Efficient MUS Extraction with Resolution

Alexander Nadel  Intel, Haifa
Vadim Ryvchin  Intel + Technion, Haifa
Ofer Strichman  Technion, Haifa

Presented by Yue Wang
Some slides are borrowed from Vadim Ryvchin
Introduction

What is Minimal Unsatisfiable Set (MUS)?

- Given an unsatisfiable CNF formula:

\[ F = (a \lor b) \land (\neg b \lor c) \land (\neg c) \land (\neg a \lor c) \land (b \lor c) \]

- An unsatisfiable core is an unsatisfiable Set of its clauses:

\[ U_1 = (a \lor b) \land (\neg b \lor c) \land (\neg c) \land (\neg a \lor c) \land (b \lor c) \]

\[ U_2 = (a \lor b) \land (\neg b \lor c) \land (\neg c) \land (\neg a \lor c) \land (b \lor c) \]

\[ U_3 = (a \lor b) \land (\neg b \lor c) \land (\neg c) \land (\neg a \lor c) \land (b \lor c) \]

- Core is minimal if removal of any clause makes it satisfiable
  - U1 and U3 are minimal
  - U2 is not minimal, since U3 \( \subset \) U2
Minimal Unsatisfiable Set (MUS) Extraction

- Given an unsatisfiable CNF $F$, find a minimal (irreducible) set of clauses $C \subseteq F$ such that $C$ is unsatisfiable.

- Two main ‘schools’ of finding cores:
  - Assumption-based (Een & Sorensson, [2003])
  - Resolution-based (Zhang et al. [2003])
Deletion-based minimization

Initially Roots are unmarked

- Choose unmarked clause \( c \in \text{Roots} \)
- \( \text{Remove}(F, c); \)
- \( \text{SAT}(F') \)?
  - yes: mark \( c \)
  - no: \( \text{Roots} := \text{core} \)

- All marked: Return Roots

- Works for both ‘assumption-based’ and ‘resolution-based’
- Only one step is different – \( \text{Remove}(F, c) \)
Assumption-based MUS

- SAT solving under assumptions
  - SAT(F, Y)
    - F: Input formula
    - Y = \{y_1, y_2, \ldots, y_m\}: Assumptions. All the literals in Y are assigned true before other variables. Then invoke standard SAT.
  - Remove(F, c)
    - For each clause c in F, add the negation of a fresh selector variable s
      - c → \neg s \lor c
    - Restore c: add s to Y (s = \text{true} → \neg s \lor c = c)
    - Remove c: add a unit clause \neg s to F (\neg s = \text{true} → \neg s \lor c = \text{true})
Assumption-based MUS

- Why not remove c directly from F?
  - Incremental SAT solving
    - Boost the solving of closely related SAT instances
    - After one SAT call, one can add new clauses and re-invoke SAT
    - Does not allow remove clauses
Resolution-based MUS

- Resolution-based example
  - Initial clauses are on the right
Resolution-based MUS

- Build resolution refutation
- non-redundant - One initial clause is dropped ($I_1$)

Resolution refutation: a resolution derivation of the empty clause (unsat) – Represented as a DAG
Resolution-based MUS

- Consider clause $l_8$

The diagram illustrates the resolution process with clauses and literals, showing the relationships and implications between them.
Resolution-based MUS

- Consider clause $I_8$
- Invoke SAT solver on $I' = \text{UnRe}(I_8)$

- $\text{UnRe}(I_8)$ - the clauses unreachable from $I_8$
Resolution-based MUS

- Invoke SAT solver on $I' = \text{UnRe}(l_8)$
- Doesn’t know about resolution relation between clauses
Resolution-based MUS

- The instance is unsatisfiable
A new refutation is composed

- We can remove $I_8$
- Get a new core
Resolution-based MUS

- Consider $I_7$
Resolution-based MUS

- UnRe(\(I_7\)) is satisfiable with \(a=b=c=d=false\)
Resolution-based MUS

- $I_7$ is marked as belonging to a MUS
- The refutation is not changed
Resolution-based MUS

- Every other initial clause also belongs to MUS
Assumption vs. Resolution

- Assumption-based MUS:
  - Pros:
    - no need to maintain the resolution derivation
  - Cons:
    - Adding new selector variable – increase the length of Formula – not scale well

- Resolution-based MUS:
  - Pros:
    - reuse the existing resolution derivation
  - Cons:
    - need extra space to maintain the resolution derivation

- Time vs. Space
A. Maintain partial resolution proofs – save space
   - Only part of proof emanating from unmarked clauses
   - The red clauses (marked clauses) can be discarded
Optimization techniques

B. Perform selective clause minimization

- Shrinking conflict clauses

\[ c_1 = (\neg v_1 \lor v_2) \]
\[ c_2 = (\neg v_2 \lor v_3) \]
\[ c_3 = (\neg v_4 \lor v_5) \]
\[ c_4 = (\neg v_5 \lor v_6) \]
\[ c_5 = (\neg v_1 \lor \neg v_3 \lor \neg v_4 \lor \neg v_6) \]

