Engineering Theories with Z3

Nikolaj Bjørner
Microsoft Research
IWIL March 10th 2012
Not so Hidden Agenda
Blatant, Shameless Propaganda

Z3 – An Efficient SMT Solver

Intro

SMT?

Z3?

Lazy Reduction

Eager Reduction

Theory Solver

Tutorial style

Many techniques apply broadly to SMT solvers: Barcelogic, CVC, Ergo, Mathsat, OpenSMT, Yices, ..

Many tools already use techniques ....

.. But many more tools should really do it too.
Rich Theories (and logics) with Efficient Decision Procedures
We review three methods:

**Theory Solver**: Optimization, Partial Orders

**Reduction**: Object Types

**Saturation**: HOL
The MUNCH Tool: automated reasoner for collections

This is the web page for the MUNCH tool. Currently the following is available for download:

- paper describing the tool
- implementation
- some examples and their output

Examples are written in the separate file (examples.txt). The tool then parses this input into a language corresponding to the grammar described in the paper and in the file ASTMultisets.scala. MUNCH invokes z3.

Playing with the MUNCH tool

The MUNCH tool is written in Scala and for testing MUNCH you need to have Scala installed. To run MUNCH, on your machine, first download the source code and compile it.
Overview of methods

New Theory

Reduction (eager reduction)

Compile

Saturation (lazy reduction)

Theory Solver (1st class solver)

New Theory

Constraints → Equalities

Model

Partial Compile

Search

New Theory
SMT solvers have specialized algorithms for $T$. 

Is formula $\varphi$ satisfiable modulo theory $T$?
Satisfiability Modulo Theories (SMT)

\[ x + 2 = y \Rightarrow f(\text{select}(\text{store}(a, x, 3), y - 2)) = f(y - x + 1) \]

Array Theory  Arithmetic  Uninterpreted Functions

select(store(a, i, v), i) = v
i \neq j \Rightarrow select(store(a, i, v), j) = select(a, j)
Job Shop Scheduling

Machines

Tasks

Jobs

$P = NP$?

$\zeta(s) = 0 \Rightarrow s = \frac{1}{2} + ir$
Job Shop Scheduling

Constraints:

**Precedence:** between two tasks of the same job

\[ [\text{start}_{2,2} \ldots \text{end}_{2,2}] \cap [\text{start}_{4,2} \ldots \text{end}_{4,2}] = \emptyset \]

**Resource:** Machines execute at most one job at a time
Job Shop Scheduling

Constraints:

Precedence:

![Diagram of precedence constraints]

Resource:

\[[\text{start}_{2,2}..\text{end}_{2,2}] \cap [\text{start}_{4,2}..\text{end}_{4,2}] = \emptyset\]

Encoding:

\[ t_{2,3} \ - \text{start time of job 2 on mach 3}\]

\[ d_{2,3} \ - \text{duration of job 2 on mach 3}\]

\[ t_{2,3} + d_{2,3} \leq t_{2,4}\]

Not convex

\[ t_{2,2} + d_{2,2} \leq t_{4,2} \lor t_{4,2} + d_{4,2} \leq t_{2,2}\]
Job Shop Scheduling

<table>
<thead>
<tr>
<th>$d_{i,j}$</th>
<th>Machine 1</th>
<th>Machine 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job 1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Job 2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Job 3</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

$\text{max} = 8$

Solution
$t_{1,1} = 5$, $t_{1,2} = 7$, $t_{2,1} = 2$, $t_{2,2} = 6$, $t_{3,1} = 0$, $t_{3,2} = 3$

