Static Program Analysis

Part 11 – abstract interpretation

https://cs.au.dk/~amoeller/spa/

Anders Møller

Computer Science, Aarhus University

Abstract interpretation

Abstract interpretation provides a solid mathematical foundation for reasoning about static program analyses

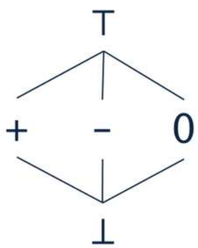
- Is my analysis **sound**? (Does it safely approximate the actual program behavior?)
- Is it as **precise** as possible for the currently used analysis lattice? If not, where can precision losses arise? Which precision losses can be avoided (without sacrificing soundness)?

Answering such questions requires a precise definition of the semantics of the programming language, and precise definitions of the analysis abstractions in terms of the semantics

Agenda

- **Collecting semantics**
- Abstraction and concretization
- Soundness
- Optimality
- Completeness
- Trace semantics

Sign analysis, recap

$Sign =$ 

$State = Var \rightarrow Sign$

$State^n$

$$\begin{aligned}
[\![v_1]\!] &= af_{v_1}([\![v_1]\!], \ldots, [\![v_n]\!]) \\
[\![v_2]\!] &= af_{v_2}([\![v_1]\!], \ldots, [\![v_n]\!]) \\
&\vdots \\
[\![v_n]\!] &= af_{v_n}([\![v_1]\!], \ldots, [\![v_n]\!])
\end{aligned}$$

$$af([\![v_1]\!], \ldots, [\![v_n]\!]) = \big(af_{v_1}([\![v_1]\!], \ldots, [\![v_n]\!]), \ldots, af_{v_n}([\![v_1]\!], \ldots, [\![v_n]\!])\big)$$

$af : State^n \rightarrow State^n$ is the analysis representation of the given program

$$[\![P]\!] = lfp(af)$$

Program semantics as constraint systems

$$ConcreteState = Var \hookrightarrow \mathbb{Z}$$

$$\{\![v]\!\} \subseteq ConcreteState$$

This is called a *reachable states collecting semantics*

The semantics of expressions

$$ceval: \ ConcreteState \times Exp \ \rightarrow \ \mathcal{P}(\mathbb{Z})$$

$$\begin{aligned}
&ceval(\rho, X) = \{\rho(X)\} \\
&ceval(\rho, I) = \{I\} \\
&ceval(\rho, \texttt{input}) = \mathbb{Z} \\
&ceval(\rho, E_1 \,\textsf{op}\, E_2) = \{v_1 \,\textsf{op}\, v_2 \mid v_1 \in ceval(\rho, E_1) \ \wedge \ v_2 \in ceval(\rho, E_2)\}
\end{aligned}$$

$$ceval(R, E) = \bigcup_{\rho \in R} ceval(\rho, E)$$

Successors and joins

$$csucc\colon ConcreteState \times Node \to \mathcal{P}(Node)$$

$$csucc(R, v) = \bigcup_{\rho \in R} csucc(\rho, v)$$

$$CJOIN(v) = \\ \{\rho \in ConcreteState \mid \exists w \in Node\colon \rho \in [\![w]\!] \wedge v \in csucc(\rho, w)\}$$

Semantics of statements

$$\{\![X\texttt{=}E]\!\} = \{\rho[X \mapsto z] \mid \rho \in CJOIN(v) \ \wedge \ z \in ceval(\rho, E)\}$$

$$\{\![\texttt{var}\ X_1\texttt{,}\ \ldots\texttt{,}X_n]\!\} = \{\rho[X_1 \mapsto z_1, \ldots, X_n \mapsto z_n] \mid \rho \in CJOIN(v) \wedge z_1 \in \mathbb{Z} \wedge \cdots \wedge z_n \in \mathbb{Z}\}$$

$$\{\![entry]\!\} = \{[\,]\}$$

$$\{\![v]\!\} = CJOIN(v)$$

The resulting constraint system

$$
\begin{aligned}
&[\![v_1]\!] = cf_{v_1}([\![v_1]\!], \ldots, [\![v_n]\!]) \\
&[\![v_2]\!] = cf_{v_2}([\![v_1]\!], \ldots, [\![v_n]\!]) \\
&\vdots \\
&[\![v_n]\!] = cf_{v_n}([\![v_1]\!], \ldots, [\![v_n]\!])
\end{aligned}
$$

$$cf_v \colon \big(\mathcal{P}(ConcreteState)\big)^n \to \mathcal{P}(ConcreteState)$$

$$cf(x_1, \ldots, x_n) = \big(cf_{v_1}(x_1, \ldots, x_n), \ldots, cf_{v_n}(x_1, \ldots, x_n)\big)$$

is the semantic representation of the given program

$$x = cf(x)$$

$$[\![P]\!] = lfp(cf)$$

Example

```
var x;
x = 0;
while (input) {
  x = x + 2;
}
```

	solution 1	solution 2
$[\![entry]\!]$	$\{[\,]\}$	$\{[\,]\}$
$[\![\texttt{var x}]\!]$	$\{[\texttt{x} \mapsto z] \mid z \in \mathbb{Z}\}$	$\{[\texttt{x} \mapsto z] \mid z \in \mathbb{Z}\}$
$[\![\texttt{x = 0}]\!]$	$\{[\texttt{x} \mapsto 0]\}$	$\{[\texttt{x} \mapsto 0]\}$
$[\![\texttt{input}]\!]$	$\{[\texttt{x} \mapsto z] \mid z \in \{0,2,4,\ldots\}\}$	$\{[\texttt{x} \mapsto z] \mid z \in \mathbb{Z}\}$
$[\![\texttt{x = x + 2}]\!]$	$\{[\texttt{x} \mapsto z] \mid z \in \{2,4,\ldots\}\}$	$\{[\texttt{x} \mapsto z] \mid z \in \mathbb{Z}\}$
$[\![exit]\!]$	$\{[\texttt{x} \mapsto z] \mid z \in \{0,2,4,\ldots\}\}$	$\{[\texttt{x} \mapsto z] \mid z \in \mathbb{Z}\}$

the least solution

Kleene's fixed point theorem for complete join morphisms

$f : L \to L$ is a *complete join morphism* if $f(\bigsqcup A) = \bigsqcup_{a \in A} f(a)$ for every $A \subseteq L$

If f is a complete join morphism:

$$lfp(f) = \bigsqcup_{i \geq 0} f^i(\bot)$$

(even when L has infinite height!)

