POPL 2014: 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages
San Diego, CA, USA January 22-24, 2014

A Galois connection calculus for abstract interpretation

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

Radhia Cousot
rcousot@ens.fr di.ens.fr/~rcousot

Thanks

We warmly thank

- the ACM SIGPLAN Awards Committee for awarding us the 2013 Programming Languages Achievement Award, and
- the whole programming language community for its warmhearted support for nearly 4 decades.

Abstract Interpretation

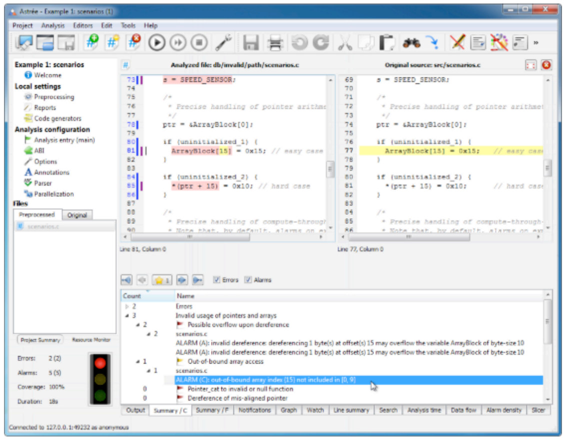
- A mathematical framework for reasoning on program behaviors (useful in program semantics, transformation/compilation, static analysis, verification, *etc*)
- The theory aims at being general (neither depending on specific languages, properties, specification methods, *etc*)
- The theory aims at being applicable to real-life software, hardware, and computer systems (must scale up: precise analysis is very easy in the small and extremely difficult in the large)

Part I

Industrial applications

Astrée

- Commercially available: www.absint.com/astree/

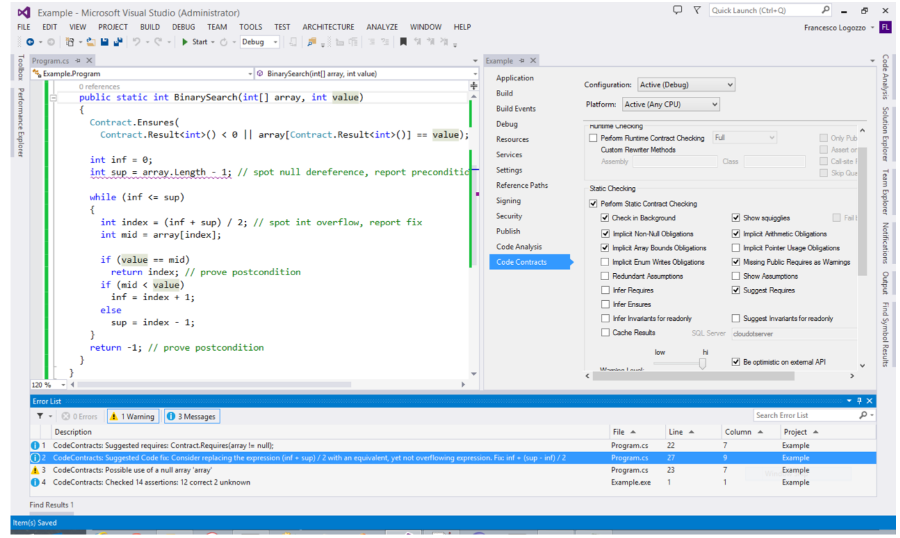


- Effectively used in production to qualify truly large and complex software in transportation, communications, medicine, *etc*

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

Code Contract Static Checker (cccheck)

- Available within MS Visual Studio



Manuel Fähndrich, Francesco Logozzo: **Static Contract Checking with Abstract Interpretation.** *FoVeOOS 2010*: 10-30

Comments on screenshot (courtesy Francesco Logozzo)

- A screenshot from Clousot/cccheck on the classic binary search.
- The screenshot shows from left to right and top to bottom
  1. C# code + CodeContracts with a buggy BinarySearch
  2. cccheck integration in VS (right pane with all the options integrated in the VS project system)
  3. cccheck messages in the VS error list
- The features of cccheck that it shows are:
  1. basic abstract interpretation:
     a. the loop invariant to prove the array access correct and that the arithmetic operation may overflow is inferred fully automatically
     b. different from deductive methods as e.g. ESC/Java or Boogie where the loop invariant must be provided by the end-user
  2. inference of necessary preconditions:
     a. Clousot finds that array may be null (message 3)
     b. Clousot suggests and propagates a necessary precondition invariant (message 1)
  3. array analysis (+ disjunctive reasoning):
     a. to prove the postcondition should infer property of the content of the array
     b. please note that the postcondition is true even if there is no precondition requiring the array to be sorted.
  4. verified code repairs:
     a. from the inferred loop invariant does not follow that index computation does not overflow
     b. suggest a code fix for it (message 2)

Part II

A short introduction to abstract interpretation

Patrick Cousot, Radhia Cousot: **Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints.** POPL 1977: 238-252

Patrick Cousot, Radhia Cousot: **Systematic Design of Program Analysis Frameworks.** POPL 1979: 269-282

Properties and their Abstractions

Concrete properties

- A concrete property is represented by the set of elements which have that property:
  - universe (set of elements) $\mathscr{D}$ (e.g. a semantic domain)
  - properties of these elements: $P \in \wp(\mathscr{D})$
  - x has property P is $x \in P$
- $\langle \wp(\mathscr{D}), \subseteq, \cup, \cap, \ldots \rangle$ is a *complete lattice* for inclusion $\subseteq$ (i.e. *logical implication*)

Abstract properties

- Abstract properties: $\overline{P} \in \mathscr{A}$
- Abstract domain $\mathscr{A}$: encodes a subset of the concrete properties (e.g. a program logic, type terms, linear algebra, *etc*)
- Poset: $\langle \mathscr{A}, \sqsubseteq, \sqcup, \sqcap, \ldots \rangle$
- Partial order: $\sqsubseteq$ is *abstract implication*

Concretization

- Concretization $\gamma \in \mathscr{A} \longrightarrow \wp(\mathscr{D})$
- $\gamma(\overline{P})$ is the semantics (concrete meaning) of $\overline{P}$
- γ is *increasing* (so $\sqsubseteq$ abstracts $\subseteq$)

Best abstraction

- A concrete property $P \in \wp(\mathscr{D})$ has a best abstraction $\overline{P} \in \mathscr{A}$ iff
  - it is sound (over-approximation):

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

  - and more precise than any sound abstraction:

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

- The best abstraction is unique (by antisymmetry)
- Under-approximation is order-dual

Galois connection

- Any $P \in \wp(\mathscr{D})$ has a (unique) best abstraction $\alpha(P)$ in $\mathscr{A}$ if and only if

$$\forall P \in \wp(\mathscr{D}) : \forall Q \in \mathscr{A} : \ \alpha(P) \sqsubseteq Q \iff P \subseteq \gamma(Q)$$

