Combining Algebraic Domains and Logical Theories by the Reduced Product

Patrick Cousot

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

joint work with

Radhia Cousot   Laurent Mauborgne

di.ens.fr/~rcousot   software.imdea.org/people/laurent.mauborgne/

Abstract

- In static analysis by abstract interpretation, algebraic abstract domains can be combined by the reduced product, or over-approximated by the iterated pairwise reductions.

  In Satisfiability Modulo Theories (SMT) solvers, the Nelson-Oppen (NO) theory combination schema provides, under various restrictions, a sound and complete decision procedure for the combining of disjoint theories by exchanges of disjunctions of equalities and disequalities. Understood as abstract domains, we show that the NO procedure is an iterated pairwise reduction.

  In the context of static analysis, theories can be understood as abstract domains. Completeness is useless in the abstract since the static analysis problem is undecidable anyway. This point of view introduces generalizations which, when combined with bounded widenings (such as interpolation), yields new cominations of algebraic and logical abstractions.
- Joint work with Radhia Cousot, ENS & CNRS, Paris and Laurent Mauborgne, IMDEA, Madrid.

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.

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

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-Oppen procedure

Objective

- Show that proof-theoretic/logical abstractions are particular cases of algebraic abstractions
- Show that the Nelson-Oppen 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

Concrete semantics

Programs (syntax)

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

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

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

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

Programs (mono-interpretation)

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

— $I_{\mathcal{V}}$ is a non-empty set of values,

— $\forall \mathtt{c} \in \mathbb{f}^0 : I_\gamma(\mathtt{c}) \in I_{\mathcal{V}}, \quad \forall n \geqslant 1 : \forall \mathtt{f} \in \mathbb{f}^n : I_\gamma(\mathtt{f}) \in I_{\mathcal{V}}^n \to I_{\mathcal{V}}$,

— $\forall n \geqslant 0 : \forall \mathtt{p} \in \mathbb{p}^n : I_\gamma(\mathtt{p}) \in I_{\mathcal{V}}^n \to \mathcal{B}$. $\qquad \mathcal{B} \triangleq \{\mathit{false}, \mathit{true}\}$

- Environments

$$\eta \in \mathcal{R}_I \triangleq \mathbb{x} \to I_{\mathcal{V}} \quad \text{environments}$$

- Expression evaluation

$\llbracket a \rrbracket_I \eta \in \mathcal{B}$ of an atomic formula $a \in \mathbb{A}(\mathbb{x}, \mathbb{f}, \mathbb{p})$

$\llbracket t \rrbracket_I \eta \in I_{\mathcal{V}}$ of the term $t \in \mathbb{T}(\mathbb{x}, \mathbb{f})$

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]
$F_{\mathfrak{I}}\llbracket \mathtt{P} \rrbracket$	$\in \mathcal{P}_{\mathfrak{I}} \to \mathcal{P}_{\mathfrak{I}}$	concrete transformer of program P
$C_{\mathfrak{I}}\llbracket \mathtt{P} \rrbracket \triangleq \mathbf{postfp}^{\subseteq} F_{\mathfrak{I}}\llbracket \mathtt{P} \rrbracket$	$\in \wp(\mathcal{P}_{\mathfrak{I}})$	concrete semantics of program P

where $\mathbf{postfp}^{\leq} f \triangleq \left\{ x \,\middle|\, f(x) \leq x \right\}$

[5] Examples of observables are set of states, set of partial or complete execution traces, infinite/transfinite execution trees, etc.

[6] A property is understood as the set of elements satisfying this property.

