is the greatest fixpoint, it turns out that $I_1 = I_4 = I_2$. ▶