Correctness Kernels of Abstract Interpretations

Roberto Giacobazzi[a], Francesco Ranzato[b]

[a]*Dipartimento di Informatica, University of Verona, Italy*
[b]*Dipartimento di Matematica, University of Padova, Italy*

Abstract

In abstract interpretation-based static analysis, approximation is encoded by abstract domains. They provide systematic guidelines for designing abstract semantic functions that approximate some concrete system behaviors under analysis. It may happen that an abstract domain contains redundant information for the specific purpose of approximating a given concrete semantic function. This paper introduces the notion of correctness kernel of an abstract interpretation, a methodology for simplifying abstract domains, i.e. removing abstract values from them, in a maximal way while retaining exactly the same approximate behavior of the system under analysis. We show that in abstract model checking correctness kernels provide a simplification paradigm of the abstract state space that is guided by examples, meaning that this simplification preserves spuriousness of examples (i.e., abstract paths). In particular, we show how correctness kernels can be integrated with the well-known CEGAR (CounterExample-Guided Abstraction Refinement) methodology.

1. Introduction

In static analysis and verification, model-driven *abstraction refinement* has emerged in the last decade as a key paradigm for enhancing abstractions towards more precise yet efficient analyses. The underlying basic idea is simple: given an abstraction A modeling some approximate properties of a system to analyze, in order to remove some artificial computations that may arise in the analysis based on A refine A by considering how the concrete model actually behaves when false alarms or spurious traces are encountered. The general idea of using spurious counterexamples for refining an abstraction stems from the CounterExample-Guided Abstraction Refinement (CEGAR) paradigm [4, 5]. The concrete model here drives the automatic identification of prefixes of the counterexample abstract path that do not correspond to an actual trace, by isolating abstract (failure) states that need to be refined in order to eliminate that spurious counterexample. Model-driven refinement strategies, such as CEGAR, provide algorithmic methods for achieving abstractions that are complete (i.e., precise [15, 19]) with respect to some given property of the concrete model.

We investigate here the dual problem of *abstraction simplification*. Instead of refining abstractions in order to eliminate spurious traces, our goal is to modify an abstraction A towards a simpler (ideally, the simplest) model A_s that gives rise to the same approximate system behavior as A does. In abstract model checking, this abstraction simplification has *to keep the same examples* of the concrete system in the following sense. Recall that an abstract path π in an abstract transition system $\mathcal{A}$ is *spurious* when no real concrete path is abstracted to π. Assume that a given abstract state space A of a system $\mathcal{A}$ gets simplified to A_s and thus gives rise to a more

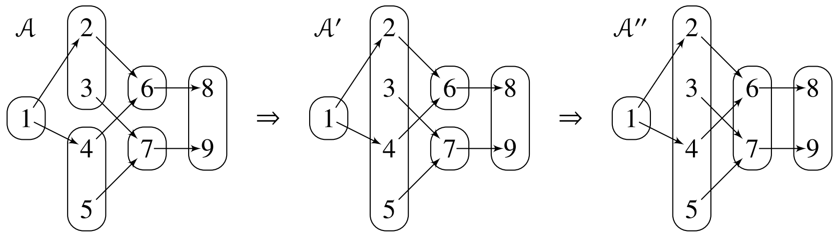
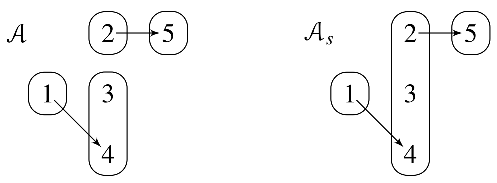


Figure 1: Some abstract transition systems.

abstract system $\mathcal{A}_s$. Then, we say that $\mathcal{A}_s$ keeps the same examples of $\mathcal{A}$ when the following condition is satisfied: if π_{A_s} is a spurious path in the simplified abstract system $\mathcal{A}_s$ then there exists a spurious path π_A in the original system $\mathcal{A}$ which is abstracted to π_{A_s}. Obviously, if $\mathcal{A}_s$ is a generic simplification of $\mathcal{A}$ then $\mathcal{A}_s$ does not necessarily keep the same examples of $\mathcal{A}$. In the following, we depict abstract transition systems by diagrams where integer numbers denote concrete states, arrows connect concrete states in a transition relation, and oval shapes indicate blocks (denoted by square brackets) of a state partition. In the example below, for the spurious path $\pi_{A_s} = \langle[1], [2,3,4], [5]\rangle$ in $\mathcal{A}_s$ there is no corresponding spurious path in $\mathcal{A}$ which can be abstracted to π_{A_s} and therefore the simplification $\mathcal{A}_s$ does not keep the same examples of $\mathcal{A}$.



Such a methodology is called EGAS, Example-Guided Abstraction Simplification, since this abstraction simplification is able to keep examples according to the meaning above. Let us illustrate how EGAS works through a simple example. Let us consider the abstract transition system $\mathcal{A}$ in Figure 1, where $\{[1], [2,3], [4,5], [6], [7], [8,9]\}$ is the underlying state partition. The abstract state space of $\mathcal{A}$ is simplified by merging the blocks $[2,3]$ and $[4,5]$: EGAS guarantees that this can be safely done because $\text{pre}^\sharp([2,3]) = \text{pre}^\sharp([4,5]) = \{[1]\}$ and $\text{post}^\sharp([2,3]) = \text{post}^\sharp([4,5]) = \{[6], [7]\}$, where $\text{pre}^\sharp$ and $\text{post}^\sharp$ denote, respectively, the abstract predecessor and successor functions. This abstraction simplification leads to the abstract system $\mathcal{A}'$ in Figure 1. Observe that the abstract path $\pi = \langle[1], [2,3,4,5], [7], [8,9]\rangle$ in $\mathcal{A}'$ is spurious because there is no concrete path whose abstraction in $\mathcal{A}'$ is π, while π is instead the abstraction of the spurious path $\langle[1], [4,5], [7], [8,9]\rangle$ in $\mathcal{A}$. On the other hand, consider the path $\sigma = \langle[1], [2,3,4,5], [6], [8,9]\rangle$ in $\mathcal{A}'$ and observe that all the paths in $\mathcal{A}$ that are abstracted to σ, i.e. $\langle[1], [2,3], [6], [8,9]\rangle$ and $\langle[1], [4,5], [6], [8,9]\rangle$, are not spurious. This is consistent with the fact that σ actually is not a spurious path. Likewise, $\mathcal{A}'$ can be further simplified to the abstract system $\mathcal{A}''$ where the blocks $[6]$ and $[7]$ are merged. This transform also keeps examples because now there is no spurious path in $\mathcal{A}''$. Let us also notice that if $\mathcal{A}$ would get simplified to an abstract system $\mathcal{A}'''$ by merging the blocks $[1]$ and $[2,3]$ into a new abstract state $[1,2,3]$ then this transform would not keep examples because we would obtain the spurious loop path

$\tau = \langle [1,2,3], [1,2,3], [1,2,3], ... \rangle$ in $\mathcal{A}'''$ (because, in $\mathcal{A}'''$, $[1,2,3]$ would have a self-loop) while there is no corresponding spurious abstract path in $\mathcal{A}$ whose abstraction in $\mathcal{A}'''$ is τ.

We show how EGAS can be formalized within the standard Galois connection-based abstract interpretation framework [9, 10]. Consider for instance the two following basic abstract domains A_1 and A_2 for sign analysis of an integer program variable, so that the concrete domain of values is the powerset $\wp(\mathbb{Z})$ of integer numbers.

$$\begin{tikzcd}[column sep=small]
 & & \mathbb{Z} \arrow[dl, dash] \arrow[dr, dash] & & & & \mathbb{Z} \arrow[dd, dash] & \\
A_1 & \mathbb{Z}_{\leq 0} \arrow[dr, dash] & & \mathbb{Z}_{\geq 0} \arrow[dl, dash] & & & & A_2 \\
 & & 0 & & & & \mathbb{Z}_{\geq 0} &
\end{tikzcd}$$

Recall that in abstract interpretation the best correct approximation of a semantic function f on an abstract domain A is given by $f^A \triangleq \alpha \circ f \circ \gamma$, where α and γ are the abstraction and concretization maps defining A [9, 10]. Let us consider a simple operation of increment x++ on an integer variable x. In this case, the best correct approximations on the abstractions A_1 and A_2 go as follows:

$$\begin{aligned}
{++}^{A_1} &= \{0 \mapsto \mathbb{Z}_{\geq 0},\ \mathbb{Z}_{\leq 0} \mapsto \mathbb{Z},\ \mathbb{Z}_{\geq 0} \mapsto \mathbb{Z}_{\geq 0},\ \mathbb{Z} \mapsto \mathbb{Z}\},\\
{++}^{A_2} &= \{\mathbb{Z}_{\geq 0} \mapsto \mathbb{Z}_{\geq 0},\ \mathbb{Z} \mapsto \mathbb{Z}\}.
\end{aligned}$$

We observe that the best correct approximations of ++ in A_1 and A_2 encode the same function, meaning that the approximations of the operation ++ in A_1 and A_2 are equivalent: In fact, we have that $\gamma_{A_1} \circ {++}^{A_1} \circ \alpha_{A_1}$ and $\gamma_{A_2} \circ {++}^{A_2} \circ \alpha_{A_2}$ denote exactly the same function in $\wp(\mathbb{Z}) \to \wp(\mathbb{Z})$. In other terms, the abstract domain A_1 contains some abstract values, i.e. 0 and $\mathbb{Z}_{\leq 0}$, that are unnecessary for the specific aim of approximating the increment operation on A_1; of course, these "redundant" abstract values might be instead needed in A_1 for different purposes, e.g. for approximating further operations.

We formalize this simplification process of an abstract domain relatively to a semantic function in standard Galois connection-based abstract interpretation. This allows us to provide, for generic continuous semantic functions, a systematic and constructive method, that we call a *correctness kernel*, for simplifying a given abstraction A relatively to a given semantic function f towards the unique minimal abstract domain that induces an equivalent approximate behavior of f as in A.

We show how correctness kernels can be embedded within the CEGAR methodology by providing a novel refinement heuristic in a CEGAR iteration step which turns out to be more accurate than the basic refinement heuristic [5]. We also describe how correctness kernels may be applied in predicate abstraction-based model checking [12, 20] for reducing the search space without applying Ball et al.'s [2] Cartesian abstractions, which typically yield additional loss of precision.

This is an extended and revised version of the conference paper [18].

2. Correctness Kernels

2.1. Abstract Interpretation Background

Abstract Domains. In standard abstract interpretation [9, 10], abstract domains (or abstractions) are specified by Galois connections/insertions (GCs/GIs for short) or, equivalently, adjunctions.

Concrete and abstract domains, $\langle C, \leq_C \rangle$ and $\langle A, \leq_A \rangle$, are assumed to be complete lattices which are related by abstraction and concretization maps $\alpha : C \to A$ and $\gamma : A \to C$ that give rise to an adjunction (α, C, A, γ): for all $a \in A$ and $c \in C$, $\alpha(c) \leq_A a \Leftrightarrow c \leq_C \gamma(a)$. A GC is a GI when $\alpha \circ \gamma = \lambda x.x$. It is known that the function $\mu_A \triangleq \gamma \circ \alpha : C \to C$ is an upper closure operator (uco) on C [10], i.e. a monotone, idempotent and increasing function. GIs of a common concrete domain C are preordered w.r.t. their relative precision as usual: $\mathcal{G}_1 = (\alpha_1, C, A_1, \gamma_1) \sqsubseteq \mathcal{G}_2 = (\alpha_2, C, A_2, \gamma_2)$ — i.e. A_1/A_2 is a refinement/simplification of A_2/A_1 — iff $\{\gamma_2(\alpha_2(c)) \mid c \in C\} \subseteq \{\gamma_1(\alpha_1(c)) \mid c \in C\}$, which is more compactly denoted using images of functions by $\gamma_2(\alpha_2(C)) \subseteq \gamma_1(\alpha_1(C))$. Moreover, $\mathcal{G}_1$ and $\mathcal{G}_2$ are equivalent when $\mathcal{G}_1 \sqsubseteq \mathcal{G}_2$ and $\mathcal{G}_2 \sqsubseteq \mathcal{G}_1$. We denote by $\mathrm{Abs}(C)$ the family of abstract domains of C, as defined by a GI, up to this latter equivalence. It is well known that $\langle \mathrm{Abs}(C), \sqsubseteq \rangle$ is a complete lattice [10]. Thus, one can consider the most concrete simplification (i.e., lub $\sqcup$) and the most abstract refinement (i.e., glb $\sqcap$) of any family of abstract domains. Abstract domains can be equivalently defined as uco's, meaning that any GI (α, C, A, γ) induces the uco μ_A, any uco $\mu : C \to C$ induces the GI $(\mu, C, \mu(C), \lambda x.x)$, and these two transforms are the inverse of each other, namely $(\mu_A, C, \mu_A(C), \lambda x.x)$ and (α, C, A, γ) are equivalent GIs and $\mu = \mu_{\mu(C)}$. In more technical terms, $\langle \mathrm{Abs}(C), \sqsubseteq \rangle$ is isomorphic to the complete lattice $\langle \mathrm{uco}(C), \sqsubseteq \rangle$ of uco's on C, where $\sqsubseteq$ denotes the standard point-wise ordering between functions, so that lub's and glb's of abstractions can be equivalently characterized in $\mathrm{uco}(C)$. Let us also recall that each closure $\mu \in \mathrm{uco}(C)$ is uniquely determined by its image $\mathrm{img}(\mu) = \mu(C)$ as follows: for any $x \in C$, $\mu(x) = \wedge\{y \in C \mid y \in \mu(C),\ x \leq y\}$. Moreover, a subset $X \subseteq C$ is the image of some uco on C iff X is meet-closed, i.e. $X = \mathrm{Cl}_\wedge(X) \triangleq \{\wedge Y \mid Y \subseteq X\}$ (note that $\top_C = \wedge\varnothing \in \mathrm{Cl}_\wedge(X)$). This allows us to equivalently use uco's both as functions in $C \to C$ or as subsets of C; this does not give rise to ambiguity, since one can distinguish their use as functions or sets according to the context. Hence, if $A, B \in \mathrm{Abs}(C)$ are two abstractions of C then they can be viewed as images of two uco's on C, denoted respectively by μ_A and μ_B, so that A is a refinement of B when $\mathrm{img}(\mu_B) \subseteq \mathrm{img}(\mu_A)$. Let us also recall that given a family of uco's $\mathcal{X} \subseteq \mathrm{uco}(C)$, then its lub and glb can be characterized as follows: $\sqcup \mathcal{X} = \cap_{\mu \in \mathcal{X}} \mathrm{img}(\mu)$ and $\sqcap \mathcal{X} = \mathrm{Cl}_\wedge(\cup_{\mu \in \mathcal{X}} \mathrm{img}(\mu))$.

