Fundamenta Informaticae XX (2010) 1–25
IOS Press

Observational Completeness on Abstract Interpretation

Gianluca Amato and Francesca Scozzari
Dipartimento di Scienze
Università di Chieti-Pescara
Pescara, Italy
{*amato,scozzari*}*@sci.unich.it*

Abstract. In the theory of abstract interpretation, a domain is complete when abstract computations are as precise as concrete computations. In addition to the standard notion of completeness, we introduce the concept of observational completeness. A domain is observationally complete for an observable π when abstract computations are as precise as concrete computations, if we only look at properties in π. We prove that continuity of state-transition functions ensures the existence of the least observationally complete domain and we provide a constructive characterization. We study the relationship between the least observationally complete domain and the complete shell. We provide sufficient conditions under which they coincide, and show several examples where they differ, included a detailed analysis of cellular automata.

Keywords: Abstract interpretation, completeness, static analysis, cellular automata.

1. Introduction

Abstract interpretation [5, 6] is a general theory for approximating the behavior of a discrete dynamic system. It has been applied to many different fields, such as program analysis, verification, model checking and semantics hierarchies. Abstract interpretation allows us to observe many kinds of properties, e.g., termination, types and security properties. The idea is to replace the formal semantics of a system with an abstract semantics, computed over a domain of abstract objects. There are many different methods to describe the semantics of a system. Most of them are based on a partially ordered set (poset) $\langle C, \leq_C \rangle$ of states and a set F of monotone state-transition functions $f : C \to C$. The semantics $\mathcal{S}$ is defined as the (least) fixpoint of a semantic function $\mathcal{F}$ obtained as a composition

Address for correspondence: Gianluca Amato, Dipartimento di Scienze, viale Pindaro 42, I-65127 Pescara, Italy

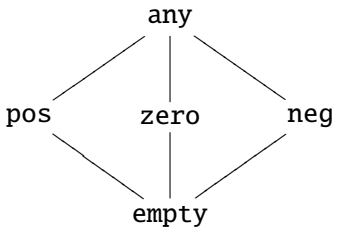


Figure 1: The abstract domain *Sign*

of state-transition functions. The poset $\langle C, \leq_C \rangle$ is called *concrete domain* and $\mathcal{S} = \mathrm{lfp}\,\mathcal{F}$ is called the *concrete semantics*.

An abstract interpretation is specified by the poset $\langle A, \leq_A \rangle$ of *abstract objects*. The abstract objects describe the properties of the system we are interested in. The relationship between concrete and abstract objects is formalized by a monotone concretization map $\gamma : A \to C$ which, given a property $a \in A$, yields the biggest concrete state $c \in C$ which enjoys the property a. Therefore, a property a is a *correct approximation* of a concrete state c when $c \leq_C \gamma(a)$.

For instance, consider the concrete domain $\wp(\mathbb{Z})$ with the standard ordering given by inclusion, and *Sign* = $\{\texttt{empty}, \texttt{pos}, \texttt{neg}, \texttt{zero}, \texttt{any}\}$ ordered as depicted in Figure 1. The intuition is that `pos` represents the set of (strictly) positive integers, `zero` represents the singleton $\{0\}$, while `empty` represents the empty set of integers. This may be formalized by defining γ as follows: $\gamma(\texttt{empty}) = \emptyset$, $\gamma(\texttt{pos}) = \{n \in \mathbb{Z} \mid n > 0\}$, $\gamma(\texttt{neg}) = \{n \in \mathbb{Z} \mid n < 0\}$, $\gamma(\texttt{zero}) = \{0\}$, $\gamma(\texttt{any}) = \mathbb{Z}$.

Often, it is possible to define a monotone abstraction map $\alpha : C \to A$ which yields the largest properties a enjoyed by a concrete object c, such that $c \leq_C \gamma(a) \iff \alpha(c) \leq_A a$. In the previous example, the abstraction is given by $\alpha(c) = \bigvee_A \{a \in A \mid c \leq_C \gamma(a)\}$. For instance, $\alpha(\{-1, -2\}) = \texttt{neg}$ while $\alpha(\{-1, 0\}) = \texttt{any}$.

An *abstract domain* is given by the poset A of the abstract objects and the pair of maps $\langle \alpha, \gamma \rangle$. However, since γ is uniquely determined by α and viceversa, in the following we specify an abstract domain just by giving α.

The goal of any abstract interpretation is to compute $\alpha(\mathcal{S})$, that is, to find out the properties enjoyed by the semantics of the system. Instead of computing $\mathcal{S}$ (which, in general, is not feasible) and then applying α, the idea is to replace, in the the definition of $\mathcal{F}$, every state-transition function f with an abstract counterpart $f^\# : A \to A$, which must be *correct*. We say that $f^\#$ is correct if, whenever a is a correct approximation of c, then $f^\#(a)$ is a correct approximation of $f(c)$. This is equivalent to say that any abstract computation $f^\#(\alpha(c))$ approximates the corresponding concrete computation $f(c)$, i.e.:

$$\alpha \circ f \leq_{A \to A} f^\# \circ \alpha \ , \tag{1}$$

where $\leq_{A \to A}$ is the pointwise extension of $\leq_A$. If we replace the state-transition functions in the definition of $\mathcal{F}$ with suitable correct abstractions, we obtain a new semantic function $\mathcal{F}_\#$ and a new abstract semantics $\mathcal{S}_\# = \mathrm{lfp}\,\mathcal{F}_\#$, and the theory of abstract interpretation ensures that

$$\alpha(\mathcal{S}) \leq_A \mathcal{S}_\# \ . \tag{2}$$

In particular, each state-transition function f has a *best correct abstraction*, denoted by f^α, and defined as $f^\alpha = \alpha \circ f \circ \gamma$. We denote by $\mathcal{F}_\alpha$ the semantic function obtained by replacing each state-transition function f in $\mathcal{F}$ with f^α, while $\mathcal{S}_\alpha = \mathrm{lfp}\,\mathcal{F}_\alpha$. Note that $\mathcal{F}_\alpha$ is not necessarily the best correct abstraction of $\mathcal{F}$.

As an example of abstract state-transition function, consider $inc : \wp(\mathbb{Z}) \to \wp(\mathbb{Z})$ such that $inc(X) = \{n + 1 \mid n \in X\}$. The best correct abstraction of inc is

$$inc^\alpha(a) = \begin{cases} \texttt{empty} & \text{if } a = \texttt{empty}, \\ \texttt{pos} & \text{if } a = \texttt{zero} \text{ or } a = \texttt{pos}, \\ \texttt{any} & \text{otherwise.} \end{cases}$$

Completeness

Generally speaking, the inequalities (1) and (2) are strict. This means that computing in the abstract domain is (strictly) less precise than computing in the concrete one. For instance, $\alpha(inc(-1)) = \texttt{zero}$ but $inc^\alpha(\alpha(-1)) = \texttt{any}$. When $\alpha \circ f = f^\alpha \circ \alpha$ we say that the abstract domain is *complete* for the function f. Intuitively, when this happens, the best correct abstraction f^α perfectly mimics the concrete function f. For example, given $sq(X) = \{x^2 \mid x \in X\}$, the best correct abstraction is

$$sq^\alpha(a) = \begin{cases} \texttt{pos} & \text{if } a = \texttt{pos} \text{ or } a = \texttt{neg}, \\ a & \text{otherwise.} \end{cases}$$

It follows that $sq(\{-1, -2\}) = \{1, 4\}$, and its abstraction is $\alpha(sq(\{-1, -2\})) = \texttt{pos}$, meaning that the square of any integer in $\{-1, -2\}$ is positive. The same result may be obtained by first abstracting $\{-1, -2\}$ and then computing sq^α, since $sq^\alpha(\alpha(\{-1, -2\})) = sq^\alpha(\texttt{neg}) = \texttt{pos}$. It is easy to show that, for any set of integers $X \in \wp(\mathbb{Z})$, it holds that $sq^\alpha(\alpha(X)) = \alpha(sq(X))$. Thus, the abstract domain *Sign* is complete for the function sq.

Completeness enjoys many good properties. If an abstract domain α is complete for f and g, then it holds that:

- α is complete for $f \circ g$ and $f^\alpha \circ g^\alpha = (f \circ g)^\alpha$;
- $\alpha(\mathrm{lfp}\, f) = \mathrm{lfp}(f^\alpha)$.

When α is complete for all the state-transition functions in F, then $\mathcal{S}_\alpha = \alpha(\mathcal{S})$, and one does not lose precision by computing on the abstract domain.

In most cases, we are interested in properties expressed by an abstract domain π which is not precise enough, that is, the abstract semantics $\mathcal{S}_\pi$ does not coincide with $\pi(\mathcal{S})$. Therefore, we need to refine the abstract domain π and replace it with a new domain α such that $\pi(\mathcal{S})$ may be recovered by $\mathcal{S}_\alpha$. The domain α is the *computational domain* and π is the *observational domain*, since it contains all the properties we want to observe. The abstract objects in α which do not belong to π are only used to compute intermediate steps in order not to lose precision. Obviously, we want to keep α as small as possible.

In the literature of abstract interpretation, the standard way of refining π in order to get α is to compute the least domain which includes π and is complete for the state-transition functions. This is called the *complete shell* of π, and may be constructively computed [15].

The Goal

In this paper, we show that the complete shell may not be the smallest abstract domain which enables us to recover the property $\pi(\mathcal{S})$. This is because a complete domain α never loses precision for any observation of objects in α, i.e., $\mathcal{S}_\alpha = \alpha(\mathcal{S})$, while we may allow some loss in precision as long as

observations in π are not affected, i.e., we only require $\pi(\mathcal{S}_\alpha) = \pi(\mathcal{S})$. This observation naturally suggests another notion of completeness, which we call *observational completeness*. A domain α is observationally complete for a set F of state-transition functions and an observational domain π when every concrete computation may be approximated in α without losing precision over the properties in π. We show that, when all the state-transition functions are continuous, the least observationally complete domain exists and we provide a constructive characterization.

We study the relationship between observational and standard completeness. We prove that any complete domain which contains π is also observationally complete for π, but the converse does not hold. We compare the least observationally complete domain and the complete shell and prove that, in the general case, the former is more abstract than the latter. This is illustrated by several examples. In particular, we develop a static analysis of cellular automata and show an instance where the least observationally complete domain is strictly more abstract than the complete shell.

We prove that, under the hypothesis of additivity of the state-transition functions, least observationally complete domains and complete shells coincide. In the meanwhile, we give an alternative constructive characterization of complete shells which simplifies the original one.

Observational completeness compares the precision of a computational domain α to the precision of the concrete domain. We generalize this notion by comparing the precision of α with the precision of a fixed domain β, different from the concrete one. However, we show that the notion of least observational complete domain cannot be generalized to this case and provide a partial solution with the concept of observational equivalence. Two abstract domains are observationally equivalent for the domain π if they have exactly the same precision on every abstract computation, when the results are observed over π. It turns out that the least domain which is observationally equivalent to a given α does exist, and can be recovered as a special case of observational completeness.

Plan of the Paper

The next section is devoted to presenting completeness and the motivations which led to the introduction of observational completeness. In Section 3 we formally define the notion of observational completeness and give a constructive characterization for the least observationally complete domain. In Section 4 we study the relationship between observational completeness and standard completeness. In Section 5 we provide an example of static analysis of cellular automata. In Section 6 we generalize observational completeness and introduce observational equivalence. Finally, in Section 7 we compare observational completeness to other notions of completeness in the literature.

