Semantics of Programs and Semantic Properties

MPRI — Cours 2.6 “Interprétation abstraite : application à la vérification et à l’analyse statique”

Xavier Rival

INRIA, ENS, CNRS

Jan, 6th. 2012

Overview of the lecture

- **Choosing the right semantics** is the first step in the design of a **static analysis**
  - it should capture the relevant properties
  - non relevant properties may be abstracted
    typically, one by one, by composing several abstractions
- **Abstract interpretation** is a good framework to **compare** various semantics (independently from the application)
  Application: designing **lattices of semantics**
- **Semantic properties** should also be classified, to better guide the choice of a base semantics to reason about them

Outline

Definition

Programs/systems and their **executions** need be formalized:

- **state**: status of the machine at a given time
- **execution**: defined by transitions from a state to the next one

Transition system (TS)

A *transition system* is a tuple $(\mathbb{S}, \rightarrow)$ where:

- $\mathbb{S}$ is the set of states of the system
- $\rightarrow \subseteq \mathcal{P}(\mathbb{S} \times \mathbb{S})$ is the transition relation of the system

Furthermore, transition systems may be enriched with

- a set of initial states $\mathbb{S}_{\mathcal{I}} \subseteq \mathbb{S}$
- a set of final states $\mathbb{S}_{\mathcal{F}} \subseteq \mathbb{S}$

Notes:

- the set of states may be infinite
- steps are *discrete* (not continuous)

Example TS: functional language

λ-terms

The set of λ-terms is defined by:

$$\begin{array}{llll} t, u, \ldots & ::= & x & \text{variable} \\ & | & \lambda x \cdot t & \text{abstraction} \\ & | & t\, u & \text{application} \end{array}$$

β reduction

- $(\lambda x \cdot t)\, u \rightarrow_\beta t[x \leftarrow u]$
- if $u \rightarrow_\beta v$ then $\lambda x \cdot u \rightarrow_\beta \lambda x \cdot v$
- if $u \rightarrow_\beta v$ then $u\, t \rightarrow_\beta v\, t$
- if $u \rightarrow_\beta v$ then $t\, u \rightarrow_\beta t\, v$

A program is a transition system:

- $\mathbb{S}$ is the set of λ-terms
- for λ-calculs $(\rightarrow)$ is $(\rightarrow_\beta)$
  in ML, execution order specified: $(\rightarrow) \subset (\rightarrow_\beta)$ (no equality)

Example TS: stack machine

The **Krivine** machine, used to compile **functional languages**:

- **Programs**: sequences of instructions

$$\begin{array}{rcl} c & ::= & i \cdot c \mid \epsilon \\ i & ::= & \textbf{Access}(n) \mid \textbf{Push}(c) \mid \textbf{Grab};\, n \in \mathbb{N} \end{array}$$

- **States** are of the form (c, e, s), where
  - c is a program
  - e is the **environment** and s is the **stack**: lists of pairs (c, e) (denoting sub-expressions and the environment they should be evaluated in)
- **Transitions**:

$$\begin{array}{rcl} (\textbf{Access}(0) \cdot c, (c_0, e_0) \cdot e, s) & \rightarrow & (c_0, e_0, s) \\ (\textbf{Access}(n+1) \cdot c, (c_0, e_0) \cdot e, s) & \rightarrow & (\textbf{Access}(n), e, s) \\ (\textbf{Push}(c') \cdot c, e, s) & \rightarrow & (c, e, (c', e) \cdot s) \\ (\textbf{Grab} \cdot c, e, (c_0, e_0) \cdot s) & \rightarrow & (c, (c_0, e_0) \cdot e, s) \end{array}$$

Labelled transition system

Definition of states:

- depends on the kinds of programs to abstract
- typically, we can separate **control** and **memory**

Labelled transition system (LTS)

A *labelled transition system* is a transition system $(\mathbb{S}, \rightarrow)$ the states of which can be described as pairs of a control state and a memory state, i.e., where:

- $\mathbb{S} = \mathbb{L} \times \mathbb{M}$
- $\mathbb{L}$ is the set of *labels* or *control states*
- $\mathbb{M}$ is the set of *memory states*

- labels may denote a point in the code and may include a call stack (languages with procedures)
- **error state**: usually added, separate Ω value, so that $\mathbb{S} = \mathbb{L} \times \mathbb{M} \uplus \{\Omega\}$

Example LTS: imperative language

$$
\begin{array}{rcl}
\texttt{i} & ::= & x := \mathrm{e}; \\
 & | & \textbf{if}(\texttt{c})\ \texttt{b}\ \textbf{else}\ \texttt{b} \\
 & | & \textbf{while}(\texttt{c})\ \texttt{b}
\end{array}
$$

$$
\texttt{b} \ ::= \ \{\texttt{i};\ldots;\texttt{i};\}
$$

- $\mathbb{X}$: finite, predefined set of variables
- $\mathbb{L}$: before and after each statement

Definition of $\rightarrow$:

transitions for all instructions

- $l_0 : x = \mathrm{e};\ l_1$:
  - if $[\![\mathrm{e}]\!](m) \neq \Omega$, then $(l_0, m) \rightarrow (l_1, m[x \leftarrow [\![\mathrm{e}]\!](\rho)])$
  - if $[\![\mathrm{e}]\!](m) = \Omega$, then $(l_0, m) \rightarrow \Omega$
- $l_0 : \textbf{while}(\texttt{c})\{l_1 : \texttt{b}_t\ l_2\}\ l_3$:
  - if $[\![\mathrm{e}]\!](m) = \textbf{true}$, then
    $(l_0, m) \rightarrow (l_1, m)$
    $(l_2, m) \rightarrow (l_1, m)$
  - if $[\![\mathrm{e}]\!](m) = \textbf{false}$, then
    $(l_0, m) \rightarrow (l_3, m)$
    $(l_2, m) \rightarrow (l_3, m)$
  - if $[\![\mathrm{e}]\!](m) = \Omega$, then
    $(l_0, m) \rightarrow \Omega$
    $(l_2, m) \rightarrow \Omega$

Outline

Traces: definitions

- a trace is a finite or infinite sequence of states

Notations

- we write $\langle s_0, \ldots, s_n \rangle$ for a **finite trace** and $\langle s_0, \ldots \rangle$ for an **infinite trace**
- $\mathbb{S}^\star$ is the **set of finite traces**
- $\mathbb{S}^\omega$ is the **set of infinite traces**
- $\mathbb{S}^\propto = \mathbb{S}^\star \cup \mathbb{S}^\omega$ is the **set of finite or infinite traces**

Operations on traces

- **length** $\mid \sigma \mid$:

$$\left\{ \begin{array}{lcl} \langle s_0, \ldots, s_n \rangle & = & n+1 \\ \langle s_0, \ldots \rangle & = & \omega \end{array} \right.$$

- **prefix** order relation:

$$\langle s_0, \ldots, s_n \rangle \prec \langle s'_0, \ldots, s'_{n'} \rangle \iff \left\{ \begin{array}{l} n \leq n' \\ \forall i \in [\![0, n]\!],\ s_i = s'_i \end{array} \right.$$

  (also defined for infinite traces)

- **concatenation** operator "$\cdot$":

$$\begin{array}{rcl} \langle s_0, \ldots, s_n \rangle \cdot \langle s'_0, \ldots, s'_{n'} \rangle & = & \langle s_0, \ldots, s_n, s'_0, \ldots, s'_{n'} \rangle \\ \langle s_0, \ldots, s_n \rangle \cdot \langle s'_0, \ldots \rangle & = & \langle s_0, \ldots, s_n, s'_0, \ldots \rangle \\ \langle s_0, \ldots, s_n, \ldots \rangle \cdot \sigma' & = & \langle s_0, \ldots, s_n, \ldots \rangle \end{array}$$

- **empty trace** ϵ, neutral element for $\cdot$

Semantics of finite traces

Goal: capture all finite executions of the program

We consider a transition system $\mathcal{S} = (\mathbb{S}, \rightarrow)$

Definition

The **finite traces semantics** $[\![\mathcal{S}]\!]^\star$ is defined by:

$$[\![\mathcal{S}]\!]^\star = \{\langle s_0, \ldots, s_n\rangle \in \mathbb{S}^\star \mid \forall i,\ s_i \rightarrow s_{i+1}\}$$

Example:

- contrived transition system $\mathcal{S} = (\{a, b, c, d\}, \{(a, b), (b, a), (b, c)\})$
- finite traces semantics:

$$\begin{array}{rclll}
[\![\mathcal{S}]\!]^\star & = & \{ & \langle a, b, \ldots, a, b, a\rangle, & \langle b, a, \ldots, a, b, a\rangle, \\
 & & & \langle a, b, \ldots, a, b, a, b\rangle, & \langle b, a, \ldots, a, b, a, b\rangle, \\
 & & & \langle a, b, \ldots, a, b, a, b, c\rangle, & \langle b, a, \ldots, a, b, a, b, c\rangle \\
 & & & \langle c\rangle \quad \} &
\end{array}$$

