Abstract Interpretation IV

Semantics and Application to Program Verification

Antoine Miné

École normale supérieure, Paris

year 2015–2016

Course 13

27 May 2016

Overview

Selected advanced topics:

- reduced products of abstract domains
- disjunctive abstract domains
- inter-procedural analysis

Practical session:

- implement a reduced product
- help with the project

Reduced products

Idea

Theory:

- the set of abstract domains is a lattice,
- ordered by abstraction, which is a partial order, i.e.:
  $(C, \leq) \xleftarrow[\alpha]{\gamma} (A, \sqsubseteq)$ denotes that C is more concrete than A.
  (every property of A can also be represented exactly in C)
- there is a least upper bound $\sqcup$ for arbitrary sets of domains and a greatest lower bound $\sqcap$.

Application: reduced product

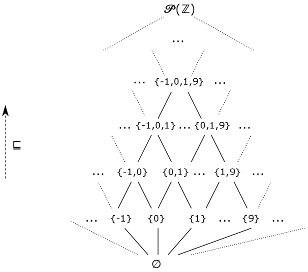
Effective construction for the least upper bound $A_1 \sqcap A_2$, able to represent properties expressible in either A_1 or A_2

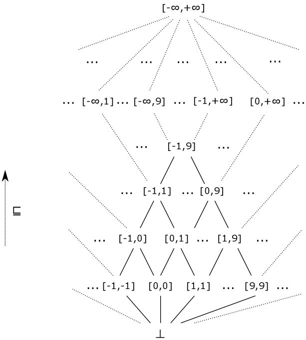
Benefit

We can design **more precise** analyses by combining existing abstractions

Abstract domain lattice

Reminder: interval abstraction

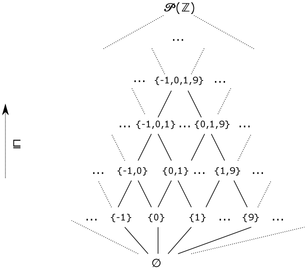


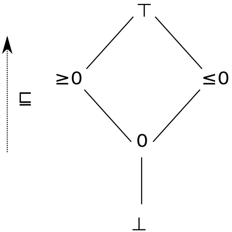


$$\mathcal{P}(\mathbb{Z}) \underset{\alpha_i}{\overset{\gamma_i}{\leftrightarrows}} \{\, [a,b] \mid a \le b \,\} \cup \{\bot\}$$

- $\alpha_i(S) \stackrel{\text{def}}{=} [\min S, \max S]$
- $\gamma_i([a,b]) \stackrel{\text{def}}{=} \{\, x \in \mathbb{Z} \mid a \le x \le b \,\}$

Reminder: sign abstraction



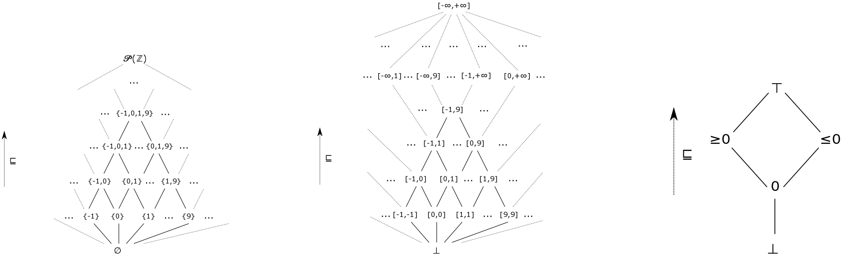


$$\mathcal{P}(\mathbb{Z}) \;\underset{\alpha_s}{\overset{\gamma_s}{\leftrightarrows}}\; \{\bot, 0, \leq 0, \geq 0, \top\}$$

$$\begin{array}{lcl}
\gamma_s(\bot) & \stackrel{\text{def}}{=} & \emptyset \\
\gamma_s(0) & \stackrel{\text{def}}{=} & \{0\} \\
\gamma_s(\geq 0) & \stackrel{\text{def}}{=} & \mathbb{N} \\
\gamma_s(\leq 0) & \stackrel{\text{def}}{=} & -\mathbb{N} \\
\gamma_s(\top) & \stackrel{\text{def}}{=} & \mathbb{Z}
\end{array}$$

$$\alpha_s(S) \stackrel{\text{def}}{=} \begin{cases} \bot & \text{if } S = \emptyset \\ 0 & \text{if } S = \{0\} \\ \geq 0 & \text{else, if } \forall s \in S,\ s \geq 0 \\ \leq 0 & \text{else, if } \forall s \in S,\ s \leq 0 \\ \top & \text{otherwise} \end{cases}$$

Composing abstractions



$$\mathcal{P}(\mathbb{Z}) \quad \underset{\alpha_i}{\overset{\gamma_i}{\leftrightarrows}} \quad \{\, [a,b] \mid a \le b \,\} \cup \{\bot\} \quad \underset{\alpha'_s}{\overset{\gamma'_s}{\leftrightarrows}} \quad \{\bot, 0, \le 0, \ge 0, \top\}$$

where: $\gamma'_s(\bot) \overset{\text{def}}{=} \bot \qquad \gamma'_s(\top) \overset{\text{def}}{=} [-\infty,+\infty]$

$\gamma'_s(\ge 0) \overset{\text{def}}{=} [0,+\infty] \qquad \gamma'_s(\le 0) \overset{\text{def}}{=} [-\infty,0] \qquad \gamma'_s(0) \overset{\text{def}}{=} [0,0]$

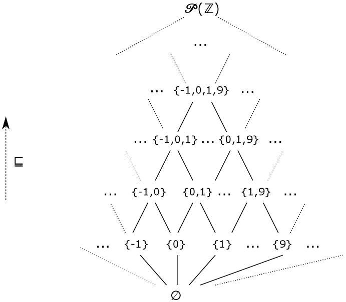
We can compose Galois connections:

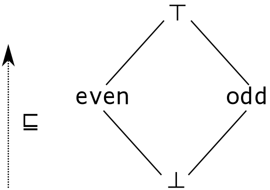
If $(X_1,\sqsubseteq_1) \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} (X_2,\sqsubseteq_2) \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} (X_3,\sqsubseteq_3)$, then

$(X_1,\sqsubseteq_1) \underset{\alpha_2\circ\alpha_1}{\overset{\gamma_1\circ\gamma_2}{\leftrightarrows}} (X_3,\sqsubseteq_3)$.

Proof: $(\alpha_2 \circ \alpha_1)(c) \sqsubseteq_3 a \iff \alpha_1(c) \sqsubseteq_2 \gamma_2(a) \iff c \sqsubseteq_1 (\gamma_1 \circ \gamma_2)(a)$

Parity domain



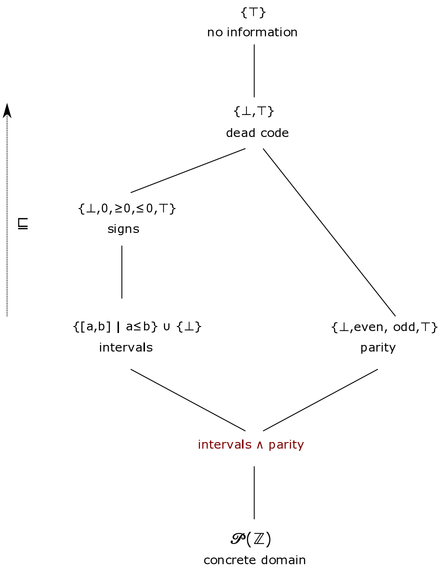


$$\mathcal{P}(\mathbb{Z}) \underset{\alpha_p}{\overset{\gamma_p}{\leftrightarrows}} \{\bot, \top, \mathit{even}, \mathit{odd}\}$$

$$\begin{array}{lcl}
\gamma_p(\bot) & \stackrel{\text{def}}{=} & \emptyset \\
\gamma_p(\mathit{even}) & \stackrel{\text{def}}{=} & 2\mathbb{Z} \\
\gamma_p(\mathit{odd}) & \stackrel{\text{def}}{=} & 2\mathbb{Z}+1 \\
\gamma_p(\top) & \stackrel{\text{def}}{=} & \mathbb{Z}
\end{array}$$

$$\alpha_p(S) \stackrel{\text{def}}{=} \begin{cases} \bot & \text{if } S = \emptyset \\ \mathit{even} & \text{else, if } S \subseteq 2\mathbb{Z} \\ \mathit{odd} & \text{else, if } S \subseteq 2\mathbb{Z}+1 \\ \top & \text{otherwise} \end{cases}$$

Part of the abstraction lattice for $\mathcal{P}(\mathbb{Z})$



- signs are more abstract than intervals;

  every sign information can be expressed as an interval

- parities and intervals are incomparable;

  no common property, except $\bot$ and $\top$

- $\mathcal{P}(\mathbb{Z})$ is the most concrete domain;
- $\{\top\}$ is the most abstract domain;
- intervals $\wedge$ parities is the coarsest abstract domain more precise than intervals and parities.

