VMCAI 2019 Winter School

Abstract Interpretation

Semantics, Verification, and Analysis

Patrick Cousot

pcousot@cs.nyu.edu   cs.nyu.edu/~pcousot

Friday, 01/11/2019, 09:00 – 12:30

Content

1. Semantics (45 mn)

2. Abstraction (45 mn)

   break (30mn)

3. Verification and proofs (45 mn)

4. Analysis (45 mn)
   - Numerical abstraction: see the VMCAI invited talk by Sylvie Putot (École polytechnique, France) on "Zonotopic abstract domains for numerical program analysis"
   - Symbolic abstraction: dependency analysis

Part 1

Semantics

Syntax

Context-free syntax of expressions

$$
\begin{array}{rclll}
\mathsf{x}, \mathsf{y}, \ldots & \in & \mathbb{V} & & \text{variables } (\mathbb{V} \text{ not empty}) \\
\mathsf{A} & \in & \mathbb{A} & ::= \mathsf{1} \mid \mathsf{x} \mid \mathsf{A}_1 - \mathsf{A}_2 & \text{arithmetic expressions} \\
\mathsf{B} & \in & \mathbb{B} & ::= \mathsf{A}_1 < \mathsf{A}_2 \mid \mathsf{B}_1 \ \mathsf{nand} \ \mathsf{B}_2 & \text{boolean expressions} \\
\mathsf{E} & \in & \mathbb{E} & ::= \mathsf{A} \mid \mathsf{B} & \text{expressions}
\end{array}
$$

Context-free syntax of program statements

S	::=		statement S $\in \mathbb{S}$
		x = A ;	assignment
	|	;	skip
	|	**if** (B) S	conditional
	|	**if** (B) S **else** S	
	|	**while** (B) S	iteration
	|	**break** ;	iteration break
	|	{ Sl }	compound statement
Sl	::=	Sl S | ϵ	statement list
P	::=	Sl	program P $\in \mathbb{P}$

Program labels

- To designate program points of program components, not part of the language
- Labels are unique
- $\text{at}\llbracket S \rrbracket$ label at entry of statement S
- $\text{after}\llbracket S \rrbracket$ label after exit of statement S
- $\text{escape}\llbracket S \rrbracket$ is it possible to break out of the statement S?
- $\text{break-to}\llbracket S \rrbracket$ where to break (exit label of enclosing loop)
- $\text{in}\llbracket S \rrbracket$ labels in statement S (excluding $\text{after}\llbracket S \rrbracket$ and $\text{break-to}\llbracket S \rrbracket$)
- $\text{labs}\llbracket S \rrbracket \triangleq \text{in}\llbracket S \rrbracket \cup \{\text{after}\llbracket S \rrbracket\}$
- $\text{labx}\llbracket S \rrbracket \triangleq \text{labs}\llbracket S \rrbracket \cup \left(\!\left|\; \text{escape}\llbracket S \rrbracket \mathrel{?} \{\text{break-to}\llbracket S \rrbracket\} \mathrel{:} \varnothing \;\right|\!\right)$

Axiomatic definition of program labelling

- We never define labels, just the properties they must satisfy
- Example $\mathtt{S} \triangleq \mathtt{if\ (B)\ S}_t\ \mathtt{else\ S}_f$:

$$\begin{aligned}
&\mathsf{in}[\![\mathtt{S}]\!] \triangleq \mathsf{at}[\![\mathtt{S}]\!] \cup \mathsf{in}[\![\mathtt{S}_t]\!] \cup \mathsf{in}[\![\mathtt{S}_f]\!]\\
&\mathsf{at}[\![\mathtt{S}]\!] \notin \mathsf{in}[\![\mathtt{S}_t]\!] \cup \mathsf{in}[\![\mathtt{S}_f]\!]\\
&\mathsf{in}[\![\mathtt{S}_t]\!] \cap \mathsf{in}[\![\mathtt{S}_f]\!] = \emptyset
\end{aligned}$$

$$\mathsf{after}[\![\mathtt{S}_t]\!] = \mathsf{after}[\![\mathtt{S}_f]\!] = \mathsf{after}[\![\mathtt{S}]\!]$$

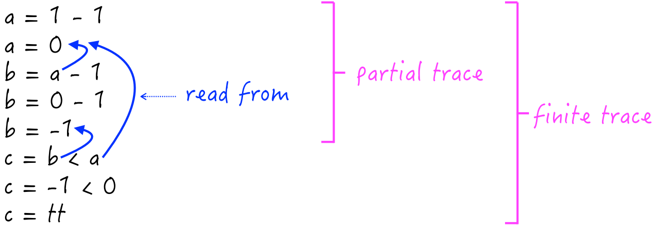
Prefix trace semantics

Trace of a hand computation

Hand computation of

$$\underbrace{\underbrace{\underbrace{(1-1)}_{a}-1}_{b} < \underbrace{(1-1)}_{a}}_{c}$$

is



$a = 1 - 1$

$a = 0$

$b = a - 1$

$b = 0 - 1$

$b = -1$

$c = b < a$

$c = -1 < 0$

$c = tt$

read from

partial trace

finite trace

Prefix trace

- A prefix trace is a finite observation of the program execution from entry
- A trace is a finite sequence of labels separated by actions (no memory state)

$$\ell_1 \xrightarrow{a_1} \ell_2 \xrightarrow{a_3} \ell_3 \xrightarrow{a_3} \ell_4 \xrightarrow{a_4} \ell_3 \xrightarrow{a_5} \ell_6 \ldots$$

  - labels ℓ_i: next action to be executed
  - actions a_i: records the computation done by a program step

Example of prefix trace

- default initialization to 0

ℓ_1 `x = x + 1 ;` (4.4)
  `while` ℓ_2 `(`tt`) {`
    ℓ_3 `x = x + 1 ;`
    `if` ℓ_4 `(x > 2)` ℓ_5 `break ;}`ℓ_6`;`ℓ_7

- $$\ell_1 \xrightarrow{\texttt{x = x + 1} = 1} \ell_2 \xrightarrow{tt} \ell_3 \xrightarrow{\texttt{x = x + 1} = 2} \ell_4 \xrightarrow{\neg(\texttt{x} > 2)} \ell_2 \xrightarrow{tt} \ell_3 \qquad (6.1)$$

- $$\ell_1 \xrightarrow{\texttt{x = x + 1} = 1} \ell_2 \xrightarrow{tt} \ell_3 \xrightarrow{\texttt{x = x + 1} = 2} \ell_4 \xrightarrow{\neg(\texttt{x} > 2)} \ell_2 \xrightarrow{tt} \ell_3 \xrightarrow{\texttt{x = x + 1} = 3}$$
  $$\ell_4 \xrightarrow{\texttt{x} > 2} \ell_5 \xrightarrow{\texttt{break}} \ell_6 \xrightarrow{\text{skip}} \ell_7$$

Values of variables

- Go back in the past to look for the last recorded assigned value (or 0 at initialization)

$$\begin{aligned} \boldsymbol{\rho}(\pi^{\ell} \xrightarrow{\mathsf{x} = \mathsf{E} = \nu} \ell')\mathsf{x} &\triangleq \nu \\ \boldsymbol{\rho}(\pi^{\ell} \xrightarrow{\cdots} \ell')\mathsf{x} &\triangleq \boldsymbol{\rho}(\pi^{\ell}) \quad \text{otherwise} \\ \boldsymbol{\rho}(\ell)\mathsf{x} &\triangleq 0 \end{aligned} \tag{6.2}$$

Prefix trace semantics

- Given a trace π_0 arriving at$[\![\mathsf{S}]\!]$,

  the prefix trace semantics $\mathcal{S}^*[\![\mathsf{S}]\!]$ of S specifies

  the trace π_1 of the execution of S from at$[\![\mathsf{S}]\!]$ with initial values defined by π_0

$$\xrightarrow{\qquad\pi_0\qquad} \underbrace{\mathsf{at}[\![\mathsf{S}]\!] \xrightarrow{\qquad\pi_1\qquad} \ell}_{\in\ \ \mathcal{S}^*[\![\mathsf{S}]\!](\pi_0\mathsf{at}[\![\mathsf{S}]\!])}$$

Structural rule-based definition of the prefix trace semantics

Structural prefix trace semantics at a statement

Prefix trace at a statement s

- $$\frac{}{\mathsf{at}[\![\mathsf{S}]\!] \in \widehat{\mathcal{S}}^{*}[\![\mathsf{S}]\!](\pi_1\mathsf{at}[\![\mathsf{S}]\!])} \qquad (6.7)$$

A prefix continuation of the traces $\pi_1\mathsf{at}[\![\mathsf{S}]\!]$ arriving at a program, statement or statement list S can be reduced to the program point $\mathsf{at}[\![\mathsf{S}]\!]$ at this program, statement or statement list S.

Semantics of arithmetic expressions

- An environment $\rho \in \mathbb{E}\mathtt{v}$ where $\mathbb{E}\mathtt{v} \triangleq \mathbb{V} \rightarrow \mathbb{Z}$ is a function ρ mapping a variable $\mathtt{x}$ to its value $\rho(\mathtt{x})$ in the set $\mathbb{Z}$ of all mathematical integers.
- Semantics of arithmetic expressions:

$$\begin{array}{rcl}
\mathscr{A}\llbracket \mathtt{1} \rrbracket\rho & \triangleq & 1 \\
\mathscr{A}\llbracket \mathtt{x} \rrbracket\rho & \triangleq & \rho(\mathtt{x}) \\
\mathscr{A}\llbracket \mathtt{A}_1 \mathtt{-} \mathtt{A}_2 \rrbracket\rho & \triangleq & \mathscr{A}\llbracket \mathtt{A}_1 \rrbracket\rho - \mathscr{A}\llbracket \mathtt{A}_2 \rrbracket\rho
\end{array} \tag{3.4}$$

Structural prefix trace semantics of an assignment statement

Prefix traces of an assignment statement S ::= ℓ x = A ;

- $$\frac{\upsilon = \mathscr{A}[\![\mathsf{A}]\!]\rho(\pi^{\ell})}{\ell \xrightarrow{\ \mathsf{x}\ =\ \mathsf{A} = \upsilon\ } \mathsf{after}[\![\mathsf{S}]\!] \in \widehat{\mathcal{S}}^{*}[\![\mathsf{S}]\!](\pi^{\ell})}$$

A prefix finite trace of an assignment ℓ x = E ; continuing some trace π^{ℓ} is ℓ followed by the event $\mathsf{x} = \upsilon$ where υ is the last value of x previously assigned to x on π^{ℓ} (otherwise initialized to 0) and finishing at the label $\mathsf{after}[\![\mathsf{S}]\!]$ after the assignment.

Structural prefix trace semantics of a conditional statement

Prefix traces of a conditional statement $\mathtt{S} ::= \mathtt{if}\ \ell\ \mathtt{(B)}\ \mathtt{S}_t$

- $$\frac{\mathscr{B}[\![\mathsf{B}]\!]\boldsymbol{\rho}(\pi_1{}^{\ell}) = \mathrm{ff}}{\ell \xrightarrow{\neg(\mathsf{B})} \mathsf{after}[\![\mathsf{S}]\!] \in \widehat{\mathcal{S}}^{\,*}[\![\mathsf{S}]\!](\pi_1{}^{\ell})} \tag{6.14}$$

- $$\frac{\mathscr{B}[\![\mathsf{B}]\!]\boldsymbol{\rho}(\pi_1{}^{\ell}) = \mathrm{tt}, \quad \pi_2 \in \widehat{\mathcal{S}}^{\,*}[\![\mathsf{S}_t]\!](\pi_1{}^{\ell} \xrightarrow{\mathsf{B}} \mathsf{at}[\![\mathsf{S}_t]\!])}{\ell \xrightarrow{\mathsf{B}} \mathsf{at}[\![\mathsf{S}_t]\!] \smallfrown \pi_2 \in \widehat{\mathcal{S}}^{\,*}[\![\mathsf{S}]\!](\pi_1{}^{\ell})} \tag{6.15}$$

$\smallfrown$ is trace concatenation

Structural prefix trace semantics of an empty statement list

Prefix traces of an empty statement list $\mathtt{Sl} ::= \epsilon$

- $$\frac{}{\mathsf{at}[\![\mathtt{Sl}]\!] \in \widehat{\mathcal{S}}^{*}[\![\mathtt{Sl}]\!](\pi\mathsf{at}[\![\mathtt{Sl}]\!])} \tag{6.11}$$

- A prefix/maximal trace π of the empty statement list ϵ continuing some trace is reduced to the program label $\mathsf{at}[\![\mathtt{Sl}]\!]$ at that empty statement.
- This case is redundant and already covered by (6.7).

Structural prefix trace semantics of a statement list

Prefix traces of a statement list `Sl ::= Sl′ S`

- $$\frac{\pi_2 \in \widehat{\mathcal{S}}^{*}[\![\mathtt{Sl}']\!](\pi_1)}{\pi_2 \in \widehat{\mathcal{S}}^{*}[\![\mathtt{Sl}]\!](\pi_1)} \tag{6.9}$$

- $$\frac{\pi_2 \in \widehat{\mathcal{S}}^{+}[\![\mathtt{Sl}']\!](\pi_1), \quad \pi_3 \in \widehat{\mathcal{S}}^{*}[\![\mathtt{S}]\!](\pi_1 \frown \pi_2)}{\pi_2 \frown \pi_3 \in \widehat{\mathcal{S}}^{*}[\![\mathtt{Sl}]\!](\pi_1)} \tag{6.10}$$

A prefix trace of `Sl′ S` continuing an initial trace π_1 can be a prefix trace of `Sl′` or a finite maximal trace of `Sl′` followed by a prefix trace of `S`.

Structural prefix trace semantics of an iteration statement

Prefix traces of an iteration statement $\texttt{S} ::= \texttt{while}\,^{\ell}\,\texttt{(B)}\ \texttt{S}_b$

- $$\frac{}{\ell \in \widehat{\mathcal{S}}^{*}[\![\texttt{S}]\!](\pi_1\ell)} \tag{6.20}$$

- $$\frac{\ell\pi_2\ell \in \widehat{\mathcal{S}}^{*}[\![\texttt{S}]\!](\pi_1\ell), \quad \mathcal{B}[\![\texttt{B}]\!]\boldsymbol{\rho}(\pi_1\ell\pi_2\ell) = \mathrm{ff}}{\ell\pi_2\ell \xrightarrow{\neg(\texttt{B})} \mathsf{after}[\![\texttt{S}]\!] \in \widehat{\mathcal{S}}^{*}[\![\texttt{S}]\!](\pi_1\ell)} \tag{6.21}$$

- $$\frac{\begin{array}{c}\ell\pi_2\ell \in \widehat{\mathcal{S}}^{*}[\![\texttt{S}]\!](\pi_1\ell), \quad \mathcal{B}[\![\texttt{B}]\!]\boldsymbol{\rho}(\pi_1\ell\pi_2\ell) = \mathrm{tt}, \\ \pi_3 \in \widehat{\mathcal{S}}^{*}[\![\texttt{S}_b]\!](\pi_1\ell\pi_2\ell \xrightarrow{\texttt{B}} \mathsf{at}[\![\texttt{S}_b]\!])\end{array}}{\ell\pi_2\ell \xrightarrow{\texttt{B}} \mathsf{at}[\![\texttt{S}_b]\!] \smallfrown \pi_3 \in \widehat{\mathcal{S}}^{*}[\![\texttt{S}]\!](\pi_1\ell)} \tag{6.22}$$

This is a forward, left recursive definition where $n+1$ iterations are n iterations followed by one more iteration.

Structural prefix trace semantics of an iteration statement

Prefix traces of an iteration statement $\mathtt{S} ::= \mathtt{while}\ \ell\ \mathtt{(B)}\ \mathtt{S}_b$

- $$\frac{}{\ell \in \widehat{\mathcal{S}}^*[\![\mathtt{S}]\!](\pi_1\ell)} \tag{6.20}$$

- $$\frac{\ell\pi_2\ell \in \boxed{\widehat{\mathcal{S}}^*[\![\mathtt{S}]\!](\pi_1\ell)}\,, \qquad \mathcal{B}[\![\mathtt{B}]\!]\rho(\pi_1\ell\pi_2\ell) = \mathit{ff}}{\ell\pi_2\ell \xrightarrow{\neg(\mathtt{B})} \mathsf{after}[\![\mathtt{S}]\!] \in \boxed{\widehat{\mathcal{S}}^*[\![\mathtt{S}]\!](\pi_1\ell)}} \tag{6.21}$$

- $$\frac{\begin{array}{c}\ell\pi_2\ell \in \boxed{\widehat{\mathcal{S}}^*[\![\mathtt{S}]\!](\pi_1\ell)}\,, \qquad \mathcal{B}[\![\mathtt{B}]\!]\rho(\pi_1\ell\pi_2\ell) = \mathit{tt}, \\ \pi_3 \in \widehat{\mathcal{S}}^*[\![\mathtt{S}_b]\!](\pi_1\ell\pi_2\ell \xrightarrow{\mathtt{B}} \mathsf{at}[\![\mathtt{S}_b]\!])\end{array}}{\ell\pi_2\ell \xrightarrow{\mathtt{B}} \mathsf{at}[\![\mathtt{S}_b]\!] \mathbin{\smallfrown} \pi_3 \in \boxed{\widehat{\mathcal{S}}^*[\![\mathtt{S}]\!](\pi_1\ell)}} \tag{6.22}$$

The definition is structural (depends on the already defined semantics of sub-components) and recursive (depends on itself) → might not be well-defined.

Structural prefix trace semantics of a break statement

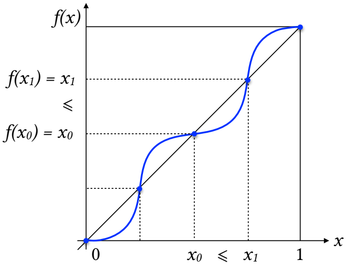
Prefix traces of a break statement $\mathtt{S} ::= {}^{\ell}\ \mathbf{break}\ \mathbf{;}$

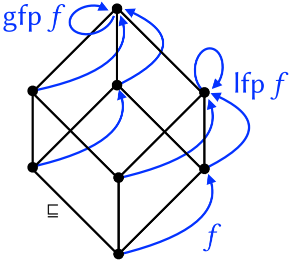
- $$\frac{}{\ell \xrightarrow{\mathbf{break}} \text{break-to}[\![\mathtt{S}]\!] \in \widehat{\mathcal{S}}^{*}[\![\mathtt{S}]\!](\pi^{\ell})} \qquad (6.25)$$

A prefix finite trace of a break ${}^{\ell}$ **break ;** continuing some initial trace π^{ℓ} is the trace ${}^{\ell}$ followed by the **break ;** event and ending at the break label break-to$[\![\mathtt{S}]\!]$ (which is the exit label of the closest enclosing iteration loop or else the program exit).

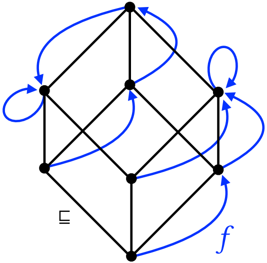
Structural fixpoint definition of the prefix trace semantics

Examples of fixpoints x such that $f(x) = x$





increasing function f



non-increasing function f

- As shown by Alfred Tarski, an increasing function on a complete lattice has at least one fixpoint and has a least one.

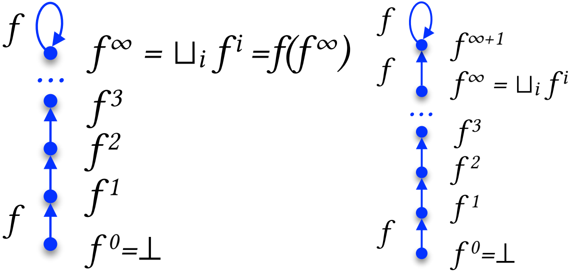
Tarski fixpoint theorem

Theorem (13.5, Tarski fixpoint theorem) An increasing function $f \in L \xrightarrow{\nearrow} L$ on a complete lattice $\langle L, \sqsubseteq, \bot, \top, \sqcap, \sqcup \rangle$ has a least fixpoint $\mathsf{lfp}^{\sqsubseteq} f = \bigsqcap\{x \in L \mid f(x) \sqsubseteq x\}$.

Tarski iterative fixpoint theorem

Theorem (13.14, Tarski iterative fixpoint)

- Let $f \in P \xrightarrow{\nearrow} P$ be an increasing function on a poset $\langle P, \sqsubseteq, \bot\rangle$ with infimum $\bot$.
- Define the iterates of f to be the sequence $f^0 = \bot$ and $f^{n+1} = f(f^n)$ for $n \in \mathbb{N}$.
- Assume that the least upper bound $\bigsqcup\{f^n \mid n \in \mathbb{N}\}$ exists and $f(\bigsqcup\{f^n \mid n \in \mathbb{N}\}) = \bigsqcup\{f(f^n) \mid n \in \mathbb{N}\}$
- Then f has a least fixpoint $\mathsf{lfp}^{\sqsubseteq} f = \bigsqcup\{f^n \mid n \in \mathbb{N}\}$.



Fixpoint prefix trace semantics of an assignment statement

Fixpoint prefix trace semantics of an assignment statement $\mathtt{S} ::= {}^{\ell}\ x\ \mathtt{=}\ \mathtt{E}\ \mathtt{;}$

$$\widehat{\mathcal{S}}^{*}[\![\mathsf{S}]\!](\pi^{\ell}) \quad = \quad \{\ell\} \cup \{\ell \xrightarrow{\mathtt{x = E} = v} \mathsf{after}[\![\mathsf{S}]\!] \mid v = \mathscr{E}[\![\mathsf{E}]\!]\boldsymbol{\rho}(\pi^{\ell})\}$$

- Example of basic case

Fixpoint prefix trace semantics of a statement list

Prefix traces of a statement list Sl ::= Sl′ S

$$\begin{aligned}\widehat{\mathcal{S}}^{*}[\![\mathtt{Sl}]\!](\pi_1) \quad = \quad & \widehat{\mathcal{S}}^{*}[\![\mathtt{Sl}']\!](\pi_1) \cup \\ & \{\pi_2 \frown \pi_3 \mid \pi_2 \in \widehat{\mathcal{S}}^{+}[\![\mathtt{Sl}']\!](\pi_1) \wedge \pi_3 \in \widehat{\mathcal{S}}^{*}[\![\mathtt{S}]\!](\pi_1 \frown \pi_2)\}\end{aligned} \qquad (15.2)$$

- $\widehat{\mathcal{S}}^{+}[\![\mathtt{Sl}']\!]$ contains the finite maximal traces of $\widehat{\mathcal{S}}^{*}[\![\mathtt{Sl}']\!]$
- Example of inductive case ($\widehat{\mathcal{S}}^{*}[\![\mathtt{Sl}]\!]$ defined in terms of $\widehat{\mathcal{S}}^{+}[\![\mathtt{Sl}']\!]$ and $\widehat{\mathcal{S}}^{*}[\![\mathtt{S}]\!]$ with $\mathtt{Sl}' \lhd \mathtt{Sl}$ and $\mathtt{S} \lhd \mathtt{Sl}$ where $\lhd$ is the strict component relation)

Fixpoint prefix trace semantics of an iteration

Prefix traces of an iteration statement $\texttt{S} ::= \texttt{while}\ ^{\ell}\ \texttt{(B)}\ \texttt{S}_b$

$$\mathcal{S}^*[\![\texttt{while}\ ^{\ell}\ \texttt{(B)}\ \texttt{S}_b]\!] \quad = \quad \mathsf{lfp}^{\dot\subseteq}\ \mathcal{F}^*[\![\texttt{while}\ ^{\ell}\ \texttt{(B)}\ \texttt{S}_b]\!] \tag{15.3}$$

$$\mathcal{F}^*[\![\texttt{while}\ ^{\ell}\ \texttt{(B)}\ \texttt{S}_b]\!](X)(\pi_1\ell') \quad \triangleq \quad \varnothing \qquad \text{when} \quad \ell' \neq \ell$$

$$\mathcal{F}^*[\![\texttt{while}\ ^{\ell}\ \texttt{(B)}\ \texttt{S}_b]\!](X)(\pi_1\ell) \quad \triangleq \quad \{\ell\} \tag{a}$$

$$\begin{aligned}\cup\ \{\ell'\pi_2\ell' \xrightarrow{\neg(\texttt{B})} \mathsf{after}[\![\texttt{S}]\!] \mid\ & \ell'\pi_2\ell' \in X(\pi_1\ell') \wedge {}\\ & \mathcal{B}[\![\texttt{B}]\!]\boldsymbol{\rho}(\pi_1\ell'\pi_2\ell') = \mathsf{ff} \wedge \ell' = \ell\}\end{aligned} \tag{b}$$

$$\begin{aligned}\cup\ \{\ell'\pi_2\ell' \xrightarrow{\texttt{B}} \mathsf{at}[\![\texttt{S}_b]\!] \smallfrown \pi_3 \mid\ & \ell'\pi_2\ell' \in X(\pi_1\ell') \wedge \mathcal{B}[\![\texttt{B}]\!]\boldsymbol{\rho}(\pi_1\ell'\pi_2\ell') = \mathsf{tt}\\ & \wedge\ \pi_3 \in \mathcal{S}^*[\![\texttt{S}_b]\!](\pi_1\ell'\pi_2\ell' \xrightarrow{\texttt{B}} \mathsf{at}[\![\texttt{S}_b]\!]) \wedge \ell' = \ell\}\end{aligned} \tag{c}$$

- Example of inductive fixpoint case
  - inductive: $\mathcal{S}^*[\![\texttt{while}\,^{\ell}\,\texttt{(B)}\ \texttt{S}_b]\!]$ defined in terms of $\mathcal{S}^*[\![\texttt{S}_b]\!]$ with $\texttt{S}_b \triangleleft \texttt{while}\,^{\ell}\,\texttt{(B)}\ \texttt{S}_b$
  - fixpoint: $\mathcal{S}^*[\![\texttt{while}\,^{\ell}\,\texttt{(B)}\ \texttt{S}_b]\!]$ recursively defined in terms of itself ($n + 1$ iterations are 1 iteration plus n iterations)

Maximal trace semantics

Maximal trace semantics, informally

- The maximal trace semantics $\mathcal{S}^{+\infty}[\![\mathsf{S}]\!] = \mathcal{S}^{+}[\![\mathsf{S}]\!] \cup \mathcal{S}^{\infty}[\![\mathsf{S}]\!]$ is derived from the prefix trace semantics $\mathcal{S}^{*}[\![\mathsf{S}]\!]$ by
  - keeping the longest finite traces $\mathcal{S}^{+}[\![\mathsf{S}]\!]$, and
  - passing to the limit $\mathcal{S}^{\infty}[\![\mathsf{S}]\!]$ of prefix-closed traces for infinite traces.

