Abstract Interpretation: Past, Present, and Future

Patrick Cousot

cims.nyu.edu/~pcousot

Radhia Cousot

di.ens.fr/~rcousot

CSL — LICS
Vienna, Austria — July 14 – 18, 2014

Motivation

Formal methods

- Reasonings on programs are
  - Reasonings on properties of their semantics (*i.e.* execution behaviors)
  - Always involve some form of abstraction

Abstract interpretation

- A theory establishing a correspondance between
  - Concrete semantic properties
    ↑ what you want to prove on the semantics
  - Abstract properties
    ↑ how to prove it in the abstract
- Objective: formalize
  - formal methods
  - algorithms for reasoning on programs

Fundamental motivations

Scientific research

- in Mathematics/Physics:

  trend towards unification and synthesis through universal principles

- in Computer science:

  trend towards dispersion and parcelization through a collection of local techniques for specific applications

  An exponential process, will stop!

Example: reasoning on computational structures

WCET
Axiomatic semantics
Confidentiality analysis
Program synthesis
Grammar analysis
Statistical model-checking
Invariance proof
Probabilistic verification
Parsing

Security protocole verification
Dataflow analysis
Partial evaluation
Effect systems
Trace semantics
Symbolic execution
Quantum entanglement detection
Type theory

Systems biology analysis
Model checking
Obfuscation
Denotational semantics
Theories combination
Code contracts
Integrity analysis
Bisimulation
SMT solvers
Steganography

Operational semantics
Database query
Dependence analysis
CEGAR
Program transformation
Interpolants
Abstract model checking
Tautology testers

Abstraction refinement
Type inference
Separation logic
Termination proof
Shape analysis
Malware detection
Code refactoring

Example: reasoning on computational structures

WCET
Axiomatic semantics
Confidentiality analysis
Program synthesis
Grammar analysis
Statistical model-checking
Invariance proof
Probabilistic verification
Parsing

Security protocole verification
Dataflow analysis
Partial evaluation
Effect systems
Trace semantics
Symbolic execution
Quantum entanglement detection
Type theory

Systems biology analysis
Model checking
Obfuscation
Denotational semantics
Theories combination
Code contracts
Integrity analysis
Bisimulation
SMT solvers
Steganography

Operational semantics
Database query
Dependence analysis
CEGAR
Program transformation
Interpolants
Abstract model checking
Tautology testers

Abstraction refinement
Type inference
Separation logic
Termination proof
Shape analysis
Malware detection
Code refactoring

Example: reasoning on computational structures

Abstract interpretation

WCET
Axiomatic semantics
Confidentiality analysis
Program synthesis
Grammar analysis
Statistical model-checking
Invariance proof
Probabilistic verification
Parsing
Security protocole verification
Dataflow analysis
Partial evaluation
Effect systems
Trace semantics
Symbolic execution
Type theory
Model checking
Obfuscation
Denotational semantics
Theories combination
Code contracts
Quantum entanglement detection
Steganography
Systems biology analysis
Database query
Dependence analysis
CEGAR
Program transformation
Interpolants
Integrity analysis
Bisimulation
SMT solvers
Tautology testers
Operational semantics
Abstraction refinement
Type inference
Separation logic
Termination proof
Abstract model checking
Shape analysis
Malware detection
Code refactoring


Practical motivations

All computer scientists have experienced bugs

Ariane 5.01 failure (overflow)

Patriot failure (float rounding)

Mars orbiter loss (unit error)
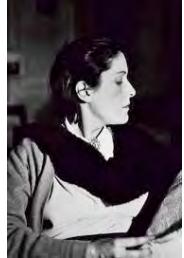
Heartbleed (buffer overrun)

- Checking the presence of bugs by debugging is great
- Proving their absence by static analysis is even better!
- Undecidability and complexity is the challenge for automation

---

Informal examples of abstraction

Abstractions of Dora Maar by Picasso
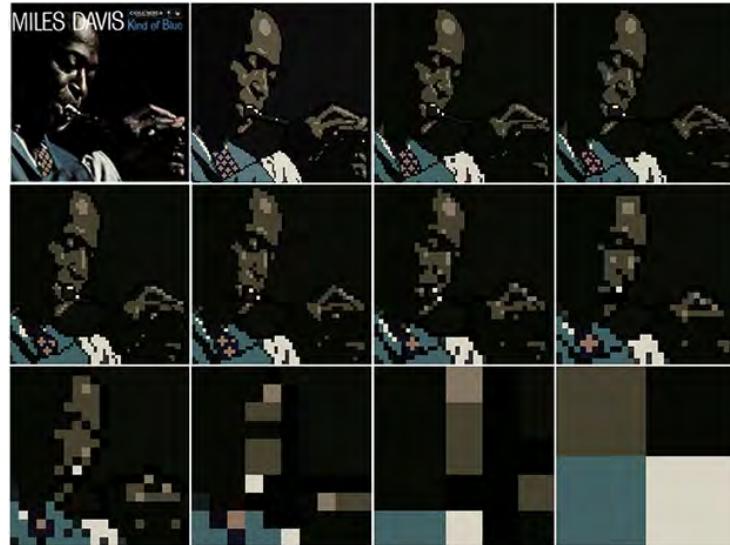
Pixelation

/www.petapixel.com/2011/06/23/how-much-pixelation-is-needed-before-a-photo-becomes-transformed/

*Image credit*: Photograph by Jay Maisel

## An old idea...

20 000 years old picture in a spanish cave:
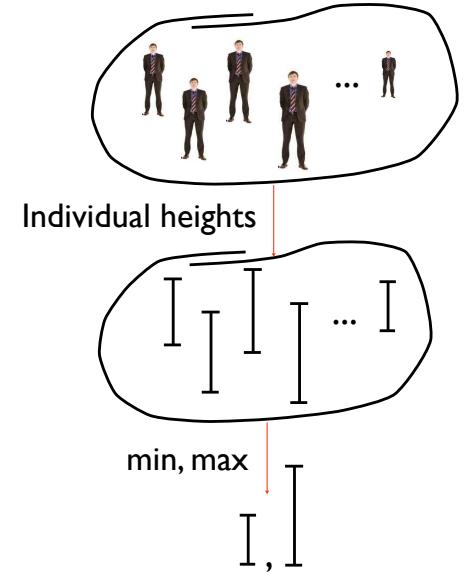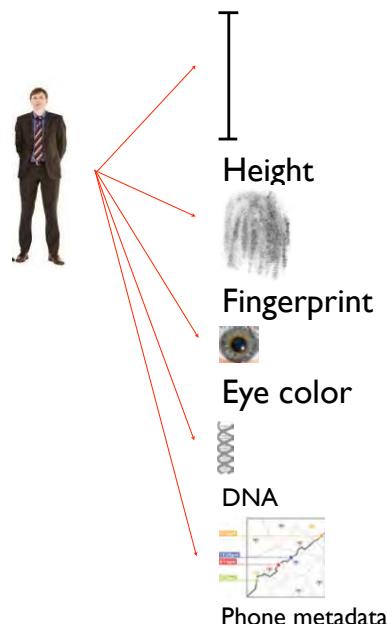
(the concrete is unknown)

Abstractions of a man / crowd

Height

Fingerprint

Eye color

DNA

Phone metadata

Individual heights

min, max
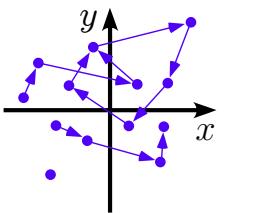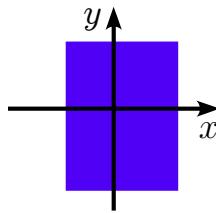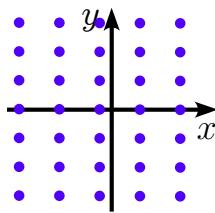
Numerical abstractions in Astrée
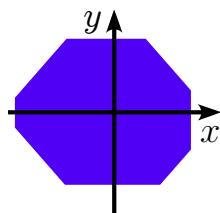
Collecting semantics:
partial traces
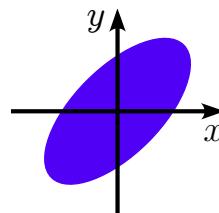
Intervals:
$x \in [a, b]$
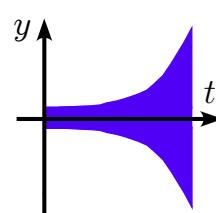
Simple congruences:
$x \equiv a[b]$

Octagons:
$\pm x \pm y \leqslant a$

Ellipses:
$x^2 + by^2 - axy \leqslant d$

Exponentials:
$-a^{bt} \leqslant y(t) \leqslant a^{bt}$

17

A very short introduction to abstract interpretation

Patrick Cousot & Radhia Cousot. Vérification statique de la cohérence dynamique des programmes. In *Rapport du contrat IRIA SESORI No 75-035*, Laboratoire IMAG, University of Grenoble, France. 125 pages. 23 September 1975.

Patrick Cousot & Radhia Cousot. Static Determination of Dynamic Properties of Programs. In B. Robinet, editor, *Proceedings of the second international symposium on Programming*, Paris, France, pages 106—130, April 13-15 1976, Dunod, Paris.

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

Patrick Cousot. Méthodes itératives de construction et d'approximation de points fixes d'opérateurs monotones sur un treillis, analyse sémantique des programmes. *Thèse Ès Sciences Mathématiques*, Université Joseph Fourier, Grenoble, France, 21 March 1978

Patrick Cousot. Semantic foundations of program analysis. In S.S. Muchnick & N.D. Jones, editors, *Program Flow Analysis: Theory and Applications*, Ch. 10, pages 303—342, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, U.S.A., 1981.

18

Properties and their Abstractions

19

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 $x \in P$

- $\langle \wp(\mathscr{D}), \subseteq, \cup, \cap, ... \rangle$ is a *complete lattice* for inclusion $\subseteq$ (*i.e. logical implication*)

20

Abstract properties

- Abstract properties: $Q \in \mathscr{A}$

- Abstract domain $\mathscr{A}$: encodes a subset of the concrete properties (*e.g.* a program logic, type terms, linear algebra, *etc*)

- Poset: $\langle \mathscr{A}, \sqsubseteq, \sqcup, \sqcap, ... \rangle$

- Partial order: $\sqsubseteq$ is *abstract implication*

Concretization

- Concretization $\gamma \in \mathscr{A} \longrightarrow \wp(\mathscr{D})$

- $\gamma(Q)$ is the semantics (concrete meaning) of Q

- γ is *increasing* (so $\sqsubseteq$ abstracts $\subseteq$)

- The concrete properties in $\gamma(\mathscr{A})$ are exactly representable in the abstract $\mathscr{A}$, all others in $\wp(\mathscr{D}) \backslash \gamma(\mathscr{A})$ can only be approximated in $\mathscr{A}$

Best abstraction

- A concrete property $P \in \wp(\mathscr{D})$ has a best abstraction $Q \in \mathscr{A}$ iff

  - it is sound (over-approximation):

    $$P \subseteq \gamma(Q)$$

  - and more precise than any sound abstraction:

    $$P \subseteq \gamma(Q') \implies Q \sqsubseteq Q' \implies \gamma(Q) \subseteq \gamma(Q')$$

