

Program Semantics and Properties

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

Antoine Miné

Year 2024–2025

Course 2
30 September 2024



Programs and executions

Language syntax

ℓ stat ℓ	$::=$	$\ell X \leftarrow \text{exp}^\ell$	(assignment)
		$\ell \text{if exp} \bowtie 0 \text{ then } \ell \text{stat}^\ell$	(conditional)
		$\ell \text{while } \ell \text{exp} \bowtie 0 \text{ do } \ell \text{stat}^\ell \text{ done}^\ell$	(loop)
		$\ell \text{stat}; \ell \text{stat}^\ell$	(sequence)
exp	$::=$	X	(variable)
		$-\text{exp}$	(negation)
		$\text{exp} \diamond \text{exp}$	(binary operation)
		c	(constant $c \in \mathbb{Z}$)
		$[c, c']$	(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: $\bowtie \in \{=, \leq, \dots\}$, $\diamond \in \{+, -, \times, /\}$
- **random inputs**: $X \leftarrow [c, c']$
model environment, parametric programs, unknown functions, ...

Example

Example

```
a $X \leftarrow [-\infty, \infty];$   

bwhile c $X \neq 0$  do d $X \leftarrow X - 1$  done e
```

Where:

- control points $\mathcal{L} = \{a, b, c, d, e\}$
- variables $\mathbb{V} = \{X\}$

We also define:

- the entry control point: $a \in \mathcal{L}$
- the exit control point: $e \in \mathcal{L}$
- the **memory states**: $\mathcal{E} \stackrel{\text{def}}{=} \mathbb{V} \rightarrow \mathbb{Z}$
- the **program states**: $\Sigma \stackrel{\text{def}}{=} \mathcal{L} \times \mathcal{E}$ (control and memory state)

Transition systems

Program execution modeled as discrete **transitions** between **states**

- Σ : set of **states**
- $\tau \subseteq \Sigma \times \Sigma$: a **transition relation**, written $\sigma \rightarrow_{\tau} \sigma'$, or $\sigma \rightarrow \sigma'$

\implies a form of small-step semantics.

and also sometimes:

- distinguished set of **initial states** $\mathcal{I} \subseteq \Sigma$
- distinguished set of **final states** $\mathcal{F} \subseteq \Sigma$
- *labelled* transition systems: $\tau \subseteq \Sigma \times \mathcal{A} \times \Sigma$, $\sigma \xrightarrow{a} \sigma'$
where \mathcal{A} is a set of labels, or actions

Transition system on our language

Application: on our programming language

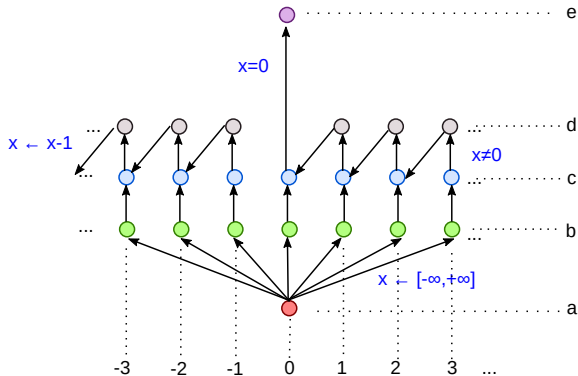
- $\Sigma \stackrel{\text{def}}{=} \mathcal{L} \times \mathcal{E}$ (a program state = a control point and a memory state)
where $\mathcal{E} \stackrel{\text{def}}{=} \mathbb{V} \rightarrow \mathbb{Z}$
- initial states $\mathcal{I} \stackrel{\text{def}}{=} \{\ell\} \times \mathcal{E}$ and
final states $\mathcal{F} \stackrel{\text{def}}{=} \{\ell'\} \times \mathcal{E}$ for program ${}^{\ell}\text{stat}^{\ell'}$
- τ is defined by structural induction on ${}^{\ell}\text{stat}^{\ell'}$ (next slides)
- τ is non-deterministic
(several possible successors for $X \leftarrow [a, b]$)

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

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

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

$$\begin{aligned} \tau[\ell^1 \text{while } \ell^2 e \bowtie 0 \text{ do } \ell^3 \text{ s } \ell^4 \text{ done } \ell^5] &\stackrel{\text{def}}{=} \\ &\{ (\ell^1, \rho) \rightarrow (\ell^2, \rho) \mid \rho \in \mathcal{E} \} \cup \\ &\{ (\ell^2, \rho) \rightarrow (\ell^3, \rho) \mid \rho \in \mathcal{E}, \exists v \in E[e] \rho: v \bowtie 0 \} \cup \tau[\ell^3 \text{ s } \ell^4] \cup \\ &\{ (\ell^4, \rho) \rightarrow (\ell^2, \rho) \mid \rho \in \mathcal{E} \} \cup \\ &\{ (\ell^2, \rho) \rightarrow (\ell^5, \rho) \mid \rho \in \mathcal{E}, \exists v \in E[e] \rho: v \not\bowtie 0 \} \end{aligned}$$

$$\tau[\ell^1 \text{ s}_1; \ell^2 \text{ s}_2 \ell^3] \stackrel{\text{def}}{=} \tau[\ell^1 \text{ s}_1 \ell^2] \cup \tau[\ell^2 \text{ s}_2 \ell^3]$$

(expression semantics $E[e]$ on next slide)

Expression semantics

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

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

$$E[[c, c']] \rho \stackrel{\text{def}}{=} \{x \in \mathbb{Z} \mid c \leq x \leq c'\}$$

$$E[[X]] \rho \stackrel{\text{def}}{=} \{\rho(X)\}$$

$$E[[-e]] \rho \stackrel{\text{def}}{=} \{-v \mid v \in E[[e]] \rho\}$$

$$E[[e_1 + e_2]] \rho \stackrel{\text{def}}{=} \{v_1 + v_2 \mid v_1 \in E[[e_1]] \rho, v_2 \in E[[e_2]] \rho\}$$

$$E[[e_1 - e_2]] \rho \stackrel{\text{def}}{=} \{v_1 - v_2 \mid v_1 \in E[[e_1]] \rho, v_2 \in E[[e_2]] \rho\}$$

$$E[[e_1 \times e_2]] \rho \stackrel{\text{def}}{=} \{v_1 \times v_2 \mid v_1 \in E[[e_1]] \rho, v_2 \in E[[e_2]] \rho\}$$

$$E[[e_1 / e_2]] \rho \stackrel{\text{def}}{=} \{v_1 / v_2 \mid v_1 \in E[[e_1]] \rho, v_2 \in E[[e_2]] \rho, v_2 \neq 0\}$$

Another example: λ -calculus

syntax: λ -terms

t	$::=$	x	(variable)
		$\lambda x.t$	(abstraction)
		$t u$	(application)

Small-step operational semantics: (call-by-value)

$$\frac{}{(\lambda x.M)N \rightsquigarrow M[x/N]} \quad \frac{M \rightsquigarrow M'}{M N \rightsquigarrow M' N} \quad \frac{N \rightsquigarrow N'}{M N \rightsquigarrow M N'}$$

Models program execution as a sequence of term-rewriting \rightsquigarrow
 exposing each transition (low level).

