Abstract Interpretation: Principles and Applications

Patrick Cousot

cims.nyu.edu/~pcousot

di.ens.fr/~cousot

SCS Distinguished Lecture Series
Gates & Hillman Centers, Rashid Auditorium 4401
CMU, Pittsburgh — April 12[th], 2012

Abstract Interpretation: Principles and Applications

Patrick Cousot

joint work Radhia Cousot

cims.nyu.edu/~pcousot

di.ens.fr/~cousot

SCS Distinguished Lecture Series
Gates & Hillman Centers, Rashid Auditorium 4401
CMU, Pittsburgh — April 12[th], 2012

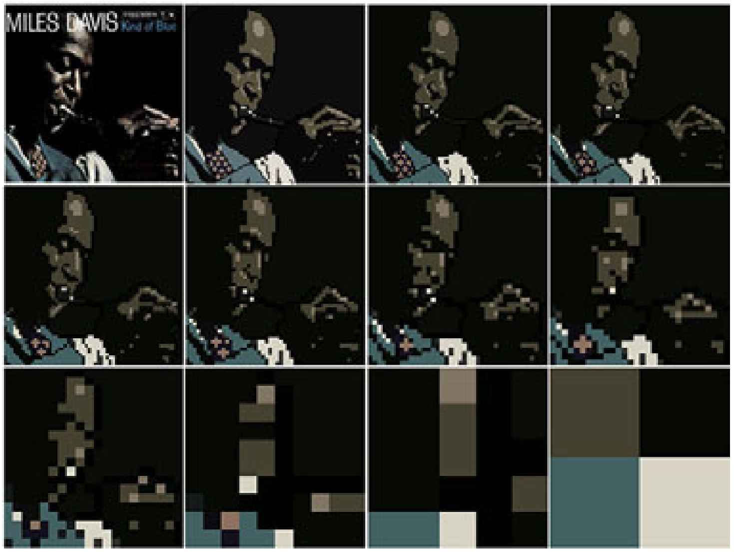
Abstract

Abstract interpretation is a theory of abstraction and constructive approximation of the mathematical structures used in the formal description of complex or infinite systems and the inference or verification of their combinatorial or undecidable properties. Developed in the late seventies with Radhia Cousot, it has been since then applied to many aspects of computer science (such as static analysis and verification, contract inference, type inference, termination inference, model-checking, abstraction refinement, program transformation (including watermarking), combination of decision procedures, security, malware detection, database queries, etc.) and more recently, to system biology.

The talk will consist in an introduction to the basic notions of abstract interpretation and the induced methodology for the systematic development of sound abstract interpretation-based tools. Examples of abstractions will be provided, from semantics to typing, grammars to safety, reachability to potential/definite termination, numerical to protein-protein abstractions, as well as applications (including in industrial use) to software, hardware and system biology.

Examples of abstraction

Pixelation of a photo by Jay Maisel

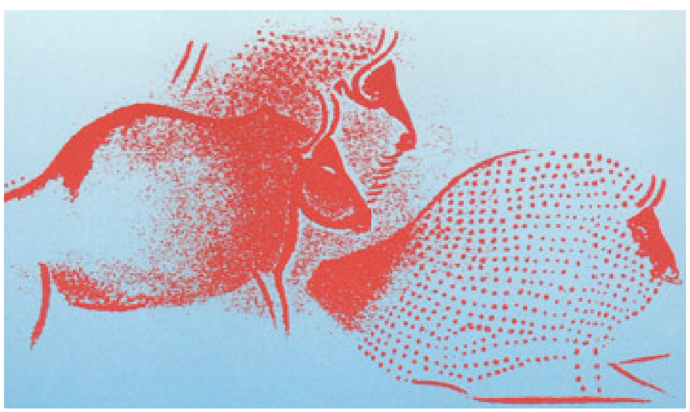


/www.petapixel.com/2011/06/23/how-much-pixelation-is-needed-before-a-photo-becomes-transformed/

***Image credit**: Photograph by Jay Maisel*

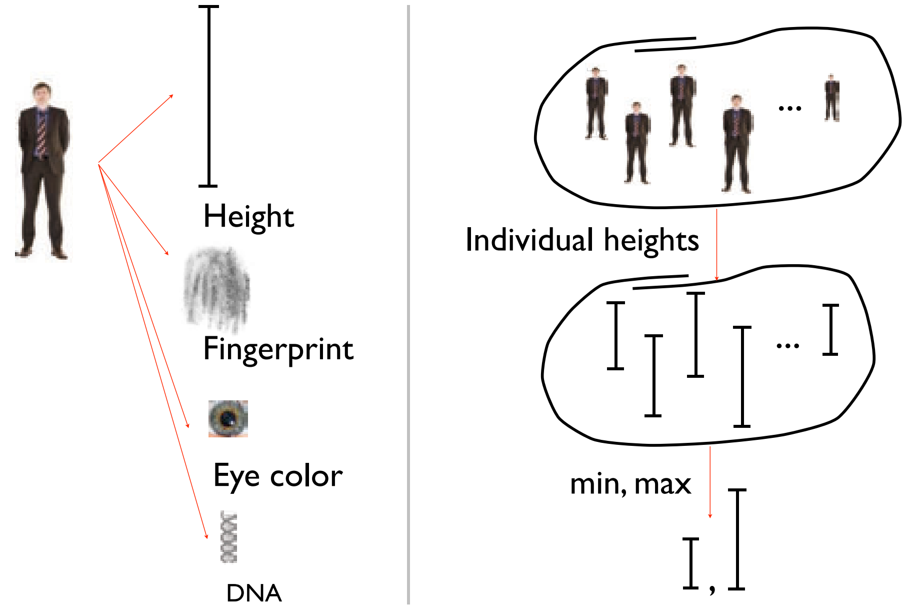
An old idea...

20 000 years old picture in a spanish cave:



The concrete is not always well-known!

Abstractions of a man / crowd



Content

- Motivation
- A touch of theory of abstract interpretation, with many examples of abstractions
- A short overview of a few applications and on-going work on software verification

For a rather complete basic introduction to abstract interpretation and applications to cyber-physical systems, see:

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.

Fundamental motivations

Scientific research

- in Mathematics/Physics:

  works towards unification and synthesis

  it is science of structure and change aiming at universal principles

- in Computer science

  works towards dispersion and parcelization

  it is a collection of local techniques for computational structures aiming at specific applications

An exponential process, will stop!

Example: reasoning on computational structures

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

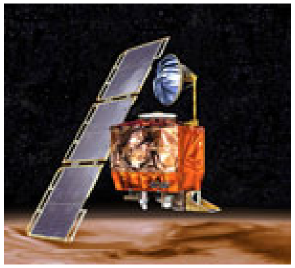
Example: reasoning on computational structures

Abstract interpretation

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

Applied motivations

All computer scientists have experienced bugs



- Checking the presence of bugs is great
- Proving their absence is even better!

A Touch of Abstract Interpretation Theory

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 És 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.

Semantics

Semantics

- Formal system: syntax to describe computations (e.g. programming language = set of programs):

$$\mathrm{P} \in \mathbb{L}$$

- Semantics: formal model of computations (e.g. set of execution traces)
- Semantic domain (set of semantics):

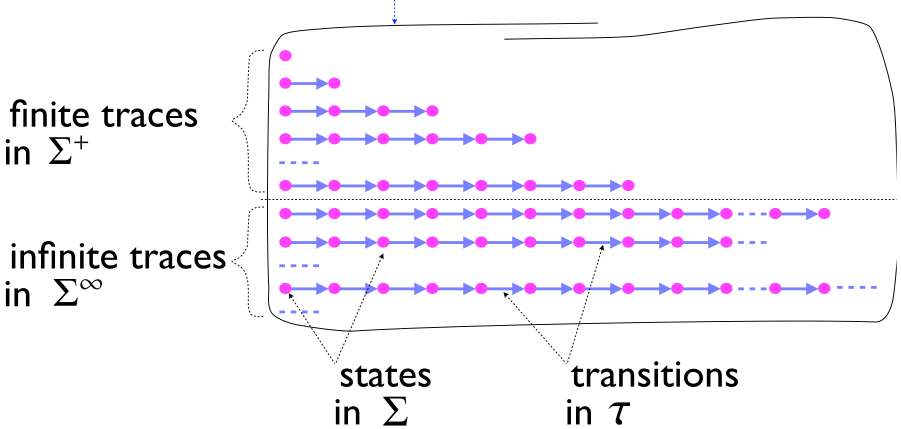
$$\mathcal{D}$$

- Formal system semantics (maps syntactic system descriptions to their semantics)

$$\mathcal{S} \in \mathbb{L} \to \mathcal{D}$$

Example: partial trace semantics

- Program $\mathrm{P} \longmapsto \tau^{\ddot{+}\infty}[\![\mathrm{P}]\!] \in \wp(\Sigma^{+\infty})$



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

Example: partial trace semantics

- Partial trace semantics $\tau^{\ddot{+}\infty}[\![\mathrm{P}]\!]$ generated by the small-step operational semantics $\langle \Sigma, \tau \rangle$ of a program P :

$$\tau^{\ddot{n}}[\![\mathrm{P}]\!] \triangleq \left\{\sigma \in \Sigma^n \;\middle|\; \forall i \in [0, n-1) : \langle \sigma_i, \sigma_{i+1} \rangle \in \tau[\![\mathrm{P}]\!]\right\}, \quad n \geqslant 0$$

$$\tau^{\ddot{+}}[\![\mathrm{P}]\!] \triangleq \bigcup_{n>0} \tau^{\ddot{n}}[\![\mathrm{P}]\!],$$

$$\tau^{\infty}[\![\mathrm{P}]\!] \triangleq \left\{\sigma \in \Sigma^\infty \;\middle|\; \forall i \in \mathbb{N} : \langle \sigma_i, \sigma_{i+1} \rangle \in \tau[\![\mathrm{P}]\!]\right\}$$

$$\tau^{\ddot{+}\infty}[\![\mathrm{P}]\!] \triangleq \tau^{\ddot{+}}[\![\mathrm{P}]\!] \cup \tau^{\infty}[\![\mathrm{P}]\!]$$

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

Concrete properties

Concrete properties

- Program concrete property: set of possible semantics of the program
- Concrete property domain:

$$\mathcal{P} \triangleq \wp(\mathcal{D}) \qquad \langle \mathcal{P},\ \subseteq,\ \emptyset,\ \mathcal{D},\ \cup,\ \cap \rangle$$

more generally $\langle \mathcal{P},\ \leqslant,\ 0,\ 1,\ \vee,\ \wedge \rangle$ or $\langle \mathcal{P},\ \leqslant,\ 0,\ \vee \rangle$

- Collecting semantics: (maps programs to their strongest property)

$$C[\![\mathrm{P}]\!] \ \triangleq\ \{\, \mathcal{S}[\![\mathrm{P}]\!] \,\}$$

(it implies "$\subseteq$" all other 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
Patrick Cousot, Radhia Cousot: Systematic Design of Program Analysis Frameworks. POPL 1979: 269-282

Example: concrete properties of trace semantics

- A trace in $\Sigma^{+\infty}$ is a finite or infinite sequence of states in Σ
- A trace semantics in $\wp(\Sigma^{+\infty})$ is a set of traces
- A trace semantics property in $\wp(\wp(\Sigma^{+\infty}))$ is a set of trace semantics
- The collecting semantics of a program P with trace semantics $\Theta^{+\infty}[\![\mathrm{P}]\!] \in \wp(\Sigma^{+\infty})$ is the *strongest trace semantics property*

$$\{\Theta^{+\infty}[\![\mathrm{P}]\!]\} \in \wp(\wp(\Sigma^{+\infty}))$$

Abstract properties

Abstract properties

- Abstract property: encodes a concrete property (e.g. a logical formula, a geometric object, etc)
- Abstract property domain:
  - a set of abstract properties
  - encodes *selected* concrete properties *of interest*

$$\langle \mathcal{A},\ \sqsubseteq,\ \bot,\ \top,\ \sqcup,\ \sqcap \rangle$$

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

Example of abstract properties: reachability

- A reachability property in $\wp(\Sigma)$ is a set of states in Σ that can be reached during execution from given initial states

Example of abstract properties: intervals

$[\ell, h]$: interval of values between ℓ and h (including -∞ and +∞)

$\perp$: empty set (false)

$$\mathcal{A} \triangleq \{[\ell, h] \mid \ell \in \mathbb{Z} \cup \{-\infty\} \wedge h \in \mathbb{Z} \cup \{+\infty\} \wedge \ell \leqslant h\} \cup \{\perp\}$$

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.