Interesting subsets of the finite trace semantics

We consider a transition system $\mathcal{S} = (\mathbb{S}, \rightarrow, \mathbb{S}_{\mathcal{I}}, \mathbb{S}_{\mathcal{F}})$

- the **traces from an initial state**:

$$\{\langle s_0, \ldots, s_n \rangle \in [\![\mathcal{S}]\!]^\star \mid s_0 \in \mathbb{S}_{\mathcal{I}}\}$$

- the **traces reaching a blocking state**:

$$\{\sigma \in [\![\mathcal{S}]\!]^\star \mid \forall \sigma' \in [\![\mathcal{S}]\!]^\star, \sigma \prec \sigma' \Longrightarrow \sigma = \sigma'\}$$

- the **traces ending in a final state**:

$$\{\langle s_0, \ldots, s_n \rangle \in [\![\mathcal{S}]\!]^\star \mid s_n \in \mathbb{S}_{\mathcal{F}}\}$$

Example (same transition system, with $\mathbb{S}_{\mathcal{I}} = \{a\}$ and $\mathbb{S}_{\mathcal{F}} = \{c\}$):

- traces from an initial state ending in a final state:

$$\{\langle a, b, \ldots, a, b, a, b, c \rangle\}$$

Fixpoint definition for of the semantics of finite traces

We consider a transition system $\mathcal{S} = (\mathbb{S}, \rightarrow)$.
The semantics of finite traces can be defined as a least-fixpoint:

Finite traces semantics as a fixpoint

Let $\mathcal{I} = \{\langle s \rangle \mid s \in \mathbb{S}\}$. Let $F_\star$ by the function defined by:

$$\begin{array}{rccl} F_\star : & \mathcal{P}(\mathbb{S}^\star) & \longrightarrow & \mathcal{P}(\mathbb{S}^\star) \\ & X & \longmapsto & \{\langle s_0, \ldots, s_n, s_{n+1} \rangle \mid \langle s_0, \ldots, s_n \rangle \in X \wedge s_n \rightarrow s_{n+1}\} \end{array}$$

Then, $F_\star$ is continuous and thus has a least-fixpoint greater than $\mathcal{I}$; moreover:

$$\mathbf{lfp}_{\mathcal{I}} F_\star = [\![\mathcal{S}]\!]^\star = \bigcup_{n \in \mathbb{N}} F_\star^n(\mathcal{I})$$

Fixpoint definition: proof (1), fixpoint existence

First, we prove that $F_\star$ is continuous. Let $\mathcal{X} \subseteq \mathcal{P}(\mathbb{S}^\star)$. Then:

$$\begin{array}{l} F_\star(\bigcup_{X \in \mathcal{X}} X) \\ \begin{array}{rl} = & \{\langle s_0, \ldots, s_n, s_{n+1}\rangle \mid (\langle s_0, \ldots, s_n\rangle \in \bigcup_{X \in \mathcal{X}} X) \wedge s_n \rightarrow s_{n+1}\} \\ = & \{\langle s_0, \ldots, s_n, s_{n+1}\rangle \mid (\exists X \in \mathcal{X},\ \langle s_0, \ldots, s_n\rangle \in X) \wedge s_n \rightarrow s_{n+1}\} \\ = & \{\langle s_0, \ldots, s_n, s_{n+1}\rangle \mid \exists X \in \mathcal{X},\ \langle s_0, \ldots, s_n\rangle \in X \wedge s_n \rightarrow s_{n+1}\} \\ = & \bigcup_{X \in \mathcal{X}} \{\langle s_0, \ldots, s_n, s_{n+1}\rangle \mid \langle s_0, \ldots, s_n\rangle \in X \wedge s_n \rightarrow s_{n+1}\} \\ = & \bigcup_{X \in \mathcal{X}} F_\star(X) \end{array} \end{array}$$

As $(\mathcal{P}(\mathbb{S}^\star), \subseteq)$ is a CPO, the continuity of $F_\star$ entails the existence of a least-fixpoint (Kleene theorem); moreover, it implies that:

$$\mathbf{lfp}_{\mathcal{I}} F_\star = \bigcup_{n \in \mathbb{N}} F_\star^n(\mathcal{I})$$

Fixpoint definition: proof (2), fixpoint equality

We now show that $[\![\mathcal{S}]\!]^\star$ is equal to $\mathbf{lfp}_{\mathcal{I}} F_\star$, by showing the property below, by induction over n:

$$\langle s_0, \ldots, s_n \rangle \in F_\star^n(\mathcal{I}) \iff \langle s_0, \ldots, s_n \rangle \in [\![\mathcal{S}]\!]^\star$$

- at rank 0:

$$\begin{array}{rcl} \langle s \rangle \in [\![\mathcal{S}]\!]^\star & \iff & s \in \mathbb{S} \\ & \iff & \langle s \rangle \in F_\star^0(\mathcal{I}) \end{array}$$

- at rank $n+1$, and assuming the property holds at rank n:

$$\begin{array}{l} \langle s_0, \ldots, s_n, s_{n+1} \rangle \in [\![\mathcal{S}]\!]^\star \\ \quad \iff \quad \langle s_0, \ldots, s_n \rangle \in [\![\mathcal{S}]\!]^\star \wedge s_n \to s_{n+1} \\ \quad \iff \quad \langle s_0, \ldots, s_n \rangle \in F_\star^n(\mathcal{I}) \wedge s_n \to s_{n+1} \\ \quad \iff \quad \langle s_0, \ldots, s_n, s_{n+1} \rangle \in F_\star^{n+1}(\mathcal{I}) \end{array}$$

Example

Example, with the same simple transition system $\mathcal{S} = (\mathbb{S}, \rightarrow)$:

- $\mathbb{S} = \{a, b, c, d\}$
- $\rightarrow$ is defined by $a \rightarrow b$, $b \rightarrow a$ and $b \rightarrow c$

Then, the first iterates are:

$$\begin{array}{rcl}
F_\star^0 & = & \{\langle a\rangle, \langle b\rangle, \langle c\rangle\} \\
F_\star^1 & = & \{\langle b, a\rangle, \langle a, b\rangle, \langle b, c\rangle\} \\
F_\star^2 & = & \{\langle a, b, a\rangle, \langle b, a, b\rangle, \langle a, b, c\rangle\} \\
F_\star^3 & = & \{\langle b, a, b, a\rangle, \langle a, b, a, b\rangle, \langle b, a, b, c\rangle\} \\
F_\star^4 & = & \{\langle a, b, a, b, a\rangle, \langle b, a, b, a, b\rangle, \langle a, b, a, b, c\rangle\} \\
F_\star^5 & = & \ldots
\end{array}$$

Semantics of infinite traces

So far, **we do not really isolate non-terminating behaviors**

We consider a transition system $\mathcal{S} = (\mathbb{S}, \rightarrow)$

Definition

The **infinite traces semantics** $[\![\mathcal{S}]\!]^\omega$ is defined by:

$$[\![\mathcal{S}]\!]^\omega = \{\langle s_0, \ldots \rangle \in \mathbb{S}^\omega \mid \forall i,\ s_i \rightarrow s_{i+1}\}$$

Example:

- contrived transition system defined by

$$\mathbb{S} = \{a, b, c, d\} \qquad (\rightarrow) = \{(a, b), (b, a), (b, c)\}$$

- the infinite traces semantics contains only two traces

$$[\![\mathcal{S}]\!]^\omega = \{\langle a, b, \ldots, a, b, a, b, \ldots \rangle, \langle b, a, \ldots, b, a, b, a, \ldots \rangle\}$$

Semantics of infinite traces: towards a fixpoint form

Can we also provide a fixpoint form for $[\![\mathcal{S}]\!]^\omega$?

