Satisfiability Modulo Theories and Network Verification

Nikolaj Bjørner
Microsoft Research
Formal Methods and Networks Summer School
Ithaca, June 10-14 2013
Lectures

Wednesday 2:00pm-2:45pm:
An Introduction to SMT with Z3

Thursday 11:00am-11:45am
Algorithmic underpinnings of SAT/SMT

Friday 9:00am-9:45am
Theories, Solvers and Applications
Plan

1. Progress in automated reasoning
   SAT, Automated Theorem Proving, SMT

1. An abstract account for SMT search (DPLL+T)

2. Integrating Theories

**Takeaway:** Theorem Proving is cool and beautiful
Symbolic Engines: SAT, FTP and SMT

SAT: Propositional Satisfiability.
\[(\text{Tie} \lor \text{Shirt}) \land (\neg \text{Tie} \lor \neg \text{Shirt}) \land (\neg \text{Tie} \lor \text{Shirt})\]

FTP: First-order Theorem Proving.
\[
\forall X, Y, Z \ [X \ast (Y \ast Z) = (X \ast Y) \ast Z]
\forall X \ [X \ast \text{inv}(X) = e] \ \forall X \ [X \ast e = e]
\]

SMT: Satisfiability Modulo background Theories
\[
b + 2 = c \land A[3] \neq A[c-b+1]
\]
### SAT - Milestones

Problems impossible 10 years ago are trivial today

<table>
<thead>
<tr>
<th>Year</th>
<th>Milestone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1960</td>
<td>Davis-Putnam procedure</td>
</tr>
<tr>
<td>1962</td>
<td>Davis-Logeman-Loveland</td>
</tr>
<tr>
<td>1984</td>
<td>Binary Decision Diagrams</td>
</tr>
<tr>
<td>1992</td>
<td>DIMACS SAT challenge</td>
</tr>
<tr>
<td>1994</td>
<td>SATO: clause indexing</td>
</tr>
<tr>
<td>1997</td>
<td>GRASP: conflict clause learning</td>
</tr>
<tr>
<td>1998</td>
<td>Search Restarts</td>
</tr>
<tr>
<td>2001</td>
<td>zChaff: 2-watch literal, VSIDS</td>
</tr>
<tr>
<td>2005</td>
<td>Preprocessing techniques</td>
</tr>
<tr>
<td>2007</td>
<td>Phase caching</td>
</tr>
<tr>
<td>2008</td>
<td>Cache optimized indexing</td>
</tr>
<tr>
<td>2009</td>
<td>In-processing, clause management</td>
</tr>
<tr>
<td>2010</td>
<td>Blocked clause elimination</td>
</tr>
</tbody>
</table>

**Concept**

Results of the SAT competition/race winners on the SAT 2009 application benchmarks, 20mn timeout

