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 unversally as fixpoints
- link them through abstractions
  $\Longrightarrow$ 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: (Σ, τ)

- set of states Σ,
  (memory states, $\lambda-$terms, configurations, etc., generally infinite)
- transition relation $\tau \subseteq \Sigma \times \Sigma$.

(Σ, τ) is a general form of small-step operational semantics.

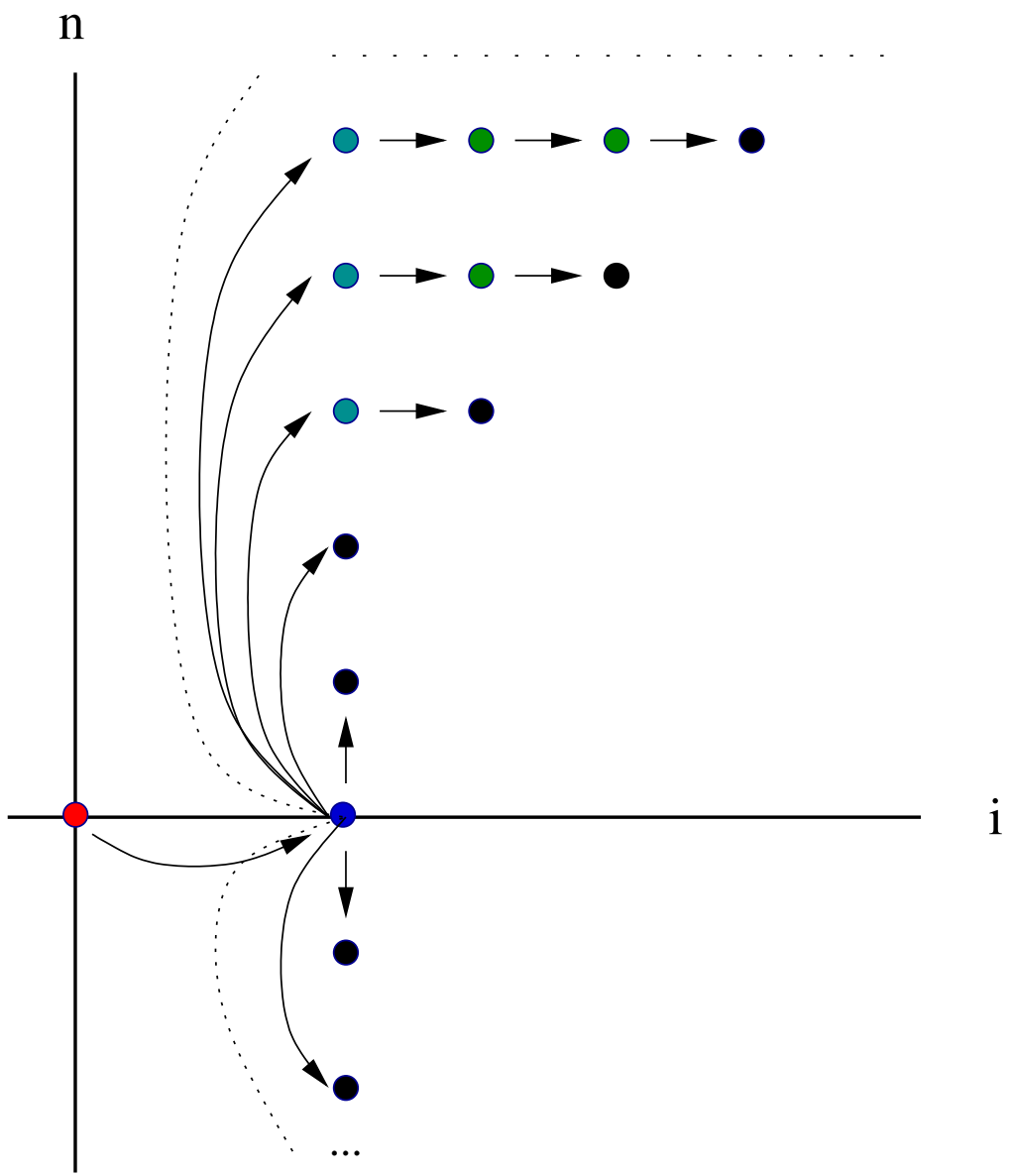
$(\sigma, \sigma') \in \tau$ is noted $\sigma \to \sigma'$:

starting in state σ, after one execution step, we can go to state σ'.

Transition system: example

$i \leftarrow 2;$
$n \leftarrow [-\infty, +\infty];$
while $i < n$ **do**
  if ? **then**
    $i \leftarrow i + 1$

$\Sigma \stackrel{\text{def}}{=} \{i, n\} \to \mathbb{Z}$



From programs to transition systems

Example: on a simple imperative language.

Language syntax

$$
\begin{array}{lcll}
{}^{\ell}stat^{\ell} & ::= & {}^{\ell}X \leftarrow expr^{\ell} & \textit{(assignment)} \\
& | & {}^{\ell}\textbf{if } expr \bowtie 0 \textbf{ then } {}^{\ell}stat^{\ell} & \textit{(conditional)} \\
& | & {}^{\ell}\textbf{while } {}^{\ell}expr \bowtie 0 \textbf{ do } {}^{\ell}stat^{\ell} & \textit{(loop)} \\
& | & {}^{\ell}stat; {}^{\ell}stat^{\ell} & \textit{(sequence)}
\end{array}
$$

- $X \in \mathbb{V}$, where $\mathbb{V}$ is a finite set of program variables,
- $\ell \in \mathcal{L}$ is a finite set of control labels,
- $\bowtie \in \{=, \leq, \ldots\}$, the syntax of *expr* is left undefined. (see next course)

Program states: $\Sigma \stackrel{\text{def}}{=} \mathcal{L} \times \mathcal{E}$ are composed of:

- a control state in $\mathcal{L}$,
- a memory state in $\mathcal{E} \stackrel{\text{def}}{=} \mathbb{V} \to \mathbb{R}$.

From programs to transition systems

Transitions: $\tau[{}^{\ell} stat^{\ell'}] \subseteq \Sigma \times \Sigma$ is defined by **structural induction**.

Assuming that expression semantics is given as $\mathsf{E}[\![e]\!] : \mathcal{E} \to \mathcal{P}(\mathbb{R})$.
(see next course)

$$\tau[{}^{\ell 1} X \leftarrow e^{\ell 2}] \stackrel{\text{def}}{=} \{ (\ell 1, \rho) \to (\ell 2, \rho[X \mapsto v]) \mid \rho \in \mathcal{E},\ v \in \mathsf{E}[\![e]\!] \rho \}$$

$$\begin{aligned}\tau[{}^{\ell 1}\textbf{if } e \bowtie 0 \textbf{ then } {}^{\ell 2} s^{\ell 3}] \stackrel{\text{def}}{=}\\ \{ (\ell 1, \rho) \to (\ell 2, \rho) \mid \rho \in \mathcal{E},\ \exists v \in \mathsf{E}[\![e]\!] \rho \colon v \bowtie 0 \} \cup\\ \{ (\ell 1, \rho) \to (\ell 3, \rho) \mid \rho \in \mathcal{E},\ \exists v \in \mathsf{E}[\![e]\!] \rho \colon v \not\bowtie 0 \} \cup \tau[{}^{\ell 2} s^{\ell 3}]\end{aligned}$$

$$\begin{aligned}\tau[{}^{\ell 1}\textbf{while } {}^{\ell 2} e \bowtie 0 \textbf{ do } {}^{\ell 3} s^{\ell 4}] \stackrel{\text{def}}{=}\\ \{ (\ell 1, \rho) \to (\ell 2, \rho) \mid \rho \in \mathcal{E} \} \cup\\ \{ (\ell 2, \rho) \to (\ell 3, \rho) \mid \rho \in \mathcal{E},\ \exists v \in \mathsf{E}[\![e]\!] \rho \colon v \bowtie 0 \} \cup\\ \{ (\ell 2, \rho) \to (\ell 4, \rho) \mid \rho \in \mathcal{E},\ \exists v \in \mathsf{E}[\![e]\!] \rho \colon v \not\bowtie 0 \} \cup \tau[{}^{\ell 3} s^{\ell 2}]\end{aligned}$$

$$\tau[{}^{\ell 1} s_1 ;\, {}^{\ell 2} s_2 {}^{\ell 3}] \stackrel{\text{def}}{=} \tau[{}^{\ell 1} s_1 {}^{\ell 2}] \cup \tau[{}^{\ell 2} s_2 {}^{\ell 3}]$$

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 $\Sigma^\sharp$ of program states Σ.

Initial, final, blocking states

Initial and final states:

Transition systems (Σ, τ) are often enriched with:

- $\mathcal{I} \subseteq \Sigma$ a set of distinguished initial states,
- $\mathcal{F} \subseteq \Sigma$ 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 $\mathcal{B}$:

- states with no successor $\mathcal{B} \stackrel{\text{def}}{=} \{ \sigma \mid \forall \sigma' \in \Sigma : \sigma \not\to \sigma' \}$,
- model both correct program termination and program errors, (correct exit, program stuck, unhandled exception, etc.)
- often include (or equal) final states $\mathcal{F}$.

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 $\mathcal{P}(\Sigma)$, is sufficient.

Post-image, pre-image

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

- successors: (forward, post-image)
  $$\mathrm{post}_\tau(S) \stackrel{\text{def}}{=} \{ \sigma' \mid \exists \sigma \in S : \sigma \to \sigma' \}$$
- predecessors: (backward, pre-image)
  $$\mathrm{pre}_\tau(S) \stackrel{\text{def}}{=} \{ \sigma \mid \exists \sigma' \in S : \sigma \to \sigma' \}$$

post_τ and pre_τ are complete $\cup-$morphisms in $(\mathcal{P}(\Sigma), \subseteq, \cup, \cap, \emptyset, \Sigma)$.
$(\mathrm{post}_\tau(\cup_{i\in I} S_i) = \cup_{i\in I} \mathrm{post}_\tau(S_i),\ \mathrm{pre}_\tau(\cup_{i\in I} S_i) = \cup_{i\in I} \mathrm{pre}_\tau(S_i))$

post_τ and pre_τ are strict. $(\mathrm{post}_\tau(\emptyset) = \mathrm{pre}_\tau(\emptyset) = \emptyset)$

Dual images

Dual post-images and pre-images:

- $\widetilde{\mathrm{pre}}_\tau(S) \stackrel{\text{def}}{=} \{\, \sigma \mid \forall \sigma' : \sigma \to \sigma' \implies \sigma' \in S \,\}$
  (states such that all successors satisfy S)

- $\widetilde{\mathrm{post}}_\tau(S) \stackrel{\text{def}}{=} \{\, \sigma' \mid \forall \sigma : \sigma \to \sigma' \implies \sigma \in S \,\}$
  (states such that all predecessors satisfy S)

$\widetilde{\mathrm{pre}}_\tau$ and $\widetilde{\mathrm{post}}_\tau$ are complete $\cap-$morphisms and not strict.

$\widetilde{\mathrm{post}}$ is not much used...

Correspondences between images and dual images

We have the following correspondences:

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

  $\widetilde{\mathrm{pre}}_\tau = \widetilde{\mathrm{post}}_{(\tau^{-1})}$ $\quad \widetilde{\mathrm{post}}_\tau = \widetilde{\mathrm{pre}}_{(\tau^{-1})}$

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

- Galois connections:

$$(\mathcal{P}(\Sigma), \subseteq) \underset{\mathrm{post}_\tau}{\overset{\widetilde{\mathrm{pre}}_\tau}{\leftrightarrows}} (\mathcal{P}(\Sigma), \subseteq) \text{ and}$$

$$(\mathcal{P}(\Sigma), \subseteq) \underset{\mathrm{pre}_\tau}{\overset{\widetilde{\mathrm{post}}_\tau}{\leftrightarrows}} (\mathcal{P}(\Sigma), \subseteq).$$

proof:

$\mathrm{post}_\tau(A) \subseteq B \iff \{ \sigma' \,|\, \exists \sigma \in A : \sigma \to \sigma' \} \subseteq B \iff (\forall \sigma \in A : \sigma \to \sigma' \implies \sigma' \in B) \iff (A \subseteq \{ \sigma \,|\, \forall \sigma' : \sigma \to \sigma' \implies \sigma' \in B \}) \iff A \subseteq \widetilde{\mathrm{pre}}_\tau(B)$;
other directions are similar.

Deterministic systems

Determinism:

- (Σ, τ) is deterministic if $\forall \sigma \in \Sigma : |\mathrm{post}_\tau(\{\sigma\})| = 1$,
  (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 τ is deterministic
  then $\mathrm{pre}_\tau = \widetilde{\mathrm{pre}}_\tau$ and $\mathrm{post}_\tau = \widetilde{\mathrm{post}}_\tau$.

Forward state reachability

Forward reachability

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

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

(reachable $\iff$ reachable from $\mathcal{I}$ in n steps of τ for some $n \geq 0$)

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

$$\mathcal{R}(\mathcal{I}) = \text{lfp } F_{\mathcal{R}} \text{ where } F_{\mathcal{R}}(S) \stackrel{\text{def}}{=} \mathcal{I} \cup \text{post}_\tau(S)$$

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

Alternate characterization: $\mathcal{R} = \text{lfp}_{\mathcal{I}}\ G_{\mathcal{R}}$ where $G_{\mathcal{R}}(S) \stackrel{\text{def}}{=} S \cup \text{post}_\tau(S)$.
($G_{\mathcal{R}}$ shifts S by τ and accumulates the result with S)

(proofs on next slide)

Forward reachability: proof

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

$(\mathcal{P}(\Sigma), \subseteq)$ is a CPO and post_τ is continuous, hence $F_{\mathcal{R}}$ is continuous: $F_{\mathcal{R}}(\cup_{i\in I} A_i) = \cup_{i\in I} F_{\mathcal{R}}(A_i)$.

By Kleene's theorem, $\mathrm{lfp}\, F_{\mathcal{R}} = \cup_{n\in\mathbb{N}} F^n_{\mathcal{R}}(\emptyset)$.

We prove by recurrence on n that: $\forall n\colon F^n_{\mathcal{R}}(\emptyset) = \cup_{i<n} \mathrm{post}^i_\tau(\mathcal{I})$.
(states reachable in less than n steps)

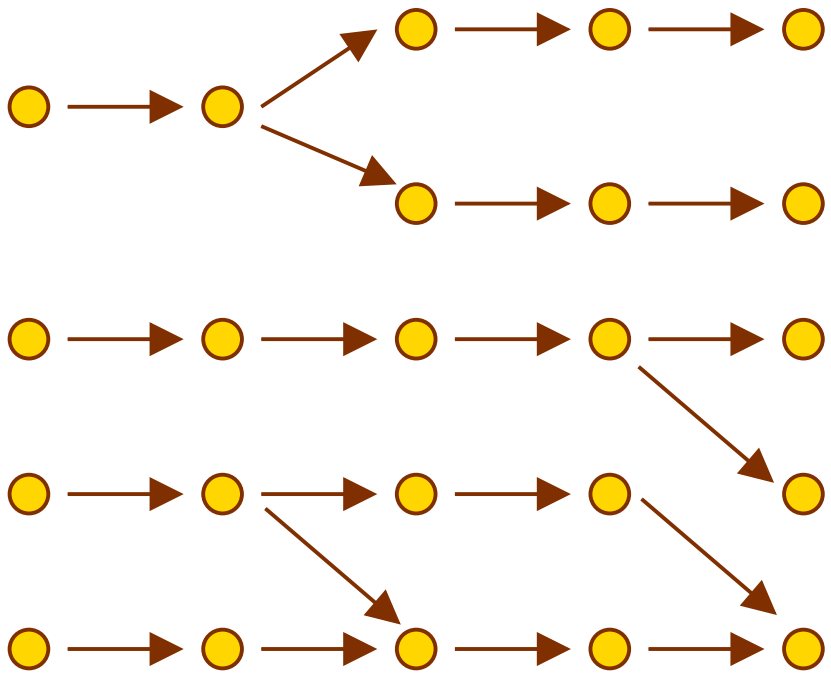
- $F^0_{\mathcal{R}}(\emptyset) = \emptyset$
- assuming the property at n,

$$\begin{aligned} F^{n+1}_{\mathcal{R}}(\emptyset) &= F_{\mathcal{R}}(\textstyle\bigcup_{i<n} \mathrm{post}^i_\tau(\mathcal{I})) \\ &= \mathcal{I} \cup \mathrm{post}_\tau(\textstyle\bigcup_{i<n} \mathrm{post}^i_\tau(\mathcal{I})) \\ &= \mathcal{I} \cup \textstyle\bigcup_{i<n} \mathrm{post}_\tau(\mathrm{post}^i_\tau(\mathcal{I})) \\ &= \mathcal{I} \cup \textstyle\bigcup_{1\le i<n+1} \mathrm{post}^i_\tau(\mathcal{I}) \\ &= \textstyle\bigcup_{i<n+1} \mathrm{post}^i_\tau(\mathcal{I}) \end{aligned}$$

Hence: $\mathrm{lfp}\, F_{\mathcal{R}} = \cup_{n\in\mathbb{N}} F^n_{\mathcal{R}}(\emptyset) = \cup_{i\in\mathbb{N}} \mathrm{post}^i_\tau(\mathcal{I}) = \mathcal{R}(\mathcal{I})$.

