Abstract Interpretation

Minakami Yuki

April 2022

1 Introduction

In Program Analysis, there is a significant question about how to abstract program source code into formal semantics. As we are going to analyze potential problems of our program without running or compiling it, the abstraction should get the approximation as optimal as possible.

Abstract Interpretation is a branch of theoretical computer science that deals with how to approximate the semantic information of programs. By using the mathematical tools such as lattice theory and category theory, we have developed a theory which satisfies the practical requirements in most cases.

This article aims at introducing the basic abstract interpretation theory to the readers with only a basic knowledge of discrete mathematics.

2 Lattice Theory

At the beginning of abstract interpretation, it is necessary to introduce binary relations.

We assume that our readers have mastered the basic concepts of binary relations, let's go straight to the definition of partial order:

Definition 1 *A binary relation $\leq$ on a set P is a **partial order** satisfying:*

(1) $\forall x \in P, x \leq x$

(2) $\forall x\ y \in P, x \leq y \wedge y \leq x \Rightarrow x = y$

(3) $\forall x\ y\ z \in P, x \leq y \wedge y \leq z \Rightarrow x \leq z$

*The set P is called a **poset**.*

In addition, if $\forall x\ y \in P \Rightarrow x \leq y \vee y \leq x$, then we call P a **chain**. Chain is closely connected with **Zorn's Lemma**, which is equivalent to the axiom of choice. Interested can refer to Zorn's Lemma.

The most natural example of a poset is $\wp(X)$, the collection of all subsets of a set X, ordered by $\subseteq$, that is $(\wp(X), \subseteq)$. Another familiar example is $\mathbb{N}$, the set of all natural numbers, ordered by $\leq$ or division, that is $(\mathbb{N}, \leq)$ or $(\mathbb{N}, |)$.

You can think of lots of examples of this type. Indeed, any nonempty collection Q of subsets of X, ordered by set containment, forms a poset.

More generally, if P is a poset and $Q \subseteq P$, then the restriction of $\leq$ to Q is a partial order, leading to a new poset Q.

In order to obtain the definition of lattice, we have to define some operations on poset. For a poset $(P, \leq)$, $\forall A \subseteq P$, we denote:

(1) $A's$ **maximal** x s.t. $x \in A \wedge \nexists y \in A, x < y$

(2) $A's$ **maximum** x s.t. $x \in A \wedge \forall y \in A, y \leq x$

(3) $A's$ **upper bounds** $U(A) = \{u | \forall x \in A, x \leq u\}$

(4) $A's$ **lower bounds** $L(A) = \{v | \forall x \in A, v \leq x\}$

(5) $A's$ **least upper bounds** $\bigvee A = \sup(A)$

(6) $A's$ **greatest lower bounds** $\bigwedge A = \inf(A)$

where $\sup(A)$ is $U(A)$'s minimum and $\inf(A)$ is $L(A)$'s maximum.

.

After that, we are going to the definition of a lattice!

Definition 2 *For a poset $(L, \leq)$, $\forall x\ y \in L$, the binary set $\{x, y\}$ s.t.*

$$\sup\{x, y\} \in L \wedge \inf\{x, y\} \in L$$

*then we call L a **lattice**.*

*In particular, $\forall A \subseteq L$ s.t. $\bigwedge A \in L \wedge \bigvee A \in L$, we call L a **complete lattice**.*

More generally, to judge if a lattice is complete, we have

Theorem 1 *$(L, \leq)$ is a poset s.t. $\bigvee L \in L \wedge \forall A \subseteq L, \bigwedge A \in L$, then L is a complete lattice. (The dual form is that if $(L, \leq)$ is a poset s.t. $\bigwedge L \in L \wedge \forall A \subseteq L, \bigvee A \in L$, then L is a complete lattice)*

Proof. We only prove the case of $\bigvee L \in L$, the other case's proof is similar.

.

The crux of the proof is realizing $\forall A \subseteq L, \exists B$ s.t. $\bigwedge B = \bigvee A$.

..

Consider $X \subseteq L$, denote $U(X) = Y$, as $\bigvee L \in L$, so X certainly has the upper bounds, which shows that $Y \neq \emptyset$.

...

By assumption, $\bigwedge Y \in L$ s.t. $\forall y \in Y, y \geq \bigwedge Y$. According to the definition of $\bigvee X$, it shows that $\bigwedge Y = \bigvee X$.

....

Notice that X is chosen arbitrarily, so $\forall X \subseteq L$, $\bigvee X \in L$, which has stated that the conclusion is valid.

Qed.

A classical complete lattice is the **closure system**.

Definition 3 *A closure system on X is a series of poset $(\mathcal{C}, \subseteq)$ s.t.*

$$\mathcal{A} \subseteq \mathcal{C} \Rightarrow \bigcap \mathcal{A} \in \mathcal{C}.$$

According to our Theorem 1, if a closure system is a complete lattice, it have to satisfy $\bigvee \mathcal{C} \in \mathcal{C}$. For a concrete set A, the least upper bound is the closure $\overline{\mathrm{A}} \in C$ where $\overline{\mathrm{A}} = A \vee \bigcap A_i$. Therefore, a closure system is a complete lattice. *

Distinctively, the power set $\wp(X)$ is a closure system, which means that it is also a complete lattice.

