Decision Procedures for Parametric Theories

Sava Krstić*, Amit Goel*, Jim Grundy*, Cesare Tinelli**

*Intel Strategic CAD Labs
**The University of Iowa
This Talk

Based on work in

*S. Krstić, A. Goel, J. Grundy, and C. Tinelli
Combined Satisfiability Modulo Parametric Theories

*S. Krstić and A. Goel
Architecting Solvers for SAT Modulo Theories: Nelson-Oppen with DPLL
Contribution

Nelson-Oppen framework for theories in parametrically polymorphic logics—a fresh foundation for design of SMT solvers.

- Endowing SMT with a rich typed input language that can model arbitrarily nested data structures
- Completeness of the Nelson-Oppen-style combination method proved for theories of all common datatypes
- Troublesome stable infinity condition replaced by a natural notion of type parametricity
- Issue of finite-cardinality constraints exposed as crucial for completeness
SAT Modulo Theories (SMT)

There are decision procedures for (fragments of) logical theories of common datatypes.

Use them to decide validity/satisfiability of formulas that involve symbols from several theories.

\[ f(x) = x \Rightarrow f(2x - f(x)) = x \quad [\mathcal{T}_{UF} + \mathcal{T}_{Int}] \]

\[ \text{head}(a) = f(x) + 1 \ldots \quad [\mathcal{T}_{UF} + \mathcal{T}_{Int} + \mathcal{T}_{List}] \]

The underlying logic is the classical (unsorted or multi-sorted) first-order logic.
SMT Solvers

G. Nelson, D.C. Oppen *Simplification by cooperating decision procedures*, 1979

Input:
- theories $T_1, \ldots, T_n$ with disjoint signatures $\Sigma_1, \ldots, \Sigma_n$
- decision procedures $P_i$ for satisfiability of sets of $T_i$-literals

Output:
- a decision procedure for $(T_1 + \cdots + T_n)$-satisfiability of sets of $(\Sigma_1 + \cdots + \Sigma_n)$-literals.

Diagram:

```
\[ \Phi \]
\[ \text{sat?} \]
\[ \text{Combination Core} \]
\[ P_1 \]
\[ P_2 \]
\[ \cdots \]
\[ P_n \]
```
G. Nelson, D.C. Oppen *Simplification by cooperating decision procedures*, 1979

Input:

- theories $T_1, \ldots, T_n$ with disjoint signatures $\Sigma_1, \ldots, \Sigma_n$
- decision procedures $P_i$ for satisfiability of sets of $T_i$-literals

Output:

- a decision procedure for satisfiability of quantifier-free $(T_1 + \cdots + T_n)$-formulas.
Nelson-Oppen Combination Algorithm

1. Transform ("purify") the mixed input formula $\Phi$ into an equisatisfiable set of pure formulas $\Phi_1, \ldots, \Phi_n$

2. If $\Phi_i \models x = y$, update all $\Phi_j$ with $\Phi_j \cup \{x = y\}$

3. Repeat 2 until $\Phi_i \not\models \bot$ for some $i$ (return $\text{UNSAT}$), or 2 no longer applies (return $\text{SAT}$)
Nelson-Oppen: Example

\( \mathcal{T}_1 = \text{theory of lists} \quad \mathcal{T}_2 = \text{linear arithmetic} \)

Input set:

\[
\Phi = \begin{cases} 
  l_1 \neq l_2 \\
  \text{head}(l_2) \leq x \\
  l = \text{tail}(l_2) \\
  l_1 = x :: l \\
  \text{head}(l) - \text{head}(\text{tail} l_1) + x \leq \text{head}(l_2)
\end{cases}
\]

Purified sets:

\[
\Phi_1 = \begin{cases} 
  l_1 \neq l_2 \\
  y_1 = \text{head}(l_2) \\
  l = \text{tail}(l_2) \\
  l_1 = x :: l \\
  y_2 = \text{head}(l) \\
  y_3 = \text{head}(\text{tail} l_1)
\end{cases}
\]

\[
\Phi_2 = \begin{cases} 
  y_1 \leq x \\
  y_2 - y_3 + x \leq y_1
\end{cases}
\]
### Nelson-Oppen: Example [ctd]