■ $\Sigma \stackrel{\text{def}}{=} \{\lambda\text{-terms}\}$

■ $\mathcal{T} \stackrel{\text{def}}{=} \rightsquigarrow$

Program executions

Intuitive model of executions:

- program **traces**
sequences of states encountered during execution
sequences are possibly unbounded
- a program can have several traces
due to non-determinism

Trace semantics:

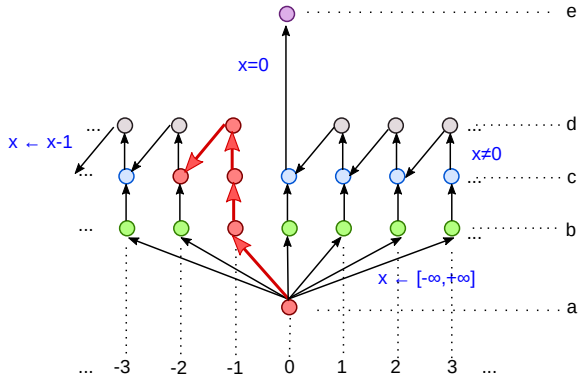
- the domain is $\mathcal{D} \stackrel{\text{def}}{=} \mathcal{P}(\Sigma^*)$
- the semantics is:
 $\mathcal{T}_p(\mathcal{I}) \stackrel{\text{def}}{=} \{ \sigma_0, \dots, \sigma_n \mid n \geq 0, \sigma_0 \in \mathcal{I}, \forall i: \sigma_i \rightarrow \sigma_{i+1} \}$
- actually, we defined here finite execution prefixes, observable in finite time

Trace semantics example

Example

^a $X \leftarrow [-\infty, \infty];$

^b**while** ^c $X \neq 0$ **do** ^d $X \leftarrow X - 1$ **done** ^e



Semantics and abstract interpretation

Other choices of semantics are possible:

- reachable states (later in this course)
- going backward as well as forward (later in this course)
- relations between input and output (relational, or denotational semantics)
- ...

these are all **uncomputable** **concrete** semantics
(next course will consider computable approximations)

Goal: use abstract interpretation to

- express all these semantics uniformly as **fixpoints**
(staying at the level of transition systems for generality, not program syntax)
- relate these semantics by **abstraction relations**
- study which semantics to **choose** for each class of properties to **prove**

Finite prefix trace semantics

Finite traces

Finite trace: finite sequence of elements from Σ

- ϵ : empty trace (unique)
- σ : trace of length 1 (assimilated to a state)
- $\sigma_0, \dots, \sigma_{n-1}$: trace of length n
- Σ^n : the set of traces of length n
- $\Sigma^{\leq n} \stackrel{\text{def}}{=} \bigcup_{i \leq n} \Sigma^i$: the set of traces of length at most n
- $\Sigma^* \stackrel{\text{def}}{=} \bigcup_{i \in \mathbb{N}} \Sigma^i$: the set of finite traces

Note: we assimilate

- a set of states $S \subseteq \Sigma$ with a set of traces of length 1
- a relation $R \subseteq \Sigma \times \Sigma$ with a set of traces of length 2

so, $\mathcal{I}, \mathcal{F}, \tau \in \mathcal{P}(\Sigma^*)$

Trace operations

Operations on traces:

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

- **concatenation** \cdot

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

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

- **junction** \frown

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

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**:

$$\blacksquare A \cdot B \stackrel{\text{def}}{=} \{a \cdot b \mid a \in A, b \in B\}$$

$\{\epsilon\}$ is the neutral element for \cdot

$$\blacksquare A \frown B \stackrel{\text{def}}{=} \{a \frown b \mid a \in A, b \in B, a \frown b \text{ defined}\}$$

Σ is the neutral element for \frown

$$A^0 \stackrel{\text{def}}{=} \{\epsilon\}$$

$$A^{n+1} \stackrel{\text{def}}{=} A \cdot A^n$$

$$A^* \stackrel{\text{def}}{=} \bigcup_{n < \omega} A^n$$

$$A^{\frown 0} \stackrel{\text{def}}{=} \Sigma$$

$$A^{\frown n+1} \stackrel{\text{def}}{=} A \frown A^{\frown n}$$

$$A^{\frown *} \stackrel{\text{def}}{=} \bigcup_{n < \omega} A^{\frown n}$$

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

Notes: distributivity

$$\blacksquare \cdot \text{ and } \frown \text{ distribute } \cup : \text{e.g., } (\bigcup_{i \in I} A_i) \frown (\bigcup_{j \in J} B_j) = \bigcup_{i \in I, j \in J} (A_i \frown B_j)$$

$$\blacksquare \cdot \text{ and } \frown \text{ do not distribute } \cap : \text{e.g., } (\bigcap_{i \in I} A_i) \frown (\bigcap_{j \in J} B_j) \subseteq \bigcap_{i \in I, j \in J} (A_i \frown B_j) \text{ by monotony of } \frown, \\ \text{but the equality does not always hold}$$

Prefix trace semantics

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

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

(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**:

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

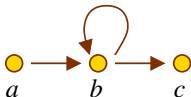
(F_p appends a transition to each trace, and adds back \mathcal{I})

Alternate characterization: $\mathcal{T}_p(\mathcal{I}) = \text{lfp}_{\mathcal{I}} G_p$ where $G_p(T) = T \cup T \frown \tau$.

G_p extends T by τ and accumulates the result with T

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

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

Prefix trace semantics: proof

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

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

$$\begin{aligned} & F_p(\bigcup_{i \in I} T_i) \\ &= \mathcal{I} \cup (\bigcup_{i \in I} T_i) \frown \tau \\ &= \mathcal{I} \cup (\bigcup_{i \in I} T_i \frown \tau) = \bigcup_{i \in I} (\mathcal{I} \cup T_i \frown \tau) \end{aligned}$$

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

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

$$\begin{aligned} \blacksquare & F_p^0(\emptyset) = \emptyset, \\ \blacksquare & \begin{aligned} & F_p^{n+1}(\emptyset) \\ &= \mathcal{I} \cup F_p^n(\emptyset) \frown \tau \\ &= \mathcal{I} \cup (\bigcup_{i < n} \mathcal{I} \frown \tau \frown^i) \frown \tau \\ &= \mathcal{I} \cup \bigcup_{i < n} (\mathcal{I} \frown \tau \frown^i) \frown \tau \\ &= \mathcal{I} \frown \tau \frown^0 \cup \bigcup_{i < n} (\mathcal{I} \frown \tau \frown^{i+1}) \\ &= \bigcup_{i < n+1} \mathcal{I} \frown \tau \frown^i \end{aligned} \end{aligned}$$

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

The proof is similar for the alternate form $\mathcal{T}_p(\mathcal{I}) = \text{lfp}_{\mathcal{I}} G_p$ where $G_p(T) = T \cup T \frown \tau$ as $G_p^n(\mathcal{I}) = F_p^{n+1}(\emptyset) = \bigcup_{i \leq n} \mathcal{I} \frown \tau \frown^i$.