Now, it's time to discuss the functions between different lattices.

There are two kinds of functions we are going to look at:

1. **Monotonic functions** $f : L \to L'$ receives $x \in L$ and returns $f(x) \in L'$ s.t.
$$x \leq y \Rightarrow f(x) \leq f(y)$$
2. **Continuous functions** $f : L \to L'$ s.t. $\forall S \subset L \Rightarrow f(\bigvee S) = \bigvee f(S)$

We have the following two lemmas concerning the relation between continuity and monotone

Lemma 1 *$f : L \to L'$ is continuous, then f is necessarily monotonic.*

Proof. Consider $S = \{x, y\} \subseteq L$ s.t. $x \leq y$, then $\bigwedge S = x \wedge \bigvee S = y$.

.

f is continuous $\Rightarrow f(y) = \bigvee f(S) \geq f(x) \Rightarrow f$ is monotonic.
Qed.

Lemma 2 *$f : L \to L'$ is monotonic, then f is continuous if and only if*

$$\forall S \subset L \text{ s.t. } f(\bigvee S) \leq \bigvee f(S).$$

Proof. $\forall x \in S \Rightarrow x \leq \bigvee S$

.

f is monotonic $\Rightarrow \forall x \in S$ s.t. $f(x) \leq f(\bigvee S)$, then $f(\bigvee S) \in U(S)$.

..

So we have $f(\bigvee S) \geq \bigvee f(S)$. With our assumption

...

$$\begin{cases} f(\bigvee S) \leq \bigvee f(S) \\ f(\bigvee S) \geq \bigvee f(S) \end{cases} \tag{1}$$

....

By using the properties of $\leq$, we have $f(\bigvee S) = \bigvee f(S) \Leftrightarrow f$ is continuous.
Qed.

.

So far the basic content of lattice theory has come to an end!

3 Fixed Point Theory

Fixed point theory is the basis of recursion. If some of our readers have experience with functional programming, you've probably heard of $\mathbb{Y}$ Combinator, which is an important kind of fixed point.

In a nutshell, the fixed points of a function f are the value of x s.t. $f(x) = x$. Obviously, not all functions have fixed points, for example, $x \in \Re, g(x) = e^x$ don't have a fixed point on $\Re$.

Fixed points of functions often have many interesting properties. Therefore, the existence of fixed points is always an important problem in economics and mathematics, while the construction of fixed points is more concerned in computer science.

When we look at lattices and the two special functions, fixed points are usually related to upper and lower bounds. For a self-mapping $f : L \to L$, we donate the **pre-fixed points** are $\{x|x \in L, x \leq f(x)\}$ and the **post-fixed points** are $\{x|x \in L, x \geq f(x)\}$.

Before going to the fixed point theorems on lattices, some further knowledge of partial order is necessary.

Definition 4 *A poset $(L, \leq)$ s.t. $\forall a, b \in L \Rightarrow \exists c \in L, a \leq c, b \leq c$, then it is called a **directed set**.*

*If $S \subset L$ is a directed set, we call S a **directed subset**.*

With the definition of directed set, we can further define

Definition 5 *A poset $(L, \leq)$ s.t. $\forall S \subset L \wedge S$ is a directed set $\Rightarrow \bigvee S \in L$, then we call $(L, \leq)$ a **directed-complete poset**.*

*If $\perp \subseteq L$, then $(L, \leq)$ is a **complete poset**, where $\perp$ is the singleton set containing the least element of L.*

Now, it's time to introduce the two most important fixed point theorems.

Theorem 2 ***Tarski Theorem**: $f : L \to L$ is a monotonic function on the complete lattice $(L, \leq)$, denoting the fixed point set $\mathcal{F} = \{x|x \in L \wedge f(x) = x\}$, then $\bigwedge \mathcal{F} \in L \wedge \bigvee \mathcal{F} \in L$ and $\mathcal{F}$ is a complete lattice.*

Proof. As for the existence, we only have to prove that $\bigvee \mathcal{F}$ exists because of duality.

*(existence of $\bigvee \mathcal{F}$)

The pre-fixed point set $P = \{x|x \in L \wedge x \leq f(x)\}$, denoting $u = \bigvee P$.

$$\forall x \in P, x \leq f(x) \wedge x \leq u \Rightarrow f(x) \leq f(u) \Rightarrow x \leq f(x) \leq f(u).$$

So $f(u)$ is an upper bound of P, $u = \bigvee P \Rightarrow u \leq f(u) \Rightarrow u \in P$.

..

Notice that f is monotonic, so $f(u) \leq f(f(u)) \Rightarrow f(u) \in P \Leftrightarrow u = f(u)$.

...

$\mathcal{F} \subseteq P \Rightarrow \bigvee \mathcal{F} \leq u$ and u is a fixed point of f, showing $\bigvee \mathcal{F} = u \neq \bot$.
**($\mathcal{F}$ is a complete lattice)

Now let's prove that $\mathcal{F}$ is a complete lattice.

.

$\forall S \subseteq \mathcal{F}$ s.t. $\bigvee S \in \mathcal{F} \Leftrightarrow \mathcal{F}$ is a complete lattice.

..

