Notes on Abstract Interpretation*

Alexandru Sălcianu
salcianu@mit.edu

November 2001

1 Introduction

This paper summarizes our view of the abstract interpretation field. It is based on the original abstract interpretation papers of Cousot and Cousot [1, 2], and on the more recent presentation of Nielsen et al [3, Chapter 4]. This paper does not try to give an exhaustive view of this field — such an overview would clearly require a lot of space. Instead, we tried to see what abstract interpretation is and to identify those things that might be useful to know when designing a novel program analysis. Funny math subtleties are not covered.

Briefly, abstract interpretation is a technique for approximating a basic analysis (at the limit, the concrete execution of the program), with a refined analysis that sacrifices precision for speed. Abstract interpretation expresses the connection between the two analyses using a *Galois connection* between the associated *property lattices* (all these terms are well explained in the paper). Ideally, given a basic analysis and a Galois connection, one could "compute" the refined analysis; however, in most of the cases this is too hard to be practical: usually, one starts with both analyses and the Galois connection, and verifies that they satisfy the conditions from the abstract interpretation theory. One interesting observation of this paper is that the connection between the basic and the refined analysis does not have to be a Galois connection. In many practical cases, all we need to reason about the analysis correctness is an *concretization* function (see Section 4.5).

If you feel interested, please read on! The rest of this paper is structured as follows: Section 2 presents the main ideas of abstract interpretation. Section 3 studies the case of a simple analysis and introduces the notion of "correctness relation". In Section 4, we show how a costly analysis over a big property lattice can be approximated by an analysis over a smaller property lattice, and the connection that has to exist between the two latices in order for the new analysis to be correct.

*This paper should be cited as "Alexandru Sălcianu. Notes on Abstract Interpretation. Unpublished manuscript. Available from `http://www.mit.edu/~salcianu`."

2 Abstract Interpretation - Main Ideas

The Monotone [Dataflow] Framework [3, Chapter 2] allows the program analysis designer to specify an analysis in a precise, mathematical format. In this framework, one has to specify the property lattice, the meaning of the elements of this lattice (i.e., how the analysis results are interpreted), and the transfer functions associated with the basic instructions. Almost all currently used static analyses can be expressed in the Monotone Framework. The main drawback of this framework is that the correctness proof for each analysis has to be done in an *ad-hoc* way, usually based on some simulation invariants.[1]

Abstract Interpretation goes one step beyond the Monotone Framework by allowing the designer not only to specify an analysis, but also to specify it in a way that will make the correctness proof mechanical.[2] Moreover, it makes it possible to systematically construct (Cousot's term is "compute") an analysis: once a property lattice and a representation function[3] are chosen, the transfer functions can be defined in a systematic way: we just have to select some computable functions that satisfy some property that we'll see later (they have to be "bigger" — i.e., more conservative — than a certain function). Therefore, the Abstract Interpretation emphasizes the importance of the abstraction function.

Funny note: Very often, the abstraction function can be computed by mechanically combining some building blocks, which should allow the fast construction of new analyses. In [2, Section 10], Cousot and Cousot claim that:

> "The ideal method in order to construct a program analyzer [...] would consist in a separate design and implementation of various program analysis frameworks which could be then systematically combined using a once for all implemented assembler."

Disturbing, isn't it? When everything is so "systematic", one could ask if there is any reason to do research in the program analysis field. However, our work is not in vain, because [2, Section 10] continues as follows:

> "[...] we show that such an automatic combination of independently designed parts would not lead to an optimal analyzer and that unfortunately the efficient combination of program analyses often necessitates the revision of the original design phase."

Still, even if systematic combination of building blocks is not always the key for an efficient analysis, it is supposed to give some precious insight into the analysis.

Abstract Interpretation advocates the incremental definition of an analysis. The typical design is as follows: We start with a very simple analysis, whose correctness is easy to prove (correctness of such an analysis is formalized in Section 3). Usually, this "analysis" is a "collecting semantics" that collects precise information about the program. This first analysis might be too difficult or even impossible to compute statically, usually because the

[1] Based on personal experience, we can say that choosing the invariants is usually more difficult than proving them.

[2] However, it's worth noting that "mechanical" is not a synonym for "easy" ;-(

[3] A function that describes how the elements of the property lattice are to be interpreted.

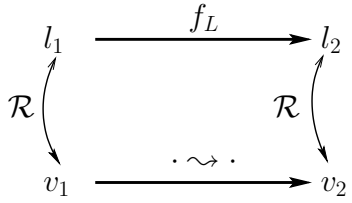


Figure 1: Basic Analysis

property space is too big or does not respect the Ascending Chain Property.[4] Therefore, we compute an approximation of it, in a smaller property space. We can iterate this refinement technique several times, till we obtain a computable analysis. In [1], the Cousots even suggested that the different analyses we obtain can be arranged in a lattice with a specific ordering relation.[5]

3 Correctness Relations

The execution of a program can be described as a succession of transitions between "concrete" values/states from the set V:

$$v_0 \rightsquigarrow v_1 \rightsquigarrow \cdots \rightsquigarrow v_i \rightsquigarrow \cdots$$

The concrete execution may be non-deterministic, i.e., the transition relation $\rightsquigarrow$ is not necessarily a function (e.g., consider a language that allows multiple threads of execution).