Intuitively, $\langle s_0, s_1, \ldots \rangle \in [\![\mathcal{S}]\!]^\omega$ if and only if $\forall i,\ s_i \rightarrow s_{i+1}$.
Let F_ω be defined by:

$$\begin{array}{rccl} F_\omega : & \mathcal{P}(\mathbb{S}^\omega) & \longrightarrow & \mathcal{P}(\mathbb{S}^\omega) \\ & X & \longmapsto & \{\langle s_0, s_1, \ldots, s_n, \ldots \rangle \mid \langle s_1, \ldots, s_n, \ldots \rangle \in X \wedge s_0 \rightarrow s_1\} \end{array}$$

Then, we can show by induction that:

$$\begin{array}{rcl} \sigma \in [\![\mathcal{S}]\!]^\star & \Longleftrightarrow & \forall n \in \mathbb{N},\ \sigma \in F_\omega^n(\mathbb{S}^\omega) \\ & \Longleftrightarrow & \bigcap_{n \in \mathbb{N}} F_\omega^n(\mathbb{S}^\omega) \end{array}$$

Note: backward expression of the finite traces semantics

With a similar definition of $F_\star$, $[\![\mathcal{S}]\!]^\star = \mathbf{lfp}_{\mathcal{I}} F_\star$:

$$F_\star(X) ::= \{\langle s_0, s_1, \ldots, s_n \rangle \in \mathbb{S}^\star \mid \langle s_1, \ldots, s_n \rangle \in X \wedge s_0 \rightarrow s_1\}$$

Duality principle

- if $\subseteq$ is an order relation, so is $\supseteq$
- all properties of $\subseteq$ are inherited by $\supseteq$, modulo some correspondance

basic order	dual order
$\subseteq$	$\supseteq$
$\cup$	$\cap$
$\cap$	$\cup$
$\bot$	$\top$
$\cup$-continuous function	$\cap$-continuous function
$\cap$-continuous function	$\cup$-continuous function
least-fixpoint (**lfp**)	greatest-fixpoint (**gfp**)
greatest-fixpoint (**gfp**)	least-fixpoint (**lfp**)

Thus, we can derive dual versions of Tarski's theorem and Kleene's theorem

Fixpoint form of the semantics of infinite traces

Infinite traces semantics as a fixpoint

Let F_ω by the function defined by:

$$\begin{array}{rlcl} F_\omega : & \mathcal{P}(\mathbb{S}^\omega) & \longrightarrow & \mathcal{P}(\mathbb{S}^\omega) \\ & X & \longmapsto & \{\langle s_0, s_1, \ldots, s_n, \ldots\rangle \mid \langle s_1, \ldots, s_n, \ldots\rangle \in X \wedge s_0 \rightarrow s_1\} \end{array}$$

Then, F_ω is $\cap$-continuous and thus has a greatest-fixpoint; moreover:

$$\mathbf{gfp}_{\mathbb{S}^\omega} F_\omega = [\![\mathcal{S}]\!]^\omega$$

Proof sketch:

- the $\cap$-contiunity proof is similar as for the $\cup$-continuity of $F_\star$
- by the dual version of Kleene's theorem, $\mathbf{gfp}_{\mathbb{S}^\omega} F_\omega$ exists and is equal to $\bigcap_{n\in\mathbb{N}} F_\omega^n(\mathbb{S}^\omega)$, i.e. to $[\![\mathcal{S}]\!]^\omega$ (induction proof)

Example

Example, with the same simple transition system:

- $\mathbb{S} = \{a, b, c, d\}$
- $\rightarrow$ is defined by $a \rightarrow b$, $b \rightarrow a$ and $b \rightarrow c$

Then, the first iterates are:

$$\begin{array}{rcl}
F_\omega^0 & = & \{\langle a, s_1, s_2, \ldots\rangle, \langle b, s_1, s_2, \ldots\rangle, \langle c, s_1, s_2, \ldots\rangle\} \\
F_\omega^1 & = & \{\langle a, b, s_2, s_3, \ldots\rangle, \langle b, a, s_2, s_3, \ldots\rangle, \langle b, c, s_2, s_3, \ldots\rangle\} \\
F_\omega^2 & = & \{\langle b, a, b, s_2, s_3, \ldots\rangle, \langle a, b, a, s_2, s_3, \ldots\rangle, \langle a, b, c, s_2, s_3, \ldots\rangle\} \\
F_\omega^3 & = & \ldots
\end{array}$$

Intuition

- at iterate n, prefixes of length $n+1$ match the traces in the infinite semantics
- only $\langle a, b, \ldots, a, b, a, b, \ldots\rangle$ and $\langle b, a, \ldots, b, a, b, a, \ldots\rangle$ belong to *all* iterates

Maximal traces semantics

The maximal traces semantics simply puts together the finite traces semantics and the infinite traces semantics:

We consider a transition system $\mathcal{S} = (\mathbb{S}, \rightarrow)$

Definition

The **maximal traces semantics** $[\![\mathcal{S}]\!]^{\propto}$ is the element of $\mathbb{S}^{\propto}$ defined by:

$$[\![\mathcal{S}]\!]^{\propto} = [\![\mathcal{S}]\!]^{\star} \cup [\![\mathcal{S}]\!]^{\omega}$$

Example

Still same simple transition system:

- $\mathbb{S} = \{a, b, c, d\}$
- $\rightarrow$ is defined by $a \rightarrow b$, $b \rightarrow a$ and $b \rightarrow c$

Then:

$$\begin{array}{rcll} [\![\mathcal{S}]\!]^{\propto} & = & \{ & \langle a, b, \ldots, a, b, a\rangle, \langle b, a, \ldots, a, b, a\rangle, \\ & & & \langle a, b, \ldots, a, b, a, b\rangle, \langle b, a, \ldots, a, b, a, b\rangle, \\ & & & \langle a, b, \ldots, a, b, a, b, c\rangle, \langle b, a, \ldots, a, b, a, b, c\rangle \\ & & & \langle c\rangle \\ & & & \langle a, b, \ldots, a, b, a, b, \ldots\rangle, \langle b, a, \ldots, b, a, b, a, \ldots\rangle \quad \} \end{array}$$

Co-induction technique

Goal of the co-induction technique

- how to set up a new fixpoint definition ?
- we need to combine a least-fixpoint and a greatest-fixpoint

- **lattice**: $\mathbb{S}^{\propto}$, with the order relation $\sqsubseteq^{\propto}$ defined by

$$X \sqsubseteq^{\propto} Y \iff \left\{ \begin{array}{ll} & X \cap \mathbb{S}^{\star} \subseteq Y \cap \mathbb{S}^{\star} \\ \wedge & X \cap \mathbb{S}^{\omega} \supseteq Y \cap \mathbb{S}^{\omega} \end{array} \right.$$

- **Join**: $X \sqcup Y = ((X \cap \mathbb{S}^{\star}) \cup (Y \cap \mathbb{S}^{\star})) \cup ((X \cap \mathbb{S}^{\omega}) \cap (Y \cap \mathbb{S}^{\omega}))$
- **assumptions**: we assume $F_{\star}$ and F_{ω} defined as before
- **semantic function** $F_{\propto}$ defined by:

$$\begin{array}{rccl} F_{\propto} : & \mathcal{P}(\mathbb{S}^{\propto}) & \longrightarrow & \mathcal{P}(\mathbb{S}^{\propto}) \\ & X & \longmapsto & F_{\star}(X \cap \mathbb{S}^{\star}) \cup F_{\omega}(X \cap \mathbb{S}^{\omega}) \end{array}$$

We could also let

$$F_{\propto}(X) = \{\langle s_0, s_1, \ldots, s_n, \ldots \rangle \mid \langle s_1, \ldots, s_n, \ldots \rangle \in X \wedge s_0 \to s_1\}$$

Fixpoint form of the maximal trace semantics

We have the following properties:

- $(\mathbb{S}^{\propto}, \sqsubseteq^{\propto}, \sqcup^{\propto})$ is a complete lattice
- $F_{\propto}$ is $\sqcup^{\propto}$-continuous
- thus, it has a least-fixpoint greater than $\mathcal{I} = \{\langle s \rangle \mid s \in \mathbb{S}\}$; furthermore:

$$\left\{ \begin{array}{rcl} \mathbf{lfp}_{\mathcal{I}} F_{\propto} \cap \mathbb{S}^{\star} & = & \mathbf{lfp}_{\mathcal{I}} F_{\star} \\ \mathbf{lfp}_{\mathcal{I}} F_{\propto} \cap \mathbb{S}^{\omega} & = & \mathbf{gfp} F_{\omega} \\ \mathbf{lfp}_{\mathcal{I}} F_{\propto} & = & \mathbf{lfp}_{\mathcal{I}} F_{\star} \cup \mathbf{gfp} F_{\omega} \end{array} \right.$$

Therefore:

Fixpoint definition of $[\![\mathcal{S}]\!]^{\propto}$

$$[\![\mathcal{S}]\!]^{\propto} = \mathbf{lfp}_{\mathcal{I}} F_{\propto}$$

Finite traces as an abstraction

- we have defined three semantics; how to relate them ? can this be done in a constructive manner ?
- abstract interpretation allows to define relation between semantics !

