Problem Solving with SAT Oracles

Joao Marques-Silva$^{1,2}$

$^1$University College Dublin, Ireland
$^2$IST/INESC-ID, Lisbon, Portugal

Theoretical Foundations of Applied SAT Solving
BIRS, Banff, Canada, January 2014
SAT solving in practice

• SAT is a success story of Computer Science
  – Hundreds (even more?) of practical applications
SAT solving in practice

- SAT is a success story of Computer Science
  - Hundreds (even more?) of practical applications
SAT solving in practice

- SAT is a success story of Computer Science
  - Hundreds (even more?) of practical applications

- Many formulated as decision problems; many others not
<table>
<thead>
<tr>
<th>Answer</th>
<th>Problem Type</th>
</tr>
</thead>
</table>

Decision vs. function problems
## Decision vs. function problems

<table>
<thead>
<tr>
<th>Answer</th>
<th>Problem Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes/No</td>
<td>Decision Problems</td>
</tr>
</tbody>
</table>
## Decision vs. function problems

<table>
<thead>
<tr>
<th>Answer</th>
<th>Problem Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes/No</td>
<td>Decision Problems</td>
</tr>
<tr>
<td>Otherwise</td>
<td></td>
</tr>
</tbody>
</table>

Recap MUS, MCS, MSS, MES, MFS, ...!
## Decision vs. function problems

<table>
<thead>
<tr>
<th>Answer</th>
<th>Problem Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes/No</td>
<td>Decision Problems</td>
</tr>
<tr>
<td>Otherwise</td>
<td>Function (Search) Problems</td>
</tr>
</tbody>
</table>
Decision vs. function problems

<table>
<thead>
<tr>
<th>Answer</th>
<th>Problem Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes/No</td>
<td>Decision Problems</td>
</tr>
<tr>
<td>Otherwise</td>
<td>Function (Search) Problems</td>
</tr>
</tbody>
</table>

### Function Problems on Propositional Formulas

- MaxSAT
- MinSAT
- Maximal Models
- Minimal Models
- Autarkies
- Backbones
- Prime Implicates
- Prime Implicants
- Indep. Vars
- MCFSes
- MFSes
- MESes
- MUSes
- MDSes
- MSSes
- Implicant Ext.
- Implicate Ext.
## Decision vs. function problems

<table>
<thead>
<tr>
<th>Answer</th>
<th>Problem Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes/No</td>
<td><strong>Decision Problems</strong></td>
</tr>
<tr>
<td><strong>Otherwise</strong></td>
<td><strong>Function (Search) Problems</strong></td>
</tr>
</tbody>
</table>

### Function Problems on Propositional Formulas

- **Optimization Problems**
  - MaxSAT
  - PBO
  - ...  
  - WBO
  - MinSAT

- **Minimal Sets**
  - Minimal Models
  - Maximal Models
  - Backbones
  - Prime Implicants
  - Autarkies
  - Prime Implicates
  - Indep. Vars
  - MUSes
  - MCSes
  - MESES
  - MDSes
  - MNSes
  - MFSes
  - MCFSes
  - Implicant Ext.
  - Implicate Ext.

Recap MUS, MCS, MSS, MES, MFS, ...!
### Decision vs. function problems

<table>
<thead>
<tr>
<th>Answer</th>
<th>Problem Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes/No</td>
<td>Decision Problems</td>
</tr>
<tr>
<td>Otherwise</td>
<td>Function (Search) Problems</td>
</tr>
</tbody>
</table>

#### Function Problems on Propositional Formulas

**Optimization Problems**
- MaxSAT
- MinSAT
- PBO
- WBO

**Minimal Sets**
- Minimal Models
- Maximal Models
- Prime Implicants
- Autarkies
- Backbones
- MCSes
- MESes
- MFSes

**Prime Implicates**
- Indep. Vars
- MDSes
- Implicate Ext.

#### Recap MUS, MCS, MSS, MES, MFS, ...!
How to solve function problems?

- Poly-time Algorithm
- Yes/No + Witness
- Decision Procedure
- Bounded # of calls / queries
- SAT, SMT, CSP, ...
- Solver / Oracle

• SAT oracle \(\neq\) (standard) NP oracle.

Why?

- SAT oracles compute witnesses for outcomes
- SAT oracle corresponds to a witness oracle (more later) [e.g. BKT93]

• SAT oracle queries can be expensive!

How to minimize queries?

- Develop more efficient algorithms, i.e. with fewer oracle calls
- Characterize query complexity

▶ But, use witness oracles instead of NP oracles
How to solve function problems?

- SAT oracle $\neq$ (standard) NP oracle. \textbf{Why?}

- SAT oracle computes witnesses for outcomes.
- SAT oracle corresponds to a witness oracle (more later) [e.g. BKT93]
- SAT oracle queries can be expensive!

- Develop more efficient algorithms, i.e. with fewer oracle calls.
- Characterize query complexity of function problems.

- Use witness oracles instead of NP oracles.
How to solve function problems?

- SAT oracle ≠ (standard) NP oracle. **Why?**
  - SAT oracles compute witnesses for \( \forall \) outcomes
How to solve function problems?

- SAT oracle ≠ (standard) NP oracle. **Why?**
  - SAT oracles compute witnesses for **Y** outcomes
  - SAT oracle corresponds to a **witness oracle** (more later) [e.g. BKT93]
How to solve function problems?

- **SAT oracle** $\neq$ (standard) NP oracle. **Why?**
  - SAT oracles compute witnesses for $\mathbf{Y}$ outcomes
  - SAT oracle corresponds to a **witness oracle** (more later) [e.g. BKT93]

- SAT oracle queries can be expensive! **How to minimize queries?**
How to solve function problems?

- SAT oracle ≠ (standard) NP oracle. **Why?**
  - SAT oracles compute witnesses for \( \mathbb{Y} \) outcomes
  - SAT oracle corresponds to a **witness oracle** (more later) [e.g. BKT93]

- SAT oracle queries can be expensive! **How to minimize queries?**
  - Develop more efficient algorithms, i.e. with **fewer** oracle calls
How to solve function problems?

- **SAT oracle** $\neq$ (standard) NP oracle. **Why?**
  - SAT oracles compute witnesses for **Y** outcomes
  - SAT oracle corresponds to a witness oracle (more later) [e.g. BKT93]

- **SAT oracle queries** can be expensive! **How to minimize queries?**
  - Develop more efficient algorithms, i.e. with fewer oracle calls
  - Characterize query complexity of function problems
How to solve function problems?

- SAT oracle ≠ (standard) NP oracle. **Why?**
  - SAT oracles compute witnesses for **Y** outcomes
  - SAT oracle corresponds to a **witness oracle** (more later) [e.g. BKT93]

- SAT oracle queries can be expensive! **How to minimize queries?**
  - Develop more efficient algorithms, i.e. with **fewer** oracle calls
  - Characterize **query complexity** of function problems
    - But, use **witness oracles** instead of NP oracles
Problem solving with SAT oracles – general case

Decision Problems
- Model Checking
- AI Planning*
- CEGAR Loops
- ...

Function Problems
- Optimization Problems
- MaxSAT
- MinSAT
- PBO
- WBO
- ...

Minimal Sets
- MUSes
- Primes
- MFSes
- Min Models
- Autarkies
- MCSes
- Backbones
- ...

MFSes
Problem solving with SAT oracles – our work (2007-...)

Decision Problems
- Model Checking
- AI Planning*
- CEGAR Loops
- ...

Function Problems
- Optimization Problems
- MaxSAT
- MinSAT
- PBO
- ...
- Min Models
- Backbones
- ...

MUSes
MCSes
Primes
MFSes

But also, MESes, groups, variables, circuits, SMUS,...
Problem solving with SAT oracles – our work (2007-...)

Problem solving with SAT oracles

Decision Problems

- Model Checking
- AI Planning*
- CEGAR Loops

Function Problems

- Optimization Problems
  - MaxSAT
  - MinSAT
  - PBO
  - WBO

- Minimal Sets
- MUSes
- MCSes
- Primes
- MFSes
- Autarkies
- Backbones

But also, MESes, groups, variables, circuits, SMUS,...
Problem solving with SAT oracles – some challenges
Problem solving with SAT oracles – some challenges

Interfacing
SAT solver

SAT oracles
Problem solving with SAT oracles – some challenges

Interfacing SAT solver

SAT oracles

Algorithms: min sets & optimization

Reduce # SAT solver calls
Problem solving with SAT oracles – some challenges

Interfacing
SAT solver

Reduce # SAT oracle queries
SAT oracles

Algorithms: min sets & optimization
Problem solving with SAT oracles – some challenges

- Interfacing SAT solver
- Reduce # SAT oracle queries
- SAT oracles
- Query complexity
- Algorithms: min sets & optimization
Problem solving with SAT oracles – this talk

Query complexity

SAT oracles

Algorithms: min sets & optimization
Brief detour – some challenges

- **MUS**: [e.g. PW88, SP88, CD91, BDTW93, J01, J04, HLSB06, KBK09, K11, MSL11, BMS11, BLMS12, MSJB13]
  - Find $\mathcal{M} \subseteq \mathcal{F}$ s.t. $\mathcal{M}$ is unsatisfiable and $\mathcal{M}$ is irreducible
  - **Q1**: Algorithms for computing one MUS?
  - **Q2**: Worst-case number of queries to NP/SAT oracle to compute one MUS?

- **MCS**: [e.g. R87, BS05, OOF05, LS08, FSZ12, NBE12, MSHJPB13]
  - Find $\mathcal{C} \subseteq \mathcal{F}$ s.t. $\mathcal{F} \setminus \mathcal{C}$ is satisfiable and $\mathcal{C}$ is irreducible
  - **Q1**: Algorithms for computing one MCS?
  - **Q2**: Worst-case number of queries to NP/SAT oracle to compute one MCS?

- **Backbone**: [e.g. MZKST99, KK01, SW01, SKK03, KSTW05, MSJL10, ZWSM11]
  - Find set of literals common to all satisfying assignments of $\mathcal{F}$
  - **Q1**: Algorithms for computing the Backbone of $\mathcal{F}$?
  - **Q2**: Worst-case number of queries to NP/SAT oracle to compute the Backbone of $\mathcal{F}$?
Brief detour – some challenges

- **MUS**: [e.g. PW88,SP88,CD91,BDTW93,J01,J04,HLSB06,KBK09,K11,MSL11,BMS11,BLMS12,MSJB13]
  - Find $\mathcal{M} \subseteq \mathcal{F}$ s.t. $\mathcal{M}$ is unsatisfiable and $\mathcal{M}$ is irreducible
  - **Q1**: Algorithms for computing one MUS?
  - **Q2**: Worst-case number of queries to NP/SAT oracle to compute one MUS?

- **MCS**: [e.g. R87,BS05,OOF05,LS08,FSZ12,NBE12,MSHJPB13]
  - Find $\mathcal{C} \subseteq \mathcal{F}$ s.t. $\mathcal{F} \setminus \mathcal{C}$ is satisfiable and $\mathcal{C}$ is irreducible
  - **Q1**: Algorithms for computing one MCS?
  - **Q2**: Worst-case number of queries to NP/SAT oracle to compute one MCS?
Brief detour – some challenges

- **MUS**: [e.g. PW88, SP88, CD91, BDTW93, J01, J04, HLSB06, KBK09, K11, MSL11, BMS11, BLMS12, MSJB13]
  - Find $M \subseteq F$ s.t. $M$ is unsatisfiable and $M$ is irreducible
  - **Q1**: Algorithms for computing one MUS?
  - **Q2**: Worst-case number of queries to NP/SAT oracle to compute one MUS?