Prefix closure

Prefix partial order: \preceq on Σ^*

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

Note: (Σ^*, \preceq) is not a CPO, as $a^n, n \in \mathbb{N}$ has no limit

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

$$\rho_p(T) \stackrel{\text{def}}{=} \{ u \in \Sigma^+ \mid \exists t \in T : u \preceq t \}$$

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)

Collecting semantics and properties

General properties

General setting:

- given a program $prog \in Prog$
- its **semantics**: $\llbracket \cdot \rrbracket : Prog \rightarrow \mathcal{P}(\Sigma^*)$ is a set of finite traces
- a **property** P is the **set** of correct program semantics
i.e., a **set of sets of traces** $P \in \mathcal{P}(\mathcal{P}(\Sigma^*))$
 - \subseteq gives an information order on properties
 - $P \subseteq P'$ means that P' is weaker than P (allows more semantics)

General collecting semantics

The **collecting semantics** $Col : Prog \rightarrow \mathcal{P}(\mathcal{P}(\Sigma^*))$
is the **strongest property** of a program

Hence: $Col(prog) \stackrel{\text{def}}{=} \{\llbracket prog \rrbracket\}$

Benefits: uniformity of semantics and properties, \subseteq information order

- given a program $prog$ and a property $P \in \mathcal{P}(\mathcal{P}(\Sigma^*))$
the **verification problem** is an inclusion check:

$$Col(prog) \subseteq P$$

- 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: $S^\sharp \subseteq P$

Restricted properties

Reasoning on (and abstracting) $\mathcal{P}(\mathcal{P}(\Sigma^*))$ is **hard!**

In the following, we use a **simpler** setting:

- a property is a **set of traces** $P \in \mathcal{P}(\Sigma^*)$
- the collecting semantics is a **set of traces**: $Col(prog) \stackrel{\text{def}}{=} \llbracket prog \rrbracket$
- the verification problem remains an inclusion check: $\llbracket prog \rrbracket \subseteq P$
- abstractions will over-approximate the set of traces $\llbracket prog \rrbracket$

Example properties:

- state property $P \stackrel{\text{def}}{=} S^*$ (remains in the set S of safe states)
- maximal execution time: $P \stackrel{\text{def}}{=} S^{\leq k}$
- ordering: $P \stackrel{\text{def}}{=} (\Sigma \setminus \{b\})^* \cdot a \cdot \Sigma^* \cdot b \cdot \Sigma^*$ (a occurs before b)

Proving restricted 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)
- $\forall \sigma_0, \dots, \sigma_n \in I: \sigma_n \rightarrow \sigma_{n+1} \implies \sigma_0, \dots, \sigma_n, \sigma_{n+1} \in I$
(invariant by program transition)
- 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 $F_p(T) \stackrel{\text{def}}{=} \mathcal{I} \cup T \cap \tau$.

$\mathcal{T}_p(\mathcal{I}) = \text{lfp } F_p$ is the most precise inductive invariant

Limitations

- Our semantics is **closed by prefix**

It cannot distinguish between:

- non-terminating executions (infinite loops)
- and unbounded executions

⇒ we **cannot prove termination** and, more generally, **liveness**

(this will be solved using *maximal trace semantics* later in this course)

- Some properties, such as **non-interferences**, cannot be expressed as sets of traces, we need **sets of sets of traces**

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

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

Forward state reachability semantics

State semantics and properties

Principle: reason on **sets of states** instead of sets of traces

- simpler semantic $Col : Prog \rightarrow \mathcal{P}(\Sigma)$
- state properties are also **sets of states** $P \in \mathcal{P}(\Sigma)$
 \implies sufficient for many purposes
- easier to abstract
- can be seen as an abstraction of traces
(forgets the ordering of states)

Forward reachability

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

$$\text{post}_\tau(S) \stackrel{\text{def}}{=} \{ \sigma' \mid \exists \sigma \in S : \sigma \rightarrow \sigma' \}$$

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: $\mathcal{B} \stackrel{\text{def}}{=} \{ \sigma \mid \forall \sigma' \in \Sigma : \sigma \not\rightarrow \sigma' \}$

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

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

$$\begin{aligned} \mathcal{R}(\mathcal{I}) &\stackrel{\text{def}}{=} \{ \sigma \mid \exists n \geq 0, \sigma_0, \dots, \sigma_n : \sigma_0 \in \mathcal{I}, \sigma = \sigma_n, \forall i : \sigma_i \rightarrow \sigma_{i+1} \} \\ &= \bigcup_{n \geq 0} \text{post}_\tau^n(\mathcal{I}) \end{aligned}$$

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

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

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

- $F_{\mathcal{R}}^0(\emptyset) = \emptyset$

- assuming the property at n ,

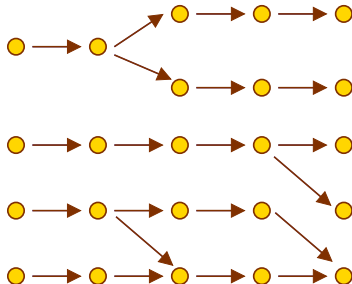
$$\begin{aligned}
 F_{\mathcal{R}}^{n+1}(\emptyset) &= F_{\mathcal{R}}\left(\bigcup_{i < n} \text{post}_{\tau}^i(\mathcal{I})\right) \\
 &= \mathcal{I} \cup \text{post}_{\tau}\left(\bigcup_{i < n} \text{post}_{\tau}^i(\mathcal{I})\right) \\
 &= \mathcal{I} \cup \bigcup_{i < n} \text{post}_{\tau}(\text{post}_{\tau}^i(\mathcal{I}))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

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

The proof is similar for the alternate form, given that $\text{lfp}_{\mathcal{I}} G_{\mathcal{R}} = \bigcup_{n \in \mathbb{N}} G_{\mathcal{R}}^n(\mathcal{I})$ and

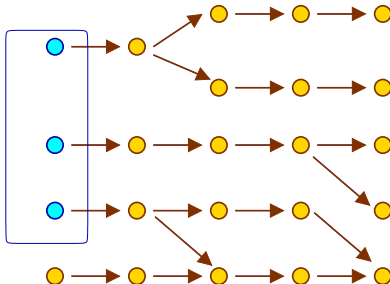
$$G_{\mathcal{R}}^n(\mathcal{I}) = F_{\mathcal{R}}^{n+1}(\emptyset) = \bigcup_{i \leq n} \text{post}_{\tau}^i(\mathcal{I}).$$

Graphical illustration



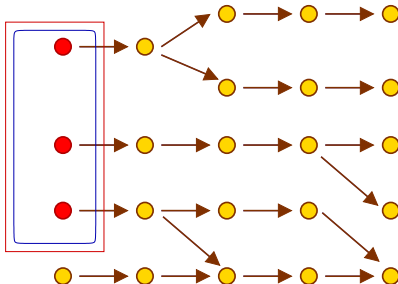
Transition system

Graphical illustration



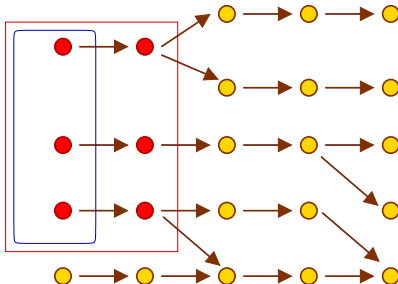
Initial states \mathcal{I}

Graphical illustration



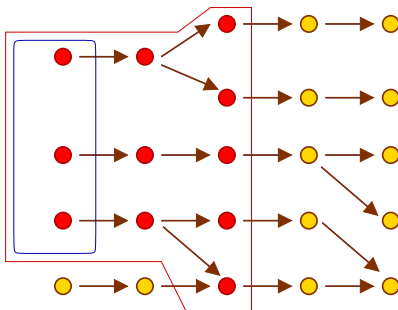
Iterate $F_{\mathcal{R}}^1(\mathcal{I})$