(Proof: Exercise 11.8)

$(\mathcal{P}(ConcreteState))^n$ is a complete lattice (a product of a powerset lattice)

cf is a complete join morphism (Exercise 11.9)

Tarski's fixed-point theorem

In a complete lattice L, every monotone function $f: L \to L$ has a unique least fixed point given by $\bigsqcap\{x \in L \mid f(x) \sqsubseteq x\}$.

(Proof in Chapter 6)

cf is monotone (Exercise 11.3)

Semantics vs. analysis

```
var a,b,c;
a = 42;
b = 87;
if (input) {
  c = a + b;
} else {
  c = a - b;
}
```

$$\{\![\texttt{b = 87}]\!\} = \{[\mathsf{a} \mapsto 42, \mathsf{b} \mapsto 87, \mathsf{c} \mapsto z] \mid z \in \mathbb{Z}\}$$
$$\{\![\texttt{c = a - b}]\!\} = \{[\mathsf{a} \mapsto 42, \mathsf{b} \mapsto 87, \mathsf{c} \mapsto -45]\}$$
$$\{\![\mathit{exit}]\!\} = \{[\mathsf{a} \mapsto 42, \mathsf{b} \mapsto 87, \mathsf{c} \mapsto 129], [\mathsf{a} \mapsto 42, \mathsf{b} \mapsto 87, \mathsf{c} \mapsto -45]\}$$

$$[\![\texttt{b = 87}]\!] = [\mathsf{a} \mapsto +, \mathsf{b} \mapsto +, \mathsf{c} \mapsto \top]$$
$$[\![\texttt{c = a - b}]\!] = [\mathsf{a} \mapsto +, \mathsf{b} \mapsto +, \mathsf{c} \mapsto \top]$$
$$[\![\mathit{exit}]\!] = [\mathsf{a} \mapsto +, \mathsf{b} \mapsto +, \mathsf{c} \mapsto \top]$$

Agenda

- Collecting semantics
- **Abstraction and concretization**
- Soundness
- Optimality
- Completeness
- Trace semantics

Abstraction functions for sign analysis

$$\alpha_{\mathrm{a}}: \mathcal{P}(\mathbb{Z}) \to Sign$$
$$\alpha_{\mathrm{b}}: \mathcal{P}(ConcreteState) \to State$$
$$\alpha_{\mathrm{c}}: \big(\mathcal{P}(ConcreteState)\big)^n \to State^n$$

$$\alpha_{\mathrm{a}}(D) = \begin{cases} \bot & \text{if } D \text{ is empty} \\ + & \text{if } D \text{ is nonempty and contains only positive integers} \\ - & \text{if } D \text{ is nonempty and contains only negative integers} \\ \mathbf{0} & \text{if } D \text{ is nonempty and contains only the integer } 0 \\ \top & \text{otherwise} \end{cases}$$

for any $D \in \mathcal{P}(\mathbb{Z})$

$\alpha_{\mathrm{b}}(R) = \sigma$ where $\sigma(X) = \alpha_{\mathrm{a}}(\{\rho(X) \mid \rho \in R\})$

for any $R \subseteq ConcreteState$ and $X \in Var$

$\alpha_{\mathrm{c}}(R_1, \ldots, R_n) = (\alpha_b(R_1), \ldots, \alpha_b(R_n))$

for any $R_1, \ldots, R_n \subseteq ConcreteState$

Concretization functions for sign analysis

$$\gamma_{\mathrm{a}} : Sign \rightarrow \mathcal{P}(\mathbb{Z})$$
$$\gamma_{\mathrm{b}} : State \rightarrow \mathcal{P}(ConcreteState)$$
$$\gamma_{\mathrm{c}} : State^n \rightarrow \big(\mathcal{P}(ConcreteState)\big)^n$$

$$\gamma_{\mathrm{a}}(s) = \begin{cases} \emptyset & \text{if } s = \bot \\ \{1, 2, 3, \ldots\} & \text{if } s = + \\ \{-1, -2, -3, \ldots\} & \text{if } s = \texttt{-} \\ \{0\} & \text{if } s = \mathtt{0} \\ \mathbb{Z} & \text{if } s = \top \end{cases}$$
for any $s \in Sign$

$$\gamma_{\mathrm{b}}(\sigma) = \{\rho \in ConcreteState \mid \rho(\mathrm{X}) \in \gamma_{\mathrm{a}}(\sigma(\mathrm{X})) \text{ for all } X \in Var\}$$
for any $\sigma \in State$

$$\gamma_{\mathrm{c}}(\sigma_1, \ldots, \sigma_n) = (\gamma_{\mathrm{b}}(\sigma_1), \ldots, \gamma_{\mathrm{b}}(\sigma_n))$$
for any $(\sigma_1, \ldots, \sigma_n) \in State^n$