- **MCS**: [e.g. R87, BS05, OOF05, LS08, FSZ12, NBE12, MSHJPB13]
  - Find $C \subseteq F$ s.t. $F \setminus C$ is satisfiable and $C$ is irreducible
  - **Q1**: Algorithms for computing one MCS?
  - **Q2**: Worst-case number of queries to NP/SAT oracle to compute one MCS?

- **Backbone**: [e.g. MZKST99, KK01, SW01, SKK03, KSTW05, MSJL10, ZWSM11]
  - Find set of literals common to all satisfying assignments of $F$
  - **Q1**: Algorithms for computing the Backbone of $F$?
  - **Q2**: Worst-case number of queries to NP/SAT oracle to compute the Backbone of $F$?
Outline

Optimization Problems

Minimal Sets

Query Complexity

Conclusion
Outline

Optimization Problems

Minimal Sets

Query Complexity

Conclusion
Maximum satisfiability

Given unsatisfiable formula, find largest subset of clauses that is satisfiable
Maximum satisfiability

- Given unsatisfiable formula, find largest subset of clauses that is satisfiable
- A Minimal Correction Subset (MCS) is an irreducible relaxation of the formula
Maximum satisfiability

- Given unsatisfiable formula, find largest subset of clauses that is satisfiable
- A Minimal Correction Subset (MCS) is an irreducible relaxation of the formula
- The MaxSAT solution is one of the smallest MCSes
<table>
<thead>
<tr>
<th>Weights?</th>
<th>Hard Clauses?</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Hard Clauses?</td>
<td>No</td>
</tr>
<tr>
<td>---------------</td>
<td>----</td>
</tr>
<tr>
<td>Weights?</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>Plain</td>
</tr>
<tr>
<td>Yes</td>
<td>Weighted</td>
</tr>
<tr>
<td>Hard Clauses?</td>
<td>No</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Weights?</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td>Plain</td>
</tr>
<tr>
<td>Yes</td>
<td>Weighted</td>
</tr>
</tbody>
</table>

- Must satisfy hard clauses, if any
- Compute set of satisfied soft clauses with maximum cost
  - Without weights, cost of each falsified soft clause is 1
- Or, compute set of falsified soft clauses with minimum cost (s.t. hard & remaining soft clauses are satisfied)
### MaxSAT problem(s)

<table>
<thead>
<tr>
<th>Weights?</th>
<th>Hard Clauses?</th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Plain</td>
<td>Partial</td>
</tr>
<tr>
<td>No</td>
<td></td>
<td>Weighted</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td></td>
<td></td>
<td>Weighted Partial</td>
</tr>
</tbody>
</table>

- **Must** satisfy **hard** clauses, if any
- Compute set of satisfied **soft** clauses with **maximum cost**
  - Without weights, cost of each falsified soft clause is 1
- **Or**, compute set of falsified **soft** clauses with **minimum cost** (s.t. **hard** & remaining **soft** clauses are satisfied)
- **Note**: goal is to compute **set** of satisfied (or falsified) clauses; **not** just the cost!
Many MaxSAT algorithms
Many MaxSAT algorithms

MaxSAT Algorithms

- Branch & Bound
  - No unit prop; No cl. learning
- Iterative
- Model Guided
- Iterative MHS
- Core Guided
- Guided Iterative MHS & SAT
  - Relax cls given unsat cores
  - All cls relaxed

- Relax cls given models
Many MaxSAT algorithms

- Branch & Bound
- Model Guided
- Iterative MHS
- Core Guided
- Iterative

All cls relaxed

No unit prop; No cl. learning
Many MaxSAT algorithms

- Branch & Bound
- Model Guided
- Iterative MHS
- Core Guided
- Iterative

Relax cls given unsat cores
Many MaxSAT algorithms

- Branch & Bound
- Model Guided
- Iterative
- Iterative MHS
- Core Guided
- Iterative MHS & SAT

No unit prop; No cl. learning
All cls relaxed
Relax cls given
unsat cores
Iterative
models
Many MaxSAT algorithms

- Branch & Bound
- Model Guided
- Iterative MHS
- Core Guided
- Iterative

Relax cls given models
More on MaxSAT algorithms

- **Iterative:**
  - Linear search SAT/UNSAT (refine UB)
  - Linear search UNSAT/SAT (refine LB)
  - Binary search
  - Bit-based
  - Mixed linear/binary search

- **Core-guided:**
  - FM/(W)MSU1.X/WPM1
  - (W)MSU3
  - (W)MSU4
  - (W)PM2
  - Core-guided binary search (w/ disjoint cores)
    - Bin-Core, Bin-Core-Dis, Bin-Core-Dis2

- **Iterative minimal hitting set (MHS) computation**

- **Model guided approaches**

- **Branch & bound search**
MaxSAT with iterative SAT solving – definitions

- **Cost of assignment:**
  - Sum of weights of *falsified* clauses
MaxSAT with iterative SAT solving – definitions

- **Cost of assignment:**
  - Sum of weights of falsified clauses

  \[ \text{Cost} = \sum w_j \]

- **Optimum solution (OPT):**
  - Assignment with minimum cost

  \[ \text{OPT} \]

- **Upper Bound (UB):**
  - Assignment with cost \( \geq \text{OPT} \)

  \[ \sum c_j \in \phi \cdot w_j + 1; \text{hard clauses may be inconsistent} \]

- **Lower Bound (LB):**
  - No assignment with cost \( \leq \text{LB} \)

  \[ -1; \text{it may be possible to satisfy all soft clauses} \]

- **Relax each soft clause** \( c_j \): \((c_j \lor r_j)\) (on-demand in core-guided)
MaxSAT with iterative SAT solving – definitions

- **Cost of assignment:**
  - Sum of weights of falsified clauses

- **Optimum solution (OPT):**
  - Assignment with minimum cost

- **Upper Bound (UB):**
  - Assignment with cost $\geq$ OPT
  - E.g. $\sum_{c_j \in \varphi} w_j + 1$; hard clauses may be inconsistent

- **Lower Bound (LB):**
  - No assignment with cost $\leq$ LB
  - E.g. $-1$; it may be possible to satisfy all soft clauses
MaxSAT with iterative SAT solving – definitions

- **Cost of assignment:**
  - Sum of weights of falsified clauses

- **Optimum solution (OPT):**
  - Assignment with minimum cost

- **Upper Bound (UB):**
  - Assignment with cost $\geq$ OPT
  - E.g. $\sum_{c_j \in \varphi} w_j + 1$; hard clauses may be inconsistent

- **Lower Bound (LB):**
  - No assignment with cost $\leq$ LB
  - E.g. -1; it may be possible to satisfy all soft clauses

- **Relax** each soft clause $c_j$: $(c_j \lor r_j)$ (on-demand in core-guided)
MaxSAT with iterative SAT solving – refine UB

\( i \leftarrow 0 \)
\( UB_i \leftarrow \text{ComputeUB} \)

\( i \leftarrow i + 1 \)
\( UB_i \leftarrow \text{UpdateUB} \)

\( G \leftarrow F \cup (\sum w_j r_j < UB_i) \)

SAT(\( G \))?

\( \text{return } UB_{i-1} \)
MaxSAT with iterative SAT solving – refine UB

\[ i \leftarrow 0 \]
\[ UB_i \leftarrow \text{ComputeUB} \]

\[ i \leftarrow i + 1 \]
\[ UB_i \leftarrow \text{UpdateUB} \]

\[ G \leftarrow F \cup (\sum w_j r_j < UB_i) \]

\[ \text{SAT}(G)? \]

\[ \text{return } UB_{i-1} \]
MaxSAT with iterative SAT solving – refine UB

\[
i \leftarrow 0
UB_i \leftarrow \text{ComputeUB}
\]

\[
i \leftarrow i + 1
UB_i \leftarrow \text{UpdateUB}
\]

\[
G \leftarrow \mathcal{F} \cup (\sum w_j r_j < UB_i)
\]

\[
\text{SAT}(G)\
\]

\[
\text{yes}
\]

\[
\text{no}
\]

\[
\text{return } UB_{i-1}
\]

\[
\text{OPT}
\]

\[
\text{LB}
\]

\[
\text{UB}_2
\]

• Worst-case # of iterations exponential on instance size (# bits)

• Improvement: use binary search instead

• Many example solvers: Minisat+, SAT4J, QMaxSat

[ES06, LBP10, KZFH12]
MaxSAT with iterative SAT solving – refine UB

\[ i \leftarrow 0 \]
\[ UB_i \leftarrow \text{ComputeUB} \]

\[ i \leftarrow i + 1 \]
\[ UB_i \leftarrow \text{UpdateUB} \]

\[ G \leftarrow F \cup (\sum w_j r_j < UB_i) \]

\[ \text{SAT}(G)? \]

yes

no

return \( UB_{i-1} \)

Worst-case # of iterations
exponential on instance size (# bits)
– Improvement: use binary search instead

Many example solvers: Minisat+, SAT4J, QMaxSat

\[ \text{LB} \]
\[ \text{OPT} \]
\[ \text{UB}_k \]

\[ \text{UB} \]

\[ \text{LB} \]

\[ \text{OPT} \]

\[ \text{UB}_k \]
MaxSAT with iterative SAT solving – refine UB

\[ i \leftarrow 0 \]
\[ UB_i \leftarrow \text{ComputeUB} \]

\[ i \leftarrow i + 1 \]
\[ UB_i \leftarrow \text{UpdateUB} \]

\[ G \leftarrow F \cup (\sum w_j r_j < UB_i) \]

\[ \text{SAT}(G)? \]

- yes
  - \[ i \leftarrow 0 \]
  - \[ UB_i \leftarrow \text{ComputeUB} \]
- no
  - return \( UB_{i-1} \)

- Worst-case \# of iterations \textbf{exponential} on instance size (\# bits)
  - Improvement: use \textbf{binary search} instead

Many example solvers: Minisat+, SAT4J, QMaxSat

[ES06,LBP10,KZFH12]
MaxSAT with iterative SAT solving – refine UB

\[ i \leftarrow 0 \]
\[ UB_i \leftarrow \text{ComputeUB} \]

\[ i \leftarrow i + 1 \]
\[ UB_i \leftarrow \text{UpdateUB} \]

\[ G \leftarrow F \cup (\sum w_j r_j < UB_i) \]

\[ \text{SAT}(G)\? \]

- yes
- no

return \( UB_{i-1} \)

- Worst-case \# of iterations \textbf{exponential} on instance size (\# bits)
  - Improvement: use \textit{binary search} instead
- Many example solvers: Minisat+, SAT4J, QMaxSat [ES06, LBP10, KZFH12]
Core-guided MaxSAT – Fu&Malik’s algorithm