Abstraction

Abstraction

- Abstraction: maps concrete to abstract properties

$$\alpha \in \mathcal{P} \to \mathcal{A}$$

α is assumed to be *increasing* (so $\sqsubseteq$ is the abstraction of $\subseteq$).

- Abstract semantics: abstraction of the collecting semantics

$$\begin{aligned} \overline{\mathcal{S}} &\in \mathbb{L} \to \mathcal{A} \\ \overline{\mathcal{S}}[\![\mathrm{P}]\!] &\triangleq \alpha(\mathcal{C}[\![\mathrm{P}]\!]) = \alpha(\{\mathcal{S}[\![\mathrm{P}]\!]\}) \end{aligned}$$

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

Concretization

Concretization

- Concretization: maps abstract properties to concrete properties

$$\gamma \in \mathcal{A} \to \mathcal{P}$$

γ is assumed to be *increasing* (so $\subseteq$ is the concretization of $\sqsubseteq$)

- Abstract properties either describe exactly the concrete properties in $\gamma(\mathcal{A})$, or
- Abstract properties must approximate the concrete properties in $\mathcal{P} \setminus \gamma(\mathcal{A})$

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

Soundness

Soundness

- Definition: An abstract property $Q \in \mathcal{A}$ over-approximates a concrete property $P \in \mathcal{P}$ if and only if

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

- Definition: an abstraction is sound if and only if

$$\forall P \in \mathcal{P} : P \subseteq \gamma(\alpha(P))$$

- Under-approximation is dual(*)

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 És Sciences Mathématiques*, Université Joseph Fourier, Grenoble, France, 21 March 1978.

Best abstraction

Best abstraction

- Concrete properties: $\langle \mathcal{P},\ \leqslant \rangle$
- Abstract properties: $\langle \mathcal{A},\ \sqsubseteq \rangle$
- If any concrete property $P \in \mathcal{P}$ has a best abstraction $\alpha(P) \in \mathcal{A}$, then the correspondence is given by a *Galois connection* [1,2]

$$\langle \mathcal{P},\ \leqslant \rangle \xleftrightarrows[\alpha]{\gamma} \langle \mathcal{A},\ \sqsubseteq \rangle$$

i.e.

$$\forall P \in \mathcal{P} : \forall Q \in \mathcal{A} : \alpha(P) \sqsubseteq Q \Leftrightarrow P \leqslant \gamma(Q)$$

Sound abstraction $\Rightarrow$

Best abstraction $\Leftarrow$

[1] 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

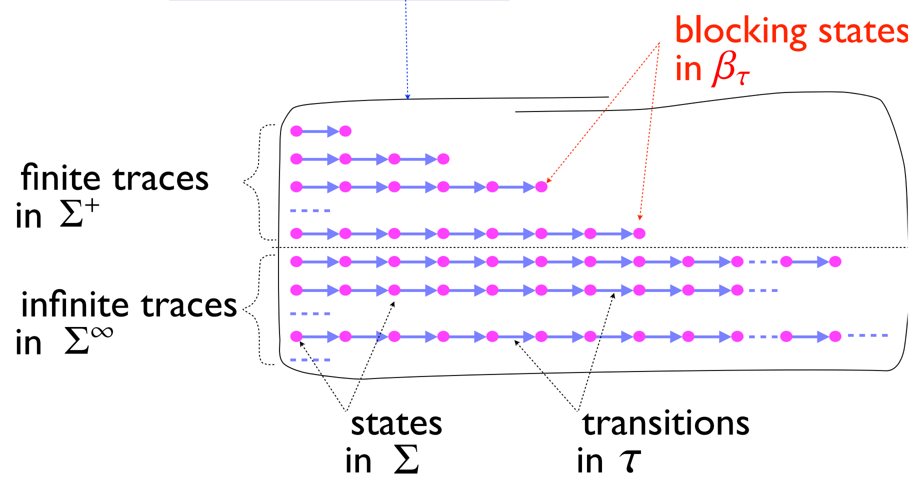
[2] Equivalently upper closures, principal ideals, complete join congruences, Moore families, etc, see [3]

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

Examples of abstraction/ concretization

Example I of abstraction: maximal trace semantics

- Program $\mathrm{P} \longmapsto \tau^{+\infty}[\![\mathrm{P}]\!] \in \wp(\Sigma^{+\infty})$



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)

Example I of abstraction: maximal trace semantics

- Blocking states of a transition system $\langle \Sigma,\ \tau \rangle$:

$$\beta_\tau[\![\mathrm{P}]\!] \triangleq \{s \in \Sigma[\![\mathrm{P}]\!] \mid \forall s' \in \Sigma[\![\mathrm{P}]\!] : \langle s,\ s' \rangle \notin \tau[\![\mathrm{P}]\!]\}$$

- Maximal trace abstraction (eliminates all traces that are not terminated):

$$\alpha_M(T) \triangleq \bigcup_{n \in \mathbb{N}} \Big\{\sigma \in T \cap \Sigma^n \;\Big|\; \sigma_{n-1} \in \beta_\tau[\![\mathrm{P}]\!]\Big\} \cup T^\infty$$

$$\langle \wp(\Sigma^{+\infty}),\ \subseteq \rangle \xleftrightarrows[\alpha_M]{\gamma_M} \langle \wp(\Sigma^{+\infty}),\ \subseteq \rangle$$

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)

Example II of abstraction: trace property

- Trace property abstraction:

$$\alpha_\Theta(P) \triangleq \bigcup P$$

$$\langle \wp(\wp(\Sigma^{+\infty})), \subseteq\rangle \underset{\alpha_\Theta}{\overset{\gamma_\Theta}{\leftrightarrows}} \langle \wp(\Sigma^{+\infty}), \subseteq\rangle$$

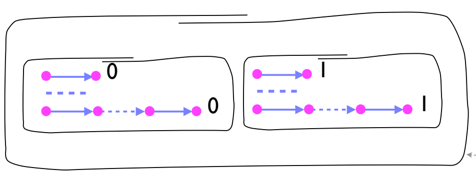
- Trace property abstraction of the collecting semantics:

$$\alpha_\Theta(\{\tau^{+\infty}[\![\mathrm{P}]\!]\}) = \tau^{+\infty}[\![\mathrm{P}]\!] \in \wp(\Sigma^{+\infty})$$

(common confusion between semantics and properties)

Loss of information in the trace property abstraction

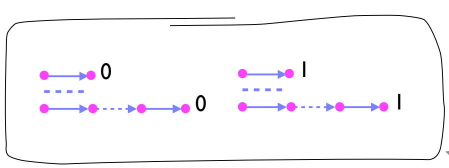
- "Always terminate with the same value, either 0 or 1"

$P =$ 

$P \in \wp(\wp(\Sigma^{+\infty}))$

always same result

- Trace property abstraction:

$\alpha_\Theta(P) =$ 

$\alpha_\Theta(P) \in \wp(\Sigma^{+\infty})$

results can be different

"Always terminate, either with 0 or 1"

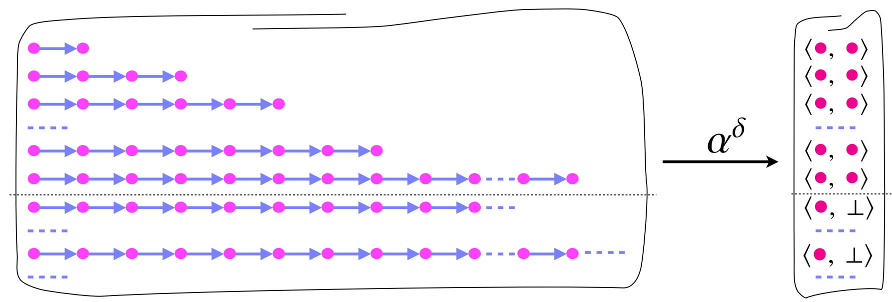
Example III of abstraction: relational abstraction

- Relational abstraction:

$$\begin{aligned}\alpha^\delta(T) \triangleq\ & \{\langle \sigma_0,\ \sigma_{n-1}\rangle \mid \sigma \in T \cap \Sigma^n\} \\ & \cup \{\langle \sigma_0,\ \bot\rangle \mid \sigma \in T \cap \Sigma^{*\infty}\}\end{aligned}$$

$$\langle \wp(\Sigma^{+\infty}),\ \subseteq\rangle \underset{\alpha^\delta}{\overset{\gamma^\delta}{\leftrightarrows}} \langle \wp((\Sigma\times\Sigma)\cup(\Sigma\times\{\bot\})),\ \subseteq\rangle$$

- Intuition:



Patrick Cousot, Radhia Cousot: Invited Talk: Higher Order Abstract Interpretation (and Application to Comportment Analysis Generalizing Strictness, Termination, Projection, and PER Analysis. ICCL 1994: 95-112
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)

Example IV of abstraction: safety trace property

- Prefix abstraction (program executions can be observed only for a finite time):

$$\mathsf{pf}(\sigma) \triangleq \{\sigma' \in \Sigma^{+\infty} \mid \exists \sigma'' \in \Sigma^{*\infty} : \sigma = \sigma'\sigma''\}$$
$$\mathsf{pf}(T) \triangleq \bigcup\{\mathsf{pf}(\sigma) \mid \sigma \in T\}\,.$$

- Limit abstraction (non-termination cannot be observed):

$$\mathsf{lm}(T) \triangleq T \cup \{\sigma \in \Sigma^\infty \mid \forall n \in \mathbb{N} : \sigma[0,n] \in T\}$$

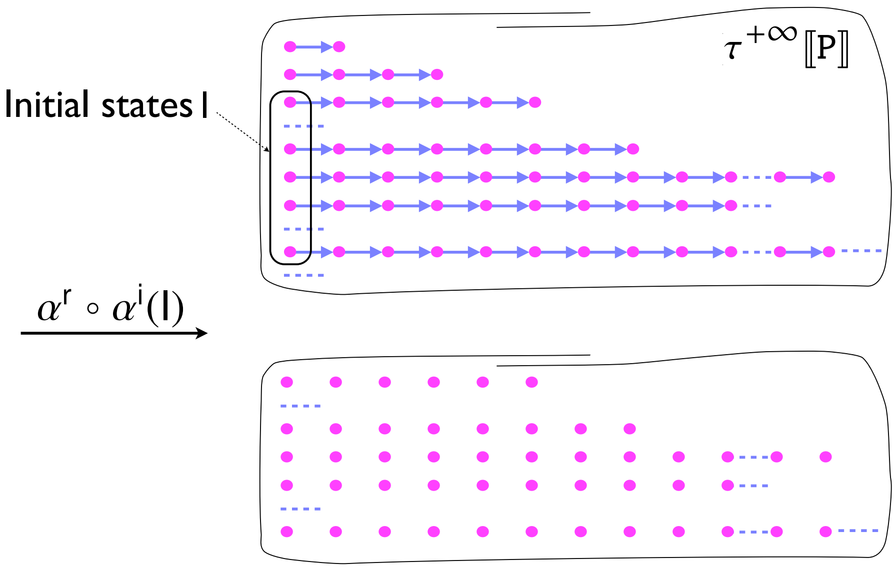
- Safety abstraction (finite observations of executions):

$$\mathsf{sf} \triangleq \mathsf{lm} \circ \mathsf{pf} = \mathsf{pf} \circ \mathsf{lm} \circ \mathsf{pf}$$

$$\langle \wp(\Sigma^{+\infty}),\ \subseteq\rangle \underset{\mathsf{sf}}{\overset{\mathbb{1}}{\leftrightarrows}} \langle \wp(\Sigma^{+\infty}),\ \subseteq\rangle$$

