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 combinations of algebraic and logical abstractions.

• Joint work with Radhia Cousot, ENS & CNRS, Paris and Laurent Mauborgne, IMDEA, Madrid.

References


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.
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
In addition, in a first-order logic with equality the interpretation of equality is always true. The set $\mathcal{A} = \{a, b, c, \ldots\}$ is the class of all such interpretations.

### 3.2. Mono-Interpreted Concrete Semantics

Note that if the concrete transformer admits a least fixpoint, then it is enough to consider only the formulae that are not in the scope of an existential quantifier. A concrete interpretation $I$ is a program $p$,$\ldots \in \mathcal{P}(x, f, \mathcal{P})$ such that:

- $I_V$ is a non-empty set of values,
- $\forall c \in \mathcal{P}^0 : I_V(c) \subseteq I_V$,
- $\forall n \geq 1 : \forall f \in \mathcal{P}^n : I_V(f) \subseteq I_V$,
- $\forall n \geq 0 : \forall p \in \mathcal{P}^n : I_V(p) \subseteq I_V$.

**Environments** $\eta \in \mathcal{R}_I \models x \rightarrow I_V$ environments

**Expression evaluation** $[a], \eta \in \mathcal{B}$ of an atomic formula $a \in \mathcal{A}(x, f, \mathcal{P})$

$[t], \eta \in I_V$ of the term $t \in \mathcal{T}(x, f)$

### Example of program concrete semantics

- **Program** $p = x = 1; \text{while } \text{true} \{x = \text{incr}(x)\}$
- **Arithmetic interpretation** $\mathcal{S}$ on integers $\mathbb{Z} \models x$.
- **Loop invariant** $\mathsf{Ifp}^C F_3[P] = \{\eta \in \mathcal{R}_3 \mid 0 < \eta(x)\}$,

where $\mathcal{R}_3 \models x \rightarrow \mathcal{S}$ is the concrete observables,

$F_3[P] \in \mathcal{P}_3 \rightarrow \mathcal{P}_3$ is the concrete transformer of program $P$,

$C_3[P] \models \mathsf{postfp}^C F_3[P]$ is the concrete semantics of program $P$.

$^3$Examples of observables are set of states, set of partial or complete execution traces, infinite/transfinite execution trees, etc.

$^4$A property is understood as the set of elements satisfying this property.

$^6$A property is understood as the set of elements satisfying this property.
Concrete domains

- The **standard semantics** describes computations of a system formalized by elements of a domain of observables $R_3$ (e.g. set of traces, states, etc).

- The **properties** $P_3 \triangleq \varphi(R_3)$ (a property is the set of elements with that property) form a complete lattice $\langle P_3, \subseteq, \emptyset, R_3, \cup, \cap \rangle$.

- The concrete semantics $C_3[P] \triangleq \text{postfp}^C F_3[P]$ defines the system properties of interest for the verification.

- The **transformer** $F_3[P]$ is defined in terms of primitives, e.g.,

  $f_3[x := e]P \triangleq \{ \eta[x \leftarrow [e], \eta] : \eta \in P \}$  
  Floyd’s assignment post-condition

  $p_3[\varphi]P \triangleq \{ \eta \in P : [\varphi]\eta = \text{true} \}$  
  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])

\[
\text{lfp}^C F_3[P] = \bigcap \text{postfp}^C F_3[P] \in \text{postfp}^C F_3[P]
\]

Concrete property satisfaction

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

\[
\exists C \in C_3[P] : C \subseteq P
\]

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)
  \[
  \begin{align*}
  x_0 &= 11/2.0; \\
  x_1 &= 61/11.0; \\
  \text{for} \ (i=1; \ i\leq 100; \ i++) \ { \text{do}} \\
  &\quad x_2 = 111 - (1130 - 3000/x_0) / x_1; \\
  x_0 &= x_1; \\
  x_1 &= x_2;
  \end{align*}
  \]
- With exact reals, converges to 6 (repulsive fixpoint)
- With any finite precision, converges to 100 (attractive fixpoint)

\[ \Rightarrow \text{Programs have many interpretations} \quad I \in \varphi(3). \]

Multi-interpreted semantics

- A generalization consists in considering multiple interpretations of logics and programs.
- Multi-interpreted properties:
  \[
  \begin{align*}
  R_I &\triangleq I \in I \in \varphi(R_I), \\
  P_I &\triangleq I \in I \in \varphi(R_I), \\
  \text{interpreted properties for the set of interpretations } I
  \end{align*}
  \]
- Multi-interpreted transformer:
  \[
  F_I[P] \in P_I \rightarrow P_I
  \]
- Multi-interpreted semantics:
  \[
  C_I[P] \in \varphi(P_I)
  \]
  where \( \subseteq \) 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 \(-\infty, +\infty, 0, \text{closest}\))
- ASTRÉE over-approximates all four semantics