Example CNF formula

\[ x_6 \lor x_2 \quad \lnot x_6 \lor x_2 \quad \lnot x_2 \lor x_1 \quad \lnot x_1 \]

\[ \lnot x_6 \lor x_8 \quad x_6 \lor \lnot x_8 \quad x_2 \lor x_4 \quad \lnot x_4 \lor x_5 \]

\[ x_7 \lor x_5 \quad \lnot x_7 \lor x_5 \quad \lnot x_5 \lor x_3 \quad \lnot x_3 \]
Core-guided MaxSAT – Fu&Malik’s algorithm

\[
\begin{align*}
\varphi & = x_6 \lor x_2 \\
\varphi & = \neg x_6 \lor x_2 \\
\varphi & = \neg x_6 \lor \neg x_8 \\
\varphi & = x_6 \lor x_7 \\
\varphi & = x_5 \lor x_7 \\
\varphi & = x_5 \lor x_3 \\
\varphi & = \neg x_1 \\
\varphi & = x_1 \\
\varphi & = x_4 \\
\varphi & = \neg x_3 \\
\neg \varphi & = x_5 \lor x_4 \lor x_3
\end{align*}
\]

Formula is UNSAT; OPT \(\leq |\varphi| - 1\); Get unsat core
Core-guided MaxSAT – Fu&Malik’s algorithm

\[ \begin{align*}
\neg x_6 \vee x_2 \\
\neg \neg x_6 \vee x_8 \\
x_7 \vee x_5 \\
\sum_{i=1}^6 r_i \leq 1
\end{align*} \]

Add relaxation variables and AtMost1 constraint
Core-guided MaxSAT – Fu&Malik’s algorithm

Formula is (again) UNSAT; \( \text{OPT} \leq |\varphi| - 2 \); Get unsat core
Core-guided MaxSAT – Fu&Malik’s algorithm

Add new relaxation variables and AtMost1 constraint
Core-guided MaxSAT – Fu&Malik’s algorithm

\[
\begin{align*}
x_6 & \lor x_2 \lor r_7 & \neg x_6 & \lor x_2 \lor r_8 & \neg x_2 & \lor x_1 \lor r_1 \lor r_9 & \neg x_1 \lor r_2 \lor r_{10} \\
\neg x_6 & \lor x_8 & x_6 & \lor \neg x_8 & x_2 & \lor x_4 \lor r_3 & \neg x_4 & \lor x_5 \lor r_4 \\
x_7 & \lor x_5 \lor r_{11} & \neg x_7 & \lor x_5 \lor r_{12} & \neg x_5 & \lor x_3 \lor r_5 \lor r_{13} & \neg x_3 & \lor r_6 \lor r_{14} \\
\sum_{i=1}^{6} r_i & \leq 1 & \sum_{i=7}^{14} r_i & \leq 1
\end{align*}
\]

Instance is now SAT
Core-guided MaxSAT – Fu&Malik’s algorithm

\[ x_6 \lor x_2 \lor r_7 \quad \neg x_6 \lor x_2 \lor r_8 \quad \neg x_2 \lor x_1 \lor r_1 \lor r_9 \quad \neg x_1 \lor r_2 \lor r_{10} \]

\[ \neg x_6 \lor x_8 \quad x_6 \lor \neg x_8 \quad x_2 \lor x_4 \lor r_3 \quad \neg x_4 \lor x_5 \lor r_4 \]

\[ x_7 \lor x_5 \lor r_{11} \quad \neg x_7 \lor x_5 \lor r_{12} \quad \neg x_5 \lor x_3 \lor r_5 \lor r_{13} \quad \neg x_3 \lor r_6 \lor r_{14} \]

\[ \sum_{i=1}^{6} r_i \leq 1 \quad \sum_{i=7}^{14} r_i \leq 1 \]

MaxSAT solution is \(|\varphi| - I = 12 - 2 = 10\)
MaxSAT solving with SAT oracles

- A sample of recent algorithms:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th># Oracle Queries</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear search SU</td>
<td>Exponential***</td>
<td>[e.g. LBP10]</td>
</tr>
<tr>
<td>Binary search</td>
<td>Linear*</td>
<td>[e.g. FM06]</td>
</tr>
<tr>
<td>FM/WMSU1/WPM1</td>
<td>Exponential**</td>
<td>[FM06, MSM08, MMSP09, ABL09a, ABGL12]</td>
</tr>
<tr>
<td>WPM2</td>
<td>Exponential**</td>
<td>[ABL10, ABGL13]</td>
</tr>
<tr>
<td>Bin-Core-Dis</td>
<td>Linear</td>
<td>[HMMS11, MHMS12]</td>
</tr>
<tr>
<td>Iterative MHS</td>
<td>Exponential</td>
<td>[DB11, DB13a, DB13b]</td>
</tr>
</tbody>
</table>

* $O(\log m)$ queries with SAT oracle, for (partial) unweighted MaxSAT
** Weighted case; depends on computed cores
*** On # bits of problem instance (due to weights)

- Example MaxSAT solvers:
  - MSUnCore; WPM1, WPM2; QMaxSAT; SAT4J; etc.
Other optimization problems

- MinSAT
- MaxSAT
- WBO
- PBO / 0-1 ILP
- BCP
- UCP

Minimize number of satisfied clauses
Minimize/maximize linear cost function given linear inequalities on boolean variables
PBO if clauses instead of linear inequalities
BCP if positive clauses
Can reduce to/from MaxSAT
Extensive work on CNF encodings
Solution approaches mimic the ones for MaxSAT
Other optimization problems

- **MinSAT**
  - Minimize # satisfied cls

- **MaxSAT**
  - Min/max linear cost function given linear inequalities on boolean vars
  - PBO if cls instead of linear inequalities
  - BCP if positive cls
  - Can reduce to/from MaxSAT

- **PBO / 0-1 ILP**
- **WBO**
- **BCP**
- **UCP**

Extensive work on CNF encodings
Solution approaches mimic the ones for MaxSAT
Other optimization problems

- **MinSAT**
- **MaxSAT**
- **WBO**
- **PBO / 0-1 ILP**
- **BCP**
- **UCP**

Min/Max linear cost function given linear inequalities on boolean vars

Extensive work on CNF encodings

Solution approaches mimic the ones for MaxSAT
Other optimization problems

Optimization Problems

MinSAT

MaxSAT

WBO

PBO / 0-1 ILP

BCP

UCP

PBO if cls instead of linear inequalities

Extensive work on CNF encodings

Solution approaches mimic the ones for MaxSAT
Other optimization problems

- MinSAT
- MaxSAT
- WBO
- PBO / 0-1 ILP
- BCP
- UCP

Optimization Problems

Minimize # satisfied cls
Min/max linear cost function given linear inequalities on boolean vars
PBO if cls instead of linear inequalities
BCP if positive cls
Can reduce to/from MaxSAT

Extensive work on CNF encodings
Solution approaches mimic the ones for MaxSAT
Other optimization problems

- **Optimization Problems**
- **MinSAT**
- **MaxSAT**
- **WBO**
- **PBO / 0-1 ILP**
- **BCP**
- **UCP**

- **Can reduce to/from MaxSAT**
- **Extensive work on CNF encodings**
- **Solution approaches mimic the ones for MaxSAT**
Optimization Problems

- MinSAT
- MaxSAT
- WBO
- PBO / 0-1 ILP
- BCP
- UCP

Can reduce to/from MaxSAT
Extensive work on CNF encodings
Solution approaches mimic the ones for MaxSAT
Outline

Optimization Problems

Minimal Sets

Query Complexity

Conclusion
Computing minimal sets is ubiquitous!

- MUSes are minimal sets
  - Extensive work since the mid 80s
Computing minimal sets is ubiquitous!

- Backbones(!) are **minimal sets**
  - Extensive work since the **late 90s**
Computing minimal sets is ubiquitous!

- MCSes are minimal sets
  - Extensive work since the mid 80s
Computing minimal sets is ubiquitous!

- Autarkies(!) & primes are also **minimal sets**
  - Extensive work since the 80s & 30s(!), resp.
Computing minimal sets is ubiquitous!

- MESes, MFSes (and many more!) are minimal sets
  - Work since the 00s & 90s, etc.
Computing minimal sets is ubiquitous!

- Develop framework for reasoning about minimal sets!
  - Why? Common algorithms & techniques; new insights & results; ...
Example – MUSes as minimal sets

\[(\bar{x}_1 \lor \bar{x}_2) \land (x_1) \land (x_5 \lor x_6) \land (\bar{x}_3 \lor \bar{x}_4) \land (x_2) \land (x_3) \land (x_4)\]

- Formula is unsatisfiable but **not** irreducible
Example – MUSes as minimal sets

\begin{align*}
(x_1 &\lor \bar{x}_2) (\bar{x}_1) (x_5 \lor x_6) (\bar{x}_3 \lor \bar{x}_4) (x_2) (x_3) (x_4)
\end{align*}

- Formula is unsatisfiable but \textbf{not} irreducible
- Can remove clauses, and formula still \textbf{unsatisfiable}
Example – MUSes as minimal sets

- Formula is unsatisfiable but **not** irreducible
- Can remove clauses, and formula still unsatisfiable

- **Minimal Unsatisfiable Subset (MUS):**
  - Irreducible subformula that is unsatisfiable
    - MUSes are minimal sets
Example – MUSes as minimal sets

- Formula is unsatisfiable but **not** irreducible
- Can remove clauses, and formula still unsatisfiable

**Minimal Unsatisfiable Subset (MUS):**
- Irreducible subformula that is unsatisfiable
  - MUSes are minimal sets
Example – MUSes as minimal sets

- Formula is unsatisfiable but **not** irreducible
- Can remove clauses, and formula still unsatisfiable

- **Minimal Unsatisfiable Subset (MUS):**
  - Irreducible subformula that is unsatisfiable
    - MUSes are minimal sets

- Complexity results:
  - Decision problem: $D^P$-complete
  - Function problem: in $FP^{NP}$ with lower bound in $FP_{\|}^{NP}$ [PW88, CT95]
Example – MCSes as minimal sets

\[(\bar{x}_1 \lor \bar{x}_2) \land (x_1) \land (x_5 \lor x_6) \land (\bar{x}_3 \lor \bar{x}_4) \land (x_2) \land (x_3) \land (x_4)\]

- Formula is unsatisfiable with satisfiable subformulas
Example – MCSes as minimal sets

- Formula is **unsatisfiable** with **satisfiable** subformulas
- Can remove clauses such that remaining clauses are **satisfiable**
Example – MCSes as minimal sets

- Formula is **unsatisfiable** with **satisfiable** subformulas
- Can remove clauses such that remaining clauses are **satisfiable**

- **Minimal Correction Subset (MCS):**
  - Irreducible subformula such that the complement is **satisfiable**
    - MCSes are minimal sets
Example – MCSes as minimal sets

\[(\bar{x}_1 \lor \bar{x}_2) (x_1) (x_5 \lor x_6) (\bar{x}_3 \lor \bar{x}_4)\]

- Formula is **unsatisfiable** with **satisfiable** subformulas
- Can remove clauses such that remaining clauses are **satisfiable**

- **Minimal Correction Subset (MCS):**
  - Irreducible subformula such that the complement is **satisfiable**
    - MCSes are minimal sets
Example – MCSes as minimal sets

\[(\bar{x}_1 \lor \bar{x}_2) (x_1) (x_5 \lor x_6) (\bar{x}_3 \lor \bar{x}_4) (x_2) (x_3) (x_4)\]