Patrick Cousot, Radhia Cousot: An abstract interpretation framework for termination. POPL 2012: 245-258

Example V of abstraction: reachability



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

Example V of abstraction: reachability

- Initialization abstraction:

$$\alpha^{\mathrm{i}}(\mathrm{I})T \triangleq \{\sigma \in T \mid \sigma_0 \in \mathrm{I}\}$$

$$\alpha^{\mathrm{i}} \in \wp(\Sigma) \rightarrow (\wp(\Sigma^{+\infty}) \rightarrow \wp(\Sigma^{+\infty}))$$

- Reachability abstraction:

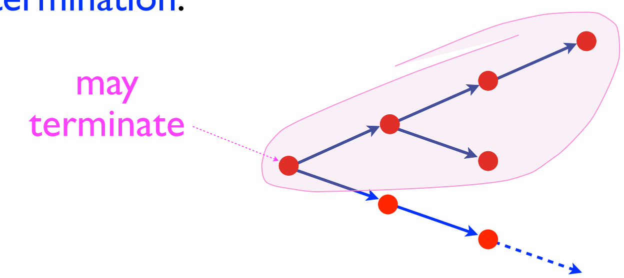
$$\alpha^{\mathrm{r}}(T) \triangleq \{\, s \mid \exists \sigma \in \Sigma^{*}, \sigma' \in \Sigma^{*\infty} : \sigma s \sigma' \in T \,\}$$

$$\langle \wp(\Sigma^{+\infty}),\ \subseteq \rangle \underset{\alpha^{\mathrm{r}}}{\overset{\gamma^{\mathrm{r}}}{\leftrightarrows}} \langle \wp(\Sigma),\ \subseteq \rangle$$

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

Example VI of abstraction: potential termination

- Potential termination:



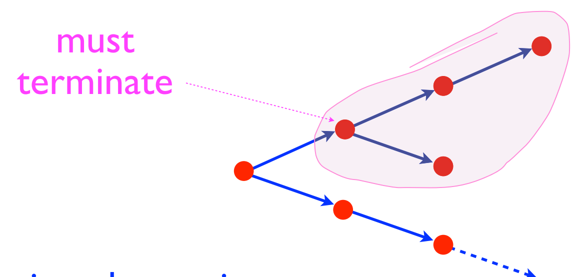
- Potential termination abstraction:

$$\alpha^{\mathrm{mt}}(T) \triangleq T \cap \Sigma^{+}$$

Patrick Cousot, Radhia Cousot: An abstract interpretation framework for termination. POPL 2012: 245-258

Example VII of abstraction: definite termination

- Definite termination:



- Definite termination abstraction:

$$\alpha^{\mathrm{Mt}}(T) \triangleq \{\sigma \in T^{+} \mid \mathrm{pf}(\sigma) \cap \mathrm{pf}(T^{\infty}) = \emptyset\}$$

Patrick Cousot, Radhia Cousot: An abstract interpretation framework for termination. POPL 2012: 245-258

Example VIII: elementwise abstraction

- Morphism

$$h \in \mathcal{P} \mapsto \mathcal{A}$$

- Abstraction

$$\alpha(X) \triangleq \{h(x) \mid x \in X\}$$

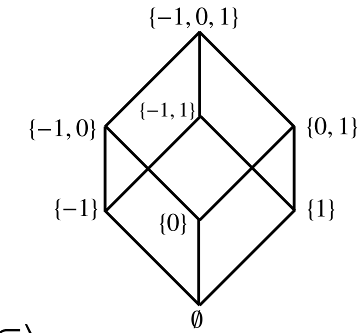
- Galois connection

$$\langle \wp(\mathcal{P}),\ \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \wp(\mathcal{A}),\ \subseteq\rangle$$

- Example: rule of signs

$$h : \mathbb{Z} \to \{-1, 0, 1\}$$

$$h(z) \triangleq z/|z|$$



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

Example IX: typing

- Eager lambda calculus:

$$\mathtt{x}, \mathtt{f}, \ldots \in \mathbb{X} \quad : \quad \text{program variables}$$

$$e \in \mathbb{E} \quad : \quad \text{program expressions}$$

$$e ::= \mathtt{x} \mid \boldsymbol{\lambda}\mathtt{x}\cdot e \mid e_1(e_2) \mid \boldsymbol{\mu}\mathtt{f}\cdot\boldsymbol{\lambda}\mathtt{x}\cdot e \mid \mathbf{1} \mid e_1 - e_2 \mid (e_1 \mathbin{\textbf{?}} e_2 \mathbin{\textbf{:}} e_3)$$

- Semantic domains:

$\mathbb{W} \triangleq \{\omega\}$	wrong
$\mathsf{z} \in \mathbb{Z}$	integers
$\mathsf{u}, \mathsf{f}, \varphi \in \mathbb{U} \cong \mathbb{W}_\perp \oplus \mathbb{Z}_\perp \oplus [\mathbb{U} \mapsto \mathbb{U}]_\perp$	values
$\mathsf{R} \in \mathbb{R} \triangleq \mathbb{X} \mapsto \mathbb{U}$	environments
$\phi \in \mathbb{S} \triangleq \mathbb{R} \mapsto \mathbb{U}$	semantic domain

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

Example IX: typing

- Denotational semantics: $\mathbf{S}[\![\ \bullet\ ]\!] \in \mathbb{E} \mapsto \mathbb{S}$

$$\mathbf{S}[\![\mathtt{x}]\!] \triangleq \Lambda\mathsf{R}\cdot \mathsf{R}(\mathtt{x})$$

$$\mathbf{S}[\![\boldsymbol{\lambda}\mathtt{x}\cdot e]\!] \triangleq \Lambda\mathsf{R}\cdot {\uparrow}\Big(\Lambda\mathsf{u}\cdot(\mathsf{u} = \perp \vee \mathsf{u} = \Omega\ ?\ \mathsf{u} \mid \mathbf{S}[\![e]\!]\mathsf{R}[\mathtt{x} \leftarrow \mathsf{u}])\Big) :: [\mathbb{U} \mapsto \mathbb{U}]_\perp$$

$$\mathbf{S}[\![e_1(e_2)]\!] \triangleq \Lambda\mathsf{R}\cdot(\mathbf{S}[\![e_1]\!]\mathsf{R} = \perp \vee \mathbf{S}[\![e_2]\!]\mathsf{R} = \perp\ ?\ \perp \mid \mathbf{S}[\![e_1]\!]\mathsf{R} = \mathsf{f} :: [\mathbb{U} \mapsto \mathbb{U}]_\perp\ ?\ {\downarrow}(\mathsf{f})\Big(\mathbf{S}[\![e_2]\!]\mathsf{R}\Big) \mid \Omega)$$

$$\mathbf{S}[\![\boldsymbol{\mu}\mathtt{f}\cdot\boldsymbol{\lambda}\mathtt{x}\cdot e]\!] \triangleq \Lambda\mathsf{R}\cdot\ \mathbf{lfp}^{\sqsubseteq}_{{\uparrow}(\Lambda\,\mathsf{u}\cdot\,\perp)::[\mathbb{U}\mapsto\mathbb{U}]_\perp}\ \Lambda\varphi\cdot \mathbf{S}[\![\boldsymbol{\lambda}\mathtt{x}\cdot e]\!]\mathsf{R}[\mathtt{f} \leftarrow \varphi]$$

$$\mathbf{S}[\![\mathbf{1}]\!] \triangleq \Lambda\mathsf{R}\cdot {\uparrow}(1) :: \mathbb{Z}_\perp$$

$$\mathbf{S}[\![e_1 - e_2]\!] \triangleq \Lambda\mathsf{R}\cdot(\mathbf{S}[\![e_1]\!]\mathsf{R} = \perp \vee \mathbf{S}[\![e_2]\!]\mathsf{R} = \perp\ ?\ \perp \mid \mathbf{S}[\![e_1]\!]\mathsf{R} = \mathsf{z}_1 :: \mathbb{Z}_\perp \wedge \mathbf{S}[\![e_2]\!]\mathsf{R} = \mathsf{z}_2 :: \mathbb{Z}_\perp\ ?\ {\uparrow}({\downarrow}(\mathsf{z}_1) - {\downarrow}(\mathsf{z}_2)) :: \mathbb{Z}_\perp \mid \Omega)$$

$$\mathbf{S}[\![(e_1 \mathbin{\textbf{?}} e_2 \mathbin{\textbf{:}} e_3)]\!] \triangleq \Lambda\mathsf{R}\cdot(\mathbf{S}[\![e_1]\!]\mathsf{R} = \perp\ ?\ \perp \mid \mathbf{S}[\![e_1]\!]\mathsf{R} = \mathsf{z} :: \mathbb{Z}_\perp\ ?\ ({\downarrow}(\mathsf{z}) = 0\ ?\ \mathbf{S}[\![e_2]\!]\mathsf{R} \mid \mathbf{S}[\![e_3]\!]\mathsf{R}) \mid \Omega)$$

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

Example IX: typing

- Church/Curry monotypes:

$m \in \mathbb{M}^{\mathrm{C}}, \quad m ::= \mathtt{int} \mid m_1 \to m_2$	monotype
$H \in \mathbb{H}^{\mathrm{C}} \triangleq \mathbb{X} \mapsto \mathbb{M}^{\mathrm{C}}$	type environment
$\theta \in \mathbb{I}^{\mathrm{C}} \triangleq \mathbb{H}^{\mathrm{C}} \times \mathbb{M}^{\mathrm{C}}$	typing
$T \in \mathbb{T}^{\mathrm{C}} \triangleq \wp(\mathbb{I}^{\mathrm{C}})$	program type

