Program Semantics and Properties

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

Antoine Miné

Year 2019–2020

Course 02
18 September 2019

Introduction

Language syntax

${}^{\ell}\mathtt{stat}^{\ell}$	$::=$	${}^{\ell}X \leftarrow \mathtt{exp}^{\ell}$	*(assignment)*
	$\mid$	${}^{\ell}\textbf{if}\ \mathtt{exp} \bowtie 0\ \textbf{then}\ {}^{\ell}\mathtt{stat}^{\ell}$	*(conditional)*
	$\mid$	${}^{\ell}\textbf{while}\ {}^{\ell}\mathtt{exp} \bowtie 0\ \textbf{do}\ {}^{\ell}\mathtt{stat}^{\ell}\ \textbf{done}^{\ell}$	*(loop)*
	$\mid$	${}^{\ell}\mathtt{stat};\ {}^{\ell}\mathtt{stat}^{\ell}$	*(sequence)*
$\mathtt{exp}$	$::=$	X	*(variable)*
	$\mid$	$-\mathtt{exp}$	*(negation)*
	$\mid$	$\mathtt{exp} \diamond \mathtt{exp}$	*(binary operation)*
	$\mid$	c	*(constant $c \in \mathbb{Z}$)*
	$\mid$	$[c, c']$	*(random input, $c, c' \in \mathbb{Z} \cup \{\pm\infty\}$)*

Simple structured, numeric language

- $X \in \mathbb{V}$, where $\mathbb{V}$ is a finite set of program variables
- $\ell \in \mathcal{L}$, where $\mathcal{L}$ is a finite set of control points
- numeric expressions: $\bowtie\ \in \{=, \leq, \ldots\}$, $\diamond \in \{+, -, \times, /\}$
- random inputs: $X \leftarrow [c, c']$
  model environment, parametric programs, unknown functions, ...

Expression semantics

$E[\![e]\!] : (\mathbb{V} \to \mathbb{Z}) \to \mathcal{P}(\mathbb{Z})$

- semantics of an expression in a memory state $\rho \in \mathcal{E} \stackrel{\text{def}}{=} \mathbb{V} \to \mathbb{Z}$
- outputs a set of values in $\mathcal{P}(\mathbb{Z})$
  - divisions by zero return no result (omit error states for simplicity)
  - random inputs lead to several values (non-determinism)
- defined by structural induction

$$
\begin{array}{lcl}
E[\![[c, c']]\!] \rho & \stackrel{\text{def}}{=} & \{ x \in \mathbb{Z} \mid c \leq x \leq c' \} \\
E[\![X]\!] \rho & \stackrel{\text{def}}{=} & \{ \rho(X) \} \\
E[\![-e]\!] \rho & \stackrel{\text{def}}{=} & \{ -v \mid v \in E[\![e]\!] \rho \} \\
E[\![e_1 + e_2]\!] \rho & \stackrel{\text{def}}{=} & \{ v_1 + v_2 \mid v_1 \in E[\![e_1]\!] \rho, v_2 \in E[\![e_2]\!] \rho \} \\
E[\![e_1 - e_2]\!] \rho & \stackrel{\text{def}}{=} & \{ v_1 - v_2 \mid v_1 \in E[\![e_1]\!] \rho, v_2 \in E[\![e_2]\!] \rho \} \\
E[\![e_1 \times e_2]\!] \rho & \stackrel{\text{def}}{=} & \{ v_1 \times v_2 \mid v_1 \in E[\![e_1]\!] \rho, v_2 \in E[\![e_2]\!] \rho \} \\
E[\![e_1 / e_2]\!] \rho & \stackrel{\text{def}}{=} & \{ v_1 / v_2 \mid v_1 \in E[\![e_1]\!] \rho, v_2 \in E[\![e_2]\!] \rho, v_2 \neq 0 \}
\end{array}
$$

Invariant semantics and properties

Invariant property: true of all program executions.

```
ℓ1 X ← [0,10]; ℓ2
   Y ← 100;
   while ℓ3 X ≥ 0 do ℓ4
      X ← X − 1; ℓ5
      Y ← Y + 10
   done ℓ6
```

$$\begin{cases} \mathcal{X}_1 = \mathcal{E} \\ \mathcal{X}_2 = \mathsf{C}[\![\, X \leftarrow [0,10] \,]\!]\, \mathcal{X}_1 \\ \mathcal{X}_3 = \mathsf{C}[\![\, Y \leftarrow 100 \,]\!]\, \mathcal{X}_2 \cup \mathsf{C}[\![\, Y \leftarrow Y + 10 \,]\!]\, \mathcal{X}_5 \\ \mathcal{X}_4 = \mathsf{C}[\![\, X \geq 0 \,]\!]\, \mathcal{X}_3 \\ \mathcal{X}_5 = \mathsf{C}[\![\, X \leftarrow X - 1 \,]\!]\, \mathcal{X}_4 \\ \mathcal{X}_6 = \mathsf{C}[\![\, X < 0 \,]\!]\, \mathcal{X}_3 \end{cases}$$

(atomic command semantics $\mathsf{C}[\![\, \texttt{com} \,]\!]$ on next slide)

- $\mathcal{X}_i \in \mathcal{P}(\mathcal{E})$: set of memory states at program point $i \in \mathcal{L}$
  e.g.: $\mathcal{X}_3 = \{\, \rho \in \mathcal{E} \mid \rho(X) \in [0,10],\ 10\rho(X) + \rho(Y) \in [100,200] \cap 10\mathbb{Z} \,\}$
- we look for the smallest solution $(\mathcal{X}_i)_{i \in \mathcal{L}}$ of the system
- $I \subseteq \mathcal{E}$ is invariant at i if $\mathcal{X}_i \subseteq I$
- state invariants I can express absence of assertion failures, overflows, memory errors, non-termination, etc.

From programs to equations

Atomic commands: $C[\![\texttt{com}]\!] : \mathcal{P}(\mathcal{E}) \to \mathcal{P}(\mathcal{E})$

$\texttt{com} \stackrel{\text{def}}{=} \{ X \leftarrow \texttt{exp},\ exp \bowtie 0 \}$: assignments and tests.

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

$C[\![\cdot]\!]$ are $\cup$−morphisms: $C[\![s]\!] \mathcal{X} = \cup_{\rho \in \mathcal{X}} C[\![s]\!] \{\rho\}$, monotonic, continuous

Systematic derivation of the equation system: $eq({}^{\ell}stat^{\ell'})$

by structural induction:

$eq({}^{\ell1}X \leftarrow e^{\ell2}) \stackrel{\text{def}}{=} \{ \mathcal{X}_{\ell2} = C[\![X \leftarrow e]\!] \mathcal{X}_{\ell1} \}$

$eq({}^{\ell1}s_1;{}^{\ell2}s_2{}^{\ell3}) \stackrel{\text{def}}{=} eq({}^{\ell1}s_1{}^{\ell2}) \cup ({}^{\ell2}s_2{}^{\ell3})$

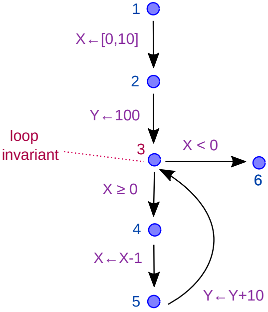
$eq({}^{\ell1}\textbf{if } e \bowtie 0 \textbf{ then } {}^{\ell2}s^{\ell3}) \stackrel{\text{def}}{=}$
$\{ \mathcal{X}_{\ell2} = C[\![e \bowtie 0]\!] \mathcal{X}_{\ell1} \} \cup eq({}^{\ell2}s^{\ell3'}) \cup \{ \mathcal{X}_{\ell3} = \mathcal{X}_{\ell3'} \cup C[\![e \not\bowtie 0]\!] \mathcal{X}_{\ell1} \}$

$eq({}^{\ell1}\textbf{while } {}^{\ell2}e \bowtie 0 \textbf{ do } {}^{\ell3}s^{\ell4} \textbf{ done}^{\ell5}) \stackrel{\text{def}}{=}$
$\{ \mathcal{X}_{\ell2} = \mathcal{X}_{\ell1} \cup \mathcal{X}_{\ell4},\ \mathcal{X}_{\ell3} = C[\![e \bowtie 0]\!] \mathcal{X}_{\ell2} \} \cup eq({}^{\ell3}s^{\ell4}) \cup \{ \mathcal{X}_{\ell5} = C[\![e \not\bowtie 0]\!] \mathcal{X}_{\ell2} \}$

where: $\mathcal{X}^{\ell3'}$ is a fresh variable storing intermediate results

From control-flow graphs to equations

Programs can also be viewed as a control-flow graphs.