Reduced product construction

Simple product

Algebraic structure:

Given two domains $(\mathcal{D}_1^\sharp, \sqsubseteq_1)$ and $(\mathcal{D}_2^\sharp, \sqsubseteq_2)$, we use pairs of abstract elements to represent conjunctions of properties.

- $\mathcal{D}_{1\times 2}^\sharp \stackrel{\text{def}}{=} \mathcal{D}_1^\sharp \times \mathcal{D}_2^\sharp$
- $\gamma_{1\times 2}(X_1^\sharp, X_2^\sharp) \stackrel{\text{def}}{=} \gamma_1(X_1^\sharp) \cap \gamma_2(X_2^\sharp)$
- $\alpha_{1\times 2}(S) \stackrel{\text{def}}{=} (\alpha_1(S), \alpha_2(S))$
- $(X_1^\sharp, X_2^\sharp) \sqsubseteq_{1\times 2} (Y_1^\sharp, Y_2^\sharp) \stackrel{\text{def}}{\Longleftrightarrow} X_1^\sharp \sqsubseteq_1 Y_1^\sharp$ and $X_2^\sharp \sqsubseteq_2 Y_2^\sharp$

Abstract operators in $\mathcal{D}^\sharp$:

Applied in parallel (independently) in each abstract domain:

- $(X_1^\sharp, X_2^\sharp) \cup_{1\times 2}^\sharp (Y_1^\sharp, Y_2^\sharp) \stackrel{\text{def}}{=} (X_1^\sharp \cup_1^\sharp Y_1^\sharp,\ X_2^\sharp \cup_2^\sharp Y_2^\sharp)$, ;
- $(X_1^\sharp, X_2^\sharp) \triangledown_{1\times 2} (Y_1^\sharp, Y_2^\sharp) \stackrel{\text{def}}{=} (X_1^\sharp \triangledown_1 Y_1^\sharp,\ X_2^\sharp \triangledown_2 Y_2^\sharp)$;
- $\mathsf{S}^\sharp[\![s]\!]_{1\times 2}(X_1^\sharp, X_2^\sharp) \stackrel{\text{def}}{=} (\mathsf{S}^\sharp[\![s]\!]_1(X_1^\sharp),\ \mathsf{S}^\sharp[\![s]\!]_2(X_2^\sharp))$.

Simple products: limitations

$V \leftarrow 1;$
while $V \leq 10$ **do** $V \leftarrow V + 2$ **done**;
$\bullet$ **if** $V \geq 12$ **then** $\bullet$ $V \leftarrow 0$ $\bullet$;

Analysis in the product domain of intervals and parities:

	intervals	parities	product: intervals $\times$ parities
$\bullet$	$V \in [11, 12]$	V odd	$(V \in [11, 12]) \wedge (V \text{ odd})$
$\bullet$	$V = 12$	V odd	$(V = 12) \wedge (V \text{ odd})$
$\bullet$	$V = 0$	V even	$(V = 0) \wedge (V \text{ even})$

Identical to two separate analyses:

- at $\bullet$, we get $(V = 12) \wedge (V \text{ odd})$, which represents $\emptyset$;
- at $\bullet$, we apply $V \leftarrow 0$ independently on intervals and parities, which gives $(V = 0) \wedge (V \text{ even})$, instead of $\emptyset$!

$\Longrightarrow$ huge loss of precision

Fully reduced product

Idea: propagate information between domains

Given Galois connections (α_1, γ_1) and (α_2, γ_2) over $\mathcal{D}_1^\sharp$ and $\mathcal{D}_2^\sharp$,
we define a reduction operator ρ as:

$$\rho : \mathcal{D}^\sharp_{1\times 2} \to \mathcal{D}^\sharp_{1\times 2}$$
$$\rho(X_1^\sharp, X_2^\sharp) \stackrel{\text{def}}{=} (\alpha_1(\gamma_1(X_1^\sharp) \cap \gamma_2(X_2^\sharp)),\ \alpha_2(\gamma_1(X_1^\sharp) \cap \gamma_2(X_2^\sharp)))$$

i.e., the best representation of $\gamma_{1\times 2}(X_1^\sharp, X_2^\sharp)$ in both domains

Application:

use ρ to transfer information between domains after abstract operations:

- $(X_1^\sharp, X_2^\sharp) \cup^\sharp_{1\times 2} (Y_1^\sharp, Y_2^\sharp) \stackrel{\text{def}}{=} \rho(X_1^\sharp \cup_1^\sharp Y_1^\sharp,\ X_2^\sharp \cup_2^\sharp Y_2^\sharp)$,
- $\mathsf{S}^\sharp[\![s]\!]_{1\times 2}(X_1^\sharp, X_2^\sharp) \stackrel{\text{def}}{=} \rho(\mathsf{S}^\sharp[\![s]\!]_1(X_1^\sharp),\ \mathsf{S}^\sharp[\![s]\!]_2(X_2^\sharp))$.

Warning:

ρ should not be used on fixpoint iterates with widening $\triangledown$ $(X_{n+1} \stackrel{\text{def}}{=} \rho(X_n \triangledown F(X_n)))$
$\Longrightarrow$ this could prevent the convergence in $\mathcal{D}_1^\sharp \times \mathcal{D}_2^\sharp$!

Analysis with reduction: exemple

$$
\begin{array}{l}
\quad V \leftarrow 1; \\
\quad \textbf{while } V \leq 10 \textbf{ do } V \leftarrow V + 2 \textbf{ done}; \\
\bullet \textbf{ if } V \geq 12 \textbf{ then } V \leftarrow 0
\end{array}
$$

Reduction ρ between intervals and parities $\rho([a,b],p)$:
First refine interval bounds $[a,b]$ using parity information p, then refine the parity information using the refined bounds:

- let $a' = a + 1$ if $a \notin \gamma_p(p)$, $a' = a$ otherwise;
- let $b' = b - 1$ if $b \notin \gamma_p(p)$, $b' = b$ otherwise;
- if $a' > b'$, return $(\perp, \perp)$;
- if $a' = b'$, return $([a', b'], \alpha_p(a))$;
- otherwise, return $([a', b'], p)$.

Example:

At $\bullet$, $\rho([11,12],\text{odd}) = ([11,11],\text{odd})$
$\Longrightarrow$ the "then" branch is not reachable.

Partial reduction

The optimal reduction ρ is well-defined but:

- ρ assumes we have Galois connections;
- there is no general effective algorithm to compute ρ.

  (similar to the case of optimal operators, defined as $F^\sharp \stackrel{\text{def}}{=} \alpha \circ F \circ \gamma$)

Partial reduction:

Practical definition, when the optimal reduction is not available:

- $\rho(X_1^\sharp, X_2^\sharp) = (Y_1^\sharp, Y_2^\sharp)$ is a partial reduction if:
- $Y_1^\sharp \sqsubseteq_1 X_1^\sharp$ and $Y_2^\sharp \sqsubseteq_2 X_2^\sharp$ (improvement)
- $\gamma_{1\times 2}(Y_1^\sharp, Y_2^\sharp) = \gamma_{1\times 2}(X_1^\sharp, X_2^\sharp)$ (soundness)

Example:

$$\rho(X_1^\sharp, X_2^\sharp) \stackrel{\text{def}}{=} \begin{cases} (\bot_1, \bot_2) & \text{if } X_1^\sharp = \bot_1 \text{ or } X_2^\sharp = \bot_2 \\ (X_1^\sharp, X_2^\sharp) & \text{otherwise} \end{cases}$$

In practice, an analyzer contains many abstract domains (for expressiveness) with limited reductions between them (for efficiency).