A program analysis is an "abstract execution" of the program, in a property space L, as indicated in Figure 1. The property space L should be a complete lattice $\langle L, \sqsubseteq \rangle$, i.e., a set L with an ordering relation[6] $\sqsubseteq \subseteq L \times L$ such that each subset of L, $L' \subseteq L$, has a least upper bound $\bigsqcup L'$ and a greatest lower bound $\bigsqcap L'$ *in* L.[7] Instead of values, the analysis works with properties (an equivalent term is "abstract values") $l_i \in L$:

$$l_0 \mapsto l_1 = f_L(l_0) \mapsto \cdots \mapsto l_i \mapsto l_{i+1} = f_L(l_i) \cdots$$

where $f_L : L \to L$ is a function (we look only at deterministic analyses). Intuitively, l_i "models" the value v_i, in a way that is problem dependent. More formally, we have a *correctness relation* (definition below) $\mathcal{R}$ such that $v_i \ \mathcal{R} \ l_i$.

Definition 1 (Correctness relation). *A relation $R \subseteq V \times L$ is said to be a* correctness relation *iff it satisfies the following two conditions:*

1. $\forall v, l_1, l_2,\ (v \ \mathcal{R} \ l_1) \wedge (l_1 \sqsubseteq l_2) \to (v \ \mathcal{R} \ l_2)$

[4] A lattice has the "Ascending Chain Property" iff there are no infinite ascending chains in it.

[5] Viktor Kuncak found an error in Cousot's proof (page x in [1]): a domain mismatch in one of the function composition operations. However, this error does not affect the Abstract Interpretation framework.

[6] A relation that is reflexive ($\forall l \in L,\ l \sqsubseteq l$), transitive ($\forall l_1, l_2, l_3 \in L,\ ((l_1 \sqsubseteq l_2) \wedge (l_2 \sqsubseteq l_3)) \to (l_1 \sqsubseteq l_3)$), and anti-symmetric ($\forall l_1, l_2 \in L,\ ((l_1 \sqsubseteq l_2) \wedge (l_2 \sqsubseteq l_1)) \to (l_1 = l_2)$).

[7] There are two special values in a complete lattice L: "bottom", the smallest element, $\bot_L = \bigsqcup_L \emptyset$, and "top", the biggest element, $\top_L = \bigsqcap_L \emptyset$.

2. $\forall v, \forall L' \subseteq L,\ (\forall l \in L',\ (v\ \mathcal{R}\ l)) \rightarrow v\ \mathcal{R}\ (\bigsqcap L')$

Due to the first condition, if l_1 approximates v and the analysis is able to compute an upper approximation l_2 of l_1, $l_1 \sqsubseteq l_2$, l_2 is also an approximation of v. Therefore, in the property lattice L, if $l_1 \sqsubseteq l_2$, then l_1 is more precise than l_2 (l_2 approximates all the values approximated by l_1 and possibly some other ones).

If a value is approximated by more than one abstract value, the second condition allows us to deterministically select a single abstract value to use in our analysis: we take the smallest (i.e., most precise) of them all, $l_p = \bigsqcap\{l \mid v\ \mathcal{R}\ l\}$. l_p is a valid approximation of v. This condition excludes correctness relations where a value v is approximated by two incomparable abstract values, and allows us to work with a single analysis instead of considering one analysis for each candidate abstract value.[8]

Lemma 1. *If $\mathcal{R} \subseteq V \times L$ is a correctness relation, then:*

$$v\ \mathcal{R}\ \top$$
$$(v\ \mathcal{R}\ l_1) \wedge (v\ \mathcal{R}\ l_2) \ \rightarrow\ v\ \mathcal{R}\ (l_1 \sqcap l_2)$$

Proof: Direct application of the second condition from Definition 1, for $L' = \emptyset$, respectively $L' = \{l_1, l_2\}$. □

Note: the top element of the property lattice, $\top$, approximates all the values; therefore, it is the most imprecise abstract value.

Discussion on the nature of the property space: The requirement that the property space L is a lattice is reasonable: we need to compare abstract values with regard to their precision; therefore, it is natural to have some ordering relation. Also, we need the meet and the join operator in order to combine abstract values:

- If a value is described by *both* l_1 and l_2, by combining these two properties, we obtain the more precise information $l_1 \sqcap l_2$.
- If a value is described by *either* l_1 or l_2, the most precise info that we can infer is $l_1 \sqcup l_2$.

Notice that in the property lattice, "smaller" means "more precise", "bigger" means "safer", $\sqcap$ is the equivalent of a logical $\wedge$, and $\sqcup$ is the equivalent of a logical $\vee$. However, do not be surprised if some people work with the opposite convention!

Meet over paths:[9] Suppose we have an abstract initial value l_0, such that $v_0\ \mathcal{R}\ l_0$ where v_0 is the initial concrete value (state). Given a program point, we can consider all concrete execution paths that reach it, "abstractly execute" each of them starting from v_0 to compute an abstract value, and join the resulting abstract values to obtain an element of the property

[8] This is of course nice (and is also valid for virtually all the analyses we know), but, at least in our opinion, is not essential for the correctness of an analysis. We discuss more on this issue at the end of Section 4.