Finite maximal trace semantics

- $\mathcal{S}^{+}[\![\mathtt{S}]\!](\pi_1\mathsf{at}[\![\mathtt{S}]\!]) \triangleq \{\pi_2\ell \in \mathcal{S}^{*}[\![\mathtt{S}]\!](\pi_1\mathsf{at}[\![\mathtt{S}]\!]) \mid \ell = \mathsf{after}[\![\mathtt{S}]\!]\}$
- $\mathcal{S}^{+}[\![\mathtt{S}]\!](\pi_1\ell) = \varnothing$ when $\ell \neq \mathsf{at}[\![\mathtt{S}]\!]$

- $\mathcal{S}^{+}[\![\mathtt{S}]\!](\pi_1\mathsf{at}[\![\mathtt{S}]\!])$ is the set of maximal finite traces $\mathsf{at}[\![\mathtt{S}]\!]\pi_2\mathsf{after}[\![\mathtt{S}]\!]$ of $\mathtt{S}$ continuing the trace $\pi_1\mathsf{at}[\![\mathtt{S}]\!]$ and reaching $\mathsf{after}[\![\mathtt{S}]\!]$.

Prefixes of a trace

- If $\pi = \ell_0 \xrightarrow{e_0} \ldots \ell_i \xrightarrow{e_i} \ldots \ell_n$ is a finite trace then its prefix $\pi[0..p]$ at p is
  - π when $p \geqslant n$
  - $\ell_0 \xrightarrow{e_0} \ldots \ell_j \xrightarrow{e_j} \ldots \ell_p$ when $0 \leqslant p \leqslant n$.
- If $\pi = \ell_0 \xrightarrow{e_0} \ldots \ell_i \xrightarrow{e_i} \ldots$ is an infinite trace then its prefix $\pi[0..p]$ at p is $\ell_0 \xrightarrow{e_0} \ldots \ell_j \xrightarrow{e_j} \ldots \ell_p$.

Limit of prefix traces

- The limit $\lim \mathcal{T}$ of a set of traces $\mathcal{T}$ is the set of infinite traces which prefixes can be extended to a trace in $\mathcal{T}$.

$$\lim \mathcal{T} \quad \triangleq \quad \{\pi \in \mathbb{T}^{\infty} \mid \forall n \in \mathbb{N} \,.\, \exists p \geqslant n \,.\, \pi[0..p] \in \mathcal{T}\} \,.$$

- Let S be an iteration. $\langle \pi,\, \pi' \rangle \in \lim \mathcal{S}^{*}[\![\mathsf{S}]\!]$ where π' is infinite if and only if, whenever we take a prefix $\pi'[0..n]$ of π', it is a possible finite observation of the execution of S and so belongs to the prefix trace semantics $\langle \pi,\, \pi'[0..n] \rangle \in \mathcal{S}^{*}[\![\mathsf{S}]\!]$.

Infinite maximal trace semantics

$$\mathcal{S}^{\infty}[\![\mathsf{s}]\!] \quad \triangleq \quad \lim(\mathcal{S}^{*}[\![\mathsf{s}]\!])$$

Memory abstraction

Memory abstraction

- Abstraction from traces $\pi \in \mathbb{T}^+$ to environments $\rho \in \mathbb{E}\mathbb{v} \triangleq \mathbb{V} \to \mathbb{V}$ mapping variables $\mathsf{x} \in \mathbb{V}$ to their value $\rho(\mathsf{x}) \in \mathbb{V}$
- $\alpha(\pi) = \boldsymbol{\rho}(\pi)$

  where

$$\begin{aligned}
\boldsymbol{\rho}(\pi^{\ell} \xrightarrow{\mathsf{x}\ =\ \mathsf{E}\ =\ \nu} \ell')\mathsf{x} &\triangleq \nu \\
\boldsymbol{\rho}(\pi^{\ell} \xrightarrow{\ \cdots\ } \ell')\mathsf{x} &\triangleq \boldsymbol{\rho}(\pi^{\ell}) \quad \text{otherwise} \\
\boldsymbol{\rho}(\ell)\mathsf{x} &\triangleq 0
\end{aligned} \tag{6.2}$$

Properties

Formal property

- A property is the set of elements that satisfy this property.

- Examples:
  - $\{2k+1 \mid k \in \mathbb{N}\}$ is the property "to be an odd natural"
  - $\{2k \mid k \in \mathbb{Z}\}$ is the property "to be an even integer"
- Formally:
  - $\mathfrak{E}$ is a set of entities
  - A property of these entities is an element of $\wp(\mathfrak{E})$
  - Examples:
    - $\varnothing$ is false (ff)
    - $\mathfrak{E}$ is true (tt)
    - $e \in P$, $P \in \wp(\mathfrak{E})$ means "e has property P"
    - $P \subseteq P'$ is implication $\Rightarrow$ (P is *stronger* that P', P' is *weaker* that P)

Program property

- Syntactic point of view: a program property is the set of all programs which have this property (*e.g.* Rice theorem)
- Semantic point of view: : a program property is the set of all semantic of programs which have this property.
- By [program] property, we mean the semantic point of view.
- A program semantics is a set of traces (in $\wp(\mathbb{T}^+)$) so a program property is a set of sets of traces (in $\wp(\wp(\mathbb{T}^+))$)[1]

[1] sometimes called "hyperproperties"

The complete (boolean) lattice of formal properties

$$\langle \wp(\mathfrak{E}),\ \subseteq,\ \varnothing,\ \mathfrak{E},\ \cup,\ \cap,\ \neg \rangle$$

- $\wp(\mathfrak{E})$ properties of entities belonging to $\mathfrak{E}$
- $\subseteq$ implication
- $\varnothing$ false
- $\mathfrak{E}$ true
- $\cup$ disjonction, or
- $\cap$ conjunction, and
- $\neg$ negation, $\neg P \triangleq \mathfrak{E} \setminus P$

(the definition of "complete lattice" is forthcoming)

Posets and complete lattices

- A *poset* $\langle \mathbb{P}, \sqsubseteq \rangle$ is a set equipped with a binary relation $\sqsubseteq$ which is (forall $x, y, z \in \mathbb{P}$)
  - *reflexive*: $x \sqsubseteq x$
  - *antisymmetric*: $x \sqsubseteq y \wedge y \sqsubseteq x \Rightarrow x = y$
  - *transitive*: $x \sqsubseteq y \wedge y \sqsubseteq z \Rightarrow x \sqsubseteq z$

- A subset $S \in \wp(\mathbb{P})$ has a *least upper bound* (denoted $\sqcup S$) if and only if
  - $\sqcup S \in \mathbb{P}$
  - $\forall x \in S \,.\, x \sqsubseteq \sqcup S$
  - $\forall x \in S \,.\, x \sqsubseteq u \Rightarrow \sqcup S \sqsubseteq u$

- A *complete lattice* is a poset $\langle \mathbb{P}, \sqsubseteq \rangle$ in which any subset $S \in \wp(\mathbb{P})$ has a lub/join $\sqcup S$ (not only the finite ones).

Collecting semantics

Collecting semantics

- The strongest semantic property of program P

$$\mathcal{S}^{\mathbb{C}}[\![\mathtt{P}]\!] \quad \triangleq \quad \{\mathcal{S}^*[\![\mathtt{P}]\!]\}\ . \tag{8.5}$$

- Program P has property $P \in \wp(\wp(\mathbb{T}^{+\infty}))$ is
  - $\mathcal{S}^*[\![\mathtt{P}]\!] \in P$, or equivalently
  - $\{\mathcal{S}^*[\![\mathtt{P}]\!]\} \subseteq P$ *i.e.* P is implied by the collecting semantics of program P.
- So we can use implication $\subseteq$ ($\Rightarrow$) instead of $\in$ (with no direct equivalent for predicates in logic).
- Program verification $\{\mathcal{S}^*[\![\mathtt{P}]\!]\} \subseteq P$ is undecidable (Rice theorem)

Bibliography on semantics

References I

Cousot, Patrick (2002). “Constructive design of a hierarchy of semantics of a transition system by abstract interpretation”. *Theor. Comput. Sci.* 277.1-2, pp. 47–103 (5, 10, 3, 16).

Cousot, Patrick and Radhia Cousot (1979). “Constructive Versions of Tarski’s Fixed Point Theorems”. *Pacific Journal of Mathematics* 81.1, pp. 43–57 (7, 5, 3).

– (1992). “Inductive Definitions, Semantics and Abstract Interpretation”. In: *POPL*. ACM Press, pp. 83–94 (5, 10).

– (1995). “Compositional and Inductive Semantic Definitions in Fixpoint, Equational, Constraint, Closure-condition, Rule-based and Game-Theoretic Form”. In: *CAV*. Vol. 939. Lecture Notes in Computer Science. Springer, pp. 293–308 (10, 29).

– (2009). “Bi-inductive structural semantics”. *Inf. Comput.* 207.2, pp. 258–283 (5, 10, 3, 8).

Plotkin, Gordon D. (2004). “A structural approach to operational semantics”. *J. Log. Algebr. Program.* 60-61, pp. 17–139 (10).

The End of Part 1

Part 2

Abstraction

Abstraction

- We formalize the *abstraction and approximation of program properties*

- We show how a structural rule-based/fixpoint *abstract semantics* can be derived from the collecting semantics by *calculational design*.

Informal introduction to abstraction

Abstraction, informally

- Let be $\langle \wp(\mathfrak{E}), \subseteq \rangle$ be properties of entities (so called the *concrete domain*)

Abstraction, informally

- Let be $\langle \wp(\mathfrak{E}), \subseteq \rangle$ be properties of entities (so called the *concrete domain*)
- Consider a subset $\mathcal{A} \subseteq \wp(\mathfrak{E})$ of *properties of interest* $\langle \mathcal{A}, \subseteq \rangle$

Abstraction, informally

- Let be $\langle \wp(\mathfrak{E}), \subseteq \rangle$ be properties of entities (so called the *concrete domain*)
- Consider a subset $\mathcal{A} \subseteq \wp(\mathfrak{E})$ of *properties of interest* $\langle \mathcal{A}, \subseteq \rangle$
- Encode these properties of interest in an *abstract domain* $\langle A, \sqsubseteq \rangle$

Abstraction, informally

- Let be $\langle\wp(\mathfrak{E}), \subseteq\rangle$ be properties of entities (so called the *concrete domain*)
- Consider a subset $\mathcal{A} \subseteq \wp(\mathfrak{E})$ of *properties of interest* $\langle\mathcal{A}, \subseteq\rangle$
- Encode these properties of interest in an *abstract domain* $\langle A, \sqsubseteq\rangle$
- The decoding function $\gamma \in A \rightarrow \mathcal{A}$ is called the *concretization function*

Abstraction, informally

- Let be $\langle\wp(\mathfrak{E}),\ \subseteq\rangle$ be properties of entities (so called the *concrete domain*)
- Consider a subset $\mathcal{A} \subseteq \wp(\mathfrak{E})$ of *properties of interest* $\langle\mathcal{A},\ \subseteq\rangle$
- Encode these properties of interest in an *abstract domain* $\langle A,\ \sqsubseteq\rangle$
- The decoding function $\gamma \in A \rightarrow \mathcal{A}$ is called the *concretization function*
- Make proofs using abstract properties only

Abstraction, informally

- Let be $\langle \wp(\mathfrak{E}), \subseteq \rangle$ be properties of entities (so called the *concrete domain*)
- Consider a subset $\mathcal{A} \subseteq \wp(\mathfrak{E})$ of *properties of interest* $\langle \mathcal{A}, \subseteq \rangle$
- Encode these properties of interest in an *abstract domain* $\langle A, \sqsubseteq \rangle$
- The decoding function $\gamma \in A \to \mathcal{A}$ is called the *concretization function*
- Make proofs using abstract properties only
- So any concrete property must be over-approximated by a abstract property in $\mathcal{A} = \gamma(A)$

Abstraction, informally

- Let be $\langle\wp(\mathfrak{E}), \subseteq\rangle$ be properties of entities (so called the *concrete domain*)
- Consider a subset $\mathcal{A} \subseteq \wp(\mathfrak{E})$ of *properties of interest* $\langle\mathcal{A}, \subseteq\rangle$
- Encode these properties of interest in an *abstract domain* $\langle A, \sqsubseteq\rangle$
- The decoding function $\gamma \in A \rightarrow \mathcal{A}$ is called the *concretization function*
- Make proofs using abstract properties only
- So any concrete property must be over-approximated by a abstract property in $\mathcal{A} = \gamma(A)$
- If the abstract proof succeeds, it is valid in the concrete (*soundness*)

Abstraction, informally

- Let be $\langle\wp(\mathfrak{E}), \subseteq\rangle$ be properties of entities (so called the *concrete domain*)
- Consider a subset $\mathcal{A} \subseteq \wp(\mathfrak{E})$ of *properties of interest* $\langle\mathcal{A}, \subseteq\rangle$
- Encode these properties of interest in an *abstract domain* $\langle A, \sqsubseteq\rangle$
- The decoding function $\gamma \in A \rightarrow \mathcal{A}$ is called the *concretization function*
- Make proofs using abstract properties only
- So any concrete property must be over-approximated by a abstract property in $\mathcal{A} = \gamma(A)$
- If the abstract proof succeeds, it is valid in the concrete (*soundness*)
- If the abstract proof fails, you missed some property in $\wp(\mathfrak{E}) \setminus \mathcal{A}$ which is essential in the concrete proof (*incompleteness*)

Brahmagupta

- Brahmagupta (born c. 598, died after 665) was an Indian mathematician and astronomer;
- Invented the rule of signs (including to compute with zero);
- We explain his rule of sign as an abstract interpretation;
- Probably the very first example of abstract interpretation.

Structural collecting semantics

- Semantics

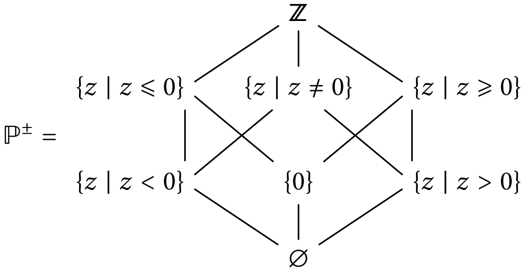
$$\begin{aligned}
\mathscr{A}[\![\mathtt{A}]\!] &\in (\mathbb{V} \to \mathbb{Z}) \to \mathbb{Z} \\
\mathscr{A}[\![\mathtt{1}]\!]\rho &\triangleq 1 \\
\mathscr{A}[\![\mathtt{x}]\!]\rho &\triangleq \rho(\mathtt{x}) \\
\mathscr{A}[\![\mathtt{A}_1 \,\mathtt{-}\, \mathtt{A}_2]\!]\rho &\triangleq \mathscr{A}[\![\mathtt{A}_1]\!]\rho - \mathscr{A}[\![\mathtt{A}_2]\!]\rho
\end{aligned}$$

- Collecting semantics

$$\begin{aligned}
\mathcal{S}^{\mathbb{C}}[\![\mathtt{A}]\!] &\in \wp((\mathbb{V} \to \mathbb{Z}) \to \mathbb{Z}) \\
\mathcal{S}^{\mathbb{C}}[\![\mathtt{1}]\!] &= \{\boldsymbol{\lambda}\,\rho \in \;\bullet\; 1\} \\
\mathcal{S}^{\mathbb{C}}[\![\mathtt{x}]\!] &= \{\boldsymbol{\lambda}\,\rho \in (\mathbb{V} \to \mathbb{Z}) \bullet \rho(\mathtt{x})\} \\
\mathcal{S}^{\mathbb{C}}[\![\mathtt{A}_1 \,\mathtt{-}\, \mathtt{A}_2]\!] &= \{\boldsymbol{\lambda}\,\rho \in (\mathbb{V} \to \mathbb{Z}) \bullet f_1(\rho) - f_2(\rho) \mid f_1 \in \mathcal{S}^{\mathbb{C}}[\![\mathtt{A}_1]\!] \wedge f_2 \in \mathcal{S}^{\mathbb{C}}[\![\mathtt{A}_2]\!]\}
\end{aligned}$$

Sign abstraction

Sign property (of an individual variable)



$\mathbb{P}^{\pm} =$

$\mathbb{Z}$

$\{z \mid z \leqslant 0\}$ $\{z \mid z \neq 0\}$ $\{z \mid z \geqslant 0\}$

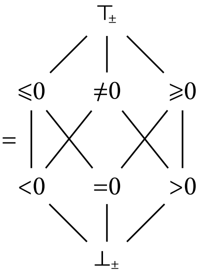
$\{z \mid z < 0\}$ $\{0\}$ $\{z \mid z > 0\}$

$\varnothing$

Example of Hasse diagram.

Encoding of sign properties (of an individual variable)

$\mathbb{P}^{\pm} =$

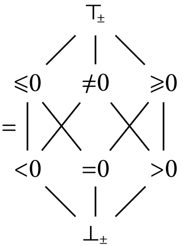


Concretization function:

$$
\begin{array}{lcl@{\qquad}lcl}
\gamma_{\pm}(\bot_{\pm}) & \triangleq & \varnothing & \gamma_{\pm}(\leqslant 0) & \triangleq & \{z \mid z \leqslant 0\} \\
\gamma_{\pm}(<0) & \triangleq & \{z \mid z < 0\} & \gamma_{\pm}(\neq 0) & \triangleq & \{z \mid z \neq 0\} \\
\gamma_{\pm}(=0) & \triangleq & \{0\} & \gamma_{\pm}(\geqslant 0) & \triangleq & \{z \mid z \geqslant 0\} \\
\gamma_{\pm}(>0) & \triangleq & \{z \mid z > 0\} & \gamma_{\pm}(\top_{\pm}) & \triangleq & \mathbb{Z}
\end{array}
$$

Encoding of sign properties (of an individual variable)

$\mathbb{P}^{\pm} =$



$\sqsubseteq$ is the partial order in $\mathbb{P}^{\pm}$

$\bigsqcup$ is the least upper bound in $\mathbb{P}^{\pm}$
e.g. $\bigsqcup\{\leqslant 0, \neq 0\} = \top_{\pm}$, $\bigsqcup \varnothing = \bot_{\pm}$

$\bigsqcap$ is the greatest lower bound in $\mathbb{P}^{\pm}$
e.g. $\bigsqcap\{\leqslant 0, \neq 0\} = <0$, $\bigsqcap \varnothing = \top_{\pm}$

Abstraction function: $$\alpha_{\pm}(P) \triangleq \begin{array}{l} \llbracket\, P \subseteq \varnothing \mathrel{?} \bot_{\pm} \\ \parallel\, P \subseteq \{z \mid z < 0\} \mathrel{?} <0 \\ \parallel\, P \subseteq \{0\} \mathrel{?} =0 \\ \parallel\, P \subseteq \{z \mid z > 0\} \mathrel{?} >0 \\ \parallel\, P \subseteq \{z \mid z \leqslant 0\} \mathrel{?} \leqslant 0 \\ \parallel\, P \subseteq \{z \mid z \neq 0\} \mathrel{?} \neq 0 \\ \parallel\, P \subseteq \{z \mid z \geqslant 0\} \mathrel{?} \geqslant 0 \\ \mathrel{:} \top_{\pm} \,\rrbracket \end{array} \tag{3.28}$$

Galois connection

- The pair $\langle\alpha_\pm,\ \gamma_\pm\rangle$ of functions satisfies $\alpha_\pm(P) \sqsubseteq Q \Leftrightarrow P \subseteq \gamma_\pm(Q)$

$$
\begin{array}{ll}
\quad \alpha_\pm(P) \sqsubseteq Q & \\
\Leftrightarrow\ \alpha_\pm(P) \sqsubseteq {\neq}0 & \wr\text{in case } Q = {\neq}0\text{, other cases are similar}\wr \\
\Leftrightarrow\ \alpha_\pm(P) \in \{\bot_\pm, {<}0, {\neq}0, {>}0\} & \wr\text{def. }\wr \\
\Leftrightarrow\ P \subseteq \varnothing \vee P \subseteq \{z \mid z < 0\} \vee P \subseteq \{z \mid z > 0\} \vee P \subseteq \{z \mid z \neq 0\} & \wr\text{def. } \alpha_\pm\wr \\
\Leftrightarrow\ P \subseteq \{z \mid z \neq 0\} & \wr\text{def. } \subseteq\wr \\
\Leftrightarrow\ P \subseteq \gamma_\pm({\neq}0) & \wr\text{def. } \gamma_\pm\wr \\
\Leftrightarrow\ P \subseteq \gamma_\pm(Q) & \wr\text{case } Q = {\neq}0\wr
\end{array}
$$

- This is the definition of a Galois connection
- We write $\langle\wp(\mathbb{Z}),\ \subseteq\rangle \xrightleftharpoons[\alpha_\pm]{\gamma_\pm} \langle\mathbb{P}^\pm,\ \sqsubseteq\rangle$
- This will be further generalized.

Sign abstract semantics

$$\begin{array}{rcl}
\mathcal{S}[\![\mathtt{A}]\!] & \in & (\mathbb{V} \rightarrow \mathbb{P}^{\pm}) \rightarrow \mathbb{P}^{\pm} \\
\\
\mathcal{S}[\![\mathtt{1}]\!]P & \triangleq & >0 \\
\mathcal{S}[\![\mathtt{x}]\!]P & \triangleq & P(\mathtt{x}) \\
\mathcal{S}[\![\mathtt{A}_1 - \mathtt{A}_2]\!]P & \triangleq & \mathcal{S}[\![\mathtt{A}_1]\!]P -_{\pm} \mathcal{S}[\![\mathtt{A}_2]\!]P\}
\end{array}$$

	$x -_{\pm} y$	y							
		$\perp_{\pm}$	<0	$=0$	>0	$\leqslant 0$	$\neq 0$	$\geqslant 0$	$\top_{\pm}$
	$\perp_{\pm}$	$\perp_{\pm}$	$\perp_{\pm}$	$\perp_{\pm}$	$\perp_{\pm}$	$\perp_{\pm}$	$\perp_{\pm}$	$\perp_{\pm}$	$\perp_{\pm}$
	<0	$\perp_{\pm}$	$\top_{\pm}$	<0	<0	$\top_{\pm}$	$\top_{\pm}$	<0	$\top_{\pm}$
	$=0$	$\perp_{\pm}$	>0	$=0$	<0	$\geqslant 0$	$\neq 0$	$\leqslant 0$	$\top_{\pm}$
x	>0	$\perp_{\pm}$	>0	>0	$\top_{\pm}$	>0	$\top_{\pm}$	$\top_{\pm}$	$\top_{\pm}$
	$\leqslant 0$	$\perp_{\pm}$	>0	$\leqslant 0$	$\top_{\pm}$	$\top_{\pm}$	$\top_{\pm}$	$\leqslant 0$	$\top_{\pm}$
	$\neq 0$	$\perp_{\pm}$	$\top_{\pm}$	$\neq 0$	$\top_{\pm}$	$\top_{\pm}$	$\top_{\pm}$	$\top_{\pm}$	$\top_{\pm}$
	$\geqslant 0$	$\perp_{\pm}$	>0	$\geqslant 0$	$\top_{\pm}$	$\geqslant 0$	$\top_{\pm}$	$\top_{\pm}$	$\top_{\pm}$
	$\top_{\pm}$	$\perp_{\pm}$	$\top_{\pm}$	$\top_{\pm}$	$\top_{\pm}$	$\top_{\pm}$	$\top_{\pm}$	$\top_{\pm}$	$\top_{\pm}$

Calculational design of the rule of signs

$$
\begin{array}{rl}
& {>}0 \mathbin{-_{\pm}} {\leqslant}0 \\
\triangleq & \alpha_{\pm}(\{x - y \mid x \in \gamma_{\pm}({>}0) \wedge y \in \gamma_{\pm}({\leqslant}0)\} \\
= & \alpha_{\pm}(\{x - y \mid x > 0 \wedge y \leqslant 0\}) \\
= & \alpha_{\pm}(\{x - y \mid x > 0 \wedge -y \geqslant 0\}) \\
\subseteq & \alpha_{\pm}(\{x - y \mid x - y > 0\}) \\
= & \alpha_{\pm}(\{z \mid z > 0\}) \\
= & {>}0
\end{array}
$$

Same calculus for all other cases (can be automated with a theorem prover, so called *predicate abstraction*).

Sign abstract semantics (revisited)

- If a variable y has sign $\bot_\pm$, then $\gamma_\pm(\bot_\pm) = \emptyset$ so the expression is not evaluated hence returns no value
- Define $\mathop{\downarrow\!\!\!\top}^{\pm}[P]s \triangleq \llparenthesis\, \exists \mathsf{y} \in \mathbb{V} \,.\, P(\mathsf{y}) = \bot_\pm \;?\; \bot_\pm \;:\; s \,\rrparenthesis$ to force returning $\bot_\pm$ if a variable has abstract value $\bot_\pm$
- The following sign abstract semantics is more precise:

$$\begin{aligned}
\mathcal{S}^{\pm}[\![\mathtt{1}]\!]P &= \mathop{\downarrow\!\!\!\top}^{\pm}[P](>0) \\
\mathcal{S}^{\pm}[\![\mathsf{x}]\!]P &= \mathop{\downarrow\!\!\!\top}^{\pm}[P](P(\mathsf{x})) \\
\mathcal{S}^{\pm}[\![\mathsf{A}_1 - \mathsf{A}_2]\!]P &= (\mathcal{S}^{\pm}[\![\mathsf{A}_1]\!]P) -_{\pm} (\mathcal{S}^{\pm}[\![\mathsf{A}_2]\!]P)
\end{aligned} \tag{3.19}$$

- It follows that $\exists \mathsf{x} \in \mathbb{V} \,.\, P(\mathsf{x}) = \bot_\pm$ implies $\mathcal{S}^{\pm}[\![\mathsf{A}]\!]P = \bot_\pm$.