The finite semantics discards the infinite executions

Finite traces abstraction

We define $\alpha_\star, \gamma_\star$ by:

$$\begin{array}{rcll} \alpha_\star : & \mathcal{P}(\mathbb{S}^\propto) & \longrightarrow & \mathcal{P}(\mathbb{S}^\star) \\ & X & \longmapsto & X \cap \mathbb{S}^\star \end{array} \qquad \begin{array}{rcll} \gamma_\star : & \mathcal{P}(\mathbb{S}^\star) & \longrightarrow & \mathcal{P}(\mathbb{S}^\propto) \\ & Y & \longmapsto & Y \cup \mathbb{S}^\omega \end{array}$$

Then:

- these define a Galois connection $(\mathcal{P}(\mathbb{S}^\propto), \subseteq) \underset{\alpha_\star}{\overset{\gamma_\star}{\leftrightarrows}} (\mathcal{P}(\mathbb{S}^\star), \subseteq)$
- moreover, $\alpha_\star([\![\mathcal{S}]\!]^\propto) = [\![\mathcal{S}]\!]^\star$

Proof: $\forall X \in \mathcal{P}(\mathbb{S}^\propto), Y \in \mathcal{P}(\mathbb{S}^\star),\ \alpha_\star(X) \subseteq Y \iff X \subseteq \gamma_\star(Y)$

Fixpoint transfer

We can actually make this statement more constructive

Exact fixpoint transfer

Let $(\mathbb{D}_0, \sqsubseteq_0)$ and $(\mathbb{D}_1, \sqsubseteq_1)$ be two domains, let α, γ be a pair of adjoint functions defining a Galois connection $(\mathbb{D}_0, \sqsubseteq_0) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (\mathbb{D}_1, \sqsubseteq_1)$.
Let $F_0 : \mathbb{D}_0 \to \mathbb{D}_0$, $F_1 : \mathbb{D}_1 \to \mathbb{D}_1$ and $x_0 \in \mathbb{D}_0, x_1 \in \mathbb{D}_1$, such that:

- F_0 is continuous
- F_1 is monotone
- $\alpha \circ F_0 = F_1 \circ \alpha$
- $\alpha(x_0) = x_1$

Then:

- both F_0 and F_1 have a least-fixpoint (Tarski's fixpoint theorem)
- $\alpha(\mathbf{lfp}_{x_0} F_0) = \mathbf{lfp}_{x_1} F_1$

Fixpoint transfer: proof

- $\alpha(\mathbf{lfp} F_0)$ is a least-fixpoint of F_1 since:

$$\begin{array}{rcll} F_1(\alpha(\mathbf{lfp} F_0)) & = & \alpha(F_0(\mathbf{lfp} F_0)) & \text{since } \alpha \circ F_0 = F_1 \circ \alpha \\ & = & \alpha(\mathbf{lfp} F_0) & \text{by definition of the fixpoints} \end{array}$$

- to show that $\alpha(\mathbf{lfp} F_0)$, we assume that X is another fixpoint of F_1 and we show that $\alpha(\mathbf{lfp} F_0) \sqsubseteq_1 X$, i.e., that $\mathbf{lfp} F_0 \sqsubseteq_0 \gamma(X)$;
  as $\mathbf{lfp} F_0 = \bigcup_{n \in \mathbb{N}} F_0^n(x_0)$, it amounts to proving that
  $\forall n \in \mathbb{N},\ F_0^n(x_0) \sqsubseteq_0 \gamma(X)$;
  by induction over n:
  - $F_0^0(x_0) = x_0$, thus $\alpha(F_0^0(x_0)) = x_1 \sqsubseteq_0 \gamma(X)$;
  - let us assume that $F_0^n(x_0) \sqsubseteq_0 \gamma(X)$, and let us show that $F_0^{n+1}(x_0) \sqsubseteq_0 \gamma(X)$, i.e. that $\alpha(F_0^{n+1}(x_0)) \sqsubseteq_1 X$:

$$\alpha(F_0^{n+1}(x_0)) = \alpha \circ F_0(F_0^n(x_0)) = F_1 \circ \alpha(F_0^n(x_0)) \sqsubseteq_1 F_1(X) = X$$

Application of the fixpoint transfer

All assumptions are satisfied:

- $\alpha_\star, \gamma_\star$ define a Galois connection between $(\mathcal{P}(\mathbb{S}^\propto), \subseteq)$ and $(\mathcal{P}(\mathbb{S}^\star), \subseteq)$
- $\alpha_\star(\mathcal{I}) = \mathcal{I}$
- $F_\propto$ is continuous
- $F_\star$ is continuous, hence montone
- $F_\star \circ \alpha_\star = \alpha_\star \circ F_\propto$

This gives another proof of the abstraction relation:

Abstraction relation

$$\alpha_\star(\llbracket \mathcal{S} \rrbracket^\propto) = \alpha_\star(\mathbf{lfp}_\mathcal{I} F_\propto) = \mathbf{lfp}_\mathcal{I} F_\omega = \llbracket \mathcal{S} \rrbracket^\omega$$

The constructive proof ties very closely the iterates
i.e., the way the semantics are computed

Infinite traces as an abstraction

The same reasoning can be applied to the infinite traces semantics:

Infinite traces abstraction

We define $\alpha_\omega, \gamma_\omega$ by:

$$\begin{array}{rlcl} \alpha_\omega : & \mathcal{P}(\mathbb{S}^\propto) & \longrightarrow & \mathcal{P}(\mathbb{S}^\omega) \\ & X & \longmapsto & X \cap \mathbb{S}^\omega \end{array} \qquad \begin{array}{rlcl} \gamma_\omega : & \mathcal{P}(\mathbb{S}^\omega) & \longrightarrow & \mathcal{P}(\mathbb{S}^\propto) \\ & Y & \longmapsto & Y \cup \mathbb{S}^\star \end{array}$$

Then:

- these define a Galois connection $(\mathcal{P}(\mathbb{S}^\propto), \subseteq) \underset{\alpha_\omega}{\overset{\gamma_\omega}{\leftrightarrows}} (\mathcal{P}(\mathbb{S}^\omega), \subseteq)$
- moreover, $\alpha_\omega(\llbracket \mathcal{S} \rrbracket^\propto) = \llbracket \mathcal{S} \rrbracket^\omega$
- the fixpoint transfer also holds: $\alpha_\omega \circ F_\propto = F_\omega \circ \alpha_\omega$, $F_\propto$ is continuous and F_ω is continuous, hence monotone

Towards a hierarchy of semantics

So far, we have:

- three forms of operational semantics
- two abstraction relations

$$\begin{tikzcd}
\llbracket \mathcal{S} \rrbracket^{\star} & & \llbracket \mathcal{S} \rrbracket^{\omega} \\
& \llbracket \mathcal{S} \rrbracket^{\propto} \arrow[ul, "\alpha_{\star}"] \arrow[ur, "\alpha_{\omega}"'] &
\end{tikzcd}$$

We can actually build lattices of semantics:
"greater" means "more abstract than"
See **[C'97]**

Outline

Denotational semantics: definition

- operational (trace) semantics is very precise:
  it records *all the history* of *all* executions of the system
- this may be too precise in many cases, e.g., when the history is not relevant

- we first focus on the *finite* behaviors
- we consider transition system $\mathcal{S} = (\mathbb{S}, \rightarrow)$

Finite denotational semantics [ST'71]

The denotational semantics $[\![\mathcal{S}]\!]_\partial$ is the function

$$\begin{array}{rccl} [\![\mathcal{S}]\!]_\partial : & \mathbb{S} & \longrightarrow & \mathcal{P}(\mathbb{S}) \\ & s & \longmapsto & \{s' \in \mathbb{S} \mid s \rightarrow^\star s'\} \end{array}$$

Semantic domain: $\mathbb{D}_\partial = \mathbb{S} \rightarrow \mathcal{P}(\mathbb{S})$, with the pointwise extension of $\subseteq$

Example

Another contrived transition system $\mathcal{S} = (\mathbb{S}, \rightarrow)$ defined by:

- $\mathbb{S} = \{a, b, c, d\}$
- $a \rightarrow b$, $c \rightarrow c$, $c \rightarrow d$

Then:

$$\begin{array}{rrcl} [\![\mathcal{S}]\!]_\partial : & a & \longmapsto & \{a, b\} \\ & b & \longmapsto & \{b\} \\ & c & \longmapsto & \{c, d\} \\ & d & \longmapsto & \{d\} \end{array}$$

Observations

- much more compact than the operational semantics
- the execution history is effectively left behind
- the semantics makes no difference between one step and a sequence of any number of steps (as observed from state c)

Denotational abstraction

We can obviously derive $[\![\mathcal{S}]\!]_\partial$ from $[\![\mathcal{S}]\!]^\star$

