Program Semantics

MPRI 2–6: Abstract Interpretation, application to verification and static analysis

Antoine Miné

Year 2018–2019

Course 02
18 September 2018
Goal

Study **broad classes** of semantics that express useful **program properties**:

- **several flavors**: state, trace and relational semantics
- at a **concrete** level:
  - express the **strongest property** of that shape that holds
  - **uncomputable**, must be further abstracted into a static analysis
    (e.g., using numeric domains as seen in the two following courses)

- **independently** from specific programming languages, using **transition systems**
  (we will quickly specialize to a simple numeric imperative language)

- express them _universally_ as **fixpoints**
- link them through abstractions

⇒ construct a **hierarchy of semantics**

A first step in analysis design is choosing the (uncomputable) concrete semantics of interest that can exactly express the properties at hand and is _complete_; sound computable abstractions come later and are guided by a target class of programs
Transition systems
Transition systems: definition

Language-neutral formalism to discuss program semantics.

**Transition system:** \((\Sigma, \tau)\)

- set of states \(\Sigma\),
  
  (memory states, \(\lambda\)-terms, configurations, etc., generally infinite)

- transition relation \(\tau \subseteq \Sigma \times \Sigma\).

\((\Sigma, \tau)\) is a general form of small-step operational semantics.

\((\sigma, \sigma') \in \tau\) is noted \(\sigma \to \sigma'\):

starting in state \(\sigma\), after one execution step, we can go to state \(\sigma'\).
Transition systems

Transition system: example

\[i \leftarrow 2;\]
\[n \leftarrow [\infty, +\infty];\]
\[\text{while } i < n \text{ do}\]
\[\text{if } ? \text{ then}\]
\[i \leftarrow i + 1\]

\[\Sigma \defeq \{i, n\} \rightarrow \mathbb{Z}\]
Transition systems

From programs to transition systems

**Example:** on a simple imperative language.

**Language syntax**

\[
\begin{align*}
\ell \text{stat} &::= \ell X &\leftarrow& \text{expr}^\ell \quad \text{(assignment)} \\
|&|\ell \text{if } \text{expr} \not\equiv 0 \text{ then } \ell \text{stat}^\ell \quad \text{(conditional)} \\
|&|\ell \text{while } \ell \text{expr} \not\equiv 0 \text{ do } \ell \text{stat}^\ell \quad \text{(loop)} \\
|&|\ell \text{stat}; \ell \text{stat}^\ell \quad \text{(sequence)}
\end{align*}
\]

- \(X \in \mathbb{V}\), where \(\mathbb{V}\) is a finite set of program variables,
- \(\ell \in \mathbb{L}\) is a finite set of control labels,
- \(\not\equiv \in \{=, \leq, \ldots\}\), the syntax of \text{expr} is left undefined.
  (see next course)

**Program states:** \(\Sigma \overset{\text{def}}{=} \mathbb{L} \times \mathbb{E}\) are composed of:
- a control state in \(\mathbb{L}\),
- a memory state in \(\mathbb{E} \overset{\text{def}}{=} \mathbb{V} \rightarrow \mathbb{R}\).
**Transitions:** \( \tau[\ell \text{ stat} \ell'] \subseteq \Sigma \times \Sigma \) is defined by **structural induction**.

Assuming that expression semantics is given as \( E[e] : \mathcal{E} \rightarrow \mathcal{P}(\mathbb{R}) \).

(see next course)

\[
\tau[\ell1 \ X \leftarrow e \ell2] \overset{\text{def}}{=} \{ (\ell1, \rho) \rightarrow (\ell2, \rho[X \mapsto v]) \mid \rho \in \mathcal{E}, \ v \in E[e] \ \rho \}
\]

\[
\tau[\ell1 \ \text{if} \ e \not\equiv 0 \ \text{then} \ \ell2 \ \ell3] \overset{\text{def}}{=} \bigcup \{ (\ell1, \rho) \rightarrow (\ell2, \rho) \mid \rho \in \mathcal{E}, \ \exists v \in E[e] \ \rho : v \not\equiv 0 \} \bigcup \{ (\ell1, \rho) \rightarrow (\ell3, \rho) \mid \rho \in \mathcal{E}, \ \exists v \in E[e] \ \rho : v \not\equiv 0 \} \bigcup \tau[\ell2 \ \ell3]
\]

\[
\tau[\ell1 \ \text{while} \ e \not\equiv 0 \ \text{do} \ \ell3 \ \ell4] \overset{\text{def}}{=} \bigcup \{ (\ell1, \rho) \rightarrow (\ell2, \rho) \mid \rho \in \mathcal{E} \} \bigcup \{ (\ell2, \rho) \rightarrow (\ell3, \rho) \mid \rho \in \mathcal{E}, \ \exists v \in E[e] \ \rho : v \not\equiv 0 \} \bigcup \{ (\ell2, \rho) \rightarrow (\ell4, \rho) \mid \rho \in \mathcal{E}, \ \exists v \in E[e] \ \rho : v \not\equiv 0 \} \bigcup \tau[\ell3 \ \ell2]
\]

\[
\tau[\ell1 \ s_1; \ \ell2 \ s_2 \ \ell3] \overset{\text{def}}{=} \tau[\ell1 \ s_1 \ s_2] \bigcup \tau[\ell2 \ s_2 \ \ell3]
\]
Use of transition systems

Transition systems are a form of structured operational semantics.

Other examples:

- semantics of $\lambda$-calculus
  (states are terms, transitions are reductions)

- abstract machines
  (states are configurations, transitions are instruction execution)

- concurrent programs
  (states are sequences of configurations, transitions model one process step)

- transitions are often labeled
  (to denote syntactic instruction, rewriting rule, process, etc.)

In practice:

Transitions systems are a theoretical tool.
We do not convert explicitly programs to transition systems to be analyzed!
Instead, the analysis proceeds directly on the AST, the CFG, or an equation system
following the same structural induction rules as the ones defining the transition system, but on an abstraction $\Sigma^\#$ of program states $\Sigma$. 
Initial and final states:

Transition systems \((\Sigma, \tau)\) are often enriched with:

- \(I \subseteq \Sigma\) a set of distinguished initial states,
- \(F \subseteq \Sigma\) a set of distinguished final states.

(e.g., limit observation to executions starting in an initial state and ending in a final state)

Blocking states \(B\):

- states with no successor \(B \overset{\text{def}}{=} \{ \sigma \mid \forall \sigma' \in \Sigma: \sigma \nrightarrow \sigma' \}\),
- model both correct program termination and program errors, (correct exit, program stuck, unhandled exception, etc.)
- often include (or equal) final states \(F\).
State semantics
Motivation

Many verification problems can be reduced to inferring the reachable program states:

- absence of run-time error, unhandled exception, deadlock, etc.  
  (no bad state is reached)
- infer variable bound inference, pointer targets  
  (application to verification and to optimization)
- infer invariants
- sometimes with some instrumentation of the semantics  
  (cost analysis by adding an instruction counter)
- etc.

Reasoning at the state level, in $P(\Sigma)$, is sufficient.
Post-image, pre-image

Forward and backward images, in $\mathcal{P}(\Sigma) \rightarrow \mathcal{P}(\Sigma)$:

- **successors**: (forward, post-image)
  \[ \text{post}_\tau(S) \overset{\text{def}}{=} \{ \sigma' \mid \exists \sigma \in S: \sigma \rightarrow \sigma' \} \]

- **predecessors**: (backward, pre-image)
  \[ \text{pre}_\tau(S) \overset{\text{def}}{=} \{ \sigma \mid \exists \sigma' \in S: \sigma \rightarrow \sigma' \} \]

$\text{post}_\tau$ and $\text{pre}_\tau$ are complete $\cup$−morphisms in $(\mathcal{P}(\Sigma), \subseteq, \cup, \cap, \emptyset, \Sigma)$.

\[ (\text{post}_\tau(\bigcup_{i \in I} S_i) = \bigcup_{i \in I} \text{post}_\tau(S_i), \text{pre}_\tau(\bigcup_{i \in I} S_i) = \bigcup_{i \in I} \text{pre}_\tau(S_i)) \]

$\text{post}_\tau$ and $\text{pre}_\tau$ are strict. (\[ \text{post}_\tau(\emptyset) = \text{pre}_\tau(\emptyset) = \emptyset \])
Dual post-images and pre-images:

\[ \tilde{\text{pre}}_\tau(S) \overset{\text{def}}{=} \{ \sigma \mid \forall \sigma' : \sigma \rightarrow \sigma' \implies \sigma' \in S \} \]

(states such that all successors satisfy \( S \))

\[ \tilde{\text{post}}_\tau(S) \overset{\text{def}}{=} \{ \sigma' \mid \forall \sigma : \sigma \rightarrow \sigma' \implies \sigma \in S \} \]

(states such that all predecessors satisfy \( S \))

\( \tilde{\text{pre}}_\tau \) and \( \tilde{\text{post}}_\tau \) are complete \( \cap \)-morphisms and not strict.

\( \tilde{\text{post}} \) is not much used...
Correspondences between images and dual images

We have the following correspondences:

- **inverse:** \( \text{pre}_\tau = \text{post}(\tau^{-1}) \) \( \quad \text{post}_\tau = \text{pre}(\tau^{-1}) \)

  \( \tilde{\text{pre}}_\tau = \tilde{\text{post}}(\tau^{-1}) \) \( \quad \tilde{\text{post}}_\tau = \tilde{\text{pre}}(\tau^{-1}) \)

  (where \( \tau^{-1} \overset{\text{def}}{=} \{(\sigma, \sigma') | (\sigma', \sigma) \in \tau\} \))

- **Galois connections:**

  \[
  (\mathcal{P}(\Sigma), \subseteq) \leftrightarrow_{\text{pre}_\tau} (\mathcal{P}(\Sigma), \subseteq) \text{ and }
  \]

  \[
  (\mathcal{P}(\Sigma), \subseteq) \leftrightarrow_{\text{post}_\tau} (\mathcal{P}(\Sigma), \subseteq).
  \]

  **proof:**

  \[
  \text{post}_\tau(A) \subseteq B \iff \{ \sigma' | \exists \sigma \in A : \sigma \rightarrow \sigma' \} \subseteq B \iff \forall \sigma \in A : \forall \sigma' : \sigma \rightarrow \sigma' \implies \sigma' \in B \iff (A \subseteq \{ \sigma \mid \forall \sigma' : \sigma \rightarrow \sigma' \implies \sigma' \in B \}) \iff A \subseteq \tilde{\text{pre}}_\tau(B);
  \]

  other directions are similar.
Deterministic systems

**Determinism:**

- $(\Sigma, \tau)$ is **deterministic** if $\forall \sigma \in \Sigma: |\text{post}_\tau(\{\sigma\})| = 1$,
  (every state has a single successor, no blocking state)

- most transition systems are **non-deterministic**.
  (e.g., effect of input $X \leftarrow [0, 10]$, program termination)

- If $\tau$ is deterministic
  then $\text{pre}_\tau = \tilde{\text{pre}}_\tau$ and $\text{post}_\tau = \tilde{\text{post}}_\tau$. 
Forward state reachability
Forward reachability

\( \mathcal{R}(I) \): states \textbf{reachable from} \( I \) in the transition system

\[
\mathcal{R}(I) \overset{\text{def}}{=} \{ \sigma \mid \exists n \geq 0, \sigma_0, \ldots, \sigma_n : \sigma_0 \in I, \sigma = \sigma_n, \forall i : \sigma_i \to \sigma_{i+1} \} \\
= \bigcup_{n \geq 0} \text{post}^n_\tau (I)
\]

(reachable \( \iff \) reachable from \( I \) in \( n \) steps of \( \tau \) for some \( n \geq 0 \))

\( \mathcal{R}(I) \) can be expressed in \textbf{fixpoint form}:

\[
\mathcal{R}(I) = \text{lfp } F_{\mathcal{R}} \text{ where } F_{\mathcal{R}}(S) \overset{\text{def}}{=} I \cup \text{post}_\tau(S)
\]

(\( F_{\mathcal{R}} \) shifts \( S \) and adds back \( I \))

Alternate characterization: \( \mathcal{R} = \text{lfp}_I G_{\mathcal{R}} \) where \( G_{\mathcal{R}}(S) \overset{\text{def}}{=} S \cup \text{post}_\tau(S) \).

(\( G_{\mathcal{R}} \) shifts \( S \) by \( \tau \) and accumulates the result with \( S \))

(proofs on next slide)
Forward reachability: proof

proof: of \( \mathcal{R}(\mathcal{I}) = \text{lfp } F_\mathcal{R} \) where \( F_\mathcal{R}(S) \overset{\text{def}}{=} \mathcal{I} \cup \text{post}_\tau(S) \)

\( (\mathcal{P}(\Sigma), \subseteq) \) is a CPO and \( \text{post}_\tau \) is continuous, hence \( F_\mathcal{R} \) is continuous:

\( F_\mathcal{R}(\bigcup_{i \in I} A_i) = \bigcup_{i \in I} F_\mathcal{R}(A_i) \).

By Kleene’s theorem, \( \text{lfp } F_\mathcal{R} = \bigcup_{n \in \mathbb{N}} F_\mathcal{R}^n(\emptyset) \).

We prove by recurrence on \( n \) that: \( \forall n: F_\mathcal{R}^n(\emptyset) = \bigcup_{i < n} \text{post}_\tau^i(\mathcal{I}) \).

(states reachable in less than \( n \) steps)

- \( F_\mathcal{R}^0(\emptyset) = \emptyset \)
- assuming the property at \( n \),

\[
F_\mathcal{R}^{n+1}(\emptyset) = F_\mathcal{R}(\bigcup_{i < n} \text{post}_\tau^i(\mathcal{I})) = \mathcal{I} \cup \text{post}_\tau(\bigcup_{i < n} \text{post}_\tau^i(\mathcal{I})) = \mathcal{I} \cup \bigcup_{i < n} \text{post}_\tau^i(\mathcal{I}) = \mathcal{I} \cup \bigcup_{1 \leq i < n+1} \text{post}_\tau^i(\mathcal{I}) = \bigcup_{i < n+1} \text{post}_\tau^i(\mathcal{I})
\]

Hence: \( \text{lfp } F_\mathcal{R} = \bigcup_{n \in \mathbb{N}} F_\mathcal{R}^n(\emptyset) = \bigcup_{i \in \mathbb{N}} \text{post}_\tau^i(\mathcal{I}) = \mathcal{R}(\mathcal{I}) \).

