Satisfiability Modulo Theories

Tjark Weber
webertj@in.tum.de

Oberseminar Statistische Analyse
November 11, 2004
Goal

To decide the satisfiability of formulas with respect to decidable background theories . . .

$$\phi ::= A \mid \neg \phi \mid \phi \lor \phi \mid \phi \land \phi$$

Applications:
- Formal verification
- Scheduling
- Compiler optimization
- . . .
Goal

To decide the satisfiability of formulas with respect to decidable background theories . . .

\[ \phi ::= A \mid \neg \phi \mid \phi \lor \phi \mid \phi \land \phi \]

. . . using a *combination* of SAT solving and theory-specific decision procedures.

Applications:

- Formal verification
- Scheduling
- Compiler optimization
- . . .
People

- Armando, Alessandro (U. of Genova)
- Barrett, Clark (New York U.)
- Berezin, Sergey (Stanford U.)
- Castellini, Claudio (U. of Genova)
- Cimatti, Alessandro (IRST-ITC)
- Cok, David (Eastman Kodak Company)
- Flanagan, Cormac (UC Santa Cruz)
- Fontaine, Pascal (U. of Liège)
- Ganesh, Vijay (Stanford U.)
- Giunchiglia, Enrico (U. of Genova)
- Kiniry, Joseph (U. of Nijmegen and KindSoftware LCC)
- Krstic, Sava (Strategic CAD Labs, Intel Corporation)
- Harrison, John (Intel)
- Janicic, Predrag (U. of Belgrade)
- Lahiri, Shuvendu (Carnegie Mellon U.)
People, cntd.

Joshi, Rajeev (NASA JPL)
de Moura, Leonardo (SRI International)
Nelson, Greg (HP Laboratories)
Ranise, Silvio (INRIA-Lorraine)
Ringeissen, Christophe (INRIA-Lorraine)
Ruess, Harald (SRI International)
Saxe, Jim (Compaq SRC)
Sebastiani, Roberto (U. of Trento)
Seshia, Sanjit (Carnegie Mellon U.)
Shankar, Natarajan (SRI International)
Strichman, Ofer (Technion U.)
Stump, Aaron (Washington U.)
Tinelli, Cesare (U. of Iowa)
Zarba, Calogero (INRIA-Lorraine)

Source: http://goedel.cs.uiowa.edu/smtlib/group.html
## Some SMT Systems

<table>
<thead>
<tr>
<th>Current:</th>
<th>Old:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Argo-lib</td>
<td>CVC</td>
</tr>
<tr>
<td>DPLL(T)</td>
<td>LPSAT</td>
</tr>
<tr>
<td>CVC Lite</td>
<td>RDL</td>
</tr>
<tr>
<td>haRVey</td>
<td>Simplify</td>
</tr>
<tr>
<td>ICS</td>
<td>STeP</td>
</tr>
<tr>
<td>Math-SAT</td>
<td>SVC</td>
</tr>
<tr>
<td>Tsat++</td>
<td>Tsat</td>
</tr>
<tr>
<td>UCLID</td>
<td></td>
</tr>
</tbody>
</table>

**Source:** [http://goedel.cs.uiowa.edu/smtlib/solvers.html](http://goedel.cs.uiowa.edu/smtlib/solvers.html)
Combining Decision Procedures

Theories:

- \( \mathcal{R} \): theory of rationals
  \[ \Sigma_\mathcal{R} = \{\leq, +, -, 0, 1\} \]

- \( \mathcal{L} \): theory of lists
  \[ \Sigma_\mathcal{L} = \{=, \text{hd}, \text{tl}, \text{nil}, \text{cons}\} \]

- \( \mathcal{E} \): theory of equality
  \[ \Sigma: \text{free function and predicate symbols} \]

Problem: Is

\[ x \leq y \land y \leq x + \text{hd}(\text{cons}(0, \text{nil})) \land P(h(x) - h(y)) \land \neg P(0) \]

satisfiable in \( \mathcal{R} \cup \mathcal{L} \cup \mathcal{E} \)?
The Nelson-Oppen Procedure


Given:

- $\mathcal{T}_1, \mathcal{T}_2$ first-order theories with signatures $\Sigma_1, \Sigma_2$
- $\Sigma_1 \cap \Sigma_2 = \emptyset$
- $\phi$ quantifier-free formula over $\Sigma_1 \cup \Sigma_2$

Obtain a decision procedure for satisfiability in $\mathcal{T}_1 \cup \mathcal{T}_2$ from decision procedures for satisfiability in $\mathcal{T}_1$ and $\mathcal{T}_2$. 
Nelson-Oppen: Example

