Abstract Interpretation
Semantics and applications to verification

Xavier Rival

École Normale Supérieure

May 5th, 2017
Program of this lecture

Studied so far:

- **semantics**: behaviors of programs
- **properties**: safety, liveness, security...
- **approaches to verification**: typing, use of proof assistants, model checking

Today’s lecture: introduction to abstract interpretation

a **general framework for comparing semantics** introduced by Patrick Cousot and Radhia Cousot (1977)

- **abstraction**: use of a lattice of predicates
- **computing abstract over-approximations**, while preserving soundness
- **computing abstract over-approximations for loops**
Outline

1. Abstraction
   - Notion of abstraction
   - Abstraction and concretization functions
   - Galois connections

2. Abstract interpretation

3. Application of abstract interpretation

4. Conclusion
Abstraction example 1: signs

Abstraction: defined by a family of properties to use in proofs

Example:
- objects under study: sets of mathematical integers
- abstract elements: signs

Lattice of signs
Abstraction example 1: signs

Definition: abstraction relation

- **concrete elements**: elements of the original lattice \((c \in \mathcal{P}(\mathbb{Z}))\)
- **abstract elements**: predicate \((a: \cdot \in \{\pm, 0, \ldots\})\)
- **abstraction relation**: \(c \vdash_S a\) when \(a\) describes \(c\)

Examples:

- \(\{1, 2, 3, 5, 7, 11, 13, 17, 19, 23, \ldots\} \vdash_S \pm\)
- \(\{1, 2, 3, 5, 7, 11, 13, 17, 19, 23, \ldots\} \vdash_S \top\)

We use abstract elements **to reason about operations**:

- if \(c_0 \vdash_S \pm\) and \(c_1 \vdash_S \pm\), then \(\{x_0 + x_1 \mid x_i \in c_i\} \vdash_S \pm\)
- if \(c_0 \vdash_S \pm\) and \(c_1 \vdash_S \pm\), then \(\{x_0 \cdot x_1 \mid x_i \in c_i\} \vdash_S \pm\)
- if \(c_0 \vdash_S \pm\) and \(c_1 \vdash_S 0\), then \(\{x_0 \cdot x_1 \mid x_i \in c_i\} \vdash_S 0\)
- if \(c_0 \vdash_S \pm\) and \(c_1 \vdash_S \bot\), then \(\{x_0 \cdot x_1 \mid x_i \in c_i\} \vdash_S \bot\)

Xavier Rival

Abstract Interpretation: Introduction

May 5th, 2017 5 / 54
Abstraction example 1: signs

We can also consider the **union operation**:
- if \( c_0 \vdash_{S} \perp \) and \( c_1 \vdash_{S} \perp \), then \( c_0 \cup c_1 \vdash_{S} \perp \)
- if \( c_0 \vdash_{S} \perp \) and \( c_1 \vdash_{S} \perp \), then \( c_0 \cup c_1 \vdash_{S} \perp \)

But, what can we say about \( c_0 \cup c_1 \), when \( c_0 \vdash_{S} 0 \) and \( c_1 \vdash_{S} \perp \)?
- clearly, \( c_0 \cup c_1 \vdash_{S} \top \)...
- but **no other relation holds**
- in the abstract, **we do not rule out negative values**

We can **extend the initial lattice**:
- \( \geq 0 \) denotes any set of positive or null integers
- \( \leq 0 \) denotes any set of negative or null integers
- \( \neq 0 \) denotes any set of non null integers
- if \( c_0 \vdash_{S} \perp \) and \( c_1 \vdash_{S} 0 \), then \( c_0 \cup c_1 \vdash_{S} \geq 0 \)
Abstraction example 2: constants

Definition: abstraction based on constants

- **concrete elements**: \( \mathcal{P}(\mathbb{Z}) \)
- **abstract elements**: \( \bot, \top, n \) where \( n \in \mathbb{Z} \)
  \( (\mathcal{D}_C^\# = \{ \bot, \top \} \cup \{ n \mid n \in \mathbb{Z} \}) \)
- **abstraction relation**: \( c \vdash_C n \iff c \subseteq \{ n \} \)

We obtain a flat lattice:

![Diagram of a flat lattice]

Abstract reasoning:

- if \( c_0 \vdash_C n_0 \) and \( c_1 \vdash_C n_1 \), then \( \{ k_0 + k_1 \mid k_i \in c_i \} \vdash_C n_0 + n_1 \)
Abstraction example 3: Parikh vector

**Definition: Parikh vector abstraction**
- **concrete elements:** $\mathcal{P}(\mathcal{A}^*)$ (sets of words over alphabet $\mathcal{A}$)
- **abstract elements:** $\{\bot, \top\} \cup (\mathcal{A} \rightarrow \mathbb{N})$
- **abstraction relation:** $c \vdash \phi : \mathcal{A} \rightarrow \mathbb{N}$ if and only if:
  $$\forall w \in c, \forall a \in \mathcal{A}, \ a \text{ appears } \phi(a) \text{ times in } w$$

**Abstract reasoning:**
- **concatenation:**
  if $\phi_0, \phi_1 : \mathcal{A} \rightarrow \mathbb{N}$ and $c_0, c_1$ are such that $c_i \vdash \phi_i$,
  $$\{w_0 \cdot w_1 \mid w_i \in c_i\} \vdash \phi_0 + \phi_1$$

**Information preserved, information deleted:**
- *very precise* information about the number of occurrences
- the order of letters is *totally abstracted away* (lost)
Abstraction example 4: interval abstraction

**Definition: abstraction based on intervals**

- **Concrete elements:** $P(\mathbb{Z})$
- **Abstract elements:** $\bot, \top, (a, b)$ where $a \in \{-\infty\} \cup \mathbb{Z}$, $b \in \mathbb{Z} \cup \{+\infty\}$ and $a \leq b$
- **Abstraction relation:**

  \[
  \emptyset \vdash_{I} \bot \quad S \vdash_{I} \top \quad S \vdash_{I} (a, b) \iff \forall x \in S, \ a \leq x \leq b
  \]