Abstract Functions. Let $f : C \to C$ be some concrete semantic function — for simplicity, we consider 1-ary functions — and let $f^\sharp : A \to A$ be a corresponding abstract function defined on some abstraction $A \in \mathrm{Abs}(C)$. Then, $\langle A, f^\sharp \rangle$ is a sound abstract interpretation when $\alpha \circ f \sqsubseteq f^\sharp \circ \alpha$ holds. Moreover, the abstract function $f^A \triangleq \alpha \circ f \circ \gamma : A \to A$ is called the *best correct approximation* (b.c.a.) of f on A because any abstract interpretation $\langle A, f^\sharp \rangle$ is sound iff for any $a \in A$, $f^A(a) \leq f^\sharp(a)$. Hence, for any abstraction A, f^A plays the role of the best possible approximation of f on the abstract domain A.

2.2. *The Problem*

Given a semantic function $f : C \to C$ on some concrete domain C and an abstraction $A \in \mathrm{Abs}(C)$, does there exist the *most abstract domain* that induces the same best correct approximation of f as A does?

Let us formalize the above question. Consider two abstractions $A, B \in \mathrm{Abs}(C)$. We say that A and B induce the same best correct approximation of f when f^A and f^B approximate any concrete computation $f(c)$ in the same way, namely, for any $c \in C$, $\gamma_A(f^A(\alpha_A(c))) = \gamma_B(f^B(\alpha_B(c)))$. By recalling that μ_A and μ_B denote the corresponding uco's, this definition boils down to the following equation:

$$\mu_A \circ f \circ \mu_A = \mu_B \circ f \circ \mu_B.$$

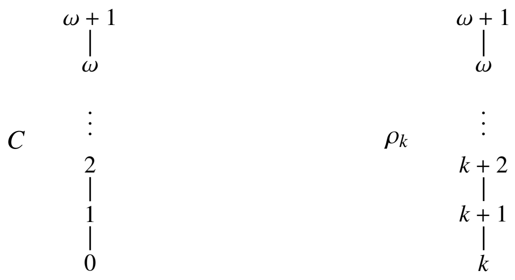


Figure 2: Concrete and abstract domains.

In order to keep the notation easy, this is denoted simply by $f^A = f^B$. Also, if $F \subseteq C \to C$ is a set of concrete functions then $F^A = F^B$ means that for any $f \in F$, $f^A = f^B$.

Given $A \in \mathrm{Abs}(C)$, the domain $\sqcup\{B \in \mathrm{Abs}(C) \mid F^B = F^A\}$ is precisely the lub in Abs(C) of all the abstractions that induce the same best correct approximations of F as A does. Hence, our question is formalized through the following notion of correctness kernel.

Definition 2.1. Given $F \subseteq C \longrightarrow C$, define $\mathcal{K}_F : \mathrm{Abs}(C) \to \mathrm{Abs}(C)$ as

$$\mathcal{K}_F(A) \triangleq \sqcup\{B \in \mathrm{Abs}(C) \mid F^B = F^A\}.$$

If $F^{\mathcal{K}_F(A)} = F^A$ then $\mathcal{K}_F(A)$ is called the *correctness kernel* of A for F. □

A correctness kernel $\mathcal{K}_F(A)$, when it exists, is therefore a simplification of the abstraction A. However, in general, a correctness kernel may not exist, as the following technical example shows.

Example 2.2. Let us consider the concrete domain C depicted in Figure 2, namely the ordinal numbers less than or equal to $\omega + 1$. Let $f : C \to C$ be defined as follows:

$$f(x) \triangleq \begin{cases} \omega & \text{if } x \notin \{\omega, \omega+1\} \\ \omega+1 & \text{otherwise} \end{cases}$$

Let $\mu \in \mathrm{uco}(C)$ be the identity uco $\lambda x.x$, so that $\mu \circ f \circ \mu = f$. Let us define $\mu_f^K \triangleq \sqcup\{\eta \in \mathrm{uco}(C) \mid \eta \circ f \circ \eta = \mu \circ f \circ \mu\}$, so that the correctness kernel of μ for f exists iff $\mu_f^K \circ f \circ \mu_f^K = \mu \circ f \circ \mu$. For any $k \geq 0$, consider $\rho_k \in \mathrm{uco}(C)$ defined as $\rho_k \triangleq C \smallsetminus [0, k)$, which is also depicted in Figure 2. It is easily seen that, for any $k \geq 0$, $\rho_k \circ f \circ \rho_k = f = \mu \circ f \circ \mu$. Hence, it turns out that for any $k \geq 0$, $\rho_k \sqsubseteq \mu_f^K$. Let us define $\hat{\rho} \triangleq \sqcup_{k \geq 0} \rho_k \in \mathrm{uco}(C)$. Hence, it turns out that $\hat{\rho} \sqsubseteq \mu_f^K$. Moreover, let us observe that $\hat{\rho} = \cap_{k \geq 0} \mathrm{img}(\rho_k) = \{\omega, \omega+1\}$, so that, for any $x \leq \omega$, $\hat{\rho}(x) = \omega$ while $\hat{\rho}(\omega+1) = \omega+1$. Since $\hat{\rho} \sqsubseteq \mu_f^K$ and $\hat{\rho} = \{\omega, \omega+1\}$, there are only two possibilities for the uco μ_f^K:

$$(1)\ \mu_f^K = \{\omega, \omega+1\}; \qquad (2)\ \mu_f^K = \{\omega+1\}.$$

In both cases (1) and (2), it is easy to check that $\mu_f^K \circ f \circ \mu_f^K = \lambda x.\omega + 1 \neq \mu \circ f \circ \mu$. As a consequence, the correctness kernel of μ for f does not exist. □

It is also worth remarking that the corresponding dual abstraction refinement may not exist even for finite concrete domains (and, consequently, finite abstract domains). In other terms, the dual question on the existence of the most concrete abstraction that induces the same best correct approximation of f as A has a negative answer even under the strong hypothesis of having a finite concrete domain C, as shown by the following simple example.

Example 2.3. Consider the finite lattice C depicted below.

$$\begin{tikzcd}
 & 5 & \\
3 \arrow[ur, dash] & & 4 \arrow[ul, dash] \\
 & 2 \arrow[ul, dash] \arrow[ur, dash] \arrow[d, dash] & \\
 & 1 &
\end{tikzcd}$$

Let us consider the monotonic function $f : C \to C$ defined as $f = \{1 \mapsto 1, 2 \mapsto 1, 3 \mapsto 5, 4 \mapsto 5, 5 \mapsto 5\}$. Let us consider the abstraction $A \in \mathrm{Abs}(C)$ defined as $A = \{1, 5\}$, so that the corresponding uco $\mu \in \mathrm{uco}(C)$ is the function: $\mu = \{1 \mapsto 1, 2 \mapsto 5, 3 \mapsto 5, 4 \mapsto 5, 5 \mapsto 5\}$. It is immediate to observe that $\mu \circ f \circ \mu = \{1 \mapsto 1, 2 \mapsto 5, 3 \mapsto 5, 4 \mapsto 5, 5 \mapsto 5\}$. Consider now the abstractions $\rho_1 = \{1, 3, 5\}$ and $\rho_2 = \{1, 4, 5\}$ and observe that $\rho_i \circ f \circ \rho_i = \mu \circ f \circ \mu$. However, we have that $\rho_1 \sqcap \rho_2 = \lambda x.x$, because the image of $\rho_1 \sqcap \rho_2$ is $\mathrm{Cl}_\wedge(\rho_1 \cup \rho_2) = \{1, 2, 3, 4, 5\}$. Hence, $(\rho_1 \sqcap \rho_2) \circ f \circ (\rho_1 \sqcap \rho_2) = f \neq \mu \circ f \circ \mu$. Let $\rho_r = \sqcap\{\rho \in \mathrm{uco}(C) \mid \rho \circ f \circ \rho = \mu \circ f \circ \mu\}$. Thus, $\rho_r = \lambda x.x$ and, in turn, $\rho_r \circ f \circ \rho_r \neq \mu \circ f \circ \mu$. Consequently, the most concrete abstraction that induces the same best correct approximation of f as μ does not exist. □

3. Characterization of Correctness Kernels

Our key technical result provides a *constructive* characterization of the property of "having the same b.c.a." for two abstract domains that are comparable w.r.t. the precision ordering. In the following, for any function $f : X \to Y$ and any subset $S \subseteq Y$, $f^{-1}(T) \triangleq \{x \in X \mid f(x) \in T\}$ denotes the inverse image of T for f. Moreover, given a complete lattice C and any subset $S \subseteq C$, $\max(S) \triangleq \{x \in S \mid \forall y \in S.\ x \leq_C y \Rightarrow x = y\}$ denotes the set of maximal elements of S in C. Also, $\downarrow y$ denotes the downward closure of some $y \in C$, i.e. $\downarrow y \triangleq \{x \in C \mid x \leq_C y\}$.

Lemma 3.1. *Let $f : C \to C$ and $A, B \in \mathrm{Abs}(C)$ such that $A \sqsubseteq B$. Assume that the function $f \circ \gamma_A \circ \alpha_A : C \to C$ is continuous (i.e., preserves lub's of chains in C). Then,*

$$f^B = f^A \;\Leftrightarrow\; \gamma_A\Big(\mathrm{img}(f^A) \cup \textstyle\bigcup_{y \in A} \max(\{x \in A \mid f^A(x) \leq_A y\})\Big) \subseteq \gamma_B(B).$$

Proof. Let μ and ρ be the uco's on C induced by, respectively, the abstractions A and B, so that $A \sqsubseteq B$ corresponds to $\mu \sqsubseteq \rho$. Let us recall (see e.g. [8, Proposition 4.2.3.0.1]) that $\mu \sqsubseteq \rho$ implies $\mu \circ \rho = \rho = \rho \circ \mu$.
Given $y \in A$, let us show that

$$\gamma_A(\max(\{x \in A \mid f^A(x) \leq_A y\})) = \max(\{x \in C \mid f(\mu(x)) \leq_C \gamma_A(y)\}).$$