Variable abstraction + equality propagation:

\[ x \leq y \land y \leq x + \text{hd}(\text{cons}(0, \text{nil})) \land P(h(x) - h(y)) \land \neg P(0) \]
Nelson-Oppen: Example

Variable abstraction + equality propagation:

\[ x \leq y \land y \leq x + \text{hd}(\text{cons}(0, \text{nil})) \land P(h(x) - h(y)) \land \neg P(0) \]
Nelson-Oppen: Example

Variable abstraction + equality propagation:

\[ x \leq y \land y \leq x + \text{hd}(\text{cons}(0, \text{nil})) \land P(h(x) - h(y)) \land \neg P(0) \]

\[ R \]
\[ \begin{array}{ll}
   x \leq y \\
y \leq x + v_1 \\
\end{array} \]

\[ L \]
\[ \begin{array}{l}
v_1 \\
v_2 \\
v_3 \\
v_4 \\
v_5 \\
\end{array} \]

\[ E \]
\[ \begin{array}{l}
P(v_2) \\
\neg P(v_5) \\
\end{array} \]
Nelson-Oppen: Example

**Variable abstraction + equality propagation:**

\[
x \leq y \land y \leq x + \text{hd}(\text{cons}(0, \text{nil})) \land P(h(x) - h(y)) \land \neg P(0)
\]

<table>
<thead>
<tr>
<th>( R )</th>
<th>( L )</th>
<th>( E )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x \leq y )</td>
<td>( v_1 = \text{hd}(\text{cons}(v_5, \text{nil})) )</td>
<td>( P(v_2) )</td>
</tr>
<tr>
<td>( y \leq x + v_1 )</td>
<td>( v_3 = h(x) )</td>
<td>( \neg P(v_5) )</td>
</tr>
<tr>
<td>( v_2 = v_3 - v_4 )</td>
<td>( v_1 = \text{hd}(\text{cons}(v_5, \text{nil})) )</td>
<td>( v_3 = h(x) )</td>
</tr>
<tr>
<td>( v_5 = 0 )</td>
<td>( v_4 = h(y) )</td>
<td>( v_1 = \text{hd}(\text{cons}(v_5, \text{nil})) )</td>
</tr>
</tbody>
</table>
Nelson-Oppen: Example

Variable abstraction + equality propagation:

\[ x \leq y \land y \leq x + \text{hd}(\text{cons}(0, \text{nil})) \land P(h(x) - h(y)) \land \neg P(0) \]

<table>
<thead>
<tr>
<th>(\mathcal{R})</th>
<th>(\mathcal{L})</th>
<th>(\mathcal{E})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x \leq y)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(y \leq x + v_1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(v_2 = v_3 - v_4)</td>
<td>(v_1 = \text{hd}(\text{cons}(v_5, \text{nil})))</td>
<td>(P(v_2))</td>
</tr>
<tr>
<td>(v_5 = 0)</td>
<td></td>
<td>(\neg P(v_5))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(v_3 = h(x))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(v_4 = h(y))</td>
</tr>
<tr>
<td></td>
<td>(v_1 = v_5)</td>
<td></td>
</tr>
</tbody>
</table>
Nelson-Oppen: Example

Variable abstraction + equality propagation:

\[ x \leq y \land y \leq x + \text{hd}(\text{cons}(0, \text{nil})) \land P(h(x) - h(y)) \land \neg P(0) \]

<table>
<thead>
<tr>
<th>$R$</th>
<th>$L$</th>
<th>$E$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \leq y$</td>
<td>$v_1 = \text{hd}(\text{cons}(v_5, \text{nil}))$</td>
<td>$P(v_2)$</td>
</tr>
<tr>
<td>$y \leq x + v_1$</td>
<td>$v_3 = h(x)$</td>
<td>$\neg P(v_5)$</td>
</tr>
<tr>
<td>$v_2 = v_3 - v_4$</td>
<td>$v_1 = h(x)$</td>
<td>$v_3 = h(x)$</td>
</tr>
<tr>
<td>$v_5 = 0$</td>
<td>$v_4 = h(y)$</td>
<td>$v_4 = h(y)$</td>
</tr>
<tr>
<td>$x = y$</td>
<td>$v_1 = v_5$</td>
<td></td>
</tr>
</tbody>
</table>
**Nelson-Oppen: Example**

**Variable abstraction + equality propagation:**

\[ x \leq y \land y \leq x + \text{hd}(\text{cons}(0, \text{nil})) \land P(h(x) - h(y)) \land \neg P(0) \]