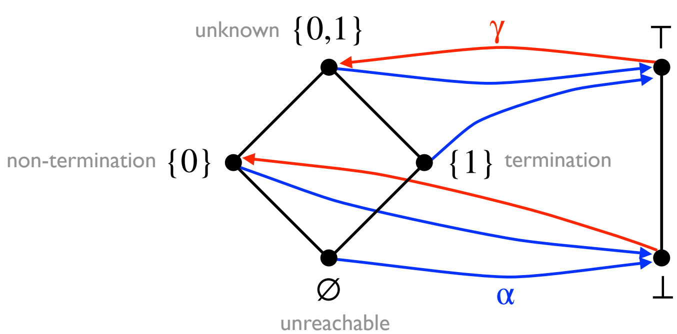
$\Rightarrow$: *over-approximation*
$\Leftarrow$: *best abstraction*

written

$$\langle \wp(\mathscr{D}), \subseteq \rangle \xleftrightarrows[\alpha]{\gamma} \langle \mathscr{A}, \sqsubseteq \rangle$$

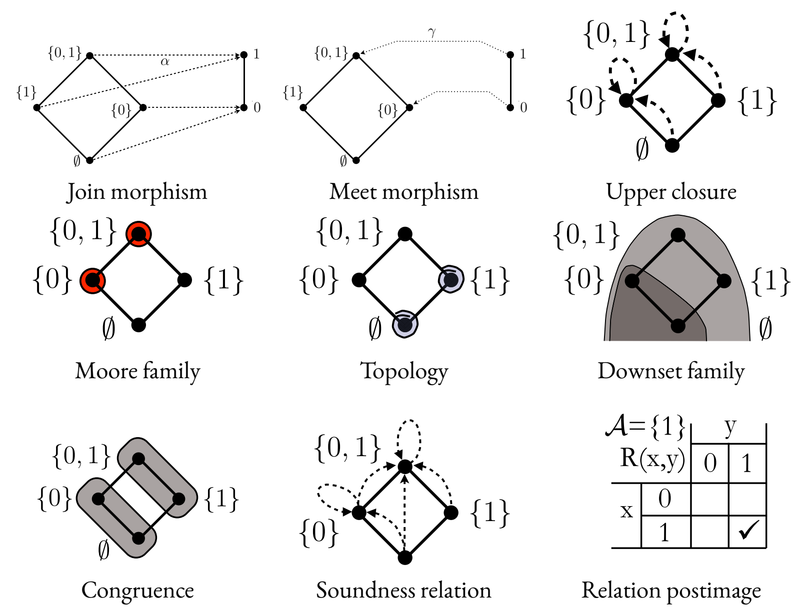
Simple example

- Neednessǀstrictness analysis (80's)



- Similar abstraction for scalable harware symbolic trajectory evaluation STE (90)

Equivalent mathematical structures



Abstraction of the Semantics of Programming Languages

Sound semantics abstraction

- program $\mathsf{P} \in \mathbb{L}$ programming language
- standard semantics $\mathsf{S}[\![\mathsf{P}]\!] \in \mathcal{D}$ semantic domain
- collecting semantics $\{\mathsf{S}[\![\mathsf{P}]\!]\} \in \wp(\mathcal{D})$ semantic property
- abstract semantics $\overline{\mathsf{S}}[\![\mathsf{P}]\!] \in \mathscr{A}$ abstract domain
- concretization $\gamma \in \mathscr{A} \longrightarrow \wp(\mathcal{D})$
- soundness $\{\mathsf{S}[\![\mathsf{P}]\!]\} \subseteq \gamma(\overline{\mathsf{S}}[\![\mathsf{P}]\!])$

i.e. $\mathsf{S}[\![\mathsf{P}]\!] \in \gamma(\overline{\mathsf{S}}[\![\mathsf{P}]\!])$, P *has abstract property* $\overline{\mathsf{S}}[\![\mathsf{P}]\!]$

Best abstract semantics

- If $\langle \wp(\mathcal{D}), \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathscr{A}, \sqsubseteq \rangle$ then the best abstract semantics is the abstraction of the collecting semantics

$$\overline{\mathsf{S}}[\![\mathsf{P}]\!] \triangleq \alpha(\{\mathsf{S}[\![\mathsf{P}]\!]\})$$

- Proof:
  - It is *sound*: $\overline{\mathsf{S}}[\![\mathsf{P}]\!] \triangleq \alpha(\{\mathsf{S}[\![\mathsf{P}]\!]\}) \sqsubseteq \overline{\mathsf{S}}[\![\mathsf{P}]\!] \Longrightarrow \{\mathsf{S}[\![\mathsf{P}]\!]\} \subseteq \gamma(\overline{\mathsf{S}}[\![\mathsf{P}]\!]) \Longrightarrow \mathsf{S}[\![\mathsf{P}]\!] \in \gamma(\overline{\mathsf{S}}[\![\mathsf{P}]\!])$
  - It is the *most precise*: $\mathsf{S}[\![\mathsf{P}]\!] \in \gamma(\overline{\overline{\mathsf{S}}}[\![\mathsf{P}]\!]) \Longrightarrow \{\mathsf{S}[\![\mathsf{P}]\!]\} \subseteq \gamma(\overline{\overline{\mathsf{S}}}[\![\mathsf{P}]\!]) \Longrightarrow \overline{\mathsf{S}}[\![\mathsf{P}]\!] \triangleq \alpha(\{\mathsf{S}[\![\mathsf{P}]\!]\}) \sqsubseteq \overline{\overline{\mathsf{S}}}[\![\mathsf{P}]\!]$ ■

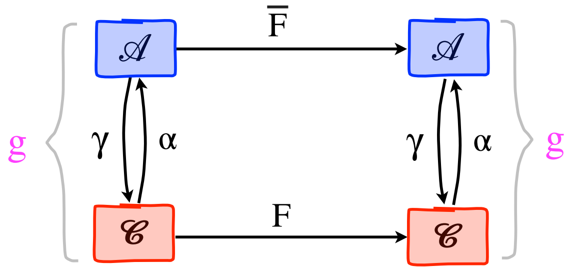
Calculational design of the abstract semantics

- The (standard hence collecting) semantics are defined by composition of mathematical structures (such as set unions, products, functions, fixpoints, *etc*)
- If you know the best abstraction of properties, you also know best abstractions of these mathematical structures
- So, by composition, you also know the best abstraction of the collecting semantics ⇝ calculational design of the abstract semantics
- Orthogonally, there are many styles of
  - *semantics* (traces, relations, transformers,…)
  - *induction* (transitional, structural, segmentation)
  - *presentations* (fixpoints, equations, constraints, rules [CAV 1995])

Example: functional connector

- If $g = \langle \mathscr{C}, \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathscr{A}, \sqsubseteq \rangle$ then

$$g \Longmapsto g = \langle \mathscr{C} \stackrel{\nearrow}{\longrightarrow} \mathscr{C}, \subseteq \rangle \underset{\lambda F.\alpha \circ F \circ \gamma}{\overset{\lambda \overline{F}.\gamma \circ \overline{F} \circ \alpha}{\leftrightarrows}} \langle \mathscr{A} \stackrel{\nearrow}{\longrightarrow} \mathscr{A}, \sqsubseteq \rangle$$



($\Longmapsto$ is a called a *Galois connector*)