Example II of abstraction of a multi-interpreted semantics

- Ignore some interpretations

\[
\begin{align*}
\langle P_I, \varnothing \rangle &\leadsto \langle P_{I^*}, \varnothing \rangle \text{ is a Galois connection where} \\
\alpha_{I \rightarrow I^*}(P) &\triangleq P \cap P_{I^*} \\
\gamma_{I \rightarrow I^*}(Q) &\triangleq \{ (I, \eta) \mid I \in I \land \eta \in R_I \land (I \in I^t \Rightarrow (I, \eta) \in Q) \}
\end{align*}
\]
4.2. Abstract Semantics

Abstract domains

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

where

\[ \bar{P}, \bar{Q}, \ldots \in A \]

\[ \sqsubseteq \in A \times A \to \mathbb{B} \]

\[ \bot, \top \in A \]

\[ \sqcup, \sqcap, \Delta \in A \times A \to A \]

\[ \bar{f} \in (x \times E(x, f, p)) \to A \to A \]

\[ \bar{b} \in (x \times E(x, f, p)) \to A \to A \]

\[ \bar{p} \in C(x, f, p) \to A \to A \]

abstract properties

abstract partial order

infimum, supremum (\( \forall \bar{P} \in A : \bot \sqsubseteq \bar{P} \sqsubseteq \top \))

abstract join, meet, widening, narrowing

abstract forward assignment transformer

abstract backward assignment transformer

abstract condition transformer.

4.3. Soundness of Abstract Domains

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

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

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

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

4.4. Soundness of Abstract Semantics

- The abstract semantics is sound iff

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

- The abstract semantics is complete iff

\[ \forall \bar{P} \in A : (\exists C \in C[P] : C \sqsubseteq \gamma(\bar{P})) \implies (\exists \bar{C} \in \bar{C}[P] : \bar{C} \sqsubseteq \bar{P}) \]
**Sufficient soundness condition**

- **Theorem 4.4 (Soundness of an abstract post-fixpoint semantics).** If $C[P] \leq \text{postfix}_S F[P]$, $\overline{C}[P] \leq \text{postfix}_S F[P]$ and $\gamma : A \rightarrow C$ increasing, then

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

implies that the abstract semantics is sound.

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

$$\gamma(f[x := e][P]) \supseteq f_g[x := e]\gamma(P)$$

$$\gamma(b[x := e][P]) \supseteq b_g[x := e]\gamma(P)$$

$$\gamma(\mathcal{b}[\varphi][P]) \supseteq p_g[\varphi]\gamma(P)$$

- **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 $(A, \sqsubseteq)$ be a poset. Then an over-approximating widening $\forall \in A \times A 
\rightarrow A$ is such that

(a) $\forall x, y \in A : x \sqsubseteq y \land y \sqsubseteq x \Rightarrow y \sqsubseteq y'$.

A terminating widening $\forall \in A \times A 
\rightarrow A$ is such that

(a) Given any sequence $(\gamma^k, n \geq 0)$, the sequence $\gamma^0 = x^0, \ldots, \gamma^{n+1} = y^n \sqsubseteq x^n, \ldots$ converges (i.e. $\exists i \in N : \forall n \geq l : \gamma^n = \gamma^i$ in which case $\gamma^i$ is called the limit of the widened sequence $(\gamma^n, n \geq 0)$).

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 $\overline{F}[P] \in A \rightarrow A$ that begins in an infimum $\bot \in A$ with widening $\forall \in A \times A 
\rightarrow A$ in a poset $(A, \sqsubseteq)$ are defined by recurrence as:

$$\bot = \bot, \overline{F}^{i+1} = \overline{F}^i \land \overline{F}^{i+1} = \overline{F} \lor \overline{F}^{i+1} = \overline{F} \lor \overline{F}[P](\overline{F})$$

otherwise.