Example of program concrete semantics

- Program $\quad \mathtt{P} \triangleq$ `x=1; while true {x=incr(x)}`
- Arithmetic interpretation $\quad \mathfrak{I}$ on integers $\mathfrak{I}_{\mathcal{V}} = \mathbb{Z}$
- Loop invariant $\quad \mathbf{lfp}^{\subseteq} F_{\mathfrak{I}}\llbracket \mathtt{P} \rrbracket = \{\eta \in \mathcal{R}_{\mathfrak{I}} \mid 0 < \eta(\mathbf{x})\}$

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

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

- The *strongest invariant* is $\mathbf{lfp}^{\subseteq} F_{\mathfrak{I}}\llbracket \mathtt{P} \rrbracket = \bigcap \mathbf{postfp}^{\subseteq} F_{\mathfrak{I}}\llbracket \mathtt{P} \rrbracket$
- *Expressivity*: the $\mathbf{lfp}$ may not be expressible in the abstract in which case we use the set of possible invariants $C_{\mathfrak{I}}\llbracket \mathtt{P} \rrbracket \triangleq \mathbf{postfp}^{\subseteq} F_{\mathfrak{I}}\llbracket \mathtt{P} \rrbracket$

Concrete domains

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

  The properties $\mathcal{P}_{\mathfrak{I}} \triangleq \wp(\mathcal{R}_{\mathfrak{I}})$ (a property is the set of elements with that property) form a complete lattice $\langle \mathcal{P}_{\mathfrak{I}}, \subseteq, \emptyset, \mathcal{R}_{\mathfrak{I}}, \cup, \cap \rangle$
- The concrete semantics $C_{\mathfrak{I}}[\![\mathrm{P}]\!] \triangleq \mathbf{postfp}^{\subseteq} F_{\mathfrak{I}}[\![\mathrm{P}]\!]$ defines the system properties of interest for the verification
- The transformer $F_{\mathfrak{I}}[\![\mathrm{P}]\!]$ is defined in terms of primitives, e.g.

$\mathsf{f}_{\mathfrak{I}}[\![\mathtt{x} := e]\!]P$	$\triangleq$	$\{\eta[\mathtt{x} \leftarrow [\![e]\!]_{\mathfrak{I}}\eta] \mid \eta \in P)\}$	Floyd's assignment post-condition
$\mathsf{p}_{\mathfrak{I}}[\![\varphi]\!]P$	$\triangleq$	$\{\eta \in P \mid [\![\varphi]\!]_{\mathfrak{I}}\eta = true\}$	test

Concrete property satisfaction

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

$$\exists C \in C_{\mathfrak{I}}[\![P]\!] : C \subseteq P$$

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])

$$\mathbf{lfp}^{\subseteq} F_{\mathfrak{I}}[\![\mathrm{P}]\!] = \bigcap \mathbf{postfp}^{\subseteq} F_{\mathfrak{I}}[\![\mathrm{P}]\!] \in \mathbf{postfp}^{\subseteq} F_{\mathfrak{I}}[\![\mathrm{P}]\!]$$

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

Multi-interpreted semantics

- A generalization consists in considering multiple interpretations of logics and programs
- Multi-interpreted properties:

$\mathcal{R}_I$ program observables for interpretation $I \in \boldsymbol{\mathcal{I}} \in \wp(\mathfrak{I})$

$\mathcal{P}_{\mathcal{I}} \triangleq I \in \mathcal{I} \not\mapsto \wp(\mathcal{R}_I)$ 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:

$$\begin{aligned} F_{\mathcal{I}}[\![\mathrm{P}]\!] &\in \mathcal{P}_{\mathcal{I}} \to \mathcal{P}_{\mathcal{I}} \\ &\triangleq \lambda P \in \mathcal{P}_{\mathcal{I}} \bullet \lambda I \in \mathcal{I} \bullet F_I[\![\mathrm{P}]\!](P(I)) \end{aligned}$$

- Multi-interpreted semantics:

$$\begin{aligned} C_{\mathcal{I}}[\![\mathrm{P}]\!] &\in \wp(\mathcal{P}_{\mathcal{I}}) \\ &\triangleq \mathbf{postfp}^{\dot{\subseteq}}\, F_{\mathcal{I}}[\![\mathrm{P}]\!] \end{aligned}$$

where $\dot{\subseteq}$ is the pointwise subset ordering.