Donating the upper bound of $\bigvee S$ is $W = \{x | x \in \mathcal{F} \wedge \bigvee S \leq x\}$.

...

We have $\bigwedge W = \bigvee S \Rightarrow \bigvee S \in W$.

....

Notice that $\bigvee W = \bigvee \mathcal{F} \wedge \bigvee W \leq \bigvee S \Rightarrow \bigvee W \in W \Rightarrow W$ is a complete lattice.

$$\forall w \in W \text{ s.t. } \bigvee S \leq w \leq f(w) \Rightarrow f(w) \in W$$

Assuming $f' : W \to W$ s.t. $f'(w) = f(w)$ then f' is a monotonic function.

.....

By the existence of least fixed point, W's least fixed point $w^\wedge$ s.t. $f(w^\wedge) = w^\wedge \Rightarrow w^\wedge \in \mathcal{F}$. We have $w^\wedge \in U(S) \Rightarrow w^\wedge = \bigvee S \in \mathcal{F}$, showing that $(\mathcal{F}, \leq)$ is a complete lattice.
Qed.

.

Tarski Theorem only tells us the existence of fixed points, although it is a great theorem that matters in game theory, it is rarely used in abstract interpretation. The Kleene Fixed-point Theorem gives us a constructive proof showing how to compute the least fixed point.

Theorem 3 ***Kleene Fixed-point Theorem****: $f : L \to L$ is a continuous function on a complete poset $(L, \leq)$, then the least fixed point $\bigwedge \mathcal{F} \in L$, we denote $\bigwedge \mathcal{F} = \ell$ s.t. $\ell = Y_L f$, where $Y_L f = \bigvee\limits_{n \in \mathbb{N}} f^n(\bot)$.*

Proof. Firstly, we divide the proof into three parts.
*(the existence of ℓ)

f is continuous $\Rightarrow$ f is monotonic $\Rightarrow \bot \leq f(\bot)$, now given an induction hypothesis $f^n(\bot) \leq f^{n+1}(\bot)$ *where* $n \in \mathbb{N}$.

.

In the case of $n+1$, $f(f^n(\bot)) \leq f(f^{n+1}(\bot)) \Leftrightarrow f^{n+1}(\bot) \leq f^{n+2}(\bot)$.

..

So $\forall n \in \mathbb{N}, f^n(\bot) \leq f^{n+1}(\bot)$.

...

Then $\{f^n(\bot) \mid n \in \mathbb{N}\}$ is a directed set $\Rightarrow \bigvee\limits_{n \in \mathbb{N}} f^n(\bot) \in L \Rightarrow \ell$ exists.

**(ℓ is a fixed point of f)

Denoting $y_n = f^n(\bot) \Rightarrow \bigvee\limits_{n \in \mathbb{N}} f(y_n) = f(\bigvee\limits_{n \in \mathbb{N}} y_n)$.

$$\ell = Y_L f = \bigvee_{n \in \mathbb{N}} f^n(\bot) = \bigvee_{n \in \mathbb{N}} f^{n+1}(\bot) = f(\bigvee_{n \in \mathbb{N}} f^n(\bot)) = f(Y_L f) = f(\ell).$$

Obviously, ℓ is a fixed point of f.

***(ℓ is the least fixed point of f)

Rename $\mathcal{F} = \{y | y = f(y)\} \Rightarrow \bot \leq y$.

.

Given an induction hypothesis $f^n(\bot) \leq y$ *where* $n \in \mathbb{N}$.

..

In the case of $n+1$, $f(f^n(\bot)) \leq f(y) = y \Rightarrow f^{n+1}(\bot) \leq y$.

...

So $\forall n \in \mathbb{N}$ s.t. $f^n(\bot) \leq y$, $\mathcal{F}$ is a upper bound of $f^n(\bot) \Rightarrow \bigvee_{n \in \mathbb{N}} f^n(\bot) \leq y$.

....

Therefore, ℓ is the least fixed point of f.

Qed.

.

According to all of the above, we are able to abstract types in any programming languages. A type Γ is a set of elements of V. Not all subsets of V are legal types: they must obey some technical properties. The subsets of V obeying such properties are called ideals(I).

All the types found in programming languages are ideals in this sense, so we don't have to worry too much about subsets of V which are not ideals.

Hence, a type is an ideal, which is a set of values. Moreover, the set of all types over V, when ordered by set inclusion ($\subseteq$), forms a lattice L. The top of this lattice is the type $\top$ (the set of all values, i.e. V itself).

The bottom of the lattice is $\bot$, essentially, the empty set $\emptyset$ (actually, it is the singleton set containing the least element of V). To be more specific, the power set of the set V forms a complete lattice $\langle \wp(V), \subseteq \rangle$. Types are ideals I of $\langle \wp(V), \subseteq \rangle$ satisfying:

(1) $\forall x \in I, y \leq x \Rightarrow y \in I$

(2) $\forall x\ y \in I, \exists z \in I$ s.t. $x \leq z \wedge y \leq z$

It is trivial to prove that Disjoint union $\oplus$, Cartesian product $\otimes$ and Function type $\rightarrow$ can form closure systems on I. Obviously, $\oplus : I \rightarrow I \rightarrow I$, $\otimes : I \rightarrow I \rightarrow I$ and $\rightarrow$: $I \rightarrow I$ are continuous functions above I.

