Complete Abstractions Everywhere

Francesco Ranzato

University of Padova, Italy

Claims

- Ubiquity of completeness properties of static analyses
- Completeness can be a beneficial tool for
  - designing new static analyses
  - understanding how existing static analyses work

Setting

- Static analyses are designed in abstract interpretation
  - Abstract domains
  - Abstract functions
- Static analyses must be sound
- Static analyses may be (covertly) complete

Approximation

- Static analysis $[\![P]\!]^\sharp$ must be a sound approximation of concrete semantics $[\![P]\!]$
- Approximation by partial orders
  - $[\![P]\!]$ ranges in $(D, \leq_D)$
  - $[\![P]\!]^\sharp$ ranges in some abstraction $(A, \leq_A)$
  - $x \leq y$: y approximates x

Soundness #1

$$[\![P]\!]\gamma(\mathit{input}^\sharp) \leq_D \gamma\left([\![P]\!]^\sharp \mathit{input}^\sharp\right)$$

- Concretization $\gamma : A \to D$

Soundness #2

$$\alpha\left([\![P]\!]\mathit{input}\right) \leq_A [\![P]\!]^\sharp \alpha(\mathit{input})$$

- Abstraction $\alpha : D \to A$

Completeness

- $[\![P]\!]_1^\sharp$ and $[\![P]\!]_2^\sharp$ on a common abstraction A
- "more precise than": $[\![P]\!]_1^\sharp\, \mathit{input}^\sharp \leq_A [\![P]\!]_2^\sharp\, \mathit{input}^\sharp$

$$\boxed{\text{Completeness means "as precise as possible"}}$$

Completeness #1 $\Rightarrow$ Exactness

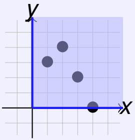
$[\![P]\!]\gamma(\mathit{input}^\sharp) = \gamma\left([\![P]\!]^\sharp \mathit{input}^\sharp\right)$

Completeness #2 $\Rightarrow$ Completeness

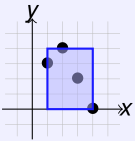
$\alpha\left([\![P]\!]\mathit{input}\right) = [\![P]\!]^\sharp \alpha(\mathit{input})$

Numerical Abstractions

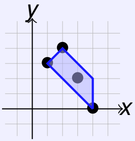
Sign abstraction



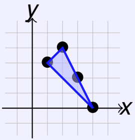
Int interval abstraction



Oct octagon abstraction



Polyhedra abstraction



Example

Sign

$$\begin{tikzcd}[column sep=small] & \mathbb{Z} \arrow[dl, dash] \arrow[dr, dash] & \\ \mathbb{Z}_{\leq 0} \arrow[dr, dash] & & \mathbb{Z}_{\geq 0} \arrow[dl, dash] \\ & 0 & \end{tikzcd}$$

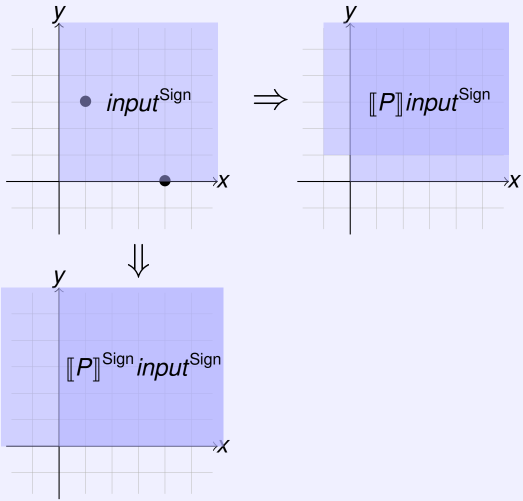
$$P \equiv \textbf{if } (*) \; \{x\text{- -};\, y\text{++};\}$$

$$input \triangleq \{(x/1, y/3), (x/4, y/0)\}$$

$$input^{\mathsf{Sign}} \triangleq \alpha_{\mathsf{Sign}}(input) = (x/\mathbb{Z}_{\geq 0}, y/\mathbb{Z}_{\geq 0})$$

Example

$P \equiv \textbf{if } (*) \ \{x\text{- -};\ y\text{++};\}$



Example

Sign

$$
\begin{tikzcd}[column sep=small]
 & \mathbb{Z} \arrow[dl, dash] \arrow[dr, dash] & \\
\mathbb{Z}_{\leq 0} \arrow[dr, dash] & & \mathbb{Z}_{\geq 0} \arrow[dl, dash] \\
 & 0 &
\end{tikzcd}
$$

$$P \equiv \textbf{if } (*) \ \{x\text{- -};\ y\text{++};\}$$

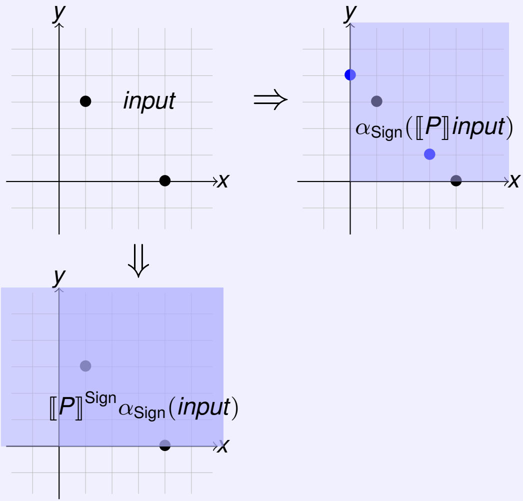
$$\mathit{input} \triangleq \{(x/1, y/3), (x/4, y/0)\}$$

$$\mathit{input}^{\mathsf{Sign}} \triangleq \alpha_{\mathsf{Sign}}(\mathit{input}) = (x/\mathbb{Z}_{\geq 0}, y/\mathbb{Z}_{\geq 0})$$