[9] This is the "historical" term; technically speaking, it would be more appropriate to call it "Join over paths."

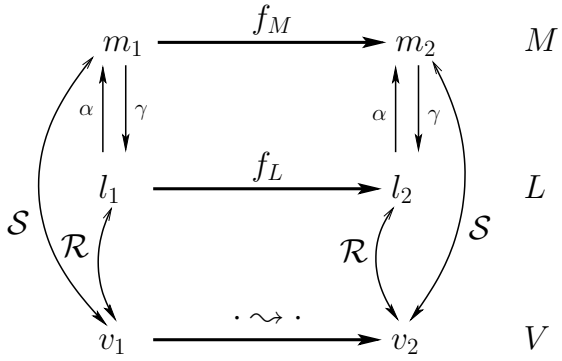


Figure 2: Refined Analysis

lattice that approximates all values (states) possible for that program point. It is usually not possible to compute the "meet over paths"; therefore, we approximate its result by computing a fixed point of a set of dataflow equations. To be sure that a fixed point exists, we require that:

1. The analysis transfer function f_L is monotone.
2. The lattice L has the "Ascending Chain Property": there is no infinite ascending chain in L.

Note that so far, except for the formal notion of *correctness relation*, there is no difference from the classic Monotone Dataflow Framework [3, Chapter 2].

Correctness issues: To prove the correctness of the analysis, it is sufficient to prove

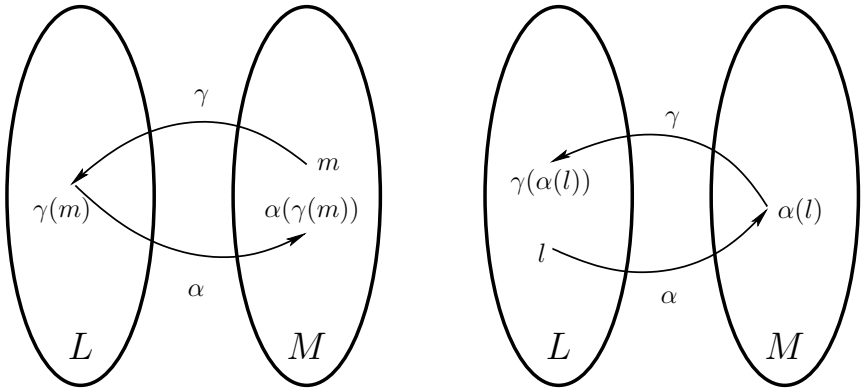
1. The initial property (abstract state) l_0 is a correct approximation of the initial value (concrete state) v_0: $v_0 \ \mathcal{R} \ l_0$.
2. Each transition preserves the correctness relation, i.e.,

$$\forall v_1, v_2, l_1, l_2, \ (v_1 \rightsquigarrow v_2) \wedge (v_1 \ \mathcal{R} \ l_1) \wedge (f_L \, (l_1) = l_2) \rightarrow v_2 \ \mathcal{R} \ l_2$$

Once we prove this, an elementary induction on the length of the execution path shows that the result of "meet-over-paths" for a specific program point describes all the concrete values that can occur at that program point (and possibly some other values too), with respect to the correctness relation $\mathcal{R}$.

4 Refining the analysis

Sometimes, the fixed point computation in L is too difficult or even impossible to perform, e.g., the analysis converges very slow or L does not satisfy the Ascending Chain Property. In this case, abstract interpretation recommends using a smaller, more approximate property lattice, M, such that there is a Galois connection $\langle L, \alpha, \gamma, M \rangle$ between L and M. Based on



$\alpha \circ \gamma \sqsubseteq \lambda m.m$ $\gamma \circ \alpha \sqsupseteq \lambda l.l$

Figure 3: Constraints for a Galois connection

the correctness relation $\mathcal{R}$ of the basic analysis and the Galois connection between L and M, we can define a correctness relation $\mathcal{S}$ for a new analysis that computes abstract values from the lattice M. The situation is depicted in Figure 2.

This section continues as follows: first, we introduce the concept of a Galois connection. Next, we show how to define the correctness relation $\mathcal{S}$ and the transfer function f_M for the new, more approximate analysis. After we look into some properties of the Galois connections, we prove that $\mathcal{S}$ is a correctness relation that is preserved by the new analysis.

4.1 Galois Connections

Definition 2 (Galois Connection). $\langle L, \alpha, \gamma, M \rangle$ *is a* Galois connection *between the lattices* $\langle L, \sqsubseteq_L \rangle$ *and* $\langle M, \sqsubseteq_M \rangle$ *iff:*

1. $\alpha : L \to M$ *and* $\gamma : M \to L$*;*
2. α *and* γ *are monotonic;*
3. $\alpha \circ \gamma \sqsubseteq_M \lambda m.m$*;*
4. $\gamma \circ \alpha \sqsupseteq_L \lambda l.l$*.*

Figure 3 presents a graphic depiction of a Galois connection. α maps elements from L to elements from the more approximate (more abstract) lattice M. Therefore, it is called the *abstraction* function. Conversely, γ is called the *concretization* function.