Example I of abstraction of a multi-interpreted semantics

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

Example II of abstraction of a multi-interpreted semantics

- Ignore some interpretations

$\langle \mathcal{P}_{\mathcal{I}},\, \subseteq\rangle \underset{\alpha_{\mathcal{I}\to\mathcal{I}^\sharp}}{\overset{\gamma_{\mathcal{I}^\sharp\to\mathcal{I}}}{\leftrightarrows}} \langle \mathcal{P}_{\mathcal{I}^\sharp},\, \subseteq\rangle$ *is a Galois connection where*

$$\alpha_{\mathcal{I}\to\mathcal{I}^\sharp}(P) \triangleq P \cap \mathcal{P}_{\mathcal{I}^\sharp}$$

$$\gamma_{\mathcal{I}^\sharp\to\mathcal{I}}(Q) \triangleq \left\{ \langle I,\, \eta\rangle \,\middle|\, I \in \mathcal{I} \wedge \eta \in \mathcal{R}_I \wedge \left(I \in \mathcal{I}^\sharp \Rightarrow \langle I,\, \eta\rangle \in Q\right) \right\}$$

Background on abstract interpretation

Abstract domains

$$\langle A, \sqsubseteq, \bot, \top, \sqcup, \sqcap, \triangledown, \vartriangle, \bar{\mathsf{f}}, \bar{\mathsf{b}}, \bar{\mathsf{p}}, \ldots \rangle$$

where

$\overline{P}, \overline{Q}, \ldots \in A$	abstract properties
$\sqsubseteq \; \in A \times A \to \mathcal{B}$	abstract partial order [9]
$\bot, \top \in A$	infimum, supremum $(\forall \overline{P} \in A : \bot \sqsubseteq \overline{P} \sqsubseteq \top)$
$\sqcup, \sqcap, \triangledown, \vartriangle \; \in A \times A \to A$	abstract join, meet, widening, narrowing
$\ldots$	
$\bar{\mathsf{f}} \in (\mathbb{x} \times \mathbb{E}(\mathbb{x}, \mathbb{f}, \mathbb{p})) \to A \to A$	abstract forward assignment transformer
$\bar{\mathsf{b}} \in (\mathbb{x} \times \mathbb{E}(\mathbb{x}, \mathbb{f}, \mathbb{p})) \to A \to A$	abstract backward assignment transformer
$\bar{\mathsf{p}} \in \mathbb{C}(\mathbb{x}, \mathbb{f}, \mathbb{p}) \to A \to A$	abstract condition transformer.

Abstract semantics

- A abstract domain
- $\sqsubseteq$ abstract logical implication
- $\overline{F}[\![\mathrm{P}]\!] \in A \to A$ abstract transformer defined in term of abstract primitives

$\bar{\mathsf{f}} \in (\mathbb{x} \times \mathbb{E}(\mathbb{x}, \mathbb{f}, \mathbb{p})) \to A \to A$	abstract forward assignment transformer
$\bar{\mathsf{b}} \in (\mathbb{x} \times \mathbb{E}(\mathbb{x}, \mathbb{f}, \mathbb{p})) \to A \to A$	abstract backward assignment transformer
$\bar{\mathsf{p}} \in \mathbb{C}(\mathbb{x}, \mathbb{f}, \mathbb{p}) \to A \to A$	abstract condition transformer.