Monotonicity of abstraction and concretization functions

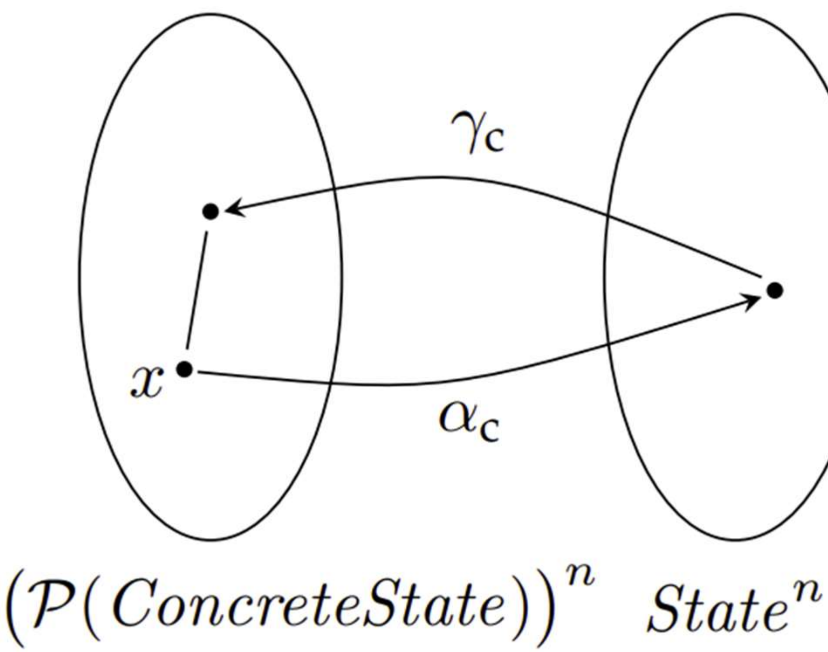
Concretatization functions are, like abstraction functions, naturally monotone.

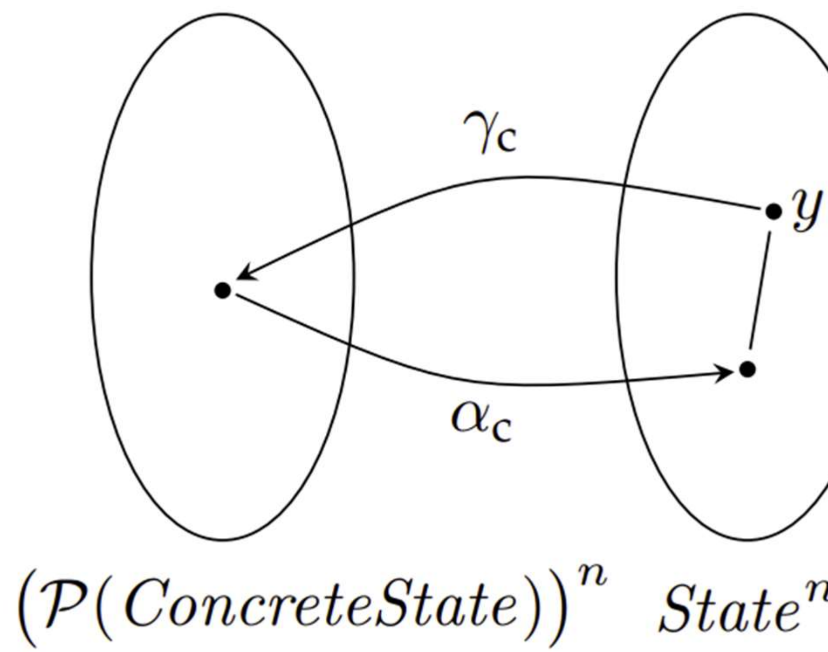
(A larger set of concrete values should correspond to a larger abstract state, and conversely)

Galois connections

The pair of monotone functions, α and γ, is called a *Galois connection* if

$$\gamma \circ \alpha \text{ is extensive}$$
$$\alpha \circ \gamma \text{ is reductive}$$





all three pairs of abstraction and concretization functions (α_a, γ_a), (α_b, γ_b), and (α_c, γ_c) from the sign analysis example are Galois connections (Exercise 11.13)

Galois connections

For Galois connections, the concretization function uniquely determines the abstraction function and vice versa:

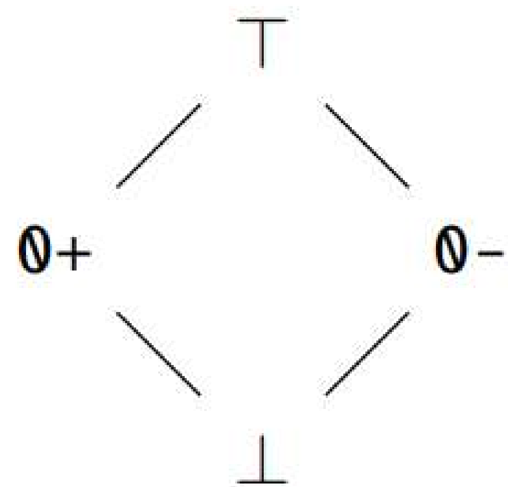
$$\gamma(y) = \bigsqcup\{x \in L_1 \mid \alpha(x) \sqsubseteq y\}$$

$$\alpha(x) = \bigsqcap\{y \in L_2 \mid x \sqsubseteq \gamma(y)\}$$

(Proof: Exercise 11.20)

Galois connections

For this lattice, given the "obvious" concretization function,
is there an abstraction function such that the concretization function
and the abstraction function form a Galois connection?



how should we define $\alpha_{\mathrm{a}}(\{0\})$? (Exercise 11.22)

Representation functions

$$\beta : \mathbb{Z} \to Sign$$

$$\beta(d) = \begin{cases} + & \text{if } d > 0 \\ - & \text{if } d < 0 \\ 0 & \text{if } d = 0 \end{cases}$$