- 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

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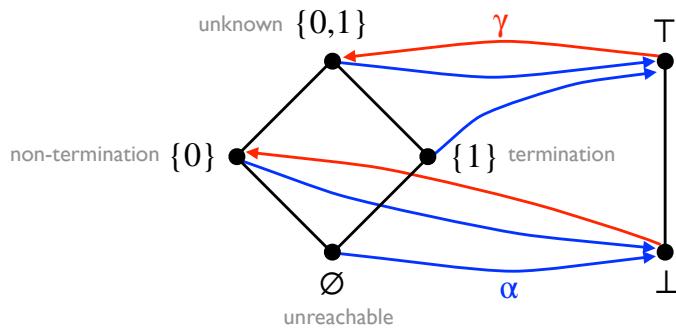
Examples

- Needness/strictness analysis (80's)

unknown {0,1}   γ   ⊤

non-termination {0}   {1} termination

∅   α   ⊥

unreachable

- Similar abstraction ($\gamma(\top) \triangleq \{\text{true}, \text{false}\}$) for scalable harware symbolic trajectory evaluation STE (90)

Alan Mycroft: The Theory and Practice of Transforming Call-by-need into Call-by-value. Symposium on Programming 1980: 269-281

Carl-Johan H. Seger, Randal E. Bryant: Formal Verification by Symbolic Evaluation of Partially-Ordered Trajectories. Formal Methods in System Design 6(2): 147-189 (1995)

Example: Homomorphic abstraction $\wp(\mathscr{D}) \longrightarrow \wp(\mathscr{A})$

- $\hbar \in \mathscr{D} \longrightarrow \mathscr{A}$

  $\alpha \triangleq \lambda X \cdot \{\hbar(x) \mid x \in X\}$

  $\gamma \triangleq \lambda Y \cdot \{x \in \mathscr{D} \mid \hbar(x) \in Y\}$

  $\Longrightarrow \langle \wp(\mathscr{D}), \subseteq \rangle \xleftrightarrow[\alpha]{\gamma} \langle \wp(\mathscr{A}), \subseteq \rangle \qquad (\longrightarrow$ iff $\hbar$ onto$)$

- Example (*): rule of signs: $A = \mathbb{Z}$, $B = \{-1, 0, 1\}$, $\hbar(z) = z/|z|$

- Counter-example (**): intervals (octagons, polyhedra, etc)

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

(**) 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


Properties of Galois connections

- α preserves existing lubs (by order-duality, γ preserves existing glbs)

- One adjoint uniquely determine the other

- α is surjective (iff γ injective iff $\alpha \circ \gamma = 1$), written

  $$\langle P, \preccurlyeq \rangle \xleftrightarrow[\alpha]{\gamma} \langle Q, \sqsubseteq \rangle$$

- The composition of Galois connections is a Galois connection

- $\alpha(x)$ is the best over-approximation of $x \in P$:

  - $x \preccurlyeq \gamma(\alpha(x))$ over-approximation

  - $x \preccurlyeq \gamma(y) \implies \alpha(x) \sqsubseteq y$ more precise than any other over-approximation

---
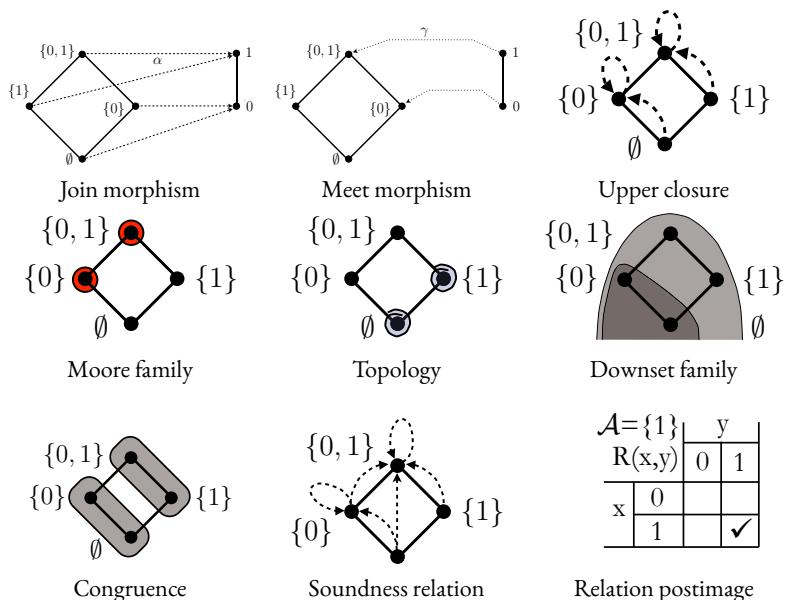
Equivalent mathematical structures

Join morphism   Meet morphism   Upper closure

Moore family   Topology   Downset family

Congruence   Soundness relation   Relation postimage

In absence of best abstraction?

- Best abstraction of a disk by a rectangular parallelogram (intervals)

- No best abstraction of a disk by a polyhedron (Euclid)

use only abstraction or concretization or widening [*]

(*) Patrick Cousot, Radhia Cousot: Abstract Interpretation Frameworks. J. Log. Comput. 2(4): 511-547 (1992)

Sound semantics abstraction

- program $\quad P \in \mathbb{L} \quad$ programming language

- standard semantics $\quad S[\![P]\!] \in \mathscr{D} \quad$ semantic domain

- collecting semantics $\{S[\![P]\!]\} \in \wp(\mathscr{D})$ semantic property

- abstract semantics $\quad \overline{S}[\![P]\!] \in \mathscr{A} \quad$ abstract domain

- concretization $\quad \gamma \in \mathscr{A} \longrightarrow \wp(\mathscr{D})$

- soundness $\quad \{S[\![P]\!]\} \subseteq \gamma(\overline{S}[\![P]\!])$

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

Best abstract semantics

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

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

- Proof:

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

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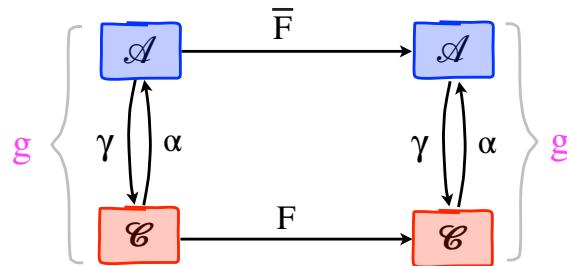

Calculational design of the abstract semantics

- The (standard hence collecting) semantics are defined by composition of mathematical structures (such as set unions, products, functions, fixpoints, *etc*)

- If you know best abstractions 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 [POPL 2012])
  - *presentations* (fixpoints, equations, constraints, rules [CAV 1995])

Example: functional connector

- If $g = \langle \mathscr{C}, \subseteq \rangle \xrightleftharpoons[\alpha]{\gamma} \langle \mathscr{A}, \sqsubseteq \rangle$ then

$$g \longmapsto \overline{g} = \langle \mathscr{C} \xrightarrow{\,\prime\,} \mathscr{C}, \subseteq \rangle \xrightleftharpoons[\lambda \overline{F}.\gamma \circ \overline{F} \circ \alpha]{\lambda F.\alpha \circ F \circ \gamma} \langle \mathscr{A} \xrightarrow{\,\prime\,} \mathscr{A}, \sqsubseteq \rangle$$

($\Longmapsto$ 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 $\alpha(\mathsf{lfp}\, F) = \mathsf{lfp}\, \overline{F}$

  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 $\alpha(\mathsf{lfp}\, F) \sqsubseteq \mathsf{lfp}\, \overline{F}$

  e.g. static analysis of infinite state systems

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

Fixpoint abstraction

Theorem 1 If $\langle C, \sqsubseteq \rangle \xrightleftharpoons[\alpha]{\gamma} \langle A, \preceq \rangle$ in cpos for infinite/transfinite chains, $F \in C \mapsto C$ and $G \in A \mapsto A$ are continuous/increasing then
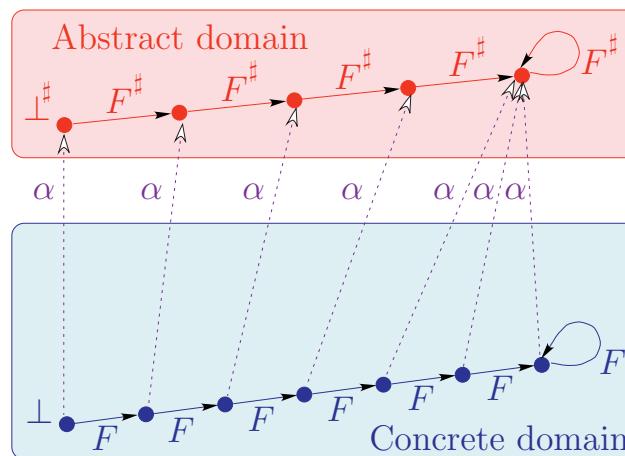
$$\alpha(\mathbf{lfp}^{\sqsubseteq} F) = \mathbf{lfp}^{\preceq} G \quad \Longleftarrow \quad \alpha \circ F = G \circ \alpha \quad \text{(commutation condition)}$$
$$G = \alpha \circ F \circ \gamma$$

$$\alpha(\mathbf{lfp}^{\sqsubseteq} F) \preceq \mathbf{lfp}^{\preceq} G \quad \Longleftarrow \quad \alpha \circ F \preceq G \circ \alpha \quad \text{(semi-commutation condition)}$$

[Cousot and Cousot, 1979b, theorem 7.1.0.4(2–3)], see also [de Bakker et al., 1984, lemma 4.3], [Apt and Plotkin, 1986, fact 2.3], [Backhouse, 2000, theorem 95], etc.
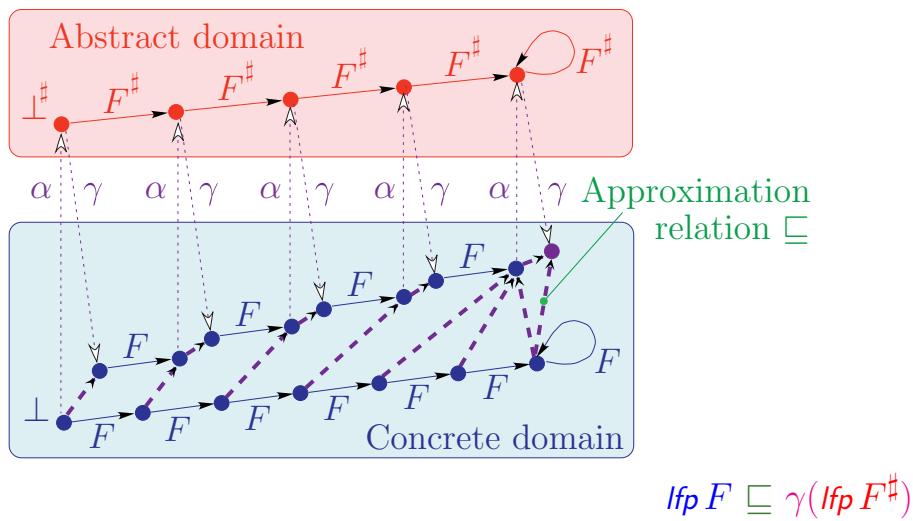
[Cousot and Cousot, 1979b] Patrick Cousot, Radhia Cousot: Systematic Design of Program Analysis Frameworks. POPL 1979: 269-282


Exact fixpoint abstraction

$$\alpha \circ F = F^{\sharp} \circ \alpha \implies \alpha(\mathit{lfp}\, F) = \mathit{lfp}\, F^{\sharp}$$

Approximate fixpoint abstraction

Abstract domain

Approximation relation ⊑

Concrete domain

$lfp\ F \sqsubseteq \gamma(lfp\ F^\sharp)$

Duality

- **Order duality**: join (∪) or meet (∩)

- **Inversion duality**: forward (→) or backward (← = (→)⁻¹)

- **Fixpoint duality**: least (↓) or greatest (↑)

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
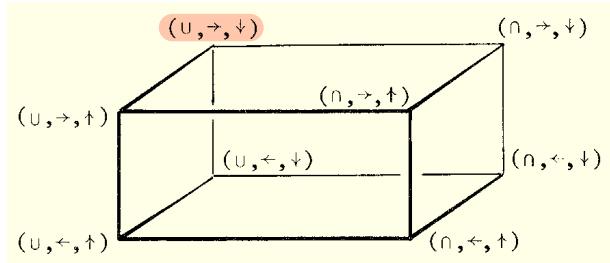
Why abstracting properties of semantics, not semantics?