The proof is similar for the alternate form, given that \( \text{lfp}_\mathcal{I} G_\mathcal{R} = \bigcup_{n \in \mathbb{N}} G_\mathcal{R}^n(\mathcal{I}) \) and \( G_\mathcal{R}^n(\mathcal{I}) = F_\mathcal{R}^{n+1}(\emptyset) = \bigcup_{i \leq n} \text{post}_\tau^i(\mathcal{I}) \).
Graphical illustration

Transition system.
Graphical illustration

Initial states $\mathcal{I}$. 
Graphical illustration

Iterate $F^1_R(I)$. 
Iterate $F^2_R(I)$. 

Graphical illustration
Graphical illustration

Iterate $F^3_R(I)$. 
Graphical illustration

States reachable from $I$: $\mathcal{R}(I) = F^5_{\mathcal{R}}(I)$. 
Multiple forward fixpoints

Recall: \( \mathcal{R}(\mathcal{I}) = \text{lfp } F_{\mathcal{R}} \) where \( F_{\mathcal{R}}(S) \overset{\text{def}}{=} \mathcal{I} \cup \text{post}_\tau(S) \).

Note that \( F_{\mathcal{R}} \) may have several fixpoints.

Example:

Initial state \( \mathcal{I} \) \hspace{1cm} \( \mathcal{R}(\mathcal{I}) = \text{lfp } F_{\mathcal{R}} \) \hspace{1cm} \text{gfp } F_{\mathcal{R}}

Exercise:
Compute all the fixpoints of \( G_{\mathcal{R}}(S) \overset{\text{def}}{=} S \cup \text{post}_\tau(S) \) on this example.
Example application of forward reachability

- Infer the set of possible states at program end: $R(I) \cap F$.

**Example**

- $i \leftarrow 0$;
  - **while** $i < 100$ **do**
    - $i \leftarrow i + 1$;
    - $j \leftarrow j + [0, 1]$  
  - **done**

- Initial states $I$: $j \in [0, 10]$ at control state $\bullet$,
- Final states $F$: any memory state at control state $\bullet$,
- $\implies R(I) \cap F$: control at $\bullet$, $i = 100$, and $j \in [0, 110]$.

- Prove the absence of run-time error: $R(I) \cap B \subseteq F$.
  (never block except when reaching the end of the program)

- To ensure soundness, over-approximations are sufficient.
  (if $R^\#(I) \supseteq R(I)$, then $R^\#(I) \cap B \subseteq F \implies R(I) \cap B \subseteq F$)
Forward reachability in equational form

Idea:
- $\Sigma \overset{\text{def}}{=} \mathcal{L} \times \mathcal{E}$: decompose states as control in $\mathcal{L}$ and memory in $\mathcal{E}$
- associate a variable $\mathcal{X}_\ell$ in $\mathcal{E}$ for each label $\ell \in \mathcal{L}$
- link $\mathcal{X}_\ell$ through the semantics of instructions

Example:

\begin{align*}
\ell_1 & \quad i \leftarrow 2; \\
\ell_2 & \quad n \leftarrow [-\infty, +\infty]; \\
\ell_3 & \quad \text{while } \ell_4 \ i < n \text{ do} \\
\ell_5 & \quad \text{if } [0, 1] = 0 \text{ then} \\
\ell_6 & \quad i \leftarrow i + 1 \\
\ell_7 & \quad \\
\ell_8 & \\
\end{align*}

\begin{align*}
\mathcal{X}_1 &= \mathcal{I}_1 \\
\mathcal{X}_2 &= \text{C}\[ i \leftarrow 2 \] \mathcal{X}_1 \\
\mathcal{X}_3 &= \text{C}\[ n \leftarrow [-\infty, +\infty] \] \mathcal{X}_2 \\
\mathcal{X}_4 &= \mathcal{X}_3 \cup \mathcal{X}_7 \\
\mathcal{X}_5 &= \text{C}\[ i < n \] \mathcal{X}_4 \\
\mathcal{X}_6 &= \mathcal{X}_5 \\
\mathcal{X}_7 &= \mathcal{X}_5 \cup \text{C}\[ i \leftarrow i + 1 \] \mathcal{X}_6 \\
\mathcal{X}_8 &= \text{C}\[ i \geq n \] \mathcal{X}_4 \\
\end{align*}

- initial states $\mathcal{I} \overset{\text{def}}{=} \{ (\ell_1, \rho) | \rho \in \mathcal{I}_1 \}$ for some $\mathcal{I}_1 \subseteq \mathcal{E}$,
- $\text{C}\[ \cdot \] : \mathcal{P}(\mathcal{E}) \rightarrow \mathcal{P}(\mathcal{E})$ model assignments and tests (see next slide).
- We get the strongest invariant at each program point.
Systematic construction of the equation system

Atomic instructions:

- \( C[ X \leftarrow e ] X \overset{\text{def}}{=} \{ \rho[X \mapsto v] | \rho \in X, v \in E[e] \rho \} \)
- \( C[ e \bowtie 0 ] X \overset{\text{def}}{=} \{ \rho \in X | \exists v \in E[\rho] \rho : v \bowtie 0 \} \)

Systematic derivation of the equation system \( eq(\ell \text{ stat}^{\ell'}) \)
from the program syntax \( \ell \text{ stat}^{\ell'} \) by structural induction:

\[
eq \begin{align*}
eq (\ell^1 X \leftarrow e^2) & \overset{\text{def}}{=} \{ X^2 = C[ X \leftarrow e ] X^1 \} \\
eq (\ell^1 \text{ if } e \bowtie 0 \text{ then } \ell^2 s^3) & \overset{\text{def}}{=} \\
& \{ X^2_1 = C[ e \bowtie 0 ] X^1_1, X^3 = X^1_3 \cup C[ e \neq 0 ] X^1_1 \} \cup eq(\ell^2 s^3') \\
eq (\ell^1 \text{ while } \ell^2 e \bowtie 0 \text{ do } \ell^3 s^4) & \overset{\text{def}}{=} \\
& \{ X^2 = X^1 \cup X^4, X^3 = C[ e \bowtie 0 ] X^2, X^4 = C[ e \neq 0 ] X^2 \} \cup eq(\ell^3 s^4') \\
eq (\ell^1 s_1; \ell^2 s_2^3) & \overset{\text{def}}{=} eq(\ell^1 s_1^2) \cup (\ell^2 s_2^3) \\
\end{align*}
\]

where: \( X^{3'}, X^{4'} \) are fresh variables storing intermediate results

\(|\cup|\)—morphisms in a complete lattice \(\implies\) a smallest solution exists
By partitioning forward reachability wrt. control states, we retrieve the equation system form of program semantics.

**Control state partitioning**

We assume $\Sigma \overset{\text{def}}{=} \mathcal{L} \times \mathcal{E}$; note that: $\mathcal{P}(\Sigma) \simeq \mathcal{L} \rightarrow \mathcal{P}(\mathcal{E})$. We have a Galois isomorphism:

$$(\mathcal{P}(\Sigma), \subseteq) \dashv \vdash (\mathcal{L} \rightarrow \mathcal{P}(\mathcal{E}), \dot{\subseteq})$$

- $X \subseteq Y \iff \forall \ell \in \mathcal{L}: X(\ell) \subseteq Y(\ell)$
- $\alpha_{\mathcal{L}}(S) \overset{\text{def}}{=} \lambda \ell.\{ \rho \mid (\ell, \rho) \in S \}$
- $\gamma_{\mathcal{L}}(X) \overset{\text{def}}{=} \{(\ell, \rho) \mid \ell \in \mathcal{L}, \rho \in X(\ell)\}$
- given $F_{eq} \overset{\text{def}}{=} \alpha_{\mathcal{L}} \circ F_{\mathcal{R}} \circ \gamma_{\mathcal{L}}$
  
  we get an equation system $\forall \ell \in \mathcal{L}: x_\ell = F_{eq,\ell}(x_1, \ldots, x_n)$
- Note that: $\alpha_{\mathcal{L}} \circ \gamma_{\mathcal{L}} = \gamma_{\mathcal{L}} \circ \alpha_{\mathcal{L}} = id$. (no abstraction)
- simply reorganize the states by control location!
Invariance proof method

**Invariance proof method:** find an *inductive invariant* \( I \subseteq \Sigma \)

- \( I \subseteq l \)
  (contains initial states)

- \( \forall \sigma \in I: \sigma \rightarrow \sigma' \implies \sigma' \in I \)
  (invariant by program transition)

that implies the desired property: \( I \subseteq P \).

**Link with the state semantics** \( \mathcal{R}(I) \):

*Given* \( F_{\mathcal{R}}(S) \overset{\text{def}}{=} I \cup \text{post}_\tau(S) \), we have \( F_{\mathcal{R}}(I) \subseteq I \)  
\( \implies I \) is a post-fixpoint of \( F_{\mathcal{R}} \).

*Recall that* \( \mathcal{R}(I) = \text{lfp} F_{\mathcal{R}} \)  
\( \implies \mathcal{R}(I) \) is the tightest inductive invariant.
Link with Hoare logic

**Hoare logic:** proof method where we
- annotate program points with local state invariants in $P(\Sigma)$
- use logic rules to prove their correctness

\[
\begin{align*}
\{P[e/X]\} X & \leftarrow e \{P\} \\
\{P\} \text{ stat}_1 \{R\} & \quad \{R\} \text{ stat}_2 \{Q\} \\
\{P \land b\} \text{ stat } \{Q\} & \quad P \land \neg b \Rightarrow Q \\
\{P\} \text{ if } b \text{ then } \text{ stat } \{Q\} & \\
\{P\} \text{ while } b \text{ do } \text{ stat } \{P \land \neg b\} \\
\{P\} \text{ stat } \{Q\} & \quad P' \Rightarrow P \quad Q \Rightarrow Q' \\
\{P'\} \text{ stat } \{Q'\}
\end{align*}
\]

Link with the state semantics $R(I)$:
Recall the equation system $\forall \ell \in \mathcal{L}: X_\ell = F_{eq,\ell}(X_1, \ldots, X_n)$
obtained by partitioning reachability $F_R$ by control point
($P(\Sigma), \subseteq) \xleftarrow{\gamma_L} (\mathcal{L} \rightarrow P(\mathcal{E}), \subseteq)$.
- any post-fixpoint of $F_{eq}$ gives valid Hoare triples
- lfp $F_{eq}$ gives the tightest Hoare triples
Backward state co-reachability
Backward co-reachability

\( \mathcal{C}(\mathcal{F}) \): states co-reachable from \( \mathcal{F} \) in the transition system:

\[
\mathcal{C}(\mathcal{F}) \overset{\text{def}}{=} \{ \sigma \mid \exists n \geq 0, \sigma_0, \ldots, \sigma_n: \sigma = \sigma_0, \sigma_n \in \mathcal{F}, \forall i: \sigma_i \to \sigma_{i+1} \}
\]

\[
= \bigcup_{n \geq 0} \text{pre}^n_\tau(\mathcal{F})
\]

\( \mathcal{C}(\mathcal{F}) \) can also be expressed in fixpoint form:

\[
\mathcal{C}(\mathcal{F}) = \text{lfp } F_C \text{ where } F_C(S) \overset{\text{def}}{=} \mathcal{F} \cup \text{pre}_\tau(S)
\]

Alternate characterization: \( \mathcal{C}(\mathcal{F}) = \text{lfp } G_C \text{ where } G_C(S) = S \cup \text{pre}_\tau(S) \)

Justification: \( \mathcal{C}(\mathcal{F}) \) in \( \tau \) is exactly \( \mathcal{R}(\mathcal{F}) \) in \( \tau^{-1} \).
Graphical illustration

Transition system.
Graphical illustration

Final states $\mathcal{F}$. 
Graphical illustration

States co-reachable from $\mathcal{F}$. 
Application of backward co-reachability

\( I \cap C(B \setminus F) \)

Initial states that have at least one erroneous execution.

- initial states \( I \): \( i \in [0, 100] \) at •
- final states \( F \): any memory state at •
- blocking states \( B \): final, or \( j > 200 \) at any location
- \( I \cap C(B \setminus F) \): at •, \( i > 20 \)

\( I \cap (\Sigma \setminus C(B)) \)

Initial states that necessarily cause the program to loop.

Over-approximating \( C \) is useful to isolate possibly incorrect executions from those guaranteed to be correct.

Iterate forward and backward analyses interactively \( \implies \) abstract debugging [Bour93].
Backward co-reachability in equational form

**Principle:**
As before, reorganize transitions by label $\ell \in \mathcal{L}$, to get an equation system on $(\mathcal{X}_\ell)_\ell$, with $\mathcal{X}_\ell \subseteq \mathcal{E}$

**Example:**

$$
\begin{align*}
\ell_1 & \quad i \leftarrow 2; \\
\ell_2 & \quad n \leftarrow [\infty, +\infty]; \\
\ell_3 & \quad \text{while } \ell_4 \quad i < n \text{ do} \\
\ell_5 & \quad \text{if } [0, 1] = 0 \text{ then} \\
\ell_6 & \quad i \leftarrow i + 1 \\
\ell_7 & \\
\ell_8 &
\end{align*}
$$

$$
\begin{align*}
\mathcal{X}_1 &= C[i \rightarrow 2] \mathcal{X}_2 \\
\mathcal{X}_2 &= C[n \rightarrow [\infty, +\infty]] \mathcal{X}_3 \\
\mathcal{X}_3 &= \mathcal{X}_4 \\
\mathcal{X}_4 &= C[i < n] \mathcal{X}_5 \cup C[i \geq n] \mathcal{X}_8 \\
\mathcal{X}_5 &= \mathcal{X}_6 \cup \mathcal{X}_7 \\
\mathcal{X}_6 &= C[i \rightarrow i + 1] \mathcal{X}_7 \\
\mathcal{X}_7 &= \mathcal{X}_4 \\
\mathcal{X}_8 &= F_8
\end{align*}
$$

- final states $\mathcal{F} \overset{\text{def}}{=} \{(\ell_8, \rho) \mid \rho \in \mathcal{F}_8\}$ for some $\mathcal{F}_8 \subseteq \mathcal{E}$,
- $C[X \rightarrow e] \mathcal{X} \overset{\text{def}}{=} \{ \rho \mid \exists \nu \in \mathcal{E}[e] \rho : \rho[X \mapsto \nu] \in X \}$. 
Backward sufficient precondition state semantics
Sufficient preconditions

\( S(\mathcal{Y}) \): states with executions staying in \( \mathcal{Y} \).

\[
S(\mathcal{Y}) \overset{\text{def}}{=} \{ \sigma \mid \forall n \geq 0, \sigma_0, \ldots, \sigma_n: (\sigma = \sigma_0 \land \forall i: \sigma_i \rightarrow \sigma_{i+1}) \implies \sigma_n \in \mathcal{Y} \}
= \bigcap_{n \geq 0} \simpre_n(\mathcal{Y})
\]

\( S(\mathcal{Y}) \) can be expressed in fixpoint form:

\[
S(\mathcal{Y}) = \text{gfp } F_S \quad \text{where } F_S(S) \overset{\text{def}}{=} \mathcal{Y} \cap \simpre(\mathcal{Y})
\]

proof sketch: similar to that of \( R(\mathcal{I}) \), in the dual.