Sign is not exact

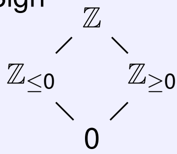
- $[\![P]\!]\mathit{input}^{\mathsf{Sign}} = (x/\mathbb{Z}_{\geq -1}, y/\mathbb{Z}_{\geq 0})$
- $[\![P]\!]^{\mathsf{Sign}}\mathit{input}^{\mathsf{Sign}} = (x/\mathbb{Z}, y/\mathbb{Z}_{\geq 0})$

$P \equiv$ **if** $(*)$ $\{x$- -; y++;$\}$


y
x
input
$\Rightarrow$
y
x
$\alpha_{\mathsf{Sign}}(\llbracket P \rrbracket input)$
$\Downarrow$
y
x
$\llbracket P \rrbracket^{\mathsf{Sign}} \alpha_{\mathsf{Sign}}(input)$

Example

Sign



$P \equiv \textbf{if } (*) \; \{x\text{- -}; \, y\text{++};\}$

$input \triangleq \{(x/1, y/3), (x/4, y/0)\}$

$input^{\mathsf{Sign}} \triangleq \alpha_{\mathsf{Sign}}(input) = (x/\mathbb{Z}_{\geq 0}, y/\mathbb{Z}_{\geq 0})$

Sign is not exact

- $[\![P]\!] input^{\mathsf{Sign}} = (x/\mathbb{Z}_{\geq -1}, y/\mathbb{Z}_{\geq 0})$
- $[\![P]\!]^{\mathsf{Sign}} input^{\mathsf{Sign}} = (x/\mathbb{Z}, y/\mathbb{Z}_{\geq 0})$

Sign is not complete

- $\alpha_{\mathsf{Sign}}([\![P]\!] input) = (x/\mathbb{Z}_{\geq 0}, y/\mathbb{Z}_{\geq 0})$
- $[\![P]\!]^{\mathsf{Sign}} \alpha_{\mathsf{Sign}}(input) = (x/\mathbb{Z}, y/\mathbb{Z}_{\geq 0})$

Interval domain Int

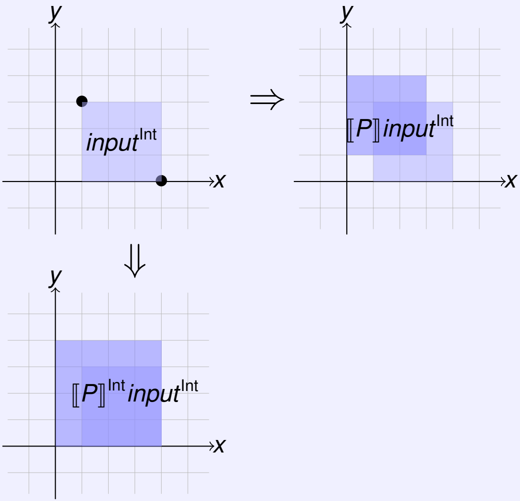
$$P \equiv \textbf{if } (*) \ \{x\text{-}\text{-}; \ y\text{++};\}$$

$$\mathit{input} \triangleq \{(x/1, y/3), (x/4, y/0)\}$$

$$\mathit{input}^{\mathsf{Int}} \triangleq \alpha_{\mathsf{Int}}(\mathit{input}) = (x/[1,4], y/[0,3])$$

Example

$P \equiv \textbf{if}\ (*)\ \{x\text{- -};\ y\text{++};\}$



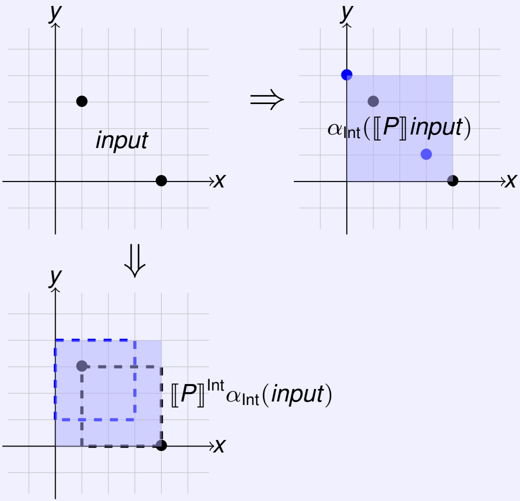
Interval domain Int

$$P \equiv \textbf{if } (*)\ \{x\text{- -};\ y\text{++};\}$$
$$\mathit{input} \triangleq \{(x/1, y/3), (x/4, y/0)\}$$
$$\mathit{input}^{\mathrm{Int}} \triangleq \alpha_{\mathrm{Int}}(\mathit{input}) = (x/[1,4], y/[0,3])$$

Int is not exact

- $[\![P]\!]\mathit{input}^{\mathrm{Int}} = (x/[0,4], y/[0,4]) \smallsetminus \{(x/0, y/0), (x/4, y/4)\}$
- $[\![P]\!]^{\mathrm{Int}}\mathit{input}^{\mathrm{Int}} = (x/[0,4], y/[0,4])$

$P \equiv$ **if** $(*)$ $\{x$- -; y++;$\}$


y
x
input
$\Rightarrow$
$\alpha_{\mathrm{Int}}([\![P]\!]\,input)$
$\Downarrow$
$[\![P]\!]^{\mathrm{Int}}\alpha_{\mathrm{Int}}(input)$

Example

Interval domain Int

$$P \equiv \textbf{if}\ (*)\ \{x\text{--};\ y\text{++};\}$$

$$input \triangleq \{(x/1, y/3), (x/4, y/0)\}$$

$$input^{\mathsf{Int}} \triangleq \alpha_{\mathsf{Int}}(input) = (x/[1,4], y/[0,3])$$

Int is not exact

- $[\![P]\!] input^{\mathsf{Int}} = (x/[0,4], y/[0,4]) \smallsetminus \{(x/0, y/0), (x/4, y/4)\}$
- $[\![P]\!]^{\mathsf{Int}} input^{\mathsf{Int}} = (x/[0,4], y/[0,4])$