Abstraction of the semantic properties

1. *Abstract interpretation* = a non-standard semantics (computations on values in the standard semantics are replaced by computations on abstract values) ⟹ extremely limited

2. *Abstract interpretation* = an abstraction of the standard semantics ⟹ limited

3. *Abstract interpretation* = an abstraction of properties of the standard semantics ⟹ more general

   i.e. (1) is an abstraction of (2), (2) is an abstraction of (3)

Example: trace semantics properties

- Domain of [in]finite traces on states: Π

- "Standard" trace semantics domain: $\mathcal{D} = \wp(\Pi)$

- "Standard" trace semantics $S[\![P]\!] \in \mathcal{D} = \wp(\Pi)$

- Domain of semantics properties is $\wp(\mathcal{D}) = \wp(\wp(\Pi))$

- Collecting semantics $C[\![P]\!] \triangleq \{S[\![P]\!]\} \in \wp(\mathcal{D}) = \wp(\wp(\Pi))$

How to abstract the standard semantics?

- The join abstraction:

$$\langle \wp(\wp(\Pi)), \subseteq \rangle \xleftarrow[\alpha_\cup]{\gamma_\cup} \langle \wp(\Pi), \subseteq \rangle$$

$$\alpha_\cup(X) \triangleq \bigcup X$$

$$\gamma_\cup(Y) \triangleq \wp(Y)$$

- Join abstraction of the collecting semantics:

$$\alpha_\cup(C[\![P]\!]) \triangleq \bigcup \{S[\![P]\!]\} \triangleq S[\![P]\!]$$

(*i.e.* the semantics is the join abstraction of its strongest property)
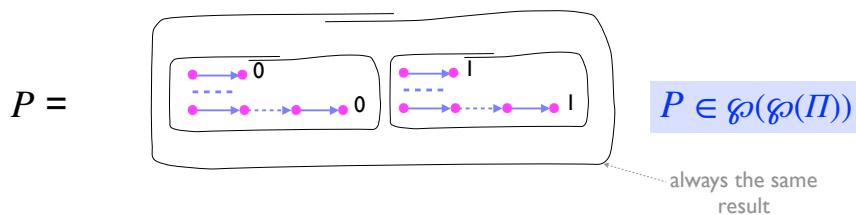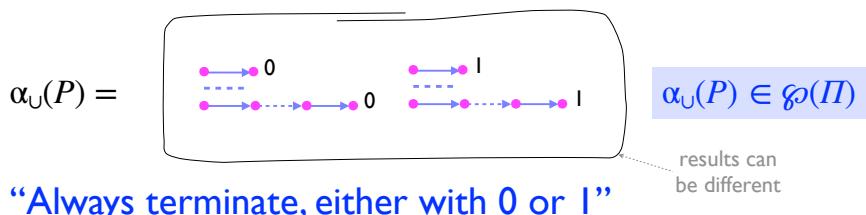
Loss of information

- "Always terminate with the same value, either 0 or 1"

$$P = \quad \boxed{P \in \wp(\wp(\Pi))}$$

always the same result

- Join abstraction:

$$\alpha_\cup(P) = \quad \boxed{\alpha_\cup(P) \in \wp(\Pi)}$$

results can be different

"Always terminate, either with 0 or 1"

Limitations of the union abstraction

- Complete iff any property of the semantics $S[\![P]\!]$ is also valid for any subset $\gamma(S[\![P]\!]) = \wp(S[\![P]\!])$:

  - Examples: safety, liveness

  - Counter-example: security (*e.g.* authentification using a random cryptographic nonce)

Exact abstractions

Exact abstractions

- The concrete properties of the standard semantics $S[\![\mathrm{P}]\!]$ that you want to prove can always be proved in the abstract (which is simpler):

$$\forall\, Q \in \mathscr{A}: S[\![\mathrm{P}]\!] \in \gamma(Q) \iff \overline{S}[\![\mathrm{P}]\!] \sqsubseteq Q$$

where

$$\overline{S}[\![\mathrm{P}]\!] \triangleq \alpha \circ S[\![\mathrm{P}]\!] \circ \gamma$$

Example 1 of exact abstraction: grammars

Patrick Cousot, Radhia Cousot: Grammar semantics, analysis and parsing by abstract interpretation. Theor. Comput. Sci. 412(44): 6135-6192 (2011)

Example: Grammars

- Context-free grammar on alphabet $A = Num \cup Var \cup \{+,-,(,),\ldots\}$:

$$E \;::=\; Num \mid Var \mid E+E \mid -E \mid (E)$$

- Chomsky-Schützenberger fixpoint semantics:

$$\mathcal{S}[\![E]\!] \;=\; \mathbf{lfp}^{\subseteq}\, \mathcal{F}[\![E]\!]$$

$$\begin{aligned}
\mathcal{F}[\![E]\!]X \;\triangleq\; & \mathcal{S}[\![Num]\!] \cup \mathcal{S}[\![Var]\!] \\
& \cup\{e_1 + e_2 \mid e_1, e_2 \in X\} \\
& \cup\{-e \mid e \in X\} \cup \{(e) \mid e \in X\}
\end{aligned}$$

Example: Grammars (cont'd)

- FIRST abstraction of a language $X \in A^*$:

$$\alpha_F(X) \triangleq \{\ell \mid \exists \sigma \in A^* : \ell\sigma \in X\} \cup \{\epsilon \mid \epsilon \in X\}$$

- Galois connection:

$$\langle \wp(A^*), \subseteq \rangle \xleftarrow[\alpha_F]{\gamma_F} \langle \wp(A \cup \{\epsilon\}), \subseteq \rangle$$

where

$$\gamma_F(Y) \triangleq \{\ell\sigma \mid \ell \in Y \wedge \sigma \in A^*\} \cup \{\epsilon \mid \epsilon \in Y\}$$

Example: Grammars (cont'd)

- Commutation:

$$\alpha_F \circ \mathcal{F}[\![E]\!] \;=\; \overline{\mathcal{F}}[\![E]\!] \circ \alpha_F$$

where for $\;E \;::=\; Num \mid Var \mid E+E \mid -E \mid (E)$

$$\overline{\mathcal{F}}[\![E]\!]Y \;\triangleq\; \mathcal{S}[\![Num]\!] \cup \mathcal{S}[\![Var]\!] \cup (Y \setminus \{\epsilon\}) \cup \{+ \mid \epsilon \in Y\} \cup \{-, (\}$$

- FIRST abstract semantics:

$$\begin{aligned}
\overline{\mathcal{S}}[\![E]\!] &\triangleq \alpha_F(\mathcal{S}[\![E]\!]) \\
&= \alpha_F(\mathbf{lfp}^{\subseteq} \mathcal{F}[\![E]\!]) \qquad \text{(Chomsky-Schützenberger)} \\
&= \mathbf{lfp}^{\subseteq} \overline{\mathcal{F}}[\![E]\!] \qquad \text{(fixpoint abstraction th.)}
\end{aligned}$$

Machine-checkable calculational design

$\alpha_F \circ \mathcal{F}[\![E]\!]$

$= \lambda X \bullet \alpha_F(\mathcal{F}[\![E]\!](X))$ ⦅def. ∘⦆

$= \lambda X \bullet \{\ell \mid \exists \sigma \in A^* : \ell\sigma \in \mathcal{F}[\![E]\!](X)\} \cup \{\epsilon \mid \epsilon \in \mathcal{F}[\![E]\!](X)\}$ ⦅def. α_F⦆

$= \lambda X \bullet \{\ell \mid \exists \sigma \in A^* : \ell\sigma \in \mathcal{F}[\![E]\!](X)\}$ ⦅since. $\forall X : \epsilon \notin \mathcal{F}[\![E]\!](X)$⦆

$= \lambda X \bullet \{\ell \mid \exists \sigma \in A^* : \ell\sigma \in \mathcal{S}[\![Num]\!] \cup \mathcal{S}[\![Var]\!] \cup \{e_1 + e_2 \mid e_1, e_2 \in X\} \cup \{-e \mid e \in X\} \cup \{(e) \mid e \in X\}\}$
⦅def. $\mathcal{F}[\![E]\!]X \triangleq \mathcal{S}[\![Num]\!] \cup \mathcal{S}[\![Var]\!] \cup \{e_1 + e_2 \mid e_1, e_2 \in X\} \cup \{-e \mid e \in X\} \cup \{(e) \mid e \in X\}$⦆

$= \lambda X \bullet \mathcal{S}[\![Num]\!] \cup \mathcal{S}[\![Var]\!] \cup \{\ell \mid \exists \sigma \in A^* : \ell\sigma \in X\} \cup \{+ \mid \epsilon \in X\} \cup \{-\} \cup \{(\}$
⦅def. $\in$ and $\epsilon + e_2 = +e_2$⦆

$= \lambda X \bullet \mathcal{S}[\![Num]\!] \cup \mathcal{S}[\![Var]\!] \cup (\alpha_F(X) \setminus \{\epsilon\}) \cup \{+ \mid \epsilon \in \alpha_F(X)\} \cup \{-\} \cup \{(\}$
⦅def. α_F and $\epsilon \in X \iff \epsilon \in \alpha_F(X)$⦆

$= \lambda X \bullet \overline{\mathcal{F}}[\![E]\!](\alpha_F(X))$
⦅by defining $\overline{\mathcal{F}}[\![E]\!]Y \triangleq \mathcal{S}[\![Num]\!] \cup \mathcal{S}[\![Var]\!] \cup (Y \setminus \{\epsilon\}) \cup \{+ \mid \epsilon \in Y\} \cup \{-, (\}$⦆

$= \overline{\mathcal{F}}[\![E]\!] \circ \alpha_F$ ⦅def. ∘⦆

Algorithm

- Read the grammar G, establish the system of equations $Y \;=\; \overline{\mathcal{F}}[\![G]\!](Y)$, solve by chaotic iterations