□

**Theorem 4.8 (Limit of the iterates with widening).** The iterates in a poset $(A, \sqsubseteq, \bot)$ of a transformer $\overline{F}[P]$ from the infimum $\bot$ with widening $\forall$ converge and their limit is a post-fixpoint of the transformer. □

---

**Best abstraction**

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

$$\langle \mathcal{P}_J, \sqsubseteq \rangle \triangleleft \gamma \alpha \rightarrow \langle A, \sqsubseteq \rangle$$

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

$$\langle \mathcal{P}_J, \sqsubseteq \rangle \triangleleft \gamma \alpha \rightarrow \langle A, \sqsubseteq \rangle$$
Implementation notes

- Each abstract domain \( \langle A, \subseteq, \sqcap, \sqcup, \land, \lor, \forall, \Delta, f, \bar{f}, \bar{p}, \ldots \rangle \) is implemented separately, by providing a specific computer representation of properties in \( A \), and algorithms for the logical operations \( \subseteq, \sqcap, \sqcup, \land, \lor, \forall \), and transformers \( \bar{f}, \bar{p}, \ldots \).
- Different abstract domains are combined into a reduced product.
- Very efficient but requires skilled experts.

First-order logical formulae & satisfaction

- Syntax
  \[ \Psi \in F(\exists, f, p) \quad \Psi ::= a \mid \neg \Psi \mid \Psi \land \Psi \mid \exists x : \Psi \]
  quantified first-order formulae
  a distinguished predicate \( (t_1, t_2) \) which we write \( t_1 \equiv t_2 \).
- Free variables \( \bar{x}_\Psi \)
- Satisfaction
  \[ I \models_\eta \Psi \]
  interpretation \( I \) and an environment \( \eta \) satisfy a formula \( \Psi \)
- Equality
  \[ \eta \models t_1 = t_2 \iff [t_1], \eta = [t_2], \eta \]
  where \( =_I \) is the unique reflexive, symmetric, antisymmetric, and transitive relation on \( I_V \).

Multi-interpreted first-order logic

- A property is described by a formula for multiple interpretations
  \[ I \in \wp(\exists) \]
- Semantics of first-order formulae
  \[ \psi^a_I \in F(\exists, f, p) \xrightarrow{\_} \mathcal{P}_I \]
  \[ \psi^a_I(\Psi) \triangleq \{ \langle I, \eta \rangle \mid I \in I \wedge I \models_\eta \Psi \} \]
- But how are we going to describe sets of interpretations \( I \in \wp(\exists) \)?
Defining multiple interpretations as models of theories

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

$$M(T) \triangleq \{ I \in \mathfrak{I} \mid \forall \Psi \in T : \exists \eta : I \models \eta \Psi \}$$

$$= \{ I \in \mathfrak{I} \mid \forall \Psi \in T : \forall \eta : I \models \eta \Psi \}$$

Classical properties of theories

- **Decidable theories**: $\forall \Psi \in \mathbb{F}(x,f,p) : \text{decide}_T(\Psi) \triangleq (\Psi \in T)$ is computable
- **Deductive theories**: closed by deduction

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

- **Satisfiable theory**:

$$M(T) \neq \emptyset$$

- **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**

$$\text{valid}_T(\Psi) \triangleq \forall I \in M(T) : \forall \eta : I \models \eta \Psi$$

- **Satisfiability modulo theory (SMT)**

$$\text{satisfiable}_T(\Psi) \triangleq \exists I \in M(T) : \exists \eta : I \models \eta \Psi$$

- **Checking satisfiability** for decidable theories

$$\text{satisfiable}_T(\Psi) \iff \neg (\text{decide}_T(\forall \xi \eta : \neg \Psi)) \quad \text{(when } T \text{ is decidable and deductive)}$$

$$\text{satisfiable}_T(\Psi) \iff (\text{decide}_T(\exists \xi \eta : \Psi)) \quad \text{(when } T \text{ is decidable and complete)}$$

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

Example of abstraction: Logical abstractions
Logical abstract domains

- $\langle A, \mathcal{T} \rangle : A \in \phi(\mathbb{F}(x, f, p))$ abstract properties
  - $\mathcal{T}$ theory of $\mathbb{F}(x, f, p)$
- Abstract domain $\langle A, \subseteq, \mathbb{F}, \mathbb{f}, \mathbb{t}, \mathbb{v}, \land, \lor, \Delta, \bar{t}_a, \bar{b}_a, \bar{p}_a, \ldots \rangle$
- Logical implication $(\Psi \subseteq \Psi') \iff ((\forall \bar{x} \Psi \cup \bar{x} \Psi' : \Psi \Rightarrow \Psi') \in \mathcal{T})$
- A lattice but in general not complete
- The concretization is
  $$\gamma^a_{\mathcal{T}}(\Psi) \triangleq \{ \langle I, \eta \rangle \mid I \in \mathfrak{M}(\mathcal{T}) \land I \models_{\eta} \Psi \}$$

