SAT/SMT/AR
Introduction and Applications

Ákos Hajdu
About me

Ákos Hajdu

- PhD student at BME MIT (2016 – )
- Supervisor: Dr. Zoltán Micskei
- Topic: Applying counterexample-guided abstraction refinement in model checking

hajdua@mit.bme.hu
inf.mit.bme.hu/members/hajdua
References

- This presentation is based on the talks presented at the SAT/SMT/AR Summer School 2016

- Some content (text, figures, examples) of this presentation is copied from the presentations of the school, available at http://ssa-school-2016.it.uu.se/programme/
History of the school

- SAT/SMT (2011 – 2015)
  - 2011: MIT, USA
  - 2012: Trento, Italy
  - 2013: Espoo, Finland
  - 2014: Semmering, Austria
  - 2015: Stanford, USA

- SAT/SMT/AR (2016 – )
  - 2016: Lisbon, Portugal
    - http://ssa-school-2016.it.uu.se/
Overview

- **SAT**
  - Boolean satisfiability
  - Boolean variables and operators
  - Problem: is the formula satisfiable?

- **AR**
  - Automated Reasoning
  - First order logic
  - Problem: theorem proving

- **SMT**
  - Satisfiability Modulo Theories
  - First order logic, but interpreted symbols
  - Problem: is the formula satisfiable?

\[ \neg p \land (p \lor q) \]

\[ \forall x, y \exists z: p(f(x, y), g(z)) \]

\[ (x \leq y + 1) \land (y \geq 3) \]
SAT basics

- Building blocks
  - Boolean variables
  - Literals (variable or negation)
  - Clauses (disjunction of literals)
  - Formulas (conjunction of clauses)

- SAT problem
  - Decide if a variable assignment exists that evaluates the formula to true
  - In theory: NP-complete
  - In practice: efficient solvers

→ Annual SAT competition: [http://baldur.iti.kit.edu/sat-competition-2016/](http://baldur.iti.kit.edu/sat-competition-2016/)
SAT in practice

SAT in practice

Randomly generated problem

Real world problem (bounded model checking)

SAT in practice

- Wide range of applications

\section*{SAT modeling}

- Graph $G = (V, E)$, coloring with $k$ colors
  - $x_{i,j} = 1$ if vertex $v_i$ has $j$th color
  - Guarantee valid coloring
    - $\neg x_{i,j} \lor \neg x_{l,j}$ for all $(v_i, v_l) \in E$ and $j \in \{1, \ldots, k\}$
  - All vertices should have some color
    - $x_{i,1} \lor x_{i,2} \lor \cdots \lor x_{i,k}$ for all $v_i \in V$

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{sat-modeling.png}
\caption{Valid and invalid colorings of a graph.}
\end{figure}
Sudoku

- $v_{i,j,k} = 1$ if the value in row $i$ and cell $j$ is $k$
- Each cell has exactly one value
  - $\sum_{k=1}^{9} v_{i,j,k} = 1$ for all $i,j \in \{1, ..., 9\}$
- Each value used exactly once in each row
  - $\sum_{k=1}^{9} v_{i,j,k} = 1$ for all $i,k \in \{1, ..., 9\}$
- Each value used exactly once in each column
  - $\sum_{i=1}^{9} v_{i,j,k} = 1$ for all $j,k \in \{1, ..., 9\}$
- Each value used exactly once in each 3x3 sub-grid
  - $\sum_{r=1}^{3} \sum_{s=1}^{3} v_{3i+r,3j+s,k} = 1$ for all $i,j \in \{0, ..., 2\}$ and $k \in \{1, ..., 9\}$
- Fixed cells
  - $v_{1,1,5} = 1, v_{1,2,3} = 1, v_{1,5,7} = 1, v_{2,1,6} = 1, ...$
Modeling challenges

- **Non-conjunctive formulas**
  - Introduce auxiliary variables (Tseitin transformation)
  - Linear increase of the formula size