→

$$\begin{cases} \mathcal{X}_1 = \mathcal{E} \\ \mathcal{X}_2 = \mathsf{C}[\![\, X \leftarrow [0,10] \,]\!]\, \mathcal{X}_1 \\ \mathcal{X}_3 = \mathsf{C}[\![\, Y \leftarrow 100 \,]\!]\, \mathcal{X}_2 \cup \mathsf{C}[\![\, Y \leftarrow Y + 10 \,]\!]\, \mathcal{X}_5 \\ \mathcal{X}_4 = \mathsf{C}[\![\, X \geq 0 \,]\!]\, \mathcal{X}_3 \\ \mathcal{X}_5 = \mathsf{C}[\![\, X \leftarrow X - 1 \,]\!]\, \mathcal{X}_4 \\ \mathcal{X}_6 = \mathsf{C}[\![\, X < 0 \,]\!]\, \mathcal{X}_3 \end{cases}$$

CFG: $(\mathcal{L}, e, A)$
nodes: $\mathcal{L}$
entry node: $e \in \mathcal{L}$
arcs: $A \subseteq \mathcal{L} \times \texttt{com} \times \mathcal{L}$

→

$(\mathcal{X}_i)_{i \in \mathcal{L}}$

$$\begin{cases} \mathcal{X}_e = \mathcal{E} \\ \mathcal{X}_i = \bigcup_{(j,c,i) \in A} \mathsf{C}[\![\, c \,]\!]\, \mathcal{X}_j \quad \text{if } i \neq e \end{cases}$$

Benefit: can also reason on unstructured programs.

Transition semantics

Program execution as discrete transitions between states.

- Σ: set of states
- $\tau \subseteq \Sigma \times \Sigma$: a transition relation, written $\sigma \rightarrow_\tau \sigma'$, or $\sigma \rightarrow \sigma'$
  (sometimes, we use *labelled* transition systems instead: $\tau \subseteq \Sigma \times \mathcal{A} \times \Sigma$, $\sigma \stackrel{a}{\rightarrow} \sigma'$)

$\Longrightarrow$ a form of small-step semantics.

Application: on our programming language

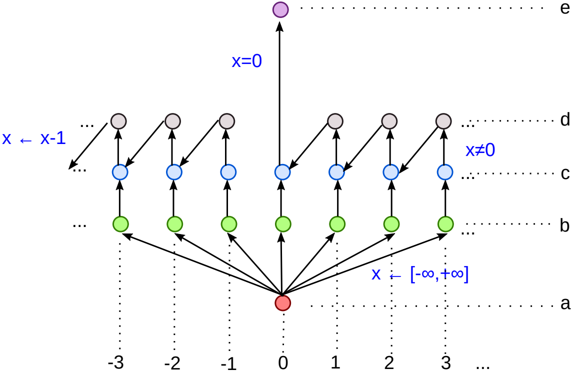
- $\Sigma \stackrel{\text{def}}{=} \mathcal{L} \times \mathcal{E}$: a control point and a memory state
- initial states $\mathcal{I} \stackrel{\text{def}}{=} \{\ell\} \times \mathcal{E}$ and
  final states $\mathcal{F} \stackrel{\text{def}}{=} \{\ell'\} \times \mathcal{E}$ for program ${}^{\ell}\texttt{stat}^{\ell'}$
- τ defined by structural induction on ${}^{\ell}\texttt{stat}^{\ell'}$ (next slides)

but transition systems can model many other languages: imperative languages, $\lambda-$calculus, abstract machines, concurrent programs, mobile systems, ...

Transition semantics example

Example

${}^{a}X \leftarrow [-\infty, \infty];$
^{b}**while** ${}^{c}X \neq 0$ **do** ${}^{d}X \leftarrow X - 1$ **done** ^{e}



From programs to transition relations

Transitions: $\tau[{}^{\ell}stat^{\ell'}] \subseteq \Sigma \times \Sigma$

$$\tau[{}^{\ell1}X \leftarrow e^{\ell2}] \stackrel{\text{def}}{=} \{\, (\ell1, \rho) \to (\ell2, \rho[X \mapsto v]) \mid \rho \in \mathcal{E},\ v \in \mathsf{E}[\![\, e \,]\!]\, \rho \,\}$$

$$\begin{array}{l}\tau[{}^{\ell1}\textbf{if } e \bowtie 0 \textbf{ then } {}^{\ell2}s^{\ell3}] \stackrel{\text{def}}{=} \\ \quad \{\, (\ell1, \rho) \to (\ell2, \rho) \mid \rho \in \mathcal{E},\ \exists v \in \mathsf{E}[\![\, e \,]\!]\, \rho \colon v \bowtie 0 \,\} \cup \\ \quad \{\, (\ell1, \rho) \to (\ell3, \rho) \mid \rho \in \mathcal{E},\ \exists v \in \mathsf{E}[\![\, e \,]\!]\, \rho \colon v \not\bowtie 0 \,\} \cup \tau[{}^{\ell2}s^{\ell3}]\end{array}$$

$$\begin{array}{l}\tau[{}^{\ell1}\textbf{while } {}^{\ell2}e \bowtie 0 \textbf{ do } {}^{\ell3}s^{\ell4} \textbf{ done}^{\ell5}] \stackrel{\text{def}}{=} \\ \quad \{\, (\ell1, \rho) \to (\ell2, \rho) \mid \rho \in \mathcal{E} \,\} \cup \\ \quad \{\, (\ell2, \rho) \to (\ell3, \rho) \mid \rho \in \mathcal{E},\ \exists v \in \mathsf{E}[\![\, e \,]\!]\, \rho \colon v \bowtie 0 \,\} \cup \tau[{}^{\ell3}s^{\ell4}] \cup \\ \quad \{\, (\ell4, \rho) \to (\ell2, \rho) \mid \rho \in \mathcal{E} \,\} \cup \\ \quad \{\, (\ell2, \rho) \to (\ell5, \rho) \mid \rho \in \mathcal{E},\ \exists v \in \mathsf{E}[\![\, e \,]\!]\, \rho \colon v \not\bowtie 0 \,\}\end{array}$$

$$\tau[{}^{\ell1}s_1; {}^{\ell2}s_2{}^{\ell3}] \stackrel{\text{def}}{=} \tau[{}^{\ell1}s_1{}^{\ell2}] \cup \tau[{}^{\ell2}s_2{}^{\ell3}]$$

Reachability semantics and post-conditions

Reachability semantics

- $\mathcal{R} \subseteq \Sigma$ states reachable from $\mathcal{I}$ by τ (transitively)
- $\mathcal{R} \cap \mathcal{F}$ final reachable states
  $\Longrightarrow$ we can check program post-conditions and non-termination

Link with the equational semantics

$\mathcal{R} \cap (\{i\} \times \mathcal{E}) = \{i\} \times \mathcal{X}_i \simeq \mathcal{X}_i$ ($\mathcal{X}_i$ are the reachable states at $i \in \mathcal{L}$)

Alternate form for reachability

$\mathsf{C}[\![\, \texttt{stat} \,]\!]\, \mathcal{I} \subseteq \mathcal{E}$ defined by structural induction:

- $\mathsf{C}[\![\, X \leftarrow e \,]\!]$ and $\mathsf{C}[\![\, e \bowtie 0 \,]\!]$ as in the equational semantics
- $\mathsf{C}[\![\, s_1;\ s_2 \,]\!]\, \mathcal{X} \stackrel{\text{def}}{=} \mathsf{C}[\![\, s_2 \,]\!]\, (\mathsf{C}[\![\, s_1 \,]\!]\, \mathcal{X})$
- $\mathsf{C}[\![\, \textbf{if } e \bowtie 0 \textbf{ then } s \,]\!]\, \mathcal{X} \stackrel{\text{def}}{=} (\mathsf{C}[\![\, s \,]\!]\, (\mathsf{C}[\![\, e \bowtie 0 \,]\!]\, \mathcal{X})) \cup (\mathsf{C}[\![\, e \not\bowtie 0 \,]\!]\, \mathcal{X})$
- $\mathsf{C}[\![\, \textbf{while } e \bowtie 0 \textbf{ do } s \textbf{ done} \,]\!]\, \mathcal{X} \stackrel{\text{def}}{=} \mathsf{C}[\![\, e \not\bowtie 0 \,]\!]\, (\cup_{i \geq 0}\, (\mathsf{C}[\![\, s \,]\!] \circ \mathsf{C}[\![\, e \bowtie 0 \,]\!])^i \mathcal{X})$

Trace semantics

Semantics:

- trace: a sequence of states (finite or infinite)
- execution trace: a sequence of states linked by the transition relation τ

The semantics of a program is now a set of traces.

