Unifying proof theoretic/logical and algebraic abstractions for inference and verification

Patrick Cousot

NYU

pcousot@cs.nyu.edu  cs.nyu.edu/~pcousot
Objective
Algebraic abstractions

• Used in abstract interpretation, model-checking,...

• 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

• Several separate abstractions can be combined with 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)

• Various theories can be combined by **Nelson-Oppen procedure**
Objective

• Show that proof-theoretic/logical abstractions are a particular case of algebraic abstractions

• Show that Nelson-Oppen procedure is a particular case of reduced product

• Use this unifying point of view to propose 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 $\langle \mathbf{f}, \mathbf{p} \rangle$)*

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x, y, z, \ldots \in x$</td>
<td>variables</td>
</tr>
<tr>
<td>$a, b, c, \ldots \in f^0$</td>
<td>constants</td>
</tr>
<tr>
<td>$f, g, h, \ldots \in f^n$, $f = \bigcup_{n \geq 0} f^n$</td>
<td>function symbols of arity $n \geq 1$</td>
</tr>
<tr>
<td>$t \in \mathbb{T}(x, f)$</td>
<td>terms</td>
</tr>
<tr>
<td>$t ::= x</td>
<td>c</td>
</tr>
<tr>
<td>$p, q, r, \ldots \in p^n$, $p^0 = { \mathsf{ff}, \mathsf{tt} }$, $p = \bigcup_{n \geq 0} p^n$</td>
<td>predicate symbols of arity $n \geq 0$,</td>
</tr>
<tr>
<td>$a \in A(x, f, p)$</td>
<td>atomic formulae</td>
</tr>
<tr>
<td>$a ::= \mathsf{ff}</td>
<td>p(t_1, \ldots, t_n)</td>
</tr>
<tr>
<td>$e \in E(x, f, p) \triangleq \mathbb{T}(x, f) \cup A(x, f, p)$</td>
<td>program expressions</td>
</tr>
<tr>
<td>$\varphi \in C(x, f, p)$</td>
<td>clauses in simple conjunctive normal form</td>
</tr>
<tr>
<td>$\varphi ::= a</td>
<td>\varphi \land \varphi$</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P, \ldots \in P(x, f, p)$</td>
<td>programs</td>
</tr>
<tr>
<td>$P ::= x := e</td>
<td>\varphi</td>
</tr>
</tbody>
</table>
Programs (interpretation)

- **Interpretation** \( I \in \mathcal{I} \) for a signature \( \langle f, p \rangle \) is such that
  
  - \( I_V \) is a non-empty set of values,
  - \( \forall c \in f^0 : I_V(c) \in I_V \), \( \forall n \geq 1 : \forall f \in f^n : I_V(f) \in I_V^n \rightarrow I_V \),
  - \( \forall n \geq 0 : \forall p \in p^n : I_V(p) \in I_V^n \rightarrow \mathcal{B} \).

- **Environments**

\[ \eta \in \mathcal{R}_I \overset{\triangle}{=} x \rightarrow I_V \text{ environments} \]

- **Expression evaluation**

\[ [a], \eta \in \mathcal{B} \text{ of an atomic formula } a \in \mathcal{A}(x, f, p) \]
\[ [t], \eta \in I_V \text{ of the term } t \in \mathcal{T}(x, f) \]
Programs (concrete semantics)

• The program semantics is usually specified relative to a standard interpretation

• 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)

\[
\begin{align*}
R_\mathcal{I} & \quad \text{concrete observables}^5 \\
P_\mathcal{I} & \triangleq \varphi(R_\mathcal{I}) \quad \text{concrete properties}^6 \\
F_\mathcal{I}[P] & \in P_\mathcal{I} \rightarrow P_\mathcal{I} \quad \text{concrete transformer of program } P \\
C_\mathcal{I}[P] & \triangleq \text{postfp} \subseteq F_\mathcal{I}[P] \in \varphi(P_\mathcal{I}) \quad \text{concrete semantics of program } P
\end{align*}
\]