2. Completeness on Abstract Interpretation

In the abstract interpretation theory, abstract domains can be equivalently specified either by *Galois connections* or by *upper closure operators* (ucos) [5]. When an abstract domain A is specified by a Galois connection, i.e., a pair of abstraction and concretization maps $\langle \alpha, \gamma \rangle$, then $\gamma \circ \alpha \in uco(C)$ is the corresponding uco on C. On the contrary, given an uco α, the corresponding Galois connection is $\langle \alpha, id \rangle$. While Galois connections are closer to implementation, since they provide a representation for abstract objects, ucos are more convenient for theoretical reasoning, expecially when comparing abstract domains. In the rest of the paper, we will use ucos. Moreover, we assume that the concrete domain C is a complete lattice, which is a standard hypothesis in the abstract interpretation theory.

An uco α on the concrete domain C is a monotone, idempotent (i.e., $\alpha(\alpha(x)) = \alpha(x)$) and extensive (i.e., $\alpha(x) \geq x$) operator on C. Each uco α on C is uniquely determined by the set of its fixpoints,

which is its image, i.e. $\alpha(C) = \{x \in C \mid \alpha(x) = x\}$, since $\alpha = \lambda x.\bigwedge\{y \in \alpha(C) \mid x \leq y\}$. Moreover, a subset $X \subseteq C$ is the set of fixpoints of an uco on C iff X is meet-closed, i.e. $X = \mathcal{M}(X) = \{\bigwedge Y \mid Y \subseteq X\}$, with the proviso that $\bigwedge \emptyset = \top_X$. For any $X \subseteq C$, $\mathcal{M}(X)$ is called the Moore-closure of X. Often, we will identify closures with their sets of fixpoints. This does not give rise to ambiguity, since one can distinguish their use as functions or sets according to the context. It is well known that the set $uco(C)$ of all ucos on C, endowed with the pointwise ordering $\subseteq$, gives rise to a complete lattice. Intuitively, $\alpha \subseteq \beta$ means that α is more abstract than β (or β is more concrete than α). The most abstract domain in $uco(C)$ is $\{\top_C\}$, while the most concrete domain in $uco(C)$ is C itself.

In order to simplify notation, we only consider the standard case when the *computational ordering* on C corresponds to the *approximation ordering*. This is always the case when working with operational semantics, while denotational semantics might need two different orderings [7]. However, the latter would only require to modify the fixpoint preservation theorem (Theorem 3.2).

2.1. Completeness

The notion of completeness in abstract interpretation first appeared in Cousot and Cousot's seminal work [5]. An abstract domain $\alpha \in uco(C)$ is *complete* for a function $f : C \to C$ iff $\alpha \circ f = \alpha \circ f \circ \alpha$ holds. Mycroft [17] introduced the idea of refining abstract domains in order to ensure completeness. Working in a predicate-based approach to abstract interpretation, he proposed to remove useless objects from the abstract domain, inspired by algorithms for automata state minimization. This idea has been fully developed in Giacobazzi et al. [15], which give a constructive characterization of complete abstract domains, under the assumption of dealing with continuous concrete functions. A function $f : C \to C$ is (Scott-)continuous if it preserves least upper bounds of chains in C, i.e., $f(\bigvee B) = \bigvee f(B)$ for any chain $B \subseteq C$. The goal is to build the least (i.e., most abstract) domain in $uco(C)$ which includes a given domain α and which is complete for a set $F \subseteq C \to C$ of continuous state-transition functions, i.e., for each function in F. Giacobazzi et al. [15] define a mapping $\mathcal{R}_F : uco(C) \to uco(C)$ as follows:

$$\mathcal{R}_F(\alpha) = \mathcal{M}\Big(\bigcup_{f\in F,\ a\in\alpha} \max(\{x \in C \mid f(x) \leq a\})\Big) \ , \tag{3}$$

where $\max(X)$ is the set of maximal elements in X. They prove that the most abstract domain which includes α and is complete for F is $\mathrm{lfp}(\lambda\eta.\mathcal{M}(\alpha \cup \mathcal{R}_F(\eta)))$. This domain is called the *complete shell* of α for F and will be denoted by $S_{\alpha,F}$.

Completeness and its many variants has been subsequently applied in many different areas, such as abstract model checking [12, 13], behavioural equivalences [18, 19], abstract domain refinements [16, 21], and security [9, 11].

2.2. Towards Observational Completeness

We argue that the notion of completeness has been often misused in the literature, for a lack of a more suitable concept. We consider an example from Giacobazzi and Ranzato [13], based on Cousot and Cousot *temporal abstract interpretation* [8]. Cousot and Cousot introduce a temporal calculus called $\overset{\frown}{\mu}$-calculus, a past- and future-time extension of Kozen's μ-calculus, endowed with a generalized quantification over traces. The semantics of a temporal formula $[\![\phi]\!]$ is the set of traces making ϕ true and is inductively defined on the structure of ϕ starting from a set of basic trace transformers. If M is the set of traces generate by a transition system, it is possible to define a *universal model checking abstraction* $\alpha_{state} : \wp(Traces) \to \wp(States)$ such that, for every linear formula ϕ, $\alpha_{state}([\![\phi]\!])$ is the set of states which satisfy ϕ.

The state-based semantics $[\![\phi]\!]_{state}$ is inductively defined by replacing each trace transformer with its corresponding best abstraction. Abstract interpretation guarantees the obvious correctness property, namely, $\alpha_{state}([\![\phi]\!]) \supseteq [\![\phi]\!]_{state}$[1], but the inequality is generally strict, essentially for the same reason that LTL and CTL have incomparable expressive powers. This means that we cannot use a state-based algorithm for universal model checking of linear $\overset{\frown}{\mu}^*$-formulas.

Giacobazzi and Ranzato [13] observe that:

> This opens the question whether it is possible to find some different approximation A of the trace-based model checking problem which (1) is still related to states, namely A refines or abstracts from sets of states, and (2) induces an approximated model checking which is instead equivalent to trace-based model checking: for any $s \in States$ and any linear formula ϕ,
>
> $$M, s \models_A \phi \Leftrightarrow M, s \models_{trace} \phi \ . \qquad (*)$$

The above formula amounts to say that $\alpha_{state}([\![\phi]\!]_A) = \alpha_{state}([\![\phi]\!])$, where $[\![\phi]\!]_A$ is the abstract semantics inductively defined from A. Giacobazzi and Ranzato [13] determine such an A by calculating the complete shell of α_{state}, which is the most abstract domain α_B including α_{state} and such that $[\![\phi]\!]_B = \alpha_B([\![\phi]\!])$. They find that α_B is essentially the powerset of traces, and therefore the only refinement of the state-based semantics that induces a trace-complete model checking is the trace-based semantics itself.

We argue that looking for complete domains is restrictive. There may be an intermediate domain between states and traces which is not complete, but nonetheless satisfies (*). In other words, we are not interested in the complete shell of α_{state}, but in the least domain which is observationally complete for α_{state}. In this paper we introduce the machinery needed to compute least observationally complete domains. As a consequence, we will be able to show that, in this case, the two abstractions coincide.

3. Observational Completeness

In abstract interpretation, it is common that, in order to observe a property π with a good deal of precision, we need to compute in a richer domain $\alpha \supseteq \pi$. In the following, we call π the *observational domain* and α the *computational domain*. In the rest of the paper, we assume given a complete lattice C (the concrete domain), a set $F \subseteq C \to C$ of monotone functions and an uco $\pi \subseteq C$ which represents the set of observable properties.

A common problem is to find a domain α such that if we perform any computation on α and we project over π, we obtain the same result of the concrete computation, projected over π. In order to formalize this notion, we first need to define the concept of computation.

Definition 3.1. (Computation)
A finite, possibly empty, sequence $\xi = \langle f_1, \ldots, f_n \rangle$ of elements of F is called *computation* (over F). For any domain $\alpha \in uco(C)$, we denote by ξ_α the function $\alpha \circ f_1 \circ \ldots \circ \alpha \circ f_n \circ \alpha$. As a special case, when ξ is the empty computation, we let $\xi_\alpha = \alpha$.

Note that, if id is the identity abstraction, then $\xi_{id} = f_1 \circ \ldots \circ f_n$. When it does not cause ambiguities, we will use ξ to denote both the computation and the function ξ_{id}. We believe that a generalization of the notion of computation to infinite sequences would be troublesome, since there is no standard

[1] Here we have $\supseteq$ instead of $\subseteq$ because the concrete and abstract domains are ordered by the superset ordering.

definition for infinite composition of functions. However, considering only finite computations is not restrictive, since the interesting cases of infinite computations, generated by least fixpoints, are covered by the fixpoint preservation theorem (Theorem 3.2).

We are now able to compare domains in terms of the precision of their computations. We say that a domain α is *more precise than* a domain β if the result of any computation on α projected over π is more precise than (it is approximated by) the result of the corresponding computation on β projected over π.

Definition 3.2. (More Precise than)
We say that α is *more precise than* β (for computing F observing π), and we write it as $\alpha \preceq_F^\pi \beta$, when

$$\pi\xi_\alpha(c) \leq \pi\xi_\beta(c)$$

for every computation ξ and $c \in C$.

When it does not cause ambiguities, we will write just $\preceq$ instead of $\preceq_F^\pi$. It is easy to check that $\preceq$ is a preorder, which may be viewed as a generalization of the standard ordering between ucos: if $\alpha \supseteq \beta$ then $\alpha \preceq \beta$. Our notion is more general than the standard ordering since it allows us to compare two different domains (α and β) w.r.t. their precision on a third domain (π), and does not require neither α nor β to be in any relation with π. Due to the empty computation, if $\pi = id$ then $\alpha \preceq \beta$ iff $\alpha \supseteq \beta$.

Our formal notion of precision suggests to define a corresponding notion of completeness. We say that a domain α is *observationally complete* (for F and π) if any computation on α projected over π gives the same result of the corresponding concrete computation, projected over π.

Definition 3.3. (Observational Completeness)
We say that a domain α is *observationally complete* (for F and π) if α is more precise than the concrete domain, i.e., $\alpha \preceq_F^\pi id$.

When $\pi = \alpha$, the above definition boils down to the standard notion of completeness of α for the set of functions F.

Among all the observationally complete domains, we are interested in the least (most abstract) one w.r.t. set inclusion, which will be denoted by $\sigma_{\pi,F}$. In general, the least observationally complete domain does not exist, as the following example shows.

Example 3.1. Let us consider the diagram on Figure 2, where the nodes are the elements of the domain $C = \{\top, \bot, a, b, c_1, c_2, \ldots, c_i, \ldots\}$, solid and dotted edges represent the ordering on C and dashed arrows represent a function $f : C \to C$.

Let $\pi = \{\top, a\}$, $\alpha = \{\top, a, b, \bot\} \cup \{c_i \mid i \text{ is even}\}$ and $\beta = \{\top, a, b, \bot\} \cup \{c_i \mid i \text{ is odd}\}$. It is easy to check that both α and β are observationally complete. However, $\gamma = \alpha \cap \beta = \{\top, a, b, \bot\}$ is not observationally complete, since, for the computation $\xi = \langle f \rangle$, we have that $\pi(\xi(c_1)) = a$ while $\pi(\xi_\gamma(c_1)) = \pi(\gamma(f(\gamma(c_1)))) = \pi(\gamma(f(a)) = \top$. □

As a key result we show that, if all the functions in F are continuous, the least observationally complete domain exists. In some proofs we will make use of the Hausdorff's maximality principle [1]. We recall that a chain Y in a poset P is maximal (with respect to set inclusion) whenever for any other chain Y' in P, $Y \subseteq Y'$ implies $Y = Y'$. The Hausdorff's maximality principle says that every chain in a poset P can be extended to a maximal chain in P.