**Operations:** TD
Abstraction example 5: non relational abstraction

**Definition: non relational abstraction**

- **Concrete elements:** $\mathcal{P}(X \to Y)$, inclusion ordering
- **Abstract elements:** $X \to \mathcal{P}(Y)$, pointwise inclusion ordering
- **Abstraction relation:** $c \vdash_{NR} a \iff \forall \phi \in c, \forall x \in X, \phi(x) \in a(x)$

**Information preserved, information deleted:**

- **Very precise** information about the image of the functions in $c$
- **Relations** such as (for given $x_0, x_1 \in X, y_0, y_1 \in Y$) the following are lost:

$$\forall \phi \in c, \phi(x_0) = \phi(x_1)$$

$$\forall \phi \in c, \forall x, x' \in X, \phi(x) \neq y_0 \lor \phi(x') \neq y_1$$
Notion of abstraction relation

**Concrete order:** so far, always inclusion
- the tighter the concrete set, the fewer behaviors
- **smaller concrete** sets correspond to **more precise** properties

**Abstraction relation:** $c \vdash a$ when $c$ satisfies $a$
- if $c_0 \subseteq c_1$ and $c_1$ satisfies $a$, in all our examples, $c_0$ also satisfies $a$

**Abstract order:** in all our examples,
- it matches the abstraction relation as well:
  - if $a_0 \sqsubseteq a_1$ and $c$ satisfies $a_0$, then $c$ also satisfies $a_1$
- **great advantage:** we can reason about implication in the abstract, without looking back at the concrete properties

We will now formalize this in detail...
Outline

1 Abstraction
   - Notion of abstraction
   - Abstraction and concretization functions
   - Galois connections

2 Abstract interpretation

3 Application of abstract interpretation

4 Conclusion
Towards adjoint functions

We consider a **concrete lattice** \((C, \subseteq)\) and an **abstract lattice** \((A, \sqsubseteq)\).

So far, we used **abstraction relations**, that are consistent with orderings:

**Abstraction relation compatibility**

\[ \forall c_0, c_1 \in C, \forall a \in A, \ c_0 \subseteq c_1 \land c_1 \vdash a \implies c_0 \vdash a \]

\[ \forall c \in C, \forall a_0, a_1 \in A, \ c \vdash a_0 \land a_0 \sqsubseteq a_1 \implies c \vdash a_1 \]

When we have a \(c\) (resp., \(a\)) and try to map it into a compatible \(a\) (resp. a \(c\)), the abstraction relation is not a convenient tool.

Hence, we shall use **adjoint functions** between \(C\) and \(A\).

- from concrete to abstract: **abstraction**
- from abstract to concrete: **concretization**
Our **first adjoint function**: 

**Definition: concretization function**

**Concretization function** $\gamma : A \rightarrow C$ (if it exists) maps abstract $a$ into the weakest (i.e., most general) concrete $c$ that satisfies $a$ (i.e., $c \vdash a$).

Note: in common cases, there exists a $\gamma$.

- $c \vdash a$ if and only if $c \subseteq \gamma(a)$
Concretization function: a few examples

**Signs abstraction:**

\[ \gamma_S : \begin{array}{c|c} \top & \mathbb{Z} \\ \pm & \mathbb{Z}^*_+ \\ 0 & \{0\} \\ \mp & \mathbb{Z}^*_- \\ \bot & \emptyset \end{array} \]

**Constants abstraction:**

\[ \gamma_c : \begin{array}{c|c} \top & \mathbb{Z} \\ n & \{n\} \\ \bot & \emptyset \end{array} \]

**Non relational abstraction:**

\[ \gamma_{NR} : (X \rightarrow \mathcal{P}(Y)) \rightarrow \mathcal{P}(X \rightarrow Y) \]
\[ \Phi \rightarrow \{\phi : X \rightarrow Y | \forall x \in X, \phi(x) \in \Phi(x)\} \]

**Parikh vector abstraction:** exercise!
Abstraction function

Our second adjoint function:

**Definition: abstraction function**

**Abstraction function** \( \alpha : C \rightarrow A \) (if it exists) maps concrete \( c \) into the most precise abstract \( a \) that soundly describes \( c \) (i.e., \( c \vdash a \)).

Note: in quite a few cases (including some in this course), there is no \( \alpha \).

**Summary on adjoint functions:**

- \( \alpha \) returns the **most precise abstract predicate** that holds true for its argument.
  - this is called the **best abstraction**
- \( \gamma \) returns the **most general concrete meaning** of its argument.
  - hence, is called the **concretization**
Abstraction: a few examples

**Constants abstraction:**

\[ \alpha_C : (c \subseteq \mathbb{Z}) \mapsto \begin{cases} \bot & \text{if } c = \emptyset \\ n & \text{if } c = \{ n \} \\ \top & \text{otherwise} \end{cases} \]

**Non relational abstraction:**

\[ \alpha_{\mathcal{N}\mathcal{R}} : \mathcal{P}(X \rightarrow Y) \mapsto X \rightarrow \mathcal{P}(Y) \]

\[ c \mapsto (x \in X) \mapsto \{ \phi(x) \mid \phi \in c \} \]

**Signs abstraction** and **Parikh vector abstraction**: exercises
Outline

1 Abstraction
   - Notion of abstraction
   - Abstraction and concretization functions
   - Galois connections

2 Abstract interpretation

3 Application of abstract interpretation

4 Conclusion
Definition

So far, we have:

- **abstraction** \( \alpha : C \to A \)
- **concretization** \( \gamma : A \to C \)

How to tie them together?