Int is complete

- $\alpha_{\mathsf{Int}}([\![P]\!] input) = (x/[0,4], y/[0,4])$
- $[\![P]\!]^{\mathsf{Int}} \alpha_{\mathsf{Int}}(input) = (x/[0,4], y/[0,4])$

Octagon domain Oct

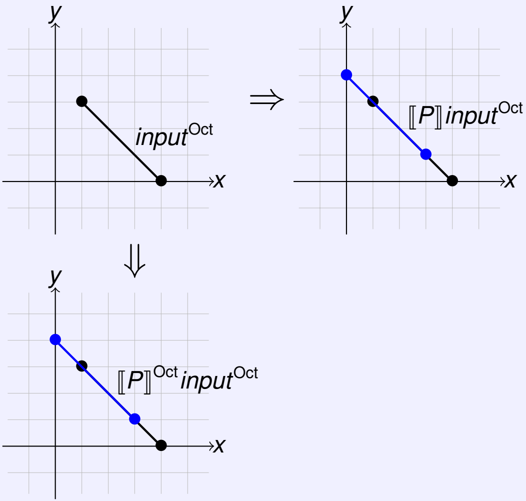
$P \equiv$ **if** $(*)$ $\{x$- -; y++;$\}$

$input \triangleq \{(x/1, y/3), (x/4, y/0)\}$

$input^{\mathrm{Oct}} \triangleq \alpha_{\mathrm{Oct}}(input) = \{x + y = 4, x \in [1, 4], y \in [0, 3]\}$

Example

$P \equiv \textbf{if } (*) \ \{x\text{- -}; \ y\text{++};\}$



Octagon domain Oct

$P \equiv \textbf{if } (*) \ \{x\text{- -};\ y\text{++};\}$

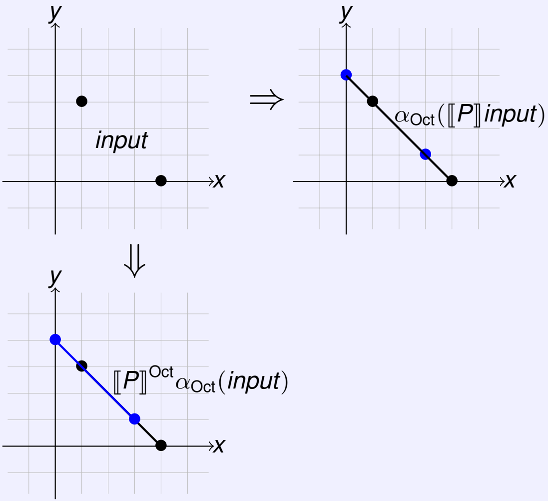
$input \triangleq \{(x/1, y/3), (x/4, y/0)\}$

$input^{\text{Oct}} \triangleq \alpha_{\text{Oct}}(input) = \{x + y = 4, x \in [1, 4], y \in [0, 3]\}$

Oct is exact

- $[\![P]\!] input^{\text{Oct}} = \{x + y = 4, x \in [0, 4], y \in [0, 4]\}$
- $[\![P]\!]^{\text{Oct}} input^{\text{Oct}} = \{x + y = 4, x \in [0, 4], y \in [0, 4]\}$

$P \equiv \textbf{if}\ (*)\ \{x\text{- -};\ y\text{++};\}$



Example

Octagon domain Oct

$$P \equiv \textbf{if } (*)\ \{x\text{- -};\ y\text{++};\}$$

$$\mathit{input} \triangleq \{(x/1, y/3), (x/4, y/0)\}$$

$$\mathit{input}^{\mathrm{Oct}} \triangleq \alpha_{\mathrm{Oct}}(\mathit{input}) = \{x + y = 4, x \in [1, 4], y \in [0, 3]\}$$

Oct is exact

- $[\![P]\!]\mathit{input}^{\mathrm{Oct}} = \{x + y = 4, x \in [0, 4], y \in [0, 4]\}$
- $[\![P]\!]^{\mathrm{Oct}}\mathit{input}^{\mathrm{Oct}} = \{x + y = 4, x \in [0, 4], y \in [0, 4]\}$

Oct is complete

- $\alpha_{\mathrm{Oct}}([\![P]\!]\mathit{input}) = \{x + y = 4, x \in [0, 4], y \in [0, 4]\}$
- $[\![P]\!]^{\mathrm{Oct}}\alpha_{\mathrm{Oct}}(\mathit{input}) = \{x + y = 4, x \in [0, 4], y \in [0, 4]\}$

- Why to care about completeness/exactness?
  - $\Rightarrow$ Completeness allow to understand abstractions in depth
- Is completeness/exactness a kind of incidental feature?
  - $\Rightarrow$ Complete abstractions are everywhere ☺
- Why completeness/exactness shoud be a useful tool?
  - $\Rightarrow$ Completeness-driven design of abstractions

Definition

Exactness: $\text{Sem} \circ \gamma = \gamma \circ \text{Analysis}$

Definition

Completeness: $\alpha \circ \text{Sem} = \text{Analysis} \circ \alpha$

Assumption

Abstractions have both α and γ, i.e. are Galois connections

Completeness is a Property of Abstractions

Complete/exact abstractions are necessarily
best correct approximations

Fact

Analysis *is exact on A* $\Leftrightarrow$

1. Analysis $= \alpha \circ$ Sem $\circ \gamma$
2. Sem $\circ \gamma = \gamma \circ \alpha$ Sem $\circ \gamma$

Fact

Analysis *is complete on A* $\Leftrightarrow$

1. Analysis $= \alpha \circ$ Sem $\circ \gamma$
2. $\alpha \circ$ Sem $= \alpha \circ$ Sem $\circ \gamma \circ \alpha$

Completeness is a Property of Abstractions

Definition

Abstraction A is exact for Sem $\Leftrightarrow$ $\mathrm{Sem} \circ \gamma = \gamma \circ \alpha\, \mathrm{Sem} \circ \gamma$