Definition of the denotational abstraction

Let $\alpha_\partial, \gamma_\partial$ be the functions defined by

$$\begin{array}{rlcl}
\alpha_\partial : & \mathcal{P}(\mathbb{S}^\star) & \longrightarrow & \mathbb{D}_\partial \\
 & X & \longmapsto & \lambda s_0 \cdot \{ s_n \in \mathbb{S} \mid \exists \sigma = \langle s_0, \ldots, s_n \rangle \in X \} \\
\gamma_\partial : & \mathbb{D}_\partial & \longrightarrow & \mathcal{P}(\mathbb{S}^\star) \\
 & \Psi & \longmapsto & \{ \langle s_0, \ldots, s_n \rangle \mid s_n \in \Psi(s_0) \}
\end{array}$$

These functions form a Galois connection

$$(\mathcal{P}(\mathbb{S}^\star), \subseteq) \underset{\alpha_\partial}{\overset{\gamma_\partial}{\leftrightarrows}} (\mathbb{D}_\partial, \dot{\subseteq})$$

Proof: straightforward computation

Denotational semantics as an abstraction

Abstraction relation

Following the definitions of $[\![.]\!]_\partial, [\![.]\!]^\star$ and α_∂:

$$[\![\mathcal{S}]\!]_\partial = \alpha_\partial([\![\mathcal{S}]\!]^\star)$$

Other similar kinds of abstractions:

- Relational semantics
- Pre-conditions (e.g., weakest pre-conditions semantics)

See **[C'97]**

Fixpoint definition

Can $[\![\mathcal{S}]\!]_\partial$ be constructively defined ? Yes, fixpoint transfer!

With the notations used so far for $\mathcal{S}$, its semantics and semantic functions, and with $X \in \mathcal{P}(\mathbb{S}^\star)$,

$$\begin{array}{rcl}\alpha_\partial \circ F_\star(X) & = & \lambda(s \in \mathbb{S}) \cdot \{s' \in \mathbb{S} \mid \exists \langle s, \ldots, s' \rangle \in F_\star(X)\} \\ & = & \lambda(s_0 \in \mathbb{S}) \cdot \{s_{n+1} \in \mathbb{S} \mid \exists \langle s_0, \ldots, s_n \rangle \in X \wedge s_n \rightarrow s_{n+1}\} \\ & = & \lambda(s_0 \in \mathbb{S}) \cdot \{s_{n+1} \in \mathbb{S} \mid \exists s_n \in \alpha_\partial(X),\ s_n \rightarrow s_{n+1}\} \\ & = & F_\partial \circ \alpha_\partial(X)\end{array}$$

where:

$$\begin{array}{rccl} F_\partial : & \mathbb{D}_\partial & \longrightarrow & \mathbb{D}_\partial \\ & \Psi & \longmapsto & \lambda(s \in \mathbb{S}) \cdot \{s' \in \mathbb{S} \mid s \rightarrow s'\} \end{array}$$

Fixpoint form of the denotational semantics

We remark that:

- $(\mathcal{P}(\mathbb{S}^\star), \subseteq)$ and $(\mathbb{D}_\partial, \dot{\subseteq})$ are complete lattices
- $\alpha_\partial, \gamma_\partial$ define a Galois connection betwee these lattices
- $F_\star$ is continuous
- F_∂ is continuous, hence monotone
- $\alpha_\partial \circ F_\star = F_\partial \circ \alpha_\partial$
- $\alpha_\partial(\mathcal{I}) = \alpha_\partial(\{\langle s \rangle \mid s \in \mathbb{S}\}) = \lambda(s \in \mathbb{S}) \cdot \{s\}$
  (we write $\mathbb{I}$ for the identity function)

Therefore, by fixpoint transfer:

Denotational semantics as a fixpoint

$$[\![\mathcal{S}]\!]_\partial = \alpha_\partial([\![\mathcal{S}]\!]^\star) = \alpha_\partial(\mathbf{lfp}_{\mathcal{I}} F_\star \mathcal{S}) = \mathbf{lfp}_{\mathbb{I}} F_\partial$$

Applications

The choice of the concrete semantics is tied to the properties to analyze

Denotational semantics is a good basis for:

- modular analyses, based on the abstraction of input-output relations
- typing analyses: types are an abstraction of the denotational semantics
- whenever intermediate states are not relevant, it is helpful to abstract them

Reachable states abstraction

We consider a transition system $\mathcal{S} = (\mathbb{S}, \rightarrow, \mathbb{S}_{\mathcal{I}})$

Definition

We let $\alpha_{\mathcal{R}}$ be defined by:

$$\begin{array}{rlcl} \alpha_{\mathcal{R}}: & \mathbb{D}_{\partial} & \longrightarrow & \mathcal{P}(\mathbb{S}) \\ & \Phi & \longmapsto & \Phi(\mathbb{S}_{\mathcal{I}}) \end{array}$$

$$\begin{array}{rlcl} \gamma_{\mathcal{R}}: & \mathcal{P}(\mathbb{S}) & \longrightarrow & \mathbb{D}_{\partial} \\ & X & \longmapsto & \lambda(s \in \mathbb{S}) \cdot \begin{cases} X & \text{if } s \in \mathbb{S}_{\mathcal{I}} \\ \mathbb{S} & \text{otherwise} \end{cases} \end{array}$$

Then, we have a Galois connection $(\mathbb{D}_{\partial}, \dot{\subseteq}) \underset{\alpha_{\mathcal{R}}}{\overset{\gamma_{\mathcal{R}}}{\leftrightarrows}} (\mathcal{P}(\mathbb{S}), \subseteq)$.
We let:

$$[\![\mathcal{S}]\!]_{\mathcal{R}} = \alpha_{\mathcal{R}}([\![\mathcal{S}]\!]_{\partial}) = \{s_n \in \mathbb{S} \mid \exists s_0 \in \mathbb{S}_{\mathcal{I}}, \langle s_0, \ldots, s_n \rangle\}$$

Example

Example, with the simple transition system $\mathcal{S}$ defined by:

- $\mathbb{S} = \{a, b, c, d\}$
- $\rightarrow$ is defined by $a \rightarrow b$, $b \rightarrow a$ and $b \rightarrow c$
- $\mathbb{S}_{\mathcal{I}} = \{a\}$

Then, the **reachable states** are:

$$[\![\mathcal{S}]\!]_{\mathcal{R}} = \{a, b, c\}$$

Composition of Galois connections

Composition property

Let $(\mathbb{D}_0, \sqsubseteq_0)$, $(\mathbb{D}_1, \sqsubseteq_1)$ and $(\mathbb{D}_2, \sqsubseteq_2)$ be three abstract domains, and let us assume the Galois connections below are defined:

$$(\mathbb{D}_0, \sqsubseteq_0) \underset{\alpha_{01}}{\overset{\gamma_{10}}{\leftrightarrows}} (\mathbb{D}_1, \sqsubseteq_1) \qquad (\mathbb{D}_1, \sqsubseteq_1) \underset{\alpha_{12}}{\overset{\gamma_{21}}{\leftrightarrows}} (\mathbb{D}_2, \sqsubseteq_2)$$

Then, we have a third Galois connection

$$(\mathbb{D}_0, \sqsubseteq_0) \underset{\alpha_{12}\circ\alpha_{01}}{\overset{\gamma_{10}\circ\gamma_{21}}{\leftrightarrows}} (\mathbb{D}_2, \sqsubseteq_2)$$

Proof: if $x_0 \in \mathbb{D}_0, x_2 \in \mathbb{D}_2$, then
$\alpha_{12} \circ \alpha_{01}(x_0) \sqsubseteq_2 x_2 \iff \alpha_{01}(x_0) \sqsubseteq_1 \gamma_{21}(x_2) \iff x_0 \sqsubseteq_0 \gamma_{10} \circ \gamma_{21}(x_2)$

Application

$[\![\mathcal{S}]\!]_{\mathcal{R}}$ is also an abstraction of $[\![\mathcal{S}]\!]^\star$

Fixpoint form of the reachable states abstraction

Fixpoint definition

We let $F_{\mathcal{R}}$ be defined by:

$$\begin{array}{rccl} F_{\mathcal{R}} : & \mathcal{P}(\mathbb{S}) & \longrightarrow & \mathcal{P}(\mathbb{S}) \\ & X & \longmapsto & \{ s \in \mathbb{S} \mid \exists s' \in X,\ s' \rightarrow s \} \end{array}$$

Then, $F_{\mathcal{R}}$ is continuous, has a least fixpoint and

$$[\![\mathcal{S}]\!]_{\mathcal{R}} = \mathbf{lfp}_{\mathbb{S}_{\mathcal{I}}} F_{\mathcal{R}}$$

Proof: exercise

Infinite denotational semantics

- (finite) denotational semantics maps inputs to outputs
- infinite operational semantics collects infinite executions
  - infinite traces have no output state...
  - ... so, at the "denotational level": begins of infinite traces

Can we propose an infinite counterpart to the denotational semantics ?