**They should agree on a same abstraction relation \( \vdash \) !**

**Definition: Galois connection**

A **Galois connection** is defined by a concrete lattice \( (C, \subseteq) \), an abstract lattice \( (A, \sqsubseteq) \), an abstraction function \( \alpha : C \to A \) and a concretization function \( \gamma : A \to C \) such that:

\[
\forall c \in C, \forall a \in A, \quad \alpha(c) \sqsubseteq a \iff c \subseteq \gamma(a) \quad (\iff c \vdash a)
\]

**Notation:** \( (C, \subseteq) \leftarrow \gamma \to (A, \sqsubseteq) \)

Note: in practice, we shall rarely use \( \vdash \); we use \( \alpha, \gamma \) instead.
Example: constants abstraction and Galois connection

**Constants lattice** $D_C^\# = \{\bot, \top\} \uplus \{n \mid n \in \mathbb{Z}\}$

\[
\begin{align*}
\alpha_C(c) &= \bot \quad \text{if } c = \emptyset \\
\alpha_C(c) &= n \quad \text{if } c = \{n\} \\
\alpha_C(c) &= \top \quad \text{otherwise}
\end{align*}
\]

\[
\begin{align*}
\gamma_C(\top) &\mapsto \mathbb{Z} \\
\gamma_C(n) &\mapsto \{n\} \\
\gamma_C(\bot) &\mapsto \emptyset
\end{align*}
\]

Thus:

- if $c = \emptyset$, $\forall a$, $c \subseteq \gamma_C(a)$, i.e., $c \subseteq \gamma_C(a) \iff \alpha_C(c) = \bot \subseteq a$
- if $c = \{n\}$,
  \[
  \alpha_C(\{n\}) = n \subseteq c \iff c = n \lor c = \top \iff c = \{n\} \subseteq \gamma_C(a)
  \]
- if $c$ has at least two distinct elements $n_0, n_1$, $\alpha_C(c) = \top$ and $c \subseteq \gamma_C(a) \Rightarrow a = \top$, i.e., $c \subseteq \gamma_C(a) \iff \alpha_C(c) = \bot \subseteq a$

**Constant abstraction: Galois connection**

$c \subseteq \gamma_C(a) \iff \alpha_C(c) \subseteq a$, therefore, $(\mathcal{P}(\mathbb{Z}), \subseteq) \xleftarrow{\alpha_C} (D_C^\#, \subseteq)$
Example: non relational abstraction Galois connection

We have defined:

\[ \alpha_{NR}(c \subseteq (X \to Y)) \quad \mapsto \quad (x \in X) \mapsto \{ f(x) \mid f \in c \} \]

\[ \gamma_{NR}(\Phi \in (X \to \mathcal{P}(Y))) \quad \mapsto \quad \{ f : X \to Y \mid \forall x \in X, f(x) \in \Phi(x) \} \]

Let \( c \in \mathcal{P}(X \to Y) \) and \( \Phi \in (X \to \mathcal{P}(Y)) \); then:

\[ \alpha_{NR}(c) \subseteq \Phi \quad \iff \quad \forall x \in X, \alpha_{NR}(c)(x) \subseteq \Phi(x) \]

\[ \iff \quad \forall x \in X, \{ f(x) \mid f \in c \} \subseteq \Phi(x) \]

\[ \iff \quad \forall f \in c, \forall x \in X, f(x) \in \Phi(x) \]

\[ \iff \quad \forall f \in c, f \in \gamma_{NR}(\Phi) \]

\[ \iff \quad c \subseteq \gamma_{NR}(\Phi) \]

Non relational abstraction: Galois connection

\[ c \subseteq \gamma_{NR}(a) \iff \alpha_{NR}(c) \subseteq a, \text{ therefore,} \]

\[ (\mathcal{P}(X \to Y), \subseteq) \xrightarrow{\alpha_{NR}} (X \to \mathcal{P}(Y), \subseteq) \xrightarrow{\gamma_{NR}} \]

Xavier Rival

Abstract Interpretation: Introduction

May 5th, 2017
Galois connection properties

Galois connections have many useful properties.

In the next few slides, we consider a Galois connection $(C, \subseteq) \xleftrightarrow{\gamma \alpha} (A, \subseteq)$ and establish a few interesting properties.

**Extensivity, contractivity**

- $\alpha \circ \gamma$ is **contractive**: $\forall a \in A, \alpha \circ \gamma(a) \subseteq a$
- $\gamma \circ \alpha$ is **extensive**: $\forall c \in C, c \subseteq \gamma \circ \alpha(c)$

**Proof:**

- let $a \in A$; then, $\gamma(a) \subseteq \gamma(a)$, thus $\alpha(\gamma(a)) \subseteq a$
- let $c \in C$; then, $\alpha(c) \subseteq \alpha(c)$, thus $c \subseteq \gamma(\alpha(a))$
Galois connection properties

Monotonicity of adjoints

- $\alpha$ is monotone
- $\gamma$ is monotone

Proof:

- monotonicity of $\alpha$: let $c_0, c_1 \in C$ such that $c_0 \subseteq c_1$; by extensivity of $\gamma \circ \alpha$, $c_1 \subseteq \gamma(\alpha(c_1))$, so by transitivity, $c_0 \subseteq \gamma(\alpha(c_1))$ by definition of the Galois connection, $\alpha(c_0) \subseteq \alpha(c_1)$

- monotonicity of $\gamma$: same principle

Note: many proofs can be derived by duality

Duality principle applied for Galois connections

If $(C, \subseteq) \xleftarrow{\gamma} (A, \sqsubseteq)$, then $(A, \sqsupseteq) \xleftarrow{\alpha} (C, \supseteq)$
Galois connection properties