Definition

Abstraction A is complete for Sem $\Leftrightarrow$ $\alpha \circ \mathrm{Sem} = \alpha \circ \mathrm{Sem} \circ \gamma \circ \alpha$

- Programs with n integer variables ($n = 1, 2$ in examples)
- Program states in $\mathbb{Z}^n$
- Program properties in $\langle \wp(\mathbb{Z}^n), \subseteq \rangle$
- P_1 is approximated by P_2: $P_1 \subseteq P_2$ (i.e., $P_1 \Rightarrow P_2$)
- Standard transfer functions and collecting semantics

Scenario

- Abstractions as Galois connections
- Abstractions of program states in $\mathsf{Abs}(\wp(\mathbb{Z}^n))$
- A_1 refines A_2: $A_1 \trianglelefteq A_2 \Leftrightarrow \forall S. \gamma_1(\alpha_1(S)) \subseteq \gamma_2(\alpha_2(S))$
- $\langle \mathsf{Abs}(\wp(\mathbb{Z}^n)), \trianglelefteq \rangle$ the lattice of abstractions

Trivial Abstraction

- Abstracts each set in $\wp(\mathbb{Z})$ to a unique abstract value
- $A^{\top} = \{\mathbb{Z}\}$
- $A^{\top}$ is always trivially exact and complete

- $\{\mathbb{Z}, K\}$ for some $K \in \wp(\mathbb{Z})$
- Consider $A^0 \triangleq \{\mathbb{Z}, \{0\}\}$
- Consider the test $(\!| x > 0? |\!)$

A^0 is not complete

1. $A^0\big((\!| x > 0? |\!)\{-1\}\big) = A^0(\varnothing) = \{0\}$
2. $(\!| x > 0? |\!)^{A^0} A^0(\{-1\}) = \mathbb{Z}$

A^0 is not complete

$$A^0\big(\llparenthesis\, x > 0?\, \rrparenthesis\{-1\}\big) = A^0(\varnothing) = \{0\}$$

$$\llparenthesis\, x > 0?\, \rrparenthesis^{A^0} A^0(\{-1\}) = \mathbb{Z}$$

Where is the problem?

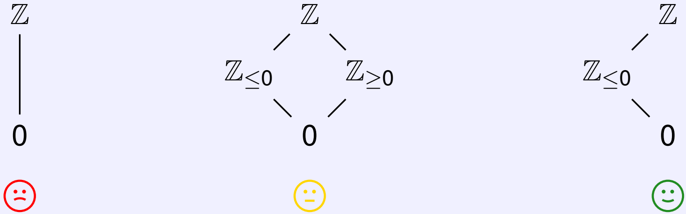
1. $S = \{-1\}$ does not satisfy the test: $\llparenthesis\, x > 0?\, \rrparenthesis S = \varnothing$
2. $A^0(\varnothing) = \{0\}$
3. $A^0(S) = \mathbb{Z}$

$\Rightarrow$ The abstract function on A^0 can only tell: $\mathbb{Z} \mapsto \mathbb{Z}$

- Sign $= \{\mathbb{Z}, \mathbb{Z}_{\leq 0}, \mathbb{Z}_{\geq 0}, 0\}$ refines A^0
- The previous problem does not happen with Sign
- Sign is complete for the test

...Sign is too refined...

...for the specific goal of being complete for $(\!| \, x > 0? \,|\!)$



What we really need is a minimal refinement of A^0 which is complete for $(\!|\, x > 0?\, |\!)$

These are called complete shell refinements

Complete Shells

- Minimal refinement of A which is complete for f
- $\mathrm{CShell}_f(A) \triangleq \sqcup\{A' \in \mathrm{Abs}(D) \mid A' \trianglelefteq A, A'$ is complete for $f\}$

Well Definedness

$\mathrm{CShell}_f(A)$ is complete for f

Constructive Characterization

1. $R_f(X) \triangleq \cup_{a \in X} \max\{d \in D \mid f(d) \leq a\}$
2. $R_f^{\omega}(A) = \cup_{i \in \mathbb{N}} R_f^i(A)$

$\Rightarrow$ $\mathrm{CShell}_f(A) =$ *Glb-Closure* of $R_f^{\omega}(A)$

- Minimal refinement of A which is exact for f
- $\mathrm{EShell}_f(A) \triangleq \sqcup \{A' \in \mathrm{Abs}(D) \mid A' \trianglelefteq A,\ A' \text{ is exact for } f\}$

Well Definedness

$\mathrm{EShell}_f(A)$ is exact for f

Constructive Characterization

1. $S_f(X) \triangleq \{f(x) \in D \mid x \in X\}$
2. $S_f^\omega(A) = \cup_{i\in\mathbb{N}} S_f^i(A)$

$\Rightarrow$ $\mathrm{EShell}_f(A) =$ *Glb-Closure* of $S_f^\omega(A)$

$$\begin{tikzcd} \mathbb{Z} \arrow[dd, dash] \\ \\ 0 \end{tikzcd} \qquad \overset{?}{\Longrightarrow} \qquad \begin{tikzcd} & \mathbb{Z} \\ \mathbb{Z}_{\leq 0} \arrow[ur, dash] \arrow[dr, dash] & \\ & 0 \end{tikzcd}$$

1. $\max\{S \in \wp(\mathbb{Z}) \mid \llparenthesis x > 0? \rrparenthesis S \subseteq \mathbb{Z}\} = \mathbb{Z}$
2. $\max\{S \in \wp(\mathbb{Z}) \mid \llparenthesis x > 0? \rrparenthesis S \subseteq \{0\}\} = \mathbb{Z}_{\leq 0}$
3. $\max\{S \in \wp(\mathbb{Z}) \mid \llparenthesis x > 0 \rrparenthesis S \subseteq \mathbb{Z}_{\leq 0}\} = \mathbb{Z}_{\leq 0}$