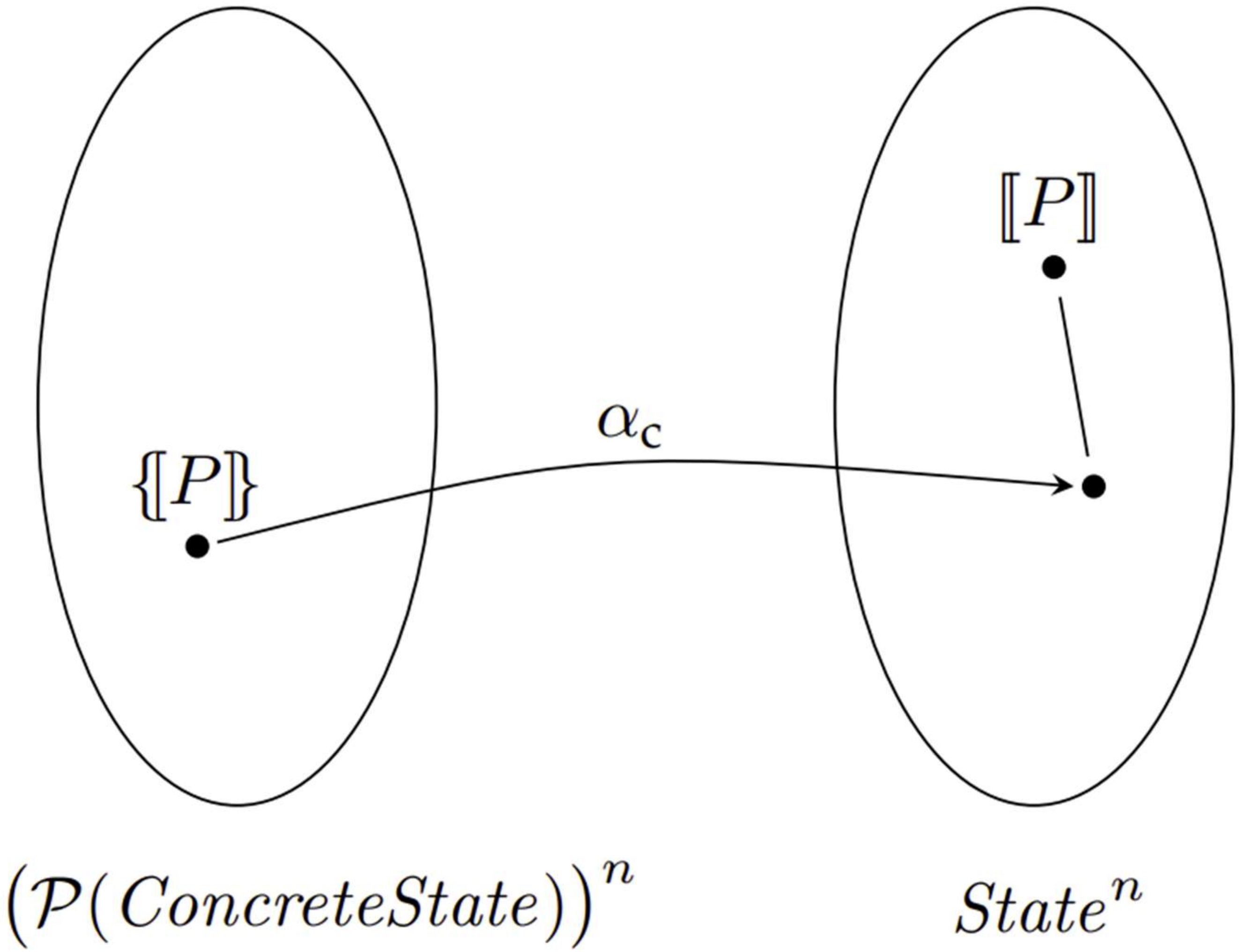
$$\alpha_{\mathrm{a}}(D) = \bigsqcup \{\beta(d) \mid d \in D\}$$

Agenda

- Collecting semantics
- Abstraction and concretization
- **Soundness**
- Optimality
- Completeness
- Trace semantics

