

Theories, Solvers and Static Analysis by Abstract Interpretation

Patrick Cousot

di.ens.fr/~cousot
cs.nyu.edu/~pcousot

Radhia Cousot

di.ens.fr/~rcousot

joint work with **Laurent Mauborgne**
software.imdea.org/people/laurent.mauborgne/

References

- Patrick Cousot, Radhia Cousot, and Laurent Mauborgne.
[Logical Abstract Domains and Interpretations](#).
In *The Future of Software Engineering*, S. Nanz (Ed.).
© Springer 2010, Pages 48–71.
- Patrick Cousot, Radhia Cousot, and Laurent Mauborgne.
[The reduced product of abstract domains and the combination of decision procedures](#).
In *14th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2011)*, March 26 — April 3, 2011, Saarbrücken, Germany, Martin Hofmann (Ed.), Lecture Notes in Computer Science, Vol. 6604,
© Springer 2011, pages 456–472.

Combined, revised and extended into:

- Patrick Cousot, Radhia Cousot, and Laurent Mauborgne.
[Theories, Solvers and Static Analysis by Abstract Interpretation](#).
Submitted to a journal.
[Available from the authors](#).

Abstract

- The algebraic/model theoretic design of static analyzers uses abstract domains based on representations of properties and pre-calculated property transformers. It is very efficient. The logical/proof theoretic approach uses SMT solvers/theorem provers and computation of property transformers on-the-fly. It is very expressive. We propose to unify both approaches, so that they can be combined to reach the sweet spot best adapted to a specific application domain in the precision/cost spectrum. We first give a new formalization of the proof theoretic approach in the abstract interpretation framework, introducing a semantics based on multiple interpretations to deal with the soundness of such approaches. Then we describe how to combine them with any other abstract interpretation-based analysis using an iterated reduction to combine abstractions. The key observation is that the Nelson-Oppen procedure which decides satisfiability in a combination of logical theories by exchanging equalities and disequalities computes a reduced product (after the state is enhanced with some new "observations" corresponding to alien terms). By abandoning restrictions ensuring completeness (such as disjointness, convexity, stably-infiniteness, or shininess, etc) we can even broaden the application scope of logical abstractions for static analysis (which is incomplete anyway).
- Joint work with Laurent Mauborgne (IMDEA, Madrid)

Objective

Algebraic abstractions

- Used in **abstract interpretation** for analysis/verification of finite/infinite systems
- System properties and specifications are abstracted as an **algebraic lattice** (abstraction-specific encoding of properties)
- **Fully automatic**: system properties are computed as fixpoints of **algebraic transformers**
- Abstractions can be combined using the **reduced product**

Objective

- Show that **proof-theoretic/logical abstractions** are particular cases of **algebraic abstractions**
- Show that the **Nelson-Open procedure** is a particular case of the **reduced product**
- Use this **unifying point of view** to introduce a **new combination of logical and algebraic abstractions**

➔ **Convergence of proof theoretic/ logical and algebraic property-inference and verification methods**

Proof theoretic/logical abstractions

- Used in **deductive methods**
- System properties and specifications are expressed with formulæ of **first-order theories** (universal encoding of properties)
- **Partly automatic**: system properties are provided manually by end-users and automatically checked to satisfy **verification conditions** (with implication defined by the theories)
- Theories can be combined using **Nelson-Open procedure**

Concrete semantics

Programs (syntax)

- Expressions (on a signature $\langle \mathbb{f}, \mathbb{p} \rangle$)

$x, y, z, \dots \in \mathbb{x}$	variables
$a, b, c, \dots \in \mathbb{f}^0$	constants
$f, g, h, \dots \in \mathbb{f}^n, \quad \mathbb{f} \triangleq \bigcup_{n \geq 0} \mathbb{f}^n$	function symbols of arity $n \geq 1$
$t \in \mathbb{T}(\mathbb{x}, \mathbb{f}) \quad t ::= x \mid c \mid f(t_1, \dots, t_n)$	terms
$p, q, r, \dots \in \mathbb{p}^n, \quad \mathbb{p}^0 \triangleq \{\text{ff}, \text{tt}\}, \quad \mathbb{p} \triangleq \bigcup_{n \geq 0} \mathbb{p}^n$	predicate symbols of arity $n \geq 0$,
$a \in \mathbb{A}(\mathbb{x}, \mathbb{f}, \mathbb{p}) \quad a ::= \text{ff} \mid p(t_1, \dots, t_n) \mid \neg a$	atomic formulae
$e \in \mathbb{E}(\mathbb{x}, \mathbb{f}, \mathbb{p}) \triangleq \mathbb{T}(\mathbb{x}, \mathbb{f}) \cup \mathbb{A}(\mathbb{x}, \mathbb{f}, \mathbb{p})$	program expressions
$\varphi \in \mathbb{C}(\mathbb{x}, \mathbb{f}, \mathbb{p}) \quad \varphi ::= a \mid \varphi \wedge \varphi$	clauses in simple conjunctive normal form

- Programs (including assignment, guards, loops, ...)

$P, \dots \in \mathbb{P}(\mathbb{x}, \mathbb{f}, \mathbb{p})$	$P ::= x := e \mid \varphi \mid \dots$	programs
---	--	----------

Programs (concrete semantics)

- The program semantics is usually specified relative to a **standard interpretation** $\mathfrak{I} \in \mathfrak{I}$.
- The **concrete semantics** is given in **post-fixpoint** form (in case the least fixpoint which is also the least post-fixpoint does not exist, e.g. *inexpressibility* in Hoare logic)

$\mathcal{R}_{\mathfrak{I}}$	concrete observables ⁵
$\mathcal{P}_{\mathfrak{I}} \triangleq \wp(\mathcal{R}_{\mathfrak{I}})$	concrete properties ⁶
$F_{\mathfrak{I}}[\mathbb{P}] \in \mathcal{P}_{\mathfrak{I}} \rightarrow \mathcal{P}_{\mathfrak{I}}$	concrete transformer of program P
$C_{\mathfrak{I}}[\mathbb{P}] \triangleq \text{postfp}^{\subseteq} F_{\mathfrak{I}}[\mathbb{P}] \in \wp(\mathcal{P}_{\mathfrak{I}})$	concrete semantics of program P

where $\text{postfp}^{\subseteq} f \triangleq \{x \mid f(x) \leq x\}$

⁵Examples of observables are set of states, set of partial or complete execution traces, infinite/transfinite execution trees, etc.
⁶A property is understood as the set of elements satisfying this property.