Soundness

Sign concretization

- Sign

$$\begin{array}{rcl@{\qquad}rcl}
\gamma_{\pm}(\bot_{\pm}) & \triangleq & \varnothing & \gamma_{\pm}(\leqslant 0) & \triangleq & \{z \in \mathbb{Z} \mid z \leqslant 0\} \\
\gamma_{\pm}(<0) & \triangleq & \{z \in \mathbb{Z} \mid z < 0\} & \gamma_{\pm}(\neq 0) & \triangleq & \{z \in \mathbb{Z} \mid z \neq 0\} \\
\gamma_{\pm}(=0) & \triangleq & \{0\} & \gamma_{\pm}(\geqslant 0) & \triangleq & \{z \in \mathbb{Z} \mid z \geqslant 0\} \\
\gamma_{\pm}(>0) & \triangleq & \{z \in \mathbb{Z} \mid z > 0\} & \gamma_{\pm}(\top_{\pm}) & \triangleq & \mathbb{Z}
\end{array} \tag{3.21}$$

- Sign environment

$$\dot{\gamma}_{\pm}(\overset{\pm}{\rho}) \triangleq \{\rho \in \mathbb{V} \to \mathbb{Z} \mid \forall \mathsf{x} \in \mathbb{V} \,.\, \rho(\mathsf{x}) \in \gamma_{\pm}(\overset{\pm}{\rho}(\mathsf{x}))\} \tag{3.22}$$

- Sign abstract property

$$\ddot{\gamma}_{\pm}(\overline{P}) \triangleq \{\boldsymbol{\mathcal{S}} \in (\mathbb{V} \to \mathbb{Z}) \to \mathbb{Z} \mid \forall \overset{\pm}{\rho} \in \mathbb{V} \to \mathbb{P}^{\pm} \,.\, \forall \rho \in \dot{\gamma}_{\pm}(\overset{\pm}{\rho}) \,.\, \boldsymbol{\mathcal{S}}(\rho) \in \gamma_{\pm}(\overline{P}(\overset{\pm}{\rho}))\} \tag{3.23}$$

Sign abstraction

- Value property

$$\begin{array}{rcll} \alpha_{\pm}(P) & \triangleq & \llparenthesis\, P \subseteq \varnothing \;?\; \bot_{\pm} & \\ & & \parallel\, P \subseteq \{z \mid z < 0\} \;?\; {<}0 & \\ & & \parallel\, P \subseteq \{0\} \;?\; {=}0 & \\ & & \parallel\, P \subseteq \{z \mid z > 0\} \;?\; {>}0 & \\ & & \parallel\, P \subseteq \{z \mid z \leqslant 0\} \;?\; {\leqslant}0 & \\ & & \parallel\, P \subseteq \{z \mid z \neq 0\} \;?\; {\neq}0 & \\ & & \parallel\, P \subseteq \{z \mid z \geqslant 0\} \;?\; {\geqslant}0 & \\ & & :\; \top_{\pm} \,\rrparenthesis & \end{array} \tag{3.28}$$

- Environment property

$$\dot{\alpha}_{\pm}(P) \quad \triangleq \quad \boldsymbol{\lambda}\, \mathsf{x} \in \mathbb{V} \bullet \alpha_{\pm}(\{\rho(\mathsf{x}) \mid \rho \in P\}) \tag{3.31}$$

- Semantics property

$$\ddot{\alpha}_{\pm}(P) \quad \triangleq \quad \boldsymbol{\lambda}\, \overset{\pm}{\rho} \in \mathbb{V} \to \mathbb{P}^{\pm} \bullet \alpha_{\pm}(\{\mathcal{S}(\rho) \mid \mathcal{S} \in P \wedge \rho \in \dot{\gamma}_{\pm}(\overset{\pm}{\rho})\}) \tag{3.32}$$

Example of environment property abstraction

- The property of environments such that x is equal to 1:

$$\{\rho \in \mathbb{V} \to \mathbb{Z} \mid \rho(\mathsf{x}) = 1\}$$

- Sign abstraction:

$$\begin{aligned}
& \dot{\alpha}_{\pm}(\{\rho \in \mathbb{V} \to \mathbb{Z} \mid \rho(\mathsf{x}) = 1\}) \\
\triangleq\ & \boldsymbol{\lambda}\, \mathsf{y} \in \mathbb{V} \centerdot \alpha_{\pm}(\{\rho(\mathsf{y}) \mid \rho \in \{\rho \in \mathbb{V} \to \mathbb{Z} \mid \rho(\mathsf{x}) = 1\}\}) \\
=\ & \boldsymbol{\lambda}\, \mathsf{y} \in \mathbb{V} \centerdot \llparenthesis\, \mathsf{y} = \mathsf{x} \mathrel{?} \alpha_{\pm}(\{1\}) \mathrel{:} \alpha_{\pm}(\mathbb{Z}) \,\rrparenthesis \\
=\ & \boldsymbol{\lambda}\, \mathsf{y} \in \mathbb{V} \centerdot \llparenthesis\, \mathsf{y} = \mathsf{x} \mathrel{?} {>}0 \mathrel{:} \top_{\pm} \,\rrparenthesis
\end{aligned}$$

- Sign concretization:

$$\begin{aligned}
& \dot{\gamma}_{\pm}(\boldsymbol{\lambda}\, \mathsf{y} \in \mathbb{V} \centerdot \llparenthesis\, \mathsf{y} = \mathsf{x} \mathrel{?} {>}0 \mathrel{:} \top_{\pm} \,\rrparenthesis) \\
\triangleq\ & \{\rho \in \mathbb{V} \to \mathbb{Z} \mid \forall \mathsf{z} \in \mathbb{V} \,.\, \rho(\mathsf{z}) \in \gamma_{\pm}(\boldsymbol{\lambda}\, \mathsf{y} \in \mathbb{V} \centerdot \llparenthesis\, \mathsf{y} = \mathsf{x} \mathrel{?} {>}0 \mathrel{:} \top_{\pm} \,\rrparenthesis(\mathsf{z}))\} \\
=\ & \{\rho \in \mathbb{V} \to \mathbb{Z} \mid \rho(\mathsf{x}) > 0\}
\end{aligned}$$

Galois connections

- Value to sign

$$\langle \wp(\mathbb{Z}),\ \subseteq\rangle \underset{\alpha_\pm}{\overset{\gamma_\pm}{\leftrightarrows}} \langle \mathbb{P}^\pm,\ \sqsubseteq\rangle$$

- Value environment to sign environment

$$\langle \wp(\mathbb{V} \rightarrow \mathbb{Z}),\ \subseteq\rangle \underset{\dot{\alpha}_\pm}{\overset{\dot{\gamma}_\pm}{\leftrightarrows}} \langle \mathbb{V} \rightarrow \mathbb{P}^\pm,\ \dot{\sqsubseteq}_\pm\rangle$$

- Semantic to sign abstract semantic property

$$\langle \wp((\mathbb{V} \rightarrow \mathbb{Z}) \rightarrow \mathbb{Z}),\ \subseteq\rangle \underset{\ddot{\alpha}_\pm}{\overset{\ddot{\gamma}_\pm}{\leftrightarrows}} \langle (\mathbb{V} \rightarrow \mathbb{P}^\pm) \rightarrow \mathbb{P}^\pm,\ \dot{\sqsubseteq}_\pm\rangle$$

Soundness of the abstract sign semantics

- The abstract sign semantics is an abstraction of the collecting property

$$\begin{array}{rrcl} & \mathcal{S}^{\mathbb{C}}[\![\mathsf{A}]\!] & \subseteq & \ddot{\gamma}_{\pm}(\mathcal{S}^{\pm}[\![\mathsf{A}]\!]) \\ \Leftrightarrow & \ddot{\alpha}_{\pm}(\mathcal{S}^{\mathbb{C}}[\![\mathsf{A}]\!]) & \ddot{\sqsubseteq} & \mathcal{S}^{\pm}[\![\mathsf{A}]\!] \end{array}$$

- Precision loss: if the sign of x is $\leqslant 0$ then the sign of x - x is $\top_{\pm}$ not $=0$
- The absolute value is abstracted away
- No precision loss for multiplication ×

Calculational design of the sign semantics

Case when $\exists \mathsf{x} \in \mathbb{V} \mathrel{.} \overset{\pm}{\rho}(\mathsf{x}) = \bot_\pm$ so that $\dot{\gamma}_\pm(\overset{\pm}{\rho}) = \varnothing$

$$
\begin{array}{ll}
- & \ddot{\alpha}_\pm(\mathcal{S}^{\mathbb{C}}[\![\mathsf{A}]\!])\overset{\pm}{\rho} \\
= & \alpha_\pm(\{\mathcal{S}(\rho) \mid \mathcal{S} \in \mathcal{S}^{\mathbb{C}}[\![\mathsf{A}]\!] \land \rho \in \dot{\gamma}_\pm(\overset{\pm}{\rho})\}) \qquad \text{⟅def. (3.32) of } \ddot{\alpha}_\pm\text{⟆} \\
= & \alpha_\pm(\{\mathcal{A}[\![\mathsf{A}]\!](\rho) \mid \rho \in \dot{\gamma}_\pm(\overset{\pm}{\rho})\}) \qquad \text{⟅def. (3.11) of } \mathcal{S}^{\mathbb{C}}[\![\mathsf{A}]\!]\text{⟆} \\
= & \alpha_\pm(\varnothing) \qquad \text{⟅}\exists \mathsf{x} \in \mathbb{V} \mathrel{.} \overset{\pm}{\rho}(\mathsf{x}) = \bot_\pm \text{ so that } \dot{\gamma}_\pm(\overset{\pm}{\rho}) = \varnothing\text{⟆} \\
= & \bot_\pm \qquad \text{⟅def. (3.28) of } \alpha_\pm\text{⟆} \\
\triangleq & \mathcal{S}^{\pm}[\![\mathsf{A}]\!]\overset{\pm}{\rho}
\end{array}
$$

⟅in accordance with (3.19) such that $\exists \mathsf{x} \in \mathbb{V} \mathrel{.} \overset{\pm}{\rho}(\mathsf{x}) = \bot_\pm$ implies $\mathcal{S}^{\pm}[\![\mathsf{A}]\!]\overset{\pm}{\rho} = \bot_\pm$.⟆

Homework: Case of a variable x

$$
\begin{array}{lll}
 & \ddot{\alpha}_\pm(\mathcal{S}^{\mathbb{C}}\llbracket \mathrm{x}\rrbracket)\overset{\pm}{\rho} & \\
= & \alpha_\pm(\{\mathcal{S}(\rho) \mid \mathcal{S} \in \mathcal{S}^{\mathbb{C}}\llbracket \mathrm{x}\rrbracket \wedge \rho \in \dot{\gamma}_\pm(\overset{\pm}{\rho})\}) & \wr\text{def. (3.32) of } \ddot{\alpha}_\pm \int \\
= & \alpha_\pm(\{\mathcal{A}\llbracket \mathrm{x}\rrbracket(\rho) \mid \rho \in \dot{\gamma}_\pm(\overset{\pm}{\rho})\}) & \wr\text{def. (3.11) of } \mathcal{S}^{\mathbb{C}}\llbracket \mathrm{x}\rrbracket \int \\
= & \alpha_\pm(\{\rho(\mathrm{x}) \mid \rho \in \dot{\gamma}_\pm(\overset{\pm}{\rho})\}) & \wr\text{def. (3.4) of } \mathcal{A}\llbracket \mathrm{x}\rrbracket \int \\
= & \alpha_\pm(\{\rho(\mathrm{x}) \mid \forall \mathrm{y} \in \mathbb{V} \,.\, \rho(\mathrm{y}) \in \gamma_\pm(\overset{\pm}{\rho}(\mathrm{y}))\}) & \wr\text{def. (3.22) of } \dot{\gamma}_\pm \int \\
= & \alpha_\pm(\{\rho(\mathrm{x}) \mid \rho(\mathrm{x}) \in \gamma_\pm(\overset{\pm}{\rho}(\mathrm{x}))\}) & \\
 & \multicolumn{2}{r}{\wr\text{since } \gamma_\pm(\overset{\pm}{\rho}(\mathrm{y})) \text{ is not empty so for } \mathrm{y} \neq \mathrm{x},\ \rho(\mathrm{y}) \text{ can be chosen arbitrarily to satisfy } \rho(\mathrm{y}) \in \gamma_\pm(\overset{\pm}{\rho}(\mathrm{y}))\int} \\
= & \alpha_\pm(\{x \mid x \in \gamma_\pm(\overset{\pm}{\rho}(\mathrm{x}))\}) & \wr\text{letting } x = \rho(\mathrm{x})\int \\
= & \alpha_\pm(\gamma_\pm(\overset{\pm}{\rho}(\mathrm{x}))) & \wr\text{since } S = \{x \mid z \in S\} \text{ for any set } S\int \\
= & \overset{\pm}{\rho}(\mathrm{x}) & \wr\text{by (3.35), } \alpha_\pm \circ \gamma_\pm \text{ is the identity}\int \\
\triangleq & \mathcal{S}^{\pm}\llbracket \mathrm{x}\rrbracket\overset{\pm}{\rho} & \wr\text{in accordance with (3.19) when } \forall \mathrm{y} \in \mathbb{V} \,.\, \overset{\pm}{\rho}(\mathrm{y}) \neq \bot_\pm\int
\end{array}
$$

Other cases

- similar for $\ddot{\alpha}_{\pm}(\mathcal{S}^{\mathbb{C}}[\![\mathtt{1}]\!])\overset{\pm}{\rho}$
- by structural induction for $\ddot{\alpha}_{\pm}(\mathcal{S}^{\mathbb{C}}[\![\mathtt{A}_1 \mathtt{-} \mathtt{A}_2]\!])$
- See the course notes in the appendix.

Chapter 11

Galois Connections and Abstraction

Galois connections

- Given posets $\langle \mathcal{C}, \sqsubseteq\rangle$ (the *concrete domain*) and $\langle \mathcal{A}, \preccurlyeq\rangle$ (the *abstract domain*), the pair $\langle \alpha, \gamma\rangle$ of functions $\alpha \in \mathcal{C} \rightarrow \mathcal{A}$ (the *lower adjoint* or *abstraction*) and $\gamma \in \mathcal{A} \rightarrow \mathcal{C}$ (the *upper-adjoint* or *concretization*) is a *Galois connection* (GC) if and only if

$$\forall P \in \mathcal{C} \mathrel{.} \forall \overline{P} \in \mathcal{A} \mathrel{.} \alpha(P) \preccurlyeq \overline{P} \Leftrightarrow P \sqsubseteq \gamma(\overline{P}) \tag{11.1}$$

which we write

$$\langle \mathcal{C}, \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \preccurlyeq\rangle \ .$$

Example: homomorphic/partition abstraction

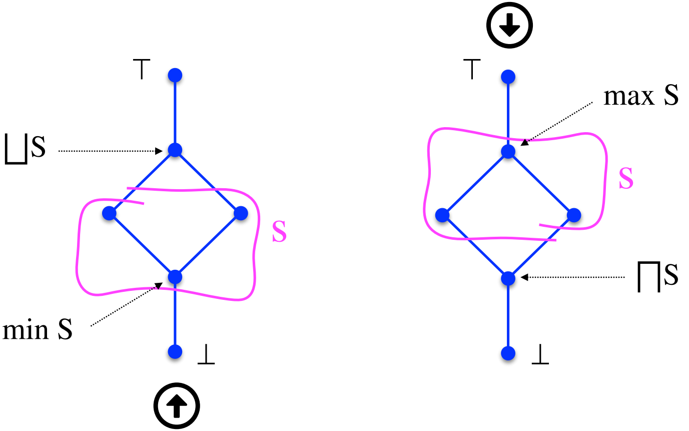
- Let C and A be sets, $h \in C \to A$
- $\alpha_h(S) \triangleq \{h(e) \mid e \in S\}$
- $\gamma_h(\overline{S}) \triangleq \{e \in S \mid h(e) \in \overline{S}\}$
- $\langle \wp(C),\ \subseteq\rangle \underset{\alpha_h}{\overset{\gamma_h}{\leftrightarrows}} \langle \wp(A),\ \subseteq\rangle$

Proof

$$
\begin{aligned}
& \alpha_h(S) \subseteq \overline{S} \\
\Leftrightarrow\ & \{h(e) \mid e \in S\} \subseteq \overline{S} && \text{(def. } \alpha_h\text{)} \\
\Leftrightarrow\ & \forall e \in S \,.\, h(e) \in \overline{S} && \text{(def. } \subseteq\text{)} \\
\Leftrightarrow\ & S \subseteq \{e \mid h(e) \in \overline{S}\} && \text{(def. } \subseteq\text{)} \\
\Leftrightarrow\ & S \subseteq \gamma_h(\overline{S}) && \text{(def. } \gamma_h\text{)} \quad \square
\end{aligned}
$$

Duality in order theory

- The properties derived for $\sqsubseteq$, $\bot$, $\top$, $\sqcup$, $\max$, $\sqcap$, $\min$, *etc*. are valid for the dual $\sqsupseteq$, $\top$, $\bot$, $\sqcap$, $\min$, $\sqcup$, $\max$, *etc*.
- Intuition:



Dual of a Galois connection

- The dual of a Galois connection $\langle \mathcal{C}, \sqsubseteq\rangle \xleftrightarrow[\alpha]{\gamma} \langle \mathcal{A}, \preccurlyeq\rangle$ is the Galois connection $\langle \mathcal{A}, \preccurlyeq\rangle \xleftrightarrow[\gamma]{\alpha} \langle \mathcal{C}, \sqsubseteq\rangle$

Proof $\langle \mathcal{C}, \sqsubseteq\rangle \xleftrightarrow[\alpha]{\gamma} \langle \mathcal{A}, \preccurlyeq\rangle$

$\Leftrightarrow \alpha(x) \preccurlyeq y \Leftrightarrow x \sqsubseteq \gamma(y)$ ⟅def. Galois connection⟆

$\alpha(x) \succcurlyeq y \Leftrightarrow x \sqsupseteq \gamma(y)$ ⟅dual statement⟆

$\Leftrightarrow \gamma(y) \sqsubseteq x \Leftrightarrow y \preccurlyeq \alpha(x)$ ⟅inverse order $x \sqsupseteq y \Leftrightarrow y \sqsubseteq x$⟆

$\Leftrightarrow \gamma(x) \sqsubseteq y \Leftrightarrow x \preccurlyeq \alpha(y)$ ⟅dummy variable renaming⟆

$\Leftrightarrow \langle \mathcal{A}, \preccurlyeq\rangle \xleftrightarrow[\gamma]{\alpha} \langle \mathcal{C}, \sqsubseteq\rangle$ ⟅def. Galois connection⟆ □

- Dualization of a statement involving Galois connections consists in exchanging the adjoints
- If an adjoint has a property, its adjoint has the dual property

Example of dualization

Lemma 1 If $\langle C,\ \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A},\ \preccurlyeq\rangle$ then α is increasing. □

Proof Assume $P \sqsubseteq P'$. By $\alpha(P') \preccurlyeq \alpha(P')$ we have $P' \sqsubseteq \gamma(\alpha(P'))$ so $P \sqsubseteq \gamma(\alpha(P'))$ by transitivity hence $\alpha(P) \sqsubseteq \alpha(P')$ by definition of a GC, proving that α is increasing. □

Lemma 2 If $\langle C,\ \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A},\ \preccurlyeq\rangle$ then γ is increasing. □

Proof By duality (increasing is self-dual so the dual of "α is increasing" is "γ is increasing"). □

Example of dualization

- In a Galois connection $\langle C, \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \preccurlyeq\rangle$ we have $\alpha \circ \gamma \circ \alpha = \alpha$

  Proof **homework** For all $x \in C$ and $y \in \mathcal{A}$,

  — $\alpha(x) \preccurlyeq \alpha(x)$ ⦃reflexivity⦄

  $\Rightarrow x \sqsubseteq \gamma(\alpha(x))$ ⦃def. GC⦄

  $\Rightarrow \alpha(x) \preccurlyeq \alpha(\gamma(\alpha(x)))$ ⦃α increasing⦄

  — $\gamma(y) \sqsubseteq \gamma(y)$ ⦃reflexivity⦄

  $\Rightarrow \alpha(\gamma(y)) \preccurlyeq y$ ⦃def. GC⦄

  $\Rightarrow \alpha(\gamma(\alpha(x))) \preccurlyeq \alpha(x)$ ⦃for $y = \alpha(x)$⦄

  — $\alpha(x) = \alpha(\gamma(\alpha(x)))$ ⦃antisymmetry⦄ □

- The dual is $\gamma \circ \alpha \circ \gamma = \gamma$.

Equivalent definition of Galois connections

- $\langle C, \sqsubseteq\rangle \xrightleftharpoons[\alpha]{\gamma} \langle \mathcal{A}, \preccurlyeq\rangle$ if and only if $\alpha \in C \to \mathcal{A}$ and $\gamma \in \mathcal{A} \to C$ satisfy

  (1) α is increasing;
  (2) γ is increasing;
  (3) $\forall x \in C \mathrel{.} x \sqsubseteq \gamma \circ \alpha(x)$ (*i.e.* $\gamma \circ \alpha$ is extensive)
  (4) $\forall y \in \mathcal{A} \mathrel{.} \alpha \circ \gamma(y) \preccurlyeq y$ (*i.e.* $\alpha \circ \gamma$ is reductive) □

α preserves existing lubs

Lemma 3 If $\langle C, \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\longleftrightarrows}} \langle \mathcal{A}, \preccurlyeq\rangle$ then α preserves lubs that may exist in C*i.e.* let $\sqcup$ be the partially defined lub for $\sqsubseteq$ in C and $\curlyvee$ be the partially defined lub for $\preccurlyeq$ in $\mathcal{A}$. Let $S \in \wp(C)$ be any subset of C. If $\bigsqcup S$ exists in C then the upper bound $\bigcurlyvee\{\alpha(e) \mid e \in S\}$ exists in C and is equal to $\alpha(\bigsqcup S)$. □

Proof By existence and definition of the lub $\bigsqcup S$, we have $\forall e \in S \mathrel{.} e \sqsubseteq \bigsqcup S$ so $\alpha(e) \preccurlyeq \alpha(\bigsqcup S)$ since α is increasing. It follows that $\alpha(\bigsqcup S)$ is an upper bound of $\{\alpha(e) \mid e \in S\}$. Let u be any upper bound of this set $\{\alpha(e) \mid e \in S\}$ so that $\forall e \in S \mathrel{.} \alpha(e) \preccurlyeq u$. By definition of a GC, $\forall e \in S \mathrel{.} e \sqsubseteq \gamma(u)$. So $\gamma(u)$ is an upper bound of S. By existence and definition of the lub $\bigsqcup S$, $\bigsqcup S \sqsubseteq \gamma(u)$ so $\alpha(\bigsqcup S) \preccurlyeq u$ proving that $\alpha(\bigsqcup S)$, which exists since α is a total function, is the lub of $\{\alpha(e) \mid e \in S\}$ denoted $\bigcurlyvee\{\alpha(e) \mid e \in S\}$. □

- By duality γ preserves existing meets.

lub-preserving α

Lemma 4 If α preserves existing lubs and $\gamma(y) \triangleq \bigsqcup\{x \in C \mid \alpha(x) \preccurlyeq y\}$ is well-defined then $\langle C, \sqsubseteq\rangle \mathrel{\underset{\alpha}{\overset{\gamma}{\leftrightarrows}}} \langle \mathcal{A}, \preccurlyeq\rangle$. □

Proof $x \sqsubseteq \gamma(y)$

$\Rightarrow x \sqsubseteq \bigsqcup\{x' \in C \mid \alpha(x') \preccurlyeq y\}$ $\quad \wr\text{def. } \gamma\int$

$\Rightarrow \alpha(x) \preccurlyeq \alpha(\bigsqcup\{x' \in C \mid \alpha(x') \preccurlyeq y\})$ $\quad \wr\alpha \text{ preserves existing lubs so is increasing}\int$

$\Rightarrow \alpha(x) \preccurlyeq \bigvee\{\alpha(x') \mid x' \in C \wedge \alpha(x') \preccurlyeq y\})$ $\quad \wr\alpha \text{ preserves existing lubs}\int$

$\Rightarrow \alpha(x) \preccurlyeq y$

$\wr$since y is an upper bound of $\{\alpha(x') \mid \alpha(x') \preccurlyeq y\}$ greater than or equal to the lub $\bigvee\{\alpha(x') \mid \alpha(x') \preccurlyeq y\}\int$

$\Rightarrow x \preccurlyeq \bigsqcup\{x' \in C \mid \alpha(x') \preccurlyeq y\}$ $\quad \wr\text{since } x \in \{x' \in C \mid \alpha(x') \preccurlyeq y\}\int$

$\Rightarrow x \preccurlyeq \gamma(y)$ $\quad \wr\text{def. } \gamma\int$ □

Uniqueness of adjoints

Lemma 5 In a Galois connection one adjoint uniquely determines the other. □

Proof Observe that $\forall P \in C \,.\, \alpha(P) = \sqcap\{\overline{P} \mid \alpha(P) \preccurlyeq \overline{P}\}$ so, by definition of a GC, $\alpha(P) = \sqcap\{\overline{P} \mid P \sqsubseteq \gamma(\overline{P})\}$ *i.e.* γ uniquely determines α. Dually α uniquely determines γ since $\forall \overline{P} \in \mathcal{A} \,.\, \gamma(\overline{P}) = \sqcup\{P \mid \alpha(P) \preccurlyeq \overline{P}\}$. □

- This lemma is useful in situations where only one adjoint is defined explicitly since then the other is also uniquely determined.
- Note: for given concrete and abstract partial orders

Galois retraction (surjection/insertion)

- If $\langle \mathcal{C}, \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \preccurlyeq\rangle$ then
  - α is surjective, if and only if
  - γ is injective, if and only if
  - $\forall \overline{P} \in \mathcal{A} \mathrel{.} \alpha \circ \gamma(\overline{P}) = \overline{P}$.
- This is denoted $\langle \mathcal{C}, \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\longleftarrow\atop\twoheadrightarrow}} \langle \mathcal{A}, \preccurlyeq\rangle$ and called a Galois retraction (Galois surjection, insertion, *etc*.).

