Math. Struct. in Comp. Science (2016), *vol.* 26, *pp.* 658–701.
doi:10.1017/S0960129514000358 First published online 12 November 2014

Making abstract models complete[†]

ROBERTO GIACOBAZZI and ISABELLA MASTROENI

Università degli Studi di Verona, Dipartimento di Informatica
Strada Le Grazie, 15, 37134 Verona, Italy
Email: roberto.giacobazzi@univr.it and isabella.mastroeni@univr.it

Received 31 December 2011; revised 10 February 2014

Completeness is a key feature of abstract interpretation. It corresponds to exactness of the abstraction of fix-points and relies upon the need of absence of false alarms in static program analysis. Making abstract interpretation complete is therefore a major problem in approximating the semantics of programming languages. In this paper, we consider the problem of making abstract interpretations complete by minimally modifying the predicate transformer, i.e. the semantics, of a program. We study the mathematical properties of complete functions on complete lattices and prove the existence of minimal transformations of monotone functions to achieve completeness. We then apply minimal complete transformers to prove the minimality of standard program transformations in security, such as static program monitoring.

1. Introduction

Abstract interpretation (Cousot and Cousot 1977) is not only a theory for the approximation of the semantics of dynamic systems, but also a way of thinking about information and computation. From this point of view a program can be seen as an *abstraction transformer* i.e. a function which maps properties of input data into properties of output data, generalizing Dijkstra's predicate transformer semantics, by considering abstractions as the objects of the computation: the way a program transforms abstractions tells us a lot about the way information flows and is manipulated during the computation. For instance the following two programs clearly compute the same input/output function, multiplying the input integer values a and b:

$$
\begin{array}{lll}
C: & \mathit{input}(\mathtt{a},\mathtt{b}); & C': \ \mathit{input}(\mathtt{a},\mathtt{b});\\
& \mathtt{x}=\mathtt{a}*\mathtt{b}; & \mathtt{x}=0;\\
& \mathit{output}(\mathtt{x}) & \mathtt{if}\ \mathtt{b}\leqslant 0\ \mathtt{then}\ \{\mathtt{a}=-\mathtt{a};\mathtt{b}=-\mathtt{b}\};\\
& & \mathtt{while}\ \mathtt{b}\neq 0\ \{\mathtt{x}=\mathtt{a}+\mathtt{x};\mathtt{b}=\mathtt{b}-1\};\\
& & \mathit{output}(\mathtt{x})
\end{array}
$$

An automated program sign analysis, replacing concrete computations with approximated ones (i.e. the rule of signs) is able to catch, with no loss of precision, the intended sign behaviour of C, while this is not possible in C'. This is because the rule of signs is imprecise

[†] This is a revised and extended version of two papers that appeared in the Proceedings of SAS'08 (Giacobazzi and Mastroeni 2008) and SEFM'08 (Giacobazzi 2008).

in approximating integer addition, while it is precise (complete) in approximating integer multiplication. The two programs therefore differ in the way they transform abstractions, although they compute the same input-output function. Similar situations hold in program transformation (e.g. in compile-time program optimization) where the transformed code, while preserving the concrete semantics of the source code, may worsen the precision of an analysis (Laviron and Logozzo 2009). It is therefore becoming more and more relevant looking at programs as *abstraction transformers* and study the properties of these transformations. This is the case in abstract non-interference (Giacobazzi and Mastroeni 2004, 2008), where in order to understand *who* can attack the code and *what* information is released, we study how data properties, namely abstractions, are manipulated and possibly released during program execution and observed by attackers which are abstract interpretations, or in code obfuscation (Dalla Preda and Giacobazzi 2009; Giacobazzi 2008), where obfuscating programs means making an (abstract) interpreter (the attacker) imprecise in its analysis. This view exposes new possibilities for abstract interpretation use, e.g. in security, code design and protection, as well as posing problems concerning the methods according to which these transformations are studied. Even if clearly previewed in the early stages of abstract interpretation (Cousot and Cousot 1979c), this approach to the use of abstract interpretation is still relatively unexplored.

1.1. *The problem*

A major challenge in abstract interpretation is precision, which is *completeness* (Giacobazzi *et al.* 2000). Completeness means exactness in the analysis: an abstraction is complete for a program `P` whenever the approximation of the semantics of `P`, with respect to some properties of interest, does not generate any loss of precision. This means that the analysis of the program `P` is *insensitive* to the specific abstraction (Cousot and Cousot 1979c; Mycroft 1993). Consider again the multiplication example, completeness of the sign analysis means that the sign of the result does not change (it does not lose precision) when we consider the sign abstract semantics of the program, namely whenever we compute the program semantics on signs instead of on values. This does not hold for the program on the right, containing the addition for which the sign analysis in incomplete. In this case, the result may be imprecise, since there are cases in which we cannot say anything about the sign of the result, e.g. when the added values have different sign.

The problem of making abstract domains complete in the standard adjoint framework of abstract interpretation by minimally transforming abstractions has been successfully solved in Giacobazzi *et al.* (2000). In this paper, the authors proposed a general theory of domain transformations that provide, under the non-restrictive hypothesis of Scott-continuous functions, the minimal refinements and simplifications of an abstract domain which is complete for those functions. In other words, this characterization provides the minimal transformation of the analysis which is precise enough for the given program semantics. Among these problems, the most relevant is refining. Refining abstractions towards completeness in fact corresponds to remove *false alarms*, therefore adapting the analysis to the particular program (or family of programs) under analysis. This has been successfully implemented with ad hoc advanced abstract domain tuning in the ASTRÉE

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

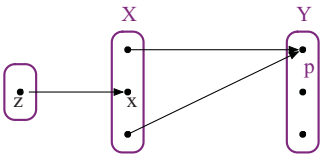


Fig. 1. Example of incomplete abstraction of transition system.

analyser (Cousot *et al.* 2007a). Although general and constructive in fact, the method presented in Giacobazzi *et al.* (2000) did not provide a direct algorithm for refining abstractions which could be applied to any program and domains. Efficient algorithmic implementations of these methods have been introduced in abstract model checking in Clarke *et al.* (2003) and predicate abstraction in Gulavani and Rajamani (2006), where abstractions correspond to partitions of the state space, by the so-called *counter example guided abstraction refinement* iterative refinement. All these methods provide solutions to the problem of deriving complete abstractions, but none of them attack the problem of modifying semantics, i.e. the program model, in order to achieve completeness. As observed in Giacobazzi *et al.* (2000), and earlier in Cousot and Cousot (1979c), we can attack the problem of making an analysis complete either by modifying the abstractions or by modifying the model (i.e. the program). While the first has been widely investigated in the last decade from the most general setting in Giacobazzi *et al.* (2000) to efficient and practical refinement strategies in Ball *et al.* (2002), Clarke *et al.* (2003), Cousot *et al.* (2007a,b) and Laviron and Logozzo (2009), the latter is still an open problem.

In order to show what we mean by *transforming semantics to achieve completeness*, let us consider the meaning of incompleteness in transition systems. In this case, we know that (e.g. see Mastroeni (2008); Ranzato and Tapparo (2007)) an abstraction is (forward) complete for a transition function if the image of an abstract state may only be the *union* of abstract states, namely if it can only contain entire blocks of states partitioned by the abstraction. This means that we have incompleteness when the image of an abstract state covers only a subset of another abstract state. This naturally derives from the so-called *existential* abstraction, in such a way that an abstract transition relates two blocks of concrete states (abstract states) if there exists at least one concrete transition from one state to another belonging respectively to the two blocks (Dams *et al.* 1997). In order to better understand this concept, consider the transition system in Figure 1. In the figure the (concrete) states are depicted as plain bullets while the transition relation f is depicted with plain arrows, X and Y are abstract states, i.e. sets of concrete states. The abstract system is characterized by defining the abstract transition relation as $f^a(W) \stackrel{\text{def}}{=} \{W' \mid \exists w \in W, w' \in W'. f(w) = w'\}$. We can show that this abstract transition system is incomplete. In particular, from a state z (an abstract state containing only one concrete state, z) we can reach the abstract state Y where the condition p holds, while in the concrete system this is not possible. In other words there are abstract traces (from z to Y) that cannot be simulated by any concrete trace in the system (from z to any state in Y). These kind of traces are called *spurious*. The partition (abstraction) refinement here, to get completeness, would split the abstract state X (Clarke *et al.* 2003). If instead we want to force completeness by *transforming* the transition function f, then we have at

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

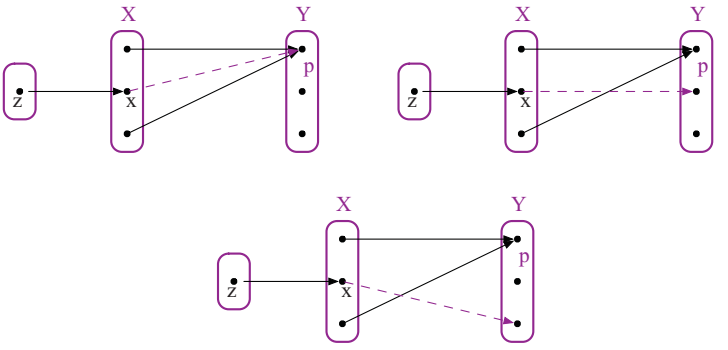


Fig. 2. Refining transition relations.

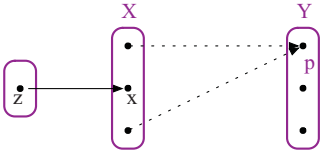


Fig. 3. Simplifying transition relations.

least to force $f(x) \subseteq Y$, for instance by adding a single transition, such as in Figure 2. In this way we force also in the concrete system the abstract traces. However, this is not the only possibility we have in order to force completeness. We can also choose to simplify the transition function by removing transitions. In this latter case the idea is to restrict the image of the function of an abstract state to the greatest union of abstract states completely contained in it, for instance consider the case in Figure 3, in this situation we can choose to erase also from the abstract system the spurious traces. Hence, we *remove* all the transitions from X to Y.

Main contribution. In this paper, we consider the mathematical problem of *minimally transforming semantics* in order to achieve completeness. Hence, we propose a *minimal* transformation and we transform *program semantics*. By minimal we mean the closest transformation in the approximation order, namely we transform functions by enriching the output images, i.e. by adding noise to the output computation, or by restricting the output images of the function, i.e. by losing output information. Hence, minimal here means that we add or remove the *minimal* amount of information from the output observation of the semantics. Moreover, as we underlined before, we propose to transform program semantics. It is clear that not all program transformations are admissible in general and, in particular, it may not be meaningful to transform semantics in all possible contexts. We show here that, whenever these transformations are allowed then they can be modelled as completeness transformation, e.g. in program monitoring as showed in Section 6.

A model for a program P is a triple $\langle A, B, f_{\mathtt{P}} \rangle$ where A and B are possibly different domains of computational objects, and $f_{\mathtt{P}} : A \longrightarrow B$ is a semantic transformer, e.g. a predicate transformer, associating with each program P a function modelling the

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

input/output behaviour of P. As usual in denotational semantics, models may involve fix-point computations. In this general setting, we study the lattice-theoretic structure of the set of all complete semantics for a given pair of abstractions, respectively for the domain A and B. We consider both backward, denoted $\mathscr{B}$, and forward, denoted $\mathscr{F}$, completeness (Giacobazzi and Quintarelli 2001). If we compare the results in the abstract domain, we obtain what is called *backward completeness*, while, if we compare the results in the concrete domain we obtain the so-called *forward completeness*.

Backward completeness corresponds precisely to the standard notion of completeness for an abstract interpretation (Cousot and Cousot 1979c; Giacobazzi *et al.* 2000), requiring that no loss of precision is accumulated in computing by using approximate (abstract) objects. A classical example is the *rule of signs* which is clearly backward complete for integer multiplication but not for addition. This means that the sign of values is sufficient to completely characterize the sign of the output multiplications, while this is not sufficient for integer addition (in the latter case, the least refinement of signs which is complete is the abstract domain of intervals (Giacobazzi *et al.* 2000)).

Forward completeness is relatively less known, and it corresponds to requiring that no loss of precision is accumulated by approximating the output of a concrete computation. An example can be found in the context of abstract transition systems. In particular, for instance strong preservation in abstract model checking (absence of spurious counter-example) is modelled as a forward completeness w.r.t. the inverse image of the transition relation (Giacobazzi and Quintarelli 2001).

It is known that forward and backward completeness are dual notions, see Giacobazzi and Quintarelli (2001), and that an abstraction is $\mathscr{B}$-complete for an additive function f if and only if it is $\mathscr{F}$-complete for its right adjoint f^+.

In the following, we consider the problem of modifying models, i.e. functions, in order to achieve completeness. Achieving $\mathscr{F}$-completeness means here *forcing* a function to reduce its output to fall into an abstraction. This is precisely restricting a function by selecting its outputs, an idea that is not new in program transformation. For instance, in Vechev *et al.* (2010), the authors combine abstraction and semantics refinement for synchronization synthesis in multi-threaded programs. The idea is that of *modifying the semantics*, the set of all the possible execution traces, by *restricting* them to a subset of traces which satisfy some synchronization constraints.

$\mathscr{B}$-completeness instead transforms a function in order to make its abstraction insensitive on the abstraction of the input. This means that the transformed function operates only on an abstraction of its input, making the non-abstract objects irrelevant for its output. Once again this corresponds to restricting the function. In both cases, we can drive a function (program) transformation towards completeness, by restricting the function on specific elements. We believe that this is suitable for modelling a variety of program transformations in security. In particular, we prove that static program monitoring can be specified in terms of making a given program complete.

The paper is structured as follows. In Section 2, we introduce the main mathematical notation and notions, including a brief presentation of the standard adjoint framework of abstract interpretation, including the main results on soundness and completeness. In Section 3, we show that software watermarking is in an ideal context where completeness

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

semantic transformers may be used for characterizing and deeply understanding different watermarking techniques. In Section 4, we present a lattice-theoretic characterization of forward complete semantics, including minimal transformations for modifying a non-complete function towards a complete one, providing both an upper and lower approximation of the transformed function. In Section 5 the same construction is developed for backward completeness. The validity of our approach is proved in Section 6, where model transformations are applied both to prove minimality in static program monitoring as complete model transformation. Section 7 concludes the paper by exploring future research directions.

2. Preliminaries

In this paper, we consider the standard definition of abstract domains as Galois connections, as formalized in Cousot and Cousot (1977, 1979c). We introduce the basic mathematical background concerning Galois connection-based abstract interpretation, residuated closures, fix-point soundness and completeness (Davey and Priestley 1990; Nielson *et al.* 1999).

2.1. *Basic lattice and fix-point theory*

If S and T are sets, then $\wp(S)$ denotes the powerset of S, $|S|$ the cardinality of S, $S \subsetneq T$ strict inclusion, $S \times T$ the cartesian product, and for a function $f : S \to T$ and $X \subseteq S$, $f(X) \stackrel{\text{def}}{=} \{f(x) \mid x \in X\}$. By $g\ f$ we denote the composition of the functions f and g, i.e. $g\ f \stackrel{\text{def}}{=} \lambda x.g(f(x))$. $\langle P, \leqslant \rangle$ denotes a poset P with ordering relation $\leqslant$, while $\langle C, \leqslant, \vee, \wedge, \top, \bot \rangle$ denotes a complete lattice C, with ordering $\leqslant$, *lub* $\vee$, *glb* $\wedge$, greatest element (top) $\top$, and least element (bottom) $\bot$. Often, $\leqslant_P$ will be used to denote the underlying ordering of a poset P, and $\vee_C$, $\wedge_C$, $\top_C$ and $\bot_C$ to denote the basic operations and elements of a complete lattice C. Let P be a poset and $S \subseteq P$. Then, $\max(S) \stackrel{\text{def}}{=} \{x \in S \mid \forall y \in S.\ x \leqslant_P y \Rightarrow x = y\}$ denotes the set of maximal elements of S in P; also, the downward closure of S is defined by $\downarrow S \stackrel{\text{def}}{=} \{x \in P \mid \exists y \in S.\ x \leqslant_P y\}$, and for $x \in P$, $\downarrow x$ is a shorthand for $\downarrow \{x\}$, while the upward closure $\uparrow$ is dually defined. We use the symbol $\sqsubseteq$ to denote point-wise ordering between functions: If S is any set, P a poset, and $f, g : S \to P$ then $f \sqsubseteq g$ if for all $x \in S$, $f(x) \leqslant_P g(x)$. An operator $f : P \longrightarrow P$ is extensive if $\forall x \in P.\ x \leqslant_P f(x)$. It is reductive if $\forall x \in P.\ x \geqslant_P f(x)$. Let C and D be complete lattices. Then, $C \xrightarrow{\text{m}} D$ and $C \xrightarrow{\text{c}} D$ denote, respectively, the set and the type of all monotone and (Scott-)continuous functions from C to D. Recall (Abramsky and Jung 1994) that $f \in C \xrightarrow{\text{c}} D$ iff f preserves *lub*'s of (non-empty) chains iff f preserves *lub*'s of directed subsets. Also, $f : C \to D$ is (completely) additive if f preserves *lub*'s of all subsets of C (emptyset included), while co-additivity is dually defined. The *additive lift* of $f : C \to D$ is a function $f^a : \wp(C) \to \wp(D)$ such that $f^a \stackrel{\text{def}}{=} \lambda X.\ \{f(x) \mid x \in X\}$. Recall that any function can be transformed into the closest (from below and from above) monotone function that approximates it. This is achieved by considering the following functionals (Cousot and Cousot 1979a) $\mathbb{M}^{\downarrow}, \mathbb{M}^{\uparrow} : (C \longrightarrow D) \longrightarrow (C \xrightarrow{\text{m}} D)$ such that

$$\mathbb{M}^{\downarrow} \stackrel{\text{def}}{=} \lambda f.\ \lambda x.\ \bigwedge\{f(y) \mid y \geqslant x\} \qquad \mathbb{M}^{\uparrow} \stackrel{\text{def}}{=} \lambda f.\ \lambda x.\ \bigvee\{f(y) \mid y \leqslant x\}.$$

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

Similarly, any function can be transformed into the closest extensive or reductive map by considering the following functionals (Cousot and Cousot 1979a) $\mathbb{E}, \mathbb{R} : (C \longrightarrow C) \longrightarrow (C \longrightarrow C)$ such that:

$$\mathbb{E} \stackrel{\text{def}}{=} \lambda f.\, \lambda x.\, f(x) \vee x \qquad \mathbb{R} \stackrel{\text{def}}{=} \lambda f.\, \lambda x.\, f(x) \wedge x$$

We denote by $lfp(f)$ and $gfp(f)$, respectively, the least and greatest fix-point, when they exist, of an operator f on a poset. The well-known Knaster–Tarski's theorem states that any monotone operator $f : C \xrightarrow{\text{m}} C$ on a complete lattice C admits both least and greatest fix-points, and the following characterizations hold:

$$lfp(f) = \wedge_C\{x \in C \mid f(x) \leqslant_C x\} \quad \text{and} \quad gfp(f) = \vee_C\{x \in C \mid x \leqslant_C f(x)\}.$$

Let us note that if $f, g : C \xrightarrow{\text{m}} C$ and $f \sqsubseteq g$ then $lfp(f) \sqsubseteq lfp(g)$. It is known that if $f : C \xrightarrow{\text{c}} C$ is continuous then $lfp(f) = lfp(\mathbb{E}(f)) = \vee_{i\in\mathbb{N}} f^i(\bot_C)$, where, for any $i \in \mathbb{N}$ and $x \in C$, the ith power of f in x is inductively defined as follows: $f^0(x) = x$; $f^{i+1}(x) = f(f^i(x))$. Dually, if $f : C \to C$ is co-continuous then $gfp(f) = gfp(\mathbb{R}(f)) = \wedge_{i\in\mathbb{N}} f^i(\top_C)$. $\{f^i(\bot_C)\}_{i\in\mathbb{N}}$ and $\{f^i(\top_C)\}_{i\in\mathbb{N}}$ are called, respectively, the upper and lower Kleene's iteration sequences of f (Cousot and Cousot 1979b). The set of all finite sequences (traces) over an alphabet Σ is denoted Σ^+. If $\sigma, \sigma' \in \Sigma^+$ then $\sigma\sigma' \in \Sigma^+$ is the concatenation of the two sequences.

2.2. *Abstract domains individually and collectively*

Concrete domains represent collections of computational objects on which the concrete semantics and models are defined. These include standard data-types (e.g. heap, stack, numerical types) control-flow structures, etc. Abstract domains are collections of approximate objects, representing properties of concrete objects in a domain-like structure. The relation between concrete and abstract domains can be specified in terms of *Galois connections*, and this sets up the so-called standard adjoint framework of abstract interpretation (Cousot and Cousot 1977). The adjoint presentation is a relatively restrictive view of abstract interpretation. Weaker frameworks could involve the weakening of the relation between concrete and abstract domains, e.g. in Cousot and Cousot (1992a), or sophisticated fix-point iteration strategies by *fix-point widening* on approximate domains (Cousot and Cousot 1992b). In this paper, we consider abstractions in the standard adjoint framework, which provides the richest mathematical environment for proving properties about abstractions.