Programs (mono-interpretation)

- Interpretation** $I \in \mathfrak{I}$ for a signature $\langle \mathbb{f}, \mathbb{p} \rangle$ is $\langle I_{\mathcal{V}}, I_{\mathcal{B}} \rangle$ such that

- $I_{\mathcal{V}}$ is a non-empty set of values,
- $\forall c \in \mathbb{f}^0 : I_{\mathcal{V}}(c) \in I_{\mathcal{V}}, \quad \forall n \geq 1 : \forall f \in \mathbb{f}^n : I_{\mathcal{V}}(f) \in I_{\mathcal{V}}^n \rightarrow I_{\mathcal{V}},$
- $\forall n \geq 0 : \forall p \in \mathbb{p}^n : I_{\mathcal{B}}(p) \in I_{\mathcal{B}}^n \rightarrow \mathcal{B}. \quad \mathcal{B} \triangleq \{\text{false}, \text{true}\}$

- Environments**

$\eta \in \mathcal{R}_I \triangleq \mathbb{x} \rightarrow I_{\mathcal{V}}$ environments

- Expression evaluation**

$\llbracket a \rrbracket, \eta \in \mathcal{B}$ of an atomic formula $a \in \mathbb{A}(\mathbb{x}, \mathbb{f}, \mathbb{p})$
 $\llbracket t \rrbracket, \eta \in I_{\mathcal{V}}$ of the term $t \in \mathbb{T}(\mathbb{x}, \mathbb{f})$

Example of program concrete semantics

- Program** $P \triangleq x=1; \text{ while true } \{x=\text{incr}(x)\}$

- Arithmetic interpretation** \mathfrak{I} on integers $\mathfrak{I}_{\mathcal{V}} = \mathbb{Z}$

- Loop invariant** $\text{Ifp}^{\subseteq} F_{\mathfrak{I}}[\mathbb{P}] = \{\eta \in \mathcal{R}_{\mathfrak{I}} \mid 0 < \eta(x)\}$

where $\mathcal{R}_{\mathfrak{I}} \triangleq \mathbb{x} \rightarrow \mathfrak{I}_{\mathcal{V}}$ concrete environments

$F_{\mathfrak{I}}[\mathbb{P}](X) \triangleq \{\eta \in \mathcal{R}_{\mathfrak{I}} \mid \eta(x) = 1\} \cup \{\eta[x \leftarrow \eta(x) + 1] \mid \eta \in X\}$

- The **strongest invariant** is $\text{Ifp}^{\subseteq} F_{\mathfrak{I}}[\mathbb{P}] = \bigcap \text{postfp}^{\subseteq} F_{\mathfrak{I}}[\mathbb{P}]$
- Expressivity**: the **Ifp** may not be expressible in the abstract in which case we use the set of possible invariants $C_{\mathfrak{I}}[\mathbb{P}] \triangleq \text{postfp}^{\subseteq} F_{\mathfrak{I}}[\mathbb{P}]$

Concrete domains

- The **standard semantics** describes computations of a system formalized by elements of a **domain of observables** $\mathcal{R}_{\mathcal{S}}$ (e.g. set of traces, states, etc)

The **properties** $\mathcal{P}_{\mathcal{S}} \triangleq \wp(\mathcal{R}_{\mathcal{S}})$ (a property is the set of elements with that property) form a complete lattice $\langle \mathcal{P}_{\mathcal{S}}, \subseteq, \emptyset, \mathcal{R}_{\mathcal{S}}, \cup, \cap \rangle$

- The concrete semantics $C_{\mathcal{S}}[[P]] \triangleq \text{postfp}^{\subseteq} F_{\mathcal{S}}[[P]]$ defines the **system properties** of interest for the verification
- The **transformer** $F_{\mathcal{S}}[[P]]$ is defined in terms of primitives, e.g.

$f_{\mathcal{S}}[[x := e]]P \triangleq \{\eta[x \leftarrow \llbracket e \rrbracket_{\mathcal{S}} \eta] \mid \eta \in P\}$ Floyd's assignment post-condition
 $p_{\mathcal{S}}[[\varphi]]P \triangleq \{\eta \in P \mid \llbracket \varphi \rrbracket_{\mathcal{S}} \eta = \text{true}\}$ test

Why using post-fixpoints is more general than using the least fixpoint?

- **The least fixpoint may not exist** (inexpressive logic) while post-fixpoints do exist (invariants)
- When the least fixpoint does exist, there is a bijection with the post-fixpoints (Tarski [1955])

$$\text{lfp}^{\subseteq} F_{\mathcal{S}}[[P]] = \bigcap \{ C \in \text{postfp}^{\subseteq} F_{\mathcal{S}}[[P]] \}$$

Concrete property satisfaction

- A program P satisfies a property P if and only if

$$\exists C \in C_{\mathcal{S}}[[P]] : C \subseteq P$$

Multiple concrete
interpretations/
semantics

About mathematical verification

- A verification relative to a purely mathematical semantics is of poor practical interest.

- Example (Muller's scheme as analyzed by Kahan)

```
x0 = 11/2.0;
x1 = 61/11.0;
for (i=1 ; i<=100 ; i++) {
  x2 = 111 - (1130 - 3000/x0) / x1;
  x0 = x1;  x1 = x2; }
```

- With exact reals, converges to 6 (repulsive fixpoint)
- With any finite precision, converges to 100 (attractive fixpoint)

⇒ Programs have many interpretations $\mathcal{I} \in \wp(\mathfrak{S})$.