($\subseteq$) Let $z \in \max(\{x \in A \mid f^A(x) \leq_A y\})$. Then:

$$\begin{aligned}
\alpha_A(f(\gamma_A(z))) \leq_A y \Leftrightarrow &\quad [\text{by adjunction } \alpha_A/\gamma_A] \\
f(\gamma_A(z)) \leq_C \gamma_A(y) \Leftrightarrow &\quad [\text{since } \alpha_A(\gamma_A(z)) = z] \\
f(\gamma_A(\alpha_A(\gamma_A(z)))) \leq_C \gamma_A(y) &
\end{aligned}$$

so that $\gamma_A(z) \in \{x \in C \mid f(\mu(x)) \leq_C \gamma_A(y)\}$. Consider now $w \in \{x \in C \mid f(\mu(x)) \leq_C \gamma_A(y)\}$ such that $\gamma_A(z) \leq_C w$. Thus, since $f(\mu(w)) \leq_C \gamma_A(y) \Leftrightarrow f^A(\alpha_A(w)) \leq_A y$ and $\gamma_A(z) \leq w \Leftrightarrow z \leq_A \alpha_A(w)$, by maximality of z, we obtain that $z = \alpha_A(w)$. Hence, by applying γ_A to this latter equality, we derive that $w \leq_C \gamma_A(\alpha_A(w)) = \gamma_A(z)$, which, together with $\gamma_A(z) \leq_C w$, implies $\gamma_A(z) = w$. We have therefore shown that $\gamma_A(z) \in \max(\{x \in C \mid f(\mu(x)) \leq_C \gamma_A(y)\})$.
($\supseteq$) Let $z \in \max(\{x \in C \mid f(\mu(x)) \leq_C \gamma_A(y)\})$. Then, $f(\gamma_A(\alpha_A(z))) \leq_C \gamma_A(y)$, hence, by adjunction α_A/γ_A, we obtain $\alpha_A(f(\gamma_A(\alpha_A(z)))) \leq_A y$, so that $\alpha_A(z) \in \{x \in A \mid f^A(x) \leq_A y\}$. Consider now $w \in \{x \in A \mid f^A(x) \leq_A y\}$ such that $\alpha_A(z) \leq_A w$. Then:

$$\begin{aligned} \alpha_A(f(\gamma_A(w))) \leq_A y \Leftrightarrow &\quad \text{[by adjunction } \alpha_A/\gamma_A] \\ f(\gamma_A(w)) \leq_C \gamma_A(y) \Leftrightarrow &\quad \text{[since } \alpha_A(\gamma_A(w)) = w] \\ f(\gamma_A(\alpha_A(\gamma_A(w)))) \leq_C \gamma_A(y) & \end{aligned}$$

so that $\gamma_A(w) \in \{x \in C \mid f(\mu(x)) \leq_C \gamma_A(y)\}$. Since $\alpha_A(z) \leq_A w \Leftrightarrow z \leq_C \gamma_A(w)$, by maximality of z, we therefore obtain $z = \gamma_A(w)$. Thus, by applying α_A to this latter equality, $\alpha_A(z) = \alpha_A(\gamma_A(w)) = w$. We have thus shown that $\alpha_A(z) \in \max(\{x \in A \mid f^A(x) \leq_A y\})$. Moreover:

$$\begin{aligned} f(\gamma_A(\alpha_A(\gamma_A(\alpha_A(z))))) = &\quad \text{[since } \gamma_A \circ \alpha_A \text{ is idempotent]} \\ f(\gamma_A(\alpha_A(z))) \leq_C &\quad \text{[by definition of } z] \\ \gamma_A(y) & \end{aligned}$$

so that $\gamma_A(\alpha_A(z)) \in \{x \in C \mid f(\mu(x)) \leq_C \gamma_A(y)\}$. Thus, since $z \leq_C \gamma_A(\alpha_A(z))$, by maximality of z we obtain $z = \gamma_A(\alpha_A(z))$. We have therefore shown that $z \in \gamma_A(\max(\{x \in A \mid f^A(x) \leq_A y\}))$.

Using the downward closure notation, $\{x \in C \mid f(\mu(x)) \leq_C \gamma_A(y)\}$ can be rewritten as the inverse image of $\downarrow\gamma_A(y)$ for the function $f \circ \mu$, i.e. $(f \circ \mu)^{-1}(\downarrow\gamma_A(y))$. Using this notation, we have shown above that $\gamma_A(\max(\{x \in A \mid f^A(x) \leq_A y\})) = \max((f \circ \mu)^{-1}(\downarrow\gamma_A(y)))$. Also, note that $\gamma_A(\mathrm{img}(f^A)) = \gamma_A(\alpha_A(f(\gamma_A(A)))) = \gamma_A(\alpha_A(f(\gamma_A(\alpha_A(C))))) = \mu(f(\mu(C)))$. Hence,

$$\gamma_A(\mathrm{img}(f^A) \cup \textstyle\bigcup_{y\in A} \max(\{x \in A \mid f^A(x) \leq_A y\})) = \mu(f(\mu(C))) \cup \textstyle\bigcup_{y\in\mu} \max((f \circ \mu)^{-1}(\downarrow y)).$$

Since $\gamma_B(B) = \rho$, the lemma can be proved by the following equivalent statement which is formalized only through uco's:

$$\rho \circ f \circ \rho = \mu \circ f \circ \mu \;\text{ iff }\; \mu(f(\mu(C))) \cup \textstyle\bigcup_{y\in\mu} \max((f \circ \mu)^{-1}(\downarrow y)) \subseteq \rho.$$

Let us first prove that

$$\rho \circ f \circ \rho = \mu \circ f \circ \mu \;\Leftrightarrow\; \rho \circ f \circ \mu = \mu \circ f \circ \mu = \mu \circ f \circ \rho \tag{$*$}$$

($\Rightarrow$) On the one hand,

$$\begin{aligned} \mu \circ f \circ \mu = \rho \circ f \circ \rho \Rightarrow &\quad \text{[by applying } \rho \text{ to both sides]} \\ \rho \circ \mu \circ f \circ \mu = \rho \circ \rho \circ f \circ \rho \Rightarrow &\quad \text{[because } \rho \circ \mu = \rho \text{ and } \rho \circ \rho = \rho] \\ \rho \circ f \circ \mu = \rho \circ f \circ \rho \Rightarrow &\quad \text{[by hypothesis]} \\ \rho \circ f \circ \mu = \mu \circ f \circ \mu & \end{aligned}$$

and on the other hand,

$$\begin{aligned}
\mu \circ f \circ \mu = \rho \circ f \circ \rho &\Rightarrow && \text{[by applying } \rho \text{ in front to both sides]}\\
\mu \circ f \circ \mu \circ \rho = \rho \circ f \circ \rho \circ \rho &\Rightarrow && \text{[because } \mu \circ \rho = \rho \text{ and } \rho \circ \rho = \rho\text{]}\\
\mu \circ f \circ \rho = \rho \circ f \circ \rho &\Rightarrow && \text{[by hypothesis]}\\
\mu \circ f \circ \rho = \mu \circ f \circ \mu &
\end{aligned}$$

so that $\rho \circ f \circ \mu = \mu \circ f \circ \mu = \mu \circ f \circ \rho$.

($\Leftarrow$) We have that:

$$\begin{aligned}
\rho \circ f \circ \mu = \mu \circ f \circ \rho &\Rightarrow && \text{[by applying } \rho \text{ to both sides]}\\
\rho \circ \rho \circ f \circ \mu = \rho \circ \mu \circ f \circ \rho &\Rightarrow && \text{[since } \rho \circ \rho = \rho \text{ and } \rho \circ \mu = \rho\text{]}\\
\rho \circ f \circ \mu = \rho \circ f \circ \rho &\Rightarrow && \text{[by hypothesis]}\\
\mu \circ f \circ \mu = \rho \circ f \circ \rho. &
\end{aligned}$$

Let us now observe that $\rho \circ f \circ \mu = \mu \circ f \circ \mu \Leftrightarrow \mu(f(\mu(C))) \subseteq \rho$: in fact, since $\rho = \rho \circ \mu$, we have that $\rho \circ f \circ \mu = \mu \circ f \circ \mu \Leftrightarrow \rho \circ \mu \circ f \circ \mu = \mu \circ f \circ \mu$, and this latter equation is clearly equivalent to $\mu(f(\mu(C))) \subseteq \rho$.
Moreover, since $\rho = \mu \circ \rho$, we have that $\mu \circ f \circ \mu = \mu \circ f \circ \rho$ is equivalent to $\mu \circ (f \circ \mu) = \mu \circ (f \circ \mu) \circ \rho$. This latter equation states the completeness of the pair of abstractions $\langle \rho, \mu \rangle$ for the function $f \circ \mu$. By the characterization of completeness in [19, Lemma 4.2], since, by hypothesis, $f \circ \mu$ is continuous, we have that the completeness equation $\mu \circ (f \circ \mu) = \mu \circ (f \circ \mu) \circ \rho$ is equivalent to $\cup_{y \in \mu} \max((f \circ \mu)^{-1}(\downarrow y)) \subseteq \rho$. Thus, $\mu \circ f \circ \mu = \mu \circ f \circ \rho \Leftrightarrow \cup_{y \in \mu} \max((f \circ \mu)^{-1}(\downarrow y)) \subseteq \rho$.
Summing up, we have shown that

$$\rho \circ f \circ \mu = \mu \circ f \circ \mu = \mu \circ f \circ \rho \;\Leftrightarrow\; \mu(f(\mu(C))) \cup \bigcup_{y \in \mu} \max((f \circ \mu)^{-1}(\downarrow y)) \subseteq \rho$$

By the above equivalence ($*$), this implies the thesis. □

It is important to stress that the above proof of Lemma 3.1 basically consists in two key steps:

(A) The equality $f^A = f^B$ between b.c.a.'s is reduced to a standard property of completeness of the abstract domains A and B for the function f: in fact the proof shows that $f^A = f^B$ iff $\mu_A \circ (f \circ \mu_A) = \mu_A \circ (f \circ \mu_A) \circ \mu_B$. Making use of the terminology related to complete abstract interpretations as introduced by Giacobazzi et al. [19], this latter equation states the completeness of the pair of abstractions $\langle B, A \rangle$ for the concrete function $f \circ \mu_A$, that is, the approximation in A of the function $f \circ \mu_A$ does not suffer a loss of precision when its input value is first approximated in B.

(B) Then, the proof exploits the constructive characterization of the completeness of $\langle B, A \rangle$ for $f \circ \mu_A$ as given by [19, Lemma 4.2], which is the key technical result used in [19] for the fixpoint-based construction of complete shells of abstract domains.

In this sense, the proof of Lemma 3.1 provides an unexpected reduction of an equivalence problem between best correct approximations to a completeness problem. This is particularly interesting because while best correct approximations can always be defined on any abstract domain, complete approximations are instead uncommon since they represent an ideal situation where fixed point computations of complete approximations do not loose precision [9, 19].

More importantly, as a consequence of Lemma 3.1 and therefore, in turn, of the constructive characterization of complete abstract domains in [19], we derive the following constructive result of existence for correctness kernels.

Theorem 3.2. *Let $A \in \mathrm{Abs}(C)$ and $F \subseteq C \to C$ such that, for any $f \in F$, $f \circ \gamma_A \circ \alpha_A$ is continuous. Then, the correctness kernel of A for F exists and it is*

$$\mathcal{K}_F(A) = \mathrm{Cl}_{\wedge}\Big(\bigcup_{f\in F}\big(\,\mathrm{img}(f^A) \cup \textstyle\bigcup_{y\in \mathrm{img}(f^A)} \max(\{x \in A \mid f^A(x) = y\})\big)\Big).$$

Proof. In this proof, in order to ease notation, in function compositions we omit the explicit $\circ$ symbol. Let $\mu = \mu_A$. Let us first prove that the correctness kernel of A for F exists, namely $F^{\mathcal{K}_F(\mu)} = F^{\mu}$. Since $\mu \sqsubseteq \mathcal{K}_F(\mu)$, we apply Lemma 3.1 with $A = \mu$ and $B = \mathcal{K}_F(\mu)$, so that for any $f \in F$, we have that:

$$f^{\mathcal{K}_F(\mu)} = f^{\mu} \Leftrightarrow \mu(f(\mu(C))) \cup \textstyle\bigcup_{y\in\mu} \max(\{x \in C \mid f(\mu(x)) \leq_C y\}) \subseteq \mathcal{K}_F(\mu).$$

Since $\mathcal{K}_F(\mu) = \cap\{\rho \in \mathrm{uco}(C) \mid \forall g \in F.\, \rho g \rho = \mu g \mu\}$, we need to prove that for any $f \in F$,

$$\mu(f(\mu(C))) \cup \textstyle\bigcup_{y\in\mu} \max(\{x \in C \mid f(\mu(x)) \leq_C y\}) \subseteq \cap\{\rho \in \mathrm{uco}(C) \mid \forall g \in F.\, \rho g \rho = \mu g \mu\}.$$

We consider some $\rho \in \mathrm{uco}(C)$ such that for any $g \in F$, $\rho g \rho = \mu g \mu$, so that, in particular, $\rho f \rho = \mu f \mu$ holds. Hence, we prove that $\mu(f(\mu(C))) \cup \bigcup_{y\in\mu} \max(\{x \in \mu \mid f(\mu(x)) \leq_C y\}) \subseteq \rho(C)$. We first observe that:

$$\begin{aligned}
\mu f \mu = \rho f \rho \Rightarrow &\quad [\text{by applying } \rho \text{ to both sides}]\\
\rho\mu f \mu = \rho\rho f \rho \Rightarrow &\quad [\text{since } \rho\rho = \rho]\\
\rho\mu f \mu = \rho f \rho \Rightarrow &\quad [\text{since } \rho f \rho = \mu f \mu]\\
\rho\mu f \mu = \mu f \mu &
\end{aligned}$$

We thus obtain the first inclusion $\mu(f(\mu(C))) \subseteq \rho(C)$. On the other hand, we also have that:

$$\begin{aligned}
\mu f \mu = \rho f \rho \Rightarrow &\quad [\text{by applying } \rho \text{ in front to both sides}]\\
\mu f \mu\rho = \rho f \rho\rho \Rightarrow &\quad [\text{since } \rho\rho = \rho]\\
\mu f \mu\rho = \rho f \rho \Rightarrow &\quad [\text{since } \rho f \rho = \mu f \mu]\\
\mu f \mu\rho = \mu f \mu &
\end{aligned}$$

As already observed in the proof of Lemma 3.1, by the characterization of completeness in [19, Lemma 4.2], since, by hypothesis, $f\mu$ is continuous, it turns out that the equation $\mu(f\mu)\rho = \mu(f\mu)$ implies (actually, is equivalent to) $\bigcup_{y\in\mu} \max(\{x \in C \mid f(\mu(x)) \leq_C y\}) \subseteq \rho$. We have therefore obtained the second inclusion. Summing up, we have shown that the correctness kernel of A for F exists.
Next, we prove that

$$\mathcal{K}_F(\mu) = \mathcal{K}'_F(\mu) \triangleq \sqcup\{\rho \in \mathrm{uco}(C) \mid \rho \sqsupseteq \mu,\ F^{\rho} = F^{\mu}\}. \qquad (*)$$

In fact, since $\{\rho \in \mathrm{uco}(C) \mid \rho \sqsupseteq \mu,\ F^{\rho} = F^{\mu}\} \subseteq \{\rho \in \mathrm{uco}(C) \mid F^{\rho} = F^{\mu}\}$, we have that $\mathcal{K}'_F(\mu) \sqsubseteq \mathcal{K}_F(\mu)$. On the other hand, since $F^{\mathcal{K}_F(\mu)} = F^{\mu}$ and $\mathcal{K}_F(\mu) \sqsupseteq \mu$, we also have that $\mathcal{K}_F(\mu) \in \{\rho \in \mathrm{uco}(C) \mid \rho \sqsupseteq \mu,\ F^{\rho} = F^{\mu}\}$ and therefore $\mathcal{K}_F(\mu) \sqsubseteq \mathcal{K}'_F(\mu)$.