Disjunctive domains

Motivation

Remark: most domains abstract **convex sets** (conjunctions of constraints)

$\Longrightarrow \cup^\sharp$ causes a loss of precision!

The need for non-convex invariants

$X \leftarrow$ **rand**$(10, 20)$;
$Y \leftarrow$ **rand**$(0, 1)$;
if $Y > 0$ **then** $X \leftarrow -X$;
• $Z \leftarrow 100/X$

Concrete semantics:

At •, $X \in [-20, -10] \cup [10, 20]$
$\Longrightarrow$ there is no division by zero

Abstract analysis:

Convex analyses (intervals, polyhedra) will find $X \in [-20, 20]$
(with intervals, $[-20, -10] \cup^\sharp [10, 20] = [-20, 20]$)
$\Longrightarrow$ possible division by zero (false alarm)

Disjunctive domains

Principle:

generic constructions to lift any numeric abstract domain to a domain able to represent disjunctions exactly

Example constructions:

- powerset completion
  unordered "soup" of abstract elements
- state partitioning
  abstract elements keyed to selected subsets of environments
- path-sensitive analyses
  partition with respect to the history of execution

each construction has its strength and weakness
they can be combined during an analysis to exploit the best of each

Powerset completion

Powerset completion

Given: $(\mathcal{E}^\sharp, \sqsubseteq, \gamma, \cup^\sharp, \cap^\sharp, \triangledown, \mathsf{S}^\sharp[\![\, stat \,]\!])$

abstract domain $\mathcal{E}^\sharp$
ordered by $\sqsubseteq$, which also acts as a sound abstraction of $\subseteq$ (i.e., $\subseteq^\sharp = \sqsubseteq$)
with concretization $\gamma : \mathcal{E}^\sharp \to \mathcal{P}(\mathcal{E})$
sound abstractions $\cup^\sharp$, $\cap^\sharp$, $\mathsf{S}^\sharp[\![\, stat \,]\!]$ of $\cup$, $\cap$, $\mathsf{S}[\![\, stat \,]\!]$, and a widening $\triangledown$

Construct: $(\hat{\mathcal{E}}^\sharp, \hat{\sqsubseteq}, \hat{\gamma}, \hat{\cup}^\sharp, \hat{\cap}^\sharp, \hat{\triangledown}, \hat{\mathsf{S}}^\sharp[\![\, stat \,]\!])$

- $\hat{\mathcal{E}}^\sharp \stackrel{\text{def}}{=} \mathcal{P}_{finite}(\mathcal{E}^\sharp)$ (finite sets of abstract elements)
- $\hat{\gamma}(A^\sharp) \stackrel{\text{def}}{=} \cup \{\, \gamma(X^\sharp) \mid X^\sharp \in A^\sharp \,\}$ (join of concretizations)

Example: using the interval domain as $\mathcal{E}^\sharp$

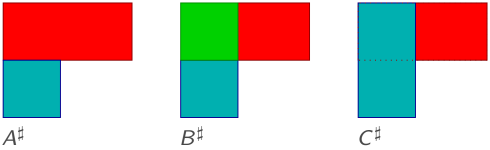
$\hat{\gamma}(\{[-10,-5],[2,4],[0,0],[2,3]\}) = [-10,-5] \cup \{0\} \cup [2,4]$

Ordering

Issue: how can we compare two elements of $\hat{\mathcal{E}}^\sharp$?

- $\hat\gamma$ is generally not injective
  there is no canonical representation for $\hat\gamma(A^\sharp)$
- testing $\hat\gamma(A^\sharp) = \hat\gamma(B^\sharp)$ or $\hat\gamma(A^\sharp) \subseteq \hat\gamma(B^\sharp)$ is difficult

Example: powerset completion of the interval domain



$A^\sharp = \{\{0\} \times \{0\},\ [0,1] \times \{1\}\}$
$B^\sharp = \{\{0\} \times \{0\},\ \{0\} \times \{1\},\ \{1\} \times \{1\}\}$
$C^\sharp = \{\{0\} \times [0,1],\ [0,1] \times \{1\}\}$

$\hat\gamma(A^\sharp) = \hat\gamma(B^\sharp) = \hat\gamma(C^\sharp)$

$B^\sharp$ is more costly to represent: it requires three abstract elements instead of two
$C^\sharp$ is a covering and not a partition (red $\cap$ blue $= \{0\} \times \{1\} \neq \emptyset$)

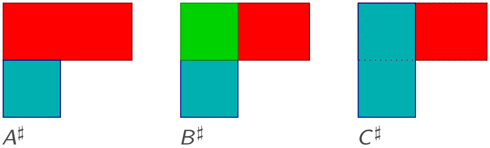
Ordering (cont.)

Solution: sound approximation of $\subseteq$

$A^\sharp \mathrel{\hat{\sqsubseteq}} B^\sharp \stackrel{\text{def}}{\iff} \forall X^\sharp \in A^\sharp : \exists Y^\sharp \in B^\sharp : X^\sharp \sqsubseteq Y^\sharp$ (Hoare powerdomain order)

- $\hat{\sqsubseteq}$ is a partial order (when $\sqsubseteq$ is)
- $\hat{\sqsubseteq}$ is a sound approximation of $\subseteq$ (when $\sqsubseteq$ is)
  $A^\sharp \mathrel{\hat{\sqsubseteq}} B^\sharp \implies \hat{\gamma}(A^\sharp) \subseteq \hat{\gamma}(B^\sharp)$ but the converse may not hold
- testing $\hat{\sqsubseteq}$ reduces to testing $\sqsubseteq$ finitely many times

Example: powerset completion of the interval domain



$\hat{\gamma}(A^\sharp) = \hat{\gamma}(B^\sharp) = \hat{\gamma}(C^\sharp)$

$B^\sharp \mathrel{\hat{\sqsubseteq}} A^\sharp \mathrel{\hat{\sqsubseteq}} C^\sharp$

Abstract operations

Abstract operators

- $\hat{\mathsf{S}}^\sharp[\![\, \mathit{stat}\,]\!]\, A^\sharp \stackrel{\text{def}}{=} \{\, \mathsf{S}^\sharp[\![\, \mathit{stat}\,]\!]\, X^\sharp \mid X^\sharp \in A^\sharp \,\}$

  apply *stat* on each abstract element independently

- $A^\sharp \;\hat{\cup}^\sharp\; B^\sharp \stackrel{\text{def}}{=} A^\sharp \cup B^\sharp$

  keep elements from both arguments without applying any abstract operation
  $\hat{\cup}^\sharp$ is exact

- $A^\sharp \;\hat{\cap}^\sharp\; B^\sharp \stackrel{\text{def}}{=} \{\, X^\sharp \cap^\sharp Y^\sharp \mid X^\sharp \in A^\sharp,\ Y^\sharp \in B^\sharp \,\}$

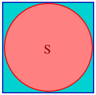
  $\hat{\cap}^\sharp$ is exact if $\cap^\sharp$ is (as $\cup$ and $\cap$ are distributive)

Galois connection:

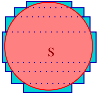
in general, there is no abstraction function $\hat{\alpha}$ corresponding to $\hat{\gamma}$

Example: powerset completion $\hat{\mathcal{E}}^\sharp$ of the interval domain $\mathcal{E}^\sharp$

given the disc $S \stackrel{\text{def}}{=} \{\, (x, y) \mid x^2 + y^2 \leq 1 \,\}$
$\alpha(S) = [-1, 1] \times [-1, 1]$ (optimal interval abstraction)
but there is no best abstraction in $\hat{\mathcal{E}}^\sharp$