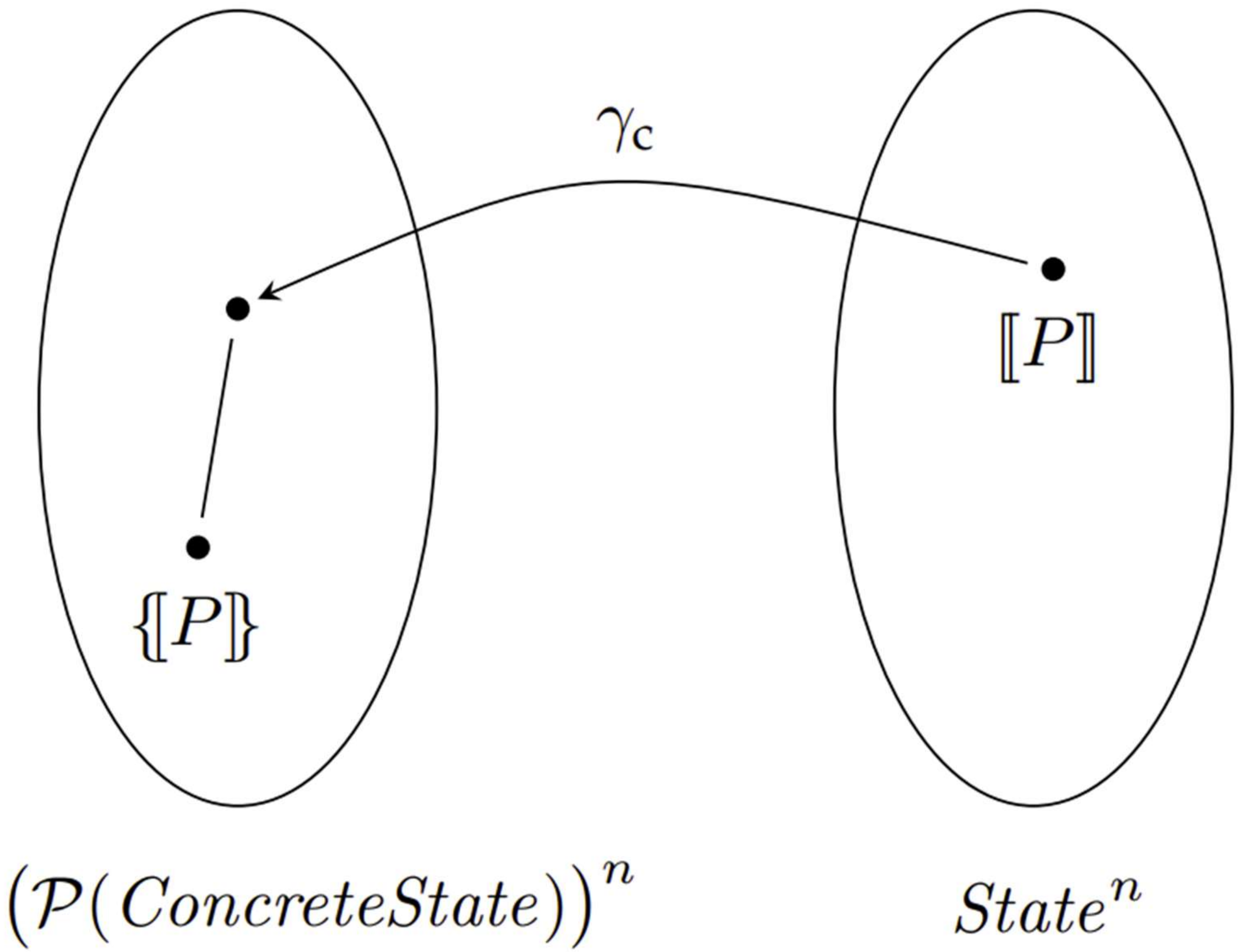
Soundness

$$\alpha(\{\![P]\!\}) \sqsubseteq [\![P]\!]$$



Soundness

$$\{\![P]\!\} \sqsubseteq \gamma(\llbracket P \rrbracket)$$



(Exercise 11.17)

Sound abstractions

$$\alpha_{\mathsf{a}}(ceval(R, E)) \sqsubseteq eval(\alpha_{\mathsf{b}}(R), E)$$

$$csucc(R, v) \subseteq succ(v) \text{ for any } R \subseteq ConcreteState$$

$$\alpha_{\mathsf{b}}(CJOIN(v)) \sqsubseteq JOIN(v)$$

$$\text{if } \alpha_{\mathsf{b}}(\{[\![w]\!]\}) \sqsubseteq [\![w]\!] \text{ for all } w \in Node$$

(Exercise 11.31 and Exercise 11.32)

Sound abstractions

if v represents an assignment statement $X = E$:

$$cf_v(\{\![v_1]\!\}, \ldots, \{\![v_n]\!\}) = \{\rho[X \mapsto z] \mid \rho \in CJOIN(v) \ \wedge \ z \in ceval(\rho, E)\}$$
$$af_v([\![v_1]\!], \ldots, [\![v_n]\!]) = \sigma[X \mapsto eval(\sigma, E)] \text{ where } \sigma = JOIN(v)$$

$$\alpha_{\mathsf{b}}(cf_v(R_1, \ldots, R_n)) \sqsubseteq af_v(\alpha_{\mathsf{b}}(R_1), \ldots, \alpha_{\mathsf{b}}(R_n))$$

(Exercise 11.33)