Trace properties:

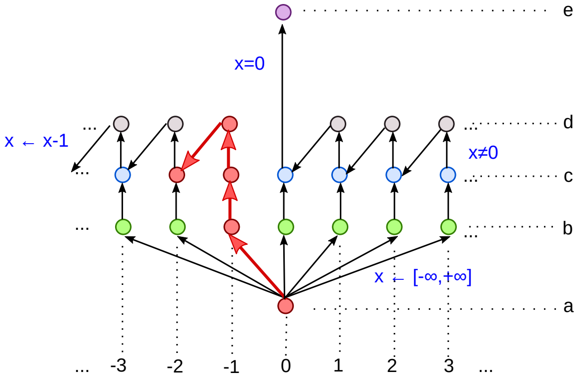
Traces carry more information than states
and can prove more expressive properties:

- temporal properties (a occurs before b)
- computation length (possibly infinite)
- liveness (termination, inevitability)

Trace semantics example

Example

$^{a}X \leftarrow [-\infty, \infty];$
^{b}**while** $^{c}X \neq 0$ **do** $^{d}X \leftarrow X - 1$ **done** ^{e}



Roadmap

Goal:

- express all these semantics as fixpoints
- relate these semantics by abstraction relations
- introduce variants (backward semantics, infinite trace semantics, ...)
- study which semantics to choose for which class of properties
- beyond trace properties

Caveat:

- start generally from transition systems (not high-level syntax)
  $\Longrightarrow$ uniform view of semantics independent from programming language
- remain at the level of concrete collecting semantics
  - express precisely all properties in a class of interest
  - uncomputable

the next course will return to numeric programs
and introduce computable abstractions to achieve computable static analysis

State semantics and properties

Forward semantics

Forward reachability

Forward image: $\text{post}_\tau : \mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma)$

$$\text{post}_\tau(S) \stackrel{\text{def}}{=} \{\, \sigma' \mid \exists \sigma \in S \colon \sigma \to \sigma' \,\}$$

post_τ is a strict, complete $\cup-$morphism in $(\mathcal{P}(\Sigma), \subseteq, \cup, \cap, \emptyset, \Sigma)$.

$\text{post}_\tau(\cup_{i \in I} S_i) = \cup_{i \in I} \text{post}_\tau(S_i)$, $\text{post}_\tau(\emptyset) = \emptyset$

Blocking states: $\mathcal{B} \stackrel{\text{def}}{=} \{\, \sigma \mid \forall \sigma' \in \Sigma \colon \sigma \not\to \sigma' \,\}$

(states with no successor: valid final states but also errors)

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

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

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

Fixpoint formulation of forward reachability

$\mathcal{R}(\mathcal{I})$ 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)

Fixpoint formulation proof

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

$(\mathcal{P}(\Sigma), \subseteq)$ 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, $\text{lfp}\, F_\mathcal{R} = \cup_{n\in\mathbb{N}} F_\mathcal{R}^n(\emptyset)$.

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

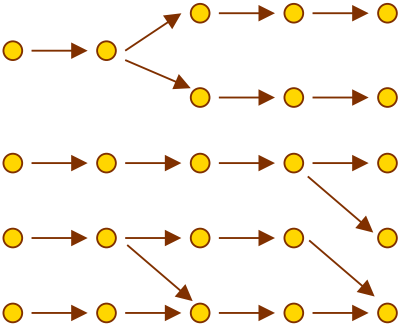
- $F_\mathcal{R}^0(\emptyset) = \emptyset$
- assuming the property at n,

$$\begin{array}{rcl} F_\mathcal{R}^{n+1}(\emptyset) & = & F_\mathcal{R}(\bigcup_{i<n} \text{post}_\tau^i(\mathcal{I})) \\ & = & \mathcal{I} \cup \text{post}_\tau(\bigcup_{i<n} \text{post}_\tau^i(\mathcal{I})) \\ & = & \mathcal{I} \cup \bigcup_{i<n} \text{post}_\tau(\text{post}_\tau^i(\mathcal{I})) \\ & = & \mathcal{I} \cup \bigcup_{1\leq i<n+1} \text{post}_\tau^i(\mathcal{I}) \\ & = & \bigcup_{i<n+1} \text{post}_\tau^i(\mathcal{I}) \end{array}$$

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

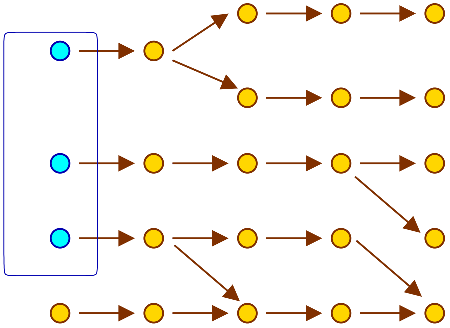
The proof is similar for the alternate form, given that $\text{lfp}_\mathcal{I}\, G_\mathcal{R} = \cup_{n\in\mathbb{N}} G_\mathcal{R}^n(\mathcal{I})$ and $G_\mathcal{R}^n(\mathcal{I}) = F_\mathcal{R}^{n+1}(\emptyset) = \cup_{i\leq n} \text{post}_\tau^i(\mathcal{I})$.

Graphical illustration



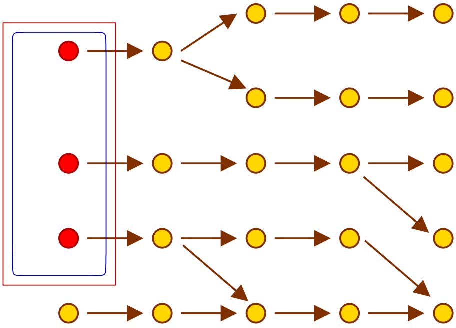
Transition system.

Graphical illustration



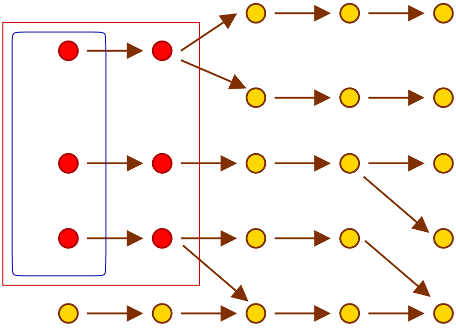
Initial states $\mathcal{I}$.

Graphical illustration



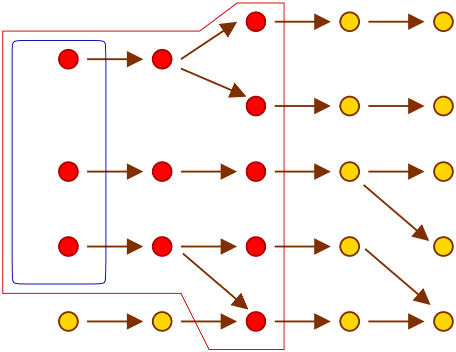
Iterate $F^1_{\mathcal{R}}(\mathcal{I})$.

Graphical illustration



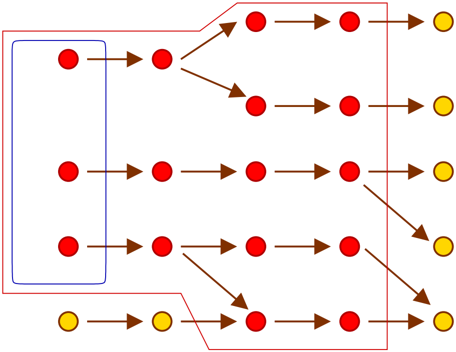
Iterate $F^2_{\mathcal{R}}(\mathcal{I})$.

Graphical illustration



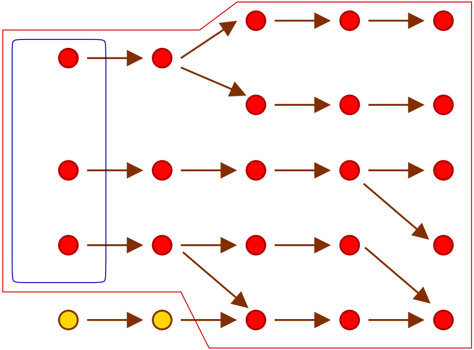
Iterate $F_{\mathcal{R}}^3(\mathcal{I})$.

Graphical illustration



Iterate $F^4_{\mathcal{R}}(\mathcal{I})$.

Graphical illustration



Iterate $F_{\mathcal{R}}^{5}(\mathcal{I})$.