$\Rightarrow$ Complete shell: $\{\mathbb{Z}, \mathbb{Z}_{\leq 0}, \{0\}\}$

Example

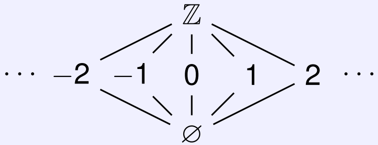
$$
\begin{tikzcd}[column sep=small]
\mathbb{Z} \arrow[d, dash] \\
0
\end{tikzcd}
\quad \overset{?}{\Longrightarrow} \quad
\begin{tikzcd}[column sep=small]
 & \mathbb{Z} \arrow[dl, dash] \arrow[dr, dash] & \\
\mathbb{Z}_{\leq 0} \arrow[dr, dash] & & \mathbb{Z}_{\geq 0} \arrow[dl, dash] \\
 & 0 &
\end{tikzcd}
$$

$$\boxed{\text{Sign is the complete shell of } A^0 \text{ for } \llparenthesis\, x > 0?\, \rrparenthesis \text{ and } \llparenthesis\, x < 0?\, \rrparenthesis}$$

- Obvious? 😐
- Interesting? 😐

$\Rightarrow$ Let's go on...

Constant Propagation

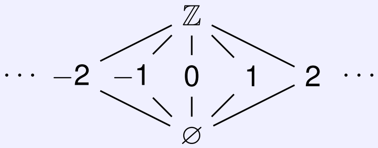


$\boxed{\text{CP is a refinement of } A^0}$

Question

Is CP a complete shell of A^0?

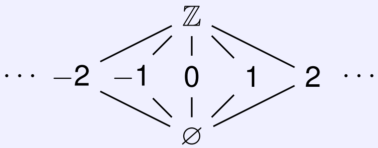
Constant Propagation



Observations

1. A^0 is not complete for $\llparenthesis x := x \pm 1 \rrparenthesis$
2. CP is complete for any $\llparenthesis x := x \pm k \rrparenthesis$
3. Compositions of $\llparenthesis x := x \pm 1 \rrparenthesis$ provide $\llparenthesis x := x \pm k \rrparenthesis$

Let us compute the complete shell of A^0 for $\llparenthesis x := x \pm 1 \rrparenthesis$



1. $\max \left\{ S \in \wp(\mathbb{Z}) \mid (\!| x := x + 1 |\!) S \subseteq \{0\} \right\} = \{-1\}$
2. $\max \left\{ S \in \wp(\mathbb{Z}) \mid (\!| x := x - 1 |\!) S \subseteq \{0\} \right\} = \{+1\}$
3. $\max \left\{ S \in \wp(\mathbb{Z}) \mid (\!| x := x \pm 1 |\!) S \subseteq \{1\} \right\} = \{\mp 2\}$

   $\ldots$

Fact

CP is the complete shell of A^0 for additions

- Obvious?
  - Maybe not at first sight! ☺
- Interesting?
  - Show me more! 😐

Incompleteness of Sign

$$\begin{tikzcd}[column sep=small]
 & \mathbb{Z} & \\
\mathbb{Z}_{\leq 0} \arrow[ur, dash] & & \mathbb{Z}_{\geq 0} \arrow[ul, dash] \\
 & 0 \arrow[ul, dash] \arrow[ur, dash] &
\end{tikzcd}$$

$$\boxed{\text{Sign is not complete for additions}}$$

1. $\mathrm{Sign}(\llparenthesis\, x := x + 1 \,\rrparenthesis\{-1\}) = \mathrm{Sign}(\{0\}) = \{0\}$
2. $\llparenthesis\, x := x + 1 \,\rrparenthesis^{\mathrm{Sign}}\,\mathrm{Sign}(\{-1\}) = \llparenthesis\, x := x + 1 \,\rrparenthesis^{\mathrm{Sign}}(\mathbb{Z}_{\leq 0}) = \mathbb{Z}$

Question

What is the complete shell of Sign for additions?

Uhm...

1. I should stretch/restrict $\mathbb{Z}_{\geq 0}$ on the left/right
2. I should restrict/stretch $\mathbb{Z}_{\leq 0}$ on the left/right
3. If I have $\mathbb{Z}_{\geq m}$ and $\mathbb{Z}_{\leq n}$ then I also have $[m, n]$

Answer

Intervals!

Let us compute the complete shell of Sign for $(\!| x := x \pm 1 |\!)$

1. $\max\left\{S \in \wp(\mathbb{Z}) \mid (\!| x := x \pm 1 |\!) S \subseteq \mathbb{Z}_{\leq 0}\right\} = \mathbb{Z}_{\leq \mp 1}$
2. $\max\left\{S \in \wp(\mathbb{Z}) \mid (\!| x := x \pm 1 |\!) S \subseteq \mathbb{Z}_{\geq 0}\right\} = \mathbb{Z}_{\geq \mp 1}$
3. $\max\left\{S \in \wp(\mathbb{Z}) \mid (\!| x := x \pm 1 |\!) S \subseteq \mathbb{Z}_{\leq -1}\right\} = \mathbb{Z}_{\leq \mp 2}$

   $\ldots$
4. Glb-closure of $\{\mathbb{Z}_{\geq n}, \mathbb{Z}_{\leq m} \mid n, m \in \mathbb{Z}\}$

Fact

Int is the complete shell of Sign for additions

Uhm...