Fixpoint abstraction

- Best abstraction (completeness case)

  if $\alpha \circ F = \overline{F} \circ \alpha$ then $\overline{F} = \alpha \circ F \circ \gamma$ and $\alpha(\mathrm{lfp}\ F) = \mathrm{lfp}\ \overline{F}$

  e.g. semantics, proof methods, static analysis of finite state systems

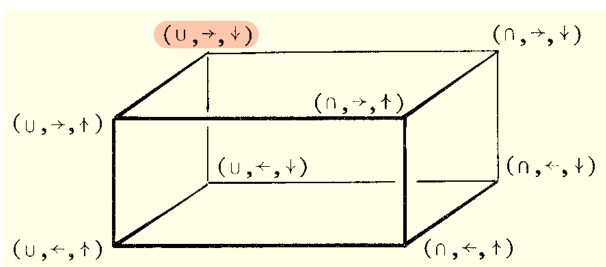
- Best approximation (incompleteness case)

  if $\overline{F} = \alpha \circ F \circ \gamma$ but $\alpha \circ F \sqsubseteq \overline{F} \circ \alpha$ then $\alpha(\mathrm{lfp}\ F) \sqsubseteq \mathrm{lfp}\ \overline{F}$

  e.g. static analysis of infinite state systems

- idem for equations, constraints, rule-based deductive systems, *etc*

Duality

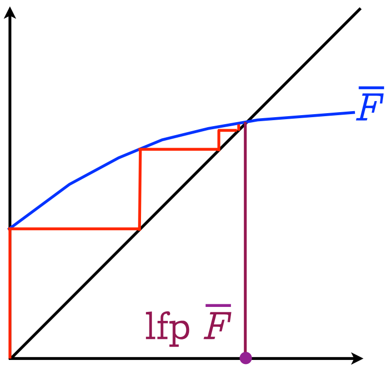


- Order duality: join ($\cup$) or meet ($\cap$)
- Inversion duality: forward ($\rightarrow$) or backward ($\leftarrow = (\rightarrow)^{-1}$)
- Fixpoint duality: least ($\downarrow$) or greatest ($\uparrow$)

Patrick Cousot, Radhia Cousot: **Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints.** POPL 1977: 238-252

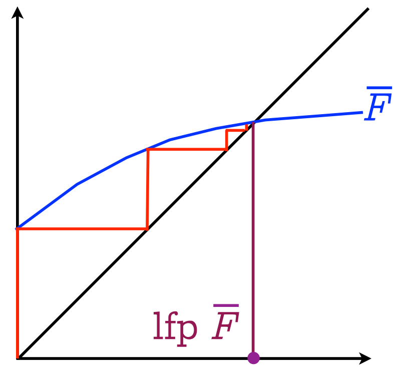
Abstract Induction
(in non-Noetherian domains)

Convergence acceleration

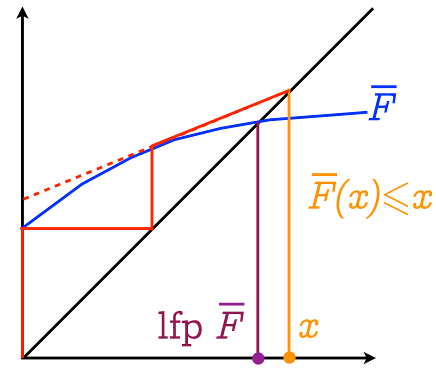


Infinite iteration

Convergence acceleration

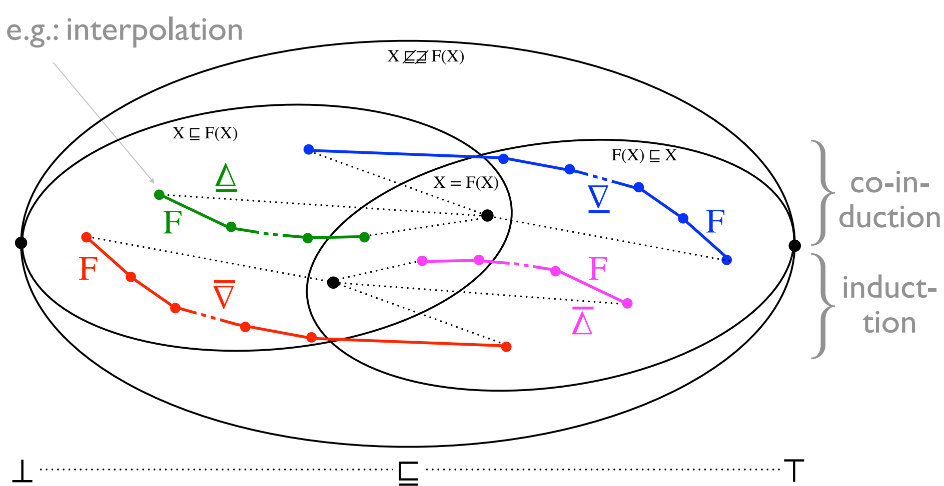


Infinite iteration



Accelerated iteration with widening
(e.g. with a widening based on the derivative as in Newton-Raphson method)

[Semi-]dual abstract induction methods



(separate from termination conditions)

Examples of widening/narrowing

- Abstract induction for intervals:
  - a widening [1,2]

```
(x ∇̄ y) = cas x ∈ Va, y ∈ Va dans
          ├ □, ? ==> y ;
          ├ ?, □ ==> x ;
          ├ [n1,m1],[n2,m2] ==>
          │      [si n2 < n1 alors -∞ sinon n1 fsi ;
          │          si m2 > m1 alors +∞ sinon m1 fsi] ;
          fincas ;
```

$$[a_1, b_1] \mathbin{\bar{\nabla}} [a_2, b_2] =$$
$$[\underline{\text{if}}\ a_2 < a_1\ \underline{\text{then}}\ -\infty\ \underline{\text{else}}\ a_1\ \underline{\text{fi}},$$
$$\underline{\text{if}}\ b_2 > b_1\ \underline{\text{then}}\ +\infty\ \underline{\text{else}}\ b_1\ \underline{\text{fi}}]$$

  - a narrowing [2]

$$[a_1,b_1] \mathbin{\bar{\Delta}} [a_2,b_2] =$$
$$[\underline{\text{if}}\ a_1 = -\infty\ \underline{\text{then}}\ a_2\ \underline{\text{else}}\ \text{MIN}\ (a_1,a_2),$$
$$\underline{\text{if}}\ b_1 = +\infty\ \underline{\text{then}}\ b_2\ \underline{\text{else}}\ \text{MAX}\ (b_1,b_2)]$$

[1] Patrick Cousot, Radhia Cousot: Vérification statique de la cohérence dynamique des programmes, Rapport du contrat IRIA-SESORI No 75-032, 23 septembre 1975.
[2] Patrick Cousot, Radhia Cousot: Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. POPL 1977: 238-252

On widening/narrowing/and their duals

- Because the abstract domain is non-Noetherian, *any* widening/narrowing/duals can be *strictly* improved infinitely many times (*i.e.* no best widening)

  E.g. widening with thresholds [1]

  $$\forall x \in \bar{L}_2, \perp \nabla_2(j)\, x = x\, \nabla_2(j) \perp = x$$

  $$\begin{aligned}[l_1, u_1]\, \nabla_2(j)\, [l_2, u_2] &= [\mathit{if}\ 0 \leq l_2 < l_1\ \mathit{then}\ 0\ \mathit{elsif}\ l_2 < l_1\ \mathit{then}\ -b-1\ \mathit{else}\ l_1\ \mathit{fi},\\ &\quad \mathit{if}\ u_1 < u_2 \leq 0\ \mathit{then}\ 0\ \mathit{elsif}\ u_1 < u_2\ \mathit{then}\ b\ \mathit{else}\ u_1\ \mathit{fi}]\end{aligned}$$

- Any *terminating* widening is *not* increasing (in its 1st parameter)
- Any abstraction done with Galois connections *can be done* with widenings (*i.e.* a widening calculus)

[1] Patrick Cousot, Semantic foundations of program analysis, Ch. 10 of Program flow analysis: theory and practice, N. Jones & S. Muchnich (eds), Prentice Hall, 1981.

