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

Patrick Cousot
NYU
pcousot@cs.nyu.edu  cs.nyu.edu/~pcousot

Algebraic abstractions
- Used in abstract interpretation, model-checking, etc.
- 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 formulas 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
• 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

Objective

Expressions (on a signature \((\mathcal{F}, \mathcal{P})\))

- \(x, y, z, \ldots \in \mathcal{X}\)
- \(a, b, c, \ldots \in \mathcal{F}^0\)
- \(f, g, h, \ldots \in \mathcal{F}^r\), \(f \in \bigcup_{n=0}^\infty \mathcal{F}^n\)
- \(t \in T(x, f)\), \(t := x \mid f(t_1, \ldots, t_n)\)
- \(p, q, r, \ldots \in \mathcal{P}^n\), \(p \in \bigcup_{n=0}^\infty \mathcal{P}^n\)
- \(a \in A(x, f, p)\)
- \(\varepsilon \in E(x, f, p)\)
- \(\varphi \in C(x, f, p)\)

Variables

Constants

Function symbols of arity \(n \geq 1\)

Terms

Predicate symbols of arity \(n \geq 0\)

Atomic formulæ

Program expressions

Clauses in simple conjunctive normal form

Programs (syntax)

Programs (interpretation)

Interpretation \(I \in \mathfrak{I}\) for a signature \((\mathcal{F}, \mathcal{P})\) is \((I_V, I_Y)\) such that

- \(I_V\) is a non-empty set of values,
- \(\forall c \in \mathcal{F}^0 : I_Y(c) \in I_V\), \(\forall n \geq 1 : \forall f \in \mathcal{F}^n : I_Y(f) \in I_V^n \rightarrow I_V\),
- \(\forall n \geq 0 : \forall p \in \mathcal{P}^n : I_Y(p) \in I_V^n \rightarrow \mathcal{B}\), where \(\mathcal{B} \doteq \{\text{false}, \text{true}\}\)

Environments

\(\eta \in \mathfrak{R}_I \doteq x \rightarrow I_V\)

Expression evaluation

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

- The program semantics is usually specified relative to a standard interpretation $\mathfrak{I} \in \mathfrak{S}$.
- 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).

\[\mathcal{R}_3\] concrete observables

\[\mathcal{P}_3 \doteq \varphi(\mathcal{R}_3)\] concrete properties

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

\[C_3[P] \doteq \text{postfp} F_3[P] \in \varphi(\mathcal{P}_3)\] concrete semantics of program $P$

where postfp $f \doteq \{ x \mid f(x) \leq x \}$

Concrete domains

- The standard semantics describes computations of a system formalized by elements of a domain of observables $\mathcal{R}_3$ (e.g., set of traces, states, etc).
- The properties $\mathcal{P}_3 \doteq \varphi(\mathcal{R}_3)$ (a property is the set of elements with that property) form a complete lattice $\langle \mathcal{P}_3, \subseteq, \emptyset, \mathcal{R}_3, \cup, \cap \rangle$.
- The concrete semantics $C_3[P] \doteq \text{postfp} F_3[P]$ defines the system properties of interest for the verification.
- The transformer $F_3[P]$ is defined in terms of primitives, e.g.

\[f_3[x := e] P \doteq \{ \eta[x \leftarrow [e]_\eta] \mid \eta \in P \} \] Floyd’s assignment post-condition

Example of program concrete semantics

- Program $P \doteq x=1; \text{while true} \{x \leftarrow \text{incr}(x)\}$
- Arithmetic interpretation $\mathfrak{I}$ on integers $\mathfrak{I}_V = \mathbb{Z}$
- Loop invariant $\text{Ifp} F_3[P] = \{ \eta \in \mathcal{R}_3 \mid 0 < \eta(x) \}$

where $\mathcal{R}_3 \doteq \mathfrak{I} \rightarrow \mathfrak{S}_V$ concrete environments

$F_3[P](X) \doteq \{ \eta \in \mathcal{R}_3 \mid \eta(x) = 1 \} \cup \{ \eta[x \leftarrow \eta(x) + 1] \mid \eta \in X \}$

- The strongest invariant is $\text{Ifp} F_3[P] = \bigcap \text{postfp} C_3[P]$.
- Expressivity: the Ifp may not be expressible in the abstract in which case we use the set of possible invariants $C_3[P] \doteq \text{postfp} F_3[P]$.