Encoding
$(t_{1,1} \geq 0) \land (t_{1,2} \geq t_{1,1} + 2) \land (t_{1,2} + 1 \leq 8) \land$
$(t_{2,1} \geq 0) \land (t_{2,2} \geq t_{2,1} + 3) \land (t_{2,2} + 1 \leq 8) \land$
$(t_{3,1} \geq 0) \land (t_{3,2} \geq t_{3,1} + 2) \land (t_{3,2} + 3 \leq 8) \land$
$((t_{1,1} \geq t_{2,1} + 3) \lor (t_{2,1} \geq t_{1,1} + 2)) \land$
$((t_{1,1} \geq t_{3,1} + 2) \lor (t_{3,1} \geq t_{1,1} + 2)) \land$
$((t_{2,1} \geq t_{3,1} + 2) \lor (t_{3,1} \geq t_{2,1} + 3)) \land$
$((t_{1,2} \geq t_{2,2} + 1) \lor (t_{2,2} \geq t_{1,2} + 1)) \land$
$((t_{1,2} \geq t_{3,2} + 3) \lor (t_{3,2} \geq t_{1,2} + 1)) \land$
$((t_{2,2} \geq t_{3,2} + 3) \lor (t_{3,2} \geq t_{2,2} + 1))$
Job Shop Scheduling

Efficient solvers:
- Floyd-Warshall algorithm
- Ford-Fulkerson algorithm

\[
\begin{align*}
(t_{1,1} \geq 0) \land (t_{1,2} \geq t_{1,1} + 2) \land (t_{1,2} + 1 \leq 8) \land \\
(t_{2,1} \geq 0) \land (t_{2,2} \geq t_{2,1} + 3) \land (t_{2,2} + 1 \leq 8) \land \\
(t_{3,1} \geq 0) \land (t_{3,2} \geq t_{3,1} + 2) \land (t_{3,2} + 3 \leq 8) \land \\
((t_{1,1} \geq t_{2,1} + 3) \lor (t_{2,1} \geq t_{1,1} + 2)) \land \\
((t_{1,1} \geq t_{3,1} + 2) \lor (t_{3,1} \geq t_{1,1} + 2)) \land \\
((t_{2,1} \geq t_{3,1} + 2) \lor (t_{3,1} \geq t_{2,1} + 3)) \land \\
((t_{1,2} \geq t_{2,2} + 1) \lor (t_{2,2} \geq t_{1,2} + 1)) \land \\
((t_{1,2} \geq t_{3,2} + 3) \lor (t_{3,2} \geq t_{1,2} + 1)) \land \\
((t_{2,2} \geq t_{3,2} + 3) \lor (t_{3,2} \geq t_{2,2} + 1))
\end{align*}
\]

\[
\begin{align*}
z - t_{1,1} & \leq 0 \\
z - t_{2,1} & \leq 0 \\
z - t_{3,1} & \leq 0 \\
t_{3,2} - z & \leq 5 \\
t_{3,1} - t_{3,2} & \leq -2 \\
t_{2,1} - t_{3,1} & \leq -3 \\
t_{1,1} - t_{2,1} & \leq -2
\end{align*}
\]

\[
z - z = 5 - 2 - 3 - 2 = -2 < 0
\]
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