- $\overline{C}[\![\mathrm{P}]\!] \triangleq \{\mathbf{lfp}^{\sqsubseteq}\, \overline{F}[\![\mathrm{P}]\!]\}$ least fixpoint semantics, if any
- $\overline{C}[\![\mathrm{P}]\!] \triangleq \{\overline{P} \mid \overline{F}[\![\mathrm{P}]\!](\overline{P}) \sqsubseteq \overline{P}\}$ or else, post-fixpoint abstract semantics

Concretization

$$\gamma \in A \xrightarrow{\nearrow} \mathcal{P}_{\mathfrak{J}}$$

- Soundness of abstract domains:

$(\overline{P} \sqsubseteq \overline{Q}) \Rightarrow (\gamma(\overline{P}) \subseteq \gamma(\overline{Q}))$	order	$\gamma(\bot) = \emptyset$	infimum
$\gamma(\overline{P} \sqcup \overline{Q}) \supseteq (\gamma(\overline{P}) \cup \gamma(\overline{Q}))$	join	$\gamma(\top) = \top_{\mathfrak{J}}$	supremum
...			

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

$$\gamma[A]$$

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

Soundness and completeness of abstract semantics

- The abstract semantics is sound iff

$$\forall \overline{P} \in A : (\exists \overline{C} \in \overline{C}[\![P]\!] : \overline{C} \sqsubseteq \overline{P}) \Rightarrow (\exists C \in C[\![P]\!] : C \leqslant \gamma(\overline{P}))$$

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

- The abstract semantics is complete iff

$$\forall \overline{P} \in A : (\exists C \in C[\![P]\!] : C \leqslant \gamma(\overline{P})) \Rightarrow (\exists \overline{C} \in \overline{C}[\![P]\!] : \overline{C} \sqsubseteq \overline{P})$$

(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* $C[\![P]\!] \triangleq \mathbf{postfp}^{\leqslant} F[\![P]\!]$, $\overline{C}[\![P]\!] \triangleq \mathbf{postfp}^{\sqsubseteq} \overline{F}[\![P]\!]$ *and* $\gamma : A \to C$ *increasing, then*

$$\forall \overline{P} \in A : F[\![\mathrm{P}]\!] \circ \gamma(\overline{P}) \leqslant \gamma \circ \overline{F}[\![\mathrm{P}]\!](\overline{P})$$

implies that the abstract semantics is sound.

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

$$\begin{aligned}
\gamma(\bar{\mathsf{f}}[\![\mathtt{x} := e]\!]\overline{P}) &\supseteq \mathsf{f}_{\mathfrak{I}}[\![\mathtt{x} := e]\!]\gamma(\overline{P})\\
\gamma(\bar{\mathsf{b}}[\![\mathtt{x} := e]\!]\overline{P}) &\supseteq \mathsf{b}_{\mathfrak{I}}[\![\mathtt{x} := e]\!]\gamma(\overline{P})\\
\gamma(\bar{\mathsf{p}}[\![\varphi]\!]\overline{P}) &\supseteq \mathsf{p}_{\mathfrak{I}}[\![\varphi]\!]\gamma(\overline{P})
\end{aligned}$$

- Compositionality:

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

Best abstraction

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

$$\langle \mathcal{P}_{\mathfrak{I}},\ \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A,\ \sqsubseteq\rangle$$

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

$$\langle \mathcal{P}_{\mathfrak{I}},\ \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\substack{\longleftarrow\\ \twoheadrightarrow}}} \langle A,\ \sqsubseteq\rangle$$

Beyond bounded verification: Widening

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

(a) $\forall x, y \in A : x \sqsubseteq x \triangledown y \wedge y \sqsubseteq x \triangledown y$[14].