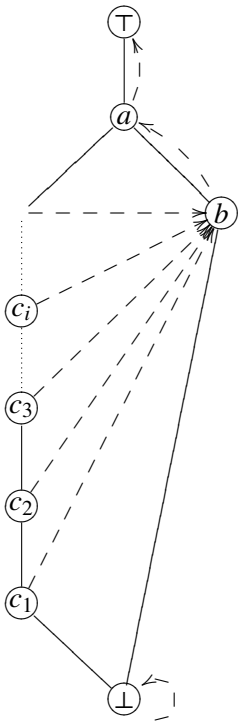


Figure 2: No least observationally complete domain for $\pi = \{\top, a\}$.

Theorem 3.1. If F is a set of continuous functions and $\pi \in uco(C)$, then the least observationally complete domain (for F and π) is

$$\sigma_{\pi,F} = \mathcal{M}\Big(\bigcup\{\max\{x \in C \mid \xi(x) \leq a\} \mid \xi \text{ computation and } a \in \pi\}\Big) \ .$$

Proof:
To ease notation, in this proof we will write σ in place of $\sigma_{\pi,F}$. First of all, we show that σ is observationally complete. We prove, by induction on the length of ξ, that $\xi(x) \leq a$ implies $\xi_\sigma(x) \leq a$ for each computation ξ and $a \in \pi$. If $|\xi| = 0$, then $\xi(x) = x$ and $\xi_\sigma(x) = \sigma(x)$. Note that $\sigma \supseteq \pi$ since if $a \in \pi$ then $a = \bigvee\{x \in C \mid x \leq a\}$. Hence $x \leq a$ implies $\sigma(x) \leq a$. Now assume $|\xi| = i + 1$. If $\xi(x) \leq a$, consider the poset $C' = \{c \in C \mid \xi(c) \leq a\}$ and the chain $\{x\} \subseteq C'$. By Hausdorff's maximality principle there exists a maximal chain $Y \supseteq \{x\}$ which is contained in C'. Let $y = \bigvee Y$. By continuity of ξ, we have that $\xi(y) \leq a$. Since Y is a maximal chain in C', then $y \in \max C'$. Moreover, by definition of σ we have that $y \in \sigma$. It follows that $\xi(\sigma(x)) \leq \xi(y) \leq a$. If $\xi = \xi' \cdot f$, we have that $\xi_\sigma(x) = \xi'_\sigma(f(\sigma(x))) \leq a$ since $\xi'(f(\sigma(x))) = \xi(\sigma(x)) \leq a$ and the inductive hypothesis.

Now we show that σ is the least observationally complete domain. Assume, without loss of generality, that $v \in \max\{x \in C \mid \xi(x) \leq a\}$ for some computation ξ and $a \in \pi$. By continuity of functions in F and by the Hausdorff's maximality principle, we have that $\xi(v) \leq a$. Let α be a domain such that $v \notin \alpha$. Then, $\xi(\alpha(v)) \not\leq a$ since $\alpha(v) > v$ and by definition of v. Hence, also $\xi_\alpha(v) \not\leq a$, which means that α is not observationally complete. □

The hypothesis of continuity for state-transition functions is absolutely non-restrictive, since we are assuming that the approximation ordering of C corresponds to the computational ordering, and continuity is a standard requirement of semantic functions. When the two orderings differ and continuity does not hold for the approximation ordering, we may move to a collecting semantics with additive state-transition functions [7].

Theorem 3.2. (Fixpoint Preservation)
Let α be observationally complete for the set of continuous functions F and π. Given a computation ξ and $c \in C$ such that $\xi(c) \geq c$, we have that

$$\pi(\mathrm{lfp}_c\, \xi) = \pi(\mathrm{lfp}_c\, \xi_\alpha) \ ,$$

where $\mathrm{lfp}_c\, f$ is the least fixpoint of f bigger than c.

Proof:
Remember that $\mathrm{lfp}_c\, \xi = \bigvee_{i<\omega} \xi^i(c)$ and $\mathrm{lfp}_c\, \xi_\alpha = \bigvee_{i<\omega} \xi^i_\alpha(c)$. It clearly holds that $\pi(\mathrm{lfp}_c\, \xi) \leq \pi(\mathrm{lfp}_c\, \xi_\alpha)$, since α is extensive. We now show the other direction. Since π is an uco, it is complete for arbitrary lubs [6]. It follows that:

$$\pi\left(\bigvee_{i<\omega} \xi^i_\alpha(c)\right) = \pi\left(\bigvee_{i<\omega} \pi(\xi^i_\alpha(c))\right) \ .$$

Since α is observationally complete for F and π, then

$$\pi\left(\bigvee_{i<\omega} \pi(\xi^i_\alpha(c))\right) = \pi\left(\bigvee_{i<\omega} \pi(\xi^i(c))\right) \ ,$$

which is equivalent to $\pi\left(\bigvee_{i<\omega} \xi^i(c)\right)$. □

The previous theorem allows us to safely approximate the concrete semantic function $\mathcal{F}$ (which is ξ_{id} for some computation ξ) with ξ_α, without losing precision on π: in other words, the abstract semantics projected over π is equal to the concrete semantics projected over π, i.e., $\pi(\mathcal{S}) = \pi(\mathrm{lfp}\, \xi) = \pi(\mathrm{lfp}\, \xi_\alpha) = \pi(\mathcal{S}_\alpha)$.

3.1. A Constructive Characterization

If we look for an explicit representation of the least observationally complete domain using the characterization given in Theorem 3.1, we need to find out the set $\max\{\xi(x) \mid x \leq a\}$ for each computation ξ and $a \in \pi$. This is in general difficult to achieve, even in the case when C is finite. Therefore, we present an alternative constructive characterization which works iteratively and does not require to compute the composition of the functions in F. This characterization is similar to the one given in [15] for the complete shell.

Definition 3.4. (Maximal preimage)
Given $S \in \wp(C)$ and a function $f : C \to C$, we denote by $f^*(S) \in \wp(C)$ the *maximal preimage of S under f*, defined as

$$f^*(S) = \max f^{-1}(\downarrow S) \ ,$$

where $\downarrow S$ is the downward closure of S, i.e., the set $\{x \in C \mid \exists a \in S\, .\, x \leq a\}$. When S is the singleton $\{a\}$, we will write $f^*(a)$ in place of $f^*(S)$.

Example 3.2. Consider the concrete domain $C = \{\top, a, b, c, d, \bot\}$ and the function f depicted in Figure 3. We consider the observable domain $\pi = \{\top, a\}$ and compute $f^*(a)$. By definition, it is given by $\max f^{-1}(\downarrow\{a\}) = \max\{x \in C \mid f(x) \leq a\} = \{b, c\}$. Intuitively, it means that we need to consider the points b and c if we want to observe the computation f on π without losing precision. Most importantly, the fact that both b and c are in the preimage of a, suggests us that we do not need to distinguish between these two points. Formally, if we know that either $x \leq c$ or $x \leq b$, we also know that $\pi(f(x)) \leq a$, even if we do not know which of the two inequalities holds. In fact, if we compute $f^*(\{b, c\})$, we find out that $f^{-1}(\downarrow\{b\}) = \{b, d, \bot\}$, while $f^{-1}(\downarrow\{c\}) = \{\bot\}$. But we only need the maximal elements of $f^{-1}(\downarrow\{b\}) \cup f^{-1}(\downarrow\{c\})$, which is the singleton $\{b\}$. Note that, in this case, we do not select any point in the preimage of c. But this choice does not lose precision. In fact, any computation starting from $\bot$ (the only object in the preimage of c) will converge towards b, and thus $\bot$ is useless for observing a. □

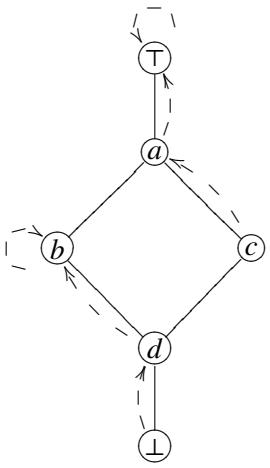


Figure 3: Examples 3.2 and 4.2

This idea, however, only works if $f^*(S)$ is a faithful representation of $f^{-1}(\downarrow S)$, i.e., if $\downarrow f^*(S) = \downarrow \max f^{-1}(\downarrow S) = f^{-1}(\downarrow S)$. In the general case, $\downarrow \max U \neq U$. However, if U is Scott-closed, i.e., it is downward closed and every chain in U has a least upper bound, then the equality holds.

Lemma 3.1. Given $U \in \wp(C)$, if U is Scott-closed, then $\downarrow \max U = U$.

Proof:
It is immediate to check that, if U is downward closed, then $\downarrow \max U \subseteq U$. On the converse, given $x \in U$, consider a maximal chain V in the set of all the elements of U bigger than x (which exists by Hausdorff's maximal principle). Then, since U is Scott-closed, $\bigvee V \in U$. Since V is maximal, v is maximal too, i.e., $v \in \max U$ and $x \in \downarrow \max U$. □

Lemma 3.2. If $U \in \wp(C)$, $\downarrow U$ is Scott-closed and $f_1, f_2 : C \to C$ are two continuous functions, then $(f_1 \circ f_2)^*(U) = f_2^*(f_1^*(U))$.

Proof:
By Lemma 3.1 and since preimages of Scott-closed sets are Scott-closed, we have that $f_2^*(f_1^*(U)) = \max f_2^{-1}(\downarrow \max f_1^{-1}(\downarrow U)) = \max f_2^{-1}(f_1^{-1}(\downarrow U)) = (f_1 \circ f_2)^*(U)$. □

Using the concept of maximal preimage, we may rewrite $\mathcal{R}_F$ in Equation 3 as follows:

$$\mathcal{R}_F(\alpha) = \mathcal{M}\left(\bigcup_{f \in F,\ a \in \alpha} f^*(a)\right) \ . \tag{4}$$

The $\mathcal{R}_F$ operator takes subsets of C into subsets of C: it computes the maximal preimage of every element in its argument and put all the results together. We now define a variant of $\mathcal{R}_F$ which works over subsets of subsets of C: at each step it takes a subset of C, computes the maximal preimage, and collect all these results without joining them together.

Definition 3.5. Given a set F of continuous functions, the refinement operator

$$\dot{\mathcal{R}}_F : \wp(\wp(C)) \to \wp(\wp(C))$$

is given by

$$\dot{\mathcal{R}}_F(H) = \{f^*(S) \mid f \in F, S \in H\} \ .$$

Note that when $H \in \wp(\wp(C))$ is a set of singletons, i.e. $H = \{\{a_1\}, \{a_2\}, \ldots\}$, we have that $\bigcup \dot{\mathcal{R}}_F(H) = \mathcal{R}_F(\bigcup H) = \mathcal{R}_F(\{a_1, a_2, \ldots\})$.

The previous definition is the key point which allows us to keep together objects which have been generated by the same property $a \in \pi$ and the same computation. Intuitively, working with sets of sets of object, keeps tracks of the history of objects.

Theorem 3.3. (Constructive characterization)
Given a set F of continuous functions and an observable domain π, the least observationally complete domain for F and π is

$$\sigma_{\pi,F} = \mathcal{M}\Big(\bigcup\{S \mid i < \omega, S \in \dot{\mathcal{R}}^i_F(H_\pi)\}\Big) \ ,$$

where $H_\pi = \{\{a\} \mid a \in \pi\}$.