\( F_S \) is continuous in the dual CPO \((\mathcal{P}(\Sigma), \supseteq)\), because \( \simpre \) is:

\[
F_S(\bigcap_{i \in I} A_i) = \bigcap_{i \in I} F_S(A_i).
\]

By Kleene’s theorem in the dual, \( \text{gfp } F_S = \bigcap_{n \in \mathbb{N}} F_S^n(\Sigma) \).

We would prove by recurrence that \( F_S^n(\Sigma) = \bigcap_{i < n} \simpre_i(\mathcal{Y}) \).
Graphical illustration

Final states $\mathcal{F}$.
Goal: when stopping, stop in $\mathcal{F}$.
Goal: avoid stopping in a non-final state (i.e., error state) but passing through a non-blocking state is not (yet) an error \( \implies \) consider \( \mathcal{Y} = \mathcal{F} \cup (\Sigma \setminus \mathcal{B}) \).
Graphical illustration

Sufficient preconditions $S(\mathcal{Y})$ to stop in $\mathcal{F}$.

(without forcing the program to stop at all)
Graphical illustration

Sufficient preconditions $S(Y)$ to stop in $F$.
(without forcing the program to stop at all)

Note: $S(Y) \subset C(F)$
Correspondence with reachability:

We have a Galois connection:

\[
(\mathcal{P}(\Sigma), \subseteq) \xrightarrow{\mathcal{S}} (\mathcal{P}(\Sigma), \subseteq)
\]

- \(\mathcal{R}(\mathcal{I}) \subseteq \mathcal{Y} \iff \mathcal{I} \subseteq \mathcal{S}(\mathcal{Y})\)
  - definition of a Galois connection
  - all executions from \(\mathcal{I}\) stay in \(\mathcal{Y}\)
  - \(\iff \mathcal{I}\) includes only sufficient pre-conditions for \(\mathcal{Y}\)

- so \(\mathcal{S}(\mathcal{Y}) = \bigcup \{ X | \mathcal{R}(X) \subseteq \mathcal{Y} \}\)
  - by Galois connection property
  - \(\mathcal{S}(\mathcal{Y})\) is the largest initial set whose reachability is in \(\mathcal{Y}\)

We retrieve Dijkstra’s weakest liberal preconditions.

(proof sketch on next slide)
proof sketch:

Recall that \( \mathcal{R}(\mathcal{I}) = \text{lfp}_\mathcal{I} \mathcal{G}_\mathcal{R} \) where \( \mathcal{G}_\mathcal{R}(S) = S \cup \text{post}_\tau(S) \).
Likewise, \( \mathcal{S}(\mathcal{Y}) = \text{gfp}_\mathcal{Y} \mathcal{G}_\mathcal{S} \) where \( \mathcal{G}_\mathcal{S}(S) = S \cap \text{pre}_\tau(S) \).

Recall the Galois connection \((\mathcal{P}(\Sigma), \subseteq) \xleftrightarrow{\text{post}_\tau} (\mathcal{P}(\Sigma), \subseteq)\).

As a consequence \((\mathcal{P}(\Sigma), \subseteq) \xleftrightarrow{\mathcal{G}_\mathcal{S}} (\mathcal{P}(\Sigma), \subseteq)\).

The Galois connection can be lifted to fixpoint operators:
\((\mathcal{P}(\Sigma), \subseteq) \xleftrightarrow{x \mapsto \text{lfp}_x \mathcal{G}_\mathcal{R}} (\mathcal{P}(\Sigma), \subseteq)\).

Exercise: complete the proof sketch.
Application of sufficient preconditions

Initial states such that all executions are correct:
\( \mathcal{I} \cap S(\mathcal{F} \cup (\Sigma \setminus \mathcal{B})) \).
(the only blocking states reachable from initial states are final states)

program

- \( i \leftarrow 0; \)
  - \textbf{while} \( i < 100 \) \textbf{do}
    - \( i \leftarrow i + 1; \)
    - \( j \leftarrow j + [0, 1] \)
  - \textbf{done} •

- initial states \( \mathcal{I} \): \( j \in [0, 10] \) at •
- final states \( \mathcal{F} \): any memory state at •
- blocking states \( \mathcal{B} \): final, or \( j > 105 \) at any location
- \( \mathcal{I} \cap S(\mathcal{F} \cup (\Sigma \setminus \mathcal{B})) \): at •, \( j \in [0, 5] \)
  (note that \( \mathcal{I} \cap C(\mathcal{F} \cup (\Sigma \setminus \mathcal{B})) \) gives \( \mathcal{I} \))
Inferring sound sufficient preconditions requires under-approximations. If $S(X)$ is a sufficient precondition, any $S^\#(X) \subset S(X)$ is stronger, thus also sufficient. Most works in abstract interpretation only target over-approximations. The search for effective under-approximations remains an uncharted area.

Applications:

- **infer function contracts**
  infer sufficient conditions on the input so that the function has no error
  infer plausible specifications

- **optimization**
  e.g., hoist dynamic checks outside loops when possible
  replace: `for i in [0,n] get(a,i)` with: `if (X) then for i in [0,n] unsafe-get(a,i) else for i in [0,n] get(a,i)`
  where $X$ ensures no array overflow in the loop

- **infer counterexamples**
  infer conditions that ensures program mis-behavior even in the presence of non-determinism
Trace semantics
Motivation

Program semantics:

A natural semantic model of program execution are traces i.e., *sequences of states* encountered during execution

- **finite executions**
  - terminating programs
  - also: partial executions, i.e., the semantics of *test*

- **extension to infinite executions**
  - models possible non-termination

**Properties:**

*Trace properties can express temporal relations as well as termination and liveness properties*

- link with temporal logic
**Trace semantics**

**Traces and trace operations**

**Sequences, traces**

**Trace:** sequence of elements from \( \Sigma \)

- \( \epsilon \): empty trace (unique)
- \( \sigma \): trace of length 1 (assimilated to a state)
- \( \sigma_0, \ldots, \sigma_{n-1} \): trace of length \( n \)
- \( \sigma_0, \ldots, \sigma_n, \ldots \): infinite trace (length \( \omega \))

**Trace sets:**

- \( \Sigma^n \): the set of traces of length \( n \)
- \( \Sigma^\leq n \overset{\text{def}}{=} \bigcup_{i \leq n} \Sigma^i \): the set of traces of length at most \( n \)
- \( \Sigma^* \overset{\text{def}}{=} \bigcup_{i \in \mathbb{N}} \Sigma^i \): the set of finite traces
- \( \Sigma^\omega \): the set of infinite traces
- \( \Sigma^\infty \overset{\text{def}}{=} \Sigma^* \cup \Sigma^\omega \): the set of all traces
Traces of a transition system

**Execution traces:**

Non-empty sequences of states linked by the transition relation $\tau$.
- can be finite (in $P(\Sigma^*)$) or infinite (in $P(\Sigma^\omega)$)
- can be anchored at initial states, or final states, or none

**Atomic traces:**

- $I$: initial states $\simeq$ set of traces of length 1
- $F$: final states $\simeq$ set of traces of length 1
- $\tau$: transition relation $\simeq$ set of traces of length 2
  \[\{ \sigma, \sigma' \mid \sigma \rightarrow \sigma' \}\]

(as $\Sigma \simeq \Sigma^1$ and $\Sigma \times \Sigma \simeq \Sigma^2$)
Trace operations

Operations on traces:

- **length**: $|t| \in \mathbb{N} \cup \{\omega\}$ of a trace $t \in \Sigma^\omega$

- **concatenation** $\cdot$
  - $(\sigma_0, \ldots, \sigma_n) \cdot (\sigma'_0, \ldots) \overset{\text{def}}{=} \sigma_0, \ldots, \sigma_n, \sigma'_0, \ldots$
    (append to a finite trace)
  - $t \cdot t' \overset{\text{def}}{=} t$ if $t \in \Sigma^\omega$
    (append to an infinite trace does nothing)
  - $\epsilon \cdot t \overset{\text{def}}{=} t \cdot \epsilon \overset{\text{def}}{=} t$ ($\epsilon$ is neutral)

- **junction** $\bowtie$
  - $(\sigma_0, \ldots, \sigma_n) \bowtie (\sigma'_0, \sigma'_1 \ldots) \overset{\text{def}}{=} \sigma_0, \ldots, \sigma_n, \sigma'_1, \ldots$ when $\sigma_n = \sigma'_0$
    undefined if $\sigma_n \neq \sigma'_0$
  - $\epsilon \bowtie t$ and $t \bowtie \epsilon$ are undefined
  - $t \bowtie t' \overset{\text{def}}{=} t$, if $t \in \Sigma^\omega$
Extension to sets of traces:

- \( A \cdot B \overset{\text{def}}{=} \{ a \cdot b \mid a \in A, b \in B \} \)
  - \( \{ \epsilon \} \) is the neutral element for \( \cdot \)

- \( A \bowtie B \overset{\text{def}}{=} \{ a \bowtie b \mid a \in A, b \in B, a \bowtie b \text{ defined} \} \)
  - \( \Sigma \) is the neutral element for \( \bowtie \)

\[
\begin{align*}
A^0 & \overset{\text{def}}{=} \{ \epsilon \} & A \bowtie^0 & \overset{\text{def}}{=} \Sigma \\
A^{n+1} & \overset{\text{def}}{=} A \cdot A^n & A \bowtie^{n+1} & \overset{\text{def}}{=} A \bowtie A \bowtie^n \\
A^\omega & \overset{\text{def}}{=} A \cdot A \cdot \ldots & A \bowtie^\omega & \overset{\text{def}}{=} A \bowtie A \bowtie \ldots \\
A^* & \overset{\text{def}}{=} \bigcup_{n < \omega} A^n & A \bowtie^* & \overset{\text{def}}{=} \bigcup_{n < \omega} A \bowtie^n \\
A^\infty & \overset{\text{def}}{=} \bigcup_{n \leq \omega} A^n & A \bowtie^\infty & \overset{\text{def}}{=} \bigcup_{n \leq \omega} A \bowtie^n
\end{align*}
\]

Note: \( A^n \neq \{ a^n \mid a \in A \} \), \( A \bowtie^n \neq \{ a \bowtie^n \mid a \in A \} \) when \( |A| > 1 \)
Distributivity of junction

- Distributes over finite and infinite $\cup$:
  \[
  A \circ (\bigcup_{i \in I} B_i) = \bigcup_{i \in I} (A \circ B_i)
  \]
  and
  \[
  (\bigcup_{i \in I} A_i) \circ B = \bigcup_{i \in I} (A_i \circ B)
  \]
  where $I$ can be finite or infinite.

- Distributes finite $\cap$ but not infinite $\cap$

  - Example:
    \[
    \{a^\omega\} \circ (\bigcap_{n \in \mathbb{N}} \{a^m \mid n \geq m \}) = \{a^\omega\} \circ \emptyset = \emptyset \text{ but }
    \bigcap_{n \in \mathbb{N}} (\{a^\omega\} \circ \{a^m \mid n \geq m \}) = \bigcap_{n \in \mathbb{N}} \{a^\omega\} = \{a^\omega\}
    \]

- But, if $A \subseteq \Sigma^*$, then
  \[
  A \circ (\bigcap_{i \in I} B_i) = \bigcup_{i \in I} (A \circ B_i)
  \]
  even for infinite $I$

Note: concatenation · distributes infinite $\cap$ and $\cup$. 
Finite prefix trace semantics
Prefix trace semantics

\( \mathcal{T}_p(\mathcal{I}) \): partial, finite execution traces starting in \( \mathcal{I} \).

\[
\mathcal{T}_p(\mathcal{I}) \overset{\text{def}}{=} \{ \sigma_0, \ldots, \sigma_n \mid n \geq 0, \sigma_0 \in \mathcal{I}, \forall i: \sigma_i \rightarrow \sigma_{i+1} \}
\]

\[
= \bigcup_{n \geq 0} \mathcal{I} \overset{\tau}{\rightarrow} (\tau \overset{n}{\rightarrow})
\]

(traces of length \( n \), for any \( n \), starting in \( \mathcal{I} \) and following \( \tau \))

\( \mathcal{T}_p(\mathcal{I}) \) can be expressed in fixpoint form:

\[
\mathcal{T}_p(\mathcal{I}) = \text{lfp } F_p \text{ where } F_p(\mathcal{T}) \overset{\text{def}}{=} \mathcal{I} \cup \mathcal{T} \overset{\tau}{\rightarrow}
\]

\((F_p \text{ appends a transition to each trace, and adds back } \mathcal{I})\)

(proof on next slide)
Prefix trace semantics: proof

Proof of: $T_p(I) = \text{lfp } F_p$ where $F_p(T) = I \cup T \tau$.

Similar to the proof of $R(I) = \text{lfp } F_R$ where $F_R(S) \overset{\text{def}}{=} I \cup \text{post}_\tau(S)$.

$F_p$ is continuous in a CPO $(\mathcal{P}(\Sigma^*), \subseteq)$:

\[
F_p(\bigcup_{i \in I} T_i) \\
= I \cup (\bigcup_{i \in I} T_i) \tau \\
= I \cup (\bigcup_{i \in I} T_i \tau) = \bigcup_{i \in I} (I \cup T_i \tau)
\]

Hence (Kleene), $\text{lfp } F_p = \bigcup_{n \geq 0} F_p^n(\emptyset)$.