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

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

Traditionally a widening *is considered to be both over-approximating and terminating.* □

Iteration with widening

DEFINITION 4.7 (ITERATES WITH WIDENING). *The iterates of a transformer* $\overline{F}[\![P]\!] \in A \mapsto A$ *from the infimum* $\bot \in A$ *with widening* $\triangledown \in A \times A \mapsto A$ *in a poset* $\langle A, \sqsubseteq\rangle$ *are defined by recurrence as* $\overline{F}^0 = \bot$, $\overline{F}^{n+1} = \overline{F}^n$ *when* $\overline{F}[\![P]\!](\overline{F}^n) \sqsubseteq \overline{F}^n$ *and* $\overline{F}^{n+1} = \overline{F}^n \triangledown \overline{F}[\![P]\!](\overline{F}^n)$ *otherwise.* □

THEOREM 4.8 (LIMIT OF THE ITERATES WITH WIDENING). *The iterates in a poset* $\langle A, \sqsubseteq, \bot\rangle$ *of a transformer* $\overline{F}[\![P]\!]$ *from the infimum* $\bot$ *with widening* $\triangledown$ *converge and their limit is a post-fixpoint of the transformer.* □

Implementation notes

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

Multi-interpreted first-order logic

First-order logical formulæ & satisfaction

- Syntax

$\Psi \in \mathbb{F}(\mathbb{x}, \mathbb{f}, \mathbb{p}) \qquad \Psi ::= a \mid \neg\Psi \mid \Psi \wedge \Psi \mid \exists \mathsf{x} : \Psi$ quantified first-order formulæ

a distinguished predicate $= (t_1, t_2)$ which we write $t_1 = t_2$

- Free variables $\vec{\mathsf{x}}_\Psi$
- Satisfaction

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

- Equality

$$I \models_\eta t_1 = t_2 \;\triangleq\; [\![t_1]\!]_I\eta =_I [\![t_2]\!]_I\eta$$

where $=_I$ is the unique reflexive, symmetric, antisymmetric, and transitive relation on $I_{\mathcal{V}}$.

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

$$\gamma^{\mathfrak{a}}_{\mathcal{I}} \in \mathbb{F}(\mathbb{x}, \mathbb{f}, \mathbb{p}) \stackrel{\nearrow}{\longrightarrow} \mathcal{P}_{\mathcal{I}}$$
$$\gamma^{\mathfrak{a}}_{\mathcal{I}}(\Psi) \triangleq \{\langle I, \eta\rangle \mid I \in \mathcal{I} \wedge I \models_\eta \Psi\}$$

- 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)

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

Classical properties of theories

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

Checking satisfiability modulo theory

- Validity modulo theory

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

- Satisfiability modulo theory (SMT)

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

- Checking satisfiability for decidable theories

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

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

- Most SMT solvers support only limited forms of quantified formulæ

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 $\langle A, \sqsubseteq, \mathtt{ff}, \mathtt{tt}, \vee, \wedge, \nabla, \Delta, \bar{f}_{\mathfrak{a}}, \bar{b}_{\mathfrak{a}}, \bar{p}_{\mathfrak{a}}, \ldots\rangle$
- Logical implication $(\Psi \sqsubseteq \Psi') \triangleq ((\forall\, \vec{x}_{\Psi} \cup \vec{x}_{\Psi'} : \Psi \Rightarrow \Psi') \in \mathcal{T})$
- A lattice but in general not complete
- The concretization is

$$\gamma^{\mathfrak{a}}_{\mathcal{T}}(\Psi) \triangleq \left\{\langle I,\ \eta\rangle \;\middle|\; I \in \mathfrak{M}(\mathcal{T}) \wedge I \models_{\eta} \Psi\right\}$$

Logical abstract semantics

- Logical abstract semantics

$$\overline{C}^{\mathfrak{a}}[\![\mathrm{P}]\!] \triangleq \left\{\Psi \;\middle|\; \overline{F}_{\mathfrak{a}}[\![\mathrm{P}]\!](\Psi) \sqsubseteq \Psi\right\}$$

- The logical abstract transformer $\overline{F}_{\mathfrak{a}}[\![\mathrm{P}]\!] \in A \rightarrow A$ is defined in terms of primitives

$\bar{\mathsf{f}}_{\mathfrak{a}} \in (\mathbb{x} \times \mathbb{T}(\mathbb{x},\mathbb{f})) \rightarrow A \rightarrow A$ abstract forward assignment transformer

$\bar{\mathsf{b}}_{\mathfrak{a}} \in (\mathbb{x} \times \mathbb{T}(\mathbb{x},\mathbb{f})) \rightarrow A \rightarrow A$ abstract backward assignment transformer

$\bar{\mathsf{p}}_{\mathfrak{a}} \in \mathbb{L} \rightarrow A \rightarrow A$ condition abstract transformer

Implementation notes ...