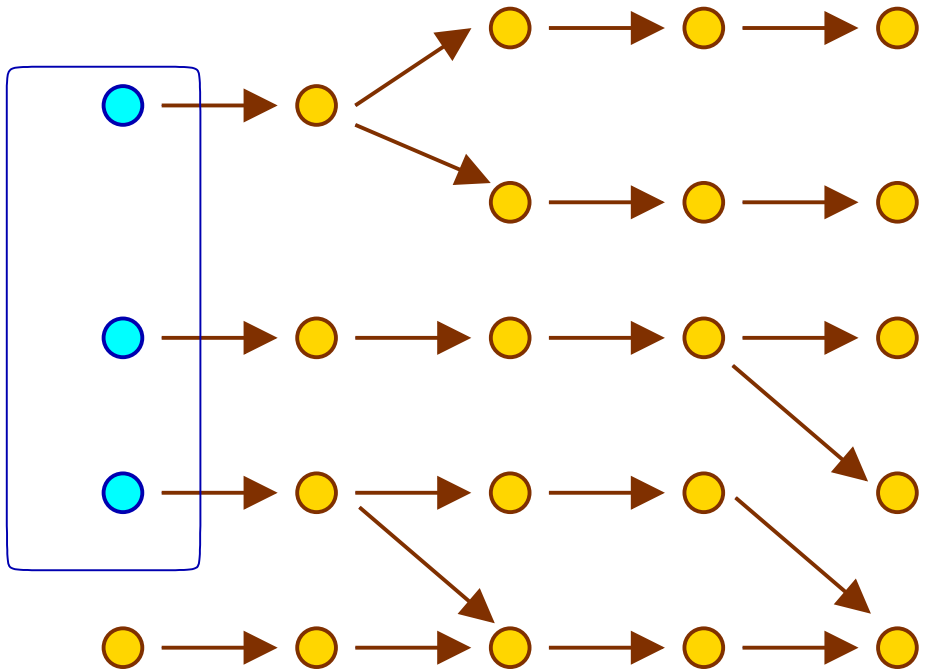
The proof is similar for the alternate form, given that $\mathrm{lfp}_{\mathcal{I}}\, G_{\mathcal{R}} = \cup_{n\in\mathbb{N}} G^n_{\mathcal{R}}(\mathcal{I})$ and $G^n_{\mathcal{R}}(\mathcal{I}) = F^{n+1}_{\mathcal{R}}(\emptyset) = \cup_{i\le n} \mathrm{post}^i_\tau(\mathcal{I})$.

Graphical illustration



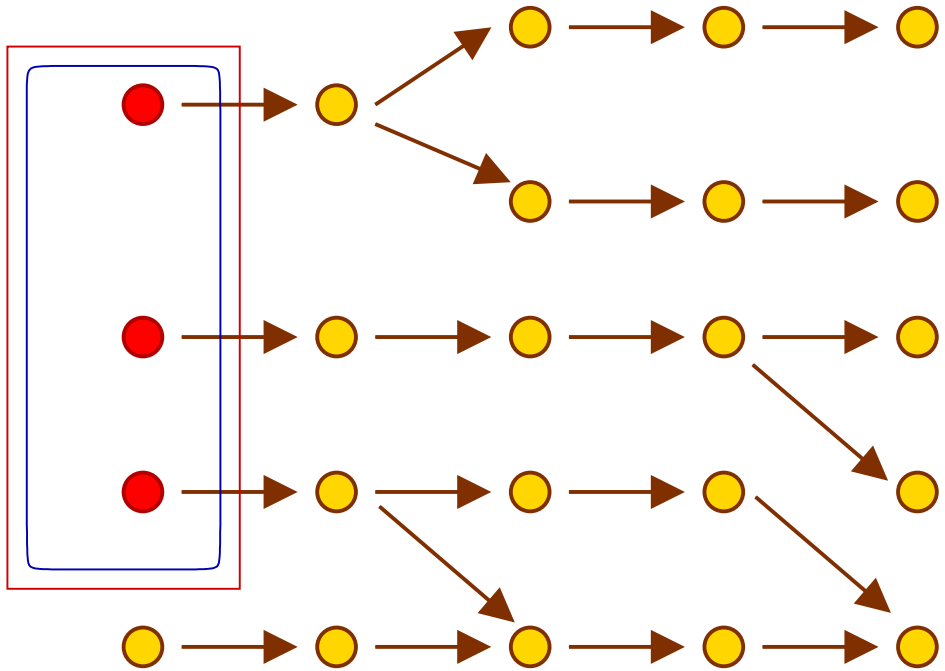
Transition system.

Graphical illustration



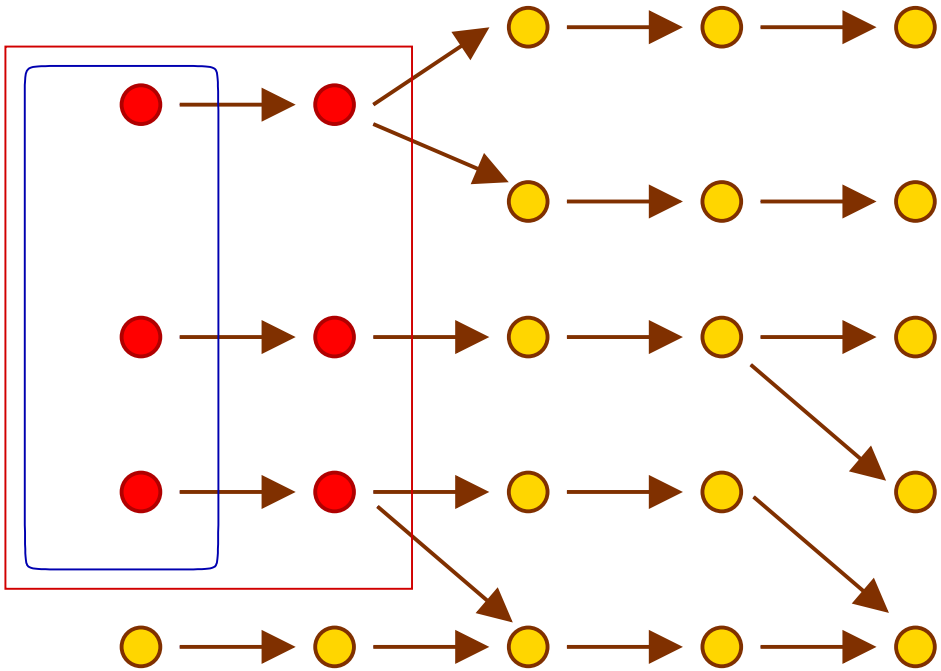
Initial states $\mathcal{I}$.

Graphical illustration



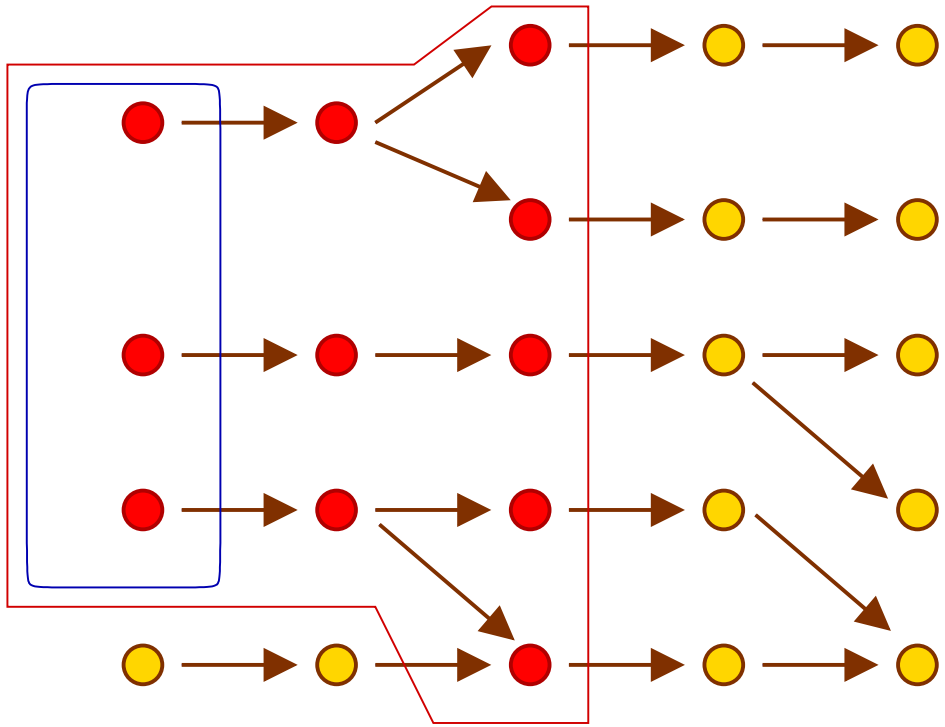
Iterate $F^1_{\mathcal{R}}(\mathcal{I})$.

Graphical illustration



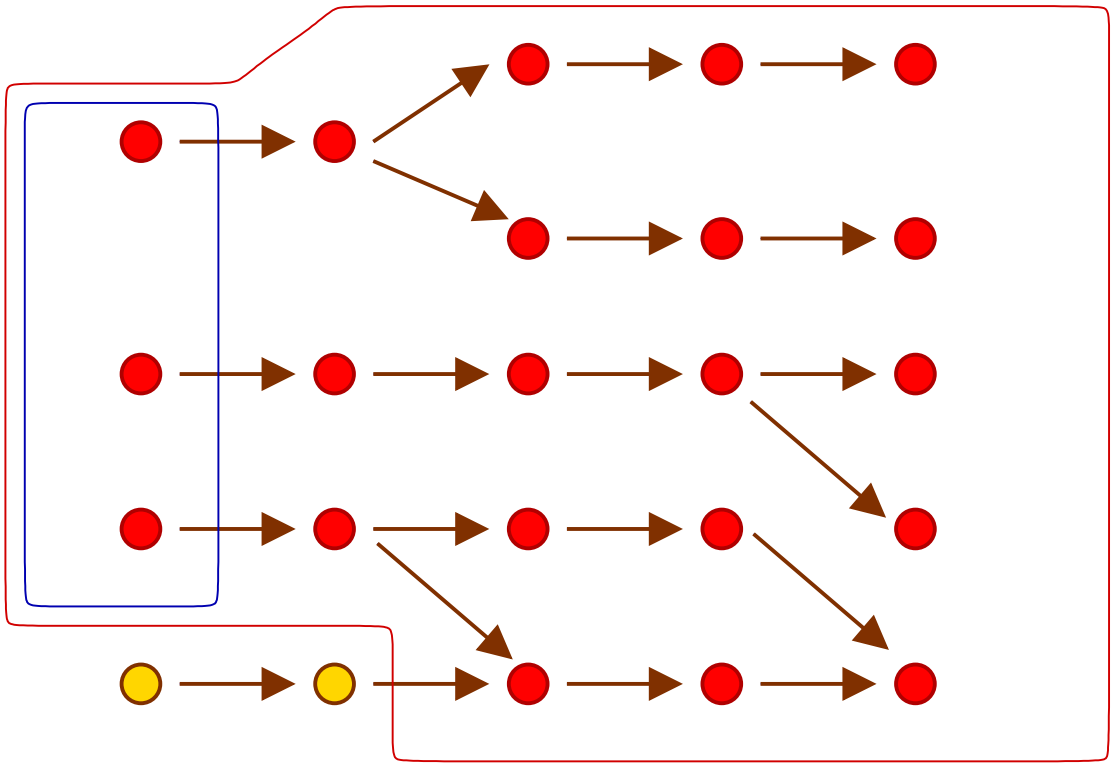
Iterate $F^2_{\mathcal{R}}(\mathcal{I})$.

Graphical illustration



Iterate $F^3_{\mathcal{R}}(\mathcal{I})$.

Graphical illustration



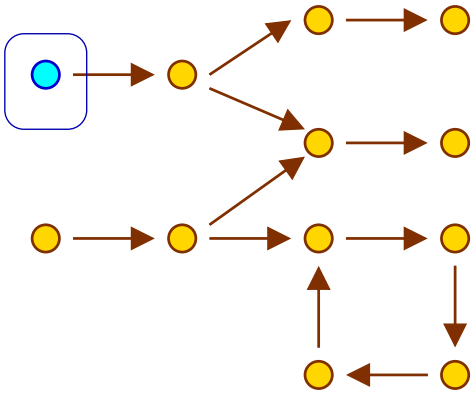
States reachable from $\mathcal{I}$: $\mathcal{R}(\mathcal{I}) = F^5_{\mathcal{R}}(\mathcal{I})$.

Multiple forward fixpoints

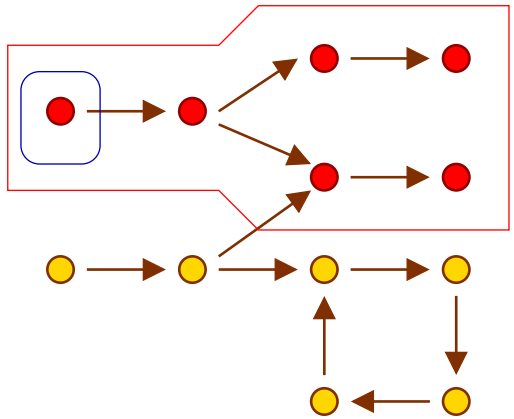
Recall: $\mathcal{R}(\mathcal{I}) = \text{lfp}\, F_{\mathcal{R}}$ where $F_{\mathcal{R}}(S) \stackrel{\text{def}}{=} \mathcal{I} \cup \text{post}_{\tau}(S)$.

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

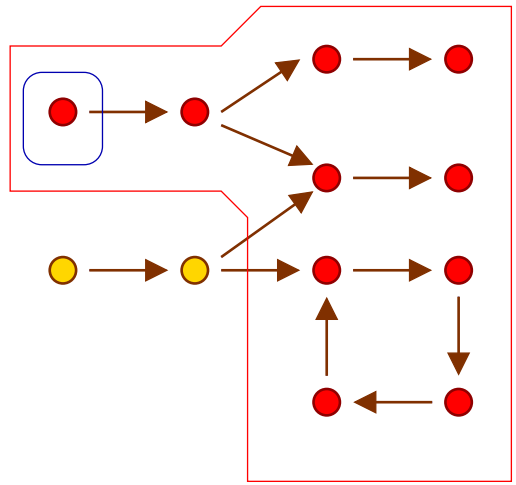
Example:



Initial state $\mathcal{I}$



$\mathcal{R}(\mathcal{I}) = \text{lfp}\, F_{\mathcal{R}}$



$\text{gfp}\, F_{\mathcal{R}}$

Exercise:

Compute all the fixpoints of $G_{\mathcal{R}}(S) \stackrel{\text{def}}{=} S \cup \text{post}_{\tau}(S)$ on this example.

Example application of forward reachability

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

example

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

  - initial states $\mathcal{I}$: $j \in [0, 10]$ at control state •,
  - final states $\mathcal{F}$: any memory state at control state •,
  - $\Longrightarrow \mathcal{R}(\mathcal{I}) \cap \mathcal{F}$: control at •, $i = 100$, and $j \in [0, 110]$.

- Prove the absence of run-time error: $\mathcal{R}(\mathcal{I}) \cap \mathcal{B} \subseteq \mathcal{F}$.
  (never block except when reaching the end of the program)
- To ensure soundness, over-approximations are sufficient.
  (if $\mathcal{R}^\sharp(\mathcal{I}) \supseteq \mathcal{R}(\mathcal{I})$, then $\mathcal{R}^\sharp(\mathcal{I}) \cap \mathcal{B} \subseteq \mathcal{F} \implies \mathcal{R}(\mathcal{I}) \cap \mathcal{B} \subseteq \mathcal{F}$)

Forward reachability in equational form

Idea:

- $\Sigma \stackrel{\text{def}}{=} \mathcal{L} \times \mathcal{E}$: 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:

$\ell 1$ $i \leftarrow 2$;
$\ell 2$ $n \leftarrow [-\infty, +\infty]$;
$\ell 3$ **while** $\ell 4$ $i < n$ **do**
  $\ell 5$ **if** $[0,1] = 0$ **then**
    $\ell 6$ $i \leftarrow i + 1$
  $\ell 7$
$\ell 8$

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

- initial states $\mathcal{I} \stackrel{\text{def}}{=} \{ (\ell 1, \rho) \mid \rho \in \mathcal{I}_1 \}$ for some $\mathcal{I}_1 \subseteq \mathcal{E}$,
- $\mathsf{C}[\![\cdot]\!] : \mathcal{P}(\mathcal{E}) \to \mathcal{P}(\mathcal{E})$ model assignments and tests (see next slide).
- We get the **strongest invariant** at each program point.