- **This is**, up to [en]coding details, **the classical algorithm:**
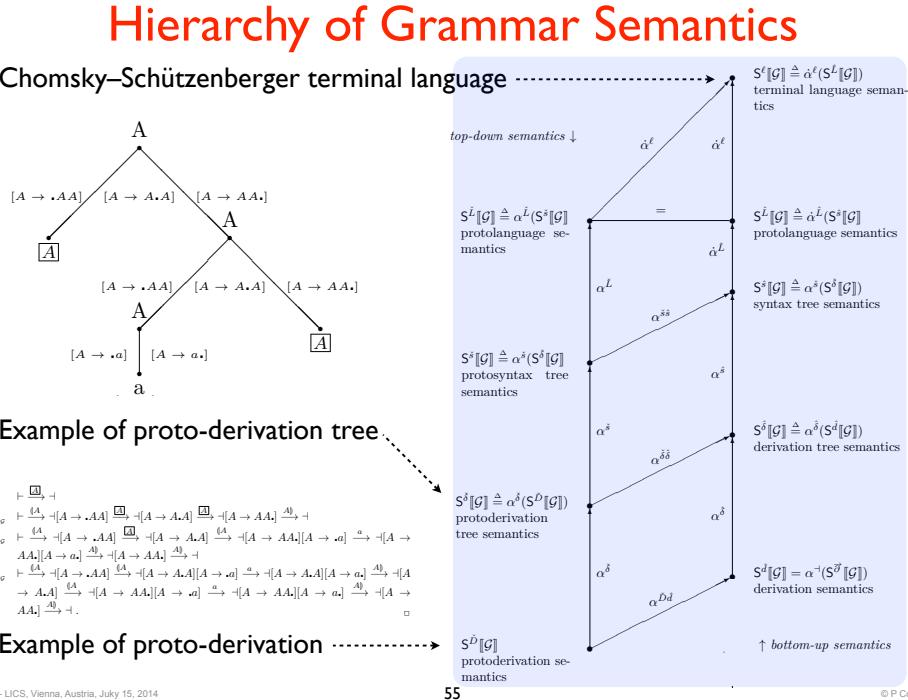
```
for each α ∈ (T ∪ ε)
    FIRST(α) ← α
for each A ∈ NT
    FIRST(A) ← ∅
while (FIRST sets are still changing)
    for each p ∈ P, where p has the form A→β
        if β is β₁β₂...βₖ, where βᵢ ∈ T ∪ NT, then
            FIRST(A) ← FIRST(A) ∪ (FIRST(β₁) − {ε})
            i ← 1
            while (ε ∈ FIRST(βᵢ) and i ≤ k-1)
                FIRST(A) ← FIRST(A) ∪ (FIRST(βᵢ₊₁) − {ε})
                i ← i + 1
        if i = k and ε ∈ FIRST(βₖ)
            then FIRST(A) ← FIRST(A) ∪ {ε}
```

Keith D. Cooper, Linda Torczon: Engineering a Compiler. Morgan Kaufmann 2004

Hierarchies of abstractions

Comparison of abstractions

- $\langle P, \preccurlyeq \rangle \xleftarrow[\alpha_1]{\gamma_1} \langle Q, \sqsubseteq \rangle$

  is more precise than

 $\langle P, \preccurlyeq \rangle \xleftarrow[\alpha_2]{\gamma_2} \langle R, \lesssim \rangle$

  iff $\gamma_2(R) \subseteq \gamma_1(Q)$
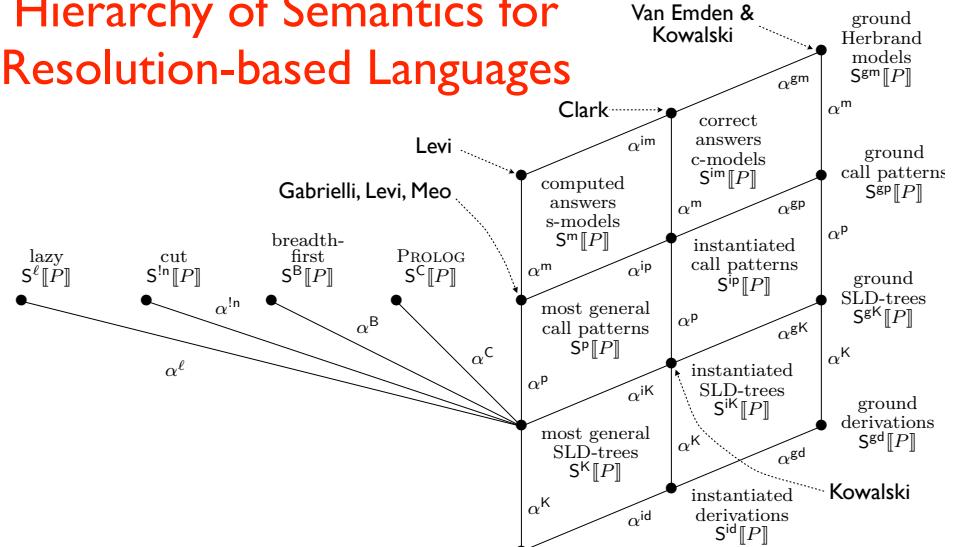
  (every abstraction in R is exactly expressible by Q)

- We say that Q is a refinement of R and R that is a abstraction of Q

- A pre-order

Hierarchy of Grammar Semantics

Chomsky–Schützenberger terminal language

Example of proto-derivation tree

Example of proto-derivation

Hierarchy of Semantics for Resolution-based Languages

Patrick Cousot, Radhia Cousot, Roberto Giacobazzi: Abstract interpretation of resolution-based semantics. Theor. Comput. Sci. 410(46): 4724-4746 (2009)

Example 11 of exact abstraction: graphs

Ilya Sergey, Jan Midtgaard, Dave Clarke: Calculating Graph Algorithms for Dominance and Shortest Path. MPC 2012: 132-156

57

Transition system

- Transition system: $\langle \Sigma,\ \mathbb{A},\ \rightarrow \rangle$

  transition relation: $\rightarrow\ \in\ \wp(\Sigma \times \mathbb{A} \times \Sigma)$

  transitions/edges: $\sigma \xrightarrow{\mathbf{A}} \sigma'$

- Example: non-negatively weighted graphs $\mathbb{A} \triangleq \mathbb{N}$

58

Finite paths

- Finite paths:

$$\Theta^+ \ \triangleq\ \{\sigma_0 \xrightarrow{\mathbf{A}_0} \sigma_1 \ldots \sigma_{n-1} \xrightarrow{\mathbf{A}_{n-1}} \sigma_n \mid n \geqslant 0 \wedge \\ \forall i \in [0,n] : \sigma_i \in \Sigma \wedge \forall i \in [0,n) : \mathbf{A}_i \in \mathbb{A}\}$$

- Paths between two vertices:

$$\Pi \ \in\ (\Sigma \times \Sigma) \mapsto \wp(\Theta^+)$$
$$\Pi(\sigma,\sigma') \ \triangleq\ \{\sigma_0 \xrightarrow{\mathbf{A}_0} \sigma_1 \ldots \sigma_{n-1} \xrightarrow{\mathbf{A}_{n-1}} \sigma_n \mid \sigma = \sigma_0 \wedge n \geqslant 0 \wedge \\ \forall i \in [0,n-1] : \sigma_i \xrightarrow{\mathbf{A}_i} \sigma_{i+1} \wedge \sigma_n = \sigma'\}$$

destination
departure

59

Fixpoint characterization

- Pointwise fixpoint charaterization:

$$\Pi \ =\ \mathbf{lfp}^{\dot{\subseteq}} F$$
$$F \ \in\ ((\Sigma \times \Sigma) \mapsto \wp(\Theta^+)) \mapsto ((\Sigma \times \Sigma) \mapsto \wp(\Theta^+))$$
$$F(X)(\sigma,\sigma') \ =\ \left(\sigma = \sigma' \ ?\ \{\sigma\} \ \text{\textsection}\ \bigcup_{\sigma'' \in \Sigma} \{\sigma \xrightarrow{\mathbf{A}} \sigma'' \pi \mid \sigma \xrightarrow{\mathbf{A}} \sigma'' \wedge \sigma'' \pi \in X(\sigma'',\sigma')\} \right)$$

(a path of n transitions is either a single vertex ($n = 0$) or an edge followed by a path of n - 1 transitions)

60

Minimal path length abstraction

- Edges have non-negative lengths $\quad \mathbb{A} = \mathbb{N}$

- Abstraction:

$$\alpha \in \Theta^+ \mapsto \mathbb{N}$$
$$\alpha(\sigma) \triangleq 0$$
$$\alpha(\sigma \xrightarrow{n} \sigma'\pi) \triangleq n + \alpha(\sigma'\pi)$$

$$\alpha \in \wp(\Theta^+) \mapsto \mathbb{N}^\infty$$
$$\alpha(X) \triangleq \min\{\alpha(\pi) \mid \pi \in X\}$$

  where
$$\min \emptyset = +\infty$$
$$\mathbb{N}^\infty \triangleq \mathbb{N} \cup \{+\infty\}$$
$\langle \mathbb{N}^\infty, \geqslant, \min \rangle$ is a complete lattice

Galois connection

- $\quad \langle \wp(\Theta^+), \subseteq \rangle \xleftrightarrow[\alpha]{\gamma} \langle \mathbb{N}^\infty, \geqslant \rangle$

- Pointwise extension:

$$\dot\alpha \in (\Sigma \times \Sigma \mapsto \wp(\Theta^+)) \mapsto (\Sigma \times \Sigma \mapsto \mathbb{N}^\infty)$$
$$\dot\alpha(X)(\sigma, \sigma') \triangleq \alpha(X(\sigma, \sigma'))$$

- Pointwise Galois connection:

$$\langle (\Sigma \times \Sigma) \mapsto \wp(\Theta^+), \dot\subseteq \rangle \xleftrightarrow[\dot\alpha]{\dot\gamma} \langle (\Sigma \times \Sigma) \mapsto \mathbb{N}^\infty, \dot\geqslant \rangle$$

Shortest distance

- Shortest distance $\Delta(\sigma, \sigma')$ between any two vertices

- $\quad \Delta \in (\Sigma \times \Sigma) \mapsto \mathbb{N}^\infty$
  $$\Delta \triangleq \dot\alpha(\Pi) = \dot\alpha(\mathbf{lfp}^{\dot\subseteq} F)$$

Calculational design of the shortest distance algorithm

$\dot\alpha \circ F$

$= \lambda X \bullet \dot\alpha(F(X))$ $\wr\text{def. } \circ\wr$

$= \lambda(\sigma, \sigma') \bullet \lambda X \bullet \dot\alpha(F(X))(\sigma, \sigma')$ $\wr\text{def. } \lambda x \bullet e\wr$

$= \lambda(\sigma, \sigma') \bullet \lambda X \bullet \dot\alpha(\lambda(\sigma, \sigma') \bullet (\![\sigma = \sigma' \, ? \, \{\sigma\} \, \text{?} \, \bigcup_{\sigma'' \in \Sigma} \{\sigma \xrightarrow{n} \sigma''\pi \mid \sigma \xrightarrow{n} \sigma'' \wedge \sigma''\pi \in X(\sigma'', \sigma')\}]\!))(\sigma, \sigma')$ $\wr\text{def. } F\wr$

$= \lambda(\sigma, \sigma') \bullet \lambda X \bullet \alpha(\![\sigma = \sigma' \, ? \, \{\sigma\} \, \text{?} \, \bigcup_{\sigma'' \in \Sigma} \{\sigma \xrightarrow{n} \sigma''\pi \mid \sigma \xrightarrow{n} \sigma'' \wedge \sigma''\pi \in X(\sigma'', \sigma')\}]\!)$ $\wr\text{def. } \dot\alpha(X)(\sigma, \sigma') \triangleq \alpha(\Delta(\sigma, \sigma'))\wr$

$= \lambda(\sigma, \sigma') \bullet \lambda X \bullet (\![\sigma = \sigma' \, ? \, \alpha(\{\sigma\}) \, \text{?} \, \alpha(\bigcup_{\sigma'' \in \Sigma} \{\sigma \xrightarrow{n} \sigma''\pi \mid \sigma \xrightarrow{n} \sigma'' \wedge \sigma''\pi \in X(\sigma'', \sigma')\})]\!)$ $\wr\text{def. conditional } (\![\ldots ? \ldots \text{?} \ldots]\!)\wr$