Example IX: typing

- **Properties**: $\mathbb{P} \triangleq \wp(\mathbb{S})$
- **Monotype concretization**:

$$
\begin{aligned}
\gamma_1^{\mathrm{C}} &\in \mathbb{M}^{\mathrm{C}} \mapsto \wp(\mathbb{U}) \\
\gamma_1^{\mathrm{C}}(\texttt{int}) &\triangleq \{\uparrow(\mathrm{z}) :: \mathbb{Z}_\perp \mid \mathrm{z} \in \mathbb{Z}\} \cup \{\perp\} \\
\gamma_1^{\mathrm{C}}(m_1 \texttt{->} m_2) &\triangleq \{\uparrow(\varphi) :: [\mathbb{U} \mapsto \mathbb{U}]_\perp \mid \varphi \in [\mathbb{U} \mapsto \mathbb{U}] \wedge \\
&\qquad \forall \mathrm{u} \in \gamma_1^{\mathrm{C}}(m_1) : \varphi(\mathrm{u}) \in \gamma_1^{\mathrm{C}}(m_2)\} \cup \{\perp\} \\
\gamma_2^{\mathrm{C}} &\in \mathbb{H}^{\mathrm{C}} \mapsto \wp(\mathbb{R}) \\
\gamma_2^{\mathrm{C}}(H) &\triangleq \{\mathsf{R} \in \mathbb{R} \mid \forall \mathtt{x} \in \mathbb{X} : \mathsf{R}(\mathtt{x}) \in \gamma_1^{\mathrm{C}}(H(\mathtt{x}))\} \\
\gamma_3^{\mathrm{C}} &\in \mathbb{I}^{\mathrm{C}} \mapsto \mathbb{P} \\
\gamma_3^{\mathrm{C}}(\langle H,\ m\rangle) &\triangleq \{\phi \in \mathbb{S} \mid \forall \mathsf{R} \in \gamma_2^{\mathrm{C}}(H) : \phi(\mathsf{R}) \in \gamma_1^{\mathrm{C}}(m)\} \\
\gamma^{\mathrm{C}} &\in \mathbb{T}^{\mathrm{C}} \mapsto \mathbb{P} \\
\gamma^{\mathrm{C}}(T) &\triangleq \bigcap_{\theta \in T} \gamma_3^{\mathrm{C}}(\theta), \qquad \gamma^{\mathrm{C}}(\emptyset) \triangleq \mathbb{S}
\end{aligned}
$$

Example IX: typing

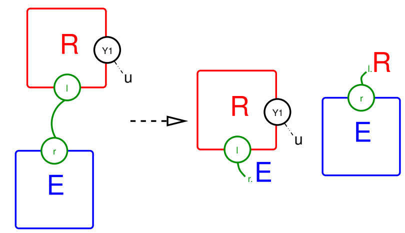
- **Galois connection**:

$$\gamma^{\mathrm{C}}\Big(\bigcup_{i \in \Delta} T_i\Big) = \bigcap_{i \in \Delta} \gamma^{\mathrm{C}}(T_i)$$

implies

$$\langle \mathbb{P},\ \subseteq\rangle \xleftarrow[\alpha^{\mathrm{C}}]{\gamma^{\mathrm{C}}} \langle \mathbb{T}^{\mathrm{C}},\ \supseteq\rangle$$

Example X: Protein–Protein interaction abstraction

- Let *Species* be the set of all chemical species ($\mathrm{C}, \mathrm{c}_1, \mathrm{c}'_1, \ldots, \mathrm{c}_k, \mathrm{c}'_k, \ldots \in$ *Species*)
- Let *Local_view* be the set of all local views
- Let $\alpha \in \wp(\textit{Species}) \rightarrow \wp(\textit{Local\_view})$ be the function that maps any set of complexes into the set of their local views.
- 

$\alpha(\{$R(Y1~u,l!1), E(r!1)$\})$

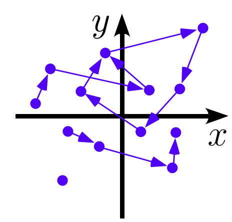
$= \{$R(Y1~u,l!r.E); E(r!l.R)$\}$

- The function α defines a Galois connexion: $\wp(\textit{Species}) \xleftarrow[\alpha]{\gamma} \wp(\textit{Local\_view})$
- (The function γ maps a set of local views into the set of complexes that can be built with these local views).

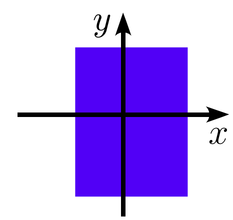
Jérôme Feret. Reachability Analysis of Biological Signalling Pathways by Abstract Interpretation. In Proceedings of the International Conference of Computational Methods in Sciences and Engineering (ICCMSE'2007), Corfu, Greece, 25--30 september, T.E. Simos(Ed.), 2007, American Institute of Physics conference proceedings 963.(2), pp 619--622.

Vincent Danos, Jérôme Feret, Walter Fontana, Jean Krivine: Abstract Interpretation of Cellular Signalling Networks. VMCAI 2008: 83-97

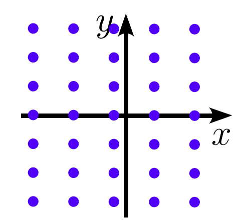
Example XI: numerical abstractions



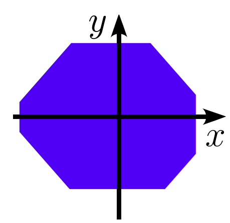
Collecting semantics: partial traces



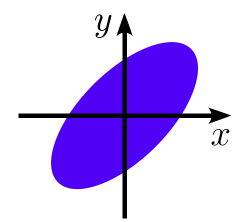
Intervals: $\mathtt{x} \in [a, b]$



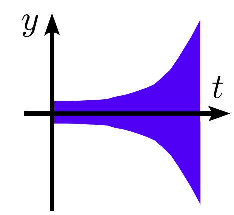
Simple congruences: $\mathtt{x} \equiv a[b]$



Octagons: $\pm\mathtt{x} \pm \mathtt{y} \leqslant a$



Ellipses: $\mathtt{x}^2 + b\mathtt{y}^2 - a\mathtt{x}\mathtt{y} \leqslant d$



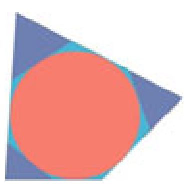
Exponentials: $-a^{bt} \leqslant \mathtt{y}(t) \leqslant a^{bt}$

In absence of best abstraction

In absence of best abstraction

- Best abstraction of a disk by a rectangular parallelogram

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



use only concretization or abstraction or widening (introduced in the following) [1]

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

Example XII of abstraction: polyhedra

- Abstract polyhedral properties:

$$\mathcal{A} \triangleq \{\langle A,\ B,\ n\rangle \mid A \in \mathbb{R}^n \times \mathbb{R}^n \wedge B \in \mathbb{R}^n \wedge n > 0\}$$

- Concretization:

$$\gamma_P(\langle A,\ B,\ n\rangle) \triangleq \{X \in \mathbb{R}^n \mid AX \leqslant B\}$$

Transformers and widenings have no more precise solution and make arbitrary choices (e.g. governed efficiency considerations)

Patrick Cousot. Méthodes itératives de construction et d'approximation de points fixes d'opérateurs monotones sur un treillis, analyse sémantique des programmes. *Thèse És 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

Transformer abstraction

Transformers

- Concrete transformer:

$$F \in \mathcal{P} \rightarrow \mathcal{P}$$

increasing (or continuous)

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

Transformer abstraction

- An abstract transformer $\overline{F} \in \mathcal{A} \rightarrow \mathcal{A}$ is
  - *Sound* iff

$$\forall P \in \mathcal{P} : \alpha \circ F(P) \sqsubseteq \overline{F} \circ \alpha(P)$$

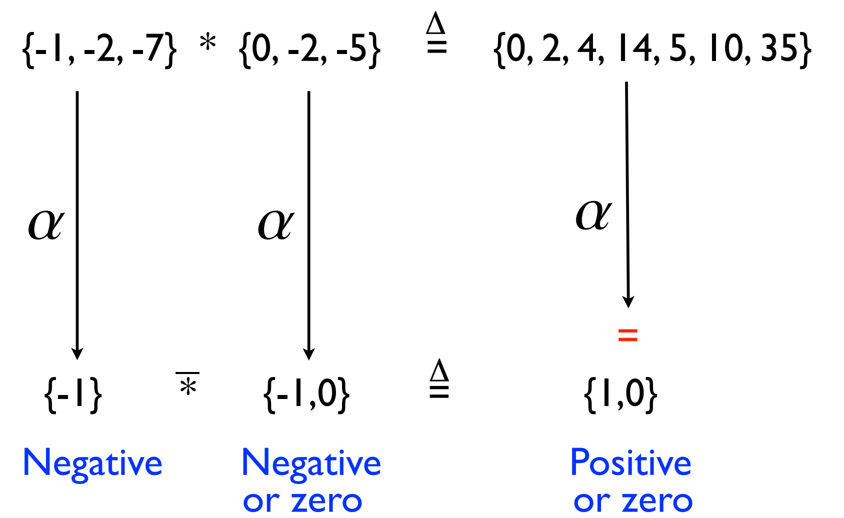
  - *Sound and complete* iff

$$\forall P \in \mathcal{P} : \alpha \circ F(P) = \overline{F} \circ \alpha(P)$$

- Example (rule of sign)
  - Addition: sound, incomplete
  - Multiplication: sound, complete

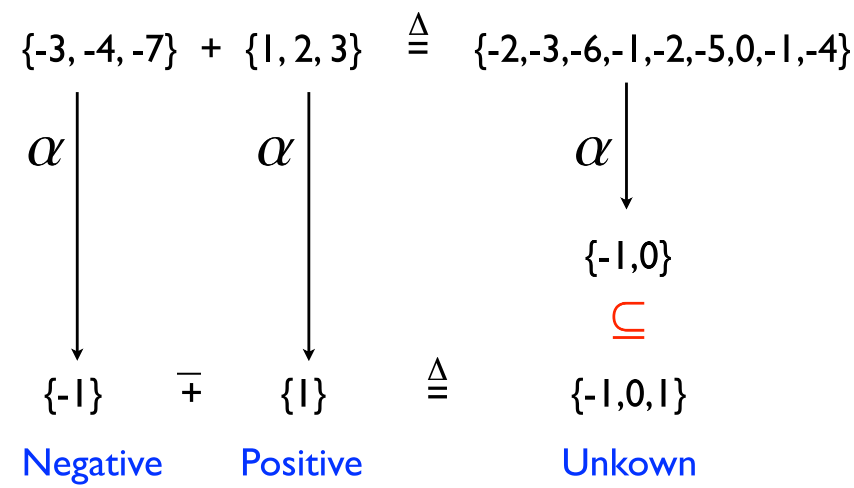
Patrick Cousot, Radhia Cousot: Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. POPL 1977: 238-252
Patrick Cousot, Radhia Cousot: Systematic Design of Program Analysis Frameworks. POPL 1979: 269-282

Example abstract transformer: rule of signs



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

Example abstract transformer: rule of signs



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

Fixpoints

Patrick Cousot & Radhia Cousot. Constructive versions of Tarski's fixed point theorems. In *Pacific Journal of Mathematics*, Vol. 82, No. 1, 1979, pp. 43—57.