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

$\mathsf{f}_{\mathfrak{a}} \in (\mathbb{x} \times \mathbb{T}(\mathbb{x},\mathbb{f})) \rightarrow \mathbb{F}(\mathbb{x},\mathbb{f},\mathbb{p}) \rightarrow \mathbb{F}(\mathbb{x},\mathbb{f},\mathbb{p})$
$\mathsf{f}_{\mathfrak{a}}[\![\mathbf{x} := t]\!]\Psi \triangleq \exists x' : \Psi[\mathbf{x} \leftarrow x'] \wedge \mathbf{x} = t[\mathbf{x} \leftarrow x']$
axiomatic forward assignment transformer

$\mathsf{b}_{\mathfrak{a}} \in (\mathbb{x} \times \mathbb{T}(\mathbb{x},\mathbb{f})) \rightarrow \mathbb{F}(\mathbb{x},\mathbb{f},\mathbb{p}) \rightarrow \mathbb{F}(\mathbb{x},\mathbb{f},\mathbb{p})$
$\mathsf{b}_{\mathfrak{a}}[\![\mathbf{x} := t]\!]\Psi \triangleq \Psi[\mathbf{x} \leftarrow t]$
axiomatic backward assignment transformer

$\mathsf{p}_{\mathfrak{a}} \in \mathbb{C}(\mathbb{x},\mathbb{f},\mathbb{p}) \rightarrow \mathbb{F}(\mathbb{x},\mathbb{f},\mathbb{p}) \rightarrow \mathbb{F}(\mathbb{x},\mathbb{f},\mathbb{p})$
$\mathsf{p}_{\mathfrak{a}}[\![\varphi]\!]\Psi \triangleq \Psi \wedge \varphi$
axiomatic transformer for program test of condition φ.

.../...

but ...

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

$\bar{\mathsf{f}}_{\mathfrak{a}}[\![\mathbf{x} := t]\!]\Psi \triangleq \alpha^{\mathcal{I}}_{A}(\mathsf{f}_{\mathfrak{a}}[\![\mathbf{x} := t]\!]\Psi)$ abstract forward assignment transformer

$\bar{\mathsf{b}}_{\mathfrak{a}}[\![\mathbf{x} := t]\!]\Psi \triangleq \alpha^{\mathcal{I}}_{A}(\mathsf{b}_{\mathfrak{a}}[\![\mathbf{x} := t]\!]\Psi)$ abstract backward assignment transformer

$\bar{\mathsf{p}}_{\mathfrak{a}}[\![\varphi]\!]\Psi \triangleq \alpha^{\mathcal{I}}_{A}(\mathsf{p}_{\mathfrak{a}}[\![\varphi]\!]\Psi)$ abstract transformer for program test of condition

- The abstraction algorithm $\alpha^{\mathcal{I}}_{A} \in \mathbb{F}(\mathbb{x},\mathbb{f},\mathbb{p}) \rightarrow A$ 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 $X \triangledown Y$ 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

Patrick Cousot & Radhia Cousot. Systematic design of program analysis frameworks. In *Conference Record of the Sixth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* pages 269—282, San Antonio, Texas, 1979. ACM Press, New York, U.S.A.