$= \lambda(\sigma, \sigma') \bullet \lambda X \bullet (\![\sigma = \sigma' \, ? \, \alpha(\{\sigma\}) \, \text{?} \, \min_{\sigma'' \in \Sigma} \alpha(\{\sigma \xrightarrow{n} \sigma''\pi \mid \sigma \xrightarrow{n} \sigma'' \wedge \sigma''\pi \in X(\sigma'', \sigma')\})]\!)$ $\wr\text{join preservation in Galois C.}\wr$

$= \lambda(\sigma, \sigma') \bullet \lambda X \bullet (\![\sigma = \sigma' \, ? \, \min\{\alpha(\pi) \mid \pi \in \{\sigma\}\} \, \text{?} \, \min_{\sigma'' \in \Sigma} \min\{\alpha(\pi) \mid \pi \in \{\sigma \xrightarrow{n} \sigma''\pi \mid \sigma \xrightarrow{n} \sigma'' \wedge \sigma''\pi \in X(\sigma'', \sigma')\}\}]\!)$ $\wr\text{def. } \alpha(X) \triangleq \min\{\alpha(\pi) \mid \pi \in X\}\wr$

Calculational design of the shortest distance algorithm

$= \lambda(\sigma,\sigma') \bullet \lambda X \bullet (\![\sigma = \sigma' \, ? \, \min\{\alpha(\sigma)\} \, \$ \, \min_{\sigma''\in\Sigma} \min\{\alpha(\sigma \xrightarrow{n} \sigma''\pi) \mid \sigma \xrightarrow{n} \sigma'' \wedge$
$\sigma''\pi \in X(\sigma'',\sigma')\}]\!)$ ⁅def. $\in$⁆

$= \lambda(\sigma,\sigma') \bullet \lambda X \bullet (\![\sigma = \sigma' \, ? \, \min\{0\} \, \$ \, \min_{\sigma''\in\Sigma} \min\{n + \alpha(\sigma''\pi) \mid \sigma \xrightarrow{n} \sigma'' \wedge \sigma''\pi \in$
$X(\sigma'',\sigma')\}]\!)$ ⁅def. $\alpha(\sigma) \triangleq 0$ and $\alpha(\sigma \xrightarrow{n} \sigma'\pi) \triangleq n + \alpha(\sigma'\pi)$⁆

$= \lambda(\sigma,\sigma') \bullet \lambda X \bullet (\![\sigma = \sigma' \, ? \, 0 \, \$ \, \min_{\sigma''\in\Sigma}\{n + \min\{\alpha(\sigma''\pi) \mid \sigma''\pi \in X(\sigma'',\sigma')\} \mid \sigma \xrightarrow{n}$
$\sigma''\}]\!)$ ⁅def. $\min$⁆

$= \lambda(\sigma,\sigma') \bullet \lambda X \bullet (\![\sigma = \sigma' \, ? \, 0 \, \$ \, \min_{\sigma''\in\Sigma}\{n + \dot\alpha(X)(\sigma'',\sigma') \mid \sigma \xrightarrow{n} \sigma''\}]\!)$
⁅def. $\dot\alpha(X)(\sigma'',\sigma') \triangleq \alpha(X(\sigma'',\sigma')) = \min\{\alpha(\pi) \mid \pi \in X(\sigma'',\sigma')\} = \min\{\alpha(\sigma''\pi) \mid \sigma''\pi \in X(\sigma'',\sigma')\}$ where $\pi = \sigma''\pi'$ and π' can be empty ⁆

$= \lambda(\sigma,\sigma') \bullet \lambda X \bullet G(\dot\alpha(X))(\sigma,\sigma')$

by defining

$G(X)(\sigma,\sigma') \;=\; (\![\sigma = \sigma' \, ? \, 0 \, \$ \, \min_{\sigma''\in\Sigma}\{n + X(\sigma'',\sigma') \mid \sigma \xrightarrow{n} \sigma''\}]\!)$

$= \lambda X \bullet G(\dot\alpha(X))$ ⁅def. $\lambda x \bullet e$⁆

$= G \circ \dot\alpha$ ⁅def. $\circ$⁆

Shortest distance in fixpoint form

- By the fixpoint abstraction theorem

$$\Delta \;=\; \alpha(\mathbf{lfp}^{\dot\subseteq} F)$$
$$=\; \mathbf{lfp}^{\dot\geqslant} G$$
$$=\; \min_{n\in\mathbb{N}} G^n(\lambda(\sigma,\sigma') \bullet +\infty)$$

where the iterates are

- $G^0(X) = X$
- $G^{n+1} = G \circ G^n, \; n \in \mathbb{N}$

Shortest distance algorithm

```
forall σ ∈ Σ do
  forall σ' ∈ Σ do
    Δ(σ,σ') := if σ = σ' then 0 else +∞;
repeat
  change := false;
  forall σ ∈ Σ do
    forall σ' ∈ Σ do
      forall σ'' ∈ Σ do
        if (σ ≠ σ' ∧ σ ⁿ→ σ'' ∧ Δ(σ,σ') > n + Δ(σ'',σ')) then
          { Δ(σ,σ') := n + Δ(σ'',σ');
            change := true }
until ¬change;
```
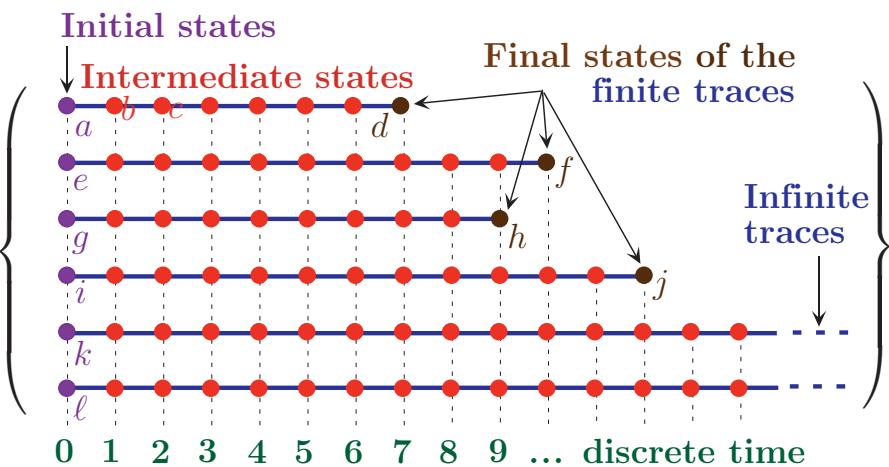
- Not Floyd-Warshall? Take instead:

$$\alpha(\sigma) \;\triangleq\; 0$$
$$\alpha(\sigma \xrightarrow{n} \sigma') \;\triangleq\; n$$
$$\alpha(\pi\sigma\pi') \;\triangleq\; \alpha(\pi\sigma) + \alpha(\sigma\pi')$$


Example III of exact abstractions: semantics

Patrick Cousot: Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. Theor. Comput. Sci. 277(1-2): 47-103 (2002)

Trace semantics

Initial states

Intermediate states

Final states of the finite traces

Infinite traces

0 1 2 3 4 5 6 7 8 9 ... discrete time
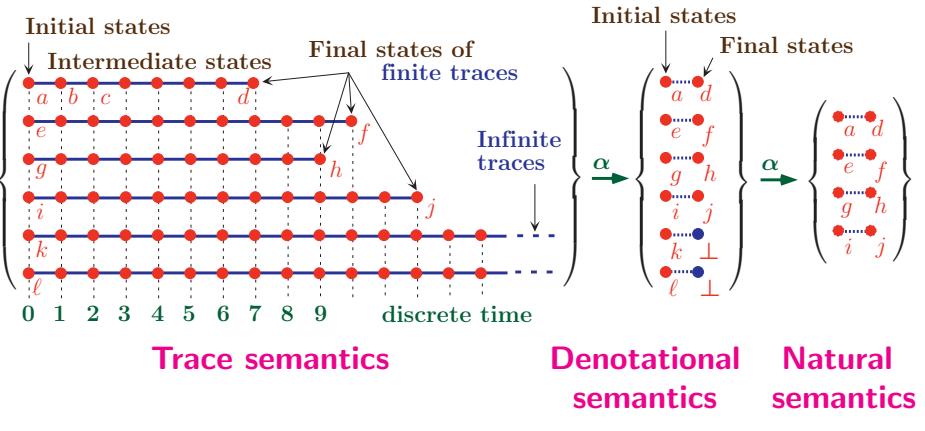
Abstraction to denotational/natural semantics

Initial states

Intermediate states

Final states of finite traces

Infinite traces

0 1 2 3 4 5 6 7 8 9   discrete time

α

α
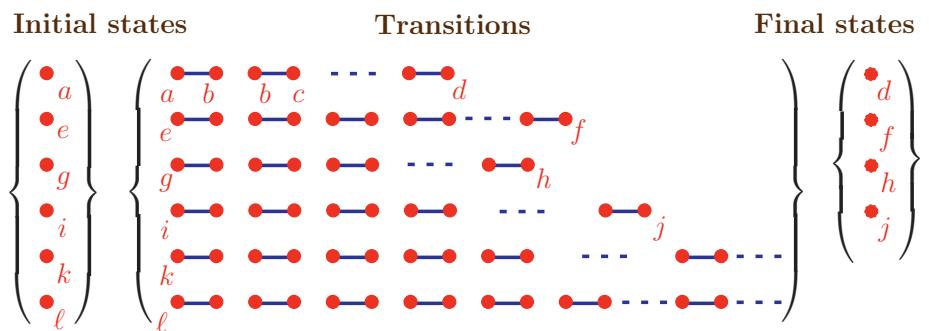
Trace semantics

Denotational semantics

Natural semantics

Abstraction to small-steps operational semantics

Initial states

Transitions

Final states
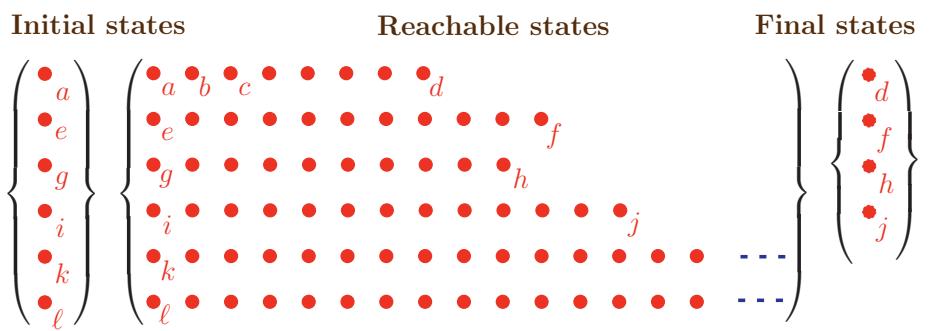
(Small-Step) Operational Semantics