<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>

Problems impossible 10 years ago are trivial today

Concept

2002

2010

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

 Millions of variables from HW designs

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>1960</td>
<td>Ordered resolution</td>
<td>Davis; Putnam</td>
<td>1974</td>
<td>Saturation algorithms</td>
<td>Overbeek</td>
</tr>
<tr>
<td>1962</td>
<td>DLL</td>
<td>Davis; Logemann; Loveland</td>
<td>1975</td>
<td>Completeness of paramodulation</td>
<td>Brand</td>
</tr>
<tr>
<td>1963</td>
<td>First-order inverse method</td>
<td>Maslov</td>
<td>1975</td>
<td>AC-unification</td>
<td>Stickel</td>
</tr>
<tr>
<td>1965</td>
<td>Unification</td>
<td>J. Robinson</td>
<td>1976</td>
<td>Resolution as a decision procedure</td>
<td>Joyner</td>
</tr>
<tr>
<td>1965</td>
<td>First-order resolution</td>
<td>J. Robinson</td>
<td>1979</td>
<td>Basic paramodulation</td>
<td>Degtyarev</td>
</tr>
<tr>
<td>1965</td>
<td>Subsumption</td>
<td>J. Robinson</td>
<td>1980</td>
<td>Lexicographic path orderings</td>
<td>Kamin; Levy</td>
</tr>
<tr>
<td>1967</td>
<td>Orderings</td>
<td>Slagle</td>
<td>1985</td>
<td>Theory resolution</td>
<td>Stickel</td>
</tr>
<tr>
<td>1967</td>
<td>Demodulation or rewriting</td>
<td>Wos; G. Robinson; Carson; Shalla</td>
<td>1986</td>
<td>Transformation</td>
<td>Plaisted; Greenbaum</td>
</tr>
<tr>
<td>1968</td>
<td>Model elimination</td>
<td>Loveland</td>
<td>1988</td>
<td>Superposition</td>
<td>Zhang</td>
</tr>
<tr>
<td>1969</td>
<td>Paramodulation</td>
<td>G. Robinson; Wos</td>
<td>1988</td>
<td>Model construction</td>
<td>Zhang</td>
</tr>
<tr>
<td>1969</td>
<td>Model elimination</td>
<td>Wos; G. Robinson; Carson; Shalla</td>
<td>1989</td>
<td>Term indexing</td>
<td>Stickel; Overbeek</td>
</tr>
<tr>
<td>1972</td>
<td>Selection function</td>
<td>Kowalski; Kuehner</td>
<td>1990</td>
<td>General theory of redundancy</td>
<td>Bachmair; Ganzinger</td>
</tr>
<tr>
<td>1974</td>
<td>Built-in equational theories</td>
<td>Plotkin</td>
<td>1992</td>
<td>Basic superposition</td>
<td>Nieuwenhuis; Rubio</td>
</tr>
<tr>
<td>1975</td>
<td>Saturation algorithms</td>
<td>Overbeek</td>
<td>1993</td>
<td>First instance-based methods</td>
<td>Billon; Plaisted</td>
</tr>
<tr>
<td>1976</td>
<td>Resolution as a decision procedure</td>
<td>Joyner</td>
<td>1993</td>
<td>Discount saturation algorithm</td>
<td>Avenhaus; Denzinger</td>
</tr>
<tr>
<td>1979</td>
<td>Basic paramodulation</td>
<td>Degtyarev</td>
<td>1998</td>
<td>Finite model finding using SAT</td>
<td>McCune</td>
</tr>
<tr>
<td>1980</td>
<td>Lexicographic path orderings</td>
<td>Kamin; Levy</td>
<td>2000</td>
<td>First-order DPLL</td>
<td>Baumgartner</td>
</tr>
<tr>
<td>1985</td>
<td>Theory resolution</td>
<td>Stickel</td>
<td>2003</td>
<td>iProver method</td>
<td>Ganzinger; Korovin</td>
</tr>
<tr>
<td>1986</td>
<td>Transformation</td>
<td>Plaisted; Greenbaum</td>
<td>2008</td>
<td>Sine selection</td>
<td>Hoder</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, Manchester U
<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:

\[
15\text{KLOC} + 285\text{KLOC} = Z3
\]
An Efficient SMT Solver
From Microsoft Research

By Leonardo de Moura, Nikolaj Bjørner, Christoph Wintersteiger
Z3: Little Engines of Proof

Freely available from http://research.microsoft.com/projects/z3
## Decision Procedures

- **Modular Difference Logic is Hard**
  - TR 08 B, Blass Gurevich, Muthuvathi.
- **Linear Functional Fixed-points.**
  - CAV 09 B. & Hendrix.
- **A Priori Reductions to Zero for Strategy-Independent Gröbner Bases**
  - SYNASC 09 M& Passmore.
- **Efficient, Generalized Array Decision Procedures**
  - FMCAD 09 M & B
- **Quantifier Elimination as an Abstract Decision Procedure**
  - IJCAR 10, B
- **Cutting to the Chase**
  - CADE 11, Jojanovich, M

## Combining Decision Procedures

- **Model-based Theory Combination**
  - SMT 07 M & B. .
- **Accelerating Lemma learning using DPLL(U)**
  - LPAR 08 B, Dutetre & M
- **Proofs, Refutations and Z3**
  - IWIL 08 M & B
- **On Locally Minimal Nullstellensatz Proofs.**
  - SMT 09 M & Passmore.
- **A Concurrent Portfolio Approach to SMT Solving**
  - CAV 09 Wintersteiger, Hamadi & M
- **Conflict Directed Theory Resolution**
  - Cambridge Univ. Press 12, M & B