Graphical illustration



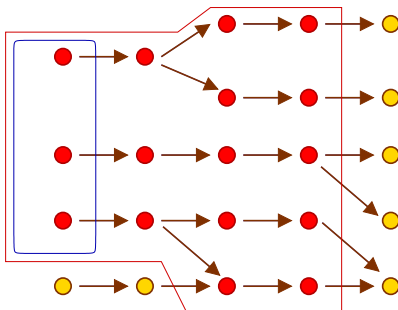
Iterate $F_{\mathcal{R}}^2(\mathcal{I})$

Graphical illustration



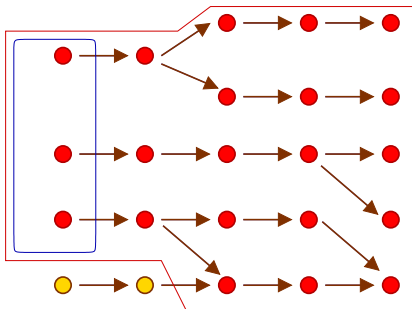
Iterate $F_{\mathcal{R}}^3(\mathcal{I})$

Graphical illustration



Iterate $F_{\mathcal{R}}^4(\mathcal{I})$

Graphical illustration



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

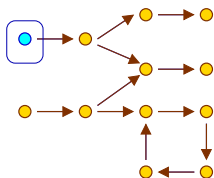
$F_{\mathcal{R}}^6(\mathcal{I}) = F_{\mathcal{R}}^5(\mathcal{I}) \Rightarrow$ we reached a fixpoint $\mathcal{R}(\mathcal{I}) = F_{\mathcal{R}}^5(\mathcal{I})$

Multiple forward fixpoints

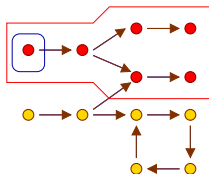
Recall: $\mathcal{R}(\mathcal{I}) = \text{lfp } F_{\mathcal{R}}$ where $F_{\mathcal{R}}(S) \stackrel{\text{def}}{=} \mathcal{I} \cup \text{post}_{\tau}(S)$

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

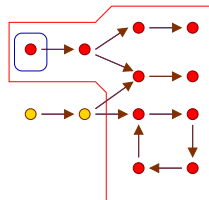
Example:



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 state-based invariance proof methods

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

- $\mathcal{I} \subseteq I$ (contains initial states)
- $\forall \sigma \in I: \sigma \rightarrow \sigma' \implies \sigma' \in I$ (invariant by program transition)
- that implies the desired property: $I \subseteq P$

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

- 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

Grouping by control location: $\mathcal{P}(\Sigma) = \mathcal{P}(\mathcal{L} \times \mathcal{E}) \simeq \mathcal{L} \rightarrow \mathcal{P}(\mathcal{E})$

We have a **Galois isomorphism**:

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

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

Example equation system

```

 $\ell^1$   $X \leftarrow [0, 10];$   $\ell^2$ 
 $Y \leftarrow 100;$ 
while  $\ell^3$   $X \geq 0$  do  $\ell^4$ 
     $X \leftarrow X - 1;$   $\ell^5$ 
     $Y \leftarrow Y + 10$ 
done  $\ell^6$ 

```

$$\left\{ \begin{array}{l} \mathcal{X}_1 = \mathcal{E} \\ \mathcal{X}_2 = \mathbb{C} \llbracket X \leftarrow [0, 10] \rrbracket \mathcal{X}_1 \\ \mathcal{X}_3 = \mathbb{C} \llbracket Y \leftarrow 100 \rrbracket \mathcal{X}_2 \cup \mathbb{C} \llbracket Y \leftarrow Y + 10 \rrbracket \mathcal{X}_5 \\ \mathcal{X}_4 = \mathbb{C} \llbracket X \geq 0 \rrbracket \mathcal{X}_3 \\ \mathcal{X}_5 = \mathbb{C} \llbracket X \leftarrow X - 1 \rrbracket \mathcal{X}_4 \\ \mathcal{X}_6 = \mathbb{C} \llbracket X < 0 \rrbracket \mathcal{X}_3 \end{array} \right.$$

(atomic command semantics $\mathbb{C} \llbracket \text{com} \rrbracket$ on next slide)

- $\mathcal{X}_i \in \mathcal{P}(\mathcal{E})$: set of memory states at program point $i \in \mathcal{L}$
e.g.: $\mathcal{X}_3 = \{ \rho \in \mathcal{E} \mid \rho(X) \in [0, 10], 10\rho(X) + \rho(Y) \in [100, 200] \cap 10\mathbb{Z} \}$
- \mathcal{R} corresponds to the **smallest** solution $(\mathcal{X}_i)_{i \in \mathcal{L}}$ of the system
- $I \subseteq \mathcal{E}$ is **invariant at i** if $\mathcal{X}_i \subseteq I$

Systematic derivation of equations

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

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

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

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

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

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

by structural induction:

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

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

$$eq(\ell^1 \text{if } e \bowtie 0 \text{ then } \ell^2 s^{\ell^3}) \stackrel{\text{def}}{=} \{ \mathcal{X}_{\ell^2} = C[\![e \bowtie 0]\!] \mathcal{X}_{\ell^1} \} \cup eq(\ell^2 s^{\ell^3'}) \cup \{ \mathcal{X}_{\ell^3} = \mathcal{X}_{\ell^3'} \cup C[\![e \bowtie 0]\!] \mathcal{X}_{\ell^1} \}$$

$$eq(\ell^1 \text{while } \ell^2 e \bowtie 0 \text{ do } \ell^3 s^{\ell^4} \text{ done } \ell^5) \stackrel{\text{def}}{=} \{ \mathcal{X}_{\ell^2} = \mathcal{X}_{\ell^1} \cup \mathcal{X}_{\ell^4}, \mathcal{X}_{\ell^3} = C[\![e \bowtie 0]\!] \mathcal{X}_{\ell^2} \} \cup eq(\ell^3 s^{\ell^4}) \cup \{ \mathcal{X}_{\ell^5} = C[\![e \bowtie 0]\!] \mathcal{X}_{\ell^2} \}$$

where: $\mathcal{X}^{\ell^3'}$ is a fresh variable storing intermediate results

Solving the equational semantics

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

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

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

By Kleene's fixpoint theorem, $\text{lfp } \vec{F}$ exists

Kleene's theorem: Jacobi iterations

$$\left\{ \begin{array}{l} \mathcal{X}_1^0 \stackrel{\text{def}}{=} \emptyset \\ \dots \\ \mathcal{X}_i^0 \stackrel{\text{def}}{=} \emptyset \\ \dots \\ \mathcal{X}_n^0 \stackrel{\text{def}}{=} \emptyset \end{array} \right. \quad \left\{ \begin{array}{l} \mathcal{X}_1^{k+1} \stackrel{\text{def}}{=} F_1(\mathcal{X}_1^k, \dots, \mathcal{X}_n^k) \\ \dots \\ \mathcal{X}_i^{k+1} \stackrel{\text{def}}{=} F_i(\mathcal{X}_1^k, \dots, \mathcal{X}_n^k) \\ \dots \\ \mathcal{X}_n^{k+1} \stackrel{\text{def}}{=} F_n(\mathcal{X}_1^k, \dots, \mathcal{X}_n^k) \end{array} \right.$$

The limit of $(\mathcal{X}_1^k, \dots, \mathcal{X}_n^k)$ is $\text{lfp } \vec{F}$

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

Solving the equational semantics (cont.)