Fixpoint

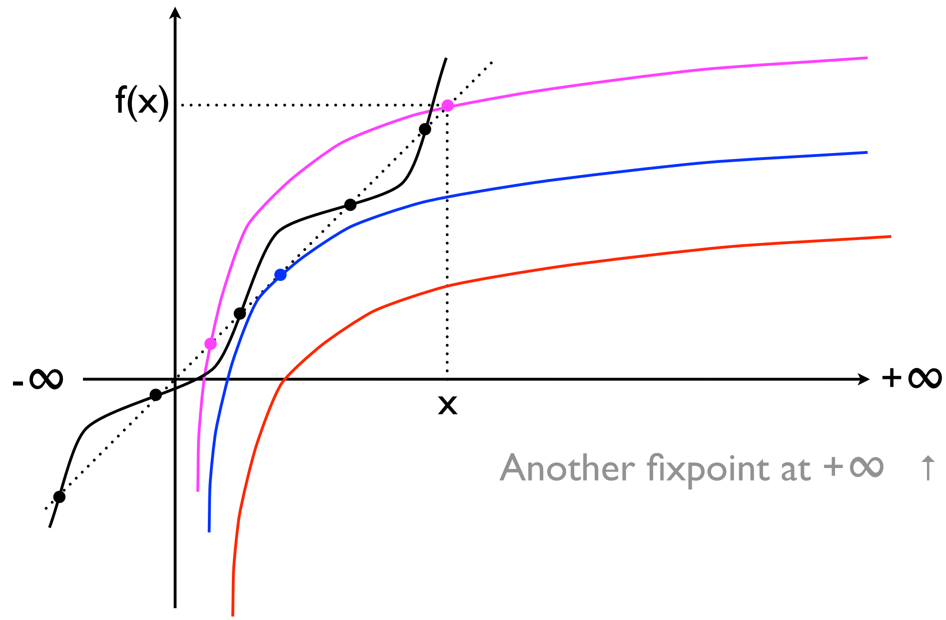
- Set $\mathcal{P}$
- Transformer $F \in \mathcal{P} \to \mathcal{P}$
- Fixpoint

$$x \in \mathcal{P} \text{ is a fixpoint of } F \iff F(x) = x$$

- Poset $\langle \mathcal{P}, \leqslant \rangle$
- Least fixpoint

$$x \in \mathcal{P} \text{ is the least fixpoint of } F \text{ (written } x = \mathsf{lfp}^{\leqslant} F) \iff F(x) = x \wedge \forall y \in \mathcal{P} : (F(y) = y) \Rightarrow (x \leqslant y)$$

Fixpoints of increasing functions (Tarski)



Program properties as fixpoints

- Program semantics and program properties can be formalized as least/greatest fixpoints of increasing transformers on complete lattices [(1)]
  - *Complete lattice / cpo of properties*

$$\langle \mathcal{P}, \leqslant, 0, 1, \vee, \wedge \rangle$$

  - *Properties* of program P

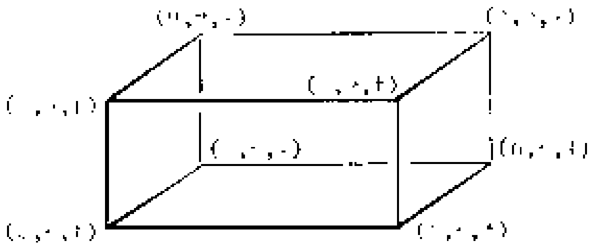
$$S[\![\mathrm{P}]\!] = \mathsf{lfp}^{\leqslant} F[\![\mathrm{P}]\!]$$

  - *Transformer* of program P

$$F[\![\mathrm{P}]\!] \in \mathcal{P} \to \mathcal{P}, \text{ increasing (or continuous)}$$

(1) 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

Fixpoints: inversion, converse and duality



- Forward (→) or backward (←) transformers
- Join (∪) or meet (∩) merge duality
- Least (↓) or greatest (↑) fixpoint duality

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 I: partial finite trace semantics

- $\langle \wp(\Sigma^{+}),\ \subseteq,\ \emptyset,\ \Sigma^{+},\ \cup,\ \cap \rangle$
- Forward transformer: $\overrightarrow{\phi}^{\ddot{+}}_{\tau}[\![\mathrm{P}]\!]T \triangleq \Sigma^1 \cup T \mathbin{;} \tau[\![\mathrm{P}]\!]$
- Backward transformer: $\overleftarrow{\phi}^{\ddot{+}}_{\tau}[\![\mathrm{P}]\!]T \triangleq \Sigma^1 \cup \tau[\![\mathrm{P}]\!] \mathbin{;} T$
- Fixpoint finite partial traces:

$$\tau^{\ddot{+}}[\![\mathrm{P}]\!] = \mathsf{lfp}^{\subseteq}_{\emptyset}\ \overleftarrow{\phi}^{\ddot{+}}_{\tau}[\![\mathrm{P}]\!] = \mathsf{lfp}^{\subseteq}_{\emptyset}\ \overrightarrow{\phi}^{\ddot{+}}_{\tau}[\![\mathrm{P}]\!]$$

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

Example II: infinite traces

- $\langle \wp(\Sigma^{\infty}),\ \subseteq,\ \emptyset,\ \Sigma^{\infty},\ \cup,\ \cap \rangle$
- Backward transformer: $\overleftarrow{\phi}^{\infty}_{\tau}[\![\mathrm{P}]\!]T \triangleq \tau[\![\mathrm{P}]\!] \mathbin{;} T$
- Fixpoint infinite traces:

$$\tau^{\infty}[\![\mathrm{P}]\!] = \mathsf{gfp}^{\subseteq}_{\Sigma^{\infty}}\ \overleftarrow{\phi}^{\infty}_{\tau}[\![\mathrm{P}]\!]$$

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)

Example III: partial finite and infinite traces (a)

- $\langle \wp(\Sigma^{+\infty}),\ \subseteq,\ \emptyset,\ \Sigma^{+\infty},\ \cup,\ \cap \rangle$
- Fixpoint partial finite and infinite traces semantics:

$$\tau^{\ddot{+}\infty}[\![\mathrm{P}]\!] = \mathsf{lfp}^{\subseteq}_{\emptyset}\ \overleftarrow{\phi}^{\ddot{+}}_{\tau}[\![\mathrm{P}]\!] \cup \mathsf{gfp}^{\subseteq}_{\Sigma^{\infty}}\ \overleftarrow{\phi}^{\infty}_{\tau}[\![\mathrm{P}]\!]$$

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)

Example III: partial finite and infinite traces (b)

- Computational order:

$$T^{+} \triangleq T \cap \Sigma^{+}$$
$$T^{\infty} \triangleq T \cap \Sigma^{\infty}$$
$$(T_1 \sqsubseteq T_2) \triangleq (T_1^{+} \subseteq T_2^{+}) \wedge (T_1^{\infty} \supseteq T_2^{\infty})$$
$$\langle \wp(\Sigma^{+\infty}),\ \sqsubseteq,\ \Sigma^{\infty},\ \Sigma^{+},\ \sqcup,\ \sqcap \rangle$$

- Transformer:

$$\overleftarrow{\phi}_{\tau}^{\ddot{+}\infty}[\![\mathrm{P}]\!]T \triangleq \Sigma^{1} \sqcup \tau[\![\mathrm{P}]\!] \,;\, T$$

- Fixpoint partial finite and infinite traces semantics:

$$\tau^{\ddot{+}\infty}[\![\mathrm{P}]\!] = \mathrm{lfp}^{\subseteq}_{\emptyset}\ \overleftarrow{\phi}_{\tau}^{\ddot{+}}[\![\mathrm{P}]\!] \cup \mathrm{gfp}^{\subseteq}_{\Sigma^{\infty}}\ \overleftarrow{\phi}_{\tau}^{\infty}[\![\mathrm{P}]\!] = \mathrm{lfp}^{\sqsubseteq}_{\Sigma^{\infty}}\ \overleftarrow{\phi}_{\tau}^{\ddot{+}\infty}[\![\mathrm{P}]\!]$$

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)

Example: reachable states

- *Transition system* (set of states Σ, initial states $\mathcal{I} \subseteq \Sigma$, transition relation τ)

$$\langle \Sigma,\ \mathcal{I},\ \tau \rangle$$

- *Right-image* of a set of states by transitions

$$\mathrm{post}[\tau]X \triangleq \{s' \mid \exists s \in X : \tau(s, s')\}$$

- *Reachable states* from initial states $\mathcal{I}$

$$\mathrm{post}[\tau^{\star}]\mathcal{I} = \mathrm{lfp}^{\subseteq}\ \lambda X \bullet \mathcal{I} \cup \mathrm{post}[\tau]X$$

Patrick Cousot. Méthodes itératives de construction et d'approximation de points fixes d'opérateurs monotones sur un treillis, analyse sémantique des programmes. *Thèse És 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.

Proof methods

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

Patrick Cousot and Radhia Cousot. Reasoning about program invariance proof methods. *Research Report CRIN-80-P050*, Institut National Polytechnique de Lorraine, Nancy, France, July 1980, 22p.

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.

Patrick Cousot & Radhia Cousot. Induction principles for proving invariance properties of programs. In D. Néel, editor, *Tools & Notions for Program Construction: an Advanced Course*, pages 75—119. Cambridge University Press, Cambridge, UK, August 1982.

Patrick Cousot. A Hoare-style axiomatization of Burstall's intermittent assertion method for non-deterministic programs *Research report LRIM-83-04*, University of Metz, September 1983.

Patrick Cousot and Radhia Cousot. "À la Burstall" induction principles for proving inevitability properties of programs. *Research Report LRIM-83-08*, University of Metz, November 1983.

Patrick Cousot & Radhia Cousot. Principe des Méthodes de Preuve de Propriétés d'Invariance et de Fatalité des Programmes Parallèles. *(Principle of invariance and inevitability proof methods of concurrent programs.)* In « *Parallélisme, communication et synchronisation* », J.-P. Verjus et G. Roucairol (Eds.), Éditions du CNRS, Paris, pp. 129—149, 1985.

Patrick Cousot & Radhia Cousot. "À la Floyd" induction principles for proving inevitability properties of programs. In «*Algebraic methods in semantics*», M. Nivat & J. Reynolds (Eds.), Cambridge University Press, Cambridge, UK, pp. 277—312, December 1985.

Patrick Cousot & Radhia Cousot. Sometime = Always + Recursion ≡ Always, On the Equivalence of the Intermittent and Invariant Assertions Methods for Proving Inevitability Properties of Programs. *Acta Informatica 24, 1—31 (1987).*

Patrick Cousot & Radhia Cousot. A language independent proof of the soundness and completeness of generalized Hoare logic. *Information and computation 80(2):165—191 (1989).*

Patrick Cousot. Methods and Logics for Proving Programs. In J. van Leeuwen, editor, *Formal Models and Semantics,* volume B of *Handbook of Theoretical Computer Science,* chapter 15, pages 843—993. Elsevier Science Publishers B.V. , 1990.

Radhia Cousot. Fondements des méthodes de preuve d'invariance et de fatalité de programmes parallèles. Thèse ès Sciences Mathématiques, Institut national polytechnique de Lorraine, Nancy, France, 15 November 1985.