- **Cardinality constraints**
  - \( \sum_{j=1}^{n} x_j \geq 1: x_1 \lor x_2 \lor \cdots \lor x_n \)
  - \( \sum_{j=1}^{n} x_j \leq 1: \)
    - Pairwise: \( O(n^2) \) clauses
    - Sequential counter: \( O(n) \) clauses + \( O(n) \) new variables
    - Bitwise: \( O(n \log n) \) clauses + \( O(\log n) \) new variables

- Can be generalized from 1 to \( k \)
Modeling challenges

- Pseudo-Boolean constraints
  - \( \sum_{j=1}^{n} a_j x_j \leq b \)
  - Binary Decision Diagram (BDD): exponential clauses
  - Watchdog encoding: \( O(n^3 \log(n) \log(a_{max})) \) clauses + \( O(n^2 \log(n) \log(a_{max})) \) new variables

- CSP constraints
  - Direct encoding
    - \( x_i \) variable with domain \( D_i, m_i = |D_i| \)
    - \( x_{i,1}, \ldots, x_{i,m_i} \) variables, \( \sum_{k=1}^{m_i} x_{i,k} = 1 \)
    - If \( x_i = v_i \land x_j = v_j \) is not valid, add a clause: \( (\neg x_{i,v_i} \lor \neg x_{j,v_j}) \)
  - There are other, more efficient encodings
SAT solver algorithms

- DP-60
  - Eliminate variables step by step with resolution

\[
\begin{align*}
\text{x}_1 \lor \text{x}_4 \\
\neg \text{x}_1 \lor \text{x}_4 \lor \text{x}_{14} \\
\neg \text{x}_1 \lor \neg \text{x}_3 \lor \neg \text{x}_8 \\
\text{x}_1 \lor \text{x}_8 \lor \text{x}_{12} \\
\text{x}_1 \lor \text{x}_5 \lor \neg \text{x}_9 \\
\text{x}_2 \lor \text{x}_{11} \\
\neg \text{x}_3 \lor \neg \text{x}_7 \lor \text{x}_{13} \\
\neg \text{x}_3 \lor \neg \text{x}_7 \lor \neg \text{x}_{13} \lor \text{x}_9 \\
\text{x}_8 \lor \neg \text{x}_7 \lor \neg \text{x}_9 \\
\end{align*}
\]

\[
\begin{align*}
\text{x}_1 \lor \left( \begin{array}{c}
\text{x}_4 \\
\text{x}_8 \lor \text{x}_{12} \\
\text{x}_5 \lor \neg \text{x}_9 \\
\end{array} \right) \\
\neg \text{x}_1 \lor \left( \begin{array}{c}
\text{x}_4 \lor \text{x}_{14} \\
\neg \text{x}_3 \lor \neg \text{x}_8 \\
\end{array} \right) \\
\end{align*}
\]

\[
\begin{align*}
\left( \begin{array}{c}
\text{x}_4 \\
\text{x}_8 \lor \text{x}_{12} \\
\text{x}_5 \lor \neg \text{x}_9 \\
\end{array} \right) \lor \left( \begin{array}{c}
\text{x}_4 \lor \text{x}_{14} \\
\neg \text{x}_3 \lor \neg \text{x}_8 \\
\end{array} \right) \\
\end{align*}
\]

\[
\begin{align*}
\text{x}_4 \lor \text{x}_4 \lor \text{x}_{14} \\
\text{x}_4 \lor \neg \text{x}_3 \lor \neg \text{x}_8 \\
\text{x}_8 \lor \text{x}_{12} \lor \text{x}_4 \lor \text{x}_{14} \\
\text{x}_8 \lor \text{x}_{12} \lor \neg \text{x}_3 \lor \neg \text{x}_8 \\
\text{x}_5 \lor \neg \text{x}_9 \lor \text{x}_4 \lor \text{x}_{14} \\
\text{x}_5 \lor \neg \text{x}_9 \lor \neg \text{x}_3 \lor \neg \text{x}_8 \\
\end{align*}
\]
SAT solver algorithms