**Iteration of adjoints**

- $\alpha \circ \gamma \circ \alpha = \alpha$
- $\gamma \circ \alpha \circ \gamma = \gamma$
- $\alpha \circ \gamma$ (resp., $\gamma \circ \alpha$) is idempotent, hence a lower (resp., upper) closure operator

**Proof:**

- $\alpha \circ \gamma \circ \alpha = \alpha$:
  
  let $c \in C$, then $\gamma \circ \alpha (c) \subseteq \gamma \circ \alpha (c)$

  hence, by the Galois connection property, $\alpha \circ \gamma \circ \alpha (c) \subseteq \alpha (c)$

  moreover, $\gamma \circ \alpha$ is extensive and $\alpha$ monotone, so $\alpha (c) \subseteq \alpha \circ \gamma \circ \alpha (c)$

  thus, $\alpha \circ \gamma \circ \alpha (c) = \alpha (c)$

- the second point can be proved similarly (duality); the others follow
Galois connection properties

Properties on iterations of adjoint functions:
Galois connection properties

\( \alpha \) preserves least upper bounds

\[ \forall c_0, c_1 \in C, \ \alpha(c_0 \cup c_1) = \alpha(c_0) \sqcup \alpha(c_1) \]

By duality:

\[ \forall a_0, a_1 \in A, \ \gamma(c_0 \sqcap c_1) = \gamma(c_0) \sqcap \gamma(c_1) \]

**Proof:**

First, we observe that \( \alpha(c_0) \sqcup \alpha(c_1) \subseteq \alpha(c_0 \cup c_1) \), i.e. \( \alpha(c_0 \cup c_1) \) is an upper bound of \( \{ \alpha(c_0), \alpha(c_1) \} \).

We now prove it is the least upper bound. For all \( a \in A \):

\[ \alpha(c_0 \cup c_1) \subseteq a \iff c_0 \cup c_1 \subseteq \gamma(a) \]
\[ \iff c_0 \subseteq \gamma(a) \land c_1 \subseteq \gamma(a) \]
\[ \iff \alpha(c_0) \subseteq a \land \alpha(c_1) \subseteq a \]
\[ \iff \alpha(c_0) \sqcup \alpha(c_1) \subseteq a \]

**Note:** when \( C, A \) are complete lattices, this extends to families of elements
Galois connection properties

Uniqueness of adjoints

- given \( \gamma : C \to A \), there exists at most one \( \alpha : A \to C \) such that
  \[ (C, \subseteq) \xleftarrow{\gamma} (A, \sqsubseteq) \], and, if it exists, \( \alpha(c) = \bigcap \{ a \in A \mid c \subseteq \gamma(a) \} \)

- similarly, given \( \alpha : A \to C \), there exists at most one \( \gamma : C \to A \) such that
  \[ (C, \subseteq) \xrightarrow{\alpha} (A, \sqsubseteq) \], and it is defined dually

Proof of the first point (the other follows by duality):

we assume that there exists an \( \alpha \) so that we have a Galois connection and prove that, \( \alpha(c) = \bigcap \{ a \in A \mid c \subseteq \gamma(a) \} \) for a given \( c \in C \).

- if \( a \in A \) is such that \( c \subseteq \gamma(a) \), then \( \alpha(a) \sqsubseteq c \) thus, \( \alpha(a) \) is a lower bound of \( \{ a \in A \mid c \subseteq \gamma(a) \} \).

- let \( a_0 \in A \) be a lower bound of \( \{ a \in A \mid c \subseteq \gamma(a) \} \). since \( \gamma \circ \alpha \) is extensive, \( c \subseteq \gamma(\alpha(c)) \) and \( \alpha(c) \in \{ a \in A \mid c \subseteq \gamma(a) \} \).
  hence, \( a_0 \sqsubseteq \alpha(c) \)

Thus, \( \alpha(c) \) is the least upper bound of \( \{ a \in A \mid c \subseteq \gamma(a) \} \)
Construction of adjoint functions

The adjoint uniqueness property is actually a very strong property:
- it allows to construct an abstraction from a concretization
- ... or to understand why no abstraction can be constructed :-)

Turning an adjoint into a Galois connection (1)

Let \((C, \subseteq)\) and \((A, \sqsubseteq)\) be two lattices, such that any subset of \(A\) as a greatest lower bound and let \(\gamma : (A, \sqsubseteq) \to (C, \subseteq)\) be a monotone function.

Then, the function below defines a Galois connection:

\[
\alpha(c) = \bigcap \{ a \in A \mid c \subseteq \gamma(a) \}
\]

Example of abstraction with no \(\alpha\): when \(\bigcap\) is not defined on all families, e.g., lattice of convex polyhedra, abstracting sets of points in \(\mathbb{R}^2\).

Exercise: state the dual property and apply the same principle to the concretization.
A characterization of Galois connections

Let \((C, \subseteq)\) and \((A, \sqsubseteq)\) be two lattices, and \(\alpha : C \rightarrow A\) and \(\gamma : A \rightarrow C\) be two monotone functions, such that:

- \(\alpha \circ \gamma\) is contractive
- \(\gamma \circ \alpha\) is extensive

Then, we have a Galois connection

\[(C, \subseteq) \leftrightarrow (A, \sqsubseteq)\]

Proof:

- let \(c \in C\) and \(a \in A\) such that \(\alpha(c) \sqsubseteq a\).
  then: \(\gamma(\alpha(c)) \subseteq \gamma(a)\) (as \(\gamma\) is monotone)
  \(c \subseteq \gamma(\alpha(c))\) (as \(\gamma \circ \alpha\) is extensive)
  thus, \(c \subseteq \gamma(a)\), by transitivity
- the other implication can be proved by duality
Abstract interpretation

Abstract computation

Outline

1. Abstraction

2. Abstract interpretation
   - Abstract computation
   - Fixpoint transfer

3. Application of abstract interpretation

4. Conclusion
Constructing a static analysis

We have set up a notion of abstraction:

- it describes sound approximations of concrete properties with abstract predicates
- there are several ways to formalize it (abstraction, concretization...)
- we now wish to compute sound abstract predicates

In the following, we assume

- a Galois connection

\[(C, \subseteq) \xleftrightarrow{\alpha, \gamma} (A, \sqsubseteq)\]

- a concrete semantics \([\cdot]\), with a constructive definition
  i.e., \([P]\) is defined by constructive equations (\([P] = f(\ldots))\), least fixpoint formula (\([P] = \text{lf}P_\emptyset f\)...
Abstract transformer

A fixed concrete element $c_0$ can be abstracted by $\alpha(c_0)$.