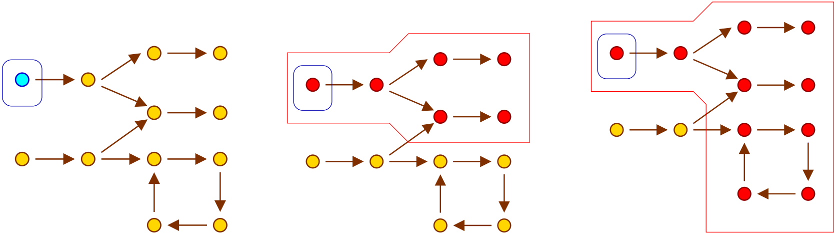
$F_{\mathcal{R}}^{6}(\mathcal{I}) = F_{\mathcal{R}}^{5}(\mathcal{I}) \Rightarrow$ we reached a fixpoint $\mathcal{R}(\mathcal{I}) = F_{\mathcal{R}}^{5}(\mathcal{I})$.

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: $\mathcal{R}(\mathcal{I}) \cap \mathcal{F}$.

```
• i ← 0;
  while i < 100 do
    i ← i + 1;
    j ← j + [0, 1]
  done •
```

  - initial states $\mathcal{I}$: $j \in [0, 10]$ at control point •,
  - final states $\mathcal{F}$: any memory state at control point •,
  - $\implies \mathcal{R}(\mathcal{I}) \cap \mathcal{F}$: 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 $\mathcal{R}^\sharp(\mathcal{I}) \cap \mathcal{B} \subseteq \mathcal{F} \implies \mathcal{R}(\mathcal{I}) \cap \mathcal{B} \subseteq \mathcal{F}$)

Link with invariance proof methods

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 $\mathcal{R}(\mathcal{I})$:

- if I is an inductive invariant, then $F_{\mathcal{R}}(I) \subseteq I$
  $F_{\mathcal{R}}(I) = \mathcal{I} \cup \text{post}_\tau(I) \subseteq I \cup I = I$
  $\implies$ an inductive invariant is a post-fixpoint of $F_{\mathcal{R}}$
- $\mathcal{R}(\mathcal{I}) = \text{lfp}\, F_{\mathcal{R}}$
  $\implies$ $\mathcal{R}(\mathcal{I})$ is the tightest inductive invariant

Link with the equational semantics

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

Control point partitioning

As $\Sigma \stackrel{\text{def}}{=} \mathcal{L} \times \mathcal{E}$, $\mathcal{P}(\Sigma) \simeq \mathcal{L} \to \mathcal{P}(\mathcal{E})$.

We have a Galois isomorphism:

$$(\mathcal{P}(\Sigma), \subseteq) \xleftarrow[\alpha_{\mathcal{L}}]{\gamma_{\mathcal{L}}} (\mathcal{L} \to \mathcal{P}(\mathcal{E}), \dot{\subseteq})$$

- $X \dot{\subseteq} Y \stackrel{\text{def}}{\iff} \forall \ell \in \mathcal{L}: X(\ell) \subseteq Y(\ell)$
- $\alpha_{\mathcal{L}}(S) \stackrel{\text{def}}{=} \lambda \ell . \{ \rho \,|\, (\ell, \rho) \in S \}$
- $\gamma_{\mathcal{L}}(X) \stackrel{\text{def}}{=} \{ (\ell, \rho) \,|\, \ell \in \mathcal{L}, \rho \in X(\ell) \}$
- given $F_{eq} \stackrel{\text{def}}{=} \alpha_{\mathcal{L}} \circ F_{\mathcal{R}} \circ \gamma_{\mathcal{L}}$
  we get back an equation system $\bigwedge_{\ell \in \mathcal{L}} \mathcal{X}_\ell = F_{eq,\ell}(\mathcal{X}_1, \ldots, \mathcal{X}_n)$
- $\alpha_{\mathcal{L}} \circ \gamma_{\mathcal{L}} = \gamma_{\mathcal{L}} \circ \alpha_{\mathcal{L}} = id$ (no abstraction)
  simply reorganize the states by control point
  after actual abstraction, partitioning makes a difference (flow-sensitivity)

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 $\mathcal{R}(\mathcal{I})$:

$F_{eq} \stackrel{\text{def}}{=} \alpha_{\mathcal{L}} \circ F_{\mathcal{R}} \circ \gamma_{\mathcal{L}}$ partitions $F_{\mathcal{R}}$ by control point and lfp $F_{\mathcal{R}}$ gives the tightest inductive invariant

- any post-fixpoint of F_{eq} gives valid Hoare triples
- lfp F_{eq} gives the tightest Hoare triples

Solving the equational semantics

Solve $\bigwedge_{i \in [1,n]} \mathcal{X}_i = F_i(\mathcal{X}_1, \ldots, \mathcal{X}_n)$

Each F_i is continuous in $\mathcal{P}(\mathcal{E})^n \to \mathcal{P}(\mathcal{E})$ (complete $\cup$−morphism)

aka $\vec{F} \stackrel{\text{def}}{=} (F_1, \ldots, F_n)$ is continuous in $\mathcal{P}(\mathcal{E})^n \to \mathcal{P}(\mathcal{E})^n$

By Taski's fixpoint theorem, lfp $\vec{F}$ exists.

Tarksi's theorem: Jacobi iterations

$$\begin{cases} \mathcal{X}_1^0 \stackrel{\text{def}}{=} \emptyset \\ \ldots \\ \mathcal{X}_i^0 \stackrel{\text{def}}{=} \emptyset \\ \ldots \\ \mathcal{X}_n^0 \stackrel{\text{def}}{=} \emptyset \end{cases} \qquad \begin{cases} \mathcal{X}_1^{k+1} \stackrel{\text{def}}{=} F_1(\mathcal{X}_1^k, \ldots, \mathcal{X}_n^k) \\ \ldots \\ \mathcal{X}_i^{k+1} \stackrel{\text{def}}{=} F_i(\mathcal{X}_1^k, \ldots, \mathcal{X}_n^k) \\ \ldots \\ \mathcal{X}_n^{k+1} \stackrel{\text{def}}{=} F_n(\mathcal{X}_1^k, \ldots, \mathcal{X}_n^k) \end{cases}$$

The limit of $(\mathcal{X}_1^k, \ldots, X_n^k)$ is lfp $\vec{F}$.

Naïve application of Tarski's theorem
called Jacobi iterations by analogy with linear algebra

Solving the equational semantics (cont.)

Other iteration techniques exist [Cous92].

Gauss-Seidl iterations

$$\begin{cases} \mathcal{X}_1^{k+1} \stackrel{\text{def}}{=} F_1(\mathcal{X}_1^k, \ldots, \mathcal{X}_n^k) \\ \ldots \\ \mathcal{X}_i^{k+1} \stackrel{\text{def}}{=} F_i(\mathcal{X}_1^{k+1}, \ldots, \mathcal{X}_{i-1}^{k+1}, \mathcal{X}_i^k, \ldots, \mathcal{X}_n^k) \\ \ldots \\ \mathcal{X}_n^{k+1} \stackrel{\text{def}}{=} F_n(\mathcal{X}_1^{k+1}, \ldots, \mathcal{X}_{n-1}^{k+1}, \mathcal{X}_n^k) \end{cases}$$

use new results as soon available

Chaotic iterations

$$\mathcal{X}_i^{k+1} \stackrel{\text{def}}{=} \begin{cases} F_i(\mathcal{X}_1^k, \ldots, \mathcal{X}_n^k) & \text{if } i = \phi(k+1) \\ \mathcal{X}_i^k & \text{otherwise} \end{cases}$$

wrt. a fair schedule $\phi : \mathbb{N} \to [1, n]$
$\forall i \in [1, n]: \forall N > 0: \exists k > N: \phi(k) = i$

- worklist algorithms
- asynchonous iterations (parallel versions of chaotic iterations)

all give the same limit! (this will not be the case for abstract static analyses. . .)