## Quantifiers, quantifiers, quantifiers

- **Efficient E-matching for SMT Solvers.**
  - CADE 07 M & B.
- **Relevancy Propagation.**
  - TR 07 M & B.
- **Deciding Effectively Propositional Logic using DPLL and substitution sets**
  - IJCAR 08 M & B.
- **Engineering DPLL(T) + saturation.**
  - IJCAR 08 M & B.
- **Complete instantiation for quantified SMT formulas**
  - CAV 09 Ge & M.
- **On deciding satisfiability by DPLL(Γ+ T) and unsound theorem proving.**
  - CADE 09 Bonachina, M & Lynch.
Fast and powerful

Summary View

The job selection includes SMT-COMP 2011, the official SMT-COMP’11 competition run

<table>
<thead>
<tr>
<th>Solver</th>
<th>Score</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z3</td>
<td>201/205</td>
<td></td>
</tr>
<tr>
<td>MathSAT5</td>
<td>199/205</td>
<td></td>
</tr>
<tr>
<td>CVC4 1.0rc4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>veriT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>opensmmt</td>
<td>187/202</td>
<td></td>
</tr>
<tr>
<td>SMT-RAT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVC3 v2.4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solver</th>
<th>Score</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z3</td>
<td>191/202</td>
<td>7459.6 s</td>
</tr>
<tr>
<td>CVC4 1.0rc4</td>
<td>156/202</td>
<td>5292.0 s</td>
</tr>
<tr>
<td>veriT</td>
<td>130/202</td>
<td>8924.1 s</td>
</tr>
<tr>
<td>opensmmt</td>
<td></td>
<td>8968.3 s</td>
</tr>
<tr>
<td>SMT-RAT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CVC3 v2.4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Z3 is Free Software for research

http://smtcomp.org
Theories

- Uninterpreted
- Arithmetic (linear)
- Bit-vectors
- **Algebraic data**
- Arrays
- User-defined

**RiSE4fun**

Click on a tool to load a sample then ask!

(declare-datatypes ((list (nil) (cons (hd Int) (tl list))))
(declare-funs ((11 list) (12 list)))
(assert (not (= 11 nil)))
(assert (not (= 12 nil)))
(assert (= (hd 11) (hd 12)))
(assert (= (tl 11) (tl 12)))
(assert (not (= 11 12)))
(check-sat)

ask z3

Is this SMT formula satisfiable? **Click 'ask z3'!** Read more or watch the video.

unsat
Some Microsoft tools using Z3

Property Driven

Execution Guided

Model Based

Over-Approximation

Under-Approximation

F7

SLAM

HAVOC

Yogi

SLAyer

SAGE
Some Microsoft tools using Z3

What does this dot graph look like? Ask AgI!

This tool requires a browser with Scalable Vector Graphics (SVG) support.

http://rise4fun.com
Get More Satisfaction with SMT

Optimization

Intro
SMT?
Z3?

Lazy Reduction
Eager Reduction
Theory Solver

New Theory

Constraints
Eqs

Search

Oliveras, Nieuwenhuis, SAT 2006
### Weighted MaxSMT

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_0$</td>
<td>$a \lor b \lor x \geq 2$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>$F_1$</td>
<td>$\neg a \lor x \geq 3$</td>
<td>3</td>
</tr>
<tr>
<td>$F_2$</td>
<td>$\neg b \lor x \geq 3$</td>
<td>4</td>
</tr>
<tr>
<td>$F_3$</td>
<td>$x &lt; 2$</td>
<td>5</td>
</tr>
</tbody>
</table>

Unsatisfiable: $F_2$
### Weighted MaxSMT

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
<th>weight</th>
<th>Penalty</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_0$</td>
<td>$a \lor b \lor x \geq 2$</td>
<td>$\infty$</td>
<td></td>
<td>$\neg a \land \neg b \land x &lt; 2$</td>
</tr>
<tr>
<td>$F_1$</td>
<td>$\neg a \lor x \geq 3$</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$F_2$</td>
<td>$\neg b \lor x \geq 3$</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$F_3$</td>
<td>$x &lt; 2$</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Weighted MaxSMT

