SMT Unsat Core Minimization

Ofer Guthmann,
Ofer Strichman, Anna Trostanetski

FMCAD2016

Technion
Israel Institute of Technology
Satisfiability Modulo Theories

Satisfiability Modulo Theories (SMT): decides satisfiability of formulas over first order theories, by combining

- a SAT solver, and

- decision procedures for conjunctions of first order literals.
SMT solvers use Boolean Abstraction

Let $\varphi$ be an SMT formula

$\varphi$'s Boolean Abstraction, $e(\varphi)$, assigns a Boolean variable to every theory literal in $\varphi$.

Example:

\begin{align*}
\varphi &= ((x = 0)) \land ((x = 1) \lor \neg(x = 2)) \\
e(\varphi) &= (e_1) \land (e_2 \lor \neg e_3) \\
\text{Boolean structure unchanged.}
\end{align*}

Decoding: $d(e_1) := (x = 0), d(e_2) := (x = 1), \text{ etc.}$
The Minimal Unsat Core Problem (MUC)

Let $\varphi$ be an unsat SMT formula (in CNF).
Find a minimal (i.e., irreducible) unsat core of $\varphi$’s clauses.

$\varphi = a \land (\neg a \lor b) \land (\neg a \lor \neg b) \land (b \lor c)$

$C = \{a, (\neg a \lor b), (\neg a \lor \neg b)\}$

$C$ is a minimal unsat core.

Many applications may benefit from finding a MUC:

- Abstraction refinement.
- Formal equivalence verification.
- Decision procedures.
- Etc.

We know of no SMT MUC extractors in the public domain.
Deletion-based MUC Extraction (propositional case)

1. **Remove unmarked clause** \( c \in C \)
2. **SAT(C)?**
   - **Yes**: Mark \( c \), and add it back to \( C \)
   - **No**: Set \( C \leftarrow \text{core} \)
3. **All clauses marked?**
   - **Yes**: Return \( C \)
   - **No**: Repeat from step 1
Z3 and Cores

Z3 is an open-source competitive SMT solver:
- Developed by Microsoft Research.
- Emits an unsat core (set of clauses used in proof).
- Uses high-level proof rules

A Deletion-based SMT MUC Extractor

1. Remove unmarked clause \( c \in C \)
2. \( Z3(C) \)?
   - Yes: Mark \( c \), and add it back to \( C \)
   - No: \( C \leftarrow \text{core} \)
3. All clauses marked?
   - Yes: Return \( C \)
   - No: \( C \leftarrow \text{core} \)
Optimization: Rotation


Let $c$ be a marked clause.
- $\varphi \setminus \{c\}$ is satisfiable.
- $\alpha \models \varphi \setminus \{c\}$.

**Rotate($c, \alpha$)**
- Find $\alpha' \neq \alpha$ and $c' \neq c$, s.t. $\alpha' \models \varphi \setminus \{c'\}$
  - By flipping variables in $\alpha$ that appear in $c$.
- If such $c'$ was found:
  - Mark $c'$
  - Rotate($c', \alpha'$)
Now in SMT: Theory Rotation

Let $c$ be a marked clause.
- $\varphi \setminus \{c\}$ is satisfiable.
- $\alpha \models e(\varphi \setminus \{c\})$.

**Rotate($c$, $\alpha$)**
- Find $\alpha' \neq \alpha$ and $c' \neq c$, s.t. $\alpha' \models e(\varphi \setminus \{c'\})$:
  - By flipping variables in $\alpha$ that appear in $c$.
- If such $c'$ was found:
  - Mark $c'$
  - Rotate($c'$, $\alpha'$)

The problem: the new assignment may not be T-consistent.
Theory Rotation – Contradiction Example

\[ \varphi = \underbrace{(x = 0)}_{c} \wedge (\neg(x = 0) \vee (x = 1)) \wedge (\neg(x = 0) \vee (x = 2)) \]

\[ e(\varphi) = \underbrace{(e_1)}_{e(c)} \wedge (\neg e_1 \vee e_2) \wedge (\neg e_1 \vee e_3) \]

For a model\interpretation where \( x \mapsto 1 \) we have:

\[ \alpha := \{e_1, e_3\} \mapsto F, \{e_2\} \mapsto T \]
Theory Rotation – Contradiction Example

\( \varphi = \underbrace{(x = 0)}_{c} \land (\neg(x = 0) \lor (x = 1)) \land (\neg(x = 0) \lor (x = 2)) \)

\( e(\varphi) = \underbrace{(e_1)}_{e(c)} \land (\neg e_1 \lor e_2) \land (\neg e_1 \lor e_3) \)

For a model interpretation where \( x \mapsto 1 \) we have:

\( \alpha := \{\{e_1, e_3\} \mapsto F, \{e_2\} \mapsto T\} \)

\( \alpha \models e(\varphi \setminus \{c\}) \)

Flipping \( e_1 \) in \( \alpha \) results in a \text{T-contradiction}.

- both \( e_1 \to (x = 0) \) and \( e_2 \to (x = 1) \) now hold.
Theory Rotation - Solution

After finding \((c', \alpha')\), check if \(\alpha'\) is T-consistent.