Inductive abstract interpreter

Principle:

- follow the control-flow of the program
- replace the global fixpoint with local fixpoints (loops)

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

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

$$\mathsf{C}[\![\, s_1; s_2 \,]\!]\,\mathcal{X} \stackrel{\text{def}}{=} \mathsf{C}[\![\, s_2 \,]\!]\,(\mathsf{C}[\![\, s_1 \,]\!]\,\mathcal{X})$$

$$\mathsf{C}[\![\, \textbf{if } e \bowtie 0 \textbf{ then } s \,]\!]\,\mathcal{X} \stackrel{\text{def}}{=} (\mathsf{C}[\![\, s \,]\!]\,(\mathsf{C}[\![\, e \bowtie 0 \,]\!]\,\mathcal{X})) \cup (\mathsf{C}[\![\, e \not\bowtie 0 \,]\!]\,\mathcal{X})$$

$$\mathsf{C}[\![\, \textbf{while } e \bowtie 0 \textbf{ do } s \textbf{ done} \,]\!]\,\mathcal{X} \stackrel{\text{def}}{=} \mathsf{C}[\![\, e \not\bowtie 0 \,]\!]\,(\mathrm{lfp}\, F)$$

$$\text{where } F(\mathcal{Y}) \stackrel{\text{def}}{=} \mathcal{X} \cup \mathsf{C}[\![\, s \,]\!]\,(\mathsf{C}[\![\, e \bowtie 0 \,]\!]\,\mathcal{Y})$$

informal justification for the loop semantics:

All the $\mathsf{C}[\![\, s \,]\!]$ functions are continuous, hence the fixoints exist.
By induction on k, $F^k(\emptyset) = \cup_{i \le k}\, (\mathsf{C}[\![\, s \,]\!] \circ \mathsf{C}[\![\, e \bowtie 0 \,]\!]\,)^i \mathcal{X}$
hence, $\mathrm{lfp}\, F = \cup_i\, (\mathsf{C}[\![\, s \,]\!] \circ \mathsf{C}[\![\, e \bowtie 0 \,]\!]\,)^i \mathcal{X}$
We fall back to a special case of (transfinite) chaotic iteration that stabilizes loops depth-first.

Backward semantics

Backward co-reachability

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

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

where $\text{pre}_\tau(S) \stackrel{\text{def}}{=} \{\, \sigma \mid \exists \sigma' \in S : \sigma \rightarrow \sigma' \,\}$ $\quad (\text{pre}_\tau = \text{post}_{\tau^{-1}})$

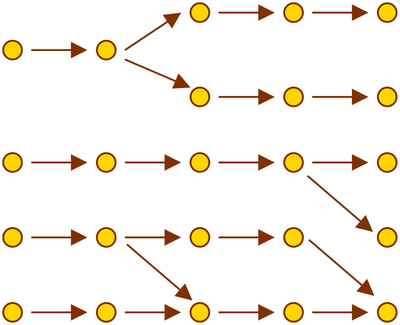
$\mathcal{C}(\mathcal{F})$ can also be expressed in fixpoint form:

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

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

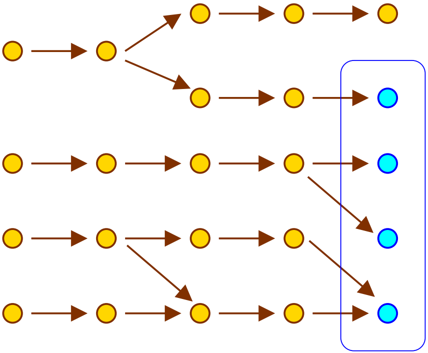
Alternate characterization: $\mathcal{C}(\mathcal{F}) = \text{lfp}_\mathcal{F}\; G_\mathcal{C}$ where $G_\mathcal{C}(S) = S \cup \text{pre}_\tau(S)$

Graphical illustration



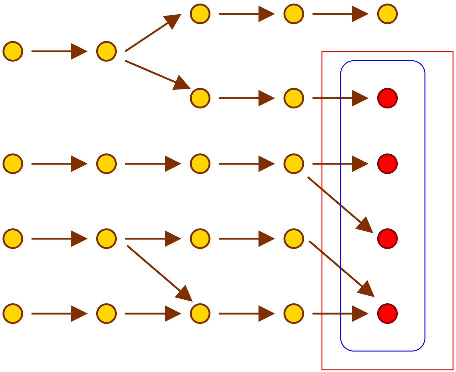
Transition system.

Graphical illustration

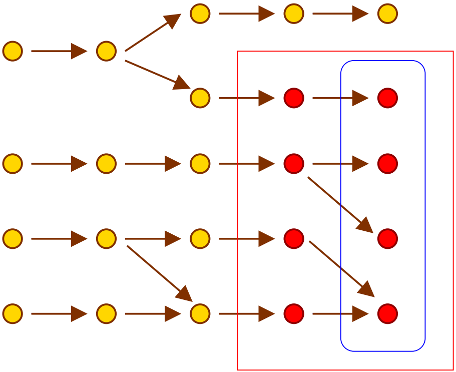


Final states $\mathcal{F}$.

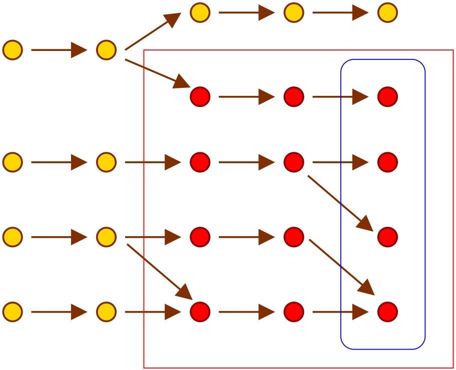
Graphical illustration



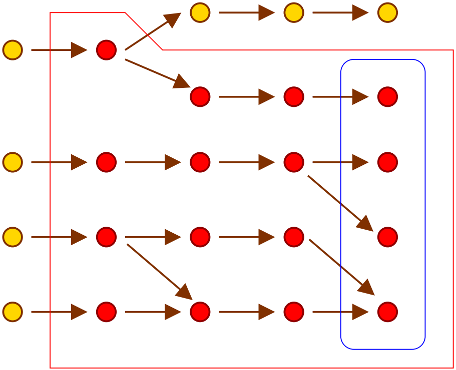
Graphical illustration



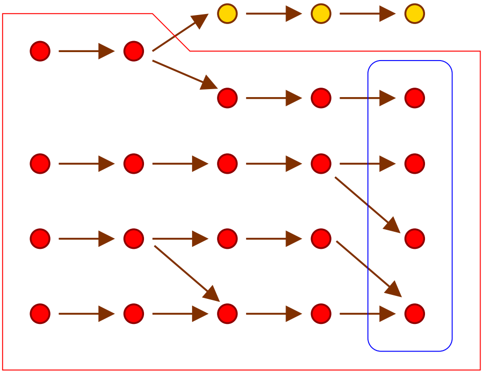
Graphical illustration



Graphical illustration



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.

```
• j ← 0;
  while i > 0 do
    i ← i − 1;
    j ← j + [0, 10]
    assert (j ≤ 200)
  done •
```

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

- Over-approximating $\mathcal{C}$ is useful to isolate possibly incorrect executions from those guaranteed to be correct.
- Iterate forward and backward analyses interactively
  $\implies$ abstract debugging [Bour93].

Backward co-reachability in equational form

Principle:

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

Example:

```
ℓ1 j ← 0;
ℓ2 while ℓ3 i > 0 do
     ℓ4 i ← i − 1;
     ℓ5 j ← j + [0, 10]
ℓ6
```

$$
\begin{aligned}
\mathcal{X}_1 &= \overleftarrow{C}[\![j \to 0]\!]\, \mathcal{X}_2 \\
\mathcal{X}_2 &= \mathcal{X}_3 \\
\mathcal{X}_3 &= \overleftarrow{C}[\![i > 0]\!]\, \mathcal{X}_4 \cup \overleftarrow{C}[\![i \leq 0]\!]\, \mathcal{X}_6 \\
\mathcal{X}_4 &= \overleftarrow{C}[\![i \leftarrow i - 1]\!]\, \mathcal{X}_5 \\
\mathcal{X}_5 &= \overleftarrow{C}[\![j \leftarrow j + [0, 10]]\!]\, \mathcal{X}_3 \\
\mathcal{X}_6 &= \mathcal{F}
\end{aligned}
$$

- final states $\{\ell 6\} \times \mathcal{F}$.
- $\overleftarrow{C}[\![X \leftarrow e]\!]\, \mathcal{X} \stackrel{\text{def}}{=} \{ \rho \mid \exists v \in E[\![e]\!]\, \rho: \rho[X \mapsto v] \in \mathcal{X} \}$.
- $\overleftarrow{C}[\![e \bowtie 0]\!]\, \mathcal{X} \stackrel{\text{def}}{=} \{ \rho \in \mathcal{X} \mid \exists v \in E[\![\rho]\!]\, \rho: v \bowtie 0 \} = C[\![e \bowtie 0]\!]\, \mathcal{X}$

(also possible on control-flow graphs. . .)