<table>
<thead>
<tr>
<th>R</th>
<th>L</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x \leq y )</td>
<td>( x \leq y )</td>
<td>( P(v_2) )</td>
</tr>
<tr>
<td>( y \leq x + v_1 )</td>
<td>( v_1 = \text{hd}(\text{cons}(v_5, \text{nil})) )</td>
<td>( \neg P(v_5) )</td>
</tr>
<tr>
<td>( v_2 = v_3 - v_4 )</td>
<td>( v_1 = \text{hd}(\text{cons}(v_5, \text{nil})) )</td>
<td>( v_3 = h(x) )</td>
</tr>
<tr>
<td>( v_5 = 0 )</td>
<td>( v_1 = \text{hd}(\text{cons}(v_5, \text{nil})) )</td>
<td>( v_3 = h(y) )</td>
</tr>
<tr>
<td>( x = y )</td>
<td>( v_1 = v_5 )</td>
<td>( v_3 = v_4 )</td>
</tr>
</tbody>
</table>
Nelson-Oppen: Example

Variable abstraction + equality propagation:

\[ x \leq y \land y \leq x + \text{hd(cons}(0, \text{nil})) \land P(h(x) - h(y)) \land \neg P(0) \]

<table>
<thead>
<tr>
<th>Variable</th>
<th>Constraint</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x \leq y )</td>
<td>( y \leq x + v_1 )</td>
<td>( v_2 = v_3 - v_4 )</td>
</tr>
<tr>
<td>( v_2 = v_3 - v_4 )</td>
<td>( v_1 = \text{hd(cons}(v_5, \text{nil})) )</td>
<td>( v_1 = v_5 )</td>
</tr>
<tr>
<td>( v_5 = 0 )</td>
<td>( v_1 = \text{hd(cons}(v_5, \text{nil})) )</td>
<td>( v_2 = v_5 )</td>
</tr>
<tr>
<td>( x = y )</td>
<td>( v_1 = v_5 )</td>
<td>( v_2 = v_5 )</td>
</tr>
<tr>
<td>( v_3 = h(x) )</td>
<td>( v_3 = h(x) )</td>
<td>( v_3 = v_4 )</td>
</tr>
<tr>
<td>( v_4 = h(y) )</td>
<td>( v_4 = h(y) )</td>
<td>( v_3 = v_4 )</td>
</tr>
<tr>
<td>( \neg P(v_5) )</td>
<td>( \neg P(v_5) )</td>
<td>( v_3 = v_4 )</td>
</tr>
<tr>
<td>( P(v_2) )</td>
<td>( P(v_2) )</td>
<td>( v_3 = v_4 )</td>
</tr>
</tbody>
</table>
### Nelson-Oppen: Example

**Variable abstraction + equality propagation:**

\[
x \leq y \land y \leq x + \text{hd}(\text{cons}(0, \text{nil})) \land P(h(x) - h(y)) \land \neg P(0)
\]

<table>
<thead>
<tr>
<th>(R)</th>
<th>(L)</th>
<th>(E)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x \leq y)</td>
<td>(y \leq x + v_1)</td>
<td>(P(v_2))</td>
</tr>
<tr>
<td>(y \leq x + v_1)</td>
<td>(v_2 = v_3 - v_4)</td>
<td>(\neg P(v_5))</td>
</tr>
<tr>
<td>(v_2 = v_3 - v_4)</td>
<td>(v_1 = \text{hd}(\text{cons}(v_5, \text{nil})))</td>
<td>(v_3 = h(x))</td>
</tr>
<tr>
<td>(v_5 = 0)</td>
<td>(v_1 = \text{hd}(\text{cons}(v_5, \text{nil})))</td>
<td>(v_4 = h(y))</td>
</tr>
<tr>
<td>(x = y)</td>
<td>(v_1 = v_5)</td>
<td>(v_3 = v_4)</td>
</tr>
<tr>
<td>(v_2 = v_5)</td>
<td></td>
<td>(\bot)</td>
</tr>
</tbody>
</table>
Extensions and Related Work

- Relaxations of the *disjointness* requirement
- Nelson-Oppen is sound for combinations of *stably-infinite* theories


- Combinations of *unification* algorithms [F. Baader, K. Schulz]
SAT Solving: DPLL


def dpll(\phi: Boolean formula, \theta: partial assignment) {
    \theta' := deduce(\phi, \theta);
    \phi' := eval(\phi, \theta');
    if \phi'=True then return \theta'
    else if \phi'=False then return UNSATISFIABLE
    else {
        x := choose_fresh_variable(\phi', \theta');
        result := dpll(\phi', \theta' \cup \{x \leftrightarrow True\});
        if result=UNSATISFIABLE then
            return dpll(\phi', \theta' \cup \{x \leftrightarrow False\})
        else return result
    }
}
Combining Nelson-Oppen and DPLL

satisfy(\phi : \text{formula}) \{
    \text{create mapping } \Gamma \text{ from Boolean variables to atomic formulas;}
    \text{while True } \{
        \theta := \text{dpll}(\Gamma^{-1}(\phi), \emptyset);
        \text{if } \theta = \text{UNSATISFIABLE then return } \theta
        \text{else } \{
            \Theta := \Gamma(\theta);
            \text{if } \text{n-o} (\Theta) = \text{SATISFIABLE then return } \Theta
            \text{else } \phi := \phi \land \neg \Theta
        \}
    \}
\}
Optimizations and Variants