Abstraction

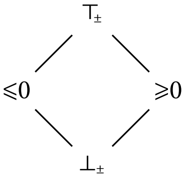
Sound abstraction

- Assume $\langle \mathcal{C}, \sqsubseteq \rangle \xleftrightarrow[\alpha]{\gamma} \langle \mathcal{A}, \preccurlyeq \rangle$
- We say that $\overline{P} \in \mathcal{A}$ is a *sound abstraction* of $P \in \mathcal{C}$ if and only if

$$P \sqsubseteq \gamma(\overline{P})$$

Examples of sound abstractions

$\langle \overline{\mathbb{P}^{\pm}}, \sqsubseteq^{\pm} \rangle =$



$$
\begin{aligned}
\gamma_{\pm}(\bot_{\pm}) &\triangleq \varnothing \\
\gamma_{\pm}(\leqslant 0) &\triangleq \{z \mid z \leqslant 0\} \\
\gamma_{\pm}(\geqslant 0) &\triangleq \{z \mid z \geqslant 0\} \\
\gamma_{\pm}(\top_{\pm}) &\triangleq \mathbb{Z}
\end{aligned}
$$

property	sound abstractions
$\{1, 42\}$	$\geqslant 0$ and $\top_{\pm}$
$\{0\}$	$\leqslant 0$, $\geqslant 0$, and $\top_{\pm}$

Better abstraction

- Assume $\langle C, \sqsubseteq\rangle \xleftarrow[\alpha]{\gamma}\!\!\!\!\longrightarrow \langle \mathcal{A}, \preceq\rangle$
- Let $\overline{P}_1, \overline{P}_2 \in \mathcal{A}$ be sound abstractions of the concrete property $P \in C$.
- We say that $\overline{P}_1$ is better/more precise/stronger/less abstract than $\overline{P}_2$ if and only if $\overline{P}_1 \preceq \overline{P}_2$.

Best abstraction

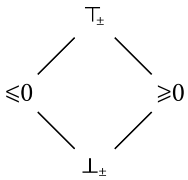
- Assume $\langle C, \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \preccurlyeq\rangle$
- Then $\alpha(P)$ is the best/most precise/strongest/least abstract property which is a sound abstraction of the concrete property P.

Proof

- $\alpha(P)$ is a sound abstraction of P since $P \sqsubseteq \gamma(\alpha(P))$.
- $\alpha(P)$ is the least sound abstraction of P since $\alpha(P) = \bigsqcap\{\overline{P} \mid P \sqsubseteq \gamma(\overline{P})\}$. □

Examples of best abstractions

$\langle\overline{\mathbb{P}^{\pm}},\ \sqsubseteq^{\pm}\rangle =$



$$\begin{array}{rcl}
\gamma_{\pm}(\bot_{\pm}) & \triangleq & \varnothing \\
\gamma_{\pm}(\leqslant 0) & \triangleq & \{z \mid z \leqslant 0\} \\
\gamma_{\pm}(\geqslant 0) & \triangleq & \{z \mid z \geqslant 0\} \\
\gamma_{\pm}(\top_{\pm}) & \triangleq & \mathbb{Z}
\end{array}$$

property	sound abstractions	best abstraction
$\{1, 42\}$	$\geqslant 0$ and $\top_{\pm}$	$\geqslant 0$
$\{0\}$	$\leqslant 0$, $\geqslant 0$, and $\top_{\pm}$	none

- There is no Galois connection between $\langle\wp(\mathbb{Z}),\ \subseteq\rangle$ and $\langle\overline{\mathbb{P}^{\pm}},\ \sqsubseteq^{\pm}\rangle$.

Combination of Galois connections

Composition of Galois connections

- The composition of Galois connections $\langle \mathcal{P}_1,\ \sqsubseteq\rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle \mathcal{P}_2,\ \preccurlyeq\rangle$ and $\langle \mathcal{P}_2,\ \preccurlyeq\rangle \underset{\alpha^2}{\overset{\gamma^2}{\leftrightarrows}} \langle \mathcal{P}_3,\ \trianglelefteq\rangle$ is the Galois connection $\langle \mathcal{P}_1,\ \sqsubseteq\rangle \underset{\alpha^2\circ\alpha_1}{\overset{\gamma_1\circ\gamma^2}{\leftrightarrows}} \langle \mathcal{P}_3,\ \trianglelefteq\rangle$.

Galois connections pairs

- Let $\langle C_1,\ \sqsubseteq_1\rangle \xleftarrow[\alpha_1]{\gamma_1} \xrightarrow{} \langle \mathcal{A}_1,\ \preccurlyeq_1\rangle$ and $\langle C_2,\ \sqsubseteq_2\rangle \xleftarrow[\alpha_2]{\gamma_2} \xrightarrow{} \langle \mathcal{A},\ \preccurlyeq_2\rangle$;
- $\langle C_1 \times C_2,\ \dot{\sqsubseteq}\rangle \xleftarrow[\alpha]{\gamma} \xrightarrow{} \langle \mathcal{A}_1 \times \mathcal{A}_2,\ \dot{\preccurlyeq}\rangle$, where
- $\alpha(\langle x,\ y\rangle) = \langle \alpha_1(x),\ \alpha_2(y)\rangle$,
- $\gamma(\langle \overline{x},\ \overline{y}\rangle) = \langle \gamma_1(\overline{x}),\ \gamma_2(\overline{y})\rangle$, and
- $\dot{\sqsubseteq}$ and $\dot{\preccurlyeq}$ are componentwise.

Higher-order Galois connections

- Let $\langle C_1, \sqsubseteq_1\rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle \mathcal{A}_1, \preccurlyeq_1\rangle$ and $\langle C_2, \sqsubseteq_2\rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle \mathcal{A}, \preccurlyeq_2\rangle$;
- $\langle C_1 \xrightarrow{\nearrow} C_2, \dot{\sqsubseteq}_2\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}_1 \xrightarrow{\nearrow} \mathcal{A}_2, \dot{\preccurlyeq}_2\rangle$, where
- $\alpha = \lambda f \centerdot \alpha_2 \circ f \circ \gamma_1$, and
- $\gamma = \lambda \overline{f} \centerdot \gamma_2 \circ \overline{f} \circ \alpha_1$.

$$\begin{tikzcd}
\mathcal{A}_1 \arrow[r, "\overline{f}"] \arrow[d, bend right, "\gamma_1"'] & \mathcal{A}_2 \arrow[d, bend right, "\gamma_2"'] \\
C_1 \arrow[r, "f"] \arrow[u, bend right, "\alpha_1"'] & C_2 \arrow[u, bend right, "\alpha_2"']
\end{tikzcd}$$

Conclusion on abstraction by Galois connections

- We can represent abstract program properties by posets and establish the correspondence with the concrete properties using a Galois connection.
- The concrete order structure is preserved in the abstract and inversely.
- Otherwise stated concrete and abstract implications coincide up to the Galois connection.
- So proofs in the abstract domain $\langle \mathcal{A}, \preccurlyeq \rangle$ using the abstract implication/order $\preccurlyeq$ is valid in the concrete $\langle \mathcal{C}, \sqsubseteq \rangle$ for $\sqsubseteq$, up to this GC.

Bibliography on abstraction

References I

Bertrane, Julien, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, and Xavier Rival (2015). "Static Analysis and Verification of Aerospace Software by Abstract Interpretation". *Foundations and Trends in Programming Languages* 2.2-3, pp. 71–190.

Blanchet, Bruno, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, and Xavier Rival (2003). "A static analyzer for large safety-critical software". In: *PLDI*. ACM, pp. 196–207.

Cousot, Patrick (Mar. 1978). "Méthodes itératives de construction et d'approximation de points fixes d'opérateurs monotones sur un treillis, analyse sémantique de programmes (in French)". Thèse d'État ès sciences mathématiques. Grenoble, France: Université de Grenoble Alpes.

– (1981). "Semantic foundations of program analysis". In: S.S. Muchnick and N.D. Jones, eds. *Program Flow Analysis: Theory and Applications*. Englewood Cliffs, New Jersey, USA: Prentice-Hall, Inc. Chap. 10, pp. 303–342.

References II

Cousot, Patrick (1997). “Types as Abstract Interpretations”. In: *POPL*. ACM Press, pp. 316–331.

– (1999). “The Calculational Design of a Generic Abstract Interpreter”. In: M. Broy and R. Steinbrüggen, eds. *Calculational System Design*. NATO ASI Series F. IOS Press, Amsterdam.

– (2000). “Partial Completeness of Abstract Fixpoint Checking”. In: *SARA*. Vol. 1864. Lecture Notes in Computer Science. Springer, pp. 1–25.

– (2015). “Abstracting Induction by Extrapolation and Interpolation”. In: *VMCAI*. Vol. 8931. Lecture Notes in Computer Science. Springer, pp. 19–42.

Cousot, Patrick and Radhia Cousot (1976). “Static determination of dynamic properties of programs”. In: *Proceedings of the Second International Symposium on Programming*. Dunod, Paris, France, pp. 106–130.

– (1977a). “Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints”. In: *POPL*. ACM, pp. 238–252.

References III

Cousot, Patrick and Radhia Cousot (1977b). “Static Determination of Dynamic Properties of Generalized Type Unions”. In: *Language Design for Reliable Software*, pp. 77–94.

– (1977c). “Static determination of dynamic properties of recursive procedures”. In: E.J. Neuhold, ed. *IFIP Conf. on Formal Description of Programming Concepts, St-Andrews, N.B., CA*. North-Holland Pub. Co., pp. 237–277.

– (1979). “Systematic Design of Program Analysis Frameworks”. In: *POPL*. ACM Press, pp. 269–282.

– (1992a). “Abstract Interpretation Frameworks”. *J. Log. Comput.* 2.4, pp. 511–547.

– (1992b). “Comparing the Galois Connection and Widening/Narrowing Approaches to Abstract Interpretation”. In: *PLILP*. Vol. 631. Lecture Notes in Computer Science. Springer, pp. 269–295.

– (1994). “Higher Order Abstract Interpretation (and Application to Comportment Analysis Generalizing Strictness, Termination, Projection, and PER Analysis”. In: *ICCL*. IEEE Computer Society, pp. 95–112.

References IV

Cousot, Patrick and Radhia Cousot (1995). “Formal Language, Grammar and Set-Constraint-Based Program Analysis by Abstract Interpretation”. In: *FPCA*. ACM, pp. 170–181.

– (2000). “Temporal Abstract Interpretation”. In: *POPL*. ACM, pp. 12–25.

– (2002). “Modular Static Program Analysis”. In: *CC*. Vol. 2304. Lecture Notes in Computer Science. Springer, pp. 159–178.

– (2004). “Basic Concepts of Abstract Interpretation”. In: René Jacquard, ed. *Building the Information Society*. Springer, pp. 359–366.

– (2012). “An abstract interpretation framework for termination”. In: *POPL*. ACM, pp. 245–258.

– (2014). “A Galois connection calculus for abstract interpretation”. In: *POPL*. ACM, pp. 3–4.

Cousot, Patrick, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, and Xavier Rival (2005). “The Astrée Analyzer”. In: *ESOP*. Vol. 3444. Lecture Notes in Computer Science. Springer, pp. 21–30.

References V

Cousot, Patrick, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, and Xavier Rival (2006). "Combination of Abstractions in the Astrée Static Analyzer". In: *ASIAN*. Vol. 4435. Lecture Notes in Computer Science. Springer, pp. 272–300.

Cousot, Patrick, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, and Xavier Rival (2009). "Why does Astrée scale up?" *Formal Methods in System Design* 35.3, pp. 229–264.

Cousot, Patrick, Roberto Giacobazzi, and Francesco Ranzato (2018). "Program Analysis Is Harder Than Verification: A Computability Perspective". In: *CAV (2)*. Vol. 10982. Lecture Notes in Computer Science. Springer, pp. 75–95.

– (Jan. 2019). "A^{2}I: Abstract^2 Interpretation". *PACMPL (POPL conference)* 3, article 42. DOI: 10.1145/3290355.

Cousot, Patrick and Nicolas Halbwachs (1978). "Automatic Discovery of Linear Restraints Among Variables of a Program". In: *POPL*. ACM Press, pp. 84–96.

The End of Part 2, 30mn break

Part 3

Verification and proofs

Verification and proofs

- We show that verification methods and program logics are (non-computable) abstractions of the program collecting semantics.

Program properties

Program semantic properties

- The entities are semantics of program `P` *i.e.* sets of maximal traces $\mathfrak{S} = \wp(\mathbb{T}^{+\infty})$
- The properties are sets of semantics of program `P` *i.e.* sets of sets of maximal traces $\wp(\mathfrak{S}) = \wp(\wp(\mathbb{T}^{+\infty}))$[2]

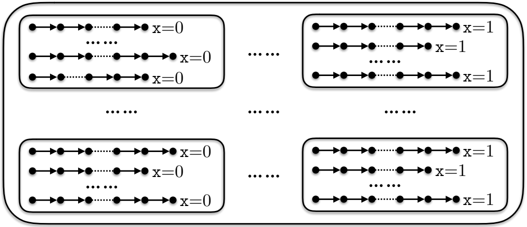
[2] also called “hyperproperties”

Example of program semantic property

$$P \triangleq \wp(\{\pi \in \mathbb{T}^{+} \mid \boldsymbol{\rho}(\pi)\mathtt{x} = 0\}) \cup \wp(\{\pi \in \mathbb{T}^{+} \mid \boldsymbol{\rho}(\pi)\mathtt{x} = 1\}) \in \wp(\wp(\mathbb{T}^{+\infty}))$$

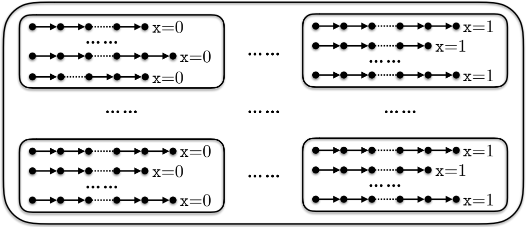
- P means “all executions of **P** always terminate with $\mathtt{x} = 0$ or all executions of **P** always terminate with $\mathtt{x} = 1$”.

$P \quad =$



Example of program semantic property (Cont'd)

P =



- Assume program P has this property P so $\mathcal{S}^{+\infty}[\![\mathsf{P}]\!] \in P$.
- Executing program P once, we know the result of all other executions.
- If the execution terminates with $\mathtt{x} = 0$ (respectively $\mathtt{x} = 1$) the property P implies that all other possible executions will always terminate with $\mathtt{x} = 0$ (respectively $\mathtt{x} = 1$).

Collecting semantics

Collecting semantics (for maximal traces)

- The strongest semantic property of program P

$$\mathcal{S}^{\mathbb{C}}[\![\mathsf{P}]\!] \quad \triangleq \quad \{\mathcal{S}^{+\infty}[\![\mathsf{P}]\!]\}\ . \tag{8.5}$$

- Program P has property $P \in \wp(\wp(\mathbb{T}^{+\infty}))$ is
  - $\mathcal{S}^{+\infty}[\![\mathsf{P}]\!] \in P$, or equivalently
  - $\{\mathcal{S}^{+\infty}[\![\mathsf{P}]\!]\} \subseteq P$ *i.e.* P is implied by the collecting semantics of program P.

- So we can use implication $\subseteq$ ($\Rightarrow$) instead of $\in$ (with no direct equivalent for predicates in logic).

Trace properties

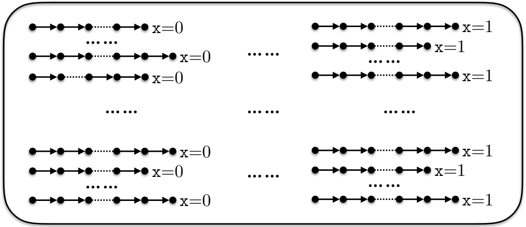
Trace properties

- By "program property" or "semantic property" most computer scientists refer to "trace properties"
- elements $\mathfrak{E} = \mathbb{T}^{+\infty}$, traces
- trace properties $\wp(\mathfrak{E}) = \wp(\mathbb{T}^{+\infty})$
- *safety* and *liveness* are trace properties

Example of trace properties

- the program trace semantics $\mathcal{S}^{+\infty}[\![\mathsf{P}]\!] \in \wp(\mathbb{T}^{+\infty})$ is a trace property.
- $\{\pi \in \mathbb{T}^{+} \mid \boldsymbol{\rho}(\pi)\mathsf{x} = 0\} \in \wp(\mathbb{T}^{+\infty})$ is the trace property of “terminating with x=0”.
- $\overline{P} = \{\pi \in \mathbb{T}^{+} \mid \boldsymbol{\rho}(\pi)\mathsf{x} \in \{0, 1\}\} \in \wp(\mathbb{T}^{+\infty})$ is the trace property of “terminating with x=0 or x=1”.

$\overline{P}$ =



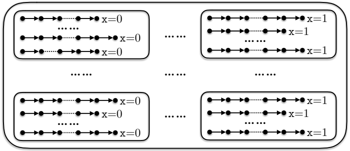
- Trace properties in $\wp(\mathbb{T}^{+\infty})$ are less expressive than semantic properties in $\wp(\wp(\mathbb{T}^{+\infty}))$

Abstraction of a semantic property into a trace property

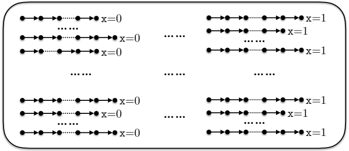
- Any semantic property P can be abstracted into a less precise trace property $\alpha^{\mathbb{T}}(P)$ defined as

$$\begin{array}{rcl} \alpha^{\mathbb{T}} & \in & \wp(\wp(\mathbb{T}^{+\infty})) \rightarrow \wp(\mathbb{T}^{+\infty}) \\ \alpha^{\mathbb{T}}(P) & = & \bigcup P \end{array} \qquad \begin{array}{rcl} \gamma^{\mathbb{T}} & \in & \wp(\mathbb{T}^{+\infty}) \rightarrow \wp(\wp(\mathbb{T}^{+\infty})) \\ \gamma^{\mathbb{T}}(\overline{P}) & = & \wp(\overline{P}) \end{array}$$

$P =$



$\overline{P}=\alpha^{\mathbb{T}}(P) =$



- P and $\overline{P}$ both express that program executions always terminate with a boolean value for x.
- P is stronger since it expresses that the result is always the same while $\overline{P}$ doesn't.

Abstraction of a semantic property into a trace property (Cont'd)

- Galois connection $\langle\wp(\wp(\mathbb{T}^{+\infty})),\ \subseteq\rangle \underset{\alpha^{\mathbb{T}}}{\overset{\gamma^{\mathbb{T}}}{\leftrightarrows}} \langle\wp(\mathbb{T}^{+\infty}),\ \subseteq\rangle$
- Proof:

$$
\begin{aligned}
& \alpha^{\mathbb{T}}(P) \subseteq \overline{P} \\
\Leftrightarrow\ & \bigcup P \subseteq \overline{P} && \text{(def. } \alpha^{\mathbb{T}}\text{)} \\
\Leftrightarrow\ & \{x \mid \exists X \in P \,.\, x \in X\} \subseteq \overline{P} && \text{(def. } \textstyle\bigcup\text{)} \\
\Leftrightarrow\ & \forall X \in P \,.\, \forall x \in X \,.\, x \in \overline{P} && \text{(def. } \subseteq\text{)} \\
\Leftrightarrow\ & \forall X \in P \,.\, X \subseteq \overline{P} && \text{(def. } \subseteq\text{)} \\
\Leftrightarrow\ & P \subseteq \{X \mid X \subseteq \overline{P}\} && \text{(def. } \subseteq\text{)} \\
\Leftrightarrow\ & P \subseteq \wp(\overline{P}) && \text{(def. } \wp\text{)} \\
\Leftrightarrow\ & P \subseteq \gamma^{\mathbb{T}}(\overline{P}) && \text{(def. } \gamma^{\mathbb{T}}.\text{)}
\end{aligned}
$$

- $\alpha^{\mathbb{T}}$ is surjective (since $\alpha^{\mathbb{T}}(\{\overline{P}\}) = \overline{P}$).

Reachability properties

Reachability property

A relation $\mathcal{I}(\ell)$ between values of variables attached to each program point ℓ that holds whenever the program point ℓ is reached during execution

```
$\ell_1$   /* $x = 0$ */
           x = x + 1 ;
           while $\ell_2$ (tt)  /* $1 \leqslant x \leqslant 2$ */ {
$\ell_3$          /* $1 \leqslant x \leqslant 2$ */
                  x = x + 1 ;
                  if $\ell_4$ (x > 2)  /* $2 \leqslant x \leqslant 3$ */
$\ell_5$                 /* $x = 3$ */
                         break ;
           }
$\ell_6$   /* $x = 3$ */
           ;
$\ell_7$   /* $x = 3$ */
```

$$\mathcal{I}(\ell_1) \triangleq \{\rho \in \mathbb{E}\mathrm{v} \mid \forall \mathrm{y} \in \mathbb{V} \,.\, \rho(\mathrm{y}) = 0\}$$

$$\mathcal{I}(\ell_2) \triangleq \mathcal{I}(\ell_3) \triangleq \{\rho \in \mathbb{E}\mathrm{v} \mid 1 \leqslant \rho(\mathrm{x}) \leqslant 2 \wedge \forall \mathrm{y} \in \mathbb{V} \setminus \{\mathrm{x}\} \,.\, \rho(\mathrm{y}) = 0\}$$

$$\mathcal{I}(\ell_4) \triangleq \{\rho \in \mathbb{E}\mathrm{v} \mid 2 \leqslant \rho(\mathrm{x}) \leqslant 3 \wedge \forall \mathrm{y} \in \mathbb{V} \setminus \{\mathrm{x}\} \,.\, \rho(\mathrm{y}) = 0\}$$

$$\mathcal{I}(\ell_5) \triangleq \mathcal{I}(\ell_6) \triangleq \mathcal{I}(\ell_7) \triangleq \{\rho \in \mathbb{E}\mathrm{v} \mid \rho(\mathrm{x}) = 3 \wedge \forall \mathrm{y} \in \mathbb{V} \setminus \{\mathrm{x}\} \,.\, \rho(\mathrm{y}) = 0\}$$

Abstraction of a trace property into a reachability property

$$\begin{array}{rcl} \alpha^{\square} & \in & \wp(\mathbb{T}^{+\infty}) \rightarrow (\mathbb{L} \rightarrow \wp(\mathbb{E}\mathbb{v})) \\ \alpha^{\square}(\Pi) & \triangleq & \boldsymbol{\lambda}\, \ell \mathrel{\raise.3ex\hbox{$\scriptscriptstyle\blacksquare$}} \{\boldsymbol{\rho}(\pi\ell) \mid \exists \pi' \,.\, \pi\ell\pi' \in \Pi\} \end{array} \tag{8.12}$$

collects at each program point ℓ of each trace the possible values of the variables at that point.

Abstraction of a trace property into a reachability property (Cont'd)

- Galois connection $\langle\wp(\mathbb{T}^{+\infty}),\ \subseteq\rangle \underset{\alpha^{\square}}{\overset{\gamma^{\square}}{\leftrightarrows}} \langle(\mathbb{L} \rightarrow \wp(\mathbb{E}\mathbb{v})),\ \dot{\subseteq}\rangle$
- Proof:

$$\alpha^{\square}(\Pi) \mathrel{\dot{\subseteq}} \mathcal{I}$$

$$\Leftrightarrow \boldsymbol{\lambda}\,\ell \centerdot \{\boldsymbol{\rho}(\pi\ell) \mid \exists\pi' \,.\, \pi\ell\pi' \in \Pi\} \mathrel{\dot{\subseteq}} \mathcal{I} \qquad \text{(def. } \alpha^{\square}\text{)}$$

$$\Leftrightarrow \forall\ell \,.\, \{\boldsymbol{\rho}(\pi\ell) \mid \exists\pi' \,.\, \pi\ell\pi' \in \Pi\} \subseteq \mathcal{I}(\ell) \qquad \text{(pointwise def. } \dot{\subseteq}\text{)}$$

$$\Leftrightarrow \forall\ell \,.\, \{\boldsymbol{\rho}(\pi\ell) \mid \exists\overline{\pi} \in \Pi \,.\, \exists\pi' \,.\, \overline{\pi} = \pi\ell\pi'\} \subseteq \mathcal{I}(\ell) \qquad \text{(def. } \in\text{)}$$

$$\Leftrightarrow \forall\ell \,.\, \forall\overline{\pi} \in \Pi \,.\, \forall\pi' \,.\, \overline{\pi} = \pi\ell\pi' \Rightarrow \boldsymbol{\rho}(\pi\ell) \in \mathcal{I}(\ell) \qquad \text{(def. } \subseteq\text{)}$$

$$\Leftrightarrow \forall\overline{\pi} \in \Pi \,.\, \forall\pi' \,.\, \forall\ell \,.\, \overline{\pi} = \pi\ell\pi' \Rightarrow \boldsymbol{\rho}(\pi\ell) \in \mathcal{I}(\ell) \qquad \text{(def. } \forall\text{)}$$

$$\Leftrightarrow \Pi \subseteq \{\overline{\pi} \mid \forall\pi' \,.\, \forall\ell \,.\, \overline{\pi} = \pi\ell\pi' \Rightarrow \boldsymbol{\rho}(\pi\ell) \in \mathcal{I}(\ell)\} \qquad \text{(def. } \subseteq\text{)}$$

$$\Leftrightarrow \Pi \subseteq \gamma^{\square}(\mathcal{I})$$

by defining $\gamma^{\square}(\mathcal{I}) \triangleq \{\overline{\pi} \mid \forall\pi' \,.\, \forall\ell \,.\, \overline{\pi} = \pi\ell\pi' \Rightarrow \boldsymbol{\rho}(\pi\ell) \in \mathcal{I}(\ell)\}$.