- Formula is **unsatisfiable** with **satisfiable** subformulas
- Can remove clauses such that remaining clauses are **satisfiable**

- **Minimal Correction Subset (MCS):**
  - Irreducible subformula such that the complement is **satisfiable**
    - MCSes are minimal sets
Example – MCSes as minimal sets

\[(\bar{x}_1 \lor \bar{x}_2) \quad (x_1) \quad (x_5 \lor x_6) \quad (\bar{x}_3 \lor \bar{x}_4) \quad (x_2) \quad (x_3) \quad (x_4)\]

- Formula is unsatisfiable with satisfiable subformulas
- Can remove clauses such that remaining clauses are satisfiable
- **Minimal Correction Subset (MCS):**
  - Irreducible subformula such that the complement is satisfiable
  - MCSes are minimal sets

- Complexity results:
  - Function problem: can be solved with $O(\log n)$ calls to a SAT oracle

Why?
Example – MCSes as minimal sets

\[(\overline{x}_1 \lor \overline{x}_2) \cdot (x_1) \cdot (x_5 \lor x_6) \cdot (x_3 \lor \overline{x}_4) \cdot (x_2) \cdot (x_3) \cdot (x_4)\]

- Formula is **unsatisfiable** with **satisfiable** subformulas
- Can remove clauses such that remaining clauses are **satisfiable**

**Minimal Correction Subset (MCS):**
- Irreducible subformula such that the complement is **satisfiable**
  - MCSes are minimal sets

**Complexity results:**
- Function problem: can be solved with \(O(\log n)\) calls to a SAT oracle. **Why?**
Monotone predicates

- Set of elements $\mathcal{R}$
- Predicate $P : 2^\mathcal{R} \rightarrow \{0, 1\}$

1. Given $\mathcal{R}$ and monotone predicate $P$ over $\mathcal{R}$,
2. compute minimal set $M \subseteq \mathcal{R}$ such that $P(M) = 1$ holds
Monotone predicates

- Set of elements $\mathcal{R}$
- Predicate $P : 2^\mathcal{R} \rightarrow \{0, 1\}$
- $P$ is **monotone** iff $P$ has the following property:
  
  \[ \Rightarrow \text{ If } P(\mathcal{R}_0) = 1 \text{ holds and } \mathcal{R}_0 \subseteq \mathcal{R}_1 \subseteq \mathcal{R}, \text{ then } P(\mathcal{R}_1) = 1 \text{ also holds} \]
  
  - Note: $P(\mathcal{R}) = 1$ must hold; otherwise no minimal set
Monotone predicates

- Set of elements $\mathcal{R}$
- Predicate $P : 2^\mathcal{R} \rightarrow \{0, 1\}$
- $P$ is monotone iff $P$ has the following property:

$$\Rightarrow \quad \text{If } P(\mathcal{R}_0) = 1 \text{ holds and } \mathcal{R}_0 \subseteq \mathcal{R}_1 \subseteq \mathcal{R}, \text{ then } P(\mathcal{R}_1) = 1 \text{ also holds}$$

- Note: $P(\mathcal{R}) = 1$ must hold; otherwise no minimal set

- Minimal Set over Monotone Predicate (MSMP) problem:  
  1. Given $\mathcal{R}$ and monotone predicate $P$ over $\mathcal{R}$,
  2. compute minimal set $\mathcal{M} \subseteq \mathcal{R}$ such that $P(\mathcal{M}) = 1$ holds
Example reductions to MSMP

<table>
<thead>
<tr>
<th>( R )</th>
<th>MUS</th>
<th>MCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P(\mathcal{W}), \mathcal{W} \subseteq R )</td>
<td>( \neg \text{SAT}(\mathcal{W}) )</td>
<td>( \text{SAT}(\mathcal{F} \setminus \mathcal{W}) )</td>
</tr>
<tr>
<td>Min. set ( \mathcal{M} ), ( P(\mathcal{M}) )</td>
<td>( \neg \text{SAT}(\mathcal{M}) ) true</td>
<td>( \text{SAT}(\mathcal{F} \setminus \mathcal{M}) ) true</td>
</tr>
<tr>
<td>( \forall \mathcal{M}' \subseteq \mathcal{M}, P(\mathcal{M'}) )</td>
<td>( \neg \text{SAT}(\mathcal{M'}) ) false</td>
<td>( \text{SAT}(\mathcal{F} \setminus \mathcal{M'}) ) false</td>
</tr>
</tbody>
</table>
Example reductions to MSMP

<table>
<thead>
<tr>
<th>$\mathcal{R}$</th>
<th>$\mathcal{F}$</th>
<th>$\mathcal{F}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(\mathcal{W}), \mathcal{W} \subseteq \mathcal{R}$</td>
<td>$\neg \text{SAT}(\mathcal{W})$</td>
<td>$\text{SAT}(\mathcal{F} \setminus \mathcal{W})$</td>
</tr>
<tr>
<td>Min. set $\mathcal{M}, P(\mathcal{M})$</td>
<td>$\neg \text{SAT}(\mathcal{M})$ true</td>
<td>$\text{SAT}(\mathcal{F} \setminus \mathcal{M})$ true</td>
</tr>
<tr>
<td>$\forall \mathcal{M}' \subset \mathcal{M}, P(\mathcal{M}')$</td>
<td>$\neg \text{SAT}(\mathcal{M}')$ false</td>
<td>$\text{SAT}(\mathcal{F} \setminus \mathcal{M}')$ false</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_4$</th>
<th>$c_5$</th>
<th>$c_6$</th>
<th>$c_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1 \lor \bar{x}_2$</td>
<td>$(x_1)$</td>
<td>$(x_5 \lor x_6)$</td>
<td>$(\bar{x}_3 \lor \bar{x}_4)$</td>
<td>$(x_2)$</td>
<td>$(x_3)$</td>
<td>$(x_4)$</td>
</tr>
</tbody>
</table>

**MUS:** $\mathcal{W}$

\[
P(\mathcal{W}) \triangleq \neg \text{SAT}(\mathcal{W})
\]

<table>
<thead>
<tr>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_4$</th>
<th>$c_5$</th>
<th>$c_6$</th>
<th>$c_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$ $C_2$ $C_3$ $C_4$ $C_5$ $C_6$ $C_7$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_2$ $C_3$ $C_4$ $C_5$ $C_6$ $C_7$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_3$ $C_4$ $C_5$ $C_6$ $C_7$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_4$ $C_5$ $C_6$ $C_7$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_4$ $C_6$ $C_7$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_4$ $C_7$</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C_6$ $C_7$</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Example reductions to MSMP

<table>
<thead>
<tr>
<th>$\mathcal{R}$</th>
<th>MUS $\mathcal{F}$</th>
<th>MCS $\mathcal{F}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(W), W \subseteq \mathcal{R}$</td>
<td>$\neg \text{SAT}(W)$</td>
<td>$\text{SAT}(\mathcal{F} \setminus W)$</td>
</tr>
<tr>
<td>Min. set $\mathcal{M}$, $P(\mathcal{M})$</td>
<td>$\neg \text{SAT}(\mathcal{M})$ true</td>
<td>$\text{SAT}(\mathcal{F} \setminus \mathcal{M})$ true</td>
</tr>
<tr>
<td>$\forall \mathcal{M'} \subset \mathcal{M}, P(\mathcal{M'})$</td>
<td>$\neg \text{SAT}(\mathcal{M'})$ false</td>
<td>$\text{SAT}(\mathcal{F} \setminus \mathcal{M'})$ false</td>
</tr>
</tbody>
</table>
Example reductions to MSMP

<table>
<thead>
<tr>
<th></th>
<th>MUS</th>
<th>MCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{R}$</td>
<td>$\mathcal{F}$</td>
<td>$\mathcal{F}$</td>
</tr>
<tr>
<td>$P(W), W \subseteq \mathcal{R}$</td>
<td>$\neg \text{SAT}(W)$</td>
<td>$\text{SAT}(\mathcal{F} \setminus W)$</td>
</tr>
<tr>
<td>Min. set $\mathcal{M}, P(\mathcal{M})$</td>
<td>$\neg \text{SAT}(\mathcal{M})$ true</td>
<td>$\text{SAT}(\mathcal{F} \setminus \mathcal{M})$ true</td>
</tr>
<tr>
<td>$\forall \mathcal{M}' \subseteq \mathcal{M}, P(\mathcal{M}')$</td>
<td>$\neg \text{SAT}(\mathcal{M}')$ false</td>
<td>$\text{SAT}(\mathcal{F} \setminus \mathcal{M}')$ false</td>
</tr>
</tbody>
</table>

### Example:

<table>
<thead>
<tr>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_4$</th>
<th>$c_5$</th>
<th>$c_6$</th>
<th>$c_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\overline{x}_1 \lor \overline{x}_2$</td>
<td>$x_1$</td>
<td>$x_5 \lor x_6$</td>
<td>$\overline{x}_3 \lor \overline{x}_4$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$x_4$</td>
</tr>
</tbody>
</table>