Sufficient precondition semantics

Sufficient preconditions

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

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

where $\widetilde{\text{pre}}_\tau(S) \stackrel{\text{def}}{=} \{\, \sigma \mid \forall \sigma' \colon \sigma \to \sigma' \implies \sigma' \in S \,\}$

(states such that all successors satisfy S, $\widetilde{\text{pre}}$ is a complete $\cap-$morphism)

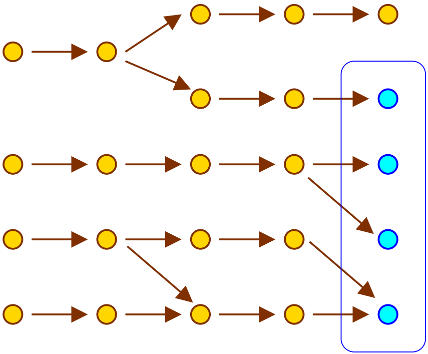
$\mathcal{S}(\mathcal{Y})$ can be expressed in fixpoint form:

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

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, $\text{gfp}\, F_\mathcal{S} = \cap_{n \in \mathbb{N}} F_\mathcal{S}^n(\Sigma)$.
We would prove by recurrence that $F_\mathcal{S}^n(\Sigma) = \cap_{i<n} \widetilde{\text{pre}}_\tau^i(\mathcal{Y})$.

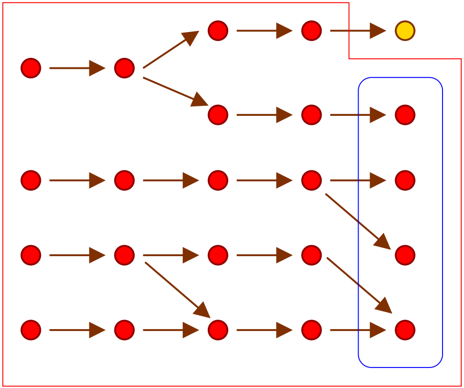
Graphical illustration



Final states $\mathcal{F}$.

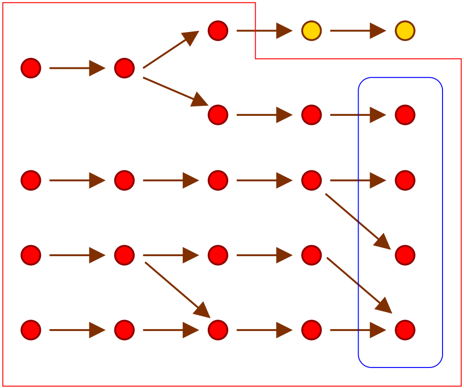
Goal: when stopping, stop in $\mathcal{F}$

Graphical illustration



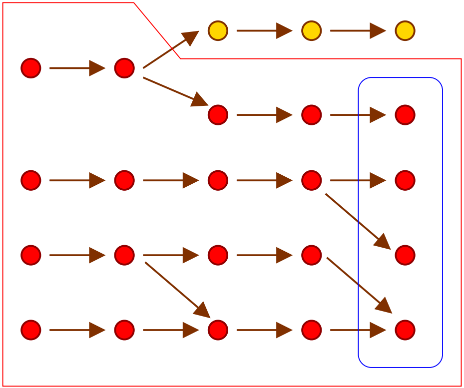
Final states $\mathcal{F}$.
Goal: stay in $\mathcal{Y} = \mathcal{F} \cup (\Sigma \setminus \mathcal{B})$
Iteration $F^0_{\mathcal{S}}(\mathcal{Y})$

Graphical illustration



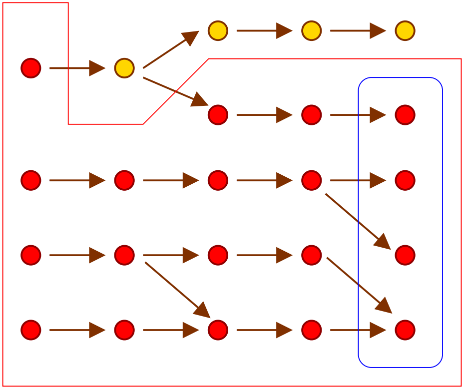
Final states $\mathcal{F}$.
Goal: stay in $\mathcal{Y} = \mathcal{F} \cup (\Sigma \setminus \mathcal{B})$
Iteration $F_{\mathcal{S}}^1(\mathcal{Y})$

Graphical illustration



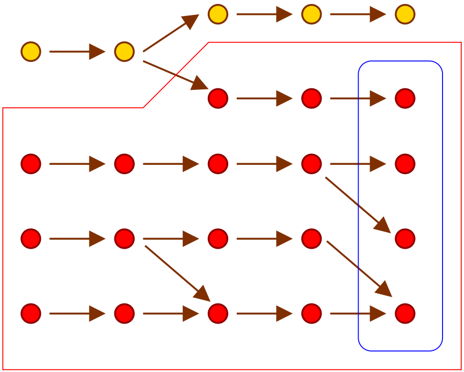
Final states $\mathcal{F}$.
Goal: stay in $\mathcal{Y} = \mathcal{F} \cup (\Sigma \setminus \mathcal{B})$
Iteration $F_{\mathcal{S}}^2(\mathcal{Y})$

Graphical illustration



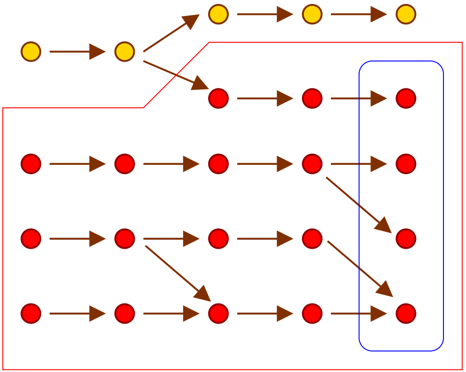
Final states $\mathcal{F}$.
Goal: stay in $\mathcal{Y} = \mathcal{F} \cup (\Sigma \setminus \mathcal{B})$
Iteration $F_{\mathcal{S}}^3(\mathcal{Y})$

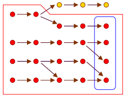
Graphical illustration



Final states $\mathcal{F}$.
Goal: stay in $\mathcal{Y} = \mathcal{F} \cup (\Sigma \setminus \mathcal{B})$
Sufficient preconditions $\mathcal{S}(\mathcal{Y})$ to stop in $\mathcal{F}$

Graphical illustration





Final states $\mathcal{F}$.
Goal: stay in $\mathcal{Y} = \mathcal{F} \cup (\Sigma \setminus \mathcal{B})$
Sufficient preconditions $\mathcal{S}(\mathcal{Y})$ to stop in $\mathcal{F}$ $\qquad \mathcal{C}(\mathcal{F})$

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

Sufficient preconditions and reachability

Correspondence with reachability:

We have a Galois connection:

$$(\mathcal{P}(\Sigma), \subseteq) \xleftarrow[\mathcal{R}]{\mathcal{S}} (\mathcal{P}(\Sigma), \subseteq)$$

- $\mathcal{R}(\mathcal{I}) \subseteq \mathcal{Y} \iff \mathcal{I} \subseteq \mathcal{S}(\mathcal{Y})$
  definition of a Galois connection
  all executions from $\mathcal{I}$ stay in $\mathcal{Y}$
  $\iff$ $\mathcal{I}$ includes only sufficient pre-conditions for $\mathcal{Y}$
- so $\mathcal{S}(\mathcal{Y}) = \bigcup \{ X \mid \mathcal{R}(X) \subseteq \mathcal{Y} \}$
  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 $\mathcal{R}(\mathcal{I}) = \text{lfp}_{\mathcal{I}}\, G_{\mathcal{R}}$ where $G_{\mathcal{R}}(S) = S \cup \text{post}_\tau(S)$.
Likewise, $\mathcal{S}(\mathcal{Y}) = \text{gfp}_{\mathcal{Y}}\, G_{\mathcal{S}}$ where $G_{\mathcal{S}}(S) = S \cap \widetilde{\text{pre}}_\tau(S)$.