Hierarchy of program properties

Hierarchy of program properties/semantics

$$\begin{tikzcd}
\bullet\ \mathbb{L} \rightarrow \wp(\mathbb{E}\mathbb{v}) \arrow[d, dash] \arrow[d, bend right, "\gamma^{\mathbb{I}}"'] \arrow[dd, bend right=60, "\gamma^{\mathbb{T}} \circ \gamma^{\mathbb{I}}"'] \\
\bullet\ \wp(\mathbb{T}^{+\infty}) \arrow[u, bend right, "\alpha^{\mathbb{I}}"'] \arrow[d, dash] \arrow[d, bend right, "\gamma^{\mathbb{T}}"'] \\
\bullet\ \wp(\wp(\mathbb{T}^{+\infty})) \arrow[u, bend right, "\alpha^{\mathbb{T}}"'] \arrow[uu, bend right=60, "\alpha^{\mathbb{I}} \circ \alpha^{\mathbb{T}}"']
\end{tikzcd}$$

$$\begin{aligned} \mathcal{S}^{\mathbb{I}}[\![\mathsf{P}]\!] &= \alpha^{\mathbb{I}}(\mathcal{S}^{\mathbb{T}}[\![\mathsf{P}]\!]) \\ &= \alpha^{\mathbb{I}} \circ \alpha^{\mathbb{T}}(\mathcal{S}^{\mathbb{C}}[\![\mathsf{P}]\!]) \end{aligned}$$

invariance/
reachability
semantics

$$\begin{aligned} \mathcal{S}^{\mathbb{T}}[\![\mathsf{P}]\!] &= \mathcal{S}^{+\infty}[\![\mathsf{P}]\!] \\ &= \alpha^{\mathbb{T}}(\mathcal{S}^{\mathbb{C}}[\![\mathsf{P}]\!]) \end{aligned}$$

trace semantics

$\mathcal{S}^{\mathbb{C}}[\![\mathsf{P}]\!] \triangleq \{\mathcal{S}^{+\infty}[\![\mathsf{P}]\!]\}$, collecting semantics

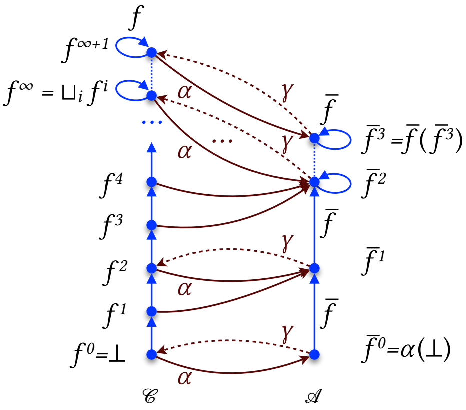
Chapter 16

Fixpoint abstraction

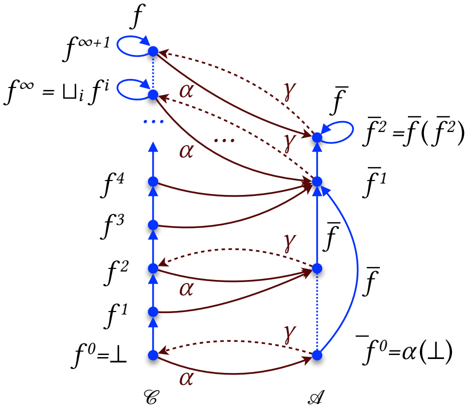
Fixpoint abstraction

- C is a concrete domain
- $f \in C \xrightarrow{\nearrow} C$ is an increasing concrete transformer
- $\langle C, \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \preccurlyeq\rangle$ is an abstraction into $\mathcal{A}$
- Problem: abstract $\mathsf{lfp}^{\sqsubseteq} f$
  - first abstract the concrete transformer f into an abstract transformer $\overline{f} \in \mathcal{A} \xrightarrow{\nearrow} \mathcal{A}$
  - then abstract $\alpha(\mathsf{lfp}^{\sqsubseteq} f)$ into $\mathsf{lfp}^{\preccurlyeq} \overline{f}$.
  - This abstraction may be
    - *exact* *i.e.* $\alpha(\mathsf{lfp}^{\sqsubseteq} f) = \mathsf{lfp}^{\preccurlyeq} \overline{f}$
    - or *sound* but imprecise, in which case we get an overapproximation $\alpha(\mathsf{lfp}^{\sqsubseteq} f) \preccurlyeq \mathsf{lfp}^{\preccurlyeq} \overline{f}$.

Example of fixpoint abstraction



exact fixpoint abstraction

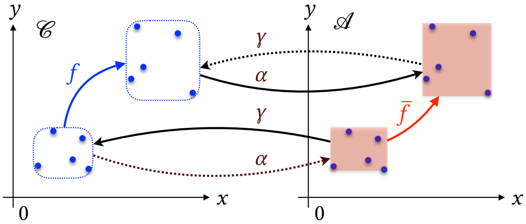


imprecise fixpoint abstraction

Transformer abstraction

Transformer abstraction

- To abstract a fixpoint $\alpha(\mathsf{lfp}^{\sqsubseteq} f)$, we first abstract its transformer f.

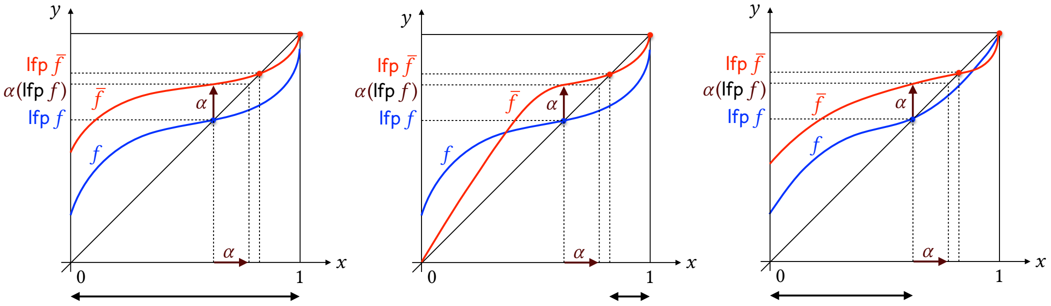


Theorem (16.1, transformer abstraction) If $\langle C, \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \preccurlyeq\rangle$ then $\langle C \xrightarrow{\nearrow} C, \dot{\sqsubseteq}\rangle \underset{\vec{\alpha}}{\overset{\vec{\gamma}}{\leftrightarrows}} \langle \mathcal{A} \xrightarrow{\nearrow} \mathcal{A}, \dot{\preccurlyeq}\rangle$ where $\dot{\sqsubseteq}$ and $\dot{\preccurlyeq}$ are pointwise (*i.e.* $f \mathrel{\dot{\sqsubseteq}} g$ if and only if $\forall x \in C \,.\, f(x) \sqsubseteq g(x)$), $\vec{\alpha}(f) = \alpha \circ f \circ \gamma$, and $\vec{\gamma}(\overline{f}) = \gamma \circ \overline{f} \circ \alpha$.

Fixpoint over-approximation

Fixpoint over-approximation

- In general abstracting the fixpoint transformer by a larger one yields a fixpoint over-approximation.



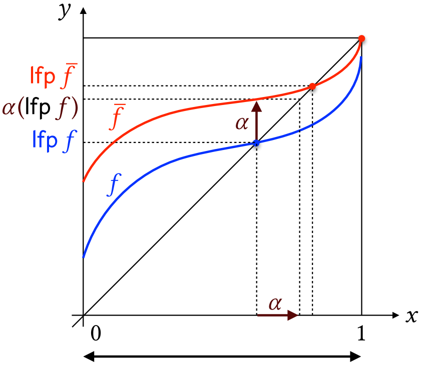
$f \mathrel{\dot{\sqsubseteq}} \overline{f}$ $\quad$ $\forall x \,.\, \overline{f}(x) \sqsubseteq x \Rightarrow f(x) \sqsubseteq x$ $\quad$ $\forall x \sqsubseteq \mathsf{lfp}^{\sqsubseteq} f \,.\, f(x) \sqsubseteq \overline{f}(x)$

fixpoint over-approximation

Fixpoint over-approximation (cont'd)

Theorem (16.3, pointwise fixpoint over-approximation) Assume that $\langle C, \sqsubseteq, \bot, \top, \sqcup, \sqcap\rangle$ is a complete lattice, $f, g \in C \xrightarrow{\nearrow} C$ are increasing, and $f \mathrel{\dot{\sqsubseteq}} g$ then $\mathsf{lfp}^{\sqsubseteq} f \sqsubseteq \mathsf{lfp}^{\sqsubseteq} g$.

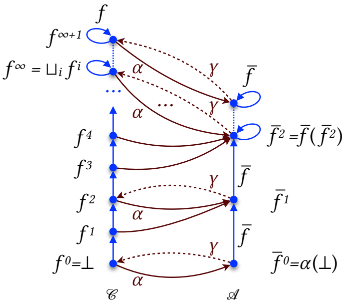
- Also valid for cpos.



Sound fixpoint abstraction

- An abstract fixpoint $\mathsf{lfp}^{\preccurlyeq}\,\overline{f}$ is a sound fixpoint abstraction of a concrete fixpoint $\mathsf{lfp}^{\sqsubseteq}\,f$ whenever $\alpha(\mathsf{lfp}^{\sqsubseteq}\,f) \preccurlyeq \mathsf{lfp}^{\preccurlyeq}\,\overline{f}$.

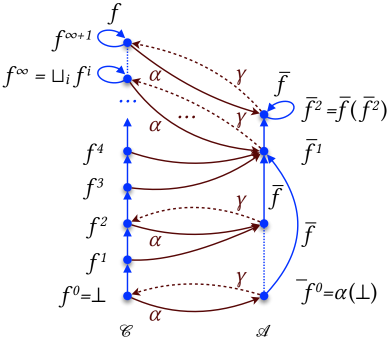
Theorem (16.6, fixpoint over-approximation in a complete lattice) Assume that $\langle \mathcal{C}, \sqsubseteq, \bot, \top, \sqcup, \sqcap\rangle$ and $\langle \mathcal{A}, \preccurlyeq, 0, 1, \curlyvee, \curlywedge\rangle$ are complete lattices, $\langle \mathcal{C}, \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \preccurlyeq\rangle$, and $f \in \mathcal{C} \overset{\nearrow}{\longrightarrow} \mathcal{C}$ is increasing. Then $\mathsf{lfp}^{\sqsubseteq}\,f \sqsubseteq \gamma(\mathsf{lfp}^{\preccurlyeq}\,\alpha \circ f \circ \gamma)$.



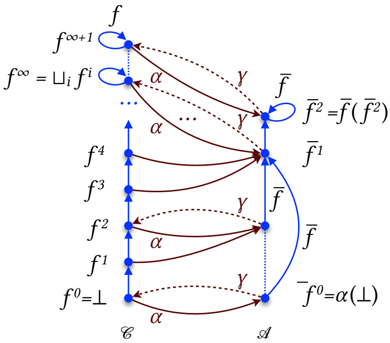
Sound fixpoint abstraction (cont'd)

Corollary (16.8, fixpoint approximation by transformer over-approximation) Assume that $\langle C, \sqsubseteq, \bot, \top, \sqcup, \sqcap\rangle$ and $\langle \mathcal{A}, \preccurlyeq, 0, 1, \curlyvee, \curlywedge\rangle$ are complete lattices, $\langle C, \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \preccurlyeq\rangle$, $f \in C \xrightarrow{\nearrow} C$ and $\overline{f} \in \mathcal{A} \xrightarrow{\nearrow} \mathcal{A}$ are increasing, and $\alpha \circ f \circ \gamma \mathrel{\dot{\preccurlyeq}} \overline{f}$. Then $\mathsf{lfp}^{\sqsubseteq} f \sqsubseteq \gamma(\mathsf{lfp}^{\preccurlyeq} \overline{f})$.

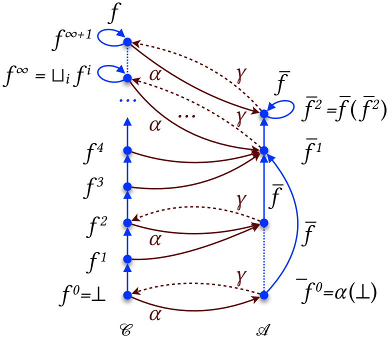
also in a cpo



Theorem (16.12, fixpoint over-approximation in a cpo) Assume that $\langle C, \sqsubseteq, \bot, \sqcup\rangle$ is a cpo and $\langle \mathcal{A}, \preccurlyeq, 0, \curlywedge\rangle$ are cpos, $\langle C, \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \preccurlyeq\rangle$, and $f \in C \xrightarrow{uc} C$ is upper continuous.
Then $\mathsf{lfp}^{\sqsubseteq} f \sqsubseteq \gamma(\mathsf{lfp}^{\preccurlyeq} \alpha \circ f \circ \gamma)$.



Corollary (16.10, fixpoint approximation by semi-commuting transformer) Under the hypotheses of Corollary 16.8 assume instead that $\alpha \circ f \mathrel{\dot{\preceq}} \overline{f} \circ \alpha$ (*semi-commutation*). Then $\mathsf{lfp}^{\sqsubseteq} f \sqsubseteq \gamma(\mathsf{lfp}^{\preceq} \overline{f})$.



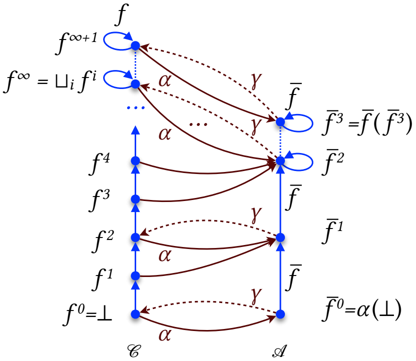
Exact fixpoint abstraction

Exact versus sound fixpoint abstraction

- A sound fixpoint abstraction $\alpha(\mathsf{lfp}^{\sqsubseteq} f) \preceq \mathsf{lfp}^{\preceq} \overline{f}$ is
  - *exact* when $\alpha(\mathsf{lfp}^{\sqsubseteq} f) = \mathsf{lfp}^{\preceq} \overline{f}$.
  - It is *sound but approximate (or imprecise)* when $\alpha(\mathsf{lfp}^{\sqsubseteq} f) \prec \mathsf{lfp}^{\preceq} \overline{f}$.

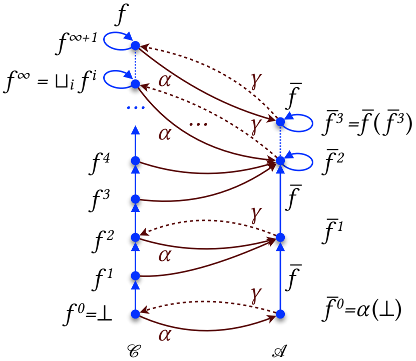
Exact fixpoint abstraction

Theorem (16.15, exact fixpoint abstraction in a complete lattice) Assume that $\langle C, \sqsubseteq, \bot, \top, \sqcup, \sqcap\rangle$ and $\langle \mathcal{A}, \preccurlyeq, 0, 1, \curlyvee, \curlywedge\rangle$ are complete lattices, $f \in C \xrightarrow{\nearrow} C$ is increasing, $\langle C, \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \preccurlyeq\rangle$, $\overline{f} \in \mathcal{A} \xrightarrow{\nearrow} \mathcal{A}$ is increasing, and $\alpha \circ f = \overline{f} \circ \alpha$ (commutation property). Then $\alpha(\mathsf{lfp}^{\sqsubseteq} f) = \mathsf{lfp}^{\preccurlyeq} \overline{f}$.



Exact fixpoint abstraction (cont'd)

Theorem (16.16, exact fixpoint abstraction in a cpo) Assume that $\langle \mathcal{C}, \sqsubseteq, \bot, \sqcup\rangle$ is a cpo, $f \in \mathcal{C} \xrightarrow{uc} \mathcal{C}$ is upper continuous, $\langle \mathcal{C}, \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \preccurlyeq\rangle$ is a Galois retraction, and $\overline{f} \in \mathcal{A} \to \mathcal{A}$ satisfies the commutation property $\alpha \circ f = \overline{f} \circ \alpha$. Then $\overline{f} = \alpha \circ f \circ \gamma$ is increasing and $\alpha(\mathsf{lfp}^{\sqsubseteq} f) = \mathsf{lfp}^{\preccurlyeq} \overline{f} = \bigvee_{n \in \mathbb{N}} \overline{f}^{\,n}(\alpha(\bot))$.

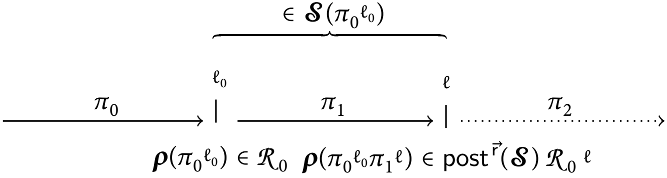


Reachability semantics

Reachability abstraction

Assertional abstraction

$$\text{post}^{\vec{r}}(\mathcal{S})\,\mathcal{R}_0\,\ell \triangleq \{\boldsymbol{\rho}(\pi_0\ell_0\pi_1\ell') \mid \boldsymbol{\rho}(\pi_0\ell_0) \in \mathcal{R}_0 \wedge \ell_0\pi_1\ell' \in \mathcal{S}(\pi_0\ell_0) \wedge \ell' = \ell\} \tag{18.1}$$



$$\overbrace{\ell_0 \xrightarrow{\pi_1} \ell}^{\in\, \mathcal{S}(\pi_0\ell_0)}$$

$\xrightarrow{\pi_0}$ ℓ_0 $\xrightarrow{\pi_1}$ ℓ $\overset{\pi_2}{\cdots\cdots\rightarrow}$

$\boldsymbol{\rho}(\pi_0\ell_0) \in \mathcal{R}_0$ $\quad$ $\boldsymbol{\rho}(\pi_0\ell_0\pi_1\ell) \in \text{post}^{\vec{r}}(\mathcal{S})\,\mathcal{R}_0\,\ell$

$$\langle \mathbb{T}^+ \rightarrow \wp(\mathbb{T}^+),\ \dot{\subseteq}\rangle \underset{\text{post}^{\vec{r}}}{\overset{\gamma^{\vec{r}}}{\leftrightarrows}} \langle \wp(\mathbb{E}\mathbb{v}) \rightarrow \mathbb{L} \mapsto \wp(\mathbb{E}\mathbb{v}),\ \dot{\subseteq}\rangle$$

Assertional abstraction, Example

$$\begin{array}{l} \ell_1\ \texttt{x = x + 1 ;} \\ \quad \texttt{while}\ \ell_2\ (\mathtt{tt})\ \{ \\ \qquad \ell_3\ \texttt{x = x + 1 ;} \\ \qquad \texttt{if}\ \ell_4\ (\texttt{x > 2})\ \ell_5\ \texttt{break ;}\}\ell_6\texttt{;}\ell_7 \end{array} \tag{4.4}$$

We assume that all variables are initialized to 0. Maximal trace semantics

$$\mathcal{S} \triangleq \{\ell_1 \xrightarrow{\mathtt{x} = 1} \ell_2 \xrightarrow{\mathtt{tt}} \ell_3 \xrightarrow{\mathtt{x} = 2} \ell_4 \xrightarrow{\neg(\mathtt{x} > 2)} \ell_2 \xrightarrow{\mathtt{tt}} \ell_3 \xrightarrow{\mathtt{x} = 3} \ell_4 \xrightarrow{\mathtt{x} > 2} \ell_5 \xrightarrow{\mathbf{break}} \ell_6 \xrightarrow{\mathtt{skip}} \ell_7\} \tag{6.1}$$

The reachable states are

ℓ	$\mathsf{post}^{\vec{\mathsf{r}}}(\mathcal{S})\,\mathcal{R}_0\ \ell$
ℓ_1	$\mathcal{R}_0 \quad = \quad \{\rho \in \mathbb{E}\mathbb{v} \mid \forall \mathtt{y} \in \mathbb{V} \,.\, \rho(\mathtt{y}) = 0\}$
ℓ_2, ℓ_3	$\{\rho[x \leftarrow i] \mid \rho \in \mathcal{R}_0 \wedge i \in [1, 2]\}$
ℓ_4	$\{\rho[x \leftarrow i] \mid \rho \in \mathcal{R}_0 \wedge i \in [2, 3]\}$
ℓ_5, ℓ_6, ℓ_7	$\{\rho[x \leftarrow 3] \mid \rho \in \mathcal{R}_0\}$

□

Calculational design of the reachability semantics

Calculational design of the reachability semantics

- by structural induction
- by calculating the exact reachability transformer from the prefix trace transformer
- by applying the exact fixpoint abstraction 16.15 for the iteration

Reachability semantics of the assignment

Reachability of an assignment statement `S ::= x = A ;`

$$
\begin{aligned}
\widehat{\mathcal{S}}^{\vec{\mathsf{r}}}[\![\mathsf{S}]\!]\,\mathcal{R}_0\,\ell \quad &= \quad (\!|\ \ell = \mathsf{at}[\![\mathsf{S}]\!] \ \text{?}\ \mathcal{R}_0 \\
&\qquad \|\ \ell = \mathsf{after}[\![\mathsf{S}]\!] \ \text{?}\ \mathsf{assign}^{\vec{\mathsf{r}}}[\![\mathsf{x}, \mathsf{A}]\!]\,\mathcal{R}_0 \\
&\qquad \text{:}\ \varnothing\ |\!) \qquad\qquad (17.10) \\
\mathsf{assign}^{\vec{\mathsf{r}}}[\![\mathsf{x}, \mathsf{A}]\!]\,\mathcal{R}_0 \quad &\triangleq \quad \{\rho[\mathsf{x} \leftarrow \boldsymbol{\mathcal{A}}[\![\mathsf{A}]\!]\rho] \mid \rho \in \mathcal{R}_0\}
\end{aligned}
$$

Reachability semantics of the conditional

Reachability of a conditional statement $\mathtt{S} ::= \mathtt{if\ (B)\ S}_t$

$$
\begin{aligned}
\widehat{\mathcal{S}}^{\vec{r}}[\![\mathsf{S}]\!]\,\mathcal{R}_0\,\ell \quad &= \quad (\!|\ \ell = \mathsf{at}[\![\mathsf{S}]\!] \mathrel{?} \mathcal{R}_0 \\
&\qquad \|\ \ell \in \mathsf{in}[\![\mathsf{S}_t]\!] \mathrel{?} \widehat{\mathcal{S}}^{\vec{r}}[\![\mathsf{S}_t]\!]\,(\mathsf{test}^{\vec{r}}[\![\mathsf{B}]\!]\mathcal{R}_0)\,\ell \\
&\qquad \|\ \ell = \mathsf{after}[\![\mathsf{S}]\!] \mathrel{?} \widehat{\mathcal{S}}^{\vec{r}}[\![\mathsf{S}_t]\!]\,(\mathsf{test}^{\vec{r}}[\![\mathsf{B}]\!]\mathcal{R}_0)\,\ell \cup (\overline{\mathsf{test}}^{\vec{r}}[\![\mathsf{B}]\!]\mathcal{R}_0) \\
&\qquad \mathrel{:} \varnothing\ |\!)
\end{aligned}
\tag{17.18}
$$

$$
\begin{aligned}
\mathsf{test}^{\vec{r}}[\![\mathsf{B}]\!]\mathcal{R}_0 \quad &\triangleq \quad \{\rho \in \mathcal{R}_0 \mid \mathcal{B}[\![\mathsf{B}]\!]\rho = \mathit{tt}\} \\
\overline{\mathsf{test}}^{\vec{r}}[\![\mathsf{B}]\!]\mathcal{R}_0 \quad &\triangleq \quad \{\rho \in \mathcal{R}_0 \mid \mathcal{B}[\![\mathsf{B}]\!]\rho = \mathit{ff}\}
\end{aligned}
$$

Reachability semantics of the statement list

Reachability of a statement list $\texttt{sl} ::= \texttt{sl}' \; \texttt{s}$

$$\begin{aligned}\widehat{\mathcal{S}}^{\,\vec{\mathsf{r}}}[\![\texttt{sl}]\!]\mathcal{R}_0\,\ell \quad = \quad & (\![\, \ell \in \mathsf{labs}[\![\texttt{sl}']\!] \setminus \{\mathsf{at}[\![\texttt{s}]\!]\} \mathrel{?} \widehat{\mathcal{S}}^{\,\vec{\mathsf{r}}}[\![\texttt{sl}']\!]\mathcal{R}_0\,\ell \\ & \,[\!]\; \ell \in \mathsf{labs}[\![\texttt{s}]\!] \mathrel{?} \widehat{\mathcal{S}}^{\,\vec{\mathsf{r}}}[\![\texttt{s}]\!](\widehat{\mathcal{S}}^{\,\vec{\mathsf{r}}}[\![\texttt{sl}']\!]\mathcal{R}_0\ \mathsf{at}[\![\texttt{s}]\!])\,\ell \\ & \mathrel{:} \varnothing \,]\!)\end{aligned} \tag{17.20}$$

Reachability semantics of the iteration

Reachability of an iteration statement $\mathtt{S} ::= \mathtt{while}\ \ell\ (\mathtt{B})\ \mathtt{S}_b$

$$\widehat{\mathcal{S}}^{\vec{r}}[\![\mathtt{S}]\!]\,\mathcal{R}_0\,\ell' \quad = \quad (\mathsf{lfp}^{\dot{\subseteq}}\,\mathcal{F}^{\vec{r}}[\![\mathtt{while}\ \ell\ (\mathtt{B})\ \mathtt{S}_b]\!]\,\mathcal{R}_0)\,\ell' \tag{17.14}$$

$$\begin{aligned}
&\mathcal{F}^{\vec{r}}[\![\mathtt{while}\ \ell\ (\mathtt{B})\ \mathtt{S}_b]\!]\,\mathcal{R}_0\,X\,\ell' \quad = \\
&\quad \llparenthesis\ \ell' = \ell \mathrel{?} \mathcal{R}_0 \cup \widehat{\mathcal{S}}^{\vec{r}}[\![\mathtt{S}_b]\!]\,(\mathsf{test}^{\vec{r}}[\![\mathtt{B}]\!]X(\ell))\,\ell \\
&\quad \parallel\ \ell' \in \mathsf{in}[\![\mathtt{S}_b]\!] \setminus \{\ell\} \mathrel{?} \widehat{\mathcal{S}}^{\vec{r}}[\![\mathtt{S}_b]\!]\,(\mathsf{test}^{\vec{r}}[\![\mathtt{B}]\!]X(\ell))\,\ell' \\
&\quad \parallel\ \ell' = \mathsf{after}[\![\mathtt{S}]\!] \mathrel{?} \overline{\mathsf{test}}^{\vec{r}}[\![\mathtt{B}]\!](X(\ell)) \cup \bigcup_{\ell'' \in \mathsf{breaks\text{-}of}[\![\mathtt{S}_b]\!]} \widehat{\mathcal{S}}^{\vec{r}}[\![\mathtt{S}_b]\!]\,(\mathsf{test}^{\vec{r}}[\![\mathtt{B}]\!]X(\ell))\,\ell'' \\
&\quad \mathbin{:} \varnothing\ \rrparenthesis
\end{aligned}$$

Abstract domain and abstract interpreter

Abstract domain

The domain of properties, inclusion (*i.e.* logical implication), and the structural definitions of the semantics have the following common structure.

semantics	prefix trace $\widehat{\mathcal{S}}^{\dot{*}}$	reachability $\widehat{\mathcal{S}}^{\vec{r}}$	abstract $\widehat{\mathcal{S}}^{¤}$
	$\wp(\mathbb{T}^+) \xrightarrow{\nearrow} (\mathbb{L} \rightarrow \wp(\mathbb{T}^+))$	$\wp(\mathbb{Ev}) \xrightarrow{\nearrow} (\mathbb{L} \rightarrow \wp(\mathbb{Ev}))$	$\mathbb{P}^{¤} \xrightarrow{\nearrow} (\mathbb{L} \rightarrow \mathbb{P}^{¤})$
domain	$\wp(\mathbb{T}^+)$	$\wp(\mathbb{Ev})$	$\mathbb{P}^{¤}$
inclusion	$\subseteq$	$\subseteq$	$\sqsubseteq^{¤}$
abstraction	$\mathbb{1}_{\wp(\mathbb{T}^+)}$ [3]	$\ddot{\alpha}_{\rho}$	$\alpha_{¤}$
infimum	$\emptyset$	$\emptyset$	$\bot^{¤}$
join	$\cup$	$\cup$	$\sqcup^{¤}$
assignment	$\mathsf{assign}^{\dot{*}}[\![\mathsf{x}, \mathsf{A}]\!]$	$\mathsf{assign}^{\vec{r}}[\![\mathsf{x}, \mathsf{A}]\!]$	$\mathsf{assign}^{¤}[\![\mathsf{x}, \mathsf{A}]\!]$
test	$\mathsf{test}^{\dot{*}}[\![\mathsf{B}]\!]$	$\mathsf{test}^{\vec{r}}[\![\mathsf{B}]\!]$	$\mathsf{test}^{¤}[\![\mathsf{B}]\!]$
	$\overline{\mathsf{test}}^{\dot{*}}[\![\mathsf{B}]\!]$	$\overline{\mathsf{test}}^{\vec{r}}[\![\mathsf{B}]\!]$	$\overline{\mathsf{test}}^{¤}[\![\mathsf{B}]\!]$

[3] $\mathbb{1}_{\mathcal{S}} \triangleq \boldsymbol{\lambda}\, x \in \mathcal{S} \bullet x$ is the identity function on the set $\mathcal{S}$.