Proof methods

- *Proof methods* directly follow from the fixpoint definition

$$S[\![\mathrm{P}]\!] \leqslant P$$
$$\Leftrightarrow \mathrm{lfp}^{\leqslant} F[\![\mathrm{P}]\!] \leqslant P$$
$$\Leftrightarrow \exists I : F[\![\mathrm{P}]\!](I) \leqslant I \wedge I \leqslant P$$

(proof by Tarski's fixpoint theorem for increasing transformers on complete lattice or Pataria for cpos)

$$\mathrm{lfp}^{\leqslant} F = \bigwedge\{x \mid F(x) \leqslant x\}$$

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

Example: Turing/Floyd Invariance Proof

- Bad states:

$$\mathcal{B} \subseteq \Sigma$$

- Prove that no bad state is reachable:

$$\mathsf{post}[\tau^\star]\mathcal{I} \subseteq \neg\mathcal{B}$$

ie $\quad \mathsf{lfp}^{\subseteq}\, \lambda X \bullet \mathcal{I} \cup \mathsf{post}[\tau]X \subseteq \neg\mathcal{B}$

- Turing/Floyd proof method:

$$\exists I \in \wp(\Sigma) : \mathcal{I} \subseteq I \wedge \mathsf{post}[\tau]I \subseteq I \wedge I \subseteq \neg\mathcal{B}$$

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

Fixpoint abstraction

Fixpoint abstraction

- For an increasing and sound abstract transformer, we have a *fixpoint approximation*

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

- For an increasing, sound, and complete abstract transformer, we have an *exact fixpoint abstraction*

$$\alpha(\mathsf{lfp}^{\leqslant} F) = \mathsf{lfp}^{\sqsubseteq} \overline{F}$$

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

Example XIII: trace to reachability abstraction

- Transition system: $\langle \Sigma[\![\mathrm{P}]\!],\ \tau[\![\mathrm{P}]\!] \rangle$
- Fixpoint concrete partial trace semantics:

$\tau^{\ddot{+}}[\![\mathrm{P}]\!] = \mathsf{lfp}^{\subseteq}_{\emptyset}\, \overrightarrow{\phi}^{\ddot{+}}_{\tau}[\![\mathrm{P}]\!]$ with $\overrightarrow{\phi}^{\ddot{+}}_{\tau}[\![\mathrm{P}]\!]T \triangleq \Sigma^1 \cup T \mathbin{;} \tau[\![\mathrm{P}]\!]$

- Reachability abstraction from initial states I:

$$\langle \wp(\Sigma^{+\infty}),\ \subseteq \rangle \xrightleftharpoons[\alpha^{\mathsf{r}} \circ \alpha^{\mathsf{i}}(\mathsf{I})]{\gamma^{\mathsf{i}}(\mathsf{I}) \circ \gamma^{\mathsf{r}}} \langle \wp(\Sigma),\ \subseteq \rangle$$

- Sound and complete abstract transformer

$$\alpha^{\mathsf{r}} \circ \alpha^{\mathsf{i}}(\mathsf{I}) \circ \overrightarrow{\phi}^{\ddot{+}}_{\tau}[\![\mathrm{P}]\!] = \lambda X \bullet \mathsf{I} \cup \mathsf{post}[\tau[\![\mathrm{P}]\!]] \circ \alpha^{\mathsf{r}} \circ \alpha^{\mathsf{i}}$$

- Fixpoint reachability:

$$\begin{aligned} \alpha^{\mathsf{r}} \circ \alpha^{\mathsf{i}}(\mathsf{I})(\tau^{\ddot{+}}[\![\mathrm{P}]\!]) &= \alpha^{\mathsf{r}} \circ \alpha^{\mathsf{i}}(\mathsf{I})\left(\mathsf{lfp}^{\subseteq}_{\emptyset}\, \overrightarrow{\phi}^{\ddot{+}}_{\tau}[\![\mathrm{P}]\!]\right) \\ &= \mathsf{lfp}^{\subseteq}_{\emptyset}\ \lambda X \bullet \mathsf{I} \cup \mathsf{post}[\tau[\![\mathrm{P}]\!]]X \end{aligned}$$

Fixpoint iteration (*) and convergence acceleration (**)

(*) In absence of direct solution (e.g. by elimination)
(**) In absence of finite convergence (e.g. ascending chain condition)

Iterative fixpoint computation

- Fixpoint of increasing transformers on cpos can be computed iteratively as limits of (transfinite) iterates

$$F^0 \triangleq \bot$$
$$F^{\beta+1} \triangleq F(F^\beta), \quad \beta+1 \text{ successor ordinal}$$
$$F^\lambda \triangleq \bigsqcup_{\beta<\lambda} F^\beta, \quad \lambda \text{ limit ordinal}$$

Ultimately stationary at rank ϵ

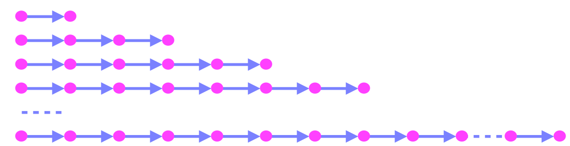
Converges to $F^\epsilon = \mathsf{lfp}^{\sqsubseteq} F$

- $\epsilon = \omega$ when F is continuous
- Finite iterates when F operates on a cpo satisfying the ascending chain condition

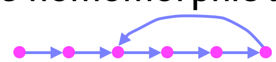
Patrick Cousot & Radhia Cousot. Constructive versions of Tarski's fixed point theorems. In *Pacific Journal of Mathematics*, Vol. 82, No. 1, 1979, pp. 43—57.

Expressiveness of finite abstractions is weak (*)

- Finite state abstraction is *impossible* for termination and *unsound* for non-termination of *unbounded programs*
  - Unbounded executions:



  - Finite homomorphic abstraction:



  - Termination: impossible (lasso)
  - Non-termination (lasso): unsound

(*) Excluding trivial solutions, see: Patrick Cousot: Partial Completeness of Abstract Fixpoint Checking. SARA 2000: 1-25

Widening

- Definition (widening $\nabla \in \mathcal{A} \times \mathcal{A} \rightarrow \mathcal{A}$)
  - $\langle \mathcal{A}, \sqsubseteq \rangle$ poset
  - Over-approximation

$$\forall x, y \in \mathcal{A} : x \sqsubseteq x \nabla y \wedge y \sqsubseteq x \nabla y$$

  - Termination

Given any sequence $\langle x^n,\ n \in \mathbb{N}\rangle$, the widened sequence $\langle y^n,\ n \in \mathbb{N}\rangle$

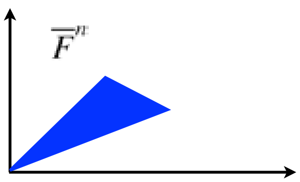
$$y^0 \triangleq x^0, \ldots, y^{n+1} \triangleq y^n \nabla x^n, \ldots$$

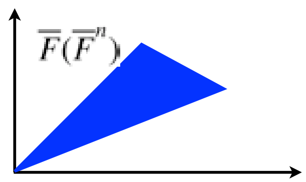
converges to a limit y^ℓ (such that $\forall m \geqslant \ell : y^m = y^\ell$)

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

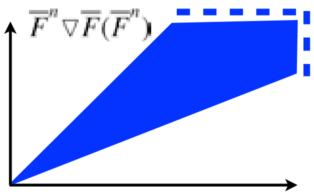
Example: (simple) widening for polyhedra

- 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 És 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

- *Iterates with widening* for transformer $\overline{F} \in \mathcal{A} \longrightarrow \mathcal{A}$

$$\begin{aligned} \overline{F}^0 &\triangleq \bot \\ \overline{F}^{n+1} &\triangleq \overline{F}^n && \text{when } \overline{F}(\overline{F}^n) \sqsubseteq \overline{F}^n \\ \overline{F}^{n+1} &\triangleq \overline{F}^n \triangledown \overline{F}(\overline{F}^n) && \text{otherwise} \end{aligned}$$

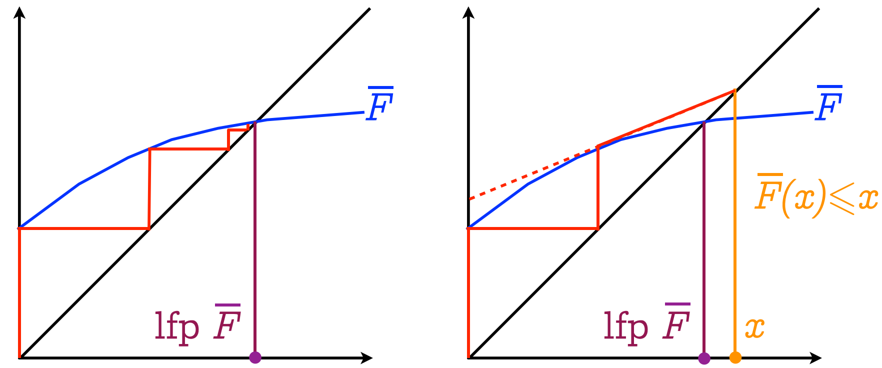
- The *widening speeds up convergence* (at the cost of imprecision)

Theorem (*Limit of iterates with widening*) The iterates of $\overline{F}$ with widening $\triangledown$ from $\bot$ on a poset $\langle \mathcal{A}, \sqsubseteq, \bot \rangle$ converge to a limit $\overline{F}^{\ell}$ such that $\overline{F}(\overline{F}^{\ell}) \sqsubseteq \overline{F}^{\ell}$ (and so $\mathsf{lfp}^{\sqsubseteq} \overline{F} \sqsubseteq \overline{F}^{\ell}$ when $\overline{F}$ is increasing).

- Can be improved by a *narrowing*.

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

Convergence acceleration with widening



Infinite iteration

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

Reduced product

- The reduced product combines abstractions by performing their conjunction in the abstract

$$\langle \mathcal{P}, \leqslant \rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle \mathcal{A}_1, \sqsubseteq_1 \rangle$$

$$\langle \mathcal{P}, \leqslant \rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle \mathcal{A}_2, \sqsubseteq_2 \rangle$$

$$\mathcal{A}_1 \otimes \mathcal{A}_2 \triangleq \{\langle \alpha_1(\gamma_1(P_1) \wedge \gamma_2(P_2)), \alpha_2(\gamma_1(P_1) \wedge \gamma_2(P_2)) \rangle \mid P_1 \in \mathcal{A}_1 \wedge P_2 \in \mathcal{A}_2\}$$

$$\langle \mathcal{P}, \leqslant \rangle \underset{\alpha_1 \times \alpha_2}{\overset{\gamma_1 \times \gamma_2}{\leftrightarrows}} \langle \mathcal{A}_1 \otimes \mathcal{A}_2, \sqsubseteq_1 \times \sqsubseteq_2 \rangle$$

- Example: (positive or zero) $\otimes$ odd = <positive,odd>

Patrick Cousot, Radhia Cousot: Systematic Design of Program Analysis Frameworks. POPL 1979: 269-282
Patrick Cousot, Radhia Cousot, Laurent Mauborgne: The Reduced Product of Abstract Domains and the Combination of Decision Procedures. FOSSACS 2011: 456-472

Undecidability and complexity

Fighting undecidability and complexity in automatic program verification

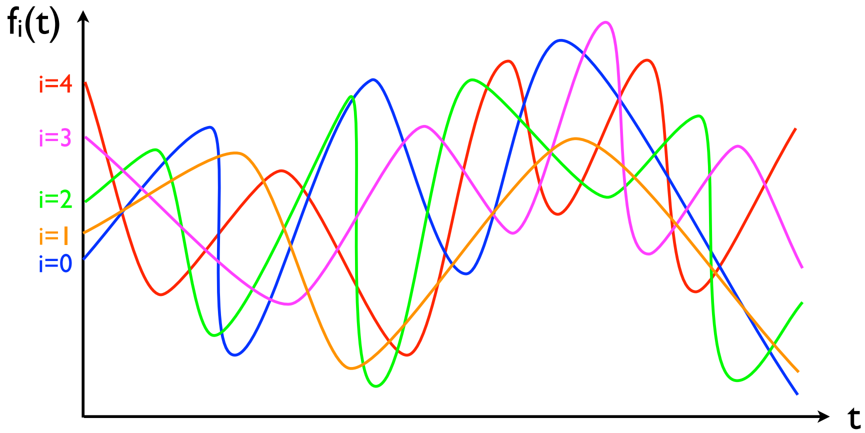
- Any *automatic* semantic program verification method will definitely fail on infinitely many programs (Gödel)
- Solutions:
  - Ask for human help (theorem-prover/proof assistant based *deductive methods*) → high labor cost
  - Consider finite/decidable systems (*model-checking*) → combinatorial explosion
  - Do sound approximations or complete abstractions (*abstract interpretation*) → *false alarms*