The monotonicity restriction is common-sense: it says that the abstraction and the concretization operations preserve the ordering (i.e., the precision). The last two restrictions are the most interesting:

- the first restriction, $\alpha \circ \gamma \sqsubseteq_M \lambda m.m$, says that the concretization doesn't lose any precision (in many cases, this relation is an equality; see Section 6.1);
- the second one, $\gamma \circ \alpha \sqsupseteq_L \lambda l.l$, says that the abstraction might lose precision, but remains correct (remember that in L, bigger means more imprecise but still correct).

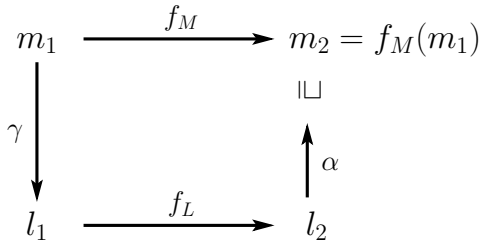


Figure 4: Transfer function for the refined analysis.

4.2 New Analysis: Correctness Relation and Transfer Function

The correctness relation $\mathcal{S} \subseteq V \times M$ for the new analysis is defined as follows:

$$v \; \mathcal{S} \; m \text{ iff } v \; \mathcal{R} \; \gamma\,(m)$$

For the new analysis transfer function, we can choose any function $f_M : M \to M$ such that:

$$f_M \sqsupseteq \alpha \circ f_L \circ \gamma$$

We shall prove later that $\mathcal{S}$ is indeed a correctness relation and that it is preserved by the transfer function f_M.

4.3 Auxiliary Results about Galois Connections

Lemma 2. *Iterating abstraction and concretization does not improve the precision:*

$$\begin{aligned} \alpha \circ \gamma \circ \alpha &= \alpha \\ \gamma \circ \alpha \circ \gamma &= \gamma \end{aligned}$$

Proof: As α is monotonic and $\gamma \circ \alpha \sqsupseteq \lambda l.l$ (by the definition of a Galois connection),

$$\forall l \in L, (\alpha \circ \gamma \circ \alpha)\,(l) = \alpha\,((\gamma \circ \alpha)\,(l)) \sqsupseteq \alpha\,(l)$$

Also, as $\alpha \circ \gamma \sqsubseteq \lambda m.m$, by putting $m = \alpha\,(l)$, we have that

$$\forall l \in L, (\alpha \circ \gamma \circ \alpha)\,(l) = (\alpha \circ \gamma)\,(\alpha\,(l)) \sqsubseteq \alpha\,(l)$$

This proves that $\alpha \circ \gamma \circ \alpha = \alpha$. The other relation is similar. □

Here is an alternative formulation of a Galois connection:

Definition 3 (Adjunction). *$\langle L, \alpha, \gamma, M\rangle$ is an adjunction between complete lattices $\langle L, \sqsubseteq_L\rangle$ and $\langle M, \sqsubseteq_M\rangle$ iff $\alpha : L \to M$ and $\gamma : M \to L$ are total functions such that*

$$\alpha\,(l) \sqsubseteq_M m \;\Leftrightarrow\; l \sqsubseteq_L \gamma\,(m)$$

The two formulations are equivalent, as the following lemma states:

Lemma 3. *$\langle L, \alpha, \gamma, M\rangle$ is an adjunction iff $\langle L, \alpha, \gamma, M\rangle$ is a Galois connection.*

Proof: Suppose $\langle L, \alpha, \gamma, M\rangle$ is a Galois connection. Suppose $\alpha(l) \sqsubseteq m$; as γ is monotonic, $\gamma(\alpha(l)) \sqsubseteq \gamma(m)$. But $\gamma(\alpha(l)) \sqsupseteq l$ by the definition of a Galois connection. Hence, $\alpha(l) \sqsubseteq m \;\Rightarrow\; l \sqsubseteq \gamma(m)$. The other implication is similar, and we prove that $\langle L, \alpha, \gamma, M\rangle$ is an adjunction.

Now, suppose $\langle L, \alpha, \gamma, M\rangle$ is an adjunction. First, $\gamma(\alpha(l)) \sqsupseteq l \;\Leftrightarrow\; \alpha(l) \sqsupseteq \alpha(l)$, which is trivially true. So, $\gamma \circ \alpha \sqsupseteq \lambda l.l$. Similarly, $\alpha \circ \gamma \sqsubseteq \lambda m.m$. The only remaining thing is the monotonicity of α and γ. Consider $l_1 \sqsubseteq l_2$. We already know that $l_2 \sqsubseteq \gamma(\alpha(l_2))$. So, $l_1 \sqsubseteq \gamma(\alpha(l_2))$, which implies $\alpha(l_1) \sqsubseteq \alpha(l_2)$, i.e., α is monotonic. The proof of γ's monotonicity is similar. □