Definition (19.1, Domain well-definedness) We say that a domain

$$\mathbb{D}^{¤} \quad \triangleq \quad \langle \mathbb{P}^{¤},\ \sqsubseteq^{¤},\ \bot^{¤},\ \sqcup^{¤},\ \mathsf{assign}^{¤}[\![\mathsf{x},\mathsf{A}]\!],\ \mathsf{test}^{¤}[\![\mathsf{B}]\!],\ \overline{\mathsf{test}}^{¤}[\![\mathsf{B}]\!]\rangle$$

is *well-defined* when $\langle \mathbb{P}^{¤}, \sqsubseteq^{¤}\rangle$ is a poset of properties with infimum $\bot^{¤}$, the lub $\sqcup^{¤}$ is well-defined for pairs of properties, and $\sqsubseteq^{¤}$-increasing chains (so $\langle \mathbb{P}^{¤}, \sqsubseteq^{¤}\rangle$ is a join-lattice and a cpo), the assignment $\mathsf{assign}^{¤}$ is well-defined in $(\mathbb{V} \times \mathbb{E}) \rightarrow \mathbb{P}^{¤} \xrightarrow{\nearrow} \mathbb{P}^{¤}$, and the tests $\mathsf{test}^{¤}[\![\mathsf{B}]\!]$ and $\overline{\mathsf{test}}^{¤}[\![\mathsf{B}]\!]$ are well-defined in $\mathbb{B} \rightarrow \mathbb{P}^{¤} \xrightarrow{\nearrow} \mathbb{P}^{¤}$.

The abstract domain $\mathbb{D}^{¤}$ is an algebra while the domain of abstract properties $\mathbb{P}^{¤}$ is a set. So the mathematical structures are different. However, following mathematicians that call $\mathbb{Z}$ the "ring of integers" where a ring is an algebraic structure and $\mathbb{Z}$ is a set, we often say, by abuse of language, that $\mathbb{P}^{¤}$ an abstract domain.

Abstract structural semantics/interpreter

The semantics can be implemented as instances of a generic abstract interpreter defined below.

- *Abstract semantics of a statement list* `Sl` ::= `Sl′ S`

$$\widehat{\mathcal{S}}^{\,¤}[\![\mathtt{Sl}]\!]\,\mathcal{R}_0\,\ell \;\triangleq\; \llparenthesis\, \ell \in \mathsf{labs}[\![\mathtt{Sl}']\!] \setminus \{\mathsf{at}[\![\mathtt{S}]\!]\} \mathrel{?} \widehat{\mathcal{S}}^{\,¤}[\![\mathtt{Sl}']\!]\,\mathcal{R}_0\,\ell \;\|\; \ell \in \mathsf{labs}[\![\mathtt{S}]\!] \mathrel{?} \widehat{\mathcal{S}}^{\,¤}[\![\mathtt{S}]\!](\widehat{\mathcal{S}}^{\,¤}[\![\mathtt{Sl}']\!]\,\mathcal{R}_0\,\mathsf{at}[\![\mathtt{S}]\!])\,\ell \mathrel{:} \bot^{¤} \,\rrparenthesis \tag{19.5}$$

- *Abstract semantics of an empty statement list* `Sl` ::= ϵ

$$\widehat{\mathcal{S}}^{\,¤}[\![\mathtt{Sl}]\!]\,\mathcal{R}_0\,\ell \;\triangleq\; \llparenthesis\, \ell = \mathsf{at}[\![\mathtt{Sl}]\!] \mathrel{?} \mathcal{R}_0 \mathrel{:} \bot^{¤} \,\rrparenthesis \tag{19.6}$$

- *Abstract semantics of an assignment statement* `S ::= x = A ;`

$$\begin{aligned}\widehat{\mathcal{S}}^{\text{¤}}[\![\mathsf{S}]\!]\,\mathcal{R}_0\,\ell \quad = \quad & (\!|\ \ell = \mathsf{at}[\![\mathsf{S}]\!]\ ?\ \mathcal{R}_0 \\ & \|\ \ell = \mathsf{after}[\![\mathsf{S}]\!]\ ?\ \mathsf{assign}^{\text{¤}}[\![\mathsf{x}, \mathsf{A}]\!]\,\mathcal{R}_0 \\ & \mathrel{:}\ \bot^{\text{¤}}\ |\!)\end{aligned} \tag{19.7}$$

where $\mathsf{assign}[\![\mathsf{x}, \mathsf{A}]\!] \circ \gamma \sqsubseteq \gamma \circ \mathsf{assign}^{\text{¤}}[\![\mathsf{x}, \mathsf{A}]\!]$.

- *Abstract semantics of a conditional statement* `S ::= if (B)` $\mathtt{S}_t$

$$\begin{aligned}\widehat{\mathcal{S}}^{\text{¤}}[\![\mathsf{S}]\!]\,\mathcal{R}_0\,\ell \quad = \quad & \llparenthesis\ \ell = \mathsf{at}[\![\mathsf{S}]\!]\ ?\ \mathcal{R}_0 \\ & [\![\ \ell \in \mathsf{in}[\![\mathsf{S}_t]\!]\ ?\ \widehat{\mathcal{S}}^{\text{¤}}[\![\mathsf{S}_t]\!]\,(\mathsf{test}^{\text{¤}}[\![\mathsf{B}]\!]\,\mathcal{R}_0)\,\ell \\ & [\![\ \ell = \mathsf{after}[\![\mathsf{S}]\!]\ ? \\ & \quad \widehat{\mathcal{S}}^{\text{¤}}[\![\mathsf{S}_t]\!]\,(\mathsf{test}^{\text{¤}}[\![\mathsf{B}]\!]\,\mathcal{R}_0)\,\ell \sqcup^{\text{¤}} \overline{\mathsf{test}}^{\text{¤}}[\![\mathsf{B}]\!]\,\mathcal{R}_0 \\ & ⁏\ \bot^{\text{¤}}\ \rrparenthesis\end{aligned} \tag{19.9}$$

where $\mathsf{test}[\![\mathsf{B}]\!] \circ \gamma \sqsubseteq \gamma \circ \mathsf{test}^{\text{¤}}[\![\mathsf{B}]\!]$ and $\overline{\mathsf{test}}[\![\mathsf{B}]\!] \circ \gamma \sqsubseteq \gamma \circ \overline{\mathsf{test}}^{\text{¤}}[\![\mathsf{B}]\!]$.

- *Abstract semantics of an iteration statement* $\mathtt{S} ::= \mathtt{while}\ \ell\ \mathtt{(B)}\ \mathtt{S}_b$

$$\widehat{\mathcal{S}}^{\text{¤}}[\![\mathtt{S}]\!]\,\mathcal{R}_0\,\ell' \quad=\quad \mathsf{lfp}^{\dot{\sqsubseteq}^{\text{¤}}}\,(\mathcal{F}^{\text{¤}}[\![\mathtt{while}\ \ell\ \mathtt{(B)}\ \mathtt{S}_b]\!]\,\mathcal{R}_0)\,\ell' \tag{19.11}$$

$$\mathcal{F}^{\text{¤}}[\![\mathtt{while}\ \ell\ \mathtt{(B)}\ \mathtt{S}_b]\!] \quad\in\quad \mathbb{P}^{\text{¤}} \to ((\mathbb{L} \to \mathbb{P}^{\text{¤}}) \to (\mathbb{L} \to \mathbb{P}^{\text{¤}}))$$

$$\begin{array}{l}
\mathcal{F}^{\text{¤}}[\![\mathtt{while}\ \ell\ \mathtt{(B)}\ \mathtt{S}_b]\!]\,\mathcal{R}_0\,X\,\ell' \quad = \\
\quad (\!|\ \ell' = \ell \mathrel{?} \mathcal{R}_0 \sqcup^{\text{¤}} \widehat{\mathcal{S}}^{\text{¤}}[\![\mathtt{S}_b]\!]\,(\mathsf{test}^{\text{¤}}[\![\mathtt{B}]\!]X(\ell))\,\ell \\
\quad\ \|\ \ell' \in \mathsf{in}[\![\mathtt{S}_b]\!] \setminus \{\ell\} \mathrel{?} \widehat{\mathcal{S}}^{\text{¤}}[\![\mathtt{S}_b]\!]\,(\mathsf{test}^{\text{¤}}[\![\mathtt{B}]\!]X(\ell))\,\ell' \\
\quad\ \|\ \ell' = \mathsf{after}[\![\mathtt{S}]\!] \mathrel{?} \overline{\mathsf{test}}^{\text{¤}}[\![\mathtt{B}]\!]X(\ell) \sqcup^{\text{¤}} \displaystyle\bigsqcup^{\text{¤}}_{\ell'' \in \mathsf{breaks\text{-}of}[\![\mathtt{S}_b]\!]} \widehat{\mathcal{S}}^{\text{¤}}[\![\mathtt{S}_b]\!]\,(\mathsf{test}^{\text{¤}}[\![\mathtt{B}]\!]X(\ell))\,\ell'' \\
\quad\ \mathrel{:} \bot^{\text{¤}}\ |\!)
\end{array}$$

- *Abstract semantics of a break statement* $\mathtt{S} ::= \ell$ **`break ;`**

$$\widehat{\mathcal{S}}^{¤}[\![\mathsf{S}]\!]\,\mathcal{R}_0\,\ell \quad = \quad (\!\![\; \ell = \mathsf{at}[\![\mathsf{S}]\!] \mathrel{?} \mathcal{R}_0 \mathrel{:} \bot^{¤} \;]\!\!) \tag{19.12}$$

Proof methods

Invariance proof methods

- Invariance proof methods derive from the reachability semantics
  - abstraction to verification conditions → Turing/Floyd/Naur proof method
  - abstraction to Hoare triples → Hoare logic
  - Fixpoints:

Theorem (22.1, Fixpoint induction) Let $f \in \mathcal{L} \xrightarrow{\nearrow} \mathcal{L}$ be an increasing function on a complete lattice $\langle \mathcal{L}, \sqsubseteq, \bot, \top, \sqcap, \sqcup \rangle$ and $P \in \mathcal{L}$.
We have $\mathsf{lfp}^{\sqsubseteq} f \sqsubseteq P \Leftrightarrow \exists I \in \mathcal{L} \,.\, f(I) \sqsubseteq I \wedge I \sqsubseteq P$.

Bibliography on verification and proofs

References I

Alglave, Jade and Patrick Cousot (2017). “Ogre and Pythia: an invariance proof method for weak consistency models”. In: *POPL*. ACM, pp. 3–18.

Cousot, Patrick (1990). “Methods and Logics for Proving Programs”. In: *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*. MIT Press Cambridge, MA, USA ©1990, pp. 841–994.

– (2002). “Constructive design of a hierarchy of semantics of a transition system by abstract interpretation”. *Theor. Comput. Sci.* 277.1-2, pp. 47–103.

– (2003). “Verification by Abstract Interpretation”. In: *Verification: Theory and Practice*. Vol. 2772. Lecture Notes in Computer Science. Springer, pp. 243–268.

Cousot, Patrick, Roberto Giacobazzi, and Francesco Ranzato (2018). “Program Analysis Is Harder Than Verification: A Computability Perspective”. In: *CAV (2)*. Vol. 10982. Lecture Notes in Computer Science. Springer, pp. 75–95.

Floyd, Robert W. (1967). “Assigning meaning to programs”. In: J.T. Schwartz, ed. *Proc. Symp. in Applied Math.* Vol. 19. Amer. Math. Soc., pp. 19–32.

References II

Hoare, C. A. R. (1978). “Some Properties of Predicate Transformers”. *J. ACM* 25.3, pp. 461–480.

Naur, Peter (1966). “Proofs of algorithms by general snapshots”. *BIT* 6, pp. 310–316.

The End of Part 3

Part 4

Symbolic abstraction: dependency analysis

Motivation

Dependency

Found in many reasonings on programs:

- Non-interference (confidentiality, integrity)
- Security, privacy
- Program slicing
- Temporal dependencies in synchronous languages (Esterelle, Lustre, Signal, ... called causality there)
- etc.

Dependency

The existing definitions

- are given a priori (*e.g*. Cheney, Ahmed, and Acar, 2011; D. E. Denning and P. J. Denning, 1977),
- without semantics justification (except Assaf, Naumann, Signoles, Totel, and Tronel, 2017 (“hyper-collecting semantics”), Urban and Müller, 2018)
- are dependencies on program exit only

Our objective is to study principles, not to get a new powerful dependency analysis

Dependency, informally

Functional dependency

- A function $f(\ldots, x, \ldots)$ depends on its parameter x if and only if changing only this parameter changes the result

$$\exists x_1, x_2 \mathrel{.} f(\ldots, x_1, \ldots) \neq f(\ldots, x_2, \ldots)$$

- Example: $f(x, y) = x - (y - y)$ depends on x but not on y
- Definition:

$$\begin{aligned}
\mathcal{F}d^{ni} &\triangleq \{ f \mid \exists x_1, \ldots, x_n, x_i{}' \mathrel{.} f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) \neq \\
&\qquad\qquad f(x_1, \ldots, x_{i-1}, x_i{}', x_{i+1}, \ldots, x_n)\}. \\
\mathcal{F}d &\triangleq \bigcup_{n\in\mathbb{N}_*} \bigcup_{1\leqslant i\leqslant n} \mathcal{F}d^{ni}
\end{aligned} \tag{44.1}$$

Non-interference

- Given low variables L (*e.g.* “public” respectively “untainted”) and high variables H (“private/conf” respectively “tainted”)
- Non-interference (Cohen, 1977; Goguen and Meseguer, 1982, 1984; Mantel, 2003) is defined as “if executions start with the same values of the low variables then, upon termination, if ever, the low variables are equal (so changing initial high variables cannot change final low variables)
- The non-interference property is therefore

$$\begin{aligned}\mathcal{N}i(L, H) \quad = \quad & \{\Pi \in \wp(\mathbb{T}^{+} \times \mathbb{T}^{\infty}) \mid \forall \langle \pi_0,\ \pi\rangle, \langle \pi'_0,\ \pi'\rangle \in \Pi \cap (\mathbb{T}^{+} \times \mathbb{T}^{+})\ .\\ & \qquad (\forall \mathsf{x} \in L\ .\ \boldsymbol{\rho}(\pi_0)\mathsf{x} = \boldsymbol{\rho}(\pi'_0)\mathsf{x}) \Rightarrow (\forall \mathsf{x} \in L\ .\ \boldsymbol{\rho}(\pi_0 \frown \pi)\mathsf{x} = \boldsymbol{\rho}(\pi'_0 \frown \pi')\mathsf{x})\}\end{aligned}$$

- Interference during the computation and non termination are not taken into account.

General idea of dependency

- y depends on the initial value x_0 of x at ℓ if and only if changing x_0 changes the future observations of y at ℓ
- We consider dependency on initial values of variables

  More generally, changing an abstraction of the past at ℓ changes an abstraction of the future after ℓ

Dependency is local

- ℓ_1 `y = 0 ;`ℓ_2 `y = x ;`ℓ_3
  - the value of `y` at ℓ_1 is the initial value y_0 of `y` at ℓ_1
    Changing the initial value of `x` does not change the value of `y` at ℓ_1 so
    `y` does not depend on the initial value of `x` at ℓ_1
  - the value of `y` at ℓ_2 is 0.
    Changing the initial value of `x` does not change the value of `y` at ℓ_2 so
    `y` does not depend on the initial value of `x` at ℓ_2
  - the value of `y` at ℓ_3 is the initial value x_0 of `x`.
    Changing the initial value of `x` changes the value of `y` at ℓ_3 so
    `y` depends on the initial value of `x` at ℓ_3

⇒ dependency upon the initial value of variables is local (may be different at different program points).

Dependency depends on values of variables

```
{if (x=0) y=x; else y=0;} ℓ
```

- The value of y at ℓ is always 0, no dependency

```
{if (x=0) y=x; else y=1;} ℓ
```

- The value of y at ℓ is
  - if $x_0 = 0$ then "0"
  - if $x_0 \neq 0$ then "1"
- y at ℓ depends on x_0 (unless $(x_0 = 0 \wedge y_0 = 0) \vee (x_0 \neq 0 \wedge y_0 = 1)$)

⇒ dependency of y upon the initial value x_0 of x depends on the initial and current values of x and y

⇒ this is ignored in D. E. Denning and P. J. Denning, 1977's dataflow analysis

Dependency depends on sequences of observations of values of variables

$\mathtt{P}_u \triangleq$ `while` $^{\ell}$ `(0==0) x=x+1;`

- One can observe $x_0 \cdot x_0+1 \cdot x_0+2 \cdot x_0+17 \cdot x_0+18 \cdot \ldots x_0+42 \cdot x_0+43 \cdot \ldots$ at $^{\ell}$
- changing the initial value x_0 of `x` changes this observation
- `x` at $^{\ell}$ depends upon x_0

$\mathtt{P}_0 \triangleq$ `x=0; while` $^{\ell}$ `(0==0) x=x+1;`

- One can observe $0 \cdot 1 \cdot 2 \cdot \ldots 17 \cdot 18 \cdot \ldots \cdot 42 \cdot 43 \cdot \ldots$ at $^{\ell}$
- changing the initial value x_0 of `x` does not change this observation
- `x` at $^{\ell}$ does not depend upon x_0

⇒ We must observe the maximal sequence of values successively taken by a variable at a program point

Counterfactual dependency: absence of observation

```
int x,y; if (x=0) { y=x; ℓ}
```

- Observation of `y` at ℓ:
  - if $x_0 = 0$ then "0"
  - if $x_0 \neq 0$ then "" (empty observations: no execution ever reaches ℓ)

⇒ Dependency if empty observations are taken into account

⇒ No dependency if empty observations are not taken into account

⇒ The choice is completely arbitrary!

Counterfactual value dependency: absence of observation

```
int x,y,z; if (x=0) { y=x; ℓ}
```

- Assume that empty observations are taken into account (so y depends on x_0)
- Observation of `z` at ℓ:
  - if $x_0 = 0$ then "z_0" (initial value of z)
  - if $x_0 \neq 0$ then "" (empty observations: no execution ever reaches ℓ)
- Two different observations at ℓ!
- Should z depends on x_0 at ℓ?

⇒ The choice is completely arbitrary!

  - No
  - Yes
  - Yes if the value of z at ℓ is different from z_0 (D. E. Denning and P. J. Denning, 1977)

Timing dependency

```
while ℓ (x > 0) x = x - 1 ;
```

- Does variable y (s.t. y ≠ x) at ℓ depends on the initial value x_0 of x?
  - The observation of y at ℓ is $y_0 \cdot y_0 \cdot \ldots \cdot y_0$ repeated $x_0 + 1$ times.
  - So changing x_0 changes the observation of y at ℓ

⇒ This is a *covert/side channel* (Lampson, 1973; Mulder, Eisenbarth, and Schaumont, 2018), more precisely, a *timing channel* (Russo, Hughes, Naumann, and Sabelfeld, 2006; Sabelfeld and Myers, 2003)

⇒ The choice of ignoring timing channel is arbitrary

⇒ Ignored in the classical definition of dependency D. E. Denning and P. J. Denning, 1977

⇒ One way of ignoring timing channels is to require that observation sequences must differ by at least one data

Counterfactual timing dependency

```
/* x {0,1} */ while (x != 0) ℓ y = x ;
```

- If $x_0 = 1$, the infinite sequence of values of y observed at ℓ is $y_0 \cdot 1 \cdot 1 \cdots$.
- If $x_0 = 0$, then the observation at ℓ is the empty sequence $\ni$.
- Does y at ℓ depends on the initial value x_0 of x?
- This depends on hypotheses on observables. Is an infinite sequence of values observable? Is the empty sequence $\ni$ of values observable?
- This is debatable and problem-specific
- For example if a program terminates it is easy to check on program termination that a program point is never reached. This may be considered impossible with non-termination.

Dependency, formally

Future observations

- initialisation trace $\pi_0 \in \mathbb{T}^{+}$
- (non empty) continuation trace $\pi \in \mathbb{T}^{+\infty}$
- $\mathsf{future}[\![\mathsf{y}]\!]\ell(\pi_0,\ \pi)$ is the sequence of values of y successively observed at program point point ℓ in the trace π continuing π_0 [4]

$$
\begin{array}{rcl}
\mathsf{future}[\![\mathsf{y}]\!]\ell(\pi_0, \ell) & \triangleq & \boldsymbol{\rho}(\pi_0)\mathsf{y} \\
\mathsf{future}[\![\mathsf{y}]\!]\ell(\pi_0, \ell') & \triangleq & \backepsilon \\
\mathsf{future}[\![\mathsf{y}]\!]\ell(\pi_0, \ell \xrightarrow{a} \ell''\pi) & \triangleq & \boldsymbol{\rho}(\pi_0)\mathsf{y} \cdot \mathsf{future}[\![\mathsf{y}]\!]\ell(\pi_0 \smallfrown \ell \xrightarrow{a} \ell'', \ell''\pi) \\
\mathsf{future}[\![\mathsf{y}]\!]\ell(\pi_0, \ell' \xrightarrow{a} \ell''\pi) & \triangleq & \mathsf{future}[\![\mathsf{y}]\!]\ell(\pi_0 \smallfrown \ell' \xrightarrow{a} \ell'', \ell''\pi)
\end{array}
$$

- $\mathsf{future}[\![\mathsf{y}]\!]\ell(\pi_0, \pi)$ is the empty sequence $\backepsilon$ if ℓ does not appear in π

[4] this should be understood as a bi-inductive definition of P. Cousot and R. Cousot, 2009 to properly handle non-termination

Observations

- An observation $\langle\underline{\nu}, \omega\rangle$ of a variable at a program point is a pair of
  - an initial value $\underline{\nu}$ of the variable
  - the future observation ω of this variable from that program point on

Differences between future observations $\langle \underline{\nu},\ \omega \rangle$ and $\langle \underline{\nu}',\ \omega' \rangle$ (I)

(1) Counterfactual timing dependency:

$$\text{ctdep}(\langle \underline{\nu},\ \omega \rangle, \langle \underline{\nu}',\ \omega' \rangle) \quad \triangleq \quad \omega \neq \omega'$$

(empty observations are allowed)

(2) Timing dependency:

$$\text{tdep}(\langle \nu,\ \omega \rangle, \langle \nu',\ \omega' \rangle) \quad \triangleq \quad \omega \neq \omega' \wedge \omega \neq \backepsilon \wedge \omega' \neq \backepsilon$$

(empty observations are disallowed)

Differences between future observations $\langle \underline{\nu}, \omega\rangle$ and $\langle \underline{\nu}', \omega'\rangle$ (II)

(3) Value dependency:

$$\begin{aligned}\mathsf{vdep}(\langle \underline{\nu}, \omega\rangle, \langle \underline{\nu}', \omega'\rangle) \quad \triangleq \quad & \exists \omega_0, \omega_1, \omega'_1, \nu, \nu' \;. \\ & \omega = \omega_0 \cdot \nu \cdot \omega_1 \wedge \omega' = \omega_0 \cdot \nu' \cdot \omega'_1 \wedge \nu \neq \nu'\end{aligned}$$

(different values of the variable must be observed)

Example 6 `if` ℓ_0 `(x == 1) {` ℓ_1 `y = x ;` ℓ_2 `}` ℓ_3
`y` does not depend on `x` at ℓ_0, ℓ_1, and ℓ_2 but `y` depends on `x` at ℓ_3 (unless $\mathtt{y} = 1$ at ℓ_0).
□

Differences between future observations $\langle\underline{\nu}, \omega\rangle$ and $\langle\underline{\nu}', \omega'\rangle$ (III)

(4) counterfactual value dependency:

$$\text{cvdep}(\langle\underline{\nu}, \omega\rangle, \langle\underline{\nu}', \omega'\rangle) \triangleq \text{vdep}(\langle\underline{\nu}, \omega\rangle, \langle\underline{\nu}', \omega'\rangle) \vee (\omega = \ni \wedge \omega' \neq \ni) \vee (\omega \neq \ni \wedge \omega' = \ni)$$

(an empty observation is allowed)

Example 7 `if` ℓ_0 `(x == 1) {` ℓ_1 `y = x ;` ℓ_2 `}` ℓ_3
y depends on x at ℓ_2 (unless $y = 1$ at ℓ_0).
Any variable depends on the initial value of x at ℓ_1 and ℓ_2. □

Differences between future observations $\langle \underline{\nu},\ \omega \rangle$ and $\langle \underline{\nu}',\ \omega' \rangle$ (IV)

(5) Counterfactual multi-values dependency:

$$\begin{aligned}\mathsf{cmvdp}(\langle \underline{\nu},\ \omega \rangle, \langle \underline{\nu}',\ \omega' \rangle) \quad \triangleq \quad & \mathsf{vdep}(\langle \underline{\nu},\ \omega \rangle, \langle \underline{\nu}',\ \omega' \rangle) \vee \\ & \quad (\omega = \ni \wedge \exists \omega_0', \nu', \omega_1' \ .\ \omega' = \omega_0' \cdot \nu' \cdot \omega_1' \wedge \underline{\nu}' \neq \nu') \vee \\ & \qquad\qquad (\omega' = \ni \wedge \exists \omega_0, \nu, \omega_1 \ .\ \omega = \omega_0 \cdot \nu \cdot \omega_1 \wedge \underline{\nu} \neq \nu)\end{aligned}$$

(an empty observation is allowed for variables which value has changed)

Example 8 `if` ℓ_0 `(x == 1) {` ℓ_1 `y = x ;` ℓ_2 `}` ℓ_3
No variable depends on the initial value of `x` at ℓ_1 and only `y` at ℓ_2 (unless `y` is initially 1).
This is D. E. Denning and P. J. Denning, 1977. □

Formal definition of dependency

- Dependency property:

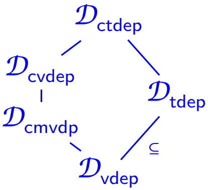
$$\begin{aligned}\mathcal{D}_{\mathsf{dep}}{}^{\ell}\langle \mathsf{x},\ \mathsf{y}\rangle \ \triangleq\ \{\Pi \in \wp(\mathbb{T}^{+} \times \mathbb{T}^{+\infty}) \mid \exists \langle \pi_0,\ \pi_1\rangle, \langle \pi'_0,\ \pi'_1\rangle \in \Pi \,.\\ (\forall \mathsf{z} \in \mathbb{V} \setminus \{\mathsf{x}\} \,.\, \boldsymbol{\rho}(\pi_0)\mathsf{z} = \boldsymbol{\rho}(\pi'_0)\mathsf{z}) \wedge\\ \mathsf{dep}(\langle \boldsymbol{\rho}(\pi_0)\mathsf{y},\ \mathsf{future}[\![\mathsf{y}]\!]^{\ell}(\pi_0, \pi_1)\rangle, \langle \boldsymbol{\rho}(\pi'_0)\mathsf{y},\ \mathsf{future}[\![\mathsf{y}]\!]^{\ell}(\pi'_0, \pi'_1)\rangle)\}\end{aligned}$$

- choose $\mathsf{dep} \in \{\mathsf{vdep}, \mathsf{cmvdp}, \mathsf{cvdep}, \mathsf{tdep}, \mathsf{ctdep}\}$ to get 5 different definitions
- y depends on the initial value of x at point ℓ of program P is:

$$\widehat{\mathcal{S}}^{+\infty}[\![\mathsf{P}]\!] \ \in\ \mathcal{D}_{\mathsf{dep}}{}^{\ell}\langle \mathsf{x},\ \mathsf{y}\rangle$$

- No necessary distinction between explicits and implicits flows as in D. E. Denning and P. J. Denning, 1977

Dependency lattice



$\mathcal{D}_{\textsf{ctdep}}$

$\mathcal{D}_{\textsf{cvdep}}$

$\mathcal{D}_{\textsf{tdep}}$

$\mathcal{D}_{\textsf{cmvdp}}$

$\subseteq$

$\mathcal{D}_{\textsf{vdep}}$

(??)