- Obvious?
  - Probably not! ☺
- Interesting?
  - Not bad! ☺
- That’s the whole story? 😐

Remark

- Completeness is not monotone for abstraction refinements
  - A_1 complete, $A_2 \trianglelefteq A_1 \not\Rightarrow A_2$ complete

$\Rightarrow$ Completeness can be lost through shell refinements

Fact

1. Sign was designed as complete shell for $(\!| x > 0? |\!)$ and $(\!| x < 0? |\!)$
2. Int was designed as complete shell of Sign for additions

$\Rightarrow$ Int lost completeness for $(\!| x > 0? |\!)$ and $(\!| x < 0? |\!)$

Fact

Complete shell of Int for all $(\!|\, x > k?\, |\!)$ leads to the concrete domain

Fact

Complete shell of Int for $(\!|\, x > 0?\, |\!)$ and $(\!|\, x < 0?\, |\!)$ is

$$\mathrm{Int}^{\neq\pm1} \triangleq \mathrm{Int} \cup \left\{ I \setminus \{+1\} \;\middle|\; I \in \mathrm{Int} \right\} \cup \left\{ I \setminus \{-1\} \;\middle|\; I \in \mathrm{Int} \right\}$$

Genesis of Octagons

- Intervals are not "relational"
- Need for relational abstractions
- Fully relational $\Rightarrow$ Polyhedra
  - Precise but expensive
- Weakly relational $\Rightarrow$ Octagons and the like
  - Tractable and still practically precise

Standard Example

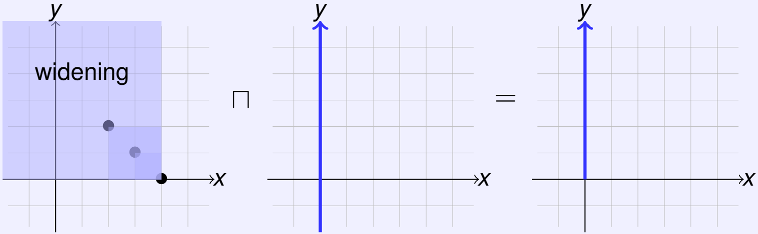
$\{x = 4, y = 0\}$ $\quad$ **while** $x \neq 0$ **do** $\{x{-}{-};\ y{+}{+}\}$ $\quad$ $\{x = 0, y = 4\}$

Int is not enough for deriving that $y = 4$ at the exit of P
i.e. Int is not complete

Genesis of Octagons

Standard Example

$\{x = 4, y = 0\}$ **while** $x \neq 0$ **do** $\{x{-}{-};\ y{+}{+}\}$ $\{x = 0, y = 4\}$



Standard Example

$\{x = 4, y = 0\}$ $\quad$ **while** $x \neq 0$ **do** $\{x{-}{-};\ y{+}{+}\}$ $\quad$ $\{x = 0, y = 4\}$

1. $\mathrm{Int}(\llbracket P \rrbracket \langle x/\{4\}, y/\{0\}\rangle) = \langle x/[0,0], y/[4,4]\rangle$
2. $\llbracket P \rrbracket^{\mathrm{Int}}\, \mathrm{Int}\langle x/\{4\}, y/\{0\}\rangle = \langle x/[0,0], y/[0,+\infty)\rangle$

Problem

Int is not able to represent the loop invariant $x + y = 4$

(Logical) Solution

Oct represents sets $\{\langle x, y \rangle \in \mathbb{Z}^2 \mid x \pm y = k\}$

Question

Int is already complete for $(\!|\, x{-}{-} \,|\!)$ and $(\!|\, y{+}{+} \,|\!)$.

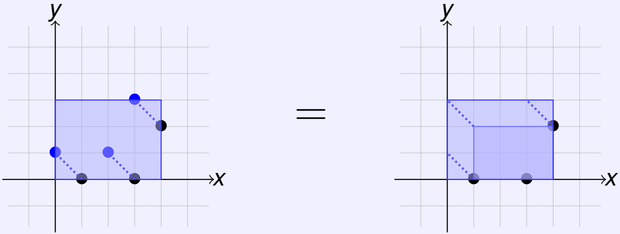
$\Rightarrow$ Why this is not enough?

Answer

What we really need is an abstraction that represents precisely the loop invariant $x + y = 4$

$\Rightarrow$ We need exactness for $\lambda S.S \cup (\!|\, x{-}{-};\ y{+}{+} \,|\!) S$

$$\boxed{\text{Int is complete for } \lambda S.S \cup (\!|\, x{-}{-};\ y{+}{+} \,|\!)\, S}$$



$$\boxed{\text{Int is not exact for } \lambda S. S \cup (\!| x{-}{-};\ y{+}{+} |\!) S}$$

1. $\{\langle x/4, y/0\rangle\} \cup (\!| x{-}{-};\ y{+}{+} |\!)\{\langle x/4, y/0\rangle\} = \{\langle x/4, y/0\rangle, \langle x/3, y/1\rangle\}$

2. $\text{Int}\left(\{\langle x/4, y/0\rangle\} \cup (\!| x{-}{-};\ y{+}{+} |\!)\{\langle x/4, y/0\rangle\}\right) \ni \langle x/4, y/1\rangle$

Let us compute the exact shell of Int for

$$\lambda S.S \cup (\!| x{-}{-};\ y{+}{+} |\!) S$$
$$\lambda S.S \cup (\!| x{-}{-};\ y{-}{-} |\!) S$$
$$\lambda S.S \cup (\!| x{+}{+};\ y{+}{+} |\!) S$$
$$\lambda S.S \cup (\!| x{+}{+};\ y{-}{-} |\!) S$$

Let us compute the exact shell of Int for
$F \triangleq \lambda S.S \cup (\!| x\text{--};\ y\text{++} |\!) S$

Consider a generic point $\langle a, b \rangle$:

1. $F^0(\{\langle a, b \rangle\}) = \{\langle a, b \rangle\}$
2. $F^1(\{\langle a, b \rangle\}) = \{\langle a, b \rangle, \langle a-1, b+1 \rangle\}$
3. $F^2(\{\langle a, b \rangle\}) = \{\langle a, b \rangle, \langle a-1, b+1 \rangle, \langle a-2, b+2 \rangle\}$

   $\ldots$

$\Rightarrow$ $F^\omega(\{\langle a, b \rangle\}) = \{\langle x, y \rangle \mid x + y = a + b,\ x \leq a,\ y \geq b\}$