Other iteration techniques exist [Cous92].

Gauss-Seidl iterations

$$\left\{ \begin{array}{l} \mathcal{X}_1^{k+1} \stackrel{\text{def}}{=} F_1(\mathcal{X}_1^k, \dots, \mathcal{X}_n^k) \\ \dots \\ \mathcal{X}_i^{k+1} \stackrel{\text{def}}{=} F_i(\mathcal{X}_1^{k+1}, \dots, \mathcal{X}_{i-1}^{k+1}, \mathcal{X}_i^k, \dots, \mathcal{X}_n^k) \\ \dots \\ \mathcal{X}_n^{k+1} \stackrel{\text{def}}{=} F_n(\mathcal{X}_1^{k+1}, \dots, \mathcal{X}_{n-1}^{k+1}, \mathcal{X}_n^k) \end{array} \right.$$

use new results **as soon as available**

Chaotic iterations

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

w.r.t. a **fair schedule** $\phi : \mathbb{N} \rightarrow [1, n]$

$\forall i \in [1, n]: \forall N > 0: \exists k > N: \phi(k) = i$

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

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

Alternate view: inductive abstract interpreter

Principle:

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

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

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

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

$$C[\text{if } e \bowtie 0 \text{ then } s] \mathcal{X} \stackrel{\text{def}}{=} (C[s] (C[e \bowtie 0] \mathcal{X})) \cup (C[e \nbowtie 0] \mathcal{X})$$

$$C[\text{while } e \bowtie 0 \text{ do } s \text{ done}] \mathcal{X} \stackrel{\text{def}}{=} C[e \nbowtie 0] (\text{lfp } F)$$

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

informal justification for the loop semantics:

All the $C[s]$ functions are continuous, hence the fixpoints exist.

By induction on k , $F^k(\emptyset) = \bigcup_{i \leq k} (C[s] \circ C[e \bowtie 0])^i \mathcal{X}$

hence, $\text{lfp } F = \bigcup_i (C[s] \circ C[e \bowtie 0])^i \mathcal{X}$

We fall back to a special case of (transfinite) chaotic iteration that stabilizes loops depth-first.

From finite traces to reachability

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:

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

$$\blacksquare \alpha_p(T) \stackrel{\text{def}}{=} \{ \sigma \in \Sigma \mid \exists \sigma_0, \dots, \sigma_n \in T : \sigma = \sigma_n \}$$

(last state in traces in T)

$$\blacksquare \gamma_p(S) \stackrel{\text{def}}{=} \{ \sigma_0, \dots, \sigma_n \in \Sigma^* \mid \sigma_n \in S \}$$

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

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

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

Abstracting traces into states (example)

program

```

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

```

- **prefix trace** semantics:

i and j are **increasing** and $0 \leq j \leq i \leq 100$

- **forward reachable state** semantics:

$0 \leq j \leq i \leq 100$

\implies the abstraction **forgets the ordering of states**

Another state/trace abstraction: ordering abstraction

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

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

$$\blacksquare \alpha_o(T) \stackrel{\text{def}}{=} \{ \sigma \mid \exists \sigma_0, \dots, \sigma_n \in T, i \leq n : \sigma = \sigma_i \}$$

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

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

(traces composed of elements from S)

proof sketch:

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

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

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

$\alpha_o(\text{lfp } F_p) = \text{lfp } F_{\mathcal{R}}$.

Backward state co-reachability semantics

Backward state co-reachability

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

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

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

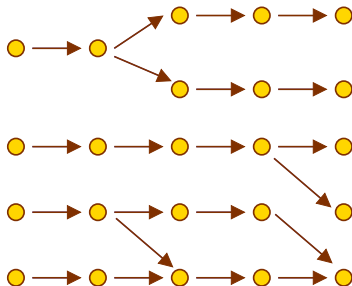
$\mathcal{C}(\mathcal{F})$ can also be expressed in **fixpoint** form:

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

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

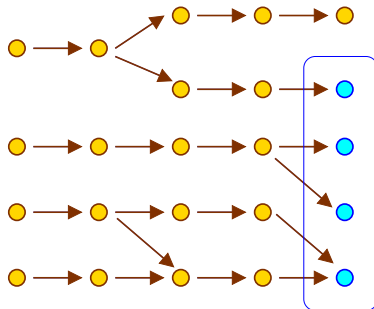
Alternate characterization: $\mathcal{C}(\mathcal{F}) = \text{lfp}_{\mathcal{F}} G_{\mathcal{C}} \text{ where } G_{\mathcal{C}}(S) = S \cup \text{pre}_{\tau}(S)$

Graphical illustration



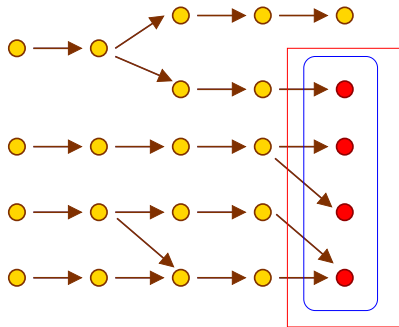
Transition system

Graphical illustration

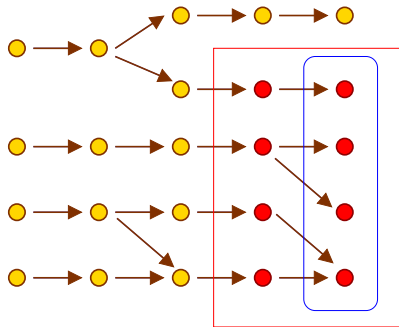


Final states \mathcal{F}

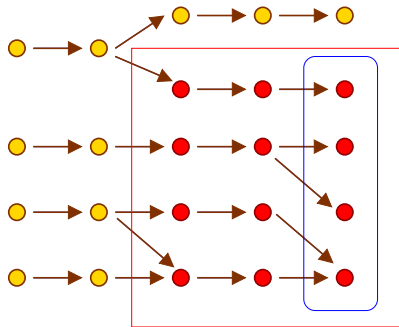
Graphical illustration



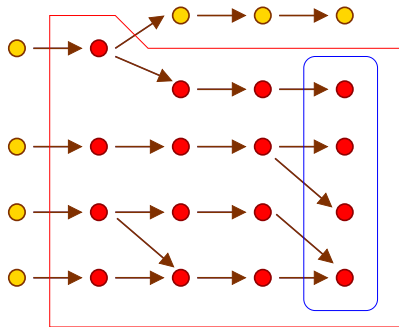
Graphical illustration



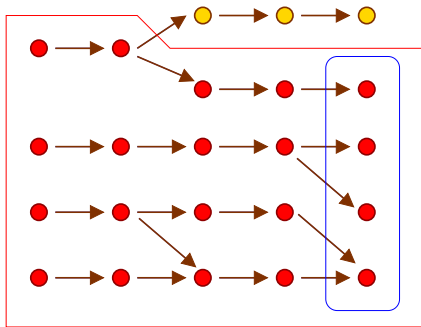
Graphical illustration



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

```

•  $j \leftarrow 0$ ;
  while  $i > 0$  do
     $i \leftarrow i - 1$ ;
     $j \leftarrow j + [0, 10]$ 
    assert ( $j \leq 200$ )
  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:

```

 $\ell 1$   $j \leftarrow 0$ ;
 $\ell 2$  while  $\ell 3$   $i > 0$  do
     $\ell 4$   $i \leftarrow i - 1$ ;
     $\ell 5$   $j \leftarrow j + [0, 10]$ 
 $\ell 6$ 

```

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

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

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

Suffix trace semantics

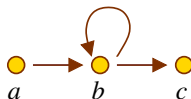
Similarly to the finite prefix trace semantics from \mathcal{I} , we can build a **suffix** trace semantics **going backwards** from \mathcal{F} :

- $\mathcal{T}_s(\mathcal{F}) \stackrel{\text{def}}{=} \{ \sigma_0, \dots, \sigma_n \mid n \geq 0, \sigma_n \in \mathcal{F}, \forall i: \sigma_i \rightarrow \sigma_{i+1} \}$
(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-reachability** abstracts the suffix trace semantics:

- $\alpha_s(\mathcal{T}_s(\mathcal{F})) = \mathcal{C}(\mathcal{F})$ where $\alpha_s(T) \stackrel{\text{def}}{=} \{ \sigma \mid \exists \sigma_0, \dots, \sigma_n \in T: \sigma = \sigma_0 \}$
- $\rho_s(\mathcal{T}_s(\mathcal{F})) = \mathcal{T}_s(\mathcal{F})$ 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 $F_s(T) \stackrel{\text{def}}{=} \mathcal{F} \cup \tau \cap T$

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

Symmetric finite partial trace semantics

Symmetric finite partial trace semantics

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

(not necessarily starting in \mathcal{I} nor ending in \mathcal{F})

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

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:

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

■ $\alpha_{\mathcal{I}}(T) \stackrel{\text{def}}{=} T \cap (\mathcal{I} \cdot \Sigma^*)$ (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 $\alpha_{\mathcal{F}}(T) \stackrel{\text{def}}{=} T \cap (\Sigma^* \cdot \mathcal{F})$

(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.

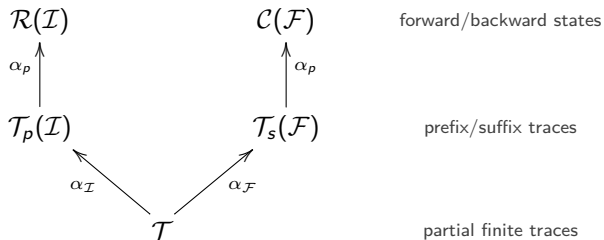
$\mathcal{T} = \text{lfp } F_{p*}$ where $F_{p*}(T) \stackrel{\text{def}}{=} \Sigma \cup T \frown \tau$.

$\mathcal{T}_p = \text{lfp } F_p$ 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



Sufficient precondition state semantics

Sufficient preconditions

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

$$\begin{aligned}\mathcal{S}(\mathcal{Y}) &\stackrel{\text{def}}{=} \{ \sigma \mid \forall n \geq 0, \sigma_0, \dots, \sigma_n : (\sigma = \sigma_0 \wedge \forall i : \sigma_i \rightarrow \sigma_{i+1}) \implies \sigma_n \in \mathcal{Y} \} \\ &= \bigcap_{n \geq 0} \widetilde{\text{pre}}_\tau^n(\mathcal{Y})\end{aligned}$$

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

(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**:

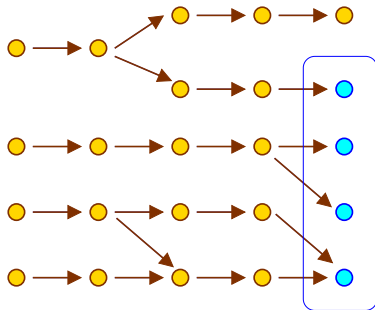
$$\mathcal{S}(\mathcal{Y}) = \text{gfp } F_S \text{ where } F_S(S) \stackrel{\text{def}}{=} \mathcal{Y} \cap \widetilde{\text{pre}}_\tau(S)$$

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

F_S is continuous in the dual CPO $(\mathcal{P}(\Sigma), \supseteq)$, because $\widetilde{\text{pre}}_\tau$ is: $F_S(\cap_{i \in I} A_i) = \cap_{i \in I} F_S(A_i)$.
By Kleene's theorem in the dual, $\text{gfp } F_S = \cap_{n \in \mathbb{N}} F_S^n(\Sigma)$.

We would prove by recurrence that $F_S^n(\Sigma) = \cap_{i < n} \widetilde{\text{pre}}_\tau^i(\mathcal{Y})$.

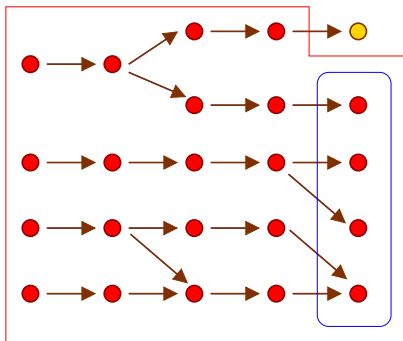
Graphical illustration



Final states \mathcal{F}

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

Graphical illustration

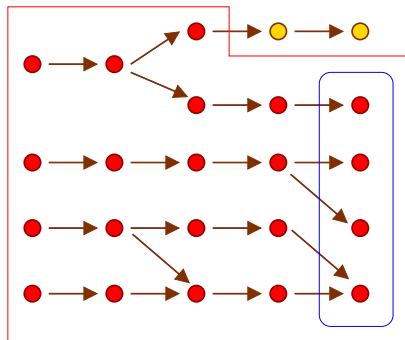


Final states \mathcal{F}

Goal: stay in $\mathcal{Y} = \mathcal{F} \cup (\Sigma \setminus \mathcal{B})$

Iteration $F_S^0(\mathcal{Y})$

Graphical illustration

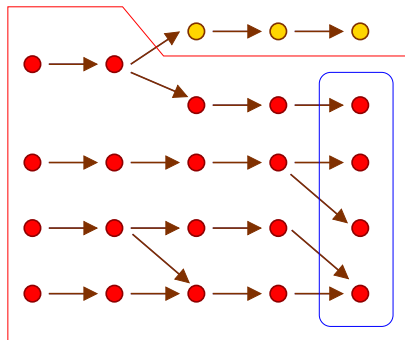


Final states \mathcal{F}

Goal: stay in $\mathcal{Y} = \mathcal{F} \cup (\Sigma \setminus \mathcal{B})$

Iteration $F_S^1(\mathcal{Y})$

Graphical illustration

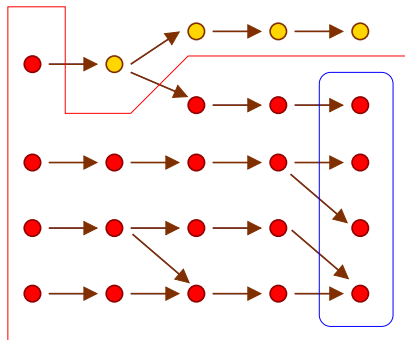


Final states \mathcal{F}

Goal: stay in $\mathcal{Y} = \mathcal{F} \cup (\Sigma \setminus \mathcal{B})$

Iteration $F_S^2(\mathcal{Y})$

Graphical illustration

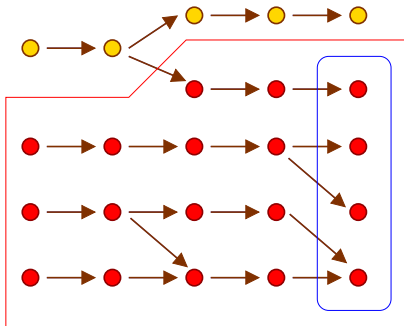


Final states \mathcal{F}

Goal: stay in $\mathcal{Y} = \mathcal{F} \cup (\Sigma \setminus \mathcal{B})$

Iteration $F_S^3(\mathcal{Y})$

Graphical illustration

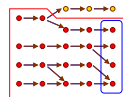
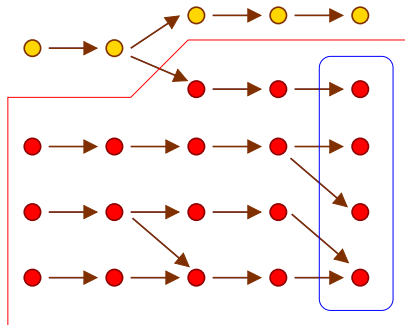


Final states \mathcal{F}

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}

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

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

Sufficient preconditions and reachability

Correspondence with reachability:

We have a **Galois connection**:

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

■ $\mathcal{R}(\mathcal{I}) \subseteq \mathcal{Y} \iff \mathcal{I} \subseteq \mathcal{S}(\mathcal{Y})$

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: $(\mathcal{P}(\Sigma), \subseteq) \xrightleftharpoons[\text{post}_{\tau}]{\widetilde{\text{pre}}_{\tau}} (\mathcal{P}(\Sigma), \subseteq)$.

$$\begin{aligned}
 \text{post}_{\tau}(A) \subseteq B & \iff \{ \sigma' \mid \exists \sigma \in A : \sigma \rightarrow \sigma' \} \subseteq B \\
 & \iff (\forall \sigma \in A : \sigma \rightarrow \sigma' \implies \sigma' \in B) \\
 & \iff (A \subseteq \{ \sigma \mid \forall \sigma' : \sigma \rightarrow \sigma' \implies \sigma' \in B \}) \\
 & \iff A \subseteq \widetilde{\text{pre}}_{\tau}(B)
 \end{aligned}$$

As a consequence $(\mathcal{P}(\Sigma), \subseteq) \xrightleftharpoons[G_{\mathcal{R}}]{G_{\mathcal{S}}} (\mathcal{P}(\Sigma), \subseteq)$.

The Galois connection can be lifted to fixpoint operators:

$$(\mathcal{P}(\Sigma), \subseteq) \xrightleftharpoons[x \mapsto \text{lfp}_x G_{\mathcal{R}}]{x \mapsto \text{gfp}_x G_{\mathcal{S}}} (\mathcal{P}(\Sigma), \subseteq).$$

Applications 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
     $i \leftarrow i + 1;$ 
     $j \leftarrow j + [0, 1]$ 
    assert  $(j \leq 105)$ 
  done •