Logical abstract semantics

- Logical abstract semantics $\overline{\mathcal{C}}^a[P] \triangleq \{ \Psi \mid \overline{F}_a[P](\Psi) \subseteq \Psi \}$
- The logical abstract transformer $\overline{F}_a[P] : A \rightarrow A$ is defined in terms of primitives
  - $\bar{t}_a \in (x \times \mathbb{T}(x, f)) \rightarrow A \rightarrow A$ abstract forward assignment transformer
  - $\bar{b}_a \in (x \times \mathbb{T}(x, f)) \rightarrow A \rightarrow A$ abstract backward assignment transformer
  - $\bar{p}_a \in \mathbb{L} \rightarrow A \rightarrow A$ condition abstract transformer

Implementation notes ...

- Universal representation of abstract properties by logical formulæ
- Trivial implementations of logical operations $\mathbb{f}, \mathbb{t}, \mathbb{v}, \land, \lor$
- Provers or SMT solvers can be used for the abstract implication $\subseteq$
- Concrete transformers are purely syntactic

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

- $\bar{t}_a[x := l] \Psi \triangleq \alpha^f_A(t_a[x := l] \Psi)$ abstract forward assignment transformer
- $\bar{b}_a[x := l] \Psi \triangleq \alpha^b_A(b_a[x := l] \Psi)$ abstract backward assignment transformer
- $\bar{p}_a[\bar{v}] \Psi \triangleq \alpha^c_A(p_a[\bar{v}] \Psi)$ abstract transformer for program test of condition

- The abstraction algorithm $\alpha^f_A \in \mathbb{F}(x, f, p) \rightarrow A$ to abstract properties in $A$ may be non-trivial (e.g. quantifiers elimination)
- A widening $\bigvee$ 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 $\subseteq$.
- Define $X \uparrow Y$ 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.

Cartesian product

- Definition of the Cartesian product:

  Let $\langle A_i, \sqsubseteq_i \rangle$, $i \in \Delta$, $\Delta$ finite, be abstract domains with increasing concretization $\gamma_i \in A_i \rightarrow \Psi_i^{\Sigma_0}$. Their Cartesian product is $\langle \mathcal{A}, \mathcal{E} \rangle$ where $\mathcal{A} = \tau_{i \in \Delta} A_i$, $(\mathcal{P}, \mathcal{Q}) \sqsubseteq (\mathcal{P'}, \mathcal{Q'})$ if $\gamma_i(\mathcal{P}) \sqsubseteq_i \gamma_i(\mathcal{Q})$ and $\gamma \in \mathcal{A} \rightarrow \Psi_i^{\Sigma_0}$ is $\gamma(\mathcal{P}) \sqsubseteq \bigwedge_{i \in \Delta} \gamma_i(\mathcal{P}_i)$.

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 $\gamma_i \in A_i \rightarrow \Psi_i^{\Sigma_0}$ where $\mathcal{A} = \tau_{i \in \Delta} A_i$ is their Cartesian product. Their reduced product is $\langle \mathcal{A}' / \mathcal{G}, \mathcal{E} \rangle$ where $(\mathcal{P}, \mathcal{Q}) \sqsubseteq (\mathcal{P'}, \mathcal{Q'})$ if $\gamma(\mathcal{P}) \equiv \gamma(\mathcal{Q})$ and $\mathcal{P}' \sqsubseteq \mathcal{Q}'$ as well as $\mathcal{G}$ are naturally extended to the equivalence classes $[\mathcal{P}] / \mathcal{G}$, $\mathcal{P} \in \mathcal{A}$, of $\mathcal{G}$ by $\gamma([\mathcal{P}] / \mathcal{G}) = \gamma(\mathcal{P})$ and $[\mathcal{P}] / \mathcal{G} \sqsubseteq [\mathcal{Q}] / \mathcal{G} \Leftrightarrow \exists \mathcal{P}' : [\mathcal{P}] / \mathcal{G} : \exists \mathcal{Q}' : [\mathcal{Q}] / \mathcal{G} : \mathcal{P}' \sqsubseteq \mathcal{Q}'$.

- 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 × congruences
  \[ \rho( x \in [-1,5] \land x = 2 \mod 4) = x \in [2,2] \land x = 2 \mod 0 \]
  are equivalent
- **Meaning-preserving reduction:**

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

---

Why is reduction to a minimal representant important?