We now consider the following chain of equalities:

$$\mathcal{K}_F(\mu) =$$

[by equation $(*)$]

$$\bigsqcup\{\rho \in \mathrm{uco}(C) \mid \rho \sqsupseteq \mu,\ F^{\rho} = F^{\mu}\} =$$

[by a characterization of lub of uco's]

$$\bigcap\{\rho \in \mathrm{uco}(C) \mid \rho \subseteq \mu,\ F^{\rho} = F^{\mu}\} =$$

[by Lemma 3.1]

$$\bigcap\{\rho \in \mathrm{uco}(C) \mid \rho \subseteq \mu,\ \textstyle\bigcup_{f\in F}(\mu(f(\mu(C))) \cup \bigcup_{y\in\mu} \max((f\mu)^{-1}(\downarrow y))) \subseteq \rho\} =$$

[because $\bigcup_{f\in F} (\mu(f(\mu(C))) \cup \bigcup_{y\in\mu} \max((f\mu)^{-1}(\downarrow y))) \subseteq$
$\mathrm{Cl}_{\wedge}\big(\bigcup_{f\in F} (\mu(f(\mu(C))) \cup \bigcup_{y\in\mu} \max((f\mu)^{-1}(\downarrow y)))\big) \subseteq \mu$]

$$\mathrm{Cl}_{\wedge}\Big(\textstyle\bigcup_{f\in F} (\mu(f(\mu(C))) \cup \bigcup_{y\in\mu} \max((f\mu)^{-1}(\downarrow y)))\Big).$$

To conclude, let us show that

$$\mathrm{Cl}_{\wedge}\Big(\textstyle\bigcup_{f\in F} (\mu(f(\mu(C))) \cup \bigcup_{y\in\mu} \max((f\mu)^{-1}(\downarrow y)))\Big) = \gamma\Big(\mathrm{Cl}_{\wedge}\Big(\textstyle\bigcup_{f\in F} (\mathrm{img}(f^A) \cup \bigcup_{y\in\mathrm{img}(f^A)} \max(\{x \in A \mid f^A(x) = y\}))\Big)\Big).$$

Since γ preserves arbitrary glb's (see e.g. [8, Theorem 4.2.7.0.3]), it is enough to show that for any $f \in F$,

$$\mu(f(\mu(C))) \cup \textstyle\bigcup_{y\in\mu} \max((f\mu)^{-1}(\downarrow y)) = \gamma(\mathrm{img}(f^A) \cup \bigcup_{y\in\mathrm{img}(f^A)} \max(\{a \in A \mid f^A(a) = y\})).$$

Let us first observe that since $\alpha(C) = A$ (see e.g. [8, Theorem 4.2.7.0.3]), it turns out that $\gamma(\mathrm{img}(f^A)) = \gamma(\alpha(f(\gamma(A)))) = \gamma(\alpha(f(\gamma(\alpha(C))))) = \mu(f(\mu(C)))$. Next, we show that

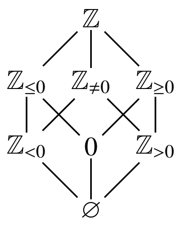
$$\textstyle\bigcup_{y\in\mu} \max((f\mu)^{-1}(\downarrow y)) = \gamma(\bigcup_{y\in\mathrm{img}(f^A)} \max(\{a \in A \mid f^A(a) = y\}))$$

($\subseteq$): Consider $y \in \mu$ and $z \in \max(\{x \in C \mid f(\mu(x)) \leq_C y\})$. Let us first observe that $z = \mu(z)$: in fact, since $f(\mu(\mu(z))) = f(\mu(z)) \leq_C y$ and $z \leq_C \mu(z)$, by maximality of z, $z = \mu(z)$. Then, define $y_z \triangleq f^A(\alpha(z)) \in \mathrm{img}(f^A)$ and let us show that $\alpha(z) \in \max(\{a \in A \mid f^A(a) = y_z\})$. First, $\alpha(z) \in \{a \in A \mid f^A(a) = y_z\}$ by definition. Then, consider some $b \in A$ such that $f^A(b) = y_z$ and $\alpha(z) \leq_A b$. We have that $z \leq_C \gamma(b)$ and $\alpha(f(\gamma(b))) = \alpha(f(\gamma(\alpha(z)))) \leq_A \alpha(y)$, so that $f(\mu(\gamma(b))) = f(\gamma(b)) \leq_C \gamma(\alpha(y)) = y$. Hence, by maximality of z, from $z \leq_C \gamma(b)$ we obtain $z = \gamma(b)$, and in turn $\alpha(z) = \alpha(\gamma(b)) = b$. Hence, from $\alpha(z) \in \max(\{a \in A \mid f^A(a) = y_z\})$, since $\gamma(\alpha(z)) = z$, we derive that $z \in \gamma(\max(\{a \in A \mid f^A(a) = y_z\}))$.
($\supseteq$): Consider $y = f^A(b)$ for some $b \in A$, and $z \in \gamma(\max(\{a \in A \mid f^A(a) = y\}))$. Therefore, $z = \gamma(a)$, for some $a \in \max(\{a \in A \mid f^A(a) = y\})$. Because $\gamma(y) \in \mu$, let us show that $z \in \max((f\mu)^{-1}(\downarrow\gamma(y)))$. Since $\alpha(f(\gamma(a))) = y$, we have that $f(\mu(z)) = f(\gamma(\alpha(z))) = f(\gamma(a)) \leq_C \gamma(y)$, and in turn we derive $z \in (f\mu)^{-1}(\downarrow\gamma(y))$. If $f(\mu(u)) \leq_C \gamma(y)$ and $z \leq_C u$, for some $u \in C$, then we have that $y = f^A(\alpha(z)) \leq_A f^A(\alpha(u)) = \alpha(f(\gamma(\alpha(u)))) = \alpha(f(\mu(u))) \leq_A \alpha(\gamma(y)) = y$, so that $f^A(\alpha(u)) = y$. Hence, from $z = \gamma(a) \leq_C u$, we obtain $a \leq_A \alpha(u)$, so that, by maximality of a, $a = \alpha(u)$, namely, $\alpha(z) = \alpha(u)$. Hence, $z = \gamma(\alpha(z)) = \gamma(\alpha(u))$. Hence, $u \leq_C \gamma(\alpha(u)) = z$, from which $z = u$. We can thus conclude that $z \in \max((f\mu)^{-1}(\downarrow\gamma(y)))$. □

The above result thus provides a constructive characterization of correctness kernels for all, possibly infinite, abstract domains A and for all, possibly not finitely computable, best correct approximations f^A under the weak hypothesis of continuity for $f \circ \gamma_A \circ \alpha_A$. In finite cases, i.e. when f^A has finite image and for any $y \in \text{img}(f^A)$, the set $\max(\{x \in A \mid f^A(x) = y\})$ is finitely computable — note that this happens when A is a finite abstraction and the best correct approximation f^A is finitely computable — this characterization obviously yields a terminating algorithm for automating the task of computing the correctness kernel of A for f. On the other hand, if either f^A has an infinite image or f^A is not finitely computable then the procedure induced by Theorem 3.2 in general cannot be automated. Of course, in this case, a correctness kernel can still be characterized through hand calculations.

Let us illustrate through a simple numerical example how to use the above result for deriving correctness kernels.

Example 3.3. Consider sets of integers $\langle \wp(\mathbb{Z}), \subseteq \rangle$ as concrete domain and a collecting square operation $sq : \wp(\mathbb{Z}) \to \wp(\mathbb{Z})$ as concrete function, i.e., $sq(X) \triangleq \{x^2 \mid x \in X\}$, which obviously preserves arbitrary lubs and therefore is continuous. Consider the abstract domain $\text{Sign} \in \text{Abs}(\wp(\mathbb{Z}))$, depicted in the following diagram, that represents the sign of an integer variable.



It is immediate to check that Sign induces the following best correct approximation of sq:

$$\begin{aligned} sq^{\text{Sign}} = \{&\varnothing \mapsto \varnothing, \mathbb{Z}_{<0} \mapsto \mathbb{Z}_{>0}, 0 \mapsto 0, \mathbb{Z}_{>0} \mapsto \mathbb{Z}_{>0}, \mathbb{Z}_{\leq 0} \mapsto \mathbb{Z}_{\geq 0},\\ &\mathbb{Z}_{\neq 0} \mapsto \mathbb{Z}_{>0}, \mathbb{Z}_{\geq 0} \mapsto \mathbb{Z}_{\geq 0}, \mathbb{Z} \mapsto \mathbb{Z}_{\geq 0}\}.\end{aligned}$$

Let us characterize the correctness kernel $\mathcal{K}_{sq}(\text{Sign})$ by Theorem 3.2. We have that $\text{img}(sq^{\text{Sign}}) = \{\varnothing, \mathbb{Z}_{>0}, 0, \mathbb{Z}_{\geq 0}\}$. Moreover,

$$\begin{gathered} \max(\{x \in \text{Sign} \mid sq^{\text{Sign}}(x) = \varnothing\}) = \{\varnothing\}\\ \max(\{x \in \text{Sign} \mid sq^{\text{Sign}}(x) = \mathbb{Z}_{>0}\}) = \{\mathbb{Z}_{\neq 0}\}\\ \max(\{x \in \text{Sign} \mid sq^{\text{Sign}}(x) = 0\}) = \{0\}\\ \max(\{x \in \text{Sign} \mid sq^{\text{Sign}}(x) = \mathbb{Z}_{\geq 0}\}) = \{\mathbb{Z}\} \end{gathered}$$

Therefore, $\bigcup_{y \in \text{img}(sq^{\text{Sign}})} \max(\{x \in \text{Sign} \mid sq^{\text{Sign}}(x) = y\}) = \{\varnothing, \mathbb{Z}_{\neq 0}, 0, \mathbb{Z}\}$ so that, by Theorem 3.2:

$$\mathcal{K}_{sq}(\text{Sign}) = \text{Cl}_{\cap}(\{\varnothing, \mathbb{Z}_{>0}, 0, \mathbb{Z}_{\geq 0}, \mathbb{Z}_{\neq 0}, \mathbb{Z}\}) = \text{Sign} \smallsetminus \{\mathbb{Z}_{<0}, \mathbb{Z}_{\leq 0}\}.$$

Thus, it turns out that we can safely remove the abstract values $\mathbb{Z}_{<0}$ and $\mathbb{Z}_{\leq 0}$ from Sign and still preserve the same b.c.a. as Sign. Besides, we cannot remove further abstract elements otherwise we do not retain the same b.c.a. as Sign. For example, this means that Sign-based analyses of programs like

$$x := k; \textbf{while } \text{condition } \textbf{do } x := x * x;$$

can be carried out by using the simpler domain Sign $\smallsetminus\{\mathbb{Z}_{<0}, \mathbb{Z}_{\leq 0}\}$, yet providing the same input/output abstract behavior. $\square$

It is worth remarking that in Theorem 3.2 the hypothesis of continuity is crucial for the existence of correctness kernels. In fact, observe that in Example 2.2 where the correctness kernel of the identical abstraction $\mu = \lambda x.x$ for f does not exist, we have that $f \circ \mu = f$ is clearly not continuous on the complete lattice C and therefore this example is consistent with the continuity hypothesis of Theorem 3.2.