We now consider a monotone concrete function $f : C \to C$

- given $c \in C$, $\alpha \circ f(c)$ abstracts the image of $c$ by $f$
- if $c \in C$ is abstracted by $a \in A$, then $f(c)$ is abstracted by $\alpha \circ f \circ \gamma(a)$:
  - $c \subseteq \gamma(a)$ by assumption
  - $f(c) \subseteq f(\gamma(a))$ by monotonicity of $f$
  - $\alpha(f(c)) \subseteq \alpha(f(\gamma(a)))$ by monotonicity of $\alpha$

Definition: best and sound abstract transformers

- the best abstract transformer approximating $f$ is $f^\# = \alpha \circ f \circ \gamma$
- a sound abstract transformer approximating $f$ is any operator $f^\# : A \to A$, such that $\alpha \circ f \circ \gamma \subseteq f^\#$ (or equivalently, $f \circ \gamma \subseteq \gamma \circ f^\#$)
Example: lattice of signs

- $f : D_C^\# \rightarrow D_C^\#, c \mapsto \{-n \mid n \in c\}$
- $f^\# = \alpha \circ f \circ \gamma$

**Lattice of signs:**

**Abstract negation operator:**

<table>
<thead>
<tr>
<th>$a$</th>
<th>$\Theta^#(a)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\perp$</td>
<td>$\perp$</td>
</tr>
<tr>
<td>$\mp$</td>
<td>$\mp$</td>
</tr>
<tr>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td>$\pm$</td>
<td>$\mp$</td>
</tr>
<tr>
<td>$\mp$</td>
<td>$\mp$</td>
</tr>
<tr>
<td>$\perp$</td>
<td>$\perp$</td>
</tr>
</tbody>
</table>

- here, the best abstract transformer is very easy to compute
- no need to use an approximate one
Abstract $n$-ary operators

We can generalize this to $n$-ary operators, such as boolean operators and arithmetic operators.

Definition: sound and exact abstract operators

Let $g : C^n \rightarrow C$ be an $n$-ary operator, monotone in each component. Then:

- the best abstract operator approximating $g$ is defined by:

$$g^\#: \ A^n \rightarrow A$$
$$\hspace{1cm} (a_0, \ldots, a_{n-1}) \mapsto \alpha \circ g(\gamma(a_0), \ldots, \gamma(a_{n-1}))$$

- a sound abstract transformer approximating $g$ is any operator $g^\#: A^n \rightarrow A$, such that

$$\forall (a_0, \ldots, a_{n-1}) \in A^n, \alpha \circ g(\gamma(a_0), \ldots, \gamma(a_{n-1})) \subseteq g^\#(a_0, \ldots, a_{n-1})$$

(i.e., equivalently, $g(\gamma(a_0), \ldots, \gamma(a_{n-1})) \subseteq \gamma \circ g^\#(a_0, \ldots, a_{n-1})$)
Abstract interpretation

Example: lattice of signs arithmetic operators

Application:
- $\oplus: C^2 \rightarrow C, (c_0, c_1) \mapsto \{n_0 + n_1 \mid n_i \in c_i\}$
- $\otimes: C^2 \rightarrow C, (c_0, c_1) \mapsto \{n_0 \cdot n_1 \mid n_i \in c_i\}$

Best abstract operators:

<table>
<thead>
<tr>
<th>$\oplus^#$</th>
<th>$\bot$</th>
<th>$=$</th>
<th>0</th>
<th>$\pm$</th>
<th>$\top$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$=$</td>
<td>$\bot$</td>
<td>$=$</td>
<td>$=$</td>
<td>$\top$</td>
<td>$\top$</td>
</tr>
<tr>
<td>0</td>
<td>$\bot$</td>
<td>$=$</td>
<td>0</td>
<td>$\pm$</td>
<td>$\top$</td>
</tr>
<tr>
<td>$\pm$</td>
<td>$\bot$</td>
<td>$\top$</td>
<td>$\pm$</td>
<td>$\top$</td>
<td>$\top$</td>
</tr>
<tr>
<td>$\top$</td>
<td>$\bot$</td>
<td>$\top$</td>
<td>$\top$</td>
<td>$\top$</td>
<td>$\top$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\otimes^#$</th>
<th>$\bot$</th>
<th>$=$</th>
<th>0</th>
<th>$\pm$</th>
<th>$\top$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$=$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\pm$</td>
<td>0</td>
<td>$\bot$</td>
</tr>
<tr>
<td>0</td>
<td>$\bot$</td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td>$\pm$</td>
<td>$\bot$</td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td>$\top$</td>
<td>$\bot$</td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
</tr>
</tbody>
</table>

Example of loss in precision:
- $\{8\} \in \gamma_S(\pm)$ and $\{-2\} \in \gamma_S(\bot)$
- $\oplus^\#(\pm, \bot) = \top$ is a lot worse than $\alpha_S(\oplus(\{8\}, \{-2\})) = \pm$
Example: lattice of signs set operators

Best abstract operators approximating $\cup$ and $\cap$:

