A taxonomy of program analyses

Gianluca Amato, Maria Chiara Meo, and Francesca Scozzari

Dipartimento di Economia, Università di Chieti-Pescara, Italy
{gianluca.amato, mariachiara.meo, francesca.scozzari}@unic.it

Abstract. The design of static analyses of programs in the abstract interpretation theory starts with the choice of a collecting semantics, which is the strongest property we can derive for a program. Starting from well-known collecting semantics for functional programs in the literature, we propose a taxonomy of program properties by considering the sets of abstract interpretations for which the collecting semantics is initial and show that they can be constructively characterized in terms of the abstraction functions.

1 Introduction

Abstract interpretation [7, 8] is a framework for approximating the behavior of discrete systems. The basic idea is to replace the formal semantics of a system with an abstract semantics computed over a domain of abstract objects, which describe the properties of the system we are interested in. Given a discrete system e, we assume that its semantics $\llbracket e \rrbracket$ is an element of a set $C$ called the concrete domain. For instance, consider a simple setting for functional programs 1 where the concrete domain $C$ is the poset of functions $D_\bot \rightarrow D_\bot$ ordered pointwise and $D_\bot$ has a distinguished value $\bot$ which represents non-terminating computations.

Given the following program over natural numbers:

\begin{verbatim}
let rec prog n = if (n = 2) then prog (n) else n
\end{verbatim}

its semantics is

$$\llbracket prog \rrbracket = \lambda n. \begin{cases} \bot & \text{if } n = 2 \text{ or } n = \bot; \\ n & \text{otherwise}, \end{cases}$$

where $D_\bot = \mathbb{N} \cup \{\bot\}$.

In most cases we are only interested in determining some properties of the semantics of a system specified by a set $A$ of abstract objects. Formally, an abstract interpretation 2 for a concrete domain $C$ is a pair $(A, \alpha)$ where $A$ is partially ordered by $\leq_A$ (the approximation ordering) and $\alpha : C \rightarrow A$ maps every semantic property to the strongest (smallest) abstract property it enjoys.

---

1 For easiness of presentation, we do not consider here higher-order functions.
2 The abstract interpretation framework here used is the one presented in [9] under the “existence of a best abstract approximation assumption”. Not all the abstract interpretations may be formalized in this way, such as polyhedral analysis [11, 3, 2, 4, 5].
example of a simple and useful abstraction is strictness [13]. We say that a function \( f : D_\bot \rightarrow D_\bot \) is strict if \( f(\bot) = \bot \). The strictness abstract interpretation is \( \text{Str} = (\{\text{str, } \top\}, \alpha_{\text{str}}) \) where \( \text{str} \leq \top \) and
\[
\alpha_{\text{str}}(f) = \begin{cases} 
\text{str} & \text{if } f(\bot) = \bot, \\
\top & \text{otherwise.}
\end{cases}
\]
Clearly, \( \alpha_{\text{str}}([\text{prog}]) = \text{str} \) since \( \text{prog} \) describes a strict function.