Courtesy Daniel le Berre
### FTP - Milestones

<table>
<thead>
<tr>
<th>Year</th>
<th>Milestone</th>
<th>Who</th>
<th>Year</th>
<th>Milestone</th>
<th>Who</th>
</tr>
</thead>
<tbody>
<tr>
<td>1930</td>
<td>Hebrand's theorem</td>
<td>Herbrand</td>
<td>1970</td>
<td>Completion and saturation procedures</td>
<td>many people and provers</td>
</tr>
<tr>
<td>1934</td>
<td>Sequent calculi</td>
<td>Gentzen</td>
<td>1970</td>
<td>Knuth-Bendix ordering</td>
<td>Knuth; Bendix</td>
</tr>
<tr>
<td>1934</td>
<td>Inverse method</td>
<td>Gentzen</td>
<td>1971</td>
<td>Selection function</td>
<td>Kowalski; Kuehner</td>
</tr>
<tr>
<td>1955</td>
<td>Semantic tableaux</td>
<td>Beth</td>
<td>1972</td>
<td>Built-in equational theories</td>
<td>Plotkin</td>
</tr>
<tr>
<td></td>
<td>Herbrand-based theorem</td>
<td>Wang Hao</td>
<td>1972</td>
<td>Prolog</td>
<td>Colmerauer</td>
</tr>
<tr>
<td>1960</td>
<td>proving</td>
<td>Davis; Putnam</td>
<td>1974</td>
<td>Saturation algorithms</td>
<td>Overbeek</td>
</tr>
<tr>
<td>1960</td>
<td>Ordered resolution</td>
<td>Davis; Logemann; Loveland</td>
<td>1975</td>
<td>Completeness of paramodulation</td>
<td>Brand</td>
</tr>
<tr>
<td>1962</td>
<td>DLL</td>
<td>Davis; Logemann; Loveland</td>
<td>1975</td>
<td>AC-unification</td>
<td>Stickel</td>
</tr>
<tr>
<td>1963</td>
<td>First-order inverse method</td>
<td>Maslov</td>
<td>1976</td>
<td>Resolution as a decision procedure</td>
<td>Joyner; Joyner</td>
</tr>
<tr>
<td>1965</td>
<td>Unification</td>
<td>J. Robinson</td>
<td>1979</td>
<td>Basic paramodulation</td>
<td>Degtyarev</td>
</tr>
<tr>
<td>1965</td>
<td>First-order resolution</td>
<td>J. Robinson</td>
<td>1980</td>
<td>Lexicographic path orderings</td>
<td>Kamin; Levy</td>
</tr>
<tr>
<td>1965</td>
<td>Subsumption</td>
<td>J. Robinson</td>
<td>1985</td>
<td>Theory resolution</td>
<td>Stickel</td>
</tr>
<tr>
<td>1967</td>
<td>Orderings</td>
<td>Slagle</td>
<td>1985</td>
<td>Transformation</td>
<td>Plaisted; Greenbaum</td>
</tr>
<tr>
<td>1967</td>
<td>Demodulation or rewriting</td>
<td>Wos; G. Robinson; Carson; Shalla</td>
<td>1988</td>
<td>Superposition</td>
<td>Zhang</td>
</tr>
<tr>
<td>1968</td>
<td>Model elimination</td>
<td>Loveland</td>
<td>1988</td>
<td>Model construction</td>
<td>Zhang</td>
</tr>
<tr>
<td>1969</td>
<td>Paramodulation</td>
<td>G. Robinson; Wos</td>
<td>1989</td>
<td>Model construction</td>
<td>Stickel; Overbeek</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Term indexing</td>
<td></td>
</tr>
<tr>
<td>1990</td>
<td>General theory of redundancy</td>
<td>Bachmair; Ganzinger</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1992</td>
<td>Basic superposition</td>
<td>Nieuwenhuis; Rubio</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1993</td>
<td>First instance-based methods</td>
<td>Billon; Plaisted</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1993</td>
<td>Discount saturation algorithm</td>
<td>Avenhaus; Denzinger</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1998</td>
<td>Finite model finding using SAT</td>
<td>McCune</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>First-order DPLL</td>
<td>Baumgartner</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2003</td>
<td>iProver method</td>
<td>Ganzinger; Korovin</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2008</td>
<td>Sine selection</td>
<td>Hoder</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Some success stories:**
- **Open Problems (of 25 years):**
  - XCB: $X \equiv ((X \equiv Y) \equiv (Z \equiv Y)) \equiv Z$
    is a single axiom for equivalence
- **Knowledge Ontologies**
  - GBs of formulas

*Courtesy Andrei Voronkov, U of Manchester*
SMT - Milestones

<table>
<thead>
<tr>
<th>Year</th>
<th>Milestone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1977</td>
<td>Efficient Equality Reasoning</td>
</tr>
<tr>
<td>1979</td>
<td>Theory Combination Foundations</td>
</tr>
<tr>
<td>1979</td>
<td>Arithmetic + Functions</td>
</tr>
<tr>
<td>1982</td>
<td>Combining Canonizing Solvers</td>
</tr>
<tr>
<td>1992-8</td>
<td>Systems: PVS, Simplify, STeP, SVC</td>
</tr>
<tr>
<td>2002</td>
<td>Theory Clause Learning</td>
</tr>
<tr>
<td>2005</td>
<td>SMT competition</td>
</tr>
<tr>
<td>2006</td>
<td>Efficient SAT + Simplex</td>
</tr>
<tr>
<td>2007</td>
<td>Efficient Equality Matching</td>
</tr>
<tr>
<td>2009</td>
<td>Combinatory Array Logic, ...</td>
</tr>
</tbody>
</table>

Includes progress from SAT:

15KLOC + 285KLOC = Z3

Z3 (of '07) Time On Boogie Regression

Z3 Time On VCC Regression

Simplify (of '01) time

VCC Regression

Nov 08 - March 09
News: Solving $\exists R$ Efficiently

A key idea: Use partial solution to guide the search

Feasible Region

$-4xy - 4x + y > 1$

$\exists x \exists y \exists z - 5 < 0$

$x = 0.5$

$\exists x \exists y \exists z < 1$

Dejan Jojanovich & Leonardo de Moura, IJCAR 2012
mc(x) = x-10 if x > 100
mc(x) = mc(mc(x+11)) if x ≤ 100
assert (x ≤ 101 ⇒ mc(x) = 91)

∀ X. X > 100 → mc(X, X−10)
∀ X, Y, R. X ≤ 100 ∧ mc(X+11, Y) ∧ mc(Y, R) → mc(X, R)
∀ X, R. mc(X, R) ∧ X ≤ 101 → R = 91
SMT SOLVING
SMT : Basic Architecture

- Equality + UF
- Arithmetic
- Bit-vectors
- ...

Case Analysis
SAT + Theory solvers

Basic Idea

\[ x \geq 0, \ y = x + 1, \ (y > 2 \lor y < 1) \]

Abstract (aka “naming” atoms)

\[ p_1, \ p_2, \ (p_3 \lor p_4) \]

\[ p_1 \equiv (x \geq 0), \ p_2 \equiv (y = x + 1), \]

\[ p_3 \equiv (y > 2), \ p_4 \equiv (y < 1) \]
SAT + Theory solvers

Basic Idea

\[ x \geq 0, \ y = x + 1, \ (y > 2 \lor y < 1) \]

Abstract (aka “naming” atoms)

\[ p_1, \ p_2, \ (p_3 \lor p_4) \]

\[ p_1 \equiv (x \geq 0), \ p_2 \equiv (y = x + 1), \]
\[ p_3 \equiv (y > 2), \ p_4 \equiv (y < 1) \]

SAT Solver
SAT + Theory solvers

**Basic Idea**

\[
x \geq 0, \ y = x + 1, \ (y > 2 \lor y < 1)
\]

Abstract (aka “naming” atoms)

\[
p_1, \ p_2, (p_3 \lor p_4)
\]

Assignment

\[
p_1, \ p_2, \neg p_3, \ p_4
\]
SAT + Theory solvers

Basic Idea

\[ x \geq 0, \ y = x + 1, \ (y > 2 \lor y < 1) \]

Abstract (aka “naming” atoms)

\[ p_1, \ p_2, \ (p_3 \lor p_4) \]

\[ p_1 \equiv (x \geq 0), \ p_2 \equiv (y = x + 1), \]
\[ p_3 \equiv (y > 2), \ p_4 \equiv (y < 1) \]

Assignment

\[ p_1, \ p_2, \ \neg p_3, \ p_4 \]

\[ x \geq 0, \ y = x + 1, \]
\[ \neg (y > 2), \ y < 1 \]

SAT Solver
SAT + Theory solvers

Basic Idea

\[ x \geq 0, \ y = x + 1, \ (y > 2 \lor y < 1) \]

Abstract (aka “naming” atoms)

\[ p_1, \ p_2, (p_3 \lor p_4) \]
\[ p_1 \equiv (x \geq 0), \ p_2 \equiv (y = x + 1), \]
\[ p_3 \equiv (y > 2), \ p_4 \equiv (y < 1) \]

Assignment

\[ p_1, \ p_2, \neg p_3, \ p_4 \]

Unsatisfiable

\[ x \geq 0, \ y = x + 1, \ y < 1 \]

Theory Solver
SAT + Theory solvers

**Basic Idea**

\[ x \geq 0, \ y = x + 1, \ (y > 2 \lor y < 1) \]

Abstract (aka “naming” atoms)

\[ p_1, \ p_2, \ (p_3 \lor p_4) \]

\[ p_1 \equiv (x \geq 0), \ p_2 \equiv (y = x + 1), \]
\[ p_3 \equiv (y > 2), \ p_4 \equiv (y < 1) \]

**SAT Solver**

```
p_1, \ p_2, \neg p_3, \ p_4
```

**Assignment**

\[ x \geq 0, \ y = x + 1, \neg(y > 2), \ y < 1 \]