Summary

- The specification of abstract semantics/proof methods/ transformers/verifiers/static analyzers reduces to the choice of:
  - The standard semantics domain $\mathscr{D}$
  - The concrete fixpoint transformers $\mathsf{F} \in \wp(\mathscr{D}) \xrightarrow{\nearrow} \wp(\mathscr{D})$
  - The abstraction $\langle \wp(\mathscr{D}), \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathscr{A}, \sqsubseteq \rangle$
  - The abstract induction ($\overline{\nabla}$, $\overline{\Delta}$, $\underline{\nabla}$, $\underline{\Delta}$)
- Maybe dualities and fixpoint combinations
- Calculational design of the verifier/analyzer by sound abstraction of the collecting semantics preferred to empirical design with a posteriory soundness checks, if any

Part III

A Galois connection calculus for abstract interpretation

How to specify $\langle \wp(\mathscr{D}), \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathscr{A}, \sqsubseteq \rangle$?

Specifying posets

$$\langle \wp(\mathscr{D}), \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathscr{A}, \sqsubseteq \rangle$$

Specifying the concrete/abstract domains

- Program variables: $x, \ldots \in \mathbb{X}$
- Program labels: $\ell, \ldots \in \mathbb{L}$
- Elements: $e \in \mathbb{E}$
  $e ::= \text{true} \mid 1 \mid \infty \mid \mathrm{x} \mid \ell \mid -e \mid \ldots$
- Sets: $s \in \mathbb{S}$
  $s ::= \mathbb{B} \mid \mathbb{Z} \mid \mathbb{X} \mid \mathbb{L} \mid \{e\} \mid [e, e]_o \mid$
  $\mathbb{I}(s, o) \mid s^\infty \mid s \cup s \mid s \mapsto s \mid$
  $s \times s \mid \wp(s) \mid \ldots$
- Partial orders: $o \in \mathbb{O}$
  $o ::= \Rightarrow \mid \Leftrightarrow \mid \leq \mid \subseteq \mid \oplus\!\!\!\subseteq \mid = \mid$
  $o^{-1} \mid o_1 \times o_2 \mid \dot{o} \mid \ddot{o} \mid \ldots$

Specifying the concrete/abstract domains (cont'd)

- Posets: $p \in \mathbb{P}$
  $p ::= \langle s,\ o\rangle$
- Trivial set-theoretic semantics (with errors)

(dynamic) error

Example: semantic properties of a simple imperative language

- values: $\langle \mathcal{V},\ \leq\rangle$ (*e.g.* $\langle \mathbb{Z},\ \leqslant\rangle$ or $\langle[\mathtt{minint}, \mathtt{maxint}],\ \leqslant\rangle$
- environments: $\mathcal{M} \triangleq \mathbb{X} \mapsto \mathcal{V}$
- states: $\Sigma \triangleq \mathbb{L} \times \mathcal{M}$
- finite or infinite sequences of states: Σ^∞
- semantic domain $\mathscr{D}$: $\mathcal{S} \triangleq \wp(\Sigma^\infty)$
- semantic properties: $\wp(\mathcal{S}) = \wp(\wp((\mathbb{L} \times (\mathbb{X} \mapsto \mathcal{V}))^\infty))$
- concrete domain: $\langle \wp(\wp((\mathbb{L} \times (\mathbb{X} \mapsto \mathcal{V}))^\infty)),\ \subseteq\rangle$

Specifying abstractions (*i.e.* Galois connections)

$$\langle \wp(\mathscr{D}), \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathscr{A}, \sqsubseteq\rangle$$

Specifying the abstraction

- A collection of basic Galois connections
- Galois connectors: to built new Galois connections out of existing ones (e.g. $\Longmapsto$)

Specifying the abstraction (cont'd)

- Basic Galois connections $g \in \mathbb{G}$:

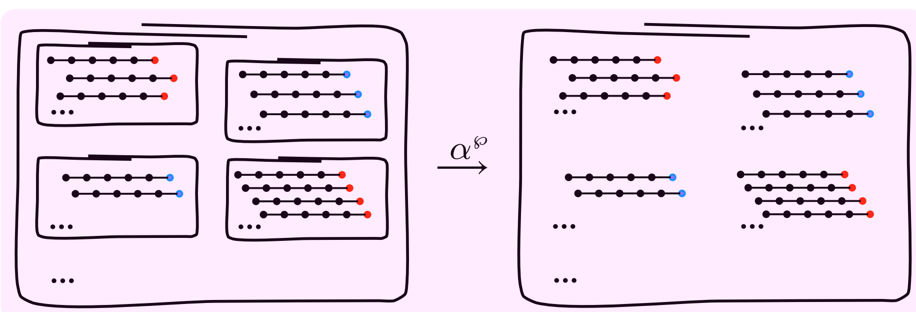
identity abstraction | top abstraction | interval abstraction | right image abstraction | join abstraction

$$g ::= \mathbb{1}[p] \mid \top[p, e] \mid \mathbb{I}[p, e, e] \mid \curvearrowright[s, s] \mid \cup[s] \mid$$
$$\neg[s] \mid \infty[s] \mid \rightsquigarrow[s, s] \mid \mapsto[s, s] \mid \times[s, s] \mid \ldots$$

complement | sequences to elements | relation to transformer | function abstraction | cartesian abstraction

Examples of basic GCs

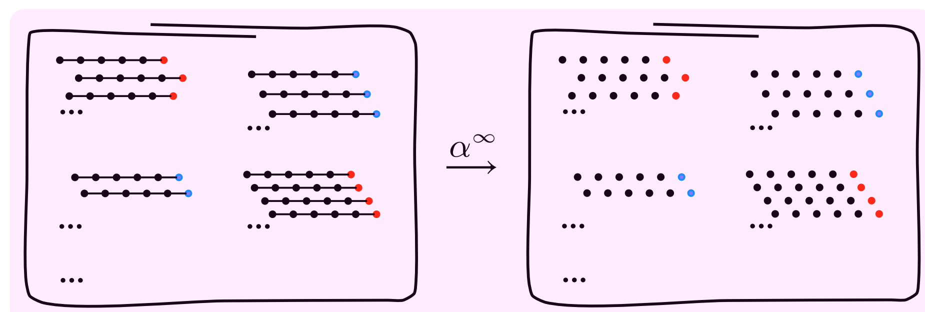
- Join abstraction $\cup[C]$:



$$\mathcal{S}[\![\cup[C]]\!] \triangleq \langle \wp(\wp(C)), \subseteq\rangle \underset{\alpha^{\wp}}{\overset{\gamma^{\wp}}{\leftrightarrows}} \langle \wp(C), \subseteq\rangle$$
$$\alpha^{\wp}(P) \triangleq \bigcup P, \quad \gamma^{\wp}(Q) \triangleq \wp(Q)$$

Examples of basic GCs (cont'd)