We have a Galois connection: $(\mathcal{P}(\Sigma), \subseteq) \underset{\text{post}_\tau}{\overset{\widetilde{\text{pre}}_\tau}{\leftrightarrows}} (\mathcal{P}(\Sigma), \subseteq)$.

$$\begin{array}{lll}
\text{post}_\tau(A) \subseteq B & \iff & \{\, \sigma' \mid \exists \sigma \in A\colon \sigma \to \sigma' \,\} \subseteq B \\
& \iff & (\forall \sigma \in A\colon \sigma \to \sigma' \implies \sigma' \in B) \\
& \iff & (A \subseteq \{\, \sigma \mid \forall \sigma'\colon \sigma \to \sigma' \implies \sigma' \in B \,\}) \\
& \iff & A \subseteq \widetilde{\text{pre}}_\tau(B)
\end{array}$$

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:

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

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

$\bullet$ $i \leftarrow 0;$
while $i < 100$ **do**
  $i \leftarrow i + 1;$
  $j \leftarrow j + [0, 1]$
  assert $(j \leq 105)$
done $\bullet$

- initial states $\mathcal{I}$: $j \in [0, 10]$ at $\bullet$
- final states $\mathcal{F}$: any memory state at $\bullet$
- blocking states $\mathcal{B}$: either final or $j > 105$ (assertion failure)
- $\mathcal{I} \cap \mathcal{S}(\mathcal{F} \cup (\Sigma \setminus \mathcal{B}))$: at $\bullet$, $j \in [0, 5]$
  (note that $\mathcal{I} \cap \mathcal{C}(\mathcal{F} \cup (\Sigma \setminus \mathcal{B}))$ gives $\mathcal{I}$)

- application to inferring function contracts
- application to inferring counter-examples
- requires under-approximations to build decidable abstractions but most analyses can only provide over-approximations!
  $\Longrightarrow$ **research topic**

Finite trace semantics

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

- 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

- state sets $\mathcal{I}, \mathcal{F} \subseteq \Sigma$ are also sets of traces, of length 1
- transition relation $\tau \subseteq \Sigma \times \Sigma$ is also a set of traces, of length 2

Trace operations

Operations on traces:

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

  $(\sigma_0, \ldots, \sigma_n) \cdot (\sigma'_0, \ldots, \sigma'_m) \stackrel{\text{def}}{=} \sigma_0, \ldots, \sigma_n, \sigma'_0, \ldots, \sigma'_m$

  $\epsilon \cdot t \stackrel{\text{def}}{=} t \cdot \epsilon \stackrel{\text{def}}{=} t$

- junction $\frown$

  $(\sigma_0, \ldots, \sigma_n) \frown (\sigma'_0, \sigma'_1 \ldots, \sigma'_m) \stackrel{\text{def}}{=} \sigma_0, \ldots, \sigma_n, \sigma'_1, \ldots, \sigma'_m$
  when $\sigma_n = \sigma'_0$

  undefined if $\sigma_n \neq \sigma'_0$, and for ϵ

  (join two consecutive traces, the common element $\sigma_n = \sigma'_0$ is not repeated)

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 $^\frown$

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

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

Note: $\cdot$ and $^\frown$ distribute $\cup$ and $\cap$

$(\cup_{i \in I} A_i)^\frown(\cup_{j \in J} B_i) = \cup_{i \in I, j \in J} (A_i{}^\frown B_j)$, etc.

Finite prefix trace semantics

Prefix trace semantics

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

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

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

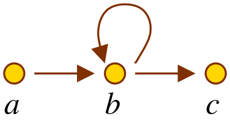
$\mathcal{T}_p(\mathcal{I})$ can be expressed in fixpoint form:

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

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

(proof on next slides)

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: 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 $\forall n$: $F_p^n(\emptyset) = \cup_{i<n}\, \mathcal{I} \frown \tau^{\frown i}$:

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

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: expressive power

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

$\Longrightarrow$ this is the 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)
- cannot distinguish between finished and unfinished executions
  $\Longrightarrow$ no liveness property (see 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:

$$(\mathcal{P}(\Sigma^*), \subseteq) \xrightleftharpoons[\alpha_p]{\gamma_p} (\mathcal{P}(\Sigma), \subseteq)$$

- $\alpha_p(T) \stackrel{\text{def}}{=} \{\, \sigma \in \Sigma \mid \exists \sigma_0, \ldots, \sigma_n \in T \colon \sigma = \sigma_n \,\}$
  (last state in traces in T)
- $\gamma_p(S) \stackrel{\text{def}}{=} \{\, \sigma_0, \ldots, \sigma_n \in \Sigma^* \mid \sigma_n \in S \,\}$
  (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
- $(\gamma_p \circ \alpha_p)(T)$
  $= \{ \sigma_0, \ldots, \sigma_n \mid \sigma_n \in \alpha_p(T) \}$
  $= \{ \sigma_0, \ldots, \sigma_n \mid \exists \sigma'_0, \ldots, \sigma'_m \in T : \sigma_n = \sigma'_m \}$
  $\supseteq T$
- $(\alpha_p \circ \gamma_p)(S)$
  $= \{ \sigma \mid \exists \sigma_0, \ldots, \sigma_n \in \gamma_p(S) : \sigma = \sigma_n \}$
  $= \{ \sigma \mid \exists \sigma_0, \ldots, \sigma_n : \sigma_n \in S,\ \sigma = \sigma_n \}$
  $= S$

Abstracting prefix trace semantics into reachability

We can abstract semantic operators and their least fixpoint.

Recall that:

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

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 $\alpha_p \circ F_p = F_{\mathcal{R}} \circ \alpha_p$

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

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.

Prefix closure

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

$$x \preceq y \overset{\text{def}}{\iff} \exists u \in \Sigma^* \colon x \cdot u = y$$

Note: $(\Sigma^*, \preceq)$ is not a CPO

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

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

ρ_p is an upper closure operator on $\mathcal{P}(\Sigma^* \setminus \{\epsilon\})$.

(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:

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

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

  (set of all states appearing in some trace in T)

- $\gamma_o(S) \stackrel{\text{def}}{=} \{ \sigma_0, \ldots, \sigma_n \mid n \geq 0, \forall i \leq n \colon \sigma_i \in S \}$

  (traces composed of elements from S)

proof sketch:

α_o and γ_o are monotonic, and $\alpha_o \circ \gamma_o = id$.

$(\gamma_o \circ \alpha_o)(T) = \{ \sigma_0, \ldots, \sigma_n \mid \forall i \leq n \colon \exists \sigma'_0, \ldots, \sigma'_m \in T, j \leq m \colon \sigma_i = \sigma'_j \} \supseteq T$.

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 $\mathcal{T}_p(\mathcal{I}) = \text{lfp}\, F_p$ where $F_p(T) \stackrel{\text{def}}{=} \mathcal{I} \cup T \frown \tau$ and $\mathcal{R}(\mathcal{I}) = \text{lfp}\, F_{\mathcal{R}}$ where $F_{\mathcal{R}}(S) \stackrel{\text{def}}{=} \mathcal{I} \cup \text{post}_\tau(S)$, 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 $\text{lfp}\, F_p$ and $\text{lfp}\, F_{\mathcal{R}}$ satisfy $\forall n: \alpha_o(a_p^n) = a_{\mathcal{R}}^n$, and so $\alpha_o(\text{lfp}\, F_p) = \text{lfp}\, F_{\mathcal{R}}$.

Finite suffix trace semantics

Suffix trace semantics

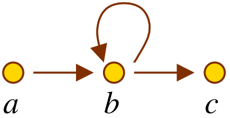
Similar results on the suffix trace semantics,
going backwards from $\mathcal{F}$:

- $\mathcal{T}_s(\mathcal{F}) \stackrel{\text{def}}{=} \{ \sigma_0, \ldots, \sigma_n \mid n \geq 0, \sigma_n \in \mathcal{F}, \forall i: \sigma_i \to \sigma_{i+1} \}$
  (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 $\mathcal{F}$)

Backward state co-rechability abstracts the suffix trace semantics:

- $\alpha_s(\mathcal{T}_s(\mathcal{F})) = \mathcal{C}(\mathcal{F})$ 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 $\rho_s(T) \stackrel{\text{def}}{=} \{ u \mid \exists t \in \Sigma^*: t \cdot u \in T, u \neq \epsilon \}$
  (closed by suffix)

Graphical illustration



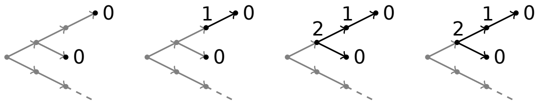
$\mathcal{F} \stackrel{\text{def}}{=} \{c\}$
$\tau \stackrel{\text{def}}{=} \{(a,b),(b,b),(b,c)\}$

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 into a static analysis
  (work by Cousot & Cousot & Urban)

Finite partial trace semantics

Symmetric finite partial trace semantics

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

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

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

The semantics (and iterates) are forward/backward symmetric:

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

Abstracting partial traces into prefix traces

Prefix traces abstract partial traces
as we forget all about partial traces not starting in $\mathcal{I}$.

Galois connection:

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

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

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

similarly for the suffix traces: $\mathcal{T}_s(\mathcal{F}) = \alpha_{\mathcal{F}}(\mathcal{T})$ where $\alpha_{\mathcal{F}}(T) \stackrel{\text{def}}{=} \mathcal{T} \cap (\Sigma^* \cdot \mathcal{F})$

(proof on next slide)

Abstracting partial traces into 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 $\mathcal{T}$.