- **DPLL-62**
  - Backtrack search
    - Pick a literal
    - Search for solution by setting the literal to true
    - If conflicting, set literal to false

\[
\begin{align*}
x_1 \lor x_4 \\
\neg x_1 \lor x_4 \lor x_{14} \\
x_1 \lor \neg x_3 \lor \neg x_8 \\
x_1 \lor x_8 \lor x_{12} \\
x_2 \lor x_{12} \\
\neg x_3 \lor \neg x_{12} \lor x_{13} \\
\neg x_3 \lor x_7 \lor \neg x_{13} \\
x_8 \lor \neg x_7 \lor \neg x_{12}
\end{align*}
\]

<table>
<thead>
<tr>
<th>Level</th>
<th>Literal</th>
<th>Backtrack</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>\neg x_1</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>x_4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>x_3</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>\neg x_8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>x_{12}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>x_{13}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>x_7</td>
<td></td>
</tr>
</tbody>
</table>

Conflict, backtrack
SAT solver algorithms

- **DPLL-62 problems**
  - Selecting the literal
    - Bad decisions at the beginning can have a high cost
  - Only chronological backtracking

- **CDCL algorithms (2001 – )**
  - Conflict Driven Clause Learning
  - Learn new clause from conflict
  - Non-chronological backtrack and restart is possible
SAT extensions

- MUS: Minimal Unsatisfiable Subset
  - Minimal explanation of inconsistency

- MCS: Minimal Correction Subset
  - Minimal relaxation to recover consistency

- Application: analysis of over-constrained systems
  - Software/hardware model checking
  - Model-based diagnosis
  - Maximum feasible subsystem
  - Timetable/scheduling
  - ...
### SAT extensions

- **MaxSAT**
  - Maximize the number of satisfied clauses
  - Weighted MaxSAT: assign weights to clauses
  - Partial MaxSAT: “hard” clauses that must be satisfied
  - Application: optimization problems

- **QBF (Quantified Boolean Formulas)**
  - Quantified formulas ($\forall, \exists$)
  - Application: e.g. bounded model checking
  - Harder than SAT if $\text{NP} \neq \text{PSPACE}$
    - This is reflected in the complexity of solvers
    - But allows for exponentially more succinct encodings than SAT
Overview

- **SAT**
  - Boolean satisfiability
  - Boolean variables and operators
  - Problem: is the formula satisfiable?

- **AR**
  - Automated Reasoning
  - First order logic
  - Problem: theorem proving

- **SMT**
  - Satisfiability Modulo Theories
  - First order logic, but interpreted symbols
  - Problem: is the formula satisfiable?

\[
¬p \land (p \lor q)
\]

\[
\forall x, y \exists z: p(f(x, y), g(z))
\]

\[
(x \leq y + 1) \land (y \geq 3)
\]
AR basics

- Expressiveness of SAT is not always enough
- First Order Logic (FOL)
  - Symbolic knowledge representation and reasoning
  - Building blocks
    - Simple terms: constants, variables
    - Complex terms: functions
    - Atoms: predicates
    - Literals: atom or negation
    - Formulas: connectives and quantifiers
    - Partial order between elements

\[
\begin{align*}
\forall x \exists y: & \ p(f(x), y) \wedge x \neq y \\
& f(x, g(x)) > g(x)
\end{align*}
\]
AR basics

- Theorem proving
  - $H$ set of formulas: assumptions, hypotheses
  - $\phi$ formula: conjecture
  - Theorem proving problem: $H \Rightarrow \phi$?
  - Drawback: infinitely many interpretations on infinitely many domains

- In theory
  - Gödel: completeness
  - Turing: undecidability
  - Herbrand: semi-decidability
AR basics

- **Applications**
  - Software and hardware verification
  - Static analysis
  - Reasoning in knowledge bases
  - Mathematical theorem proving