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_0$</td>
<td>$a \lor b \lor x \geq 2$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>$F_1$</td>
<td>$\neg a \lor x \geq 3$</td>
<td>3</td>
</tr>
<tr>
<td>$F_2$</td>
<td>$\neg b \lor x \geq 3$</td>
<td>4</td>
</tr>
<tr>
<td>$F_3$</td>
<td>$x &lt; 2$</td>
<td>5</td>
</tr>
</tbody>
</table>

Sat $\neg a \land b \land x = 2$

Penalty: $9 = 4 + 5$
## Weighted MaxSMT

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
<th>weight</th>
<th>Penalty</th>
<th>Satisfaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_0$</td>
<td>$a \lor b \lor x \geq 2$</td>
<td>$\infty$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$F_1$</td>
<td>$\neg a \lor x \geq 3$</td>
<td>3</td>
<td></td>
<td>$\neg a \land \neg b \land x \geq 2$</td>
</tr>
<tr>
<td>$F_2$</td>
<td>$\neg b \lor x \geq 3$</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$F_3$</td>
<td>$x &lt; 2$</td>
<td>5</td>
<td></td>
<td>$\text{Penalty: 5}$</td>
</tr>
</tbody>
</table>
## Weighted MaxSMT

<table>
<thead>
<tr>
<th>Name</th>
<th>Formula</th>
<th>weight</th>
<th>Penalty</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_0$</td>
<td>$a \lor b \lor x \geq 2$</td>
<td>$\infty$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$F_1$</td>
<td>$\neg a \lor x \geq 3$</td>
<td>$3$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$F_2$</td>
<td>$\neg b \lor x \geq 3$</td>
<td>$4$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$F_3$</td>
<td>$x &lt; 2$</td>
<td>$5$</td>
<td></td>
<td>$a \land \neg b \land x &lt; 2$</td>
</tr>
</tbody>
</table>
Weighted MaxSMT

**Formula**

\[ \begin{align*}
  a \lor b \lor x & \geq 2 & \text{weight} \quad \infty \\
  F_1 \lor \neg a \lor x & \geq 3 & 3 \\
  F_2 \lor \neg b \lor x & \geq 3 & 4 \\
  F_3 \lor x & < 2 & 5 
\end{align*} \]

**Initially:** All atoms are unassigned
\[ Cost = 0 \]

**Assert** \( \neg a \land b \land x < 2 \)

**Propagate:** \( F_2: \) Cost := Cost + 4 := 4

**Best so far:** MinCost = 4

**Add Axiom** \( \neg F_2 - \text{backtrack} \)

**Assert** \( F_3 \) Cost = 5 > MinCost

**Add Axiom** \( \neg F_3 - \text{backtrack} \)

**…. Assert** \( a \land \neg b \land x < 2 \land F_1 \)
Principles of Modern SMT solvers in two slides
# Modern DPLL in a nutshell

<table>
<thead>
<tr>
<th>Step</th>
<th>Rule</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initialize</td>
<td>$\epsilon \mid F$</td>
<td>$F$ is a set of clauses</td>
</tr>
<tr>
<td>Decide</td>
<td>$M \mid F \Rightarrow M, \ell \mid F$</td>
<td>$\ell$ is unassigned</td>
</tr>
<tr>
<td>Propagate</td>
<td>$M \mid F, C \lor \ell \Rightarrow M, C^{\ell\lor} \mid F, C \lor \ell$</td>
<td>$C$ is false under $M$</td>
</tr>
<tr>
<td>Conflict</td>
<td>$M \mid F, C \Rightarrow M \mid F, C \mid C$</td>
<td>$C$ is false under $M$</td>
</tr>
<tr>
<td>Resolve</td>
<td>$M \mid F \mid C' \lor \neg \ell \Rightarrow M \mid F \mid C' \lor C$</td>
<td>$C^{\ell\lor} \in M$</td>
</tr>
<tr>
<td>Learn</td>
<td>$M \mid F \mid C \Rightarrow M \mid F, C \mid C$</td>
<td></td>
</tr>
<tr>
<td>Backjump</td>
<td>$M \neg \ell M' \mid F \mid C \lor \ell \Rightarrow M^{\ell\lor} \mid F$</td>
<td>$C$ has no literals in $M'$</td>
</tr>
<tr>
<td>Unsat</td>
<td>$M \mid F \mid \emptyset \Rightarrow \text{Unsat}$</td>
<td></td>
</tr>
<tr>
<td>Sat</td>
<td>$M \mid F \Rightarrow M$</td>
<td>$F$ true under $M$</td>
</tr>
<tr>
<td>Restart</td>
<td>$M \mid F \Rightarrow \epsilon \mid F$</td>
<td></td>
</tr>
</tbody>
</table>