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* $\gamma_i \in A_i \xrightarrow{\nearrow} \mathfrak{P}^{\Sigma^o}_{\mathcal{I}}$. *Their* Cartesian product *is* $\langle\vec{A},\ \vec{\sqsubseteq}\rangle$ *where* $\vec{A} \triangleq \bigtimes_{i\in\Delta} A_i$, $(\vec{P}\ \vec{\sqsubseteq}\ \vec{Q}) \triangleq \bigwedge_{i\in\Delta}(\vec{P}_i \sqsubseteq_i \vec{Q}_i)$ *and* $\vec{\gamma} \in \vec{A} \rightarrow \mathfrak{P}^{\Sigma^o}_{\mathcal{I}}$ *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* $\gamma_i \in A_i \xrightarrow{\nearrow} \mathfrak{P}^{\Sigma^o}_{\mathcal{I}}$ *where* $\vec{A} \triangleq \bigtimes_{i\in\Delta} A_i$ *is their Cartesian product. Their* reduced product *is* $\langle\vec{A}/_{\vec{\equiv}},\ \vec{\sqsubseteq}\rangle$ *where* $(\vec{P}\ \vec{\equiv}\ \vec{Q}) \triangleq (\vec{\gamma}(\vec{P}) = \vec{\gamma}(\vec{Q}))$ *and* $\vec{\gamma}$ *as well as* $\vec{\sqsubseteq}$ *are naturally extended to the equivalence classes* $[\vec{P}]/_{\vec{\equiv}}$, $\vec{P} \in \vec{A}$, *of* $\vec{\equiv}$ *by* $\vec{\gamma}([\vec{P}]/_{\vec{\equiv}}) = \vec{\gamma}(\vec{P})$ *and* $[\vec{P}]/_{\vec{\equiv}}\ \vec{\sqsubseteq}\ [\vec{Q}]/_{\vec{\equiv}} \triangleq \exists\vec{P}' \in [\vec{P}]/_{\vec{\equiv}} : \exists\vec{Q}' \in [\vec{Q}]/_{\vec{\equiv}} : \vec{P}'\ \vec{\sqsubseteq}\ \vec{Q}'$. □

- 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

  ρ(x ∈ [-1,5] ∧ x = 2 mod 4) ≡ x ∈ [2,2] ∧ x = 2 mod 0

  are equivalent

- Meaning-preserving reduction:

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

Implementing the reduced product

- Mathematically, we can choose any representant of the equivalence class (and normalize to this rerpesentant)
- In practice, normalization is hard to do
- It is better to choose a minimal representant

Why is reduction to a minimal representant 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)          ☜ minimal representant