Lemma 4. *If $\langle L, \alpha, \gamma, M\rangle$ is a Galois connection then α uniquely determines γ by*

$$\gamma(m) = \bigsqcup\{l \mid \alpha(l) \sqsubseteq m\}$$

and γ uniquely determines α by

$$\alpha(l) = \bigsqcap\{m \mid l \sqsubseteq \gamma(m)\}$$

Proof: $\gamma(m) = \bigsqcup\{l \mid l \sqsubseteq \gamma(m)\} = \bigsqcup\{l \mid \alpha(l) \sqsubseteq m\}$ (we've used the fact that any Galois connection is also an adjunction). If $\langle L, \alpha, \gamma_1, M\rangle$ and $\langle L, \alpha, \gamma_2, M\rangle$ are Galois connections, it is trivial to show that $\forall m \in M,\ \gamma_1(m) = \gamma_2(m)$ which proves that α uniquely determines γ. The other half of the lemma is similar. □

Lemma 5. *If $\langle L, \alpha, \gamma, M\rangle$ is a Galois connection then α is completely additive and γ is completely multiplicative, i.e.,*

$$\begin{array}{lll} \forall L' \subseteq L, & \alpha(\bigsqcup L') & = \bigsqcup\{\alpha(l) \mid l \in L'\} \\ \forall M' \subseteq M, & \gamma(\bigsqcap M') & = \bigsqcap\{\gamma(m) \mid m \in M'\} \end{array}$$

In particular, $\alpha(\bot_L) = \bot_M$ and $\gamma(\top_M) = \top_L$.

Proof: Consider $L' \subseteq L$ and an arbitrary $m \in M$. By using the fact that any Galois connection is also an adjunction, we have that:

$$\begin{array}{lll} \alpha(\bigsqcup L') \sqsubseteq m & \Leftrightarrow & \bigsqcup L' \sqsubseteq \gamma(m) \\ & \Leftrightarrow & \forall l \in L',\ l \sqsubseteq \gamma(m) \\ & \Leftrightarrow & \forall l \in L',\ \alpha(l) \sqsubseteq m \\ & \Leftrightarrow & \bigsqcup\{\alpha(l) \mid l \in L'\} \sqsubseteq m \end{array}$$

By indirect equality[10], $\alpha(\bigsqcup L') = \bigsqcup\{\alpha(l) \mid l \in L'\}$, and α is completely additive. In particular, $\alpha(\bot_L) = \alpha(\bigsqcup_L \emptyset) = \bigsqcup_M \emptyset = \bot_M$. The other half of the lemma is similar. □

Lemma 6. *If $\alpha : L \to M$, is completely additive then there exists $\gamma : M \to L$ such that $\langle L, \alpha, \gamma, M\rangle$ is a Galois connection. Similarly, if $\gamma : M \to L$ is completely multiplicative, then there exists $\alpha : L \to M$ such that $\langle L, \alpha, \gamma, M\rangle$ is a Galois connection.*

Proof Sketch: Given α or γ, consider γ (respectively α) defined as in the text of Lemma 4 and prove that they constitute a Galois adjunction. By Lemma 3, $\langle L, \alpha, \gamma, M\rangle$ is a Galois connection. Full proof in [3, pp. 237] □

[10] In a lattice $\langle L, \sqsubseteq\rangle$, $(\forall l,\ (l_1 \sqsubseteq l) \leftrightarrow (l_2 \sqsubseteq l)) \to (l_1 = l_2)$; to see why, just consider $l = l_1$, $l = l_2$, and apply $\sqsubseteq$'s anti-symmetry.

Note: Lemma 6 proves that for specifying the new, refined analysis, we need to specify just a new property lattice M and a multiplicative concretization function $\gamma : M \to L$ that specifies the meaning of M's elements. α can then be directly inferred.

4.4 Correctness of the New Analysis

First of all, we need to prove that $\mathcal{S}$ is indeed a correctness relation:

Lemma 7. *$\mathcal{S}$ is a correctness relation.*

Proof: First, consider $v \in V$ and $m_1, m_2 \in M$ such that $v \ \mathcal{S} \ m_1$ and $m_1 \sqsubseteq_M m_2$. Then $v \ \mathcal{R} \ \gamma(m_1)$, and, as $\gamma(m_1) \sqsubseteq_L \gamma(m_2)$ (γ is monotonic) and $\mathcal{R}$ is a correctness relation, we have that $v \ \mathcal{R} \ \gamma(m_2)$ and hence, $v \ \mathcal{S} \ m_2$.

Next, consider $v \in V$ and $M' \subseteq M$ such that, $\forall m \in M', v \ \mathcal{S} \ m$. By the definition of $\mathcal{S}$, $\forall m \in M', v \ \mathcal{R} \ \gamma(m)$, and, as $\mathcal{R}$ is a correctness relation:

$$v \ \mathcal{R} \ (\bigsqcap \{\gamma(m) \mid m \in M'\})$$

As γ is fully multiplicative,

$$\bigsqcap \{\gamma(m) \mid m \in M'\} = \gamma(\bigsqcap \{m \mid m \in M'\})$$

and we have proved that $v \ \mathcal{S} \ (\bigsqcap \{m \mid m \in M'\})$, which finishes our proof. □

Next, we show that the correctness relation $\mathcal{S}$ is preserved under computation. Consider first the following auxiliary result:

Lemma 8. $\gamma(f_M(m_1)) \sqsupseteq f_L(\gamma(m_1))$

Proof: By the definition of f_M,

$$f_M(m_1) \sqsupseteq (\alpha \circ f_L \circ \gamma)(m_1) = \alpha(f_L(\gamma(m_1)))$$

As γ is monotonic and $\gamma \circ \alpha \sqsupseteq \lambda l.l$, we have that

$$\gamma(f_M(m_1)) \sqsupseteq (\gamma \circ \alpha)(f_L(\gamma(m_1))) \sqsupseteq f_L(\gamma(m_1))$$

□

Note: Basically, the previous lemma tells that the analysis in M is a conservative[11] approximation of the analysis in L.