Adapted and modified from [Nieuwenhuis, Oliveras, Tinelli J.ACM 06]
**DPLL(\(T\)) solver interaction**

**T- Propagate**

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

**T- Conflict**

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

**T- Propagate**

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

**T- Conflict**

\[ M \mid F \implies M \mid F, a \leq b \lor b \leq c \lor c < a \quad \text{where } a > b, b > c, a \leq c \subseteq M \]
How does Z3 enable $T$ solvers?
DPLL(T) Solver Interaction

Calls into DPLL engine

Callbacks from DPLL engine

T-Propagate

T-Propagate

T-Conflict

T-Conflict

Callbacks from DPLL engine

```
let mutable cost = 0
let mutable costs = []
let mutable min_cost = Int32.max_int
let mutable weights = Dictionary<Term,int>()

let th = ctx.MkTheory("opt")
let block() =
    let offender = optimize_cost ctx costs min_cost
    th.AssertTheoryAxiom(ctx.MkNot(ctx.MkAnd offender))

let Assign p vl =
    if vl then
        let w = weights.[p]
        costs <- cost + w;
        trail.Add (fun () -> cost <- cost - w)
        costs <- (w,p)::costs;
        trail.Add (fun () -> costs <- List.tail costs)
        if cost > min_cost then
            block()

let FinalCheck() =
    if cost < min_cost then
        min_cost <- cost
        block()
    true

let _ = th.NewAssignment <- (fun p vl -> Assign p vl)
let _ = th.FinalCheck <- (fun () -> FinalCheck())
let _ = th.Pop <- (fun () -> trail.Pop())
let _ = th.Push <- (fun () -> trail.Push())
```
Partial Orders & Object Hierarchies

Acyclic graphs and SMT

Intro
SMT?
Z3?

Lazy Reduction
Eager Reduction
Theory Solver
Partial Orders as Acyclic Graphs

\[ \forall x. x \preceq x \]
\[ \forall x, y. x \preceq y \land y \preceq x \rightarrow x = y \]
\[ \forall x, y, z. x \preceq y \land y \preceq z \rightarrow x \preceq z \]

Elements are equal in strongly connected components
Partial Orders as Acyclic Graphs

Checking negations

\[ \forall x. x \leq x \]
\[ \forall x, y. x \leq y \land y \leq x \rightarrow x = y \]
\[ \forall x, y, z. x \leq y \land y \leq z \rightarrow x \leq z \]

OK

Not OK
Partial Orders as Acyclic Graphs

Checking Consistency of \( \neg (x \preceq y) \):

Is there a \( \preceq \) path from \( \bullet \) to \( \bullet \) ?

Extracting Equalities from \( \preceq \) using strongly connected components:

```
let FinalCheck() =
    for t in terms do   // connect equal terms.
        let tr = th.GetEqcRoot t
        if tr <> t then
            add_eq t tr
    done
check_not_leqs()    // check negations
add IMPLIED_eqs()   // propagate equalities
true
```

```
let NewAssignment (a:Term) v =
    assert (is_leq a)
let args = a.GetAppArgs()
let t1, t2 = args[0], args[1]
add_term t1; add_term t2
if v then
    g.AddEdge t1 t2
else
    trail.Add (fun () -> not_leqs <- List.tail not_leqs)
    not_leqs <- (t1,t2)::not_leqs
```

```
let initialize() =
    th.FinalCheck <- (fun () -> FinalCheck())
    th.NewAssignment <- (fun a v -> NewAssignment a v)
    th.Push <- (fun () -> trail.Push())
    th.Pop <- (fun () -> trail.Pop())
```
Inheritance as table-lookup