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

\(^5\)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**
  \[ P \triangleq x=1; \text{while true } \{x=\text{incr}(x)\} \]

- **Arithmetic interpretation**
  \[ \mathcal{I} \text{ on integers } \mathbb{Z} \]

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

  where
  \[ \mathcal{R}_{\mathcal{I}} \triangleq x \rightarrow \mathcal{I}_\mathbb{Z} \] concrete environments
  \[ F_{\mathcal{I}}[P](X) \triangleq \{\eta \in \mathcal{R}_{\mathcal{I}} \mid \eta(x) = 1\} \cup \{\eta[x \leftarrow \eta(x) + 1] \mid \eta \in X\} \]

- **The strongest invariant is**
  \[ \text{lfp} \subseteq F_{\mathcal{I}}[P] = \bigcap \text{postfp} \subseteq F_{\mathcal{I}}[P] \]

- **Expressivity**: the lfp may not be expressible in the abstract in which case we use the set of possible invariants
  \[ C_{\mathcal{I}}[P] \triangleq \text{postfp} \subseteq F_{\mathcal{I}}[P] \]
Concrete domains

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

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

- The **concrete semantics** $C_\mathcal{J}[P] \triangleq \text{postfp} \subseteq F_\mathcal{J}[P]$ defines the system properties of interest for the verification.

- The **transformer** $F_\mathcal{J}[P]$ is defined in terms of primitives, e.g.,

  $$f_\mathcal{J}[x := e]P \triangleq \{ \eta[x \leftarrow [e]_\mathcal{J}\eta] \mid \eta \in P \}$$  
  Floyd’s assignment post-condition

  $$p_\mathcal{J}[\varphi]P \triangleq \{ \eta \in P \mid [\varphi]_\mathcal{J}\eta = \text{true} \}$$  
  test
Extension to multi-interpretations

- Programs have many interpretations \( \mathcal{I} \in \wp(\mathfrak{S}) \).
- Multi-interpreted semantics

\[
\begin{align*}
\mathcal{R}_I &\triangleq I \in \mathcal{I} \not\rightarrow \wp(\mathcal{R}_I) \\
\mathcal{P}_I &\triangleq \wp(\{\langle I, \eta \rangle \mid I \in \mathcal{I} \land \eta \in \mathcal{R}_I\})^8
\end{align*}
\]

\[
\begin{align*}
F_I[P] &\in \mathcal{P}_I \rightarrow \mathcal{P}_I \\
&\triangleq \lambda P \in \mathcal{P}_I \cdot \lambda I \in \mathcal{I} \cdot F_I[P](P(I)) \\
C_I[P] &\in \wp(\mathcal{P}_I) \\
&\triangleq \text{postfp}^\subset F_I[P]
\end{align*}
\]

\[\text{where } \subset \text{ is the pointwise subset ordering.}\]

---

\[8\text{A partial function } f \in A \rightarrow B \text{ with domain } \text{dom}(f) \in \wp(A) \text{ is understood as the relation } \{\langle x, f(x) \rangle \in A \times B \mid x \in \text{dom}(f)\}\text{ and maps } x \in A \text{ to } f(x) \in B, \text{ written } x \in A \not\rightarrow f(x) \in B \text{ or } x \in A \not\rightarrow B_x \text{ when } \forall x \in A : f(x) \in B_x \subseteq B.\]
Algebraic Abstractions
Abstract domains

\[\langle A, \sqsubseteq, \bot, \top, \sqcup, \sqcap, \bigtriangleup, \bar{f}, \bar{b}, \bar{p}, \ldots \rangle\]

where

\[\bar{P}, \bar{Q}, \ldots \in A\]
\[\sqsubseteq \in A \times A \to \mathcal{B}\]
\[\bot, \top \in A\]
\[\sqcup, \sqcap, \bigtriangleup \in A \times A \to A\]
\[\ldots\]
\[\bar{f} \in (x \times \mathcal{E}(x, f, p)) \to A \to A\]
\[\bar{b} \in (x \times \mathcal{E}(x, f, p)) \to A \to A\]
\[\bar{p} \in \mathcal{C}(x, f, p) \to A \to A\]

abstract properties
abstract partial order
infimum, supremum
abstract join, meet, widening, narrowing

abstract forward assignment transformer
abstract backward assignment transformer
abstract condition transformer.
Abstract semantics