The aim of the following part is that of recalling the basic concepts and properties in abstract interpretation that will be used in the paper, for a formal introduction of abstract interpretation see (Cousot and Cousot 1977, 1979c). If $\langle C, \leqslant, \top, \bot, \vee, \wedge\rangle$ is a complete lattice, a pair of monotone functions $\alpha : C \xrightarrow{\text{m}} A$ and $\gamma : A \xrightarrow{\text{m}} C$ forms an *adjunction* or a *Galois connection* if for any $x \in C$ and $y \in A$: $\alpha(x) \leqslant_A y \Leftrightarrow x \leqslant_C \gamma(y)$. The function α is the *left- adjoint* to γ and it is additive, i.e. it preserves *lub*'s of all subsets of the domain (emptyset included). The function γ is the *right-adjoint* to α and it is additive, i.e. it preserves *glb*'s of all subsets of the domain (emptyset included). The right adjoint of a function α is $\alpha^+ \stackrel{\text{def}}{=} \lambda x.\ \bigvee\{y \mid \alpha(y) \leqslant x\}$. Conversely the left adjoint

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

of γ is $\gamma^- \stackrel{\text{def}}{=} \lambda x.\ \bigwedge\{y |\ x \leqslant \gamma(y)\}$. In this case: $\gamma^- = \alpha$ and $\alpha^+ = \gamma$. It is known that abstract domains can be formalized as closure operators on the concrete domain (Cousot and Cousot 1979c). An *upper [lower] closure operator* $\rho : P \longrightarrow P$ on a poset P is monotone, idempotent, and extensive [reductive]. Closures are uniquely determined by their fix-points $\rho(C)$. In the following, we will write $x \in \rho$ as shorthand for x fix-point of ρ, i.e. for $x \in \rho(C)$. In the following, we will often use closures both as functions and as sets (viz., domains). Given $X \subseteq C$, the least abstract domain containing X is the least closure including X as fix-points, which is the *Moore-closure* or *Moore family* $\mathcal{M}(X) \stackrel{\text{def}}{=} \{\bigwedge S \mid S \subseteq X\}$. It turns out that $\langle \rho(C), \leqslant \rangle$ is a complete meet subsemilattice of C (i.e. $\wedge$ is its *glb*), but, in general, it is not a complete sublattice of C, since the *lub* in $\rho(C)$ – defined by $\lambda Y \subseteq \rho(C).\ \rho(\vee Y)$ – might be different from that in C. In fact, $\rho(C)$ is a complete sublattice of C iff ρ is additive. The set of all upper closure operators on P is denoted by *uco*(P), while lower closure operators are denoted *lco*(P). If C is a complete lattice, then $\langle uco(C), \sqsubseteq, \sqcup, \sqcap, \lambda x.\top, id \rangle$ is a complete lattice (Ward 1942), which is isomorphic to the *lattice of abstract domains* of C (cf. (Cousot and Cousot 1977, Section 7) and (Cousot and Cousot 1979c, Section 8)). where $id \stackrel{\text{def}}{=} \lambda x.x$ and for every $\rho, \eta \in uco(C)$, $\{\rho_i\}_{i \in I} \subseteq uco(C)$ and $x \in C$:

— $\rho \sqsubseteq \eta$ iff $\forall y \in C.\ \rho(y) \leqslant \eta(y)$ iff $\eta(C) \subseteq \rho(C)$;
— $(\sqcap_{i \in I} \rho_i)(x) = \wedge_{i \in I} \rho_i(x)$;
— $(\sqcup_{i \in I} \rho_i)(x) = x \Leftrightarrow \forall i \in I.\ \rho_i(x) = x$;
— $\lambda x.\top$ is the top element and $\lambda x.x$ is the bottom element.

Thus, the *glb* in $uco(C)$ is defined point-wise, while the *lub* of a set of closures $\{\rho_i\}_{i \in I} \subseteq uco(C)$ is the closure whose set of fix-points is given by the set-intersection $\cap_{i \in I} \rho_i(C)$. In the following, we will make use of the following basic properties for $\rho, \eta \in uco(C)$ and $Y \subseteq C$:

i. $\rho(\wedge \rho(Y)) = \wedge \rho(Y)$;
ii. $\rho(\vee Y) = \rho(\vee \rho(Y))$;
iii. $\eta \sqsubseteq \rho \Leftrightarrow \eta\rho = \rho \Leftrightarrow \rho\eta = \rho$.

In abstract interpretation, A_1 is more precise (viz. more concrete) than A_2 (i.e. A_2 is an abstraction of A_1) iff $A_1 \sqsubseteq A_2$ in *uco*(C) iff $A_2 \in$ *uco*(A_1).

2.3. *Adjoining closure operators*

In the following, we will make an extensive use of adjunction, in particular of closure operators. Janowitz (1967) characterized the structure of *residuated* (adjoint) closure operators by the following basic result (see also Blyth and Janowitz (1972)).

Theorem 2.1 (Janowitz 1967). Let $\langle \eta, \eta^+ \rangle$ and $\langle \eta^-, \eta \rangle$ be pairs of adjoint operators on C.

$$(1)\ \ \eta \in uco(C) \Leftrightarrow \eta^+ \in lco(C) \Leftrightarrow \begin{cases} \eta\eta^+ = \eta^+ \\ \eta^+\eta = \eta \end{cases}$$

$$(2)\ \ \eta \in uco(C) \Leftrightarrow \eta^- \in lco(C) \Leftrightarrow \begin{cases} \eta\eta^- = \eta \\ \eta^-\eta = \eta^- \end{cases}$$

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

This theorem says that the adjoint of a closure operator is always a closure operator, when it exists. In particular, the adjoints of upper closure operators are lower operators, and, by duality, the adjoint of a lower operator is an upper one. Let us explain the relations described in the theorem, existing between closure adjoints, in the lco case. Let $\tau \in lco(C)$. By Theorem 2.1, if τ^- exists then $\tau^-(\tau(X)) = \tau(X)$ and $\tau(\tau^-(X)) = \tau^-(X)$. This means that τ^- is such that both τ and τ^- have the same sets of fix-points, namely τ^- extends any object X to the largest object Y such that $\tau(Y) = Y$. Conversely, the right adjoint of τ, when it exists, is quite different. By Theorem 2.1, we have that if τ^+ exists then $\tau^+(\tau(X)) = \tau^+(X)$ and $\tau(\tau^+(X)) = \tau(X)$. In this case $\tau^+(X)$ is not a fix-point of τ. Instead, it is the least element Y that can be lifted by τ to the same object as X does, i.e. such that $\tau(X) = X = \tau(Y)$. The following result strengthen Theorem 2.1 by showing the order-theoretic structure of residuated closures.

Proposition 2.2. Let $\tau \in lco(C)$ and $\eta \in uco(C)$.

1. If $\langle \tau^-, \tau \rangle$ and $\langle \eta, \eta^+ \rangle$ are pairs of adjoint functions then $\tau^- = \lambda X.\ \bigwedge\{\tau(Y) | \tau(Y) \geqslant X\}$ and $\eta^+ = \lambda X.\ \bigvee\{\eta(Y) | X \geqslant \eta(Y)\}$.
2. If $\langle \tau, \tau^+ \rangle$ and $\langle \eta^-, \eta \rangle$ are pairs of adjoint functions then $\tau^+ = \lambda X.\ \bigvee\{Y | \tau(Y) = \tau(X)\}$ and $\eta^- = \lambda X.\ \bigwedge\{Y | \eta(X) = \eta(Y)\}$.

Proof.

1. Let us prove that $\bigwedge\{\tau(y) |\ \tau(y) \geqslant x\} = \bigwedge\{y |\ \tau(y) \geqslant x\}$ (the proof for η can be obtained by duality). We suppose that $\langle \tau^-, \tau \rangle$ is a pair of adjoint functions, namely that τ is co-additive. We prove that the two implications of equality separately. Note that, by idempotence of τ

$$\{\tau(y) |\ \tau(y) \geqslant x\} \subseteq \{y |\ \tau(y) \geqslant x\} \Rightarrow \bigwedge\{\tau(y) |\ \tau(y) \geqslant x\} \geqslant \bigwedge\{y |\ \tau(y) \geqslant x\}.$$

   On the other hand, τ is co-additive, therefore

$$\tau(\bigwedge\{y |\ \tau(y) \geqslant x\}) = \bigwedge\{\tau(y) |\ \tau(y) \geqslant x\} \geqslant x \Rightarrow \tau(\bigwedge\{y |\ \tau(y) \geqslant x\}) \in \{\tau(y) |\ \tau(y) \geqslant x\}.$$

   This means, since τ is reductive, that

$$\bigwedge\{y |\ \tau(y) \geqslant x\} \geqslant \tau\left(\bigwedge\{y |\ \tau(y) \geqslant x\}\right) \geqslant \bigwedge\{\tau(y) |\ \tau(y) \geqslant x\}.$$

   In this way we proved the equality.
2. Let us prove the result for τ, the other case is obtained by duality. Since, we suppose that $\langle \tau, \tau^+ \rangle$ is a pair of adjoint functions, we are supposing that τ is additive. We prove the two implication of equality separately. First of all note that, since τ is reductive, i.e. $\tau(x) \leqslant x$, then

$$\{y |\ \tau(y) = \tau(x)\} \subseteq \{y |\ \tau(y) \leqslant x\} \Rightarrow \bigvee\{y |\ \tau(y) = \tau(x)\} \leqslant \bigvee\{y |\ \tau(y) \leqslant x\}.$$

   On the other hand, since $\tau(x) \in \{\tau(y) |\ \tau(y) \leqslant x\}$, by additivity of τ, we have that

$$\tau\Big(\bigvee\{y |\ \tau(y) \leqslant x\}\Big) = \bigvee\{\tau(y) |\ \tau(y) \leqslant x\} \geqslant \tau(x).$$

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

Moreover, for each y, such that $\tau(y) \leqslant x$, we have $\tau(y) \leqslant \tau(x)$, by idempotence and monotonicity of τ, therefore

$$\tau(x) \geqslant \bigvee\{\tau(y) |\ \tau(y) \leqslant x\} \ \Rightarrow\ \tau(x) \geqslant \tau(\bigvee\{y |\ \tau(y) \leqslant x\}).$$

Hence, $\tau(\bigvee\{y |\ \tau(y) \leqslant x\}) = \tau(x)$, i.e. $\bigvee\{y |\ \tau(y) \leqslant x\} \in \{y |\ \tau(y) = \tau(x)\}$, which implies

$$\bigvee\{y |\ \tau(y) = \tau(x)\} \geqslant \bigvee\{y |\ \tau(y) \leqslant x\}.$$

In this way we proved the equality. □

Uniform closures have been introduced in Giacobazzi and Ranzato (1998b) for specifying the notion of *abstract domain compression*, namely the operation for reducing abstract domains to their minimal structure with respect to some given abstraction refinement $\eta \in \mathit{lco}(\mathit{uco}(C))$. An upper closure η is *meet-uniform* (Giacobazzi and Ranzato 1998b) if $\eta(\bigwedge\{Y | \eta(X) = \eta(Y)\}) = \eta(X)$. Join-uniformity is dually defined for lower closures. Well-known non-co-additive upper closures are meet-uniform, such as the downward closure $\downarrow$ of a subset of a partially ordered set (Giacobazzi and Ranzato 1998b). The following is an example of meet-uniform closure.

Example 2.3. It is easy to prove that the *limited-interval abstract domain* with $\mathtt{m} \in \mathbb{N}$:

$$\begin{aligned} \imath \ \stackrel{\text{def}}{=}\ \lambda x.\ \textbf{let}\ a = \textbf{if}\ \mathit{min}(x) < -\mathtt{m}\ \textbf{then}\ -\mathtt{m}\ \textbf{else}\ \mathit{min}(x), \\ b = \textbf{if}\ \mathit{max}(x) > \mathtt{m}\ \textbf{then}\ -\mathtt{m}\ \textbf{else}\ \mathit{max}(x)\ \textbf{in}\ [a, b] \in \mathit{uco}(\wp([-\mathtt{m}, \mathtt{m}])) \end{aligned}$$

is a meet-uniform closure: If $Y \subseteq \wp([-\mathtt{m}, \mathtt{m}])$ and for any $x, y \in Y$: $\imath(x) = \imath(y)$, then for any $x, y \in Y$. $\min(x) = \min(y) \wedge \max(x) = \max(y)$. Therefore there exists $z \in Y$ such that $\imath(\bigcap Y) = \imath(z)$. In this case, $\imath^- = \lambda x.\ \{\min(x), \max(x)\} \in \mathit{lco}(\wp([-\mathtt{m}, \mathtt{m}]))$.

It is known that any $\rho \in \mathit{uco}(C)$ is join-uniform and the set of meet-uniform upper closures $\mathit{uco}^*(C)$ is a Moore-family of $\mathit{uco}(C)$. The following is immediate by Theorem 2.1 and Proposition 2.2.

Corollary 2.4. Let $\eta \in \mathit{uco}(C)$. $\langle \eta^-, \eta \rangle$ is a pair of adjoint closures iff η is meet-uniform.

2.4. *Soundness and completeness*

Let $f : C \xrightarrow{\mathrm{m}} D$ be a concrete semantic operation defined over some concrete domains C and D. Let an abstract interpretation be specified by Galois connections with abstract domains $\rho(C)$ and $\eta(D)$ corresponding to closure operators $\rho \in \mathit{uco}(C)$ and $\eta \in \mathit{uco}(D)$ respectively, and by a corresponding abstract semantics $f^\sharp : \rho \xrightarrow{\mathrm{m}} \eta$. Then, $f^\sharp$ is *sound* for (or is a *correct approximation* of) f if $\eta f \sqsubseteq f^\sharp \rho$. This holds iff $\eta f \rho \sqsubseteq f^\sharp$. The function $\eta f \rho$ is called *best correct approximation of f in ρ and η*. Whenever $f : C \xrightarrow{\mathrm{m}} C$ and $f^\sharp : \rho \xrightarrow{\mathrm{m}} \rho$, $f^\sharp$ is *fix-point sound* for f if $\rho(\mathit{lfp}(f)) \leqslant \mathit{lfp}(f^\sharp)$. A sound over-approximation means that no error can be forgotten by the analysis, i.e. the approximate semantics includes a full coverage of all possible concrete computations, e.g. the collections of all reachable states. As we recalled in the introduction, a well-known basic result of abstract

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

interpretation (Cousot and Cousot 1979c, Theorem 7.1.0.4) states that soundness implies fix-point soundness. It is worth remarking that fix-point soundness is in general a strictly weaker property than soundness.

Precision of an abstract interpretation is typically defined in terms of *completeness* (Cousot and Cousot 1979c; Mycroft 1993). Depending on where we compare the concrete and the abstract computations, we obtain two different notions of completeness (Giacobazzi *et al.* 2000; Giacobazzi and Quintarelli 2001). If we compare the results in the abstract domain, we obtain what is called *backward completeness* ($\mathscr{B}$-completeness), while, if we compare the results in the concrete domain we obtain the so-called *forward completeness* ($\mathscr{F}$-completeness). Formally, if $f : C \xrightarrow{\text{m}} C$ and $\rho \in \mathit{uco}(C)$, then ρ is $\mathscr{B}$-complete if $\rho f \rho = \rho f$, while it is $\mathscr{F}$-complete if $\rho f \rho = f \rho$. A complete over-approximation means that no false-alarms are returned by the analysis, i.e. in $\mathscr{B}$-completeness the approximate semantics computed by manipulating abstract objects corresponds precisely to the abstraction of the concrete semantics, while in $\mathscr{F}$-completeness the concrete semantics do not lose precision by computing on abstract objects. The problem of making abstract domains $\mathscr{B}$-complete has been solved in Giacobazzi *et al.* (2000) and later extended to $\mathscr{F}$-completeness in Giacobazzi and Quintarelli (2001). Let $f : C_1 \longrightarrow C_2$ and $\rho \in \mathit{uco}(C_2)$ and $\eta \in \mathit{uco}(C_1)$. $\langle \rho, \eta \rangle$ is a pair of $\mathscr{B}$-complete abstractions for f if $\rho f = \rho f \eta$, it is a pair of $\mathscr{F}$-complete abstractions for f if $f \eta = \rho f \eta$. A pair of domain transformers has been associated with any completeness problem, which are respectively a *domain refinement* and *simplification* (Filé *et al.* 1996; Giacobazzi and Ranzato 1997). In Giacobazzi *et al.* (2000) and Giacobazzi and Quintarelli (2001), a constructive characterization of the most abstract refinement, called *complete shell*, and of the most concrete simplification, called *complete core*, of any abstract domain, making it $\mathscr{F}$ or $\mathscr{B}$-complete for a given continuous function f, is given as a solution of simple abstract domain equations given by the following basic operators:

$$\begin{array}{l|l} R_f^{\mathscr{F}} \stackrel{\text{def}}{=} \lambda X.\, \mathcal{M}(f(X)) & R_f^{\mathscr{B}} \stackrel{\text{def}}{=} \lambda X.\, \mathcal{M}(\bigcup_{y \in X} \max(f^{-1}(\downarrow y))) \\ C_f^{\mathscr{F}} \stackrel{\text{def}}{=} \lambda X.\, \{ y \in L \mid f(y) \subseteq X \} & C_f^{\mathscr{B}} \stackrel{\text{def}}{=} \lambda X.\, \{ y \in L \mid \max(f^{-1}(\downarrow y)) \subseteq X \} \end{array}$$

Let $\ell \in \{\mathscr{F}, \mathscr{B}\}$. In Giacobazzi *et al.* (2000) the authors proved that the most concrete $\beta \sqsupseteq \rho$ such that $\langle \beta, \eta \rangle$ is ℓ-complete and the most abstract $\beta \sqsubseteq \eta$ such that $\langle \rho, \beta \rangle$ is ℓ-complete are respectively the ℓ-complete core and ℓ-complete shell, which are: $\mathcal{C}_f^{\ell,\eta}(\rho) \stackrel{\text{def}}{=} \rho \sqcup C_f^{\ell}(\eta)$ and $\mathcal{R}_f^{\ell,\rho}(\eta) \stackrel{\text{def}}{=} \eta \sqcap R_f^{\ell}(\rho)$. When $\eta = \rho$, we need a fix-point iteration on abstract domains, i.e. $\mathcal{R}_f^{\ell}(\rho) = \mathit{gfp}(\lambda X.\, \rho \sqcap R_f^{\ell}(X)) \in \mathit{lco}(\mathit{uco}(C))$ which is called *absolute ℓ-complete shell*. By construction if f is additive then $\mathcal{R}_f^{\mathscr{B}} = \mathcal{R}_{f^+}^{\mathscr{F}}$ (Giacobazzi and Quintarelli 2001). This means that when we have to solve a problem of $\mathscr{B}$-completeness for an additive function then we can equivalently solve the corresponding $\mathscr{F}$-completeness problem for its right adjoint. The following example from Giacobazzi and Quintarelli (2001), exemplifies the duality of forward and backward abstract domain completeness when dealing with additive functions. Assume $\mathtt{S}$ to be the domain in Figure 4, which is an obvious abstraction of $\langle \wp(\mathbb{Z}), \subseteq \rangle$ for the analysis of integer variables and $sq : \wp(\mathbb{Z}) \to \wp(\mathbb{Z})$ be the square operation defined as follows: $sq(X) = \{ x^2 \mid x \in X \}$ for $X \in \wp(\mathbb{Z})$. The arrows in Figure 4a and b show the function $sq^{\sharp}$. Let $\rho_{\mathtt{S}} \in \mathit{uco}(\wp(\mathbb{Z}))$ be the closure operator associated with $\mathtt{S}$. The best correct approximation of sq in $\mathtt{S}$ is $sq^{\sharp} : \mathtt{S} \to \mathtt{S}$

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

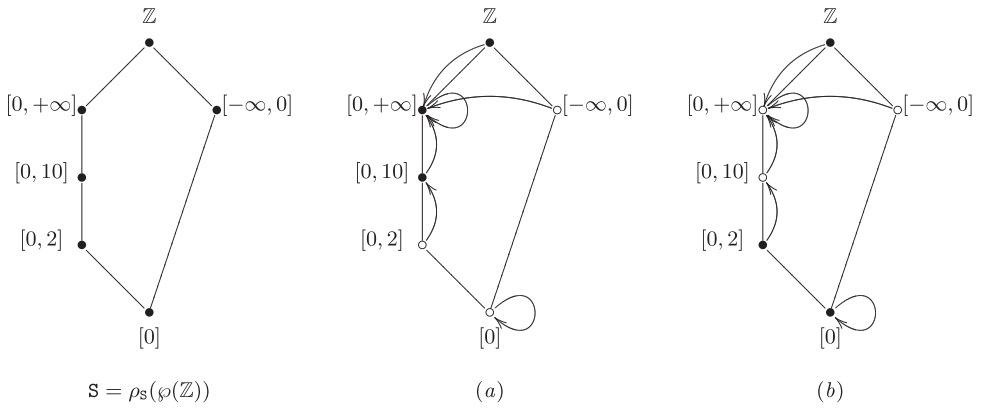


Fig. 4. The abstract domain S and two abstractions.

such that $sq^{\sharp}(X) = \rho_{\mathtt{S}}(sq(X))$, with $X \in \mathtt{S}$. It is easy to see that the abstractions $\rho_a = \{\mathbb{Z}, [0, +\infty], [0, 10]\}$ (black dots in Figure 4a) and $\rho_b = \{\mathbb{Z}, [0, 2], [0]\}$ of S (black dots in Figure 4b), respect the following facts: $\rho_a = \{\mathbb{Z}, [0, +\infty], [0, 10]\}$ is $\mathscr{F}$-complete but not $\mathscr{B}$-complete on the concrete domain S for $sq^{\sharp}$ (for instance $\rho_a(sq^{\sharp}(\rho_a([0]))) = [0, +\infty]$ but $\rho_a(sq^{\sharp}([0])) = [0, 10]$) and $\rho_b = \{\mathbb{Z}, [0, 2], [0]\}$ is $\mathscr{B}$-complete but not $\mathscr{F}$-complete on the concrete domain S for $sq^{\sharp}$ (for instance $\rho_b(sq^{\sharp}(\rho_b([0, 2]))) = \mathbb{Z}$ but $sq^{\sharp}(\rho_b([0, 2])) = [0, 10]$).

3. A motivating example: incompleteness for information hiding

Among the different methods for hiding secrets in programs, software watermarking is one of the most common and widely used (Collberg and Nagra 2010). Consider a programming language defined as the collection of all well-formed programs $\mathbb{P}$. We consider a steganographic approach to software watermarking, i.e. program transformations where the intended (typically a copyright) signature is hidden from external observers. We follow (Cousot and Cousot 2004) by defining the *stegomarker* $\mathfrak{M} : \mathcal{S} \longrightarrow \mathbb{P}$ as the encoding of the signature $s \in \mathcal{S} \subseteq A^+$ over a finite alphabet A, into a program $\mathfrak{M}(s) \in \mathbb{P}$, called the *stegomark*. The *stegolayer* $\mathfrak{L} : \mathbb{P} \times \mathbb{P} \longrightarrow \mathbb{P}$ is used to compose the stegomark with the source (cover) program. The *(watermarked) stegoprogram* $\mathfrak{S} : \mathbb{P} \times \mathcal{S} \longrightarrow \mathbb{P}$ is such that $\mathfrak{S}(\mathtt{P}, s) = \mathfrak{L}(\mathtt{P}, \mathfrak{M}(s))$ for any program $\mathtt{P} \in \mathbb{P}$ and *signature* $s \in \mathcal{S}$.

The standard taxonomy of software watermarking in Collberg and Nagra (2010), Collberg and Thomborson (1999, 2000) and Nagra *et al.* (2002), distinguish between *static watermarking*, where signatures are encoded as properties of the code text, and *dynamic watermarking*, where the signature is encoded in the state computed by the stegoprogram under suitable inputs. Abstract watermarking, introduced in Cousot and Cousot (2004), is different: the signature is encoded as a stegomark in the cover program and can be extracted by suitable static program analysis. In some sense it is dynamic being encoded in the states computed by the stegoprogram and static being observable and removable by a static analysis under suitable (secret) hypothesis (the secret key).