What to do about false alarms? abstraction refinement

What to do about false alarms? (I) Automatic refinement

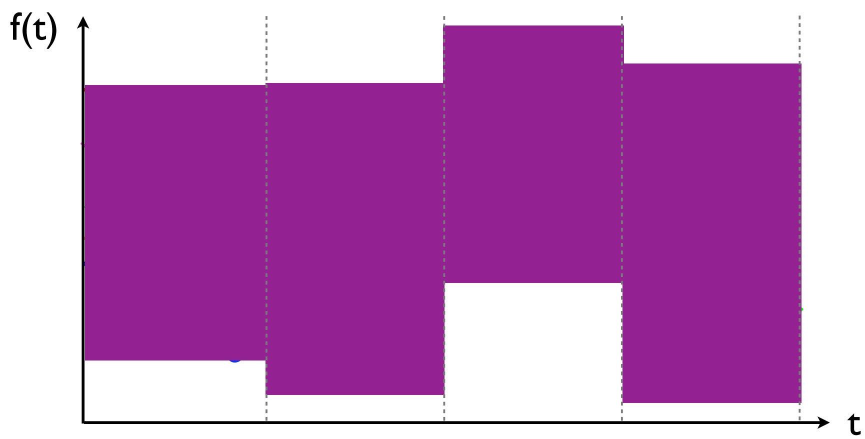
- Inefficient and may not terminate (Gödel)
- Refinement needs intelligence

Set of functions

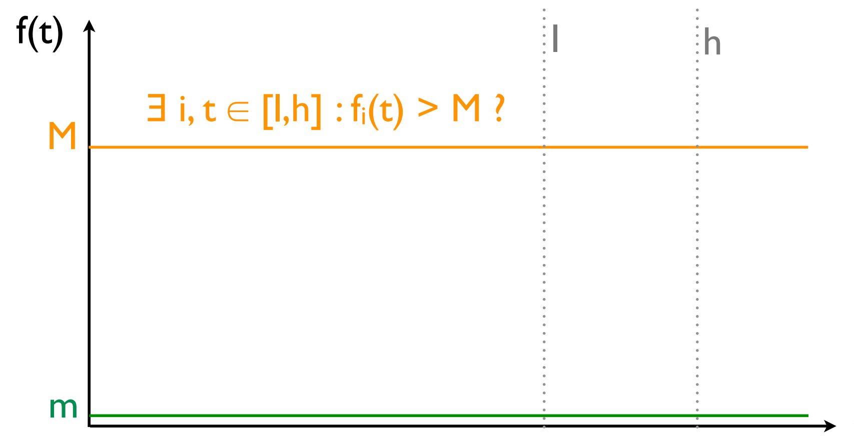


How to approximate $\{ f_1, f_2, f_3, f_4 \}$?

Set of functions abstraction



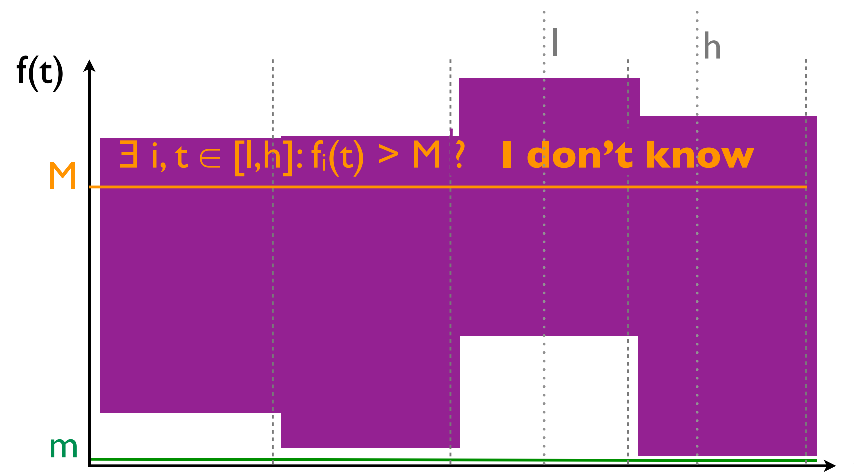
Concrete questions on the f_i



$\exists$ i, t $\in$ [l, h]: $f_i(t)$ < m ?

Min/max questions on the f_i

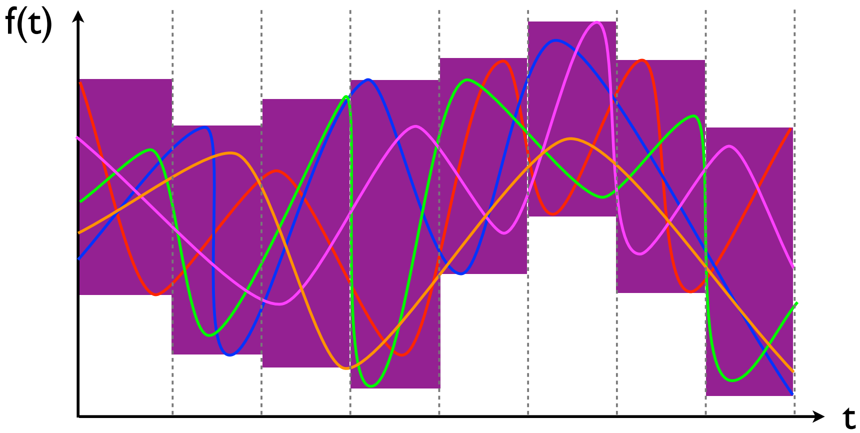
Concrete questions on the f_i answered in the abstract



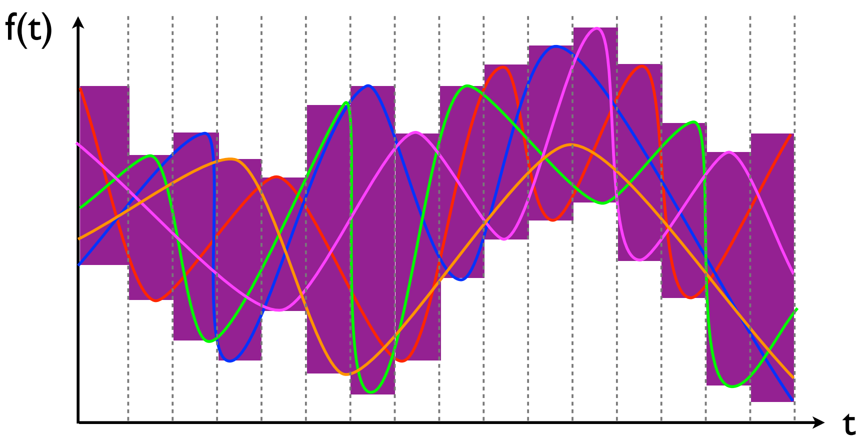
$\exists$ i, t $\in$ [l,h] : $f_i(t)$ < m ? **No**

Min/max questions on the f_i

A more precise/refined abstraction

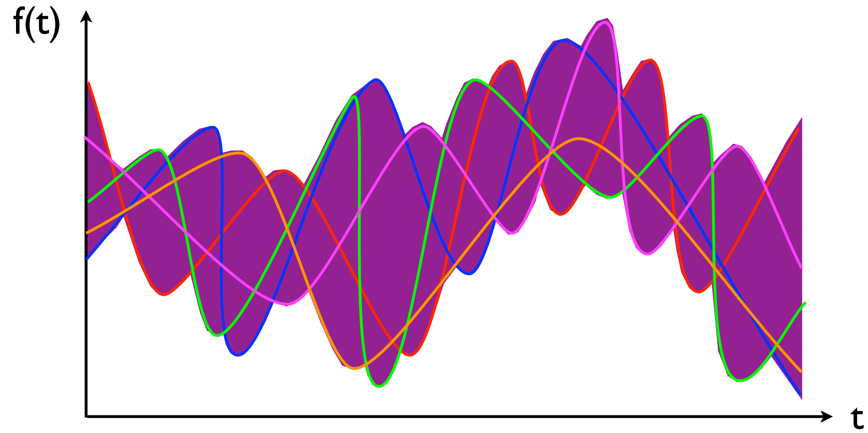


An even more precise/refined abstraction



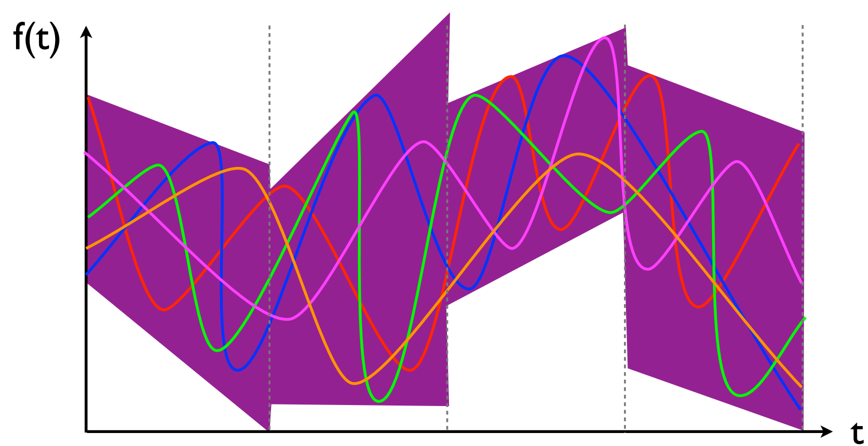
Note: this is already much more elaborate than CEGAR that goes counter-example by counter-example!

Intelligent passing to the limit



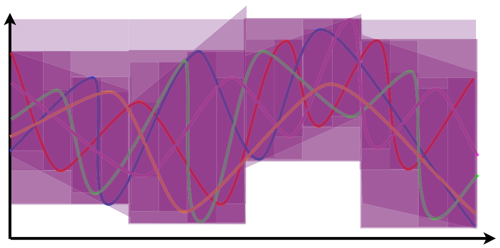
Sound and *complete* abstraction for min/max questions on the f_i

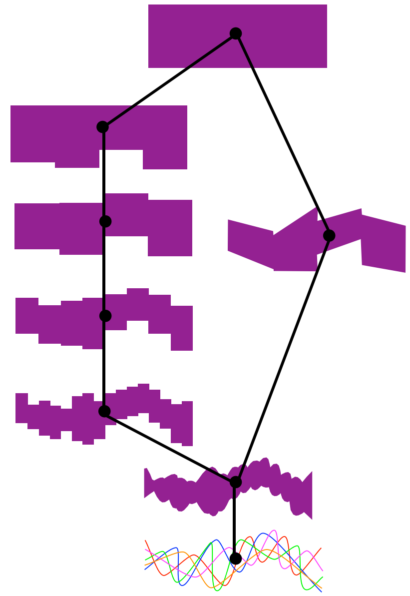
A non-comparable abstraction



Sound and *incomplete* abstraction for min/max questions on the f_i

The hierarchy of abstractions



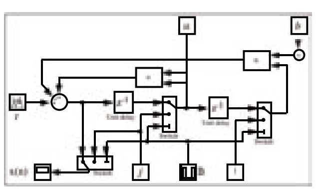


Patrick Cousot, Radhia Cousot: Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. POPL 1977: 238-252

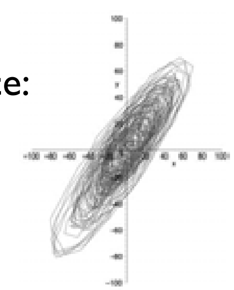
(I) Automatic refinement: Astrée example

- Filter invariant abstraction:

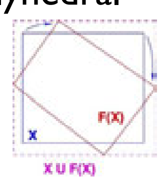
2nd order filter:



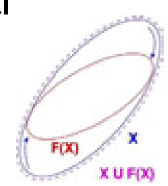
Execution trace:



Unstable polyhedral abstraction:



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.

What to do about false alarms? (II) Domain specific refinement

- Adapt the abstraction to the *programming paradigms* typically used in given *domain-specific applications*
- e.g. Astrée for *synchronous control/command programs*: no recursion, no dynamic memory allocation, maximum execution time, filters, integrators, quaternions, etc.