- \( A \) abstract domain
- \( \subseteq \) abstract logical implication
- \( \bar{F}[P] \in A \rightarrow A \) abstract transformer defined in terms of abstract primitives
  - \( \bar{f} \in (x \times E(x, f, p)) \rightarrow A \rightarrow A \) abstract forward assignment transformer
  - \( \bar{b} \in (x \times E(x, f, p)) \rightarrow A \rightarrow A \) abstract backward assignment transformer
  - \( \bar{p} \in C(x, f, p) \rightarrow A \rightarrow A \) abstract condition transformer
- \( \bar{C}[P] \triangleq \{ lfp \subseteq \bar{F}[P] \} \) least fixpoint semantics, if any
- \( \bar{C}[P] \triangleq \{ \bar{P} \mid \bar{F}[P](\bar{P}) \subseteq \bar{P} \} \) or else, post-fixpoint abstract semantics
Soundness of the abstract semantics

- **Concretization**
  \[ \gamma \in A \xrightarrow{\gamma} \mathcal{P}_\gamma \]

- **Soundness** of the abstract semantics
  \[ \forall \overline{P} \in A : (\exists \overline{C} \in \overline{C}[P] : \overline{C} \subseteq \overline{P}) \Rightarrow (\exists C \in C[P] : C \subseteq \gamma(\overline{P})) \]

- **Sufficient local soundness conditions:**
  \[ \begin{align*}
  (\overline{P} \subseteq \overline{Q}) & \Rightarrow (\gamma(\overline{P}) \subseteq \gamma(\overline{Q})) & \text{order} \\
  \gamma(\overline{P} \sqcup \overline{Q}) & \supseteq (\gamma(\overline{P}) \cup \gamma(\overline{Q})) & \text{join} \\
  \gamma(\overline{f}[x := e][\overline{P}]) & \supseteq f_{\mathcal{P}}[x := e]\gamma(\overline{P}) & \text{assignment} \text{ post-condition} \\
  \gamma(\overline{b}[x := e][\overline{P}]) & \supseteq b_{\mathcal{P}}[x := e]\gamma(\overline{P}) & \text{assignment} \text{ pre-condition} \\
  \gamma(\overline{p}[\varphi][\overline{P}]) & \supseteq p_{\mathcal{P}}[\varphi]\gamma(\overline{P}) & \text{test} \\
  \end{align*} \]

  implying
  \[ \forall \overline{P} \in A : F [P] \circ \gamma(\overline{P}) \subseteq \gamma \circ F [P](\overline{P}) \]
Beyond bounded verification: Widening

- **Definition of widening:**

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

  \[(a) \quad \forall x, y \in A : x \sqsubseteq x \triangledown y \land y \leq x \triangledown y^{14}.\]

  A terminating widening \( \triangledown \in A \times A \mapsto A \) is such that

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

  Traditionally a widening is considered to be both over-approximating and terminating.\[\square\]
Beyond bounded verification: Widening

- **Iterations 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.