4. Correctness Kernels in Abstract Model Checking

4.1. Background on Partitioning Abstractions

Following [22, 23], partitions of a finite state space Σ can be viewed as abstractions of the concrete domain $\langle \wp(\Sigma), \subseteq \rangle$. Let Part($\Sigma$) denote the set of partitions of Σ and let us recall that $\langle \text{Part}(\Sigma), \preceq, \curlyvee, \curlywedge \rangle$ is a complete lattice, where $P_1 \preceq P_2$ iff for all $s \in \Sigma$, $P_1(s) \subseteq P_2(s)$ [23]. Given a partition $P \in \text{Part}(\Sigma)$, we consider the corresponding set $\wp(P)$ of all (possibly empty) sets of blocks of P. Then, $\langle \wp(P), \subseteq \rangle$ can be viewed as an abstract domain of $\langle \wp(\Sigma), \subseteq \rangle$, which is called partitioning abstraction, by means of the following Galois insertion $(\alpha_P, \wp(\Sigma), \wp(P), \gamma_P)$:

$$\alpha_P(X) \triangleq \{B \in P \mid B \cap X \neq \varnothing\} \text{ and } \gamma_P(\mathcal{B}) \triangleq \cup_{B \in \mathcal{B}} B.$$

Hence, the abstraction $\alpha_P(X)$ provides the minimal over-approximation of a set X of states through blocks of P.

Also, an abstraction $A \in \text{Abs}(\wp(\Sigma))$ is called partitioning when there exists a partition $P \in \text{Part}(\Sigma)$ such that $(\alpha_A, \wp(\Sigma), A, \gamma_A)$ is equivalent to $(\alpha_P, \wp(\Sigma), \wp(P), \gamma_P)$. This happens exactly when $\gamma_A(A) \subseteq \wp(\Sigma)$ is closed under set intersections and complementations [23].

Finally, let us recall that any abstraction $A \in \text{Abs}(\wp(\Sigma))$ induces a partition $P_A \in \text{Part}(\Sigma)$ as follows [23]: for any $s, t \in \Sigma$, $P_A(s) = P_A(t) \Leftrightarrow \alpha_A(\{s\}) = \alpha_A(\{t\})$. This is particularly interesting because the corresponding partitioning abstraction $(\alpha_{P_A}, \wp(\Sigma), \wp(P_A), \gamma_{P_A})$ turns out to be the least partitioning abstraction refinement of A.

4.2. Abstract Transition Systems

Consider a finite state transition system $\mathcal{S} = \langle \Sigma, \to \rangle$ and a corresponding abstract transition system $\mathcal{A} = \langle P, \to^\sharp \rangle$ defined over a state partition $P \in \text{Part}(\Sigma)$. Equivalently, the abstract transition system $\mathcal{A}$ could be defined over a set A of abstract states which is defined by a surjective function $h : \Sigma \to A$ that induces a partition of Σ (see e.g. [7]). Fixpoint-based verification of a temporal specification on the abstract model $\mathcal{A}$ relies on computing some least/greatest fixpoints of operators which are defined using Boolean connectives (union, intersection, complementation) on abstract states and abstract successor/predecessor functions post$^\sharp$/pre$^\sharp$ on the abstract transition system $\langle P, \to^\sharp \rangle$. The key point here is that successor/predecessor functions are defined as best correct approximations on the partitioning abstract domain $\wp(P)$ of the corresponding concrete successor/predecessor functions on $\wp(\Sigma)$. In standard abstract model checking [1, 6, 7], the abstract transition relation is defined as the existential/existential relation $\to^{\exists\exists}$ between blocks of P: for any $B, C \in P$,

$$B \to^{\exists\exists} C \quad \text{iff} \quad \exists x \in B. \exists y \in C.\ x \to y$$

Accordingly, abstract predecessor and successor in $\langle P, \rightarrow^{\exists\exists}\rangle$ are given by the functions $\mathrm{pre}_P^{\exists\exists} : \wp(P) \to \wp(P)$ and $\mathrm{post}_P^{\exists\exists} : \wp(P) \to \wp(P)$ defined as follows:

$$\begin{aligned}
\mathrm{pre}_P^{\exists\exists}(\mathcal{C}) &\triangleq \{B \in P \mid \exists C \in \mathcal{C}.\, B \rightarrow^{\exists\exists} C\};\\
\mathrm{post}_P^{\exists\exists}(\mathcal{B}) &\triangleq \{C \in P \mid \exists B \in \mathcal{B}.\, B \rightarrow^{\exists\exists} C\}.
\end{aligned}$$

As shown in [22, 23], it turns out that $\mathrm{pre}_P^{\exists\exists}$ and $\mathrm{post}_P^{\exists\exists}$ are the best correct approximations of, respectively, $\mathrm{pre} : \wp(\Sigma) \to \wp(\Sigma)$ and $\mathrm{post} : \wp(\Sigma) \to \wp(\Sigma)$ on the abstraction $(\alpha_P, \wp(\Sigma), \wp(P), \gamma_P)$. In fact, for a set of blocks $\mathcal{C} \in \wp(P)$, we have that

$$\begin{aligned}
\alpha_P(\mathrm{pre}(\gamma_P(\mathcal{C}))) &= \{B \in P \mid B \cap \mathrm{pre}(\cup_{C \in \mathcal{C}} C) \neq \varnothing\}\\
&= \{B \in P \mid \cup_{C \in \mathcal{C}} B \cap \mathrm{pre}(C) \neq \varnothing\}\\
&= \{B \in P \mid \exists C \in \mathcal{C}.\, B \rightarrow^{\exists\exists} C\}\\
&= \mathrm{pre}_P^{\exists\exists}(\mathcal{C})
\end{aligned}$$

and analogous equations hold for post. We thus have that

$$\mathrm{pre}_P^{\exists\exists} = \alpha_P \circ \mathrm{pre} \circ \gamma_P \ \text{ and } \ \mathrm{post}_P^{\exists\exists} = \alpha_P \circ \mathrm{post} \circ \gamma_P.$$

4.3. Correctness Kernels

The above abstract interpretation-based approach allows us to apply correctness kernels in abstract model checking as follows. The abstract transition system $\mathcal{A} = \langle P, \rightarrow^{\exists\exists}\rangle$ is viewed as an abstract interpretation which is defined by the abstract domain $(\alpha_P, \wp(\Sigma), \wp(P), \gamma_P)$ and the abstract functions $\mathrm{pre}_P^{\exists\exists} = \alpha_P \circ \mathrm{pre} \circ \gamma_P$ and $\mathrm{post}_P^{\exists\exists} = \alpha_P \circ \mathrm{post} \circ \gamma_P$. We are thus interested in the correctness kernel of the partitioning abstraction $\wp(P)$ for the concrete predecessor/successor functions $\{\mathrm{pre}, \mathrm{post}\}$, that we denote simply by $\mathcal{K}_{\rightarrow}(P)$. Observe that, by Theorem 3.2, the kernel $\mathcal{K}_{\rightarrow}(P) \in \mathrm{Abs}(\wp(\Sigma))$ clearly exists since pre, post and $\gamma_P \circ \alpha_P$ preserve arbitrary unions on $\wp(\Sigma)$. The abstraction $\mathcal{K}_{\rightarrow}(P)$ provides a simplification of the abstract domain $\wp(P)$ that preserves the best correct approximations of both predecessor and successor functions. In general, it turns out that $\mathcal{K}_{\rightarrow}(P)$ is not a partitioning abstraction, as shown by the following example.

Example 4.1. Consider the following 3-state transition system.

$$\begin{tikzcd}
1 \arrow[r] \arrow[rr, bend left] & 2 \arrow[r] & 3
\end{tikzcd}$$

Let us consider the finest partition $P = \{[1], [2], [3]\}$, so that $\mathrm{pre}_P^{\exists\exists} = \mathrm{pre}$ and $\mathrm{post}_P^{\exists\exists} = \mathrm{post}$. In order to apply Theorem 3.2, here we have that $\mathrm{img}(\mathrm{pre}) = \{\varnothing, [1], [1, 2]\}$, $\mathrm{img}(\mathrm{post}) = \{\varnothing, [3], [2, 3]\}$ and

$$\begin{aligned}
\max(\{S \in \wp(\Sigma) \mid \mathrm{pre}(S) = \varnothing\}) &= \{1\}\\
\max(\{S \in \wp(\Sigma) \mid \mathrm{pre}(S) = \{1\}\}) &= \{1, 2\}\\
\max(\{S \in \wp(\Sigma) \mid \mathrm{pre}(S) = \{1, 2\}\}) &= \{1, 2, 3\}\\
\max(\{S \in \wp(\Sigma) \mid \mathrm{post}(S) = \varnothing\}) &= \{3\}\\
\max(\{S \in \wp(\Sigma) \mid \mathrm{post}(S) = \{3\}\}) &= \{2, 3\}\\
\max(\{S \in \wp(\Sigma) \mid \mathrm{post}(S) = \{2, 3\}\}) &= \{1, 2, 3\}
\end{aligned}$$

Thus, by Theorem 3.2,

$$\mathcal{K}_{\to}(P) = \mathrm{Cl}_{\cap}(\{\varnothing, [1], [3], [1,2], [2,3], [1,2,3]\}) = \{\varnothing, [1], [2], [3], [1,2], [2,3], [1,2,3]\}.$$

Notice that $[2] \in \mathcal{K}_{\to}(P)$ while its complement $[1,3] \notin \mathcal{K}_{\to}(P)$. Thus, $\mathcal{K}_{\to}(P)$ is not a partitioning abstraction. ☐

Since the abstract domain $\mathcal{K}_{\to}(P)$ in general is not partitioning, we are thus interested in its partitioning abstraction, which is characterized as follows.

Corollary 4.2. *Let $B_1, B_2 \in P$. Then, $\mathcal{K}_{\to}(P)(B_1) = \mathcal{K}_{\to}(P)(B_2)$ if and only if* $\mathrm{pre}_P^{\exists\exists}(\{B_1\}) = \mathrm{pre}_P^{\exists\exists}(\{B_2\})$ *and* $\mathrm{post}_P^{\exists\exists}(\{B_1\}) = \mathrm{post}_P^{\exists\exists}(\{B_2\})$.

Proof. The kernel $\mathcal{K}_{\to}(P)$ can be obtained by applying Theorem 3.2 to the GI $(\alpha_P, \wp(\Sigma), \wp(P), \gamma_P)$ and to the functions $\{\mathrm{pre}, \mathrm{post}\}$. Since the best correct approximations $\mathrm{pre}_P^{\exists\exists}$ and $\mathrm{post}_P^{\exists\exists}$ preserve arbitrary lub's (i.e., unions) on $\langle \wp(P), \subseteq\rangle$, max's can be replaced by lub's in $\langle \wp(P), \subseteq\rangle$, namely set unions. Hence, we have that:

$$\begin{aligned}\mathcal{K}_{\to}(P) = \mathrm{Cl}_{\cap}\Big(&\mathrm{img}(\mathrm{pre}_P^{\exists\exists}) \bigcup \{\cup\{\mathcal{C} \in \wp(P) \mid \mathrm{pre}^{\exists\exists}(\mathcal{C}) = \mathcal{B}\} \,\big|\, \mathcal{B} \in \mathrm{img}(\mathrm{pre}_P^{\exists\exists})\} \\ &\bigcup \mathrm{img}(\mathrm{post}_P^{\exists\exists}) \bigcup \{\cup\{\mathcal{B} \in \wp(P) \mid \mathrm{post}^{\exists\exists}(\mathcal{B}) = \mathcal{C}\} \,\big|\, \mathcal{C} \in \mathrm{img}(\mathrm{post}_P^{\exists\exists})\}\Big).\end{aligned}$$

Let us then show the stated equivalence.

($\Rightarrow$) More in general, it is enough to observe that if ρ is the correctness kernel of some μ for some f then for any $c_1, c_2 \in C$ such that $\rho(c_1) = \rho(c_2)$ we have that $\mu(f(\mu(c_1))) = \rho(f(\rho(c_1))) = \rho(f(\rho(c_2))) = \mu(f(\mu(c_2)))$.