#### MCS:

<table>
<thead>
<tr>
<th>$\mathcal{W}$</th>
<th>$\mathcal{F} \setminus \mathcal{W}$</th>
<th>$P(\mathcal{W}) \triangleq \text{SAT}(\mathcal{F} \setminus \mathcal{W})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1 c_2 c_3 c_4 c_5 c_6 c_7$</td>
<td>$\emptyset$</td>
<td>1</td>
</tr>
<tr>
<td>$c_1 c_2 c_3 c_4 c_5 c_7$</td>
<td>$c_6$</td>
<td>1</td>
</tr>
<tr>
<td>$c_1 c_2 c_3 c_5 c_7$</td>
<td>$c_4 c_6$</td>
<td>1</td>
</tr>
<tr>
<td>$c_1 c_2 c_5 c_7$</td>
<td>$c_3 c_4 c_6$</td>
<td>1</td>
</tr>
<tr>
<td>$c_1 c_5 c_7$</td>
<td>$c_2 c_3 c_4 c_6$</td>
<td>1</td>
</tr>
<tr>
<td>$c_5 c_7$</td>
<td>$c_1 c_2 c_3 c_4 c_6$</td>
<td>1</td>
</tr>
<tr>
<td>$c_5$</td>
<td>$c_1 c_2 c_3 c_4 c_6 c_7$</td>
<td>0</td>
</tr>
<tr>
<td>$c_7$</td>
<td>$c_1 c_2 c_3 c_4 c_5 c_6$</td>
<td>0</td>
</tr>
<tr>
<td>Problem</td>
<td>$\mathcal{R}$</td>
<td>$P(\mathcal{W}), \mathcal{W} \subseteq \mathcal{R}$</td>
</tr>
<tr>
<td>----------</td>
<td>---------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>FMUS</td>
<td>$\mathcal{F}$</td>
<td>$\neg \text{SAT}(\bigwedge_{c \in \mathcal{W}} (c))$</td>
</tr>
<tr>
<td>FMCS</td>
<td>$\mathcal{F}$</td>
<td>$\text{SAT}(\bigwedge_{c \in \mathcal{R} \setminus \mathcal{W}} (c))$</td>
</tr>
<tr>
<td>FMES</td>
<td>$\mathcal{F}$</td>
<td>$\neg \text{SAT}(\neg \mathcal{F} \land \bigwedge_{c \in \mathcal{W}} (c))$</td>
</tr>
<tr>
<td>FMDS</td>
<td>$\mathcal{F}$</td>
<td>$\text{SAT}(\neg \mathcal{F} \land \bigwedge_{c \in \mathcal{R} \setminus \mathcal{W}} (c))$</td>
</tr>
<tr>
<td>FCMFS</td>
<td>$\mathcal{F}$</td>
<td>$\text{SAT}(\bigwedge_{c \in \mathcal{R} \setminus \mathcal{W}} (\neg c))$</td>
</tr>
<tr>
<td>FMnM</td>
<td>$\mathcal{X}$</td>
<td>$\text{SAT}(\mathcal{F} \land \bigwedge_{x \in \mathcal{R} \setminus \mathcal{W}} (\neg x))$</td>
</tr>
<tr>
<td>FPlt</td>
<td>$L(t)$</td>
<td>$\neg \text{SAT}(\neg \mathcal{F} \land \bigwedge_{l \in \mathcal{W}} (l))$</td>
</tr>
<tr>
<td>FPlc</td>
<td>$L(c)$</td>
<td>$\neg \text{SAT}(\mathcal{F} \land \bigwedge_{l \in \mathcal{W}} (\neg l))$</td>
</tr>
<tr>
<td>FLEIt</td>
<td>$L_t$</td>
<td>$\neg \text{SAT}(\mathcal{F}^{ltX} \land (\forall l \in \mathcal{R} \setminus \mathcal{W} \neg l))$</td>
</tr>
<tr>
<td>FLElc</td>
<td>$L_c$</td>
<td>$\neg \text{SAT}(\mathcal{F}^{lcX} \land (\forall l \in \mathcal{R} \setminus \mathcal{W} l))$</td>
</tr>
<tr>
<td>FMnES</td>
<td>$\mathcal{I}$</td>
<td>$\neg \text{SAT}(\neg \mathcal{I} \land \bigwedge_{c \in \mathcal{W}} (c))$</td>
</tr>
<tr>
<td>FMxEs</td>
<td>$\mathcal{N}$</td>
<td>$\neg \text{SAT}(\mathcal{I} \land (\forall c \in \mathcal{R} \setminus \mathcal{W} \neg c))$</td>
</tr>
<tr>
<td>FBBr</td>
<td>$\mathcal{V}$</td>
<td>$\neg \text{SAT}(\mathcal{F} \land (\forall l \in \mathcal{R} \setminus \mathcal{W} l))$</td>
</tr>
<tr>
<td>FBB</td>
<td>$\mathcal{X}$</td>
<td>$\neg \text{SAT}(\mathcal{F}^{BB} \land (\forall x \in \mathcal{R} \setminus \mathcal{W} x \land \neg x'))$</td>
</tr>
<tr>
<td>FVInd</td>
<td>$\mathcal{X}$</td>
<td>$\neg \text{SAT}(\mathcal{F}^{VInd} \land (\forall x_i \in \mathcal{W} (x_i \leftrightarrow y_i))$</td>
</tr>
<tr>
<td>FAut</td>
<td>$\mathcal{X}^+$</td>
<td>$\text{SAT}(\mathcal{F}^{Aut} \land \bigwedge_{x^+ \in \mathcal{R} \setminus \mathcal{W}} (x^+))$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
### Reductions to MSMP – a glimpse

<table>
<thead>
<tr>
<th>Problem</th>
<th>$\mathcal{R}$</th>
<th>$P(\mathcal{W}), \mathcal{W} \subseteq \mathcal{R}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMUS</td>
<td>$\mathcal{F}$</td>
<td>$\neg \text{SAT}(\land_{c \in \mathcal{W}}(c))$</td>
</tr>
<tr>
<td>FMCS</td>
<td>$\mathcal{F}$</td>
<td>$\text{SAT}(\land_{c \in \mathcal{R} \setminus \mathcal{W}}(c))$</td>
</tr>
<tr>
<td>FMES</td>
<td>$\mathcal{F}$</td>
<td>$\neg \text{SAT}(\neg \mathcal{F} \land \land_{c \in \mathcal{W}}(c))$</td>
</tr>
<tr>
<td>FMDS</td>
<td>$\mathcal{F}$</td>
<td>$\text{SAT}(\neg \mathcal{F} \land \land_{c \in \mathcal{R} \setminus \mathcal{W}}(c))$</td>
</tr>
<tr>
<td>FCMFS</td>
<td>$\mathcal{F}$</td>
<td>$\text{SAT}(\land_{c \in \mathcal{R} \setminus \mathcal{W}}(\neg c))$</td>
</tr>
<tr>
<td>FMnM</td>
<td>$\mathcal{X}$</td>
<td>$\text{SAT}(\mathcal{F} \land \land_{x \in \mathcal{R} \setminus \mathcal{W}}(\neg x))$</td>
</tr>
<tr>
<td>FPlt</td>
<td>$L(t)$</td>
<td>$\neg \text{SAT}(\neg \mathcal{F} \land \land_{l \in \mathcal{W}}(l))$</td>
</tr>
<tr>
<td>FPlc</td>
<td>$L(c)$</td>
<td>$\neg \text{SAT}(\mathcal{F} \land \land_{l \in \mathcal{W}}(\neg l))$</td>
</tr>
<tr>
<td>FLEIt</td>
<td>$\mathcal{L}_t$</td>
<td>$\neg \text{SAT}(\mathcal{F}^{ltX} \land (\lor_{l \in \mathcal{R} \setminus \mathcal{W}}(\neg l)))$</td>
</tr>
<tr>
<td>FLElc</td>
<td>$\mathcal{L}_c$</td>
<td>$\neg \text{SAT}(\mathcal{F}^{lcX} \land (\lor_{l \in \mathcal{R} \setminus \mathcal{W}}(l)))$</td>
</tr>
<tr>
<td>FMnES</td>
<td>$\mathcal{J}$</td>
<td>$\neg \text{SAT}(\neg \mathcal{I} \land \land_{c \in \mathcal{W}}(c))$</td>
</tr>
<tr>
<td>FMxEs</td>
<td>$\mathcal{N}$</td>
<td>$\neg \text{SAT}(\mathcal{J} \land (\lor_{c \in \mathcal{R} \setminus \mathcal{W}}(\neg c)))$</td>
</tr>
<tr>
<td>FBBr</td>
<td>$\mathcal{V}$</td>
<td>$\neg \text{SAT}(\mathcal{F} \land (\lor_{l \in \mathcal{R} \setminus \mathcal{W}}(\neg l)))$</td>
</tr>
<tr>
<td>FBB</td>
<td>$\mathcal{X}$</td>
<td>$\neg \text{SAT}(\mathcal{F}^{BB} \land (\lor_{x \in \mathcal{R} \setminus \mathcal{W}}(x \land \neg x'))) $</td>
</tr>
<tr>
<td>FVInd</td>
<td>$\mathcal{X}$</td>
<td>$\neg \text{SAT}(\mathcal{F}^{VInd} \land \land_{x_i \in \mathcal{W}}(x_i \leftrightarrow y_i))$</td>
</tr>
<tr>
<td>FAut</td>
<td>$\mathcal{X}^+$</td>
<td>$\text{SAT}(\mathcal{F}^{Aut} \land \land_{x^+ \in \mathcal{R} \setminus \mathcal{W}}(x^+))$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Why MSMP algorithms?

Adapt algorithms for MUS extraction

- Insertion; Deletion; Dichotomic; QuickXplain; Progression

Worst-case number of predicate tests:

- Set $R$ with $m$ elements and $k$ the size of largest minimal subset

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Predicate tests</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insertion (Default)</td>
<td>$O(m \times k)$</td>
<td>[SP88,vMW08]</td>
</tr>
<tr>
<td>Deletion (Default)</td>
<td>$O(m)$</td>
<td>[CD91,BDTW93]</td>
</tr>
<tr>
<td>Dichotomic</td>
<td>$O(k \times \log m)$</td>
<td>[HLSB06]</td>
</tr>
<tr>
<td>QuickXplain</td>
<td>$O(k \times (1 + \log_m k))$</td>
<td>[J01,J04]</td>
</tr>
<tr>
<td>Progression</td>
<td>$O(k \times \log(1 + m/k))$</td>
<td>[MSJB13]</td>
</tr>
</tbody>
</table>

- For MUSes/MCSes/PIs/MMs/MESes/etc. each predicate test represents one query to a SAT oracle

MSMP algorithms can integrate well-known pruning techniques

- Clause set refinement; Model rotation; etc.*

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MSMP algorithms

- Why MSMP algorithms? Common algorithms & techniques, ...

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Predicate tests</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insertion</td>
<td>$O(m \times k)$</td>
<td>[SP88,vMW08]</td>
</tr>
<tr>
<td>Deletion</td>
<td>$O(m)$</td>
<td>[CD91,BDTW93]</td>
</tr>
<tr>
<td>Dichotomic</td>
<td>$O(k \times \log m)$</td>
<td>[HLSB06]</td>
</tr>
<tr>
<td>QuickXplain</td>
<td>$O(k \times (1 + \log m/k))$</td>
<td>[J01,J04]</td>
</tr>
<tr>
<td>Progression</td>
<td>$O(k \times \log(1 + m/k))$</td>
<td>[MSJB13]</td>
</tr>
</tbody>
</table>

- For MUSes/MCSes/PIs/MMs/MESes/etc. each predicate test represents one query to a SAT oracle.
- MSMP algorithms can integrate well-known pruning techniques - Clause set refinement; Model rotation; etc. [BDTW93,DHN06,MSL11,BLMS12]
MSMP algorithms

- Why MSMP algorithms? Common algorithms & techniques, ...
- Adapt algorithms for MUS extraction
  - Insertion; Deletion; Dichotomic; QuickXplain; Progression
- Worst-case number of predicate tests:
  - Set $\mathcal{R}$ with $m$ elements and $k$ the size of largest minimal subset

<table>
<thead>
<tr>
<th>Algorithm</th>
<th># Predicate tests</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insertion (Default)</td>
<td>$O(m \times k)$</td>
<td>[SP88, VMW08]</td>
</tr>
<tr>
<td>Deletion (Default)</td>
<td>$O(m)$</td>
<td>[CD91, BDTW93]</td>
</tr>
<tr>
<td>Dichotomic</td>
<td>$O(k \times \log m)$</td>
<td>[HLSB06]</td>
</tr>
<tr>
<td>QuickXplain</td>
<td>$O(k \times (1 + \log \frac{m}{k}))$</td>
<td>[J01, J04]</td>
</tr>
<tr>
<td>Progression</td>
<td>$O(k \times \log(1 + \frac{m}{k}))$</td>
<td>[MSJB13]</td>
</tr>
</tbody>
</table>

- For MUSes/MCSes/PIs/MMs/MESes/etc. each predicate test represents one query to a SAT oracle
MSMP algorithms

- Why MSMP algorithms? Common algorithms & techniques, ...
- Adapt algorithms for MUS extraction
  - Insertion; Deletion; Dichotomic; QuickXplain; Progression
- Worst-case number of predicate tests:
  - Set $\mathcal{R}$ with $m$ elements and $k$ the size of largest minimal subset

<table>
<thead>
<tr>
<th>Algorithm</th>
<th># Predicate tests</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insertion (Default)</td>
<td>$\mathcal{O}(m \times k)$</td>
<td>[SP88, vMW08]</td>
</tr>
<tr>
<td>Deletion (Default)</td>
<td>$\mathcal{O}(m)$</td>
<td>[CD91, BDTW93]</td>
</tr>
<tr>
<td>Dichotomic</td>
<td>$\mathcal{O}(k \times \log m)$</td>
<td>[HLSB06]</td>
</tr>
<tr>
<td>QuickXplain</td>
<td>$\mathcal{O}(k \times (1 + \log \frac{m}{k}))$</td>
<td>[J01, J04]</td>
</tr>
<tr>
<td>Progression</td>
<td>$\mathcal{O}(k \times \log(1 + \frac{m}{k}))$</td>
<td>[MSJB13]</td>
</tr>
</tbody>
</table>

- For MUSes/MCSes/PIs/MMs/MESes/etc. each predicate test represents one query to a SAT oracle

$\mathcal{O}(m)$ calls for last 4!
MSMP algorithms

• Why MSMP algorithms? Common algorithms & techniques, ...