$x \leq \text{java.lang.Comparable}$

$x \leq \text{java.lang.Cloneable}$

$x = \text{java.util.Date}$

Efficient propagators using Type Slicing algorithm
Leverages ordering of children
J. Gil and Y. Zibin. [TOPLAS 2007]

Available as F#/Z3 sample

Sherman, Garvin, Dwyer. IJCAR 2010
Object Graphs

To Cycle and not to Cycle

Intro
SMT?
Z3?

Lazy Reduction
Eager Reduction
Theory Solver
A Theory of Objects

```java
class O {
    public readonly D d;
    public readonly O left;
    public O right;
    public O(D data, O left, O right) {
        this.data = data;
        this.left = left;
        this.right = right;
    }
}
```
So far so good, but what about read-only fields?

**A Theory of Objects**

**sorts:**

\[ O, \]

**constructors:**

null : \( O, O : H \times D \times O \times O \rightarrow H \times O, \)

**accessors:**

data : \( H \times O \rightarrow D, \) left : \( H \times O \rightarrow O, \) right : \( H \times O \rightarrow O, \)

**modifiers:**

update_right : \( H \times O \times O \rightarrow H \)

\[
(h', o) = O(h, d, l, r) \implies o \neq \text{null}
\]

\[
(h', o) = O(h, d, l, r) \implies \text{data}(h', o) = d
\]

\[
(h', o) = O(h, d, l, r) \implies \text{left}(h', o) = l
\]

\[
o \neq \text{null} \implies \text{left}(h_1, \text{left}(h_2, \text{left}(... \text{left}(h_n, o)))) \neq o
\]

\[
h' = \text{update_right}(h, o, r) \land o' \neq o \implies \text{right}(h', o') = \text{right}(h, o')
\]

\[
h' = \text{update_right}(h, o, r) \implies \text{left}(h', o') = \text{left}(h, o')
\]

\[
h' = \text{update_right}(h, o, r) \implies \text{data}(h', o') = \text{data}(h, o')
\]
**Encoding: Heaps as Arrays**

Domains: objects are Natural numbers, left child is a smaller number

\[ O = N \]
\[ H = \langle data : O \Rightarrow D, left : O \Rightarrow O, right : O \Rightarrow O, clock : N \rangle \]

Most axioms follow by function definitions.

\[
\text{right}(\langle data, left, right, clock \rangle, o) = \text{select}(right, o) \\
\text{update_right}(\langle data, left, right, clock \rangle, o, r) = \langle data, left, \text{store}(right, o, r), clock \rangle
\]

**Only Axiom:** Instantiate for every occurrence of \( \text{left}(h, o) \)

\[
\forall h : H, o : O . o \neq \text{null} \implies 0 \leq \text{left}(h, o) < o
\]
Domains: read-only fields use algebraic data-types

\[ O = \text{null} \mid O(\text{id} : N, \text{data} : D, \text{left} : O) \]
\[ H = \langle \text{right} : O \Rightarrow O, \text{clock} : N \rangle \]

Most axioms follow by function definitions.

\[
\begin{align*}
\text{left}(h, O(\text{id}, \text{d}, l)) &= l \\
\text{left}(h, \text{null}) &= \text{null} \\
\text{right}(\langle \text{right}, \text{clock} \rangle, o) &= \text{select}(\text{right}, o) \\
\text{update_right}(\langle \text{right}, \text{clock} \rangle, o, r) &= \langle \text{store}(\text{right}, o, r), \text{clock} \rangle
\end{align*}
\]

No Extra Axiom: Data-type theory enforces acyclicity over \textit{left} → More efficient search
Z3 at the service of $\Gamma, \Pi, \Sigma, \alpha, \beta, \lambda, \eta, \kappa, *, \square$

SMT version of Satalax, Brown, CADE 2011
But

Used for First-Order Theorems
Sure, often HOL (problem) is just FO (solution) in disguise.