Systematic construction of the equation system

Atomic instructions:

- $\mathsf{C}[\![X \leftarrow e]\!]\,\mathcal{X} \stackrel{\text{def}}{=} \{\, \rho[X \mapsto v] \mid \rho \in \mathcal{X},\ v \in \mathsf{E}[\![e]\!]\,\rho \,\}$
- $\mathsf{C}[\![e \bowtie 0]\!]\,\mathcal{X} \stackrel{\text{def}}{=} \{\, \rho \in \mathcal{X} \mid \exists v \in \mathsf{E}[\![\rho]\!]\,\rho\colon v \bowtie 0 \,\}$

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

$$eq(^{\ell 1}X \leftarrow e^{\ell 2}) \stackrel{\text{def}}{=} \{\, \mathcal{X}_{\ell 2} = \mathsf{C}[\![X \leftarrow e]\!]\,\mathcal{X}_{\ell 1} \,\}$$

$$eq(^{\ell 1}\textbf{if } e \bowtie 0 \textbf{ then } ^{\ell 2}s^{\ell 3}) \stackrel{\text{def}}{=} \{\, \mathcal{X}_{\ell 2} = \mathsf{C}[\![e \bowtie 0]\!]\,\mathcal{X}_{\ell 1},\ \mathcal{X}_{\ell 3} = \mathcal{X}_{\ell 3'} \cup \mathsf{C}[\![e \not\bowtie 0]\!]\,\mathcal{X}_{\ell 1} \,\} \cup eq(^{\ell 2}s^{\ell 3'})$$

$$eq(^{\ell 1}\textbf{while } ^{\ell 2}e \bowtie 0 \textbf{ do } ^{\ell 3}s^{\ell 4}) \stackrel{\text{def}}{=} \{\, \mathcal{X}_{\ell 2} = \mathcal{X}_{\ell 1} \cup \mathcal{X}_{\ell 4'},\ \mathcal{X}_{\ell 3} = \mathsf{C}[\![e \bowtie 0]\!]\,\mathcal{X}_{\ell 2},\ \mathcal{X}_{\ell 4} = \mathsf{C}[\![e \not\bowtie 0]\!]\,\mathcal{X}_{\ell 2} \,\} \cup eq(^{\ell 3}s^{\ell 4'})$$

$$eq(^{\ell 1}s_1;^{\ell 2}s_2{}^{\ell 3}) \stackrel{\text{def}}{=} eq(^{\ell 1}s_1{}^{\ell 2}) \cup (^{\ell 2}s_2{}^{\ell 3})$$

where: $\mathcal{X}^{\ell 3'}$, $\mathcal{X}^{\ell 4'}$ are fresh variables storing intermediate results

$\cup$−morphisms in a complete lattice $\Longrightarrow$ a smallest solution exists

Link between the fixpoint and equational presentation

By partitioning forward reachability wrt. control states,
we retrieve the equation system form of program semantics.

Control state partitioning

We assume $\Sigma \stackrel{\text{def}}{=} \mathcal{L} \times \mathcal{E}$; note that: $\mathcal{P}(\Sigma) \simeq \mathcal{L} \to \mathcal{P}(\mathcal{E})$.

We have a Galois isomorphism:

$$(\mathcal{P}(\Sigma), \subseteq) \underset{\alpha_{\mathcal{L}}}{\overset{\gamma_{\mathcal{L}}}{\leftrightarrows}} (\mathcal{L} \to \mathcal{P}(\mathcal{E}), \dot{\subseteq})$$

- $X \dot{\subseteq} Y \iff^{\text{def}} \forall \ell \in \mathcal{L}: X(\ell) \subseteq Y(\ell)$
- $\alpha_{\mathcal{L}}(S) \stackrel{\text{def}}{=} \lambda \ell.\{\, \rho \mid (\ell, \rho) \in S \,\}$
- $\gamma_{\mathcal{L}}(X) \stackrel{\text{def}}{=} \{\, (\ell, \rho) \mid \ell \in \mathcal{L}, \rho \in X(\ell) \,\}$
- given $F_{eq} \stackrel{\text{def}}{=} \alpha_{\mathcal{L}} \circ F_{\mathcal{R}} \circ \gamma_{\mathcal{L}}$
  we get an equation system $\forall \ell \in \mathcal{L}: \mathcal{X}_\ell = F_{eq,\ell}(\mathcal{X}_1, \ldots, \mathcal{X}_n)$
- 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 $I \subseteq \Sigma$

- $\mathcal{I} \subseteq I$
  (contains initial states)
- $\forall \sigma \in I : \sigma \to \sigma' \implies \sigma' \in I$
  (invariant by program transition)

that implies the desired property: $I \subseteq P$.

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

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

Recall that $\mathcal{R}(\mathcal{I}) = \text{lfp}\, F_{\mathcal{R}}$
$\implies$ $\mathcal{R}(\mathcal{I})$ is the tightest inductive invariant.

Link with Hoare logic

Hoare logic: proof method where we

- annotate program points with local sate invariants in $\mathcal{P}(\Sigma)$
- use logic rules to prove their correctness

$$\frac{}{\{P[e/X]\}\, X \leftarrow e\, \{P\}} \qquad \frac{\{P\}\, stat_1\, \{R\} \quad \{R\}\, stat_2\, \{Q\}}{\{P\}\, stat_1; stat_2\, \{Q\}}$$

$$\frac{\{P \wedge b\}\, stat\, \{Q\} \quad P \wedge \neg b \Rightarrow Q}{\{P\}\, \textbf{if}\ b\ \textbf{then}\ stat\, \{Q\}} \qquad \frac{\{P \wedge b\}\, stat\, \{P\}}{\{P\}\, \textbf{while}\ b\ \textbf{do}\ stat\, \{P \wedge \neg b\}}$$

$$\frac{\{P\}\, stat\, \{Q\} \quad P' \Rightarrow P \quad Q \Rightarrow Q'}{\{P'\}\, stat\, \{Q'\}}$$

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

Recall the equation system $\forall \ell \in \mathcal{L}: \mathcal{X}_\ell = F_{eq,\ell}(\mathcal{X}_1, \ldots, \mathcal{X}_n)$ obtained by partitioning reachability $F_{\mathcal{R}}$ by control point $(\mathcal{P}(\Sigma), \subseteq) \underset{\alpha_{\mathcal{L}}}{\overset{\gamma_{\mathcal{L}}}{\leftrightarrows}} (\mathcal{L} \to \mathcal{P}(\mathcal{E}), \dot{\subseteq})$.

- 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:

$$\mathcal{C}(\mathcal{F}) \stackrel{\text{def}}{=} \{\, \sigma \mid \exists n \geq 0, \sigma_0, \ldots, \sigma_n : \sigma = \sigma_0, \sigma_n \in \mathcal{F}, \forall i : \sigma_i \to \sigma_{i+1} \,\}$$
$$= \bigcup_{n \geq 0} \mathrm{pre}_\tau^n(\mathcal{F})$$

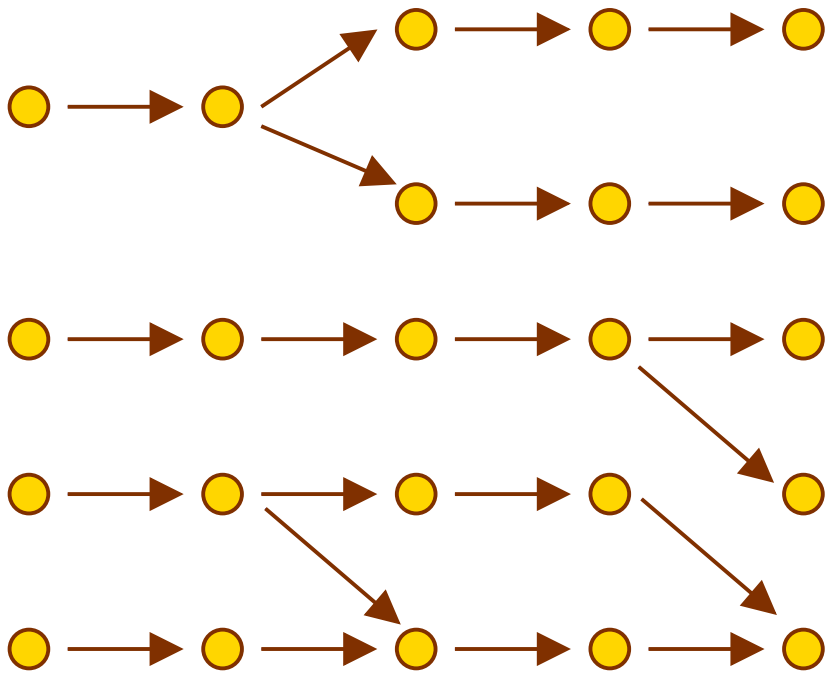
$\mathcal{C}(\mathcal{F})$ can also be expressed in fixpoint form:

$$\mathcal{C}(\mathcal{F}) = \mathrm{lfp}\, F_{\mathcal{C}} \text{ where } F_{\mathcal{C}}(S) \stackrel{\text{def}}{=} \mathcal{F} \cup \mathrm{pre}_\tau(S)$$

Alternate characterization: $\mathcal{C}(\mathcal{F}) = \mathrm{lfp}_{\mathcal{F}}\ G_{\mathcal{C}}$ where $G_{\mathcal{C}}(S) = S \cup \mathrm{pre}_\tau(S)$

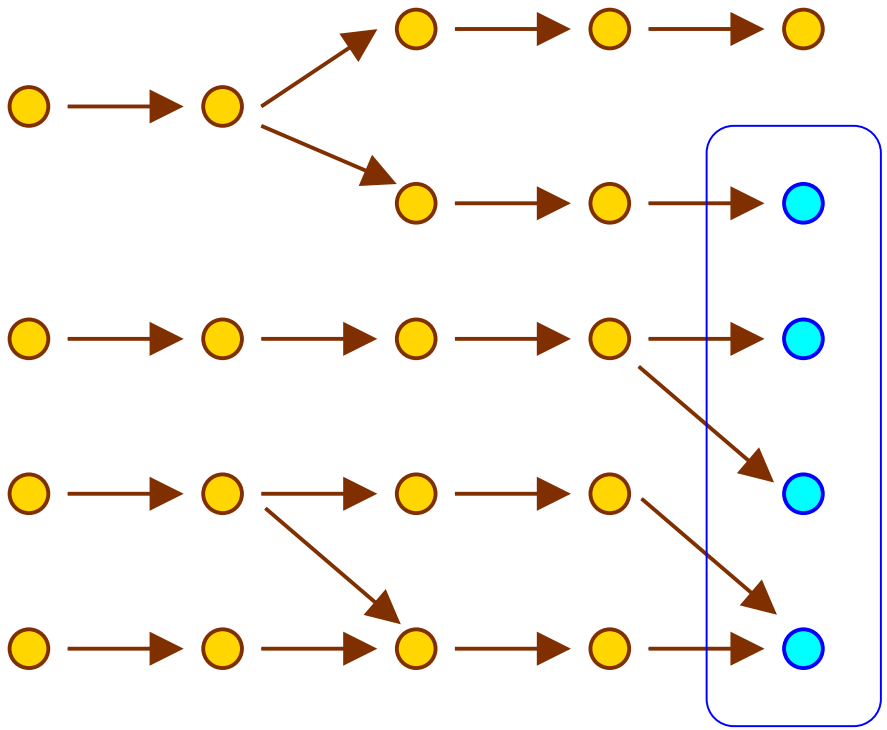
Justification: $\mathcal{C}(\mathcal{F})$ in τ is exactly $\mathcal{R}(\mathcal{F})$ in τ^{-1}.

Graphical illustration



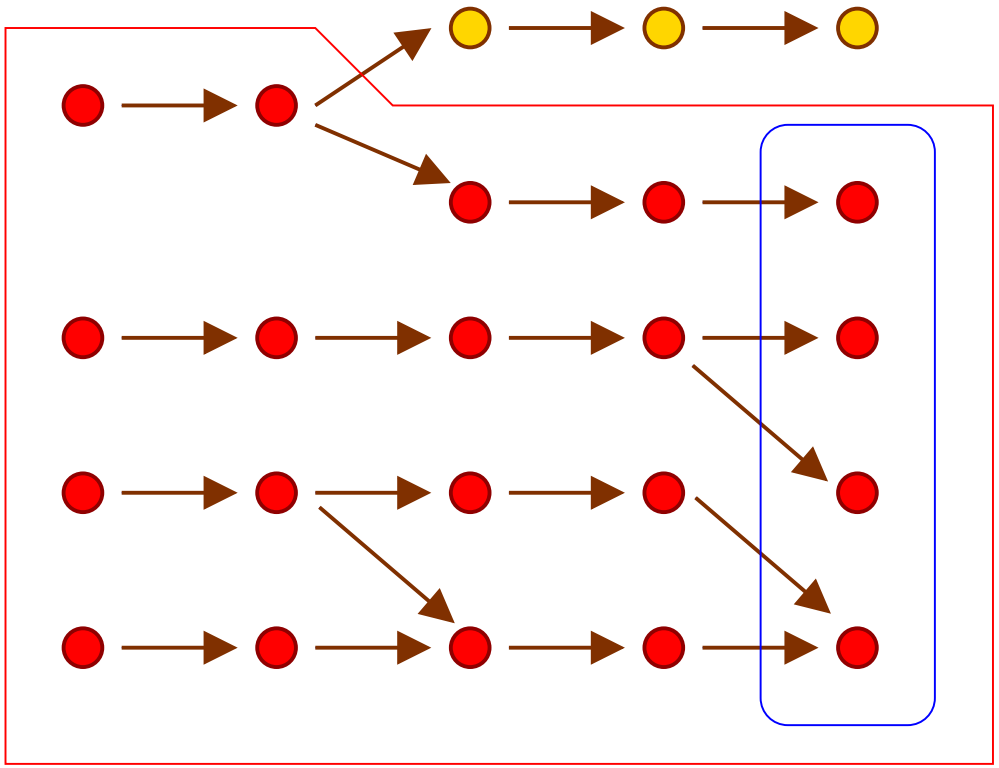
Transition system.

Graphical illustration



Final states $\mathcal{F}$.

Graphical illustration



States co-reachable from $\mathcal{F}$.

Application of backward co-reachability

- $\mathcal{I} \cap \mathcal{C}(\mathcal{B} \setminus \mathcal{F})$
  Initial states that have at least one erroneous execution.

program

- $j \leftarrow 0;$
  while $i > 0$ **do**
  $i \leftarrow i - 1;$
  $j \leftarrow j + [0, 10]$
  done •

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

- $\mathcal{I} \cap (\Sigma \setminus \mathcal{C}(\mathcal{B}))$
  Initial states that necessarily cause the program to loop.
- Over-approximating $\mathcal{C}$ is useful to isolate possibly incorrect executions from those guaranteed to be correct.
- Iterate forward and backward analyses interactively
  $\Longrightarrow$ 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:

$\ell 1$ $i \leftarrow 2$;
$\ell 2$ $n \leftarrow [-\infty, +\infty]$;
$\ell 3$ **while** $\ell 4$ $i < n$ **do**
  $\ell 5$ **if** $[0,1] = 0$ **then**
    $\ell 6$ $i \leftarrow i + 1$
  $\ell 7$
$\ell 8$

$$
\begin{aligned}
\mathcal{X}_1 &= \mathsf{C}[\![\, i \to 2 \,]\!]\, \mathcal{X}_2 \\
\mathcal{X}_2 &= \mathsf{C}[\![\, n \to [-\infty, +\infty] \,]\!]\, \mathcal{X}_3 \\
\mathcal{X}_3 &= \mathcal{X}_4 \\
\mathcal{X}_4 &= \mathsf{C}[\![\, i < n \,]\!]\, \mathcal{X}_5 \cup \mathsf{C}[\![\, i \geq n \,]\!]\, \mathcal{X}_8 \\
\mathcal{X}_5 &= \mathcal{X}_6 \cup \mathcal{X}_7 \\
\mathcal{X}_6 &= \mathsf{C}[\![\, i \to i + 1 \,]\!]\, \mathcal{X}_7 \\
\mathcal{X}_7 &= \mathcal{X}_4 \\
\mathcal{X}_8 &= \mathcal{F}_8
\end{aligned}
$$

- final states $\mathcal{F} \stackrel{\text{def}}{=} \{\, (\ell 8, \rho) \mid \rho \in \mathcal{F}_8 \,\}$ for some $\mathcal{F}_8 \subseteq \mathcal{E}$,
- $\mathsf{C}[\![\, X \to e \,]\!]\, \mathcal{X} \stackrel{\text{def}}{=} \{\, \rho \mid \exists v \in \mathsf{E}[\![\, e \,]\!]\, \rho \colon \rho[X \mapsto v] \in X \,\}$.