Multi-interpreted semantics

- Programs have many interpretations $I \in \varphi(\mathfrak{S})$.
- Multi-interpreted concrete semantics

\[\mathcal{R}_I \doteq I \in I \implies \varphi(\mathcal{R}_I)\]

multi-interpreted concrete transformer of program $P$

\[C_I[P] \doteq \varphi(\mathcal{P}_I)\]

multi-interpreted concrete semantics

where $\subseteq$ is the pointwise subset ordering.

Extension to multi-interpretations

- Program observables for interpretation $I \in I$ interpreted properties for the set of interpretations $I$.
Abstract domains

where

\(\overline{p}, \overline{q}, \ldots \in A\)

\(\subseteq \in A \times A \rightarrow B\)

\(\bot, \top \in A\)

\(\sqcup, \sqcap, \forall, \Delta, f, b, \ldots\) abstract properties

\(\text{abstract partial order}^9\)

\(\text{infimum, supremum}\)

\(\text{abstract join, meet, widening, narrowing}\)

\(\overline{f} \in (x \times E(x, f, p)) \rightarrow A \rightarrow A\)

\(\overline{b} \in (x \times E(x, f, p)) \rightarrow A \rightarrow A\)

\(\overline{p} \in C(x, f, p) \rightarrow A \rightarrow A\)

abstract forward assignment transformer

abstract backward assignment transformer

abstract condition transformer.

Abstract semantics

- **\(A\)** abstract domain
- **\(\subseteq\)** abstract logical implication
- **\(\overline{F}[P] \in A \rightarrow A\)** abstract transformer defined in term of abstract primitives
- **\(\overline{C}[P] \triangleq \{lfp(\overline{F}[P])\}\)** least fixpoint semantics, if any
- **\(\overline{C}[P] \triangleq \{\overline{P} | \overline{F}[P] \subseteq \overline{P}\}\)** or else, post-fixpoint abstract semantics

Soundness of the abstract semantics

- Concretization

\(\gamma \in A \hookrightarrow P_3\)

- **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:**

\(\overline{f} \in (x \times E(x, f, p)) \rightarrow A \rightarrow A\)

\(\overline{b} \in (x \times E(x, f, p)) \rightarrow A \rightarrow A\)

\(\overline{p} \in C(x, f, p) \rightarrow A \rightarrow A\)

\(\text{assignment post-condition}\)

\(\text{assignment pre-condition}\)

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

- **order** \(\gamma(\bot) = 0\)

- **supremum** \(\gamma(\top) = T_3\)
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

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

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

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

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

---

Implementation notes

- Each abstract domain \( \langle A, \sqsubseteq, \bot, \top, \sqcap, \sqcup, \triangleleft, \triangleright, \trianglerighteq, \bowtie \rangle \) is implemented separately by hand, by providing a specific computer representation of properties in \( A \), and algorithms for the logical operations \( \sqsubseteq, \bot, \top, \sqcap, \sqcup \), and transformers \( \triangleleft, \triangleright, \bowtie \).

- Different abstract domains are combined into a reduced product.

- Very efficient but implemented manually (requires skilled specialists)

---

First-order logic
First-order logical formulae & 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 formulae
  a distinguished predicate \( = (t_1, t_2) \) which we write \( t_1 = t_2 \).