```

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

- 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!

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

Solution: restrict the semantics to **complete** executions only

- keep only executions finishing in a **blocking state** B
- add **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!

Benefits:

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

Infinite traces

Notations:

- $\sigma_0, \dots, \sigma_n, \dots$: 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 (finite and infinite)

Extending the operators:

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

\cdot and \frown still distribute \cup , and still does not always distribute \cap

Counterexample:

$$\{a^\omega\} \frown (\cap_{n \in \mathbb{N}} \{a^m \mid m \geq n\}) = \{a^\omega\} \frown \emptyset = \emptyset \text{ but } \cap_{n \in \mathbb{N}} (\{a^\omega\} \frown \{a^m \mid m \geq n\}) = \cap_{n \in \mathbb{N}} \{a^\omega\} = \{a^\omega\}$$

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 ($\mathcal{I} = \Sigma$, technical requirement for the fixpoint characterization)
- either finite and **stop in a blocking state** ($\mathcal{F} = \mathcal{B}$)
- or **infinite**

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

(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}_∞

We consider separately **finite** and **infinite** maximal traces

- **Finite traces:** already done!

From the **suffix partial trace semantics**, recall:

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

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

- **Infinite traces:**

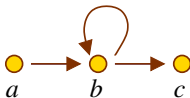
Additionally, we will prove: $\mathcal{M}_\infty \cap \Sigma^\omega = \text{gfp } G_s$

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 $G_s(T) \stackrel{\text{def}}{=} \tau \cap T$

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

Infinite trace semantics: proof

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

where $G_s(T) \stackrel{\text{def}}{=} \tau \cap T$ 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 $\forall n: G_s^n(\Sigma^\omega) = (\tau \cap^n) \cap \Sigma^\omega$:

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

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

Least fixpoint formulation of maximal traces

Idea: To get a **least fixpoint** formulation for whole \mathcal{M}_∞ ,
we 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 $(\mathcal{P}(\Sigma^\infty), \sqsubseteq, \sqcup, \sqcap, \perp, \top)$:

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

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

(proof on next slides)

Fixpoint fusion theorem

Theorem: fixpoint fusion

If $X_1 = \text{lfp } F_1$ in $(\mathcal{P}(\mathcal{D}_1), \sqsubseteq_1)$ and $X_2 = \text{lfp } F_2$ in $(\mathcal{P}(\mathcal{D}_2), \sqsubseteq_2)$

and $\mathcal{D}_1 \cap \mathcal{D}_2 = \emptyset$,

then $X_1 \cup X_2 = \text{lfp } F$ in $(\mathcal{P}(\mathcal{D}_1 \cup \mathcal{D}_2), \sqsubseteq)$ where:

$$\blacksquare F(X) \stackrel{\text{def}}{=} F_1(X \cap \mathcal{D}_1) \cup F_2(X \cap \mathcal{D}_2)$$

$$\blacksquare A \sqsubseteq B \stackrel{\text{def}}{\iff} (A \cap \mathcal{D}_1) \sqsubseteq_1 (B \cap \mathcal{D}_1) \wedge (A \cap \mathcal{D}_2) \sqsubseteq_2 (B \cap \mathcal{D}_2)$$

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 $\text{gfp } F = \text{gfp } F_1 \cup \text{gfp } F_2$.

Least fixpoint formulation of maximal traces (proof)

We are now ready to finish the proof that $\mathcal{M}_\infty = \text{lfp } F_s$ in $(\mathcal{P}(\Sigma^\infty), \sqsubseteq)$ with $F_s(T) \stackrel{\text{def}}{=} \mathcal{B} \cup \tau \cap T$

proof:

We have:

- $\mathcal{M}_\infty \cap \Sigma^* = \text{lfp } F_s$ in $(\mathcal{P}(\Sigma^*), \sqsubseteq)$,
- $\mathcal{M}_\infty \cap \Sigma^\omega = \text{lfp } G_s$ in $(\mathcal{P}(\Sigma^\omega), \supseteq)$ where $G_s(T) \stackrel{\text{def}}{=} \tau \cap T$,
- 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 $(\Sigma^\infty, \supseteq)$ also exists!