Definition

We define $\alpha_{\partial\omega}, \gamma_{\partial\omega}$ by:

$$\begin{array}{rlcl} \alpha_{\partial\omega}: & \mathcal{P}(\mathbb{S}^\omega) & \longrightarrow & \mathcal{P}(\mathbb{S}) \\ & X & \longmapsto & \{s \in \mathbb{S} \mid \exists \langle s, s_1, s_2, \ldots \rangle \in X\} \\ \gamma_{\partial\omega}: & \mathcal{P}(\mathbb{S}) & \longrightarrow & \mathcal{P}(\mathbb{S}^\omega) \\ & X & \longmapsto & X^\omega \end{array}$$

These form a Galois connection $(\mathcal{P}(\mathbb{S}^\omega), \subseteq) \underset{\alpha_{\partial\omega}}{\overset{\gamma_{\partial\omega}}{\leftrightarrows}} (\mathcal{P}(\mathbb{S}), \subseteq)$

Then $[\![\mathcal{S}]\!]_{\partial\omega} = \alpha_{\partial\omega}([\![\mathcal{S}]\!]^\omega)$

Denotational semantics for both finite and infinite behaviors

Many other kinds of semantics can be defined:

- denotational semantics for both finite and infinite behaviors
- same for other forms of semantics

Lattice of abstractions

- abstraction is a **pre-order relation** among semantics
- these semantics can be compared by abstraction
- they form a **lattice** of semantics **[C'97]**

Outline

Semantic properties of programs

Second part of the lecture:

- how to formalize properties that we want to verify about programs ?
- how does this choice impact the choice of a base semantics, of abstractions, and of analysis ?

Examples of semantics properties

- is the program exempt of **runtime errors** ?
- does the program compute the **expected result** ?
- does the program **terminate** ?
- does the program **terminate in less than t seconds** ?
- do program execution leak any secrete information ?

State properties

As usual, we consider $\mathcal{S} = (\mathbb{S}, \rightarrow, \mathbb{S}_{\mathcal{I}})$

First approach: properties as sets of states

- a property $\mathcal{P}$ is a set of states $\mathcal{P} \subseteq \mathbb{S}$
- $\mathcal{P}$ is satisfied if and only if all reachable states belong to $\mathcal{P}$, i.e., $[\![\mathcal{S}]\!]_{\mathcal{R}} \subseteq \mathcal{P}$

Examples:

- **absence of runtime errors**:

$$\mathcal{P} = \mathbb{S} \setminus \{\Omega\} \quad \text{where } \Omega \text{ is the error state}$$

- **non termination** (e.g., operating system):

$$\mathcal{P} = \{s \in \mathbb{S} \mid \exists s' \in \mathbb{S}, s \rightarrow s'\}$$

(set of non blocking states)

Verification of state properties

Invariance proof method, soundness and completeness

Considering state property $\mathcal{P}$, $\mathcal{S}$ satisfies $\mathcal{P}$ if and only if there exists a set of states $\mathbb{I}$ called **invariant** such that

- $\mathbb{S}_{\mathcal{I}} \subseteq \mathbb{I}$
- $\forall s \in \mathbb{I},\ \forall s' \in \mathbb{S},\ (s \rightarrow s') \Longrightarrow s' \in \mathbb{I}$
- $\mathbb{I} \subseteq \mathcal{P}$

Proof:

- soundness: if there exists such a $\mathbb{I}$, we can show by induction that $[\![\mathcal{S}]\!]_{\mathcal{R}} \subseteq \mathbb{I}$, hence $[\![\mathcal{S}]\!]_{\mathcal{R}} \subseteq \mathcal{P}$
- completness: if $\mathcal{P}$ holds, $\mathbb{I} = \mathbb{S} \setminus \mathcal{P}$ works

Trace properties

Second approach: properties as sets of traces

- a property $\mathcal{T}$ is a set of traces: $\mathcal{T} \subseteq \mathbb{S}^{\propto}$
- $\mathcal{T}$ is satisfied if and only if all traces belong to $\mathcal{T}$, i.e., $[\![\mathcal{S}]\!]^{\propto} \subseteq \mathcal{T}$

Examples:

- obviously, **state properties** are trace properties
- **functional properties**
  e.g., "program P takes one integer input x and returns its absolute value"
- **termination**: $\mathcal{T} = \mathbb{S}^{\star}$ (i.e., the system should have no infinite execution)

There is a wide range of trace properties, how to classify them ?
$\Rightarrow$ we are going to see two important families of properties

A monotony property

Remark

If:

- $\mathcal{T}$ is a trace property
- system $\mathcal{S}_0$ satisfies $\mathcal{T}$
- system $\mathcal{S}_1$ has **fewer** behaviors than $\mathcal{S}_0$ (i.e., $[\![\mathcal{S}_1]\!]^\propto \subseteq [\![\mathcal{S}_0]\!]^\propto$)

Then $\mathcal{S}_1$ also **satisfies** $\mathcal{T}$

Proof: trivial composition of inclusions

Safety properties

Intuition:

- a safety property is a property which specifies that some (bad) thing **will never occur**
- it is possible to **disprove** a safety property with a single, finite trace

- **absence of runtime errors** is a safety property ("bad thing": error)
- **state properties** is a safety property ("bad thing": reaching $\mathbb{S} \setminus \mathcal{P}$)
- **non termination** is a safety property ("bad thing": reaching a blocking state)
- **"not reaching state b after visiting state a"** is a safety property (and **not** a trace property)
- **termination** is **not** a safety property

We intend to provide a **formal** definition of safety

Some operators on sets of traces: prefix closure

Prefix closure

We write $\sigma_{\lceil i}$ for the prefix of length i of trace σ:

$$\langle s_0, \ldots, s_n \rangle_{\lceil i+1} = \begin{cases} \langle s_0, \ldots, s_i \rangle & \text{if } i \leq n \\ \langle s_0, \ldots, s_n \rangle & \text{otherwise} \end{cases}$$

The prefix closure operator is defined by:

$$\begin{array}{rlcl} \mathbf{PCl}: & \mathcal{P}(\mathbb{S}^{\propto}) & \longrightarrow & \mathcal{P}(\mathbb{S}^{\star}) \\ & X & \longmapsto & \{\sigma_{\lceil i} \mid \sigma \in X, i \in \mathbb{N}\} \end{array}$$

Properties:

- **PCl** is monotone
- **PCl** is idempotent, i.e., $\mathbf{PCl} \circ \mathbf{PCl}(X) = \mathbf{PCl}(X)$

Some operators on sets of traces: limit

Limit

The limit operator is defined by:

$$\begin{array}{rccl} \mathbf{Lim}: & \mathcal{P}(\mathbb{S}^{\propto}) & \longrightarrow & \mathcal{P}(\mathbb{S}^{\propto}) \\ & X & \longmapsto & X \cup \{\sigma \in \mathbb{S}^{\propto} \mid \forall i \in \mathbb{N},\ \sigma_{\lceil i} \in X\} \end{array}$$

Properties:

- **Lim** is extensive, monotone and idempotent (i.e., it defines an upper closure operator over $\mathcal{P}(\mathbb{S}^{\propto})$)

Safety: formal definition

An upper closure operator

Operator **Safe** is defined by $\mathbf{Safe} = \mathbf{Lim} \circ \mathbf{PCl}$.
It is an upper closure operator over $\mathcal{P}(\mathbb{S}^{\propto})$

Proof:

- it is monotone and idempotent as **Lim** and **PCl** are
- it is extensive; indeed if $X \subseteq \mathbb{S}^{\propto}$ and $\sigma \in X$, we can show that $\sigma \in \mathbf{Safe}(X)$:
    - if σ is a finite trace, it is one of its prefixes, so $\sigma \in \mathbf{PCl}(X) \subseteq \mathbf{Lim}(\mathbf{PCl}(X))$
    - if σ is an infinite trace, all its prefixes belong to $\mathbf{PCl}(X)$, so $\sigma \in \mathbf{Lim}(\mathbf{PCl}(X))$

Safety: definition [AS'87]

A trace property $\mathcal{T}$ is a **safety** property if and only if $\mathbf{Safe}(\mathcal{T}) = \mathcal{T}$

Example

Let us consider **state property** $\mathcal{P}$.
It is equivalent to **trace property** $\mathcal{T} = \mathcal{P}^{\propto}$:

$$\begin{array}{rcl}
\mathbf{Safe}(\mathcal{T}) & = & \mathbf{Lim}(\mathbf{PCl}(\mathcal{P}^{\propto})) \\
& = & \mathbf{Lim}(\mathcal{P}^{\star}) \\
& = & \mathcal{P}^{\star} \cup \mathcal{P}^{\omega} \\
& = & \mathcal{P}^{\propto} \\
& = & \mathcal{T}
\end{array}$$