Abstraction to reachability/invariance

Initial states

Reachable states

Final states
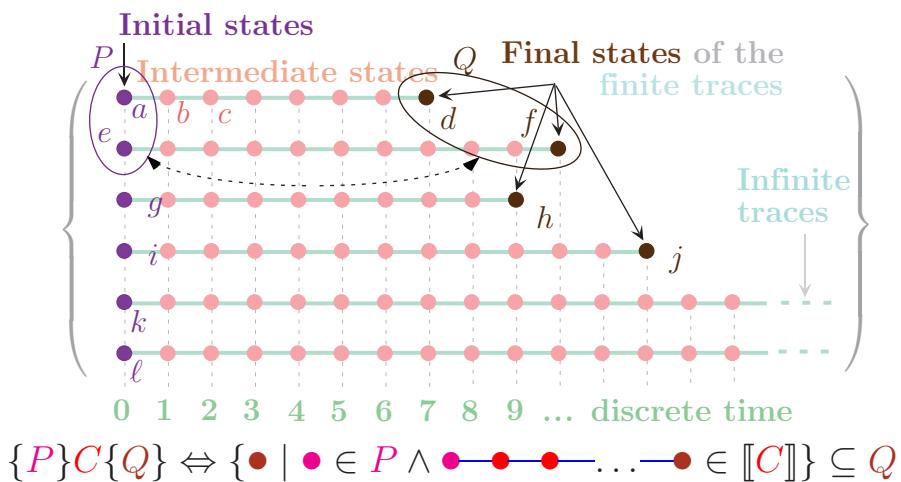
Partial Correctness / Invariance Semantics

Abstraction to Hoare logic
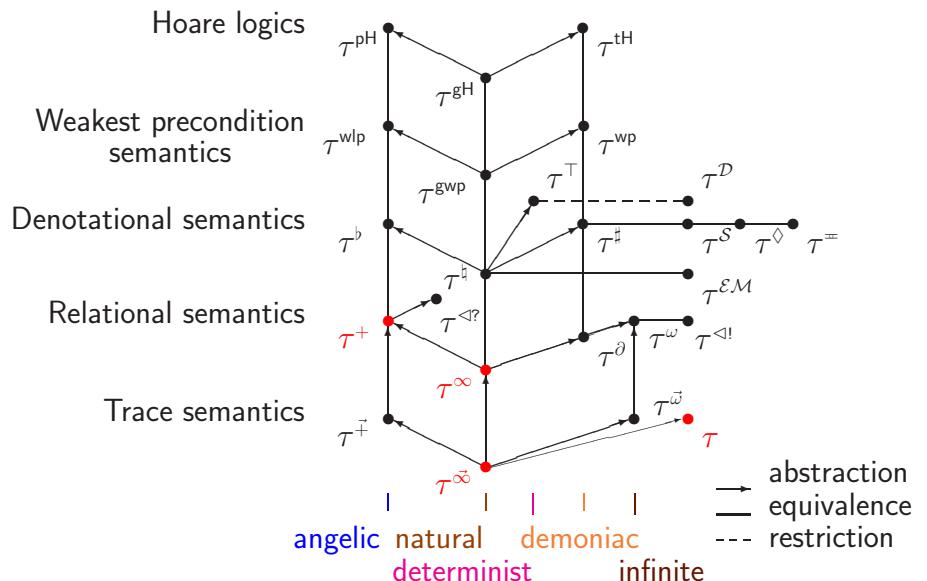
Initial states

P

Intermediate states Q **Final states** of the finite traces

Infinite traces

a b c
e
d f
g
h
i j
k
ℓ

0 1 2 3 4 5 6 7 8 9 ... discrete time

$$\{P\}C\{Q\} \Leftrightarrow \{\bullet \mid \bullet \in P \wedge \bullet\!-\!\bullet\!-\!\bullet\ldots\!-\!\bullet \in [\![C]\!]\} \subseteq Q$$

Poset of semantics

Hoare logics τ^{pH} τ^{gH} τ^{tH}

Weakest precondition semantics τ^{wlp} τ^{wp}

τ^{gwp} $\tau^{\top}$ $\tau^{\mathcal{D}}$

Denotational semantics $\tau^{\flat}$ $\tau^{\natural}$ $\tau^{\sharp}$ $\tau^{\mathcal{S}}$ $\tau^{\lozenge}$ $\tau^{=}$

$\tau^{\mathcal{EM}}$

Relational semantics τ^{+} $\tau^{\triangleleft?}$ τ^{∂} τ^{ω} $\tau^{\triangleleft!}$

τ^{∞}

Trace semantics $\tau^{\vec{+}}$ $\tau^{\vec{\omega}}$ τ

$\tau^{\vec{\infty}}$

angelic natural demoniac
determinist infinite

→ abstraction
— equivalence
- - - restriction

Analysis & Verification

Verification/static analysis by abstract interpretation

- Define the syntax of programs $\mathbf{P} \in \mathbb{L}$

- Define the concrete semantics of programs:

  - $\mathscr{D}[\![\mathbf{P}]\!]$ concrete semantic domain

  - $\forall\, \mathbf{P} \in \mathbb{L} : S[\![\mathbf{P}]\!] \in \mathscr{D}[\![\mathbf{P}]\!]$ concrete semantics

- Concrete/semantic properties: $\wp(\mathscr{D}[\![\mathbf{P}]\!])$

- Collecting semantics: $\{S[\![\mathbf{P}]\!]\} \in \wp(\mathscr{D}[\![\mathbf{P}]\!])$

  (the strongest property of the semantics, which implies all other semantic properties)

Verification/static analysis by abstract interpretation

- Define the abstraction:
  - $\langle \wp(\mathscr{D}[\![\mathrm{P}]\!]), \subseteq \rangle \xrightleftharpoons[\alpha[\![\mathrm{P}]\!]]{\gamma[\![\mathrm{P}]\!]} \langle \mathscr{A}[\![\mathrm{P}]\!], \sqsubseteq \rangle$
- Calculate the abstract semantics:
  - $S^{\#}[\![\mathrm{P}]\!] = \alpha[\![\mathrm{P}]\!](\{S[\![\mathrm{P}]\!]\})$ exact abstraction
  - $S^{\#}[\![\mathrm{P}]\!] \sqsupseteq \alpha[\![\mathrm{P}]\!](\{S[\![\mathrm{P}]\!]\})$ approximate abstraction
- Soundness (by construction):

  $$\forall \, \mathrm{P} \in \mathbb{L} : \forall \, Q \in \mathscr{A} : S^{\#}[\![\mathrm{P}]\!] \sqsubseteq Q \implies S[\![\mathrm{P}]\!] \in \gamma[\![\mathrm{P}]\!](Q)$$

Verification/static analysis by abstract interpretation

- Completeness (for exact abstractions only)

  $$\forall \, \mathrm{P} \in \mathbb{L} : \forall \, Q \in \mathscr{A}[\![\mathrm{P}]\!] : S[\![\mathrm{P}]\!] \in \gamma[\![\mathrm{P}]\!](Q) \implies S^{\#}[\![\mathrm{P}]\!] \sqsubseteq Q$$

- Methodolody:
  - Structural induction on programs P
  - Compositional definition[*] of $\mathscr{A}[\![\mathrm{P}]\!]$ and $\alpha[\![\mathrm{P}]\!]/\gamma[\![\mathrm{P}]\!]$
  - Fixpoint abstraction/approximation for recursion
- Verification for fixpoints is the main problem:

  $$\mathsf{lfp}^{\sqsubseteq} F^{\#}[\![\mathrm{P}]\!] \sqsubseteq Q$$

[*] Patrick Cousot, Radhia Cousot: A Galois connection calculus for abstract interpretation. POPL 2014: 3-4 + Aux. mat. 15p.

Verification/static analysis by abstract interpretation

- Method: find $I \in \mathscr{A}[\![\mathrm{P}]\!]$ such that $F^{\#}[\![\mathrm{P}]\!]I \sqsubseteq I \wedge I \sqsubseteq Q$

  (so that $\mathsf{lfp}^{\sqsubseteq} F^{\#}[\![\mathrm{P}]\!] \sqsubseteq Q$, by Tarski)

  - Verification/deductive/proof methods:
    - ask the end-user for the inductive argument I
  - Static analysis:

    1. compute I knowing $F^{\#}[\![\mathrm{P}]\!]$ and Q

    2. compute I knowing $F^{\#}[\![\mathrm{P}]\!]$ (and later given any Q check that $I \sqsubseteq Q$)

Approximate abstractions

Approximate abstractions

- The concrete properties of the standard semantics $S[\![P]\!]$ that you want to prove may not always be provable in the abstract:

$$\forall\, Q \in \mathscr{A}: S[\![P]\!] \in \gamma(Q) \;\overset{\Longleftarrow}{\underset{\not\Longrightarrow}{}}\; \overline{S}[\![P]\!] \sqsubseteq Q$$

where

$$\overline{S}[\![P]\!] \overset{\triangle}{=} \alpha \circ S[\![P]\!] \circ \gamma$$

Why abstraction may be approximate?

- Example

$$\{\, x = y \wedge 0 \leqslant x \leqslant 10 \,\}$$
```
x := x - y;
```
$$\{\, x = 0 \wedge 0 \leqslant y \leqslant 10 \,\}$$

Interval abstraction:

$$\{\, x \in [0, 10] \wedge y \in [0, 10] \,\}$$
```
x := x - y;
```
$$\{\, x \in [\text{-}10, 10] \wedge y \in [0, 10] \,\}$$

(but for constants, the interval abstraction can't express equality)

Refinement

Refinement: good news
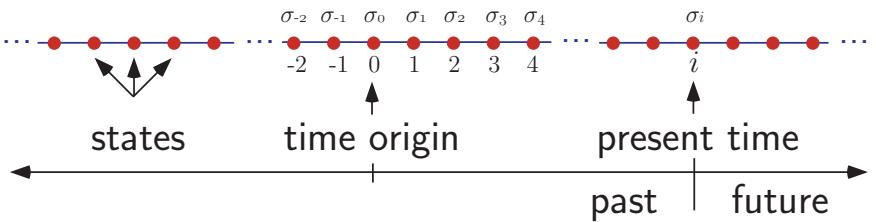
- Problem: how to prove a valid abstract property $\alpha(\{\mathsf{lfp}\, F[\![P]\!]\}) \sqsubseteq Q$ when $\alpha \circ F \sqsubseteq F^{\#} \circ \alpha$ but $\mathsf{lfp}\, F^{\#}[\![P]\!] \not\sqsubseteq Q$?