- **Example: arrays**
  - Axioms
    - $\forall a, i, j: i = j \rightarrow a[i] = a[j]$
    - $\forall a, v, i, j: i = j \rightarrow \text{store}(a, i, v)[j] = v$
    - $\forall a, v, i, j: i \neq j \rightarrow \text{store}(a, i, v)[j] = a[j]$
  - Theorem
    - $\forall a, i, j: \text{store}(\text{store}(a, i, a[j]), j, a[i]) = \text{store}(\text{store}(a, j, a[i]), i, a[j])$
Theorem provers

- Inference system
  - Set of inference rules
  - Expansion or contraction of a set of formulas

- Expansion
  - Resolution
    \[
    \frac{p(x) \lor q(x), \neg p(f(a)) \lor r(a)}{q(f(a)) \lor r(a)}
    \]
  - Superposition
    \[
    \frac{f(z, e) = z, f(l(x, y), y) = x}{l(x, e) = x}
    \]

- Contraction
  - Subsumption
    \[
    \frac{p(x, y) \lor q(z), p(a, a) \lor q(b) \lor r(a)}{p(x, y) \lor q(z)}
    \]
  - Simplification
    \[
    \frac{f(x) = x, p(f(a)) \lor q(a)}{f(x) = x, p(a) \lor q(a)}
    \]
Theorem provers

- Search plan
  - Which rule to use?
  - Fairness: use all rules that are needed
  - Redundancy: avoid redundant clauses

- Strategy = inference system + search plan

- Algorithms
  - Ordering-based
  - Subgoal-reduction
  - Semantically-Guided Goal-sensitive
AR in practice

- **Tools**
  - Otter, EQP, Prover9
  - SNARK
  - SPASS
  - E
  - Vampire
  - leanCoP
  - iProver
  - Meris, MetiTarski

- **CASC**
  - The CADE ATP System Competition
Overview

- **SAT**
  - Boolean satisfiability
  - Boolean variables and operators
  - Problem: is the formula satisfiable?

- **AR**
  - Automated Reasoning
  - First order logic
  - Problem: theorem proving

- **SMT**
  - Satisfiability Modulo Theories
  - First order logic, but interpreted symbols
  - Problem: is the formula satisfiable?
SMT basics

- First order logic is undecidable 😞
- In practice, symbols often have an interpretation in some theory
  - Integers, reals, arrays, ...
  - Decidable theories (or fragments) 😊

- SMT problem
  - Check if a FOL formula is satisfiable modulo the background theories
  - Example (QF_UFLRA)
    \[
    (z = 1 \lor z = 0) \land (x - y + z = 1) \land (f(x) > f(y))
    \]
Example: schedule 3 jobs, each composed of 2 consecutive tasks, on 2 machines in 8 time units.

<table>
<thead>
<tr>
<th>Machine1</th>
<th>Machine2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job1</td>
<td>2</td>
</tr>
<tr>
<td>Job2</td>
<td>3</td>
</tr>
<tr>
<td>Job3</td>
<td>2</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
t_{1,1} &= 5 \\
t_{1,2} &= 7 \\
t_{2,1} &= 2 \\
t_{2,2} &= 6 \\
t_{3,1} &= 0 \\
t_{3,2} &= 3
\end{align*}
\]

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

- Example: is the code correct?

```c
void swap(int* a, int* b) {
    *a = *a + *b;
    *b = *a - *b;
    *a = *a - *b;
}
```

\[
\begin{align*}
    h_1 &= \text{store}(h_0, a, h_0[a] +_{32} h_0[b]) \\
    h_2 &= \text{store}(h_1, b, h_1[a] -_{32} h_1[b]) \\
    h_3 &= \text{store}(h_2, a, h_2[a] -_{32} h_2[b]) \\
    \neg(h_3[a] = h_0[b] \land h_3[b] = h_0[a])
\end{align*}
\]

\[
\begin{align*}
a &= 0, b &= 0 \\
h_0[0] &= 1, h_1[0] &= 2 \\
h_2[0] &= 0, h_3[0] &= 0
\end{align*}
\]
SMT theories