<table>
<thead>
<tr>
<th>$\Phi_1$</th>
<th>$\Phi_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_1 \neq l_2$</td>
<td>$y_1 \leq x$</td>
</tr>
<tr>
<td>$y_1 = \text{head}(l_2)$</td>
<td>$y_2 - y_3 + x \leq y_1$</td>
</tr>
<tr>
<td>$l = \text{tail}(l_2)$</td>
<td></td>
</tr>
<tr>
<td>$l_1 = x :: l$</td>
<td></td>
</tr>
<tr>
<td>$y_2 = \text{head}(l)$</td>
<td></td>
</tr>
<tr>
<td>$y_3 = \text{head}(\text{tail } l_1)$</td>
<td></td>
</tr>
</tbody>
</table>

Unsatisfiable!
Nelson-Oppen: Example [ctd]

<table>
<thead>
<tr>
<th>$\Phi_1$</th>
<th>$\Phi_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_1 \neq l_2$</td>
<td>$y_1 \leq x$</td>
</tr>
<tr>
<td>$y_1 = \text{head}(l_2)$</td>
<td>$y_2 - y_3 + x \leq y_1$</td>
</tr>
<tr>
<td>$l = \text{tail}(l_2)$</td>
<td></td>
</tr>
<tr>
<td>$l_1 = x :: l$</td>
<td></td>
</tr>
<tr>
<td>$y_2 = \text{head}(l)$</td>
<td></td>
</tr>
<tr>
<td>$y_3 = \text{head}(\text{tail } l_1)$</td>
<td>$y_2 = y_3$</td>
</tr>
</tbody>
</table>

Unsatisfied!
Nelson-Oppen: Example [ctd]

<table>
<thead>
<tr>
<th>$\Phi_1$</th>
<th>$\Phi_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_1 \neq l_2$</td>
<td>$y_1 \leq x$</td>
</tr>
<tr>
<td>$y_1 = \text{head}(l_2)$</td>
<td>$y_2 - y_3 + x \leq y_1$</td>
</tr>
<tr>
<td>$l = \text{tail}(l_2)$</td>
<td></td>
</tr>
<tr>
<td>$l_1 = x :: l$</td>
<td></td>
</tr>
<tr>
<td>$y_2 = \text{head}(l)$</td>
<td></td>
</tr>
<tr>
<td>$y_3 = \text{head}(\text{tail } l_1)$</td>
<td>$y_2 = y_3$</td>
</tr>
<tr>
<td>$x = y_1$</td>
<td>$x = y_1$</td>
</tr>
</tbody>
</table>
### Nelson-Oppen: Example [ctd]

<table>
<thead>
<tr>
<th>$\Phi_1$</th>
<th>$\Phi_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_1 \neq l_2$</td>
<td>$y_1 \leq x$</td>
</tr>
<tr>
<td>$y_1 = \text{head}(l_2)$</td>
<td>$y_2 - y_3 + x \leq y_1$</td>
</tr>
<tr>
<td>$l = \text{tail}(l_2)$</td>
<td></td>
</tr>
<tr>
<td>$l_1 = x :: l$</td>
<td></td>
</tr>
<tr>
<td>$y_2 = \text{head}(l)$</td>
<td></td>
</tr>
<tr>
<td>$y_3 = \text{head}(\text{tail } l_1)$</td>
<td></td>
</tr>
<tr>
<td>$x = y_1$</td>
<td>$y_2 = y_3$</td>
</tr>
<tr>
<td><strong>UNSAT</strong></td>
<td></td>
</tr>
</tbody>
</table>


Completeness of Nelson-Oppen

- Suppose N-O halts in a state \( \{\Phi_1, \Phi_2\} \) and \( E \) is the set of all equalities between shared variables that have been derived.

- Let \( \Delta = E \cup \{x \neq y \mid x = y \notin E\} \) (arrangement)
  - \( \Phi_1 \cup \Delta \) has a \( T_1 \)-model \( M_1 \)
  - \( \Phi_2 \cup \Delta \) has a \( T_2 \)-model \( M_2 \)

- If \( M_1 \) and \( M_2 \) have the same cardinality *, we can “glue” them and obtain a model of \( \Phi_1 \cup \Phi_2 \cup \Delta \).