Consider the set of types T, which is recursively generated by these three operations. By Kleene fixed point theorem, T has a least fixed point. Moreover, T is also a ideal of $\langle \wp(V), \subseteq \rangle$. Finally, we are able to confirm all of types are the ideals of $\langle \wp(V), \subseteq \rangle$.

More generally, the least fixed point logic can perfectly work with finite model theory and have a strong power of expression. It is not only used in abstract interpretation, but also many other domains of theoretical computer science. Interested can refer to Model Fixpoint Logics.

After introducing the theoretical basis of abstract interpretation, we are going to get into the real abstract interpretation!

4 Galois Connection

Most of the interesting properties of programs are undecidable since they can be reduced to the Halting Problem. To analyze the behavior of programs, we introduce approximations. The approximations must be **sound**: if the system gives a definite answer, then this answer is true.

The state of programs consists of 1) a program counter, which indicates the next command to execute, 2) an environment, which contains a partial map from variables to values.

We can formalize the process as a state machine $\langle \Sigma, \rightsquigarrow \rangle$, where Σ is a **state set** and $\rightsquigarrow \subseteq \Sigma \times \Sigma$ is a **transition**.

A sequence of states is a **trace**. A program may have several traces, depending on the user input; so the semantics of a program is a set of traces.

$$v_0 \rightsquigarrow v_1 \rightsquigarrow \cdots \rightsquigarrow v_i \rightsquigarrow \cdots, \text{where } v_0 \text{ is the initial state.}$$

We want to prove properties like "all traces of the program satisfy a given condition". To do this, we over-approximate the set of traces, and obtain a superset of the set of concrete traces. An analysis is more precise when it yields a small superset of the set of concrete traces. All correct approximations are the ones are greater than the smallest correct one, namely, the set of concrete traces.

Collecting Semantics computes the set of possible traces, formally:

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

The set of all traces, of all states, are ordered by inclusion. A program analysis is an "abstract execution" of the program, in a property space L. The property space L should be a complete lattice $\langle L, \leq \rangle$. Instead of values, the analysis works with properties (equivalent terms are "abstract values") $l_i \in L$.

$$\begin{array}{ccc} l_1 & \xrightarrow{f_L} & l_2 \\ \mathcal{R}\uparrow\downarrow & & \mathcal{R}\uparrow\downarrow \\ v_1 & \xrightarrow{\cdot\rightsquigarrow\cdot} & v_2 \end{array} \tag{2}$$

More formally, we have an **correctness relation** $\mathcal{R}$ such that $v_i \ R \ l_i$.

Definition 6 *A relation $\mathcal{R} \in \Sigma \times L$ s.t.*

(1) $\forall v, l_1, l_2, (v \ \mathcal{R} \ l_1) \wedge (l_1 \leq l_2) \rightarrow (v \ \mathcal{R} \ l_2)$

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

is called an correctness relation.

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, $l_1 \leq l_2$, then l_2 is also an approximation of v.

Therefore, in the property lattice L, if $l_1 \leq l_2$, l_1 is more **precise** than l_2 (l_2 approximates all the values approximated by l_1 and possibly some other ones). More geneally, in the lattice L, the **smaller** means more precise and the **bigger** means **safer**. We donate $\top$ representing all possible information of our program(i.e. the most imprecise value), while $\bot$ means no information.

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, $L_p = \bigwedge L'$.

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.

From the two conditions, we can draw a simple corollary.

Corollary 1 *$\mathcal{R} \in \Sigma \times L$ is an correctness relation s.t.*

(1) $\forall v, (v\ \mathcal{R}\ \top)$

(2) $\forall v, (v\ \mathcal{R}\ l_1) \wedge (v\ \mathcal{R}\ l_2) \rightarrow (v\ \mathcal{R}\ (l_1 \wedge l_2))$

(3) $\forall v, (v\ \mathcal{R}\ l_1) \vee (v\ \mathcal{R}\ l_2) \rightarrow (v\ \mathcal{R}\ (l_1 \vee l_2))$

I'll leave the proof as an exercise for our readers.

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 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, from Tarski Theorem, we require that:

(1) The analysis transfer function f_L is monotone

(2) $\langle L, \leq \rangle$ is a complete lattice

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.

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

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, with respect to the correctness relation $\mathcal{R}$.

For example, we want to analyze the possible values of an integer variable at runtime, the state set is $\Sigma = I \times \mathbb{Z}$ where I represents the set of instructions for the program. $v_i \in \mathbb{Z}$ is the value at $i \in I$; the transition $\rightsquigarrow_i$ is defined by the specific semantics of the program. Then corresponding to the program analysis given by collecting semantics $\langle L, f_L \rangle$ can be defined as $L = \wp(\mathbb{Z})^I$ where $\wp(\mathbb{Z})$ is the power set of $\mathbb{Z}$. A subset of L is regraded as an indexed family for I.

$$\forall S \in \wp(\mathbb{Z})^I \Leftrightarrow S = \{S_i : i \in I \wedge S_i \in \wp(\mathbb{Z})\}$$

The transfer function f_L can be defined as

$$f_L(i \mapsto S_i) = i \mapsto \bigcup_{v \in S_i} \{v' : v \rightsquigarrow_i v'\}$$

By definition, f_L is obviously monotonic. Assume the initial state $v_{i_0} = 0 \wedge S_{i_0} = \{0\}$, the correctness relation $\mathcal{R}$ s.t. $(v_i \ \mathcal{R} \ S_i) \Leftrightarrow v_i \in S_i$.

