Comparing Completeness Properties of Static Analyses and Their Logics

David A. Schmidt*

Kansas State University, Manhattan, Kansas, USA
schmidt@cis.ksu.edu.

Abstract. Static analyses calculate abstract states, and their logics validate properties of the abstract states. We place into perspective the variety of forwards, backwards, functional, and logical completeness used in abstract-interpretation-based static analysis by giving examples and by proving equivalences, implications, and independences. We expose two fundamental Galois connections that underlie the logics for static analyses and reveal a new completeness variant, *O-completeness*. We also show that the key concept underlying logical completeness is *covering*, which we use to relate the various forms of completeness.

When we use a static analysis, like data-flow analysis or model checking, to validate a program for correctness or code improvement, we must carefully define the domain of properties the analysis can calculate so that it includes both the goal properties we seek to validate as well as intermediate properties that lead to the goals. Say we try to validate $\{?\}\mathtt{y} := -\mathtt{y}; \mathtt{x} := \mathtt{y} + 1\{\mathit{isPositive}(\mathtt{x})\}$; our analysis requires properties like *isNegative* to calculate a sound precondition: $\{\mathit{isNegative}(\mathtt{y})\}\ \mathtt{y} := -\mathtt{y}\ \{\mathit{isPositive}(\mathtt{y})\}\ \mathtt{x} := \mathtt{y} + 1\ \{\mathit{isPositive}(\mathtt{x})\}$. But, is the analysis *complete* — as expressive as possible? If we can express the properties, *isNonNegative* and *isNonPositive*, then a complete analysis calculates the weakest precondition: $\{\mathit{isNonPositive}(\mathtt{y})\}\ \mathtt{y} := -\mathtt{y}; \mathtt{x} := \mathtt{y} + 1\ \{\mathit{isPositive}(\mathtt{x})\}$.

The example suggests that "completeness" is a property of both static analyses as well as logics. Thanks to Cousot and Cousot [6,7,8,11], we have a well-defined notion of *functional completeness*: it is when a static analysis's abstract state-transition function precisely mimicks the concrete state-transition function, modulo the Galois connection between concrete and abstract domains.

Giacobazzi, Ranzato, and Scozarri [17] showed how to refine an abstract interpretation to synthesize functionally complete transition functions; Giacobazzi and Quintarelli [16] showed that there are, in fact, two, independent notions of functional completeness — *forwards* and *backwards*. Cousot and Cousot [11] applied functional completeness to define the *logical completeness* of a logic that judges abstract values as compared to the logic that judges the concrete values. Recently, Ranzato and Tapparo [23,24] applied Giacobazzi, et al.'s refinement techniques to build logically complete abstract logics.

The present paper's contribution is to place into perspective the variants of forwards, backwards, functional, and logical completeness by giving examples

* Supported by NSF ITR-0086154 and ITR-0326577.

N. Kobayashi (Ed.): APLAS 2006, LNCS 4279, pp. 183–199, 2006.

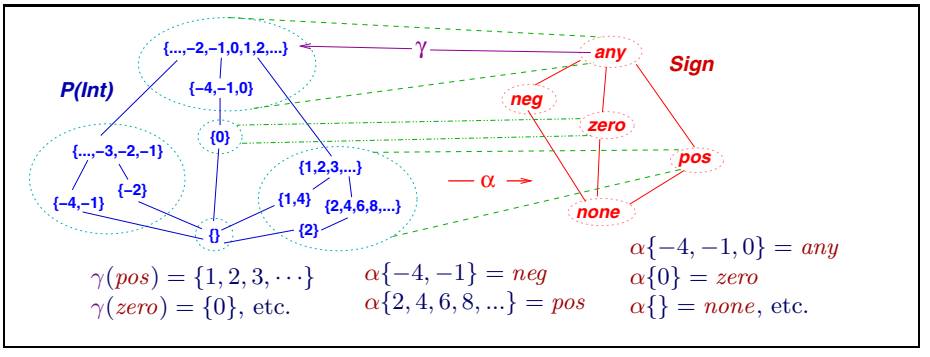

Fig. 1. Galois connection for signs; equivalence classes circled

and by proving equivalences, implications, and independences. By exposing two fundamental Galois connections that underlie logics for abstract values, we reveal yet another completeness variant, *O-logical-completeness*. We also show that the key concept underlying logical completeness notions is *covering*, which we use to relate the various forms of completeness.

1 Galois Connections and Functional Completeness

We use Galois connections to abstract concrete data into properties. A *Galois connection* [8,15] between two partially ordered sets, $(C, \subseteq)$ and $(A, \sqsubseteq)$, written $C\langle\alpha, \gamma\rangle A$, is a pair of functions, $\alpha : C \to A$ and $\gamma : A \to C$, such that for all $c \in C$ and $a \in A$,

$$c \subseteq \gamma(a) \text{ iff } \alpha(c) \sqsubseteq a.$$

The adjunction is equivalently defined by requiring that α and γ are monotone maps such that $id_{C\to C} \sqsubseteq \gamma \circ \alpha$ and $\alpha \circ \gamma \sqsubseteq id_{A\to A}$.

C is the *concrete domain* and A is the *abstract domain*. γ's adjoint, α, is uniquely defined as $\alpha(c) = \sqcap\{a \mid c \subseteq \gamma(a)\}$ and α's adjoint must be $\gamma(a) = \cup\{c \mid \alpha(c) \sqsubseteq a\}$[15]. γ is an *upper adjoint* of a Galois connection iff it preserves meets: $\gamma(\sqcap T) = \cap_{a\in T}\gamma(a)$, for all $T \subseteq A$. Similarly, α is a *lower adjoint* iff it preserves joins: $\alpha(\cup S) = \sqcup_{c\in S}\alpha(c)$, for all $S \subseteq C$ [15].

Figure 1 displays the classic Galois connection that abstracts sets of integers to their signs [8]. (In the Figure, C is $\mathcal{P}(Int)$ and A is $Sign$.)Each $S \in \mathcal{P}(Int)$ is abstracted to $\alpha(S) \in Sign$. Values like pos and any can be read as primitive logical propositions ($isPositive$ and $true$, respectively) or they can be used as abstract arguments and answers to static-analysis functions (e.g. $succ^\sharp(zero) = pos$). The Galois connection is *overapproximating* because $S \subseteq \gamma(\alpha(S))$, for all $S \in \mathcal{P}(C)$.

The following little-known result [21] exposes the inner structure of Galois connections:[1] *There is a Galois connection between* $(C, \subseteq)$ *and* $(A, \sqsubseteq)$ *iff*

[1] In this paper, definitions and previously proved results are embedded into the text narrative. New results and new variations of known results are stated as Propositions, Theorems, and Corollaries. Due to lack of space, some proofs are omitted but can be found in the paper's accompanying technical report [28].