In the following, we show that static and dynamic watermarking are indeed both instances of abstract watermarking, under suitable choices for $\mathfrak{M}$ and $\mathfrak{L}$. In particular,

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

they are instances of a common pattern which corresponds precisely to the program transformations making semantics $\mathscr{F}$-complete.

Let $\mathtt{P} \in \mathbb{P}$ be a deterministic program on concrete data-types $\mathcal{D}$ and $\alpha, \omega \in \mathit{uco}(\wp(\Sigma^+))$ be program properties on program traces, i.e. sequences of program states in Σ, such that $\alpha \sqsubseteq \omega$. Denote by $[\![\cdot]\!] : \mathbb{P} \times \mathcal{D} \longrightarrow \Sigma^+$ the concrete semantics associating with each program $\mathtt{P}$ and input $d \in \mathcal{D}$ the computed trace $[\![\mathtt{P}]\!](d) \in \Sigma^+$. $[\![\cdot]\!]^\alpha : \mathbb{P} \times \mathcal{D} \longrightarrow \alpha$ is the abstract interpretation with respect to an abstraction $\alpha \in \mathit{uco}(\wp(\Sigma^+))$. If $\eta \in \mathit{uco}(\mathcal{D})$ and $\forall x.\ [\![\mathfrak{M}(s)]\!]^\alpha(x) \in \omega$ then $\mathfrak{L}$ is a stegolayer for P and $\mathfrak{M}(s)$ if

$$[\![\mathfrak{S}(s,\mathtt{P})]\!]^\alpha = [\![\mathfrak{L}(\mathtt{P},\mathfrak{M}(s))]\!]^\alpha \stackrel{\text{def}}{=} \lambda x.\ \begin{cases} [\![\mathfrak{M}(s)]\!]^\alpha(x) & \text{if } x \in \eta \\ [\![\mathtt{P}]\!]^\alpha(x) & \text{otherwise} \end{cases}$$

where we recall that $x \in \eta$ denotes x fix-point of η. Static software watermarking corresponds here to set $\eta = \mathit{id}$, i.e.

$$\forall x \in \mathcal{D}.\ [\![\mathfrak{S}(s,\mathtt{P})]\!]^\alpha(x) = [\![\mathfrak{M}(s)]\!]^\alpha(x) \in \omega$$

and α is a decidable abstraction. This means that the interpretation of the stegoprogram always reveals the watermark, independently from the input, provided that a (static) decidable analysis is performed on the stegoprogram $\mathfrak{S}(s,\mathtt{P})$. In dynamic watermarking instead $\eta \neq \mathit{id}$, meaning that only suitable inputs may reveal the watermark. In this case, the inputs revealing the watermark are all and only the inputs satisfying the property η. In this context, the syntactic stegomarker $\mathfrak{M}(\cdot)$ can be associated with a semantic stegomarker $\mathfrak{M}[\![\cdot]\!] : \mathcal{S} \longrightarrow \mathit{uco}(\wp(\Sigma^+))$. $\omega = \mathfrak{M}[\![s]\!]$ can be any property of the watermark, for instance $\mathfrak{M}[\![s]\!] = \{\Sigma^+, X_s\} \in \mathit{uco}(\wp(\Sigma^+))$, which is the least closure including the a suitable set of execution traces revealing the watermark $X_s \subseteq \Sigma^+$, for some signature $s \in \mathcal{S}$. It is immediate to recognize a $\mathcal{F}$-completeness transformation here: *a stegoprogram reveals the watermark* $\omega = \mathfrak{M}[\![s]\!]$ *under input* η *if its abstract semantics is* $\mathcal{F}$*-complete for* ω *and* η. In this case the abstract semantics $[\![\cdot]\!]^\alpha$ performs *watermark extraction* which can be, as in Cousot and Cousot (2004), implemented as abstract interpretation. Therefore $\mathfrak{S}(s,\mathtt{P})$ is a stegoprogram if its abstract semantics $[\![\cdot]\!]^\alpha$ is a $\mathscr{F}$-complete transformation of the abstract semantics of P w.r.t. the input observation η and the output observation $\mathfrak{M}[\![s]\!]$, recognizing the watermarks. This means that by formalizing and characterizing completeness transformers we obtain a framework where we can understand, study and even generalize well-known existing watermarking techniques.

Note that if $\langle \eta, \mathfrak{M}[\![s]\!] \rangle$ is $\mathscr{F}$-complete for $[\![\mathfrak{S}(s,\mathtt{P})]\!]^\alpha$ it may happen that $\langle \mathfrak{M}[\![s]\!], \eta \rangle$ is not $\mathscr{F}$-complete for $[\![\mathfrak{S}(s,\mathtt{P})]\!]$ (Giacobazzi *et al.* 2000). This means that the knowledge of the stegomarker may not be sufficient in order to extract the watermark. This makes the extraction completely dependent on the suitable choice of the abstract semantics $[\![\cdot]\!]^\alpha$. In this sense, code obfuscation can be used in order to design appropriate stegolayers making $\langle \eta, \omega \rangle$ incomplete for the standard interpreter $[\![\cdot]\!]$ (e.g. see Giacobazzi *et al.* (2012)). This is a further weakening with respect to abstract watermarking in Cousot and Cousot (2004), where $\omega = \alpha$ and the secrecy relies upon the difficulty to guess ω out of any blind static or dynamic analysis of the stegoprogram.

It is easy to provide the intuition of how it is possible to encode, within this schema, most well-known watermarking methods. This of course does not reduce the impact

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

of these methods but rather shows a common invariant pattern in most methods for information hiding in programs, which is precisely modelled by fixing some abstractions, which play here the role of the watermark recognition process and then making the cover program complete with respect to these abstractions, letting the corresponding abstract interpretation be a valid watermark extractor. We sketch two of these encodings for popular watermarking methods, see Collberg and Nagra (2010) for details, showing how they both correspond to program (semantics) modifications inducing $\mathscr{F}$-completeness with respect to an observer (the watermark extractor) and a key (the key for watermark extraction).

Abstract watermarking. Abstract watermarking (Cousot and Cousot 2002) is immediate. In abstract watermarking, a signature is a *large* number $c \in \mathbb{N}$ which is embedded in the program as a tuple of smaller numbers $\langle c_1, \ldots, c_n \rangle \in \mathbb{N}^n$, where each c_i is a natural number modulo a prime number p_i, i.e. c_i ranges in $[0, p_i - 1]$. The tuple of relatively prime numbers $\langle p_1, \ldots, p_n \rangle$ is the secret key. $\langle c_1, \ldots, c_n \rangle$ is the watermark which is embedded in the cover program P by including a stegomarker which is a set of program modules given, for each c_i, by the code computing (by the Horner's method) two polynomials R_i and Q_i such that $c_i \equiv_{p_i} R_i(I)$ for some input value $I \in \mathbb{N}$ and $c_i \equiv_{p_i} Q_i(c_i)$, where $x \equiv_p y$ if $x = y + kp$ and $k \in \mathbb{Z}$. The watermark extraction is obtained by extracting the invariants of the computed polynomials. Because the numbers computed in $\mathbb{N}$ by the embedded polynomials may appear random, this can only be achieved by a constant propagation analysis in an abstract domain $\alpha_i \in \mathit{uco}(\wp(\mathbb{N}))$ of values representing congruences modulo p_i, i.e. $\alpha_i = \{\mathbb{N}, 0, \ldots, p_i - 1, \varnothing\}$. The abstract domain $\alpha = \prod_{i=1}^{n} \alpha_i$ is therefore unambiguously determined by the secret key $\langle p_1, \ldots, p_n \rangle$. In this case, the obtained stegoprogram $\mathfrak{S}(s, \mathtt{P})$ is such that $[\![\mathfrak{S}(s, \mathtt{P})]\!]^\alpha$ is a complete transformation of $[\![\mathtt{P}]\!]^\alpha$ w.r.t. η and α, where $\eta = \{\mathbb{N}, I\} \in \mathit{uco}(\wp(\mathbb{N}))$ is the closure determining whether the (input) value is I. A similar notion of abstract watermarking can be found in Mastroeni (2004).

Block reordering. Let $\mathcal{G}$ be the set of all possible directed graphs which can be obtained from the *basic blocks* (or any given partition of code statements) of the cover program P. Assume that the concrete semantics extracts program execution traces in Σ^+ where each program state includes the executed statements of P. Let also $\mathrm{CFG}(\sigma) \in \mathcal{G}$, for $\sigma \in \Sigma^+$, be the graph of basic blocks visited in σ. The block reordering static watermarking corresponds to choose: $\eta = \mathit{id}$ (static); given a (numerical) signature s and an encoding of numbers as graphs, i.e. sequences of basic blocks $\mathcal{E} : \mathbb{N} \longrightarrow \mathbb{G}$ then $\mathfrak{M}[\![s]\!]$ is the atomic closure $\{\Sigma^+, \mathcal{G}_s\} \in \mathit{uco}(\wp(\Sigma^+))$ where $\mathcal{G}_s = \{\sigma \in \Sigma^+ \mid \mathcal{E}(s) = \mathrm{CFG}(\sigma)\}$ and $[\![P]\!]^\alpha$ extracts the CFG of P (Giacobazzi *et al.* 2012), which is an (incomplete) abstract interpretation of the trace semantics $[\![P]\!]$ provided that states include code instructions with labels. The abstraction α forgets about memory locations and computed values and just keeps track of the sequence of program instructions isolating basic blocks (consecutive instructions) as graph nodes and determining possible jumps between blocks as graph edges (Rival and Mauborgne 2007). Incompleteness results here from the inability of $[\![P]\!]^\alpha$ in distinguishing true or false branches and iterations (i.e. dead code), being a purely static extraction of the control flow graph of P. A dynamic version of block reordering can be implemented by choosing $\eta \neq \mathit{id}$. In this case

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

$\mathfrak{S}(s, P)$ has to include within the stegomarker a block reordering algorithm, which is activated when x is a fix-point of η, as in metamorphic malware (Dalla Preda *et al.* 2007). A similar model, with analogous abstractions can be used for encoding the Venkatesan *et al.* CFG-based watermarking (Venkatesan *et al.* 2001).

Other watermarking methods can be encoded as making a transformed program complete for an abstraction. For instance, the *Threading watermarking* in Nagra and Thomborson (2004) would need a different computational model, including multithreading and concurrency. In this case the extractor should correspond to a complete abstract interpretation modelling execution paths, which encode the watermark in the sequence of thread interleavings. *Credibility*, *data-rate*, *resilience* and *stealthy* (Collberg and Thomborson 2000) rely upon the choice of the properties α and $\mathfrak{M}[\![s]\!]$. High credibility corresponds to $\alpha, \mathfrak{M}[\![s]\!] \in uco(\wp(\Sigma^+))$ such that $[\![\mathtt{P}]\!]^\alpha \notin \mathfrak{M}[\![s]\!]$ (i.e. $\mathfrak{M}[\![s]\!]([\![\mathtt{P}]\!]^\alpha) \approx \Sigma^+$ minimizes false positives). Data-rate depends upon the choice of $\mathfrak{M}[\![s]\!]$. Resilience is high when $\mathfrak{M}[\![s]\!]$, and therefore α, are both hard to guess and they are preserved by most common program transformations. Stealthy instead depends upon the implementation of the stegolayer, which has to produce output code which is as similar as possible to P. Note that, the $\mathscr{F}$-completeness transformers will be formalized as idempotent transformations, hence they provide also a code tamper-detection method similar to the one used for images in mathematical morphology (Kihara *et al.* 2007).

4. Making semantics forward complete

In this section, we face the problem of (minimally) transforming semantics in order to make them $\mathscr{F}$-complete. Minimal here is defined on the approximation order, namely we characterize the transformation with the least loss of precision w.r.t. the original semantics outputs, both adding noise or removing information. The transformation is made in two steps: first we induce $\mathscr{F}$-completeness, and then we force monotonicity by using standard results on function transformers in Cousot and Cousot (1979a). This is because the completeness transformation may generate nonmonotone functions. Before introducing the completeness transformers we have to formally specify what we mean by *minimally transforming semantics*. As usual we consider a lattice of functions where maps are point-wise ordered. Hence, a minimal transformation of f finds the closest function, by reducing or increasing the images of f, w.r.t. a given property we want to hold for f, which is in this context, $\mathscr{F}$-completeness. In abstract interpretation this corresponds to find the closest (viz., least abstraction or concretization) of the semantics such that $\mathscr{F}$-completeness holds for a given pair of abstractions. Consider a function $f : C \longrightarrow C$ and consider two abstract domains $\eta, \rho \in uco(C)$, we are interested in transforming f in order to make $\langle \eta, \rho \rangle$ $\mathscr{F}$-complete for f, namely in order to force the equation $\rho f \eta = f \eta$ to hold.

4.1. *Order-theoretic $\mathscr{F}$-complete semantics*

We first observe that the set of all the $\mathscr{F}$-complete functions with respect to two given abstractions η and ρ, namely the set $\{h : C \xrightarrow{\text{m}} C \mid \rho h \eta = h \eta\}$, has a least element. The

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

following theorem proves exactly this fact, implying that we can always minimally increase a given monotone function f in order to induce completeness.

Theorem 4.1. Given $\eta, \rho \in \mathit{uco}(C)$ the set $\{f : C \longrightarrow C \mid \rho f \eta = f \eta\}$ is a Moore family on $\langle C \longrightarrow C, \sqsubseteq \rangle$.

Proof. Let us prove that the set $F \stackrel{\text{def}}{=} \{f : C \longrightarrow C \mid \rho f \eta = f \eta\}$ is closed under greatest lower bound, namely it has minimum. We prove that it is closed under infinitary *glb*. Consider $\{f_i\}_i \subseteq F$, we have to prove that also $\bigsqcap_i f_i \in F$, i.e. $\rho \bigsqcap_i f_i \, \eta = \bigsqcap_i f_i \, \eta$. First of all we recall that, by properties of uco, $\rho(\wedge \rho(x)) = \wedge \rho(x)$ for any $\rho \in \mathit{uco}(C)$. Let $x \in C$,

$$\begin{array}{rcll} \bigsqcap_i f_i \, \eta(x) & = & \bigwedge_i f_i \eta(x) & \text{By def. of } \sqcap \\ & = & \bigwedge_i \rho f_i \eta(x) & \text{By complet. hyp. on } f_i \\ & = & \rho(\bigwedge_i \rho f_i \eta(x)) & \text{Since } \rho(\wedge \rho(x)) = \wedge \rho(x) \\ & = & \rho(\bigwedge_i f_i \eta(x)) & \text{By complet. hyp. on } f_i \\ & = & \rho \bigsqcap_i f_i \, \eta(x) & \end{array}$$

□

Corollary 4.2. Given $\eta, \rho \in \mathit{uco}(C)$ the set $\{f : C \xrightarrow{\text{m}} C \mid \rho f \eta = f \eta\}$ of monotone functions is Moore family on $\langle C \xrightarrow{\text{m}} C, \sqsubseteq \rangle$.

Proof. Immediate by Theorem 4.1 because the *glb* of monotone functions is monotone. □

On the other hand we observe that, under additivity hypothesis on ρ, the set of all the complete functions has also the greatest element, implying that we can always minimally decrease a given monotone function f in order to induce $\mathscr{F}$-completeness.

Theorem 4.3. Given $\eta, \rho \in \mathit{uco}(C)$ the set $\{f : C \longrightarrow C \mid \rho f \eta = f \eta\}$ is a dual Moore family on the domain $\langle C \longrightarrow C, \sqsubseteq \rangle$ iff ρ is additive.

Proof. Let us prove that the set $F \stackrel{\text{def}}{=} \{f : C \longrightarrow C \mid \rho f \eta = f \eta\}$ is closed under least upper bound. We prove that it is closed under infinitary *lub*. Consider $\{f_i\}_i \subseteq F$, we have to prove that also $\bigsqcup_i f_i \in F$, i.e. $\rho \bigsqcup_i f_i \, \eta = \bigsqcup_i f_i \, \eta$. Let $x \in C$ and suppose ρ additive. Then the following equalities hold

$$\begin{array}{rcll} \bigsqcup_i f_i \, \eta(x) & = & \bigvee_i f_i \eta(x) & \text{By def. of } \sqcup \\ & = & \bigvee_i \rho f_i \eta(x) & \text{By complet. hyp. on } f_i \\ & = & \rho(\bigvee_i f_i \eta(x)) & \text{By additivity of } \rho \\ & = & \rho \bigsqcup_i f_i \, \eta(x) & \end{array}$$

Hence we have completeness.

Suppose now that completeness holds for *lub* of complete functions, we have to prove that ρ is additive. We prove that it is infinitary additive, i.e. we prove that for any $x_i \in C$, $\rho(\bigvee_i x_i) = \bigvee_i \rho(x_i)$. Let us define the functions $f_x \stackrel{\text{def}}{=} \lambda z.\rho(x)$, then these functions are trivially forward complete, so it is their *lub* by hypothesis. Moreover, let us recall that for

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

any uco ρ we have that $\rho(\vee\rho(x)) = \rho(\vee x)$ (*). Then the following equalities holds:

$$\begin{array}{rcll}
\rho(\bigvee_i x_i) & = & \rho(\bigvee_i \rho(x_i)) & \text{By Equation (*)} \\
& = & \rho(\bigvee_i f_{x_i}\eta(z)) & \text{By Definition of } f_{x_i} \\
& = & \rho \bigsqcup_i f_{x_i}\, \eta(z) & \\
& = & (\bigsqcup_i f_{x_i})\, \eta(z) & \text{By hypothesis} \\
& = & \bigvee_i f_{x_i}\rho(z) & \\
& = & \bigvee_i \rho(x_i) & \text{By Definition of } f_{x_i}
\end{array}$$

□

Corollary 4.4. Given $\eta, \rho \in uco(C)$ the set $\{f : C \xrightarrow{m} C \mid \rho f\eta = f\eta\}$ is a dual Moore family on the domain $\langle C \xrightarrow{m} C, \sqsubseteq\rangle$ iff ρ is additive.

Proof. By Theorem 4.3 since the *lub* of monotone functions is monotone. □

4.2. *Functional extension towards $\mathscr{F}$-completeness*

In this section, we aim to characterize the operator associating with each function the closest (from above) function for which a given pair of abstract domains is $\mathscr{F}$-complete for this function. This is indeed a closure operator on the lattice of monotone functions, point-wise ordered. For any $f \in C \xrightarrow{m} C$ and $\eta, \rho \in uco(C)$, let us define:

$$\mathbb{F}^{\uparrow}_{\eta,\rho} \stackrel{\text{def}}{=} \lambda f.\lambda x. \begin{cases} \rho f(x) & \text{if } x \in \eta \\ f(x) & \text{otherwise} \end{cases}$$

We can observe that, $\mathscr{F}$-completeness is checked only on η-closed elements, i.e. on fix-points of η. The intuition behind this transformation is that we force all these η-closed elements to be mapped into ρ (namely $\mathbb{F}^{\uparrow}_{\eta,\rho}(x) = \rho f(x)$), obtaining trivially completeness. The interesting aspect is that this, quite straightforward, transformation is the *minimal* complete extension of f.

Lemma 4.5. Let $f : C \xrightarrow{m} C$. Then $\mathbb{F}^{\uparrow}_{\eta,\rho}(f)$ is $\mathscr{F}$-complete and

$$\mathbb{F}^{\uparrow}_{\eta,\rho}(f) = \bigsqcap\{h : C \longrightarrow C \mid f \sqsubseteq h,\ \rho h\eta = h\eta\}.$$

Proof. Let $A \stackrel{\text{def}}{=} \{h : C \longrightarrow C \mid f \sqsubseteq h,\ \rho h\eta = h\eta\}$, and let $f^* \stackrel{\text{def}}{=} \mathbb{F}^{\uparrow}_{\eta,\rho}(f)$.
First of all we prove that $f^* \in A$, namely that $f^* \sqsupseteq f$ and $\rho f^*\eta = f^*\eta$. Clearly the fact that $f^* \sqsupseteq f$, where $h_1 \sqsupseteq h_2$ if $\forall x\,.\, h_1(x) \geqslant h_2(x)$, trivially holds since $\rho \in uco(C)$. Consider $\rho f^*\eta = f^*\eta$:

$$\rho(f^*(\eta(x))) = \rho(\rho(f(\eta(x)))) = \rho(f(\eta(x))) = f^*(\eta(x)).$$

Therefore $f^* \in A$, let us prove that it is the *glb*, namely for each $g : C \longrightarrow C$ such that $g \sqsupseteq f$ and $\rho g\eta = g\eta$, then $g \sqsupseteq f^*$. If $x \notin \eta$, then $f^*(x) = f(x)$, therefore $g(x) \geqslant f(x) = f^*(x)$. Consider $x \in \eta$, namely such that $\eta(x) = x$:

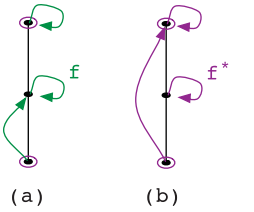
$$\begin{array}{l}
g(x) \geqslant f(x) \ \Rightarrow\ g(x) \geqslant f(\eta(x)) \ \Rightarrow \text{(By completeness hyp.)} \\
g(x) \geqslant \rho(f(\eta(x)) \ \Rightarrow\ g(x) \geqslant \rho(f(x)) = f^*(x).
\end{array}$$

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

Note that, if f is complete, i.e. $f\eta = \rho f\eta$, we have that, when $x \in \eta$, $f^*(x) = \rho f(x)$ and being $x \in \eta$ we have $\rho f(x) = \rho f\eta(x)$. Hence by completeness we conclude that $f^*(x) = \rho f\eta(x) = f\eta(x) = f(x)$. □

It is worth noting that $\mathbb{F}^{\uparrow}_{\eta,\rho}(f)$ may not be monotone, as shown in the following example.

Example 4.6. Consider the lattices depicted on the right. The circled points are those in ρ and the arrows in the picture (a) represent f. The arrows in the picture (b) are those of the map obtained from f by applying $\mathbb{F}^{\uparrow}$, which is clearly not monotone.



The lack of monotonicity is due to the fact that, in order to minimally transform f, only the images of the elements in η are modified, leaving unchanged the images of all the other elements. Indeed monotonicity fails when we check it between the transformed image of one element in η and one taken outside η. We consider the basic transformer $\mathbb{M}^{\uparrow}$ (see Section 2.1) for finding the closest *monotone* transformation of f which is $\mathscr{F}$-complete for the pair of abstractions $\eta, \rho \in uco(C)$. Next lemma proves that $\mathbb{M}^{\uparrow}$ preserves $\mathscr{F}$-completeness, namely $\mathbb{M}$ does not destroy the transformations made by $\mathbb{F}$ towards completeness.