• Adapt algorithms for MUS extraction
  – Insertion; Deletion; Dichotomic; QuickXplain; Progression

• Worst-case number of predicate tests:
  – Set $\mathcal{R}$ with $m$ elements and $k$ the size of largest minimal subset

<table>
<thead>
<tr>
<th>Algorithm</th>
<th># Predicate tests</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insertion (Default)</td>
<td>$\mathcal{O}(m \times k)$</td>
<td>[SP88,vMW08]</td>
</tr>
<tr>
<td>Deletion (Default)</td>
<td>$\mathcal{O}(m)$</td>
<td>[CD91,BDTW93]</td>
</tr>
<tr>
<td>Dichotomic</td>
<td>$\mathcal{O}(k \times \log m)$</td>
<td>[HLSB06]</td>
</tr>
<tr>
<td>QuickXplain</td>
<td>$\mathcal{O}(k \times (1 + \log \frac{m}{k}))$</td>
<td>[J01,J04]</td>
</tr>
<tr>
<td>Progression</td>
<td>$\mathcal{O}(k \times \log(1 + \frac{m}{k}))$</td>
<td>[MSJB13]</td>
</tr>
</tbody>
</table>

– For MUSes/MCSes/PIs/MMs/MESes/etc. each predicate test represents one query to a SAT oracle

$\mathcal{O}(m)$ calls for last 4!

• MSMP algorithms can integrate well-known pruning techniques
  – Clause set refinement; Model rotation; etc.* [BDTW93,DHN06,MSL11,BLMS12]
Deletion algorithm

Input: Target set $T$
Output: Minimal subset $M$

begin

$M \leftarrow T$  \hspace{1cm} // Precondition: $P(T)$ holds

foreach $u \in M$ do

  if $P(M \setminus \{u\})$ then  \hspace{1cm} // $P$ holds without element

    $M \leftarrow M \setminus \{u\}$  \hspace{1cm} // Drop element

return $M$  \hspace{1cm} // Postcondition: $M$ is minimal set s.t. $P(M)$ holds

end


- Number of predicate tests: $O(m)$
Deletion – MUS example

\[
\begin{array}{cccccccc}
  c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \\
  (\bar{x}_1 \lor \bar{x}_2) & (x_1) & (x_5 \lor x_6) & (\bar{x}_3 \lor \bar{x}_4) & (x_2) & (x_3) & (x_4)
\end{array}
\]

- **MUS** predicate test: \( W \triangleq M \setminus \{c_i\}, P(W) \triangleq \neg \text{SAT}(W) \)

\[
\begin{array}{cccccc}
  c_i & M & M \setminus \{c_i\} & P(W) & \text{Outcome}
\end{array}
\]
Deletion – MUS example

<table>
<thead>
<tr>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_4$</th>
<th>$c_5$</th>
<th>$c_6$</th>
<th>$c_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\bar{x}_1 \lor \bar{x}_2)$</td>
<td>$(x_1)$</td>
<td>$(x_5 \lor x_6)$</td>
<td>$(\bar{x}_3 \lor \bar{x}_4)$</td>
<td>$(x_2)$</td>
<td>$(x_3)$</td>
<td>$(x_4)$</td>
</tr>
</tbody>
</table>

- **MUS predicate test:** $\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}$, $P(\mathcal{W}) \triangleq \neg \text{SAT}(\mathcal{W})$

<table>
<thead>
<tr>
<th>$c_i$</th>
<th>$\mathcal{M}$</th>
<th>$\mathcal{M} \setminus {c_i}$</th>
<th>$P(\mathcal{W})$</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>$c_1..c_7$</td>
<td>$c_2..c_7$</td>
<td>$1$</td>
<td>Drop $c_1$</td>
</tr>
</tbody>
</table>
Deletion – MUS example

\[
\begin{array}{ccccccc}
  c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \\
  (\bar{x}_1 \lor \bar{x}_2) & (x_1) & (x_5 \lor x_6) & (\bar{x}_3 \lor \bar{x}_4) & (x_2) & (x_3) & (x_4) \\
\end{array}
\]

- **MUS** predicate test: \( W \triangleq M \setminus \{c_i\}, \quad P(W) \triangleq \neg SAT(W) \)

<table>
<thead>
<tr>
<th>( c_i )</th>
<th>( M )</th>
<th>( M \setminus {c_i} )</th>
<th>( P(W) )</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_1 )</td>
<td>( c_1..c_7 )</td>
<td>( c_2..c_7 )</td>
<td>( 1 )</td>
<td>Drop ( c_1 )</td>
</tr>
<tr>
<td>( c_2 )</td>
<td>( c_2..c_7 )</td>
<td>( c_3..c_7 )</td>
<td>( 1 )</td>
<td>Drop ( c_2 )</td>
</tr>
</tbody>
</table>
Deletion – MUS example

\[
\begin{array}{cccccccc}
  c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \\
  (\bar{x}_1 \lor \bar{x}_2) & (x_1) & (x_5 \lor x_6) & (\bar{x}_3 \lor \bar{x}_4) & (x_2) & (x_3) & (x_4)
\end{array}
\]

- **MUS** predicate test: \( \mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}, \ P(\mathcal{W}) \triangleq \neg \text{SAT}(\mathcal{W}) \)

<table>
<thead>
<tr>
<th>( c_i )</th>
<th>( \mathcal{M} )</th>
<th>( \mathcal{M} \setminus {c_i} )</th>
<th>( P(\mathcal{W}) )</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_1 )</td>
<td>( c_1..c_7 )</td>
<td>( c_2..c_7 )</td>
<td>( 1 )</td>
<td>Drop ( c_1 )</td>
</tr>
<tr>
<td>( c_2 )</td>
<td>( c_2..c_7 )</td>
<td>( c_3..c_7 )</td>
<td>( 1 )</td>
<td>Drop ( c_2 )</td>
</tr>
<tr>
<td>( c_3 )</td>
<td>( c_3..c_7 )</td>
<td>( c_4..c_7 )</td>
<td>( 1 )</td>
<td>Drop ( c_3 )</td>
</tr>
</tbody>
</table>
Deletion – MUS example

\[
\begin{array}{cccccccc}
  c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \\
  (\overline{x}_1 \lor \overline{x}_2) & (x_1) & (x_5 \lor x_6) & (\overline{x}_3 \lor \overline{x}_4) & (x_2) & (x_3) & (x_4) \\
\end{array}
\]

- **MUS** predicate test: \[ W \triangleq \mathcal{M} \setminus \{c_i\}, \quad P(W) \triangleq \neg \text{SAT}(W) \]

<table>
<thead>
<tr>
<th>(c_i)</th>
<th>(\mathcal{M})</th>
<th>(\mathcal{M} \setminus {c_i})</th>
<th>(P(W))</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c_1)</td>
<td>(c_1..c_7)</td>
<td>(c_2..c_7)</td>
<td>1</td>
<td>Drop (c_1)</td>
</tr>
<tr>
<td>(c_2)</td>
<td>(c_2..c_7)</td>
<td>(c_3..c_7)</td>
<td>1</td>
<td>Drop (c_2)</td>
</tr>
<tr>
<td>(c_3)</td>
<td>(c_3..c_7)</td>
<td>(c_4..c_7)</td>
<td>1</td>
<td>Drop (c_3)</td>
</tr>
<tr>
<td>(c_4)</td>
<td>(c_4..c_7)</td>
<td>(c_5..c_7)</td>
<td>0</td>
<td>Keep (c_4)</td>
</tr>
</tbody>
</table>
Deletion – MUS example

$\begin{array}{ccccccc}
  c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \\
  (\bar{x}_1 \lor \bar{x}_2) & (x_1) & (x_5 \lor x_6) & (\bar{x}_3 \lor \bar{x}_4) & (x_2) & (x_3) & (x_4) \\
\end{array}$

- **MUS predicate test:** $\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}$, $P(\mathcal{W}) \triangleq \neg \text{SAT}(\mathcal{W})$

<table>
<thead>
<tr>
<th>$c_i$</th>
<th>$\mathcal{M}$</th>
<th>$\mathcal{M} \setminus {c_i}$</th>
<th>$P(\mathcal{W})$</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>$c_1..c_7$</td>
<td>$c_2..c_7$</td>
<td>1</td>
<td>Drop $c_1$</td>
</tr>
<tr>
<td>$c_2$</td>
<td>$c_2..c_7$</td>
<td>$c_3..c_7$</td>
<td>1</td>
<td>Drop $c_2$</td>
</tr>
<tr>
<td>$c_3$</td>
<td>$c_3..c_7$</td>
<td>$c_4..c_7$</td>
<td>1</td>
<td>Drop $c_3$</td>
</tr>
<tr>
<td>$c_4$</td>
<td>$c_4..c_7$</td>
<td>$c_5..c_7$</td>
<td>0</td>
<td>Keep $c_4$</td>
</tr>
<tr>
<td>$c_5$</td>
<td>$c_5..c_7$</td>
<td>$c_4, c_6, c_7$</td>
<td>1</td>
<td>Drop $c_5$</td>
</tr>
</tbody>
</table>
Deletion – MUS example

\[
\begin{align*}
\begin{array}{cccccccc}
\quad & c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \\
\hline
(\bar{x}_1 \lor \bar{x}_2) & (x_1) & (x_5 \lor x_6) & (\bar{x}_3 \lor \bar{x}_4) & (x_2) & (x_3) & (x_4) \\
\end{array}
\end{align*}
\]

- **MUS** predicate test: \( W \triangleq M \setminus \{c_i\}, \quad P(W) \triangleq \neg \text{SAT}(W) \)