Therefore $\mathcal{T}$ is indeed a safety property

Example

We assume that:

- $\mathbb{S} = \{a, b\}$
- $\mathcal{T}$ states that a should not be visited after state b is visited; elements of $\mathcal{T}$ are of the general form

$$\langle a, a, a, \ldots, a, b, b, b, b, \ldots\rangle \text{ or } \langle a, a, a, \ldots, a, a, \ldots\rangle$$

Then:

- **PCl**$(\mathcal{T})$ elements are all finite traces which are of the above form (i.e., made of n occurrences of a followed by m occurrences of b, where n, m are positive integers)
- **Lim**(**PCl**$(\mathcal{T})$) adds to this set the trace made made of infinitely many occurrences of a and the infinite traces made of n occurrences of a followed by infinitely many occurrneces of b
- thus, **Safe**$(\mathcal{T})$ = **Lim**(**PCl**$(\mathcal{T})$) $= \mathcal{T}$

Therefore $\mathcal{T}$ is a safety property

A characterization

Property

A safety properties $\mathcal{T}$ can be disproved **by looking only at finite behaviors**:

$$\forall \sigma \in \mathbb{S}^{\propto}, (\sigma \notin \mathcal{T}) \Longleftrightarrow (\exists i,\ \sigma_{\lceil i} \notin \mathcal{T})$$

Proof by invariance

We consider transition system $\mathcal{S} = (\mathbb{S}, \rightarrow, \mathbb{S}_{\mathcal{I}}, \mathbb{S}_{\mathcal{F}})$, and safety property $\mathcal{T}$

Principle of invariance proofs

Let $\mathbb{I}$ be a set of finite traces; it is said to be an **invariant** if and only if:

- $\forall s \in \mathbb{S}_{\mathcal{I}},\ \langle s \rangle \in \mathbb{I}$
- $F_{\star}(\mathbb{I}) \subseteq \mathbb{I}$

It is stronger than $\mathcal{T}$ if and only if $\mathbb{I} \subseteq \mathcal{T}$

Other lectures of this course:

how to calculate the invariant by abstract interpretation ?

Soundness and completeness

The invariance proof method is sound and complete for safety properties: $[\![\mathcal{S}]\!]^{\propto}$ satisfies $\mathcal{T}$ if and only if we can find an invariant for $\mathcal{S}$, which is stronger than $\mathcal{T}$

Proof

- **Soundness**:
  we assume that $\mathbb{I}$ is an invariant of $\mathcal{S}$ and that it is stronger than $\mathcal{T}$, and we show that $\mathcal{S}$ satisfies $\mathcal{T}$;
  - by induction over n, we can prove that $F_\star^n(\mathcal{I}) \subseteq F_\star^n(\mathbb{I}) \subseteq \mathbb{I}$
  - therefore $[\![\mathcal{S}]\!]^\star \subseteq \mathbb{I}$
  - we remark that $[\![\mathcal{S}]\!]^\propto = \mathbf{Safe}([\![\mathcal{S}]\!]^\star)$
  - thus, $[\![\mathcal{S}]\!]^\propto = \mathbf{Safe}(\mathbb{I}) \subseteq \mathbf{Safe}(\mathcal{T})$ since **Safe** is monotone
  - $\mathcal{T}$ is a safety property so $\mathbf{Safe}(\mathcal{T}) = \mathcal{T}$
  - we conclude $[\![\mathcal{S}]\!]^\propto \subseteq \mathcal{T}$, i.e., $\mathcal{S}$ satisfies property $\mathcal{T}$
- **Completeness**: we assume that $[\![\mathcal{S}]\!]^\propto$ satisfies $\mathcal{T}$
  then, $\mathbb{I} = [\![\mathcal{S}]\!]^\propto$ is an invariant of $\mathcal{S}$ by definition of $[\![.]\!]^\propto$, and it is stronger than $\mathcal{T}$

Liveness properties

Intuition:

- a liveness property is a property which specifies that some (good) thing **will eventually occur**
- it is not possible to disprove a liveness property by looking at finite traces only
  i.e., it requires reasoning about infinite behaviors

- **termination** is a liveness property ("good thing": reaching a blocking state)
- **"state a will eventually be reached by all execution"** is a liveness property
- **absence of runtime errors** is *not* a liveness property

Liveness: formal definition

Formal definition [AS'87]

Operator **Live** is defined by $\mathbf{Live}(\mathcal{T}) = \mathcal{T} \cup (\mathbb{S}^\omega \setminus \mathbf{Safe}(\mathcal{T}))$. Given property $\mathcal{T}$, the following three statements are equivalent:

(*i*) $\mathbf{Live}(\mathcal{T}) = \mathcal{T}$

(*ii*) $\mathbf{PCl}(\mathcal{T}) = \mathbb{S}^\star$

(*iii*) $\mathbf{Lim} \circ \mathbf{PCl}(\mathcal{T}) = \mathbb{S}^\omega$

When they are satisfied, $\mathcal{T}$ is said to be a **liveness property**

Example: **termination**

- the property is $\mathcal{T} = \mathbb{S}^\star$
  (i.e., there should be no infinite execution)
- clearly, it satisfies (*ii*): $\mathbf{PCl}(\mathcal{T}) = \mathbb{S}^\star$
  thus termination indeed satisfies this definition

Formal definition

Proof of equivalence:

- *(i)* **implies** *(iii)*:
  we assume that $\mathbf{Live}(\mathcal{T}) = \mathcal{T}$, i.e., $\mathcal{T} \cup (\mathbb{S}^\omega \setminus \mathbf{Safe}(\mathcal{T})) = \mathcal{T}$
  therefore, $\mathbb{S}^\omega \setminus \mathbf{Safe}(\mathcal{T}) \subseteq \mathcal{T}$;
  let $\sigma \in \mathbb{S}^\star$, and let us show that $\sigma \in \mathbf{PCl}(\mathcal{T})$:
  let $\sigma' \in \mathbb{S}^\omega$; then $\sigma \cdot \sigma' \in \mathbb{S}^\omega$, thus:
    - either $\sigma \cdot \sigma' \in \mathbf{Safe}(\mathcal{T}) = \mathbf{Lim}(\mathbf{PCl}(\mathcal{T}))$, so all its prefixes are in $\mathbf{PCl}(\mathcal{T})$ and $\sigma \in \mathbf{PCl}(\mathcal{T})$
    - or $\sigma \cdot \sigma' \in \mathcal{T}$, which means that $\sigma \in \mathbf{PCl}(\mathcal{T})$
- *(ii)* **implies** *(iii)*:
  if $\mathbf{PCl}(\mathcal{T}) = \mathbb{S}^\star$, then $\mathbf{Lim} \circ \mathbf{PCl}(\mathcal{T}) = \mathbb{S}^\omega$
- *(iii)* **implies** *(i)*:
  if $\mathbf{Lim} \circ \mathbf{PCl}(\mathcal{T}) = \mathbb{S}^\omega$, then
  $\mathbf{Live}(\mathcal{T}) = \mathcal{T} \cup (\mathbb{S}^\omega \setminus (\mathcal{T} \cup \mathbf{Lim} \circ \mathbf{PCl}(\mathcal{T}))) = \mathcal{T} \cup (\mathbb{S}^\omega \setminus \mathbb{S}^\omega) = \mathcal{T}$

Proof by variance

We consider transition system $\mathcal{S} = (\mathbb{S}, \rightarrow, \mathbb{S}_{\mathcal{I}})$, and safety property $\mathcal{T}$

Principle of variance proofs

Let $(\mathbb{I}_n)_{n \in \mathbb{N}}$, $\mathbb{I}_\omega$ be elements of $\mathbb{S}^\propto$; these are said to form a variance proof of $\mathcal{T}$ if and only if:

- $\mathbb{S}^\propto \subseteq \mathbb{I}_0$
- for all $k \in \{1, 2, \ldots, \omega\}$, $\forall s \in \mathbb{S}$, $\langle s \rangle \in \mathbb{I}_k$
- for all $k \in \{1, 2, \ldots, \omega\}$, there exists $l < k$ such that $F_\omega(\mathbb{I}_l) \subseteq \mathbb{I}_k$
- $\mathbb{I}_\omega \subseteq \mathcal{T}$

Soundness and completeness

The variance proof method is sound and complete for liveness properties: $[\![\mathcal{S}]\!]^\propto$ satisfies $\mathcal{T}$ if and only if we can find $(\mathbb{I}_n)_{n \in \mathbb{N}}$ and $\mathbb{I}_\omega$ satisfying the above conditions