We are going to prove the correctness of our $\langle L, f_L \rangle$.

1. $v_{i_0} \ \mathcal{R} \ S_i \Leftrightarrow 0 \in \{0\}$ is tenable
2. $\forall v_{i_1}, v_{i_2}, \forall S_{i_1}, S_{i_2}, (v_{i_1} \ \mathcal{R} \ S_{i_1}) \wedge (v_{i_1} \rightsquigarrow_i v_{i_2}) \wedge (S_{i_2} = f_L(S_{i_1}))$
   $\rightarrow v_{i_2} \in S_{i_2} \Leftrightarrow (v_{i_2} \ \mathcal{R} \ S_{i_2})$ is tenable

We have confirmed the correctness of our approximation!

To get all the properties of the program, we have to calculate the least fixed point of the transfer function f_L on the property space L. As Tarski theorem only tells us the existence of least fixed point, we have to think of the Kleene fixed point theorem.

According to Kleene fixed point theorem, if f_L is continuous, the least fixed point $\text{lfp}(f_L) = Y f_L = \bigvee_{n \in \mathbb{N}} f^n(\bot)$. Our task is to determine in which case does a monotonic function $f : L \rightarrow L$ be a continuous one on a complete lattice.

For instance, a monotonic self-mapping $f : \wp(S) \rightarrow \wp(S)$ is continuous on the complete lattice $\langle \wp(S), \subseteq \rangle$.

Theorem 4 *More generally, if a complete lattice L has the* ***ascending chain property****(there is no infinite ascending chain in L), a monotonic function f_L is also continuous.*

Proof.

From the Lemma 2, our goals is to prove $\forall S \subset L$ s.t. $f(\bigvee S) \leq \bigvee f(S)$.

.

Consider a binary set $l = \{l_1, l_2\}$ s.t. $l_1 \leq l_2$, $f(l_1 \vee l_2) = f(l_2) = f(l_1) \vee f(l_2)$. By this conclusion, we can unfold a finite subset of L.

..

If S is finite, we can write $S = \{s_1, \cdots, s_n\}$ for $n \geq 1$ and

$$f(\bigvee S) = f(s_1 \vee \cdots \vee s_n) = f(s_1) \vee \cdots \vee f(s_n) \leq \bigvee f(S).$$

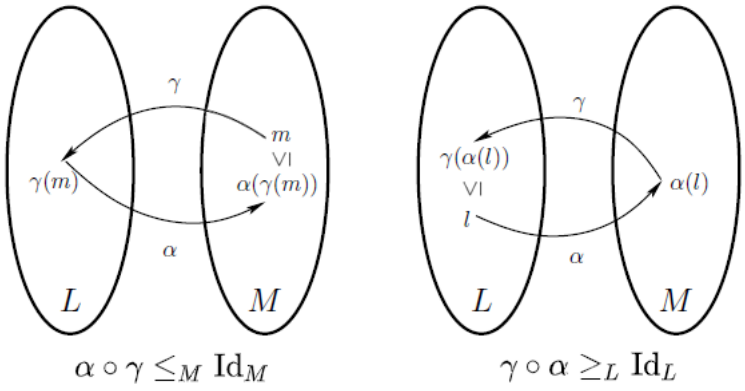

Figure 1: Galois Connection

If S is infinite, by the ascending chain property, $\bigvee S$ must be a upper bound of an ascending chain $l = \{l_1, \cdots, l_n\} \wedge l \subseteq L$.

...

Denoting $\bigvee S = l_n = s_n \vee \cdots \vee s_0$ for some $s_i \in S$ and $0 \leq i \leq n$. We then have

$$f(\bigvee S) = f(l_n) = f(s_n \vee \cdots \vee s_0) = f(s_n) \vee \cdots \vee f(s_0) \leq \bigvee f(S).$$

This completes the proof.
Qed.

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** $\langle L, \alpha, M, \gamma \rangle$ between L and M.

Definition 7 ***Galois Connection*** $\langle L, \alpha, M, \gamma \rangle$ *is a connection between two lattices* $\langle L, \leq_L \rangle$ *and* $\langle M, \leq_M \rangle$ *s.t.*

(1) $\alpha : L \to M$ *and* $\gamma : M \to L$ *are both monotonic functions*

(2) $\alpha \circ \gamma \leq_M \mathrm{Id}_M$

(3) $\gamma \circ \alpha \geq_L \mathrm{Id}_L$

Figure 1 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**. Conversely, γ is called the **concretization**.