*Single-sorted FOL!
Completeness of Nelson-Oppen

• Suppose N-O halts in a state \{\Phi_1, \Phi_2\} and \(E\) is the set of all equalities between shared variables that have been derived.

• Let \(\Delta = E \cup \{x \neq y \mid x = y \notin E\}\) (arrangement)
  
  - \(\Phi_1 \cup \Delta\) has a \(T_1\)-model \(M_1\)
  
  - \(\Phi_2 \cup \Delta\) has a \(T_2\)-model \(M_2\)

• If \(M_1\) and \(M_2\) have the same cardinality, then we can “glue” them and obtain a model of \(\Phi_1 \cup \Phi_2 \cup \Delta\).

Q But what if the cardinalities of \(M_1\) and \(M_2\) are not the same?
Problem with Cardinality Mismatch

\[ \mathcal{T}_1 = \text{“theory of uninterpreted functions”} \]
\[ \mathcal{T}_2 = \text{theory of bits (not stably-infinite)} \]

Purified Input:

<table>
<thead>
<tr>
<th>( \Phi_1 )</th>
<th>( \Phi_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(f(x)) \neq x )</td>
<td>( x = 0 )</td>
</tr>
<tr>
<td>( f(f(y)) \neq y )</td>
<td>( y = 1 )</td>
</tr>
</tbody>
</table>

✻ There are no equations to propagate; N-O returns SAT

✻ \( \Phi_1 \) requires at least 3 elements for a model
The Notorious Stable Infiniteness Restriction

**Definition:** A first-order theory $\mathcal{T}$ is **stably infinite** if every satisfiable formula is satisfiable in an infinite model.

- This condition guarantees completeness of N-O, but
  - it’s not easy to prove
  - it’s not true in some important cases (eg, bitvectors)

- General understanding: the condition doesn’t matter much

- Lot of research shows completeness of N-O without it: [Tinelli-Zarba’04], [Fontaine-Gribomont’04], [Zarba’04], [Ghilardi-Nicolini-Zuchelli’07], [Ranise-Ringeissen-Zarba’05]
Parametricity, Not Stable Infiniteness: Example

\[ \Phi = \left\{ \begin{array}{l} \text{head}(\text{tail } a) = \text{head } a + x \\ \text{head } b = \text{head } a + x \\ \text{tail } a = \text{tail } b \end{array} \right\} \]

\[ \Phi_{\text{List}} = \left\{ \begin{array}{l} \text{tail } a = \text{tail } b \\ u = \text{head } a \\ v = \text{head } b \\ w = \text{head}(\text{tail } a) \end{array} \right\} \]

\[ \Phi_{\text{Int}} = \left\{ \begin{array}{l} w = u + x \\ v = u + x \end{array} \right\} \]

\[ \Delta = \{ v = w \neq u \} \]

\[ \left( \begin{array}{cccc} u & v & w & a \\ 5 & 9 & 9 & [5, 9] \\ \end{array} \right) \models \Phi_{\text{List}} \cup \Delta \]

\[ \left( \begin{array}{cccc} u & v & w & x \\ 1 & 2 & 2 & 1 \end{array} \right) \models \Phi_{\text{Int}} \cup \Delta \]

\[ \mathcal{T}_{\text{List}} \] knows nothing about \( \mathbb{Z} \) and cannot distinguish the pair \((5, 9)\) from any pair \((\text{\textbullet}, \text{\textbullet})\) of distinct symbols:

\[ \left( \begin{array}{cccc} u & v & w & a \\ \text{\textbullet} & \text{\textbullet} & \text{\textbullet} & [\text{\textbullet}, \text{\textbullet}] \end{array} \right) \models \Phi_{\text{List}} \cup \Delta \]

\[ \therefore \quad \text{To construct a model for } \Phi_{\text{List}} \cup \Phi_{\text{Int}} \cup \Delta, \text{ use the blue assignment to } u, v, w \]
All Theories of Practical Interest Are Parametric