“For every problem there is a solution which is simple, clean and wrong.”

Henry Louis Mencken

“We are all faced with a series of great opportunities brilliantly disguised as unsolvable problems.”

John W. Gardner
**Digression: CAL**

CAL – Combinatory Array Logic

\[
\text{store}(a, i, v) = \lambda j. \text{if } i = j \text{ then } v \text{ else } a[j]
\]

\[
K(v) = \lambda j. v
\]

\[
\text{map}_f(a, b) = \lambda j. f(a[j], b[j])
\]

Existential fragment is in NP by reduction to congruence closure using polynomial set of instances.
but can we do something more HOLish?

e.g.,

$$\forall f. (\forall x, y. f(x) = f(y) \rightarrow x = y) \rightarrow \exists g. \forall x. x = g(f(x))$$
Idea: Saturate for Henkin Models

Types
\[ \sigma ::= i \mid o \quad \tau ::= \sigma \mid \tau \to \tau \]

Terms
\[ M, N ::= \lambda x: \tau. M \mid (M N) \mid x \]
\[ \text{false} : o \quad \Rightarrow : o \to o \to o \]

Constants
\[ \varepsilon : (\tau \to o) \to \tau, \quad \forall : (\tau \to o) \to o, \]
\[ = : \tau \to \tau \to o \]

Axioms
\[ (\forall (\lambda x : \tau. \neg(M x))) \lor (M (\varepsilon M)) \quad \text{for every } M : \tau \to o \]
\[ M = N \iff (\forall \lambda x : \tau . (M x) = (N x)) \quad \text{for every } M, N : \tau \to \tau' \]
\[ (\forall M) \implies (M N) \quad \text{for every } M : \tau \to o, N : \tau \]
Lazy Saturation loop

HOL formula $F$

Assert $\lbrack F \rbrack$

Check SAT

Unsat

Model

Instantiate

$F \leftarrow F \land F_{\text{Inst}}$

$(\forall (\lambda x : \tau. \neg(M x))) \lor (M (\epsilon M))$

$M = N \iff (\forall \lambda x : \tau. (M x) = (N x))$

$(\forall M) \implies (M N)$
Propositional reasoning

SAT

Equalities

SMT

Congruence

Closure

Extensional arrays

\([\_]: \text{HOL} \rightarrow \text{SMT}\)

\[
\begin{align*}
[(\forall \ M)] &= [(\forall \ M)] \\
[(\epsilon \ M)] &= [(\epsilon \ M)] \\
[M \rightarrow N] &= [M] \rightarrow [N] \\
[M = N] &= [M] = [N] \\
[(M \ N)] &= \text{select}([M], [N]) \\
[\lambda x : \tau . \ M] &= [\lambda x : \tau . \ M] \\
[f] &= f \quad \text{for constant } f
\end{align*}
\]
Set of $\beta\eta$ long NF terms with free variables from $\Gamma$ of type $\tau$

Enumerate $T[\Gamma; \tau]$ by depth:

$$(\lambda x : \tau . M) \in T[\Gamma; \tau \rightarrow \tau'] \quad \text{if} \quad M \in T[\Gamma, x : \tau; \tau']$$

$$(x \ M_1 \ldots \ M_k) \in T[\Gamma; \sigma] \quad \text{if} \quad (x : \bar{\tau} \rightarrow \sigma) \in \Gamma, \ M_i \in T[\Gamma; \tau_i]$$

Many more algorithms (matching, unification)/optimizations required for anything viable… ... but main task of Boolean search, equalities, functions is delegated
We surveyed three methods for adding new theories (logics) to Z3:

- As 1st class Theory Solver
- Eager reduction: embed theory in Z3
- Lazy reduction: add facts on demand

Choose one that fits your theory!

[Stan Rosenberg, Anindya Banerjee and David Naumann. Decision Procedures for Region Logic. VMCAI 2012]
Applications often generate problems with particular characteristics (many ground clauses/bit-vectors + predicates/arithmetic + transcendental/..)

New Z3 feature by de Moura & Passmore:
- Compose strategies using tactical interface.