($\Leftarrow$) In the following, let $\mu \in \mathrm{uco}(\wp(\Sigma))$ denote the uco induced by the abstraction $\mathcal{K}_{\to}(P)$. Let us consider two blocks $B_1, B_2 \in P$. If $\mathrm{pre}_P^{\exists\exists}(\mathcal{C}) \in \mathrm{img}(\mathrm{pre}_P^{\exists\exists})$, for some $\mathcal{C} \in \wp(P)$, then we have that:

$$\begin{aligned} B_1 \in \mathrm{pre}_P^{\exists\exists}(\mathcal{C}) \Leftrightarrow &\quad [\text{by definition of } \mathrm{pre}_P^{\exists\exists}] \\ \exists C \in \mathcal{C}.\, B_1 \to^{\exists\exists} C \Leftrightarrow &\quad [\text{by definition of } \mathrm{post}_P^{\exists\exists}] \\ \exists C \in \mathcal{C}.\, C \in \mathrm{post}_P^{\exists\exists}(\{B_1\}) \Leftrightarrow &\quad [\text{by hypothesis}] \\ \exists C \in \mathcal{C}.\, C \in \mathrm{post}_P^{\exists\exists}(\{B_2\}) \Leftrightarrow &\quad [\text{by replicating the previous arguments}] \\ B_2 \in \mathrm{pre}_P^{\exists\exists}(\mathcal{C}) & \end{aligned}$$

Also, for any $\mathcal{B} \in \mathrm{img}(\mathrm{pre}_P^{\exists\exists})$, we have that:

$$\begin{aligned} B_1 \in \cup\{\mathcal{C} \in \wp(P) \mid \mathrm{pre}_P^{\exists\exists}(\mathcal{C}) = \mathcal{B}\} \Leftrightarrow &\quad [\text{by definition of } \mathrm{pre}_P^{\exists\exists}] \\ B_1 \in \{C \in P \mid \mathrm{pre}_P^{\exists\exists}(\{C\}) \subseteq \mathcal{B}\} \Leftrightarrow &\quad [\text{by definition of } \in] \\ \mathrm{pre}_P^{\exists\exists}(\{B_1\}) \subseteq \mathcal{B} \Leftrightarrow &\quad [\text{by hypothesis}] \\ \mathrm{pre}_P^{\exists\exists}(\{B_2\}) \subseteq \mathcal{B} \Leftrightarrow &\quad [\text{by replicating the previous arguments}] \\ B_2 \in \cup\{\mathcal{C} \in \wp(P) \mid \mathrm{pre}_P^{\exists\exists}(\mathcal{C}) = \mathcal{B}\} & \end{aligned}$$

Likewise, for any $\mathcal{B} \in \wp(P)$ and $\mathcal{C} \in \mathrm{img}(\mathrm{post}_P^{\exists\exists})$ we also have that:

$$\begin{aligned} &B_1 \in \mathrm{post}_P^{\exists\exists}(\mathcal{B}) \Leftrightarrow B_2 \in \mathrm{post}_P^{\exists\exists}(\mathcal{B}) \\ &B_1 \in \cup\{\mathcal{B} \in \wp(P) \mid \mathrm{post}_P^{\exists\exists}(\mathcal{B}) = \mathcal{C}\} \Leftrightarrow B_2 \in \cup\{\mathcal{B} \in \wp(P) \mid \mathrm{post}_P^{\exists\exists}(\mathcal{B}) = \mathcal{C}\}. \end{aligned}$$

Consequently, $\mathcal{K}_{\to}(P)(B_1) = \mathcal{K}_{\to}(P)(B_2)$. ☐

We denote by $P_{\mathcal{K}} \in \text{Part}(\Sigma)$ the partitioning abstraction of $\mathcal{K}_{\rightarrow}(P)$. We therefore have that in $P_{\mathcal{K}}$ a block $B \in P$ is merged together with all the blocks $B' \in P$ such that $\text{pre}^{\exists\exists}(\{B\}) = \text{pre}^{\exists\exists}(\{B'\})$ and $\text{post}^{\exists\exists}(\{B\}) = \text{post}^{\exists\exists}(\{B'\})$.

Given $P, Q \in \text{Part}(\Sigma)$, let $\text{pre}_Q^{\exists\exists} = \text{pre}_P^{\exists\exists}$ denote the fact that for all $s \in \Sigma$, $\cup\, \text{pre}_Q^{\exists\exists}(Q(s)) = \cup\, \text{pre}_P^{\exists\exists}(P(s))$, and analogously for post. We thus derive the following characterization of $P_{\mathcal{K}}$.

Corollary 4.3. $P_{\mathcal{K}} = \curlyvee\{Q \in \text{Part}(\Sigma) \mid \text{pre}_Q^{\exists\exists} = \text{pre}_P^{\exists\exists},\ \text{post}_Q^{\exists\exists} = \text{post}_P^{\exists\exists}\}$.

Proof. Let $\mu \triangleq \mathcal{K}_{\rightarrow}(P) \in \text{Abs}(\wp(\Sigma))$. Let us first check that $\text{pre}_{P_{\mathcal{K}}}^{\exists\exists} = \text{pre}_P^{\exists\exists}$. Given $s \in \Sigma$, we have that:

$$\begin{aligned}
\cup\, \text{pre}_P^{\exists\exists}(P(s)) \subseteq &\quad [\text{since } P \leq P_{\mathcal{K}}]\\
\cup\, \text{pre}_{P_{\mathcal{K}}}^{\exists\exists}(P_{\mathcal{K}}(s)) \subseteq &\quad [\text{since, for any } S,\ P_{\mathcal{K}}(S) \subseteq \mu(S)]\\
\mu(\text{pre}(\mu(s))) = &\quad [\text{since } \mu \text{ is the correctness kernel of } P \text{ for pre and post}]\\
\cup\, \text{pre}_P^{\exists\exists}(P(s)) &
\end{aligned}$$

Hence, $\text{pre}_{P_{\mathcal{K}}}^{\exists\exists} = \text{pre}_P^{\exists\exists}$. Likewise, $\text{post}_{P_{\mathcal{K}}}^{\exists\exists} = \text{post}_P^{\exists\exists}$. Therefore, we obtain that $P_{\mathcal{K}} \leq \curlyvee\{Q \in \text{Part}(\Sigma) \mid \text{pre}_Q^{\exists\exists} = \text{pre}_P^{\exists\exists}, \text{post}_Q^{\exists\exists} = \text{post}_P^{\exists\exists}\}$. On the other hand, if $Q \in \text{Part}(\Sigma)$ is such that $\text{pre}_Q^{\exists\exists} = \text{pre}_P^{\exists\exists}$ and $\text{post}_Q^{\exists\exists} = \text{post}_P^{\exists\exists}$ then, since μ is the correctness kernel of P for pre and post, $\gamma_Q \circ \alpha_Q \sqsubseteq \mu$. Hence, since the partitioning abstraction refinement is monotonic, we obtain that $Q \leq P_{\mathcal{K}}$. Consequently, $\{Q \in \text{Part}(\Sigma) \mid \text{pre}_Q^{\exists\exists} = \text{pre}_P^{\exists\exists}, \text{post}_Q^{\exists\exists} = \text{post}_P^{\exists\exists}\} \leq P_{\mathcal{K}}$. ☐

Example 4.4. Reconsider the abstract transition system $\mathcal{A}$ in Figure 1 where the underlying state partition is $P = \{[1], [2,3], [4,5], [6], [7], [8,9]\}$. Here, by Corollary 4.2, the block $[2,3]$ is merged with $[4,5]$ while $[6]$ is merged with $[7]$. This therefore simplifies the partition P to $P_{\mathcal{K}} = \{[1], [2,3,4,5], [6,7], [8,9]\}$, that is, we obtain the abstract transition system $\mathcal{A}''$ depicted in Figure 1. ☐

5. Example Guided Abstraction Simplification

Let us discuss how correctness kernels give rise to an Example-Guided Abstraction Simplification (EGAS) paradigm in abstract transition systems.

5.1. CEGAR Background

Let us first recall some basic notions of CEGAR [4, 5]. Consider an abstract transition system $\mathcal{A} = \langle P, \rightarrow^{\exists\exists}\rangle$ defined over a state partition $P \in \text{Part}(\Sigma)$ and some finite abstract path $\pi = \langle B_1, ..., B_n\rangle$ in $\mathcal{A}$, where each B_i is a block of P. Typically, this path is a counterexample to the validity in $\mathcal{A}$ of a temporal formula and it originated as output of a model checker running on $\mathcal{A}$ (for simplicity we do not consider here loop path counterexamples). The set of concrete paths that are abstracted to π are defined as follows:

$$\text{paths}(\pi) \triangleq \{\langle s_1, ..., s_n\rangle \in \Sigma^n \mid \forall i \in [1,n].s_i \in B_i \ \&\ \forall i \in [1,n).s_i \rightarrow s_{i+1}\}.$$

The abstract path π is *spurious* when it represents no real concrete path, that is, when $\text{paths}(\pi) = \varnothing$. A corresponding sequence $\text{sp}(\pi) = \langle S_1, ..., S_n\rangle$ of sets of states in Σ is inductively defined as follows: $S_1 \triangleq B_1$; $S_{i+1} \triangleq \text{post}(S_i) \cap B_{i+1}$. As observed in [5], it turns out that π is spurious iff

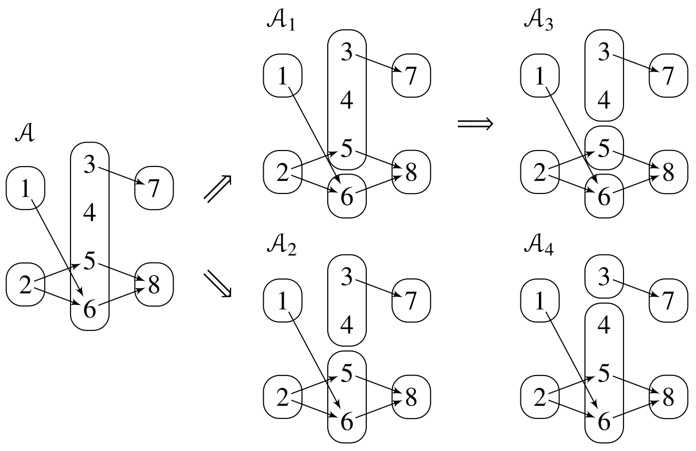


Figure 3: Some abstract transition systems.

there exists a least $k \in [1, n-1]$ such that $S_{k+1} = \varnothing$. In such a case, the partition P is refined by splitting the block B_k. The three following sets partition the states of the block B_k:

dead-end states: $B_k^{\text{dead}} \triangleq S_k \neq \varnothing$

bad states: $B_k^{\text{bad}} \triangleq B_k \cap \text{pre}(B_{k+1}) \neq \varnothing$

irrelevant states: $B_k^{\text{irr}} \triangleq B_k \smallsetminus (B_k^{\text{dead}} \cup B_k^{\text{bad}})$

The split of the block B_k must separate dead-end states from bad states, while irrelevant states may be joined indifferently with dead-end or bad states. However, when states are memory stores, the problem of finding the coarsest refinement of P that separates dead-end and bad states is NP-hard [5, Theorem 4.17] and thus some refinement heuristics are necessarily used. According to the basic heuristic of CEGAR [5, Section 4], B_k is simply split into B_k^{dead} and $B_k^{\text{bad}} \cup B_k^{\text{irr}}$.

Let us see a simple example. Consider the abstract path $\pi = \langle [1], [3,4,5,6], [7] \rangle$ in the abstract transition system $\mathcal{A}$ depicted in Figure 3. This is a spurious path and the block $[3,4,5,6]$ needs to be split. This block is therefore partitioned as follows: $[6]$ dead-end states, $[3]$ bad states and $[4,5]$ irrelevant states. The refinement heuristic of CEGAR tells us that irrelevant states are joined with bad states so that $\mathcal{A}$ is refined to the abstract transition system $\mathcal{A}_1$. In turn, consider the spurious path $\pi' = \langle [2], [3,4,5], [7] \rangle$ in $\mathcal{A}_1$, so that CEGAR refines $\mathcal{A}_1$ to $\mathcal{A}_3$ by splitting the block $[3,4,5]$ into $[3,4]$ and $[5]$, i.e., bad and irrelevant states in $[3,4]$ and dead-end states in $[5]$. In the first abstraction refinement, let us observe that if irrelevant states in $[4,5]$ would have been joined together with dead-end states in $[6]$ rather than with bad states in $[3]$ we would have obtained the abstract system $\mathcal{A}_4$, and $\mathcal{A}_4$ does not contain spurious paths so that it does not need to be further refined. Let us also notice that if the irrelevant state 5 would have been joined with dead-end states $[6]$ while the irrelevant state 4 would have been joined with bad states $[3]$ we would have obtained the abstract system $\mathcal{A}_2$ that still does not need to be further refined since it does not contain spurious paths.

5.2. EGAS

EGAS can be integrated within the CEGAR loop thanks to the following remark. If π_1 and π_2 are paths, respectively, in $\langle P_1, \rightarrow^{\exists\exists}\rangle$ and $\langle P_2, \rightarrow^{\exists\exists}\rangle$, where $P_1, P_2 \in \mathrm{Part}(\Sigma)$ and $P_1 \preceq P_2$, then we say that π_1 is abstracted to π_2, denoted by $\pi_1 \sqsubseteq \pi_2$, when $\mathrm{length}(\pi_1) = \mathrm{length}(\pi_2)$ and for any $j \in [1, \mathrm{length}(\pi_1)]$, $\pi_1(j) \subseteq \pi_2(j)$.