Decomposition theorem

Theorem

Let $\mathcal{T} \subseteq \mathbb{S}^{\propto}$; it can be decomposed into the **conjunction** of **safety property** $\mathbf{Safe}(\mathcal{T})$ and **liveness property** $\mathbf{Live}(\mathcal{T})$:

$$\mathcal{T} = \mathbf{Safe}(\mathcal{T}) \cap \mathbf{Live}(\mathcal{T})$$

Proof:

$$\begin{array}{rcl}\mathbf{Safe}(\mathcal{T}) \cap \mathbf{Live}(\mathcal{T}) & = & (\mathbb{S}^{\propto} \setminus \mathbf{Safe}(\mathcal{T}) \cup \mathcal{T}) \cap \mathbf{Safe}(\mathcal{T}) \\ & = & (\mathbb{S}^{\propto} \setminus \mathbf{Safe}(\mathcal{T}) \cap \mathbf{Safe}(\mathcal{T})) \cup (\mathcal{T} \cap \mathbf{Safe}(\mathcal{T})) \\ & = & \mathcal{T}\end{array}$$

- Application: any trace property can be **decomposed**
- **Proofs** can also be decomposed (Floyd)
  prove $\mathbf{Safe}(\mathcal{T})$ by invariance and prove $\mathbf{Live}(\mathcal{T})$ by variance

Interference, non interference

Assumptions:

- states are of the form $(l, m) \in \mathbb{L} \times \mathbb{M}$
- memory states are of the form $\mathbb{X} \to \mathbb{V}$

Let $l, l' \in \mathbb{L}$ and $x, x' \in \mathbb{X}$

Definition

We say x' at l' depends on x at l if and only if observing the values of x' at point l' allows to gain information about the value x took at point l, before reaching point l'

Applications:

- **security**: can sensitive information x be leaked to a non trusted agent who gets to see x'
- **dependences**: what part of the program should be considered to understand the value of x' (this question arises in program understanding techniques, slicing...)

Interference, non interference

We seek for a more rigorous definition of property "x' at point l' depends on x at point l":

Formal definition: interference

We derive function $\Phi_{l,l'}$ from the denotational semantics of the system:

$$\begin{array}{rccl} \Phi_{l,l'}(\psi) : & \mathbb{M} & \longrightarrow & \mathcal{P}(\mathbb{M}) \\ & m & \longmapsto & \{m \in \mathbb{M} \mid (l', m') \in \psi(l, m)\} \end{array}$$

We write $(l', x') \rightsquigarrow (l, x)$ if and only if there exist two memory states m_0, m_1 such that:

- for all variable $y \neq x$, $m_0(y) = m_1(y)$
  (i.e., m_0 and m_1 may differ only on x)
- $\Phi_{l,l'}([\![\mathcal{S}]\!]_\partial)(m_0)(x') \neq \Phi_{l,l'}([\![\mathcal{S}]\!]_\partial)(m_1)(x')$
  (i.e., output values of x' are different)

Interference, non interference

We seek for a more rigorous definition of property "x' at point l' does not depend on x at point l":

Formal definition: non interference

We derive function $\Phi_{l,l'}$ from the denotational semantics of the system:

$$\begin{array}{rrcl} \Phi_{l,l'}(\psi): & \mathbb{M} & \longrightarrow & \mathcal{P}(\mathbb{M}) \\ & m & \longmapsto & \{m \in \mathbb{M} \mid (l', m') \in \psi(l, m)\} \end{array}$$

We write $(l', x') \not\leadsto (l, x)$ if and only if, for all pair of memory states m_0, m_1 such that for all variable $y \neq x$, $m_0(y) = m_1(y)$ (i.e., m_0 and m_1 may differ only on x), then $\Phi_{l,l'}([\![\mathcal{S}]\!]_\partial)(m_0)(x') = \Phi_{l,l'}([\![\mathcal{S}]\!]_\partial)(m_1)(x')$ (i.e., output values observed for x' are similar).

Non interference is not a trace property

- we assume $\mathbb{V} = \{0, 1\}$ and $\mathbb{X} = \{x, y\}$
- we assume $\mathbb{L} = \{l, l'\}$ and consider systems such that all transitions are of the form $(l, m) \rightarrow (l', m')$
  (i.e., the systems are isomorphic to $\Phi_{l,l'}$)
- we write (v_x, v_y) for the $m \in \mathbb{M}$ such that $m(x) = v_x$ and $m(y) = v_y$

$$\Phi^0_{l,l'}(\mathcal{S}_0) : \begin{array}{rcl} (0,0) & \longmapsto & \mathbb{M} \\ (0,1) & \longmapsto & \mathbb{M} \\ (1,0) & \longmapsto & \mathbb{M} \\ (1,1) & \longmapsto & \mathbb{M} \end{array} \qquad \Phi^0_{l,l'}(\mathcal{S}_1) : \begin{array}{rcl} (0,0) & \longmapsto & \mathbb{M} \\ (0,1) & \longmapsto & \mathbb{M} \\ (1,0) & \longmapsto & \{(1,1)\} \\ (1,1) & \longmapsto & \{(1,1)\} \end{array}$$

- $\mathcal{S}_0$ **has the non-interference property**, but $\mathcal{S}_0$ **does not**
- $\mathcal{S}_1$ has fewer behavior than $\mathcal{S}_0$
- **thus, the non interference property is not a trace property**

Interference is not a trace property

$$\begin{array}{rlcl} \Phi^0_{l,l'}(\mathcal{S}_0): & (0,0) & \longmapsto & \mathbb{M} \\ & (0,1) & \longmapsto & \mathbb{M} \\ & (1,0) & \longmapsto & \{(1,1)\} \\ & (1,1) & \longmapsto & \{(1,1)\} \end{array}$$

$$\begin{array}{rlcl} \Phi^0_{l,l'}(\mathcal{S}_1): & (0,0) & \longmapsto & \{(1,1)\} \\ & (0,1) & \longmapsto & \{(1,1)\} \\ & (1,0) & \longmapsto & \{(1,1)\} \\ & (1,1) & \longmapsto & \{(1,1)\} \end{array}$$

- $\mathcal{S}_0$ **has the interference property**, but $\mathcal{S}_0$ **does not**
- $\mathcal{S}_1$ has fewer behavior than $\mathcal{S}_0$
- **thus, the interference property is not a trace property**

Interference and non-interference not trace properties

- interference and non interference **cannot be observed on a single trace**
- to exhibit interference or non-interference, we need to consider at least **two traces**

  it is not possible to say a trace satisfies the property independently from the other executions of the program
- **interference** and **non interference** **are not** trace properties

Hyperproperties

Definition [CS'08]

A **hyperproperty** is a set of sets traces, i.e. an element of

$$\mathcal{P}(\mathcal{P}(\mathbb{S}^{\propto}))$$

Transition system satisfies hyperproperty $\mathcal{H}$ if and only if $[\![\mathcal{S}]\!]^{\star} \in \mathcal{H}$

- trace property $\mathcal{T}$ is a hyperproperty $\mathcal{H} = \{\mathcal{T}' \in \mathcal{P}(\mathbb{S}^{\propto}) \mid \mathcal{T} \subseteq \mathcal{T}'\}$
- non interference is a hyperproperty:

$$\begin{array}{rcl}\mathcal{H} & = & \{X \in \mathcal{P}(\mathbb{S}^{\propto}) \mid \forall m \in \mathbb{M}, v, v' \in \mathbb{V}, \\ & & \qquad \Phi_{l,l'}(\alpha_{\partial}([\![\mathcal{S}]\!]^{\propto}))(m[x \leftarrow v])(x') \\ & & \qquad\qquad = \Phi_{l,l'}(\alpha_{\partial}([\![\mathcal{S}]\!]^{\propto}))(m[x \leftarrow v'])(x')\}\end{array}$$

Outline

Main items to remember

- **Semantics** can be **compared** by **abstract interpretation**
  - precision: **more abstract** means less precise, less verbose
  - computation: fixpoint transfers produce **constructive** definitions
  - constructive definitions are a good basis for **static analysis**
- **Semantic properties** can be classified in various groups
  This classification can serve as a **guidance**:
  - to discover what is hard to reason about
  - to select the right concrete semantics

Bibliography: semantics and abstraction

- **[C'97]**: **Constructive Design of a Hierarchy of Semantics of a Transition System by Abstract Interpretation**.
  Patrick Cousot.
  In Electronic Notes in Theoretical Computer Science, 6 (1997)
- **[ST'71]**: **Towards a mathematical semantics of computer languages**.
  Dana Scott and **Christopher Strachey**
  In Symposium on Computers and Automata, 1971.
- **[AS'87]**: **Recognizing Safety and Liveness**.
  Bowen Alpern and **Fred B. Schneider**.
  In Distributed Computing, Springer, 1987.
- **[CS'08]**: **Hyperproperties**.
  Michael R. Clarkson and **Fred B. Schneider**.
  In IEEE Computer Security Symposium, 2008.