- **Soundness** of iterations 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 \( \langle A, \sqsubseteq, \bot, \top, \sqcup, \sqcap, \bigvee, \bigwedge, \bar{f}, \bar{b}, \bar{p}, \ldots \rangle \)
is implemented separately by hand, by providing a specific computer representation of properties in \( A \)
and algorithms for the logical operations \( \sqsubseteq, \bot, \top, \sqcup, \sqcap, \) and transformers \( \bar{f}, \bar{b}, \bar{p}, \ldots \)

• Different abstract domains are combined into a reduced product

• Very efficient but implemented manually (requires skilled specialists)
First-order logic
First-order logical formulæ & satisfaction

• **Syntax**

$$\Psi \in F(x, f, p) \quad \Psi ::= a \mid \neg \Psi \mid \Psi \land \Psi \mid \exists x : \Psi$$

quantified first-order formulæ

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

• **Free variables** $\bar{x}_\Psi$

• **Satisfaction**

$$I \models_\eta \Psi,$$

interpretation $I$ and an environment $\eta$ satisfy a formula $\Psi$

• **Equality**

$$I \models_\eta t_1 = t_2 \equiv [t_1], \eta =_I [t_2], \eta$$

where $=_I$ is the unique reflexive, symmetric, antisymmetric, and transitive relation on $I.\Psi$. 
Extension to multi-interpretations

• Property described by a formula for multiple interpretations

\[ \mathcal{I} \in \wp(\Theta) \]

• Semantics of first-order formulæ

\[
\gamma^a_{\mathcal{I}} \in \mathbb{F}(x, f, p) \rightarrow \mathcal{P}_{\mathcal{I}} \\
\gamma^a_{\mathcal{I}}(\Psi) \triangleq \{ \langle I, \eta \rangle \mid I \in \mathcal{I} \land I \models_{\eta} \Psi \}
\]

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

\[
M(\mathcal{T}) \triangleq \{ I \in \mathfrak{I} | \forall \Psi \in \mathcal{T} : \exists \eta : I \models_{\eta} \Psi \} \\
= \{ I \in \mathfrak{I} | \forall \Psi \in \mathcal{T} : \forall \eta : I \models_{\eta} \Psi \}
\]
Classical properties of theories

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

- **Deductive theories:** closed by deduction
  \[ \forall \Psi \in \mathcal{T} : \forall \Psi' \in F(x, f, p), \text{if } \Psi \Rightarrow \Psi' \text{ implies } \Psi' \in \mathcal{T} \]

- **Satisfiable theory:**
  \[ M(\mathcal{T}) \neq \emptyset \]

- **Complete theory:**
  for all sentences \( \Psi \) in the language of the theory, either \( \Psi \) is in the theory or \( \neg \Psi \) is in the theory.
Checking satisfiability modulo theory

- **Validity modulo theory**

\[
\text{valid}_T(\Psi) \triangleq \forall I \in \mathcal{M}(T) : \forall \eta : I \models_\eta \Psi
\]

- **Satisfiability modulo theory (SMT)**

\[
\text{satisfiable}_T(\Psi) \triangleq \exists I \in \mathcal{M}(T) : \exists \eta : I \models_\eta \Psi
\]

- **Checking satisfiability** for decidable theories

\[
\text{satisfiable}_T(\Psi) \iff \neg (\text{decide}_T(\forall \overrightarrow{x} \psi : \neg \Psi)) \quad \text{(when } T \text{ is decidable and deductive)}
\]

\[
\text{satisfiable}_T(\Psi) \iff (\text{decide}_T(\exists \overrightarrow{x} \psi : \psi)) \quad \text{(when } T \text{ is decidable and complete)}
\]

- **Most SMT solvers support only quantifier-free formulæ**
Logical Abstractions
Logical abstract domains

- \( \langle A, \mathcal{T} \rangle : A \in \wp(\mathcal{F}(x, f, p)) \) abstract properties
  \( \mathcal{T} \) theory of \( \mathcal{F}(x, f, p) \)

- Abstract domain \( \langle A, \sqsubseteq, \mathbb{F}, \mathtt{tt}, \lor, \land, \neg, f_a, b_a, p_a, \ldots \rangle \)