$\alpha(S)$



not $\hat{\alpha}(S)$

Dynamic approximation

Issue: the size $|A^\sharp|$ of elements $A^\sharp \in \hat{\mathcal{E}}^\sharp$ is unbounded

every application of $\hat{\cup}^\sharp$ adds some more elements

$\implies$ efficiency and convergence problems

Solution: to reduce the size of elements

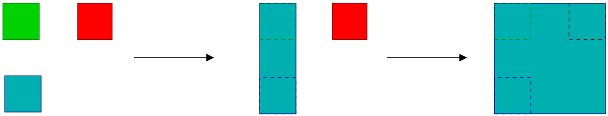
- redundancy removal

  $\textit{simplify}(A^\sharp) \stackrel{\text{def}}{=} \{\, X^\sharp \in A^\sharp \,|\, \forall Y^\sharp \neq X^\sharp \in A^\sharp : X^\sharp \not\sqsubseteq Y^\sharp \,\}$

  no loss of precision: $\hat{\gamma}(\textit{simplify}(A^\sharp)) = \hat{\gamma}(A^\sharp)$

- collapse: join elements in $\mathcal{E}^\sharp$

  $\textit{collapse}(A^\sharp) \stackrel{\text{def}}{=} \{\, \cup^\sharp \{\, X^\sharp \in A^\sharp \,\}\,\}$

  large loss of precision, but very effective: $|\textit{collapse}(A^\sharp)| = 1$

- partial collapse: limit $|A^\sharp|$ to a fixed size k by $\cup^\sharp$

  but how to choose which elements to merge? no easy solution!

Widening

Issue: for loops, abstract iterations $(A^\sharp_n)_{n\in\mathbb{N}}$ may not converge

- the size of $A^\sharp_n$ may grow arbitrarily large
- even if $|A^\sharp_n|$ is stable, some elements in $A^\sharp_n$ may not converge
  if $\mathcal{E}^\sharp$ has infinite increasing sequences

$\Longrightarrow$ we need a widening $\hat{\triangledown}$

Widenings for powerset domains are difficult to design

Example widening: collapse after a fixed number N of iterations

$$A^\sharp_{n+1} \stackrel{\text{def}}{=} A^\sharp_n \mathbin{\hat{\triangledown}} B^\sharp_n \stackrel{\text{def}}{=} \begin{cases} \mathit{simplify}(A^\sharp_n \mathbin{\hat{\cup}^\sharp} B^\sharp_n) & \text{if } n < N \\ \mathit{collapse}(A^\sharp_n) \mathbin{\triangledown} \mathit{collapse}(B^\sharp_n) & \text{otherwise} \end{cases}$$

(this is very naïve, see Bagnara et al. STTT06 for more interesting widenings)

State partitioning

State partitioning

Principle:

- partition *a priori* $\mathcal{E}$ into finitely many sets
- abstract each partition of $\mathcal{E}$ independently using an element of $\mathcal{E}^\sharp$

Abstract domain:

Given an abstract partition $P^\sharp \subseteq \mathcal{E}^\sharp$, i.e., a set such that:

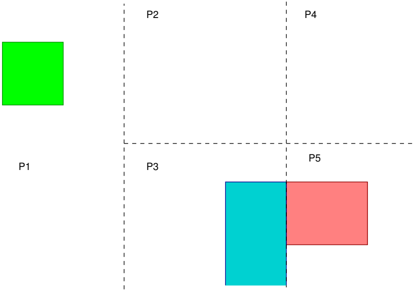
- $P^\sharp$ is finite
- $\cup \{ \gamma(X^\sharp) \mid X^\sharp \in P^\sharp \} = \mathcal{E}$

  for generality, we have in fact a covering, not a partitioning of $\mathcal{E}$
  i.e., we can have $X^\sharp \neq Y^\sharp \in P^\sharp$ with $\gamma(X^\sharp) \cap \gamma(Y^\sharp) \neq \emptyset$

We define $\tilde{\mathcal{E}}^\sharp \stackrel{\text{def}}{=} P^\sharp \to \mathcal{E}^\sharp$

representable in memory, as $P^\sharp$ is finite

Ordering



Example: $\mathcal{E}^\sharp$ is the interval domain

$$
\begin{aligned}
P^\sharp = \{P_1, P_2, P_3, P_4, P_5\} \text{ where } \\
P_1 &= [-\infty, 0] \times [-\infty, +\infty] \\
P_2 &= [0, 10] \times [0, +\infty] \\
P_3 &= [0, 10] \times [-\infty, 0] \\
P_4 &= [10, +\infty] \times [0, +\infty] \\
P_5 &= [10, +\infty] \times [-\infty, 0]
\end{aligned}
$$

$$
\begin{aligned}
X^\sharp = [P_1 &\mapsto [-6, -5] \times [5, 6], \\
P_2 &\mapsto \bot, \\
P_3 &\mapsto [9, 10] \times [-\infty, -1], \\
P_4 &\mapsto \bot, \\
P_5 &\mapsto [10, 12] \times [-3, -1]]
\end{aligned}
$$

- $\tilde{\mathcal{E}}^\sharp \stackrel{\text{def}}{=} P^\sharp \to \mathcal{E}^\sharp$
- $\tilde{\gamma}(A^\sharp) \stackrel{\text{def}}{=} \cup \, \{ \gamma(A^\sharp(X^\sharp)) \cap \gamma(X^\sharp) \,|\, X^\sharp \in P^\sharp \}$
- $A^\sharp \mathrel{\tilde{\sqsubseteq}} B^\sharp \stackrel{\text{def}}{\iff} \forall X^\sharp \in P^\sharp: A^\sharp(X^\sharp) \sqsubseteq B^\sharp(X^\sharp)$ (point-wise order)
- $\tilde{\alpha}(S) \stackrel{\text{def}}{=} \lambda X^\sharp \in P^\sharp . \alpha(S \cap \gamma(X^\sharp))$

  if $\mathcal{E}^\sharp$ enjoys a Galois connection, so does $\tilde{\mathcal{E}}^\sharp$

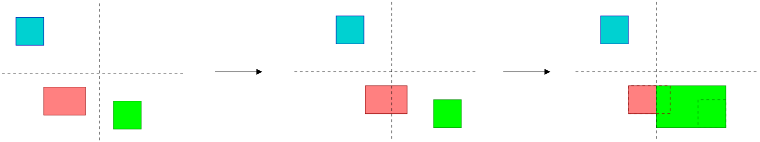
Abstract operators

Abstract operators: point-wise extension from $\mathcal{E}^\sharp$ to $P^\sharp \to \mathcal{E}^\sharp$

- $A \mathrel{\tilde{\cup}^\sharp} B \stackrel{\text{def}}{=} \lambda X^\sharp \in P^\sharp . A(X^\sharp) \cup^\sharp B(X^\sharp)$
- $A \mathrel{\tilde{\cap}^\sharp} B \stackrel{\text{def}}{=} \lambda X^\sharp \in P^\sharp . A(X^\sharp) \cap^\sharp B(X^\sharp)$
- $A \mathrel{\tilde{\triangledown}} B \stackrel{\text{def}}{=} \lambda X^\sharp \in P^\sharp . A(X^\sharp) \triangledown B(X^\sharp)$
- $\tilde{\mathsf{S}}^\sharp[\![\, e \leq 0 ?\,]\!]\, A^\sharp \stackrel{\text{def}}{=} \lambda X^\sharp \in P^\sharp . \mathsf{S}^\sharp[\![\, e \leq 0 ?\,]\!]\, A^\sharp(X^\sharp)$
- $\tilde{\mathsf{S}}^\sharp[\![\, V \leftarrow e \,]\!]\, A^\sharp$ is more complex

  any $\mathsf{S}^\sharp[\![\, V \leftarrow e \,]\!]\, A^\sharp(X^\sharp)$ may escape its partition $X^\sharp$; we must cut them at partition borders and glue the pieces falling into the same partition

example: $X \leftarrow X + 2$



$$\tilde{\mathsf{S}}^\sharp[\![\, V \leftarrow e \,]\!]\, A^\sharp \stackrel{\text{def}}{=} \lambda X^\sharp . \cup^\sharp \{\, X^\sharp \cap^\sharp \mathsf{S}^\sharp[\![\, V \leftarrow e \,]\!]\, A(Y^\sharp) \mid Y^\sharp \in P^\sharp \,\}$$