**Theory Solver**

```
\neg p_1 \lor \neg p_2 \lor \neg p_4
```

**New Lemma**

\[ x \geq 0, \ y = x + 1, \ y < 1 \]

**Unsatisfiable**
SAT + Theory solvers

New Lemma
\[ \neg p_1 \lor \neg p_2 \lor \neg p_4 \]

Unsatisfiable
\[ x \geq 0, \ y = x + 1, \ y < 1 \]

AKA Theory conflict

Theory Solver
SAT/SMT SOLVING USING DPLL(T)

[DAVIS PUTNAM LOGEMAN LOVELAND MODULO THEORIES]
Resolution

Formula must be in CNF

**Resolution rule:**

\[ C \lor p \quad D \lor \neg p / C \lor D \]

**Example:**

\[ q \lor t \lor p \quad q \lor r \lor \neg p / q \lor t \lor r \]

The result of resolution is the resolvent (clause). Original clauses are kept (not deleted).
Duplicate literals are deleted from the resolvent.

**Note:** No branching.

**Termination:** Only finite number of possible derived clauses.
Resolution (example)

A refutation of \( \neg p \lor \neg q \lor r, \ p \lor r, \ q \lor r, \ \neg r \):

Ex: Implement a naïve resolution procedure.
Unit & Input Resolution

**Unit resolution:** \( \text{Unit} \quad \text{Input} \quad \text{Resolution} \)

\( C \lor \ell \lor \neg \ell \) / \( C \lor \neg \ell \)  \( (C \lor \ell \) is subsumed by \( C)\)

**Input resolution:** \( \text{Input} \quad \text{resolution} \)

\( C \lor \ell \quad D \lor \neg \ell / C \lor D \)  \( (C \lor \ell \) member of input \( F)\).

Exercise:

**Set of clauses \( F:\)**

\( F \) has an input refutation iff \( F \) has a unit refutation.
DPLL

DPLL: David Putnam Logeman Loveland = Unit resolution + split rule.

\[ F/F, p \mid F, \neg p \text{ split} \quad p \text{ and } \neg p \text{ are not in } F \]

\[ F, C \lor l, \neg l/F, C, \neg l \text{ unit} \]

Ingredient of most efficient SAT solvers
**Pure Literals**

A literal is *pure* if only occurs positively or negatively.

Example:

$$\varphi = (\neg x_1 \lor x_2) \land (x_3 \lor \neg x_2) \land (x_4 \lor \neg x_5) \land (x_5 \lor \neg x_4)$$

$\neg x_1$ and $x_3$ are pure literals

**Pure literal rule:**

Clauses containing pure literals can be removed from the formula (i.e. just satisfy those pure literals)

$$\varphi_{\neg x_1,x_3} = (x_4 \lor \neg x_5) \land (x_5 \lor \neg x_4)$$

Preserve satisfiability, not logical equivalency!
DPLL (as a procedure)

- Standard **backtrack search**
- **DPLL**(F):
  - Apply unit propagation
  - If conflict identified, return **UNSAT**
  - Apply the pure literal rule
  - If F is satisfied (empty), return **SAT**
  - Select decision variable x
    - If DPLL\((F \land x)\) = **SAT** return **SAT**
    - return DPLL\((F \land \neg x)\)
DPLL

Guessing

\[ p \mid p \lor q, \neg q \lor r \]

\[ p, \neg q \mid p \lor q, \neg q \lor r \]
DPLL

Deducing

\[ p \mid p \lor q, \neg p \lor s \]

\[ p, s \mid p \lor q, \neg p \lor s \]
DPLL

Backtracking

\[ p, \neg s, q \implies p \lor q, s \lor q, \neg p \lor \neg q \]

\[ p, s \implies p \lor q, s \lor q, \neg p \lor \neg q \]
Modern DPLL

• Non-chronological backtracking (backjumping)
• Lemma learning

and

• Efficient indexing (two-watch literal)
• ...
CDCL – Conflict Directed Clause Learning

Lemma learning

\[ \neg t, p, q, s \mid t \lor \neg p \lor q, \neg q \lor s, \neg p \lor \neg s \]

\[ \neg t, p, q, s \mid t \lor \neg p \lor q, \neg q \lor s, \neg p \lor \neg s \mid \neg p \lor \neg s \]

\[ \neg t, p, q, s \mid t \lor \neg p \lor q, \neg q \lor s, \neg p \lor \neg s \mid \neg p \lor \neg q \]

\[ \neg t, p, q, s \mid t \lor \neg p \lor q, \neg q \lor s, \neg p \lor \neg s \mid \neg p \lor t \]
Core Engine in Z3: Modern DPLL/CDCL