Proof:
First of all, if $\xi = \langle f_1, \ldots, f_n\rangle$ is a computation and $c \in C$, then $\max\{x \in C \mid \xi(x) \leq c\} = \xi^*(c)$. We may rewrite the characterization of $\sigma_{\pi,F}$ given in Theorem 3.1 as follows:

$$\sigma_{\pi,F} = \mathcal{M}\Big(\bigcup\{\xi^*(a) \mid \xi \text{ computation and } a \in \pi\}\Big) \ .$$

Therefore, it is enough to prove that, for each $i < \omega$,

$$\dot{\mathcal{R}}^i_F(H_\pi) = \{\xi^*(a) \mid \xi \text{ computation of length } i \text{ and } a \in \pi\} \ ,$$

The proof is by induction. For $i = 0$, we have that $\dot{\mathcal{R}}^0_F(H_\pi) = H_\pi$ by definition. The only computation of length 0 is the empty computation, which corresponds to the identity map. Hence $\{id^*(a) \mid a \in \pi\} = \{\{a\} \mid a \in \pi\} = H_\pi$.

If $i = j+1$, we have that $\dot{\mathcal{R}}^i_F(H_\pi) = \dot{\mathcal{R}}_F(\{\xi^*(\{a\}) \mid \xi \text{ computation of length } j \text{ and } a \in \pi\}) = \{f^*(\xi^*(a)) \mid f \in F, \xi \text{ computation of length } j \text{ and } a \in \pi\}$. Every computation $\bar{\xi}$ of length i is of the form $\xi \cdot f$ with $f \in F$ and ξ of length j. By Proposition 3.2, and since $\downarrow\{a\}$ is Scott-closed, we have that $\bar{\xi}^*(a) = f^*(\xi^*(a))$, hence $\dot{\mathcal{R}}^i_F(H_\pi) = \{\bar{\xi}^*(a)) \mid \bar{\xi}$ is a computation of length i and $a \in \pi\}$, which concludes the proof. □

Example 3.3. Consider the concrete domain $C = \{\top, a, b, c, d, \bot\}$ depicted in Fig. 3. Assume $\pi = \{\top, a\}$. Therefore $H_\pi = \{\{\top\}, \{a\}\}$. At the first step we compute $\dot{\mathcal{R}}^1_F(\{\{\top\}, \{a\}\}) = \{f^*(\top), f^*(a)\}$. It is immediate to see that $f^*(\top) = \max f^{-1}(C) = \{\top\}$ (it holds for any function f) and

$$f^*(a) = \max\{x \in C \mid f(x) \leq a\} = \max\{b, c, d, \bot\} = \{b, c\} \ .$$

At the second step, we need to compute $\dot{\mathcal{R}}^2_F(\{\{\top\},\{a\}\}) = \dot{\mathcal{R}}_F(\{\{\top\},\{b,c\}\}) = \{f^*(\top), f^*(\{b,c\})\}$. We have that:

$$f^*(\{b,c\}) = \max(f^{-1}(\downarrow\{b\}) \cup f^{-1}(\downarrow\{c\})) = \max(\{b,d,\bot\} \cup \{\bot\}) = \max\{b,d,\bot\} = \{b\} \ .$$

Therefore $\dot{\mathcal{R}}^2_F(\{\{\top\},\{a\}\}) = \{\{\top\},\{b\}\}$. With another step we get $\dot{\mathcal{R}}^3_F(\{\{\top\},\{a\}\}) = \{\{\top\},\{b\}\} = \dot{\mathcal{R}}^2_F(\{\{\top\},\{a\}\})$. We may stop the iteration and $\bigcup\{S \mid i < \omega, S \in \dot{\mathcal{R}}^i_F(H_\pi)\} = \{\top, a, b, c\}$. Now we compute the Moore-closure $\mathcal{M}(\{\top, a, b, c\}) = \{\top, a, b, c, d\}$ and obtain the least observationally complete domain (for f and π). It is worth noting that the object $\bot$ is not included in this domain, even if it is a maximal element of $f^{-1}(\downarrow\{c\})$. This is because we only need to consider the maximal elements in $f^{-1}(\downarrow\{b\}) \cup f^{-1}(\downarrow\{c\})$, since both b and c are generated by the same computation and object a in the observable π (see Example 3.2). □

4. Observational Completeness and Complete Shell

In this section we study the relationship between observational completeness and the standard notion of completeness, in particular between the least observationally complete domain and the complete shell.

It is immediate to show that if α is complete for F and $\alpha \supseteq \pi$, then α is observationally complete for F and π. In particular, α is observationally complete for F and α. We wonder whether:

a) every observationally complete domain for F and π is complete for F;

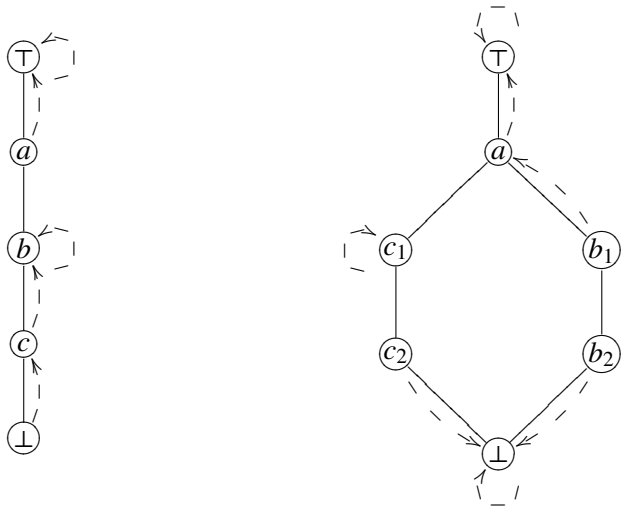
b) the least observationally complete domain for F and π is the complete shell of π for F.

With respect to the first question, note that if α is observationally complete for F and π, every $\beta \supseteq \alpha$ is still observationally complete. This does not hold for completeness: α may be complete for F, although some $\beta \supseteq \alpha$ may not. This is because in the observational completeness the observable properties remain fixed when we refine, while for standard completeness the notion of observational domain coincides with the computational domain.

Example 4.1. Consider the concrete domain $C = \{\top, a, b, c, \bot\}$ depicted in Fig. 4a. The domain $\alpha = \{\top, a, b\}$ is complete for the function depicted in the diagram, hence it is also observationally complete for $\pi = \{\top, a\}$. However, the domain $\{\top, a, b, c\}$ is observationally complete for π but it is not complete. □

Since completeness implies observational completeness, we may argue that the least observationally complete domain for F and π coincides with the complete shell of π. In the general case this is not true and the least observationally complete domain is more abstract than the complete shell. The next examples illustrate this point.

Example 4.2. Consider the concrete domain $C = \{\top, a, b, c, d, \bot\}$ depicted in Fig. 3. Assume $\pi = \{\top, a\}$. If we build the complete shell of π for f, in the first step we include the elements b and c, since they are the maximal $x \in C$ such that $f(x) \leq a$, and the element d since it is the meet of b and c. At the second step, we also include $\bot$, which is the greatest element $x \in C$ such that $f(x) \leq c$. Thus the complete shell of π for f is the whole concrete domain C. Note that, in each step, we consider all the elements generated in the previous steps, forgetting the observational domain π which started the process. However, from Example 3.3 we know that the domain $\alpha = C \setminus \{\bot\}$ is observationally complete, and thus it has the same precision of C when observing π, i.e. $\pi f^i(x) = \pi(\alpha f \alpha)^i(x)$ for



(a) Example 4.1 (b) Example 4.3

Figure 4: Counterexamples

every $x \in C$ and $i \in \mathbb{N}$. When $x \in \alpha$, the stronger property $f^i(x) = (\alpha f \alpha)^i(x)$ holds. When $x = \bot$, it is not true that $f^i(\bot) = (\alpha f \alpha)^i(\bot)$: for example it does not hold for $i = 1$. However, for each i, $\pi f^i(\bot) = a = \pi(\alpha f \alpha)^i(\bot)$, since α is observationally complete. □

Example 4.3. Consider the concrete domain $C = \{\top, a, b_1, b_2, c_1, c_2, \bot\}$ depicted in Fig. 4b. If $\pi = \{\top, a\}$, the complete shell is the entire domain C. However, c_2 is useless when observing π, since the least observationally complete domain is $C \setminus \{c_2\}$. □

4.1. The Case of Additive Functions

When all the functions in F are (completely) additive, the least observationally complete domain and the complete shell coincide. However, in order to prove this result, we need to give an alternative construction for the complete shell, more similar to the construction for the least observationally complete domain. We replace the standard refinement operator $\mathcal{R}_F : uco(C) \to uco(C)$ given in Section 2 with a new operator $\hat{\mathcal{R}}_F : \wp(C) \to \wp(C)$ simply obtained by removing the Moore-closure from the definition of $\mathcal{R}_F$. Therefore, we define

$$\hat{\mathcal{R}}_F(X) = \bigcup_{f \in F,\ a \in X} f^*(a) \ .$$

Note that $\hat{\mathcal{R}}_F(X)$ may not be an uco even if X is an uco, and that $\mathcal{R}_F(X) = \mathcal{M}\big(\hat{\mathcal{R}}_F(X)\big)$.

Lemma 4.1. For every set F of continuous functions and every $X \in \wp(C)$, we have

$$\mathcal{M}\big(\hat{\mathcal{R}}_F(\mathcal{M}(X))\big) = \mathcal{M}\big(\hat{\mathcal{R}}_F(X)\big) \ .$$

Proof:
It is immediate by monotonicity of $\hat{\mathcal{R}}_F$ and $\mathcal{M}(\_)$ that $\mathcal{M}\big(\hat{\mathcal{R}}_F(\mathcal{M}(X))\big) \supseteq \mathcal{M}\big(\hat{\mathcal{R}}_F(X)\big)$. For the other

direction, since $\mathcal{M}(\_)$ is an uco on $\wp(C)$, it is enough to prove that $\hat{\mathcal{R}}_F(\mathcal{M}(X)) \subseteq \mathcal{M}\left(\hat{\mathcal{R}}_F(X)\right)$. Given $y \in \hat{\mathcal{R}}_F(\mathcal{M}(X))$, we have $y \in f^*(a)$ and $a = \bigwedge_{i\in I} a_i$ where $f \in F$ and $\{a_i\}_{i\in I} \subseteq X$.