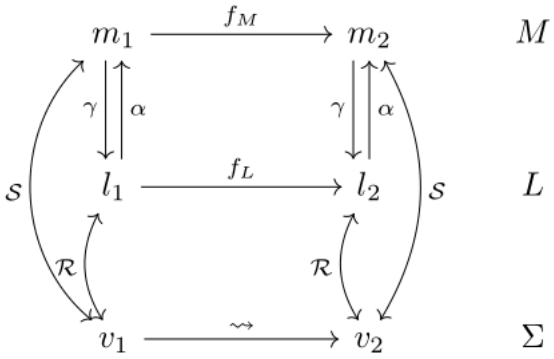

Figure 2: Correctness Relation of M

The second part of the Galois Connection definition $\alpha \circ \gamma \leq_M \mathrm{Id}_M$ shows that concretization does not lose analysis precision, and the third which is abstraction can lose analysis precision but does not affect the correctness of the analysis, resulting in more secure results.

$$\begin{array}{ccc} m_1 & \xrightarrow{f_M} & m_2 \geq_M \\ \downarrow{\scriptstyle \gamma} & & \uparrow{\scriptstyle \alpha} \\ l_1 & \xrightarrow{f_L} & l_2 \end{array} \tag{3}$$

Given a program analysis $\langle L, f_L \rangle$ and a lattice M with the Galois Connection, we can define the correctness relation $\mathcal{S} \in \Sigma \times M$ of M according to the correctness relation $\mathcal{R} \in \Sigma \times L$ of L.

$$v \; \mathcal{S} \; m \Leftrightarrow v \; \mathcal{R} \; \gamma(m)$$

At the same time, we can define the transfer function f_M s.t.

$$f_M \geq_M \alpha \circ f_L \circ \gamma$$

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

Let's start with a different definition of Galois Connection.

Definition 8 *(**Adjunction**) $\langle L, \alpha, M, \gamma \rangle$ is the Galois Connection between two lattices $\langle L, \leq_L \rangle$ and $\langle M, \leq_M \rangle$ if and only if $\alpha : L \to M$ and $\gamma : M \to L$ s.t.*

$$\alpha(l) \leq_M m \Leftrightarrow l \leq_L \gamma(m)$$

In a categorical view, a poset $\langle X, \leq_X \rangle$ is an order category, the objects are the elements of X and the morphisms are $x \to x' \Leftrightarrow x \leq_X x'$. Moreover, a

monotonic function $f : X \to Y$ can be regarded as a functor from category $\mathbb{X}$ to category $\mathbb{Y}$.

So we can rewrite the Definition 8 as

Definition 9 *$\langle L, \alpha, M, \gamma \rangle$ is the Galois Connection between two lattices $\langle L, \leq_L \rangle$ and $\langle M, \leq_M \rangle$ if and only if $\alpha : L \to M$ is the left adjoint of $\gamma : M \to L$.*

We are going to prove the two different definitions are equivalence.

Lemma 3 *$\langle L, \alpha, M, \gamma \rangle$ is an adjunction s.t. $\alpha \circ \gamma \leq_M \mathrm{Id}_M \wedge \gamma \circ \alpha \geq_L \mathrm{Id}_L$.*

Proof.

Suppose $m = \alpha(l)$, by the definition of adjunction, we have

$$\alpha(l) \leq_M \alpha(l) \Leftrightarrow l \leq_L \gamma(\alpha(l))$$

So $\gamma \circ \alpha \geq_L \mathrm{Id}_L$. The rest of the proof is similar.
Qed.

Lemma 4 *$\langle L, \alpha, M, \gamma \rangle$ is an adjunction, $\alpha : L \to M$ and $\gamma : M \to L$ s.t. both of them are monotonic.*

Proof.

Suppose $l_1 \leq_L l_2 \wedge m = \alpha(l_2)$, by the definition of adjunction

$$\alpha(l_1) \leq_M \alpha(l_2) \Leftrightarrow l_1 \leq_L \gamma(\alpha(l_2))$$

According to Lemma 3, we have $l_1 \leq_L l_2 \leq_L \gamma(\alpha(l_2)) \Leftrightarrow \alpha(l_1) \leq_M \alpha(l_2)$, it shows $\alpha : L \to M$ is monotonic. The monotonicity of γ's proof is similar.
Qed.

Lemma 5 *A Galois Connection $\langle L, \alpha, M, \gamma \rangle$ is an adjunction.*

Proof.

Suppose $\alpha(l) \leq_M m$, γ is monotonic $\Rightarrow \gamma \circ \alpha(l) \leq_L \gamma(m)$.

.

By the definition of Galois Connection, we have $\gamma \circ \alpha(l) \geq_L l \Rightarrow l \leq_L \gamma(m)$.

..

Therefore, $\alpha(l) \leq_M m \Rightarrow l \leq_L \gamma(m)$. The proof in the opposite direction is similar.
Qed.

Theorem 5 *$\langle L, \alpha, M, \gamma \rangle$ is a Galois Connection if and only if it is an adjunction.*

Proof.

Lemma 3 and Lemma 4 show that $\langle L, \alpha, M, \gamma \rangle$ is an adjunction $\Rightarrow$ it is a Galois Connection.

.

Lemma 5 shows that $\langle L, \alpha, M, \gamma\rangle$ is a Galois Connection $\Rightarrow$ it is an adjunction.
..
In a conclusion, we have Adjunction $\Leftrightarrow$ Galois Connection.
Qed.

Theorem 5 proves that the two different definitions are equivalent. The following results show how to construct a Galois Connection. Firstly, let's deal with an easy corollary given by the definition of Galois Connection.