Backward sufficient precondition state semantics

Sufficient preconditions

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

$$\mathcal{S}(\mathcal{Y}) \stackrel{\text{def}}{=} \{\, \sigma \mid \forall n \geq 0, \sigma_0, \ldots, \sigma_n : (\sigma = \sigma_0 \wedge \forall i : \sigma_i \to \sigma_{i+1}) \implies \sigma_n \in \mathcal{Y} \,\}$$
$$= \bigcap_{n \geq 0} \widetilde{\text{pre}}^n_\tau(\mathcal{Y})$$

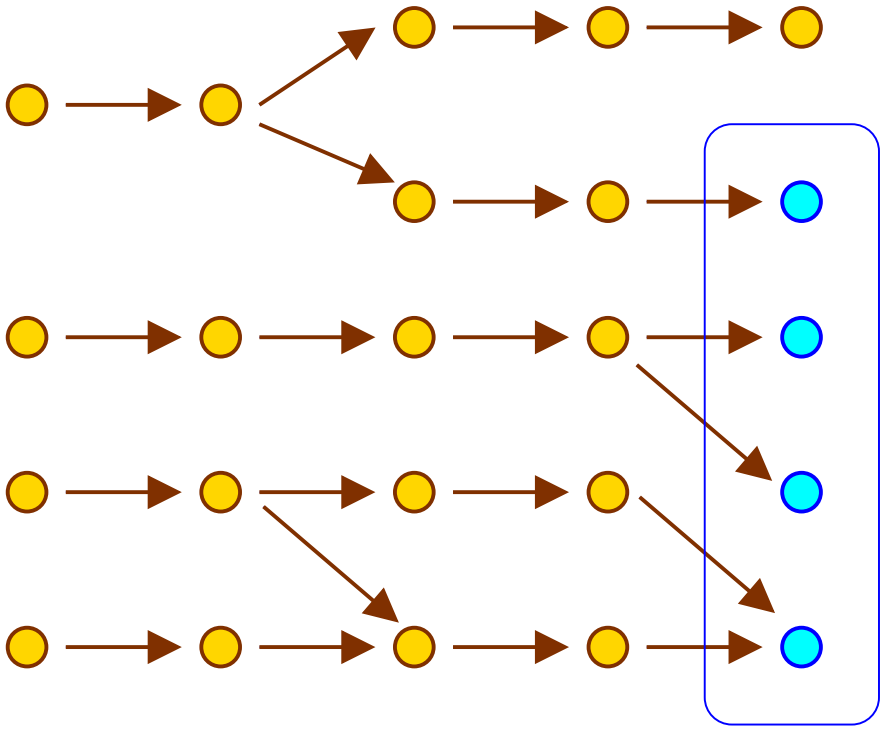
$\mathcal{S}(\mathcal{Y})$ can be expressed in fixpoint form:

$$\mathcal{S}(\mathcal{Y}) = \text{gfp}\, F_{\mathcal{S}} \text{ where } F_{\mathcal{S}}(S) \stackrel{\text{def}}{=} \mathcal{Y} \cap \widetilde{\text{pre}}_\tau(S)$$

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

$F_{\mathcal{S}}$ is continuous in the dual CPO $(\mathcal{P}(\Sigma), \supseteq)$, because $\widetilde{\text{pre}}_\tau$ is:
$F_{\mathcal{S}}(\cap_{i \in I} A_i) = \cap_{i \in I} F_{\mathcal{S}}(A_i)$.
By Kleene's theorem in the dual, $\text{gfp}\, F_{\mathcal{S}} = \cap_{n \in \mathbb{N}} F^n_{\mathcal{S}}(\Sigma)$.
We would prove by recurrence that $F^n_{\mathcal{S}}(\Sigma) = \cap_{i < n} \widetilde{\text{pre}}^i_\tau(\mathcal{Y})$.

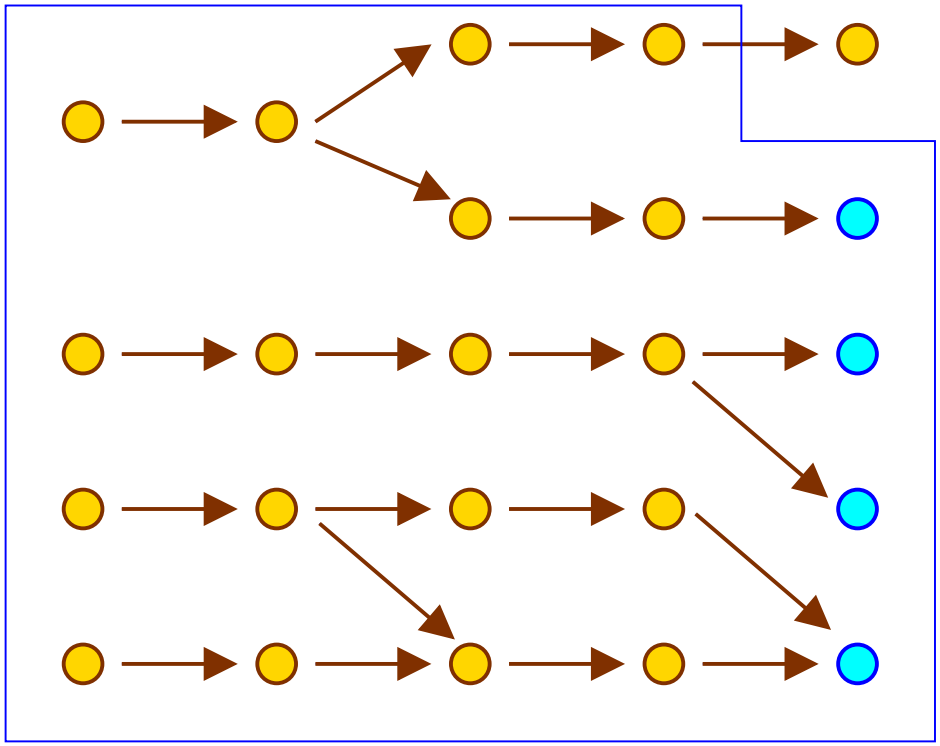
Graphical illustration



Final states $\mathcal{F}$.

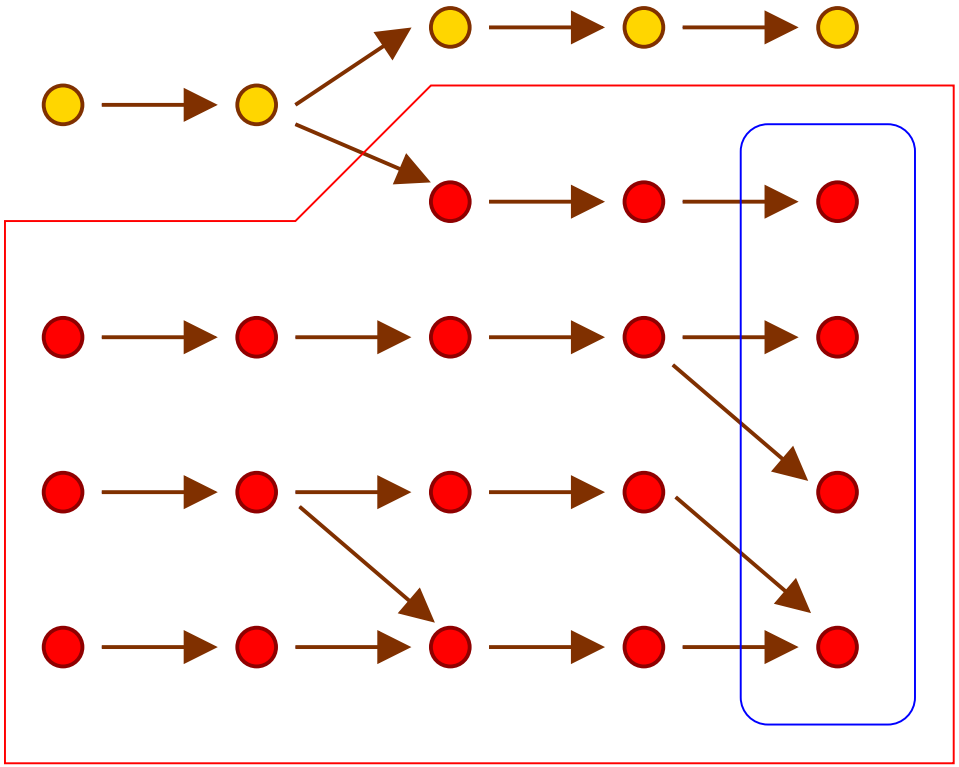
Goal: when stopping, stop in $\mathcal{F}$.

Graphical illustration



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

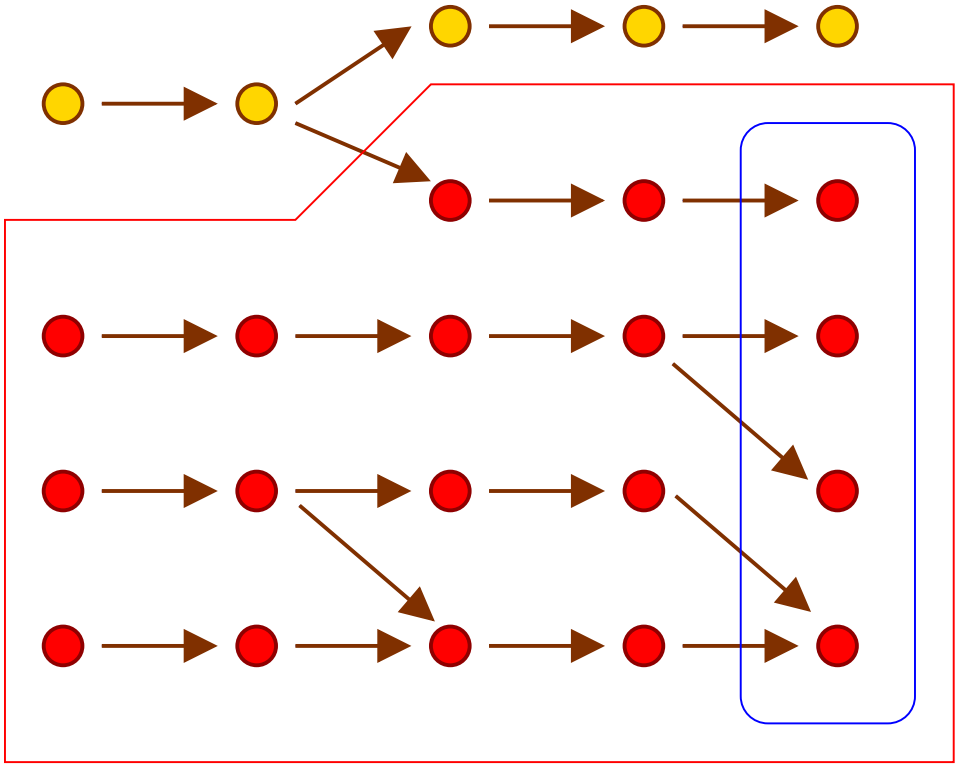
Graphical illustration

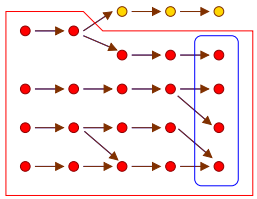


Sufficient preconditions $\mathcal{S}(\mathcal{Y})$ to stop in $\mathcal{F}$.

(without forcing the program to stop at all)

Graphical illustration





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

(without forcing the program to stop at all)

$\mathcal{C}(\mathcal{F})$

Note: $\mathcal{S}(\mathcal{Y}) \subset \mathcal{C}(\mathcal{F})$

Sufficient preconditions and reachability

Correspondence with reachability:

We have a Galois connection:

$$(\mathcal{P}(\Sigma), \subseteq) \underset{\mathcal{R}}{\overset{\mathcal{S}}{\leftrightarrows}} (\mathcal{P}(\Sigma), \subseteq)$$

- $\mathcal{R}(\mathcal{I}) \subseteq \mathcal{Y} \iff \mathcal{I} \subseteq \mathcal{S}(\mathcal{Y})$
  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 $\mathcal{S}(\mathcal{Y}) = \bigcup \{ X \mid \mathcal{R}(X) \subseteq \mathcal{Y} \}$
  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)

Sufficient preconditions and reachability (proof)

proof sketch:

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

Recall the Galois connection $(\mathcal{P}(\Sigma), \subseteq) \underset{\mathrm{post}_\tau}{\overset{\widetilde{\mathrm{pre}}_\tau}{\leftrightarrows}} (\mathcal{P}(\Sigma), \subseteq)$.

As a consequence $(\mathcal{P}(\Sigma), \subseteq) \underset{G_{\mathcal{R}}}{\overset{G_{\mathcal{S}}}{\leftrightarrows}} (\mathcal{P}(\Sigma), \subseteq)$.

The Galois connection can be lifted to fixpoint operators:

$(\mathcal{P}(\Sigma), \subseteq) \underset{x \mapsto \mathrm{lfp}_x\, G_{\mathcal{R}}}{\overset{x \mapsto \mathrm{gfp}_x\, G_{\mathcal{S}}}{\leftrightarrows}} (\mathcal{P}(\Sigma), \subseteq)$.

Exercise: complete the proof sketch.

Application of sufficient preconditions

Initial states such that all executions are correct:
$\mathcal{I} \cap \mathcal{S}(\mathcal{F} \cup (\Sigma \setminus \mathcal{B}))$.

(the only blocking states reachable from initial states are final states)

program

- $i \leftarrow 0;$
  while $i < 100$ **do**
    $i \leftarrow i + 1;$
    $j \leftarrow j + [0, 1]$
  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
- $\mathcal{I} \cap \mathcal{S}(\mathcal{F} \cup (\Sigma \setminus \mathcal{B}))$: at •, $j \in [0, 5]$
  (note that $\mathcal{I} \cap \mathcal{C}(\mathcal{F} \cup (\Sigma \setminus \mathcal{B}))$ gives $\mathcal{I}$)

⋆⋆ Research topic ⋆⋆

Inferring sound sufficient preconditions requires under-approximations.

if $\mathcal{S}(\mathcal{X})$ is a sufficient precondition, any $\mathcal{S}^\sharp(\mathcal{X}) \subset \mathcal{S}(\mathcal{X})$ 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 (`$\mathcal{X}$`) then for i in [0,n] unsafe-get(a,i)`
  `else for i in [0,n] get(a,i)`
  where $\mathcal{X}$ 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

Sequences, traces

Trace: sequence of elements from Σ

- ϵ: empty trace (unique)
- σ: 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 ω)

Trace sets:

- Σ^n: the set of traces of length n
- $\Sigma^{\leq n} \stackrel{\text{def}}{=} \cup_{i \leq n} \Sigma^i$: the set of traces of length at most n
- $\Sigma^* \stackrel{\text{def}}{=} \cup_{i \in \mathbb{N}} \Sigma^i$: the set of finite traces
- Σ^ω: the set of infinite traces
- $\Sigma^\infty \stackrel{\text{def}}{=} \Sigma^* \cup \Sigma^\omega$: the set of all traces

Traces of a transition system

Execution traces:

Non-empty sequences of states linked by the transition relation τ.

- can be finite (in $\mathcal{P}(\Sigma^*)$) or infinite (in $\mathcal{P}(\Sigma^\omega)$)
- can be anchored at initial states, or final states, or none

Atomic traces:

- $\mathcal{I}$: initial states $\simeq$ set of traces of length 1
- $\mathcal{F}$: final states $\simeq$ set of traces of length 1
- τ: transition relation $\simeq$ set of traces of length 2
  $(\{\, \sigma, \sigma' \mid \sigma \to \sigma' \,\})$

(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 $t \in \Sigma^\infty$
- concatenation $\cdot$
  - $(\sigma_0, \ldots, \sigma_n) \cdot (\sigma'_0, \ldots) \stackrel{\text{def}}{=} \sigma_0, \ldots, \sigma_n, \sigma'_0, \ldots$
    (append to a finite trace)
  - $t \cdot t' \stackrel{\text{def}}{=} t$ if $t \in \Sigma^\omega$ (append to an infinite trace does nothing)
  - $\epsilon \cdot t \stackrel{\text{def}}{=} t \cdot \epsilon \stackrel{\text{def}}{=} t$ (ϵ is neutral)
- junction $\frown$
  - $(\sigma_0, \ldots, \sigma_n) \frown (\sigma'_0, \sigma'_1 \ldots) \stackrel{\text{def}}{=} \sigma_0, \ldots, \sigma_n, \sigma'_1, \ldots$ when $\sigma_n = \sigma'_0$
    undefined if $\sigma_n \neq \sigma'_0$
  - $\epsilon \frown t$ and $t \frown \epsilon$ are undefined
  - $t \frown t' \stackrel{\text{def}}{=} t$, if $t \in \Sigma^\omega$

Trace operations (cont.)