<table>
<thead>
<tr>
<th>( c_i )</th>
<th>( M )</th>
<th>( M \setminus {c_i} )</th>
<th>( P(W) )</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_1 )</td>
<td>( c_1..c_7 )</td>
<td>( c_2..c_7 )</td>
<td>1</td>
<td>Drop ( c_1 )</td>
</tr>
<tr>
<td>( c_2 )</td>
<td>( c_2..c_7 )</td>
<td>( c_3..c_7 )</td>
<td>1</td>
<td>Drop ( c_2 )</td>
</tr>
<tr>
<td>( c_3 )</td>
<td>( c_3..c_7 )</td>
<td>( c_4..c_7 )</td>
<td>1</td>
<td>Drop ( c_3 )</td>
</tr>
<tr>
<td>( c_4 )</td>
<td>( c_4..c_7 )</td>
<td>( c_5..c_7 )</td>
<td>0</td>
<td>Keep ( c_4 )</td>
</tr>
<tr>
<td>( c_5 )</td>
<td>( c_5..c_7 )</td>
<td>( c_4, c_6, c_7 )</td>
<td>1</td>
<td>Drop ( c_5 )</td>
</tr>
<tr>
<td>( c_6 )</td>
<td>( c_4, c_6, c_7 )</td>
<td>( c_4, c_7 )</td>
<td>0</td>
<td>Keep ( c_6 )</td>
</tr>
</tbody>
</table>
Deletion – MUS example

<table>
<thead>
<tr>
<th>$c_i$</th>
<th>$\mathcal{M}$</th>
<th>$\mathcal{M} \setminus {c_i}$</th>
<th>$P(\mathcal{W})$</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>$c_1..c_7$</td>
<td>$c_2..c_7$</td>
<td>1</td>
<td>Drop $c_1$</td>
</tr>
<tr>
<td>$c_2$</td>
<td>$c_2..c_7$</td>
<td>$c_3..c_7$</td>
<td>1</td>
<td>Drop $c_2$</td>
</tr>
<tr>
<td>$c_3$</td>
<td>$c_3..c_7$</td>
<td>$c_4..c_7$</td>
<td>1</td>
<td>Drop $c_3$</td>
</tr>
<tr>
<td>$c_4$</td>
<td>$c_4..c_7$</td>
<td>$c_5..c_7$</td>
<td>0</td>
<td>Keep $c_4$</td>
</tr>
<tr>
<td>$c_5$</td>
<td>$c_5..c_7$</td>
<td>$c_4, c_6, c_7$</td>
<td>1</td>
<td>Drop $c_5$</td>
</tr>
<tr>
<td>$c_6$</td>
<td>$c_4, c_6, c_7$</td>
<td>$c_4, c_7$</td>
<td>0</td>
<td>Keep $c_6$</td>
</tr>
<tr>
<td>$c_7$</td>
<td>$c_4, c_6, c_7$</td>
<td>$c_4, c_6$</td>
<td>0</td>
<td>Keep $c_7$</td>
</tr>
</tbody>
</table>

**MUS predicate test:**

$\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}$, $P(\mathcal{W}) \triangleq \lnot \text{SAT}(\mathcal{W})$
Deletion – MUS example

\[
\begin{array}{cccccccc}
  c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \\
  (\bar{x}_1 \lor \bar{x}_2) & (x_1) & (x_5 \lor x_6) & (\bar{x}_3 \lor \bar{x}_4) & (x_2) & (x_3) & (x_4)
\end{array}
\]

- **MUS predicate test:** \( \mathcal{W} \triangleq M \setminus \{c_i\}, P(\mathcal{W}) \triangleq \neg \text{SAT}(\mathcal{W}) \)

<table>
<thead>
<tr>
<th>(c_i)</th>
<th>(\mathcal{M})</th>
<th>(\mathcal{M} \setminus {c_i})</th>
<th>(P(\mathcal{W}))</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c_1)</td>
<td>(c_1..c_7)</td>
<td>(c_2..c_7)</td>
<td>1</td>
<td>Drop (c_1)</td>
</tr>
<tr>
<td>(c_2)</td>
<td>(c_2..c_7)</td>
<td>(c_3..c_7)</td>
<td>1</td>
<td>Drop (c_2)</td>
</tr>
<tr>
<td>(c_3)</td>
<td>(c_3..c_7)</td>
<td>(c_4..c_7)</td>
<td>1</td>
<td>Drop (c_3)</td>
</tr>
<tr>
<td>(c_4)</td>
<td>(c_4..c_7)</td>
<td>(c_5..c_7)</td>
<td>0</td>
<td>Keep (c_4)</td>
</tr>
<tr>
<td>(c_5)</td>
<td>(c_5..c_7)</td>
<td>(c_4, c_6, c_7)</td>
<td>1</td>
<td>Drop (c_5)</td>
</tr>
<tr>
<td>(c_6)</td>
<td>(c_4, c_6, c_7)</td>
<td>(c_4, c_7)</td>
<td>0</td>
<td>Keep (c_6)</td>
</tr>
<tr>
<td>(c_7)</td>
<td>(c_4, c_6, c_7)</td>
<td>(c_4, c_6)</td>
<td>0</td>
<td>Keep (c_7)</td>
</tr>
</tbody>
</table>

- **MUS:** \(\{c_4, c_6, c_7\}\)
Deletion – MCS example

\[(\bar{x}_1 \lor \bar{x}_2) (x_1) (x_5 \lor x_6) (\bar{x}_3 \lor \bar{x}_4) (x_2) (x_3) (x_4)\]

- **MCS predicate test:**
  \[W \triangleq M \setminus \{c_i\}, \quad P(W) \triangleq \text{SAT}(F \setminus W)\]

<table>
<thead>
<tr>
<th>(c_i)</th>
<th>(M)</th>
<th>(M \setminus {c_i})</th>
<th>(F \setminus (M \setminus {c_i}))</th>
<th>(P(W))</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Deletion – MCS example

\[
\begin{align*}
\begin{array}{cccccccc}
  c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \\
(\bar{x}_1 \lor \bar{x}_2) & (x_1) & (x_5 \lor x_6) & (\bar{x}_3 \lor \bar{x}_4) & (x_2) & (x_3) & (x_4)
\end{array}
\end{align*}
\]

- **MCS predicate test:** \[\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}, \quad P(\mathcal{W}) \triangleq \text{SAT}(\mathcal{F} \setminus \mathcal{W})\]

<table>
<thead>
<tr>
<th>(c_i)</th>
<th>(\mathcal{M})</th>
<th>(\mathcal{M} \setminus {c_i})</th>
<th>(\mathcal{F} \setminus (\mathcal{M} \setminus {c_i}))</th>
<th>(P(\mathcal{W}))</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c_1)</td>
<td>(c_1..c_7)</td>
<td>(c_2..c_7)</td>
<td>(c_1)</td>
<td>(1)</td>
<td>Drop (c_1)</td>
</tr>
</tbody>
</table>
Deletion – MCS example

\[
\begin{array}{cccccccc}
\text{c}_1 & \text{c}_2 & \text{c}_3 & \text{c}_4 & \text{c}_5 & \text{c}_6 & \text{c}_7 \\
(\bar{x}_1 \lor \bar{x}_2) & (x_1) & (x_5 \lor x_6) & (\bar{x}_3 \lor \bar{x}_4) & (x_2) & (x_3) & (x_4) \\
\end{array}
\]

- **MCS predicate test:** \( W \triangleq M \setminus \{c_i\}, P(W) \triangleq \text{SAT}(F \setminus W) \)

<table>
<thead>
<tr>
<th>( c_i )</th>
<th>( M )</th>
<th>( M \setminus {c_i} )</th>
<th>( F \setminus (M \setminus {c_i}) )</th>
<th>( P(W) )</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_1 )</td>
<td>( c_1..c_7 )</td>
<td>( c_2..c_7 )</td>
<td>( c_1 )</td>
<td>1</td>
<td>Drop ( c_1 )</td>
</tr>
<tr>
<td>( c_2 )</td>
<td>( c_2..c_7 )</td>
<td>( c_3..c_7 )</td>
<td>( c_1, c_2 )</td>
<td>1</td>
<td>Drop ( c_2 )</td>
</tr>
</tbody>
</table>
### Deletion – MCS example

<table>
<thead>
<tr>
<th></th>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_4$</th>
<th>$c_5$</th>
<th>$c_6$</th>
<th>$c_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$(\bar{x}_1 \lor x_2)$</td>
<td>$x_1$</td>
<td>$(x_5 \lor x_6)$</td>
<td>$(\bar{x}_3 \lor \bar{x}_4)$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$x_4$</td>
</tr>
</tbody>
</table>

#### MCS predicate test:

$$\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}, \quad P(\mathcal{W}) \triangleq \text{SAT}(\mathcal{F} \setminus \mathcal{W})$$

<table>
<thead>
<tr>
<th>$c_i$</th>
<th>$\mathcal{M}$</th>
<th>$\mathcal{M} \setminus {c_i}$</th>
<th>$\mathcal{F} \setminus (\mathcal{M} \setminus {c_i})$</th>
<th>$P(\mathcal{W})$</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>$c_1..c_7$</td>
<td>$c_2..c_7$</td>
<td>$c_1$</td>
<td>1</td>
<td>Drop $c_1$</td>
</tr>
<tr>
<td>$c_2$</td>
<td>$c_2..c_7$</td>
<td>$c_3..c_7$</td>
<td>$c_1, c_2$</td>
<td>1</td>
<td>Drop $c_2$</td>
</tr>
<tr>
<td>$c_3$</td>
<td>$c_3..c_7$</td>
<td>$c_4..c_7$</td>
<td>$c_1..c_3$</td>
<td>1</td>
<td>Drop $c_3$</td>
</tr>
</tbody>
</table>

Compare with std MCS grow procedure!
Deletion – MCS example

\[
\begin{array}{cccccccc}
  c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \\
  (\bar{x}_1 \lor \bar{x}_2) & (x_1) & (x_5 \lor x_6) & (\bar{x}_3 \lor \bar{x}_4) & (x_2) & (x_3) & (x_4) \\
\end{array}
\]

- **MCS predicate test:**

  \[ \mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}, \ P(\mathcal{W}) \triangleq \text{SAT}(\mathcal{F} \setminus \mathcal{W}) \]

<table>
<thead>
<tr>
<th>(c_i)</th>
<th>(\mathcal{M})</th>
<th>(\mathcal{M} \setminus {c_i})</th>
<th>(\mathcal{F} \setminus (\mathcal{M} \setminus {c_i}))</th>
<th>(P(\mathcal{W}))</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c_1)</td>
<td>(c_1..c_7)</td>
<td>(c_2..c_7)</td>
<td>(c_1)</td>
<td>1</td>
<td>Drop (c_1)</td>
</tr>
<tr>
<td>(c_2)</td>
<td>(c_2..c_7)</td>
<td>(c_3..c_7)</td>
<td>(c_1, c_2)</td>
<td>1</td>
<td>Drop (c_2)</td>
</tr>
<tr>
<td>(c_3)</td>
<td>(c_3..c_7)</td>
<td>(c_4..c_7)</td>
<td>(c_1..c_3)</td>
<td>1</td>
<td>Drop (c_3)</td>
</tr>
<tr>
<td>(c_4)</td>
<td>(c_4..c_7)</td>
<td>(c_5..c_7)</td>
<td>(c_1..c_4)</td>
<td>1</td>
<td>Drop (c_4)</td>
</tr>
</tbody>
</table>