Lemma 4.7. For any $\eta, \rho \in uco(C)$ we have $\mathbb{F}^{\uparrow}_{\eta,\rho}\mathbb{M}^{\uparrow}\mathbb{F}^{\uparrow}_{\eta,\rho} = \mathbb{M}^{\uparrow}\mathbb{F}^{\uparrow}_{\eta,\rho}$

Proof. For the sake of readability let us simply use $\mathbb{F}^{\uparrow}$, omitting the closure operators η and ρ.

$$\begin{aligned}
\mathbb{M}^{\uparrow}\mathbb{F}^{\uparrow}(f) &= \lambda x.\ \bigvee\{\mathbb{F}^{\uparrow}(f)(y) |\ y \leqslant x\} \\
&= \lambda x.\ \bigvee\{\rho f(y) |\ y \leqslant x, y \in \eta\} \vee \bigvee\{f(y) |\ y \leqslant x, y \notin \eta\} \\
&= \lambda x.\ \begin{cases} \rho f(x) \vee \bigvee\{f(y) |\ y \leqslant x, y \notin \eta\} & x \in \eta \\ \bigvee\{\rho f(y) |\ y \leqslant x, y \in \eta\} \vee f(x) & x \notin \eta \end{cases} \quad (*) \\
&= \lambda x.\ \begin{cases} \rho f(x) & x \in \eta \\ \bigvee\{\rho f(y) |\ y \leqslant x, y \in \eta\} \vee f(x) & x \notin \eta \end{cases} \quad (**)
\end{aligned}$$

where the step $(*)$ holds because if $x \in \eta$ then $\rho f(x) \in \{\rho f(y) |\ y \leqslant x, y \in \eta\}$, hence $\rho f(x) \leqslant \bigvee\{\rho f(y) |\ y \leqslant x, y \in \eta\}$, while by monotonicity of ρ and f we have $\rho f(x) \geqslant \bigvee\{\rho f(y) |\ y \leqslant x, y \in \eta\}$. On the other hand, if $x \notin \eta$ then $f(x) \in \{f(y) |\ y \leqslant x, y \notin \eta\}$, hence $f(x) \leqslant \bigvee\{f(y) |\ y \leqslant x, y \notin \eta\}$, while the other inclusion holds by f monotonicity. Step $(**)$ holds because $\bigvee\{f(y) |\ y \leqslant x, y \notin \eta\} \leqslant f(x)$, and therefore we have that

$$\rho f(x) \vee \bigvee\{f(y) |\ y \leqslant x, y \notin \eta\} = \rho f(x) \vee f(x) = \rho f(x).$$

Hence, we can compute the complete transformation of the map above

$$\begin{aligned}
\mathbb{F}^{\uparrow}\mathbb{M}^{\uparrow}\mathbb{F}^{\uparrow}(f) &= \mathbb{F}^{\uparrow}\left(\lambda x.\ \begin{cases} \rho f(x) & x \in \eta \\ \bigvee\{\rho f(y) |\ y \leqslant x, y \in \eta\} \vee f(x) & x \notin \eta \end{cases}\right) \\
&= \lambda x.\ \begin{cases} \rho\rho f(x) = \rho f(x) & x \in \eta \\ \bigvee\{\rho f(y) |\ y \leqslant x, y \in \eta\} \vee f(x) & x \notin \eta \end{cases} \\
&= \mathbb{M}^{\uparrow}\mathbb{F}^{\uparrow}(f)
\end{aligned}$$

□

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

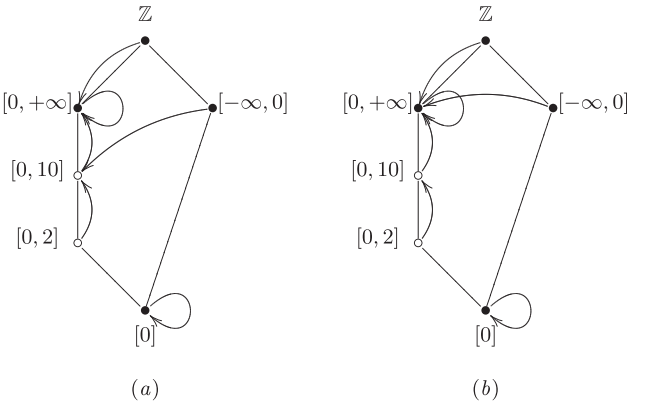


Fig. 5. The abstract domain S with sq and $\mathcal{F}^{\uparrow}_{\rho_c,\rho_c}(sq)$.

The following theorem shows that the *minimal monotone transformation for $\mathscr{F}$-completeness* of a monotone function is:

$$\mathcal{F}^{\uparrow}_{\eta,\rho} \stackrel{\text{def}}{=} \lambda f.\, \mathbb{M}^{\uparrow}\mathbb{F}^{\uparrow}_{\eta,\rho}(f).$$

Theorem 4.8. If $f : C \xrightarrow{\text{m}} C$ then $\mathcal{F}^{\uparrow}_{\eta,\rho}(f) = \bigsqcap\{h : C \xrightarrow{\text{m}} C \mid f \sqsubseteq h,\ \rho h\eta = h\eta\}$.

Proof. Straightforward consequence of Lemmas 4.5 and 4.7. □

The construction presented so far for obtaining $\mathscr{F}$-completeness focuses first on the idea of inducing completeness by guaranteeing extensivity ($\mathbb{F}^{\uparrow}$) and then by forcing monotonicity ($\mathbb{M}^{\uparrow}$). It is worth noting that these two transformers deal with different aspects of f:

- $\mathbb{F}^{\uparrow}$ transforms η-closed elements on which $\mathscr{F}$-completeness fails, i.e. $x \in \eta.\, \rho f\eta(x) \neq f\eta(x)$ implies that $\mathbb{F}^{\uparrow}(f)(x) > f(x)$.
- $\mathbb{M}^{\uparrow}$ transforms only those elements x outside η which are above an incomplete point of η, i.e. $y \in \eta.\, x \leqslant y$ and $\rho f\eta(y) \neq f\eta(y)$. This implies $\forall x \in \eta.\, \mathbb{M}^{\uparrow}\mathbb{F}^{\uparrow}(f)(x) = \mathbb{F}^{\uparrow}(f)(x)$.
- All the other elements are left unchanged.

In particular, we can note that:

$$\forall x \in \eta.\, \mathbb{F}^{\uparrow}(f)(x) = \rho f\eta(x)$$

This observation is interesting because it means that the closest $\mathscr{F}$-completeness transformation (by ignoring extensivity and monotonicity) of f is indeed its *best correct approximation.*

Example 4.9. Consider the concrete domain S in Figure 4, an erroneous square operation sq' and an abstraction $\rho_c = \{\mathbb{Z}, [-\infty,0], [0,+\infty], [0]\}$, depicted in Figure 5 with black dots. Figure 5 shows the transformation of sq' induced by $\mathcal{F}^{\uparrow}_{\rho_c,\rho_c}(sq')$. Note that $\mathcal{F}^{\uparrow}_{\rho_c,\rho_c}(sq') = sq^{\sharp}$.

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

4.3. *Preserving monotonicity and $\mathscr{F}$-completeness towards extension*

In this section, we characterize the closest *monotone* transformation of a function f that induces $\mathscr{F}$-completeness, and which is extensive. Let us define the following function:

$$\mathbb{F}^{\mathbf{E}}_{\eta,\rho} \stackrel{\text{def}}{=} \lambda f.\, \lambda x.\, \rho f \eta^{+}(x)$$

where η^{+} is the right adjoint of η. This transformation transforms incomplete elements of η exactly as it was done by the previous transformer, i.e. being $\eta^{+}(x) = x$ for each $x \in \eta$, we have

$$\forall x \in \eta.\, \mathbb{F}^{\mathbf{E}}_{\eta,\rho}(f)(x) = \rho f(x) = \mathbb{F}^{\uparrow}_{\eta,\rho}(f)(x) = \mathbb{M}^{\uparrow}\mathbb{F}^{\uparrow}(f)(x)$$

while it transforms all the other elements in order to preserve monotonicity, in particular for all $x \notin \eta$ above an incomplete point, the transformation coincides with $\mathbb{M}^{\uparrow}\mathbb{F}^{\uparrow}$, while for all the other elements we have that $\mathbb{F}^{\mathbf{E}}(f)$ is incomparable with f. The following results tells us that the order of composition of extensivity (recall that $\mathbb{E}$ forces extensivity and it is defined in Section 2.1) and monotonicity is irrelevant.

Proposition 4.10. If $\eta \in \mathit{uco}(C)$ is additive then

$$\mathcal{F}^{\uparrow}_{\eta,\rho}(f) \stackrel{\text{def}}{=} \mathbb{M}^{\uparrow}\mathbb{F}^{\uparrow}(f) = \mathbb{E}\mathbb{F}^{\mathbf{E}}(f) = \lambda x.\, \rho f \eta^{+}(x) \vee f(x).$$

Proof. By Lemma 4.7 we know that

$$\mathcal{F}^{\uparrow}_{\eta,\rho}(f) \stackrel{\text{def}}{=} \lambda f.\, \mathbb{M}^{\uparrow}\mathbb{F}^{\uparrow}_{\eta,\rho}(f) = \lambda x. \begin{cases} \rho f(x) & x \in \eta \\ \bigvee\{\rho f(y) |\ y \leqslant x, y \in \eta\} \vee f(x) & x \notin \eta \end{cases}$$

Let us prove that this function corresponds to $\lambda x.\, \rho f \eta^{+}(x) \vee f(x)$.
If $x \in \eta$, then $x = \eta(x)$, moreover we recall that $\eta^{+}\eta(x) = \eta(x)$, therefore

$$\rho f \eta^{+} \eta(x) \vee f \eta(x) = \rho f \eta(x) \vee f \eta(x) = \rho f(x)$$

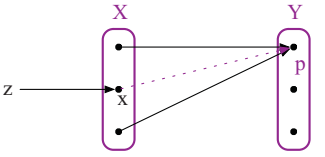
being $\rho f \eta(x) \geqslant f \eta(x) = f(x)$.

Consider $x \notin \eta$. We can prove that $\bigvee\{\rho f(y) |\ y \leqslant x, y \in \eta\} \vee f(x) = \rho f \eta^{+}(x) \vee f(x)$, in particular we prove that $\bigvee\{\rho f(y) |\ y \leqslant x, y \in \eta\} = \rho f \eta^{+}(x)$. Being $\eta^{+}(x) \leqslant x$ and, by construction, $\eta^{+}(x) \in \eta$, we have that $\rho f \eta^{+}(x) \in \{\rho f(y) |\ y \leqslant x, y \in \eta\}$ and therefore $\rho f \eta^{+}(x) \leqslant \bigvee\{\rho f(y) |\ y \leqslant x, y \in \eta\}$. On the other hand, for each $y \in \eta$ such that $y \leqslant x$ we have $y \leqslant \eta^{+}(x)$, being $\eta^{+}(x)$ the greatest fix-point of η less or equal to x by construction. Hence $\rho f(y) \leqslant \rho f \eta^{+}(x)$ for each such y, therefore $\bigvee\{\rho f(y) |\ y \leqslant x, y \in \eta\} \leqslant \rho f \eta^{+}(x)$. □

Example 4.11. Let us consider transition functions on a transition system as shown in the introduction. In this case, we have one more constraint since the resulting function has to be additive in order to be the transition relation of a transition system. In order to obtain an additive function we simply transform the image of singletons and we obtain the whole function by additive lift. It is clear that we cannot use the transformation of Section 4.2 since in that transformation we only modify the abstract elements that usually (for all the abstractions different from the identity) are not the singletons. Hence, we have to consider the transformation $\mathbb{F}^{\mathbf{E}}_{\eta,\rho}$. Consider for instance Figure 1. In this case $\eta = \rho$ is the equivalence relation depicted in the picture, and clearly η^{+} applied to singletons

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

is the identity, hence on transition systems, by denoting $\mathtt{R}$ the equivalence relation, the transformation become $\lambda x \in C.\, [f(x)]_{\mathtt{R}}$ where given a set X, we define $[X]_{\mathtt{R}} = \bigcup_{x \in X} [x]_{\mathtt{R}}$. This transformation, which is unique, corresponds to the following transition system



Even if in this example we obtain a minimal transformation, we have to observe that, in general, we lose optimality because we transform the image of an abstract state (i.e. a set of states) by transforming the image of all the singletons states in the set.

4.4. *Functional reduction towards $\mathscr{F}$-completeness*

In this section, we characterize the operator transforming each function into the closest one (from below) for which a given pair of domains is $\mathscr{F}$-complete. For any $f : C \xrightarrow{\text{m}} C$ and $\eta, \rho \in \mathit{uco}(C)$, with ρ additive, let us define:

$$\mathbb{F}^{\downarrow}_{\eta,\rho} \stackrel{\text{def}}{=} \lambda f.\lambda x. \begin{cases} \rho^{+} f(x) & \text{if } x \in \eta \\ f(x) & \text{otherwise} \end{cases}$$

Intuitively, ρ^{+} is the operator associating with a generic element z the greatest fix-point of ρ smaller than z. Hence, it can be used to restrict the f-image of x in order to be mapped precisely in the greatest ρ-closed element contained in $f(x)$, when x is η-closed, forcing again completeness.

Lemma 4.12. Let $f : C \xrightarrow{\text{m}} C$, then $\mathbb{F}^{\downarrow}_{\eta,\rho}(f)$ is $\mathscr{F}$-complete and

$$\mathbb{F}^{\downarrow}_{\eta,\rho}(f) = \bigsqcup \{ h : C \longrightarrow C \mid f \sqsupseteq h,\ \rho h \eta = h \eta \}$$

Proof. Let A be $\{ h : C \longrightarrow C \mid f \sqsupseteq h,\ \rho h \eta = h \eta \}$, and $f^{*} \stackrel{\text{def}}{=} \mathbb{F}^{\downarrow}_{\eta,\rho}(f)$. We first prove that $f^{*} \in A$, namely that $f^{*} \sqsubseteq f$ and $\rho f^{*} \eta = f^{*} \eta$. Clearly $f^{*} \sqsubseteq f$ holds trivially since $\rho^{+} \in \mathit{lco}(C)$, being the adjoint function of an uco. Consider $\rho f^{*} \eta = f^{*} \eta$:

$$\rho(f^{*}(\eta(x))) = \rho(\rho^{+}(f(\eta(x)))) = \rho^{+}(f(\eta(x))) \text{ [By Theorem 2.1(1)]} = f^{*}(\eta(x))$$

Finally, we have to prove that f^{*} is the *lub* of A, namely for each $g \in A$ we have $g \sqsubseteq f^{*}$. If $x \notin \eta$ then $f^{*}(x) = f(x)$, and therefore $g(x) \leqslant f(x) = f^{*}(x)$. Consider $x \in \eta$, namely $x = \eta(x)$:

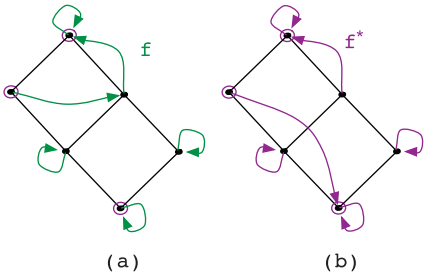
$$\begin{array}{l} g(x) \leqslant f(x) \ \Rightarrow\ \rho^{+}(g(x)) \leqslant \rho^{+}(f(x)) \ \Rightarrow \\ \rho^{+}(g(\eta(x))) \leqslant f^{*}(x) \ \Rightarrow\ \rho^{+}(\rho(g(\eta(x)))) \leqslant f^{*}(x) \text{ (By complet.)} \Rightarrow \\ \rho(g(x)) \leqslant f^{*}(x) \text{ (By Theorem 2.1(1) and } x \in \eta) \ \Rightarrow \\ g(x) \leqslant \rho(g(x)) \leqslant f^{*}(x)) \end{array}$$

If f is complete, then we have that when $x \in \eta$, $f^{*}(x) = \rho^{+} f(x) = \rho^{+} f \eta(x)$, hence by completeness $f^{*}(x) = \rho^{+} \rho f \eta(x) = \rho f \eta(x) = f \eta(x) = f(x)$. □

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

Also in this case, the transformed function may fail monotonicity as shown in the next example.

Example 4.13. Consider the lattices depicted on the right. The circled points are those in ρ and the arrows on picture (a) are those of f. The arrows on the picture (b) are those of the map obtained from f by means of $\mathbb{F}^{\downarrow}$, and this map is clearly not monotone.



Again, the lack of monotonicity is due to the fact that, for the sake of minimality, the transformer changes the f-image of only some elements, those in η. Again we apply the monotonicity transformer, in this case $\mathbb{M}^{\downarrow}$, for finding the best *monotone* transformation of f. As above, the next theorem shows that it is not necessary a fix-point transformation since the monotone transformer does not change the $\mathscr{F}$-completeness of functions obtained by $\mathbb{F}^{\downarrow}$.

Lemma 4.14. For any $\eta, \rho \in uco(C)$ we have $\mathbb{F}^{\downarrow}_{\eta,\rho}\mathbb{M}^{\downarrow}\mathbb{F}^{\downarrow}_{\eta,\rho} = \mathbb{M}^{\downarrow}\mathbb{F}^{\downarrow}_{\eta,\rho}$.

Proof. For the sake of readability let us simply use $\mathbb{F}^{\uparrow}$, omitting the closure operators η and ρ.

$$\begin{aligned}
\mathbb{M}^{\downarrow}\mathbb{F}^{\downarrow}(f) &= \lambda x.\ \bigwedge\{\mathbb{F}^{\downarrow}(f)(y) |\ y \geqslant x\} \\
&= \lambda x.\ \bigwedge\{\rho^{+}f(y) |\ y \geqslant x, y \in \eta\} \wedge \bigwedge\{f(y) |\ y \geqslant x, y \notin \eta\} \\
&= \lambda x.\ \begin{cases} \rho^{+}f(x) \wedge \bigwedge\{f(y) |\ y \geqslant x, y \notin \eta\} & x \in \eta \\ \bigwedge\{\rho^{+}f(y) |\ y \geqslant x, y \in \eta\} \wedge f(x) & x \notin \eta \end{cases} \qquad (*) \\
&= \lambda x.\ \begin{cases} \rho^{+}f(x) & x \in \eta \\ \bigwedge\{\rho^{+}f(y) |\ y \geqslant x, y \in \eta\} \wedge f(x) & x \notin \eta \end{cases} \qquad (**)
\end{aligned}$$

where the step $(*)$ holds because if $x \in \eta$ then $\rho^{+}f(x) \in \{\rho^{+}f(y) |\ y \leqslant x, y \in \eta\}$, hence $\rho^{+}f(x) \geqslant \bigwedge\{\rho^{+}f(y) |\ y \geqslant x, y \in \eta\}$, while by monotonicity of ρ^{+} and f we have $\rho^{+}f(x) \leqslant \bigwedge\{\rho^{+}f(y) |\ y \geqslant x, y \in \eta\}$. On the other hand, if $x \notin \eta$ then $f(x) \in \{f(y) |\ y \geqslant x, y \notin \eta\}$, hence $f(x) \geqslant \bigwedge\{f(y) |\ y \geqslant x, y \notin \eta\}$, while the other inclusion holds by monotonicity of f. Step $(**)$ holds since $\bigwedge\{f(y) |\ y \geqslant x, y \notin \eta\} \geqslant f(x)$, and therefore we have $\rho^{+}f(x) \wedge \bigwedge\{f(y) |\ y \geqslant x, y \notin \eta\} = \rho^{+}f(x) \wedge f(x) = \rho^{+}f(x)$. Hence, we can compute the complete transformation of the map above

$$\begin{aligned}
\mathbb{F}^{\downarrow}\mathbb{M}^{\downarrow}\mathbb{F}^{\downarrow}(f) &= \mathbb{F}^{\downarrow}\left(\lambda x.\ \begin{cases} \rho^{+}f(x) & x \in \eta \\ \bigwedge\{\rho^{+}f(y) |\ y \geqslant x, y \in \eta\} \wedge f(x) & x \notin \eta \end{cases}\right) \\
&= \lambda x.\ \begin{cases} \rho^{+}\rho^{+}f(x) = \rho^{+}f(x) & x \in \eta \\ \bigwedge\{\rho^{+}f(y) |\ y \geqslant x, y \in \eta\} \wedge f(x) & x \notin \eta \end{cases} \\
&= \mathbb{M}^{\downarrow}\mathbb{F}^{\downarrow}(f).
\end{aligned}$$

□

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

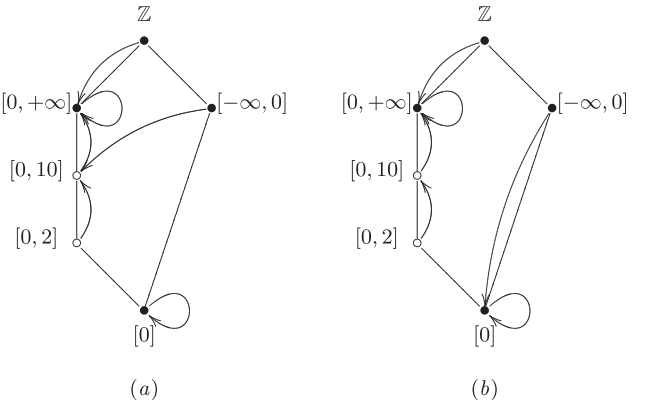


Fig. 6. The abstract domain S with sq' and $\mathcal{F}^{\downarrow}_{\rho_c,\rho_c}(sq')$.

The next result states that the minimal *monotone* transformation of a function towards completeness is defined as $\mathcal{F}^{\downarrow}_{\eta,\rho} \stackrel{\text{def}}{=} \lambda f.\, \mathbb{M}^{\downarrow}\mathbb{F}^{\downarrow}(f)$.