- 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

\[
\gamma^a_\mathcal{T}(\Psi) \triangleq \{ \langle I, \eta \rangle \mid I \in \mathcal{M}(\mathcal{T}) \land I \models_\eta \Psi \}
\]
Logical abstract semantics

- Logical abstract semantics
  
  \[ \overline{C}^a[P] \triangleq \{ \psi \mid \overline{F}_a[P](\psi) \subseteq \psi \} \]

- The **logical abstract transformer** is defined in terms of primitives

  \[
  \overline{f}_a \in (x \times T(x, f)) \rightarrow A \rightarrow A
  \]

  \[
  \overline{b}_a \in (x \times T(x, f)) \rightarrow A \rightarrow A
  \]

  \[
  \overline{p}_a \in \mathbb{L} \rightarrow A \rightarrow A
  \]

  abstract forward assignment transformer

  abstract backward assignment transformer

  condition abstract transformer
Implementation notes ...

- Universal representation of abstract properties by logical formulæ
- Trival implementations of logical operations \( \& \&, \&\&, \lor, \land, \)
- Provers or SMT solvers can be used for the abstract implication \( \subseteq \)
- Concrete transformers are purely syntactic

\[
\begin{align*}
  f_a \in (x \times T(x, f)) &\rightarrow F(x, f, p) \rightarrow F(x, f, p) \\
  f_a[x := t]\Psi &\triangleq \exists x' : \Psi[x \leftarrow x'] \wedge x = t[x \leftarrow x'] \ \\
  b_a \in (x \times T(x, f)) &\rightarrow F(x, f, p) \rightarrow F(x, f, p) \\
  b_a[x := t]\Psi &\triangleq \Psi[x \leftarrow t] \ \\
  p_a \in C(x, f, p) &\rightarrow F(x, f, p) \rightarrow F(x, f, p) \\
  p_a[\varphi]\Psi &\triangleq \Psi \wedge \varphi
\end{align*}
\]

axiomatic forward assignment transformer

axiomatic backward assignment transformer

axiomatic transformer for program test of condition \( \varphi \).
but ...

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

\[ \tilde{f}_a[x := t] \Psi \triangleq \alpha^T_A(f_a[x := t] \Psi) \]

abstract forward assignment transformer

\[ \overline{b}_a[x := t] \Psi \triangleq \alpha^T_A(b_a[x := t] \Psi) \]

abstract backward assignment transformer

\[ \overline{p}_a[\varphi] \Psi \triangleq \alpha^T_A(p_a[\varphi] \Psi) \]

abstract transformer for program test of condition

- The abstraction algorithm to abstract properties in \( A \) may be non-trivial (e.g. quantifiers elimination)

- A widening \( \triangledown \) is needed to ensure convergence of the fixpoint iterates (or else ask the end-user)
Example I of widening: \textbf{thresholds}

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

Example II of bounded widening: \textbf{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 \( \langle A_i, \sqsubseteq_i \rangle, i \in \Delta, \Delta \text{ finite} \), be abstract domains with increasing concretization \( \gamma_i \in A_i \rightarrow \Psi^\Sigma_I \). Their Cartesian product is \( \langle \tilde{A}, \sqsubseteq \rangle \) where \( \tilde{A} \triangleq \times_{i \in \Delta} A_i \), \( (\tilde{P} \sqsubseteq \tilde{Q}) \triangleq \bigwedge_{i \in \Delta}(\tilde{P}_i \sqsubseteq_i \tilde{Q}_i) \) and \( \tilde{\gamma} \in \tilde{A} \rightarrow \Psi^\Sigma_I \) is \( \tilde{\gamma}(\tilde{P}) \triangleq \bigcap_{i \in \Delta} \gamma_i(\tilde{P}_i) \).
Reduced product