For each $i \in I$, consider the set $Y_i = f^*(a_i) \subseteq \hat{\mathcal{R}}_F(X)$. Since $f(y) \leq a \leq a_i$ and f is continuous, there exists $y_i \in Y_i$ such that $y_i \geq y$. We may find y_i as the least upper bound of a maximal chain in Y_i containing y, which exists by Hausdorff's maximality principle. It is enough to prove that $y = \bigwedge_{i\in I} y_i$. By definition of the y_i's, we have $y \leq \bigwedge_{i\in I} y_i$. Moreover, $f(\bigwedge_{i\in I} y_i) \leq f(y_i) \leq a_i$ hence $f(\bigwedge_{i\in I} y_i) \leq a$. Since y is a maximal element such that $f(y) \leq a$, this means that $y = \bigwedge_{i\in I} y_i$. □

Theorem 4.1. For every set F of continuous maps, the complete shell of π for F is given by

$$S_{\pi,F} = \mathcal{M}\left(\hat{\mathcal{R}}_F^{\omega}(\pi)\right) .$$

Proof:
We recall that the complete shell of π for F may be computed as:

$$S_{\pi,F} = \mathcal{M}(G^{\omega}(\{\top_C\})) ,$$

where $G : uco(C) \to uco(C)$ is the map

$$G(\alpha) = \mathcal{M}(\pi \cup \mathcal{R}_F(\alpha)) = \mathcal{M}\left(\pi \cup \hat{\mathcal{R}}_F(\alpha)\right) .$$

It is enough to prove that $\mathcal{M}\left(\hat{\mathcal{R}}_F^{\omega}(\pi)\right)$ is a subset of $S_{\pi,F}$ and a fixpoint of G. In order to prove $\mathcal{M}\left(\hat{\mathcal{R}}_F^{\omega}(\pi)\right) \subseteq S$ it is enough to show that $\hat{\mathcal{R}}_F^{i}(\pi) \subseteq G^{i+1}(\{\top_C\})$ for every $i < \omega$. The proof is by induction over i. For $i = 0$, we have $\hat{\mathcal{R}}_F^{0}(\pi) = \pi \subseteq G(\{\top_C\})$. If $i = j + 1$, $\hat{\mathcal{R}}_F^{i}(\pi) = \hat{\mathcal{R}}_F(\hat{\mathcal{R}}_F^{j}(\pi)) \subseteq \hat{\mathcal{R}}_F(G^{i}(\{\top_C\})) \subseteq G(G^{i}(\{\top_C\})) = G^{i+1}(\{\top_C\})$. Now we prove that $\mathcal{M}\left(\hat{\mathcal{R}}_F^{\omega}(\pi)\right)$ is a fixpoint of G. We have that

$$\begin{aligned} G\left(\mathcal{M}\left(\hat{\mathcal{R}}_F^{\omega}(\pi)\right)\right) &= \mathcal{M}\left(\pi \cup \hat{\mathcal{R}}_F\left(\mathcal{M}\left(\hat{\mathcal{R}}_F^{\omega}(\pi)\right)\right)\right) = \mathcal{M}\left(\pi \cup \mathcal{M}\left(\hat{\mathcal{R}}_F\left(\mathcal{M}\left(\hat{\mathcal{R}}_F^{\omega}(\pi)\right)\right)\right)\right) \\ &= \mathcal{M}\left(\pi \cup \mathcal{M}\left(\hat{\mathcal{R}}_F(\hat{\mathcal{R}}_F^{\omega}(\pi))\right)\right) = \mathcal{M}\left(\pi \cup \bigcup\{\hat{\mathcal{R}}_F^{i+1}(\pi) \mid i < \omega\}\right) = \mathcal{M}\left(\hat{\mathcal{R}}_F^{\omega}(\pi)\right) . \end{aligned}$$

This concludes the proof. □

This new construction, which is the key result to prove Theorem 4.2, is interesting in itself, since it sheds a new light on the construction of the complete shell. It shows that it is not necessary to compute the Moore-closure at each step of the refinement, but it suffices to compute it at the end.

We recall that a function $f : C \to C$ is (completely) additive if it preserves arbitrary least upper bounds, i.e., $f(\bigvee B) = \bigvee f(B)$ for any $B \subseteq C$.

Theorem 4.2. If F is a set of completely additive functions, the complete shell $S_{\pi,F}$ of π for F is the least observationally complete domain $\sigma_{\pi,F}$ for F and π.

Proof:
By Theorems 3.3 and 4.1, we have that

$$S_{\pi,F} = \mathcal{M}\left(\hat{\mathcal{R}}_F^{\omega}(\pi)\right) , \qquad \sigma_{\pi,F} = \mathcal{M}\left(\bigcup \dot{\mathcal{R}}_F^{\omega}(H_\pi)\right) .$$

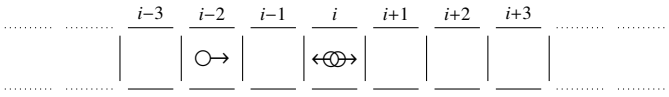
Note that, if f is completely additive, $f^*(y)$ is a singleton for any $y \in C$. Hence, if $H \in \wp(\wp(C))$ is a set of singletons, like H_π, we have that $\dot{\mathcal{R}}_F(H)$ is a set of singletons. Hence, $\dot{\mathcal{R}}^i_F(H_\pi)$ is a set of singletons for each $i < \omega$. This allows us to prove that $\hat{\mathcal{R}}^i_F(\pi) = \cup\dot{\mathcal{R}}^i_F(H_\pi)$ for every $i \leq \omega$. The proof is by induction on i. For $i = 0$, we have $\hat{\mathcal{R}}^0_F(\pi) = \pi = \cup H_\pi = \cup\dot{\mathcal{R}}^0_F(H_\pi)$. If $i = j + 1$, we have that $\hat{\mathcal{R}}^i_F(\pi) = \hat{\mathcal{R}}_F(\hat{\mathcal{R}}^j_F(\pi)) = \hat{\mathcal{R}}_F(\bigcup \dot{\mathcal{R}}^j_F(H_\pi))$ Since $\dot{\mathcal{R}}^j_F(H_\pi)$ is a set of singletons, we have that $\hat{\mathcal{R}}_F(\bigcup \dot{\mathcal{R}}^j_F(H_\pi)) = \bigcup \dot{\mathcal{R}}_F(\dot{\mathcal{R}}^j_F(H_\pi)) = \cup\dot{\mathcal{R}}^i_F(H_\pi)$. The case $i = \omega$ is trivial. This concludes the proof. □

It is worth noting that, even if F is a set of additive functions, this theorem does not imply that observational completeness and completeness are the same thing: in Example 4.1 the function f is additive, yet there is an observationally complete domain which is not complete.

Example 4.4. (Temporal abstract interpretation)
We may now come back to the question in Section 2.2. Giacobazzi and Ranzato [13] show that, even considering the two connectives $\cup$ and $\curvearrowleft$ alone, the complete shell is the same as for the entire logic. Since these operators are additive, their complete shell corresponds to the least observationally complete domain. Hence, there is no intermediate refinement of states which does not loose precision w.r.t. the trace semantics when observing properties of states only.

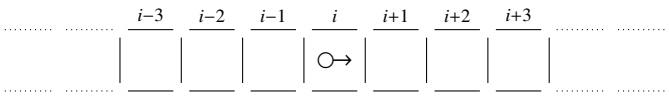
5. An Example

In this section, we present a detailed example of static analysis of cellular automata (CA), and we show a case where the least observationally complete domain differs from the complete shell. A cellular automaton consists of a grid of cells, and each cell can be in a finite number of states. In our example, we consider one-dimensional CA, where each cell can be in four different states: $\perp$ (empty), $\bigcirc\!\!\rightarrow$ (right-directed), $\leftarrow\!\!\bigcirc$ (left-directed) and $\leftarrow\!\!\bigcirc\!\!\rightarrow$ (collision). Intuitively, the state $\bigcirc\!\!\rightarrow$ can be thought of as a right-directed entity in a cell, and the state $\leftarrow\!\!\bigcirc\!\!\rightarrow$ as a collision of two (right-directed and left-directed) entities. A configuration (at time t) assigns a state to each cell. For instance:

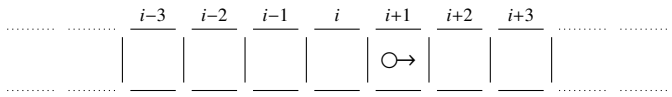


is a configuration where the cell in position $i - 2$ is in the state $\bigcirc\!\!\rightarrow$ and the cell i is in the state $\leftarrow\!\!\bigcirc\!\!\rightarrow$. All the other cells are in the empty state. To improve readability, we prefer to draw them empty instead of using the symbol $\perp$. More formally, if States $= \{\perp, \leftarrow\!\!\bigcirc, \bigcirc\!\!\rightarrow, \leftarrow\!\!\bigcirc\!\!\rightarrow\}$ is the set of states, a configuration is a map $\mathbb{Z} \to$ States.

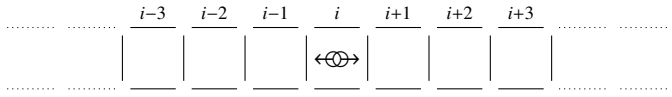
A cellular automaton evolves from time t to time $t + 1$ according to some fixed rules. In our example, we consider two rules: the *moving rule* and the *collision rule*. The moving rule applies when the state of a cell is either $\bigcirc\!\!\rightarrow$ or $\leftarrow\!\!\bigcirc$. Starting from the configuration at time t:



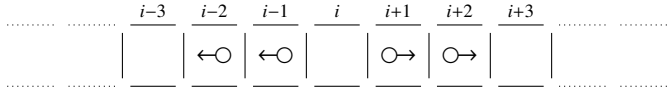
the configuration at time $t + 1$ will be:



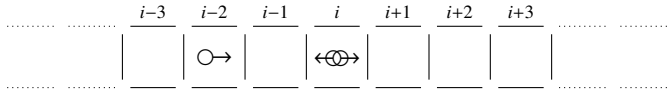
Symmetrically for $\leftarrow\bigcirc$, where the affected cell is $i - 1$. The collision rule applies when the state of a cell is $\leftrightarrow\!\!\bigcirc\!\!\rightarrow$. Starting from a configuration at time t:



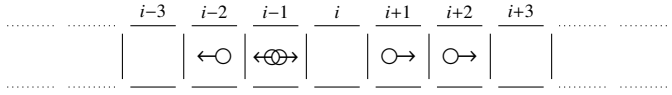
the configuration at time $t + 1$ is:



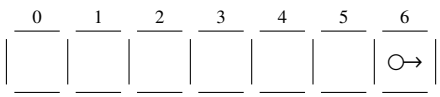
Intuitively, the collision generates two new entities. The effects of the two rules sum up. For instance, from the configuration at time t:



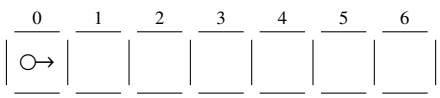
we get at time $t + 1$:



Often, cellular automata only have a finite number n of cells. In this case, it is generally assumed that the grid space is circular and a configuration becomes a map $\mathbb{Z}_n \to$ States, where $\mathbb{Z}_n$ is the set of integers modulo n. We denote the set of all the configurations by Conf_n. For instance, in a cellular automaton with 7 cells and configuration at time t:



we have, at time $t + 1$, the configuration:



The rules are typically formalized by a function which computes the new state of a cell in terms of the current state of the cell and of some of its neighbors. The function for updating the state of a cell is the same for each cell, does not change over time, and is applied to the whole grid simultaneously.

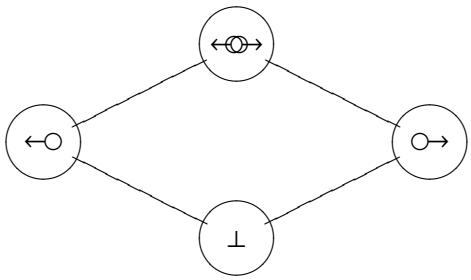


Figure 5: Ordering on States

The updating function $f : \mathrm{Conf}_n \to \mathrm{Conf}_n$ corresponding to our rules is the following:

$$f(c)(i) = \begin{cases} \leftrightarrow\!\bigcirc\!\!\rightarrow & \text{if} \begin{cases} (c(i-1) \in \{\bigcirc\!\rightarrow, \leftrightarrow\!\bigcirc\!\!\rightarrow\} \; OR \; c(i-2) = \leftrightarrow\!\bigcirc\!\!\rightarrow) \\ AND \; (c(i+1) \in \{\leftarrow\!\bigcirc, \leftrightarrow\!\bigcirc\!\!\rightarrow\} \; OR \; c(i+2) = \leftrightarrow\!\bigcirc\!\!\rightarrow) \end{cases} \\ \bigcirc\!\rightarrow & \text{otherwise, if } c(i-1) \in \{\bigcirc\!\rightarrow, \leftrightarrow\!\bigcirc\!\!\rightarrow\} \; OR \; c(i-2) = \leftrightarrow\!\bigcirc\!\!\rightarrow \\ \leftarrow\!\bigcirc & \text{otherwise, if } c(i+1) \in \{\leftarrow\!\bigcirc, \leftrightarrow\!\bigcirc\!\!\rightarrow\} \; OR \; c(i+2) = \leftrightarrow\!\bigcirc\!\!\rightarrow \\ \bot & \text{otherwise,} \end{cases}$$

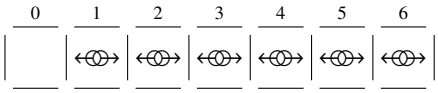
where $c \in \mathrm{Conf}_n$ and $i \in \mathbb{Z}_n$ is a position.