<table>
<thead>
<tr>
<th>Step</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize</td>
<td>$\emptyset</td>
</tr>
<tr>
<td>Decide</td>
<td>$M</td>
</tr>
<tr>
<td>Propagate</td>
<td>$M</td>
</tr>
<tr>
<td>Sat</td>
<td>$M</td>
</tr>
</tbody>
</table>
| Conflict  | $M | F, C \Rightarrow M | F, C | C$  

- $C$ is false under $M$
- $C \subseteq M$, $\neg \ell \in M$
- $\ell \uparrow C \lor \ell \in M$
- $C$ is a learned clause

We will **now** motivate the CDCL algorithm as a cooperative procedure between model and proof search.

“*It took me a year to understand the Mini-SAT FUIP code*”
Mate Soos to Niklas Sörenson over ice-cream in Trento

[Nieuwenhuis, Oliveras, Tinelli J.ACM 06] customized
Mile High: Modern SAT/SMT search
The Farkas Lemma Dichotomy

1. There is an $x$ such that: $Ax = b \land x \geq 0$

2. There is a $y$ such that: $yA \geq 0 \land yb < 0$

For every matrix $A$, vector $b$ it is the case that either (1) or (2) holds (and not both).
A Dichotomy of Models and Proofs

1. There is a model $M$ such that $M \models F$

2. There is a proof $\Pi$ such that $F \vdash \bot \Pi \emptyset$

For every formula $F$ (set of clauses) it is the case that either (1) or (2) holds (and not both).
A Dichotomy of Models and Proofs

1. There is $M' \supseteq M$ such that $M' \models F$

2. There is $M' \subseteq M$ and proof $\Pi$ such that $F \vdash \Pi M'$

For every formula $F$ (set of clauses) and partial model $M$ it is the case that either (1) or (2) holds (and not both).
A Dichotomy of Models and Proofs

1. There is $M' \supseteq M$ such that $M \models F$

2. There is $M' \subseteq M$ and proof $\Pi$ such that $F \vdash \downarrow \Pi M'$

Given $M$ can it be extended to $M'$ to satisfy (1)? If not, find subset $M'$ to establish (2). (that is inconsistent with $F$)
A Dichotomy of Models and Proofs

**Corollary:**
If $F \vdash \Pi C$ then it is not possible to extend $C$ to satisfy $F$

**Corollary:**
If $M \models \neg F$ then
- $C, \ell \subseteq M$ for some $F \vdash C \lor \ell$ (or $F$ contains $\emptyset$)
- for every $D$, where
  - $D, C \subseteq M' \subseteq M$,
  - $M' \vdash (D \lor \neg \ell)$
  
  it is not possible to extend $M'$ to satisfy $F$
CDCL Search – Data structures

Invariant:

For state $M \mid F \mid C$:

$$ C \subseteq M \quad F \vdash C $$

Invariant:

For states $M \mid F$ and $M \mid F \mid D$ where $M = M\downarrow_1 \uparrow C \uparrow \downarrow_2$:

$$ C \subseteq M\downarrow_1 \quad F \vdash C \uparrow \downarrow_1 $$
CDCL steps

Initialize $\epsilon \models F$

$F$ is a set of clauses

No model candidate has been fixed
CDCL steps

Decide \( M \mid F \Rightarrow M, \ell \mid F \quad \ell \) is unassigned

Case split on \( \ell \)
If \( M \) can be extended to satisfy \( F \),
then the extension contains \( M, p \) or \( M, \neg p \)
CDCL steps

Propagate

\[ M \mid F, C \lor \ell \Rightarrow M, \ell \uparrow C \lor \ell \mid F, C \lor \ell \]

\( C \) is false under \( M \)

\( \ell \) must be true if \( M \) has any chance of being a model for \( F, C \lor \ell \)
CDCL steps

Sat  \[ M | F \implies M \quad \text{F true under } M \]