Example I of abstraction of a multi-interpreted semantics

- The float operations have 4 possible interpretations depending on the rounding mode (towards $-\infty$, $+\infty$, 0, closest)
- ASTRÉE over-approximates all four semantics

Example II of abstraction of a multi-interpreted semantics

- Ignore some interpretations

$\langle \mathcal{P}_{\mathcal{I}}, \subseteq \rangle \xleftrightarrow[\alpha_{\mathcal{I} \rightarrow \mathcal{I}^\#}]{\gamma_{\mathcal{I}^\# \rightarrow \mathcal{I}}} \langle \mathcal{P}_{\mathcal{I}^\#}, \subseteq \rangle$ is a Galois connection where

$$\alpha_{\mathcal{I} \rightarrow \mathcal{I}^\#}(P) \triangleq P \cap \mathcal{P}_{\mathcal{I}^\#}$$

$$\gamma_{\mathcal{I}^\# \rightarrow \mathcal{I}}(Q) \triangleq \left\{ \langle I, \eta \rangle \mid I \in \mathcal{I} \wedge \eta \in \mathcal{R}_I \wedge (I \in \mathcal{I}^\# \Rightarrow \langle I, \eta \rangle \in Q) \right\}$$

Multi-interpreted semantics

- A generalization consists in considering multiple interpretations of logics and programs

- Multi-interpreted **properties**:

\mathcal{R}_I program observables for interpretation $I \in \mathcal{I} \in \wp(\mathfrak{S})$,
 $\mathcal{P}_{\mathcal{I}} \triangleq \{ I \in \mathcal{I} \mid \wp(\mathcal{R}_I) \}$ interpreted properties for the set of interpretations \mathcal{I}
 $\simeq \wp(\{ \langle I, \eta \rangle \mid I \in \mathcal{I} \wedge \eta \in \mathcal{R}_I \})$ ⁸

- Multi-interpreted **transformer**:

$$F_{\mathcal{I}}[\mathbb{P}] \in \mathcal{P}_{\mathcal{I}} \rightarrow \mathcal{P}_{\mathcal{I}} \\ \triangleq \lambda P \in \mathcal{P}_{\mathcal{I}} \cdot \lambda I \in \mathcal{I} \cdot F_I[\mathbb{P}](P(I))$$

- Multi-interpreted **semantics**:

$$C_{\mathcal{I}}[\mathbb{P}] \in \wp(\mathcal{P}_{\mathcal{I}}) \\ \triangleq \text{postfp}^{\subseteq} F_{\mathcal{I}}[\mathbb{P}]$$

where \subseteq is the pointwise subset ordering.

Background on abstract interpretation

Abstract domains

$$\langle A, \sqsubseteq, \perp, \top, \sqcup, \sqcap, \nabla, \Delta, \bar{f}, \bar{b}, \bar{p}, \dots \rangle$$

where

$\bar{P}, \bar{Q}, \dots \in A$	abstract properties
$\sqsubseteq \in A \times A \rightarrow \mathcal{B}$	abstract partial order ⁹
$\perp, \top \in A$	infimum, supremum ($\forall \bar{P} \in A : \perp \sqsubseteq \bar{P} \sqsubseteq \top$)
$\sqcup, \sqcap, \nabla, \Delta \in A \times A \rightarrow A$	abstract join, meet, widening, narrowing
...	
$\bar{f} \in (\mathbb{x} \times \mathbb{E}(\mathbb{x}, \mathbb{f}, \mathbb{p})) \rightarrow A \rightarrow A$	abstract forward assignment transformer
$\bar{b} \in (\mathbb{x} \times \mathbb{E}(\mathbb{x}, \mathbb{f}, \mathbb{p})) \rightarrow A \rightarrow A$	abstract backward assignment transformer
$\bar{p} \in \mathbb{C}(\mathbb{x}, \mathbb{f}, \mathbb{p}) \rightarrow A \rightarrow A$	abstract condition transformer.

Concretization

$$\gamma \in A \xrightarrow{\Delta} \mathcal{P}_{\mathcal{G}}$$

- **Soundness** of abstract domains:

$$\begin{aligned} (\bar{P} \sqsubseteq \bar{Q}) \Rightarrow (\gamma(\bar{P}) \subseteq \gamma(\bar{Q})) & \quad \text{order} & \quad \gamma(\perp) = \emptyset & \quad \text{infimum} \\ \gamma(\bar{P} \sqcup \bar{Q}) \supseteq (\gamma(\bar{P}) \cup \gamma(\bar{Q})) & \quad \text{join} & \quad \gamma(\top) = \top_{\mathcal{G}} & \quad \text{supremum} \\ \dots & & & \end{aligned}$$

- Up to an encoding, the abstraction consists in reasoning on a *subset* of the concrete properties

$$\gamma[A]$$

where

$$\gamma[X] \triangleq \{\gamma(x) \mid x \in X\}$$

Abstract semantics

- A abstract domain
- \sqsubseteq abstract logical implication
- $\bar{F}[[P]] \in A \rightarrow A$ abstract transformer defined in term of abstract primitives

$\bar{f} \in (\mathbb{x} \times \mathbb{E}(\mathbb{x}, \mathbb{f}, \mathbb{p})) \rightarrow A \rightarrow A$	abstract forward assignment transformer
$\bar{b} \in (\mathbb{x} \times \mathbb{E}(\mathbb{x}, \mathbb{f}, \mathbb{p})) \rightarrow A \rightarrow A$	abstract backward assignment transformer
$\bar{p} \in \mathbb{C}(\mathbb{x}, \mathbb{f}, \mathbb{p}) \rightarrow A \rightarrow A$	abstract condition transformer.
- $\bar{C}[[P]] \triangleq \{\text{lfp}^{\sqsubseteq} \bar{F}[[P]]\}$ least fixpoint semantics, if any
- $\bar{C}[[P]] \triangleq \{\bar{P} \mid \bar{F}[[P]](\bar{P}) \sqsubseteq \bar{P}\}$ or else, post-fixpoint abstract semantics

Soundness and completeness of abstract semantics

- The abstract semantics is **sound** iff

$$\forall \bar{P} \in A : (\exists \bar{C} \in \bar{C}[[P]] : \bar{C} \sqsubseteq \bar{P}) \Rightarrow (\exists C \in C[[P]] : C \subseteq \gamma(\bar{P}))$$

(any abstract proof of an abstract property can be done in the concrete)

- The abstract semantics is **complete** iff

$$\forall \bar{P} \in A : (\exists C \in C[[P]] : C \subseteq \gamma(\bar{P})) \Rightarrow (\exists \bar{C} \in \bar{C}[[P]] : \bar{C} \sqsubseteq \bar{P})$$

(any concrete proof of an abstract property can be done in the abstract)

Sufficient soundness condition

- THEOREM 4.4 (SOUNDNESS OF AN ABSTRACT POST-FIXPOINT SEMANTICS). If $C \llbracket P \rrbracket \triangleq \text{postfp}^{\leq} F \llbracket P \rrbracket$, $\bar{C} \llbracket P \rrbracket \triangleq \text{postfp}^{\sqsubseteq} \bar{F} \llbracket P \rrbracket$ and $\gamma : A \rightarrow C$ increasing, then

$$\forall \bar{P} \in A : F \llbracket P \rrbracket \circ \gamma(\bar{P}) \leq \gamma \circ \bar{F} \llbracket P \rrbracket(\bar{P})$$

implies that the abstract semantics is sound.