- **Free variables** \( \vec{x} \)

- **Satisfaction**
  \[ I \models \eta \Psi, \]
  interpretation \( I \) and an environment \( \eta \) satisfy a formula \( \Psi \)

- **Equality**
  \[ I \models \eta t_1 = t_2 \iff [t_1], \eta = [t_2], \eta \]

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

Extension to multi-interpretations

- **Property described by a formula for multiple interpretations**
  \[ I \in \varphi(\mathfrak{I}) \]

- **Semantics of first-order formulae**
  \[ \gamma^I_\eta (\Psi) \equiv \{ \langle I, \eta \rangle \mid I \in I \land I \models \eta \Psi \} \]

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

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

  \[ = \{ I \in \mathfrak{I} \mid \forall \Psi \in \mathcal{T} : \forall \eta : I \models \eta \Psi \} \]

Classical properties of theories

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

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

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

- **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
  \[ \text{valid}_T(\Psi) \equiv \forall I \in \mathcal{M}(T) : \forall \eta : I \models_\eta \Psi \]
- Satisfiability modulo theory (SMT)
  \[ \text{satisfiable}_T(\Psi) \equiv \exists I \in \mathcal{M}(T) : \exists \eta : I \models_\eta \Psi \]
- Checking satisfiability for decidable theories
  \[ \text{satisfiable}_T(\Psi) \Leftrightarrow (\text{decide}_T(\forall x \phi : \neg \Psi)) \] (when \(T\) is decidable and deductive)
  \[ \text{satisfiable}_T(\Psi) \Leftrightarrow (\text{decide}_T(\exists x \phi : \Psi)) \] (when \(T\) is decidable and complete)
- Most SMT solvers support only quantifier-free formulae

Logical Abstract Domains

- \(\langle A, \mathcal{T} \rangle : A \in \varphi(\mathbb{F}(x, f, p))\) abstract properties
  \[ \mathcal{T} : \text{theory of } \mathbb{F}(x, f, p) \]
- Abstract domain \(\langle A, \sqsubseteq, \sqcup, \sqcap, \land, \lor, \top, \bot, \bar{a}, \overline{p}, \ldots \rangle\)
- Logical implication \(\langle \Psi \sqsubseteq \Psi' \rangle \equiv (\forall \bar{x}_\Psi \cup \bar{x}_\Psi : \Psi \Rightarrow \Psi' \in \mathcal{T})\)
- A lattice but in general not complete
- The concretization is
  \[ \gamma^a_T(\Psi) \equiv \{ \langle I, \eta \rangle | I \in \mathcal{M}(\mathcal{T}) \land I \models_\eta \Psi \} \]

Logical Abstractions

Logical abstract semantics

- Logical abstract semantics
  \[ \overline{C}^a[P] \equiv \left\{ \Psi | \overline{F}_a[P](\Psi) \subseteq \Psi \right\} \]
- The logical abstract transformer \(\overline{F}_a[a] : A \to A\) is defined in terms of primitives
  \[ \bar{t}_a \in (x \times \mathcal{T}(x, f)) \to A \to A \] abstract forward assignment transformer
  \[ \bar{b}_a \in (x \times \mathcal{T}(x, f)) \to A \to A \] abstract backward assignment transformer
  \[ \bar{p}_a \in \mathbb{L} \to A \to A \] condition abstract transformer
Implementation notes ...

- Universal representation of abstract properties by logical formulae
- Trivial implementations of logical operations $\&$, $\lor$, $\exists$, $\forall$
- Provers or SMT solvers can be used for the abstract implication $\models$
- Concrete transformers are purely syntactic

$$
\begin{align*}
\alpha^f_A &\in F(x, f, p) \rightarrow A \\
\alpha^b_A &\in C(x, f, p) \rightarrow F(x, f, p) \rightarrow F(x, f, p)
\end{align*}
$$

Example I of widening: thresholds

- Choose a subset $W$ of $A$ satisfying the ascending chain condition for $\subseteq$
- Define $x \lor y$ 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

but ...