Theorem 4.15. Let $f : C \xrightarrow{\text{m}} C$, then

$$\mathcal{F}^{\downarrow}_{\eta,\rho}(f) = \bigsqcup\{h : C \xrightarrow{\text{m}} C \mid f \sqsupseteq h,\ \rho h\eta = h\eta\}.$$

Proof. Straightforward consequence of Lemmas 4.12 and 4.14. □

The construction presented so far for obtaining forward completeness focuses first on the idea of inducing completeness by guaranteeing reductivity ($\mathbb{F}^{\downarrow}$) and then, if necessary, by forcing monotonicity ($\mathbb{M}^{\downarrow}$). Once again these two transformers deal with different aspects of f:

- $\mathbb{F}^{\downarrow}$ transforms η-closed elements on which completeness fails, i.e. $x \in \eta.\, \rho f\eta(x) \neq f\eta(x)$ implies that $\mathbb{F}^{\downarrow}(f)(x) < f(x)$.
- $\mathbb{M}^{\downarrow}$ transforms only those elements $x \notin \eta$ which are below an incomplete point of η, i.e. $y \in \eta.\, x \geqslant y$ and $\rho f\eta(y) \neq f\eta(y)$. This in particular implies that $\forall x \in \eta.\, \mathbb{M}^{\downarrow}\mathbb{F}^{\downarrow}(f)(x) = \mathbb{F}^{\downarrow}(f)(x)$.
- All the other elements are left unchanged.

Again, we can observe that $\forall x \in \eta.\, \mathbb{F}^{\downarrow}(f)(x) = \rho^{+} f \eta^{+}(x)$. As above, this observation is interesting because it means that the closest $\mathscr{F}$-completeness transformation (ignoring extensivity and monotonicity) of f is its best correct approximation with respect to the corresponding adjoint closures.

Example 4.16. Consider the situation in Example 4.9. Figure 6 shows the transformation of sq' induced by $\mathcal{F}^{\downarrow}_{\rho_c,\rho_c}(sq')$.

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

4.5. *Preserving monotonicity and $\mathscr{F}$-completeness towards reduction*

In this section, we characterize the closest *monotone* transformation of a function f that induces $\mathscr{F}$-completeness, and which is reductive. Let us define the following function:

$$\mathbb{F}^{\mathtt{R}}_{\eta,\rho} \stackrel{\text{def}}{=} \lambda f.\, \lambda x.\, \rho^{+} f \eta(x).$$

This transformation transforms incomplete elements of η exactly as it was done for $\mathbb{F}^{\downarrow}$, i.e. being $\eta^{+}(x) = x$, we have

$$\forall x \in \eta.\, \mathbb{F}^{\mathtt{R}}_{\eta,\rho}(f)(x) = \rho^{+} f(x) = \mathbb{F}^{\downarrow}_{\eta,\rho}(f)(x) = \mathbb{M}^{\downarrow}\mathbb{F}^{\downarrow}(f)(x)$$

while it transforms all the other elements in order to preserve monotonicity, in particular for those non η-closed elements below an incomplete point, the transformation coincide with $\mathbb{M}^{\downarrow}\mathbb{F}^{\downarrow}$, while for all the other elements we have that $\mathbb{F}^{\mathtt{R}}(f)$ is incomparable with f. The following results tells us that inducing reductivity (recall that $\mathbb{R}$ forces reductivity and it is defined in Section 2.1) and then monotonicity or vice versa, leads to the same transformation.

Proposition 4.17. $\mathcal{F}^{\downarrow}_{\eta,\rho}(f) = \mathbb{M}^{\downarrow}\mathbb{F}^{\downarrow}(f) = \mathbb{R}\mathbb{F}^{\mathtt{R}}(f) = \lambda x.\, \rho^{+} f \eta(x) \wedge f(x)$.

Proof. By Lemma 4.14 we know that

$$\mathcal{F}^{\downarrow}_{\eta,\rho}(f) \stackrel{\text{def}}{=} \lambda f.\, \mathbb{M}^{\downarrow}\mathbb{F}^{\downarrow}_{\eta,\rho}(f) = \lambda x. \begin{cases} \rho^{+} f(x) & x \in \eta \\ \bigwedge\{\rho^{+} f(y) |\ y \geqslant x, y \in \eta\} \wedge f(x) & x \notin \eta \end{cases}$$

Let us prove that this function corresponds to $\lambda x.\, \rho^{+} f \eta(x) \wedge f(x)$.
If $x \in \eta$, then $x = \eta(x)$ for all $x \in \eta$, therefore

$$\rho^{+} f \eta \eta(x) \wedge f \eta(x) = \rho^{+} f(x) \wedge f(x) = \rho^{+} f(x)$$

being $\rho^{+} f \eta(x) = \rho^{+} f(x) \leqslant f(x)$.
Consider $x \notin \eta$. We can prove that $\bigwedge\{\rho^{+} f(y) |\ y \geqslant x, y \in \eta\} \wedge f(x) = \rho^{+} f \eta(x) \wedge f(x)$, in particular, we prove that $\bigwedge\{\rho^{+} f(y) |\ y \geqslant x, y \in \eta\} = \rho^{+} f \eta(x)$. Being $\eta(x) \geqslant x$ and $\eta(x) \in \eta$, we have that $\rho^{+} f \eta(x) \in \{\rho^{+} f(y) |\ y \geqslant x, y \in \eta\}$ and therefore $\rho^{+} f \eta(x) \geqslant \bigwedge\{\rho^{+} f(y) |\ y \geqslant x, y \in \eta\}$. On the other hand, for each $y \in \eta$ such that $y \geqslant x$ we have $y \geqslant \eta(x)$, being $\eta(x)$ the smallest fix-point of η greater or equal to x by construction. Hence $\rho^{+} f(y) \geqslant \rho^{+} f \eta(x)$ for each such y, therefore $\bigwedge\{\rho^{+} f(y) |\ y \geqslant x, y \in \eta\} \geqslant \rho^{+} f \eta(x)$. □

Example 4.18. Let us consider again transitions on a transition system. As we underlined previously, in this case the resulting function has to be additive in order to be the transition relation of a transition system. In order to obtain an additive function we simply transform the image of singletons and we obtain the whole function by additive lift. As above we cannot use the transformation of Section 4.4 since in that transformation we only modify the abstract elements that usually (for all the abstractions different from the identity) are not the singletons. Hence, we have to consider the transformation $\mathbb{F}^{\mathtt{R}}_{\eta,\rho}$. Consider for instance the transition system in Figure 1. In this case $\eta = \rho$ is the equivalence relation depicted in the picture. Let $\mathtt{R}$ such equivalence relation, and note that

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

$[X]^+_{\mathtt{R}} \stackrel{\text{def}}{=} \bigcup\{[y]_{\mathtt{R}} |\ [y]_{\mathtt{R}} \subseteq X\}$, then the transformation become $\lambda x \in C.\ \ [f([x]_{\mathtt{R}})]^+_{\mathtt{R}} \cap f(x)$. The resulting transition system is the one depicted in Figure 3.

4.6. *Weakening additivity*

Additivity in the construction of the reduction of a generic function f towards $\mathscr{F}$-completeness is a relatively restrictive hypothesis for closure operators, which corresponds in abstract interpretation to have disjunctive abstract domains (Cousot and Cousot 1979c; Giacobazzi and Ranzato 1998a). The problem is that ρ^+ may not exist in general. In this case however we can still induce $\mathscr{F}$-completeness by weakening the uniqueness of the function. Instead of a unique transformed function we may have a family of equally optimal incomparable functions, all $\mathscr{F}$-complete. In other words, when we have incompleteness on some input x, we can *choose* as output any element in $\max\{\rho(y) |\ \rho(y) \leqslant f(x)\}$, yet keeping $\mathscr{F}$-completeness:

$$\widetilde{\mathbb{F}}^{\downarrow}_{\eta,\rho} \stackrel{\text{def}}{=} \lambda f.\lambda x. \begin{cases} y & \text{if } x \in \eta\ \ \wedge\ \ y \in \max\{\rho(y) |\ \rho(y) \leqslant f(x)\} \\ f(x) & \text{otherwise.} \end{cases}$$

Proposition 4.19. Let $f : C \xrightarrow{\text{m}} C$, then $\widetilde{\mathbb{F}}^{\downarrow}_{\eta,\rho}(f)$ is $\mathscr{F}$-complete and

$$\widetilde{\mathbb{F}}^{\downarrow}_{\eta,\rho}(f) \in \max\{h : C \longrightarrow C |\ f \sqsupseteq h,\ \rho h\eta = h\eta\}.$$

Proof. Let A be $\{h : C \longrightarrow C |\ f \sqsupseteq h,\ \rho h\eta = h\eta\}$, and $f^* \stackrel{\text{def}}{=} \widetilde{\mathbb{F}}^{\downarrow}_{\eta,\rho}(f)$. We first prove that $f^* \in A$, namely that $f^* \sqsubseteq f$ and $\rho f^*\eta = f^*\eta$. Clearly $f^* \sqsubseteq f$ holds trivially by construction. Consider $\rho f^*\eta = f^*\eta$, let $f^*(\eta(x)) = y$, with $y \in \rho$ by definition of $\widetilde{\mathbb{F}}$:

$$\rho(f^*(\eta(x))) = \rho(y) = y = f^*(\eta(x)).$$

Finally we have to prove that f^* is a maximal element of A, namely for each $g \in A$, if $g \sqsupseteq f^*$ we have $g = f^*$. If $x \notin \eta$ then $f^*(x) = f(x)$, and therefore $g(x) \leqslant f(x) = f^*(x)$, namely $f^*(x) = g(x)$. Consider $x \in \eta$, namely $x = \eta(x)$:

$$\begin{array}{l} g(x) \geqslant f^*(x) = y \in \max\{\rho(z) |\ \rho(z) \leqslant f(x)\}\ \Rightarrow \\ g(x) \in \{\rho(z) |\ \rho(z) \leqslant f(x)\} \text{ being } g \text{ complete } (g\eta(x) \in \rho) \text{ and } g \sqsubseteq f\ \Rightarrow \\ g(x) = g(\eta(x)) = y = f^*(x). \end{array}$$

□

It is worth noting that, by weakening the maximality condition, we can still obtain a $\mathscr{F}$-complete transformation by choosing *any* element $y \in \{\rho(z) |\ \rho(z) \leqslant f(x)\}$ for transforming $f(x)$.

5. Making semantics backward complete

In this section, we describe the semantic transformers that induce $\mathscr{B}$-completeness. Unfortunately, as far as $\mathscr{B}$-completeness is concerned, we lose the modularity between

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

monotonicity and extensivity/reductivity, which holds in the case of $\mathscr{F}$-completeness. In fact, in this case, we have only one possible way for both extending and reducing a function towards $\mathscr{B}$-completeness, yet preserving monotonicity. First we observe that we can always transform a function in order to force $\mathscr{B}$-completeness. In particular, we have that while we can always reduce the function for inducing completeness, we have some restrictions when we need to extend the function.

5.1. *Order-theoretic $\mathscr{B}$-complete semantics*

Consider $f : C \longrightarrow C$ and $\eta, \rho \in \mathit{uco}(C)$. We prove that the set of $\mathscr{B}$-complete monotone functions

$$\{h : C \xrightarrow{\text{m}} C |\ \rho h \eta = \rho h\}$$

has a least element under certain conditions. The following result characterizes when we can minimally increase a given monotone function f in order to induce $\mathscr{B}$-completeness.

Theorem 5.1. Given $\eta, \rho \in \mathit{uco}(C)$, the set $\{f : C \longrightarrow C |\ \rho f \eta = \rho f\}$ is an upper closure operator on $\langle C \longrightarrow C, \sqsubseteq \rangle$ iff ρ is co-additive.

Proof. Let us prove that the set $F \stackrel{\text{def}}{=} \{f : C \longrightarrow C |\ \rho \circ f \circ \eta = \rho \circ f\}$ is closed under greatest lower bound. We prove that it is closed under infinitary *glb*. Consider $\{f_i\}_i \subseteq F$, we have to prove that also $\bigsqcap_i f_i \in F$, i.e. $\rho \circ (\bigsqcap_i f_i) \circ \eta = \rho \circ (\bigsqcap_i f_i)$. Let $x \in C$ and suppose ρ co-additive. Then the following equalities hold

$$\begin{array}{rcll}
\rho \circ (\bigsqcap_i f_i)(x) & = & \rho(\bigwedge_i f_i(x)) & \text{By def. of } \wedge \\
& = & \bigwedge_i \rho f_i(x) & \text{By co-additivity of } \rho \\
& = & \bigwedge_i \rho f_i \eta(x) & \text{By complet. hyp. on } f_i \\
& = & \rho(\bigwedge_i f_i \eta(x)) & \text{By co-additivity of } \rho \\
& = & \rho \circ (\bigsqcap_i f_i) \circ \eta(x) &
\end{array}$$

Hence we have completeness.

Suppose now that completeness holds for *glb* of complete functions, we have to prove that ρ is co-additive. We prove that it is infinitary co-additive, i.e. we prove that for any $x_i \in C$, $\rho(\bigwedge_i x_i) = \bigwedge_i \rho(x_i)$. Let us define the following family of functions, where $x \in C$

$$f_x \stackrel{\text{def}}{=} \lambda z. \begin{cases} \rho(x) & \text{if } z \in \eta \\ x & \text{otherwise.} \end{cases}$$

These functions are trivially backward complete, so it is the *glb* among subsets of them, by hypothesis. Finally note that, for any uco ρ we have that $\rho(\wedge\rho(x)) = \wedge\rho(x)$ (**). Consider a family of such elements x_i, then the following equalities holds:

$$\begin{array}{rcll}
\rho(\bigwedge_i x_i) & = & \rho(\bigwedge_i f_{x_i}(z)) & \text{By Def. of } f_{x_i},\ z \notin \rho \\
& = & \rho \circ (\bigsqcap_i f_{x_i})(z) & \\
& = & \rho \circ (\bigsqcap_i f_{x_i}) \circ \eta(z) & \text{By hypothesis} \\
& = & \rho(\bigwedge_i f_{x_i} \eta(z)) & \\
& = & \rho(\bigwedge_i \rho(x_i)) & \text{By Def. of } f_{x_i} \\
& = & \bigwedge_i \rho(x_i) & \text{By Eq. (**)}
\end{array}$$

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

□

Corollary 5.2. Given $\eta, \rho \in uco(C)$, the set $\{f : C \xrightarrow{m} C |\ \rho f \eta = \rho f\}$ of $\mathscr{B}$-complete monotone functions is an upper closure operator on $\langle C \xrightarrow{m} C, \sqsubseteq \rangle$ iff ρ is co-additive.

Proof. By Theorem 5.1 since the *glb* of monotone functions is always monotone. □

On the other hand, next theorem proves that the set of all $\mathscr{B}$-complete functions admits a greatest elements, therefore we can always minimally reduce a function towards completeness.

Theorem 5.3. Given $\eta, \rho \in uco(C)$, the set $\{f : C \longrightarrow C |\ \rho f \eta = \rho f\}$ is a dual Moore family on the domain $\langle C \longrightarrow C, \sqsubseteq \rangle$.

Proof. Let us prove that the set $F \stackrel{\text{def}}{=} \{f : C \longrightarrow C |\ \rho f \eta = \rho f\}$ is closed under least upper bound. We prove that it is closed under infinitary *lub*. Consider $\{f_i\}_i \subseteq F$, we have to prove that also $\bigsqcup_i f_i \in F$, i.e. $\rho(\bigsqcup_i f_i)\eta = \rho(\bigsqcup_i f_i)$. First of all we recall that $\rho(\vee\rho(x)) = \rho(\vee x)$ (*) for any $\rho \in uco(C)$. Let $x \in C$,

$$\begin{array}{rcll} \rho(\bigsqcup_i f_i)(x) & = & \rho(\bigvee_i f_i(x)) & \text{By def. of } \vee \\ & = & \rho(\bigvee_i \rho f_i(x)) & \text{By Eq. (*)} \\ & = & \rho(\bigvee_i \rho f_i \eta(x)) & \text{By complet. hyp. on } f_i \\ & = & \rho(\bigvee_i f_i \eta(x)) & \text{By Eq. (*)} \\ & = & \rho(\bigsqcup_i f_i)\eta(x) & \end{array}$$

□

Corollary 5.4. Given $\eta, \rho \in uco(C)$, the set $\{f : C \xrightarrow{m} C \mid \rho f \eta = \rho f\}$ is a dual Moore family on the domain $\langle C \xrightarrow{m} C, \sqsubseteq \rangle$.

Proof. By Theorem 5.3 since the *lub* of monotone functions is monotone. □

5.2. *Functional extension towards monotone $\mathcal{B}$-completeness*

In this section, we aim to characterize the upper closure operator extending a function in order to make it $\mathscr{B}$-complete. Theorem 5.1 tells us that if ρ is meet-uniform then there exists the best correct $\mathscr{B}$-complete approximation of a function. Let us call this functional $\mathcal{B}^{\uparrow}_{\eta,\rho}$. Consider the function:

$$\mathbb{B}^{\mathbf{I}}_{\eta,\rho} = \lambda f.\, \lambda x.\, \rho^{-} f \eta(x)$$

where $f \in C \xrightarrow{m} C$ and $\eta, \rho \in uco(C)$. Then, we can define the completeness transformer as:

$$\mathcal{B}^{\uparrow}_{\eta,\rho} \stackrel{\text{def}}{=} \mathbb{E}\mathbb{B}^{\mathbf{I}}_{\eta,\rho} = \lambda f.\lambda x.\rho^{-} f \eta(x) \vee f(x).$$

Hence, for backward completeness, the only way we can follow for inducing completeness, when possible, is to transform preserving monotonicity ($\mathbb{B}^{\mathbf{I}}_{\eta,\rho}$) and then to force extensivity ($\mathbb{E}$). Note that, if $x \in \eta$ then $\rho^{-} f \eta(x) \vee f(x) = \rho^{-} f \eta(x) \vee f \eta(x) = f \eta(x) = f(x)$ being $\eta(x) = x$ and $\rho^{-} \in lco(C)$. Otherwise, $\mathcal{B}^{\uparrow}_{\eta,\rho} = \bigwedge\{y \in C |\ y \geqslant f(x),\ \rho(y) \geqslant f \eta(x)\}$ by construction of ρ^{-}.

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

Theorem 5.5. Assume ρ co-additive. Let $f : C \xrightarrow{m} C$, then $\mathcal{B}^{\uparrow}_{\eta,\rho}(f)$ is $\mathscr{B}$-complete and

$$\mathcal{B}^{\uparrow}_{\eta,\rho}(f) = \bigsqcap\{h : C \xrightarrow{m} C |\ f \sqsubseteq h,\ \rho h\eta = \rho h\}$$

Proof. Being ρ co-additive, let $\rho^{-} = \lambda x.\ \bigwedge\{y |\ x \leqslant \rho(y)\} \in \mathit{lco}$. Let us first prove that $f^{*} = \mathcal{B}^{\uparrow}_{\eta,\rho}(f)$ is monotone. Consider $x, y \in C$ such that $x \geqslant y$:

- If $x, y \in \eta$ then $f^{*}(x) = f(x) \geqslant f(y) = f^{*}(y)$, by monotonicity of f;
- If $x \in \eta$ and $y \notin \eta$, then since $x \geqslant y$ we have $x \geqslant \eta(y)$. Now, by monotonicity of f we have $f\eta(y) \leqslant f(x) \leqslant \rho f(x)$ and $f(x) \geqslant f(y)$, i.e. $f(x) \in \{z \in C |\ z \geqslant f(y),\ \rho(z) \geqslant f\eta(y))\}$.
  So we obtain that, $f^{*}(y) = \bigwedge\{y \in C |\ y \geqslant f(x),\ \rho(y) \geqslant f\eta(x)\} \leqslant f(x) = f^{*}(x)$;
- If $x \notin \eta$ and $y \in \eta$, then by definition we have $f^{*}(x) \geqslant f(x) \geqslant f(y) = f^{*}(y)$;
- If $x, y \notin \eta$, then note that by monotonicity of f we have $f(x) \geqslant f(y)$ and hence $f\eta(x) \geqslant f\eta(y)$. Consider $k \in \{k |\ k \geqslant f(x),\ \rho(k) \geqslant f\eta(x)\}$ then $k \geqslant f(x) \geqslant f(y)$ and $\rho(k) \geqslant f\eta(x) \geqslant f\eta(y)$, hence $k \in \{z |\ z \geqslant f(y),\ \rho(z) \geqslant f\eta(y)\}$, i.e. $\{k |\ k \geqslant f(x),\ \rho(k) \geqslant f\eta(x)\} \subseteq \{z |\ z \geqslant f(y),\ \rho(z) \geqslant f\eta(y)\}$, which implies

$$\begin{aligned} f^{*}(x) &= \bigwedge\{k |\ k \geqslant f(x),\ \rho(k) \geqslant f\eta(x)\} \\ &\geqslant \bigwedge\{z |\ z \geqslant f(y),\ \rho(z) \geqslant f\eta(y)\} = f^{*}(y). \end{aligned}$$

It is straightforward to note that $f^{*} \geqslant f$. Let us prove that $\mathcal{B}^{\uparrow}_{\eta,\rho}(f)$ is $\mathscr{B}$-complete. First of all, note that by definition we have $\rho f^{*}\eta = \rho f\eta$. On the other side

$$\rho f^{*}(x) = \begin{cases} \rho f(x) = \rho f\eta(x) = \rho f^{*}\eta(x) & \text{if } x \in \eta \\ \rho(\bigwedge\{y |\ y \geqslant f(x),\ \rho(y) \geqslant f\eta(x)\}) & \text{otherwise.} \end{cases}$$

For this last case we have that

$$\begin{aligned} &\rho(\bigwedge\{y |\ y \geqslant f(x),\ \rho(y) \geqslant f\eta(x)\}) \\ &\quad \geqslant \rho(\bigwedge\{y |\ \rho(y) \geqslant f\eta(x)\}) && \text{Since it is a greater set} \\ &\quad = \rho(\rho^{-} f\eta(x)) && \text{By Def. of } \rho^{-} \\ &\quad = \rho f\eta(x) && \text{By Th. 2.1} \\ &\quad = \rho f^{*}\eta(x). \end{aligned}$$

Hence, we have $\rho f^{*}(x) \geqslant \rho f^{*}\eta(x)$, but the other inclusion is trivial, so we have the equality.

