

# Abstract Interpretation

## Chapter 12: Collecting Semantics & Galois Connections

Oral Exam Script

Target: 15 Minutes

### 1. Introduction (Approx. 2 Minutes)

Good morning. Today I will be presenting Chapter 12 on Abstract Interpretation.

In the previous parts of this course, we have built various static analyses intuitively. However, a critical question remains: *How do we verify that these analyses are actually correct?*

We know from Rice's Theorem that calculating non-trivial properties of runtime behavior perfectly is undecidable. We cannot compute the exact set of all possible states for infinite loops or recursive structures.

To solve this, we use Abstract Interpretation. This is a formal mathematical framework that bridges the gap between the infinite "Concrete" execution of a program and a finite "Abstract" approximation.

My goal is to show how we define the "Ground Truth" (Collecting Semantics) and strictly link it to our analysis using Galois Connections to guarantee Soundness.

#### Whiteboard Action:

- **The Problem:** Undecidability (Rice's Theorem).
- **The Solution:** Finite Approximation.
- **Concrete Domain ( $C$ ):**  $\mathcal{P}(\mathbb{Z})$  (e.g.,  $\{0, 2, 4, \dots\}$ )
- **Abstract Domain ( $A$ ):** Signs, Intervals, Parity, etc.

### Part 1: The Framework & Galois Connections (4 Minutes)

To perform this analysis formally, we need two domains and a strict way to translate between them. First, we have the **Collecting Semantics**. This represents the ground truth—the set of all concrete states reachable at a program point. Since this set is often infinite, we map it to a finite **Abstract Domain** ( $L$ ).

To link them, we define a **Galois Connection**, which consists of two monotone functions:

1. **Abstraction ( $\alpha$ ):** Maps a concrete set to its "best" abstract representation.
2. **Concretization ( $\gamma$ ):** Maps an abstract value back to the set of real values it represents.

### Whiteboard Action:

#### The Galois Connection Pair $(\alpha, \gamma)$ :

1. **Extension (Safety):**

$$x \subseteq \gamma(\alpha(x))$$

2. **Reduction (Precision):**

$$\alpha(\gamma(y)) \sqsubseteq y$$

#### The Adjunction (Shortcut):

$$\alpha(x) \sqsubseteq y \iff x \subseteq \gamma(y)$$

For this pair to be a valid Galois Connection, it must satisfy two specific properties.

The first is **Extension**. This is our **Safety** guarantee. It says: If you abstract a value and then concretize it back, the result must be *larger or equal* to what you started with. We are allowed to add "noise" (imprecision), but we are strictly forbidden from losing original data.

The second is **Reduction**. This is our **Precision** guarantee. It says: If you concretize an abstract value and re-abstract it, you must get a result *smaller or equal* to the original. This forces  $\alpha$  to pick the most specific symbol available. It prevents the abstraction from being "lazy" and just returning  $\top$  (Unknown) for everything.

Finally, we often combine these into the **Adjunction Property**. This is a powerful shortcut: It tells us that checking if a concrete set  $x$  is safely represented by abstract value  $y$  is mathematically identical to checking if  $x$  is contained in the concretization of  $y$ .

## Part 2: Precision - Optimality & Completeness (Approx. 5 Minutes)

Now that we have Soundness (safety), we must ask: Is our analysis \*good\*? We define "good" using two concepts: **Optimality** and **Completeness**.

First, **Optimality**. This concept applies to our transfer functions (the rules we write for operations like  $+$  or  $-$ ). Optimality basically asks: *Is this rule as precise as it mathematically can be, given our abstract domain?*

The mathematical definition provides a recipe for deriving the perfect rule:

### Whiteboard Action:

#### Optimality (The "Best" Local Rule):

$$af_{opt} = \alpha \circ cf \circ \gamma$$

#### Recipe:

1. **Concretize** ( $\gamma$ ): Go to the real world.
2. **Compute** ( $cf$ ): Do the math concretely.
3. **Abstract** ( $\alpha$ ): Come back to the abstract world.

If we follow this loop, we get the optimal result. For example, consider the expression  $x - x$  where the value of  $x$  is unknown ( $\top$ ).

In a naive analysis, we might say: "I don't know  $x$ , so I don't know  $x - x$ ." The result would be  $\top$ . But let's apply the optimality recipe:

1. We concretize  $\top$  to "all integers".
2. We compute  $n - n$  for every integer. The result is always 0.
3. We abstract  $\{0\}$  to the abstract value **Zero**.

So, the *optimal* abstract result for  $x - x$  is **Zero**, not  $\top$ . Optimality ensures we don't lose precision just because our transfer function is lazy.

Next is **Completeness**. This is a much stricter, global requirement. Completeness implies there is **zero** loss of precision during the entire analysis compared to the concrete execution.

### Whiteboard Action:

#### Completeness (Perfect Simulation):

$$\alpha \circ cf = af \circ \alpha$$

#### Commutative Diagram:

$$\begin{array}{ccc} \textit{Concrete} & \xrightarrow{cf} & \textit{Concrete} \\ \downarrow \alpha & & \downarrow \alpha \\ \textit{Abstract} & \xrightarrow{af} & \textit{Abstract} \end{array}$$

This formula represents a "Commutative Diagram." It says: It shouldn't matter if I calculate concretely and then abstract ( $\alpha \circ cf$ ), or abstract first and then calculate abstractly ( $af \circ \alpha$ ). The result should be identical.

In Abstract Interpretation, completeness is extremely rare. We almost always lose information. Consider adding (+1) and (-1).

### Whiteboard Action:

#### Example of Incompleteness:

- **Concrete Path:**  $1 + (-1) = 0$ . Abstraction  $\rightarrow$  **Zero**.
- **Abstract Path:**  $(+) + (-) = \top$  (could be anything).

Since **Zero**  $\sqsubset \top$ , the Abstract path lost precision.

Because the abstract calculation gave us  $\top$  while the concrete calculation (abstracted) gave us **Zero**, our analysis is **Incomplete**. We lost the specific relationship that the magnitudes were equal.

## Part 3: Precision & Completeness (4 Minutes)

Soundness means we aren't wrong, but it doesn't mean we are useful. If we always returned "Top" (Unknown), we would be sound, but useless.

We strive for two higher goals: **Optimality** and **Completeness**.

**Optimality** defines the best possible abstract function. Mathematically, it is defined by inducing the function from the concrete world:

### Whiteboard Action:

#### Write Optimality:

$$af_{opt} = \alpha \circ cf \circ \gamma$$

**Completeness** is an even stricter requirement. Completeness implies there is NO loss of precision during the analysis.

### Whiteboard Action:

#### Write Completeness:

$$\alpha \circ cf = af \circ \alpha$$

However, in Abstract Interpretation, completeness is rare. Consider the expression  $(+1) + (-1)$ .

In the concrete world, the result is exactly 0. In the abstract world of signs, adding a Positive to a Negative results in  $\top$  (it could be anything). Because  $\alpha(0) \sqsubset \top$ , the analysis is Incomplete.

## Conclusion (<1 Minute)

To summarize:

- We use **Collecting Semantics** to define the concrete ground truth.

- We use **Galois Connections** ( $\alpha$  and  $\gamma$ ) to formally link the concrete and abstract worlds.
- We rely on the **Soundness Square** to prove that our abstract transfer functions never miss a possible program state.

This framework provides the mathematical proof required to trust static analysis tools.

Thank you. I am happy to take any questions.