5.1. Static Analysis of Cellular Automata

The choice of the concrete domain depends on the properties we want to analyze. Here we are interested in properties of the kind: "the cell i is empty", "the cell i does not contain a right-directed entity", "every cell whose index is odd does not contain a left-directed entity". We endow the set States with the ordering $\leq_s$ depicted in Figure 5, and we lift $\leq_s$ pointwise to the ordering $\leq_c$ on configurations. This allows us to formalize the above properties. For instance, the property "starting from the configuration c, the cell 0 will always be empty", is formalized by

$$\forall j < \omega, f^j(c) \leq_c c' \ ,$$

where c' is the configuration:



Hence, in our settings, Conf_n plays the role of the concrete domain.

We compute the least observationally complete domain and the complete shell, in the case of a simple CA with 3 cells. Since f is continuous w.r.t. $\leq_c$, both domains exist and we show a case where they are different (due to the fact that f is not additive).

To simplify the notation, a configuration $c \in \mathrm{Conf}_3$ will be denoted by the triple $\langle c(0), c(1), c(2)\rangle$. We consider the observable domain $\pi = \{\langle \leftrightarrow\!\bigcirc\!\!\rightarrow, \leftrightarrow\!\bigcirc\!\!\rightarrow, \leftrightarrow\!\bigcirc\!\!\rightarrow\rangle, \langle \leftrightarrow\!\bigcirc\!\!\rightarrow, \bigcirc\!\rightarrow, \bigcirc\!\rightarrow\rangle\}$, which corresponds to the property that the cells in position 1 and 2 can be only in the states $\bot$ or $\bigcirc\!\rightarrow$, i.e., they cannot contain any left-directed entity.

In order to carry on the constructions given in Theorems 3.3 and 4.1, we need to compute f^*. Although would be theoretically possible to use the definition of f^* as an algorithm, this would require

a long search in the space of all the configurations. However, it is possible to compute f^* with a simpler method. We first define f^* on the configurations where two cells are in the state $\leftarrow\bigcirc\rightarrow$. The state of the third cell will enforce some constraints on the configurations c' such that $f(c') \leq_c c$. For example, given $c = \langle \leftarrow\bigcirc\rightarrow, \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow \rangle$, the state $\bigcirc\rightarrow$ forbids the state $\leftarrow\bigcirc$ in position 0 at the previous time. This may be formalized as follows:

$$f^*(\langle s, \leftarrow\bigcirc\rightarrow, \leftarrow\bigcirc\rightarrow\rangle) = \begin{cases} \{\langle \leftarrow\bigcirc\rightarrow, \leftarrow\bigcirc\rightarrow, \leftarrow\bigcirc\rightarrow\rangle\} & \text{if } s = \leftarrow\bigcirc\rightarrow, \\ \{\langle \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle, \langle \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow, \leftarrow\bigcirc\rangle\} & \text{if } s = \bigcirc\rightarrow, \\ \{\langle \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow, \leftarrow\bigcirc\rangle, \langle \leftarrow\bigcirc\rightarrow, \leftarrow\bigcirc, \leftarrow\bigcirc\rangle\} & \text{if } s = \leftarrow\bigcirc, \\ \{\langle \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow, \leftarrow\bigcirc\rangle\} & \text{if } s = \bot. \end{cases}$$

The definitions for the cases when the state s is in the positions 1 or 2 are the obvious permutations of the above definition. If c is a generic configuration, the constraints which originate by the different cells of c sum up, hence we may define

$$f^*(c) = \max\left(f^*(\langle c(0), \leftarrow\bigcirc\rightarrow, \leftarrow\bigcirc\rightarrow\rangle) \wedge_c f^*(\langle \leftarrow\bigcirc\rightarrow, c(1), \leftarrow\bigcirc\rightarrow\rangle) \wedge_c f^*(\langle \leftarrow\bigcirc\rightarrow, \leftarrow\bigcirc\rightarrow, c(2)\rangle)\right) \ ,$$

where $\wedge_c$ is the greatest lower bound of configurations according to the ordering $\leq_c$, applied pointwise to sets of configurations. Finally, $f^*(S) = \max \bigcup_{c\in S} f^*(c)$.

For instance, given the configuration $c = \langle \bigcirc\rightarrow, \leftarrow\bigcirc, \bigcirc\rightarrow \rangle$, we have

$$\begin{aligned} f^*(\langle \bigcirc\rightarrow, \leftarrow\bigcirc\rightarrow, \leftarrow\bigcirc\rightarrow\rangle) &= \{\langle \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle, \langle \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow, \leftarrow\bigcirc\rangle\} \\ f^*(\langle \leftarrow\bigcirc\rightarrow, \leftarrow\bigcirc, \leftarrow\bigcirc\rightarrow\rangle) &= \{\langle \leftarrow\bigcirc, \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow\rangle, \langle \leftarrow\bigcirc, \leftarrow\bigcirc\rightarrow, \leftarrow\bigcirc\rangle\} \\ f^*(\langle \leftarrow\bigcirc\rightarrow, \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow\rangle) &= \{\langle \bigcirc\rightarrow, \bigcirc\rightarrow, \leftarrow\bigcirc\rightarrow\rangle, \langle \bigcirc\rightarrow, \leftarrow\bigcirc, \leftarrow\bigcirc\rightarrow\rangle\} \end{aligned}$$

hence $f^*(c) = \{\langle \bot, \bigcirc\rightarrow, \leftarrow\bigcirc\rangle, \langle \bot, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}$. In the same way, we have $f^*(\{\langle \bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}) = \{\langle \bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}$ and therefore

$$\begin{aligned} f^*(\{\langle \bigcirc\rightarrow, \leftarrow\bigcirc, \bigcirc\rightarrow\rangle, \langle \bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}) &= \max\{\langle \bot, \bigcirc\rightarrow, \leftarrow\bigcirc\rangle, \langle \bot, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle, \langle \bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\} \\ &= \{\langle \bot, \bigcirc\rightarrow, \leftarrow\bigcirc\rangle, \langle \bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\} \ . \end{aligned}$$

Now, using f^*, we compute the least observationally complete domain for f and π, defined as

$$\mathcal{M}\Big(\bigcup\{S \mid i < \omega, S \in \dot{\mathcal{R}}^i_{\{f\}}(\{\{\langle \leftarrow\bigcirc\rightarrow, \leftarrow\bigcirc\rightarrow, \leftarrow\bigcirc\rightarrow\rangle\}, \{\langle \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}\})\}\Big)$$

We now show the iterates of $\dot{\mathcal{R}}_{\{f\}}$. For the sake of conciseness, we will omit the configuration $\top_c = \langle \leftarrow\bigcirc\rightarrow, \leftarrow\bigcirc\rightarrow, \leftarrow\bigcirc\rightarrow\rangle$ since $f^*(\top_c) = \{\top_c\}$.

$$\begin{aligned} \dot{\mathcal{R}}^0_{\{f\}}(\{\{\langle \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}\}) &= \{\{\langle \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}\} \\ \dot{\mathcal{R}}^1_{\{f\}}(\{\{\langle \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}\}) &= \{\{\langle \bigcirc\rightarrow, \leftarrow\bigcirc, \bigcirc\rightarrow\rangle, \langle \bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\} \\ \dot{\mathcal{R}}^2_{\{f\}}(\{\{\langle \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}\}) &= \{\{\langle \bot, \bigcirc\rightarrow, \leftarrow\bigcirc\rangle, \langle \bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}) \\ \dot{\mathcal{R}}^3_{\{f\}}(\{\{\langle \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}\}) &= \{\{\langle \leftarrow\bigcirc, \bot, \bot\rangle, \langle \bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}) \\ \dot{\mathcal{R}}^4_{\{f\}}(\{\{\langle \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}\}) &= \{\{\langle \bot, \leftarrow\bigcirc, \bot\rangle, \langle \bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}) \\ \dot{\mathcal{R}}^5_{\{f\}}(\{\{\langle \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}\}) &= \{\{\langle \bot, \bot, \leftarrow\bigcirc\rangle, \langle \bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}) \\ \dot{\mathcal{R}}^6_{\{f\}}(\{\{\langle \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}\}) &= \dot{\mathcal{R}}^3_{\{f\}}(\{\{\langle \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}\}) \end{aligned}$$

Since $\dot{\mathcal{R}}^6_{\{f\}}(\{\{\langle \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}\}) = \dot{\mathcal{R}}^3_{\{f\}}(\{\{\langle \leftarrow\bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}\})$, we cannot generate any new point and we may stop the computation. It follows that:

$$\begin{aligned}\sigma_{\pi,\{f\}} = \mathcal{M}\{&\langle \leftarrow\!\bigcirc\!\rightarrow, \leftarrow\!\bigcirc\!\rightarrow, \leftarrow\!\bigcirc\!\rightarrow\rangle\,\langle \leftarrow\!\bigcirc\!\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle, \langle \bigcirc\rightarrow, \leftarrow\bigcirc, \bigcirc\rightarrow\rangle, \langle \bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle,\\ &\langle \bot, \bigcirc\rightarrow, \leftarrow\bigcirc\rangle, \langle \leftarrow\bigcirc, \bot, \bot\rangle, \langle \bot, \leftarrow\bigcirc, \bot\rangle, \langle \bot, \bot, \leftarrow\bigcirc\rangle\}\\ = \{&\langle \leftarrow\!\bigcirc\!\rightarrow, \leftarrow\!\bigcirc\!\rightarrow, \leftarrow\!\bigcirc\!\rightarrow\rangle\,\langle \leftarrow\!\bigcirc\!\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle, \langle \bigcirc\rightarrow, \leftarrow\bigcirc, \bigcirc\rightarrow\rangle, \langle \bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle,\\ &\langle \bot, \bigcirc\rightarrow, \leftarrow\bigcirc\rangle, \langle \leftarrow\bigcirc, \bot, \bot\rangle, \langle \bot, \leftarrow\bigcirc, \bot\rangle, \langle \bot, \bot, \leftarrow\bigcirc\rangle,\\ &\langle \bigcirc\rightarrow, \bot, \bigcirc\rightarrow\rangle, \langle \bot, \bigcirc\rightarrow, \bot\rangle, \langle \bot, \bot, \bot\rangle\}\ .\end{aligned}$$

We now show that the complete shell of π for $\{f\}$ is strictly more concrete than the least observationally complete domain.

$$\begin{aligned}\hat{\mathcal{R}}^0_{\{f\}}(\pi) &= \pi\\ \hat{\mathcal{R}}^1_{\{f\}}(\pi) &= \{\langle \leftarrow\!\bigcirc\!\rightarrow, \leftarrow\!\bigcirc\!\rightarrow, \leftarrow\!\bigcirc\!\rightarrow\rangle, \langle \bigcirc\rightarrow, \leftarrow\bigcirc, \bigcirc\rightarrow\rangle, \langle \bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}\\ \hat{\mathcal{R}}^2_{\{f\}}(\pi) &= \{\langle \leftarrow\!\bigcirc\!\rightarrow, \leftarrow\!\bigcirc\!\rightarrow, \leftarrow\!\bigcirc\!\rightarrow\rangle, \langle \bot, \bigcirc\rightarrow, \leftarrow\bigcirc\rangle, \langle \bot, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle, \langle \bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}\end{aligned}$$

Note that, the object $\langle \bot, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle$ does not appear in the least observationally complete domain. In fact, it is generated by $f^*(\{\langle \bigcirc\rightarrow, \leftarrow\bigcirc, \bigcirc\rightarrow\rangle\} = \{\langle \bot, \bigcirc\rightarrow, \leftarrow\bigcirc\rangle, \langle \bot, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}$. But this object is useless, since $f^*(\{\langle \bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle\}) = \langle \bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle$, but $\langle \bot, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle \leq_C \langle \bigcirc\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle$ and they are generated by the same object $\langle \leftarrow\!\bigcirc\!\rightarrow, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle$ at the same iteration. This result agrees with our intuition. In fact, since we are interested in observing the property that the cells 1 and 2 cannot contain the entity $\leftarrow\bigcirc$, it follows that we are not interested in distinguish whether the cell 0 contains an entity $\bigcirc\rightarrow$, as long as it cannot cause collisions, since it does not affect the observable property in any future time.