Lemma 9 ($\mathcal{S}$ is preserved under the computation).

$$\forall v_1, v_2, m_1, m_2, \ (v_1 \rightsquigarrow v_2) \wedge (v_1 \ \mathcal{S} \ m_1) \wedge (f_M(m_1) = m_2) \to v_2 \ \mathcal{S} \ m_2$$

Proof: By Lemma 8,

$$\gamma(m_2) \sqsupseteq f_L(\gamma(m_1))$$

As $\mathcal{R}$ is preserved under computation, $v_1 \rightsquigarrow v_2$ and $v_1 \ \mathcal{R} \ \gamma(m_1)$, we obtain that $v_2 \ \mathcal{R} \ f_L(\gamma(m_1))$. But $f_L(\gamma(m_1)) \sqsubseteq \gamma(m_2)$, and $\mathcal{R}$ is a correctness relation. Therefore, $v_2 \ \mathcal{R} \ \gamma(m_2)$ and hence, $v_2 \ \mathcal{S} \ m_2$, which completes our proof. □

[11] Remember that in the property lattice L "bigger", means more imprecise but safe.

4.5 Discussion

Patrick and Radhia Cousot [1, 2] claim that the relation between L and M, i.e., the relation between the basic and the refined analysis, has to be a Galois connection. We (respectfully) dare to disagree!

It is interesting to consider each requirement from the definition of a Galois connection and identify the parts of the formalism where that requirement is used. As we explained before, it is reasonable to demand the property spaces, L and M, to be complete lattices. The existence of a Galois connection between L and M is used to construct f_M (the transfer function in M) and to prove that $\mathcal{S}$ (the new correctness relation) respects the second condition from the definition of a correctness relation (Definition 1). More precisely, the proof of that condition uses the fact that γ is fully multiplicative, which is a direct consequence of the fact that $\langle L, \alpha, \gamma, M\rangle$ is a Galois connection ($\Leftrightarrow$ adjunction).

But Condition 2 of Definition 1 was imposed only to simplify the design (we always use the most precise abstract value from the possibly many candidates); we can imagine correct analyses that do not respect that condition. Consider the case when we relax Definition 1 by removing Condition 2. In this case, if we have an abstraction function $\gamma : M \to L$ and a transfer function $f_M : M \to M$ such that $\gamma(f_M(m_1)) \sqsupseteq f_L(\gamma(m_1))$, the correctness of the analysis in M (according to the new definition of the correctness relation) is trivial to prove (note that the proof of the fact that $\mathcal{S}$ is preserved under the computation requires just the aforementioned inequality). In an alternative formulation, we can require α and γ that respects conditions 1, 2 and 4 (but not 3) from the Definition 2 for a Galois connection, and choose $f_M \sqsupseteq \alpha \circ f_L \circ \gamma$ as before. Note that we have a weaker connection between L and M: we don't require $\alpha \circ \gamma \sqsubseteq \lambda m.m$.

We believe that the existence of a Galois connection is more of a "methodological" requirement that makes some parts of the formalism nicer and possibly allows some advanced extensions of the framework. We don't think it is essential for the intrinsic correctness of an analysis that is an approximation of another analysis.

5 Fixed Point Computation Issues

To compute the result of an analysis, we usually have to compute a fixed point of a monotonic transfer function $f : L \to L$, where L is a complete lattice.

This is usually done by computing the limit of an Ascending/Descending Kleene Sequence (denoted AKS, respectively DKS), $(f^n(\tau))_n$, where the start value τ is the bottom element of L, $\bot$, for AKS, respectively the top element, $\top$, for DKS. If AKS stabilizes, it will stabilize to the least fixed point of f, $lfp(f)$. This is not true in general but it is true if f is continuous[12], which is almost always the case.[13] Similarly, if DKS stabilizes, it stabilizes to an upper approximation of the greatest fixed point of f, $gfp(f)$. For the rest of this section, we focus on AKS (DKS is simply its dual).

[12] $\forall L' \subseteq L, f(\bigsqcup L') = \bigsqcup\{f(l) \mid l \in L'\}$.

[13] We usually work with finite property lattices, and on a finite lattice, continuity is equivalent to monotonicity.