Finally, we prove that f^{*} is the *glb* of the set, i.e. $\forall g : C \xrightarrow{m} C$ such that $\rho g = \rho g\eta$, then $f^{*} \sqsubseteq g$. If $x \in \eta$ then $f^{*}(x) = f(x)$, which implies by definition that $g(x) \geqslant f^{*}(x)$. Suppose $x \notin \eta$:

$$\begin{aligned} \forall x.\ g(x) \geqslant f(x) (\text{By Def. of } g) &\Rightarrow g\eta(x) \geqslant f\eta(x) \\ &\Rightarrow \rho g\eta(x) \geqslant f\eta(x) (\text{Since } \rho g\eta(x) \geqslant g\eta(x)) \\ &\Rightarrow \rho g(x) \geqslant f\eta(x) (g \text{ is complete}) \\ &\Rightarrow g(x) \in \{y |\ y \geqslant f(x),\ \rho(y) \geqslant f\eta(x)\} \\ &\Rightarrow g(x) \geqslant \bigwedge\{y |\ y \geqslant f(x),\ \rho(y) \geqslant f\eta(x)\} \\ &\Rightarrow g(x) \geqslant f^{*}(x) \end{aligned}$$

□

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

5.3. *Weakening co-additivity*

As above for $\mathscr{F}$-completeness, the extension of an arbitrary (monotone) function towards $\mathscr{B}$-completeness exists only when the output observation is a co-additive closure. Let us see in this section what happens when the observation ρ is not co-additive. In this case, the problem is that ρ^- may not exist. As above, we can still induce $\mathscr{B}$-completeness by weakening the uniqueness of the transformed function. In other words, when we have incompleteness on some input x, we can *choose* to transform $f(x)$ in any element in $\min\{y|\ \rho(y) \geqslant f(x)\}$:

$$\begin{array}{rcl} \widetilde{\mathcal{B}}^{\uparrow}_{\eta,\rho} & \stackrel{\text{def}}{=} & \lambda f.\lambda x.y \vee f(x) \text{ where } y \in \min\{y|\ \rho(y) \geqslant f\eta(x)\} \\ & = & \lambda x.y \in \min\{y \in C|\ y \geqslant f(x),\ \rho(y) \geqslant f\eta(x)\} \end{array}$$

Proposition 5.6. Let $f : C \xrightarrow{\text{m}} C$, then $\mathcal{B}^{\uparrow}_{\eta,\rho}(f)$ is $\mathscr{B}$-complete and

$$\widetilde{\mathcal{B}}^{\uparrow}_{\eta,\rho}(f) \ \in\ \min\{h : C \xrightarrow{\text{m}} C|\ f \sqsubseteq h,\ \rho h\eta = h\eta\}$$

Proof. Let us first prove that $f^* = \mathcal{B}^{\uparrow}_{\eta,\rho}(f)$ is monotone. Consider $x, y \in C$ such that $x \geqslant y$:

- If $x, y \in \eta$ then $f^*(x) = f(x) \geqslant f(y) = f^*(y)$, by monotonicity of f.
- If $x \in \eta$ and $y \notin \eta$, then since $x \geqslant y$ we have $x \geqslant \eta(y)$. Now, by monotonicity of f we have $f\eta(y) \leqslant f(x) \leqslant \rho f(x)$ and $f(x) \geqslant f(y)$, hence $f(x) \in \{z \in C|\ z \geqslant f(y),\ \rho(z) \geqslant f\eta(y))\}$.
  So we obtain that, $f^*(y) \in \min\{y \in C|\ y \geqslant f(x),\ \rho(y) \geqslant f\eta(x)\} \leqslant f(x) = f^*(x)$.
- If $x \notin \eta$ and $y \in \eta$, then by definition we have $f^*(x) \geqslant f(x) \geqslant f(y) = f^*(y)$.
- If $x, y \notin \eta$, then note that by monotonicity of f we have $f(x) \geqslant f(y)$ and hence $f\eta(x) \geqslant f\eta(y)$. Consider $k \in \{k|\ k \geqslant f(x),\ \rho(k) \geqslant f\eta(x)\}$ then $k \geqslant f(x) \geqslant f(y)$ and $\rho(k) \geqslant f\eta(x) \geqslant f\eta(y)$, hence $k \in \{z|\ z \geqslant f(y),\ \rho(z) \geqslant f\eta(y)\}$, i.e.
  $\{k|\ k \geqslant f(x),\ \rho(k) \geqslant f\eta(x)\} \subseteq \{z|\ z \geqslant f(y),\ \rho(z) \geqslant f\eta(y)\}$, which implies

$$\begin{array}{rcl} f^*(x) & \in & \min\{k|\ k \geqslant f(x),\ \rho(k) \geqslant f\eta(x)\} \\ & \geqslant & f^*(y) \in \min\{z|\ z \geqslant f(y),\ \rho(z) \geqslant f\eta(y)\}. \end{array}$$

It is straightforward to note that $f^* \geqslant f$, let us prove it is complete. First of all, note that by definition we have $\rho f^*\eta = \rho f\eta$. On the other side

$$\rho f^*(x) = \begin{cases} \rho f(x) = \rho f\eta(x) = \rho f^*\eta(x) & \text{if } x \in \eta \\ \rho(\min\{y|\ y \geqslant f(x),\ \rho(y) \geqslant f\eta(x)\}) & \text{otherwise} \end{cases}$$

For this last case we have that

$$\begin{array}{ll} \rho(\min\{y|\ y \geqslant f(x),\ \rho(y) \geqslant f\eta(x)\}) & \\ \quad \geqslant \rho(\min\{y|\ \rho(y) \geqslant f\eta(x)\}) & \text{Since it is a greater set} \\ \quad \geqslant f\eta(x) & \text{By construction} \\ \quad \Rightarrow\ \rho f^* \geqslant \rho f\eta = \rho f^*\eta(x) & \text{By idempotence of } \rho \end{array}$$

Since the other inclusion is trivial, so we have the equality.

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

Finally, we prove that f^* is minimal in the set, i.e. $\forall g : C \xrightarrow{m} C$ such that $\rho g = \rho g \eta$ and $g \sqsubseteq f^*$, then $f^* = g$. If $x \in \eta$ then $f^*(x) = f(x)$, which implies by definition that $g(x) \geqslant f^*(x)$. Suppose $x \notin \eta$:

$$\begin{aligned}
\forall x.\, g(x) \geqslant f(x) (\text{By Def. of } g) &\Rightarrow g\eta(x) \geqslant f\eta(x) \\
&\Rightarrow \rho g\eta(x) \geqslant f\eta(x) (\text{Since } \rho g\eta(x) \geqslant g\eta(x)) \\
&\Rightarrow \rho g(x) \geqslant f\eta(x) (g \text{ is complete}) \\
&\Rightarrow g(x) \in \{y |\ y \geqslant f(x),\ \rho(y) \geqslant f\eta(x)\} \\
&\Rightarrow g(x) = f^*(x) \\
&\quad (\text{Being } g(x) \leqslant f^*(x) \text{ and } f^*(x) \text{ minimal in the set})
\end{aligned}$$

□

It is worth noting that, as for $\mathscr{F}$-completeness, also in this case we can relax the minimality condition. In particular, we can choose *any* element in $\{y |\ y \geqslant f(x),\ \rho(y) \geqslant f\eta(x)\}$ as a possible image of the transformed function $\mathcal{B}^{\uparrow}_{\eta,\rho}(f)(x)$. An interesting and easy choice could be $\mathcal{B}^{\uparrow}_{\eta,\rho}(f) = \lambda x.f\eta(x)$, which is obviously complete by idempotence of η.

5.4. *Functional reduction towards monotone $\mathcal{B}$-completeness*

In this section, we characterize the lower closure operator transforming a function towards $\mathscr{B}$-completeness. Theorem 5.3 tells us that there always exists the best correct $\mathscr{B}$-complete concretization of a function. Let us call this functional $\mathcal{B}^{\downarrow}_{\eta,\rho}$.

Lemma 5.7. Let $f : C \xrightarrow{m} C$ and $\rho, \eta \in \mathit{uco}(C)$.

$$\mathbb{B}^{\downarrow}_{\eta,\rho} = \lambda f.\lambda x. \bigvee \{w |\ \forall z \in [x]_\eta.\ [w]_\rho \cap \downarrow f(z) \neq \varnothing\}$$

where, for each $\varphi \in \mathit{uco}$ we have that $[x]_\varphi \stackrel{\text{def}}{=} \{z |\ \varphi(z) = \varphi(x)\}$ is the kernel of ρ[†]. $\mathbb{B}^{\downarrow}_{\eta,\rho}(f)$ is $\mathscr{B}$-complete.

Proof. First of all let us prove that $\{w |\ \forall z \in [x]_\eta.\ [w]_\rho \cap \downarrow f(z) \neq \varnothing\}$ contains its maximum. Consider a family of its elements, $\{w_i\}_{i \in I}$. Hence they are such that $\forall z \in [x]_\eta$ $[w_i]_\rho \cap \downarrow f(x) \neq \varnothing$. This means that $\forall z \in [x]_\eta$ we have that $\exists y_i.\, \rho(y_i) = \rho(w_i) \wedge y_i \leqslant f(z)$. Then we have that $\rho(\bigvee_i y_i) = \rho(\bigvee_i \rho(y_i)) = \rho(\bigvee_i \rho(w_i)) = \rho(\bigvee w_i)$ and $\bigvee_i y_i \leqslant f(z)$, hence $\bigvee_i w_i$ belongs to the set.

At this point it is worth noting that $\mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x) = \mathbb{B}^{\downarrow}_{\eta,\rho}(f)(\eta(x))$, by construction, hence the resulting function is trivially $\mathscr{B}$-complete. □

This transformation induces $\mathscr{B}$-completeness, but it may lack both reductivity and monotonicity. Next results shows that, by forcing reductivity, we preserve completeness and we obtain the corresponding closest reductive transformation.

Lemma 5.8. $\mathbb{RB}^{\downarrow}_{\eta,\rho}$ is idempotent on $\mathscr{B}$-complete functions and $\mathbb{RB}^{\downarrow}_{\eta,\rho}(f)$ is a $\mathscr{B}$-complete function.

[†] The use of the kernel of an uco is an equivalent way for representing closures (Cousot and Cousot 1979c)

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

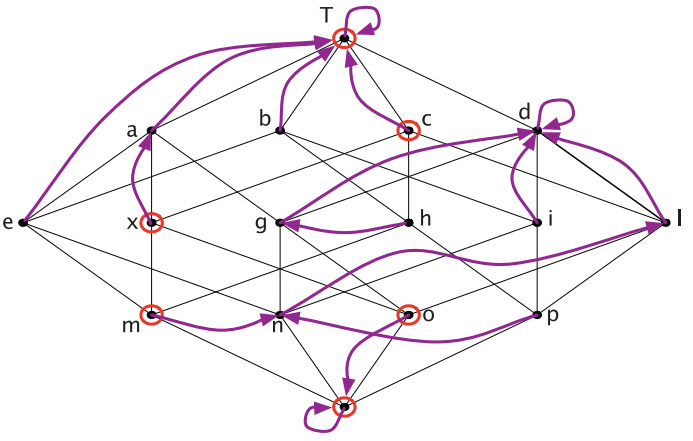


Fig. 7. $\mathscr{B}$-complete transformation (1).

Proof. Let us denote $f^*(x) \stackrel{\text{def}}{=} \mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x) \wedge f(x)$. Consider f complete, then we have that $\forall z \in [x]_\eta.\ \rho f(z) = \rho f \eta(z) = \rho f \eta(x)$. This means that $[\rho f \eta(x)]_\rho \cap \downarrow f(z) \neq \emptyset$ for any $z \in [x]_\rho$, since $\rho f \eta(x)$ is the image by ρf of each one of these z. Hence $\rho f \eta(x)$ is in $\mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x)$. Let $\overline{w} = \mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x)$, then $\overline{w} \geqslant \rho f \eta(x)$, but then $f(x) \leqslant \rho f \eta(x) \leqslant \overline{w}$, hence $f^*(x) = f(x) \wedge \mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x) = f(x)$. At this point note that $\mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x) \in \rho$. Indeed, if $\overline{w} = \mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x)$, then also $\rho(\overline{w})$ is inside the set, hence it is lower than the *lub*, which is $\overline{w}$, which implies that $\overline{w} = \rho(\overline{w}) \in \rho$. Now, we can prove that $\rho f^*(x) = \rho(\mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x) \wedge f(x)) = \rho(\mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x)) = \mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x)$.

Clearly, $\mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x) = \rho(\mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x)) \geqslant \rho(\mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x) \wedge f(x))$. Now, suppose again $\overline{w} = \mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x)$, then for what we proved before, $\overline{w}$ is in the set, hence for each $z \in [x]_\eta$ there exists $y \in [\overline{w}]_\rho$ such that $y \leqslant f(z)$. Therefore, $\rho(y) = \rho(\overline{w}) = \mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x)$. This implies that $y \leqslant \mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x)$ which together with $y \leqslant f(x)$ implies that $y \leqslant \mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x) \wedge f(x)$. Finally, this means that $\mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x) = \rho(y) \leqslant \rho(\mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x) \wedge f(x))$, which implies the equality. So, $\rho f^*(x) = \mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x) = \mathbb{B}^{\downarrow}_{\eta,\rho}(f)(\eta(x)) = \rho f^* \eta(x)$, i.e. f^* is complete. □

Lemma 5.9. $\mathbb{B}^{\downarrow}_{\eta,\rho}(f)$ is optimal, i.e. for any $g \in \{h : C \xrightarrow{\text{m}} C \mid f \sqsupseteq h,\ \rho h \eta = \rho h\}$, we have $g \sqsubseteq \mathbb{B}^{\downarrow}_{\eta,\rho}(f)$.

Proof. Consider $g \in \{h : C \xrightarrow{\text{m}} C \mid f \geqslant h,\ \rho h \eta = \rho h\}$. Then by completeness we have that $\forall z \in [x]_\eta.\ \rho g(z) = \rho g \eta(z) = \rho g \eta(x)$, hence $[\rho g \eta(x)]_\rho \cap \downarrow g(z) \neq \emptyset$. By hypothesis, we have that $\forall x.\ g(x) \leqslant f(x)$, hence that $\downarrow g(x) \subseteq \downarrow f(x)$, therefore we can conclude that $\forall z \in [x]_\eta$ we have $[\rho g \eta(x)]_\rho \cap \downarrow f(z) \neq \emptyset$. Clearly this means that $\rho g \eta(x) \in \{w \mid \forall z \in [x]_\eta.\ [w]_\rho \cap \downarrow f(z) \neq \emptyset\}$ and therefore we have the following disequalities: $g(x) \leqslant \rho g \eta(x) \leqslant \bigvee \{w \mid \forall z \in [x]_\eta.\ [w]_\rho \cap \downarrow f(z) \neq \emptyset\}$. At this point, being $g(x) \leqslant f(x)$ we obtain the result, i.e. $g(x) \leqslant \mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x) \wedge f(x) = f^*(x)$. □

Note that $\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho}(f)$, and therefore $\mathbb{B}^{\downarrow}_{\eta,\rho}$, may not return a monotone function, as shown in Figure 7. In this picture, circled points are those belonging to the closure ρ, while the plain arrows represent the input function f. Consider $\rho = \eta$. We can see, that the domain is not $\mathscr{B}$-complete for the input function f. In particular, note that $f(n) = l$

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

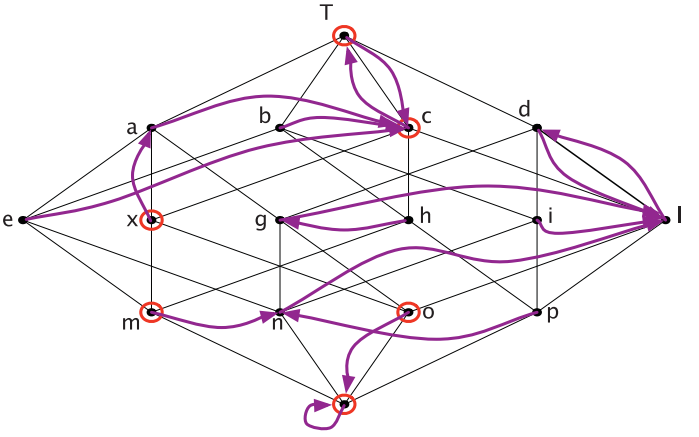


Fig. 8. $\mathscr{B}$-complete transformation (2).

while $\rho(n) = \top$, hence while $\rho(f(n)) = c$, we have $\rho(f(\rho(n))) = \top$. The new function $f' \stackrel{\text{def}}{=} \mathbb{B}^{\downarrow}_{\eta,\rho}(f)$, depicted in Figure 8, makes the following transformations:

$$f'(e) = f'(a) = f'(b) = f'(\top) = c \qquad f'(g) = f'(i) = f'(d) = l$$

for all the other points $f' = f$. We can note that the resulting function is no longer monotone, for instance note that $l \leqslant d$ while $f'(d) = l < f'(l) = d$.

Lemma 5.10. The functional $\mathbb{B}^{\downarrow}_{\eta,\rho}$ is monotone on $\langle C \xrightarrow{\text{m}} C, \sqsubseteq \rangle$.

Proof. Consider $f, g : C \xrightarrow{\text{m}} C$, such that $f \sqsubseteq g$ (i.e. $\forall x \in C.\, f(x) \leqslant g(x)$). We have to prove that $\mathbb{B}^{\downarrow}_{\eta,\rho}(f) \leqslant \mathbb{B}^{\downarrow}_{\eta,\rho}(g)$, i.e. $\mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x) \leqslant \mathbb{B}^{\downarrow}_{\eta,\rho}(g)(x)$.
Hence, consider $\mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x) = \bigvee\{w |\, \forall z \in [x]_\eta.\, [w]_\rho \cap \downarrow f(z) \neq \varnothing\}$ and $\mathbb{B}^{\downarrow}_{\eta,\rho}(g)(x) = \bigvee\{w |\, \forall z \in [x]_\eta.\, [w]_\rho \cap \downarrow g(z) \neq \varnothing\}$. Being $f \leqslant g$ we have that $\forall z.\, \downarrow f(z) \subseteq \downarrow g(z)$, hence $[w]_\rho \cap \downarrow f(z) \neq \varnothing$ implies $[w]_\rho \cap \downarrow g(z)$. This implies $\{w |\, \forall z \in [x]_\eta.\, [w]_\rho \cap \downarrow f(z) \neq \varnothing\} \subseteq \{w |\, \forall z \in [x]_\eta.\, [w]_\rho \cap \downarrow g(z) \neq \varnothing\}$, i.e. we have the result. □

Lemma 5.11. The following properties hold:

1. $\mathbb{M}^{\downarrow}$ is co-additive;
2. $\mathbb{R}$ is co-additive;
3. $\mathbb{M}^{\downarrow}\,\mathbb{R} = \mathbb{R}\,\mathbb{M}^{\downarrow}$ is an upper closure operator.

Proof.

1. Let us prove that $\mathbb{M}^{\downarrow}$ is co-additive. Hence we have to prove that, given a family of functions f_{ii}, then $\mathbb{M}^{\downarrow}(\bigsqcap_i f_i) = \bigsqcap_i \mathbb{M}^{\downarrow}(f_i)$.

$$\begin{aligned}\mathbb{M}^{\downarrow}(\textstyle\bigsqcap_i f_i)(x) &= \textstyle\bigwedge\{(\bigsqcap_i f_i)(y) |\, y \geqslant x\} = \bigwedge\{\bigwedge_i f_i(y) |\, y \geqslant x\} \\ &= \textstyle\bigwedge_i \bigwedge\{f_i(y) |\, y \geqslant x\} = \bigsqcap_i \mathbb{M}^{\downarrow}(f_i)(x).\end{aligned}$$

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

2. Let us prove that $\mathbb{R}$ is co-additive. Hence we have to prove that, given a family of functions $f_{i\,i}$, then $\mathbb{R}(\bigsqcap_i f_i) = \bigsqcap_i \mathbb{R}(f_i)$.

$$\begin{aligned}\mathbb{R}(\textstyle\bigsqcap_i f_i)(x) &= (\textstyle\bigsqcap_i f_i)(x) \wedge x = (\textstyle\bigwedge_i f_i(x)) \wedge x \\ &= \textstyle\bigwedge_i (f_i(x) \wedge x) = \bigsqcap_i \mathbb{R}(f_i(x)).\end{aligned}$$

3. By duality from (Cousot and Cousot 1979a, Lemma 3.4) we have the result.

□

Next result specifies the minimal reduction towards $\mathscr{B}$-completeness as a fix-point functional, iteratively combining the functionals for achieving monotonicity, reductivity and $\mathscr{B}$-completeness.

Theorem 5.12.

$$\mathcal{B}^{\downarrow}_{\eta,\rho} \stackrel{\text{def}}{=} gfp^{\preccurlyeq}_{f}(\mathbb{M}^{\downarrow}\ \mathbb{R}\ \mathbb{B}^{\downarrow}_{\eta,\rho}) = \bigsqcup\{h : C \xrightarrow{\text{m}} C \mid f \sqsupseteq h,\ \rho h\eta = \rho h\}.$$

Proof. By Lemma 5.9, we have that the function $\mathbb{M}^{\downarrow}\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho}$ is monotone, so a fix-point can be reached as greatest fix-point. Let us prove that indeed the fix-points of this function are monotone, reductive and $\mathscr{B}$-complete. Let us denote the greatest fix-point as $(\mathbb{M}^{\downarrow}\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho})_{\omega} = \bigsqcap_i (\mathbb{M}^{\downarrow}\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho})^i$, then we have that $\mathbb{M}^{\downarrow}\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho}((\mathbb{M}^{\downarrow}\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho})_{\omega}) = (\mathbb{M}^{\downarrow}\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho})_{\omega}$.

Monotonicity:

$$\begin{aligned}\mathbb{M}^{\downarrow}((\mathbb{M}^{\downarrow}\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho})_{\omega}) &= \mathbb{M}^{\downarrow}(\textstyle\bigsqcap_i (\mathbb{M}^{\downarrow}\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho})^i) \\ &= \textstyle\bigsqcap_i (\mathbb{M}^{\downarrow}\mathbb{M}^{\downarrow}\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho})^i \qquad \text{(By Lemma 5.11(1))} \\ &= \textstyle\bigsqcap_i (\mathbb{M}^{\downarrow}\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho})^i \qquad \text{(By idempotence of } \mathbb{M}^{\downarrow}) \\ &= (\mathbb{M}^{\downarrow}\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho})_{\omega}\end{aligned}$$

By the properties of $\mathbb{M}^{\downarrow}$, this means that the fix-point is monotone.

Reductivity:

$$\begin{aligned}\mathbb{R}((\mathbb{M}^{\downarrow}\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho})_{\omega}) &= \mathbb{R}(\textstyle\bigsqcap_i (\mathbb{M}^{\downarrow}\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho})^i) \\ &= \textstyle\bigsqcap_i (\mathbb{R}\mathbb{M}^{\downarrow}\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho})^i \qquad \text{(By Lemma 5.11(2))} \\ &= \textstyle\bigsqcap_i (\mathbb{M}^{\downarrow}\mathbb{R}\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho})^i \qquad \text{(By Lemma 4.5(3))} \\ &= \textstyle\bigsqcap_i (\mathbb{M}^{\downarrow}\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho})^i \qquad \text{(By idempotence of } \mathbb{R}) \\ &= (\mathbb{M}^{\downarrow}\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho})_{\omega}\end{aligned}$$

By the properties of $\mathbb{R}$, this means that the fix-point is reductive.