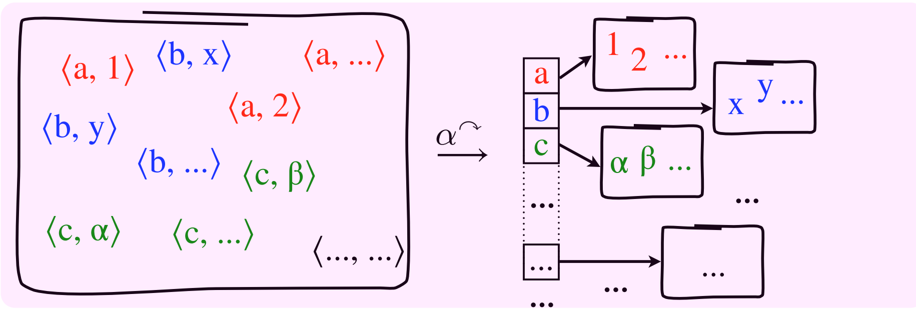
- Sequence abstraction $\infty[C]$:



$$\mathcal{S}[\![\infty[C]]\!] \triangleq \langle \wp(C^{\infty}), \subseteq\rangle \underset{\alpha^{\infty}}{\overset{\gamma^{\infty}}{\leftrightarrows}} \langle \wp(C), \subseteq\rangle$$
$$\alpha^{\infty}(P) \triangleq \{\sigma_i \mid \sigma \in P \wedge i \in \mathsf{dom}(\sigma)\}$$
$$\gamma^{\infty}(Q) \triangleq \{\sigma \in C^{\infty} \mid \forall i \in \mathsf{dom}(\sigma) : \sigma_i \in Q\}$$

Examples of basic GCs (cont'd)

- Right-image abstraction (isomorphism) $\curvearrowright[\mathbb{L}, \mathcal{M}]$:



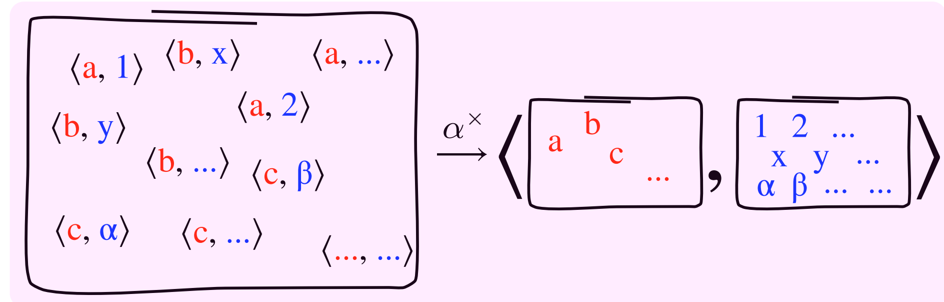
$$\mathcal{S}[\![\curvearrowright[\mathbb{L}, \mathcal{M}]]\!] \triangleq \langle \wp(\mathbb{L}\times\mathcal{M}), \subseteq\rangle \xleftrightarrow[\alpha^{\curvearrowright}]{\gamma^{\curvearrowright}} \langle \mathbb{L} \mapsto \wp(\mathcal{M}), \dot{\subseteq}\rangle$$

$$\alpha^{\curvearrowright}(P) \triangleq \lambda \ell \bullet \{m \mid \langle \ell, m\rangle \in P\}$$

$$\gamma^{\curvearrowright}(Q) \triangleq \{\langle \ell, m\rangle \mid m \in Q(\ell)\}$$

Examples of basic GCs (cont'd)

- Cartesian abstraction $\times[s_1, s_2]$:

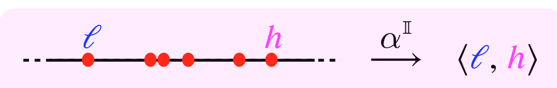


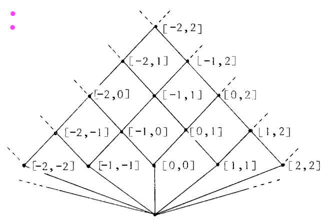
$$\mathcal{S}[\![\times[s_1, s_2]]\!] \triangleq$$
$$\langle \wp(\mathcal{S}[\![s_1]\!] \mapsto \mathcal{S}[\![s_2]\!]), \subseteq\rangle \xleftrightarrow[\alpha^{\times}]{\gamma^{\times}} \langle \mathcal{S}[\![s_1]\!] \mapsto \wp(\mathcal{S}[\![s_2]\!]), \dot{\subseteq}\rangle$$

$$\alpha^{\times}(X) \triangleq \lambda i \in \mathcal{S}[\![s_1]\!] \bullet \{x \in \mathcal{S}[\![s_2]\!] \mid \exists f \in \mathcal{S}[\![s_1]\!] \mapsto \mathcal{S}[\![s_2]\!] : f[i \leftarrow x] \in X\}$$

Examples of basic GCs (cont'd)

- Interval abstraction $\mathbb{I}[\langle s, o\rangle, e_1, e_2]$:





$$\mathcal{S}[\![\mathbb{I}[\langle s, o\rangle, e_1, e_2]]\!] \triangleq$$
$$\langle \wp(\mathcal{S}[\![s]\!]), \subseteq\rangle \xleftrightarrow[\alpha^{\mathbb{I}}]{\gamma^{\mathbb{I}}} \langle \Im(\mathcal{S}[\![s]\!] \cup \{\mathcal{S}[\![e_1]\!], \mathcal{S}[\![e_2]\!]\}, \mathcal{S}[\![o]\!]), \subseteq\rangle$$

$\Im(S, \leqslant) \triangleq \{[v_1, v_2] \mid v_1, v_2 \in S\}$ set of intervals

$[v_1, v_2] \triangleq \{v \in S \mid v_1 \leqslant v \wedge v \leqslant v_2\}$ interval

$$\alpha^{\mathbb{I}}(X) \triangleq [\min_{\mathcal{S}[\![o]\!]} X, \max_{\mathcal{S}[\![o]\!]} X]$$

Specifying the abstraction (cont'd)

- Galois connectors:

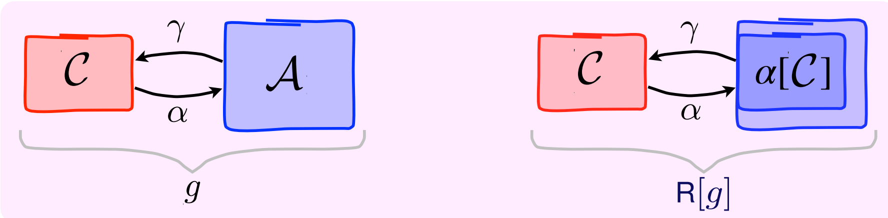
$$g \in \mathbb{G}$$

$$g ::= \ldots \mid \mathsf{R}[g] \mid s \twoheadrightarrow g \mid g \,⨾\, g \mid g \circledast g \mid g \Longmapsto g \mid \ldots$$

reduction | pointwise extension | composition connector | pairwise connector | functional connector

Examples of Galois connectors

- Reduction $\mathsf{R}[g]$ of Galois connection g :



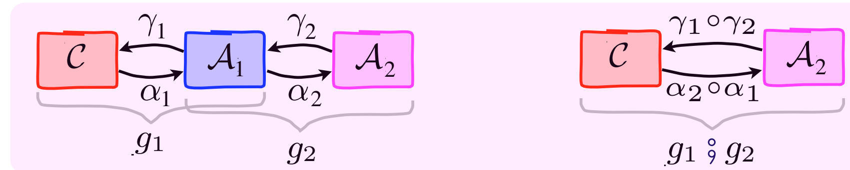
$\mathcal{S}[\![\mathsf{R}[g]]\!] \triangleq (\!|\, \mathcal{S}[\![g]\!] = \langle \mathcal{C},\ \sqsubseteq\rangle \xleftarrow[\alpha]{\gamma} \langle \mathcal{A},\ \leqslant\rangle \mathbin{?} \langle \mathcal{C},\ \sqsubseteq\rangle \xleftarrow[\alpha]{\gamma} \langle\{\alpha(P) \mid P \in \mathcal{C}\},\ \leqslant\rangle \mathbin{:} (\!|\, \mathcal{S}[\![g]\!] = \omega \mathbin{?} \omega \mathbin{:} \Omega \,|\!) \,|\!)$

dynamic error

static error

Examples of Galois connectors (cont'd)