However, sometimes the fixed point computation might be difficult to perform directly, e.g., because the AKS doesn't stabilize or stabilizes very slowly. For these cases, the Cousots [1] advocated the use of a technique called "widening" which computes an upper approximation of the AKS — an Ascending Approximation Sequence (AAS) — which converges faster than AKS. This technique will produce an upper approximation of $\mathit{lfp}\,(f)$. By applying a companion technique called "narrowing", we can compute a Truncated Descending Sequence (TDS) that goes down in the lattice toward $\mathit{lfp}\,(f)$, and stabilizes to a better (but still correct) approximation of it. Both techniques are based on binary operators $\nabla, \Delta : L \to L$. The rest of this section is structured as follows: we first introduce a few basic notions about fixed points in Section 5.1. We describe the widening technique in Section 5.2 and the narrowing technique in Section 5.3.

5.1 Basic Fixed Points Notions

Consider a monotone function $f : L \to L$ on a complete lattice $L = \langle L, \sqsubseteq, \sqcup, \sqcap, \bot, \top \rangle$. We introduce the following definitions:

- $l \in L$ is a *fixed point* for f iff $f\,(l) = l$;
- f is *reductive* at $l \in L$ iff $f\,(l) \sqsubseteq l$;
- f is *extensive* at $l \in L$ iff $f\,(l) \sqsupseteq l$.

Accordingly, we introduce the following sets:

$$\begin{array}{rcll} \mathit{Fix}\,(f) & = & \{l \mid f\,(l) = l\} & \text{set of fixed points} \\ \mathit{Red}\,(f) & = & \{l \mid f\,(l) \sqsubseteq l\} & \text{set of elements upon which } f \text{ is reductive} \\ \mathit{Ext}\,(f) & = & \{l \mid f\,(l) \sqsupseteq l\} & \text{set of elements upon which } f \text{ is extensive} \end{array}$$

Obviously, $\mathit{Fix}\,(f) = \mathit{Red}\,(f) \cap \mathit{Ext}\,(f)$. As L is a complete lattice, each of these sets has both a least upper bound and a greatest lower bound. In particular, $\bigsqcap \mathit{Fix}\,(f) = \mathit{lfp}\,(f)$ (the least fixed point of f) and $\bigsqcup \mathit{Fix}\,(f) = \mathit{gfp}\,(f)$ (the greatest fixed point of f). By Tarski's theorem, we have

$$\begin{array}{rcccccc} \mathit{lfp}\,(f) & \stackrel{def}{=} & \bigsqcap \mathit{Fix}\,(f) & = & \bigsqcap \mathit{Red}\,(f) & \in & \mathit{Fix}\,(f) \\ \mathit{gfp}\,(f) & \stackrel{def}{=} & \bigsqcup \mathit{Fix}\,(f) & = & \bigsqcup \mathit{Ext}\,(f) & \in & \mathit{Fix}\,(f) \end{array}$$

We can also prove the following relations for the elements of the Ascending/Descending Kleene Sequences:

$$f^n\,(\bot) \;\sqsubseteq\; \bigsqcup\{f^m\,(\bot) \mid m \in \mathbb{N}\} \;\sqsubseteq\; \mathit{lfp}\,(f) \;\sqsubseteq\; \mathit{gfp}\,(f) \;\sqsubseteq\; \bigsqcap\{f^m\,(\top) \mid m \in \mathbb{N}\} \;\sqsubseteq\; f^n\,(\top)$$

If we impose further restrictions on f (e.g., continuity), some of the inequalities will be transformed into equalities. However, in general, all the above inequalities can be strict.

5.2 Widening

Idea: given a (possibly ascending) chain $(l_n)_n$, use a special "widening" operator to construct a fast-converging sequence of elements that stabilizes to an upper approximation of $lfp\,(f)$.

First, we introduce a new notation, which, given an operator and a sequence, allows us to construct a new sequence (with possibly better properties than the initial one):

Definition 4. *Given a chain $(l_n)_n$ in L and an operator $\phi : L \times L \to L$, $(l_n^\phi)_n$ is a chain in L, constructed as follows:*

$$l_n^\phi = \begin{cases} l_n & \text{if } n = 0 \\ l_{n-1}^\phi \ \phi \ l_n & \text{otherwise} \end{cases}$$

Now, we are ready to present the widening technique.

Definition 5. $\nabla : L \times L \to L$ *is a* widening operator *iff:*

1. ∇ *is an* upper bound operator*, i.e.,* $\forall l_1, l_2,\ l_1 \sqsubseteq l_1 \nabla l_2,\ l_2 \sqsubseteq l_1 \nabla l_2$.
2. *For all ascending chains $(l_n)_n$ in L, the ascending chain $(l_n^\nabla)_n$ (also in L) eventually stabilizes*

Given a monotone function $f : L \to L$ on a complete lattice L, and a widening operator ∇ on L, we define the sequence $(f_\nabla^n)_n$ as follows:

$$f_\nabla^n = \begin{cases} \bot & \text{if } n = 0 \\ f_\nabla^{n-1} & \text{if } n > 0 \ \wedge \ f\left(f_\nabla^{n-1}\right) \sqsubseteq f_\nabla^{n-1} \\ f_\nabla^{n-1} \ \nabla \ f\left(f_\nabla^{n-1}\right) & \text{otherwise} \end{cases}$$

We can prove that $(f_\nabla^n)_n$ stabilizes at a value f_∇^m (i.e., $\forall n > m, f_\nabla^n = f_\nabla^m$) such that $f\left(f_\nabla^m\right) \sqsubseteq f_\nabla^m$ (i.e., f is reductive at f_∇^m). Furthermore, using Tarski's theorem, we infer that $f_\nabla^m \sqsupseteq \bigsqcap Red\,(f) = lfp\,(f)$, i.e., the value where $(f_\nabla^n)_n$ stabilizes is a safe approximation of $lfp\,(f)$. See [3, pp. 222-227] for details.