Extension to sets of traces:

- $A \cdot B \stackrel{\text{def}}{=} \{ a \cdot b \mid a \in A,\ b \in B \}$
  $\{\epsilon\}$ is the neutral element for $\cdot$
- $A^\frown B \stackrel{\text{def}}{=} \{ a^\frown b \mid a \in A,\ b \in B,\ a^\frown b \text{ defined} \}$
  Σ is the neutral element for $^\frown$

$$
\begin{array}{llll}
A^0 & \stackrel{\text{def}}{=} & \{\epsilon\} & A^{\frown 0} \stackrel{\text{def}}{=} \Sigma \\
A^{n+1} & \stackrel{\text{def}}{=} & A \cdot A^n & A^{\frown n+1} \stackrel{\text{def}}{=} A^\frown A^{\frown n} \\
A^\omega & \stackrel{\text{def}}{=} & A \cdot A \cdot \ldots & A^{\frown \omega} \stackrel{\text{def}}{=} A^\frown A^\frown \ldots \\
A^* & \stackrel{\text{def}}{=} & \cup_{n<\omega} A^n & A^{\frown *} \stackrel{\text{def}}{=} \cup_{n<\omega} A^{\frown n} \\
A^\infty & \stackrel{\text{def}}{=} & \cup_{n\leq\omega} A^n & A^{\frown \infty} \stackrel{\text{def}}{=} \cup_{n\leq\omega} A^{\frown n}
\end{array}
$$

Note: $A^n \neq \{ a^n \mid a \in A \}$, $A^{\frown n} \neq \{ a^{\frown n} \mid a \in A \}$ when $|A| > 1$

Distributivity of junction

- $\frown$ distributes over finite and infinite $\cup$:
  $A \frown (\cup_{i \in I} B_i) = \cup_{i \in I} (A \frown B_i)$ and
  $(\cup_{i \in I} A_i) \frown B = \cup_{i \in I} (A_i \frown B)$
  where I can be finite or infinite.

- $\frown$ distributes finite $\cap$ but not infinite $\cap$
  example:
  $\{a^\omega\} \frown (\cap_{n \in \mathbb{N}} \{ a^m \mid n \geq m \}) = \{a^\omega\} \frown \emptyset = \emptyset$ but
  $\cap_{n \in \mathbb{N}} (\{a^\omega\} \frown \{ a^m \mid n \geq m \}) = \cap_{n \in \mathbb{N}} \{a^\omega\} = \{a^\omega\}$

- but, if $A \subseteq \Sigma^*$, then $A \frown (\cap_{i \in I} B_i) = \cup_{i \in I} (A \frown B_i)$
  even for infinite I

Note: concatenation $\cdot$ distributes infinite $\cap$ and $\cup$.

Finite prefix trace semantics

Prefix trace semantics

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

$$\mathcal{T}_p(\mathcal{I}) \stackrel{\text{def}}{=} \{\, \sigma_0, \ldots, \sigma_n \mid n \geq 0, \sigma_0 \in \mathcal{I}, \forall i: \sigma_i \to \sigma_{i+1} \,\}$$
$$= \bigcup_{n \geq 0} \mathcal{I}^\frown(\tau^{\frown n})$$

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

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

$$\mathcal{T}_p(\mathcal{I}) = \text{lfp}\, F_p \text{ where } F_p(T) \stackrel{\text{def}}{=} \mathcal{I} \cup T^\frown\tau$$

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

(proof on next slide)

Prefix trace semantics: proof

proof of: $\mathcal{T}_p(\mathcal{I}) = \text{lfp}\, F_p$ where $F_p(T) = \mathcal{I} \cup T^\frown\tau$

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

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

$$\begin{array}{rl} & F_p(\cup_{i\in I}\, T_i) \\ = & \mathcal{I} \cup (\cup_{i\in I}\, T_i)^\frown\tau \\ = & \mathcal{I} \cup (\cup_{i\in I}\, T_i^\frown\tau) = \cup_{i\in I}\, (\mathcal{I} \cup T_i^\frown\tau) \end{array}$$

hence (Kleene), $\text{lfp}\, F_p = \cup_{n\geq 0}\, F_p^i(\emptyset)$

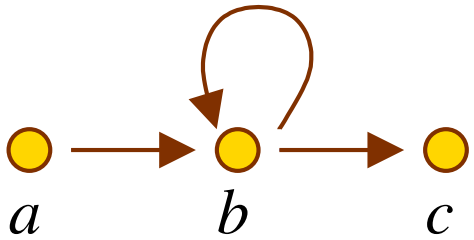
We prove by recurrence on n that $\forall n\text{: } F_p^n(\emptyset) = \cup_{i<n}\, \mathcal{I}^\frown\tau^{\frown i}$:

- $F_p^0(\emptyset) = \emptyset$,
- $$\begin{array}{rl} & F_p^{n+1}(\emptyset) \\ = & \mathcal{I} \cup F_p^n(\emptyset)^\frown\tau \\ = & \mathcal{I} \cup (\cup_{i<n}\, \mathcal{I}^\frown\tau^{\frown i})^\frown\tau \\ = & \mathcal{I} \cup \cup_{i<n}\, (\mathcal{I}^\frown\tau^{\frown i})^\frown\tau \\ = & \mathcal{I}^\frown\tau^{\frown 0} \cup \cup_{i<n}\, (\mathcal{I}^\frown\tau^{\frown i+1}) \\ = & \cup_{i<n+1}\, \mathcal{I}^\frown\tau^{\frown i} \end{array}$$

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

Note: we also have $\mathcal{T}_p(\mathcal{I}) = \text{lfp}_\mathcal{I}\, G_p$ where $G_p(T) = T \cup T^\frown\tau$.

Prefix trace semantics: graphical illustration



$$\mathcal{I} \stackrel{\text{def}}{=} \{a\}$$
$$\tau \stackrel{\text{def}}{=} \{(a,b),(b,b),(b,c)\}$$

Iterates: $\mathcal{T}_p(\mathcal{I}) = \text{lfp}\, F_p$ where $F_p(T) \stackrel{\text{def}}{=} \mathcal{I} \cup T^\frown\tau$.

- $F_p^0(\emptyset) = \emptyset$
- $F_p^1(\emptyset) = \mathcal{I} = \{a\}$
- $F_p^2(\emptyset) = \{a, ab\}$
- $F_p^3(\emptyset) = \{a, ab, abb, abc\}$
- $F_p^n(\emptyset) = \{\, a, ab^i, ab^j c \mid i \in [1, n-1], j \in [1, n-2] \,\}$
- $\mathcal{T}_p(\mathcal{I}) = \cup_{n \geq 0} F_p^n(\emptyset) = \{\, a, ab^i, ab^i c \mid i \geq 1 \,\}$

Prefix trace semantics: expressive power

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

$\Longrightarrow$ Semantics of testing.

Limitations:

- no information on infinite executions,
  (we will add infinite traces later)
- can bound maximal execution time: $\mathcal{T}_p(\mathcal{I}) \subseteq \Sigma^{\leq n}$
  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:

$$(\mathcal{P}(\Sigma^*), \subseteq) \underset{\alpha_p}{\overset{\gamma_p}{\leftrightarrows}} (\mathcal{P}(\Sigma), \subseteq)$$

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

(proof on next slide)

Abstracting traces into states (proof)

proof of: (α_p, γ_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: α and γ are monotonic, $\gamma \circ \alpha$ is extensive, and $\alpha \circ \gamma$ is reductive.
Embedding means that, additionally, $\alpha \circ \gamma = id$.

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

Abstracting prefix traces into reachability

We can abstract semantic operators and their least fixpoint.

Recall that:

- $\mathcal{T}_p(\mathcal{I}) = \mathrm{lfp}\, F_p$ where $F_p(T) \stackrel{\text{def}}{=} \mathcal{I} \cup T \frown \tau$,
- $\mathcal{R}(\mathcal{I}) = \mathrm{lfp}\, F_{\mathcal{R}}$ where $F_{\mathcal{R}}(S) \stackrel{\text{def}}{=} \mathcal{I} \cup \mathrm{post}_\tau(S)$,
- $(\mathcal{P}(\Sigma^*), \subseteq) \underset{\alpha_p}{\overset{\gamma_p}{\leftrightarrows}} (\mathcal{P}(\Sigma), \subseteq)$.

We have: $\alpha_p \circ F_p = F_{\mathcal{R}} \circ \alpha_p$;

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)

proof: of $\alpha_p \circ F_p = F_{\mathcal{R}} \circ \alpha_p$

$$
\begin{aligned}
&(\alpha_p \circ F_p)(T) \\
&= \alpha_p(\mathcal{I} \cup T^\frown\tau) \\
&= \{\, \sigma \mid \exists \sigma_0, \dots, \sigma_n \in \mathcal{I} \cup T^\frown\tau : \sigma = \sigma_n \,\} \\
&= \mathcal{I} \cup \{\, \sigma \mid \exists \sigma_0, \dots, \sigma_n \in T^\frown\tau : \sigma = \sigma_n \,\} \\
&= \mathcal{I} \cup \{\, \sigma \mid \exists \sigma_0, \dots, \sigma_n \in T : \sigma_n \to \sigma \,\} \\
&= \mathcal{I} \cup \mathsf{post}_\tau(\{\, \sigma \mid \exists \sigma_0, \dots, \sigma_n \in T : \sigma = \sigma_n \,\}) \\
&= \mathcal{I} \cup \mathsf{post}_\tau(\alpha_p(T)) \\
&= (F_{\mathcal{R}} \circ \alpha_p)(T)
\end{aligned}
$$

Abstracting traces into states (example)

program

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

- prefix trace semantics:
  i and j are increasing and $0 \leq j \leq i \leq 100$
- forward reachable state semantics:
  $0 \leq j \leq i \leq 100$

$\Longrightarrow$ the abstraction forgets the ordering of states.

Application: partitioning

program

$X \leftarrow [10, 20]$;
$Y \leftarrow [0, 1]$;
if $Y \geq 1$ **then** $X \leftarrow -X$;
• **assert** $X \neq 0$

- 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 $\Longrightarrow$ 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

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

More in Xavier Rival's course on partitioning

Prefix closure

Prefix partial order: $\preceq$ on Σ^∞

$$x \preceq y \stackrel{\text{def}}{\iff} \exists u \in \Sigma^\infty : x \cdot u = y$$

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

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

$$\rho_p(T) \stackrel{\text{def}}{=} \{ u \mid \exists t \in T : u \preceq t,\ u \neq \epsilon \}$$

ρ_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:

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

(note that $\epsilon \notin \mathcal{T}_p(\mathcal{I})$, which is why we disallowed ϵ in ρ_p)

Another state/trace abstraction: Ordering abstraction

Another Galois embedding between finite traces and states:

$$(\mathcal{P}(\Sigma^*), \subseteq) \underset{\alpha_o}{\overset{\gamma_o}{\leftrightarrows}} (\mathcal{P}(\Sigma), \subseteq)$$

- $\alpha_o(T) \overset{\text{def}}{=} \{ \sigma \mid \exists \sigma_0, \ldots, \sigma_n \in T, i \leq n : \sigma = \sigma_i \}$
  (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:

α_o and γ_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: $\alpha_o(\mathcal{T}_p(\mathcal{I})) = \mathcal{R}(\mathcal{I})$.

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: $\mathcal{R}(\mathcal{I}) = \alpha_p(\mathcal{T}_p(\mathcal{I}))$ and the fact that the prefix trace semantics is closed by prefix: $\rho_p(\mathcal{T}_p(\mathcal{I})) = \mathcal{T}_p(\mathcal{I})$.
We get $\alpha_o(\mathcal{T}_p(\mathcal{I})) = \alpha_p(\rho_p(\mathcal{T}_p(\mathcal{I}))) = \alpha_p(\mathcal{T}_p(\mathcal{I})) = \mathcal{R}(\mathcal{I})$.

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

alternate proof: generalized fixpoint transfer

Recall that $\mathcal{T}_p(\mathcal{I}) = \mathrm{lfp}\, F_p$ where $F_p(T) \stackrel{\text{def}}{=} \mathcal{I} \cup T \frown \tau$ and $\mathcal{R}(\mathcal{I}) = \mathrm{lfp}\, F_{\mathcal{R}}$ where $F_{\mathcal{R}}(S) \stackrel{\text{def}}{=} \mathcal{I} \cup \mathrm{post}_\tau(S)$, but $\alpha_o \circ F_p = F_{\mathcal{R}} \circ \alpha_o$ does not hold in general, so, fixpoint transfer theorems do not apply directly.

However, $\alpha_o \circ F_p = F_{\mathcal{R}} \circ \alpha_o$ holds for sets of traces closed by prefix. By induction, the Kleene iterates a_p^n and $a_{\mathcal{R}}^n$ involved in the computation of $\mathrm{lfp}\, F_p$ and $\mathrm{lfp}\, F_{\mathcal{R}}$ satisfy $\forall n\colon \alpha_o(a_p^n) = a_{\mathcal{R}}^n$, and so $\alpha_o(\mathrm{lfp}\, F_p) = \mathrm{lfp}\, F_{\mathcal{R}}$.

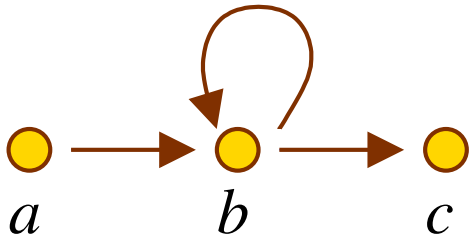
Finite suffix trace semantics

Suffix trace semantics

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

- $\mathcal{T}_s(\mathcal{F}) \stackrel{\text{def}}{=} \{ \sigma_0, \ldots, \sigma_n \mid n \geq 0, \sigma_n \in \mathcal{F}, \forall i : \sigma_i \to \sigma_{i+1} \}$
  (traces following τ and ending in a state in $\mathcal{F}$)
- $\mathcal{T}_s(\mathcal{F}) = \bigcup_{n \geq 0} \tau^{\frown n} \frown \mathcal{F}$
- $\mathcal{T}_s(\mathcal{F}) = \text{lfp}\, F_s$ where $F_s(T) \stackrel{\text{def}}{=} \mathcal{F} \cup \tau^{\frown} T$
  (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 $\alpha_s(T) \stackrel{\text{def}}{=} \{ \sigma \mid \exists \sigma_0, \ldots, \sigma_n \in T : \sigma = \sigma_0 \}$
- $\rho_s(\mathcal{T}_s(\mathcal{F})) = \mathcal{T}_s(\mathcal{F})$
  where $\rho_s(T) \stackrel{\text{def}}{=} \{ u \mid \exists t \in \Sigma^\infty : t \cdot u \in T, u \neq \epsilon \}$
  (closed by suffix)

Graphical illustration



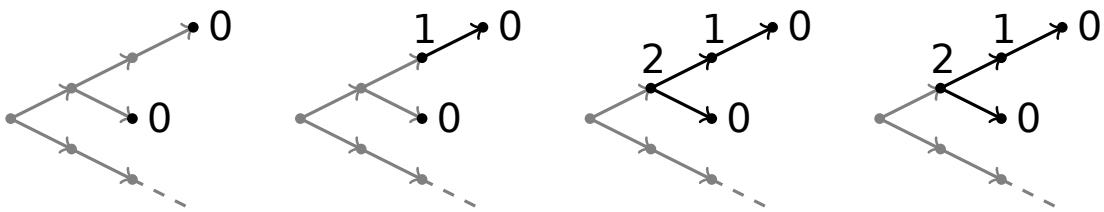
$$\mathcal{F} \stackrel{\text{def}}{=} \{c\}$$
$$\tau \stackrel{\text{def}}{=} \{(a,b),(b,b),(b,c)\}$$

Iterates: $\mathcal{T}_s(\mathcal{F}) = \text{lfp}\, F_s$ where $F_s(T) \stackrel{\text{def}}{=} \mathcal{F} \cup \tau^\frown T$.