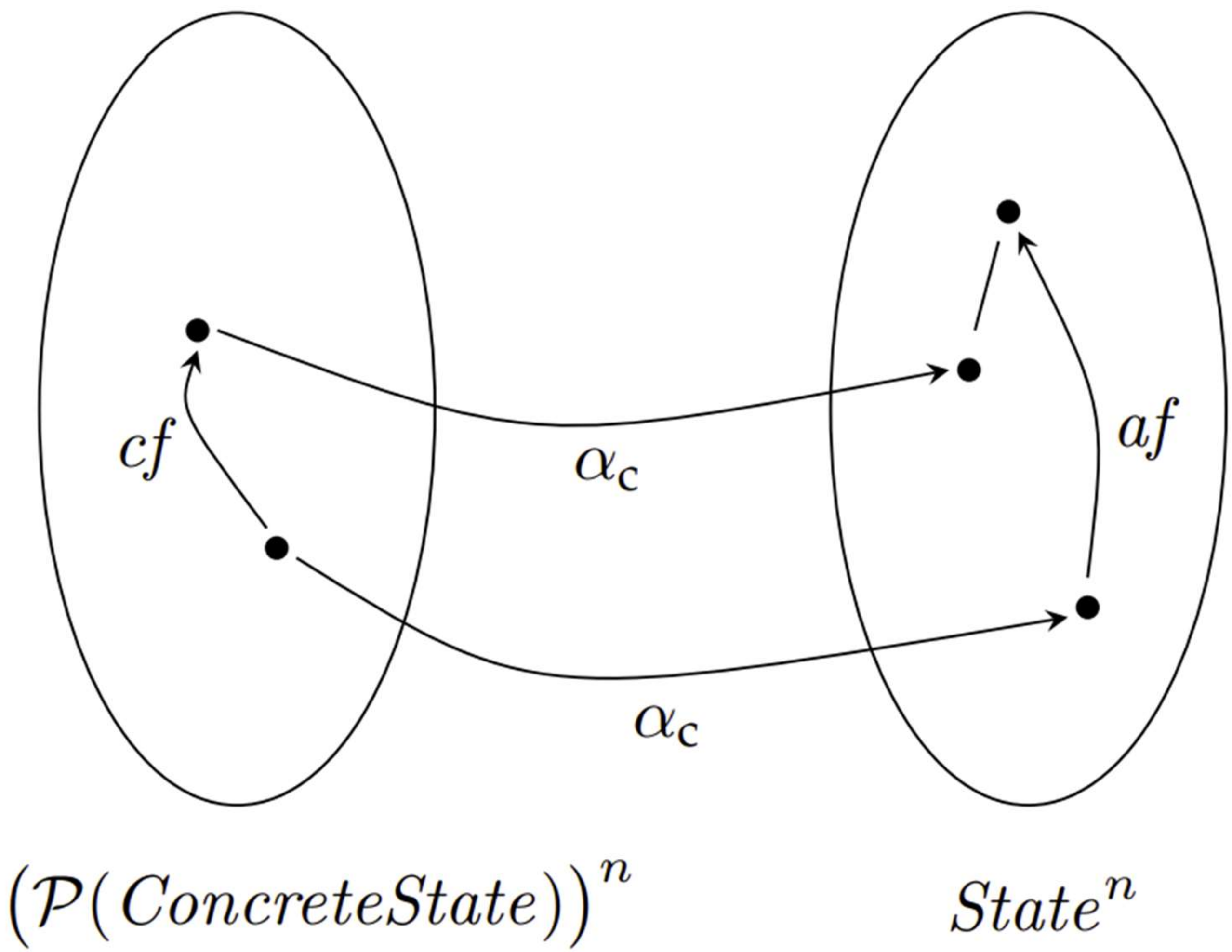
The two constraint systems

$$cf(\{\![v_1]\!\}, \ldots, \{\![v_n]\!\}) = \left(cf_{v_1}(\{\![v_1]\!\}, \ldots, \{\![v_n]\!\}), \ldots, cf_{v_n}(\{\![v_1]\!\}, \ldots, \{\![v_n]\!\})\right)$$

$$af([\![v_1]\!], \ldots, [\![v_n]\!]) = \left(af_{v_1}([\![v_1]\!], \ldots, [\![v_n]\!]), \ldots, af_{v_n}([\![v_1]\!], \ldots, [\![v_n]\!])\right)$$

Sound abstractions

$$\alpha_{\mathsf{c}}(cf(R_1, \ldots, R_n)) \sqsubseteq af(\alpha_{\mathsf{c}}(R_1, \ldots, R_n))$$



Sound abstractions

$$\alpha \circ cf \sqsubseteq af \circ \alpha$$

$$cf \circ \gamma \sqsubseteq \gamma \circ af$$

Equivalent, if α and γ form a Galois connection

(Proof: Exercise 11.34)

The soundness theorem

If L_1 and L_2 are complete lattices with a concretization function $\gamma : L_2 \to L_1$, $cf : L_1 \to L_1$ and $af : L_2 \to L_2$ are monotone, and af is a sound abstraction of cf with respect to γ, i.e., $cf \circ \gamma \sqsubseteq \gamma \circ af$, then $lfp(cf) \sqsubseteq \gamma(lfp(af))$.

