Objects

When analyzing/proving programs we have to consider “objects” that represent some part of the computation state, such as:

- **Values**: booleans, integers, \ldots \mathcal{V}
- **Variable names**: \textbf{X}
- **Environments**: \textbf{X} \mapsto \mathcal{V}
- **Heaps**: dynamic allocation;
- **Control points**: procedure names, labels, \ldots ;
- **States**: control & memory states;
- **Finite prefix traces**;
- **Maximal finite or infinite traces** (for deterministic programs);
- **Sets of maximal finite or infinite traces** (for nondeterministic programs);
- \ldots
Properties

Properties are \textit{“sets of objects”} (which have that property).

Examples:

- \textit{odd naturals}: \(\{1, 3, 5, \ldots, 2n + 1, \ldots\}\)
- \textit{even integers}: \(\{2z \mid z \in \mathbb{Z}\}\)
- \textit{values of integer variables}: \(\{z \in \mathbb{Z} \mid \text{minint} \leq z \leq \text{maxint}\}\)
- \textit{values of maybe uninitialized integer variables}: \(\{z \in \mathbb{Z} \mid \text{minint} \leq z \leq \text{maxint}\} \cup \{\Omega_m \mid m \in \mathcal{M}\}\) where \(\mathcal{M}\) is a set of error messages
- \textit{equality of two variables} \(x\) and \(y\): \(\{\rho \in X \leftrightarrow \mathcal{V} \mid x, y \in \text{dom}(\rho) \land \rho(x) = \rho(y)\}\)
- \textit{invariance property} (of a program with states in \(\Sigma\)): \(I \in \wp(\Sigma)\)
- \textit{trace property}: \(T \in \wp(\Sigma^\infty)\)
Complete Lattice of Concrete Properties

The set of properties $\wp(\Sigma)$ of objects in $\Sigma$ is a complete boolean lattice:

$$\langle \wp(\Sigma), \subseteq, \emptyset, \Sigma, \cup, \cap, \neg \rangle$$

where

- A property $P \in \wp(\Sigma)$ is the set of objects which have the property $P$
Complete Lattice of Concrete Properties

The set of properties $\varphi(\Sigma)$ of objects in $\Sigma$ is a complete boolean lattice:

$$\langle \varphi(\Sigma), \subseteq, \emptyset, \Sigma, \cup, \cap, \neg \rangle$$

where

- A property $P \in \varphi(\Sigma)$ is the set of objects which have the property $P$

  - $\subseteq$ is logical implication since $P \subseteq Q$ means that all objects with property $P$ have property $Q$ ($o \in P \implies o \in Q$)
  - $\emptyset$ is false (ff)
  - $\Sigma$ is true (tt)
  - $\cup$ is disjunction (objects which have either property $P$ and/or have property $Q$ belong to $P \cup Q$)
  - $\cap$ is conjunction (object which have property $P$ and have property $Q$ belong to $P \cap Q$)
  - $\neg$ is negation (objects not having property $P$ are those in $\Sigma \setminus P$)


Abstraction - Informally

- Abstraction replace something “concrete” by a schematic description that account for some, and in general not all properties, either known or inferred i.e. an “abstract” model or concept
- In practice, such an abstract model of a concrete object \( \sigma \)
  - can describe some of the properties of the concrete object
  - cannot describe all properties of this concrete object
Abstraction - Informally

- Abstraction replace something “concrete” by a schematic description that account for some, and in general not all properties, either known or inferred i.e. an “abstract” model or concept

- In practice, such an abstract model of a concrete object σ
  
  - can describe some of the properties of the concrete object
  
  - cannot describe all properties of this concrete object

- So an abstraction of properties in $\varphi(\Sigma)$ of objects in $\Sigma$ is essentially a subset $A \subseteq \varphi(\Sigma)$ such that:
  
  - The properties in $A$ are the concrete properties that can be described exactly by the abstraction, without any loss of information
  
  - The properties in $\varphi(\Sigma) \setminus A$ are the properties that cannot be described exactly by the abstraction, and have to be referred to by being approximated in some way or another by abstract properties in $A$
Abstraction - Examples

Abstraction, informal introduction

– Abstraction replace something "concrete" with a "mathematical" description that account for some, and in general not all properties, either known or inferred i.e. an "abstract" model or concept.