\[ \Sigma_{\text{Int}} = \langle \text{Int} | \text{0}_{\text{Int}}, \text{1}_{\text{Int}}, (-1)_{\text{Int}}, \ldots, +_{\text{Int}^2 \rightarrow \text{Int}}, -_{\text{Int}^2 \rightarrow \text{Int}}, \times_{\text{Int}^2 \rightarrow \text{Int}}, \leq_{\text{Int}^2 \rightarrow \text{Bool}}, \ldots \rangle \]

\[ \Sigma_{\text{Array}} = \langle \text{Array} | \text{mk}_{\text{arr}}: \beta \rightarrow \text{Array}(\alpha, \beta), \text{read}[\text{Array}(\alpha, \beta), \alpha] \rightarrow \beta, \text{write}[\text{Array}(\alpha, \beta), \alpha, \beta] \rightarrow \text{Array}(\alpha, \beta) \rangle \]

\[ \Sigma_{\text{List}} = \langle \text{List} | \text{cons}[\alpha, \text{List}(\alpha)] \rightarrow \text{List}(\alpha), \text{nil}\text{List}(\alpha), \text{head}\text{List}(\alpha) \rightarrow \alpha, \text{tail}\text{List}(\alpha) \rightarrow \text{List}(\alpha) \rangle \]

\[ \Sigma_{\text{UF}} = \langle \Rightarrow | @[\alpha \Rightarrow \beta, \alpha] \rightarrow \beta \rangle \]

\[ \Sigma_{\times} = \langle \times | (\cdot, \cdot) : [\alpha, \beta] \rightarrow \alpha \times \beta, \text{fst}\alpha \times \beta \rightarrow \alpha, \text{snd}\alpha \times \beta \rightarrow \beta \rangle \]

\[ \Sigma_{\text{BitVec32}} = \ldots \]

\[ \Sigma_{\text{Sets}} = \ldots \]

\[ \Sigma_{\text{Multisets}} = \ldots \]
Syntax for Parametric Theories

- Parametrically typed FOL (PTFOL), an applicative fragment of HOL
  - A signature is a pair \( \Sigma = \langle O | K \rangle \), where \( O \) is the set of type operators and \( K \) is a set of constants
  - A type operator is a symbol, with an arity \( \geq 0 \)
  - Build types using type operators and type variables, as usual
  - A constant is a symbol, with an arity \( [\sigma_1, \ldots, \sigma_k] \rightarrow \sigma \)
  - Build (well-typed) terms by applying constants to term variables and simpler terms

- The logical signature, to be included in all others:

\[
\Sigma_{\text{Eq}} = \langle \text{Bool} | \equiv^{\alpha^2 \rightarrow \text{Bool}}, \text{ite}^{[\text{Bool}, \alpha, \alpha] \rightarrow \alpha}, \text{true}, \text{false}, \neg \rightarrow \text{Bool}, \wedge \rightarrow \text{Bool}, \ldots \rangle
\]
Theory Semantics: Parametric Structures

- $\mathcal{T}_{\text{Array}}$—like any other datatype theory—is a theory of a single model: there is a unique semantic structure associated with the signature

$$\Sigma_{\text{Array}} = \langle \text{Array} \mid \text{mk\_arr} : \beta \to \text{Array}(\alpha, \beta), \text{read} : [\text{Array}(\alpha, \beta), \alpha] \to \beta, \text{write} : [\text{Array}(\alpha, \beta), \alpha, \beta] \to \text{Array}(\alpha, \beta) \rangle$$

- Parametric types: $\text{Array}$ is a binary set operation: $\text{Array}(I, E)$ is the set of arrays indexed by $I$, with elements in $A$

- Parametric elements: $\text{read}$ is an indexed family of binary operations $\text{read}_{I, E} : [\text{Array}(I, E), I] \to E$

Q But what does it mean that $\text{Array}$ and $\text{read}$ are parametric?
Semantics: Parametric Set Operators

- The meaning of the type operator \( \text{List} \) is the function 
\[
\text{[List]} : \mathcal{U} \rightarrow \mathcal{U},
\]
where \( \mathcal{U} \) is a universe of sets.