- **Without reduction** (signs and parity):
  \[
  \begin{align*}
  \text{if } (x \leq 0) & \text{ then } \\
  \text{ } & \text{// } x = = 0 - \text{odd}(x)
  \end{align*}
  \]
- **With reduction:**
  \[
  \begin{align*}
  \text{if } (x \leq 0) & \text{ then } \\
  \text{ } & \text{// } \text{false} - \text{odd}(x) \\
  \text{ } & \text{// } \text{false} - \text{false}
  \end{align*}
  \]

---

Implementing the reduced product

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

---

Iterated reduction

- **Definition of iterated reduction:**

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

  The iterates are well-defined when the greatest lower bounds \( \bigcap (\text{glb}) \) 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 \( \gamma_i \in A_i \to L \) into the concrete domain \( \langle L, \subseteq \rangle \).

  For \( i, j \in \Delta, i \neq j \), let \( \rho_{ij} \in \langle A_i \times A_j, \sqsubseteq_{ij} \rangle \) 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 \( (\gamma_i \times \gamma_j) \circ \rho_{ij} = (\gamma_i \times \gamma_j) \)).

  Define the pairwise reductions \( \bar{\rho}_{ij} \in \langle \bar{A}, \bar{\sqsubseteq} \rangle \) of the Cartesian product as

  \[
  \bar{\rho}_{ij}(\bar{P}) \triangleq \langle \bar{P}^i, \bar{P}^j \rangle \triangleq \rho_{ij}(\langle P^i, P^j \rangle) \text{ in } \bar{P}[i \leftarrow \bar{P}^i][j \leftarrow \bar{P}^j]
  \]

  where \( \bar{P}[i \leftarrow x]^i = x \) and \( \bar{P}[i \leftarrow x]^j = P^j \) when \( i \neq j \).

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

Pairwise reduction (cont’d)

Define the iterated pairwise reductions \( \bar{\rho}^n, \bar{\rho}^A, \bar{\rho}^* \in \langle \bar{A}, \bar{\sqsubseteq} \rangle \) \( \bar{\rho}^n \triangleq \bigcirc_{i,j \in \Delta} \bar{\rho}_{ij} \) where \( \bigcirc f_i \triangleq f_{\pi_1} \circ \ldots \circ f_{\pi_n} \) is the function composition for some arbitrary permutation \( \pi \in [1, n] \).

Iterated pairwise reduction

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

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

  If, moreover, γ preserves greatest lower bounds then \( \bar{\rho}^*(\bar{P}) \in [\bar{P}]_{\bar{\subseteq}} \).
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

- \( L = \mathcal{V}([a, b, c]) \)
- \( A_1 = \{\emptyset, \{a\}, T\} \)
- \( A_2 = \{\emptyset, \{a, b\}, T\} \)
- \( A_3 = \{\emptyset, \{a, c\}, T\} \)
- \( \langle T, \{a, b\}, \{a, c\} \rangle \) preserves lower bounds then
- \( \vec{P}_{ij} \langle \Delta, \{a, b\}, \{a, c\} \rangle = \langle \Delta, \{a, b\}, \{a, c\} \rangle \)
- for \( \Delta = \{1, 2, 3\}, i,j \in \Delta, i \neq j \)
- \( \vec{P}^* \langle \Delta, \{a, b\}, \{a, c\} \rangle = \langle \Delta, \{a, b\}, \{a, c\} \rangle \) is not a minimal element of \( \langle \Delta, \{a, b\}, \{a, c\} \rangle \).

Nelson-Oppen combination procedure

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

\[
\begin{align*}
\varphi &= \varphi_1 \land \varphi_2 \\
\varphi_1 &= (x = a \lor x = b) \land y = a \lor z = b \\
\varphi_2 &= f(x) = f(y) \land f(x) = f(z)
\end{align*}
\]

\( \ldots/\ldots \)
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.


The Nelson-Oppen procedure is an iterated pairwise reduced product
Observables in Abstract Interpretation

- (Relational) abstractions of values \((v_1,\ldots,v_n)\) of program variables \((x_1,\ldots,x_n)\) 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 \((v_1,\ldots,v_n)\) of the program variables \((x_1,\ldots,x_n)\) assigned to a fresh auxiliary variable \(x_o\) (with a precise abstraction of \(f\))

\[
x_o = f(v_1,\ldots,v_n)
\]

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

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

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 $\varphi_1 \land \varphi_2 \land ... \land \varphi_n$, 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 $T_i$ 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 $T_i$).}

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 $\implies$ 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