– In practice, such an abstract model of a concrete object can describe some of the properties of the concrete object, cannot describe all properties of this concrete object.

Since otherwise this property would have to be "exactly that object" i.e. a real, actual, material, corporeal, property.

– So an abstraction of properties in (˚) of objects in (˚) is essentially a subset A (˚) such that:
  - The properties in A are the concrete properties that can be described exactly without any loss of information.
  - The properties in (˚) \ A are the properties that cannot be described exactly by the abstraction, and have to be referred to by being approximated in some way or another by abstract properties in A.

Intuitive example 1 of abstraction

Cars → Trademarks

– A concrete property of cars is a set of cars.
– It can be abstracted by the set of their trademarks.
– A trademark is a set of cars.
– An abstract property of cars is a set of cars which, whenever it contains one car of some trademark, also contains all cars of that trademark.

Formally, if t → Cars yield the trademark t(c) of a car c → Cars then the abstraction of P → Cars is α → (Cars) = {t(c) | c → P} and the set of cars described by an abstract property T → Trademarks is \( \{ c | t(c) \in T \} \).

Intuitive example 2 of abstraction

Scientific papers → set of keywords

– A concrete property of scientific papers is a set of scientific papers.
– Each scientific paper is abstracted by a list of keywords.
– A property of scientific papers can be abstracted by the list of keywords appearing in all papers with that property.
– An abstract property of scientific papers is therefore a set of papers which have all keywords belonging to the list.

Formally if w → Scientific papers provides the set of words w(p) appearing in a paper p → Scientific papers and Keywords → Words is a given list of keywords, then a property P → Scientific papers is abstracted by the set of keywords α → (Scientific papers) = {w(p) \ Keywords | p → P} and a set K of keywords stands for the concrete property \( \{ p | w(p) \in Keywords \} \).

Course 16.399: "Abstract interpretation", Tuesday, April 12, 2005

P. Cousot, 2005
Abstraction - Examples

Cars $\xrightarrow{\alpha}$ Trademarks

- A concrete property of cars is a set of cars
- It can be abstracted by the set of their trademarks
- A trademark is a set of cars
- An abstract property of cars is a set of cars which, whenever it contains one car of some trademark, also contains all cars of that trademark
Abstraction - Examples

Cars $\xrightarrow{\alpha} \text{Trademarks}$

- A concrete property of cars is a set of cars
- It can be abstracted by the set of their trademarks
- A trademark is a set of cars
- An abstract property of cars is a set of cars which, whenever it contains one car of some trademark, also contains all cars of that trademark

Scientific papers $\xrightarrow{\alpha} \text{set of keywords}$
Abstraction - Examples

Cars $\xrightarrow{\alpha} \text{Trademarks}$

- A concrete property of cars is a set of cars
- It can be abstracted by the set of their trademarks
- A trademark is a set of cars
- An abstract property of cars is a set of cars which, whenever it contains one car of some trademark, also contains all cars of that trademark

Scientific papers $\xrightarrow{\alpha} \text{set of keywords}$

- A concrete property of scientific papers is a set of scientific papers
- Each scientific paper is abstracted by a list of keywords
- A property of scientific papers can be abstracted by the list of keywords appearing in all papers with that property
- An abstract property of scientific papers is therefore a set of papers which have all keywords belonging to the list
Abstract/Concrete Properties

*Abstraction* in a reasoning/computation such that:

– Only some properties \( A \subseteq \wp(\Sigma) \) of the objects in \( \Sigma \) can be used;
– The properties \( P \in A \) that can be used are called *abstract*;
– The properties \( P \in \wp(\Sigma) \) are called *concrete*;
Abstract/Concrete Properties

**Abstraction** in a reasoning/computation such that:
- Only some properties \( A \subseteq \wp(\Sigma) \) of the objects in \( \Sigma \) can be used;
- The properties \( P \in A \) that can be used are called *abstract*;
- The properties \( P \in \wp(\Sigma) \) are called *concrete*;

- When approximating a concrete property \( P \in \wp(\Sigma) \), by an abstract property \( \overline{P} \in A \), with \( \overline{P} \neq P \), a relation must be established between the concrete \( P \) and abstract property \( \overline{P} \) to establish that