We prove by recurrence on $n$ that $\forall n: F_p^n(\emptyset) = \bigcup_{i < n} I \tau^i$:

\begin{itemize}
  \item $F_p^0(\emptyset) = \emptyset$,
  \item $F_p^{n+1}(\emptyset)$
    \[
      = I \cup F_p^n(\emptyset) \tau \\
      = I \cup (\bigcup_{i < n} I \tau^i) \tau \\
      = I \cup \bigcup_{i < n} (I \tau^i \tau) \\
      = I \tau^0 \cup \bigcup_{i < n} (I \tau^i) \\
      = \bigcup_{i < n+1} I \tau^i
    \end{itemize}

Thus, $\text{lfp } F_p = \bigcup_{n \in \mathbb{N}} F_p^n(\emptyset) = \bigcup_{n \in \mathbb{N}} \bigcup_{i < n} I \tau^i = \bigcup_{i \in \mathbb{N}} I \tau^i$.

Note: we also have $T_p(I) = \text{lfp}_I G_p$ where $G_p(T) = T \cup T \tau$.
Trace semantics

Prefix trace semantics: graphical illustration

\[ I \overset{\text{def}}{=} \{ a \} \]

\[ \tau \overset{\text{def}}{=} \{ (a, b), (b, b), (b, c) \} \]

Iterates:

\[ T_p(I) = \text{lfp } F_p \text{ where } F_p(T) \overset{\text{def}}{=} I \cup T \bowtie \tau. \]

- \[ F^0_p(\emptyset) = \emptyset \]
- \[ F^1_p(\emptyset) = I = \{ a \} \]
- \[ F^2_p(\emptyset) = \{ a, ab \} \]
- \[ F^3_p(\emptyset) = \{ a, ab, ab^2, abc \} \]
- \[ F^n_p(\emptyset) = \{ a, ab^i, ab^j c \mid i \in [1, n - 1], j \in [1, n - 2] \} \]
- \[ T_p(I) = \bigcup_{n \geq 0} F^n_p(\emptyset) = \{ a, ab^i, ab^i c \mid i \geq 1 \} \]
Prefix trace semantics: expressive power

The prefix trace semantics is the collection of finite observations of program executions.  

\[ \Rightarrow \text{Semantics of testing.} \]

Limitations:

- no information on infinite executions, (we will add infinite traces later)
- can bound maximal execution time: \( T_p(I) \subseteq \Sigma^{\leq n} \) but cannot bound minimal execution time. (we will consider maximal traces later)
Abstracting traces into states

**Idea:** view state semantics as abstractions of traces semantics.

A state in the state semantics corresponds to any partial execution trace terminating in this state.

We have a Galois embedding between finite traces and states:

\[(\mathcal{P}(\Sigma^*), \subseteq) \xrightarrow{\gamma_p} (\mathcal{P}(\Sigma), \subseteq)\]

- \(\alpha_p(T) \overset{\text{def}}{=} \{ \sigma \in \Sigma \mid \exists \sigma_0, \ldots, \sigma_n \in T : \sigma = \sigma_n \}\)  
  (last state in traces in \(T\))

- \(\gamma_p(S) \overset{\text{def}}{=} \{ \sigma_0, \ldots, \sigma_n \in \Sigma^* \mid \sigma_n \in S \}\)  
  (traces ending in a state in \(S\))
Abstracting traces into states (proof)

proof of: \((\alpha_p, \gamma_p)\) forms a Galois embedding.

Instead of the definition \(\alpha(c) \subseteq a \iff c \subseteq \gamma(a)\), we use the alternate characterization of Galois connections: \(\alpha\) and \(\gamma\) are monotonic, \(\gamma \circ \alpha\) is extensive, and \(\alpha \circ \gamma\) is reductive.

Embedding means that, additionally, \(\alpha \circ \gamma = id\).

- \(\alpha_p, \gamma_p\) are \(\cup\)-morphisms, hence monotonic
- \((\gamma_p \circ \alpha_p)(T)\)
  \(= \{ \sigma_0, \ldots, \sigma_n \mid \sigma_n \in \alpha_p(T) \}\)
  \(= \{ \sigma_0, \ldots, \sigma_n \mid \exists \sigma'_0, \ldots, \sigma'_m \in T: \sigma_n = \sigma'_m \}\)
  \(\supseteq T\)
- \((\alpha_p \circ \gamma_p)(S)\)
  \(= \{ \sigma \mid \exists \sigma_0, \ldots, \sigma_n \in \gamma_p(S): \sigma = \sigma_n \}\)
  \(= \{ \sigma \mid \exists \sigma_0, \ldots, \sigma_n: \sigma_n \in S, \sigma = \sigma_n \}\)
  \(= S\)
Trace semantics

Abstracting prefix traces into reachability

We can abstract semantic operators and their least fixpoint.

Recall that:

- $\mathcal{T}_p(\mathcal{I}) = \text{lfp } F_p$ where $F_p(T) \overset{\text{def}}{=} \mathcal{I} \cup T \cdot \tau$,
- $\mathcal{R}(\mathcal{I}) = \text{lfp } F_R$ where $F_R(S) \overset{\text{def}}{=} \mathcal{I} \cup \text{post}_\tau(S)$,
- $(\mathcal{P}(\Sigma^*), \subseteq) \xrightarrow{\alpha_p} (\mathcal{P}(\Sigma), \subseteq)$.

We have: $\alpha_p \circ F_p = F_R \circ \alpha_p$;
by fixpoint transfer, we get: $\alpha_p(\mathcal{T}_p(\mathcal{I})) = \mathcal{R}(\mathcal{I})$.

(proof on next slide)
Abstracting prefix traces into reachability (proof)

\textbf{proof:} of \( \alpha_p \circ F_p = F_R \circ \alpha_p \)

\[
(\alpha_p \circ F_p)(T) = \alpha_p(I \cup T \setminus \tau)
\]

= \{ \sigma \mid \exists \sigma_0, \ldots, \sigma_n \in I \cup T \setminus \tau : \sigma = \sigma_n \}

= I \cup \{ \sigma \mid \exists \sigma_0, \ldots, \sigma_n \in T \setminus \tau : \sigma = \sigma_n \}

= I \cup \{ \sigma \mid \exists \sigma_0, \ldots, \sigma_n \in T : \sigma_n \rightarrow \sigma \}

= I \cup \text{post}_\tau(\{ \sigma \mid \exists \sigma_0, \ldots, \sigma_n \in T : \sigma = \sigma_n \})

= I \cup \text{post}_\tau(\alpha_p(T))

= (F_R \circ \alpha_p)(T)
Abstracting traces into states (example)

program

\[ j \leftarrow 0; \]
\[ i \leftarrow 0; \]
\[ \textbf{while } i < 100 \textbf{ do} \]
\[ \quad i \leftarrow i + 1; \]
\[ \quad j \leftarrow j + [0, 1] \]
\[ \textbf{done} \]

- **prefix trace semantics:**
  \[ i \text{ and } j \text{ are increasing and } 0 \leq j \leq i \leq 100 \]
- **forward reachable state semantics:**
  \[ 0 \leq j \leq i \leq 100 \]

\[ \implies \text{the abstraction forgets the ordering of states.} \]
a state semantics states that $X \in [-20, -10] \cup [10, 20]$ at •:
this implies that assert $X \neq 0$ is correct but it is difficult to abstract (intervals are not sufficient: we need sets of intervals $\implies$ costly)

a path sensitive analysis can state that, at •:
- $X \in [-20, 10]$ if we went through the then branch
- $X \in [10, 20]$ if we went through the else branch
- in both cases, assert $X \neq 0$ is correct

$\implies$ we partition the (interval) state abstraction with respect to the history of computation (trace abstraction)

More in Xavier Rival’s course on partitioning
Prefix partial order: \( \preceq \) on \( \Sigma^\infty \)

\[
x \preceq y \iff \exists u \in \Sigma^\infty: x \cdot u = y
\]

\((\Sigma^\infty, \preceq)\) is a CPO, while \((\Sigma^*, \preceq)\) is not complete.

Prefix closure: \( \rho_p : \mathcal{P}(\Sigma^\infty) \to \mathcal{P}(\Sigma^\infty) \)

\[
\rho_p(T) \overset{\text{def}}{=} \{ u | \exists t \in T: u \preceq t, u \neq \epsilon \}
\]

\(\rho_p\) is an upper closure operator on \(\mathcal{P}(\Sigma^\infty \setminus \{\epsilon\})\).

(monotonic, extensive \( T \subseteq \rho_p(T) \), idempotent \( \rho_p \circ \rho_p = \rho_p \))

The prefix trace semantics is closed by prefix:

\[
\rho_p(\mathcal{T}_p(I)) = \mathcal{T}_p(I).
\]

(note that \( \epsilon \notin \mathcal{T}_p(I) \), which is why we disallowed \( \epsilon \) in \( \rho_p \))
Another state/trace abstraction: Ordering abstraction

Another **Galois embedding** between finite traces and states:

\[
(\mathcal{P}(\Sigma^*), \subseteq) \xrightarrow{\gamma_o} (\mathcal{P}(\Sigma), \subseteq) \xleftarrow{\alpha_o} (\mathcal{P}(\Sigma), \subseteq)
\]

- \(\alpha_o(T) \overset{\text{def}}{=} \{ \sigma \mid \exists \sigma_0, \ldots, \sigma_n \in T, i \leq n: \sigma = \sigma_i \} \)
  (set of all states appearing in some trace in \(T\))
- \(\gamma_o(S) \overset{\text{def}}{=} \{ \sigma_0, \ldots, \sigma_n \mid n \geq 0, \forall i \leq n: \sigma_i \in S \} \)
  (traces composed of elements from \(S\))

**proof sketch:**

\(\alpha_o \) and \(\gamma_o\) are monotonic, and \(\alpha_o \circ \gamma_o = id\).

\[(\gamma_o \circ \alpha_o)(T) = \{ \sigma_0, \ldots, \sigma_n \mid \forall i \leq n: \exists \sigma'_0, \ldots, \sigma'_m \in T, j \leq m: \sigma_i = \sigma'_j \} \supseteq T.\]
Semantic correspondence by ordering abstraction

We have: \( \alpha_o(T_p(I)) = R(I) \).

**proof:**
We have \( \alpha_o = \alpha_p \circ \rho_p \) (i.e.: a state is in a trace if it is the last state of one of its prefix).
Recall the prefix trace abstraction into states: \( R(I) = \alpha_p(T_p(I)) \) and the fact that the prefix trace semantics is closed by prefix: \( \rho_p(T_p(I)) = T_p(I) \).
We get \( \alpha_o(T_p(I)) = \alpha_p(\rho_p(T_p(I))) = \alpha_p(T_p(I)) = R(I) \).

This is a direct proof, not a fixpoint transfer proof (our theorems do not apply . . . )

**alternate proof:** generalized fixpoint transfer
Recall that \( T_p(I) = \text{lfp}\ F_p \) where \( F_p(T) \overset{\text{def}}{=} I \cup T \triangleright \tau \) and \( R(I) = \text{lfp}\ F_R \) where \( F_R(S) \overset{\text{def}}{=} I \cup \text{post}_\tau(S) \), but \( \alpha_o \circ F_p = F_R \circ \alpha_o \) does not hold in general, so, fixpoint transfer theorems do not apply directly.

However, \( \alpha_o \circ F_p = F_R \circ \alpha_o \) holds for sets of traces closed by prefix. By induction, the Kleene iterates \( a^n_p \) and \( a^n_R \) involved in the computation of \( \text{lfp}\ F_p \) and \( \text{lfp}\ F_R \) satisfy \( \forall n: \alpha_o(a^n_p) = a^n_R \), and so \( \alpha_o(\text{lfp}\ F_p) = \text{lfp}\ F_R \).
Finite suffix trace semantics
Suffix trace semantics

Similar results on the suffix trace semantics, going backwards from $\mathcal{F}$:

- $\mathcal{T}_s(\mathcal{F}) \overset{\text{def}}{=} \{ \sigma_0, \ldots, \sigma_n \mid n \geq 0, \sigma_n \in \mathcal{F}, \forall i : \sigma_i \rightarrow \sigma_{i+1} \}$
  (traces following $\tau$ and ending in a state in $\mathcal{F}$)

- $\mathcal{T}_s(\mathcal{F}) = \bigcup_{n \geq 0} \tau \triangleright n \triangleright \mathcal{F}$

- $\mathcal{T}_s(\mathcal{F}) = \text{lfp } F_s$ where $F_s(T) \overset{\text{def}}{=} \mathcal{F} \cup \tau \triangleright T$
  ($F_s$ prepends a transition to each trace, and adds back $\mathcal{F}$)

- $\alpha_s(\mathcal{T}_s(\mathcal{F})) = \mathcal{C}(\mathcal{F})$
  where $\alpha_s(T) \overset{\text{def}}{=} \{ \sigma \mid \exists \sigma_0, \ldots, \sigma_n \in T : \sigma = \sigma_0 \}$

- $\rho_s(\mathcal{T}_s(\mathcal{F})) = \mathcal{T}_s(\mathcal{F})$
  where $\rho_s(T) \overset{\text{def}}{=} \{ u \mid \exists t \in \Sigma^\infty : t \cdot u \in T, u \neq \epsilon \}$
  (closed by suffix)
Trace semantics

Finite suffix trace semantics

Graphical illustration

\[ F \overset{\text{def}}{=} \{ c \} \]
\[ \tau \overset{\text{def}}{=} \{ (a, b), (b, b), (b, c) \} \]

Iterates: \( \mathcal{T}_s(\mathcal{F}) = \text{lfp } F_s \) where \( F_s(T) \overset{\text{def}}{=} \mathcal{F} \cup \tau \circ T \).

- \( F_s^0(\emptyset) = \emptyset \)
- \( F_s^1(\emptyset) = \mathcal{F} = \{ c \} \)
- \( F_s^2(\emptyset) = \{ c, bc \} \)
- \( F_s^3(\emptyset) = \{ c, bc, bbc, abc \} \)
- \( F_s^n(\emptyset) = \{ c, b^i c, ab^j c \mid i \in [1, n - 1], j \in [1, n - 2] \} \)
- \( \mathcal{T}_s(\mathcal{F}) = \bigcup_{n \geq 0} F_s^n(\emptyset) = \{ c, b^i c, ab^i c \mid i \geq 1 \} \)
Application: termination inference

A program terminates if we can find a ranking function
strictly decreasing function with a lower bound

Termination semantics:
- start with final states, that terminate in 0 step
- go backwards in the program traces
  and annotate with one more step

This semantics:
- infers the optimal ranking function
- discovers initial states for which the program terminates
- can be abstracted using numeric domain
  (Work by Cousot & Cousot & Urban)
Finite partial trace semantics
Finite partial trace semantics

$\mathcal{T}$: all finite partial finite execution traces.