$$
\begin{align*}
\bar{t}_x &\in (x \times T(x, f)) \rightarrow F(x, f, p) \rightarrow F(x, f, p) \\
\bar{b}_x &\in (x \times T(x, f)) \rightarrow F(x, f, p) \rightarrow F(x, f, p)
\end{align*}
$$

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

$$
\begin{align*}
\bar{t}_x[x := i] \Psi &\equiv \alpha^f_A(\bar{t}_x[x := i]\Psi) \\
\bar{b}_x[x := i] \Psi &\equiv \alpha^b_A(\bar{b}_x[x := i]\Psi)
\end{align*}
$$

- The abstraction algorithm $\alpha^f_A \in F(x, f, p) \rightarrow A$ to abstract properties in $A$ may be non-trivial (e.g. quantifiers elimination)
- A widening $\lor$ is needed to ensure convergence of the fixpoint iterates (or else ask the end-user)
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 \( \gamma_i \in A_i \xrightarrow{\sqsubseteq_i} \Psi_i^{\Sigma_0} \). Their Cartesian product is \( \langle \vec{A}, \sqsubseteq \rangle \) where \( \vec{A} \equiv \times_{i \in \Delta} A_i \), \( \vec{P} \sqsubseteq \vec{Q} \equiv \bigwedge_{i \in \Delta} (P_i \sqsubseteq_i Q_i) \) and \( \vec{\gamma} \in \vec{A} \rightarrow \Psi_i^{\Sigma_0} \) is \( \vec{\gamma}(\vec{P}) \equiv \bigcap_{i \in \Delta} \gamma_i(P_i) \).

Reduction

- Example: intervals x 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 \( \langle C, \sqsubseteq \rangle \) is the concrete domain. A meaning-preserving map is \( \rho \in A \rightarrow A \) such that \( \forall \vec{P} \in A : \gamma(\rho(\vec{P})) = \gamma(\vec{P}) \). The map is a reduction if and only if it is reductive that is \( \forall \vec{P} \in A : \rho(\vec{P}) \sqsubseteq \vec{P} \).

Reduced product

- Definition of the Reduced product:

Let \( \langle A_i, \sqsubseteq_i \rangle, \ i \in \Delta, \Delta \text{ finite}, \) be abstract domains with increasing concretization \( \gamma_i \in A_i \xrightarrow{\sqsubseteq_i} \Psi_i^{\Sigma_0} \) where \( \vec{A} \equiv \times_{i \in \Delta} A_i \) is their Cartesian product. Their reduced product is \( \langle \vec{A}/\equiv, \sqsubseteq \rangle \) where \( \vec{P} \equiv \vec{Q} \equiv (\vec{\gamma}(\vec{P}) = \vec{\gamma}(\vec{Q})) \) and \( \vec{\gamma} \) as well as \( \vec{\equiv} \) are naturally extended to the equivalence classes \( [\vec{P}]/\equiv, [\vec{P}] \equiv [\vec{Q}] \equiv [\vec{P}]/\equiv \equiv \exists \vec{P}' \in [\vec{P}] /\equiv : \exists \vec{Q}' \in [\vec{Q}] /\equiv : \vec{P}' \sqsubseteq \vec{Q}' \).

- 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.

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 \( \langle C, \sqsubseteq \rangle \) is the concrete domain and \( \rho \in A \rightarrow A \) be a meaning-preserving reduction.

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

The iterates are well-defined when the greatest lower bounds \( \bigcap (\text{glb}) \) 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 \( \gamma \) preserves glbs.