- This is usually implied by local conditions to be checked on the abstract domain

$$\begin{aligned} \gamma(\bar{f} \llbracket x := e \rrbracket \bar{P}) &\supseteq f_{\mathcal{J}} \llbracket x := e \rrbracket \gamma(\bar{P}) \\ \gamma(\bar{b} \llbracket x := e \rrbracket \bar{P}) &\supseteq b_{\mathcal{J}} \llbracket x := e \rrbracket \gamma(\bar{P}) \\ \gamma(\bar{p} \llbracket \varphi \rrbracket \bar{P}) &\supseteq p_{\mathcal{J}} \llbracket \varphi \rrbracket \gamma(\bar{P}) \end{aligned}$$

- Compositionality:

THEOREM 4.3 (COMPOSITIONALITY OF ABSTRACTIONS). The composition of sound (resp. complete) abstractions is sound (resp. complete).

Beyond bounded verification: Widening

DEFINITION 4.6 (WIDENING). Let $\langle A, \sqsubseteq \rangle$ be a poset. Then an **over-approximating widening** $\nabla \in A \times A \mapsto A$ is such that

$$(a) \forall x, y \in A : x \sqsubseteq x \nabla y \wedge y \sqsubseteq x \nabla y^{14}.$$

A **terminating widening** $\nabla \in A \times A \mapsto A$ is such that

$$(a) \text{ Given any sequence } \langle x^n, n \geq 0 \rangle, \text{ the sequence } y^0 = x^0, \dots, y^{n+1} = y^n \nabla x^n, \dots \text{ converges (i.e. } \exists \ell \in \mathbb{N} : \forall n \geq \ell : y^n = y^\ell \text{ in which case } y^\ell \text{ is called the limit of the widened sequence } \langle y^n, n \geq 0 \rangle).$$

Traditionally a widening is considered to be both over-approximating and terminating. \square

Best abstraction

- If the concretization preserves existing meets then we have a Galois connection

$$\langle \mathcal{P}_{\mathcal{J}}, \sqsubseteq \rangle \xleftrightarrow[\alpha]{\gamma} \langle A, \sqsubseteq \rangle$$

- If no two abstract properties have the same concretization, the abstraction is surjective

$$\langle \mathcal{P}_{\mathcal{J}}, \sqsubseteq \rangle \xleftrightarrow[\alpha]{\gamma} \langle A, \sqsubseteq \rangle$$

Iteration with widening

DEFINITION 4.7 (ITERATES WITH WIDENING). The iterates of a transformer $\bar{F} \llbracket P \rrbracket \in A \mapsto A$ from the infimum $\perp \in A$ with widening $\nabla \in A \times A \mapsto A$ in a poset $\langle A, \sqsubseteq \rangle$ are defined by recurrence as $\bar{F}^0 = \perp$, $\bar{F}^{n+1} = \bar{F}^n$ when $\bar{F} \llbracket P \rrbracket(\bar{F}^n) \sqsubseteq \bar{F}^n$ and $\bar{F}^{n+1} = \bar{F}^n \nabla \bar{F} \llbracket P \rrbracket(\bar{F}^n)$ otherwise. \square

THEOREM 4.8 (LIMIT OF THE ITERATES WITH WIDENING). The iterates in a poset $\langle A, \sqsubseteq, \perp \rangle$ of a transformer $\bar{F} \llbracket P \rrbracket$ from the infimum \perp with widening ∇ converge and their limit is a post-fixpoint of the transformer. \square

Implementation notes

- Each abstract domain $\langle A, \sqsubseteq, \perp, \top, \sqcup, \sqcap, \nabla, \Delta, \bar{f}, \bar{b}, \bar{p}, \dots \rangle$ is implemented separately, by providing a specific computer representation of properties in A , and algorithms for the logical operations $\sqsubseteq, \perp, \top, \sqcup, \sqcap$, and transformers $\bar{f}, \bar{b}, \bar{p}, \dots$
- Different abstract domains are combined into a reduced product
- Very efficient but requires skilled experts

First-order logical formulæ & satisfaction

- Syntax

$\Psi \in \mathbb{F}(\mathbb{x}, \mathbb{f}, \mathbb{p})$ $\Psi ::= a \mid \neg\Psi \mid \Psi \wedge \Psi \mid \exists \mathbf{x} : \Psi$ quantified first-order formulæ
 a distinguished predicate $= (t_1, t_2)$ which we write $t_1 = t_2$.

- Free variables $\vec{\mathbf{x}}_\Psi$

- Satisfaction

$I \models_\eta \Psi$, interpretation I and an environment η satisfy a formula Ψ

- Equality

$$I \models_\eta t_1 = t_2 \triangleq \llbracket t_1 \rrbracket, \eta =_I \llbracket t_2 \rrbracket, \eta$$

where $=_I$ is the unique reflexive, symmetric, antisymmetric, and transitive relation on I_V .

Multi-interpreted first-order logic

Extension to multi-interpretations

- A property is described by a formula for multiple interpretations

$$\mathcal{I} \in \wp(\mathfrak{I})$$

- Semantics of first-order formulæ

$$\gamma_I^a \in \mathbb{F}(\mathbb{x}, \mathbb{f}, \mathbb{p}) \xrightarrow{\lambda} \mathcal{P}_I$$

$$\gamma_I^a(\Psi) \triangleq \{ \langle I, \eta \rangle \mid I \in \mathcal{I} \wedge I \models_\eta \Psi \}$$

- But how are we going to describe sets of interpretations $\mathcal{I} \in \wp(\mathfrak{I})$?