If it is T-consistent use \textbf{Rotate} \((c', \alpha')\) as before.

If it’s not...

- One possibility is to give up and stop the recursion.
- Let’s try and do better.
Theory Rotation – Fixing a T-Contradiction

Try and find more variables to flip in $\alpha'$.  

Variables to flip: choose from $\text{core}(\alpha')$.  
- If resulting $\alpha''$ still contradictory, recursively flip more vars.  
- Recursion depth is determined heuristically.

$$\alpha'' \models \varphi \setminus \{c''\} \text{ and is T-consistent} \Rightarrow$$
- mark $c''$, and  
- Rotate $(c'', \alpha'')$.  

SMT MUCS
Adaptive Activation of Theory Rotation

Failed Theory Rotation can be costly.

Determine at runtime whether rotations is be continued:

First option:
  - **Fail Bound**: stop after $x$ consecutive failures.
  - Failure: no clauses were marked.

Observation: Rotation success-rate declines through time.
Adaptive Activation of Theory Rotation

Another option

- **Dynamic Measurement:** estimate $t_{smt} < \frac{t_r}{n_r}$ to stop rotation.
- **Problem:** measurement is non-monotonic.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>SMT SAT check time</th>
<th>Rotation time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Graph showing time cost per clause marking](image)

SMT MUCS
Adaptive Activation of Theory Rotation

Exponential smoothing: Given a stream of measurements \(\{(t_{smt}^i, t_{rot}^i, n_{rot}^i)\}_{i=1}^n\) define:

\[
\begin{align*}
T_{smt}^0 &= t_{smt}^0 \\
T_{smt}^i &= \alpha \cdot t_{smt}^i + (1 - \alpha) \cdot T_{smt}^{i-1}, \quad 0 \leq \alpha \leq 1
\end{align*}
\]

- Do the same for \(T_{rot}^i\) and \(N_{rot}^i\)

Stop rotation when \(T_{smt}^i < \frac{T_{rot}^i}{N_{rot}^i}\) holds.

\(\alpha\) chosen heuristically.
Adaptive Activation of Theory Rotation

Back to the example, now with exponential smoothing:

Time cost per clause marking
(Uses exp. smoothing w. alpha = 0.1)
Experimental Results – Avg. core size reduction

561 unsat SMT-LIB instances*

Avg. core size:
- Z3: 820 clauses.
- Min: 454 clauses.

Experimental Results – Theory Rotation

Reduces the number of (deletion) iterations.
Experimental Results – Theory Rotation

Translates to a modest run-time improvement (~6%-10%)

<table>
<thead>
<tr>
<th>Config.</th>
<th>Time (sec.)</th>
<th>T-check Time (sec.)</th>
<th>T-Conflicts Resolved</th>
</tr>
</thead>
<tbody>
<tr>
<td>(base)</td>
<td>30.5</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>T-Rotate</td>
<td>29.7</td>
<td>1.4</td>
<td>20.8</td>
</tr>
<tr>
<td>T-Rotate b 5</td>
<td>28.9</td>
<td>1.0</td>
<td>10.2</td>
</tr>
<tr>
<td>T-Rotate b 7</td>
<td>29.2</td>
<td>1.2</td>
<td>12.3</td>
</tr>
<tr>
<td>T-Rotate exp</td>
<td>29.6</td>
<td>1.2</td>
<td>11.2</td>
</tr>
</tbody>
</table>

Can be attributed to time spent on failed rotations, T-contradiction checks and additional var. flipping.

Best configuration is for Theory Rotation w. fail bound = 5
And now... Small Unsatisfiable Core (SUC)

[1] suggested an algorithm that finds a small (not necessarily minimal) SMT core

- Based on MathSat and the propos. MUC extractor Muser2

We re-implemented [1] based on Z3 + HaifaMuc

We also tested a hybrid approach in which we find a small core and then minimize it with HSmtMuc

Small Unsatisfiable Core (SUC)

Our re-implementation with Z3 and HaifaMUC:

- Requires proof logging (slows Z3 a lot).
- Requires a propositional encoding of Z3’s proof objects.
- Produces much larger proofs on avg. comparing to MathSat.

- Turned-out to be slower
We also tried a hybrid approach

MathSat-based SUC + minimization with HSmtMuc.

- Result is minimal.

The overall winner.

Less time-outs (HSmtMuc alone: **171** vs. Hybrid: **138**).

- (but higher runtime than HSmtMuc on instances that completed, HSmtMuc: **22.9 sec.** vs. Hybrid: **27.9 sec.**).
Summary

HSmtMuc is the first SMT-MUC extractor in the public domain.
- Based on Z3.

Best observed results:

MUC: the Hybrid algorithm
- MathSat SUC extraction, followed by HSmtMuc.

SUC:
- MathSat SUC extraction.

More information & our implementation is available at
http://strichman.net.technion.ac.il/
Questions?
Thank you!

- offers@ie.technion.ac.il
- ofer.guthmann@cs.technion.aci.il
- annat@cs.technion.ac.il