1.1 Collecting Semantics
In most cases abstract interpretations are defined starting from a so-called collecting semantics. According to [9], a collecting semantics is “a version of the standard semantics reduced to essentials in order to ignore irrelevant details about program execution”. Thus, a collecting semantics should be an abstraction precise enough that many other abstractions may be derived from it. We say that the abstraction \((B, \alpha_B)\) may be derived from \((A, \alpha_A)\), or that \((A, \alpha_A)\) is more precise than \((B, \alpha_B)\), when for each \( b \in B \) there exists \( a \in A \) such that, for any \( c \in C \),
\[
\alpha_B(c) \leq b \iff \alpha_A(c) \leq a.
\]
For example, consider the collecting semantics \( \text{CS}_1 \) for the concrete domain \( C = D_\bot \rightarrow D_\bot \) defined in [10]: the abstract domain is \( \mathcal{P}(D_\bot) \cup \mathcal{P}(D_\bot) \), the set of complete join-morphisms from \( \mathcal{P}(D_\bot) \) to itself ordered pointwise, and \( \alpha_{\text{CS}_1}(f) = \lambda X \in \mathcal{P}(D_\bot). f(X) \), where \( f(X) \) is the image of \( f \) through \( X \). It turns out that strictness may be derived from \( \text{CS}_1 \). Actually, we have that:
- \( \alpha_{\text{str}}(f) \leq \top \) is always true, hence it is equivalent to \( \alpha_{\text{CS}_1}(f) \leq \lambda X. D_\bot \);
- \( \alpha_{\text{str}}(f) \leq \text{str} \) means that \( f \) is a strict function, which is equivalent to \( \alpha_{\text{CS}_1}((\bot)) \subseteq (\bot) \), i.e., \( \alpha_{\text{CS}_1}(f) \leq \phi_{\text{str}} \) by defining
\[
\phi_{\text{str}} = \lambda X. \begin{cases} X & \text{if } X \subseteq (\bot), \\
D_\bot & \text{otherwise.}
\end{cases}
\]
However, not all the abstractions may be recovered from \( \text{CS}_1 \). Consider the property of absence. We say that a function is absent if it ignores its arguments. This gives origin to the absence abstract interpretation [16] defined as \( \text{Abs} = (\{\text{abs, } \top\}, \alpha_{\text{abs}}) \) where \( \text{abs} \leq \top \) and
\[
\alpha_{\text{abs}}(f) = \begin{cases} \text{abs} & \text{if } \forall x \in D_\bot. f(x) = f(\bot), \\
\top & \text{otherwise.}
\end{cases}
\]
It turns out that absence cannot be recovered from \( \text{CS}_1 \), since there is no element in the abstract domain of \( \text{CS}_1 \) which exactly corresponds to the set of all the absent functions. However, more precise collecting semantics may be used to derive \( \text{abs} \), such as \( \text{CS}_2 [10] \) which has \( \mathcal{P}(D_\bot \rightarrow D_\bot) \) as the abstract domain and \( \alpha_{\text{CS}_2}(f) = \{f\} \) as the abstraction function. Then, \( \alpha_{\text{abs}}(c) \leq \text{abs} \) is equivalent to \( \alpha_{\text{CS}_2}(f) \subseteq \{f \mid \alpha_{\text{abs}}(f) \leq \text{abs}\} \). We can also compare the relative precision of two collecting semantics. It is easily shown that \( \text{CS}_1 \) may be derived from \( \text{CS}_2 \), since \( \alpha_{\text{CS}_1}(f) \leq \phi \) iff \( \alpha_{\text{CS}_2}(f) \subseteq \{f' \mid \alpha_{\text{CS}_1}(f') \leq \phi\} \).
2 A category of abstract interpretations

We now formalize the notion of precision which arises from the previous considerations and characterize the collecting semantics using the category theory\(^3\). We recall that a Galois connection (GC) is a pair of maps \((\alpha, \gamma) : A \leftrightarrows B\) such that \(\alpha(a) \leq b \leftrightarrow a \leq \gamma(b)\). Given a set \(C\), we define the category \(\mathbb{A}_i(C)\) whose objects are abstract interpretations and whose morphisms from \((A, \alpha_A)\) to \((B, \alpha_B)\) are Galois connections \((\alpha, \gamma) : A \leftrightarrows B\) such that \(\alpha_B = \alpha \circ \alpha_A\). Identity and composition of arrows work as in the category of Galois connections. This definition of the category of abstract interpretations naturally arises from the practical uses of abstract interpretation and represents a strengthening of the notion of derivability between abstractions, since the existence of a morphism \((\alpha, \gamma) : A \leftrightarrows B\) implies that \(B\) is derivable from \(A\). The following proposition shows that there is a GC from \(CS_2\) to \(CS_1\) but not from \(CS_1\) to \(CS_2\), which reflects our intuition that \(CS_2\) is strictly more precise than \(CS_1\), and that \(CS_1\) is a suitable collecting semantics for strictness but not for the absence property.

**Proposition 1.** The following properties hold:

- there is no GC \((\alpha_{12}, \gamma_{12}) : CS_1 \leftrightarrows CS_2\) such that \(\alpha_{CS_2} = \alpha_{12} \circ \alpha_{CS_1}\);
- there is a GC \((\alpha_{21}, \gamma_{21}) : CS_2 \leftrightarrows CS_1\) such that \(\alpha_{CS_1} = \alpha_{21} \circ \alpha_{CS_2}\);
- there is a GC \((\alpha_{1, str}, \gamma_{1, str}) : CS_1 \leftrightarrows \text{STR}\) such that \(\alpha_{str} = \alpha_{1, str} \circ \alpha_{CS_1}\) and
  \[
  \alpha_{1, str}(\phi) = \begin{cases} 
  \text{str} & \text{if } \phi(\{\bot\}) \subseteq \{\bot\} \\
  \top & \text{otherwise.}
  \end{cases}
  \]
  \[
  \gamma_{1, str}(\text{str}) = \lambda X. \begin{cases} 
  X & \text{if } X \subseteq \{\bot\} \\
  \mathcal{D}_\bot & \text{otherwise}
  \end{cases}
  \]
  \[
  \gamma_{1, str}(\top) = \lambda X. \mathcal{D}_\bot
  \]
- there is no GC \((\alpha_{1, abs}, \gamma_{1, abs}) : CS_1 \leftrightarrows CS_{abs}\) such that \(\alpha_{abs} = \alpha_{1, abs} \circ \alpha_{CS_1}\).