Defining multiple interpretations as models of theories

- **Theory**: set \mathcal{T} of theorems (closed sentences without any free variable)
- **Models** of a theory (interpretations making true all theorems of the theory)

$$\begin{aligned}\mathfrak{M}(\mathcal{T}) &\triangleq \{I \in \mathfrak{I} \mid \forall \Psi \in \mathcal{T} : \exists \eta : I \models_{\eta} \Psi\} \\ &= \{I \in \mathfrak{I} \mid \forall \Psi \in \mathcal{T} : \forall \eta : I \models_{\eta} \Psi\}\end{aligned}$$

Checking satisfiability modulo theory

- **Validity modulo theory**

$$\text{valid}_{\mathcal{T}}(\Psi) \triangleq \forall I \in \mathfrak{M}(\mathcal{T}) : \forall \eta : I \models_{\eta} \Psi$$

- **Satisfiability modulo theory (SMT)**

$$\text{satisfiable}_{\mathcal{T}}(\Psi) \triangleq \exists I \in \mathfrak{M}(\mathcal{T}) : \exists \eta : I \models_{\eta} \Psi$$

- **Checking satisfiability for decidable theories**

$$\text{satisfiable}_{\mathcal{T}}(\Psi) \Leftrightarrow \neg(\text{decide}_{\mathcal{T}}(\forall \vec{x}_{\Psi} : \neg \Psi)) \quad (\text{when } \mathcal{T} \text{ is decidable and deductive})$$

$$\text{satisfiable}_{\mathcal{T}}(\Psi) \Leftrightarrow (\text{decide}_{\mathcal{T}}(\exists \vec{x}_{\Psi} : \Psi)) \quad (\text{when } \mathcal{T} \text{ is decidable and complete})$$

- **Most SMT solvers support only limited forms of quantified formulæ**

Classical properties of theories

- **Decidable theories**: $\forall \Psi \in \mathbb{F}(\mathbb{x}, \mathbb{f}, \mathbb{p}) : \text{decide}_{\mathcal{T}}(\Psi) \triangleq (\Psi \in \mathcal{T})$ is computable

- **Deductive theories**: closed by deduction

$$\forall \Psi \in \mathcal{T} : \forall \Psi' \in \mathbb{F}(\mathbb{x}, \mathbb{f}, \mathbb{p}), \text{ if } \Psi \Rightarrow \Psi' \text{ implies } \Psi' \in \mathcal{T}$$

- **Satisfiable theory**:

$$\mathfrak{M}(\mathcal{T}) \neq \emptyset$$

- **Complete theory**:

for all sentences Ψ in the language of the theory, either Ψ is in the theory or $\neg \Psi$ is in the theory.

Example of abstraction: Logical abstractions

Logical abstract domains

- $\langle A, \mathcal{T} \rangle$: $A \in \wp(\mathbb{F}(\mathbb{x}, \mathbb{f}, \mathbb{p}))$ abstract properties
 \mathcal{T} theory of $\mathbb{F}(\mathbb{x}, \mathbb{f}, \mathbb{p})$
- Abstract domain $\langle A, \sqsubseteq, \text{ff}, \text{tt}, \vee, \wedge, \nabla, \Delta, \bar{f}_a, \bar{b}_a, \bar{p}_a, \dots \rangle$
- Logical implication $(\Psi \sqsubseteq \Psi') \triangleq ((\forall \bar{x}_\Psi \cup \bar{x}_{\Psi'} : \Psi \Rightarrow \Psi') \in \mathcal{T})$
- A lattice but in general not complete
- The concretization is

$$\gamma_{\mathcal{T}}^a(\Psi) \triangleq \left\{ \langle I, \eta \rangle \mid I \in \mathfrak{M}(\mathcal{T}) \wedge I \models_{\eta} \Psi \right\}$$

Implementation notes ...

- Universal representation of abstract properties by logical formulæ
- Trivial implementations of logical operations $\text{ff}, \text{tt}, \vee, \wedge,$
- Provers or SMT solvers can be used for the abstract implication $\sqsubseteq,$
- Concrete transformers are purely syntactic

$f_a \in (\mathbb{x} \times \mathbb{T}(\mathbb{x}, \mathbb{f})) \rightarrow \mathbb{F}(\mathbb{x}, \mathbb{f}, \mathbb{p}) \rightarrow \mathbb{F}(\mathbb{x}, \mathbb{f}, \mathbb{p})$	axiomatic forward assignment transformer
$f_a[[x := t]]\Psi \triangleq \exists x' : \Psi[x \leftarrow x'] \wedge x = t[x \leftarrow x']$	
$b_a \in (\mathbb{x} \times \mathbb{T}(\mathbb{x}, \mathbb{f})) \rightarrow \mathbb{F}(\mathbb{x}, \mathbb{f}, \mathbb{p}) \rightarrow \mathbb{F}(\mathbb{x}, \mathbb{f}, \mathbb{p})$	axiomatic backward assignment transformer
$b_a[[x := t]]\Psi \triangleq \Psi[x \leftarrow t]$	
$p_a \in \mathbb{C}(\mathbb{x}, \mathbb{f}, \mathbb{p}) \rightarrow \mathbb{F}(\mathbb{x}, \mathbb{f}, \mathbb{p}) \rightarrow \mathbb{F}(\mathbb{x}, \mathbb{f}, \mathbb{p})$	axiomatic transformer for program test of condition φ .
$p_a[[\varphi]]\Psi \triangleq \Psi \wedge \varphi$	

.../...

Logical abstract semantics

- Logical abstract semantics
 $\bar{C}^a[[P]] \triangleq \{ \Psi \mid \bar{F}_a[[P]](\Psi) \sqsubseteq \Psi \}$
- The logical abstract transformer $\bar{F}_a[[P]] \in A \rightarrow A$ is defined in terms of primitives

$\bar{f}_a \in (\mathbb{x} \times \mathbb{T}(\mathbb{x}, \mathbb{f})) \rightarrow A \rightarrow A$	abstract forward assignment transformer
$\bar{b}_a \in (\mathbb{x} \times \mathbb{T}(\mathbb{x}, \mathbb{f})) \rightarrow A \rightarrow A$	abstract backward assignment transformer
$\bar{p}_a \in \mathbb{L} \rightarrow A \rightarrow A$	condition abstract transformer

but ...