(not necessarily starting in $\mathcal{I}$ or ending in $\mathcal{F}$)

$\mathcal{T} \overset{\text{def}}{=} \{ \sigma_0, \ldots, \sigma_n \mid n \geq 0, \forall i: \sigma_i \rightarrow \sigma_{i+1} \}$

$= \bigcup_{n \geq 0} \Sigma \triangleright^\tau \triangleright^n$

$= \bigcup_{n \geq 0} \tau \triangleright^n \triangleright \Sigma$

- $\mathcal{T} = \mathcal{T}_p(\Sigma)$, hence $\mathcal{T} = \text{lfp } F_{p^*}$ where $F_{p^*}(\mathcal{T}) \overset{\text{def}}{=} \Sigma \cup \mathcal{T} \triangleright \tau$
  (prefix partial traces from any initial state)

- $\mathcal{T} = \mathcal{T}_s(\Sigma)$, hence $\mathcal{T} = \text{lfp } F_{s^*}$ where $F_{s^*}(\mathcal{T}) \overset{\text{def}}{=} \Sigma \cup \tau \triangleright \mathcal{T}$
  (suffix partial traces to any final state)

- $F_{p^*}^n(\emptyset) = F_{s^*}^n(\emptyset) = \bigcup_{i < n} \Sigma \triangleright \tau \triangleright i = \bigcup_{i < n} \tau \triangleright i \triangleright \Sigma = \mathcal{T} \cap \Sigma <^n$

- $\mathcal{T}_p(\mathcal{I}) = \mathcal{T} \cap (\mathcal{I} \cdot \Sigma^*)$ (restricted initial states)

- $\mathcal{T}_s(\mathcal{F}) = \mathcal{T} \cap (\Sigma^* \cdot \mathcal{F})$ (restricted final states)
Partial trace semantics: graphical illustration

\[ \tau \overset{\text{def}}{=} \{(a, b), (b, b), (b, c)\} \]

**Iterates:** \( T(\Sigma) = \text{lfp} \ F_{p^*} \) where \( F_{p^*}(T) \overset{\text{def}}{=} \Sigma \cup T \triangleleft T \).

- \( F_{p^*}^0(\emptyset) = \emptyset \)
- \( F_{p^*}^1(\emptyset) = \Sigma = \{a, b, c\} \)
- \( F_{p^*}^2(\emptyset) = \{a, b, c, ab, bb, bc\} \)
- \( F_{p^*}^3(\emptyset) = \{a, b, c, ab, bb, bc, abb, abc, bbb, bbc\} \)
- \( F_{p^*}^n(\emptyset) = \{ab^i, ab^j c, b^i c, b^k | i \in [0, n - 1], j \in [1, n - 2], k \in [1, n]\} \)
- \( T = \bigcup_{n \geq 0} F_{p^*}^n(\emptyset) = \{ab^i, ab^j c, b^i c, b^j | i \geq 0, j > 1\} \)

(using \( F_{s^*}(T) \overset{\text{def}}{=} \Sigma \cup \tau \triangleleft T \), we get the exact same iterates)
Abstracting partial traces to prefix traces

Idea: anchor partial traces at initial states $\mathcal{I}$.

We have a Galois connection:

$\left( \mathcal{P}(\Sigma^*), \subseteq \right) \xleftrightarrow{\alpha_{\mathcal{I}}} \left( \mathcal{P}(\Sigma^*), \subseteq \right)$

- $\alpha_{\mathcal{I}}(T) \overset{\text{def}}{=} T \cap (\mathcal{I} \cdot \Sigma^*)$ (keep only traces starting in $\mathcal{I}$)
- $\gamma_{\mathcal{I}}(T) \overset{\text{def}}{=} T \cup ((\Sigma \setminus \mathcal{I}) \cdot \Sigma^*)$ (add all traces not starting in $\mathcal{I}$)

We then have: $T_p(\mathcal{I}) = \alpha_{\mathcal{I}}(T)$.

(similarly $T_s(\mathcal{F}) = \alpha_{\mathcal{F}}(T)$ where $\alpha_{\mathcal{F}}(T) \overset{\text{def}}{=} T \cap (\Sigma^* \cdot \mathcal{F})$)

(proof on next slide)
Abstracting partial traces to prefix traces (proof)

**Proof**

\(\alpha_I\) and \(\gamma_I\) are monotonic.

\[
(\alpha_I \circ \gamma_I)(T) = (T \cup (\Sigma \setminus I) \cdot \Sigma^*) \cap I \cdot \Sigma^* = T \cap I \cdot \Sigma^* \subseteq T.
\]

\[
(\gamma_I \circ \alpha_I)(T) = (T \cap I \cdot \Sigma^*) \cup (\Sigma \setminus I) \cdot \Sigma^* = T \cup (\Sigma \setminus I) \cdot \Sigma^* \supseteq T.
\]

So, we have a Galois connection.

A direct proof of \(T_p(I) = \alpha_I(T)\) is straightforward, by definition of \(T_p\), \(\alpha_I\), and \(T\).

We can also retrieve the result by fixpoint transfer.

\[
T = \text{lfp } F_p \quad \text{where } F_p(T) \overset{\text{def}}{=} \Sigma \cup T \cap \tau.
\]

\[
T_p = \text{lfp } F \quad \text{where } F(T) \overset{\text{def}}{=} I \cup T \cap \tau.
\]

We have:

\[
(\alpha_I \circ F_p)(T) = (\Sigma \cup T \cap \tau) \cap (I \cdot \Sigma^*) = I \cup ((T \cap \tau) \cap (I \cdot \Sigma^*)) = I \cup ((T \cap (I \cdot \Sigma^*)) \cap \tau) = (F \circ \alpha_I)(T).
\]
(Partial) hierarchy of semantics

\[ \mathcal{R}(I) \xrightarrow{\alpha_p} \mathcal{T}_p(I) \]

\[ \mathcal{C}(F) \xrightarrow{\alpha_p} \mathcal{T}_s(F) \]

states

anchored traces

partial finite traces
Maximal finite and infinite trace semantics
Maximal traces

Maximal traces: $\mathcal{M}_\infty \in \mathcal{P}(\Sigma^\infty)$

- sequences of states linked by the transition relation $\tau$,
- start in any state ($\mathcal{I} = \Sigma$),
- either finite and stop in a blocking state ($\mathcal{F} = \mathcal{B}$),
- or infinite.

Maximal traces cannot be “extended” by adding a new transition in $\tau$ at their end.

Maximal traces defined as:

$$\mathcal{M}_\infty \overset{\text{def}}{=} \left\{ \sigma_0, \ldots, \sigma_n \in \Sigma^* \mid \sigma_n \in \mathcal{B}, \forall i < n: \sigma_i \to \sigma_{i+1} \right\} \cup$$

$$\left\{ \sigma_0, \ldots, \sigma_n, \ldots \in \Sigma^\omega \mid \forall i < \omega: \sigma_i \to \sigma_{i+1} \right\}$$

(can be anchored at $\mathcal{I}$ and $\mathcal{F}$ as: $\mathcal{M}_\infty \cap (\mathcal{I} \cdot \Sigma^\infty) \cap ((\Sigma^* \cdot \mathcal{F}) \cup \Sigma^\omega)$)
Goal: we look for a fixpoint characterization of $\mathcal{M}_\infty$.

We consider separately finite and infinite maximal traces.

- **Finite traces:** already done!

  From the suffix partial trace semantics, recall:
  
  $$\mathcal{M}_\infty \cap \Sigma^* = \mathcal{T}_s(\mathcal{B}) = \text{lfp } F_s$$

  recall that $F_s(T) \overset{\text{def}}{=} \mathcal{B} \cup \tau \mathcal{C} T$ in $(\mathcal{P}(\Sigma^*), \subseteq)$.

- **Infinite traces:**

  Additionally, we will prove:  
  
  $$\mathcal{M}_\infty \cap \Sigma^\omega = \text{gfp } G_s$$

  where $G_s(T) \overset{\text{def}}{=} \tau \mathcal{C} T$ in $(\mathcal{P}(\Sigma^\omega), \subseteq)$.

(proof in following slides)
Infinite trace semantics: graphical illustration

\[ \mathcal{B} \overset{\text{def}}{=} \{ c \} \]
\[ \tau \overset{\text{def}}{=} \{(a, b), (b, b), (b, c)\} \]

Iterates:
\[ \mathcal{M}_\infty \cap \Sigma^\omega = \text{gfp } G_s \text{ where } G_s(T) \overset{\text{def}}{=} \tau \bowtie T. \]

- \[ G_s^0(\Sigma^\omega) = \Sigma^\omega \]
- \[ G_s^1(\Sigma^\omega) = ab\Sigma^\omega \cup bb\Sigma^\omega \cup bc\Sigma^\omega \]
- \[ G_s^2(\Sigma^\omega) = abb\Sigma^\omega \cup bbb\Sigma^\omega \cup abc\Sigma^\omega \cup bbc\Sigma^\omega \]
- \[ G_s^3(\Sigma^\omega) = abbb\Sigma^\omega \cup bbbb\Sigma^\omega \cup abbc\Sigma^\omega \cup bbbc\Sigma^\omega \]
- \[ G_s^n(\Sigma^\omega) = \{ ab^n t, b^{n+1} t, ab^{n-1} ct, b^n ct | t \in \Sigma^\omega \} \]
- \[ \mathcal{M}_\infty \cap \Sigma^\omega = \bigcap_{n \geq 0} G_s^n(\Sigma^\omega) = \{ ab^\omega, b^\omega \} \]
Infinite trace semantics: proof

\[ \mathcal{M}_\infty \cap \Sigma^\omega = \text{gfp } G_s \]

where \( G_s(T) \overset{\text{def}}{=} \tau \triangleright T \) in \( (\mathcal{P}(\Sigma^\omega), \subseteq) \)

proof:

\( G_s \) is continuous in \( (\mathcal{P}(\Sigma^\omega), \supseteq) \): \( G_s(\bigcap_{i \in I} T_i) = \bigcap_{i \in I} G_s(T_i) \).

By Kleene’s theorem in the dual: \( \text{gfp } G_s = \bigcap_{n \in \mathbb{N}} G_s^n(\Sigma^\omega) \).

We prove by recurrence on \( n \) that \( \forall n : G_s^n(\Sigma^\omega) = \tau \triangleright^n \Sigma^\omega \):

- \( G_s^0(\Sigma^\omega) = \Sigma^\omega = \tau \triangleright^0 \Sigma^\omega \),
- \( G_s^{n+1}(\Sigma^\omega) = \tau \triangleright G_s^n(\Sigma^\omega) = \tau \triangleright (\tau \triangleright^n \Sigma^\omega) = \tau \triangleright^n \Sigma^\omega \).

\[ \text{gfp } G_s = \bigcap_{n \in \mathbb{N}} \tau \triangleright^n \Sigma^\omega \]

\[ = \{ \sigma_0, \ldots \in \Sigma^\omega \mid \forall n \geq 0: \sigma_0, \ldots, \sigma_{n-1} \in \tau \triangleright^n \} \]

\[ = \{ \sigma_0, \ldots \in \Sigma^\omega \mid \forall n \geq 0: \forall i < n: \sigma_i \rightarrow \sigma_{i+1} \} \]

\[ = \mathcal{M}_\infty \cap \Sigma^\omega \]
Least fixpoint formulation of maximal traces

**Idea:** To get a least fixpoint formulation for whole $\mathcal{M}_\infty$, merge finite and infinite maximal trace least fixpoint forms.

**Fixpoint fusion**

$\mathcal{M}_\infty \cap \Sigma^*$ is best defined on $(\Sigma^*, \subseteq, \cup, \cap, \emptyset, \Sigma^*)$.

$\mathcal{M}_\infty \cap \Sigma^\omega$ is best defined on $(\Sigma^\omega, \supseteq, \cap, \cup, \Sigma^\omega, \emptyset)$, the dual lattice (we transform the greatest fixpoint into a least fixpoint!)

We mix them into a new complete lattice $(\Sigma^\infty, \subseteq, \cup, \cap, \bot, \top)$:

- $A \sqsubseteq B \overset{\text{def}}{\iff} (A \cap \Sigma^*) \subseteq (B \cap \Sigma^*) \land (A \cap \Sigma^\omega) \supseteq (B \cap \Sigma^\omega)$
- $A \sqcup B \overset{\text{def}}{=} ((A \cap \Sigma^*) \cup (B \cap \Sigma^*)) \cup ((A \cap \Sigma^\omega) \cap (B \cap \Sigma^\omega))$
- $A \sqcap B \overset{\text{def}}{=} ((A \cap \Sigma^*) \cap (B \cap \Sigma^*)) \cup ((A \cap \Sigma^\omega) \cup (B \cap \Sigma^\omega))$
- $\bot \overset{\text{def}}{=} \Sigma^\omega$
- $\top \overset{\text{def}}{=} \Sigma^*$

In this lattice, $\mathcal{M}_\infty = \text{lfp} \ F_S$ where $F_S(T) \overset{\text{def}}{=} B \cup \tau \cap T$. (proof on next slides)
**Theorem:** fixpoint fusion

If \( X_1 = \text{lfp } F_1 \) in \((\mathcal{P}(D_1), \sqsubseteq_1)\) and \( X_2 = \text{lfp } F_2 \) in \((\mathcal{P}(D_2), \sqsubseteq_2)\)
and \( D_1 \cap D_2 = \emptyset \), then \( X_1 \cup X_2 = \text{lfp } F \) in \((\mathcal{P}(D_1 \cup D_2), \sqsubseteq)\) where:

- \( F(X) \overset{\text{def}}{=} F_1(X \cap D_1) \cup F_2(X \cap D_2) \),
- \( A \sqsubseteq B \iff (A \cap D_1) \sqsubseteq_1 (B \cap D_1) \land (A \cap D_2) \sqsubseteq_2 (B \cap D_2) \).