Example analysis

Example

$X \leftarrow \textbf{rand}(10, 20);$
$Y \leftarrow \textbf{rand}(0, 1);$
$\textbf{if } Y > 0 \textbf{ then } X \leftarrow -X;$
• $Z \leftarrow 100/X$

Analysis:

- $\mathcal{E}^\sharp$ is the interval domain
- partition with respect to the sign of X
  $P^\sharp \stackrel{\text{def}}{=} \{ X^+, X^- \}$ where
  $X^+ \stackrel{\text{def}}{=} [0, +\infty] \times \mathbb{Z} \times \mathbb{Z}$ and $X^- \stackrel{\text{def}}{=} [-\infty, 0] \times \mathbb{Z} \times \mathbb{Z}$
- at • we find:
  $X^+ \mapsto [X \in [10, 20], Y \mapsto [0, 0], Z \mapsto [0, 0]]$
  $X^- \mapsto [X \in [-20, -10], Y \mapsto [1, 1], Z \mapsto [0, 0]]$

  $\Longrightarrow$ no division by zero

Path partitioning

Path sensitivity

Principle: partition wrt. the history of computation

- keep different abstract elements for different execution paths
  e.g., different branches taken, different loop iterations
- avoid merging with $\cup^\sharp$ elements at control-flow joins
  at the end of **if** $\cdots$ **then** $\cdots$ **else**, or at loop head

Intuition: as a program transformation

$X \leftarrow$ **rand**$(-50, 50)$;
if $X \geq 0$ **then**
  $Y \leftarrow X + 10$
else
  $Y \leftarrow X - 10$;
assert $Y \neq 0$

$\longrightarrow$

$X \leftarrow$ **rand**$(-50, 50)$;
if $X \geq 0$ **then**
  $Y \leftarrow X + 10$;
  assert $Y \neq 0$
else
  $Y \leftarrow X - 10$;
  assert $Y \neq 0$

the **assert** is tested in the context of each branch
instead of after the control-flow join

the interval domain can prove the assertion on the right, but not on the left

Abstract domain

Formalization: we consider here only **if** $\cdots$ **then** $\cdots$ **else**

- $\mathcal{L}$ denote syntactic labels of **if** $\cdots$ **then** $\cdots$ **else** instructions
- history abstraction $\mathbb{H} \stackrel{\text{def}}{=} \mathcal{L} \to \{\text{true}, \text{false}, \perp\}$

  $H \in \mathbb{H}$ indicates the outcome of the last time we executed each test:
  - $H(\ell) = \text{true}$: we took the **then** branch
  - $H(\ell) = \text{false}$: we took the **else** branch
  - $H(\ell) = \perp$: we never executed the test

  Notes:

  – $\mathbb{H}$ can remember the outcome of several successive tests
  ℓ_1 : **if** $\cdots$ **then** $\cdots$ **else**; ℓ_2 : **if** $\cdots$ **then** $\cdots$ **else**

  – for tests in loops, $\mathbb{H}$ remembers only the last outcome
  while $\cdots$ **do** ℓ : **if** $\cdots$ **then** $\cdots$ **else**

  – we could extend $\mathbb{H}$ to longer histories with $\mathbb{H} = (\mathcal{L} \to \{\,\text{true}, \text{false}, \perp\,\})^*$
  – we could extend $\mathbb{H}$ to track loop iterations with $\mathbb{H} = \mathcal{L} \to \mathbb{N}$

- $\breve{\mathcal{E}}^\sharp \stackrel{\text{def}}{=} \mathbb{H} \to \mathcal{E}^\sharp$

  use a different abstract element for each abstract history

Abstract operators

- $\breve{\mathcal{E}}^\sharp \stackrel{\text{def}}{=} \mathbb{H} \to \mathcal{E}^\sharp$
- $\breve{\gamma}(A^\sharp) = \cup \{ \gamma(A^\sharp(H)) \mid H \in \mathbb{H} \}$
- $\breve{\sqsubseteq}$, $\breve{\cup}^\sharp$, $\breve{\cap}^\sharp$, $\breve{\triangledown}$ are point-wise
- $\breve{S}^\sharp[\![V \leftarrow e]\!]$ and $\breve{S}^\sharp[\![e \leq 0?]\!]$ are point-wise
- $\breve{S}^\sharp[\![\ell : \textbf{if } c \textbf{ then } s_1 \textbf{ else } s_2]\!] A^\sharp$ is more complex
  - we merge all information about ℓ
    $C^\sharp = \lambda H.A^\sharp(H[\ell \mapsto \text{true}]) \cup^\sharp A^\sharp(H[\ell \mapsto \text{false}]) \cup^\sharp A^\sharp(H[\ell \mapsto \perp])$
  - we compute the then branch, where $H(\ell) = \text{true}$
    $T'^\sharp = \breve{S}^\sharp[\![s_1]\!] (\breve{S}^\sharp[\![c?]\!] T^\sharp)$ where
    $T^\sharp = \lambda H.C^\sharp(H)$ if $H(\ell) = \text{true}$, $\perp$ otherwise
  - we compute the else branch, where $H(\ell) = \text{false}$
    $F'^\sharp = \breve{S}^\sharp[\![s_2]\!] (\breve{S}^\sharp[\![\neg c?]\!] F^\sharp)$ where
    $F^\sharp = \lambda H.C^\sharp(H)$ if $H(\ell) = \text{false}$, $\perp$ otherwise
  - we join both branches: $T'^\sharp \,\breve{\cup}^\sharp\, F'^\sharp$
    the join is exact as $\forall H \in \mathbb{H}$: either $T'^\sharp(H) = \perp$ or $F'^\sharp(H) = \perp$

$\Longrightarrow$ we get a semantic by induction on the syntax of the original program

Complex example

Linear interpolation

$X \leftarrow$ **rand**$(TX[0], TX[N])$;
$I \leftarrow 0$;
while $I < N \wedge X > TX[I+1]$ **do**
    $I \leftarrow I + 1$;
done;
$Y \leftarrow TY[I] + (X - TX[I]) \times TS[I]$

Concrete semantics: table-based interpolation based on the value of X

- look-up index I in the interpolation table: $TX[I] \leq X \leq TX[I+1]$
- interpolate from value $TY[I]$ when $X = TX[I]$ with slope $TS[I]$

Analysis: in the interval domain

- without partitioning:
  $Y \in [\min\ TY, \max\ TY] + (X - [\min\ TX, \max\ TX]) \times [\min\ TS, \max\ TS]$
- partitioning with respect to the number of loop iterations:
  $Y \in \cup_{I \in [0,N]}\ TY[I] + ([0, TX[I+1] - TX[I]]) \times TS[I]$
  more precise as it keeps the relation between table indices

Inter-procedural analyses

Overview

- Analysis on the control-flow graph

  reduce function calls and returns to *gotos*
  useful for the project!