$\mathscr{B}$-Completeness:

Suppose $f = (\mathbb{M}^{\downarrow}\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho})_{\omega}$ not complete, hence $\mathbb{B}^{\downarrow}_{\eta,\rho}(f) \neq f$, note that $\mathbb{B}^{\downarrow}_{\eta,\rho}(f) \not\geqslant f$, because otherwise $\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho}(f) = f$ would be complete, against the hypothesis. This means that $\exists x\,.\,\mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x) < f(x)$ or not comparable with it. This implies that $\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x) < f(x)$. Hence, $\mathbb{M}^{\downarrow}\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho}(f)(x) \leqslant \mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho}(f(x)) < f(x)$, since $\mathbb{M}^{\downarrow}(f(x)) \leqslant f(x)$. But, if $f(x)$ is a fix-point of $\mathbb{M}^{\downarrow}\mathbb{R}\mathbb{B}^{\downarrow}_{\eta,\rho}$ this is absurd, so fix-points are complete. □

Consider again the function in Figure 7 and its transformation in Figure 8. As explained, the function in Figure 8 is not monotone after one application of $\mathbb{B}^{\downarrow}_{\eta,\rho}$, hence we have to apply the transformation $\mathbb{M}^{\downarrow}$, towards monotonicity, obtaining the function $f' = \mathbb{M}^{\downarrow}(f)$,

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

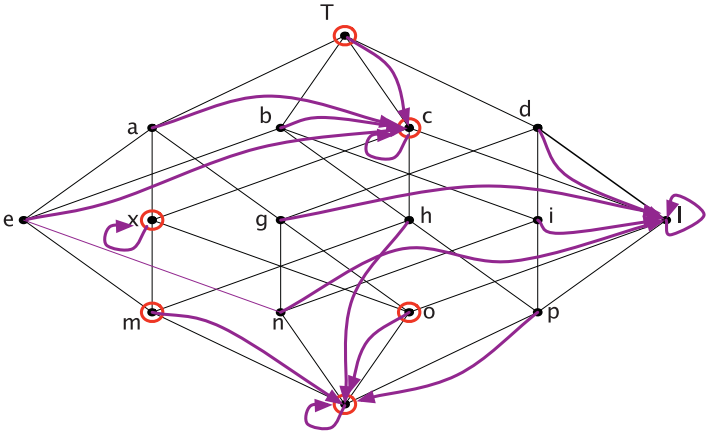


Fig. 9. $\mathscr{B}$-complete transformation (3).

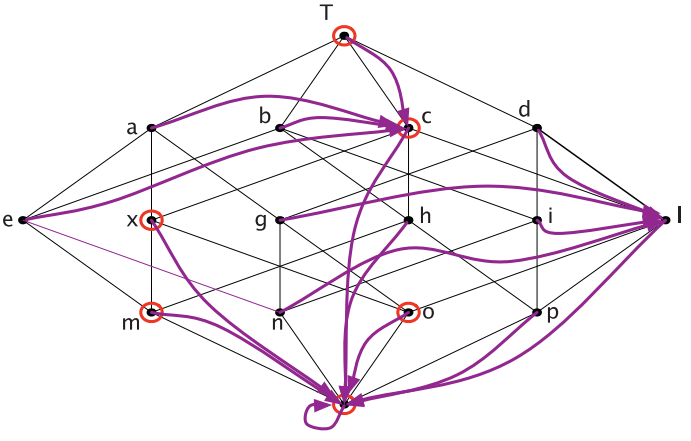


Fig. 10. $\mathscr{B}$-complete transformation (4).

depicted in Figure 9:

$$f'(m) = f'(p) = f'(h) = \bot \qquad f'(x) = x \qquad f'(l) = l \qquad f'(c) = c.$$

Note that the new function is no more $\mathscr{B}$-complete. Indeed, note that for instance $f(h) = \bot$ while $\rho(h) = c$, hence $\rho(f(h)) = \bot$ while $\rho(f(\rho(h))) = c$. So, we have to apply again the two transformers. $\mathbb{B}^{\downarrow}_{\eta,\rho}$ does the following transformations:

$$f''(c) = f''(l) = \bot$$

At this point note that the function, is no more monotone. Indeed, for instance $x \leqslant c$ while $f''(c) = \bot < f''(x) = f'(x) = x$. Then we apply for the last time $\mathbb{M}^{\downarrow}$, obtaining a function, represented in Figure 10, which is both monotone and $\mathscr{B}$-complete, such that $f'''(x) = \bot$.

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

We conclude this section by observing that, under meet-uniformity conditions, we can again find a modularity in the composition of the functionals $\mathbb{R}$ and $\mathbb{B}^{\downarrow}_{\eta,\rho}$, which is similar to the one observed *moving upwards*. In other words, when the closures are meet-uniform, we can first induce completeness by preserving monotonicity and then we can force reductivity.

Proposition 5.13. Let ρ, η be meet-uniform closures. Let $\mathbb{B}^{\mathrm{O}}_{\eta,\rho} \stackrel{\text{def}}{=} \lambda f.\, \lambda x.\, \rho f \eta^{-}(x)$, then

1. $\mathbb{B}^{\downarrow}_{\eta,\rho} = \mathbb{B}^{\mathrm{O}}_{\eta,\rho}$
2. $\mathcal{B}^{\downarrow}_{\eta,\rho}(f) = \mathbb{R}\mathbb{B}^{\mathrm{O}}_{\eta,\rho}(f) = \lambda x.\, \rho f \eta^{-}(x) \wedge f(x)$.

Proof.

We have to prove that when η and ρ are meet-uniform then

$$\bigvee \{w \mid \forall z \in [x]_\eta.\, [w]_\rho \cap \downarrow f(z) \neq \varnothing\} = \rho f \eta^{-}(x).$$

First of all let us note that by definition $\eta^{-}(x) = \bigwedge \{y \mid \eta(y) = \eta(x)\} = \bigwedge [x]_\eta$, moreover by meet-uniformity of ρ we have that $\rho(\bigwedge [x]_\rho) = \rho(x)$.

Consider now the following equalities:

$$\begin{aligned}
&\forall z \in [x]_\eta.\, [w]_\rho \cap \downarrow f(z) \neq \varnothing \\
&\quad \Leftrightarrow \forall z \in [x]_\eta.\, \{y \mid \rho(y) = \rho(x)\} \cap \{y \mid y \leqslant f(z)\} \neq \varnothing \\
&\quad \Leftrightarrow \forall z \in [x]_\eta.\, \exists y.\, \rho(y) = \rho(x) \ \wedge \ y \leqslant f(z) \\
&\quad \Leftrightarrow \forall z \in [x]_\eta.\, \textstyle\bigwedge [w]_\rho \leqslant f(z) \qquad \text{By meet-uniformity of } \rho \\
&\quad \Leftrightarrow \textstyle\bigwedge [w]_\rho \leqslant f(\bigwedge [x]_\eta) \qquad \text{By meet-uniformity of } \eta
\end{aligned}$$

Therefore $\{w \mid \forall z \in [x]_\eta.\, [w]_\rho \cap \downarrow f(z) \neq \varnothing\} = \{w \mid \bigwedge [w]_\rho \leqslant f(\bigwedge [x]_\eta)\}$.

At this point, we have to prove that $\mathbb{B}^{\downarrow}_{\eta,\rho}(f) = \bigvee \{w \mid \bigwedge [w]_\rho \leqslant f(\bigwedge [x]_\eta)\}$ is equal to $\rho f(\bigwedge [x]_\eta)$. Note that $\rho f(\bigwedge [x]\eta) \in \{w \mid \bigwedge [w]_\rho \leqslant f(\bigwedge [x]_\eta)\}$ being trivially

$$\bigwedge [\rho f(\bigwedge [x]_\eta)]_\rho = \bigwedge [f(\bigwedge [x]_\eta)]_\rho \leqslant f(\bigwedge [x]_\eta)$$

hence $\rho f(\bigwedge [x]_\eta)]_\rho \leqslant \mathbb{B}^{\downarrow}_{\eta,\rho}(f)$. On the other hand, $\forall w \in \{w \mid \bigwedge [w]_\rho \leqslant f(\bigwedge [x]_\eta)\}$ we have $w \leqslant \rho(w) = \rho(\bigwedge [w]_\rho) \leqslant \rho f(\bigwedge [x]_\eta)$ by monotonicity of ρ. Hence we have the equality. □

Note that the functional $\mathbb{B}^{\mathrm{O}}_{\eta,\rho}$, when it exists, is always monotone. The problem here is that it exists only when η is meet-uniform. Moreover, we can prove that $\mathbb{B}^{\downarrow}_{\eta,\rho} = \mathbb{B}^{\mathrm{O}}_{\eta,\rho}$ only when also ρ is meet-uniform.

Example 5.14. Consider the concrete domain S in Figure 4, the abstract square operation $sq^{\sharp}$ and an abstraction $\eta = \{\mathbb{Z}, [-\infty, 0], [0, +\infty], [0, 10], [0]\}$, depicted with black dots in Figure 11. This figure shows the transformation of $sq^{\sharp}$ induced by $\mathcal{B}^{\uparrow}_{\eta,\eta}(sq^{\sharp})$.

6. Completeness in program security: static program monitoring

In this section, we consider an example of program transformations used in program security and protection. The idea is simple: Security is enforced by selecting reliable computations and discarding untrustworthy ones. We prove that the selection can be

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

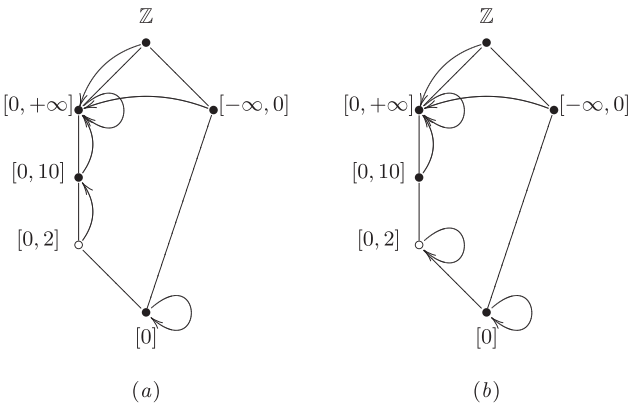


Fig. 11. The abstract domain S with $sq^\sharp$ and $\mathcal{B}^{\downarrow}_{\eta,\eta}(sq^\sharp)$.

specified in terms of a minimal (in the approximation order) completeness transformation of the semantics of the program. This provides both a proof of minimality for standard program transformations employed in software security, such as program monitoring, and a deeper comprehension of these techniques in terms of the precision of an observer (the attacker) with respect to the expected behaviour of the program (its semantics).

Consider *Program Monitoring* formalized in the context of program transformation by abstract interpretation (Cousot and Cousot 2002). Program monitoring consists in restricting the possible executions of a program P in order to enforce a given safety property Π. Examples of program monitoring are: the insertion of run-time checks for checking errors such as the division by zero and out of bound array indexing; the transformation of a target system before the system is executed in order to make it halt whenever it is about to violate some security property of interest (Erlingsson and Schneider 1999; Schneider 2000). In this section, we aim to show that program monitoring, rejecting computations violating a given safety property are indeed $\mathcal{F}$-completeness program transformations. In order to show this correspondence let us first briefly and informally introduce the characterization of program monitoring provided in Cousot and Cousot (2002).

Let σ be the trace of actions modelling the computation of a program P executed together with the monitor M, then we denote by $\sigma{\downarrow}_{\mathtt{P}}$ the projection on the variables in P only of the actions of P, while $\sigma{\downarrow}_{\mathtt{M}}$ projects on the variables of M all the actions. Then, the semantics of the monitored program is $\mathtt{t}^{\mathtt{M}}(\mathtt{P}) = \{\sigma | \sigma{\downarrow}_{\mathtt{P}}$ is a computation of P and $\sigma{\downarrow}_{\mathtt{M}}$ is a computation of M$\}$. This semantics can be expressed in fix-point form Cousot and Cousot (2002):

$$\mathtt{t}^{\mathtt{M}}(\mathtt{P}) = \mathit{lfp}^{\subseteq}\lambda\mathcal{T}.\ \mathtt{Init} \cup \mathtt{Next}(\mathcal{T})$$

where Init is the set of all the sequences of actions executed by M before the start of the execution of P, while $\mathtt{Next}(\mathcal{T})$ models the situation where after the execution of an action of P there is a sequence of actions executed by M while P does not continue its execution

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

(see Cousot and Cousot (2002) for details):

$$\mathtt{Init} = \left\{ s_1 \dots s_n \;\middle|\; \begin{array}{l} \forall i < n.\ s_i \text{ is not an action of } \mathtt{P}, s_1 \dots s_{n-1} \text{ is an execution of } \mathtt{M}, \\ s_n \text{ is an action of } \mathtt{P} \end{array} \right\}$$

$$\mathtt{Next}(\mathcal{T}) = \left\{ \sigma s_1 \dots s_n \;\middle|\; \begin{array}{l} \sigma s_1 \in \mathcal{T}, \forall\ 2 \leqslant i < n.\ s_i \text{ is not an action of } \mathtt{P}, s_1 \text{ is an action of } \mathtt{P} \\ s_1 \dots s_{n-1} \text{ is an execution of } \mathtt{M}, \text{and} \langle s_1, s_n \rangle \text{ is a transition in } \mathtt{P} \end{array} \right\}$$

The intuition is that each time P executes an action then M has to execute a sequence of actions for checking whether the computation still satisfies the safety property, namely one step of P is controlled by several steps of M. Hence, when M finds that the execution satisfies no more property, it stops the execution.

At this point, we can simplify the description of the semantics of the monitored program by considering only the projection of traces on P, and supposing that the check of the property Π requires some execution steps of M: Let $\mathcal{I}[\![\mathtt{P}]\!]_\Pi$ be the set of initial states of P satisfying Π

$$\mathtt{F}_\Pi(\mathcal{T}) = \mathcal{I}[\![\mathtt{P}]\!]_\Pi \cup \{\sigma s s' |\ \sigma s \in \mathcal{T},\ \langle s, s' \rangle \text{ transition in } \mathtt{P},\ \text{and } \Pi(\sigma s s')\}$$

Hence, we can simplify the characterization in Cousot and Cousot (2002).

Theorem 6.1. $\mathtt{t}^{\mathtt{M}}(\mathtt{P})\downarrow_{\mathtt{P}} = \mathit{lfp}^{\subseteq}_{\emptyset}\ \mathtt{F}_\Pi$.

Proof. First of all let us show what is formally $\mathtt{t}^{\mathtt{M}}(\mathtt{P})\downarrow_{\mathtt{P}}$. Suppose that Π is the property enforced by the monitor M.

$$\begin{aligned} \mathtt{Init}\downarrow_{\mathtt{P}} &= \left\{ s_1 \dots s_n \;\middle|\; \begin{array}{l} \forall i < n. s_i \text{ is not an action of } \mathtt{P}, s_1 \dots s_{n-1} \\ \text{is an execution of } \mathtt{M}, \\ s_n \text{ is an action of } \mathtt{P} \end{array} \right\}\downarrow_{\mathtt{P}} \\ &= \left\{ s_n\downarrow_{\mathtt{P}} \;\middle|\; s_n\downarrow_{\mathtt{P}} \in \mathcal{I}[\![\mathtt{P}]\!], \Pi(s_n\downarrow_{\mathtt{P}}) \right\} \stackrel{\text{def}}{=} \mathcal{I}[\![\mathtt{P}]\!]_\Pi \end{aligned}$$

while

$$\begin{aligned} \mathtt{Next}(\mathcal{T})\downarrow_{\mathtt{P}} &= \left\{ \sigma s_1 \dots s_n \;\middle|\; \begin{array}{l} \sigma s_1 \in \mathcal{T}, \forall 2 \leqslant i < n. s_i \text{ is not an action of } \mathtt{P}, \\ s_1 \text{ is an action of } \mathtt{P}, s_1 \dots s_{n-1} \text{ is an execution of } \mathtt{M}, \\ \langle s_1, s_n \rangle \text{ is a transition in } \mathtt{P} \end{array} \right\}\downarrow_{\mathtt{P}} \\ &= \left\{ \sigma\downarrow_{\mathtt{P}}\ s_1 s_n \;\middle|\; \begin{array}{l} \sigma\downarrow_{\mathtt{P}}\ s_1 \in \mathcal{T}\downarrow_{\mathtt{P}}, \langle s_1, s_n \rangle \text{ transition of } \mathtt{P}, \\ \Pi(\sigma\downarrow_{\mathtt{P}}\ s_1 s_n) \end{array} \right\} \end{aligned}$$

where $\Pi(\sigma)$ means that M has not stopped the execution of σ, namely the property Π of σ holds. Moreover note that, in each iteration in the computation of $\mathtt{t}^{\mathtt{M}}(\mathtt{P})$ the execution of M does not consider the history of the computation, but depends only on the last state of the trace, hence it is trivial to show that $\mathtt{Next}(\mathcal{T})\downarrow_{\mathtt{P}} = \mathtt{Next}(\mathcal{T}\downarrow_{\mathtt{P}})\downarrow_{\mathtt{P}}$, hence $\mathtt{t}^{\mathtt{M}}(\mathtt{P})\downarrow_{\mathtt{P}} = (\mathit{lfp}^{\subseteq}\lambda\mathcal{T}.\mathtt{Init} \cup \mathtt{Next}(\mathcal{T}))\downarrow_{\mathtt{P}} = \lambda\mathcal{T}.\mathtt{Init}\downarrow_{\mathtt{P}} \cup \mathtt{Next}(\mathcal{T})\downarrow_{\mathtt{P}}$. Therefore we have the result. □

Let us introduce, now, the completeness characterization of monitoring. In Mastroeni and Giacobazzi (2011) we formalize safety semantics as abstract interpretations in the hierarchy of semantics (Cousot 2002). A safety property is characterized as an abstract interpretation considering the abstraction (Gumm 1993) which maps a set X of (finite)

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

traces in the set of all their finite prefixes $\varphi_\omega(X) = \{\sigma \in \Sigma^+ | \exists \delta \in X.\, \sigma \preccurlyeq \delta\}$, where $\preccurlyeq$ means 'prefix of'. By using this abstraction we can model the fact that a safety property is maximal with respect to a given set of partial executions. At this point let us note that, if we consider the finite partial trace semantics of P (Cousot and Cousot 2002), i.e. $[\![\mathtt{P}]\!] \stackrel{\text{def}}{=} \mathit{lfp}^{\subseteq}_{\varnothing} \mathtt{F}_\mathtt{P}$, where $\mathtt{F}_\mathtt{P} \stackrel{\text{def}}{=} \lambda \mathcal{T}.\, \mathcal{I}[\![\mathtt{P}]\!] \cup \{\sigma s s' | \sigma s \in \mathcal{T}, \langle s, s' \rangle \text{ transition in } \mathtt{P}\}$, then we have the following property.

Proposition 6.2. $\varphi_\omega([\![\mathtt{P}]\!]) = [\![\mathtt{P}]\!]$.

Proof. In order to prove the thesis we first prove by induction that $\varphi_\omega(\mathtt{F}^i_\mathtt{P}) = \bigcup_{j \leqslant i} \mathtt{F}^j_\mathtt{P}$, where $\mathtt{F}^0_\mathtt{P} \stackrel{\text{def}}{=} \mathtt{F}_\mathtt{P}(\varnothing)$, while $\mathtt{F}^i_\mathtt{P} \stackrel{\text{def}}{=} \mathtt{F}_\mathtt{P}(\mathtt{F}^{i-1}_\mathtt{P})$. Indeed, $\varphi_\omega(\mathtt{F}^0_\mathtt{P}) = \varphi_\omega(\mathcal{I}[\![\mathtt{P}]\!]) = \mathcal{I}[\![\mathtt{P}]\!]$ since $\mathcal{I}[\![\mathtt{P}]\!]$ contains single states. Let us suppose by induction that $\varphi_\omega(\mathtt{F}^i_\mathtt{P}) = \bigcup_{j \leqslant i} \mathtt{F}^j_\mathtt{P}$, and let us compute $\varphi_\omega(\mathtt{F}^{i+1}_\mathtt{P})$.

$$
\begin{aligned}
\varphi_\omega(\mathtt{F}^{i+1}_\mathtt{P}) &= \varphi_\omega(\mathtt{F}_\mathtt{P}(\mathtt{F}^i_\mathtt{P})) \\
&= \varphi_\omega(\mathcal{I}[\![\mathtt{P}]\!] \cup \{\sigma s s' | \sigma s \in \mathtt{F}^i_\mathtt{P},\ \langle s, s' \rangle \text{ transition of } \mathtt{P}\}) \\
&= \varphi_\omega(\mathcal{I}[\![\mathtt{P}]\!]) \cup \varphi_\omega(\mathtt{F}^i_\mathtt{P}) \cup \{\sigma s s' | \sigma s \in \mathtt{F}^i_\mathtt{P},\ \langle s, s' \rangle \text{ transition of } \mathtt{P}\} \\
&\quad \text{(Being } \varphi_\omega \text{ additive (Gumm 1993))} \\
&= \mathcal{I}[\![\mathtt{P}]\!] \cup \textstyle\bigcup_{j \leqslant i} \mathtt{F}^j_\mathtt{P} \cup \{\sigma s s' | \sigma s \in \mathtt{F}^i_\mathtt{P},\ \langle s, s' \rangle \text{ transition of } \mathtt{P}\} \\
&\quad \text{(By inductive hypothesis)} \\
&= \textstyle\bigcup_{j \leqslant i} \mathtt{F}^j_\mathtt{P} \cup \{\sigma s s' | \sigma s \in \mathtt{F}^i_\mathtt{P},\ \langle s, s' \rangle \text{ transition of } \mathtt{P}\} \\
&\quad \text{(Being } \mathcal{I}[\![\mathtt{P}]\!] = \mathtt{F}^0_\mathtt{P}\text{)} \\
&= \textstyle\bigcup_{j \leqslant i} \mathtt{F}^j_\mathtt{P} \cup \mathtt{F}^{i+1}_\mathtt{P} \text{ (By definition of } \mathtt{F}^{i+1}_\mathtt{P}\text{)} \\
&= \textstyle\bigcup_{j \leqslant i+1} \mathtt{F}^j_\mathtt{P}
\end{aligned}
$$

Hence, $\varphi_\omega([\![\mathtt{P}]\!]) = \varphi_\omega(\mathit{lfp}^{\subseteq}_{\varnothing} \mathtt{F}_\mathtt{P}) = \varphi_\omega(\bigcup_{i<\omega} \mathtt{F}^i_\mathtt{P}) = \bigcup_{i<\omega} \mathtt{F}^i_\mathtt{P} = [\![\mathtt{P}]\!]$. □

When we model a (safety) property as the set of all the computations satisfying it, we can say that a program P satisfies Π iff $[\![\mathtt{P}]\!] \subseteq \Pi$. By Proposition 6.2 this corresponds to saying $\varphi_\omega([\![\mathtt{P}]\!]) \subseteq \Pi$. Moreover, Π being a safety property it is prefix closed (Gumm 1993) and therefore it is such that $\sigma \in \Pi$ implies $\varphi_\omega(\sigma) \subseteq \Pi$.