**proof:**

We have:

\[
F(X_1 \cup X_2) = F_1((X_1 \cup X_2) \cap D_1) \cup F_2((X_1 \cup X_2) \cap D_2) = F_1(X_1) \cup F_2(X_2) = X_1 \cup X_2,
\]
hence \( X_1 \cup X_2 \) is a fixpoint of \( F \).

Let \( Y \) be a fixpoint. Then \( Y = F(Y) = F_1(Y \cap D_1) \cup F_2(Y \cap D_2) \), hence, \( Y \cap D_1 = F_1(Y \cap D_1) \) and \( Y \cap D_1 \) is a fixpoint of \( F_1 \). Thus, \( X_1 \sqsubseteq_1 Y \cap D_1 \). Likewise, \( X_2 \sqsubseteq_2 Y \cap D_2 \). We deduce that \( X = X_1 \cup X_2 \sqsubseteq (Y \cap D_1) \cup (Y \cap D_2) = Y \), and so, \( X \) is \( F \)'s least fixpoint.

**note:** we also have \( \text{gfp } F = \text{gfp } F_1 \cup \text{gfp } F_2 \).
Least fixpoint formulation of maximal traces (proof)

We are now ready to finish the proof that $\mathcal{M}_\infty = \text{lfp} \ F_s$
where $F_s(T) \overset{\text{def}}{=} B \cup \tau \leadsto T$

**proof:**

We have:

- $\mathcal{M}_\infty \cap \Sigma^* = \text{lfp} \ F_s$ in $(\mathcal{P}(\Sigma^*), \subseteq)$,
- $\mathcal{M}_\infty \cap \Sigma^\omega = \text{lfp} \ G_s$ in $(\mathcal{P}(\Sigma^\omega), \supseteq)$ where $G_s(T) \overset{\text{def}}{=} \tau \leadsto T$,
- in $\mathcal{P}(\Sigma^\infty)$, we have
  
  $F_s(A) = (F_s(A) \cap \Sigma^*) \cup (F_s(A) \cap \Sigma^\omega) = F_s(A \cap \Sigma^*) \cup G_s(A \cap \Sigma^\omega)$.

So, by fixpoint fusion in $(\mathcal{P}(\Sigma^\infty), \subseteq)$, we have:

$\mathcal{M}_\infty = (\mathcal{M}_\infty \cap \Sigma^*) \cup (\mathcal{M}_\infty \cap \Sigma^\omega) = \text{lfp} \ F_s$.

**Note:** a greatest fixpoint formulation in $(\Sigma^\infty, \subseteq)$ also exists!
Actually, a fixpoint formulation in \((\Sigma^\infty, \subseteq)\) also exists.

Alternate fixpoint for finite maximal traces:

We saw that \(\mathcal{M}_\infty \cap \Sigma^* = \text{lfp } F_s\)
where \(F_s(T) \overset{\text{def}}{=} \mathcal{B} \cup \tau \triangleleft T\) in \((\mathcal{P}(\Sigma^*), \subseteq)\).

Additionally, we have \(\mathcal{M}_\infty \cap \Sigma^* = \text{gfp } F_s\) in \((\mathcal{P}(\Sigma^*), \subseteq)\).

\(F_s\) has a unique fixpoint in \((\mathcal{P}(\Sigma^*), \subseteq)\).

(proof on next slide)
Greatest fixpoint formulation of finite maximal traces

proof: of \( M_\infty \cap \Sigma^* = \text{gfp } F_s \) where \( F_s(T) \overset{\text{def}}{=} B \cup \tau \triangleright T \).

\( F_s \) is continuous in the dual \( (\mathcal{P}(\Sigma^*), \supseteq) \): \( F_s(\cap_{i \in I} A_i) = \cap_{i \in I} F_s(A_i) \).

By Kleene’s theorem in the dual \( (\mathcal{P}(\Sigma^*), \supseteq) \), we get: \( \text{gfp } F_s = \cap_{n \in \mathbb{N}} F_s^n(\Sigma^*) \).

We prove by recurrence on \( n \) that \( \forall n: F_s^n(\Sigma^*) = (\cup_{i < n} \tau \triangleright^i B) \cup (\tau \triangleright^n \Sigma^*) \): i.e., \( F_s^n(\Sigma^*) \) are the maximal finite traces of length at most \( n - 1 \), and the partial traces of length exactly \( n \) followed by any sequence of states:

- \( F_s^0(\Sigma^*) = \Sigma^* = \tau \triangleright^0 \Sigma^* \)
- \( F_s(F_s^n(\Sigma^*)) = B \cup (\tau \triangleright F_s^n(\Sigma^*)) \)
  \[ = B \cup \tau \triangleright ((\cup_{i < n} \tau \triangleright^i B) \cup (\tau \triangleright^n \Sigma^*)) \]
  \[ = B \cup (\cup_{i < n} \tau \triangleright \tau \triangleright^i B) \cup (\tau \triangleright^n \Sigma^*) \]
  \[ = B \cup (\cup_{1 < i < n+1} \tau \triangleright^i B) \cup (\tau \triangleright^n \Sigma^*) \]
  \[ = (\cup_{i < n+1} \tau \triangleright^i B) \cup (\tau \triangleright^n \Sigma^*) \]

We get:
\[ \cap_{n \in \mathbb{N}} F_s^n(\Sigma^*) = \cap_{n \in \mathbb{N}} (\cup_{i < n} \tau \triangleright^i B) \cup (\tau \triangleright^n \Sigma^*) = \cup_{n \in \mathbb{N}} \tau \triangleright^n B = M_\infty \cap \Sigma^*. \]
Greatest fixpoint of finite traces: graphical illustration

\[ M_\infty \cap \Sigma^* = \text{gfp } F_s \text{ where } F_s(T) \overset{\text{def}}{=} B \cup \tau \cup T. \]

- \( F_s^0(\Sigma^*) = \Sigma^* \)
- \( F_s^1(\Sigma^*) = \{b,b\} \cup ab\Sigma^* \cup bb\Sigma^* \cup bc\Sigma^* \)
- \( F_s^2(\Sigma^*) = \{bc,c\} \cup abb\Sigma^* \cup bbb\Sigma^* \cup abc\Sigma^* \cup bbc\Sigma^* \)
- \( F_s^3(\Sigma^*) = \{abc,bbc, bc, c\} \cup abbb\Sigma^* \cup bbbb\Sigma^* \cup abbc\Sigma^* \cup bbbc\Sigma^* \)
- \( F_s^n(\Sigma^*) = \{ab^i c, b^j c \mid i \in [1, n - 2], j \in [0, n - 1]\} \cup \{ab^n t, b^{n+1} t, ab^{n-1} ct, b^n ct \mid t \in \Sigma^* \} \)
- \( M_\infty \cap \Sigma^* = \bigcap_{n \geq 0} F_s^n(\Sigma^*) = \{ab^i c, b^j c \mid i \geq 1, j \geq 0\} \)
Greatest fixpoint formulation of maximal traces

From:
1. $\mathcal{M}_\infty \cap \Sigma^* = \text{gfp } F_s \text{ in } (\mathcal{P}(\Sigma^*), \subseteq) \text{ where } F_s(T) \overset{\text{def}}{=} B \cup \tau \bowtie T$
2. $\mathcal{M}_\infty \cap \Sigma^\omega = \text{gfp } G_s \text{ in } (\mathcal{P}(\Sigma^\omega), \subseteq) \text{ where } G_s(T) \overset{\text{def}}{=} \tau \bowtie T$

we deduce: $\mathcal{M}_\infty = \text{gfp } F_s \text{ in } (\mathcal{P}(\Sigma^\infty), \subseteq)$.

proof: similar to $\mathcal{M}_\infty = \text{lfp } F_s \text{ in } (\mathcal{P}(\Sigma^\infty), \subseteq)$, by fixpoint fusion.
Abstracting maximal traces into partial traces
Finite and infinite partial trace semantics

**Idea:** complete the partial traces $T$ with infinite traces.

$T_\infty$: all finite and infinite sequences of states linked by the transition relation $\tau$:

$$T_\infty \overset{\text{def}}{=} \{ \sigma_0, \ldots, \sigma_n \in \Sigma^* \mid \forall i < n: \sigma_i \rightarrow \sigma_{i+1} \} \cup \{ \sigma_0, \ldots, \sigma_n, \ldots \in \Sigma^\omega \mid \forall i < \omega: \sigma_i \rightarrow \sigma_{i+1} \}$$

(partial finite traces do not necessarily end in a blocking state)

Fixpoint form similar to $M_\infty$:

$T_\infty = \text{lfp } F_{s*}$ in $(\mathcal{P}(\Sigma^\infty), \subseteq)$ where $F_{s*}(T) \overset{\text{def}}{=} \Sigma \cup \tau \bowtie T$,

**proof:** similar to the proof of $M_\infty = \text{lfp } F_s$. 
Finite trace abstraction

Finite partial traces $\mathcal{T}$ are an abstraction of all partial traces $\mathcal{T}_\infty$.

We have a Galois embedding:

$$(\mathcal{P}(\Sigma^\infty), \subseteq) \xrightarrow{\gamma^*} (\mathcal{P}(\Sigma^*), \subseteq) \xleftarrow{\alpha^*} (\mathcal{P}(\Sigma^\omega), \subseteq)$$

- $\subseteq$ is the fused ordering on $\Sigma^* \cup \Sigma^\omega$:
  $$A \subseteq B \iff (A \cap \Sigma^*) \subseteq (B \cap \Sigma^*) \land (A \cap \Sigma^\omega) \supseteq (B \cap \Sigma^\omega)$$

- $\alpha^*(\mathcal{T}) \overset{\text{def}}{=} \mathcal{T} \cap \Sigma^*$
  (remove infinite traces)

- $\gamma^*(\mathcal{T}) \overset{\text{def}}{=} \mathcal{T}$
  (embedding)

- $\mathcal{T} = \alpha^*(\mathcal{T}_\infty)$
  (proof on next slide)
Finite trace abstraction (proof)

proof:

We have Galois embedding because:

- $\alpha^\ast$ and $\gamma^\ast$ are monotonic,
- given $T \subseteq \Sigma^\ast$, we have $(\alpha^\ast \circ \gamma^\ast)(T) = T \cap \Sigma^\ast = T$,
- $(\gamma^\ast \circ \alpha^\ast)(T) = T \cap \Sigma^\ast \sqsupseteq T$, as we only remove infinite traces.

Recall that $T_\infty = \text{lfp} \ F_{S^\ast}$ in $(\mathcal{P}(\Sigma^\infty), \sqsubseteq)$ and $T = \text{lfp} \ F_{S^\ast}$ in $(\mathcal{P}(\Sigma^\ast), \subseteq)$, where $F_{S^\ast}(T) \overset{\text{def}}{=} \Sigma \cup T \setminus \tau$.

As $\alpha^\ast \circ F_{S^\ast} = F_{S^\ast} \circ \alpha^\ast$ and $\alpha^\ast(\emptyset) = \emptyset$, we can apply the fixpoint transfer theorem to get $\alpha^\ast(T_\infty) = T$. 
Prefix abstraction

**Idea:** complete maximal traces by adding (non-empty) prefixes.

We have a Galois connection:

\[
(\mathcal{P}(\Sigma^\infty \setminus \{\epsilon\}), \subseteq) \xleftrightarrow{\gamma \preceq \alpha \preceq} (\mathcal{P}(\Sigma^\infty \setminus \{\epsilon\}), \subseteq)
\]

- \(\alpha \preceq (T) \overset{\text{def}}{=} \{ t \in \Sigma^\infty \setminus \{\epsilon\} | \exists u \in T: t \preceq u \}\) (set of all non-empty prefixes of traces in \(T\))

- \(\gamma \preceq (T) \overset{\text{def}}{=} \{ t \in \Sigma^\infty \setminus \{\epsilon\} | \forall u \in \Sigma^\infty \setminus \{\epsilon\}: u \preceq t \implies u \in T \}\) (traces with non-empty prefixes in \(T\))

**proof:**

\(\alpha \preceq \) and \(\gamma \preceq \) are monotonic.

\((\alpha \preceq \circ \gamma \preceq)(T) = \{ t \in T | \rho_p(t) \subseteq T \} \subseteq T \) (prefix-closed trace sets).

\((\gamma \preceq \circ \alpha \preceq)(T) = \rho_p(T) \supseteq T \).
Finite and infinite partial traces $T_{\infty}$ are an abstraction of maximal traces $M_{\infty}$: $T_{\infty} = \alpha_{\preceq}(M_{\infty})$.

**proof:**
Firstly, $T_{\infty}$ and $\alpha_{\preceq}(M_{\infty})$ coincide on infinite traces. Indeed, $T_{\infty} \cap \Sigma^\omega = M_{\infty} \cap \Sigma^\omega$ and $\alpha_{\preceq}$ does not add infinite traces, so: $T_{\infty} \cap \Sigma^\omega = \alpha_{\preceq}(M_{\infty}) \cap \Sigma^\omega$.

We now prove that they also coincide on finite traces. Assume $\sigma_0, \ldots, \sigma_n \in \alpha_{\preceq}(M_{\infty})$, then $\forall i < n: \sigma_i \rightarrow \sigma_{i+1}$, so, $\sigma_0, \ldots, \sigma_n \in T_{\infty}$.
Assume $\sigma_0, \ldots, \sigma_n \in T_{\infty}$, then it can be completed into a maximal trace, either finite or infinite, and so, $\sigma_0, \ldots, \sigma_n \in \alpha_{\preceq}(M_{\infty})$.

Note: no fixpoint transfer applies here.
(Partial) hierarchy of semantics

\[ \begin{align*}
\mathcal{R}(I) & \xrightarrow{\alpha_p} \mathcal{T}_p(I) \\
\mathcal{C}(\mathcal{F}) & \xrightarrow{\alpha_p} \mathcal{T}_s(\mathcal{F}) \\
\alpha_I & \xrightarrow{\alpha_I} \mathcal{T} \\
\alpha_F & \xrightarrow{\alpha_F} \mathcal{T} \\
\alpha_* & \xrightarrow{\alpha_*} \mathcal{T}_\infty \\
\alpha_< & \xrightarrow{\alpha_<} \mathcal{M}_\infty
\end{align*} \]

states
anchored traces
partial finite traces
partial traces
maximal traces
Trace properties
Reminder: state properties

State property: \( P \in \mathcal{P}(\Sigma) \).