Pairwise reduction (cont’d)

Define the iterated pairwise reductions \( \tilde{\rho}^n, \tilde{\rho}^{-1}, \tilde{\rho}^* \in \langle \tilde{A}, \tilde{E} \rangle \mapsto \langle \tilde{A}, \tilde{E} \rangle, n \geq 0 \) of the Cartesian product for

\[
\tilde{\rho} \doteq \bigcirc_{i,j \in \Delta} \tilde{\rho}_{ij}
\]

where \( \bigcirc_{i=1}^n f_i \doteq f_{i_1} \circ \ldots \circ f_{i_n} \) is the function composition for some arbitrary permutation \( \pi \) of \([1, n]\). □

Pairwise reduction

- **Definition of pairwise reduction**

  Let \( \langle A_i, \sqsubseteq \rangle \) be abstract domains with increasing concretization \( \gamma_i \in A_i \to L \) 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 \( \forall (x, y) \in A_i \times A_j : \rho_{ij}(\langle x, y \rangle) \sqsubseteq_{ij} \langle x, y \rangle \) and \( (\gamma_i \times \gamma_j) \circ \rho_{ij} = (\gamma_i \times \gamma_j)^2 \)).

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

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

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

Iterated pairwise reduction

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

  If the limit \( \tilde{\rho}^* \) of the iterated reductions is well defined then the reductions are such that \( \forall \tilde{P} \in \tilde{A} : \forall n \in \mathbb{N}_+ : \tilde{\rho}^* (\tilde{P}) \sqsubseteq \tilde{\rho}^n (\tilde{P}) \sqsubseteq \tilde{\rho}_{ij}(\tilde{P}) \sqsubseteq \tilde{P}, i, j \in \Delta, i \neq j \) and meaning-preserving since \( \tilde{\rho}^{-1}(\tilde{P}), \tilde{\rho}_{ij}(\tilde{P}), \tilde{P} \in [\tilde{P}]_{/\neq} \).

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

- \( L = \varphi([a, b, c]) \)
- \( A_1 = \{\emptyset, \{a\}, T\} \) where \( T = \{a, b, c\} \)
- \( A_2 = \{\emptyset, \{a, b\}, T\} \)
- \( A_3 = \{\emptyset, \{a, c\}, T\} \)
- \( \langle T, \{a, b\}, \{a, c\}\rangle/\not\equiv = \langle\{a\}, \{a, b\}, \{a, c\}\rangle \)
- \( \tilde{\rho}^i_j(\langle T, \{a, b\}, \{a, c\}\rangle) = \langle T, \{a, b\}, \{a, c\}\rangle \) for \( \Delta = \{1, 2, 3\}, i, j \in \Delta, i \neq j \)
- \( \tilde{\rho}^* (\langle T, \{a, b\}, \{a, c\}\rangle) = \langle T, \{a, b\}, \{a, c\}\rangle \) is not a minimal element of \( \langle T, \{a, b\}, \{a, c\}\rangle/\not\equiv \)

Nelson–Oppen combination procedure

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

The Nelson-Oppen combination procedure

- \( \varphi \triangleq \varphi_1 \land \varphi_2 \) where
  - \( \varphi_1 \triangleq (x = a \lor x = b) \land y = a \land z = b \)
  - \( \varphi_2 \triangleq f(x) \neq f(y) \land f(x) \neq f(z) \).

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

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

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

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

- **Iterate** $\rho^i(\varphi)$: 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.

---

**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 $x_0$

  \[
  x_0 \equiv f(v_1, \ldots, v_n)
  \]

  (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):
  \[ y = f(x) == f(a+1) \land f(x) == f(2*b) \]
  becomes
  \[ z=a+1; t=2*b; y = f(x) == f(z) \land f(x) = f(t) \]

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

Reduced product of logical and algebraic domains

- When checking satisfiability of \( \varphi_1 \land \varphi_2 \land \ldots \land \varphi_n \), the Nelson-Oppen procedure generates (dis-)equalities that can be propagated by \( \rho_{la} \) to reduce the \( P_i, i=1,\ldots,m \), or
- \( \alpha_i(\varphi_1 \land \varphi_2 \land \ldots \land \varphi_n) \) can be propagated by \( \rho_{la} \) 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 \( \varphi_i \) 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