We can model this condition as a completeness problem. Let us consider $\pi \in \mathit{uco}(\wp(\Sigma^+))$ defined as follows: $\pi(X) = X$ if $X \subseteq \Pi$, $\pi(X) = \top$ otherwise, i.e. π is the closure operator characterizing the sets of computations satisfying Π. Then we have

$$[\![\mathtt{P}]\!] \subseteq \Pi \qquad \text{iff} \qquad \pi[\![\mathtt{P}]\!] = [\![\mathtt{P}]\!]$$

which is clearly a forward completeness equation where the input abstraction is the identity. Hence we can transform the semantics $[\![\mathtt{P}]\!]$ in order to make the safety property Π hold. Consider, the right adjoint π^+ of π:

$$\pi^+ = \lambda X.\, \bigcup \{Y | \pi(Y) \subseteq X\} = \lambda X.\, \bigcup \{\pi(Y) | \pi(Y) \subseteq X\} \text{ (By Proposition 2.2)}$$

Then we have that $\mathbb{F}^{\downarrow}_{id,\pi}([\![\mathtt{P}]\!]) = \pi^+([\![\mathtt{P}]\!])$ which is exactly the transformation which, among all the (partial) computations of P, keeps only those satisfying Π. This is precisely the semantics of the monitored program, as proved in the following result.

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

Theorem 6.3. $\mathtt{t}^{\mathtt{M}}(\mathtt{P})\downarrow_{\mathtt{P}} = \mathbb{F}^{\downarrow}_{id,\pi}([\![\mathtt{P}]\!])$

Proof. By Theorem 6.1 and by definition of fix-point, we have that $\mathtt{t}^{\mathtt{M}}(\mathtt{P})\downarrow_{\mathtt{P}} = \bigcup_{i<\omega} \mathtt{F}^i_\Pi$, where $\mathtt{F}^0_\Pi = \mathtt{F}_\Pi$ and $\mathtt{F}^{i+1}_\Pi = \mathtt{F}_\Pi \mathtt{F}^i_\Pi$. We can trivially note that $\mathtt{F}_\Pi(X) = \mathtt{F}_{\mathtt{P}}(X) \cap \Pi$, where we abuse notation denoting by Π the set of all the traces satisfying Π.

On the other hand, by definition, $\mathbb{F}^{\downarrow}([\![\mathtt{P}]\!]) = \pi^+([\![\mathtt{P}]\!]) = \bigcup\{\pi(Y) |\ \pi(Y) \subseteq [\![\mathtt{P}]\!]\} = \bigcup\{Y \subseteq \Pi |\ Y \subseteq [\![\mathtt{P}]\!]\} = \bigcup\{Y |\ Y \subseteq [\![\mathtt{P}]\!] \cap \Pi\} = [\![\mathtt{P}]\!] \cap \Pi = \bigcup_{i<\omega} \mathtt{F}^i_{\mathtt{P}} \cap \Pi$.
Hence, we have to show that $(\bigcup_{i<\omega} \mathtt{F}^i_{\mathtt{P}}) \cap \Pi = \bigcup_{i<\omega} (\mathtt{F}^i_{\mathtt{P}} \cap \Pi) = \bigcup_{i<\omega} (\mathtt{F}_{\mathtt{P}} \cap \Pi)^i = \bigcup_{i<\omega} \mathtt{F}^i_\Pi$, where the first equality holds by distributivity of intersection on union, and the last one holds by definition. Let us prove by induction that $\mathtt{F}^i_\Pi = \mathtt{F}^i_{\mathtt{P}} \cap \Pi$. The base is trivial, $\mathtt{F}^0_\Pi = \mathtt{F}_\Pi(\varnothing) = \mathtt{F}_{\mathtt{P}}(\varnothing) \cap \Pi = \mathtt{F}^0_{\mathtt{P}} \cap \Pi$. Suppose it holds the inductive hypothesis. Let us prove separately the two inclusions. Consider $\mathtt{F}^i_\Pi \subseteq \mathtt{F}^i_{\mathtt{P}} \cap \Pi$, then $\mathtt{F}^i_\Pi \subseteq \mathtt{F}^i_{\mathtt{P}}$. By definition of $\mathtt{F}_\Pi$.

$$\begin{aligned} \mathtt{F}^i_\Pi = \mathtt{F}_\Pi(\mathtt{F}^i_\Pi) &= \mathtt{F}_{\mathtt{P}}(\mathtt{F}^{i-1}_\Pi) \cap \Pi \\ &\subseteq \mathtt{F}_{\mathtt{P}}(\mathtt{F}^{i-1}_{\mathtt{P}}) \cap \Pi \\ &= \mathtt{F}^i_{\mathtt{P}} \cap \Pi \end{aligned}$$

Let us prove now that $\mathtt{F}^i_\Pi \supseteq \mathtt{F}^i_{\mathtt{P}} \cap \Pi$. In particular

$$\begin{aligned} \delta \in \mathtt{F}^i_{\mathtt{P}} \cap \Pi &\Rightarrow \delta \in \Pi \ \wedge\ \delta \in \{\sigma s s' |\ \sigma s \in \mathtt{F}^{i-1}_{\mathtt{P}},\ \langle s, s' \rangle \text{ transition of } \mathtt{P}\} \\ &\Rightarrow \delta \in \{\sigma s s' |\ \sigma s \in \mathtt{F}^{i-1}_{\mathtt{P}},\ \langle s, s' \rangle \text{ transition of } \mathtt{P}, \Pi(\sigma s s')\} \end{aligned}$$

Note that by definition Π is a safety property, namely we have $\Pi(\sigma s s') \Rightarrow \Pi(\sigma s)$ hence we have the following relations

$$\begin{aligned} \delta \in \mathtt{F}^i_{\mathtt{P}} \cap \Pi &\Rightarrow \delta \in \{\sigma s s' |\ \sigma s \in \mathtt{F}^{i-1}_{\mathtt{P}},\ \langle s, s' \rangle \text{ transition of } \mathtt{P}, \Pi(\sigma s s')\} \\ &\Rightarrow \delta \in \{\sigma s s' |\ \sigma s \in \mathtt{F}^{i-1}_{\mathtt{P}} \cap \Pi,\ \langle s, s' \rangle \text{ transition of } \mathtt{P}, \Pi(\sigma s s')\} \\ &\Rightarrow \delta \in \{\sigma s s' |\ \sigma s \in \mathtt{F}^{i-1}_{\mathtt{P}} \cap \Pi,\ \langle s, s' \rangle \text{ transition of } \mathtt{P}\} \cap \Pi \\ &\Rightarrow \delta \in \{\sigma s s' |\ \sigma s \in (\mathtt{F}_{\mathtt{P}} \cap \Pi)^{i-1},\ \langle s, s' \rangle \text{ transition of } \mathtt{P}\} \cap \Pi \\ &\quad \text{(By inductive hypothesis)} \\ &\Rightarrow \delta \in \mathtt{F}_{\mathtt{P}}((\mathtt{F}_{\mathtt{P}} \cap \Pi)^{i-1}) \cap \Pi = \mathtt{F}_{\mathtt{P}}(\mathtt{F}^{i-1}_\Pi) \cap \Pi = \mathtt{F}^i_\Pi \end{aligned}$$

Hence we proved that $\mathtt{F}^i_\Pi = \mathtt{F}^i_{\mathtt{P}} \cap \Pi$. Finally,

$$\begin{aligned} \delta \in (\bigcup_{i<\omega} \mathtt{F}^i_{\mathtt{P}}) \cap \Pi &\Leftrightarrow \exists i.\ \delta \in \mathtt{F}^i_{\mathtt{P}} \cap \Pi = \mathtt{F}^i_\Pi \\ &\Leftrightarrow \delta \in \bigcup_{i<\omega} \mathtt{F}^i_\Pi \end{aligned}$$

□

7. Conclusion

We studied the lattice-structure of complete semantics in the standard adjoint framework of abstract interpretation. In particular we proved that it is possible, under often non-restrictive hypothesis, to characterize the *minimal* model transformations making a given *semantics* complete, both in the forward and backward sense.

The transformations considered here concern the semantics, instead of the abstract domains, completing, together with (Giacobazzi *et al.* 2000), the picture of domain and semantics transformers for making abstract interpretations complete, either by minimally

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

modifying abstractions or by minimally modifying semantics. In comparison with abstract domain refinement for completeness in Giacobazzi *et al.* (2000), the interest in semantics transformations for achieving completeness may appear less critical. This is a consequence of the way abstract interpretation is typically considered and used, where the program is a constant and the abstraction is a variable to determine. In contrast, making models complete requires that abstractions are constants, and the semantics, i.e. the program, is a variable to be determined. Moreover, the transformations considered here concern the semantics instead of the *syntax*, even if they are justified/applied in the context of program transformation, which indeed concerns syntax. This way of driving transformations, starting from the semantics, opens several big problems, among all: how do we move from the semantics to the syntax transformation? and how this transformations affect the computational complexity? The first question finds partial answer in a recent work (Giacobazzi *et al.* 2012) where obfuscation is obtained by a completeness-driven design of language interpreters. This is the first attempt to connect (*in*)completeness semantic transformations with syntactic transformations: the interpreter transforms the program introducing some syntactic structure that surely makes the given analysis imprecise. In the completeness case we should go in the opposite direction, the interpreter should have to erase form the program all the syntactic structures inducing imprecision in the analysis, but it is clear that this idea needs further work in order to be completely understood. As far as computational complexity is concerned we believe that the problem is even harder to attack. Semantic transformations, in principle, may change the scope of a program making a complexity comparison quite meaningless. Anyway it may be the case that, being the transformation guided by a particular analysis, it may affect the relation between the complexity of the original program and that of the transformed one, but this is quite far to be understood and therefore deserves further research.

We considered here language-based security as an application ground for interpreting our transformations, but of course the validity of our results are more general. We believe that the idea of making an abstract interpretation complete by modifying the program is an underlying feature of most algorithm and methods for program construction and design. Preliminary examples in this direction can be found in Vechev *et al.* (2010) where the authors propose a procedure combining abstraction and semantics refinement for synchronization synthesis in multi-threaded programs. The theory of minimal model transformations for completeness may provide here the appropriate mathematical structures for proving minimality on all these cases. An important aspect where complete model transformations may play a key role is in program transformation for refining program analysis. It is often the case that the result of an optimized compilation induces a loss of precision in the abstract interpreter w.r.t. the original source code. This is the case, for instance, when a C program is compiled into an intermediate three-address language. We believe that *program hints* refining program operations (see Laviron and Logozzo (2009)) can be systematically designed as complete model transformations, providing the static analyser with new advanced tools for refinement both at domain and program level. We are currently interested in expanding the theory of model transformations by considering also semantics transformations inducing maximal incompleteness (Giacobazzi and Mastroeni 2008). Possible applications of the basic

https://doi.org/10.1017/S0960129514000358 Published online by Cambridge University Press

semantic transformers for achieving maximal incompleteness are in code obfuscation (see Giacobazzi (2008) and Giacobazzi *et al.* (2012) for some preliminary result in this direction). In this case, an obfuscated program defeating an attacker which performs approximate analysis driven reverse engineering can be viewed as the maximal incomplete transformation of the program with respect to the abstractions used by the analyser. Unfortunately, for these latter operators, the conditions for the existence of the optimal deformed model are far more restrictive, making hard the derivation of concrete algorithms and tools. Under this perspective, the road towards an automated deforming compiler assisting the developer in program synthesis, repair and protection is still long to run.

Acknowledgements

Part of the material developed in this paper was conceived with Francesco Ranzato and Elisa Quintarelli. We are grateful to the endless discussions we had together during the last decade, discussions that helped us in better understanding the beauty of abstract interpretation theory. The paper has been written when Roberto Giacobazzi was visiting the ENS, École Normale Supérieure in rue d'Ulm, Paris. We thank ENS for always providing a perfect environment for researchers. Finally, we would like to thank the anonymous referees for their helpful comments and suggestions.

References

Abramsky, S. and Jung, A. (1994) Domain theory. In: Abramsky, S., Gabbay, D. M. and Maibaum, T. S. E. (eds.) *Handbook of Logic in Computer Science*, volume 3, Oxford University Press, Inc. 1–168.

Ball, T., Podelski, A. and Rajamani, S. (2002) Relative completeness of abstraction refinement for software model checking. In: Kaoen, J.-P. and Stevens, P. (eds.) Proceedings of TACAS: Tools and Algorithms for the Construction and Analysis of Systems. *Springer-Verlag Lecture Notes in Computer Science* **2280** 158–172.

Blyth, T. and Janowitz, M. (1972) *Residuation Theory*, Pergamon Press.

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

Collberg, C. and Nagra, J. (2010) *Surreptitious Software*, Addison Wesley.

Collberg, C. and Thomborson, C. D. (1999) Software watermarking: Models and dynamic embeddings. In: *POPL'99: Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ACM, New York, NY, USA 311–324.

Collberg, C. and Thomborson, C. D. (2000) Watermarking, tamper-proofing, and obfuscation-tools for software protection. *IEEE Transactions on Software Engineering* **28** 735–746.

Cousot, P. (2002) Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. *Theoretical Computer Science* **277** (1-2) 47–103.

Cousot, P. and Cousot, R. (1977) Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fix-points. In: *Proceedings of Conference Record of the 4th ACM Symposium on Principles of Programming Languages (POPL'77)*, ACM Press, New York 238–252.

Cousot, P. and Cousot, R. (1979a) A constructive characterization of the lattices of all retractions, preclosure, quasi-closure and closure operators on a complete lattice. *Portuguese Mathematical* **38** (2) 185–198.

Cousot, P. and Cousot, R. (1979b) Constructive versions of Tarski's fixed point theorems. *Pacific Journal of Mathematics* **82** (1) 43–57.

Cousot, P. and Cousot, R. (1979c) Systematic design of program analysis frameworks. In: *Proceedings of Conference Record of the 6th ACM Symposium on Principles of Programming Languages (POPL'79)* , ACM Press, New York 269–282.

Cousot, P. and Cousot, R. (1992a) Abstract interpretation frameworks. *Journal of Logic and Computation* **2** (4) 511–547.

Cousot, P. and Cousot, R. (1992b) Comparing the Galois connection and widening/narrowing approaches to abstract interpretation (Invited Paper) In: Bruynooghe, M. and Wirsing, M. (eds.) Proceedings of the 4th International Symposium on Programming Language Implementation and Logic Programming (PLILP'92). *Springer-Verlag Lecture Notes in Computer Science* **631** 269–295.

Cousot, P. and Cousot, R. (2002) Systematic design of program transformation frameworks by abstract interpretation. In: *Proceedings of Conference Record of the 29th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ACM Press, New York 178–190.

Cousot, P. and Cousot, R. (2004) An abstract interpretation-based framework for software watermarking. In: *Conference Record of the 31st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Venice, Italy*. ACM Press 173–185.

Cousot, P., Cousot, R., Feret, J., Mauborgne, L., Min, A., Monniaux, D. and Rival, X. (2007a) Combination of abstractions in the ASTRÉE static analyzer. In: Okada, M. and Satoh, I. (eds.) 11th Annual Asian Computing Science Conference (ASIAN'06). *Springer Lecture Notes in Computer Science* **4435** 1–24.

Cousot, P., Ganty, P. and Raskin, J.-F. (2007b) Fixpoint-guided abstraction refinements. In: Filé, G. and Riis Nielson, H. (eds.) Proceedings of the 14th International Symposium on Static Analysis, SAS'07, Kongens Lyngby, Denmark. *Springer Lecture Notes in Computer Science* **4634** 333–348.

Dalla Preda, M., Christodorescu, M., Jha, S. and Debray, S. (2007) A semantics-based approach to malware detection. In: *POPL'07: Proceedings of the 34th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, ACM Press, New York, NY, USA 377–388.

Dalla Preda, M. and Giacobazzi, R. (2009) Semantic-based code obfuscation by abstract interpretation. *Journal of Computer Security* **17** (6) 855–908.

Dams, D., Gerth, R. and Grumberg, O. (1997) Abstract interpretation of reactive systems. *ACM Transactions on Programming Languages and Systems* **19** (2) 253–291.

Davey, B. A. and Priestley, H. A. (1990) *Introduction to Lattices and Order*, Cambridge University Press, Cambridge, U.K.

Erlingsson, U. and Schneider, F. (1999) Sasi enforcement of security policies: A retrospective. In: *NSPW'99*, ACM Press, New York 87–95.

Filé, G., Giacobazzi, R. and Ranzato, F. (1996) A unifying view of abstract domain design. *ACM Computing Surveys* **28** (2) 333–336.

Giacobazzi, R. (2008) Hiding information in completeness holes – new perspectives in code obfuscation and watermarking. In: *Proceedings of The 6th IEEE International Conferences on Software Engineering and Formal Methods (SEFM'08)*, IEEE Press 7–20.

Giacobazzi, R., Jones, N. and Mastroeni, I. (2012) Obfuscation by partial evaluation of distorted interpreters. In: *Proceedings of the ACM Symposium on Partial Evaluation and Program Manipulation (PEPM'12)*, ACM Press, New York 179–185.

Giacobazzi, R. and Mastroeni, I. (2004) Abstract non-interference: Parameterizing non-interference by abstract interpretation. In: *Proceedings of the 31st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '04)*, ACM-Press, New York 186–197.

Giacobazzi, R. and Mastroeni, I. (2008) Transforming abstract interpretations by abstract interpretation. In: Alpuente, M. (ed.) Proceedings of the 15th International Static Analysis Symposium, SAS'08. *Springer-Verlag Lecture Notes in Computer Science* **5079** 1–17.

Giacobazzi, R. and Quintarelli, E. (2001) Incompleteness, counterexamples and refinements in abstract model-checking. In: Cousot, P. (ed.) Proceedings of The 8th International Static Analysis Symposium (SAS'01). *Springer-Verlag Lecture Notes in Computer Science* **2126** 356–373.

Giacobazzi, R. and Ranzato, F. (1997) Refining and compressing abstract domains. In: Degano, P., Gorrieri, R. and Marchetti-Spaccamela, A. (eds.) Proceedings of the 24th International Colloquium on Automata, Languages and Programming (ICALP '97). *Springer-Verlag Lecture Notes in Computer Science* **1256** 771–781.

Giacobazzi, R. and Ranzato, F. (1998a) Optimal domains for disjunctive abstract interpretation. *Science of Computer Programming* **32** (1-3) 177–210.

Giacobazzi, R. and Ranzato, F. (1998b) Uniform closures: Order-theoretically reconstructing logic program semantics and abstract domain refinements. *Information and Computation* **145** (2) 153–190.

Giacobazzi, R., Ranzato, F. and Scozzari, F. (2000) Making abstract interpretations complete. *Journal of the ACM* **47** (2) 361–416.

Gulavani, B. S. and Rajamani, S. K. (2006) Counterexample driven refinement for abstract interpretation. In: TACAS 06: Tools and Algorithms for Construction and Analysis of Systems. *Springer Lecture Notes in Computer Science* **3920** 474–488.

Gumm, H. P. (1993) Another glance at the Alpern–Schneider theorem. *Information Processing Letters* **47** 291–294.

Janowitz, M. F. (1967) Residuated closure operators. *Portuguese Mathematical* **26** (2) 221–252.

Kihara, M., Fujiyoshi, M., Wan, Q. T. and Kiya, H. (2007) Image tamper detection using mathematical morphology. In: *ICIP 2007: IEEE International Conference on Image Processing*, IEEE 101–104.

Laviron, V. and Logozzo, F. (2009) Refining abstract interpretation-based static analyses with hints. In: Proceedings of APLAS'09. *Springer-Verlag Lecture Notes in Computer Science* **5904** 343–358.

Mastroeni, I. (2004) Algebraic power analysis by abstract interpretation. *Higher-Order and Symbolic Computation (HOSC)* **17** (4) 299–347.

Mastroeni, I. (2008) Deriving bisimulations by simplifying partitions. In: Proceedings of the 9th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'08). *Springer-Verlag Lecture Notes in Computer Science* **4905** 147–171.

Mastroeni, I. and Giacobazzi, R. (2011) An abstract interpretation-based model for safety semantics. *Journal of Computer Mathematics* **88** (4) 665–694.

Mycroft, A. (1993) Completeness and predicate-based abstract interpretation. In: *Proceedings of the ACM Symposium on Partial Evaluation and Program Manipulation (PEPM'93)*, ACM Press, New York 179–185.

Nagra, J. and Thomborson, C. D. (2004) Threading software watermarks. In: Proceedings of 6th International Workshop on Information Hiding. *Springer-Verlag Lecture Notes in Computer Science* **3200** 208–233.

Nagra, J., Thomborson, C. D. and Collberg, C. (2002) A functional taxonomy for software watermarking. *Australian Computer Science Communications* **24** (1) 177–186.

Nielson, F., Nielson, H. and Hankin, C. (1999) *Principles of Program Analysis*, Springer.

Ranzato, F. and Tapparo, F. (2007) Generalized strong preservation by abstract interpretation. *Journal of Logic and Computation* **17** (1) 157–197.

Rival, X. and Mauborgne, L. (2007) The trace partitioning abstract domain. *ACM Transactions on Programming Languages and Systems* **29** (5) 26.

Schneider, F. B. (2000) Enforceable security policies. *Information and System Security* **3** (1) 30–50.
Vechev, M. T., Yahav, E. and Yorsh, G. (2010) Abstraction-guided synthesis of synchronization. In: *Proceedings of the 37th ACM SIGPLAN-SIGACT POPL 2010*, ACM 327–338.
Venkatesan, R., Vazirani, V. and Sinha, S. (2001) A graph theoretic approach to software watermarking. In: Information Hiding. *Lecture Notes in Computer Science* **2137** 157–168.
Ward, M. (1942) The closure operators of a lattice. *Annals of Mathematical* **43** (2) 191–196.