For the sake of completeness, the complete shell is given by:

$$S_{\pi,\{f\}} = \sigma_{\pi,\{f\}} \cup \{\langle \bot, \bigcirc\rightarrow, \bigcirc\rightarrow\rangle, \langle \bigcirc\rightarrow, \bot, \bot\rangle, \langle \bigcirc\rightarrow, \bigcirc\rightarrow, \bot\rangle, \langle \bot, \bot, \bigcirc\rightarrow\rangle\}\ .$$

6. Observational Equivalence

The least observationally complete domain is generally quite "big", i.e., its size is comparable with that of the concrete domain. Most of the time, if C is infinite, the least observationally complete domain is infinite, too. It turns out that completeness is a tool which is well suited for comparing different semantics of a dynamic system (see, for example, [2, 13]) but not so much for static analysis.

Often, in the latter case, a computational domain α is already known, but it may be computationally too expensive. Therefore, we look for a different computational domain which is simpler, yet more precise than α. In other words, we want to compare the precision of abstract domains with reference to the computational domain α, instead of the concrete domain. Formally, we look for the least abstract domain β such that $\beta \preceq^{\pi}_{F} \alpha$. In the general case, this problem has no solution.

Example 6.1. Consider the concrete domain $C = \{\top, a, b, \bot\}$ depicted in Figure 6, the observable $\pi = \{\top, a\}$ and the computational domain $\alpha = \{\top, a, \bot\}$. It is immediate to see that both α and $\beta = \{\top, a, b\}$ are more precise then α when observing π. However, their intersection $\alpha \cap \beta = \{\top, a\}$ is strictly less precise than α. Therefore, the problem has no solution in this case. Anyway, one may prefer β to α, since β is more precise than α in observing π (while the opposite is not true). □

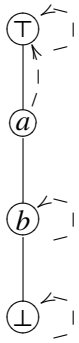


Figure 6: Counterexample

It is worth noting that, in Example 6.1, the function f is (completely) additive. This prevents us from finding a suitable condition on f which guarantees the existence of such a least domain.

Given an observable π and a domain α, the previous example shows that, in general, there are several minimal domains β such that $\beta \preceq_F^\pi \alpha$. It is possible to choose among those β's on the base of several criteria, such as precision or size. A sensible choice may be to restrict the search space to the abstractions of α. We look for the least $\beta \subseteq \alpha$ (if it exists) such that $\beta \preceq_F^\pi \alpha$. This domain exists when α is continuous, and may be obtained using the definition of least observationally complete domain, by simply considering α as the concrete domain.

Theorem 6.1. Given a continuous[2] domain α, an observable domain $\pi \subseteq \alpha$ and a set of continuous functions F, let $F_\alpha = \{f^\alpha : \alpha \to \alpha \mid f \in F\}$. The least abstract domain $\beta \subseteq \alpha$ such that $\beta \preceq_F^\pi \alpha$ is the least observationally complete domain σ_{π,F_α}.

Proof:
Since α is continuous, then each f^α is continuous too, for any $f \in F$. Thus, we may apply Theorem 3.1 by considering α as the concrete domain and F_α as the set of functions. As a result, σ_{π,F_α} is the least abstract domain $\beta \subseteq \alpha$ such that $\beta \preceq_{F_\alpha}^\pi \alpha$. We first show that $\sigma_{\pi,F_\alpha} \preceq_F^\pi \alpha$. For each computation $\xi = \langle f_1, \ldots, f_n\rangle$ in F and $c \in C$, we have that $\pi\sigma_{\pi,F_\alpha} f_1 \ldots \sigma_{\pi,F_\alpha} f_n \sigma_{\pi,F_\alpha}(c) = \pi\sigma_{\pi,F_\alpha}\alpha f_1 \ldots \sigma_{\pi,F_\alpha}\alpha f_n \sigma_{\pi,F_\alpha}\alpha(c)$ since $\sigma_{\pi,F_\alpha} \subseteq \alpha$. Moreover, $\pi\sigma_{\pi,F_\alpha}\alpha f_1 \ldots \sigma_{\pi,F_\alpha}\alpha f_n \sigma_{\pi,F_\alpha}\alpha(c) \leq \pi\alpha f_1 \ldots \alpha f_n\alpha(c)$ since σ_{π,F_α} is observationally complete for F_α and π.

We now show that for any $\beta \subseteq \alpha$ such that $\beta \preceq_F^\pi \alpha$, then it holds that $\sigma_{\pi,F_\alpha} \subseteq \beta$. We prove that $\beta \preceq_{F_\alpha}^\pi \alpha$. For each computation $\xi = \langle \alpha f_1, \ldots, \alpha f_n\rangle$ in F_α and $c \in C$, we have that $\pi\beta(\alpha f_1)\ldots\beta(\alpha f_n)\beta(c) = \pi\beta f_1 \ldots \beta f_n\beta(c)$ since $\beta \subseteq \alpha$. Moreover, $\pi\beta f_1 \ldots \beta f_n\beta(c) \leq \pi\alpha f_1 \ldots \alpha f_n\alpha(c)$ since $\beta \preceq_F^\pi \alpha$. This shows that $\beta \preceq_{F_\alpha}^\pi \alpha$. Since σ_{π,F_α} is, by definition, the least abstraction of α which is more precise than α for F_α and π, it immediately follows that $\sigma_{\pi,F_\alpha} \subseteq \beta$. □

Actually, the domain σ_{π,F_α} enjoys a stronger characterization, since it is the least domain which is exactly as precise as α, among all the domains in $uco(C)$.

Definition 6.1. (Observational Equivalence)
Two domains α and β are *observationally equivalent* for F and π if $\pi \circ \xi^\alpha = \pi \circ \xi^\beta$ for every computation ξ in F, i.e., if $\alpha \preceq_F^\pi \beta$ and $\beta \preceq_F^\pi \alpha$.

[2] An abstract domain α is continuous when the function α is continuous on C.

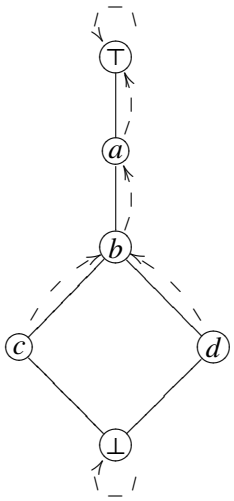


Figure 7: Observational equivalence

As a special case, it holds that α is observationally equivalent to the concrete domain iff it is observationally complete.

Theorem 6.2. In the hypothesis of the previous Theorem, σ_{π,F_α} is the least domain in $uco(C)$ which is observationally equivalent to α.

Proof:
Clearly σ_{π,F_α} is observationally equivalent to α for F and π. To show that it is the least one in $uco(C)$, assume that β is observationally equivalent to α, but β is incomparable with σ_{π,F_α}, i.e., there exists $s \in \sigma_{\pi,F_\alpha} \setminus \beta$ such that $s \in \max\{x \in \alpha \mid \xi(x) \leq a\}$ for some computation ξ in F_α and $a \in \pi$. Therefore, for each $s' > s$ it holds that $\pi\xi_\alpha(s') > \pi\xi_\alpha(s)$. Let $\bar{\xi}$ the corresponding computation in F. Since $s \notin \beta$, then $\beta(s) > s$, hence $\pi\bar{\xi}_\alpha(\beta(s)) = \pi\xi(\alpha(\beta(s)) > \pi\bar{\xi}_\alpha(s)$. On the converse, $\pi\bar{\xi}_\beta(\beta(s)) = \pi\bar{\xi}_\beta(s) = \pi\bar{\xi}_\alpha(s)$ since $\beta \preceq^\pi_F \alpha$. Therefore $\alpha \not\preceq^\pi_F \beta$, which is a contradiction. □

As a consequence, if β is observationally equivalent to α, it follows that $\sigma_{\pi,F_\alpha} = \sigma_{\pi,F_\beta}$.

Coming back to the original problem, which is to find out minimal domains β such that $\beta \preceq^F_\pi \alpha$, it is important to observe that restricting the focus on subsets of α may not be a good choice. More precise or smaller domains may be found, which are not abstractions of α. The following example shows a case where a domain β exists which is more precise and smaller in size of σ_{π,F_α}. A more complex example, in the field of sharing analysis of logic programs, may be found in [21].

Example 6.2. Consider the concrete domain $C = \{\top, a, b, c, d, \bot\}$ depicted in Figure 7, the observable $\pi = \{\top, a\}$ and the computational domain $\alpha = \{\top, a, c, d, \bot\}$. The least domain which is observationally equivalent to α for f and π is α itself, but $\{\top, a, b\}$ is strictly more precise than α (when observing π) and smaller (in terms of number of points) than α.

7. Conclusions and Related Work

Different kinds of completeness have been proposed in the literature. The first notion of completeness appears in Cousot and Cousot [5]. In the same paper, the notion of *fixpoint completeness* is formalized. A domain α is fixpoint complete for a function f when it preserves the least fixpoint of f, in formulas $\alpha(\mathrm{lfp}\, f) = \mathrm{lfp}(\alpha \circ f \circ \alpha)$. Cousot and Cousot have shown that complete domains are also fixpoint complete. A detailed study on completeness and fixpoint completeness can be found in Giacobazzi et al. [15], where the authors solve the problem of synthesizing complete abstract domains.

Cousot and Cousot [4] introduced a different notion of completeness called *exactness*. The same notion has been renamed as *forward completeness* (F-completeness) by Giacobazzi and Quintarelli [12] who apply the completeness results on model checking. Moreover, to distinguish between standard completeness and F-completeness, Giacobazzi and Quintarelli renamed the former as *backward completeness* (B-completeness). A domain α is F-complete for a function f when $f \circ \alpha = \alpha \circ f \circ \alpha$. Intuitively, this means that the result of any abstract computation coincides with the result of the corresponding concrete one, when the starting object is an abstract object.