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

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

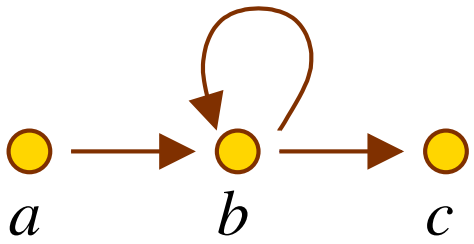
$\mathcal{T}$: all finite partial finite execution traces.

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

$$\begin{aligned}\mathcal{T} &\stackrel{\text{def}}{=} \{\, \sigma_0, \ldots, \sigma_n \mid n \geq 0, \forall i\colon \sigma_i \to \sigma_{i+1} \,\} \\ &= \bigcup_{n\geq 0} \Sigma^\frown \tau^{\frown n} \\ &= \bigcup_{n\geq 0} \tau^{\frown n} \frown \Sigma\end{aligned}$$

- $\mathcal{T} = \mathcal{T}_p(\Sigma)$, hence $\mathcal{T} = \text{lfp}\, F_{p*}$ where $F_{p*}(T) \stackrel{\text{def}}{=} \Sigma \cup T^\frown \tau$
  (prefix partial traces from any initial state)
- $\mathcal{T} = \mathcal{T}_s(\Sigma)$, hence $\mathcal{T} = \text{lfp}\, F_{s*}$ where $F_{s*}(T) \stackrel{\text{def}}{=} \Sigma \cup \tau^\frown T$
  (suffix partial traces to any final state)
- $F^n_{p*}(\emptyset) = F^n_{s*}(\emptyset) = \bigcup_{i<n} \Sigma^\frown \tau^{\frown i} = \bigcup_{i<n} \tau^{\frown i} \frown \Sigma = \mathcal{T} \cap \Sigma^{<n}$
- $\mathcal{T}_p(\mathcal{I}) = \mathcal{T} \cap (\mathcal{I} \cdot \Sigma^*)$ (restricted initial states)
- $\mathcal{T}_s(\mathcal{F}) = \mathcal{T} \cap (\Sigma^* \cdot \mathcal{F})$ (restricted final states)

Partial trace semantics: graphical illustration



$$\tau \stackrel{\text{def}}{=} \{(a,b),(b,b),(b,c)\}$$

Iterates: $\mathcal{T}(\Sigma) = \text{lfp}\, F_{p*}$ where $F_{p*}(T) \stackrel{\text{def}}{=} \Sigma \cup T \frown \tau$.

- $F^0_{p*}(\emptyset) = \emptyset$
- $F^1_{p*}(\emptyset) = \Sigma = \{a, b, c\}$
- $F^2_{p*}(\emptyset) = \{a, b, c, ab, bb, bc\}$
- $F^3_{p*}(\emptyset) = \{a, b, c, ab, bb, bc, abb, abc, bbb, bbc\}$
- $F^n_{p*}(\emptyset) = \{\, ab^i, ab^j c, b^i c, b^k \mid i \in [0, n-1], j \in [1, n-2], k \in [1, n] \,\}$
- $\mathcal{T} = \cup_{n \geq 0} F^n_{p*}(\emptyset) = \{\, ab^i, ab^j c, b^i c, b^j \mid i \geq 0, j > 1 \,\}$

(using $F_{s*}(T) \stackrel{\text{def}}{=} \Sigma \cup \tau \frown T$, 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:

$$(\mathcal{P}(\Sigma^*), \subseteq) \underset{\alpha_{\mathcal{I}}}{\overset{\gamma_{\mathcal{I}}}{\leftrightarrows}} (\mathcal{P}(\Sigma^*), \subseteq)$$

- $\alpha_{\mathcal{I}}(T) \overset{\text{def}}{=} T \cap (\mathcal{I} \cdot \Sigma^*)$ (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: $\mathcal{T}_p(\mathcal{I}) = \alpha_{\mathcal{I}}(\mathcal{T})$.

(similarly $\mathcal{T}_s(\mathcal{F}) = \alpha_{\mathcal{F}}(\mathcal{T})$ 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

$\alpha_{\mathcal{I}}$ and $\gamma_{\mathcal{I}}$ are monotonic.
$(\alpha_{\mathcal{I}} \circ \gamma_{\mathcal{I}})(T) = (T \cup (\Sigma \setminus \mathcal{I}) \cdot \Sigma^*) \cap \mathcal{I} \cdot \Sigma^* = T \cap \mathcal{I} \cdot \Sigma^* \subseteq T$.
$(\gamma_{\mathcal{I}} \circ \alpha_{\mathcal{I}})(T) = (T \cap \mathcal{I} \cdot \Sigma^*) \cup (\Sigma \setminus \mathcal{I}) \cdot \Sigma^* = T \cup (\Sigma \setminus \mathcal{I}) \cdot \Sigma^* \supseteq T$.
So, we have a Galois connection.

A direct proof of $\mathcal{T}_p(\mathcal{I}) = \alpha_{\mathcal{I}}(\mathcal{T})$ is straightforward,
by definition of $\mathcal{T}_p$, $\alpha_{\mathcal{I}}$, and $\mathcal{T}$.

We can also retrieve the result by fixpoint transfer.

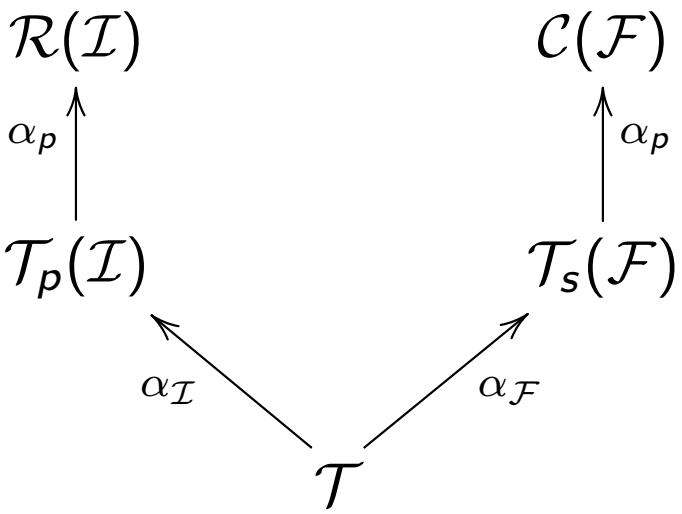
$\mathcal{T} = \text{lfp}\, F_{p*}$ where $F_{p*}(T) \overset{\text{def}}{=} \Sigma \cup T^\frown\tau$.

$\mathcal{T}_p = \text{lfp}\, F_p$ where $F_p(T) \overset{\text{def}}{=} \mathcal{I} \cup T^\frown\tau$.

We have: $(\alpha_{\mathcal{I}} \circ F_{p*})(T) = (\Sigma \cup T^\frown\tau) \cap (\mathcal{I} \cdot \Sigma^*) = \mathcal{I} \cup ((T^\frown\tau) \cap (\mathcal{I} \cdot \Sigma^*) =$

$\mathcal{I} \cup ((T \cap (\mathcal{I} \cdot \Sigma^*))^\frown\tau) = (F_p \circ \alpha_{\mathcal{I}})(T)$.

(Partial) hierarchy of semantics



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 τ,
- 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 τ at their end

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

(can be anchored at $\mathcal{I}$ and $\mathcal{F}$ as: $\mathcal{M}_\infty \cap (\mathcal{I} \cdot \Sigma^\infty) \cap ((\Sigma^* \cdot \mathcal{F}) \cup \Sigma^\omega)$)

Partitioned fixpoint formulation of maximal traces

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:

  $\mathcal{M}_\infty \cap \Sigma^* = \mathcal{T}_s(\mathcal{B}) = \mathrm{lfp}\, F_s$

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

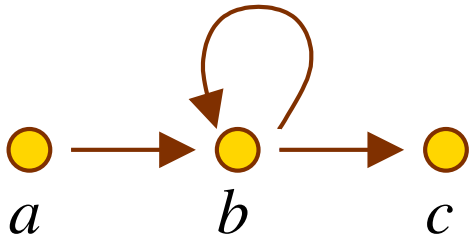
- Infinite traces:

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

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

(proof in following slides)

Infinite trace semantics: graphical illustration



$$\mathcal{B} \stackrel{\text{def}}{=} \{c\}$$
$$\tau \stackrel{\text{def}}{=} \{(a,b),(b,b),(b,c)\}$$

Iterates: $\mathcal{M}_\infty \cap \Sigma^\omega = \text{gfp}\, G_s$ where $G_s(T) \stackrel{\text{def}}{=} \tau^\frown T$.

- $G_s^0(\Sigma^\omega) = \Sigma^\omega$
- $G_s^1(\Sigma^\omega) = ab\Sigma^\omega \cup bb\Sigma^\omega \cup bc\Sigma^\omega$
- $G_s^2(\Sigma^\omega) = abb\Sigma^\omega \cup bbb\Sigma^\omega \cup abc\Sigma^\omega \cup bbc\Sigma^\omega$
- $G_s^3(\Sigma^\omega) = abbb\Sigma^\omega \cup bbbb\Sigma^\omega \cup abbc\Sigma^\omega \cup bbbc\Sigma^\omega$
- $G_s^n(\Sigma^\omega) = \{\, ab^n t,\ b^{n+1}t,\ ab^{n-1}ct,\ b^n ct \mid t \in \Sigma^\omega \,\}$
- $\mathcal{M}_\infty \cap \Sigma^\omega = \cap_{n\geq 0} G_s^n(\Sigma^\omega) = \{ab^\omega,\ b^\omega\}$

Infinite trace semantics: proof

$\mathcal{M}_\infty \cap \Sigma^\omega = \text{gfp}\, G_s$
where $G_s(T) \stackrel{\text{def}}{=} \tau^\frown T$ in $(\mathcal{P}(\Sigma^\omega), \subseteq)$

proof:

G_s is continuous in $(\mathcal{P}(\Sigma^\omega), \supseteq)$: $G_s(\cap_{i\in I} T_i) = \cap_{i\in I} G_s(T_i)$.
By Kleene's theorem in the dual: $\text{gfp}\, G_s = \cap_{n\in\mathbb{N}} G_s^n(\Sigma^\omega)$.
We prove by recurrence on n that $\forall n\colon G_s^n(\Sigma^\omega) = \tau^{\frown n}\frown\Sigma^\omega$:

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

$$\begin{aligned}
\text{gfp}\, G_s &= \cap_{n\in\mathbb{N}} \tau^{\frown n}\frown\Sigma^\omega \\
&= \{\, \sigma_0, \ldots \in \Sigma^\omega \mid \forall n \geq 0\colon \sigma_0, \ldots, \sigma_{n-1} \in \tau^{\frown n} \,\} \\
&= \{\, \sigma_0, \ldots \in \Sigma^\omega \mid \forall n \geq 0\colon \forall i < n\colon \sigma_i \to \sigma_{i+1} \,\} \\
&= \mathcal{M}_\infty \cap \Sigma^\omega
\end{aligned}$$

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 $(\Sigma^\infty, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$:

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

In this lattice, $\mathcal{M}_\infty = \text{lfp}\ F_s$ where $F_s(T) \overset{\text{def}}{=} \mathcal{B} \cup \tau^\frown T$.

(proof on next slides)

Fixpoint fusion theorem

Theorem: fixpoint fusion

If $X_1 = \mathrm{lfp}\, F_1$ in $(\mathcal{P}(\mathcal{D}_1), \sqsubseteq_1)$ and $X_2 = \mathrm{lfp}\, F_2$ in $(\mathcal{P}(\mathcal{D}_2), \sqsubseteq_2)$
and $\mathcal{D}_1 \cap \mathcal{D}_2 = \emptyset$,
then $X_1 \cup X_2 = \mathrm{lfp}\, F$ in $(\mathcal{P}(\mathcal{D}_1 \cup \mathcal{D}_2), \sqsubseteq)$ where:

- $F(X) \stackrel{\text{def}}{=} F_1(X \cap \mathcal{D}_1) \cup F_2(X \cap \mathcal{D}_2)$,
- $A \sqsubseteq B \iff^{\text{def}} (A \cap \mathcal{D}_1) \sqsubseteq_1 (B \cap \mathcal{D}_1) \wedge (A \cap \mathcal{D}_2) \sqsubseteq_2 (B \cap \mathcal{D}_2)$.

proof:

We have:
$F(X_1 \cup X_2) = F_1((X_1 \cup X_2) \cap \mathcal{D}_1) \cup F_2((X_1 \cup X_2) \cap \mathcal{D}_2) = F_1(X_1) \cup F_2(X_2) = X_1 \cup X_2$,
hence $X_1 \cup X_2$ is a fixpoint of F.

Let Y be a fixpoint. Then $Y = F(Y) = F_1(Y \cap \mathcal{D}_1) \cup F_2(Y \cap \mathcal{D}_2)$, hence,
$Y \cap \mathcal{D}_1 = F_1(Y \cap \mathcal{D}_1)$ and $Y \cap \mathcal{D}_1$ is a fixpoint of F_1. Thus, $X_1 \sqsubseteq_1 Y \cap \mathcal{D}_1$. Likewise, $X_2 \sqsubseteq_2 Y \cap \mathcal{D}_2$. We deduce that $X = X_1 \cup X_2 \sqsubseteq (Y \cap \mathcal{D}_1) \cup (Y \cap \mathcal{D}_2) = Y$, and so, X is F's least fixpoint.

note: we also have $\mathrm{gfp}\, F = \mathrm{gfp}\, F_1 \cup \mathrm{gfp}\, F_2$.

Least fixpoint formulation of maximal traces (proof)

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

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 $G_s(T) \stackrel{\text{def}}{=} \tau^\frown T$,
- 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 $(\mathcal{P}(\Sigma^\infty), \sqsubseteq)$, 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!

Greatest fixpoint formulation of finite maximal traces

Actually, a fixpoint formulation in $(\Sigma^\infty, \subseteq)$ also exists.

Alternate fixpoint for finite maximal traces:

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

Additionally, we have $\mathcal{M}_\infty \cap \Sigma^* = \mathrm{gfp}\, F_s$ 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 $\mathcal{M}_\infty \cap \Sigma^* = \mathrm{gfp}\, F_s$ where $F_s(T) \stackrel{\text{def}}{=} \mathcal{B} \cup \tau^\frown T$.

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: $\mathrm{gfp}\, F_s = \cap_{n\in\mathbb{N}} F_s^n(\Sigma^*)$.

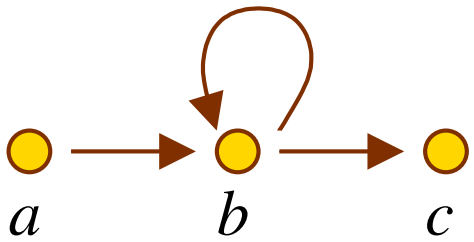
We prove by recurrence on n that $\forall n$: $F_s^n(\Sigma^*) = (\cup_{i<n} \tau^{\frown i} \frown \mathcal{B}) \cup (\tau^{\frown n} \frown \Sigma^*)$: i.e., $F_s^n(\Sigma^*)$ are the maximal finite traces of length at most $n-1$, and the partial traces of length exactly n followed by any sequence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

- $F_s^0(\Sigma^*) = \Sigma^* = \tau^{\frown 0} \frown \Sigma^*$
- $F_s(F_s^n(\Sigma^*)) = \mathcal{B} \cup (\tau^\frown F_s^n(\Sigma^*))$
  $= \mathcal{B} \cup \tau^\frown((\cup_{i<n} \tau^{\frown i} \frown \mathcal{B}) \cup (\tau^{\frown n} \frown \Sigma^*))$
  $= \mathcal{B} \cup (\cup_{i<n} \tau^\frown \tau^{\frown i} \frown \mathcal{B}) \cup (\tau^\frown \tau^{\frown n} \frown \Sigma^*)$
  $= \mathcal{B} \cup (\cup_{1<i<n+1} \tau^{\frown i} \frown \mathcal{B}) \cup (\tau^{\frown n+1} \frown \Sigma^*)$
  $= (\cup_{i<n+1} \tau^{\frown i} \frown \mathcal{B}) \cup (\tau^{\frown n+1} \frown \Sigma^*)$

We get:

$\cap_{n\in\mathbb{N}} F_s^n(\Sigma^*) = \cap_{n\in\mathbb{N}} (\cup_{i<n} \tau^{\frown i} \frown \mathcal{B}) \cup (\tau^{\frown n} \frown \Sigma^*) = \cup_{n\in\mathbb{N}} \tau^{\frown n} \frown \mathcal{B} = \mathcal{M}_\infty \cap \Sigma^*$.

Greatest fixpoint of finite traces: graphical illustration



$$\mathcal{B} \stackrel{\text{def}}{=} \{c\}$$
$$\tau \stackrel{\text{def}}{=} \{(a,b),(b,b),(b,c)\}$$

Iterates: $\mathcal{M}_\infty \cap \Sigma^* = \text{gfp}\, F_s$ where $F_s(T) \stackrel{\text{def}}{=} \mathcal{B} \cup \tau^\frown T$.