Let us compute the exact shell of Int for
$G \triangleq \lambda S.S \cup (\!| x\text{++};\ y\text{++} |\!) S$

Consider a generic point $\langle a, b\rangle$:

1. $G^0(\{\langle a, b\rangle\}) = \{\langle a, b\rangle\}$
2. $G^1(\{\langle a, b\rangle\}) = \{\langle a, b\rangle, \langle a+1, b+1\rangle\}$
3. $G^2(\{\langle a, b\rangle\}) = \{\langle a, b\rangle, \langle a+1, b+1\rangle, \langle a+2, b+2\rangle\}$

$\ldots$

$\Rightarrow\ G^\omega(\{\langle a, b\rangle\}) = \{\langle x, y\rangle \mid x - y = a + b,\ x \geq a,\ y \geq b\}$

Octagons

Closure under intersections of...

1. Intervals
2. $\{\langle x, y\rangle \mid x + y = a + b,\ x \leq a,\ y \geq b\}$
3. $\{\langle x, y\rangle \mid x - y = a + b,\ x \geq a,\ y \geq b\}$
4. $\{\langle x, y\rangle \mid x + y = a + b,\ x \geq a,\ y \leq b\}$
5. $\{\langle x, y\rangle \mid x - y = a + b,\ x \leq a,\ y \leq b\}$

...generates all the octagons!

Uhm...

- Obvious?
  - Nice observation! ☺
- Interesting?
  - Nice story! ☺

The whole story shifts from

Intervals $\Rightarrow$ Octagons

to

Octagons $\Rightarrow$ Polyhedra

Example Program

$x := x_0;\ y := y_0;\ \textbf{while}\ (*)\ \textbf{do}\ \{x := x + k_x;\ \ y := y + k_y\}$

Loop invariant: $k_x y - k_y x = k_x y_0 - k_y x_0$

Example Program

$x := x_0;\ y := y_0;\ \textbf{while}\,(*)\,\textbf{do}\,\{x := x + k_x;\ y := y + k_y\}$

Loop invariant: $k_x y - k_y x = k_x y_0 - k_y x_0$

Fact

Polyhedra are the exact shell of Octagons for the functions in

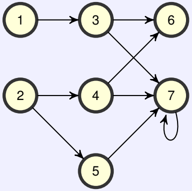
$$F = \{\lambda\, S.S \cup (\!|\, x := x + k_x;\ y := y + k_y \,|\!) S\}_{\langle k_x, k_y\rangle \in \mathbb{Z}^2}$$

Lessons

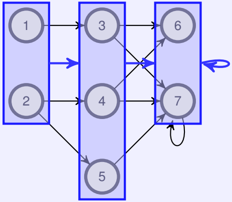
1. Lack of precision in analyzing P on A means lack of completeness of A for some functions of P
2. Abstractions are designed to remedy to some (hidden) lack of completeness of a simpler abstraction
3. Complete shells as a systematic tool for driving the design of abstractions

Let's change static analysis paradigm...

Concrete Model $\mathcal{K}$



Abstract Model $\mathcal{A}$



Preservation

Some temporal specification language $\mathfrak{L}$
e.g., CTL, ACTL, μ-calculus

$\mathcal{A}$ preserves $\mathfrak{L}$

$\forall \varphi \in \mathfrak{L}, \forall s \in \text{State}, P(s) \models^{\mathcal{A}} \varphi \Rightarrow s \models^{\mathcal{K}} \varphi$

$\mathcal{A}$ strongly preserves $\mathfrak{L}$

$\forall \varphi \in \mathfrak{L}, \forall s \in \text{State}, P(s) \models^{\mathcal{A}} \varphi \Leftrightarrow s \models^{\mathcal{K}} \varphi$

Problem

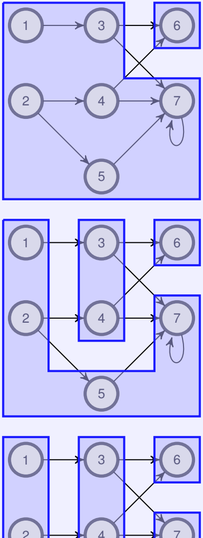
Compute the smallest abstract space $\text{State}^\sharp_{\mathfrak{L}}$ where to define an abstract model $\mathcal{A}_{\mathfrak{L}} = (\text{State}^\sharp_{\mathfrak{L}}, \rightarrow^\sharp)$ that strongly preserves $\mathfrak{L}$

Reduction Algorithms

1. CTL $\Rightarrow$ bisimulation algorithms
2. ACTL $\Rightarrow$ simulation algorithms
3. CTL-X $\Rightarrow$ stuttering bisimulation algorithms
4. ACTL-X $\Rightarrow$ stuttering simulation algorithms

...

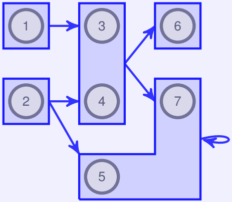
These are coarsest partition refinement algorithms



A state partition P is a bisimulation

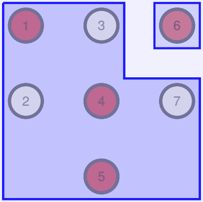
$$\Leftrightarrow$$

the abstract model $\langle P, \rightarrow^{\exists} \rangle$ strongly preserves CTL



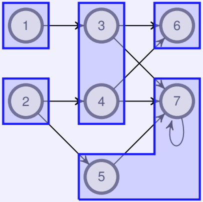
$B_1 \rightarrow^{\exists} B_2$ iff $\exists\, s_i \in B_i$ s.t. $s_1 \rightarrow s_2$

State partitions can be viewed as abstractions of $\wp(\text{State})$ that are exact for the complementation $\text{State} \smallsetminus S$