Corollary 5.1. *Consider an abstract transition system $\mathcal{A} = \langle P, \rightarrow^{\exists\exists}\rangle$ over a partition $P \in \mathrm{Part}(\Sigma)$ and its simplification $\mathcal{A}_s = \langle P_{\mathcal{K}}, \rightarrow^{\exists\exists}\rangle$ induced by the correctness kernel $\mathcal{K}_{\rightarrow}(P)$. If π is a spurious abstract path in $\mathcal{A}_s$ then there exists a spurious abstract path π' in $\mathcal{A}$ such that $\pi' \sqsubseteq \pi$.*

Proof. Let $\pi = \langle B_1, ..., B_n\rangle$, where, for any $i \in [1, n]$, $B_i \in P_{\mathcal{K}}$, and let B_k be the block of π that generates the spuriousness of π. Since $P \preceq P_{\mathcal{K}}$, we have that for each $i \in [1, n]$, $B_i = \cup_{j_i \in J_i} C_i^{j_i}$, for some set of blocks $C_i^{j_i} \in P$. By Corollary 4.3, for each $i \in [1, n)$ and $j_i \in J_i$, $\cup\,\mathrm{post}^{\exists\exists}_{P_{\mathcal{K}}}(B_i) = \cup\,\mathrm{post}^{\exists\exists}_{P}(C_i^{j_i})$ and for each $i \in (1, n]$ and $j_i \in J_i$, $\cup\,\mathrm{pre}^{\exists\exists}_{P_{\mathcal{K}}}(B_i) = \cup\,\mathrm{pre}^{\exists\exists}_{P}(C_i^{j_i})$. Then, in order to define the path π' in $\mathcal{A}$, for any $i \in [1, n]$, one can choose any block $C_i^{j_i}$ in P such that $C_i^{j_i} \subseteq B_i$. The key point to note is that by Corollary 4.3, it turns out that $C_k^{j_k}$ causes the spuriousness of the path π'. Moreover, $\pi' \sqsubseteq \pi$, and this concludes the proof. □

This means that the abstraction simplification induced by the correctness kernel does not add spurious paths.

5.3. Bad- and Dead-irrelevant States

The above observations suggest us a new refinement strategy within the CEGAR loop. Let $\pi = \langle B_1, ..., B_n\rangle$ be a spurious path in $\mathcal{A}$ and let $\mathrm{sp}(\pi) = \langle S_1, ..., S_n\rangle$ such that $S_{k+1} = \varnothing$ for some minimum $k \in [1, n-1]$, so that the block B_k needs to be split. The set of irrelevant states in B_k^{irr} is thus partitioned as specified by the following strategy. An irrelevant state $s \in B_k^{\mathrm{irr}}$ is called *bad-irrelevant* when

$$\mathrm{pre}^{\exists\exists}_P(B_k^{\mathrm{bad}} \cup \{s\}) = \mathrm{pre}^{\exists\exists}_P(B_k^{\mathrm{bad}}) \;\text{ and }\; \mathrm{post}^{\exists\exists}_P(B_k^{\mathrm{bad}} \cup \{s\}) = \mathrm{post}^{\exists\exists}_P(B_k^{\mathrm{bad}})$$

Thus, a bad-irrelevant state can be joined to bad states without affecting the set of abstract paths in P that go through B_k^{bad}. *Dead-irrelevant* states are analogously defined w.r.t. the set B_k^{dead} of dead-end states. It may happen that an irrelevant state s is both bad- and dead-irrelevant: in this case, s could be equivalently merged with bad or dead states since in both cases no spurious path would be added. Clearly, it may also happen that an irrelevant state is neither bad- nor dead-irrelevant. These states are called *fully-irrelevant*.

Let us denote by $S_k^{\text{bad-irr}}$ and $S_k^{\text{dead-irr}}$, respectively, the set of all bad- and dead-irrelevant states in B_k^{irr}. We can therefore partition the set of irrelevant states in B_k^{irr} as follows:

bad-irrelevant block: $B_k^{\text{bad-irr}} \triangleq S_k^{\text{bad-irr}} \smallsetminus S_k^{\text{dead-irr}}$

dead-irrelevant block: $B_k^{\text{dead-irr}} \triangleq S_k^{\text{dead-irr}} \smallsetminus S_k^{\text{bad-irr}}$

fully-irrelevant block: $B_k^{\text{fully-irr}} \triangleq (S_k^{\text{bad-irr}} \cap S_k^{\text{dead-irr}}) \cup (B_k^{\mathrm{irr}} \smallsetminus (S_k^{\text{bad-irr}} \cup S_k^{\text{dead-irr}}))$

Hence, the set of irrelevant states in B_k^{irr} is partitioned into three disjoint blocks: $B_k^{\text{bad-irr}}$, $B_k^{\text{dead-irr}}$ and $B_k^{\text{fully-irr}}$. Notice that it may happen that one or two of these sets is empty, whereas at least one of them must be non-empty.

We denote by P^π the refined partition obtained from P by replacing the block B_k with at most five (and at least three) non-empty blocks: B_k^{bad}, $B_k^{\text{bad-irr}} \neq \varnothing$, B_k^{dead}, $B_k^{\text{dead-irr}} \neq \varnothing$ and $B_k^{\text{fully-irr}} \neq \varnothing$. By Corollary 4.3, it is clear that in the partition $P^\pi_{\mathcal{K}}$ obtained from the correctness kernel of P^π, B_k^{bad} is merged with $B_k^{\text{bad-irr}}$, B_k^{dead} is merged with $B_k^{\text{dead-irr}}$, while $B_k^{\text{fully-irr}}$ remains a separate block in $\mathcal{K}_\rightarrow(P_\pi)$. Also, by Corollary 5.1, it turns out that no spurious path is added in the abstract system $\langle P^\pi_{\mathcal{K}}, \rightarrow^{\exists\exists}\rangle$ w.r.t. the system $\langle P^\pi, \rightarrow^{\exists\exists}\rangle$.

Summing up, the refinement strategy EGAS goes as follows:

(A) If $B_k^{\text{bad-irr}} \neq \varnothing$ then merge $B_k^{\text{bad-irr}}$ with bad states.

(B) If $B_k^{\text{dead-irr}} \neq \varnothing$ then merge $B_k^{\text{dead-irr}}$ with dead-end states.

(C) If $B_k^{\text{fully-irr}} \neq \varnothing$ then these fully-irrelevant states can be indifferently merged with bad or dead states; for these states, one could use, for example, the basic refinement heuristic of CEGAR that merge them with bad states.

In the above example, for the spurious path $\langle [1], [3,4,5,6], [7]\rangle$ in $\mathcal{A}$, the block $B = [3,4,5,6]$ needs to be refined. We have that:

$$B^{\text{bad}} = [3],\ \ B^{\text{dead}} = [6],\ \ B^{\text{irr}} = [4,5].$$

Here, 5 is a dead-irrelevant state because $\text{pre}^{\exists\exists}([5,6]) = \{[1],[2]\} = \text{pre}^{\exists\exists}([6])$ and $\text{post}^{\exists\exists}([5,6]) = \{[8]\} = \text{post}^{\exists\exists}([6])$; also, 5 is not bad-irrelevant because $\text{pre}^{\exists\exists}([3,5]) \neq \text{pre}^{\exists\exists}([3])$. Moreover, 4 is both dead- and bad-irrelevant and therefore it is fully-irrelevant. Hence, according to the EGAS refinement strategy, the block $[3,4,5,6]$ is split into $[3,4]$ and $[5,6]$, so that EGAS gives rise to the abstract system $\mathcal{A}_2$ that does not need further refinements.

6. Correctness Kernels in Predicate Abstraction

Let us discuss how correctness kernels can be also used in the context of predicate abstraction-based model checking [12, 20]. Following Ball et al.'s approach [2], predicate abstraction can be formalized by abstract interpretation as follows. Let us consider a program P with k integer variables $x_1, \ldots, x_k$. The concrete domain of computation of P is $\langle \wp(\text{States}), \subseteq\rangle$ where $\text{States} \triangleq \{x_1, \ldots, x_k\} \rightarrow \mathbb{Z}$. Values in States are denoted by tuples $\langle z_1, \ldots, z_k\rangle \in \mathbb{Z}^k$. The program P generates a transition system $\langle \text{States}, \rightarrow\rangle$ so that the concrete semantics of P is defined by the corresponding successor function $\text{post} : \wp(\text{States}) \rightarrow \wp(\text{States})$.

A finite set $\mathcal{P} = \{p_1, ..., p_n\}$ of state predicates is considered, where each predicate p_i denotes the subset of states that satisfy p_i, i.e. $\{s \in \text{States} \mid s \models p_i\}$. These predicates give rise to the so-called *Boolean abstraction* $B \triangleq \langle \wp(\{0,1\}^n), \subseteq\rangle$ which is related to $\wp(\text{States})$ through the following abstraction and concretization maps (here, $s \models p_i$ is understood to assume values in $\{0,1\}$):

$$\alpha_B(S) \triangleq \{\langle s \models p_1, ..., s \models p_n\rangle \in \{0,1\}^n \mid s \in S\},$$
$$\gamma_B(V) \triangleq \{s \in \text{States} \mid \langle s \models p_1, ..., s \models p_n\rangle \in V\}.$$

These functions give rise to a disjunctive (i.e., γ_B preserves arbitrary lub's in $\langle \wp(\{0,1\}^n), \subseteq\rangle$) Galois connection $(\alpha_B, \wp(\text{States})_\subseteq, \wp(\{0,1\}^n)_\subseteq, \gamma_B)$.

Verification of reachability properties based on predicate abstraction consists in computing the least fixpoint of the best correct approximation of post on the Boolean abstraction B, i.e., $\text{post}^B \triangleq \alpha_B \circ \text{post} \circ \gamma_B$. As argued in [2], the Boolean abstraction B may be too costly for the purpose of reachability verification, so that one usually abstracts B through the so-called *Cartesian abstraction*. The Cartesian abstraction is defined as

$$C \triangleq \langle \{0, 1, *\}^n \cup \{\bot_C\}, \leq \rangle$$

where $\leq$ is the component-wise ordering between tuples of values in $\{0, 1, *\}$ ordered by $0 < *$ and $1 < *$, while $\bot_C$ is a bottom element that represents the empty set of states. The concretization function $\gamma_C : C \to \wp(\text{States})$ is as follows:

$$\gamma_C(\langle v_1, ..., v_n \rangle) \triangleq \{s \in \text{States} \mid \langle s \models p_1, ..., s \models p_n \rangle \leq \langle v_1, ..., v_n \rangle\}.$$

This latter abstraction formalizes precisely the abstract post operator computed by the verification algorithm of the c2bp tool in SLAM [3]. However, the Cartesian abstraction of B may cause a loss of precision, so that this abstraction is successively refined by reduced disjunctive completion and the so-called focus operation, and this formalizes the bebop tool in SLAM [2].

Let us consider the following example program P, taken from [2], where the goal is that of verifying that the **assert**(0) statement at line $(*)$ is never reached, regardless of the context in which the function *foo*() is called.

```
int x, y, z, w;
void foo() {
  do {
    z := 0;  x := y;
    if (w) { x++; z := 1; }
  } while (!(x = y))
  if (z)
    assert(0);  // (*)
}
```

Ball et al. [2] consider the following set of predicates $\mathcal{P} \triangleq \{p_1 \equiv (z = 0), p_2 \equiv (x = y)\}$ so that the Boolean abstraction is $B = \wp(\{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle, \langle 1,1\rangle\})_{\subseteq}$. Clearly, the analysis based on B allows us to conclude that line $(*)$ is not reachable. This comes as a consequence of the fact that the least fixpoint computation of the best correct approximation post^B for the do-while loop provides as result $\{\langle 0,0\rangle, \langle 1,1\rangle\} \in B$ because:

$$\varnothing \xrightarrow{z:=0;\ x:=y} \{\langle 1,1\rangle\} \xrightarrow{\textbf{if}(w)\{x++;\ z:=1;\}} \{\langle 1,1\rangle\} \cup \{\langle 0,0\rangle\}$$

where, according to a standard approach, the Boolean guard b of an if statement $\textbf{if}(b) S$ is ignored so that the corresponding successor function is defined as $\text{post}_{\textbf{if}(b)\,S}(X) = X \cup \text{post}_S(X)$. Hence, at the exit of the do-while loop, where p_2 holds, one can conclude that

$$\{\langle 1,1\rangle, \langle 0,0\rangle\} \cap p_2 = \{\langle 1,1\rangle, \langle 0,0\rangle\} \cap \{\langle 0,1\rangle, \langle 1,1\rangle\} = \{\langle 1,1\rangle\}$$

holds, hence p_1 is satisfied, so that $z = 0$ and therefore line $(*)$ can never be reached.