- **Definition of the Reduced product:**

  Let \( \langle A_i, \sqsubseteq_i \rangle, i \in \Delta, \Delta \) finite, be abstract domains with increasing concretization \( \gamma_i \in A_i \rightarrow \Psi_{\sum_0}^\Delta \) where \( \overline{A} \triangleq \prod_{i \in \Delta} A_i \) is their Cartesian product. Their reduced product is \( \langle \overline{A}/\equiv, \sqsubseteq \rangle \) where \( (\overline{P} \equiv \overline{Q}) \triangleq (\overline{\gamma}(\overline{P}) = \overline{\gamma}(\overline{Q})) \) and \( \overline{\gamma} \) as well as \( \sqsubseteq \) are naturally extended to the equivalence classes \( [\overline{P}]/\equiv \), \( \overline{P} \in \overline{A} \), of \( \equiv \) by \( \overline{\gamma}([\overline{P}]/\equiv) = \overline{\gamma}(\overline{P}) \) and \( [\overline{P}]/\equiv \sqsubseteq [\overline{Q}]/\equiv \triangleq \exists \overline{P}' \in [\overline{P}]/\equiv : \exists \overline{Q}' \in [\overline{Q}]/\equiv : \overline{P}' \sqsubseteq \overline{Q}' \). □

- 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

\[
\rho( x \in [-1,5] \land x = 2 \mod 4) \equiv x \in [2,2] \land x = 2 \mod 0
\]

are equivalent

- **Meaning-preserving reduction:**

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

- Definition of iterated reduction:

Let \( \langle A, \subseteq \rangle \) be a poset which is an abstract domain with concretization \( \gamma \in A \rightarrow C \) 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 \( \rho^0 = \lambda \overline{P} \cdot \overline{P} \), \( \rho^{\lambda+1} = \rho(\rho^\lambda) \) for successor ordinals and \( \rho^\lambda = \bigcap_{\beta < \lambda} \rho^\beta \) for limit ordinals.

The iterates are well-defined when the greatest lower bounds \( \prod (\text{glb}) \) do exist in the poset \( \langle A, \subseteq \rangle \). \( \square \)
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 \( \gamma \) preserves glbs.
Pairwise reduction

**Definition of pairwise reduction**

Let \( \langle A_i, \sqsubseteq_i \rangle \) be abstract domains with increasing concretization \( \gamma_i \in A_i \rightarrow L \) 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 \( \tilde{\rho}_{ij} \in \langle \tilde{A}, \sqsubseteq \rangle \mapsto \langle \tilde{A}, \sqsubseteq \rangle \) of the Cartesian product as

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

where \( \tilde{P}[i \leftarrow x]_i = x \) and \( \tilde{P}[i \leftarrow x]_j = \tilde{P}_j \) when \( i \neq j \).

---

24 We define \( (f \times g)(\langle x, y \rangle) \triangleq \langle f(x), g(y) \rangle \).
Define the iterated pairwise reductions $\bar{\rho}^n$, $\bar{\rho}^\lambda$, $\bar{\rho}^*$ $\in \langle \hat{\mathcal{A}}, \hat{\mathcal{C}} \rangle \mapsto \langle \hat{\mathcal{A}}, \hat{\mathcal{C}} \rangle$, $n \geq 0$ of the Cartesian product for

$$\bar{\rho} \triangleq \bigodot_{i,j \in \Delta, i \neq j} \bar{\rho}_{ij}$$

where $\bigodot_{i=1}^n f_i \triangleq f_{\pi_1} \circ \ldots \circ f_{\pi_n}$ is the function composition for some arbitrary permutation $\pi$ of $[1, n]$. \hfill \square
Iterated pairwise reduction

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

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

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

- $L = \emptyset(\{a, b, c\})$
- $A_1 = \emptyset, \{a\}, \top$
- $A_2 = \emptyset, \{a, b\}, \top$
- $A_3 = \emptyset, \{a, c\}, \top$
- $\langle \top, \{a, b\}, \{a, c\}\rangle / \equiv = \langle \{a\}, \{a, b\}, \{a, c\}\rangle$
- $\vec{\rho}_{ij}(\langle \top, \{a, b\}, \{a, c\}\rangle) = \langle \top, \{a, b\}, \{a, c\}\rangle$
  \hspace{1cm} 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 $[\langle \top, \{a, b\}, \{a, c\}\rangle]/\equiv$
Nelson–Oppen combination procedure
The Nelson-Oppen combination procedure