- List also acts on relations: for every \( R : A \leftrightarrow B \), there is an induced relation 
\[
\text{[List]} \#(R) : \text{[List]}(A) \leftrightarrow \text{[List]}(B)
\]

- Functoriality:
  - \( \text{[List]} \#(\text{id}_A) = \text{id}_{\text{[List]}(A)} \)
  - \( \text{[List]} \#(R \circ R') = \text{[List]} \#(R) \circ \text{[List]} \#(R') \)

- We require that interpretations \([F]\) of all type operators \( F \) be functorial on partial bijections
  - This captures the uniformity of \([\text{List}]\) and other parametric set operations

NB Reynolds Parametricity is related, but not the same
Semantics: Parametric Elements

• The meaning of the constant $\text{fst}^\alpha \times \beta \to \alpha$ is the family of functions $\text{fst}_{A,B} : A \times B \to A$

• This family is parametric:

$$\text{fst}_{A,B}(a,b) (R \times S) \text{fst}_{A',B'}(a',b')$$

holds for all partial bijections $R : A \leftrightarrow A'$, $S : B \leftrightarrow B'$, and related elements $a \; R \; a'$, $b \; S \; b'$
Semantics: Satisfiability

Fix a theory $T$.

- Interpretation of types $\llbracket \tau \rrbracket \iota$ requires a type environment $\iota$ — a map from type variables to sets.

- Interpretation of terms $\llbracket t \rrbracket \iota \rho$ requires a type environment and a term environment $\rho$ — a map from term variables to elements of sets that $\iota$ associates to the types of these variables.

- A formula (term of boolean type) is $T$-satisfiable if $\llbracket \phi \rrbracket \iota \rho = \text{true}$ for some $\iota$ and $\rho$.

(Nothing fancy here)
Combination of Theories

* The signatures $\Sigma = \langle O \mid K \rangle$ and $\Sigma' = \langle O' \mid K' \rangle$ are **disjoint** if the $O \cap O' = \emptyset$ and $K \cap K' = \emptyset$

* . . . and then their **union signature** is $\Sigma + \Sigma' = \langle O \cup O' \mid K \cup K' \rangle$

* If $\mathcal{T}$ is a $\Sigma$-theory and $\mathcal{T}'$ is a $\Sigma'$-theory, there is a well-defined **union theory** $\mathcal{T} + \mathcal{T}'$ over $(\Sigma + \Sigma')$
Mixed Formulas and Purification

- In FOL, the formula $1 + f(x) = f(1 + f(x))$ purifies into
  $\Phi_{UF} = \{y = f(x), u = f(z)\}$, $\Phi_{Int} = \{z = 1 + y, z = u\}$

- For us, $\Phi_{UF} = \{y^{\text{Int}} = f^{\text{Int} \Rightarrow \text{Int}} x^{\text{Int}}, u^{\text{Int}} = f^{\text{Int} \Rightarrow \text{Int}} z^{\text{Int}}\}$

- This $\Phi_{UF}$ is not $\Sigma_{UF}$-pure because of presence of Int
  - $\Phi_{UF}$ is semipure—it's impurity is at the type level only

- We can use a pure approximation
  $\Phi_{UF}^{\text{pure}} = \{y^\alpha = f^{\alpha \Rightarrow \alpha} x^\alpha, u^\alpha = f^{\alpha \Rightarrow \alpha} z^\alpha\}$

- ... but this transformation is not safe!
Cardinality Constraints

\[ \Phi : \text{distinct}(x_1^{\text{List}(\alpha)}, \ldots, x_5^{\text{List}(\alpha)}) \quad \text{tail}(\text{tail } x_i^{\text{List}(\alpha)}) = \text{nil} \]

\[ \Phi_1 : \text{distinct}(x_1^{\text{List}(\text{Bool})}, \ldots, x_5^{\text{List}(\text{Bool})}) \quad \text{tail}(\text{tail } x_i^{\text{List}(\text{Bool})}) = \text{nil} \quad (i = 1, \ldots, 5) \]