Specifying and proving soundness of an analysis

1. Define the analysis (lattice and constraint rules), prove monotonicity

2. Define the collecting semantics, prove monotonicity

3. Define Galois connection (e.g., as concretization function) between the analysis lattice and the semantic lattice

4. Prove sound abstractions (analysis constraints vs. semantic constraints)

5. Soundness then follows from the soundness theorem

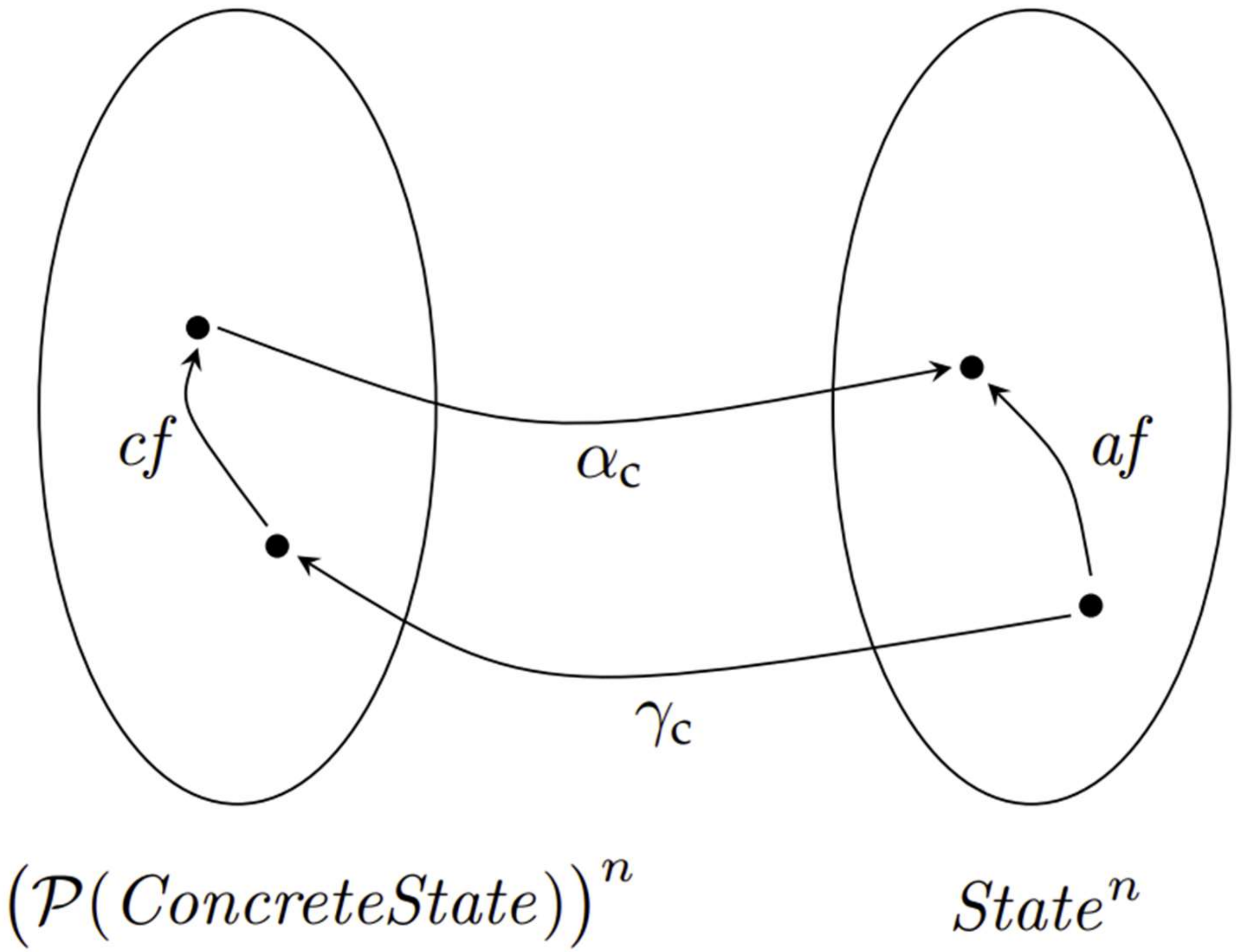
Agenda

- Collecting semantics
- Abstraction and concretization
- Soundness
- **Optimality**
- Completeness
- Trace semantics

Optimal abstractions

af is an optimal abstraction of *cf* if

$$af = \alpha \circ cf \circ \gamma$$



(compare with slide 28)

Optimal abstractions in sign analysis?

$\widehat{*}$ is optimal:

$$s_1 \widehat{*} s_2 = \alpha_a(\gamma_a(s_1) \cdot \gamma_a(s_2))$$

eval is *not* optimal:

$$\sigma(\mathbf{x}) = \top$$

$$eval(\sigma, \mathtt{x-x}) = \top$$

$$\alpha_b(ceval(\gamma_b(\sigma), \mathtt{x-x})) = \mathbf{0}$$

Even if we could make *eval* optimal, the analysis result is not always optimal:

```
x = input;
y = x;
z = x - y;
```

Agenda

- Collecting semantics
- Abstraction and concretization
- Soundness
- Optimality
- **Completeness**
- Trace semantics

Completeness

$$\alpha(\{\![P]\!\}) \sqsupseteq [\![P]\!]$$

(compare with slide 23)

Sound *and* complete: $$\alpha(\{\![P]\!\}) = [\![P]\!]$$

(Intuitively, the analysis result is the most precise possible for the currently used lattice)

Not the same as $\{\![P]\!\} = \gamma([\![P]\!])$ (called “exact”)

(Intuitively, the analysis result exactly captures the semantics of the program)

Completeness in sign analysis?

$\widehat{*}$ is complete:

$$\alpha_a(D_1)\widehat{*}\alpha_a(D_2) \sqsubseteq \alpha_a(D_1 \cdot D_2)$$

$\widehat{+}$ is *not* complete

$$\alpha_a(D_1)\widehat{+}\alpha_a(D_2) \not\sqsubseteq \alpha_a(D_1 + D_2)$$

Sign analysis is sound and complete for *some* programs, but not for all programs

Agenda

- Collecting semantics
- Abstraction and concretization
- Soundness
- Optimality
- Completeness
- **Trace semantics**

Limitations of the reachable states collecting semantics

The reachable states collecting semantics “collects” a set of concrete states for each program point

$$ConcreteState = Var \hookrightarrow \mathbb{Z}$$

$$\{\![v]\!\} \subseteq ConcreteState$$

That is not always sufficient – a trivial example:

```
main(x) {
  return x;
}
```

For this program, the collecting semantics doesn’t allow us to express properties such as “in any execution, the return value is the same as the input value”

Trace semantics

$$Trace = (Node \times ConcreteState)^*$$

$$ct_v : ConcreteState \to \mathcal{P}(ConcreteState)$$

$$ct_{\text{X=E}}(\rho) = \{\rho[X \mapsto z] \mid z \in ceval(\rho, E)\}$$

$ct_v(\rho) = \{\rho\}$ for other kinds of nodes

$$\langle\!|P|\!\rangle \in \mathcal{P}(Trace)$$

$$(entry, []) \in \langle\!|P|\!\rangle$$

$$\pi \cdot (v, \rho) \in \langle\!|P|\!\rangle \ \wedge \ v' \in csucc(\rho, v) \ \wedge \ \rho' \in ct_{v'}(\rho) \implies \pi \cdot (v, \rho) \cdot (v', \rho') \in \langle\!|P|\!\rangle$$

Example

```
main(x) {
  return x;
}
```

$$\langle\!|P|\!\rangle = \{(entry, [\mathbf{x} \mapsto 0]) \cdot (\mathbf{return\ \ x}, [\mathbf{x} \mapsto 0]) \cdot (exit, [\mathbf{x} \mapsto 0]),\\ (entry, [\mathbf{x} \mapsto 1]) \cdot (\mathbf{return\ \ x}, [\mathbf{x} \mapsto 1]) \cdot (exit, [\mathbf{x} \mapsto 1]),\\ \ldots\}$$

Reachable states is an abstraction of traces

the relation between the reachable states collecting semantics
and the trace semantics can be expressed as a Galois connection induced by an abstraction function

$$\alpha_{\mathsf{t}} : \mathcal{P}(\mathit{Trace}) \to \big(\mathcal{P}(\mathit{ConcreteState})\big)^n$$