<table>
<thead>
<tr>
<th>$\cup^#$</th>
<th>$\bot$</th>
<th>$\neg$</th>
<th>0</th>
<th>$\pm$</th>
<th>$T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\neg$</td>
<td>0</td>
<td>$\pm$</td>
<td>$T$</td>
</tr>
<tr>
<td>$\neg$</td>
<td>$\neg$</td>
<td>$\neg$</td>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>$T$</td>
<td>0</td>
<td>$T$</td>
<td>$T$</td>
</tr>
<tr>
<td>$\pm$</td>
<td>$\pm$</td>
<td>$T$</td>
<td>$T$</td>
<td>$\pm$</td>
<td>$T$</td>
</tr>
<tr>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\cap^#$</th>
<th>$\bot$</th>
<th>$\neg$</th>
<th>0</th>
<th>$\pm$</th>
<th>$T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$\neg$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>0</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>0</td>
<td>$\bot$</td>
<td>$0$</td>
</tr>
<tr>
<td>$\pm$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
</tr>
<tr>
<td>$T$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
<td>$\bot$</td>
</tr>
</tbody>
</table>

Example of loss in precision:

$\gamma(\bot) \cup \gamma(\pm) = \{ n \in \mathbb{Z} \mid n \neq 0 \} \subset \gamma(T)$
Outline

1 Abstraction

2 Abstract interpretation
   - Abstract computation
   - Fixpoint transfer

3 Application of abstract interpretation

4 Conclusion
Fixpoint transfer

What about **loops**? semantic functions defined by **fixpoints**?

### Theorem: exact fixpoint transfer

We assume \((C, \subseteq)\) and \((A, \subseteq)\) are complete lattices. We consider a Galois connection \((C, \subseteq) \dashv \vdash (A, \subseteq)\), two functions \(f : C \rightarrow C\) and \(f^\# : A \rightarrow A\) and two elements \(c_0 \in C, a_0 \in A\) such that:

- \(f\) is continuous
- \(f^\#\) is monotone
- \(\alpha \circ f = f^\# \circ \alpha\)
- \(\alpha(c_0) = a_0\)

Then:

- **both** \(f\) and \(f^\#\) **have a least-fixpoint** (by Tarski’s fixpoint theorem)
- \(\alpha(\text{lfp}_{c_0} f) = \text{lfp}_{a_0} f^\#\)
Fixpoint transfer: proof

- $\alpha(\text{lfp}_{c_0} f)$ is a fixpoint of $f^\#$ since:
  \[
  f^\#(\alpha(\text{lfp}_{c_0} f)) = \alpha(f(\text{lfp}_{c_0} f)) = \alpha(\text{lfp}_{c_0} f)
  \]
  since $\alpha \circ f = f^\# \circ \alpha$
  by definition of the fixpoints

- To show that $\alpha(\text{lfp}_{c_0} f)$ is the least-fixpoint of $f^\#$,
  we assume that $X$ is another fixpoint of $f^\#$ greater than $a_0$ and we
  show that $\alpha(\text{lfp}_{c_0} f) \subseteq X$, i.e., that $\text{lfp}_{c_0} f \subseteq \gamma(X)$.

As $\text{lfp}_{c_0} f = \bigcup_{n \in \mathbb{N}} f_0^n(c_0)$ (by Kleene’s fixpoint theorem), it amounts
to proving that $\forall n \in \mathbb{N}, f_0^n(c_0) \subseteq \gamma(X)$.

By induction over $n$:
  - $f^0(c_0) = c_0$, thus $\alpha(f^0(c_0)) = a_0 \subseteq X$; thus, $f^0(c_0) \subseteq \gamma(X)$.
  - Let us assume that $f^n(c_0) \subseteq \gamma(X)$, and let us show that
    $f^{n+1}(c_0) \subseteq \gamma(X)$, i.e. that $\alpha(f^{n+1}(c_0)) \subseteq X$:
    \[
    \alpha(f^{n+1}(c_0)) = \alpha \circ f(f^n(c_0)) = f^\# \circ \alpha(f^n(c_0)) \subseteq f^\#(X) = X
    \]
    as $\alpha(f^n(c_0)) \subseteq X$ and $f^\#$ is monotone.
Constructive analysis of loops

How to get a constructive fixpoint transfer theorem?

Theorem: fixpoint abstraction

Under the assumptions of the previous theorem, and with the following additional hypothesis:

- lattice $A$ is of finite height

We compute the sequence $(a_n)_{n \in \mathbb{N}}$ defined by $a_{n+1} = a_n \sqcup f^\#(a_n)$.

Then, $(a_n)_{n \in \mathbb{N}}$ converges and its limit $a_\infty$ is such that $\alpha(\text{lfp}_{c_0} f) = a_\infty$.

Proof: exercise.

Note:

- the assumptions we have made are too restrictive in practice
- more general fixpoint abstraction methods in the next lectures
Outline

1. Abstraction
2. Abstract interpretation
3. Application of abstract interpretation
4. Conclusion
Comparing existing semantics

1. A concrete semantics $[P]$ is given: e.g., big steps operational semantics
2. An abstract semantics $[P]^\#$ is given: e.g., denotational semantics
3. Search for an abstraction relation between them
e.g., $[P]^\# = \alpha([P])$, or $[P] \subseteq \gamma([P]^\#)$

Examples:
- finite traces semantics as an abstraction of bi-finitary trace semantics
- denotational semantics as an abstraction of trace semantics
- types as an abstraction of denotational semantics
- ...

Payoff:
- better understanding of ties across semantics
- chance to generalize existing definitions
Derivation of a static analysis

1. Start from a **concrete semantics** \([P]\)

2. **Choose an abstraction** defined by a Galois connection or a concretization function (usually)

3. **Derive an abstract semantics** \([P]^{\#}\) such that \([P] \subseteq \gamma([P]^{\#})\)

**Examples:**
- derivation of an analysis with a numerical lattice (constants, intervals...)
- construction of an analysis for a complex programming language

**Payoff:**
- the derivation of the abstract semantics is quite systematic
- this process offers good opportunities for a modular analysis design

There are many ways to apply abstract interpretation.
A very simple language and its semantics

We now apply this to a very simple language, and derive a static analysis step by step, from a concrete semantics and an abstraction.