Gilles Audemard, Piergiorgio Bertoli, Alessandro Cimatti, Artur Kornilowicz, Roberto Sebastiani: A SAT Based Approach for Solving Formulas over Boolean and Linear Mathematical Propositions. 18th International Conference on Automated Deduction (CADE 2002), Copenhagen, Denmark, July 2002. Math-SAT


Preprocessing atoms
Atoms are rewritten into *normal form*, using theory-specific facts (associativity, commutativity, ...).
Optimizations: Math-SAT

- Preprocessing atoms
  Atoms are rewritten into normal form, using theory-specific facts (associativity, commutativity, ...).

- Several layers of decision procedures
  More powerful procedures are invoked only when weaker ones fail to show unsatisfiability.
Optimizations: Math-SAT

- **Preprocessing atoms**
  Atoms are rewritten into *normal form*, using theory-specific facts (associativity, commutativity, ...).

- **Several layers of decision procedures**
  More powerful procedures are *invoked only when weaker ones fail* to show unsatisfiability.

- **Early pruning**
  *Partial Boolean assignments* are tested by the theory-specific decision procedure.
Optimizations: Math-SAT

- Preprocessing atoms
  Atoms are rewritten into *normal form*, using theory-specific facts (associativity, commutativity, ...).

- Several layers of decision procedures
  More powerful procedures are *invoked only when weaker ones fail* to show unsatisfiability.

- Early pruning
  *Partial Boolean assignments* are tested by the theory-specific decision procedure.

- Enhanced early pruning
  Information gained from partial assignments is *passed back* to the SAT solver.
Online SAT solving
The SAT solver *continues* its search after accepting additional clauses (rather than to restart from scratch).
Optimizations: Math-SAT, Verifun

- Online SAT solving
  The SAT solver *continues* its search after accepting additional clauses (rather than to restart from scratch).

- Proof explication/mathematical learning
  The theory-specific decision procedures generate *lemmas*. 

---

Satisfiability Modulo Theories -- p.14/16
Optimizations: Math-SAT, Verifun

- Online SAT solving
  The SAT solver *continues* its search after accepting additional clauses (rather than to restart from scratch).

- Proof explication/mathematical learning
  The theory-specific decision procedures generate *lemmas*.

- Lazy/eager
Optimizations: Math-SAT, Verifun

- Online SAT solving
  The SAT solver *continues* its search after accepting additional clauses (rather than to restart from scratch).

- Proof explication/mathematical learning
  The theory-specific decision procedures generate *lemmas*.

  - Lazy/eager
  - Fine-grain/coarse-grain
Optimizations: Math-SAT, Verifun

- Online SAT solving
  The SAT solver *continues* its search after accepting additional clauses (rather than to restart from scratch).

- Proof explication/mathematical learning
  The theory-specific decision procedures generate *lemmas*.

  - Lazy/eager
  - Fine-grain/coarse-grain
  - Hiding of new proxy variables
Optimizations: Math-SAT, Verifun

- **Online SAT solving**  
The SAT solver *continues* its search after accepting additional clauses (rather than to restart from scratch).

- **Proof explication/mathematical learning**  
The theory-specific decision procedures generate *lemmas*.

  - Lazy/eager
  - Fine-grain/coarse-grain
  - Hiding of new proxy variables

- **Mathematical backjumping**  
The solver jumps back to the deepest branching point in which a literal *contributing to a conflict* was assigned a value.
Optimizations: DPLL(T)

*Tight integration* of the theory-specific decision procedure with the DPLL framework:

- **Initialize**($\mathcal{L}$: literal set)
- **SetTrue**($l:\mathcal{L}$-literal): $\mathcal{L}$-literal set
- **IsTrue**($l:\mathcal{L}$-literal): bool
- **Backtrack**($n:\mathbb{N}$)
- **Explanation**($l:\mathcal{L}$-literal): $\mathcal{L}$-literal set

The solver maintains a stack of all $\mathcal{L}$-literals that are true in a partial interpretation.
Future Work

- Better (theory-dependent) *heuristics* for ...
  - lemma management
  - literal selection
  - restarting

- Extension of existing SMT systems with decision procedures for *other theories*