\[ \Phi_2 : \text{distinct}(x_1^{\text{List}(\text{Int})}, \ldots, x_5^{\text{List}(\text{Int})}) \quad \text{tail}(\text{tail } x_i^{\text{List}(\text{Int})}) = \text{nil} \]

\[ \star \Phi = \Phi_1^{\text{pure}}, \Phi = \Phi_2^{\text{pure}}; \Phi_2 \text{ is satisfiable, } \Phi_1 \text{ is not} \]

\[ \star \text{Instead of } \Phi_1, \text{the } T_{\text{List}}\text{-solver gets } \Phi \text{ together with the cardinality constraint } \alpha \div 2 \]

\[ \star \text{Instead of } \Phi_2, \text{the } T_{\text{List}}\text{-solver just gets } \Phi \]

Lemma A semipure query \( \Phi \) is satisfiable iff \( \Phi^{\text{pure}} \) is satisfiable together with the cardinality constraints
Purification

Turn a mixed \((T_1 + \cdots + T_n)\)-query \(\Phi\) into the **purified form**

\[
\Phi_B \cup \Phi_E \cup \Phi_1 \cup \cdots \cup \Phi_n
\]

where

- \(\Phi_B\) is a set of propositional formulas
- \(\Phi_E = \{\ldots \ p_{\text{Bool}} \leftrightarrow x^r = y^r \ \ldots\}\)
- \(\Phi_i = \{\ldots \ p_{\text{Bool}} \leftrightarrow \psi \ \ldots\} \cup \{\ldots \ x^r = t \ \ldots\}\) (\(i\)-semipure)

**Ex:** \(f(x) = x \lor f(2 \times x - f(x)) > x\) becomes

\[
\Phi_B = \{p \lor q\} \quad \quad \quad \Phi_E = \{p \leftrightarrow y = x\},
\]

\[
\Phi_{UF} = \{y = f(x), \ u = f(z)\} \quad \quad \Phi_{Int} = \{q \leftrightarrow u > x, \ z = 2 \times x - y\}
\]
A PTFOL Nelson-Oppen Combination Theorem

**Theorem** The query

\[ \Phi = \Phi_B \cup \Phi_E \cup \Phi_1 \cup \cdots \cup \Phi_n \]

is \((T_1 + \cdots + T_n)\)-satisfiable iff there exist

- an assignment \(M\) of the atoms in \(\Phi_B\)
- an arrangement \(\Delta\) of the non-Bool variables in \(\Phi\)

such that

- \(M \models \Phi_B\)
- \(M, \Delta \models \Phi_E\)
- \((\Phi_i \cup M \cup \Delta)^{\text{pure}} \cup \Phi_i^{\text{card}}\) is \(T_i\)-satisfiable (for \(i = 1, \ldots, n\))
A theory $\mathcal{T}$ is **flexible** if for

- every query $\Phi$
- every injective $\langle \iota, \rho \rangle$ such that $\langle \iota, \rho \rangle \models \Phi$
- every $\alpha$ in the domain of $\iota$
- every $\kappa > |\iota(\alpha)|$

there exist injective $\langle \iota^{\text{up}(\kappa)}, \rho^{\text{up}(\kappa)} \rangle$ and $\langle \iota^{\text{down}}, \rho^{\text{down}} \rangle$ satisfying $\Phi$ such that

- $\iota^{\text{up}(\kappa)}(\beta) = \iota(\beta) = \iota^{\text{down}}(\beta)$ for every $\beta \neq \alpha$
- $\iota^{\text{up}(\kappa)}(\alpha)$ has cardinality $\kappa$ [up-flexibility]
- $\iota^{\text{down}}(\alpha)$ is countable [down-flexibility]

**Lemma?** All parametric theories are flexible.

- Proved for a large class, including all datatype theories
Combined Solver: Architecture & Implementation

**NODPLL**: Top-level architecture of an SMT solver
- Nelson-Oppen with DPLL
- presented as a transition system
- precisely formulates the main algorithmic features
- guides the implementation

**DPT**: Decision Procedure Toolkit
- SMT solver developed at Intel Strategic CAD Labs
- written in OCaml
- open source (SourceForge)
- clarity with competitive performance