- Composition connector $g_1 \mathbin{;} g_2$:

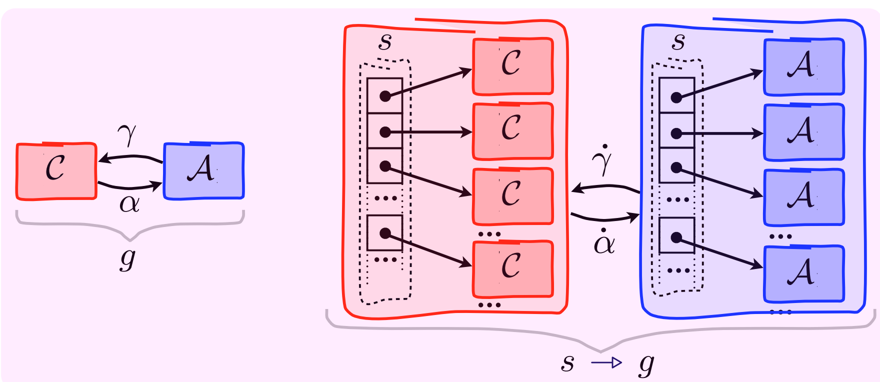


$\mathcal{S}[\![g_1 \mathbin{;} g_2]\!] \triangleq (\!|\, \mathcal{S}[\![g_1]\!] = p_1 \xleftarrow[\alpha_1]{\gamma_1} p_2 \wedge \mathcal{S}[\![g_2]\!] = p_3 \xleftarrow[\alpha_2]{\gamma_2} p_4 \mathbin{?} (\!|\, p_2 = p_3 \mathbin{?} p_1 \xleftarrow[\alpha_2 \circ \alpha_1]{\gamma_1 \circ \gamma_2} p_4 \mathbin{:} \omega \,|\!) \mathbin{:} \mathtt{error} \,|\!)$

where **error** is *static* (Ω) when $\mathcal{S}[\![g_1]\!]$ or $\mathcal{S}[\![g_2]\!]$ returns a static error, else *dynamic* (ω)

Examples of Galois connectors (cont'd)

- Componentwise/pointwise connector $s \dashrightarrow g$:



$\mathcal{S}[\![s \dashrightarrow g]\!] \triangleq (\!|\, \mathcal{S}[\![s]\!] = X \notin \{\omega, \Omega\} \wedge \mathcal{S}[\![g]\!] = \langle \mathcal{C},\ \sqsubseteq\rangle \xleftarrow[\alpha]{\gamma} \langle \mathcal{A},\ \leqslant\rangle \mathbin{?} \langle X \mapsto \mathcal{C},\ \dot\sqsubseteq\rangle \xleftarrow[\lambda \rho \bullet \alpha \circ \rho]{\lambda \overline{\rho} \bullet \gamma \circ \overline{\rho}} \langle X \mapsto \mathcal{A},\ \dot\leqslant\rangle \mathbin{:} \mathtt{error} \,|\!)$

Examples of abstractions

Reachability abstraction

- Reachability abstraction:

$$G^* \triangleq \cup[\Sigma^\infty] \fatsemi \infty[\Sigma] \fatsemi \curvearrowright[\mathbb{L}, \mathcal{M}]$$

properties to trace properties — traces to global invariant — global to to local invariant

- Applying abstract interpretation theory, you get by calculational design:
  - A proof method (Floyd/Hoare)
  - A fixpoint reachability-checking algorithm (Σ finite)

Patrick Cousot, Radhia Cousot: **Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints.** POPL 1977: 238-252

Patrick Cousot, Radhia Cousot: **Systematic Design of Program Analysis Frameworks.** POPL 1979: 269-282

Interval abstraction

- Interval abstraction :

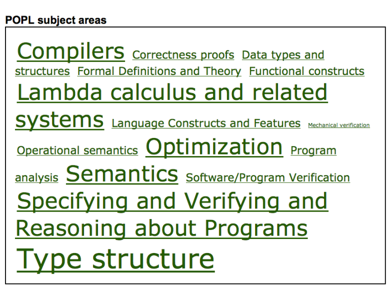
$$G^{\Im^*} \triangleq \mathsf{R}[G^* \fatsemi (\mathbb{L} \to (\times[\mathbb{X}, \mathcal{V}] \fatsemi (\mathbb{X} \to \mathbb{I}[\langle\mathcal{V}, \le\rangle, -\infty, \infty]))))]$$

reachability: properties to local invariants — for each program point — cartesian abstraction on variables — for each variable — interval abstraction

- Exactly the example of POPL'77, page 247

Patrick Cousot, Radhia Cousot: **Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints.** POPL 1977: 238-252

Typing the Galois connection calculus



Types as abstract interpretations, POPL'97

- The Galois connection calculus is a syntax which semantics has domain

$$\mathfrak{Gc} \triangleq \{\langle\mathcal{C}, \sqsubseteq\rangle \xleftrightarrow[\alpha]{\gamma} \langle\mathcal{A}, \preccurlyeq\rangle \mid \mathcal{C}, \mathcal{A} \text{ are sets} \wedge \sqsubseteq \in \wp(\mathcal{C} \times \mathcal{C}) \wedge \preccurlyeq \in \wp(\mathcal{A} \times \mathcal{A})\} \cup \{\Omega, \omega\}$$

- Design a type system to check statically that Galois connection expressions "cannot go wrong" (*i.e.* have the property $\mathfrak{Gc}\backslash\{\Omega\}$)

- Typing is an abstract interpretation

$$\langle\wp(\mathfrak{Gc}), \subseteq\rangle \xleftrightarrow[\alpha^{\mathfrak{T}}]{\gamma^{\mathfrak{T}}} \langle\mathfrak{T}_{/\cong}, \trianglelefteq\rangle$$

where $\mathsf{T} \trianglelefteq \mathsf{T}' \triangleq \gamma^{\mathfrak{T}}(\mathsf{T}) \subseteq \gamma^{\mathfrak{T}}(\mathsf{T}')$

Types

- Element types: $\mathsf{E} \in \mathfrak{E}$
  $\mathsf{E} ::= \texttt{var} \mid \texttt{lab} \mid \texttt{bool} \mid \texttt{int} \mid \texttt{err}$
- Set types: $\mathsf{S} \in \mathfrak{S}$
  $\mathsf{S} ::= \texttt{P}\,\mathsf{E} \mid \texttt{P}\,\mathsf{S} \mid \texttt{seq}\,\mathsf{S} \mid \mathsf{S} \ast\!\!\rightarrow \mathsf{S} \mid \mathsf{S} \ast \mathsf{S} \mid \texttt{err}$
- Partial order types:
  $\mathsf{O} \in \mathfrak{O}$
  $\mathsf{O} ::= \Rightarrow \mid \Leftrightarrow \mid \leq \mid \subseteq \mid = \mid \mathsf{O}^{-1} \mid \mathsf{O} \star \mathsf{O} \mid \dot{\mathsf{O}} \mid \ldots \mid \texttt{err}$
- Poset types: $\mathsf{P} \in \mathfrak{P}$
  $\mathsf{P} ::= \mathsf{S} \circledast \mathsf{O} \mid \texttt{err}$
- Galois connection types:
  $\mathsf{T} \in \mathfrak{T}$
  $\mathsf{T} ::= \mathsf{P} \leftrightarrows \mathsf{P} \mid \texttt{err}$