- It is always possible to refine $\langle \mathscr{A}, \sqsubseteq \rangle$ into a most abstract more precise abstraction $\langle \mathscr{A}', \sqsubseteq' \rangle$ such that $\langle \wp(\mathscr{D}), \subseteq \rangle \xrightleftharpoons[\alpha']{\gamma'} \langle \mathscr{A}', \sqsubseteq' \rangle$ and $\alpha' \circ F = F' \circ \alpha$ with $\mathsf{lfp}\, F'[\![P]\!] \sqsubseteq' \alpha' \circ \gamma\,(Q)$

(thus proving $\mathsf{lfp}\, F[\![P]\!] \in \gamma'(Q)$ which implies $\mathsf{lfp}\, F[\![P]\!] \in \gamma(Q)$)

Roberto Giacobazzi, Francesco Ranzato, Francesca Scozzari: Making abstract interpretations complete. J. ACM 47(2): 361-416 (2000)

Refinement: bad news

- But, refinements of an abstraction can be intrinsically incomplete

- The only complete refinement of that abstraction for the collecting semantics is :
  - the identity (i.e. no abstraction at all)

- In that case, the only complete refinement of the abstraction is to the collecting semantics and any other refinement is always imprecise

---

Example of intrinsic approximate refinement

- Consider executions traces $\langle i, \sigma \rangle$ with infinite past and future:

Patrick Cousot, Radhia Cousot: Temporal Abstract Interpretation. POPL 2000: 12-25

Example of intrinsic approximate refinement

- Consider the temporal specification language $\widehat{\mu^*}$ (containing LTL, CTL, CTL*, and Kozen's μ-calculus as fragments):

$$
\begin{array}{llll}
\varphi ::= & \boldsymbol{\sigma}_S & S \in \wp(\mathbb{S}) & \text{state predicate} \\
 | & \boldsymbol{\pi}_t & t \in \wp(\mathbb{S} \times \mathbb{S}) & \text{transition predicate} \\
 | & \oplus \varphi_1 & & \text{next} \\
 | & \varphi_1{}^{\curvearrowleft} & & \text{reversal} \\
 | & \varphi_1 \vee \varphi_2 & & \text{disjunction} \\
 | & \neg \varphi_1 & & \text{negation} \\
 | & X & X \in \mathbb{X} & \text{variable} \\
 | & \boldsymbol{\mu} X \cdot \varphi_1 & & \text{least fixpoint} \\
 | & \boldsymbol{\nu} X \cdot \varphi_1 & & \text{greatest fixpoint} \\
 | & \forall \varphi_1 : \varphi_2 & & \text{universal state closure}
\end{array}
$$

Patrick Cousot, Radhia Cousot: Temporal Abstract Interpretation. POPL 2000: 12-25

Example of intrinsic approximate refinement

- Consider universal model-checking abstraction:
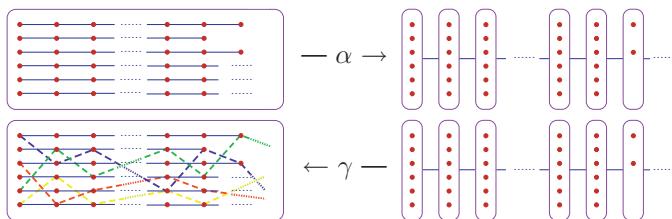
$$
\mathrm{MC}^{\forall}_M(\phi) = \alpha^{\forall}_M(\llbracket \phi \rrbracket) \in \wp(\textit{Traces}) \to \wp(\textit{States})
$$

$$
= \{ s \in \textit{States} \mid \forall \langle i, \sigma \rangle \in \textit{Traces}_M . (\sigma_i = s) \Rightarrow \langle i, \sigma \rangle \in \llbracket \phi \rrbracket \}
$$

where M is defined by a transition system

(and dually the existential model-checking abstraction)

Example of intrinsic approximate refinement

- The abstraction from a set of traces to a trace of sets is sound but *incomplete*, even for finite systems [*]

$$— \alpha →$$
$$← \gamma —$$
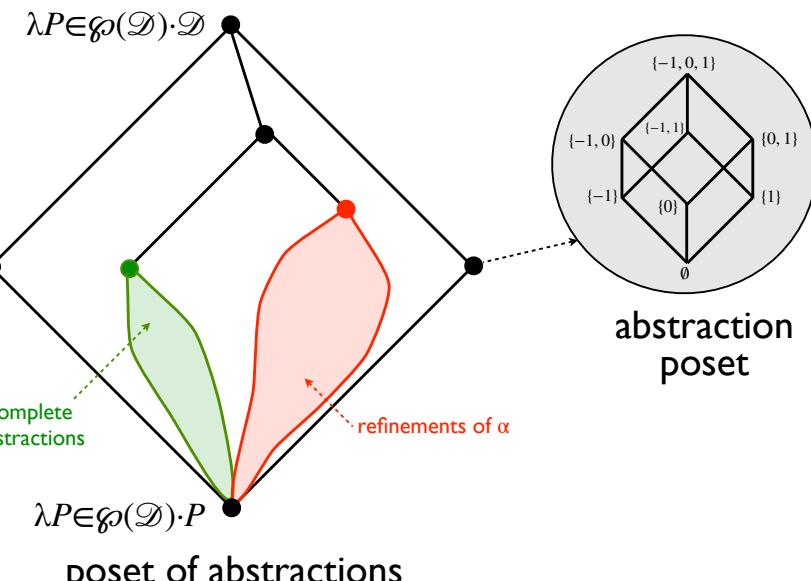
- Any refinement of this abstraction is *incomplete* (but to the infinite past/future trace semantics itself) [**]

[*] Patrick Cousot, Radhia Cousot: Temporal Abstract Interpretation. POPL 2000: 12-25

[**] Roberto Giacobazzi, Francesco Ranzato: Incompleteness of states w.r.t. traces in model checking. Inf. Comput. 204(3): 376-407 (2006)

Intrinsic approximate refinement

$$\lambda P \in \wp(\mathscr{D}) \cdot \mathscr{D}$$

$$\{-1, 0, 1\}$$
$$\{-1, 0\} \quad \{-1, 1\} \quad \{0, 1\}$$
$$\{-1\} \quad \{0\} \quad \{1\}$$
$$\emptyset$$

abstraction poset

complete abstractions

refinements of α

$$\lambda P \in \wp(\mathscr{D}) \cdot P$$

poset of abstractions
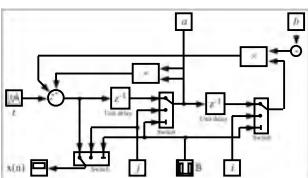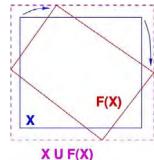
In general refinement does not terminate

- Example: filter invariant abstraction:
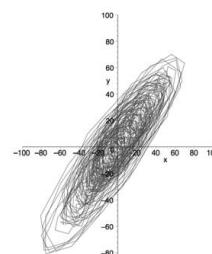
2nd order filter:
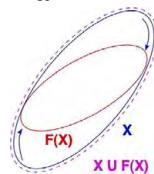
Unstable polyhedral abstraction:

Counter-example guided refinement will indefinitely add missing points according to the execution trace:

Stable ellipsoidal abstraction:

Julien Bertrane, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, & Xavier Rival. Static Analysis and Verification of Aerospace Software by Abstract Interpretation. In *AIAA Infotech@@Aerospace 2010*, Atlanta, Georgia. American Institute of Aeronautics and Astronautics, 20—22 April 2010. © AIAA.

In general refinement does not terminate

- Narrowing is needed to stop infinite iterated automatic refinements:

  e.g. SLAM stops refinement after 20mn

- Intelligence is needed for refinement:

  e.g. human-driven refinement of Astrée

Thomas Ball, Vladimir Levin, Sriram K. Rajamani: A decade of software model checking with SLAM. Commun. ACM 54(7): 68-76 (2011)

Julien Bertrane, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, & Xavier Rival. Static Analysis and Verification of Aerospace Software by Abstract Interpretation. In *AIAA Infotech@@Aerospace 2010*, Atlanta, Georgia. American Institute of Aeronautics and Astronautics, 20—22 April 2010. © AIAA.

Slide 93

Finite versus infinite abstractions

Slide 94

[In]finite abstractions

- Given a program P and a program property Q which holds (*i.e.* lfp $F[\![P]\!] \in Q$) there exists a most abstract abstraction in a **finite** domain $\mathscr{A}[\![P]\!]$ to prove it [*]

- Example:

```
x=0; while x<1 do x++ ⟶ {⊥, [0,0], [0,1],[-∞,∞]}

x=0; while x<2 do x++ ⟶ {⊥, [0,0], [0,1], [0,2],[-∞,∞]}

...

x=0; while x<n do x++ ⟶ {⊥, [0,0], [0,1], [0,2], [0,3], ..., [0,n],[-∞,∞]}

...
```

[*] Patrick Cousot: Partial Completeness of Abstract Fixpoint Checking. SARA 2000: 1-25

Slide 95

[In]finite abstractions

- No such domain exists for infinitely many programs

    - $\bigcup_{P \in \mathbb{L}} \mathscr{A}[\![P]\!]$ is infinite

    Example: $\{\bot, [0,0], [0,1], [0,2], [0,3], \ldots, [0,n], [0,n+1], \ldots, [-\infty,\infty]\}$

    - $\lambda P \in \mathbb{L} . \mathscr{A}[\![P]\!]$ is not computable (for undecidable properties)

$\Longrightarrow$ finite abstractions will fail infinitely often while infinite abstractions will succeed!

## Slide 96

Fixpoint approximation in infinite abstractions
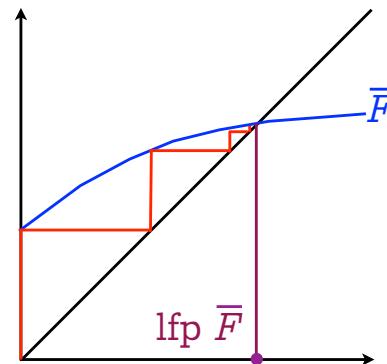
Abstract Induction
(in non-Noetherian domains)

Convergence acceleration

$\overline{F}$

$\mathrm{lfp}\ \overline{F}$

Infinite iteration

Convergence acceleration

$\overline{F}$

$\mathrm{lfp}\ \overline{F}$

Infinite iteration

$\overline{F}$

$\overline{F}(x) \leqslant x$

$\mathrm{lfp}\ \overline{F}$ x
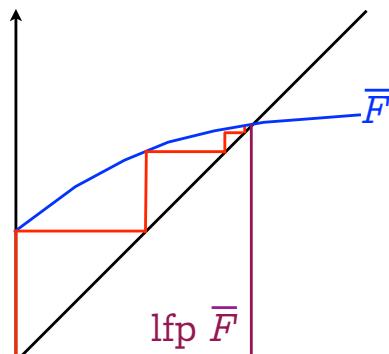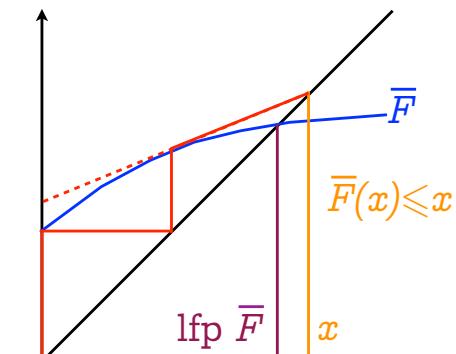
Accelerated iteration with widening
(e.g. with a widening based on the derivative
as in Newton-Raphson method[*])

---

[*] Javier Esparza, Stefan Kiefer, Michael Luttenberger: Newtonian program analysis. J. ACM 57(6): 33 (2010)

Problem with infinite abstractions

- For non-Noetherian iterations, we need
  - finitary abstract induction, and
  - finitary passage to the limit

$$X^0 = \bot, \ldots, X^{n+1} = \mathfrak{I}(X^0, \ldots, X^n, F(X^0), \ldots, F(X^n)), \ldots, \lim_{n \to \infty} X^n$$
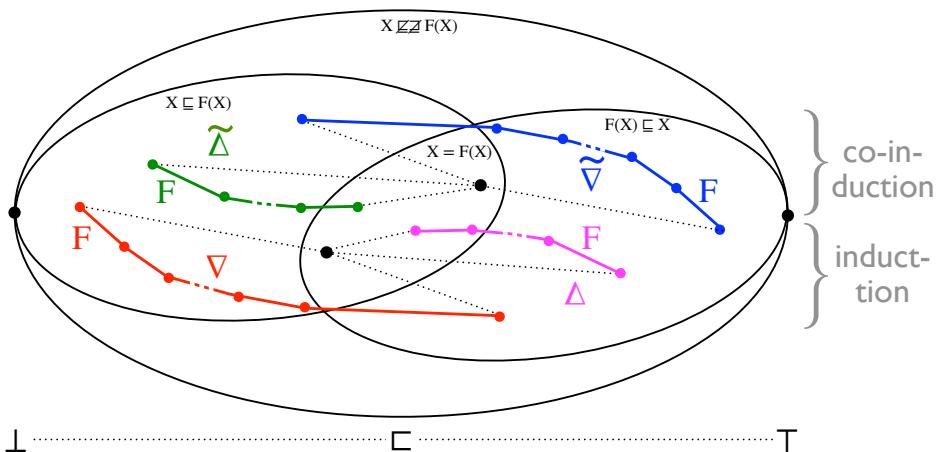
		iteration converging	
	$\mathfrak{I}$	above the limit	below the limit
Iteration starting from	below the limit	widening $\triangledown$	dual narrowing $\widetilde{\triangle}$
	above the limit	narrowing $\triangle$	dual widening $\widetilde{\triangledown}$

[Semi-]dual abstract induction methods

(separate from termination conditions)

Examples of widening/narrowing

- Abstract induction for intervals:

  - a widening [1,2]

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

  - a narrowing [2]

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

[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]

$$\forall x \in \bar{L}_2, \perp \nabla_2(j) \, x = x \nabla_2(j) \perp = x$$
$$[l_1, u_1] \, \nabla_2(j) \, [l_2, u_2]$$
$$= [if \; 0 \le l_2 < l_1 \; then \; 0 \; elsif \; l_2 < l_1 \; then \; -b - 1 \; else \; l_1 \; fi,$$
$$if \; u_1 < u_2 \le 0 \; then \; 0 \; elsif \; u_1 < u_2 \; then \; b \; else \; u_1 \; fi]$$

- Any *terminating* widening is <u>not</u> 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.