if (x ≤ 0) then
   // false — odd(x)
   // false — false        ☜ minimal representant
```

Iterated reduction

- Definition of iterated reduction:

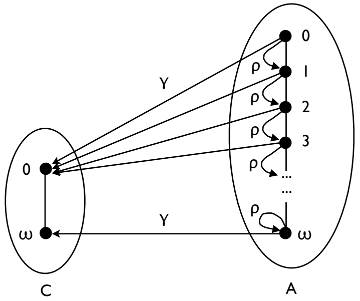
Let $\langle A, \sqsubseteq\rangle$ be a poset which is an abstract domain with concretization $\gamma \in A \xrightarrow{\nearrow} C$ where $\langle C, \subseteq\rangle$ is the concrete domain and $\rho \in A \rightarrow A$ be a meaning-preserving reduction.

The iterates *of the reduction are $\rho^0 \triangleq \lambda \overline{P} \bullet \overline{P}$, $\rho^{\lambda+1} = \rho(\rho^{\lambda})$ 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 $\langle A, \sqsubseteq\rangle$.* □

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 $\gamma_i \in A_i \stackrel{\nearrow}{\longrightarrow} L$ into the concrete domain $\langle L, \leqslant\rangle$.

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

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

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

Pairwise reduction (cont'd)

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

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

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

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 $\forall \vec{P} \in \vec{A} : \forall n \in \mathbb{N}_+ : \vec{\rho}^{\,\star}(\vec{P}) \mathrel{\vec{\sqsubseteq}} \vec{\rho}^{\,n}(\vec{P}) \mathrel{\vec{\sqsubseteq}} \vec{\rho}_{ij}(\vec{P}) \mathrel{\vec{\sqsubseteq}} \vec{P}$, $i, j \in \Delta$, $i \neq j$ and meaning-preserving since $\vec{\rho}^{\,\lambda}(\vec{P}), \vec{\rho}_{ij}(\vec{P}), \vec{P} \in [\vec{P}]/_{\vec{\equiv}}$.

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

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 $\top = \{a, b, c\}$
- $A_2 = \{\emptyset, \{a, b\}, \top\}$
- $A_3 = \{\emptyset, \{a, c\}, \top\}$
- $\langle \top,\ \{a, b\},\ \{a, c\}\rangle/_{\vec{\equiv}} = \langle \{a\},\ \{a, b\},\ \{a, c\}\rangle$
- $\vec{\rho}_{\vec{ij}}(\langle \top,\ \{a, b\},\ \{a, c\}\rangle) = \langle \top,\ \{a, b\},\ \{a, c\}\rangle$
  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 $[\langle \top,\ \{a, b\},\ \{a, c\}\rangle]/_{\vec{\equiv}}$

Nelson-Oppen combination procedure

The Nelson-Oppen combination procedure

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

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

.../...

[22] where $\mathtt{a}, \mathtt{b}$ and $\mathtt{f}$ are in different theories

The Nelson-Oppen combination procedure

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

- 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 componants 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

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

The Nelson-Oppen procedure is an iterated pairwise reduced product

Observables in Abstract Interpretation

- (Relational) abstractions of values $(v_1,...,v_n)$ of program variables $(\mathtt{x}_1,...,\mathtt{x}_n)$ 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 $(v_1,...,v_n)$ of the program variables $(\mathtt{x}_1,...,\mathtt{x}_n)$ assigned to a fresh auxiliary variable $\mathtt{x}_o$

$$\mathtt{x}_o == f(v_1,...,v_n)$$

(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):

  $y = f(x) == f(a+1)$ & $f(x) == f(2*b)$

  becomes

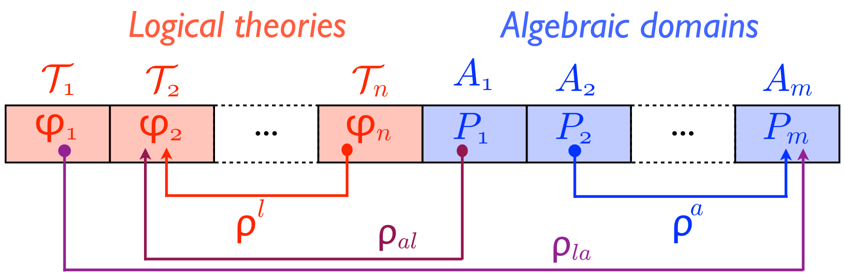
  $z=a+1$; $t=2*b$; $y = f(x) == f(z)$ & $f(x) = f(t)$

Reduction

- The transfer of a (disjunction of) conjunctions of variable (dis-)equalities is a pairwise iterated reduction
- This can be *incomplete* when the signatures are not disjoint

Static analysis combining logical and algebraic abstractions

Reduced product of logical and algebraic domains



- When checking satisfiability of $\varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n$, the Nelson-Oppen procedure generates (dis)-equalities that can be propagated by ρ_{la} to reduce the P_i, $i=1,\ldots,m$, or
- $\alpha_i(\varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n)$ can be propagated by ρ_{la} to reduce the P_i, $i=1,\ldots,m$
- The purification to theory $\mathcal{T}_i$ 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 $\mathcal{T}_i$)

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

$\Longrightarrow$ Extensible expressive static analyzers / verifiers

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

Future work

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

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

The End

Thank You