Corollary 2 *$\langle L, \alpha, M, \gamma\rangle$ is a Galois Connection s.t.*

(1) $\alpha \circ \gamma \circ \alpha = \alpha$

(2) $\gamma \circ \alpha \circ \gamma = \gamma$

Proof.

α is monotonic s.t. $\gamma \circ \alpha \geq_L \mathrm{Id}_L \wedge \alpha \circ \gamma \leq_M \mathrm{Id}_M$.

$$\forall l \in L, (\alpha \circ \gamma \circ \alpha)(l) = (\alpha \circ \gamma)(\alpha(l)) \leq_M \alpha(l)$$
$$(\alpha \circ \gamma \circ \alpha)(l) = \alpha \circ ((\gamma \circ \alpha)(l)) \geq_M \alpha(l)$$
$$\Leftrightarrow \forall l \in L, (\alpha \circ \gamma \circ \alpha)(l) = \alpha(l) \Leftrightarrow \alpha \circ \gamma \circ \alpha = \alpha$$

The proof of the second equation is similar.
Qed.

Lemma 6 *$\langle L, \alpha, M, \gamma_1\rangle \wedge \langle L, \alpha, M, \gamma_2\rangle$ are both Galois Connection $\Rightarrow \gamma_1 = \gamma_2$. (Analogously, $\langle L, \alpha_1, M, \gamma\rangle \wedge \langle L, \alpha_2, M, \gamma\rangle$ are both Galois Connection $\Rightarrow \alpha_1 = \alpha_2$).*

Proof.

By the definition of Galois Connection and our corollary, we have

$$\gamma_1(m) \leq_L (\gamma_2 \circ \alpha) \circ \gamma_1(m) = \gamma_2 \circ (\alpha \circ \gamma_1)(m) \leq_L \gamma_2(m)$$

Swap γ_1 and $\gamma_2 \Rightarrow \gamma_2(m) \leq \gamma_1(m) \Rightarrow \gamma_1 = \gamma_2$.
.
The proof of $\alpha_1 = \alpha_2$ is similar.
Qed.
.

Lemma 6 reveals that α uniquely determines γ, v.v. So we can explicitly construct γ from α and vice versa.

Lemma 7 *$\langle L, \alpha, M, \gamma\rangle$ is a Galois Connection s.t.*

(1) *α uniquely determines* $\gamma \wedge \gamma(m) = \bigvee\{l : \alpha(l) \leq_M m\}$

(2) *γ uniquely determines* $\alpha \wedge \alpha(l) = \bigwedge\{m : l \leq_L \gamma(m)\}$

Proof.

Notice that $\gamma(m) = \bigvee\{l : l \leq_L \gamma(m)\}$, by the definition of adjunction, we have

$$\gamma(m) = \bigvee\{l : l \leq_L \gamma(m)\} = \bigvee\{l : \alpha(l) \leq_M m\}$$

This uniquely identifies γ. The proof of construction for α is similar.
Qed.

.

Now, an important question to ask is, what kind of qualitative constraints do we have on α and γ? The following lemma provides a basic characterization of transfer functions:

Lemma 8 *$\langle L, \alpha, M, \gamma\rangle$ is a Galois Connection texts.t.*

*(1) α is **completely additive**, $\forall L' \subseteq L, \alpha(\bigvee L') = \bigvee \alpha(L')$*

*(2) γ is **completely multiplicative**, $\forall L' \subseteq L, \gamma(\bigwedge L') = \bigwedge \gamma(L')$*

We have $\alpha(\bot_L) = \bot_M \wedge \gamma(\top_M) = \top_L$ because of $\bigvee \emptyset = \bot \wedge \bigwedge \emptyset = \top$.

Proof.

$$\alpha(\bigvee L') \Leftrightarrow \bigvee L' \leq_L \gamma(m) \Leftrightarrow \forall l' \in L', l' \leq_L \gamma(m)$$

$$\Leftrightarrow \forall l' \in L', \alpha(l') \leq_L \gamma(m) \Leftrightarrow \bigvee \alpha(L') \leq_M m$$

Let's take $m = \bigvee \alpha(L')$ and $m = \alpha(\bigvee L') \Rightarrow \alpha(\bigvee L') = \bigvee \alpha(L')$.

.

The rest of the proof is similar.
Qed.

Corollary 3 *There are two important propositions:*

(1) If $\alpha : L \to M$ is monotonic and completely additive, we can launch that $\exists \gamma : M \to L$ s.t. $\langle L, \alpha, M, \gamma\rangle$ is a Galois Connection.

(2) If $\gamma : M \to L$ is monotonic and completely multiplicative, we can launch that $\exists \alpha : L \to M$ s.t. $\langle L, \alpha, M, \gamma\rangle$ is a Galois Connection.

The proof of this corollary is left to the readers.

.

Corollary 3 provides a way to construct abstract analysis. We only need to provide the abstract field M and a completely multiplicative concretization γ, then the abstraction α exists and $\langle L, \alpha, M, \gamma\rangle$ constitutes a Galois Connection.