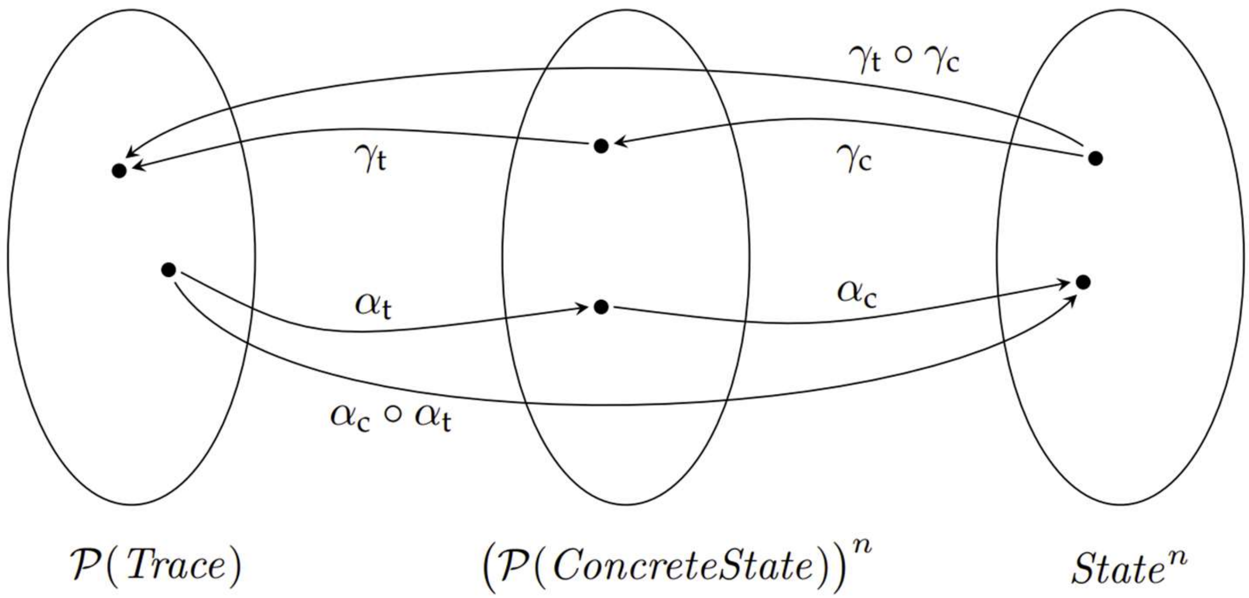
defined by

$$\alpha_{\mathsf{t}}(T) = (R_1, \ldots, R_n)$$

$$\text{where } R_i = \{\rho \mid \ \cdots \cdot (v_i, \rho) \cdot \cdots \in T\} \text{ for each } i = 1, \ldots, n.$$

Composition of Galois connections

Exercise 11.57: Let $\alpha_1 : L_1 \to L_2$, $\gamma_1 : L_2 \to L_1$, $\alpha_2 : L_2 \to L_3$, and $\gamma_2 : L_3 \to L_2$. Assume both (α_1, γ_1) and (α_2, γ_2) are Galois connections. Prove that $(\alpha_2 \circ \alpha_1, \gamma_1 \circ \gamma_2)$ is then also a Galois connection.



Exercise 11.58: Prove that the reachable states collecting semantics is sound with respect to the trace semantics. (Even though the collecting semantics is not a computable analysis, we can still apply the notion of soundness and the proof techniques from Section 11.3.)

Soundness of reaching definitions analysis

Instrumented trace semantics:

$$Trace_{RD} = (Node \times ConcreteState \times ReachingDef)^*$$

$$ReachingDef = Var \hookrightarrow Node$$

$$(entry, [], []) \in \langle\!| P |\!\rangle_{RD}$$

$$\pi \cdot (v, \rho, \delta) \in \langle\!| P |\!\rangle_{RD} \;\wedge\; v' \in csucc(\rho, v) \;\wedge\; \rho' \in ct_{v'}(\rho) \implies \pi \cdot (v, \rho, \delta) \cdot (v', \rho', \delta') \in \langle\!| P |\!\rangle_{RD}$$

$$\delta' = \begin{cases} \delta[X \mapsto v'] & \text{if } v' \text{ is an assignment } X = E \\ \delta & \text{otherwise} \end{cases}$$

Soundness of reaching definitions analysis

Abstraction:

$$\alpha_{RD} : \mathcal{P}(Trace_{RD}) \to (\mathcal{P}(Node))^n$$

$$\alpha_{RD}(T) = (R_1, \ldots, R_n)$$

$$R_i = \{w \mid \cdots (v_i, \rho, \delta) \cdots \in T \text{ where } \delta(X) = w \text{ for some } \rho, \delta, X\}$$

Soundness:

$$\alpha_{RD}(\langle\!|P|\!\rangle_{RD}) \sqsubseteq [\![P]\!]_{RD}$$

Soundness proofs for other analyses

Exercise 11.61: Use the approach from Section 11.3 and an appropriate instrumented trace semantics to prove that the available expressions analysis from Section 5.5 is sound. (This is more tricky than Exercise 11.60, because available expressions analysis is a "must" analysis!)

Exercise 11.62: Use the approach from Section 11.3 and an appropriate instrumented trace semantics to prove that the live variables analysis from Section 5.4 is sound. As part of this, you need to specify an appropriate collecting semantics that formally captures what it means for a variable to be live (see the informal definition in Section 5.4). (This is more tricky than Exercise 11.60, because live variables analysis is a "backward" analysis!)

Conclusions

Abstract interpretation provides a solid mathematical foundation for reasoning about soundness and precision of static program analyses

We need

- the static analysis (the analysis lattices and constraint rules)
- the language semantics (a suitable collecting semantics)
- abstraction/concretization functions that specify the meaning of the elements in the analysis lattice in terms of the semantic lattice

... and then

- if each constituent of the analysis is a *sound abstraction* of its semantic counterpart, then the analysis is sound (according to the soundness theorem)
- if an abstraction is *optimal*, then it is as precise as possible (yet sound), relative to the choice of analysis lattice
- if the analysis is *sound and complete*, then the analysis result is as precise as possible (yet sound), relative to the choice of analysis lattice