- The more differences between observed futures, the more dependencies;
- Not clear with postulated definitions (such as the hydraulic model where dependency depends on the rules to mix colors)

Why maximal traces?

- For prefix traces, if a trace is in the semantics, all of its prefixes are also in the semantics, which introduces artificial timing channels

Prefix traces for dependency on values

- For value dependencies, the maximal trace semantics can be replaced by the prefix trace semantics withou problem:

Lemma $\mathcal{S}^{+\infty}[\![\mathsf{P}]\!] \in \mathcal{D}_{\mathsf{vdep}}{}^{\ell}\langle \mathsf{x},\ \mathsf{y}\rangle \Leftrightarrow \mathcal{S}^{*}[\![\mathsf{P}]\!] \in \mathcal{D}_{\mathsf{vdep}}{}^{\ell}\langle \mathsf{x},\ \mathsf{y}\rangle$

- Idem if we include empty observations (the prefixes of $\mathcal{S}^{*}[\![\mathsf{P}]\!]\pi_0$ are never empty, so no possible confusion)

Dependency abstraction

Abstraction of data dependency

- The abstraction of a semantic property $\mathcal{S} \in \wp(\wp(\mathbb{T}^+ \times \mathbb{T}^{+\infty}))$ into a data dependency property $\alpha^{\mathsf{vdep}}(\mathcal{S}) \in \mathbb{L} \rightarrow \wp(\mathbb{V} \times \mathbb{V})$ is:

$$\boxed{\quad \alpha^{\mathsf{vdep}}((\mathcal{S})\ell \quad \triangleq \quad \{\langle \mathsf{x},\ \mathsf{y}\rangle \mid \mathcal{S} \in \mathcal{D}_{\mathsf{vdep}}{}^{\ell}\langle \mathsf{x},\ \mathsf{y}\rangle\} \quad}$$

- This is a Galois connection:

Lemma 10 $\langle \wp(\wp(\mathbb{T}^+ \times \mathbb{T}^{+\infty})), \subseteq\rangle \mathrel{\underset{\alpha^{\mathsf{vdep}}}{\overset{\gamma^{\mathsf{vdep}}}{\leftrightarrows}}} \langle \mathbb{L} \rightarrow \wp(\mathbb{V} \times \mathbb{V}), \supseteq^{\mathrm{d}}\rangle$ where the concretization of a dependency property $\mathbf{D} \in \mathbb{L} \rightarrow \wp(\mathbb{V} \times \mathbb{V})$ is:

$$\gamma^{\mathsf{vdep}}(\mathbf{D}) \quad \triangleq \quad \bigcap_{\ell \in \mathbb{L}} \ \bigcap_{\langle \mathsf{x},\ \mathsf{y}\rangle \in \mathbf{D}(\ell)} \ \mathcal{D}_{\mathsf{vdep}}{}^{\ell}\langle \mathsf{x},\ \mathsf{y}\rangle$$

(the more semantics, the less dependencies)

Value dependency static analysis

Potential value dependency

- $\alpha^{\mathsf{vdep}}(\{\mathcal{S}^{+\infty}[\![\mathsf{s}]\!]\}) = \alpha^{\mathsf{vdep}}(\{\mathcal{S}^{*}[\![\mathsf{s}]\!]\})$ is not computable (Rice theorem)
- We design an over-approximation:

Potential value dependency semantics $\widehat{\mathcal{S}}^{\mathsf{vdep}}_{\exists}$ *:*

$$\alpha^{\mathsf{vdep}}(\{\mathcal{S}^{+\infty}[\![\mathsf{s}]\!]\}) \quad \dot{\subseteq} \quad \widehat{\mathcal{S}}^{\mathsf{vdep}}_{\exists}[\![\mathsf{s}]\!]$$

- The abstraction of D. E. Denning and P. J. Denning, 1977 is purely syntactic (in dataflow analysis style)
- We do slightly better, by taking values into account, in a very simple way

Example

```
if ℓ0 (x == 1) { ℓ1 y = z ;ℓ2 };ℓ3
```

- we have the potential value dependency:

ℓ	ℓ_0	ℓ_1	ℓ_2	ℓ_3
$\widehat{\overline{\mathcal{S}}}_{\exists}^{\text{vdep}}[\![\mathtt{S}]\!]\,\ell$	$\{\langle x, x\rangle, \langle y, y\rangle, \langle z, z\rangle\}$	$\{\langle y, y\rangle, \langle z, z\rangle\}$	$\{\langle z, y\rangle, \langle z, z\rangle\}$	$\{\langle x, x\rangle, \langle x, y\rangle, \langle y, y\rangle, \langle z, y\rangle, \langle z, z\rangle\}$

- this is an over-approximation since *e.g.* z flows to y at ℓ_3 only when $x = 1$ at ℓ_0.

Calculational design

- By calculus (in principle, can be checked with Coq like Jourdan, Laporte, Blazy, Leroy, and Pichardie, 2015)
- By structural induction on the program syntax
- By fixpoint over-approximation for iterations:

  Theorem (over-approximation of fixpoints) If $\langle C, \sqsubseteq, \perp, \top, \sqcup, \sqcap\rangle$ and $\langle \mathcal{A}, \preccurlyeq, 0, 1, \curlyvee, \curlywedge\rangle$ are complete lattices, $\langle C, \sqsubseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathcal{A}, \preccurlyeq\rangle$ is a Galois connection, $f \in C \xrightarrow{\nearrow} C$ and $\overline{f} \in \mathcal{A} \xrightarrow{\nearrow} \mathcal{A}$ are increasing and $\alpha \circ f \mathrel{\dot{\preccurlyeq}} \overline{f} \circ \alpha$ (*semi-commutation*) then $\mathsf{lfp}^{\sqsubseteq} f \sqsubseteq \gamma(\mathsf{lfp}^{\preccurlyeq} \overline{f})$.

- Finite domain, no widening needed

Potential dependency semantics of assignment $\mathtt{S} ::= \mathtt{x} = \mathtt{A}\ ;$

$$\begin{aligned}
\widehat{\mathcal{S}}^{\text{vdep}}_{\exists}[\![\mathsf{S}]\!]\,\ell \ &= \ (\!|\ \ell = \mathsf{at}[\![\mathsf{S}]\!] \ ?\ \mathbb{1}_{\mathbb{V}} \\
&\qquad \|\ \ell = \mathsf{after}[\![\mathsf{S}]\!] \ ?\ \{\langle \mathsf{y},\ \mathsf{x}\rangle \mid \mathsf{y} \in \widehat{\mathcal{S}}^{\text{vdep}}_{\exists}[\![\mathsf{A}]\!]\} \cup \\
&\qquad \qquad \qquad \qquad \qquad \{\langle \mathsf{y},\ \mathsf{y}\rangle \mid \mathsf{y} \neq \mathsf{x}\} \\
&\qquad \ \text{:}\ \varnothing\ |\!) \\
\widehat{\mathcal{S}}^{\text{vdep}}_{\exists}[\![\mathsf{A}]\!] \ &\triangleq \ \{\mathsf{y} \mid \exists \rho \in \mathbb{E}\mathbb{v}\ .\ \exists \nu \in \mathbb{V}\ .\ \mathcal{A}[\![\mathsf{A}]\!]\rho \neq \mathcal{A}[\![\mathsf{A}]\!]\rho[\mathsf{y} \leftarrow \nu]\} \\
&\subseteq \ \mathbb{vars}[\![\mathsf{A}]\!]
\end{aligned}$$

Example:

- after $\mathtt{x = y - y\ ;}$, $\mathtt{x}$ depends on $\mathtt{y}$.
- after $\mathtt{x = y\ ;\ x = y - x\ ;}$, $\mathtt{x}$ depends on the initial values of $\mathtt{x}$ and $\mathtt{y}$
- To be more precise we would have to preserve information on the values of variables (eg. $\mathsf{x} = \mathsf{y}$)

Proof (don't read ☺) I

The cases ℓ = at$\llbracket$S$\rrbracket$ was handled in (44.38) and $\ell \notin$ labx$\llbracket$S$\rrbracket$ in (44.39). It remains the case ℓ = after$\llbracket$S$\rrbracket$.

$$\begin{aligned}
&\alpha^{\mathsf{vdep}}(\{\mathcal{S}^{+\infty}\llbracket \mathsf{S}\rrbracket\})\ \mathsf{after}\llbracket \mathsf{S}\rrbracket\\
=\ &\alpha^{\mathsf{vdep}}(\{\mathcal{S}^{*}\llbracket \mathsf{S}\rrbracket\})\ \mathsf{after}\llbracket \mathsf{S}\rrbracket && \wr\text{Lemma 44.25}\wr\\
=\ &\{\langle \mathsf{x}',\ \mathsf{y}\rangle \mid \mathcal{S}^{*}\llbracket \mathsf{S}\rrbracket \in \mathcal{D}_{\mathsf{vdep}}(\mathsf{after}\llbracket \mathsf{S}\rrbracket)\langle \mathsf{x}',\ \mathsf{y}\rangle\} && \wr\text{def. (44.29) of } \alpha^{\mathsf{vdep}} \text{ and def. } \subseteq\wr\\
=\ &\{\langle \mathsf{x}',\ \mathsf{y}\rangle \mid \exists\langle \pi_0,\ \pi_1\rangle, \langle \pi'_0,\ \pi'_1\rangle \in \mathcal{S}^{*}\llbracket \mathsf{S}\rrbracket\ .\ \forall \mathsf{z} \in \mathbb{V} \setminus \{\mathsf{x}'\}\ .\ \boldsymbol{\rho}(\pi_0)\mathsf{z} = \boldsymbol{\rho}(\pi'_0)\mathsf{z} \wedge \mathsf{vdep}(\langle \boldsymbol{\rho}(\pi_0)\mathsf{y},\\
&\quad \mathsf{future}\llbracket \mathsf{y}\rrbracket(\mathsf{after}\llbracket \mathsf{S}\rrbracket)(\pi_0, \pi_1)\rangle, \langle \boldsymbol{\rho}(\pi'_0)\mathsf{y},\ \mathsf{future}\llbracket \mathsf{y}\rrbracket(\mathsf{after}\llbracket \mathsf{S}\rrbracket)(\pi'_0, \pi'_1)\rangle)\}\\
& && \wr\text{def. } \in \text{ and (44.20) of } \mathcal{D}_{\mathsf{vdep}}\,\ell\langle \mathsf{x}',\ \mathsf{y}\rangle\wr\\
=\ &\{\langle \mathsf{x}',\ \mathsf{y}\rangle \mid \exists\langle \pi_0,\ \pi_1\rangle, \langle \pi'_0,\ \pi'_1\rangle \in \{\langle \pi\mathsf{at}\llbracket \mathsf{S}\rrbracket,\ \mathsf{at}\llbracket \mathsf{S}\rrbracket \xrightarrow{\mathsf{x}=\mathcal{A}\llbracket \mathsf{A}\rrbracket\boldsymbol{\rho}(\pi\mathsf{at}\llbracket \mathsf{S}\rrbracket)} \mathsf{after}\llbracket \mathsf{S}\rrbracket\rangle \mid \pi\mathsf{at}\llbracket \mathsf{S}\rrbracket \in\\
&\quad \mathbb{T}^{+}\}\ .\ \forall \mathsf{z} \in \mathbb{V} \setminus \{\mathsf{x}'\}\ .\ \boldsymbol{\rho}(\pi_0)\mathsf{z} = \boldsymbol{\rho}(\pi'_0)\mathsf{z} \wedge \mathsf{vdep}(\langle \boldsymbol{\rho}(\pi_0)\mathsf{y},\ \mathsf{future}\llbracket \mathsf{y}\rrbracket(\mathsf{after}\llbracket \mathsf{S}\rrbracket)(\pi_0, \pi_1)\rangle, \langle \boldsymbol{\rho}(\pi'_0)\mathsf{y},\\
&\quad \mathsf{future}\llbracket \mathsf{y}\rrbracket(\mathsf{after}\llbracket \mathsf{S}\rrbracket)(\pi'_0, \pi'_1)\rangle)\}\\
& && \wr\text{def. (15.1) of the assignment prefix finite trace semantics}\wr
\end{aligned}$$

Proof (don't read ☺) II

$$= \{\langle x', y\rangle \mid \exists\langle \pi_0\mathsf{at}\llbracket S\rrbracket,\ \mathsf{at}\llbracket S\rrbracket \xrightarrow{x=\mathscr{A}\llbracket A\rrbracket\boldsymbol{\rho}(\pi_0\mathsf{at}\llbracket S\rrbracket)} \mathsf{after}\llbracket S\rrbracket\rangle, \langle \pi'_0\mathsf{at}\llbracket S\rrbracket,\ \mathsf{at}\llbracket S\rrbracket \xrightarrow{x=\mathscr{A}\llbracket A\rrbracket\boldsymbol{\rho}(\pi'_0\mathsf{at}\llbracket S\rrbracket)} \mathsf{after}\llbracket S\rrbracket\rangle \ .\ \forall z \in \mathbb{V} \setminus \{x'\} \ .\ \boldsymbol{\rho}(\pi_0\mathsf{at}\llbracket S\rrbracket)z = \boldsymbol{\rho}(\pi'_0\mathsf{at}\llbracket S\rrbracket)z \wedge \mathsf{vdep}(\langle\boldsymbol{\rho}(\pi_0)y,\ \mathsf{future}\llbracket y\rrbracket(\mathsf{after}\llbracket S\rrbracket)(\pi_0\mathsf{at}\llbracket S\rrbracket, \mathsf{at}\llbracket S\rrbracket \xrightarrow{x=\mathscr{A}\llbracket A\rrbracket\boldsymbol{\rho}(\pi_0\mathsf{at}\llbracket S\rrbracket)} \mathsf{after}\llbracket S\rrbracket)\rangle, \langle\boldsymbol{\rho}(\pi'_0)y,\ \mathsf{future}\llbracket y\rrbracket(\mathsf{after}\llbracket S\rrbracket)(\pi'_0\mathsf{at}\llbracket S\rrbracket, \mathsf{at}\llbracket S\rrbracket \xrightarrow{x=\mathscr{A}\llbracket A\rrbracket\boldsymbol{\rho}(\pi'_0\mathsf{at}\llbracket S\rrbracket)} \mathsf{after}\llbracket S\rrbracket)\rangle)\} \qquad \wr\text{def. }\in\wr$$

$$= \{\langle x', y\rangle \mid \exists\langle \pi_0\mathsf{at}\llbracket S\rrbracket,\ \mathsf{at}\llbracket S\rrbracket \xrightarrow{x=\mathscr{A}\llbracket A\rrbracket\boldsymbol{\rho}(\pi_0\mathsf{at}\llbracket S\rrbracket)} \mathsf{after}\llbracket S\rrbracket\rangle, \langle \pi'_0\mathsf{at}\llbracket S\rrbracket,\ \mathsf{at}\llbracket S\rrbracket \xrightarrow{x=\mathscr{A}\llbracket A\rrbracket\boldsymbol{\rho}(\pi'_0\mathsf{at}\llbracket S\rrbracket)} \mathsf{after}\llbracket S\rrbracket\rangle \ .\ (\forall z \in \mathbb{V}\setminus\{x'\} \ .\ \boldsymbol{\rho}(\pi_0\mathsf{at}\llbracket S\rrbracket)z = \boldsymbol{\rho}(\pi'_0\mathsf{at}\llbracket S\rrbracket)z) \wedge \mathsf{vdep}(\langle\boldsymbol{\rho}(\pi_0)y,\ \boldsymbol{\rho}(\pi_0\mathsf{at}\llbracket S\rrbracket \xrightarrow{x=\mathscr{A}\llbracket A\rrbracket\boldsymbol{\rho}(\pi_0\mathsf{at}\llbracket S\rrbracket)} \mathsf{after}\llbracket S\rrbracket)y\rangle, \langle\boldsymbol{\rho}(\pi'_0)y,\ \boldsymbol{\rho}(\pi'_0\mathsf{at}\llbracket S\rrbracket \xrightarrow{x=\mathscr{A}\llbracket A\rrbracket\boldsymbol{\rho}(\pi'_0\mathsf{at}\llbracket S\rrbracket)} \mathsf{after}\llbracket S\rrbracket)y\rangle)\}$$

$\wr$def. (44.14) of the future $\mathsf{future}\llbracket y\rrbracket\wr$

$$= \{\langle x', y\rangle \mid \exists\langle \pi_0\mathsf{at}\llbracket S\rrbracket,\ \mathsf{at}\llbracket S\rrbracket \xrightarrow{x=\mathscr{A}\llbracket A\rrbracket\boldsymbol{\rho}(\pi_0\mathsf{at}\llbracket S\rrbracket)} \mathsf{after}\llbracket S\rrbracket\rangle, \langle \pi'_0\mathsf{at}\llbracket S\rrbracket,\ \mathsf{at}\llbracket S\rrbracket \xrightarrow{x=\mathscr{A}\llbracket A\rrbracket\boldsymbol{\rho}(\pi'_0\mathsf{at}\llbracket S\rrbracket)} \mathsf{after}\llbracket S\rrbracket\rangle \ .\ (\forall z \in \mathbb{V} \setminus \{x'\} \ .\ \boldsymbol{\rho}(\pi_0\mathsf{at}\llbracket S\rrbracket)z = \boldsymbol{\rho}(\pi'_0\mathsf{at}\llbracket S\rrbracket)z) \wedge ((\boldsymbol{\rho}(\pi_0\mathsf{at}\llbracket S\rrbracket)y \neq \boldsymbol{\rho}(\pi'_0\mathsf{at}\llbracket S\rrbracket)y) \vee (\boldsymbol{\rho}(\pi_0\mathsf{at}\llbracket S\rrbracket)y = \boldsymbol{\rho}(\pi'_0\mathsf{at}\llbracket S\rrbracket)y \wedge \boldsymbol{\rho}(\pi_0\mathsf{at}\llbracket S\rrbracket \xrightarrow{x=\mathscr{A}\llbracket A\rrbracket\boldsymbol{\rho}(\pi_0\mathsf{at}\llbracket S\rrbracket)} \mathsf{after}\llbracket S\rrbracket)y \neq \boldsymbol{\rho}(\pi'_0\mathsf{at}\llbracket S\rrbracket \xrightarrow{x=\mathscr{A}\llbracket A\rrbracket\boldsymbol{\rho}(\pi'_0\mathsf{at}\llbracket S\rrbracket)} \mathsf{after}\llbracket S\rrbracket)y))\}$$

Proof (don't read ☺) III

⟅(44.18) so that $\mathsf{vdep}(\langle x, a \cdot b\rangle, \langle y, c \cdot d\rangle)$ if and only if (1) $a \neq c$ or (2) $a = c \wedge b \neq d$.⟆

$$= \{\langle \mathsf{x}', \mathsf{y}\rangle \mid \exists \langle \pi_0\mathsf{at}[\![\mathsf{S}]\!], \mathsf{at}[\![\mathsf{S}]\!] \xrightarrow{\mathsf{x}=\mathscr{A}[\![\mathsf{A}]\!]\boldsymbol{\rho}(\pi_0\mathsf{at}[\![\mathsf{S}]\!])} \mathsf{after}[\![\mathsf{S}]\!]\rangle, \langle \pi'_0\mathsf{at}[\![\mathsf{S}]\!], \mathsf{at}[\![\mathsf{S}]\!] \xrightarrow{\mathsf{x}=\mathscr{A}[\![\mathsf{A}]\!]\boldsymbol{\rho}(\pi'_0\mathsf{at}[\![\mathsf{S}]\!])} \mathsf{after}[\![\mathsf{S}]\!]\rangle \,.\, (\forall \mathsf{z} \in \mathbb{V} \setminus \{\mathsf{x}'\} \,.\, \boldsymbol{\rho}(\pi_0\mathsf{at}[\![\mathsf{S}]\!])\mathsf{z} = \boldsymbol{\rho}(\pi'_0\mathsf{at}[\![\mathsf{S}]\!])\mathsf{z}) \wedge ((\mathsf{y} = \mathsf{x}') \vee (\mathsf{y} = \mathsf{x} \wedge \mathscr{A}[\![\mathsf{A}]\!]\boldsymbol{\rho}(\pi_0\mathsf{at}[\![\mathsf{S}]\!]) \neq \mathscr{A}[\![\mathsf{A}]\!]\boldsymbol{\rho}(\pi'_0\mathsf{at}[\![\mathsf{S}]\!])))\}$$

⟅def. (6.2) of $\boldsymbol{\rho}$⟆

$$\subseteq \{\langle \mathsf{x}', \mathsf{y}\rangle \mid ((\mathsf{y} = \mathsf{x}') \vee (\mathsf{y} = \mathsf{x} \wedge \exists \rho, \nu \,.\, \mathscr{A}[\![\mathsf{A}]\!]\rho \neq \mathscr{A}[\![\mathsf{A}]\!]\rho[\mathsf{x}' \leftarrow \nu]))\} \qquad (11)$$

⟅letting $\rho = \boldsymbol{\rho}(\pi_0\mathsf{at}[\![\mathsf{S}]\!])$ and $\nu = \boldsymbol{\rho}(\pi'_0\mathsf{at}[\![\mathsf{S}]\!])(\mathsf{x}')$ so that $\forall \mathsf{z} \in \mathbb{V} \setminus \{\mathsf{x}'\} \,.\, \boldsymbol{\rho}(\pi_0\mathsf{at}[\![\mathsf{S}]\!])\mathsf{z} = \boldsymbol{\rho}(\pi'_0\mathsf{at}[\![\mathsf{S}]\!])\mathsf{z}$ implies that $\boldsymbol{\rho}(\pi'_0\mathsf{at}[\![\mathsf{S}]\!]) = \rho[\mathsf{x}' \leftarrow \nu]$.⟆

$$= \{\langle \mathsf{x}', \mathsf{x}'\rangle \mid \mathsf{x}' \neq \mathsf{x}\} \cup \{\langle \mathsf{x}', \mathsf{x}\rangle \mid \exists \rho, \nu \,.\, \mathscr{A}[\![\mathsf{A}]\!]\rho \neq \mathscr{A}[\![\mathsf{A}]\!]\rho[\mathsf{x}' \leftarrow \nu]\}$$

⟅case analysis⟆

$$= \{\langle \mathsf{x}', \mathsf{x}'\rangle \mid \mathsf{x}' \neq \mathsf{x}\} \cup \{\langle \mathsf{x}', \mathsf{x}\rangle \mid \mathsf{x}' \in \widehat{\mathcal{S}}^{\mathsf{vdep}}_{\exists}[\![\mathsf{A}]\!]\}$$

⟅by defining the functional dependency of an expression A as $\widehat{\mathcal{S}}^{\mathsf{vdep}}_{\exists}[\![\mathsf{A}]\!] \triangleq \{\mathsf{x}' \mid \exists \rho, \nu \,.\, \mathscr{A}[\![\mathsf{A}]\!]\rho \neq \mathscr{A}[\![\mathsf{A}]\!]\rho[\mathsf{x}' \leftarrow \nu]\}$ in (44.41)⟆ □