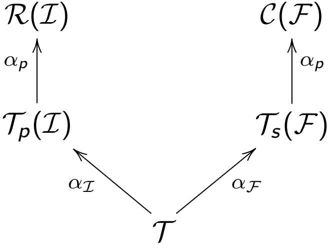
We can also retrieve the result by fixpoint transfer.

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

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

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

A first hierarchy of semantics



forward/backward states

prefix/suffix traces

partial finite traces

Maximal trace semantics

The need for maximal traces

The partial trace semantics cannot distinguish between:

while ^{a} $0 = 0$ **do done**

while ^{a} $[0, 1] = 0$ **do done**

(we get a^* for both programs)

Principle: restrict the semantics to complete executions only

- keep only executions finishing in a blocking state $\mathcal{B}$
- add back infinite executions

  the partial semantics took into account infinite execution by including all their finite parts, but we no longer keep them as they are not maximal!

Benefit:

- avoid confusing prefix of infinite executions with finite executions
- allow reasoning on trace length
- allow reasoning on infinite traces (non-termination, inevitability, liveness)

Infinite traces

Notations:

- $\sigma_0, \ldots, \sigma_n, \ldots$: an infinite trace (length ω)
- Σ^ω: the set of all infinite traces
- $\Sigma^\infty \stackrel{\text{def}}{=} \Sigma^* \cup \Sigma^\omega$: the set of all traces

Extending the operators:

- $(\sigma_0, \ldots, \sigma_n) \cdot (\sigma'_0, \ldots) \stackrel{\text{def}}{=} \sigma_0, \ldots, \sigma_n, \sigma'_0, \ldots$
  (append to a finite trace)
- $t \cdot t' \stackrel{\text{def}}{=} t$ if $t \in \Sigma^\omega$ (append to an infinite trace does nothing)
- $(\sigma_0, \ldots, \sigma_n) ^\frown (\sigma'_0, \sigma'_1 \ldots) \stackrel{\text{def}}{=} \sigma_0, \ldots, \sigma_n, \sigma'_1, \ldots$ when $\sigma_n = \sigma'_0$
- $t ^\frown t' \stackrel{\text{def}}{=} t$, if $t \in \Sigma^\omega$
- prefix: $x \preceq y \stackrel{\text{def}}{\iff} \exists u \in \Sigma^\omega : x \cdot u = y$ $(\Sigma^\omega, \preceq)$ is a CPO

$\cdot$ distributes infinite $\cup$ and $\cap$

$^\frown$ distributes infinite $\cup$, but not infinite $\cap$

$\{a^\omega\} ^\frown (\cap_{n \in \mathbb{N}} \{ a^m \mid n \geq m \}) = \{a^\omega\} ^\frown \emptyset = \emptyset$ but
$\cap_{n \in \mathbb{N}} (\{a^\omega\} ^\frown \{ a^m \mid n \geq m \}) = \cap_{n \in \mathbb{N}} \{a^\omega\} = \{a^\omega\}$
However $A ^\frown (\cap_{i \in I} B_i) = \cup_{i \in I} (A ^\frown B_i)$ if $A \subseteq \Sigma^*$.

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.

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

(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:

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

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

- Infinite traces:

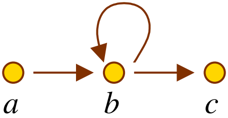
  Additionally, we will prove: $\mathcal{M}_\infty \cap \Sigma^\omega = \text{gfp } G_s$

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

Note: only backward fixpoint formulation of maximal traces exist!

(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: gfp $G_s = \cap_{n\in\mathbb{N}} G_s^n(\Sigma^\omega)$.
We prove by recurrence on n that $\forall n$: $G_s^n(\Sigma^\omega) = (\tau^{\frown n})^\frown \Sigma^\omega$:

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

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

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

We mix them into a new complete lattice $(\mathcal{P}(\Sigma^\infty), \sqsubseteq, \sqcup, \sqcap, \bot, \top)$:

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

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:

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

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 $\mathcal{M}_\infty = \text{lfp } F_s$ in $(\mathcal{P}(\Sigma^\infty), \sqsubseteq)$ with $F_s(T) \stackrel{\text{def}}{=} \mathcal{B} \cup \tau \frown T$

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!

Abstracting maximal traces into partial traces

Finite and infinite partial trace semantics

Two steps to go from maximal to finite partial traces:

- add all partial traces
- remove infinite traces (in this order!)

Partial trace semantics $\mathcal{T}_\infty$

all finite and infinite sequences of states
linked by the transition relation τ:

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

(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 $\mathcal{T}_\infty$
(forget about infinite executions)

We have a Galois embedding:

$$(\mathcal{P}(\Sigma^\infty), \sqsubseteq) \xleftarrow[\alpha_*]{\gamma_*} (\mathcal{P}(\Sigma^*), \subseteq)$$

- $\sqsubseteq$ 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 $\mathcal{T}_\infty = \text{lfp}\, F_{s*}$ in $(\mathcal{P}(\Sigma^\infty), \sqsubseteq)$ and $\mathcal{T} = \text{lfp}\, F_{s*}$ 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:

$$(\mathcal{P}(\Sigma^\infty \setminus \{\epsilon\}), \subseteq) \xrightleftharpoons[\alpha_\preceq]{\gamma_\preceq} (\mathcal{P}(\Sigma^\infty \setminus \{\epsilon\}), \subseteq)$$

- $\alpha_\preceq(T) \stackrel{\text{def}}{=} \{\, t \in \Sigma^\infty \setminus \{\epsilon\} \mid \exists u \in T : t \preceq u \,\}$

  (set of all non-empty prefixes of traces in T)

- $\gamma_\preceq(T) \stackrel{\text{def}}{=} \{\, t \in \Sigma^\infty \setminus \{\epsilon\} \mid \forall u \in \Sigma^\infty \setminus \{\epsilon\} : u \preceq t \implies u \in T \,\}$

  (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 $\mathcal{M}_\infty$: $\mathcal{T}_\infty = \alpha_{\preceq}(\mathcal{M}_\infty)$.

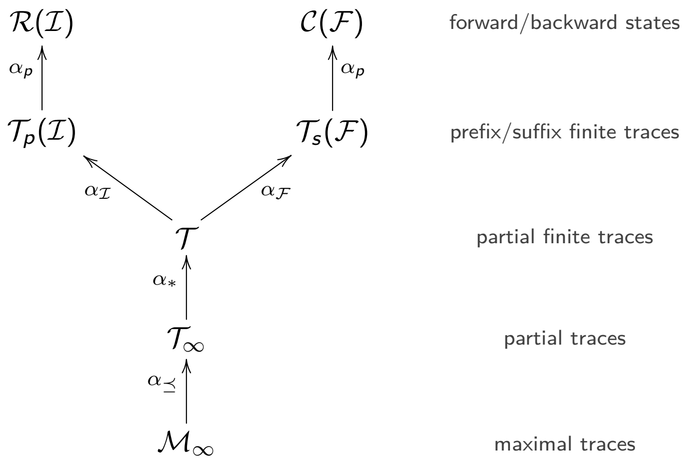
proof:

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.

Enriched hierarchy of semantics



See [Cous02] for more semantics in this diagram.

Trace properties

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)

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

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

The associated upper closure $\rho_{*\preceq} \stackrel{\text{def}}{=} \gamma_{\preceq} \circ \alpha_{\preceq}$ is:
$\rho_{*\preceq} = \lim \circ \rho_p$ where:

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

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 $S \subseteq \Sigma$:

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

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

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

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

  $\rho_p((\Sigma^2)^\infty) = \Sigma^\infty \neq (\Sigma^2)^\infty \implies$ 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:

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

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 $\lim(\Sigma^*) = \Sigma^\infty \implies$ liveness;

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

  $\rho_p(P) = P \cup \Sigma^*$ and $\lim(P \cup \Sigma^*) = \Sigma^\infty \implies$ 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 $S \neq \Sigma \implies$ not liveness;

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

  $\rho_p(\Sigma^{\leq k}) = \lim(\Sigma^{\leq k}) = \Sigma^{\leq k} \neq \Sigma^\infty \implies$ 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: $\rho(x \cup (X \setminus \rho(x))) \supseteq \rho(x) \cup \rho(X \setminus \rho(x))$
  $\supseteq \rho(x) \cup (X \setminus \rho(x))$
  $= X$

Consequence: on trace properties

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

proving a trace property can be decomposed into
a soundness proof and a liveness proof

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

(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.

[Cous92] **P. Cousot & R. Cousot**. *Abstract interpretation and application to logic programs.* In Journal of Logic Programming, 13(2–3):103–179, 1992..

[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.