- we assume a fixed set of $n$ integer variables $x_0, \ldots, x_{n-1}$
- we consider the language defined by the grammar below:

$$P ::= x_i = n \quad \text{where } n \in \mathbb{Z}$$
$$\quad | \quad x_i = x_j + x_k \quad \text{basic, three-addresses arithmetics}$$
$$\quad | \quad x_i = x_j - x_k \quad \text{basic, three-addresses arithmetics}$$
$$\quad | \quad x_i = x_j \cdot x_k \quad \text{basic, three-addresses arithmetics}$$
$$\quad | \quad P; P \quad \text{concatenation}$$
$$\quad | \quad \text{input}(x_i) \quad \text{reading of a positive input}$$
$$\quad | \quad \text{if}(x_i > 0) P \ \text{else} \ P$$
$$\quad | \quad \text{while}(x_i > 0) P$$

- a state is a vector $\sigma = (\sigma_0, \ldots, \sigma_{n-1}) \in \mathbb{Z}^n$
- a single initial state $\sigma_{\text{init}} = (0, \ldots, 0)$
Concrete semantics

We let $[P]: \mathcal{P}(\mathbb{Z}^n) \to \mathcal{P}(\mathbb{Z}^n)$ be defined by:

\[
\begin{align*}
[x_i = n](S) &= \{\sigma[i \leftarrow n] | \sigma \in S\} \\
[x_i = x_j + x_k](S) &= \{\sigma[i \leftarrow \sigma_j + \sigma_k] | \sigma \in S\} \\
[x_i = x_j - x_k](S) &= \{\sigma[i \leftarrow \sigma_j - \sigma_k] | \sigma \in S\} \\
[x_i = x_j \cdot x_k](S) &= \{\sigma[i \leftarrow \sigma_j \cdot \sigma_k] | \sigma \in S\} \\
[\text{input}(x_i)](S) &= \{\sigma[i \leftarrow n] | \sigma \in S \land n > 0\} \\
[\text{if}(x_i > 0) P_0 \text{ else } P_1](S) &= [P_0](\{\sigma \in S | \sigma_i > 0\}) \\
&\quad \cup [P_1](\{\sigma \in S | \sigma_i \leq 0\}) \\
[\text{while}(x_i > 0) P](S) &= \{\sigma \in \text{lfp}_S f | \sigma_i \leq 0\} \text{ where} \\
f : S' \mapsto S' \cup [P](\{\sigma \in S' | \sigma_i > 0\})
\end{align*}
\]

- given a complete program $P$, the \textbf{reachable states} are defined by $[P](\{\sigma_{\text{init}}\})$
Abstraction

We compose two abstractions:

- **non relational abstraction**: the values a variable may take is abstracted separately from the other variables
- **sign abstraction**: the set of values observed for each variable is abstracted into the lattice of signs

**Abstraction**

- **concrete domain**: \((\mathcal{P}(\mathbb{Z}^n), \subseteq)\)
- **abstract domain**: \((D\#, \subseteq)\), where \(D\# = (D_S\#)^n\) and \(\subseteq\) is the pointwise ordering
- **Galois connection** \((\mathcal{P}(\mathbb{Z}), \subseteq) \xleftarrow{\gamma} (D\#, \subseteq)\), defined by

\[
\begin{align*}
\alpha : & \quad S 
\quad \mapsto 
\quad (\alpha_S(\{\sigma_0 \mid \sigma \in S\}), \ldots, \alpha_S(\{\sigma_{n-1} \mid \sigma \in S\})) \\
\gamma : & \quad S\# 
\quad \mapsto 
\quad \{\sigma \in \mathbb{Z}^n \mid \forall i, \sigma_i \in \gamma_S(S_i\#)\}
\end{align*}
\]
Example

Factorial function:

```plaintext
input(x_0);
x_1 = 1;
x_2 = 1;
while(x_0 > 0){
    x_1 = x_0 \cdot x_1;
    x_0 = x_0 - x_2;
}
```

Abstraction of the semantics:
- abstract pre-condition: $(\top, \top, \top)$
- abstract state before the loop: $(\pm, \pm, \pm)$
- abstract post-condition (after the loop): $(\top, \pm, \pm)$
Computation of the abstract semantics

We search for an abstract semantics $\llbracket P \rrbracket^\# : D^\# \to D^\#$ such that:

$$\alpha \circ \llbracket P \rrbracket = \llbracket P \rrbracket^\# \circ \alpha$$

We observe that:

$$\alpha(S) = (\alpha_S(\{\sigma_0 \mid \sigma \in S\}), \ldots, \alpha_S(\{\sigma_{n-1} \mid \sigma \in S\}))$$

$$\alpha \circ \llbracket P \rrbracket(S) = (\alpha_S(\{\sigma_0 \mid \sigma \in \llbracket P \rrbracket(S)\}), \ldots, \alpha_S(\{\sigma_{n-1} \mid \sigma \in \llbracket P \rrbracket(S)\}))$$

We start with $x_i = n$:

$$\alpha \circ \llbracket x_i = n \rrbracket(S)$$

$$= (\alpha_S(\{\sigma_0 \mid \sigma \in \llbracket P \rrbracket(\{\sigma[i \leftarrow n] \mid \sigma \in S\}))\}), \ldots,$$

$$\alpha_S(\{\sigma_{n-1} \mid \sigma \in \llbracket P \rrbracket(\{\sigma[i \leftarrow n] \mid \sigma \in S\}))\}))$$

$$= (\alpha_S(\{\sigma_0 \mid \sigma \in S\}), \ldots, \alpha_S(\{\sigma_{n-1} \mid \sigma \in S\}))[i \leftarrow \alpha_S(\{n\})]$$

$$= \alpha(S)[i \leftarrow \alpha_S(\{n\})]$$

$$= \llbracket x_i = n \rrbracket^\#(\alpha(S))$$

where

$$\llbracket x_i = n \rrbracket^\#(S^\#) = S^\#[i \leftarrow \alpha_S(\{n\})]$$
Computation of the abstract semantics