- Uninterpreted functions and equality
  \[ f(g(x)) = x \]
- Arrays
- Arithmetic (integer/real)
  - Difference logic
  - Linear arithmetic
  - Nonlinear arithmetic
  \[ x - y \leq 1 \]
  \[ 2x - 3y + 4z \leq 5 \]
  \[ x^2 + 3xy + y^2 > 0 \]
- Bitvectors
- Quantifiers
- ...
- Combinations are also possible
  \[ (\sim a \& (a +_{32} 1)) > a \]
  \[ \forall x \exists y \]
SMT logics

http://smtlib.cs.uiowa.edu/logics.shtml
SMT algorithms

- Theory solver: decision procedure for a conjunction of literals in a theory
  - Example: linear real arithmetic
    - Literals: linear (in)equalities
    - Conjunction: system of linear (in)equalities \(\rightarrow\) simplex algorithm

- Basic algorithm
  - Convert formula to disjunctive normal form
    - \((-l_0 \land -l_1 \land l_2) \lor (l_3 \land -l_4) \lor (l_5 \land l_6)\)
  - Check if any of the disjuncts is satisfiable

- Lazy algorithm
  - Introduce Boolean variables for SMT literals
  - Apply SAT solver, check literals selected by the solver
    - If not satisfiable \(\rightarrow\) add a blocking clause
Lazy SMT

Example

Theory solver
\[
\neg (a \geq 3) \land (a \geq 3 \lor a \geq 5)
\]
\[
\neg (a \geq 3) \land (a \geq 5)
\]
UNSAT
\[
(a \geq 3) \lor \neg (a \geq 5)
\]
Blocking clause

SAT solver
\[
\neg x \land (x \lor y)
\]
\[
x = 0, y = 1
\]
\[
(x \lor \neg y) \land \neg x \land (x \lor y)
\]
UNSAT
SMT algorithms

- **DPLL(T) algorithm**
  - Lazy SMT + tight integration with SAT solver
    - Incremental
    - Backtrack
    - Propagation
    - Conflicts

Theory solvers

- Uninterpreted functions
  - Congruence closure
- Difference logic
  - Map to a graph and search for negative cycle
- Linear arithmetic
  - Reals: variant of simplex algorithm
  - Integers: branch & bound algorithm
- Arrays
  - Use uninterpreted functions and refine if needed
- Bitvectors
  - Map to SAT (bit blasting)
SMT in practice

- SMT-LIB: international initiative
  - Standard descriptions of theories
  - Common input/output language
  - Connect the community
  - Large benchmark library
  - Related software tools

- Actively developed solvers (as of 2015)
  - Alt-Ergo, AProVE, Boolector, CVC4, MathSAT 5, OpenSMT 2, raSAT, SMTInterpol, SMT-RAT, STP, veriT, Yices 2, Z3

http://smtlib.cs.uiowa.edu/
SMT in practice

> (set-option :print-success false)
> (set-option :produce-models true)
> (set-option :interactive-mode true)
> (set-logic QF_LIA)
> (declare-fun x () Int)
> (declare-fun y () Int)
> (assert (= (+ x (* 2 y)) 20))  
\[ x + 2y = 20 \]
> (assert (= (- x y) 2))  
\[ x - y = 2 \]
> (check-sat)  
sat
> (get-value (x y))  
\[ (x 8)(y 6) \]
Summary

- **SAT**
  - Success story, engine of the engines, low level

- **AR**
  - High level, undecidable in general

- **SMT**
  - High level, but decidable theories and fragments

\[ \neg p \land (p \lor q) \]

\[ \forall x, y \exists z: p(f(x, y), g(z)) \]

\[ (x \leq y + 1) \land (y \geq 3) \]

Questions?

hajdua@mit.bme.hu
inf.mit.bme.hu/members/hajdua