- Inlining

  simple and precise
  but not efficient and may not terminate

- Call-site and call-stack abstraction

  terminates even for recursive programs
  parametric cost-precision trade-off

- Tabulated abstraction

  optimal reuse of analysis partial results

- We also mentioned summary-based abstractions last week, leveraging relational domains for modular bottom-up analysis

in general, these different abstractions give incomparable results;
there is no clear winner

Analysis on the control-flow graph

Inter-procedural control-flow graphs

Extend control-flow graphs:

- one subgraph for each function
- additional arcs to denote function calls and returns

we get one big graph without procedures nor calls, only gotos

$\Longrightarrow$ reduced to a classic analysis based on equation systems

but difficult to use in a denotational-style analysis by induction on the syntax

Note: to simplify, we assume here no local variable and no function argument:

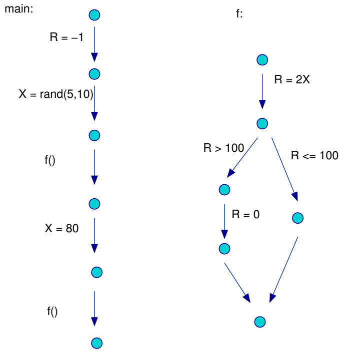
- locals and arguments are transformed into globals
- only possible if there are no recursive calls

Example: Control-flow graph

Example

main :
$R \leftarrow -1;$
$X \leftarrow \textbf{rand}(5, 10); f();$
$X \leftarrow 80; f()$

f :
$R \leftarrow 2 \times X;$
if $R > 100$ **then** $R \leftarrow 0$



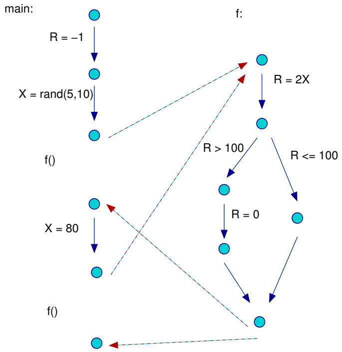
create one control-flow graph for each function

Example: Control-flow graph

Example

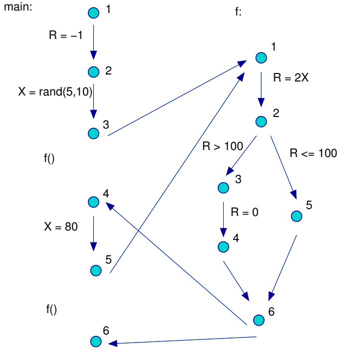
main :
$R \leftarrow -1$;
$X \leftarrow$ **rand**$(5,10)$; $f()$;
$X \leftarrow 80$; $f()$

f :
$R \leftarrow 2 \times X$;
if $R > 100$ **then** $R \leftarrow 0$



replace call instructions with gotos

Example: Equation system

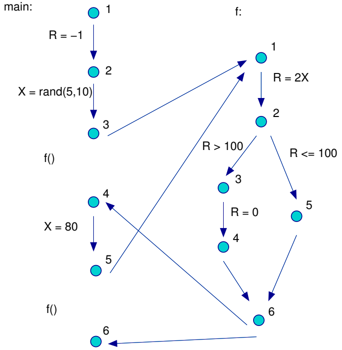


$\mathcal{S}_{\text{main},1} = \top$
$\mathcal{S}_{\text{main},2} = \mathsf{S}[\![\, R \leftarrow 1 \,]\!]\, \mathcal{S}_{\text{main},1}$
$\mathcal{S}_{\text{main},3} = \mathsf{S}[\![\, X \leftarrow \textbf{rand}(5,10) \,]\!]\, \mathcal{S}_{\text{main},2}$
$\mathcal{S}_{\text{main},4} = \mathcal{S}_{\text{f},6}$
$\mathcal{S}_{\text{main},5} = \mathsf{S}[\![\, X \leftarrow 80 \,]\!]\, \mathcal{S}_{\text{main},4}$
$\mathcal{S}_{\text{main},6} = \mathcal{S}_{\text{f},6}$

$\mathcal{S}_{\text{f},1} = \mathcal{S}_{\text{main},3} \cup \mathcal{S}_{\text{main},5}$
$\mathcal{S}_{\text{f},2} = \mathsf{S}[\![\, R \leftarrow 2X \,]\!]\, \mathcal{S}_{\text{f},1}$
$\mathcal{S}_{\text{f},3} = \mathsf{S}[\![\, R > 100 \,]\!]\, \mathcal{S}_{\text{f},2}$
$\mathcal{S}_{\text{f},4} = \mathsf{S}[\![\, R \leftarrow 0 \,]\!]\, \mathcal{S}_{\text{f},3}$
$\mathcal{S}_{\text{f},5} = \mathsf{S}[\![\, R \leq 100 \,]\!]\, \mathcal{S}_{\text{f},2}$
$\mathcal{S}_{\text{f},6} = \mathcal{S}_{\text{f},4} \cup \mathcal{S}_{\text{f},5}$

- each variable $\mathcal{S}_i$ denotes a set of environments at a control location i
- we can derive an abstract version of the system
  e.g.: $\mathcal{S}^\sharp_{\text{f},2} = \mathsf{S}^\sharp[\![\, R \leftarrow 2X \,]\!]\, \mathcal{S}^\sharp_{\text{f},1}$, $\mathcal{S}^\sharp_{\text{f},6} = \mathcal{S}^\sharp_{\text{f},4} \cup^\sharp \mathcal{S}^\sharp_{\text{f},5}$, etc.
- we can solve the abstract system, using widenings to terminate
  c.f. project

Example: Equation system



$$\begin{aligned}
\mathcal{S}_{\text{main},1} &= \top \\
\mathcal{S}_{\text{main},2} &= \mathsf{S}[\![\, R \leftarrow 1 \,]\!]\, \mathcal{S}_{\text{main},1} \\
\mathcal{S}_{\text{main},3} &= \mathsf{S}[\![\, X \leftarrow \textbf{rand}(5,10) \,]\!]\, \mathcal{S}_{\text{main},2} \\
\mathcal{S}_{\text{main},4} &= \mathcal{S}_{\text{f},6} \\
\mathcal{S}_{\text{main},5} &= \mathsf{S}[\![\, X \leftarrow 80 \,]\!]\, \mathcal{S}_{\text{main},4} \\
\mathcal{S}_{\text{main},6} &= \mathcal{S}_{\text{f},6}
\end{aligned}$$

$$\begin{aligned}
\mathcal{S}_{\text{f},1} &= \mathcal{S}_{\text{main},3} \cup \mathcal{S}_{\text{main},5} \\
\mathcal{S}_{\text{f},2} &= \mathsf{S}[\![\, R \leftarrow 2X \,]\!]\, \mathcal{S}_{\text{f},1} \\
\mathcal{S}_{\text{f},3} &= \mathsf{S}[\![\, R > 100 \,]\!]\, \mathcal{S}_{\text{f},2} \\
\mathcal{S}_{\text{f},4} &= \mathsf{S}[\![\, R \leftarrow 0 \,]\!]\, \mathcal{S}_{\text{f},3} \\
\mathcal{S}_{\text{f},5} &= \mathsf{S}[\![\, R \leq 100 \,]\!]\, \mathcal{S}_{\text{f},2} \\
\mathcal{S}_{\text{f},6} &= \mathcal{S}_{\text{f},4} \cup \mathcal{S}_{\text{f},5}
\end{aligned}$$

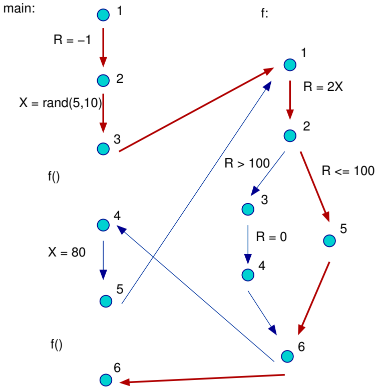
using intervals we get the following solution:

$$\begin{aligned}
\mathcal{S}^\sharp_{\text{main},1} &: X, R \in \mathbb{Z} & \mathcal{S}^\sharp_{\text{f},1} &: X \in [5,80], R \in [-1,100] \\
\mathcal{S}^\sharp_{\text{main},2} &: X \in \mathbb{Z}, R = -1 & \mathcal{S}^\sharp_{\text{f},2} &: X \in [5,80], R \in [10,160] \\
\mathcal{S}^\sharp_{\text{main},3} &: X \in [5,10], R = -1 & \mathcal{S}^\sharp_{\text{f},3} &: X \in [5,80], R \in [101,160] \\
\mathcal{S}^\sharp_{\text{main},4} &: X \in [5,80], R \in [0,100] & \mathcal{S}^\sharp_{\text{f},4} &: X \in [5,80], R = 0 \\
\mathcal{S}^\sharp_{\text{main},5} &: X = 80, R \in [0,100] & \mathcal{S}^\sharp_{\text{f},5} &: X \in [5,80], R \in [10,100] \\
\mathcal{S}^\sharp_{\text{main},6} &: X \in [5,80], R \in [0,100] & \mathcal{S}^\sharp_{\text{f},6} &: X \in [5,80], R \in [0,100]
\end{aligned}$$