I-UIP based conflict analysis:

\[ c_6 = (\neg v_1 \lor \neg v_3 \lor \neg v_4) \]
\[ c_6 = (\neg v_3 \lor \neg v_4) \]
Optimization techniques

Optimization 3

c. Postpone propagation of unmarked clauses
   - Perform Boolean Constraint Propagation on marked clauses first
D. Reclassify unmarked clauses

- When marking a new clause \( c \), look for any unmarked clauses that were derived from \( c \) and can also be marked.
Model Rotation*

Model Rotation

\( F \) is unsat, but \( \alpha \models F/c \)

\( \alpha' = \alpha[l \leftarrow \neg l] \) for some \( l \in c \)

\textbf{if} \quad (\text{UnsatSet}(F, \alpha') = \{c'\} \land c' \text{ is not in } M) \quad \textbf{then}

\begin{itemize}
  \item add \( c' \) to \( M \);
  \item Apply recursively with \((F, c', \alpha')\);
\end{itemize}

\textbf{Searching for other clauses that can be marked}

\textbf{c' is unmarked}
Model Rotation, eager
Optimization 5

\( F \) is unsat, but \( \alpha \models F/c \)

In the current call, \( K = \{c\} \)

\( \alpha' = \alpha[l \leftarrow \neg l] \) for some \( l \in c \)

\[
\text{if } (\text{UnsatSet}(F, \alpha') = \{c'\} \land c' \text{ is not in } K) \text{ then}
\]

add \( c' \) to \( K \);

if \( c' \) is not in \( M \)

add \( c' \) to \( M \);

Apply recursively with \( (F, c', \alpha') \);

}\}
The impact of E-rotation

Optimization 5

<table>
<thead>
<tr>
<th></th>
<th>E-rotation</th>
<th>Rotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial calls</td>
<td>259244</td>
<td>506908</td>
</tr>
<tr>
<td>Iterations</td>
<td>310680</td>
<td>556493</td>
</tr>
<tr>
<td>Iterations/calls</td>
<td>1.19</td>
<td>1.09</td>
</tr>
<tr>
<td>Clauses/iterations</td>
<td>7.27</td>
<td>3.63</td>
</tr>
<tr>
<td>Clauses/calls</td>
<td>8.72</td>
<td>3.98</td>
</tr>
</tbody>
</table>
Redundancy removal [BS’11]

Optimization 6

- $S$ is unsat $\Rightarrow$ ($\text{SAT}(S / c) \iff \text{SAT}((S / c) \land \lnot c)$
- Hence, add $\lnot c$ literals as assumptions.
- This is implemented already in MUSer-2 [BS’11].

- Our improvement ("path falsification"):
  - Add $\lnot c$, $\lnot c_1$, ..., $\lnot c_n$ literals as assumptions.
Path falsification

Optimization 6

- Vertex cut: separates \( \bot \) from the roots
- Implies what’s on its left
- \( \Rightarrow \) must be unsat
Path falsification

Optimization 6

- Consider $\alpha$, $\alpha \models \text{Roots} / c$
- $\alpha$ cannot satisfy a cut $\Rightarrow$
  $$\alpha \models (\neg c \land \neg c_1 \land \neg c_2) \lor (\neg c \land \neg c_1 \land \neg c_3)$$
  $\Rightarrow \alpha \models \neg c \land \neg c_1$

2 paths from $c$ to empty clause

$c_1, c$ – longest shared path prefix
## Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Time</th>
<th>Unsolved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base</td>
<td>93931</td>
<td>30</td>
</tr>
<tr>
<td>rot</td>
<td>48018</td>
<td>12</td>
</tr>
<tr>
<td>erot</td>
<td>44335</td>
<td>10</td>
</tr>
<tr>
<td>erot_AD</td>
<td>36295</td>
<td>8</td>
</tr>
<tr>
<td>erot_ABD</td>
<td>37798</td>
<td>13</td>
</tr>
<tr>
<td>erot_AB2D</td>
<td>32968</td>
<td>8</td>
</tr>
<tr>
<td>erot_AB2CD</td>
<td>32918</td>
<td>8</td>
</tr>
<tr>
<td>erot_AB2CD_rr</td>
<td>30800</td>
<td>6</td>
</tr>
<tr>
<td>erot_AB2CD_ps20</td>
<td>27263</td>
<td>4</td>
</tr>
<tr>
<td>MUSer2</td>
<td>59502</td>
<td>17</td>
</tr>
<tr>
<td>Minisatabb</td>
<td>40485</td>
<td>8</td>
</tr>
</tbody>
</table>

Other MUS Extractors

295 benchmarks of the 2011 MUS competition
Results

295 benchmarks of the 2011 MUS competition
HaifaMUC vs. Minisatbb*

* Lagniez, Biere, SAT’13
Conclusion

- Apply a number of optimization techniques for speeding up MUS extraction:
  - Maintain partial resolution proof: reduce memory
  - Perform selective clause minimization
  - Postpone propagation over interesting clauses
  - Reclassify interesting clauses
  - Model rotation / Eager Model rotation
  - Path strengthening

- All implemented in HaifaMUS
  - Outperform other leading MUS extractors: MUSer2 and Minisatabble
Thank you!