Review the previous definition of correctness relation, a Galois Connection $\langle L, \alpha, M, \gamma\rangle$ induces an correctness relation $\mathcal{S} \in \Sigma \times M$ s.t. $v \; \mathcal{S} \; m \Leftrightarrow v \; \mathcal{R} \; \gamma(m)$ and the transfer function s.t. $f_M \geq_M \alpha \circ f_L \circ \gamma$. Now, we can give a proof that $\mathcal{S}$ is an correctness relation.

Theorem 6 *$\mathcal{S} \in \Sigma \times M$ is an correctness relation.*

Proof.
*

$$\forall v \in \Sigma, \forall m_1, m_2 \in M \text{ s.t. } v \ \mathcal{S} \ m_1 \wedge m_1 \leq_M m_2$$
$$\Rightarrow v \ \mathcal{R} \ \gamma(m_1) \wedge \gamma(m_1) \leq_L \gamma(m_2) \Rightarrow v \ \mathcal{R} \ \gamma(m_2) \Leftrightarrow v \ \mathcal{S} \ m_2$$

**

$$\forall M' \subseteq M, \forall m' \in M' \text{ s.t. } v \ \mathcal{S} \ m' \Leftrightarrow v \ \mathcal{R} \ \gamma(m')$$
$$\Rightarrow v \ \mathcal{R} \ (\bigwedge \gamma(m')) \stackrel{\gamma}{\Longrightarrow} v \ \mathcal{R} \ \gamma(\bigwedge m') \Rightarrow v \ \mathcal{S} \ \bigwedge m'$$

Qed.

.

In addition to this, we need to ensure that correctness is maintained in state transitions.

Lemma 9 $\gamma \circ f_M \geq_L f_L \circ \gamma$

Proof.

By the definition of f_M

$$\forall m \in M, f_M(m) \geq_L (\alpha \circ f_L \circ \gamma)(m)$$

As γ is monotonic s.t. $\gamma \circ \alpha \geq_L \lambda l.l$, we have that

$$\gamma(f_M(m)) \geq_L (\gamma \circ \alpha)(f_L(\gamma(m))) \geq_L f_L(\gamma(m))$$
$$\Leftrightarrow \gamma \circ f_M \geq_L f_L \circ \gamma$$

Qed.

Theorem 7 *$\mathcal{S}$ is preserved under the computation s.t.*
$\forall v_1, v_2, \forall m_1, m_2, (v_1 \ \mathcal{S} \ m_1) \wedge (v_1 \rightsquigarrow v_2) \wedge (m_2 = f_M(m_1)) \rightarrow (v_2 \ \mathcal{S} \ m_2)$

.

The proof is left to our readers.

Having established the theoretical basis of Galois connection, we return to the example of integer analysis, $L = \wp(\mathbb{Z})^I$, supposing $J(\mathbb{Z}) = \{[a, b] : a, b \in \mathbb{Z}\}$.

We can use an interval $[a, b] \in J(\mathbb{Z})$ to approximate a subset of integers $\mathbb{Z}$, denoting $S \in \wp(\mathbb{Z})$, where $a = \min S, b = \max S$.

Define partial order on $J(\mathbb{Z})$ as inclusion s.t.

1. $[a_1, b_1] \vee [a_2, b_2] = [\min\{a_1, a_2\}, \max\{b_1, b_2\}]$
2. $[a_1, b_1] \wedge [a_2, b_2] = [\max\{a_1, a_2\}, \min\{b_1, b_2\}]$

Naturally there are abstract functions $\alpha(S) = [\min S, \max S]$ and representational functions $\gamma : J(\mathbb{Z}) \to \wp(\mathbb{Z})$ for canonical embedding. Select transfer function $f = \alpha \circ f_L \circ \gamma$, f induces $J(\mathbb{Z})$ on the algebraic structure, such as addition

$$[a_1, b_1] + [a_2, b_2] = [a_1 + a_2, b_1 + b_2]$$

We say, Galois connection guarantees a very good approximation.

After using abstract analysis, we often get a smaller analysis. For example, in the example above, the interval is used to approximate the set of integers, and the data that needs to be saved is reduced to two ends of the interval. But the set J is still an infinite set and does not satisfy the ascending chain property, the least fixed point is still incalculable.

One solution is to directly limit the range of possible values in a program, for example, by limiting $\mathbb{Z}$ to the familiar unit_{32} range, namely $\mathbb{Z}_{2^{32}} = \mathbb{Z}/2^{32}\mathbb{Z}$. Thus both $\wp(\mathbb{Z}_{2^{32}})$ and $J(\mathbb{Z}_{2^{32}})$ are finite sets, which naturally satisfy the ascending chain property. The calculation of the fixed point will be stopped for a finite time.

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 $\langle L, \alpha, M, \gamma \rangle$ is a Galois connection ($\Leftrightarrow$ Adjunction).

Galois connections can be combined in serial or 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. There will be a lot of interesting works on this area!

5 Conclusion

We try to construct a complete abstract interpretation theory with mathematical tools in this article. Perhaps these theories are so abstract that they are incomprehensible to many readers.

However, it is significant for us to recognize that over-generalization makes no sense for either mathematics or theoretical computer science. No one likes anything too abstract to be practical.

In my opinion, abstract interpretation shows how necessary abstraction can be made to analyze a practical problem in theoretical computer science. We can apply this idea to many areas beyond abstract interpretation.

All in all, this is where the impact of the whole theoretical computer science comes in.