Verification problem: \( \mathcal{R}(\mathcal{I}) \subseteq P \).
(all the states reachable from \( \mathcal{I} \) are in \( P \))

Examples:

- absence of blocking: \( P \overset{\text{def}}{=} \Sigma \setminus \mathcal{B} \),
- the variables remain in a safe range,
- dangerous program locations cannot be reached.
Trace properties

Trace property: $P \in \mathcal{P}(\Sigma^\infty)$

Verification problem: $\mathcal{M}_\infty \cap (\mathcal{I} \cdot \Sigma^\infty) \subseteq P$

(or, equivalently, as $\mathcal{M}_\infty \subseteq P'$ where $P' \overset{\text{def}}{=} P \cup ((\Sigma \setminus \mathcal{I}) \cdot \Sigma^\infty)$)

Examples:

- termination: $P \overset{\text{def}}{=} \Sigma^*$,
- non-termination: $P \overset{\text{def}}{=} \Sigma^\omega$,
- any state property $S \subseteq \Sigma$: $P \overset{\text{def}}{=} S^\infty$,
- maximal execution time: $P \overset{\text{def}}{=} \Sigma^{\leq k}$,
- minimal execution time: $P \overset{\text{def}}{=} \Sigma^{\geq k}$,
- ordering, e.g.: $P \overset{\text{def}}{=} (\Sigma \setminus \{b\})^* \cdot a \cdot \Sigma^* \cdot b \cdot \Sigma^\infty$.
  ($a$ and $b$ occur, and $a$ occurs before $b$)
Safety properties for traces

Idea: a safety property $P$ models that “nothing bad ever occurs”

- $P$ is provable by exhaustive testing;
  (observe the prefix trace semantics: $T_p(I) \subseteq P$)

- $P$ is disprovable by finding a single finite execution not in $P$.

Examples:

- any state property: $P \overset{\text{def}}{=} S^\infty$ for $S \subseteq \Sigma$,

- ordering: $P \overset{\text{def}}{=} \Sigma^\infty \setminus ((\Sigma \setminus \{a\})^* \cdot b \cdot \Sigma^\infty)$,
  (no $b$ can appear without an $a$ before,
  but we can have only $a$, or neither $a$ nor $b$)
  (not a state property)

- but termination $P \overset{\text{def}}{=} \Sigma^*$ is not a safety property.
  (disproving requires exhibiting an infinite execution)
Definition of safety properties

**Reminder:** finite prefix abstraction (simplified to allow $\epsilon$)

\[
(P(\Sigma^\infty), \subseteq) \xrightarrow{\gamma_* \preceq} (P(\Sigma^*), \subseteq) \xleftarrow{\alpha_* \preceq} (P(\Sigma^*), \subseteq)
\]

- \(\alpha_* \preceq(T) \overset{\text{def}}{=} \{ t \in \Sigma^* \mid \exists u \in T : t \preceq u \}\)
- \(\gamma_* \preceq(T) \overset{\text{def}}{=} \{ t \in \Sigma^\infty \mid \forall u \in \Sigma^* : u \preceq t \implies u \in T \}\)

The associated upper closure \(\rho_* \preceq \overset{\text{def}}{=} \gamma \preceq \circ \alpha \preceq\) is:

\(\rho_* \preceq = \lim \circ \rho_p\) where:

- \(\rho_p(T) \overset{\text{def}}{=} \{ u \in \Sigma^\infty \mid \exists t \in T : u \preceq t \}\),
- \(\lim(T) \overset{\text{def}}{=} T \cup \{ t \in \Sigma^\omega \mid \forall u \in \Sigma^* : u \preceq t \implies u \in T \}\).

**Definition:** \(P \in P(\Sigma^\infty)\) is a safety property if \(P = \rho_* \preceq(P)\).
Definition: $P \subseteq \mathcal{P}(\Sigma^\infty)$ is a safety property if $P = \rho_* \leq (P)$.

Examples and counter-examples:

- state property $P \overset{\text{def}}{=} S^\infty$ for $S \subseteq \Sigma$:
  \[
  \rho_p(S^\infty) = \lim(S^\infty) = S^\infty \implies \text{safety};
  \]

- termination $P \overset{\text{def}}{=} \Sigma^*$:
  \[
  \rho_p(\Sigma^*) = \Sigma^*, \text{ but } \lim(\Sigma^*) = \Sigma^\infty \neq \Sigma^* \implies \text{not safety};
  \]

- even number of steps $P \overset{\text{def}}{=} (\Sigma^2)^\infty$:
  \[
  \rho_p(\left((\Sigma^2)^\infty\right)) = \Sigma^\infty \neq (\Sigma^2)^\infty \implies \text{not safety}.
  \]
Proving safety properties

**Invariance proof method:** find an inductive invariant \( I \)

- set of finite traces \( I \subseteq \Sigma^* \)
- \( \mathcal{I} \subseteq I \)
  (contains traces reduced to an initial state)
- \( \forall \sigma_0, \ldots, \sigma_n \in I : \sigma_n \rightarrow \sigma_{n+1} \implies \sigma_0, \ldots, \sigma_n, \sigma_{n+1} \in I \)
  (invariant by program transition)

and implies the desired property: \( I \subseteq P \).

Link with the finite prefix trace semantics \( T_p(\mathcal{I}) \):

An inductive invariant is a post-fixpoint of \( F_p \): \( F_p(I) \subseteq I \)
where \( F_p(T) \overset{\text{def}}{=} \mathcal{I} \cup T \cup \tau \).

\( T_p(\mathcal{I}) = \text{lfp} F_p \) is the tightest inductive invariant.
Correctness of the invariant method for safety

**Soundness:**

If $P$ is a safety property and an inductive invariant $I$ exists then: $\mathcal{M}_\infty \cap (I \cdot \Sigma^\infty) \subseteq P$

**proof:**

Using the Galois connection between $\mathcal{M}_\infty$ and $\mathcal{T}$, we get:

$\mathcal{M}_\infty \cap (I \cdot \Sigma^\infty) \subseteq \rho_* (\mathcal{M}_\infty \cap (I \cdot \Sigma^\infty)) = \gamma_* (\alpha_* (\mathcal{M}_\infty \cap (I \cdot \Sigma^\infty))) = \gamma_* (\alpha_* (\mathcal{M}_\infty) \cap (I \cdot \Sigma^*)) = \gamma_* (\mathcal{T} \cap (I \cdot \Sigma^*)) = \gamma_* (\mathcal{T}_p (I))$.

Using the link between invariants and the finite prefix trace semantics, we have: $\mathcal{T}_p (I) \subseteq I \subseteq P$.

As $P$ is a safety property, $P = \gamma_* (P)$, so, $\gamma_* (\mathcal{T}_p (I)) \subseteq \gamma_* (P) = P$, and so, $\mathcal{M}_\infty \cap (I \cdot \Sigma^\infty) \subseteq P$.

**Completeness:** an inductive invariant always exists

**proof:** $\mathcal{T}_p (I)$ provides an inductive invariant.
Disproving safety properties

**Proof method:**

A safety property \( P \) can be disproved by constructing a finite prefix of execution that does not satisfy the property:

\[
\mathcal{M}_\infty \cap (\mathcal{I} \cdot \Sigma^\infty) \not\subseteq P \quad \Rightarrow \quad \exists t \in T_p(\mathcal{I}) : t \notin P
\]

**proof:**

By contradiction, assume that no such trace exists, i.e., \( T_p(\mathcal{I}) \subseteq P \).
We proved in the previous slide that this implies \( \mathcal{M}_\infty \cap (\mathcal{I} \cdot \Sigma^\infty) \subseteq P \).

**Examples:**

- disproving a state property \( P \overset{\text{def}}{=} S^\infty \):
  \[ \Rightarrow \text{find a partial execution containing a state in } \Sigma \setminus S; \]

- disproving an order property \( P \overset{\text{def}}{=} \Sigma^\infty \setminus ((\Sigma \setminus \{a\})^* \cdot b \cdot \Sigma^\infty) \):
  \[ \Rightarrow \text{find a partial execution where } b \text{ appears and not } a. \]
Liveness properties

Idea: liveness property \( P \in \mathcal{P}(\Sigma^\infty) \)

Liveness properties model that “something good eventually occurs”

- \( P \) cannot be proved by testing
  (if nothing good happens in a prefix execution, it can still happen in the rest of the execution)

- disproving \( P \) requires exhibiting an infinite execution not in \( P \)

Examples:

- termination: \( P \overset{\text{def}}{=} \Sigma^* \),

- inevitability: \( P \overset{\text{def}}{=} \Sigma^* \cdot a \cdot \Sigma^\infty \),
  (\( a \) eventually occurs in all executions)

- state properties are not liveness properties.
Definition of liveness properties

**Definition:** \( P \in \mathcal{P}(\Sigma^\infty) \) is a liveness property if \( \rho_{\ast\leq}(P) = \Sigma^\infty \).

Examples and counter-examples:

- **termination** \( P \overset{\text{def}}{=} \Sigma^* \):
  \[
  \rho_P(\Sigma^*) = \Sigma^* \text{ and } \lim(\Sigma^*) = \Sigma^\infty \implies \text{liveness;} \]

- **inevitability**: \( P \overset{\text{def}}{=} \Sigma^* \cdot a \cdot \Sigma^\infty \)
  \[
  \rho_P(P) = P \cup \Sigma^* \text{ and } \lim(P \cup \Sigma^*) = \Sigma^\infty \implies \text{liveness;} \]

- **state property** \( P \overset{\text{def}}{=} S^\infty \) for \( S \subseteq \Sigma \):
  \[
  \rho_P(S^\infty) = \lim(S^\infty) = S^\infty \neq \Sigma^\infty \text{ if } S \neq \Sigma \implies \text{not liveness;} \]

- **maximal execution time** \( P \overset{\text{def}}{=} \Sigma^{\leq k} \):
  \[
  \rho_P(\Sigma^{\leq k}) = \lim(\Sigma^{\leq k}) = \Sigma^{\leq k} \neq \Sigma^\infty \implies \text{not liveness;} \]

- The only property which is both safety and liveness is \( \Sigma^\infty \).
Proving liveness properties

**Variance proof method:** (informal definition)

Find a **decreasing quantity** until something good happens.

**Example:** termination proof

- find $f : \Sigma \rightarrow S$ where $(S, \sqsubseteq)$ is **well-ordered**;
  
  ($f$ is called a “ranking function”)

- $\sigma \in \mathcal{B} \implies f = \min S$;

- $\sigma \rightarrow \sigma' \implies f(\sigma') \sqsubseteq f(\sigma)$.

($f$ counts the number of steps remaining before termination)
Disproving liveness properties

**Property:**

If $P$ is a liveness property, then $\forall t \in \Sigma^*: \exists u \in P: t \preceq u$.

**proof:**

By definition of liveness, $\rho_{\preceq}(P) = \Sigma^\infty$, so $t \in \rho_{\preceq}(P) = \lim(\alpha_p(P))$.

As $t \in \Sigma^*$ and lim only adds infinite traces, $t \in \alpha_p(P)$.

By definition of $\alpha_p$, $\exists u \in P: t \preceq u$.

**Consequence:**

- liveness cannot be disproved by testing.
Trace topology

A topology on a set can be defined as:
- either a family of open sets (closed under union)
- or family of closed sets (closed under intersection)

**Trace topology:** on sets of traces in $\Sigma^\infty$

- the closed sets are: $C \overset{\text{def}}{=} \{ P \in \mathcal{P}(\Sigma^\infty) \mid P \text{ is a safety property} \} $
- the open sets can be derived as $O \overset{\text{def}}{=} \{ \Sigma^\infty \setminus c \mid c \in C \} $

**Topological closure:** $\rho : \mathcal{P}(X) \to \mathcal{P}(X)$

- $\rho(x) \overset{\text{def}}{=} \bigcap \{ c \in C \mid x \subseteq c \}$ (upper closure operator in $(\mathcal{P}(X), \subseteq)$)
- on our trace topology, $\rho = \rho_{\ast \leq}$. 

**Dense sets:**
- $x \subseteq X$ is dense if $\rho(x) = X$;
- on our trace topology, dense sets are liveness properties.
**Theorem:** decomposition on a topological space

Any set $x \subseteq X$ is the intersection of a closed set and a dense set.

**proof:**

We have $x = \rho(x) \cap (x \cup (X \setminus \rho(x)))$. Indeed:

\[
\rho(x) \cap (x \cup (X \setminus \rho(x))) = (\rho(x) \cap x) \cup (\rho(x) \cap (X \setminus \rho(x))) = \rho(x) \cap x = x \quad \text{as} \quad x \subseteq \rho(x).
\]

- $\rho(x)$ is closed
- $x \cup (X \setminus \rho(x))$ is dense because:
  
  \[
  \rho(x \cup (X \setminus \rho(x))) \supseteq \rho(x) \cup \rho(X \setminus \rho(x))
  \]
  
  \[
  \supseteq \rho(x) \cup (X \setminus \rho(x)) = X
  \]

**Consequence:** on trace properties

Every trace property is the conjunction of a safety property and a liveness property.

proving a trace property can be decomposed into a soundness proof and a liveness proof
Relational semantics
Big-step semantics
Finite big-step semantics

Pairs of states linked by a sequence of transitions in $\tau$.

$$\mathcal{BS} \overset{\text{def}}{=} \{ (\sigma_0, \sigma_n) \in \Sigma \times \Sigma \mid n \geq 0, \exists \sigma_1, \ldots, \sigma_{n-1} : \forall i < n : \sigma_i \rightarrow \sigma_{i+1} \}$$

(symmetric and transitive closure of $\tau$)

**Fixpoint form:**

$$\mathcal{BS} = \text{lfp } F_{B}$$

where $F_{B}(R) \overset{\text{def}}{=} \text{id} \cup \{ (\sigma, \sigma'') \mid \exists \sigma' : (\sigma, \sigma') \in R, \sigma' \rightarrow \sigma'' \}$. 
Relational abstraction

Relational abstraction: allows skipping intermediate steps.

We have a Galois embedding:

\[(\mathcal{P}(\Sigma^*), \subseteq) \xleftarrow{\gamma_{io}} \xrightarrow{\alpha_{io}} (\mathcal{P}(\Sigma \times \Sigma), \subseteq)\]

- \(\alpha_{io}(T) \overset{\text{def}}{=} \{ (\sigma, \sigma') | \exists \sigma_0, \ldots, \sigma_n \in T \colon \sigma = \sigma_0, \sigma' = \sigma_n \}\)
  (first and last state of a trace in \(T\))

- \(\gamma_{io}(R) \overset{\text{def}}{=} \{ \sigma_0, \ldots, \sigma_n \in \Sigma^* | \exists (\sigma, \sigma') \in R \colon \sigma = \sigma_0, \sigma' = \sigma_n \}\)
  (traces respecting the first and last states from \(R\))

proof sketch:

\(\gamma_{io}\) and \(\alpha_{io}\) are monotonic.