  "\( \overline{P} \in A \) is an approximation/abstraction of \( P \in \wp(\Sigma) \)"

so as to ensure the soundness of the reasoning in the abstract with respect to the concrete, exact one.
Abstract/Concrete Properties

Abstraction in a reasoning/computation such that:
- Only some properties $A \subseteq \wp(\Sigma)$ of the objects in $\Sigma$ can be used;
- The properties $P \in A$ that can be used are called abstract;
- The properties $P \in \wp(\Sigma)$ are called concrete;
- When approximating a concrete property $P \in \wp(\Sigma)$, by an abstract property $\overline{P} \in A$, with $\overline{P} \neq P$, a relation must be established between the concrete $P$ and abstract property $\overline{P}$ to establish that

\[
\text{“} \overline{P} \in A \text{ is an approximation/abstraction of } P \in \wp(\Sigma) \text{”}
\]

so as to ensure the soundness of the reasoning in the abstract with respect to the concrete,

- Approximation from above: $P \subseteq \overline{P}$
- Approximation from below: $P \supseteq \overline{P}$
Minimal Abstraction

- Assume concrete properties \( P \in \wp(\Sigma) \) must be approximated from above by \( \overline{P} \in A \subset \wp(\Sigma) \) such that \( P \subseteq \overline{P} \)
- The smaller the abstract property \( \overline{P} \) is, the most precise the approximation will be
- Obviously, there might be no minimal abstract property at all in \( A \)

- If a concrete property \( P \in \wp(\Sigma) \) has minimal upper approximations \( \overline{P} \in A \):
  - \( P \subseteq \overline{P} \)
  - \( \forall P' : P \subseteq P' \subset \overline{P} \)
  then such minimal approximations are more precise than the non-minimal ones
- So minimal abstract upper approximations, if any, should be prefered
- In particular, an abstract property \( \overline{P} \in A \) is best approximated by itself
Minimal Abstraction

- Assume concrete properties \( P \in \wp(\Sigma) \) must be approximated from above by \( \overline{P} \in A \subset \wp(\Sigma) \) such that \( P \subseteq \overline{P} \).
- The smaller the abstract property \( \overline{P} \) is, the most precise the approximation will be.
- Obviously, there might be no minimal abstract property at all in \( A \).

- If a concrete property \( P \in \wp(\Sigma) \) has minimal upper approximations \( \overline{P} \in A \):
  - \( P \subseteq \overline{P} \)
  - \( \forall \overline{P}' : P \subseteq \overline{P}' \subseteq \overline{P} \)
then such minimal approximations are more precise than the non-minimal ones.
- So minimal abstract upper approximations, if any, should be preferred.
- In particular, an abstract property \( \overline{P} \in A \) is best approximated by itself.
Best Abstraction

- A very handy choice of the abstract properties $A \subseteq \wp(\Sigma)$ is when every concrete property $P$ has a best approximation $\overline{P} \in P$:
  - $P \subseteq \overline{P}$
  - $\forall \overline{P'} \in A : (P \subseteq \overline{P'}) \implies (\overline{P} \subseteq \overline{P'})$
- It follows that $\overline{P}$ is the glb of the over-approximations of $P$ in $A$:
  \[
  \overline{P} = \bigcap \{ \overline{P'} \in A \mid P \subseteq \overline{P'} \} \in A
  \]
Abstraction based on Closure Operator

concrete domain $\langle \wp(\Sigma'), \subseteq, \emptyset, \Sigma, \cup, \cap, \neg \rangle$

abstraction map $\rho \in \wp(\Sigma') \mapsto A$

$$\rho(P) \overset{\text{def}}{=} \bigcap\{ \overline{P} \in A \mid P \subseteq \overline{P} \}$$

Then $\rho$ is an upper closure operator on $\wp(\Sigma')$. 
Abstraction based on Closure Operator

concrete domain \( \langle \wp(\Sigma'), \subseteq, \emptyset, \Sigma, \cup, \cap, \neg \rangle \)

abstraction map \( \rho \in \wp(\Sigma') \mapsto A \)

\[
\rho(P) \overset{\text{def}}{=} \bigcap \{ \overline{P} \in A \mid P \subseteq \overline{P} \}
\]

Then \( \rho \) is an upper closure operator on \( \wp(\Sigma') \).