Let us characterize the correctness kernel of the Boolean abstraction B in this example. Let us define $S_1 \triangleq \{z := 0;\; x := y\}$ and $S_2 \triangleq \{x{+}{+};\; z := 1\}$. The best correct approximations of post_{S_1} and post_{S_2} on the abstract domain B turn out to be as follows:

$$\begin{aligned}
\alpha_B \circ \text{post}_{S_1} \circ \gamma_B &= \Big\{\{\langle 0,0\rangle\} \mapsto \{\langle 1,1\rangle\}, \{\langle 0,1\rangle\} \mapsto \{\langle 1,1\rangle\},\\
&\qquad \{\langle 1,0\rangle\} \mapsto \{\langle 1,1\rangle\}, \{\langle 1,1\rangle\} \mapsto \{\langle 1,1\rangle\}\Big\}\\
\alpha_B \circ \text{post}_{S_2} \circ \gamma_B &= \Big\{\{\langle 0,0\rangle\} \mapsto \{\langle 0,0\rangle, \langle 0,1\rangle\}, \{\langle 0,1\rangle\} \mapsto \{\langle 0,0\rangle, \langle 0,1\rangle\},\\
&\qquad \{\langle 1,0\rangle\} \mapsto \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle\}, \{\langle 1,1\rangle\} \mapsto \{\langle 0,0\rangle, \langle 1,1\rangle\}\Big\}
\end{aligned}$$

where we define these functions only for singletons values in B, since their image on a (possibly empty) set S of values can be retrieved by additivity, i.e. by joining their images on the singleton values that are in S. Thus, it turns out that

$$\begin{aligned}
\text{img}(\alpha_B \circ \text{post}_{S_1} \circ \gamma_B) &= \{\varnothing, \{\langle 1,1\rangle\}\},\\
\text{img}(\alpha_B \circ \text{post}_{S_2} \circ \gamma_B) &= \{\varnothing, \{\langle 0,0\rangle, \langle 0,1\rangle\}, \{\langle 0,0\rangle, \langle 1,1\rangle\}, \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle\},\\
&\qquad \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,1\rangle\}, \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle, \langle 1,1\rangle\}\}.
\end{aligned}$$

We thus have that:

$$\begin{gathered}
\max\left(\{V \in B \mid \alpha_B(\text{post}_{S_1}(\gamma_B(V))) = \varnothing\}\right) = \varnothing\\
\max\left(\{V \in B \mid \alpha_B(\text{post}_{S_1}(\gamma_B(V))) = \{\langle 1,1\rangle\}\}\right) = \{\{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle, \langle 1,1\rangle\}\}
\end{gathered}$$

$$\begin{gathered}
\max\left(\{V \in B \mid \alpha_B(\text{post}_{S_2}(\gamma_B(V))) = \varnothing\}\right) = \varnothing\\
\max\left(\{V \in B \mid \alpha_B(\text{post}_{S_2}(\gamma_B(V))) = \{\langle 0,0\rangle, \langle 0,1\rangle\}\}\right) = \{\langle 0,0\rangle, \langle 0,1\rangle\}\\
\max\left(\{V \in B \mid \alpha_B(\text{post}_{S_2}(\gamma_B(V))) = \{\langle 0,0\rangle, \langle 1,1\rangle\}\}\right) = \{\langle 1,1\rangle\}\\
\max\left(\{V \in B \mid \alpha_B(\text{post}_{S_2}(\gamma_B(V))) = \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle\}\}\right) = \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle\}\\
\max\left(\{V \in B \mid \alpha_B(\text{post}_{S_2}(\gamma_B(V))) = \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,1\rangle\}\}\right) = \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,1\rangle\}\\
\max\left(\{V \in B \mid \alpha_B(\text{post}_{S_2}(\gamma_B(V))) = \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle, \langle 1,1\rangle\}\}\right) = \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle, \langle 1,1\rangle\}
\end{gathered}$$

Hence, by Theorem 3.2, the kernel $\mathcal{K}_F(B)$ of B for $F \triangleq \{\text{post}_{S_1}, \text{post}_{S_2}\}$ is:

$$\begin{aligned}
&\text{Cl}_\cap\big(\{\varnothing, \{\langle 1,1\rangle\}, \{\langle 0,0\rangle, \langle 0,1\rangle\}, \{\langle 0,0\rangle, \langle 1,1\rangle\}, \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle\},\\
&\qquad\qquad \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,1\rangle\}, \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle, \langle 1,1\rangle\}\}\big) =\\
&= \{\varnothing, \{\langle 0,0\rangle\}, \{\langle 1,1\rangle\}, \{\langle 0,0\rangle, \langle 0,1\rangle\}, \{\langle 0,0\rangle, \langle 1,1\rangle\}, \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle\},\\
&\qquad\qquad \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,1\rangle\}, \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle, \langle 1,1\rangle\}\}.
\end{aligned}$$

Let us point out that $\{\langle 0,0\rangle\}$ has been obtained as the intersection $\{\langle 0,0\rangle, \langle 0,1\rangle\} \cap \{\langle 0,0\rangle, \langle 1,1\rangle\}$.

Let us also observe that $\mathcal{K}_F(B)$ can be logically represented as follows:

$$\begin{aligned}
\varnothing &= \textbf{false}\\
\{\langle 0,0\rangle\} &= \neg p_1 \wedge \neg p_2\\
\{\langle 1,1\rangle\} &= p_1 \wedge p_2\\
\{\langle 0,0\rangle,\langle 0,1\rangle\} &= \neg p_1\\
\{\langle 0,0\rangle,\langle 1,1\rangle\} &= p_1 \leftrightarrow p_2\\
\{\langle 0,0\rangle,\langle 0,1\rangle,\langle 1,0\rangle\} &= p_1 \rightarrow \neg p_2\\
\{\langle 0,0\rangle,\langle 0,1\rangle,\langle 1,1\rangle\} &= p_1 \rightarrow p_2\\
\{\langle 0,0\rangle,\langle 0,1\rangle,\langle 1,0\rangle,\langle 1,1\rangle\} &= \textbf{true}
\end{aligned}$$

Thus, $\mathcal{K}_F(B)$ is a proper abstraction of the Boolean abstraction B that, for example, is not able to express precisely the property $p_1 \wedge \neg p_2$.

It is interesting to compare this correctness kernel $\mathcal{K}_F(B)$ with Ball et al.'s [2] Cartesian abstraction C whose definition has been recalled above. It turns out that these two abstractions are not comparable. For instance, $\langle 1,0\rangle \in C$ represents $p_1 \wedge \neg p_2$ which is not represented in $\mathcal{K}_F(B)$, while $p_1 \leftrightarrow p_2 \in \mathcal{K}_F(B)$ cannot be represented in C. However, while the correctness kernel $\mathcal{K}_F(B)$ guarantees no loss of information in analyzing the program P — therefore the analysis with $\mathcal{K}_F(B)$ is able to conclude that $(*)$ cannot be reached — the analysis of P with the Cartesian abstraction C is inconclusive because:

$$\bot_C \xrightarrow{z:=0;\ x:=y} \langle 1,1\rangle \xrightarrow{\textbf{if}(w)\{x{+}{+};\ z:=1;\}} \langle 0,0\rangle \vee_C \langle 1,1\rangle = \langle *,*\rangle$$

where $\gamma_C(\langle *,*\rangle) = \text{States}$, so that with the abstraction C at the exit of the do-while loop one cannot infer that line $(*)$ is unreachable.

7. Related and Future Work

Few examples of abstraction simplifications are known. A general notion of domain simplification and compression in abstract interpretation has been introduced in [13, 16] as a formal dual of abstraction refinement. This duality has been further investigated in [14] to include semantic transforms in a general theory for transforming abstractions based on abstract interpretation. Our domain transformation does not fit directly in this framework. Following [16], given a property $\mathcal{P}$ of abstract domains, the so-called core of an abstract domain A, when it exists, provides the most concrete simplification of A that satisfies the property $\mathcal{P}$, while the so-called compressor of A, when it exists, provides the most abstract simplification of A that induces the same refined abstraction in $\mathcal{P}$ as A does. Examples of compressors include the least disjunctive basis [17], where $\mathcal{P}$ is the abstract domain property of being disjunctive, and examples of cores include the completeness core [19], where $\mathcal{P}$ is the domain property of being complete for some semantic function. The correctness kernel defined in this paper is neither an instance of a domain core nor an instance of a domain compression. The first because, given an abstraction A, the correctness kernel of A characterizes the most abstract domain that induces the same best correct approximation of a function f on A, whilst the notion of domain core for the domain property $\mathcal{P}^f_A$ of inducing the same b.c.a. of f as A would not be meaningful, as this would trivially yield A itself. The second because there is no (unique) maximal domain refinement of an abstract domain which induces the same property $\mathcal{P}^f_A$, as shown by Example 2.3.

The EGAS methodology opens some directions for future work, such as (1) the formalization of a precise relationship between EGAS and CEGAR and (2) an experimental evaluation of the integration in the CEGAR loop of the EGAS-based refinement strategy of Section 5. It is here useful to recall that some work formalizing CEGAR in abstract interpretation has already been done [11, 15, 21]. On the one hand, [15] shows that CEGAR corresponds to iteratively compute a so-called complete shell [19] of the underlying abstract model A with respect to the concrete successor transformer, while [11, 21] formally compare CEGAR with an abstraction refinement strategy based on the computations of abstract fixpoints in an abstract domain. These works can therefore provide a starting point for studying the relationship between EGAS and CEGAR in a common abstract interpretation setting.

Acknowledgements. We are grateful to the anonymous referees for their helpful comments. This research was mostly carried out during a visit of the authors to the Equipe "Abstraction" lead by P. and R. Cousot, at École Normale Supérieure, Paris. This work was partially supported by Microsoft Research Software Engineering Innovation Foundation 2013 Award (SEIF 2013) and by the University of Padova under the Project BECOM.

References

[1] C. Baier and J.-P. Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[2] T. Ball, A. Podelski and S.K. Rajamani. Boolean and Cartesian abstraction for model checking C programs. *Int. J. Softw. Tools Technol. Transfer*, 5:49-58, 2003.

[3] T. Ball and S.K. Rajamani. The SLAM Project: Debugging system software via static analysis. In *Proc. 29th ACM Symposium on Principles of Programming Languages (POPL'02)*, pp. 1-3, ACM Press, 2002.

[4] E.M. Clarke, O. Grumberg, S. Jha, Y. Lu and H. Veith. Counterexample-guided abstraction refinement. In *Proc. 12th International Conference on Computer Aided Verification (CAV'00)*, LNCS 1855, pp. 154-169, Springer, 2000

[5] E.M. Clarke, O. Grumberg, S. Jha, Y. Lu and H. Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5):752-794, 2003.

[6] E.M. Clarke, O. Grumberg and D. Long. Model checking and abstraction. *ACM Trans. Program. Lang. Syst.*, 16(5):1512–1542, 1994.

[7] E.M. Clarke, O. Grumberg and D.A. Peled. *Model checking*. The MIT Press, 1999.

[8] P. Cousot. *Méthodes itératives de construction et d'approximation de points fixes d'opérateurs monotones sur un treillis, analyse sémantique des programmes*. Ph.D. dissertation, Université Scientifique et Médicale de Grenoble, Grenoble, France, 1978.

[9] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. 4th ACM Symposium on Principles of Programming Languages (POPL'77)*, pp. 238–252, ACM Press, 1977.

[10] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Proc. 6th ACM Symposium on Principles of Programming Languages (POPL'79)*, pp. 269–282, ACM Press, 1979.

[11] P. Cousot, P. Ganty and J.-F. Raskin Fixpoint-guided abstraction refinements. In *Proc. 14th International Static Analysis Symposium (SAS'07)*, LNCS 4634, pp. 333-348, Springer, 2007.

[12] S. Das, D.L. Dill, S. Park. Experience with predicate abstraction. In *Proc. 11th International Conference on Computer Aided Verification (CAV'99)*, LNCS 1633, pp. 160-171, Springer, 1999.

[13] G. Filé, R. Giacobazzi, and F. Ranzato. A unifying view of abstract domain design. *ACM Comp. Surveys*, 28(2):333-336, 1996.

[14] R. Giacobazzi and I. Mastroeni. Transforming abstract interpretations by abstract interpretation (Invited Lecture). In *Proc. 15th International Static Analysis Symposium (SAS'08)*, LNCS 5079, pp. 1-17, Springer, 2008.

[15] R. Giacobazzi and E. Quintarelli. Incompleteness, counterexamples, and refinements in abstract model checking. In *Proc. 8th International Static Analysis Symposium (SAS'01)*, LNCS 2126, pp. 356-373, Springer, 2001.

[16] R. Giacobazzi and F. Ranzato. Refining and compressing abstract domains. In *Proc. 24th International Colloquium on Automata, Languages, and Programming (ICALP'97)*, LNCS 1256, pp. 771-781, Springer, 1997.

[17] R. Giacobazzi and F. Ranzato. Optimal domains for disjunctive abstract interpretation. *Sci. Comp. Program.*, 32:177–210, 1998.

[18] R. Giacobazzi and F. Ranzato. Example-guided abstraction simplification. In *Proc. 37th International Colloquium on Automata, Languages, and Programming (ICALP'10)*, LNCS 6199, pp. 211-222, Springer, 2010.

[19] R. Giacobazzi, F. Ranzato and F. Scozzari. Making abstract interpretations complete. *J. ACM*, 47(2):361-416, 2000.

[20] S. Graf and H. Saïdi. Construction of abstract state graphs with PVS. In *Proc. 9th International Conference on Computer Aided Verification (CAV'97)*, LNCS 1254, pp. 72-83, Springer, 1997.

[21] F. Ranzato, O. Rossi Doria and F. Tapparo. A forward-backward abstraction refinement algorithm. In *Proc. 9th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'08)*, LNCS 4905, pp. 248-262, Springer, 2008.

[22] F. Ranzato and F. Tapparo. Strong preservation as completeness in abstract interpretation. In *Proc. 13th European Symposium on Programming (ESOP'04)*, LNCS 2986, pp. 18-32, Springer, 2004.

[23] F. Ranzato and F. Tapparo. Generalized strong preservation by abstract interpretation. *J. Logic and Computation*, 17(1):157-197, 2007.