So, what is Abstract Interpretation

A narrow view ...

- Define the syntax of the system descriptions
- Define the semantics of the system descriptions
- Define the collecting semantics (strongest property of interest)
- Preferably express the collecting semantics in fixpoint form
- Define abstractions of properties
- Infer abstractions of transformers
- Infer abstractions of fixpoints to get abstract semantics
- Iterate to compute fixpoints with convergence acceleration (widening/narrowing)
- Combine abstractions (e.g. reduced product) to refine

Example XIV: grammar abstraction

- Meta-syntax of grammars
- Semantics of grammars (by induction on the meta-syntax): the language generated by the grammar
- Fixpoint semantics: Chomsky-Schützenberger th.

$$\mathcal{S}[\![\texttt{X ::= Xa | b}]\!] = \mathrm{lfp}^{\subseteq}\,\lambda X \bullet X \cdot \{\mathrm{a}\} \cup \{\mathrm{b}\}$$

- Example of abstraction: FIRST

$$\alpha_{\mathrm{FIRST}}(X) \triangleq \{\mathrm{a} \mid \exists \sigma : \mathrm{a}\sigma \in X\}$$

- Fixpoint abstraction: FIRST classical algorithm (expressed as a fixpoint)

$$\begin{aligned}\mathcal{F}[\![\texttt{X ::= Xa | b}]\!] &\triangleq \alpha_{\mathrm{FIRST}}(\mathcal{S}[\![\texttt{X ::= Xa | b}]\!]) \\ &= \mathrm{lfp}^{\subseteq}\,\lambda X \bullet X \cup \{\mathrm{a} \mid \varepsilon \in X\} \cup \{\mathrm{b}\}\end{aligned}$$

Patrick Cousot, Radhia Cousot: Grammar semantics, analysis and parsing by abstract interpretation. Theor. Comput. Sci. 412(44): 6135-6192 (2011)
Patrick Cousot, Radhia Cousot: Grammar Analysis and Parsing by Abstract Interpretation. Program Analysis and Compilation, LNCS 4444, 2006: 175-200
Patrick Cousot, Radhia Cousot: Parsing as abstract interpretation of grammar semantics. Theor. Comput. Sci. 290(1): 531-544 (2003)

Abstraction in a more general setting...

- Reasoning on complex [computer] system behaviors is too complex (for humans)
- Analyzing/verifying [computer] system behaviors is undecidable or subject to combinatorial explosion (for machines)
- Abstraction is necessary to apprehend complexity
- Abstract interpretation is a formal framework for reasoning/computing on formal models of [computer] objects, systems and computations and their relations
- Applications include the systematic construction of methods and effective algorithms to solve/approximate undecidable or very complex problems in various areas of computer science (and more recently system biology)

Recent advances

- The same principles apply to *termination verification*

Patrick Cousot, Radhia Cousot: *An abstract interpretation framework for termination*. POPL 2012: 245-258

- and to *probabilistic verification*

Patrick Cousot and Michaël Monerau. Probabilistic Abstract Interpretation. In H. Seidel (Ed), *22nd European Symposium on Programming (ESOP 2012)*, Tallinn, Estonia, 24 March—1 April 2012. Lecture Notes in Computer Science, vol. 7211, pp. 166—190, © Springer, 2012.

Applications of abstract interpretation

Static analysis and verification

Software

- Ait: static analysis of the worst-case execution time of control/command software (www.absint.com/ait/)
- Astrée: proof of absence of runtime errors in embedded synchronous real time control/command software (www.absint.com/astree/), AstréeA for asynchronous programs (www.astreea.ens.fr/)
- C Global Surveyor, NASA, static analyzer for flight software of NASA missions (www.cmu.edu/silicon-valley/faculty-staff/venet-arnaud.html)
- IKOS (Inference Kernel for Open Static Analyzers), (www.cmu.edu/silicon-valley/software-systems-management/software-verification.html)
- Checkmate: static analyzer of multi-threaded Java programs (www.pietro.ferrara.name/checkmate/)
- CodeContracts Static Checker, Microsoft (msdn.microsoft.com/en-us/devlabs/dd491992.aspx)
- Fluctuat: static analysis of the precision of numerical computations (www-list.cea.fr/labos/gb/LSL/fluctuat/index.html)

Software

- Infer: Static analyzer for C/C++ (monoidics.com/)
- Julia: static analyzer for Java and Android programs (www.juliasoft.com/juliasoft-android-java-verification.aspx?Id=201177234649)
- Predator: static analyzer of C dynamic data structures using separation logic (www.fit.vutbr.cz/research/groups/verifit/tools/predator/)
- Terminator: termination proof (www.cs.ucl.ac.uk/staff/p.ohearn/Invader/Invader/Invader_Home.html)
- etc.

- Apron numerical domains library (apron.cri.ensmp.fr/library/)
- Parma Polyhedral Library (bugseng.com/products/ppl/)
- etc.

Hardware

- (Generalized) symbolic trajectory evaluation (Intel)

Intel's Successes with Formal Methods

John Harrison

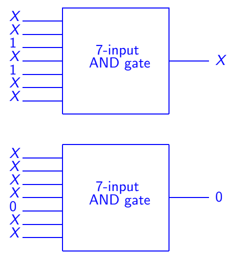
Intel Corporation

15 March 2012

Tsinghua software day, March 15, 2012, Tsinghua University, Beijing, China

Example of ternary simulation

If some inputs are undefined, the output often is too, but not always:



$$0 = \{0\}$$
$$1 = \{1\}$$
$$X = \{0, 1\}$$

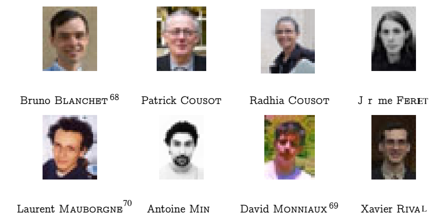
Jin Yang and Carl-Johan H. Seger, *Generalized Symbolic Trajectory Evaluation — Abstraction in Action*, Formal Methods in Computer-Aided Design, Lecture Notes in Computer Science, 2002, Volume 2517/2002, 70–87.

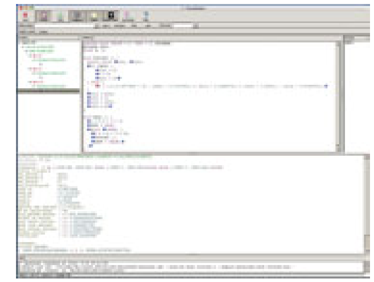
Jin Yang; Seger, C.-J.H.; *Introduction to generalized symbolic trajectory evaluation*, IEEE Transactions onVery Large Scale Integration (VLSI) Systems 11(3), June 2003, 345–353.

Biology

- Kappa – A language for modeling protein interaction networks by a set of rules and analyse that set directly deploying techniques from abstract interpretation (www.kappalanguage.org/ and fontana.med.harvard.edu/www/Documents/Lab/research.signaling.htm)

ASTRÉE





[68] Nov. 2001 Nov. 2003.
[69] Nov. 2001 Aug. 2007.
[70] Nov. 2001 – Aug. 2010.

Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, Xavier Rival: Why does Astrée scale up? Formal Methods in System Design 35(3): 229-264 (2009)

Patrick Cousot, Radhia Cousot, Jérôme Feret, Antoine Miné, Laurent Mauborgne, David Monniaux, Xavier Rival: Varieties of Static Analyzers: A Comparison with ASTREE. TASE 2007: 3-20

Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, Xavier Rival: Combination of Abstractions in the ASTRÉE Static Analyzer. ASIAN 2006: 272-300

Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, Xavier Rival: The ASTREÉ Analyzer. ESOP 2005: 21-30

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

Bruno Blanchet, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, Xavier Rival: Design and Implementation of a Special-Purpose Static Program Analyzer for Safety-Critical Real-Time Embedded Software. The Essence of Computation 2002: 85-108

Target language and applications

- C programming language
  - Without recursion, `longjump`, dynamic memory allocation, conflicting side effects, backward jumps, system calls (stubs)
  - With all its horrors (`union`, pointer arithmetics, etc)
  - Reasonably extending the standard (e.g. size & endianess of integers, IEEE 754-1985 floats, etc)
- Originally for synchronous control/command
  - e.g. generated from Scade

The semantics of C implementations is very hard to define

What is the effect of out-of-bounds array indexing?

```
% cat unpredictable.c
#include <stdio.h>
int main () { int n, T[1];
 n = 2147483647;
 printf("n = %i, T[n] = %i\n", n, T[n]);
}
```

Yields different results on different machines:

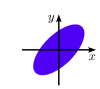
`n = 2147483647, T[n] = 2147483647`	Macintosh PPC
`n = 2147483647, T[n] = -1208492044`	Macintosh Intel
`n = 2147483647, T[n] = -135294988`	PC Intel 32 bits
`Bus error`	PC Intel 64 bits

Implicit specification

- Absence of runtime errors: overflows, division by zero, buffer overflow, null & dangling pointers, alignment errors, ...
- Semantics of runtime errors:
  - Terminating execution: stop (e.g. floating-point exceptions when traps are activated)
  - Predictable outcome: go on with worst case (e.g. signed integer overflows result in some integer, some options: e.g. modulo arithmetics)
  - Unpredictable outcome: stop (e.g. memory corruption)

Example of domain-specific abstraction: ellipses

```
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; }
}
```



An erroneous common belief on static analyzers

"The properties that can be proved by static analyzers are often simple" [2]

Like in mathematics:

- May be simple to state (no overflow)
- But harder to discover (`S[0], S[1] in [-1327.02698354, 1327.02698354]`)
- And difficult to prove (since it requires finding a non trivial non-linear invariant for second order filters with complex roots [Fer04], which can hardly be found by exhaustive enumeration)

Reference

[2] Vijay D'Silva, Daniel Kroening, and Georg Weissenbacher. A Survey of Automated Techniques for Formal Software Verification. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 27, No. 7, July 2008.

[Fer04] Jérôme Feret: Static Analysis of Digital Filters. ESOP 2004: 33-48

Industrial applications

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

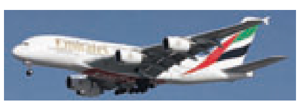
David Delmas, Jean Souyris: Astrée: From Research to Industry. SAS 2007: 437-451

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

Examples of applications

- Verification of the absence of runtime-errors in
  - Fly-by-wire flight control systems (*)

  - ATV docking system (*)

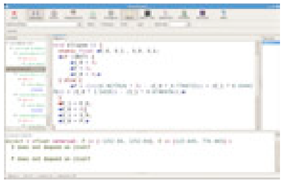
  - Flight warning system (on-going work)

(*) No false alarm at all!

Industrialization

- 8 years of research/development (CNRS/ENS/INRIA):

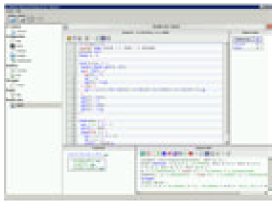
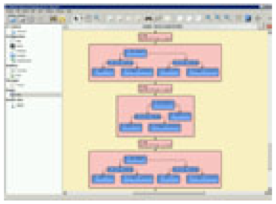
www.astree.ens.fr



- Industrialization by AbsInt (since Jan. 2010):

www.absint.com/astree/

AbsInt Angewandte Informatik

- Can be used for formal software certification in avionics (DO–178C & DO–333)

Conclusion

On research

If you reason/compute on computer/biological/... systems behaviors, you probably do abstract interpretation

On applications

If the simulation/analysis/checking of your computer/biological/… systems model does not scale up, consider using (sound (and complete)) abstract interpretations

The End, Thank You