Semantics of types

- $\gamma^{\mathfrak{E}}(\texttt{bool}) \triangleq \mathbb{B}$

…

- $\gamma^{\mathfrak{S}}(\texttt{seq}\,\mathsf{S}) \triangleq \{X^{\infty} \mid X \in \gamma^{\mathfrak{S}}(\mathsf{S})\}$

…

- $\gamma^{\mathfrak{O}}(\dot{\mathsf{O}}) \triangleq \{\dot{\leqslant} \mid \leqslant \in \gamma^{\mathfrak{O}}(\mathsf{O})\}$

…

- $\gamma^{\mathfrak{P}}(\mathsf{S} \circledast \mathsf{O}) \triangleq \gamma^{\mathfrak{S}}(\mathsf{S}) \times \gamma^{\mathfrak{O}}(\mathsf{O})$

…

- $\gamma^{\mathfrak{T}}(\mathsf{P} \leftrightarrows \mathsf{P}') \triangleq \{P \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} P' \mid P \in \gamma^{\mathfrak{P}}(\mathsf{P}) \wedge P' \in \gamma^{\mathfrak{P}}(\mathsf{P}')\}$

the semantics of a type is the set of elements with that type (never ω / Ω)

Type equivalence

- Definition: $\mathsf{T}_1 \trianglelefteq \mathsf{T}_2 \triangleq \gamma^{\mathfrak{T}}(\mathsf{T}_1) \subseteq \gamma^{\mathfrak{T}}(\mathsf{T}_2)$
  $\mathsf{T}_1 \cong \mathsf{T}_2 \triangleq \mathsf{T}_1 \trianglelefteq \mathsf{T}_2 \wedge \mathsf{T}_2 \trianglelefteq \mathsf{T}_1$
- Rules:
  - $\mathsf{E} \trianglelefteq \mathsf{E}' \Rightarrow \texttt{P}\,\mathsf{E} \trianglelefteq \texttt{P}\,\mathsf{E}'$
  - $\mathsf{S} \trianglelefteq \mathsf{S}' \Rightarrow \texttt{P}\,\mathsf{S} \trianglelefteq \texttt{P}\,\mathsf{S}'$
  - $\mathsf{S} \trianglelefteq \mathsf{S}' \Rightarrow \texttt{seq}\,\mathsf{S} \trianglelefteq \texttt{seq}\,\mathsf{S}'$
  - $\mathsf{S}_1 \trianglelefteq \mathsf{S}_1' \wedge \mathsf{S}_2 \trianglelefteq \mathsf{S}_2' \Rightarrow \mathsf{S}_1 \ast\!\!\rightarrow \mathsf{S}_2 \trianglelefteq \mathsf{S}_1' \ast\!\!\rightarrow \mathsf{S}_2'$
  - $\mathsf{S}_1 \trianglelefteq \mathsf{S}_1' \wedge \mathsf{S}_2 \trianglelefteq \mathsf{S}_2' \Rightarrow \mathsf{S}_1 \ast \mathsf{S}_2 \trianglelefteq \mathsf{S}_1' \ast \mathsf{S}_2'$
  - $\Leftrightarrow \trianglelefteq \Rightarrow, = \trianglelefteq \leq, = \trianglelefteq \subseteq, = \trianglelefteq \sqsubseteq$
  - $\mathsf{O} \trianglelefteq \mathsf{O}' \Rightarrow \mathsf{O}^{-1} \trianglelefteq \mathsf{O}'^{-1}$
  - $\mathsf{O}_1 \trianglelefteq \mathsf{O}_1' \wedge \mathsf{O}_2 \trianglelefteq \mathsf{O}_2' \Rightarrow \mathsf{O}_1 \star \mathsf{O}_2 \trianglelefteq \mathsf{O}_1' \star \mathsf{O}_2'$
  - $\mathsf{O} \trianglelefteq \mathsf{O}' \Rightarrow \dot{\mathsf{O}} \trianglelefteq \dot{\mathsf{O}}'$
  - $\mathsf{S} \trianglelefteq \mathsf{S}' \wedge \mathsf{O} \trianglelefteq \mathsf{O}' \Rightarrow \mathsf{S} \circledast \mathsf{O} \trianglelefteq \mathsf{S}' \circledast \mathsf{O}'$
  - $\mathsf{P}_1 \trianglelefteq \mathsf{P}_1' \wedge \mathsf{P}_2 \trianglelefteq \mathsf{P}_2' \Rightarrow \mathsf{P}_1 \leftrightarrows \mathsf{P}_2 \trianglelefteq \mathsf{P}_1' \leftrightarrows \mathsf{P}_2'$
  - $\mathsf{S} \trianglelefteq \mathsf{S}' \wedge \mathsf{T} \trianglelefteq \mathsf{T}' \Rightarrow \mathsf{S} \ast\!\!\rightarrow \mathsf{T} \trianglelefteq \mathsf{S}' \ast\!\!\rightarrow \mathsf{T}'$

  …

Soundness of types

- The *calculational design* of the type inference algorithm $\mathscr{T}[\![g]\!]$ is by approximation of the collecting semantics
- As usual in abstract interpretation (*), we know the type system will be sound *before* designing the inference rules
- Typable Galois connection expressions ($\neq$ **err**) cannot go wrong (be Ω)

$$\left(\mathscr{T}[\![g]\!] \neq \texttt{err} \;?\; \mathcal{S}[\![g]\!] \in \gamma^{\mathfrak{T}}(\mathscr{T}[\![g]\!]) \cup \{\omega\}\right)$$

- Typing rules are an equivalent rule-based presentation

(*) Patrick Cousot: Types as Abstract Interpretations. POPL 1997: 316-331

Type inference algorithm

- …
- $\mathcal{S}[\![s_1 \cup s_2]\!] \triangleq \big(\mathtt{err} \neq \mathcal{S}[\![s_1]\!] \cong \mathcal{S}[\![s_2]\!] \neq \mathtt{err} \mathrel{?} \mathcal{S}[\![s_1]\!] \mathrel{:} \mathtt{err}\big)$

  same type (like alternatives in conditionals), correct expressions may be rejected

- …
- $\mathcal{T}[\![g_1 \mathbin{;} g_2]\!] \triangleq \big(\mathcal{T}[\![g_1]\!] = \mathsf{P}_1 \leftrightarrows \mathsf{P}_2 \wedge \mathcal{T}[\![g_2]\!] = \mathsf{P}_3 \leftrightarrows \mathsf{P}_4 \wedge \mathsf{P}_2 \cong \mathsf{P}_3 \mathrel{?} \mathsf{P}_1 \leftrightarrows \mathsf{P}_4 \mathrel{:} \mathtt{err}\big)$

  same type (does not exclude dynamic errors, same type $\not\Rightarrow$ same set)

- …

Typing rules

- true $\vdash$ $\mathtt{bool}$ ($x \vdash \mathsf{T}$ is $\mathcal{T}[\![x]\!] = \mathsf{T}$)