Abstracting maximal traces into partial traces

Finite and infinite partial trace semantics

Two steps to go from maximal traces to finite partial traces:

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

Partial trace semantics \mathcal{T}_∞

all finite and infinite sequences of states

linked by the transition relation τ :

$$\mathcal{T}_\infty \stackrel{\text{def}}{=} \left\{ \sigma_0, \dots, \sigma_n \in \Sigma^* \mid \forall i < n: \sigma_i \rightarrow \sigma_{i+1} \right\} \cup \left\{ \sigma_0, \dots, \sigma_n, \dots \in \Sigma^\omega \mid \forall i < \omega: \sigma_i \rightarrow \sigma_{i+1} \right\}$$

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

Fixpoint form similar to \mathcal{M}_∞ :

$$\mathcal{T}_\infty = \text{lfp } F_{s*} \text{ in } (\mathcal{P}(\Sigma^\infty), \sqsubseteq) \text{ where } F_{s*}(T) \stackrel{\text{def}}{=} \Sigma \cup \tau \cap T$$

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

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:

α_\preceq and γ_\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}_∞ are an **abstraction** of **maximal traces** \mathcal{M}_∞ : $\mathcal{T}_\infty = \alpha_{\preceq}(\mathcal{M}_\infty)$.

proof:

Firstly, \mathcal{T}_∞ and $\alpha_{\preceq}(\mathcal{M}_\infty)$ coincide on infinite traces.

Indeed, $\mathcal{T}_\infty \cap \Sigma^\omega = \mathcal{M}_\infty \cap \Sigma^\omega$ and α_{\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 $\sigma_0, \dots, \sigma_n \in \alpha_{\preceq}(\mathcal{M}_\infty)$, then

$\forall i < n: \sigma_i \rightarrow \sigma_{i+1}$, so, $\sigma_0, \dots, \sigma_n \in \mathcal{T}_\infty$.

Assume $\sigma_0, \dots, \sigma_n \in \mathcal{T}_\infty$, then it can be completed into a maximal trace, either finite or infinite, and so, $\sigma_0, \dots, \sigma_n \in \alpha_{\preceq}(\mathcal{M}_\infty)$.

Note: no fixpoint transfer applies here.

Finite trace abstraction

Finite partial traces \mathcal{T} are an **abstraction** of all partial traces \mathcal{T}_∞
 (forget about infinite executions)

We have a **Galois embedding**:

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

- \sqsubseteq is the fused ordering on $\Sigma^* \cup \Sigma^\omega$:

$$A \sqsubseteq B \stackrel{\text{def}}{\iff} (A \cap \Sigma^*) \subseteq (B \cap \Sigma^*) \wedge (A \cap \Sigma^\omega) \supseteq (B \cap \Sigma^\omega)$$

- $\alpha_*(T) \stackrel{\text{def}}{=} T \cap \Sigma^*$
 (remove infinite traces)

- $\gamma_*(T) \stackrel{\text{def}}{=} T$
 (embedding)

- $\mathcal{T} = \alpha_*(\mathcal{T}_\infty)$

(proof on next slide)

Finite trace abstraction (proof)

proof:

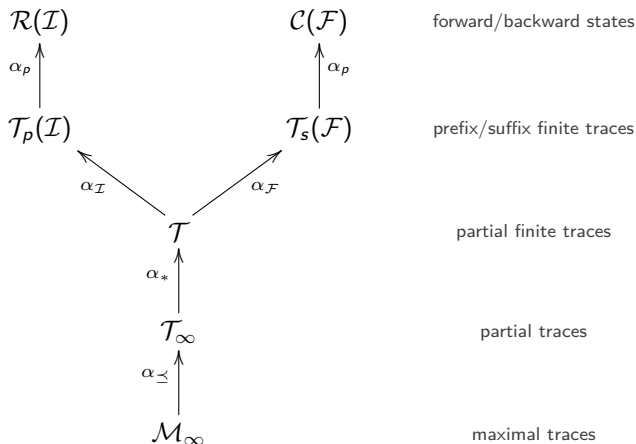
We have Galois embedding because:

- α_* and γ_* are monotonic,
- given $T \subseteq \Sigma^*$, we have $(\alpha_* \circ \gamma_*)(T) = T \cap \Sigma^* = T$,
- $(\gamma_* \circ \alpha_*)(T) = T \cap \Sigma^* \supseteq T$, 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 $(\mathcal{P}(\Sigma^*), \sqsubseteq)$, where $F_{s*}(T) \stackrel{\text{def}}{=} \Sigma \cup T \cap \tau$.

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

Enriched hierarchy of semantics



See [Cous02] for more semantics in this diagram.

Safety and liveness trace properties

Maximal 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 will ever occur”

- 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)

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

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

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 $S \subseteq \Sigma$:

$$\rho_p(S^\infty) = \lim(S^\infty) = S^\infty \implies \text{safety}$$

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

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

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

$$\rho_p((\Sigma^2)^\infty) = \Sigma^\infty \neq (\Sigma^2)^\infty \implies \text{not safety}$$

Proving safety properties

Proving that a program satisfies a safety property P is equivalent to proving that its **finite prefix abstraction** does

$$\mathcal{T}_p(\mathcal{I}) \subseteq P$$

proof sketch:

Soundness. Using the Galois connection between \mathcal{M}_∞ and \mathcal{T} , we get:

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

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

As $\mathcal{T}_p(\mathcal{I}) \subseteq P$, we have, by monotony, $\gamma_{*\prec}(\mathcal{T}_p(\mathcal{I})) \subseteq \gamma_{*\prec}(P) = P$.

Hence $\mathcal{M}_\infty \cap (\mathcal{I} \cdot \Sigma^\infty) \subseteq P$.

Completeness. $\mathcal{T}_p(\mathcal{I})$ provides an inductive invariant for P .

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 $\rho_{*\leq}(P) = \Sigma^\infty$

Examples and counter-examples:

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

$$\rho_P(\Sigma^*) = \Sigma^* \text{ and } \lim(\Sigma^*) = \Sigma^\infty \implies \text{liveness}$$

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

$$\rho_P(P) = P \cup \Sigma^* \text{ and } \lim(P \cup \Sigma^*) = \Sigma^\infty \implies \text{liveness}$$

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

$$\rho_P(S^\infty) = \lim(S^\infty) = S^\infty \neq \Sigma^\infty \text{ if } S \neq \Sigma \implies \text{not liveness}$$

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

$$\rho_P(\Sigma^{\leq k}) = \lim(\Sigma^{\leq k}) = \Sigma^{\leq k} \neq \Sigma^\infty \implies \text{not liveness}$$

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

Proving liveness properties

Variance proof method: (informal definition)

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

Example: termination proof

- find $f : \Sigma \rightarrow \mathcal{S}$ where $(\mathcal{S}, \sqsubseteq)$ is **well-ordered** (cf. previous course)
 f is called a “ranking function”
- $\sigma \in \mathcal{B} \implies f = \min \mathcal{S}$
- $\sigma \rightarrow \sigma' \implies f(\sigma') \sqsubset f(\sigma)$

generalizes the idea that f “counts” the number of steps remaining before termination

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: $\rho : \mathcal{P}(X) \rightarrow \mathcal{P}(X)$

- $\rho(x) \stackrel{\text{def}}{=} \bigcap \{ c \in \mathcal{C} \mid x \subseteq c \}$ (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 of a set in 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:

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

■ $\rho(x)$ 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

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.