.../... so the abstract transformers follow by abstraction

$\bar{f}_a[[x := t]]\Psi \triangleq \alpha_A^f(\bar{f}_a[x := t]\Psi)$	abstract forward assignment transformer
$\bar{b}_a[[x := t]]\Psi \triangleq \alpha_A^b(\bar{b}_a[x := t]\Psi)$	abstract backward assignment transformer
$\bar{p}_a[[\varphi]]\Psi \triangleq \alpha_A^p(\bar{p}_a[[\varphi]]\Psi)$	abstract transformer for program test of condition

- The abstraction algorithm $\alpha_A^f \in \mathbb{F}(\mathbb{x}, \mathbb{f}, \mathbb{p}) \rightarrow A$ to abstract properties in A may be non-trivial (e.g. quantifiers elimination)
- A widening ∇ is needed to ensure convergence of the fixpoint iterates (or else ask the end-user)

Example I of widening: thresholds

- Choose a subset W of A satisfying the ascending chain condition for \sqsubseteq ,
- Define $X \nabla Y$ to be (one of) the strongest $\Psi \in W$ such that $Y \Rightarrow \Psi$

Example II of bounded widening: Craig interpolation

- Use Craig interpolation (knowing a bound e.g. the specification)
- Move to thresholds to enforced convergence after k widenings with Craig interpolation

Reduced Product

Cartesian product

- Definition of the Cartesian product:

Let $\langle A_i, \sqsubseteq_i \rangle$, $i \in \Delta$, Δ finite, be abstract domains with increasing concretization $\gamma_i \in A_i \mapsto \mathfrak{P}_I^{\Sigma O}$. Their Cartesian product is $\langle \vec{A}, \vec{\sqsubseteq} \rangle$ where $\vec{A} \triangleq \times_{i \in \Delta} A_i$, $(\vec{P} \vec{\sqsubseteq} \vec{Q}) \triangleq \bigwedge_{i \in \Delta} (\vec{P}_i \sqsubseteq_i \vec{Q}_i)$ and $\vec{\gamma} \in \vec{A} \rightarrow \mathfrak{P}_I^{\Sigma O}$ is $\vec{\gamma}(\vec{P}) \triangleq \bigcap_{i \in \Delta} \gamma_i(\vec{P}_i)$.

Reduced product

- Definition of the Reduced product:

Let $\langle A_i, \sqsubseteq_i \rangle$, $i \in \Delta$, Δ finite, be abstract domains with increasing concretization $\gamma_i \in A_i \mapsto \mathfrak{P}_I^{\Sigma O}$ where $\vec{A} \triangleq \times_{i \in \Delta} A_i$ is their Cartesian product. Their reduced product is $\langle \vec{A}/\cong, \vec{\sqsubseteq} \rangle$ where $(\vec{P} \cong \vec{Q}) \triangleq (\vec{\gamma}(\vec{P}) = \vec{\gamma}(\vec{Q}))$ and $\vec{\gamma}$ as well as $\vec{\sqsubseteq}$ are naturally extended to the equivalence classes $[\vec{P}]/\cong$, $\vec{P} \in \vec{A}$, of \cong by $\vec{\gamma}([\vec{P}]/\cong) = \vec{\gamma}(\vec{P})$ and $[\vec{P}]/\cong \vec{\sqsubseteq} [\vec{Q}]/\cong \triangleq \exists \vec{P}' \in [\vec{P}]/\cong : \exists \vec{Q}' \in [\vec{Q}]/\cong : \vec{P}' \vec{\sqsubseteq} \vec{Q}'$. \square

- In practice, the reduced product may be complex to compute but we can use approximations such as the iterated pairwise reduction of the Cartesian product

Reduction

- **Example:** intervals x congruences

$$\rho(x \in [-1,5] \wedge x = 2 \bmod 4) \equiv x \in [2,2] \wedge x = 2 \bmod 4$$

are equivalent

- **Meaning-preserving reduction:**

Let $\langle A, \sqsubseteq \rangle$ be a poset which is an abstract domain with concretization $\gamma \in A \rightarrow C$ where $\langle C, \leq \rangle$ is the concrete domain. A meaning-preserving map is $\rho \in A \rightarrow A$ such that $\forall \bar{P} \in A : \gamma(\rho(\bar{P})) = \gamma(\bar{P})$. The map is a reduction if and only if it is reductive that is $\forall \bar{P} \in A : \rho(\bar{P}) \sqsubseteq \bar{P}$. \square

Iterated reduction

- Definition of **iterated reduction**:

Let $\langle A, \sqsubseteq \rangle$ be a poset which is an abstract domain with concretization $\gamma \in A \rightarrow C$ where $\langle C, \leq \rangle$ is the concrete domain and $\rho \in A \rightarrow A$ be a meaning-preserving reduction.

The iterates of the reduction are $\rho^0 \triangleq \lambda \bar{P} \cdot \bar{P}$, $\rho^{\lambda+1} = \rho(\rho^\lambda)$ for successor ordinals and $\rho^\lambda = \bigsqcap_{\beta < \lambda} \rho^\beta$ for limit ordinals.

The iterates are well-defined when the greatest lower bounds \bigsqcap (glb) do exist in the poset $\langle A, \sqsubseteq \rangle$. \square

Why is reduction important

- **Without reduction** (signs and parity):

```

\\ x ≥ 0 – odd(x)
if (x ≤ 0) then
  // x == 0 – odd(x)
  
```

- **With reduction:**

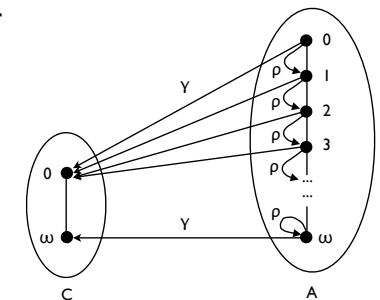
```

\\ x ≥ 0 – odd(x)
\\ x > 0 – odd(x)           ➡ reduction
if (x ≤ 0) then
  // false – odd(x)
  // false                   ➡ reduction
  
```

Finite versus infinite iterated reduction

- **Finite iterations** of a meaning preserving reduction are meaning preserving (and more precise)

- **Infinite iterations**, limits of meaning-preserving reduction, may not be meaning-preserving (although more precise). It is when γ preserves glbs.