Personal Note: The widening technique can be useful even outside abstract interpretation. For example, in a normal dataflow analysis, we can use it to make sure that the series of abstract values computed for a given program point by the analysis iterations is an ascending chain, even if the transfer functions are not monotonic.

5.3 Narrowing

Idea: In the previous section, we've seen how to use a widening operator to compute an ascending chain $(f_\nabla^n)_n$ which stabilizes to f_∇^m, an upper approximation of $lfp\,(f)$. Moreover, $f\left(f_\nabla^m\right) \sqsubseteq f_\nabla^m$, i.e., f is reductive in the point of stabilization f_∇^m. Therefore, $(f^n\left(f_\nabla^m\right))_n$ is a descending chain of elements in $Red\,(f)$. Any element of this chain is an upper approximation of $lfp\,(f)$; the more elements of the chain we compute, the more precise is our approximation. As the chain might not converge, we can use a "narrowing" operator to construct a chain that descends slower, and stabilizes faster.

Definition 6. $\Delta : L \times L \to L$ *is a* narrowing operator *iff:*

1. $\forall l_1, l_2,\ l_2 \sqsubseteq l_1 \ \rightarrow\ l_2 \sqsubseteq l_1 \Delta l_2 \sqsubseteq l_1$.
2. *For all descending chains* $(l_n)_n$ *in* L*, the sequence* $(l_n^{\Delta})_n$ *eventually stabilizes.*

Given a narrowing operator Δ, we can construct the sequence $([f]_{\Delta}^n)_n$, called the "Truncated Descending Sequence" (TDS) in [1], by

$$[f]_{\Delta}^n = \begin{cases} f_{\nabla}^m & \text{if } n = 0 \\ [f]_{\Delta}^{n-1} \ \Delta\ f\left([f]_{\Delta}^{n-1}\right) & \text{if } n > 0 \end{cases}$$

We can prove that $([f]_{\Delta}^n)_n$ is a descending chain which stabilizes to an upper approximation of $lfp\,(f)$. See [3, pp. 228-231] for details.

6 Advanced Galois Connection Issues

6.1 Galois Insertions

In [1], Patrick and Radhia Cousot use Galois insertions instead of Galois connections. Future papers switched to Galois connections. A Galois insertion is very similar to a Galois connection, but now the abstract domain M does not contain multiple elements that describe the same concrete values, i.e., M does not contain superfluous elements. Given a Galois connection, it is possible to transform it into a Galois insertion (see [3, pp. 242-244]).

Definition 7 (Galois Insertion). $\langle L, \alpha, \gamma, M\rangle$ *is a* Galois insertion *between the lattices* $\langle L, \sqsubseteq_L\rangle$ *and* $\langle M, \sqsubseteq_M\rangle$ *iff:*

1. $\alpha : L \to M$ *and* $\gamma : M \to L$*;*
2. α *and* γ *are monotonic;*
3. $\alpha \circ \gamma = \lambda m.m$*;*
4. $\gamma \circ \alpha \sqsupseteq_L \lambda l.l$.

Lemma 10. *If* $\langle L, \alpha, \gamma, M\rangle$ *is a Galois insertion, then*

- α *is surjective (i.e., each abstract value models some concrete value(s));*
- γ *is injective (i.e., each concrete value is the concretization of at most one abstract value).*

Proof: Consider an arbitrary $m \in M$. By condition 3, $\alpha(\gamma(m)) = m$. Therefore, $\forall m \in M, \exists l = \gamma(m) \in L,$ such that $\alpha(l) = m$, i.e., α is surjective. For the other half, consider arbitrary $m_1, m_2 \in M$ such that $\gamma(m_1) = \gamma(m_2) = l$. We have that $m_1 = \alpha(\gamma(m_1)) = \alpha(\gamma(m_2)) = m_2$. Therefore γ is injective. □

6.2 Systematic Design of Galois Connections

Galois connections can be combined in serial / parallel ways to compose new analyses. Serial composition corresponds to the merging of two successive layers of approximation in the analysis design, while the parallel compositions (a couple of them are possible) are used when we have several analyses of individual components of a structure and we want to combine them in a single analysis. See [3, Section 4.4] for details.

7 Acknowledgements

Many thanks to Viktor Kuncak and Darko Marinov for many interesting (and time consuming!) discussions on abstract interpretation and program analysis in general.

References

[1] Patrick Cousot and Radhia Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. 4th POPL*, 1977.

[2] Patrick Cousot and Radhia Cousot. Systematic design of program analysis frameworks. In *Proc. 6th POPL*, pages 269–282, San Antonio, Texas, 1979. ACM Press, New York, NY.

[3] Flemming Nielson, Hanne Riis Nielson, and Chris Hankin. *Principles of Program Analysis*. Springer-Verlag, 1999.