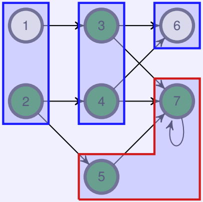
1. A state partition P is a bisimulation $\Leftrightarrow$ the abstract model $\langle P, \rightarrow^{\exists} \rangle$ strongly preserves CTL
2. State partitions are abstractions of $\wp(\mathrm{State})$
3. Predecessor pre : $\wp(\mathrm{State}) \longrightarrow \wp(\mathrm{State})$ is defined as $\mathrm{pre}(T) \triangleq \{s \in \mathrm{State} \mid s \rightarrow t,\ t \in T\}$

P is a bisimulation $\Leftrightarrow$ P is exact for pre



P is exact for pre:
for any block B, $\text{pre}(B)$ is a union of blocks of P

P is a bisimulation $\Leftrightarrow$ P as abstraction is exact for pre



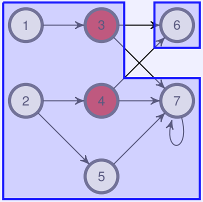
P is not exact for pre:

$\text{pre}([5, 7]) = \{2, 3, 4, 5, 7\}$ is not a union of blocks of P

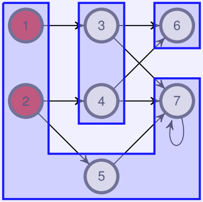
Fact

Bisimulation algorithms are exact shells of abstractions for:

1. complementation $\Rightarrow$ ensures that abstractions are partitions
2. predecessor $\Rightarrow$ ensures that partitions are bisimulations



$P \;\Rightarrow\;$ Glb-Closure of $P \cup \{\text{pre}([6])\}$



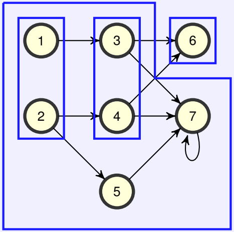
$P \;\Rightarrow\;$ Glb-Closure of $P \cup \{\text{pre}([3,4])\}$

- Obvious?
  - Nice observation! ☺
- Interesting?
  - Nice story! ☺
- Tell me more! 😐

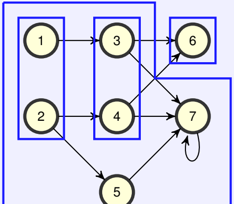
The whole story shifts from bisimulation to

1. simulation
2. stuttering bisimulation/simulation
3. generic temporal languages
4. bisimulation/simulation in probabilistic automata

A state preorder R is a simulation
$\Leftrightarrow$
$\forall\, s \in$ State, pre$(R(s))$ is a union of some $R(t)$

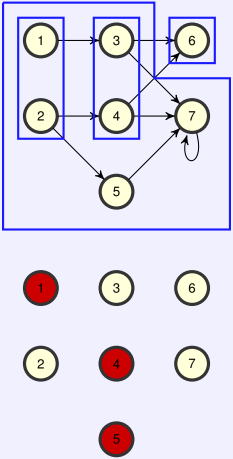


$$
\begin{aligned}
R(1) &= \{1,2\}\\
R(2) &= \{1,2\}\\
R(3) &= \{3,4\}\\
R(4) &= \{3,4\}\\
R(5) &= \{1,2,3,4,5,7\}\\
R(6) &= \{6\}\\
R(7) &= \{1,2,3,4,5,7\}
\end{aligned}
$$

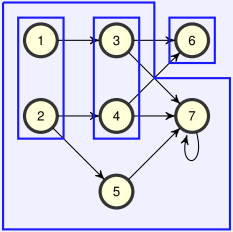


$$
\begin{aligned}
\mathrm{pre}(R(1)) = \mathrm{pre}(R(2)) &= \varnothing\\
\mathrm{pre}(R(3)) = \mathrm{pre}(R(4)) &= \{1,2\}\\
\mathrm{pre}(R(5)) = \mathrm{pre}(R(7)) &= \{1,2,3,4,5,7\}\\
\mathrm{pre}(R(6)) &= \{3,4\}
\end{aligned}
$$

State preorders can be viewed as abstractions of $\wp(\text{State})$ that are exact for the union, i.e., closed under unions

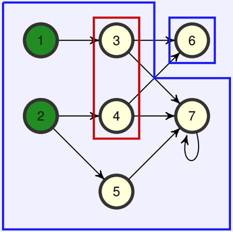


R is a simulation $\Leftrightarrow$ R as abstraction is exact for pre



R is exact for pre:
for any $R(s)$, $\mathrm{pre}(R(s))$ is a union of some $R(t)$

R is a simulation $\Leftrightarrow$ R as abstraction is exact for pre

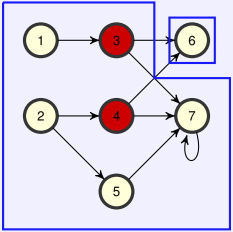


R is not exact for pre:
pre$(\{3,4\}) = \{1,2\}$ is not a union of $R(t)$

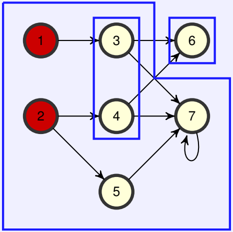
Fact

Simulation algorithms are exact shells of abstractions for:

1. union $\Rightarrow$ ensures that abstractions are preorders
2. predecessor $\Rightarrow$ ensures that preorders are simulations



$$R \;\Rightarrow\; \text{Glb-Closure of } R \cup \{\text{pre}(\{6\})\}$$



Initial claims:

1. Ubiquity of completeness properties of static analyses
   - Numerical abstractions
   - Abstract model checking
   - Probabilistic systems
   - ...interpolation, non-interference, obfuscation,...
   - Just ask and you get it! ☺
2. Completeness as a tool for designing new static analyses
   - When designing an abstraction, think a priori about its completeness properties
3. Completeness as a tool for understanding how existing static analyses work
   - Otherwise, completeness can explain a posteriori the genesis of your abstraction ☺