- **Prove** satisfiability in a combination of theories by exchanging equalities and disequalities

- **Example:** \( \varphi \overset{\triangle}{=} (x = a \lor x = b) \land f(x) \neq f(a) \land f(x) \neq f(b) \) \(^{22}\).

- **Purify:** introduce auxiliary variables to separate alien terms and put in conjunctive form

\[
\varphi \overset{\triangle}{=} \varphi_1 \land \varphi_2 \text{ where } \\
\varphi_1 \overset{\triangle}{=} (x = a \lor x = b) \land y = a \land z = b \\
\varphi_2 \overset{\triangle}{=} f(x) \neq f(y) \land f(x) \neq f(z)
\]

\(^{22}\)where \(a, b\) and \(f\) are in different theories
The Nelson-Oppen combination procedure

\[ \varphi \triangleq \varphi_1 \land \varphi_2 \text{ where} \]
\[ \varphi_1 \triangleq (x = a \lor x = b) \land y = a \land z = b \]
\[ \varphi_2 \triangleq f(x) \neq f(y) \land f(x) \neq f(z) \]

- **Reduce** \( \bar{\rho}(\varphi) \): each theory \( T_i \) determines \( E_{ij} \), a (disjunction) of conjunctions of variable (dis)equalities implied by \( \varphi_j \) and propagate it in all other components \( \varphi_i \)

\[ E_{12} \triangleq (x = y) \lor (x = z) \]
\[ E_{21} \triangleq (x \neq y) \land (x \neq z) \]

- **Iterate** \( \bar{\rho}^*(\varphi) \): 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

Program static analysis/verification is **undecidable** so requiring completeness is useless. Therefore the hypotheses can be lifted, the procedure is then sound and incomplete. No change to SMT solvers is needed.
The Nelson-Oppen procedure is an iterated pairwise reduced product
Observables in Abstract Interpretation

• (Relational) **abstracts of values** \((v_1,...,v_n)\) of 
program variables \((x_1,...,x_n)\) 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 
\((x_1,...,x_n)\) assigned to a fresh auxiliary variable \(x_0\)

\[ x_0 == f(v_1,...,v_n) \]

(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):

\[ y = f(x) == f(a+1) \land f(x) == f(2*\text{b}) \]

becomes

\[ z = a+1; t = 2*\text{b}; y = f(x) == f(z) \land f(x) = f(t) \]
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
When checking satisfiability of $\varphi_1 \land \varphi_2 \land ... \land \varphi_n$, the Nelson-Oppen procedure generates (dis)-equalities that can be propagated by $\rho_{la}$ to reduce the $P_i$, $i=1,...,m$, or $\alpha_i(\varphi_1 \land \varphi_2 \land ... \land \varphi_n)$ can be propagated by $\rho_{la}$ to reduce the $P_i$, $i=1,...,m$.

The purification to theory $\mathcal{T}_i$ of $\gamma_i(P_i)$ can be propagated to $\varphi_i$ by $\rho_{al}$ in order to reduce it to $\varphi_i \land \gamma_i(P_i)$ (in $\mathcal{T}_i$).
Advantages

• No need for completeness hypotheses on theories
• Bidirectional reduction between logical and algebraic abstraction
• 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
Future work

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

References


Conclusion

- **Convergence** between logic-based proof-theoretic deductive methods using SMT solvers/theorem provers and algebraic methods using model-checking/abstract interpretation for infinite-state systems

Garrett Birkhoff (1911–1996) abstracted *logic/set theory* into *lattice theory*

The End,

Thank You