  …

- $\mathbb{B} \vdash \mathbf{P}\ \mathtt{bool}$

  …

- $\dfrac{s \vdash \mathsf{S}}{\cup[s] \vdash \mathbf{P}\ (\mathbf{P}\ \mathsf{S}) \circledast \subseteq \leftrightarrows \mathbf{P}\ \mathsf{S} \circledast \subseteq}$
- $\dfrac{s \vdash \mathsf{S}}{\infty[s] \vdash \mathbf{P}\ (\mathbf{seq}\ \mathsf{S}) \circledast \subseteq \leftrightarrows \mathbf{P}\ \mathsf{S} \circledast \subseteq}$
- $\dfrac{g_1 \vdash \mathsf{P}_1 \leftrightarrows \mathsf{P}_2, \quad g_2 \vdash \mathsf{P}_3 \leftrightarrows \mathsf{P}_4, \quad \mathsf{P}_2 \cong \mathsf{P}_3}{g_1 \mathbin{;} g_2 \vdash \mathsf{P}_1 \leftrightarrows \mathsf{P}_4}$
- $\dfrac{s_{\mathbb{L}} \vdash \mathsf{S}_{\mathbb{L}}, \quad s_{\mathcal{M}} \vdash \mathsf{S}_{\mathcal{M}}}{\curvearrowright[s_{\mathbb{L}}, s_{\mathcal{M}}] \vdash \mathbf{P}\ (\mathsf{S}_{\mathbb{L}} * \mathsf{S}_{\mathcal{M}}) \circledast \subseteq \leftrightarrows \mathsf{S}_{\mathbb{L}} \ast\!\!\rightarrow \mathbf{P}\ \mathsf{S}_{\mathcal{M}} \circledast \dot{\subseteq}}$

  …

Type of interval analysis

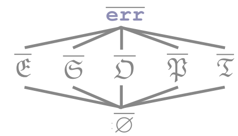
- $\mathcal{T}[\![\cup[(\mathbb{L} \times (\mathbb{X} \mapsto \mathbb{Z}))^{\infty}] \mathbin{;} \infty[\mathbb{L} \times (\mathbb{X} \mapsto \mathbb{Z})] \mathbin{;} \curvearrowright[\mathbb{L}, \mathbb{X} \mapsto \mathbb{Z}] \mathbin{;} \mathbb{L} \twoheadrightarrow (\times[\mathbb{X}, \mathbb{Z}] \mathbin{;} (\mathbb{X} \twoheadrightarrow \mathbb{I}[\langle \mathbb{Z}, \leq \rangle, -\infty, \infty]))]\!]$

  $= \mathbf{P}\ (\mathbf{P}\ (\mathbf{seq}\ (\mathbf{P}\ \mathtt{lab} * (\mathbf{P}\ \mathtt{var} \ast\!\!\rightarrow \mathbf{P}\ \mathtt{int})))) \circledast \subseteq \leftrightarrows (\mathbf{P}\ \mathtt{lab} \ast\!\!\rightarrow \mathbf{P}\ \mathtt{var} \ast\!\!\rightarrow \mathbf{P}\ \mathbf{P}\ \mathtt{int} \circledast \ddot{\subseteq})$

  (intervals / interval inclusion are abstracted by sets / set inclusion in the type system)

Typing the type system of the Galois connection calculus

Types of types



- Sorts of types: $\mathcal{T} \triangleq \{\mathfrak{E}, \mathfrak{S}, \mathfrak{O}, \mathfrak{P}, \mathfrak{T}\}$
- Domain of all types: $\mathfrak{T} = \bigcup \mathcal{T} \setminus \{\texttt{err}\}$
- Properties of types: $\mathfrak{P} = \wp(\mathfrak{T})$
- Types of types: $\overline{\mathfrak{T}} ::= \overline{\varnothing} \mid \overline{\mathfrak{E}} \mid \overline{\mathfrak{S}} \mid \overline{\mathfrak{O}} \mid \overline{\mathfrak{P}} \mid \overline{\mathfrak{T}} \mid \overline{\texttt{err}}$
- Abstraction of properties of types to types of types

$$\alpha^{\overline{\mathfrak{T}}} \in \mathfrak{P} \longrightarrow \overline{\mathfrak{T}}$$

$$\alpha^{\overline{\mathfrak{T}}}(P) \triangleq (\!| P = \emptyset \;?\; \overline{\varnothing} \;\|\; P \subseteq \mathsf{T}, \mathsf{T} \in \mathcal{T} \;?\; \overline{\mathsf{T}} \;:\; \overline{\texttt{err}} |\!)$$

- Typable types cannot go wrong $\overline{\texttt{err}}$ (e.g. an element cannot be typed as a set)

Conclusion

Abstract interpretation

- Any human or automated reasoning (on programs) involves abstractions
- Abstract interpretation aims at formalizing abstractions in the abstract
- Hopefully useful to grasp the literature (vast, eclectic, and exploding collection of recipes mostly lacking unifying principles)
- Provides a methodology to design sound abstract semantics/transformers/proof methods/verifiers/analyzers/*etc*

Perspectives

- A Galois connection calculus for specifying abstractions
  - can be implemented in programming languages or better in mathematical higher-level languages (to include formal soundness proofs)
  - can be extended to specify abstract domains (with transformers, widenings, *etc*.)
- The calculus should be useful for
  - the certification of abstract semantics/transformers/proof methods/verifiers/static analysers
  - advance towards unrestricted automatic static analyser generation

Perceval le Gallois' Wondrous Grail Quest [(*)]

- To design a programming language:
  - specify its syntax and semantics
  - specify abstractions to automatically get:
    - abstract semantics and proof methods
    - interpreters and compilers (for known machines with well-specified semantics)
    - types systems
    - verifiers
    - static analyzers

(*) *Perceval, le Conte du Graal,* novel by Chrétien de Troyes, 12th century & *Perceval le Gallois*, movie by Éric Rohmer (1978)

The End, Thank You

References

- Patrick Cousot, Radhia Cousot: A Galois connection calculus for abstract interpretation. POPL 2014: 3-4
- Patrick Cousot, Radhia Cousot: An abstract interpretation framework for termination. POPL 2012: 245-258
- Patrick Cousot, Radhia Cousot, Francesco Logozzo: A parametric segmentation functor for fully automatic and scalable array content analysis. POPL 2011: 105-118
- Patrick Cousot, Radhia Cousot: An abstract interpretation-based framework for software watermarking. POPL 2004: 173-185
- Bruno Blanchet, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, Xavier Rival: A static analyzer for large safety-critical software. PLDI 2003: 196-207
- Patrick Cousot, Radhia Cousot: Systematic design of program transformation frameworks by abstract interpretation. POPL 2002: 178-190
- Patrick Cousot, Radhia Cousot: Temporal Abstract Interpretation. POPL 2000: 12-25
- Patrick Cousot: Types as Abstract Interpretations. POPL 1997: 316-331
- Patrick Cousot, Radhia Cousot: Inductive Definitions, Semantics and Abstract Interpretation. POPL 1992: 83-94
- Patrick Cousot, Radhia Cousot: Systematic Design of Program Analysis Frameworks. POPL 1979: 269-282
- Patrick Cousot, Nicolas Halbwachs: Automatic Discovery of Linear Restraints Among Variables of a Program. POPL 1978: 84-96
- Patrick Cousot, Radhia Cousot: Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. POPL 1977: 238-252