3 Subcategories of program analyses

When an abstract interpretation \((A, \alpha_A)\) is designed starting from the collecting semantics \((S, \alpha_S)\), it means that \((A, \alpha_A)\) is derivable from \((S, \alpha_S)\), hence there is a map from \((S, \alpha_S)\) to \((A, \alpha_A)\) in our category \(\mathbb{A}_i(C)\). In addition, it would be preferable to have a canonical way of deriving one from the other. This leads to the notion of initial object in a category: if \((S, \alpha_S)\) is initial for a given full subcategory \(\mathcal{D}\) of \(\mathbb{A}_i(C)\), it means that all the abstract interpretations in \(\mathcal{D}\) may be reduced to \((S, \alpha_S)\) in a unique canonical way.

Given a collecting semantics \((S, \alpha_S)\), we are interested in characterizing the maximal full subcategory \(\mathcal{D}\) of \(\mathbb{A}_i(C)\) such that \((S, \alpha_S)\) is initial for \(\mathcal{D}\). Such subcategories immediately induce a taxonomy on program analysis which precisely characterizes the program properties suitable for a given collecting semantics.

\(^3\) Since the very beginning of the abstract interpretation theory, there have been work on categorical approaches to abstract interpretation (see, for instance, [1, 6, 14, 15]).

3
In the rest of the section, we show that for the two collecting semantics $CS_1$ and $CS_2$, such a full subcategory can be constructively described.

We first show that $CS_2$ is initial for all the abstract interpretations which have enough joins, and that this is the largest class of abstract interpretations which enjoys this property.

**Definition 2 (Having enough joins).** We say that the abstract interpretation $(A, \alpha_A)$ has enough joins when $\bigvee_{\alpha_A X}$ exists for each $X$ which is a subset of the image of $\alpha_A$.

Obviously, if $A$ is a complete join-semilattice, than $(A, \alpha_A)$ has enough joins.

**Theorem 3.** The full subcategory of all the abstract interpretations which have enough joins is the largest class of abstractions for which the collecting semantics $CS_2$ is initial.

The maximality of the class of abstract interpretations which have enough joins allows us to completely characterize the analyses which reduce to the collecting semantics $CS_2$. We now show that the collecting semantics $CS_1$ is initial for a large class of abstract interpretations, which can be constructively characterized.

**Definition 4 (Mix of functions).** We say that a function $g : \mathcal{D}_\perp \to \mathcal{D}_\perp$ is a mix of the set of functions $F \subseteq \mathcal{D}_\perp \to \mathcal{D}_\perp$ iff for each $x \in \mathcal{D}_\perp$ there exists $f \in F$ such that $g(x) = f(x)$.

**Definition 5 (Mixable interpretations).** Let $(A, \alpha_A)$ be an abstract interpretation such that $A$ has enough joins. We call $(A, \alpha_A)$ mixable if, for any set of functions $F \subseteq \mathcal{D}_\perp \to \mathcal{D}_\perp$, whenever $g$ is a mix of functions in $F$, we have $\alpha_A(g) \leq \bigvee_{f \in F} \alpha_A(f)$.

An interpretation is mixable when deciding whether $\alpha(f) \leq a$ may be done by looking at the values of $f$ for a single element of the domain at a time. This observation is formalized by the following lemma.

**Lemma 6.** Let $(A, \alpha_A)$ be a mixable interpretation. Then $\alpha_A(f) \leq a \iff \forall x \in \mathcal{D}_\perp \exists f' \text{ s.t. } f'(x) = f(x) \land \alpha_A(f') \leq a$.

This lemma can be exploited to characterize mixable abstract interpretations:

- the strictness abstract interpretation is mixable, since we only need to check the single value of $f(\perp)$ in order to decide whether a function is strict;
- the absence abstract interpretation is not mixable, since we need to compare the values computed by $f$ for different arguments;
- the totality abstract interpretation, which decides whether a function is total, i.e., for all $x \in \mathcal{D}_\perp \setminus \{\perp\}$ we have that $f(x) \neq \perp$ is mixable, since we just need to see the value of $f$ for (many) single arguments, without the need of comparing them.

We now show that all the mixable interpretations may be designed starting from the collecting semantics $CS_1$.

**Theorem 7.** The full subcategory of all the mixable abstract interpretations is the largest class of abstractions for which the collecting semantics $CS_1$ is initial.
4 Conclusion

Static analyses formalized in the abstract interpretation framework are defined starting from a collecting semantics, which is a fundamental choice in the design of any abstract interpretation, but few works in the literature systematically study collecting semantics. [9,10] present many collecting semantics, without giving a general method for choosing the right one. Mostly authors simply use one of the already defined collecting semantics or invent a new one for a specific abstraction (e.g., [12] for logic programs). On the contrary, we start from the definition of two most commonly used collecting semantics and derive a taxonomy on program analysis. Most importantly we show that the sets of abstract interpretations that can be derived from them can be constructively characterized.

References