Imprecision

In fact, in our example, $R = 0$ holds at the end of the program
but we find $R \in [0, 100]$!
$\Longrightarrow$ the analysis is imprecise

Explanation: the control-flow graph adds impossible executions paths



General case: concrete semantics

Procedures

Syntax:

- $\mathcal{F}$ finite set of procedure names
- $body : \mathcal{F} \to stat$: procedure bodies
- $main \in stat$: entry point body
- $\mathbb{V}_G$: set of global variables
- $\mathbb{V}_f$: set of local variables for procedure $f \in \mathcal{F}$
  procedure f can only access $\mathbb{V}_f \cup \mathbb{V}_G$
  main has no local variable and can only access $\mathbb{V}_G$
- $stat ::= f(expr_1, \ldots, expr_{|\mathbb{V}_f|}) \mid \cdots$

  procedure call, $f \in \mathcal{F}$, setting all its local variables

  local variables double as procedure arguments
  no special mechanism to return a value (a global variable can be used)

Concrete environments

Notes:

- when f calls g, we must remember the value of f's locals $\mathbb{V}_f$ in the semantics of g and restore them when returning
- several copies of each $V \in \mathbb{V}_f$ may exist at a given time
  due to recursive calls, i.e.: cycles in the call graph

$\implies$ concrete environments use per-variable stacks

Stacks: $\mathcal{S} \stackrel{\text{def}}{=} \mathbb{Z}^*$ (finite sequences of integers)

- **push**$(v, s) \stackrel{\text{def}}{=} v \cdot s$ $\quad (v, v' \in \mathbb{Z},\ s, s' \in \mathcal{S})$
- **pop**$(s) \stackrel{\text{def}}{=} s'$ when $\exists v: s = v \cdot s'$, undefined otherwise
- **peek**$(s) \stackrel{\text{def}}{=} v$ when $\exists s': s = v \cdot s'$, undefined otherwise
- **set**$(v, s) \stackrel{\text{def}}{=} v \cdot s'$ when $\exists v': s = v' \cdot s'$, undefined otherwise

Environments: $\mathcal{E} \stackrel{\text{def}}{=} (\cup_{f \in \mathcal{F}} \mathbb{V}_f \cup \mathbb{V}_G) \to \mathcal{S}$

for $\mathbb{V}_G$, stacks are not necessary but simplify the presentation

traditionally, there is a single global stack for all local variables
using per-variable stacks instead also makes the presentation simpler

Concrete semantics

Concrete semantics: on $\mathcal{E} \stackrel{\text{def}}{=} (\cup_{f \in \mathcal{F}} \mathbb{V}_f \cup \mathbb{V}_G) \to \mathcal{S}$

variable reads and updates only consider the top of the stack;
procedure calls push and pop local variables

- $\mathsf{E}[\![\, V \,]\!]\, \rho \stackrel{\text{def}}{=} \mathbf{peek}(\rho(V))$
- $\mathsf{S}[\![\, V \leftarrow e \,]\!]\, R \stackrel{\text{def}}{=} \{\, \rho[V \mapsto \mathbf{set}(x, \rho(V))] \mid \rho \in R, x \in \mathsf{E}[\![\, e \,]\!]\, \rho \,\}$
- $\mathsf{S}[\![\, f(e_{V_1}, \ldots, e_{V_n}) \,]\!]\, R = R_3$, where:

  $R_1 \stackrel{\text{def}}{=} \{\, \rho[\forall V \in \mathbb{V}_f \colon V \mapsto \mathbf{push}(x_V, \rho(V))] \mid \rho \in R, \forall V \in \mathbb{V}_f \colon x_V \in \mathsf{E}[\![\, e_V \,]\!]\, \rho \,\}$
  (evaluate each argument e_V and push its value x_V on the stack $\rho(V)$)

  $R_2 \stackrel{\text{def}}{=} \mathsf{S}[\![\, \mathit{body}(f) \,]\!]\, R_1$ (evaluate the procedure body)

  $R_3 \stackrel{\text{def}}{=} \{\, \rho[\forall V \in \mathbb{V}_f \colon V \mapsto \mathbf{pop}(\rho(V))] \mid \rho \in R_2 \,\}$ (pop local variables)

- initial environment: $\rho_0 \stackrel{\text{def}}{=} \lambda V \in \mathbb{V}_G . 0$

other statements are unchanged

Semantic inlining

Semantic inlining

Naïve abstract procedure call: mimic the concrete semantics

- assign abstract variables to stack positions:
  $\mathbb{V}^\sharp \stackrel{\text{def}}{=} \mathbb{V}_G \cup (\cup_{f \in \mathcal{F}} \mathbb{V}_f \times \mathbb{N})$
  $\mathbb{V}^\sharp$ is infinite, but each abstract environment uses finitely many variables
- $\mathcal{E}^\sharp_{\mathbb{V}}$ abstracts $\mathcal{P}(\mathbb{V} \to \mathbb{Z})$, for any finite $\mathbb{V} \subseteq \mathbb{V}^\sharp$
  $V \in \mathbb{V}_f$ denotes $(V, 0)$ in $\mathbb{V}^\sharp$
  push V: shift variables, replacing (V, i) with $(V, i+1)$, then add $(V, 0)$
  pop V: remove $(V, 0)$ and shift each (V, i) to $(V, i-1)$
- $\mathsf{S}^\sharp[\![\, f(e_1, \ldots, e_n) \,]\!]\, X^\sharp$ is then reduced to:

$$\begin{array}{lr}
X_1^\sharp = \mathsf{S}^\sharp[\![\, \textbf{push } V_1; \ldots; \textbf{push } V_n \,]\!]\, X^\sharp & \text{(add fresh variables for } \mathbb{V}_f\text{)} \\
X_2^\sharp = \mathsf{S}^\sharp[\![\, V_1 \leftarrow e_1; \ldots; V_n \leftarrow e_n \,]\!]\, X_1^\sharp & \text{(bind arguments to locals)} \\
X_3^\sharp = \mathsf{S}^\sharp[\![\, body(f) \,]\!]\, X_2^\sharp & \text{(execute the procedure body)} \\
X_4^\sharp = \mathsf{S}^\sharp[\![\, \textbf{pop } V_1; \ldots; \textbf{pop } V_n \,]\!]\, X_3^\sharp & \text{(delete local variables)}
\end{array}$$

Limitations:

- does not terminate in case of unbounded recursivity
- requires many abstract variables to represent the stacks
- procedures must be re-analyzed for every call
  full context-sensitivity: precise but costly

Example

Example

main :
$R \leftarrow -1;$
$f(\textbf{rand}(5,10));$
$f(80)$

$f(X)$:
$R \leftarrow 2 \times X;$
if $R > 100$ **then** $R \leftarrow 0$

Analysis using intervals

- after the first call to f, we get $R \in [10, 20]$
- after the second call to f, we get $R = 0$

Call-site abstraction

Call-site abstraction

Abstracting stacks: into a fixed, bounded set $\mathbb{V}^\sharp$ of variables

- $\mathbb{V}^\sharp \stackrel{\text{def}}{=} \cup_{f \in \mathcal{F}} \{ V, \hat{V} \mid V \in \mathbb{V}_f \} \cup \mathbb{V}_G$
  two copies of each local variable
  V abstracts the value at the top of the stack (current call)
  $\hat{V}$ abstracts the rest of the stack
- $S^\sharp[\![\textbf{push } V]\!] X^\sharp \stackrel{\text{def}}{=} X^\sharp \cup^\sharp S^\sharp[\![\hat{V} \leftarrow V]\!] X^\sharp$
  $S^\sharp[\![\textbf{pop } V]\!] X^\sharp \stackrel{\text{def}}{=} X^\sharp \cup^\sharp S^\sharp[\![V \leftarrow \hat{V}]\!] X^\sharp$
  weak updates, similar to array manipulation
  no need to create and delete variables dynamically
- assignments and tests always access V, not $\hat{V}$
  $\Longrightarrow$ strong update (precise)