\((\gamma_{io} \circ \alpha_{io})(T) = \{ \sigma_0, \ldots, \sigma_n | \exists \sigma'_0, \ldots, \sigma'_m \in T \colon \sigma_0 = \sigma'_0, \sigma_n = \sigma'_m \}\).

\((\alpha_{io} \circ \gamma_{io})(R) = R.\)
The finite big-step semantics is an abstraction of the finite trace semantics: \( BS = \alpha_{\text{io}}(T) \).

**proof sketch:** by fixpoint transfer.

We have \( T = \text{lfp} \ F_p \) where \( F_p(T) \overset{\text{def}}{=} \Sigma \cup T \bowtie \tau \).

Moreover, \( F_B(R) \overset{\text{def}}{=} \text{id} \cup \{(\sigma, \sigma'') | \exists \sigma': (\sigma, \sigma') \in R, \sigma' \rightarrow \sigma'' \} \).

Then, \( \alpha_{\text{io}} \circ F_p = F_B \circ \alpha_{\text{io}} \) because \( \alpha_{\text{io}}(\Sigma) = \text{id} \) and \( \alpha_{\text{io}}(T \bowtie \tau) = \{(\sigma, \sigma'') | \exists \sigma': (\sigma, \sigma') \in \alpha_{\text{io}}(T) \wedge \sigma' \rightarrow \sigma'' \} \).

By fixpoint transfer: \( \alpha_{\text{io}}(T) = \text{lfp} \ F_B \).

We have a similar result using \( F_s(T) \overset{\text{def}}{=} \Sigma \cup \tau \bowtie T \) and \( F'_B(R) \overset{\text{def}}{=} \text{id} \cup \{(\sigma, \sigma'') | \exists \sigma': (\sigma', \sigma'') \in R \land \sigma \rightarrow \sigma' \} \).
Finite big-step semantics (example)

Program

\[
\begin{align*}
&i \leftarrow [0, +\infty]; \\
&\textbf{while } i > 0 \textbf{ do} \quad \textbf{done} \\
&\quad i \leftarrow i - [0, 1];
\end{align*}
\]

Finite big-step semantics \(BS\): \(\{ (\rho, \rho') \mid 0 \leq \rho'(i) \leq \rho(i) \}\).
Relational denotational semantics
Denotational semantics (in relation form)

In the denotational semantics, we forget all the intermediate steps and only keep the input / output relation:

- \((\sigma, \sigma') \in \Sigma \times B\): finite execution starting in \(\sigma\), stopping in \(\sigma'\),
- \((\sigma, \emptyset)\): non-terminating execution starting in \(\sigma\).

\((\neq \) big-step semantics: we no longer include \((\sigma, \sigma')\) if \(\sigma'\) is not blocking!\)

Construction by abstraction: of the maximal trace semantics \(M_\infty\).

\[
(P(\Sigma^\infty), \subseteq) \xrightarrow[\alpha_d]{} (P(\Sigma \times (\Sigma \cup \{\emptyset\})), \subseteq)
\]

\[
\alpha_d(T) \overset{\text{def}}{=} \alpha_{io}(T \cap \Sigma^*) \cup \{(\sigma, \emptyset) \mid \exists t \in \Sigma^\omega : \sigma \cdot t \in T\}
\]

\[
\gamma_d(R) \overset{\text{def}}{=} \gamma_{io}(R \cap (\Sigma \times \Sigma)) \cup \{\sigma \cdot t \mid (\sigma, \emptyset) \in R, t \in \Sigma^\omega\}
\]

(extension of \((\alpha_{io}, \gamma_{io})\) to infinite traces)

The denotational semantics is \(DS \overset{\text{def}}{=} \alpha_d(M_\infty)\).
Denotational fixpoint semantics

**Idea:** as $\mathcal{M}_\infty$, separate terminating and non-terminating behaviors, and use a fixpoint fusion theorem.

We have: $\mathcal{DS} = \text{lfp } F_d$

in $(\mathcal{P}(\Sigma \times (\Sigma \cup \{\circ\})), \sqsubseteq^*, \sqcup^*, \sqcap^*, \perp^*, \top^*)$, where

- $\perp^* \overset{\text{def}}{=} \{ (\sigma, \circ) | \sigma \in \Sigma \}$
- $\top^* \overset{\text{def}}{=} \{ (\sigma, \sigma') | \sigma, \sigma' \in \Sigma \} $
- $A \sqsubseteq^* B \iff ((A \cap \top^*) \subseteq (B \cap \top^*)) \land ((A \cap \perp^*) \supseteq (B \cap \perp^*))$
- $A \sqcup^* B \overset{\text{def}}{=} ((A \cap \top^*) \cup (B \cap \top^*)) \cup ((A \cap \perp^*) \cap (B \cap \perp^*))$
- $A \sqcap^* B \overset{\text{def}}{=} ((A \cap \top^*) \cap (B \cap \top^*)) \cup ((A \cap \perp^*) \cup (B \cap \perp^*))$
- $F_d(R) \overset{\text{def}}{=} \{ (\sigma, \sigma) | \sigma \in \mathcal{B} \} \cup \{ (\sigma, \sigma'') | \exists \sigma' : \sigma \rightarrow \sigma' \land (\sigma', \sigma'') \in R \}$
Denotational fixpoint semantics (proof)

proof:
We cannot use directly a fixpoint transfer on \( M_\infty = \text{lfp} F_s \) in \((P(\Sigma^\infty), \sqsubseteq)\) because our Galois connection \((\alpha_d, \gamma_d)\) uses the \( \subseteq \) order, not \( \sqsubseteq \)!
Instead, we use fixpoint transfer separately on finite and infinite executions, and then apply fixpoint fusion.

Recall that \( M_\infty \cap \Sigma^* = \text{lfp} F_s \) in \((P(\Sigma^*), \subseteq)\) where \( F_s(T) \overset{\text{def}}{=} B \cup \tau \triangleleft T \)
and \( M_\infty \cap \Sigma^\omega = \text{gfp} G_s \) in \((P(\Sigma^\omega), \subseteq)\) where \( G_s(T) \overset{\text{def}}{=} \bigcup \tau \triangleleft T \).

For finite execution, we have \( \alpha_d \circ F_s = F_d \circ \alpha_d \) in \( P(\Sigma^*) \rightarrow P(\Sigma \times \Sigma) \).
We can apply directly fixpoint transfer and get that: \( DS \cap (\Sigma \times \Sigma) = \text{lfp} F_d \).

(proof continued on next slide)
Denotational fixpoint semantics (proof cont.)

proof (continued): proof sketch for infinite executions

We have $\alpha_d \circ G_s = G_d \circ \alpha_d$ in $\mathcal{P}(\Sigma^\omega) \to \mathcal{P}(\Sigma \times \{\bigcirc\})$, where

$$G_d(R) \overset{\text{def}}{=} \{(\sigma, \sigma'\prime\prime) | \exists \sigma' : \sigma \rightarrow \sigma' \land (\sigma', \sigma'' \prime\prime) \in R\}.$$

A candidate proof would be to apply a fixpoint transfer theorem to $M_\infty \cap \Sigma^\omega = \text{gfp } G_s$, in the dual, replacing lfp with gfp, and $\cup$ with $\cap$.

However, the proof of the theorem, which required $\alpha$ to be continuous, would require $\alpha$ to be co-continuous in the dual, i.e., $\alpha_d(\bigcap_{i \in I} S_i) = \bigcap_{i \in I} \alpha_d(S_i)$.

This does not hold. Consider for example: $I = \mathbb{N}$ and $S_i = \{ a^n b^\omega | n > i \}$:

$\bigcap_{i \in \mathbb{N}} S_i = \emptyset$, but $\forall i : \alpha_d(S_i) = \{(a, \bigcirc)\}$.

We use instead a fixpoint transfer based on Tarksi’s theorem.

We have $\text{gfp } G_s = \cup \{ X | X \subseteq G_s(X) \}$.

Thus, $\alpha_d(\text{gfp } G_s) = \alpha_d(\cup \{ X | X \subseteq G_s(X) \}) = \cup \{ \alpha_d(X) | X \subseteq G_s(X) \}$ as $\alpha_d$ is a complete $\cup$ morphism. The proof is finished by noting that the commutation $\alpha_d \circ G_s = G_d \circ \alpha_d$ and the Galois embedding $(\alpha_d, \gamma_d)$ imply that

$\{ \alpha_d(X) | X \subseteq G_s(X) \} = \{ \alpha_d(X) | \alpha_d(X) \subseteq G_d(\alpha_d(X)) \} = \{ Y | Y \subseteq G_d(Y) \}$.

(the complete proof can be found in [Cous02])
Denotational semantics (example)

Denotational semantics $\mathcal{DS}$:
\[
\{ (\rho, \rho') \mid \rho(i) \geq 0 \land \rho'(i) = 0 \} \cup \{ (\rho, \emptyset) \mid \rho(i) \geq 0 \}.
\]

(quite different from the big-step semantics)
Relational denotational semantics

Note: denotational semantics are often presented as functions, not relations

This is possible using the following Galois isomorphism:

\[(\mathcal{P}(\Sigma \times (\Sigma \cup \{\emptyset\})), \subseteq^*) \iff (\Sigma \rightarrow \mathcal{P}(\Sigma \cup \{\emptyset\}), \subseteq^*)\]

- \(\alpha_{df}(R) \overset{\text{def}}{=} \lambda \sigma. \{ \sigma' \mid (\sigma, \sigma') \in R \}\)
- \(\gamma_{df}(f) \overset{\text{def}}{=} \{ (\sigma, \sigma') \mid \sigma' \in f(\sigma) \}\)
- \(f \subseteq^* f \iff \forall \sigma: (f(\sigma) \cap \Sigma \subseteq g(\sigma) \cap \Sigma) \land (\emptyset \in g(\sigma) \implies \emptyset \in f(\sigma))\)

We get that: \(\alpha_{df}(\mathcal{D}S) = \text{lfp } F'_d\) where

\(F'_d(f) \overset{\text{def}}{=} (\alpha_{df} \circ F_d \circ \gamma_{df})(f) = (\lambda \sigma. \{ \sigma \mid \sigma \in \mathcal{B} \}) \cup (f \circ \text{post}_\tau)\).

(proof by fixpoint transfer, as \(F'_d \circ \alpha_{df} = F_d \circ \alpha_{df}\))
Another part of the hierarchy of semantics

See [Cou82] for more semantics in this diagram.

Note: we show transition systems as an abstraction of the partial trace semantics this is left as exercise (see assignment).
Beyond trace properties
We generalize the notion of properties and program verification.

**General setting:**

- **programs:** $prog \in Prog$
- **semantics:** $[\cdot] : Prog \to \mathcal{D}$ in some semantic domain $\mathcal{D}$
- **property:** the set of allowed program semantics $P \in \mathcal{P}(\mathcal{D})$
  - $\subseteq$ gives an information order on properties
    - $P \subseteq P'$ means that $P'$ is weaker than $P$ (allows more semantics)
- **verification problem:** $[prog] \in P$
Collecting semantics: \( \text{Col} : \text{Prog} \rightarrow \mathcal{P}(\mathcal{D}) \)

- \( \text{Col}(\text{prog}) \overset{\text{def}}{=} \{ \llbracket \text{prog} \rrbracket \} \)

- \( \text{Col}(\text{prog}) \) is the strongest property of a program in \( \mathcal{P}(\mathcal{D}) \)
  (relative to the choice of the semantic domain \( \mathcal{D} \) and function \( \llbracket \cdot \rrbracket \))

- we can interpret program verification as property inclusion:
  \( \text{Col}(\text{prog}) \subseteq P \)

  \( P \) is weaker than \( \text{Col}(\text{prog}) \) in the information order of properties

- generally, the collecting semantics cannot be computed; we settle for a weaker property \( S^\# \) that
  - is sound: \( \text{Col}(\text{prog}) \subseteq S^\# \)
  - implies the desired property: \( S^\# \subseteq P \)
Beyond trace properties

Retrieving state and trace properties

Reachability state semantics:

- $\mathcal{D} \overset{\text{def}}{=} \mathcal{P}(\Sigma)$
- $\llbracket \cdot \rrbracket \overset{\text{def}}{=} \mathcal{R}(\mathcal{I})$

Trace semantics:

- $\mathcal{D} \overset{\text{def}}{=} \mathcal{P}(\Sigma^\infty)$
- $\llbracket \cdot \rrbracket \overset{\text{def}}{=} \mathcal{M}_\infty \cap (\mathcal{I} \cdot \Sigma^\infty)$

State and trace properties: interpreted in $\mathcal{P}(\mathcal{D})$

- $\rho_\downarrow(x)$ for some $x \in \mathcal{D}$
- where $\rho_\downarrow(x) \overset{\text{def}}{=} \{ y \in \mathcal{D} \mid y \subseteq x \} \in \mathcal{P}(\mathcal{D})$

(proof: $A \subseteq B \iff A \in \rho_\downarrow(B)$)
Note: expressing properties in $\mathcal{P}(\mathcal{D})$ is more general than expressing properties in $\mathcal{D}$.

**Example:** non-interference for variable $X$

\[ P \overset{\text{def}}{=} \{ T \in \mathcal{P}(\Sigma^*) \mid \forall \sigma_0, \ldots, \sigma_n \in T : \forall \sigma' : \sigma_0 \equiv \sigma'_0 \implies \exists \sigma'_0, \ldots, \sigma'_m \in T : \sigma'_m \equiv \sigma_m \} \]

where \((\ell, \rho) \equiv (\ell', \rho') \iff \ell = \ell' \land \forall V \neq X : \rho(V) = \rho'(V)\)

(changing the initial value of $X$ does not affect the set of final environments up to the value of $X$)

There is no $Q \subseteq \Sigma^\infty$ such that $P = \rho_{\downarrow}(Q)$. \[ \implies \] non-interference is not a trace property in $\mathcal{P}(\Sigma^\infty)$.

Reading assignment: hyperproperties.