- $F_s^0(\Sigma^*) = \Sigma^*$
- $F_s^1(\Sigma^*) = \{c\} \cup ab\Sigma^* \cup bb\Sigma^* \cup bc\Sigma^*$
- $F_s^2(\Sigma^*) = \{bc, c\} \cup abb\Sigma^* \cup bbb\Sigma^* \cup abc\Sigma^* \cup bbc\Sigma^*$
- $F_s^3(\Sigma^*) = \{abc, bbc, bc, c\} \cup abbb\Sigma^* \cup bbbb\Sigma^* \cup abbc\Sigma^* \cup bbbc\Sigma^*$
- $F_s^n(\Sigma^*) = \{\, ab^i c, b^j c \mid i \in [1, n-2], j \in [0, n-1] \,\} \cup$
  $\{\, ab^n t, b^{n+1} t, ab^{n-1} ct, b^n ct \mid t \in \Sigma^* \,\}$
- $\mathcal{M}_\infty \cap \Sigma^* = \cap_{n \geq 0} F_s^n(\Sigma^*) = \{\, ab^i c, b^j c \mid i \geq 1, j \geq 0 \,\}$

Greatest fixpoint formulation of maximal traces

From:

- $\mathcal{M}_\infty \cap \Sigma^* = \mathrm{gfp}\, F_s$ in $(\mathcal{P}(\Sigma^*), \subseteq)$ where $F_s(T) \stackrel{\text{def}}{=} \mathcal{B} \cup \tau^\frown T$
- $\mathcal{M}_\infty \cap \Sigma^\omega = \mathrm{gfp}\, G_s$ in $(\mathcal{P}(\Sigma^\omega), \subseteq)$ where $G_s(T) \stackrel{\text{def}}{=} \tau^\frown T$

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

proof: similar to $\mathcal{M}_\infty = \mathrm{lfp}\, F_s$ in $(\mathcal{P}(\Sigma^\infty), \sqsubseteq)$, by fixpoint fusion.

Abstracting maximal traces into partial traces

Finite and infinite partial trace semantics

Idea: complete the partial traces $\mathcal{T}$ with infinite traces.

$\mathcal{T}_\infty$: all finite and infinite sequences of states
linked by the transition relation τ:

$$\begin{aligned}\mathcal{T}_\infty \stackrel{\text{def}}{=} & \{ \sigma_0, \ldots, \sigma_n \in \Sigma^* \mid \forall i < n : \sigma_i \to \sigma_{i+1} \} \cup \\ & \{ \sigma_0, \ldots, \sigma_n, \ldots \in \Sigma^\omega \mid \forall i < \omega : \sigma_i \to \sigma_{i+1} \}\end{aligned}$$

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

Fixpoint form similar to $\mathcal{M}_\infty$:
$\mathcal{T}_\infty = \text{lfp}\, F_{s*}$ in $(\mathcal{P}(\Sigma^\infty), \sqsubseteq)$ where $F_{s*}(T) \stackrel{\text{def}}{=} \Sigma \cup \tau^\frown T$,

proof: similar to the proof of $\mathcal{M}_\infty = \text{lfp}\, F_s$.

Finite trace abstraction

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

We have a Galois embedding:

$$(\mathcal{P}(\Sigma^\infty), \sqsubseteq) \underset{\alpha_*}{\overset{\gamma_*}{\leftrightarrows}} (\mathcal{P}(\Sigma^*), \subseteq)$$

- $\sqsubseteq$ is the fused ordering on $\Sigma^* \cup \Sigma^\omega$:
  $A \sqsubseteq B \overset{\text{def}}{\iff} (A \cap \Sigma^*) \subseteq (B \cap \Sigma^*) \wedge (A \cap \Sigma^\omega) \supseteq (B \cap \Sigma^\omega)$
- $\alpha_*(T) \overset{\text{def}}{=} T \cap \Sigma^*$
  (remove infinite traces)
- $\gamma_*(T) \overset{\text{def}}{=} T$
  (embedding)
- $\mathcal{T} = \alpha_*(\mathcal{T}_\infty)$

(proof on next slide)

Finite trace abstraction (proof)

proof:

We have Galois embedding because:

- α_* and γ_* are monotonic,
- given $T \subseteq \Sigma^*$, we have $(\alpha_* \circ \gamma_*)(T) = T \cap \Sigma^* = T$,
- $(\gamma_* \circ \alpha_*)(T) = T \cap \Sigma^* \sqsupseteq T$, as we only remove infinite traces.

Recall that $\mathcal{T}_\infty = \mathrm{lfp}\, F_{s*}$ in $(\mathcal{P}(\Sigma^\infty), \sqsubseteq)$ and $\mathcal{T} = \mathrm{lfp}\, F_{s*}$ in $(\mathcal{P}(\Sigma^*), \subseteq)$, where $F_{s*}(T) \stackrel{\text{def}}{=} \Sigma \cup T^\frown\tau$.

As $\alpha_* \circ F_{s*} = F_{s*} \circ \alpha_*$ and $\alpha_*(\emptyset) = \emptyset$, we can apply the fixpoint transfer theorem to get $\alpha_*(\mathcal{T}_\infty) = \mathcal{T}$.

Prefix abstraction

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

We have a Galois connection:

$$(\mathcal{P}(\Sigma^\infty \setminus \{\epsilon\}), \subseteq) \underset{\alpha_\preceq}{\overset{\gamma_\preceq}{\leftrightarrows}} (\mathcal{P}(\Sigma^\infty \setminus \{\epsilon\}), \subseteq)$$

- $\alpha_\preceq(T) \overset{\text{def}}{=} \{ t \in \Sigma^\infty \setminus \{\epsilon\} \mid \exists u \in T : t \preceq u \}$
  (set of all non-empty prefixes of traces in T)
- $\gamma_\preceq(T) \overset{\text{def}}{=} \{ t \in \Sigma^\infty \setminus \{\epsilon\} \mid \forall u \in \Sigma^\infty \setminus \{\epsilon\} : u \preceq t \implies u \in T \}$
  (traces with non-empty prefixes in T)

proof:

$\alpha_\preceq$ and $\gamma_\preceq$ are monotonic.

$(\alpha_\preceq \circ \gamma_\preceq)(T) = \{ t \in T \mid \rho_p(t) \subseteq T \} \subseteq T$ (prefix-closed trace sets).

$(\gamma_\preceq \circ \alpha_\preceq)(T) = \rho_p(T) \supseteq T$.

Abstraction from maximal traces to partial traces

Finite and infinite partial traces $\mathcal{T}_\infty$ are an abstraction of maximal traces $\mathcal{M}_\infty$: $\mathcal{T}_\infty = \alpha_\preceq(\mathcal{M}_\infty)$.

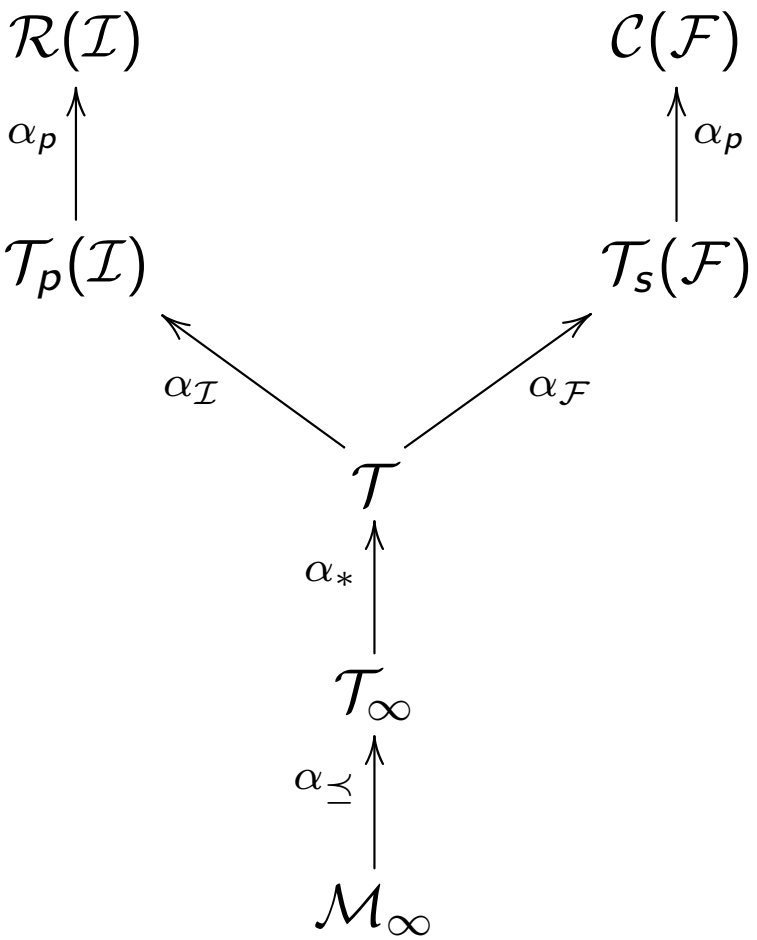
proof:

Firstly, $\mathcal{T}_\infty$ and $\alpha_\preceq(\mathcal{M}_\infty)$ coincide on infinite traces. Indeed, $\mathcal{T}_\infty \cap \Sigma^\omega = \mathcal{M}_\infty \cap \Sigma^\omega$ and $\alpha_\preceq$ does not add infinite traces, so: $\mathcal{T}_\infty \cap \Sigma^\omega = \alpha_\preceq(\mathcal{M}_\infty) \cap \Sigma^\omega$.

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

Note: no fixpoint transfer applies here.

(Partial) hierarchy of semantics



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: $P \stackrel{\text{def}}{=} \Sigma \setminus \mathcal{B}$,
- 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 $P' \stackrel{\text{def}}{=} P \cup ((\Sigma \setminus \mathcal{I}) \cdot \Sigma^\infty)$)

Examples:

- termination: $P \stackrel{\text{def}}{=} \Sigma^*$,
- non-termination: $P \stackrel{\text{def}}{=} \Sigma^\omega$,
- any state property $S \subseteq \Sigma$: $P \stackrel{\text{def}}{=} S^\infty$,
- maximal execution time: $P \stackrel{\text{def}}{=} \Sigma^{\leq k}$,
- minimal execution time: $P \stackrel{\text{def}}{=} \Sigma^{\geq k}$,
- ordering, e.g.: $P \stackrel{\text{def}}{=} (\Sigma \setminus \{b\})^* \cdot a \cdot \Sigma^* \cdot b \cdot \Sigma^\infty$.
  (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: $\mathcal{T}_p(\mathcal{I}) \subseteq P$)
- P is disprovable by finding a single finite execution not in P.

Examples:

- any state property: $P \stackrel{\text{def}}{=} S^\infty$ for $S \subseteq \Sigma$,
- ordering: $P \stackrel{\text{def}}{=} \Sigma^\infty \setminus ((\Sigma \setminus \{a\})^* \cdot b \cdot \Sigma^\infty)$,
  (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 $P \stackrel{\text{def}}{=} \Sigma^*$ is not a safety property.
  (disproving requires exhibiting an *infinite* execution)

Definition of safety properties

Reminder: finite prefix abstraction (simplified to allow ϵ)

$$(\mathcal{P}(\Sigma^\infty), \subseteq) \underset{\alpha_{*\preceq}}{\overset{\gamma_{*\preceq}}{\leftrightarrows}} (\mathcal{P}(\Sigma^*), \subseteq)$$

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

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

Definition of safety properties (examples)

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

Examples and counter-examples:

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

  $\rho_p(S^\infty) = \lim(S^\infty) = S^\infty \Longrightarrow$ safety;

- termination $P \stackrel{\text{def}}{=} \Sigma^*$:

  $\rho_p(\Sigma^*) = \Sigma^*$, but $\lim(\Sigma^*) = \Sigma^\infty \neq \Sigma^* \Longrightarrow$ not safety;

- even number of steps $P \stackrel{\text{def}}{=} (\Sigma^2)^\infty$:

  $\rho_p((\Sigma^2)^\infty) = \Sigma^\infty \neq (\Sigma^2)^\infty \Longrightarrow$ 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)
- $\forall \sigma_0, \ldots, \sigma_n \in I : \sigma_n \to \sigma_{n+1} \implies \sigma_0, \ldots, \sigma_n, \sigma_{n+1} \in I$

  (invariant by program transition)

and implies the desired property: $I \subseteq P$.

Link with the finite prefix trace semantics $\mathcal{T}_p(\mathcal{I})$:

An inductive invariant is a post-fixpoint of F_p: $F_p(I) \subseteq I$

where $F_p(T) \stackrel{\text{def}}{=} \mathcal{I} \cup T \frown \tau$.

$\mathcal{T}_p(\mathcal{I}) = \text{lfp}\, F_p$ 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: $\mathcal{M}_\infty \cap (\mathcal{I} \cdot \Sigma^\infty) \subseteq P$

proof:

Using the Galois connection between $\mathcal{M}_\infty$ and $\mathcal{T}$, we get:
$\mathcal{M}_\infty \cap (\mathcal{I} \cdot \Sigma^\infty) \subseteq \rho_{*\preceq}(\mathcal{M}_\infty \cap (\mathcal{I} \cdot \Sigma^\infty)) = \gamma_{*\preceq}(\alpha_{*\preceq}(\mathcal{M}_\infty \cap (\mathcal{I} \cdot \Sigma^\infty))) =$
$\gamma_{*\preceq}(\alpha_{*\preceq}(\mathcal{M}_\infty) \cap (\mathcal{I} \cdot \Sigma^*)) = \gamma_{*\preceq}(\mathcal{T} \cap (\mathcal{I} \cdot \Sigma^*)) = \gamma_{*\preceq}(\mathcal{T}_p(\mathcal{I}))$.
Using the link between invariants and the finite prefix trace semantics, we have:
$\mathcal{T}_p(\mathcal{I}) \subseteq I \subseteq P$.

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

Completeness: an inductive invariant always exists

proof: $\mathcal{T}_p(\mathcal{I})$ 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:

$$\mathcal{M}_\infty \cap (\mathcal{I} \cdot \Sigma^\infty) \not\subseteq P \implies \exists t \in \mathcal{T}_p(\mathcal{I}) : t \notin P$$

proof:

By contradiction, assume that no such trace exists, i.e., $\mathcal{T}_p(\mathcal{I}) \subseteq P$.

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 $P \stackrel{\text{def}}{=} S^\infty$:
  $\Rightarrow$ find a partial execution containing a state in $\Sigma \setminus S$;
- disproving an order property $P \stackrel{\text{def}}{=} \Sigma^\infty \setminus ((\Sigma \setminus \{a\})^* \cdot b \cdot \Sigma^\infty)$
  $\Rightarrow$ find a partial execution where b 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: $P \stackrel{\text{def}}{=} \Sigma^*$,
- inevitability: $P \stackrel{\text{def}}{=} \Sigma^* \cdot a \cdot \Sigma^\infty$,
  (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 $\rho_{*\preceq}(P) = \Sigma^\infty$.

Examples and counter-examples:

- termination $P \stackrel{\text{def}}{=} \Sigma^*$:

  $\rho_p(\Sigma^*) = \Sigma^*$ and $\lim(\Sigma^*) = \Sigma^\infty \Longrightarrow$ liveness;

- inevitability: $P \stackrel{\text{def}}{=} \Sigma^* \cdot a \cdot \Sigma^\infty$

  $\rho_p(P) = P \cup \Sigma^*$ and $\lim(P \cup \Sigma^*) = \Sigma^\infty \Longrightarrow$ liveness;

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

  $\rho_p(S^\infty) = \lim(S^\infty) = S^\infty \neq \Sigma^\infty$ if $S \neq \Sigma \Longrightarrow$ not liveness;

- maximal execution time $P \stackrel{\text{def}}{=} \Sigma^{\leq k}$:

  $\rho_p(\Sigma^{\leq k}) = \lim(\Sigma^{\leq k}) = \Sigma^{\leq k} \neq \Sigma^\infty \Longrightarrow$ not liveness;

- the only property which is both safety and liveness is Σ^∞.

Proving liveness properties

Variance proof method: (informal definition)

Find a decreasing quantity until something good happens.

Example: termination proof

- find $f : \Sigma \to \mathcal{S}$ where $(\mathcal{S}, \sqsubseteq)$ is well-ordered;

  (f is called a "ranking function")

- $\sigma \in \mathcal{B} \implies f = \min \mathcal{S}$;
- $\sigma \to \sigma' \implies f(\sigma') \sqsubset f(\sigma)$.