Note: when there is no recursivity, $\hat{V}$, **push** and **pop** can be omitted

Call-site abstraction

Principle: merge all the contexts in which each function is called

- we maintain two global maps $\mathcal{F} \to \mathcal{E}^\sharp$:

  $C^\sharp(f)$: abstracts the environments when calling f
  $R^\sharp(f)$: abstracts the environments when returning from f
  gather environments from all possible calls to f, disregarding the call sites

- during the analysis, when encountering a call $\mathsf{S}^\sharp[\![\, body(f) \,]\!]\, X^\sharp$:

  we return $R^\sharp(f)$
  but we also replace $C^\sharp$ with $C^\sharp[f \mapsto C^\sharp(f) \cup^\sharp X^\sharp]$

- $R^\sharp(f)$ is computed from $C^\sharp(f)$ as

  $R^\sharp(f) = \mathsf{S}^\sharp[\![\, body(f) \,]\!]\,(C^\sharp(f))$

Call-site abstraction

Fixpoint:

there may be circular dependencies between $C^\sharp$ and $R^\sharp$

e.g., in $f(2); f(3)$, the input for $f(3)$ depends on the output from $f(2)$

$\Longrightarrow$ we compute a fixpoint for $C^\sharp$ by iteration:

- initially, $\forall f\colon C^\sharp(f) = R^\sharp(f) = \bot$
- analyze *main*
- while $\exists f\colon C^\sharp(f)$ not stable

  apply widening $\triangledown$ to the iterates of $C^\sharp(f)$

  update $R^\sharp(f) = \mathsf{S}^\sharp[\![\, body(f) \,]\!]\, C^\sharp(f)$

  analyze *main* and all the procedures again

  (this may modify some $C^\sharp(g)$)

$\Longrightarrow$ using $\triangledown$, the analysis always terminates in finite time

we can be more efficient and avoid re-analyzing procedures when not needed
e.g., use a workset algorithm, track procedure dependencies, etc.

Example

Example

$main$:
  $R \leftarrow -1$;
  $f(\mathbf{rand}(5,10))$;
  $f(80)$

$f(X)$:
  $R \leftarrow 2 \times X$;
  if $R > 100$ **then** $R \leftarrow 0$

Analysis: using intervals (without widening as there is no dependency)

- first analysis of *main*: we get $\perp$ (as $R^\sharp(f) = \perp$)
  but $C^\sharp(f) = [R \mapsto [-1,-1], X \mapsto [5,10]]$
- first analysis of f: $R^\sharp(f) = [R \mapsto [10,20], X \mapsto [5,10]]$
- second analysis of *main*: we get
  $C^\sharp(f) = [R \mapsto [-1,20], X \mapsto [5,80]]$
- second analysis of f: $R^\sharp(f) = [R \mapsto [0,100], X \mapsto [5,80]]$
- final analysis of *main*, we find $R \in [0,100]$ at the program end
  less precise than $R = 0$ found by semantic inlining

Partial context-sensitivity

Variants: $k-$limiting, k is a constant

- stack:

  assign a distinct variable for the k highest levels of V
  abstract the lower (unbounded) stack part with $\hat{V}$
  more precise than keeping only the top of the stack separately

- context-sensitivity:

  each syntactic call has a unique call-site $\ell \in \mathcal{L}$
  a call stack is a sequence of nested call sites: $c \in \mathcal{L}^*$
  an abstract call stack remembers the last k call sites: $c^\sharp \in \mathcal{L}^k$
  the $C^\sharp$ and $R^\sharp$ maps now distinguish abstract call stacks
  $C^\sharp, R^\sharp : \mathcal{L}^k \to \mathcal{E}^\sharp$
  more precise than a partitioning by function only

larger k give more precision but less efficiency

Example: context-sensitivity

Example

main :
  $R \leftarrow -1$;
  $\ell_1 : f(\textbf{rand}(5,10))$;
  $\ell_2 : f(80)$

$f(X)$:
  $R \leftarrow 2 \times X$;
  if $R > 100$ **then** $R \leftarrow 0$

Analysis: using intervals and $k = 1$

- $C^\sharp(\ell_1) = [R \mapsto [-1,1], X \mapsto [5,10]]$
  $\Longrightarrow R^\sharp(\ell_1) = [R \mapsto [10,20], X \mapsto [5,10]]$
- $C^\sharp(\ell_2) = [R \mapsto [10,20], X \mapsto [80,80]]$
  $\Longrightarrow R^\sharp(\ell_2) = [R \mapsto [0,0], X \mapsto [80,80]]$
- at the end of the analysis, we get $R = 0$
  more precise than $R \in [0,100]$ found without context-sensitivity

Tabulation abstraction

Cardinal power

Principle:

the semantic of a function is $\mathsf{S}[\![\, body(f) \,]\!] : \mathcal{P}(\mathcal{E}) \to \mathcal{P}(\mathcal{E})$

$\Longrightarrow$ abstract it as an abstract function in $\mathcal{E}^\sharp \rightharpoonup \mathcal{E}^\sharp$

we use a partial function as the image of most abstract elements is not useful

Analysis: tabulated analysis

- use a global partial map $F^\sharp : \mathcal{F} \times \mathcal{E}^\sharp \rightharpoonup \mathcal{E}^\sharp$
- $F^\sharp$ is initially empty, and is filled on-demand
- when encountering $\mathsf{S}^\sharp[\![\, body(f) \,]\!]\, X^\sharp$

  return $F^\sharp(f, X^\sharp)$ if defined

  else, compute $\mathsf{S}^\sharp[\![\, body(f) \,]\!]\, X^\sharp$, store it in $F^\sharp(f, X^\sharp)$ and return it

Optimizations: trade precision for efficiency

- if $X^\sharp \sqsubseteq Y^\sharp$ and $F^\sharp(f, X^\sharp)$ is not defined, we can use $F^\sharp(f, Y^\sharp)$ instead
- if the size of $F^\sharp$ grows too large, use $F^\sharp(f, \top)$ instead

  sound, and ensures that the analysis terminates in finite time

Example

Example

main :
$R \leftarrow -1;$
$f(\mathbf{rand}(5,10));$
$f(80)$

$f(X)$:
$R \leftarrow 2 \times X;$
if $R > 100$ **then** $R \leftarrow 0$

Analysis using intervals

- $F^\sharp =$
  $$\begin{aligned}[\ &(f,[R \mapsto [-1,-1], X \mapsto [5,10]]) \mapsto [R \mapsto [10,20], X \mapsto [5,10]],\\ &(f,[R \mapsto [10,20], X \mapsto [80,80]]) \mapsto [R \mapsto [0,0], X \mapsto [80,80]]\ ]\end{aligned}$$
- at the end of the analysis, we get again $R = 0$

here, the function partitioning gives the same result as the call-site partitioning

Dynamic partitioning: complex example

Example: McCarthy's 91 function

main :
$\quad Mc(\textbf{rand}(0,+\infty))$

$Mc(n)$:
$\quad$ **if** $n > 100$ **then** $r \leftarrow n - 10$
$\quad$ **else** $Mc(n+11)$; $Mc(r)$

- in the concrete, when terminating:
  $r = n - 10$ when $n > 101$, and $r = 91$ wen $n \in [0, 101]$
- using a widening $\triangledown$ to choose tabulated abstract values $F^\sharp(f, X^\sharp)$
  we find:
  $$\begin{array}{lcl} n \in [0,72] & \Rightarrow & r = 91 \\ n \in [73,90] & \Rightarrow & r \in [91,101] \\ n \in [91,101] & \Rightarrow & r = 91 \\ n \in [102,111] & \Rightarrow & r \in [91,101] \\ n \in [112,+\infty] & \Rightarrow & r \in [91,+\infty] \end{array}$$

(source: Bourdoncle, JFP 1992)