Potential dependency semantics of the conditional $\texttt{S} ::= \texttt{if (B) S}_t$

$$
\begin{array}{lll}
\widehat{\mathcal{S}}^{\,\text{vdep}}_{\exists}[\![\texttt{S}]\!]\,\ell & = & (\!|\ \ell = \text{at}[\![\texttt{S}]\!] \mathrel{?} \mathbb{1}_{\mathbb{V}} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(a)}\\
 & & \|\ \ell \in \text{in}[\![\texttt{S}_t]\!] \mathrel{?} \widehat{\mathcal{S}}^{\,\text{vdep}}_{\exists}[\![\texttt{S}_t]\!]\,\ell \,\rceil\, \text{nondet}(\texttt{B},\texttt{B}) \qquad\qquad\qquad\qquad \text{(b)}\\
 & & \|\ \ell = \text{after}[\![\texttt{S}]\!] \mathrel{?}\\
 & & \qquad \widehat{\mathcal{S}}^{\,\text{vdep}}_{\exists}[\![\texttt{S}_t]\!]\,\text{after}[\![\texttt{S}_t]\!] \,\rceil\, \text{nondet}(\texttt{B},\texttt{B}) \qquad\qquad\qquad\qquad \text{(c.1)}\\
 & & \qquad\quad \cup\ \mathbb{1}_{\mathbb{V}} \,\rceil\, \text{nondet}(\neg\texttt{B},\neg\texttt{B}) \qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(c.2)}\\
 & & \qquad\qquad \cup\ \text{nondet}(\neg\texttt{B},\neg\texttt{B}) \times \text{mod}[\![\texttt{S}_t]\!] \qquad\qquad\qquad\qquad\qquad \text{(c.3)}\\
 & & \mathrel{:} \varnothing\ |\!) \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(d)}
\end{array}
$$

$$
\begin{array}{rcll}
\text{det}(\texttt{B}_1,\texttt{B}_2) & \subseteq & \{\texttt{x} \mid \forall \rho,\rho' \,.\, (\mathscr{B}[\![\texttt{B}_1]\!]\rho \wedge \mathscr{B}[\![\texttt{B}_2]\!]\rho') \Rightarrow (\rho(\texttt{x}) = \rho'(\texttt{x}))\} & \text{determinacy}\\
\text{nondet}(\texttt{B}_1,\texttt{B}_2) & \supseteq & \mathbb{V} \setminus \text{det}(\texttt{B}_1,\texttt{B}_2) & \text{non-determinacy}
\end{array}
$$

$$
\begin{array}{rcl}
\text{mod}[\![\texttt{x = E ;}]\!] & \triangleq & \{\texttt{x}\} \qquad\qquad\qquad\qquad \text{modified variables}\\
\text{mod}[\![\texttt{;}]\!] \ \triangleq\ \text{mod}[\![\,\epsilon\,]\!] \ \triangleq\ \text{mod}[\![\texttt{break ;}]\!] & \triangleq & \varnothing\\
\text{mod}[\![\texttt{while (B) S}]\!] \ =\ \text{mod}[\![\texttt{if (B) S}]\!] & \triangleq & \text{mod}[\![\texttt{S}]\!]\\
\text{mod}[\![\texttt{if (B) S}_t\ \texttt{else S}_f]\!] & \triangleq & \text{mod}[\![\texttt{S}_t]\!] \cup \text{mod}[\![\texttt{S}_f]\!]\\
\text{mod}[\![\texttt{\{ Sl \}}]\!] & \triangleq & \text{mod}[\![\texttt{Sl}]\!]\\
\text{mod}[\![\texttt{Sl S}]\!] & \triangleq & \text{mod}[\![\texttt{Sl}]\!] \cup \text{mod}[\![\texttt{S}]\!]
\end{array}
$$

- On entry (a), variables in $\mathbb{V}$ only depend upon themselves as specified by the identity relation $\mathbb{1}_{\mathbb{V}}$.
- The reasoning in (b) is that if a variable y depends at ℓ on the initial value of a variable x at $\mathrm{at}[\![\mathtt{S}_t]\!]$, it depends in the same way on that initial value of the variable x at $\mathrm{at}[\![\mathtt{S}]\!]$ since the test B has no side effect.

  However, (b) also takes into account that if $\mathtt{S}_t$ can only be reached for a unique value of the variable x and the branch is not taken for all other values of x then the variable y does not depend on x in $\mathtt{S}_t$ since empty observations are disallowed by vdep.
- (c) determines dependencies after S so compare two possible executions of that statement. In case (c.1) both executions go through the true branch. In case (c.2) both executions go through the false branch, while in case (c.3) the executions take different branches.

- In case (c.1) when the test is true tt for both executions, the executions of the true branch $\mathtt{S}_t$ terminate and control after $\mathtt{S}_t$ reaches the program point after $\mathtt{S}$ (recall that $\mathsf{after}[\![\mathtt{S}_t]\!] = \mathsf{after}[\![\mathtt{S}]\!]$). The dependencies after $\mathtt{S}_t$ propagate after $\mathtt{S}$ but only in case of non-determinism, *e.g*. for variables that are not constant.
- The second case in (c.2) is for those executions for which the test $\mathtt{B}$ is false ff. Variables depend on themselves $\mathsf{at}[\![\mathtt{S}]\!]$ and control moves to $\mathsf{after}[\![\mathtt{S}]\!]$ so that dependencies are the same there, but only for variables that can reach $\mathsf{after}[\![\mathtt{S}]\!]$ with different values on different executions as indicated by the restriction to $\mathsf{nondet}(\neg\mathtt{B}, \neg\mathtt{B})$.
- The third case in (c.3) is for pairs of executions, one through the true branch and the other through the false branch. In that case $\mathtt{y}$ depends on $\mathtt{x}$ only if $\mathtt{x}$ does not force execution to always take the same branch, meaning that $\mathtt{x} \in \mathsf{nondet}(\neg\mathtt{B}, \neg\mathtt{B})$. If $\mathtt{y}$ is not modified by the execution through $\mathtt{S}_t$ then its value after $\mathtt{S}$ is always the same as its value $\mathsf{at}[\![\mathtt{S}]\!]$ (since $\mathtt{y}$ is not modified on the false branch either). In that case changing $\mathtt{y}$ $\mathsf{at}[\![\mathtt{S}]\!]$ would not change $\mathtt{y}$ after $\mathtt{S}$ so that, in that situation, $\mathtt{y}$ does not depend on $\mathtt{x}$. Therefore (c.3) requires that $\mathtt{y} \in \mathsf{mod}[\![\mathtt{S}_t]\!]$.

Note on the potential dependency semantics of the conditional $\mathtt{S} ::= \mathtt{if}\ (\mathtt{B})\ \mathtt{S}_t$

- Empty observations are not taken into account
- ℓ_0 `if (x=0) { y=x;` ℓ_1`}` ℓ_2
  - `y` does not depend on `x` at ℓ_0 neither at ℓ_1
  - `y` depends on `x` at ℓ_2
- As already stated, this is different from D. E. Denning and P. J. Denning, 1977 implicitly allowing for counterfactual multi-values dependency cmvdp.

Potential dependency semantics of the statement list $\mathtt{Sl} ::= \mathtt{Sl}'\ \mathtt{S}$

$$
\begin{aligned}
\widehat{\mathcal{S}}^{\,\mathsf{vdep}}_{\exists}[\![\mathtt{Sl}]\!]\,\ell \quad\triangleq\quad & (\!|\ \ell \in \mathsf{labx}[\![\mathtt{Sl}']\!] \mathrel{?} \widehat{\mathcal{S}}^{\,\mathsf{vdep}}_{\exists}[\![\mathtt{Sl}']\!]\,\ell && \text{(a)}\\
& \|\ \ell \in \mathsf{labx}[\![\mathtt{S}]\!] \setminus \{\mathsf{at}[\![\mathtt{S}]\!]\} \mathrel{?} \\
& \qquad\quad \widehat{\mathcal{S}}^{\,\mathsf{vdep}}_{\exists}[\![\mathtt{Sl}']\!]\,\mathsf{at}[\![\mathtt{S}]\!] \fatsemi \widehat{\mathcal{S}}^{\,\mathsf{vdep}}_{\exists}[\![\mathtt{S}]\!]\,\ell && \text{(b)}\\
& :\ \varnothing\ |\!)
\end{aligned}
$$

Potential dependency semantics of the iteration $\texttt{S} ::= \texttt{while}\ \ell\ (\texttt{B})\ \texttt{S}_b$

$$\widehat{\mathcal{S}}^{\,\text{vdep}}_{\exists}[\![\texttt{S}]\!]\,\ell' \quad = \quad (\mathsf{lfp}^{\dot\subseteq}\ \mathcal{F}^{\text{vdep}}[\![\texttt{while}\ \ell\ (\texttt{B})\ \texttt{S}_b]\!])\ \ell'$$

$$\begin{aligned}
&\mathcal{F}^{\text{vdep}}[\![\texttt{while}\ \ell\ (\texttt{B})\ \texttt{S}_b]\!]\ X\ \ell' \quad = \\
&\quad \llbracket\ \ell' = \ell\ \mathrel{?} \\
&\qquad \mathbb{1}_{\mathbb{V}} \cup \big(X(\ell) \fatsemi (\widehat{\mathcal{S}}^{\,\text{vdep}}_{\exists}[\![\texttt{S}_b]\!]\,\ell\ \rceil\ \mathsf{nondet}(\texttt{B},\texttt{B}))\big) && \text{(a)}\\
&\quad \parallel\ \ell' \in \mathsf{in}[\![\texttt{S}_b]\!]\ \mathrel{?} \\
&\qquad X(\ell) \fatsemi (\widehat{\mathcal{S}}^{\,\text{vdep}}_{\exists}[\![\texttt{S}_b]\!]\,\ell'\ \rceil\ \mathsf{nondet}(\texttt{B},\texttt{B})) && \text{(b)}\\
&\quad \parallel\ \ell' = \mathsf{after}[\![\texttt{S}]\!]\ \mathrel{?} \\
&\qquad X(\ell) \cup \big(X(\ell) \fatsemi (\mathbb{V} \times \mathsf{mod}[\![\texttt{S}_b]\!])\big) \cup && \text{(c)}\\
&\qquad X(\ell) \fatsemi \Big(\Big(\bigcup_{\ell'' \in \mathsf{breaks\text{-}of}[\![\texttt{S}_b]\!]} \widehat{\mathcal{S}}^{\,\text{vdep}}_{\exists}[\![\texttt{S}_b]\!]\,\ell''\Big)\ \rceil\ \mathsf{nondet}(\texttt{B},\texttt{B})\Big) \\
&\quad \fatsemi\ \varnothing\ \rrbracket && \text{(d)}
\end{aligned}$$

Example

$$\texttt{S} \quad = \quad \texttt{while}\ \ell_0\ (\mathtt{tt})\ \{\ \ell_1\ \texttt{y = z ;}\ell_2\ \texttt{z = x ; }\}\ell_3.$$

The system of equations $X = \mathscr{F}^{\mathrm{d}}[\![\mathsf{S}]\!](X)$ is

$$\begin{cases} X(\ell_0) & = & \{\langle \mathsf{v},\ \mathsf{v}\rangle \mid \mathsf{v} \in \mathbb{V}\} \cup (X(\ell_2) \mathbin{\mathring{9}} \{\langle \mathsf{x},\ \mathsf{x}\rangle, \langle \mathsf{x},\ \mathsf{z}\rangle, \langle \mathsf{y},\ \mathsf{y}\rangle\}) \\ X(\ell_1) & = & X(\ell_0) \\ X(\ell_2) & = & X(\ell_2) \cup (X(\ell_1) \mathbin{\mathring{9}} \{\langle \mathsf{x},\ \mathsf{x}\rangle, \langle \mathsf{z},\ \mathsf{y}\rangle, \langle \mathsf{z},\ \mathsf{z}\rangle\}) \\ X(\ell_3) & = & \varnothing \end{cases}$$

The chaotic iterations are

ℓ	ℓ_0, ℓ_1	ℓ_2	ℓ_3
$X^0(\ell)$	$\varnothing$	$\varnothing$	$\varnothing$
$X^1(\ell)$	$\{\langle \mathsf{x},\ \mathsf{x}\rangle, \langle \mathsf{y},\ \mathsf{y}\rangle, \langle \mathsf{z},\ \mathsf{z}\rangle\}$	$\{\langle \mathsf{x},\ \mathsf{x}\rangle, \langle \mathsf{z},\ \mathsf{y}\rangle, \langle \mathsf{z},\ \mathsf{z}\rangle\}$	$\varnothing$
$X^2(\ell)$	$\{\langle \mathsf{x},\ \mathsf{x}\rangle, \langle \mathsf{x},\ \mathsf{z}\rangle, \langle \mathsf{y},\ \mathsf{y}\rangle, \langle \mathsf{z},\ \mathsf{y}\rangle, \langle \mathsf{z},\ \mathsf{z}\rangle\}$	$\{\langle \mathsf{x},\ \mathsf{x}\rangle, \langle \mathsf{x},\ \mathsf{y}\rangle, \langle \mathsf{x},\ \mathsf{z}\rangle, \langle \mathsf{z},\ \mathsf{y}\rangle, \langle \mathsf{z},\ \mathsf{z}\rangle\}$	$\varnothing$
$X^3(\ell)$	$\{\langle \mathsf{x},\ \mathsf{x}\rangle, \langle \mathsf{x},\ \mathsf{y}\rangle, \langle \mathsf{x},\ \mathsf{z}\rangle, \langle \mathsf{y},\ \mathsf{y}\rangle, \langle \mathsf{z},\ \mathsf{y}\rangle, \langle \mathsf{z},\ \mathsf{z}\rangle\}$	$\{\langle \mathsf{x},\ \mathsf{x}\rangle, \langle \mathsf{x},\ \mathsf{y}\rangle, \langle \mathsf{x},\ \mathsf{z}\rangle, \langle \mathsf{z},\ \mathsf{y}\rangle, \langle \mathsf{z},\ \mathsf{z}\rangle\}$	$\varnothing$
$X^4(\ell)$	$X^3(\ell_0) = X^3(\ell_1)$	$X^3(\ell_2)$	$\varnothing$

- The initial value x_0 of x flows to x at ℓ_0 on iteration entry, to z after the first iteration and so to y after the first iteration.
- The initial value y_0 of y flows only to y at ℓ_0 on iteration entry.
- The initial value z_0 of z flows to z at ℓ_0 on iteration entry and then to y after the first iteration.

The potential dependency semantics is not purely structural [5]

- Separate analysis of statements:

ℓ_0 y = x ;
ℓ_1

x and y at ℓ_1 depend on x at ℓ_0.

ℓ_1 y = y - x ;
ℓ_2

x and y at ℓ_2 depend on x at ℓ_1.

- Dependency analysis of the statement list:

ℓ_0 y = x ;
ℓ_1 y = y - x ;
ℓ_2

y at ℓ_2 depends on x at ℓ_1 which depends on x at ℓ_0 so, by composition, y at ℓ_2 depends on x at ℓ_0.

- Yet, $y = 0$ at ℓ_2 and so y at ℓ_2 do not depend on x at ℓ_0.
- A purely syntactic structural definition of dependency like $\widehat{\mathcal{S}}_{\exists}^{\text{vdep}}[\![\mathsf{S}]\!]$ is necessarily imprecise (since values of variables are not taken into account)

[5] one would say compositional in denotational semantics.

Improving precision

- To be more precise, values of variables must be taken into account
- Reduced product with a reachability analysis (for example Cortesi, Ferrara, Halder, and Zanioli, 2018; Zanioli and Cortesi, 2011)

Examples of derived dependency semantics and analyzes

Dye instrumented semantics

Postulated definition of dependency (I)

- dye-tracer tests in hydrology: determine the possible origins of spring discharges or resurgences by water source coloring and flow tracing
- dye instrumented semantics: decorate the initial values of variables with labels such as color annotations and to track their diffusion and mixtures to determine dependencies Cheney, Ahmed, and Acar, 2011.

Postulated definition of dependency (II)

- This postulated definition of dependency can be proved sound by observing that the initial color of variables can be designated by the name of these variables and that the color mix at point ℓ for variable y is

$$\{\mathsf{x} \mid \mathcal{S}^{+\infty}[\![\mathsf{P}]\!] \in \mathcal{D}_{\mathsf{dep}}{}^{\ell}\langle\mathsf{x},\ \mathsf{y}\rangle\}$$

- Note that in the postulated instrumented semantics, the choice of dep remains implicit as defined by the arbitrarily selected color mixing rules.
- Like all instrumented semantics Jones and Nielson, 1995, it must be semantically justified with respect to the non-instrumented semantics, in which case the non-instrumented semantics can be used as well to justify dependency, as we do.

Tracking analysis

- Assume the initial values of variables (more generally inputs) are partitioned into tracked $\mathbb{T}$ and untracked $\mathbb{U}$ variables,

$$\mathbb{V} = \mathbb{T} \cup \mathbb{U} \text{ and } \mathbb{T} \cap \mathbb{U} = \varnothing$$

- The tracking abstraction $\alpha^{\tau}(\mathbf{D})$ of a dependency property $\mathbf{D} \in \mathbb{L} \rightarrow \wp(\mathbb{V} \times \mathbb{V})$ attaches to each program point ℓ the set of variables y which, at that program point ℓ, depend upon the initial value of at least one tracked variable $\mathsf{x} \in \mathbb{T}$.

$$\alpha^{\tau}(\mathbf{D})^{\ell} \quad \triangleq \quad \{\mathsf{y} \mid \exists \mathsf{x} \in \mathbb{T} \;.\; \langle \mathsf{x},\ \mathsf{y} \rangle \in \mathbf{D}(\ell)\}$$

- A tracking analysis is an over-approximation of the abstract tracking semantics

$$\mathcal{S}^{\tau}[\![\mathsf{S}]\!] \quad \supseteq \quad \alpha^{\tau}(\alpha^{\mathsf{dep}}(\{\mathcal{S}^{+\infty}[\![\mathsf{S}]\!]\}))$$

assigning the each program point ℓ, a set $\mathcal{S}^{\tau}[\![\mathsf{S}]\!]^{\ell} \in \wp(\mathbb{V})$ of variables potentially depending on tracked variables.

Examples of tracking analyses

- taint analysis in privacy/security checks Ferrara, Olivieri, and Spoto, 2018; Li, Bissyandé, Papadakis, Rasthofer, Bartel, Octeau, Klein, and Traon, 2017 (tracked is tainted, untracked is untainted);
- binding time analysis in offline partial evaluation Hatcliff, 1998; Jones, Sestoft, and Søndergaard, 1989 (tracked is dynamic, untracked is static)
- absence of interference Bowman and Ahmed, 2015; Cohen, 1977; Goguen and Meseguer, 1982, 1984; Volpano, Irvine, and Smith, 1996 (tracked is high (private/untrusted), untracked is low (public/trusted)).

Conclusion

Dependency is an abstract interpretation of the program semantics

- Dependency analysis is an abstract interpretation of the program semantics
- This include non-interference, “taint” analysis, *etc*.
- Data dependency analysis to detect parallelism in sequential codes Padua and Wolfe, 1986 is also an abstract interpretation Tzolovski, 1997, Tzolovski, 2002, Ch. 5.
- We have considered particular cases of dependency.

Conjecture: all dependencies are abstract interpretations

- The semantics is a set of computations $\langle \pi\ell, \ell\pi' \rangle$ (where $\ell \notin \pi$).
- We define an abstraction of the past $\pi\ell$ (the initial state in our case)
- We define an abstraction of the future (the sequence of values of a variable y observées dans $\ell\pi'$ à each point ℓ dans $\ell\pi'$).
- We define a difference on pasts (changing the value of only one variable in our case)
- We define a difference on futures (tdep, ctdep, vdep or cvdep in our case)
- Dependency is then the future abstraction depends on the past abstraction iff a change of the past changing its abstraction change the abstraction of the future.
- By varying abstractions and the difference we change the notions of dependency (and we should be able to recover the whole literature in that way).
- Good examples are Giacobazzi and Mastroeni, 2018 for non-interference and Barthe, Grégoire, and Laporte, 2017 for the protection against side channels attacks

Bibliography on dependency

References I

Assaf, Mounir, David A. Naumann, Julien Signoles, Eric Totel, and Frédéric Tronel (2017). "Hypercollecting semantics and its application to static analysis of information flow". In: *POPL*. ACM, pp. 874–887 (176).

Barthe, Gilles, Benjamin Grégoire, and Vincent Laporte (2017). "Provably secure compilation of side-channel countermeasures". *IACR Cryptology ePrint Archive* 2017, p. 1233 (227).

Bowman, William J. and Amal Ahmed (2015). "Noninterference for free". In: *ICFP*. ACM, pp. 101–113 (224).

Cheney, James, Amal Ahmed, and Umut A. Acar (2011). "Provenance as dependency analysis". *Mathematical Structures in Computer Science* 21.6, pp. 1301–1337 (176, 220).

Cohen, Ellis S. (1977). "Information Transmission in Computational Systems". In: *SOSP*. ACM, pp. 133–139 (179, 224).

References II

Cortesi, Agostino, Pietro Ferrara, Raju Halder, and Matteo Zanioli (2018). "Combining Symbolic and Numerical Domains for Information Leakage Analysis". *Trans. Computational Science* 31, pp. 98–135 (217).

Cousot, Patrick and Radhia Cousot (2009). "Bi-inductive structural semantics". *Inf. Comput.* 207.2, pp. 258–283 (189).

Denning, Dorothy E. and Peter J. Denning (1977). "Certification of Programs for Secure Information Flow". *Commun. ACM* 20.7, pp. 504–513 (176, 182, 185, 186, 194, 195, 202, 212).

Ferrara, Pietro, Luca Olivieri, and Fausto Spoto (June 2018). "Tailoring Taint Analysis to GDPR". In: *Privacy Technologies and Policy*. 6th Annual Privacy Forum, APF 2018, Barcelona, Spain, June 13-14, 2018, Revised Selected Papers. DOI: 10.1007/978-3-030-02547-2_4 (224).

Giacobazzi, Roberto and Isabella Mastroeni (2018). "Abstract Non-Interference: A Unifying Framework for Weakening Information-flow". *ACM Trans. Priv. Secur.* 21.2, 9:1–9:31 (227).

References III

Goguen, Joseph A. and José Meseguer (1982). “Security Policies and Security Models”. In: *IEEE Symposium on Security and Privacy*. IEEE Computer Society, pp. 11–20 (179, 224).

– (1984). “Unwinding and Inference Control”. In: *IEEE Symposium on Security and Privacy*. IEEE Computer Society, pp. 75–87 (179, 224).

Hatcliff, John (1998). “An Introduction to Online and Offline Partial Evaluation using a Simple Flowchart Language”. In: *Partial Evaluation*. Vol. 1706. Lecture Notes in Computer Science. Springer, pp. 20–82 (224).

Jones, Neil D. and Flemming Nielson (1995). “Abstract Interpretation: a Semantics-Based Tool for Program Analysis”. In: Samson Abramsky and Dov M. Gabbay, eds. *Handbook of Logic in Computer Science*. Vol. 4, Semantic Modelling. Oxford University Press, pp. 527–636 (221).

Jones, Neil D., Peter Sestoft, and Harald Søndergaard (1989). “Mix: A Self-Applicable Partial Evaluator for Experiments in Compiler Generation”. *Lisp and Symbolic Computation* 2.1, pp. 9–50 (224).

References IV

Jourdan, Jacques-Henri, Vincent Laporte, Sandrine Blazy, Xavier Leroy, and David Pichardie (2015). "A Formally-Verified C Static Analyzer". In: *POPL*. ACM, pp. 247–259 (204).

Lampson, Butler W. (1973). "A Note on the Confinement Problem". *Commun. ACM* 16.10, pp. 613–615 (186).

Li, Li, Tegawendé F. Bissyandé, Mike Papadakis, Siegfried Rasthofer, Alexandre Bartel, Damien Octeau, Jacques Klein, and Yves Le Traon (2017). "Static analysis of Android apps: A systematic literature review". *Information & Software Technology* 88, pp. 67–95 (224).

Mantel, Heiko (July 2003). "A Uniform Framework for the Formal Specification and Verification of Information Flow Security". Dr.-Ing. Thesis. Saarbrücken, Germany: Fakultät I der Universität des Saarlandes (179).

Mulder, Elke De, Thomas Eisenbarth, and Patrick Schaumont (2018). "Identifying and Eliminating Side-Channel Leaks in Programmable Systems". *IEEE Design & Test* 35.1, pp. 74–89 (186).

References V

Padua, David A. and Michael Wolfe (1986). “Advanced Compiler Optimizations for Supercomputers”. *Commun. ACM* 29.12, pp. 1184–1201 (226).

Russo, Alejandro, John Hughes, David A. Naumann, and Andrei Sabelfeld (2006). “Closing Internal Timing Channels by Transformation”. In: *ASIAN*. Vol. 4435. Lecture Notes in Computer Science. Springer, pp. 120–135 (186).

Sabelfeld, Andrei and Andrew C. Myers (2003). “Language-based information-flow security”. *IEEE Journal on Selected Areas in Communications* 21.1, pp. 5–19 (186).

Tzolovski, Stanislav (1997). “Data Dependence as Abstract Interpretations”. In: *SAS*. Vol. 1302. Lecture Notes in Computer Science. Springer, p. 366 (226).

– (15 June 2002). “Raffinement d’analyses par interprétation abstraite”. Thèse de doctorat. Palaiseau, France: École polytechnique (226).

Urban, Caterina and Peter Müller (2018). “An Abstract Interpretation Framework for Input Data Usage”. In: *ESOP*. Vol. 10801. Lecture Notes in Computer Science. Springer, pp. 683–710 (176).

References VI

Volpano, Dennis M., Cynthia E. Irvine, and Geoffrey Smith (1996). "A Sound Type System for Secure Flow Analysis". *Journal of Computer Security* 4.2/3, pp. 167–188 (224).

Zanioli, Matteo and Agostino Cortesi (2011). "Information Leakage Analysis by Abstract Interpretation". In: *SOFSEM*. Vol. 6543. Lecture Notes in Computer Science. Springer, pp. 545–557 (217).

The End, Thank you

Appendix