(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 α_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 Σ^∞

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

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

- $\rho(x) \stackrel{\text{def}}{=} \cap \{ c \in \mathcal{C} \mid x \subseteq c \}$ (upper closure operator in $(\mathcal{P}(X), \subseteq)$)
- on our trace topology, $\rho = \rho_{*\preceq}$.

Dense sets:

- $x \subseteq X$ is dense if $\rho(x) = X$;
- on our trace topology, dense sets are liveness properties.

Decomposition theorem

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:
$\rho(x) \cap (x \cup (X \setminus \rho(x))) = (\rho(x) \cap x) \cup (\rho(x) \cap (X \setminus \rho(x))) = \rho(x) \cap x = x$ as $x \subseteq \rho(x)$.

- $\rho(x)$ is closed
- $x \cup (X \setminus \rho(x))$ is dense because:
$$\begin{aligned} \rho(x \cup (X \setminus \rho(x))) &\supseteq \rho(x) \cup \rho(X \setminus \rho(x)) \\ &\supseteq \rho(x) \cup (X \setminus \rho(x)) \\ &= X \end{aligned}$$

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 τ.

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

(symmetric and transitive closure of τ)

Fixpoint form:

$\mathcal{BS} = \text{lfp}\, F_B$
where $F_B(R) \stackrel{\text{def}}{=} id \cup \{ (\sigma, \sigma'') \mid \exists \sigma' : (\sigma, \sigma') \in R, \sigma' \to \sigma'' \}$.

Relational abstraction

Relational abstraction: allows skipping intermediate steps.

We have a Galois embedding:

$$(\mathcal{P}(\Sigma^*), \subseteq) \underset{\alpha_{io}}{\overset{\gamma_{io}}{\leftrightarrows}} (\mathcal{P}(\Sigma \times \Sigma), \subseteq)$$

- $\alpha_{io}(T) \overset{\text{def}}{=} \{ (\sigma, \sigma') \mid \exists \sigma_0, \ldots, \sigma_n \in T : \sigma = \sigma_0, \sigma' = \sigma_n \}$
  (first and last state of a trace in T)
- $\gamma_{io}(R) \overset{\text{def}}{=} \{ \sigma_0, \ldots, \sigma_n \in \Sigma^* \mid \exists (\sigma, \sigma') \in R : \sigma = \sigma_0, \sigma' = \sigma_n \}$
  (traces respecting the first and last states from R)

proof sketch:

γ_{io} and α_{io} are monotonic.
$(\gamma_{io} \circ \alpha_{io})(T) = \{ \sigma_0, \ldots, \sigma_n \mid \exists \sigma'_0, \ldots, \sigma'_m \in T : \sigma_0 = \sigma'_0, \sigma_n = \sigma'_m \}$.
$(\alpha_{io} \circ \gamma_{io})(R) = R$.

Finite big-step semantics as an abstraction

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

proof sketch: by fixpoint transfer.

We have $\mathcal{T} = \text{lfp}\, F_{p*}$ where $F_{p*}(T) \overset{\text{def}}{=} \Sigma \cup T^\frown\tau$.
Moreover, $F_B(R) \overset{\text{def}}{=} id \cup \{ (\sigma, \sigma'') \mid \exists \sigma' : (\sigma, \sigma') \in R, \sigma' \to \sigma'' \}$.
Then, $\alpha_{io} \circ F_{p*} = F_B \circ \alpha_{io}$ because $\alpha_{io}(\Sigma) = id$ and
$\alpha_{io}(T^\frown\tau) = \{ (\sigma, \sigma'') \mid \exists \sigma' : (\sigma, \sigma') \in \alpha_{io}(T) \wedge \sigma' \to \sigma'' \}$.
By fixpoint transfer: $\alpha_{io}(\mathcal{T}) = \text{lfp}\, F_B$.

We have a similar result using $F_{s*}(T) \overset{\text{def}}{=} \Sigma \cup \tau^\frown T$ and
$F'_B(R) \overset{\text{def}}{=} id \cup \{ (\sigma, \sigma'') \mid \exists \sigma' : (\sigma', \sigma'') \in R \wedge \sigma \to \sigma' \}$.

Finite big-step semantics (example)

program

$i \leftarrow [0, +\infty]$;
while $i > 0$ **do**
  $i \leftarrow i - [0, 1]$;
done

Finite big-step semantics $\mathcal{BS}$: $\{ (\rho, \rho') \mid 0 \leq \rho'(i) \leq \rho(i) \}$.

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:

- $(\sigma, \sigma') \in \Sigma \times \mathcal{B}$: finite execution starting in σ, stopping in σ',
- $(\sigma, \circlearrowleft)$: non-terminating execution starting in σ.

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

Construction by abstraction: of the maximal trace semantics $\mathcal{M}_\infty$.

$$(\mathcal{P}(\Sigma^\infty), \subseteq) \underset{\alpha_d}{\overset{\gamma_d}{\leftrightarrows}} (\mathcal{P}(\Sigma \times (\Sigma \cup \{\circlearrowleft\})), \subseteq)$$

- $\alpha_d(T) \overset{\text{def}}{=} \alpha_{io}(T \cap \Sigma^*) \cup \{ (\sigma, \circlearrowleft) \mid \exists t \in \Sigma^\omega : \sigma \cdot t \in T \}$
- $\gamma_d(R) \overset{\text{def}}{=} \gamma_{io}(R \cap (\Sigma \times \Sigma)) \cup \{ \sigma \cdot t \mid (\sigma, \circlearrowleft) \in R, t \in \Sigma^\omega \}$

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

The denotational semantics is $\mathcal{DS} \overset{\text{def}}{=} \alpha_d(\mathcal{M}_\infty)$.

Denotational fixpoint semantics

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

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

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

- $\bot^* \stackrel{\text{def}}{=} \{(\sigma, \circlearrowleft) \mid \sigma \in \Sigma\}$
- $\top^* \stackrel{\text{def}}{=} \{(\sigma, \sigma') \mid \sigma, \sigma' \in \Sigma\}$
- $A \sqsubseteq^* B \iff ((A \cap \top^*) \subseteq (B \cap \top^*)) \wedge ((A \cap \bot^*) \supseteq (B \cap \bot^*))$
- $A \sqcup^* B \stackrel{\text{def}}{=} ((A \cap \top^*) \cup (B \cap \top^*)) \cup ((A \cap \bot^*) \cap (B \cap \bot^*))$
- $A \sqcap^* B \stackrel{\text{def}}{=} ((A \cap \top^*) \cap (B \cap \top^*)) \cup ((A \cap \bot^*) \cup (B \cap \bot^*))$
- $F_d(R) \stackrel{\text{def}}{=} \{(\sigma, \sigma) \mid \sigma \in \mathcal{B}\} \cup \{(\sigma, \sigma'') \mid \exists \sigma' : \sigma \to \sigma' \wedge (\sigma', \sigma'') \in R\}$

Denotational fixpoint semantics (proof)

proof:

We cannot use directly a fixpoint transfer on $\mathcal{M}_\infty = \text{lfp}\, F_s$ in $(\mathcal{P}(\Sigma^\infty), \sqsubseteq)$ because our Galois connection (α_d, γ_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 $\mathcal{M}_\infty \cap \Sigma^* = \text{lfp}\, F_s$ in $(\mathcal{P}(\Sigma^*), \subseteq)$ where $F_s(T) \overset{\text{def}}{=} \mathcal{B} \cup \tau^\frown T$

and $\mathcal{M}_\infty \cap \Sigma^\omega = \text{gfp}\, G_s$ in $(\mathcal{P}(\Sigma^\omega), \subseteq)$ where $G_s(T) \overset{\text{def}}{=} \cup \tau^\frown T$.

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

(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 $\mathcal{P}(\Sigma^\omega) \to \mathcal{P}(\Sigma \times \{\circlearrowleft\})$, where
$G_d(R) \stackrel{\text{def}}{=} \{ (\sigma, \sigma'') \mid \exists \sigma' : \sigma \to \sigma' \wedge (\sigma', \sigma'') \in R \}$.

A candidate proof would be to apply a fixpoint transfer theorem to
$\mathcal{M}_\infty \cap \Sigma^\omega = \text{gfp}\, G_s$, in the dual, replacing lfp with gfp, and $\cup$ with $\cap$.
However, the proof of the theorem, which required α to be continous, would require α to be co-continuous in the dual, i.e., $\alpha_d(\cap_{i \in I} S_i) = \cap_{\in I} \alpha_d(S_i)$.
This does not hold. Consider for example: $I = \mathbb{N}$ and $S_i = \{ a^n b^\omega \mid n > i \}$:
$\cap_{i \in \mathbb{N}} S_i = \emptyset$, but $\forall i : \alpha_d(S_i) = \{(a, \circlearrowleft)\}$.

We use instead a fixpoint transfer based on Tarksi's theorem.
We have $\text{gfp}\, G_s = \cup \{ X \mid X \subseteq G_s(X) \}$.
Thus, $\alpha_d(\text{gfp}\, G_s) = \alpha_d(\cup \{ X \mid X \subseteq G_s(X) \}) = \cup \{ \alpha_d(X) \mid X \subseteq G_s(X) \}$ as α_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 (α_d, γ_d) imply that
$\{ \alpha_d(X) \mid X \subseteq G_s(X) \} = \{ \alpha_d(X) \mid \alpha_d(X) \subseteq G_d(\alpha_d(X)) \} = \{ Y \mid Y \subseteq G_d(Y) \}$.

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

Denotational semantics (example)

program

$i \leftarrow [0, +\infty];$
while $i > 0$ **do**
  $i \leftarrow i - [0, 1];$
done

Denotational semantics $\mathcal{DS}$:

$$\{ (\rho, \rho') \mid \rho(i) \geq 0 \wedge \rho'(i) = 0 \} \cup \{ (\rho, \circlearrowleft) \mid \rho(i) \geq 0 \}.$$

(quite different from the big-step semantics)

Denotational semantics (functional form)

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

This is possible using the following Galois isomorphism:

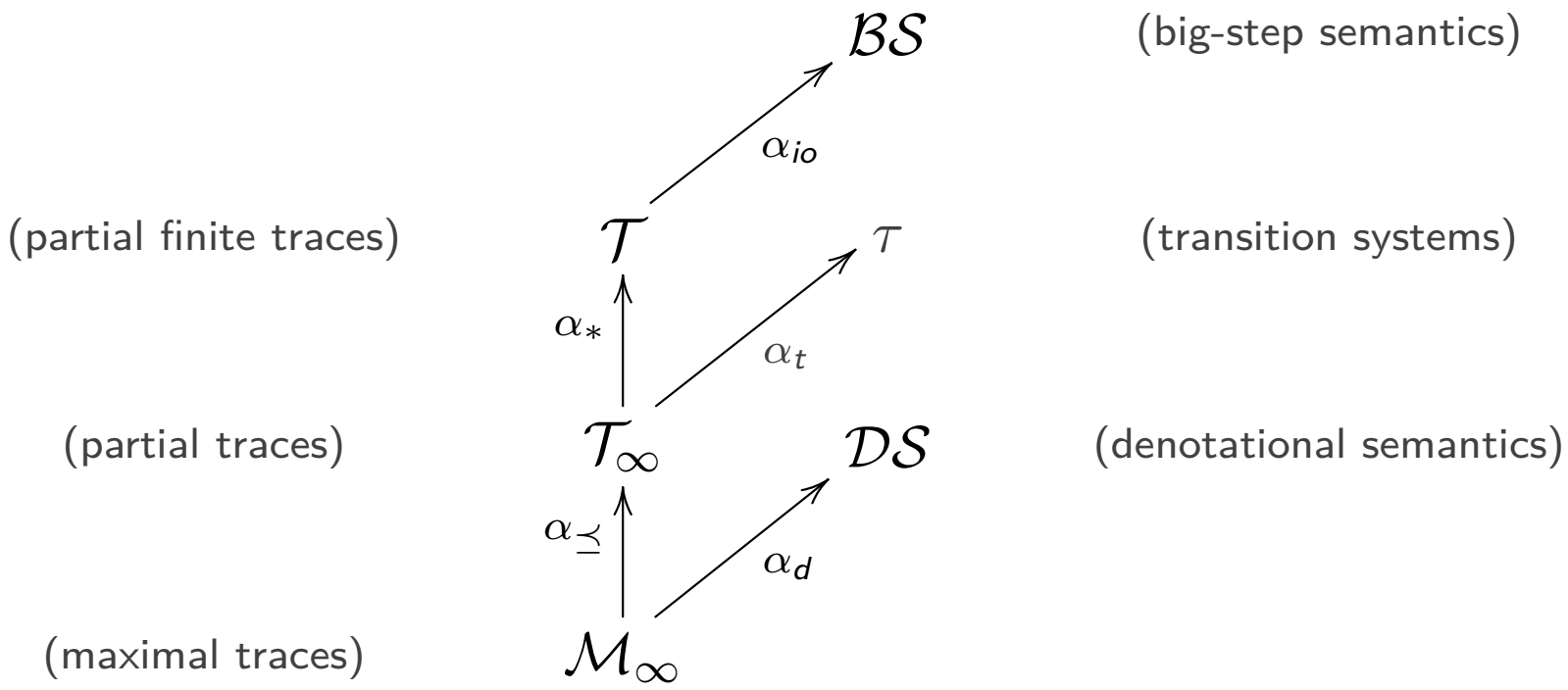
$$(\mathcal{P}(\Sigma \times (\Sigma \cup \{\circlearrowleft\})), \sqsubseteq^*) \underset{\alpha_{df}}{\overset{\gamma_{df}}{\leftrightarrows}} (\Sigma \to \mathcal{P}(\Sigma \cup \{\circlearrowleft\}), \dot{\sqsubseteq}^*)$$

- $\alpha_{df}(R) \overset{\text{def}}{=} \lambda\sigma.\{\, \sigma' \mid (\sigma, \sigma') \in R \,\}$
- $\gamma_{df}(f) \overset{\text{def}}{=} \{\, (\sigma, \sigma') \mid \sigma' \in f(\sigma) \,\}$
- $f \dot{\sqsubseteq}^* f \overset{\text{def}}{\iff} \forall\sigma\colon\ (f(\sigma) \cap \Sigma \subseteq g(\sigma) \cap \Sigma) \wedge (\circlearrowleft \in g(\sigma) \implies \circlearrowleft \in f(\sigma))$

We get that: $\alpha_{df}(\mathcal{DS}) = \text{lfp}\, F'_d$ where
$F'_d(f) \overset{\text{def}}{=} (\alpha_{df} \circ F_d \circ \gamma_{df})(f) = (\lambda\sigma.\{\, \sigma \mid \sigma \in \mathcal{B} \,\}) \dot{\cup} (f \circ \text{post}_\tau)$.

(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

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

Collecting semantics: $Col : Prog \to \mathcal{P}(\mathcal{D})$

- $Col(prog) \stackrel{\text{def}}{=} \{ [\![prog]\!] \}$
- $Col(prog)$ is the strongest property of a program in $\mathcal{P}(\mathcal{D})$
  (relative to the choice of the semantic domain $\mathcal{D}$ and function $[\![\cdot]\!]$)
- we can interpret program verification as property inclusion:
  $Col(prog) \subseteq P$
  P is weaker than $Col(prog)$ in the information order of properties
- generally, the collecting semantics cannot be computed;
  we settle for a weaker property $S^\sharp$ that
  - is sound: $Col(prog) \subseteq S^\sharp$
  - implies the desired property: $S^\sharp \subseteq P$

Retrieving state and trace properties

Reachability state semantics:

- $\mathcal{D} \stackrel{\text{def}}{=} \mathcal{P}(\Sigma)$
- $[\![\cdot]\!] \stackrel{\text{def}}{=} \mathcal{R}(\mathcal{I})$

Trace semantics:

- $\mathcal{D} \stackrel{\text{def}}{=} \mathcal{P}(\Sigma^\infty)$
- $[\![\cdot]\!] \stackrel{\text{def}}{=} \mathcal{M}_\infty \cap (\mathcal{I} \cdot \Sigma^\infty)$

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

$\rho_\downarrow(x)$ for some $x \in \mathcal{D}$

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

(proof: $A \subseteq B \iff A \in \rho_\downarrow(B)$)

Non-trace properties

Note: expressing properties in $\mathcal{P}(\mathcal{D})$
is more general than expressing properties in $\mathcal{D}$

Example: non-interference for variable X

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

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

(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.

Bibliography

Bibliography

[Bour93] **F. Bourdoncle**. *Abstract debugging of higher-order imperative languages.* In PLDI, 46-55, ACM Press, 1993.

[Cous02] **P. Cousot**. *Constructive design of a hierarchy of semantics of a transition system by abstract interpretation.* In Theoretical Comp. Sc., 277(1–2):47–103.

[Plot81] **G. Plotkin**. *The origins of structural operational semantics.* In J. of Logic and Algebraic Prog., 60:60-61, 1981.