The most precise abstraction?
Abstraction based on Closure Operator

concrete domain $\langle \mathcal{O}(\Sigma), \subseteq, \emptyset, \Sigma, \cup, \cap, \neg \rangle$

abstraction map $\rho \in \mathcal{O}(\Sigma) \mapsto A$

$$\rho(P) \overset{\text{def}}{=} \bigcap \{ \overline{P} \in A \mid P \subseteq \overline{P} \}$$

Then $\rho$ is an upper closure operator on $\mathcal{O}(\Sigma)$.

The most precise abstraction?

The most imprecise abstraction?
Correspondence Between Abstract and Concrete Properties

Given a closure operator $\rho$ on a poset $\langle L, \sqsubseteq \rangle$ (typically $L$ is $\varnothing(\Sigma)$), Morgado’s theorem states that for all $P, P' \in L$

$$\rho(P) \sqsubseteq \rho(P') \iff P \sqsubseteq \rho(P')$$

that is, by definition of Galois connections ($1_L \overset{\text{def}}{=} \lambda x \in L. x$):

$$\langle L, \sqsubseteq \rangle \xleftarrow{\rho} \langle \rho(L), \sqsubseteq \rangle$$
Let \( \langle A, \leq \rangle \) be an order-isomorphic representation of the abstract domain \( \langle \rho(L), \sqsubseteq \rangle \). We have
\[
\langle \rho(L), \sqsubseteq \rangle \xlongequal{\epsilon, \epsilon^{-1}} \langle A, \leq \rangle
\]
where \( \epsilon^{-1} \) is the inverse of the bijection \( \epsilon \in \rho(L) \mapsto A \) and \( \epsilon \in \rho(L) \mapsto A \).
Let \( \langle A, \leq \rangle \) be an order-isomorphic representation of the abstract domain \( \langle \rho(L), \sqsubseteq \rangle \). We have

\[
\langle \rho(L), \sqsubseteq \rangle \xleftarrow{\epsilon} \epsilon^{-1} \xrightarrow{\epsilon} \langle A, \leq \rangle
\]

where \( \epsilon^{-1} \) is the inverse of the bijection \( \epsilon \in \rho(L) \mapsto A \) and \( \epsilon \in \rho(L) \xrightarrow{m} A \).

By composition, we get:

\[
\langle L, \sqsubseteq \rangle \xleftarrow{1_L \circ \epsilon^{-1}} \epsilon \circ \rho \xrightarrow{\epsilon \circ \rho} \langle A, \leq \rangle
\]
Abstract Domain by Galois Surjection

- Inversely, we can consider a Galois surjection

\[ \langle L, \sqsubseteq \rangle \leftrightarrow \begin{array}{c} \gamma \\ \alpha \end{array} \langle A, \leq \rangle \]

- Then \( \rho = \gamma \circ \alpha \) is a closure operator and \( \langle A, \leq \rangle \) is order-isomorphically to \( \langle \rho(L), \sqsubseteq \rangle \)

- We have an order-isomorphically representation of the abstract domain \( \langle \rho(L), \sqsubseteq \rangle \), which is a Moore family.
Abstract Domain by Galois Surjection

- Inversely, we can consider a Galois surjection

\[ \langle L, \sqsubseteq \rangle \leftrightarrow \Arrow{\gamma}{\alpha} \langle A, \leq \rangle \]

- Then \( \rho = \gamma \circ \alpha \) is a closure operator order-isomorphic to \( \langle \rho(L), \sqsubseteq \rangle \).
- We have an order-isomorphic abstract domain \( \langle \rho(L), \sqsubseteq \rangle \), which directly follows from P. Cousot, 2005.
Abstract Domain by Galois Surjection

- Inversely, we can consider a Galois surjection

\[
\langle L, \sqsubseteq \rangle \leftrightarrow \frac{\gamma}{\alpha} \langle A, \leq \rangle
\]

- Then \( \rho = \gamma \circ \alpha \) is a closure operator and \( \langle A, \leq \rangle \) is order-isomorphic to \( \langle \rho(L), \sqsubseteq \rangle \)

- We have an order-isomorphic representation of the abstract domain \( \langle \rho(L), \sqsubseteq \rangle \), which is a Moore family.

Example: Intervals