Observational completeness differs from all the previous notions (B-completeness, fixpoint completeness, F-completeness). The main point is that we have two concepts of observational and computational domain and, most importantly, the observational domain is kept fixed when refining. We believe that, in any static analysis or semantics definition, the observable property does not change when looking for better domains. On the contrary, B-completeness is self-referential, since the observational domain changes when refining the domain. More precisely, given a domain π, the complete shell of π for f is the least abstract domain β containing π which is observationally complete for β (and thus it is observationally complete for π). Moreover, the self-referentiality of completeness yields some counter-intuitive behaviors. For instance, if α is complete and β contains α, it may well happen that β is not complete even if, according to our intuition, β is "richer" than α. This does not happen for observational completeness, where supersets of observationally complete domains are still observationally complete (see Example 4.1).

The notion of F-completeness does not fix any observable property. This kind of completeness is useful when we are interested in a subset of the concrete domain closed for the application of any state-transition function.

Finally, fixpoint completeness does not take into consideration intermediates steps during the abstract computation. In fact, it is only required that the abstract least fixpoint (computed on the abstract domain) coincides with the abstraction of the concrete least fixpoint. Giacobazzi et al. [15] show that, even under strong hypotheses, the existence of the least fixpoint complete domain containing π cannot be ensured. They show that, even if the concrete domain is a complete Boolean algebra or a finite chain, and the concrete function f is both additive and co-additive, the least fixpoint complete domain containing π does not necessarily exist. The counterexamples suggest that finding reasonable conditions for the existence of least fixpoint complete domains is not viable.

Relative Completeness

In [15] a different notion of completeness is introduced. Given complete lattices C and D and two ucos $\rho \in uco(C)$ and $\eta \in uco(D)$, the pair $\langle \rho, \eta \rangle$ is complete for the monotone map $g : C \to D$ when $\eta \circ g = \eta \circ g \circ \rho$. The pair $\langle \rho, \eta \rangle$ is complete for a set G of monotonic maps from C to D when it is complete for all $g \in G$. In the following, we call *relative completeness* this variant of completeness. Relative completeness has been recently used for modeling secrecy in language-based security problems, in particular (abstract) non-interference [11].

Observational completeness is a particular case of relative completeness. Given a set $F \subseteq C \to C$ of monotonic maps and $\pi \in uco(C)$, it turns out that $\alpha \in uco(C)$ is observationally complete for F and π iff $\langle \alpha, \pi \rangle$ is complete for the set Ξ_F of all the computations built from F. Actually, the latter means that $\pi \circ \xi \circ \alpha = \pi \circ \xi$ for each computation ξ, while observational completeness means that $\pi \circ \xi_\alpha = \pi \circ \xi$. Even if the two notions may seem different, it is easy to prove that they are actually the same.

Giacobazzi et al. [15] also defines an operation on abstract domains called *relative complete shell*. The relative complete shell of $\rho \in uco(C)$ w.r.t. $\eta \in uco(D)$, denoted by $\mathcal{S}^\eta_G(\rho)$, is the most abstract domain α such that $\alpha \supseteq \rho$ and $\langle \alpha, \eta \rangle$ is complete for G. By the previous correspondence of observational completeness with relative completeness, it turns out that $\sigma_{\pi,F}$ is $\mathcal{S}^\pi_{\Xi_F}(\{\top_C\})$, i.e., the most abstract domain α such that $\langle \alpha, \pi \rangle$ is complete for Ξ_F. However, [15] does not further develop the special case when the set F of functions is closed by composition, as it happens in observational completeness. Therefore, they do not prove any other property, neither the constructive characterization in Theorem 3.3, nor the fixpoint preservation in Theorem 3.2, and they do not investigate on the relationship between the least observationally complete domain and the complete shell.

Quotient of Abstract Interpretation

Another construction related to observational completeness has been introduced by Cortesi and Filè in [3]. In that paper, the authors define a transformation of abstract domains called *quotient*. Given an observable domain $\pi \subseteq \alpha$ and a function $f : \alpha \to \alpha$, they define an equivalence relation $r_\pi \subseteq \alpha \times \alpha$ as follows:

$$\langle a_1, a_2 \rangle \in r_\pi \iff \forall i < \omega . \pi(f^i(a_1)) = \pi(f^i(a_2)) \ .$$

Roughly speaking, $\langle a_1, a_2 \rangle \in r_\pi$ when a_1 and a_2 are equivalent w.r.t. the observable π. Then quotient $Q_\pi(\alpha)$ is the subset of α composed of all the lubs of all equivalence classes in r_π. If $[a]$ is the equivalence class of a, then $Q_\pi(\alpha) = \{\vee[a] \mid a \in \alpha\}$. The paper shows that if r_π is additive, then $Q_\pi(\alpha)$ is an uco and $\pi \subseteq Q_\pi(\alpha) \subseteq \alpha$.

Later, [14] shows that $Q_\pi(\alpha)$ is $\mathcal{S}^\pi_{\{f^i \mid i<\omega\}}(\{\top_\alpha\})$, the relative complete shell of π w.r.t. α. According to what we said before,s $Q_\pi(\alpha) = \sigma_{\pi,\{f\}}$, the least observationally complete domain for π and f (where α is the concrete domain).

Other Definitions of Completeness

Other notions of completeness have been proposed for dealing with logic (see, for instance, Cousot and Cousot [8], Schmidt [20], Dams et al. [10]). In general, completeness problems on fragments of temporal logic are considered (covering, preservation, strong preservation). All these notions are very different from the other forms of completeness, since they consider only fixed logical operators, and, in general, one is not interested in least fixpoint preservation.

Observational completeness naturally arises once we fix the preorder $\preceq$ on domains, which formalizes the intuitive notion of precision. The novelty with respect to the standard treatment of completeness is that we have two orderings on ucos: standard inclusion $\subseteq$ and precision $\preceq$. The latter is used to define what an observationally complete domain is, while the former selects, among observationally complete domains, the preferred one. In the complete shell construction the two orderings coincide. In principle, we could change the standard inclusion ordering, obtaining a different notion of "least" observationally complete domain. For instance, we could compare two abstract domains on the base of their cardinality or of a suitable notion of "complexity" of their abstract objects.

References

[1] Birkhoff, G.: *Lattice Theory*, vol. XXV of *AMS Colloquium Publications*, third edition, American Mathematical Society, 1967.

[2] Comini, M., Levi, G., Meo, M. C.: A Theory of Observables for Logic Programs, *Information and Computation*, **169**(1), 2001, 23–80, ISSN 0890-5401.

[3] Cortesi, A., Filé, G., Winsborough, W. W.: The Quotient of an Abstract Interpretation, *Theoretical Computer Science*, **202**(1–2), July 1998, 163–192, ISSN 0304-3975.

[4] Cousot, P.: Types as Abstract Interpretations, invited paper, in: *POPL '97: Proceedings of the 24th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, ACM Press, New York, NY, USA, January 1997, ISBN 0-89791-853-3, 316–331.

[5] Cousot, P., Cousot, R.: Systematic Design of Program Analysis Frameworks, in: *POPL '79: Proceedings of the 6th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, ACM Press, New York, NY, USA, January 1979, 269–282.

[6] Cousot, P., Cousot, R.: Abstract Interpretation and Applications to Logic Programs, *The Journal of Logic Programming*, **13**(2–3), July 1992, 103–179, ISSN 0743-1066.

[7] Cousot, P., Cousot, R.: Higher-Order Abstract Interpretation (and Application to Comportment Analysis Generalizing Strictness, Termination, Projection and PER Analysis of Functional Languages), *Proceedings of the 1994 International Conference on Computer Languages*, IEEE Computer Society Press, Los Alamitos, CA, USA, May 1994, ISBN 0-8186-5640-X, Invited paper.

[8] Cousot, P., Cousot, R.: Temporal Abstract Interpretation, in: *POPL '00: Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, ACM Press, New York, NY, USA, January 2000, ISBN 1-58113-125-9, 12–25.

[9] Dalla Preda, M., Giacobazzi, R.: Semantic-based Code Obfuscation by Abstract Interpretation, *Journal of Computer Security*, **17**(6), 2009, 855–908, ISSN 0926-227X.

[10] Dams, D., Gerth, R., Grumberg, O.: Abstract interpretation of reactive systems, *ACM Transactions on Programming Languages and Systems*, **19**(2), 1997, 253–291, ISSN 0164-0925.

[11] Giacobazzi, R., Mastroeni, I.: Adjoining classified and unclassified information by abstract interpretation, *Journal of Computer Security*. To appear.

[12] Giacobazzi, R., Quintarelli, E.: Incompleteness, Counterexamples, and Refinements in Abstract Model-Checking, in: *Static Analysis, 8th International Symposium, SAS 2001 Paris, France, July 16–18, 2001 Proceedings* (P. Cousot, Ed.), vol. 2126 of *Lecture Notes in Computer Science*, Springer, Berlin Heidelberg, 2001, ISBN 978-3-540-42314-0, 356–373.

[13] Giacobazzi, R., Ranzato, F.: Incompleteness of states w.r.t. traces in model checking, *Information and Computation*, **204**(3), 2006, 376–407, ISSN 0890-5401.

[14] Giacobazzi, R., Ranzato, F., Scozzari, F.: Complete Abstract Interpretations Made Constructive, in: *Mathematical Foundations of Computer Science 1998, 23rd International Symposium, MFCS'98 Brno, Czech Republic, August 2428, 1998, Proceedings* (L. Brim, J. Gruska, J. Zlatuska, Eds.), vol. 1450 of *Lecture Notes in Computer Science*, Springer, Berlin Heidelberg, 1998, ISBN 978-3-540-64827-7, 366–377.

[15] Giacobazzi, R., Ranzato, F., Scozzari, F.: Making Abstract Interpretations Complete, *Journal of the ACM*, **47**(2), 2000, 361–416, ISSN 0004-5411.

[16] Giacobazzi, R., Ranzato, F., Scozzari, F.: Making abstract domains condensing, *ACM Transactions on Computational Logic*, **6**(1), 2005, 33–60, ISSN 1529-3785.

[17] Mycroft, A.: Completeness and predicate-based abstract interpretation, *Proceedings of the ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation, PEPM'93*, ACM Press, New York, NY, USA, 1993, ISBN 0-89791-594-1.

[18] Ranzato, F., Tapparo, F.: Generalizing strong preservation by abstract interpretation, *Journal of Logic and Computation*, **17**(1), 2007, 157–197, ISSN 1465-363X (online) 0955-792X (print).

[19] Ranzato, F., Tapparo, F.: Generalizing the Paige-Tarjan algorithm by abstract interpretation, *Information and Computation*, **206**(5), 2008, 620–651, ISSN 0890-5401.

[20] Schmidt, D. A.: Comparing Completeness Properties of Static Analyses and Their Logics, in: *Programming Languages and Systems, 4th Asian Symposium, APLAS 2006, Sydney, Australia, November 8–10, 2006. Proceedings* (N. Kobayashi, Ed.), vol. 4279 of *Lecture Notes in Computer Science*, Springer, Berlin Heidelberg, 2006, ISBN 978-3-540-48937-5, 183–199.

[21] Scozzari, F.: Abstract domains for sharing analysis by optimal semantics, in: *Static Analysis, 7th International Symposium, SAS 2000, Santa Barbara, CA, USA, June 29 – July 1, 2000, Proceedings* (J. Palsberg, Ed.), vol. 1824 of *Lecture Notes in Computer Science*, Springer, Berlin Heidelberg, 2000, ISBN 978-3-540-67668-6, 397–412.