Pairwise reduction

- Definition of pairwise reduction

Let $\langle A_i, \sqsubseteq_i \rangle$ be abstract domains with increasing concretization $\gamma_i \in A_i \xrightarrow{\sqsubseteq_i} L$ into the concrete domain $\langle L, \leq \rangle$.

For $i, j \in \Delta$, $i \neq j$, let $\rho_{ij} \in \langle A_i \times A_j, \sqsubseteq_{ij} \rangle \mapsto \langle A_i \times A_j, \sqsubseteq_{ij} \rangle$ be pairwise meaning-preserving reductions (so that $\forall \langle x, y \rangle \in A_i \times A_j : \rho_{ij}(\langle x, y \rangle) \sqsubseteq_{ij} \langle x, y \rangle$ and $(\gamma_i \times \gamma_j) \circ \rho_{ij} = (\gamma_i \times \gamma_j)^{24}$).

Define the pairwise reductions $\vec{\rho}_{ij} \in \langle \vec{A}, \vec{\sqsubseteq} \rangle \mapsto \langle \vec{A}, \vec{\sqsubseteq} \rangle$ of the Cartesian product as

$\vec{\rho}_{ij}(\vec{P}) \triangleq \text{let } \langle \vec{P}'_i, \vec{P}'_j \rangle \triangleq \rho_{ij}(\langle \vec{P}_i, \vec{P}_j \rangle) \text{ in } \vec{P}[i \leftarrow \vec{P}'_i][j \leftarrow \vec{P}'_j]$

where $\vec{P}[i \leftarrow x]_i = x$ and $\vec{P}[i \leftarrow x]_j = \vec{P}_j$ when $i \neq j$.

²⁴ We define $(f \times g)(\langle x, y \rangle) \triangleq \langle f(x), g(y) \rangle$.

Iterated pairwise reduction

- The iterated pairwise reduction of the Cartesian product is meaning preserving

If the limit $\vec{\rho}^*$ of the iterated reductions is well defined then the reductions are such that $\forall \vec{P} \in \vec{A} : \forall n \in \mathbb{N}_+ : \vec{\rho}^*(\vec{P}) \sqsubseteq \vec{\rho}^n(\vec{P}) \sqsubseteq \vec{\rho}_{ij}(\vec{P}) \sqsubseteq \vec{P}$, $i, j \in \Delta$, $i \neq j$ and meaning-preserving since $\vec{\rho}^\lambda(\vec{P}), \vec{\rho}_{ij}(\vec{P}), \vec{P} \in [\vec{P}] / \equiv$.

If, moreover, γ preserves greatest lower bounds then $\vec{\rho}^*(\vec{P}) \in [\vec{P}] / \equiv$. \square

Pairwise reduction (cont'd)

Define the iterated pairwise reductions $\vec{\rho}^n, \vec{\rho}^\lambda, \vec{\rho}^* \in \langle \vec{A}, \vec{\sqsubseteq} \rangle \mapsto \langle \vec{A}, \vec{\sqsubseteq} \rangle$, $n \geq 0$ of the Cartesian product for

$$\vec{\rho} \triangleq \bigcirc_{\substack{i, j \in \Delta, \\ i \neq j}} \vec{\rho}_{ij}$$

where $\bigcirc_{i=1}^n f_i \triangleq f_{\pi_1} \circ \dots \circ f_{\pi_n}$ is the function composition for some arbitrary permutation π of $[1, n]$. \square

Iterated pairwise reduction

- In general, the iterated pairwise reduction of the Cartesian product is not as precise as the reduced product
- Sufficient conditions do exist for their equivalence

Counter-example

- $L = \wp(\{a, b, c\})$
- $A_1 = \{\emptyset, \{a\}, \top\}$ where $\top = \{a, b, c\}$
- $A_2 = \{\emptyset, \{a, b\}, \top\}$
- $A_3 = \{\emptyset, \{a, c\}, \top\}$
- $\langle \top, \{a, b\}, \{a, c\} \rangle / \cong = \langle \{a\}, \{a, b\}, \{a, c\} \rangle$
- $\vec{\rho}_{ij}(\langle \top, \{a, b\}, \{a, c\} \rangle) = \langle \top, \{a, b\}, \{a, c\} \rangle$
for $\Delta = \{1, 2, 3\}, i, j \in \Delta, i \neq j$
- $\vec{\rho}^*(\langle \top, \{a, b\}, \{a, c\} \rangle) = \langle \top, \{a, b\}, \{a, c\} \rangle$ is **not** a minimal element of $[\langle \top, \{a, b\}, \{a, c\} \rangle] / \cong$

The Nelson-Oppen combination procedure

- Prove **satisfiability in a combination of theories** by exchanging equalities and disequalities
- **Example:** $\varphi \triangleq (x = a \vee x = b) \wedge f(x) \neq f(a) \wedge f(x) \neq f(b)$ ²².
 - **Purify:** introduce auxiliary variables to separate alien terms and put in conjunctive form

$$\begin{aligned} \varphi &\triangleq \varphi_1 \wedge \varphi_2 \text{ where} \\ \varphi_1 &\triangleq (x = a \vee x = b) \wedge y = a \wedge z = b \\ \varphi_2 &\triangleq f(x) \neq f(y) \wedge f(x) \neq f(z) \end{aligned}$$

.../...