Other assignments are treated in a similar manner:

\[
\begin{align*}
[x_i = x_j + x_k]^\#(S^\#) &= S^\#[i \leftarrow S_j^\# \oplus^\# S_k^\#] \\
[x_i = x_j - x_k]^\#(S^\#) &= S^\#[i \leftarrow S_j^\# \ominus^\# S_k^\#] \\
[x_i = x_j \cdot x_k]^\#(S^\#) &= S^\#[i \leftarrow S_j^\# \otimes^\# S_k^\#] \\
\text{input}(x_i)^\#(S^\#) &= S^\#[i \leftarrow \pm]
\end{align*}
\]

Proofs are left as exercises
Computation of the abstract semantics

We now consider the case of tests:

\[ \alpha \circ \left[ \text{if} (x_i > 0) \ P_0 \ \text{else} \ P_1 \right] (S) \]
\[ = \alpha(\left[ P_0 \right](\{\sigma \in S \mid \sigma_i > 0\}) \cup \left[ P_1 \right](\{\sigma \in S \mid \sigma_i \leq 0\})) \]
\[ = \alpha(\left[ P_0 \right](\{\sigma \in S \mid \sigma_i > 0\})) \cup \alpha(\left[ P_1 \right](\{\sigma \in S \mid \sigma_i \leq 0\})) \]

as \( \alpha \) preserves least upper bounds
\[ = \left[ P_0 \right]^\#(\alpha(\{\sigma \in S \mid \sigma_i > 0\})) \cup \left[ P_1 \right]^\#(\alpha(\{\sigma \in S \mid \sigma_i \leq 0\})) \]
\[ = \left[ P_0 \right]^\#(\alpha(S) \cap \top[i \leftarrow \pm]) \cup \left[ P_1 \right]^\#(\alpha(S)) \]
\[ = \left[ \text{if} (x_i > 0) \ P_0 \ \text{else} \ P_1 \right]^\#(\alpha(S)) \]

where \( \left[ \text{if} (x_i > 0) \ P_0 \ \text{else} \ P_1 \right]^\#(S^\#) = \left[ P_0 \right]^\#(S^\# \cap \top[i \leftarrow \pm]) \cup \left[ P_1 \right]^\#(S^\#) \)

In the case of loops:

\[ \left[ \text{while} (x_i > 0) \ P \right]^\#(S^\#) = \text{lfp}_{S^\#} f^\# \]

where \( f^\# : S^\# \mapsto S^\# \cup \left[ P \right]^\#(S^\# \cap \top[i \leftarrow \pm]) \)

Proof: exercise
Abstract semantics

Abstract semantics and soundness

We have derived the following definition of $[[P]]^\#$:

$$[[x_i = n]]^\#(S^\#) = S^\#[i \leftarrow \alpha_S(\{n\})]$$
$$[[x_i = x_j + x_k]]^\#(S^\#) = S^#[i \leftarrow S_j^\# \oplus^\# S_k^\#]$$
$$[[x_i = x_j - x_k]]^\#(S^\#) = S^#[i \leftarrow S_j^\# \ominus^\# S_k^\#]$$
$$[[x_i = x_j \cdot x_k]]^\#(S^\#) = S^#[i \leftarrow S_j^\# \otimes^\# S_k^\#]$$
$$[[\text{input}(x_i)]]^\#(S^\#) = S^#[i \leftarrow +]$$
$$[[\text{if}(x_i > 0) \; P_0 \; \text{else} \; P_1]]^\#(S^\#) = [[P_0]]^\#(S^\# \sqcap \top[i \leftarrow +]) \sqcup [[P_1]]^\#(S^\#)$$
$$[[\text{while}(x_i > 0) \; P]]^\#(S^\#) = \text{lfp}_{S^\#} f^\# \text{ where } f^\#: S^\# \mapsto S^\# \sqcup [[P]]^\#(S^\# \sqcap \top[i \leftarrow +])$$

Furthermore, for all program $P$: $\alpha \circ [[P]] = [[P]]^\# \circ \alpha$

An over-approximation of the final states is computed by $[[P]]^\#(\top)$. 
Example

Factorial function:

\begin{verbatim}
input(x_0);
x_1 = 1;
x_2 = 1;
while(x_0 > 0){
    x_1 = x_0 \cdot x_1;
    x_0 = x_0 - x_2;
}
\end{verbatim}

Abstract state \textbf{before the loop:} \((\pm, \pm, \pm)\)

Iterates on the loop:

<table>
<thead>
<tr>
<th>iterate</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>x_0</td>
<td>\pm</td>
<td>\top</td>
<td>\top</td>
</tr>
<tr>
<td>x_1</td>
<td>\pm</td>
<td>\pm</td>
<td>\pm</td>
</tr>
<tr>
<td>x_2</td>
<td>\pm</td>
<td>\pm</td>
<td>\pm</td>
</tr>
</tbody>
</table>

Abstract state \textbf{after the loop:} \((\top, \pm, \pm)\)
Outline

1. Abstraction
2. Abstract interpretation
3. Application of abstract interpretation
4. Conclusion
Summary

This lecture:
- **abstraction** and its formalization
- **computation of an abstract semantics** in a very simplified case

Next lectures:
- **construction** of a few **non trivial abstractions**
- **more general** ways to **compute sound abstract properties**

Update on projects...