Unsat  \[ M | F | \emptyset \implies \text{Unsat} \]
CDCL steps

Conflict\[ M \mid F, C \Rightarrow M \mid F, C \mid C \quad \text{C is false under } M\]

\[ C \text{ is a } \textbf{sufficient} \text{ explanation why } M \text{ is not a model of } F \]
CDCL steps

Resolve\[ M \models F \mid C \lor \neg \ell \Rightarrow M \models F \mid C \lor D \]

\[ \ell \uparrow D \lor \ell \models M \]

Recall

Corollary:
If \( M \models \neg F \) then
- \( C, \ell \subseteq M \) for some \( F \models C \lor \ell \)
  (or \( F \) contains \( \emptyset \))
- for every \( D \), where
  - \( D, C \subseteq M \uparrow \subseteq M \),
  - \( M \uparrow \vdash (D \lor \neg \ell) \)
  it is not possible to extend \( M \uparrow \) to satisfy \( F \)

\( C \lor D \) is a sufficient and earlier explanation why \( M \) is not a model of \( F \)
CDCL steps

Backjump \( MM' \mid F \mid C \lor \ell \Rightarrow M \uparrow C \lor \ell \mid F \)
\[
C \subseteq M, \neg \ell \in M'
\]

- \( C \lor \ell \) is a sufficient explanation why \( M \) is not a model of \( F \)
- Prefixes of \( MM' \) that contain \( \neg \ell \) cannot become a model of \( F \)

**FUIP** *First Unique Implication Point* strategy when \# of decision literals in \( M \) is minimal.

Why is FUIP better?
- Minimizes \# of backtracking points before learned fact \( \ell \uparrow C \lor \ell \)
- What if \( \ell \uparrow C \lor \ell \) implies negation of removed backtracking point?
  - We would *forget* the learned fact \( \ell \uparrow C \lor \ell \) during backjumping.
  - ... only to then re-learn it.
CDCL steps

Learn $M | F | C \Rightarrow M | F, C | C$

Re-use proof step for later: build DAG proof instead of TREE proof
CDCL steps

Forget $M \mid F, C \Rightarrow M \mid F$

$c$ is a learned clause

Don’t forget to forget:
- Learned clauses could turn out to be useless.
- They could hog resources

Blocked Clause Elimination:
- Remove clauses that will not be used in proofs
CDCL steps

Restart \( M \mid F \Rightarrow \epsilon \mid F \)

Avoid getting trapped in one part of search space

\( S_{\downarrow 1}, S_{\downarrow 2}, \ldots = 1,1,2,1,2,4,1,1,2,1,2,4,8,1,1,2,1,2,4,1,1,2,4,8,1,\ldots \)

[Reluctant doubling sequence: Luby, Sinclair, Zuckerman, IPL 47]
Modern DPLL - tuning

• Restart frequency
  – Why is restarting good?
  – Efficient replay trick for frequent restart
• Which variable to split on
• Which branch to explore first
• Which lemmas to learn
• Blocked clause elimination
• Cache binary propagations
  – This is just scratching the surface
DPLL($\mathcal{T}$) solver interaction

**T- Propagate**

\[
M \mid F, C \lor \ell \Rightarrow M, \ell^{\text{c} \lor \ell} \mid F, C \lor \ell \quad \text{C is false under } T + M
\]

**T- Conflict**

\[
M \mid F \Rightarrow M \mid F \mid \neg M' \quad \text{M' } \subseteq \text{M and M' is false under } T
\]

**T- Propagate**

\[
a > b, b > c \mid F, a \leq c \lor b \leq d \Rightarrow
\]

\[
a > b, b > c, b \leq d_a^{\leq c \lor b \leq d} \mid F, a \leq c \lor b \leq d
\]

**T- Conflict**

\[
M \mid F \Rightarrow M \mid F, a \leq b \lor b \leq c \lor c < a
\]

where $a > b, b > c, a \leq c \subseteq M$
Model based Theory Combination

Challenge:
• Solvers need to exchange what is equal.
• Computing all implied equalities is expensive.

Idea:
• Have solvers produce models.
• Use models to introduce equalities on demand.
  If $M \models \varphi$ then guess $x = y$. 
Summary

1. Progress in automated reasoning
   SAT, Automated Theorem Proving, SMT

1. An abstract account for SMT search (DPLL+T)

2. Integrating Theories

Takeaway: Theorem Proving is cool and beautiful