²²where a, b and f are in different theories

Nelson-Oppen combination procedure

The Nelson-Oppen combination procedure

$$\begin{aligned} \varphi &\triangleq \varphi_1 \wedge \varphi_2 \text{ where} \\ \varphi_1 &\triangleq (x = a \vee x = b) \wedge y = a \wedge z = b \\ \varphi_2 &\triangleq f(x) \neq f(y) \wedge f(x) \neq f(z) \end{aligned}$$

- **Reduce** $\vec{\rho}(\varphi)$: each theory \mathcal{T}_i determines E_{ij} , a (disjunction) of conjunctions of variable (dis)equalities implied by φ_j and propagates it in all other components φ_i

$$\begin{aligned} E_{12} &\triangleq (x = y) \vee (x = z) \\ E_{21} &\triangleq (x \neq y) \wedge (x \neq z) \end{aligned}$$

- **Iterate** $\vec{\rho}^*(\varphi)$: until satisfiability is proved in each theory or stabilization of the iterates

The Nelson-Oppen combination procedure

Under **appropriate hypotheses** (disjointness of the theory signatures, stably-infiniteness/shininess, convexity to avoid disjunctions, etc), the Nelson-Oppen procedure:

- **Terminates** (finitely many possible (dis)equalities)
- Is **sound** (meaning-preserving)
- Is **complete** (always succeeds if formula is satisfiable)
- Similar techniques are used in theorem provers

The Nelson-Oppen procedure is an iterated pairwise reduced product

Is completeness of the Nelson-Oppen procedure needed?

- **Yes**, if you want to win the **SMT-COMP** competition (*)
- **No**, for program static analysis/verification
 - Verification is **undecidable** anyway so requiring completeness is useless.
 - Therefore these hypotheses (disjointness of the theory signatures, stably-infiniteness/shininess, convexity, etc) can be lifted, the procedure is then sound and incomplete.
- No change to SMT solvers is needed.

(*) congratulations to Z3 for SMT-COMP 2011, <http://www.smtexec.org/exec/?jobs=856>

Observables in Abstract Interpretation

- (Relational) **abstractions of values** (v_1, \dots, v_n) of program variables (x_1, \dots, x_n) is often too **imprecise**.

Example : when analyzing *quaternions* (a, b, c, d) we need to observe the evolution of $\sqrt{a^2+b^2+c^2+d^2}$ during execution to get a precise analysis of the normalization

- An **observable** is specified as the value of a function f of the values (v_1, \dots, v_n) of the program variables (x_1, \dots, x_n) assigned to a fresh auxiliary variable x_0

$$x_0 == f(v_1, \dots, v_n)$$

(with a precise abstraction of f)

Purification = Observables in A.I.

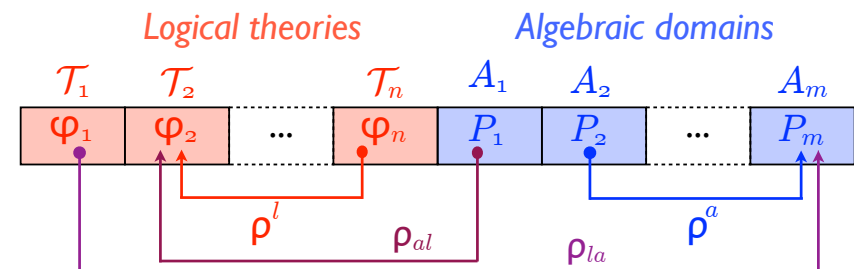
- The **purification** phase consists in introducing new **observables**
- The **program can be purified** by introducing auxiliary assignments of pure sub-expressions so that forward/backward transformers of purified formulæ always yield purified formulæ
- Example (f and a, b are in different theories):

$$y = f(x) == f(a+l) \ \& \ f(x) == f(2*b)$$
 becomes

$$z=a+l; t=2*b; y = f(x) == f(z) \ \& \ f(x) = f(t)$$

Static analysis combining logical and algebraic abstractions

Reduced product of logical and algebraic domains



- When checking satisfiability of $\varphi_1 \wedge \varphi_2 \wedge \dots \wedge \varphi_n$, the Nelson-Oppen procedure generates (dis-)equalities that can be propagated by ρ^{la} to reduce the $P_i, i=1, \dots, m$, or
- $\alpha_i(\varphi_1 \wedge \varphi_2 \wedge \dots \wedge \varphi_n)$ can be propagated by ρ^{la} to reduce the $P_i, i=1, \dots, m$
- The purification to theory T_i of $\gamma_i(P_i)$ can be propagated to φ_i by ρ^{al} in order to reduce it to $\varphi_i \wedge \gamma_i(P_i)$ (in T_i)

Advantages

- No need for completeness hypotheses on theories
 - Bidirectional reduction between logical and algebraic abstractions
 - No need for end-users to provide inductive invariants (discovered by static analysis)^(*)
 - Easy interaction with end-user (through logical formulæ)
 - Easy introduction of new abstractions on either side
- ⇒ Extensible expressive static analyzers / verifiers

^(*) may need occasionally to be strengthened by the end-user

Conclusion

- Future convergence between logic-based proof-theoretic deductive methods using SMT solvers/ theorem provers and algebraic methods using abstract interpretation for infinite-state systems?
- Expressiveness is important
- Efficiency is decisive
- Reproducibility is crucial

Future work

- Still at a conceptual stage
- More experimental work on a prototype is needed to validate the concept

Another relation between SAT-solvers and abstract interpretation

*** Vendredi 10 Juin 2011, Salle R, 14h00–15h00 *****

DPLL is Abstract Interpretation

Leopold Haller (Oxford University)

Re'sume' / Abstract:

The DPLL algorithm for deciding propositional satisfiability is a fundamental algorithm with applications in several, diverse areas of computer science. In its modern formulation, it combines intelligent model search, deduction, and lemma generation in an elegant and highly efficient manner. An important question is whether the DPLL algorithm can be generalised to richer classes of problems.

In this talk, I will outline how DPLL can naturally be viewed as an instance of abstract interpretation. The resulting abstract DPLL framework is a strict generalisation of DPLL to significantly richer logics and to non-Boolean domains. The instantiation of this framework with safety properties and abstract domains yields a powerful class of program analysis techniques which perform dynamic, property-driven domain-refinement.

These results have immediate consequences for building program analysis tools. I will present an instantiation of the abstract DPLL framework in the domain of floating point intervals. This instantiation dynamically constructs a trace-partitioning over the base domain that is precise enough to prove complex programs correct, yet coarse enough to enable efficient analysis.

**The End
Thank You**