Infinitary static analysis with abstract induction

Widening

- $\langle \mathscr{A}, \sqsubseteq \rangle$ poset

- $\triangledown \in \mathscr{A} \times \mathscr{A} \longrightarrow \mathscr{A}$

- Sound widening (upper bound):

  $$\forall x, y \in \mathscr{A}: x \sqsubseteq x \triangledown y \ \wedge\ y \sqsubseteq x \triangledown y$$

- Terminating widening: for any $\langle x^n \in \mathscr{A}, n \in \mathbb{N} \rangle$, the sequence $y^0 \triangleq x^0, \ldots, y^{n+1} \triangleq y^n \triangledown x^n, \ldots$ is *ultimately stationary* ($\exists \varepsilon \in \mathbb{N}: \forall n \geqslant \varepsilon: y^n = y^\varepsilon$)

(Note: sound and terminating are independent properties)

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
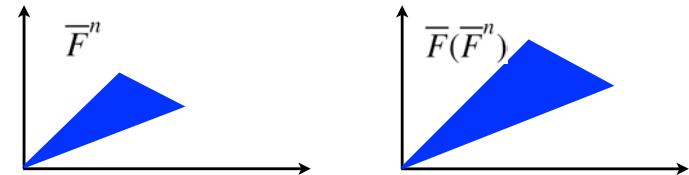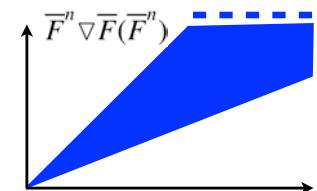
Example: (simple) widening for polyhedra

- Iterates

- Widening

Patrick Cousot. Méthodes itératives de construction et d'approximation de points fixes d'opérateurs monotones sur un treillis, analyse sémantique des programmes. *Thèse Ès Sciences Mathématiques*, Université Joseph Fourier, Grenoble, France, 21 March 1978.
Patrick Cousot, Nicolas Halbwachs: Automatic Discovery of Linear Restraints Among Variables of a Program. POPL 1978: 84-96

Iteration with widening for static analysis

- Problem: compute I such that $\mathsf{lfp}^{\sqsubseteq} F \sqsubseteq I \sqsubseteq Q$

- Compute I as the limit of the iterates:

  - $X^0 \triangleq \bot$,

  - $X^{n+1} \triangleq X^n$ when $F(X^n) \sqsubseteq X^n$ so $I = X^n$

  - $X^{n+1} \triangleq (X^n \triangledown F(X^n)) \triangle Q$ otherwise

- I can be improved by an iteration with narrowing $\triangle$

- Check that $F(I) \sqsubseteq Q$

- Example: Astrée

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

Dual narrowing

- $\langle \mathscr{A}, \sqsubseteq \rangle$ poset

- $\widetilde{\triangle} \in \mathscr{A} \times \mathscr{A} \longrightarrow \mathscr{A}$

- Sound dual narrowing (interpolation):

  $$\forall x, y \in \mathscr{A}: x \sqsubseteq y \implies x \sqsubseteq x \widetilde{\triangle} y \sqsubseteq y$$

- Terminating dual narrowing: for any $\langle x^n \in \mathscr{A}, n \in \mathbb{N} \rangle$, the sequence $y^0 \triangleq x^0, \ldots, y^{n+1} \triangleq y^n \widetilde{\triangle} x^n, \ldots$ is ultimately stationary ($\exists \varepsilon \in \mathbb{N}: \forall n \geqslant \varepsilon: y^n = y^\varepsilon$)

(Note: sound and terminating are independent properties)

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

Iteration with dual narrowing for static checking

- Problem: find I such that $\mathsf{lfp}^{\sqsubseteq} F \sqsubseteq I \sqsubseteq Q$

- Compute I as the limit of the iterates:

  - $X^0 \triangleq \bot,$

  - $X^{n+1} \triangleq X^n$ when $F(X^n) \sqsubseteq X^n$ so $I = X^n$

  - $X^{n+1} \triangleq F(X^n) \mathrel{\widetilde{\triangle}} Q,$ otherwise

- Check that $F(I) \sqsubseteq Q$

- Example: First-order logic + Graig interpolation (with some choice of one of the solutions, control of combinatorial explosion, and convergence enforcement)

Kenneth L. McMillan: Applications of Craig Interpolants in Model Checking. TACAS 2005: 1-12

Industrialization

Daniel Kästner, Christian Ferdinand, Stephan Wilhelm, Stefana Nevona, Olha Honcharova, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, Xavier Rival, and Élodie-Jane Sims. Astrée: Nachweis der Abwesenheit von Laufzeitfehlern. In *Workshop ``Entwicklung zuverlässiger Software-Systeme''*, Regensburg, Germany, June 18ᵗʰ, 2009.

Olivier Bouissou, Éric Conquet, Patrick Cousot, Radhia Cousot, Jérôme Feret, Khalil Ghorbal, Éric Goubault, David Lesens, Laurent Mauborgne, Antoine Miné, Sylvie Putot, Xavier Rival, & Michel Turin. Space Software Validation using Abstract Interpretation. In *Proc. of the Int. Space System Engineering Conf., Data Systems in Aerospace (DASIA 2009)*. Istambul, Turkey, May 2009, 7 pages. ESA.

Jean Souyris, David Delmas: Experimental Assessment of Astrée on Safety-Critical Avionics Software. SAFECOMP 2007: 479-490

David Delmas, Jean Souyris: Astrée: From Research to Industry. SAS 2007: 437-451

Jean Souyris: Industrial experience of abstract interpretation-based static analyzers. IFIP Congress Topical Sessions 2004: 393-400
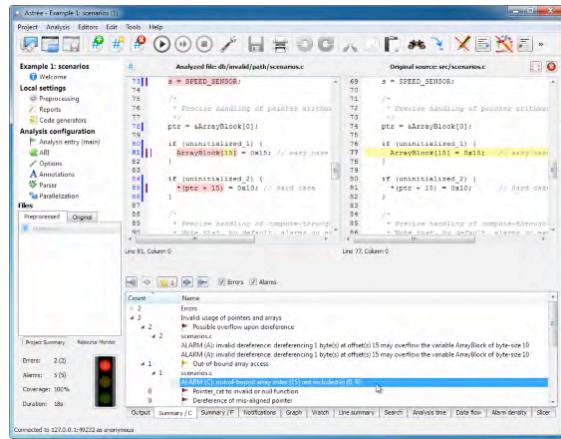
Stephan Thesing, Jean Souyris, Reinhold Heckmann, Famantanantsoa Randimbivololona, Marc Langenbach, Reinhard Wilhelm, Christian Ferdinand: An Abstract Interpretation-Based Timing Validation of Hard Real-Time Avionics Software. DSN 2003: 625-632


Astrée

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

- <u>Effectively</u> 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

Example of domain-specific abstraction: ellipses

```c
typedef enum {FALSE = 0, TRUE = 1} BOOLEAN;
BOOLEAN INIT; float P, X;

void filter () {
  static float E[2], S[2];
  if (INIT) { S[0] = X; P = X; E[0] = X; }
  else { P = (((((0.5 * X) - (E[0] * 0.7)) + (E[1] * 0.4))
             + (S[0] * 1.5)) - (S[1] * 0.7)); }
  E[1] = E[0]; E[0] = X; S[1] = S[0]; S[0] = P;
  /* S[0], S[1] in [-1327.02698354, 1327.02698354] */
}

void main () { X = 0.2 * X + 5; INIT = TRUE;
  while (1) {
    X = 0.9 * X + 35; /* simulated filter input */
    filter (); INIT = FALSE; }
}
```
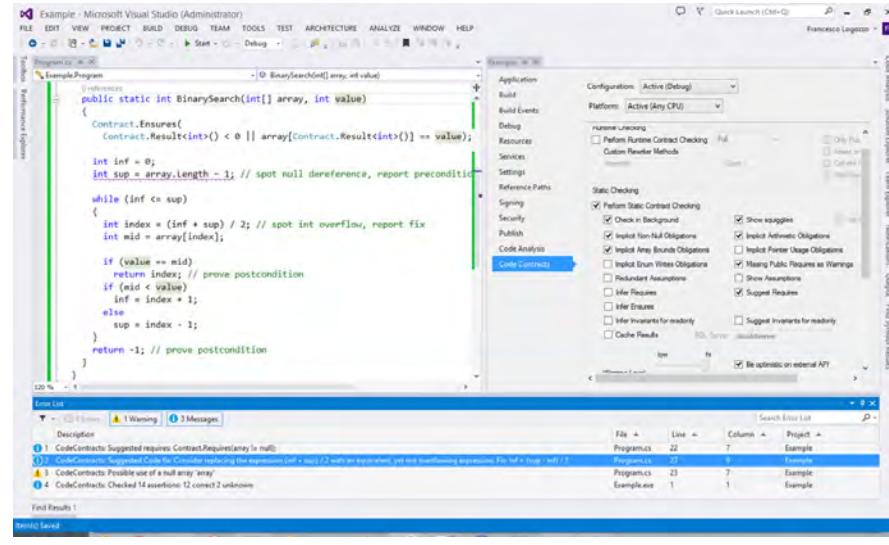
Code Contract Static Checker (`cccheck`)

- 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/ccheck 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. ccheck integration in VS (right pane with all the options integrated in the VS project system)
  3. ccheck messages in the VS error list
- The features of ccheck 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)

Conclusion

Abstract interpretation

- Intellectual tool (not to be confused with its specific application to iterative static analysis with ∇ & $\triangle$)

- No cathedral would have been built without plumb-line and square, certainly not enough for skyscrapers:

  Powerful tools are needed for progress and applicability of formal methods

Abstract interpretation

- See further developments in the proceedings with many (but unfortunately not all) references:

Abstract interpretation

- Varieties of researchers in formal methods:

  (i) explicitly use abstract interpretation, and are happy to extend its scope and broaden its applicability

  (ii) implicitly use abstract interpretation, and hide it

  (iii) pretend to use abstract interpretation, but misuse it

  (iv) don't know that they use abstract interpretation, but would benefit from it

- Never too late to upgrade

The End

The End
Thank You