Compare with std MCS grow procedure!
### Deletion – MCS example

\[
\begin{array}{cccccccc}
  c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \\
  (\bar{x}_1 \lor \bar{x}_2) & (x_1) & (x_5 \lor x_6) & (\bar{x}_3 \lor \bar{x}_4) & (x_2) & (x_3) & (x_4) \\
\end{array}
\]

- **MCS predicate test:** \( W \triangleq M \setminus \{c_i\}, \quad P(W) \triangleq \text{SAT}(F \setminus W) \)

<table>
<thead>
<tr>
<th>( c_i )</th>
<th>( M )</th>
<th>( M \setminus {c_i} )</th>
<th>( F \setminus (M \setminus {c_i}) )</th>
<th>( P(W) )</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_1 )</td>
<td>( c_1\ldots c_7 )</td>
<td>( c_2\ldots c_7 )</td>
<td>( c_1 )</td>
<td>( 1 )</td>
<td>Drop ( c_1 )</td>
</tr>
<tr>
<td>( c_2 )</td>
<td>( c_2\ldots c_7 )</td>
<td>( c_3\ldots c_7 )</td>
<td>( c_1, c_2 )</td>
<td>( 1 )</td>
<td>Drop ( c_2 )</td>
</tr>
<tr>
<td>( c_3 )</td>
<td>( c_3\ldots c_7 )</td>
<td>( c_4\ldots c_7 )</td>
<td>( c_1\ldots c_3 )</td>
<td>( 1 )</td>
<td>Drop ( c_3 )</td>
</tr>
<tr>
<td>( c_4 )</td>
<td>( c_4\ldots c_7 )</td>
<td>( c_5\ldots c_7 )</td>
<td>( c_1\ldots c_4 )</td>
<td>( 1 )</td>
<td>Drop ( c_4 )</td>
</tr>
<tr>
<td>( c_5 )</td>
<td>( c_5\ldots c_7 )</td>
<td>( c_6, c_7 )</td>
<td>( c_1\ldots c_5 )</td>
<td>( 0 )</td>
<td>Keep ( c_5 )</td>
</tr>
</tbody>
</table>

Compare with std MCS grow procedure!
Deletion – MCS example

<table>
<thead>
<tr>
<th>$c_i$</th>
<th>$M$</th>
<th>$M\setminus{c_i}$</th>
<th>$F \setminus (M \setminus {c_i})$</th>
<th>$P(W)$</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>$c_1..c_7$</td>
<td>$c_2..c_7$</td>
<td>$c_1$</td>
<td>1</td>
<td>Drop $c_1$</td>
</tr>
<tr>
<td>$c_2$</td>
<td>$c_2..c_7$</td>
<td>$c_3..c_7$</td>
<td>$c_1, c_2$</td>
<td>1</td>
<td>Drop $c_2$</td>
</tr>
<tr>
<td>$c_3$</td>
<td>$c_3..c_7$</td>
<td>$c_4..c_7$</td>
<td>$c_1..c_3$</td>
<td>1</td>
<td>Drop $c_3$</td>
</tr>
<tr>
<td>$c_4$</td>
<td>$c_4..c_7$</td>
<td>$c_5..c_7$</td>
<td>$c_1..c_4$</td>
<td>1</td>
<td>Drop $c_4$</td>
</tr>
<tr>
<td>$c_5$</td>
<td>$c_5..c_7$</td>
<td>$c_6, c_7$</td>
<td>$c_1..c_5$</td>
<td>0</td>
<td>Keep $c_5$</td>
</tr>
<tr>
<td>$c_6$</td>
<td>$c_5, c_6, c_7$</td>
<td>$c_5, c_7$</td>
<td>$c_1..c_4, c_6$</td>
<td>1</td>
<td>Drop $c_6$</td>
</tr>
</tbody>
</table>

MCS predicate test: \( \mathcal{W} \triangleq M \setminus \{c_i\}, \ P(\mathcal{W}) \triangleq \text{SAT}(F \setminus \mathcal{W}) \)
**Deletion – MCS example**

\[
\begin{array}{ccccccc}
  c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \\
  (\bar{x}_1 \lor \bar{x}_2) & x_1 & (x_5 \lor x_6) & (\bar{x}_3 \lor \bar{x}_4) & x_2 & x_3 & x_4 \\
\end{array}
\]

- **MCS predicate test:**

\[\mathcal{W} \triangleq \mathcal{M} \setminus \{c_i\}, \quad P(\mathcal{W}) \triangleq \text{SAT}(\mathcal{F} \setminus \mathcal{W})\]

<table>
<thead>
<tr>
<th>(c_i)</th>
<th>(\mathcal{M})</th>
<th>(\mathcal{M} \setminus {c_i})</th>
<th>(\mathcal{F} \setminus (\mathcal{M} \setminus {c_i}))</th>
<th>(P(\mathcal{W}))</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c_1)</td>
<td>(c_1..c_7)</td>
<td>(c_2..c_7)</td>
<td>(c_1)</td>
<td>(1)</td>
<td>Drop (c_1)</td>
</tr>
<tr>
<td>(c_2)</td>
<td>(c_2..c_7)</td>
<td>(c_3..c_7)</td>
<td>(c_1, c_2)</td>
<td>(1)</td>
<td>Drop (c_2)</td>
</tr>
<tr>
<td>(c_3)</td>
<td>(c_3..c_7)</td>
<td>(c_4..c_7)</td>
<td>(c_1..c_3)</td>
<td>(1)</td>
<td>Drop (c_3)</td>
</tr>
<tr>
<td>(c_4)</td>
<td>(c_4..c_7)</td>
<td>(c_5..c_7)</td>
<td>(c_1..c_4)</td>
<td>(1)</td>
<td>Drop (c_4)</td>
</tr>
<tr>
<td>(c_5)</td>
<td>(c_5..c_7)</td>
<td>(c_6, c_7)</td>
<td>(c_1..c_5)</td>
<td>(0)</td>
<td>Keep (c_5)</td>
</tr>
<tr>
<td>(c_6)</td>
<td>(c_5, c_6, c_7)</td>
<td>(c_5, c_7)</td>
<td>(c_1..c_4, c_6)</td>
<td>(1)</td>
<td>Drop (c_6)</td>
</tr>
<tr>
<td>(c_7)</td>
<td>(c_5, c_7)</td>
<td>(c_5)</td>
<td>(c_1..c_4, c_6, c_7)</td>
<td>(0)</td>
<td>Keep (c_7)</td>
</tr>
</tbody>
</table>

Compare with std MCS grow procedure!
Deletion – MCS example

\[
\begin{array}{cccccc}
\text{c}_1 & \text{c}_2 & \text{c}_3 & \text{c}_4 & \text{c}_5 & \text{c}_6 & \text{c}_7 \\
(\bar{x}_1 \lor \bar{x}_2) & (x_1) & (x_5 \lor x_6) & (\bar{x}_3 \lor \bar{x}_4) & (x_2) & (x_3) & (x_4) \\
\end{array}
\]

- **MCS predicate test:** \[W \triangleq \mathcal{M} \setminus \{c_i\}, \quad P(W) \triangleq \text{SAT}(\mathcal{F} \setminus W)\]

<table>
<thead>
<tr>
<th>(c_i)</th>
<th>(\mathcal{M})</th>
<th>(\mathcal{M} \setminus {c_i})</th>
<th>(\mathcal{F} \setminus (\mathcal{M} \setminus {c_i}))</th>
<th>(P(W))</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c_1)</td>
<td>(c_1..c_7)</td>
<td>(c_2..c_7)</td>
<td>(c_1)</td>
<td>1</td>
<td>Drop (c_1)</td>
</tr>
<tr>
<td>(c_2)</td>
<td>(c_2..c_7)</td>
<td>(c_3..c_7)</td>
<td>(c_1, c_2)</td>
<td>1</td>
<td>Drop (c_2)</td>
</tr>
<tr>
<td>(c_3)</td>
<td>(c_3..c_7)</td>
<td>(c_4..c_7)</td>
<td>(c_1..c_3)</td>
<td>1</td>
<td>Drop (c_3)</td>
</tr>
<tr>
<td>(c_4)</td>
<td>(c_4..c_7)</td>
<td>(c_5..c_7)</td>
<td>(c_1..c_4)</td>
<td>1</td>
<td>Drop (c_4)</td>
</tr>
<tr>
<td>(c_5)</td>
<td>(c_5..c_7)</td>
<td>(c_6, c_7)</td>
<td>(c_1..c_5)</td>
<td>0</td>
<td>Keep (c_5)</td>
</tr>
<tr>
<td>(c_6)</td>
<td>(c_5, c_6, c_7)</td>
<td>(c_5, c_7)</td>
<td>(c_1..c_4, c_6)</td>
<td>1</td>
<td>Drop (c_6)</td>
</tr>
<tr>
<td>(c_7)</td>
<td>(c_5, c_7)</td>
<td>(c_5)</td>
<td>(c_1..c_4, c_6, c_7)</td>
<td>0</td>
<td>Keep (c_7)</td>
</tr>
</tbody>
</table>

- **MCS:** \(\{c_5, c_7\}\)
Deletion – MCS example

\[
\begin{align*}
\bar{x}_1 \vee \bar{x}_2 & \quad (x_1) \quad (x_5 \vee x_6) \quad (\bar{x}_3 \vee \bar{x}_4) \quad (x_2) \quad (x_3) \quad (x_4)
\end{align*}
\]

- **MCS** predicate test: \( W \triangleq \mathcal{M} \setminus \{c_i\}, \ P(W) \triangleq \text{SAT}(\mathcal{F} \setminus W) \)

<table>
<thead>
<tr>
<th>( c_i )</th>
<th>( \mathcal{M} )</th>
<th>( \mathcal{M} \setminus {c_i} )</th>
<th>( \mathcal{F} \setminus (\mathcal{M} \setminus {c_i}) )</th>
<th>( P(W) )</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_1 )</td>
<td>( c_1..c_7 )</td>
<td>( c_2..c_7 )</td>
<td>( c_1 )</td>
<td>1</td>
<td>Drop ( c_1 )</td>
</tr>
<tr>
<td>( c_2 )</td>
<td>( c_2..c_7 )</td>
<td>( c_3..c_7 )</td>
<td>( c_1, c_2 )</td>
<td>1</td>
<td>Drop ( c_2 )</td>
</tr>
<tr>
<td>( c_3 )</td>
<td>( c_3..c_7 )</td>
<td>( c_4..c_7 )</td>
<td>( c_1..c_3 )</td>
<td>1</td>
<td>Drop ( c_3 )</td>
</tr>
<tr>
<td>( c_4 )</td>
<td>( c_4..c_7 )</td>
<td>( c_5..c_7 )</td>
<td>( c_1..c_4 )</td>
<td>1</td>
<td>Drop ( c_4 )</td>
</tr>
<tr>
<td>( c_5 )</td>
<td>( c_5..c_7 )</td>
<td>( c_6, c_7 )</td>
<td>( c_1..c_5 )</td>
<td>0</td>
<td>Keep ( c_5 )</td>
</tr>
<tr>
<td>( c_6 )</td>
<td>( c_5, c_6, c_7 )</td>
<td>( c_5, c_7 )</td>
<td>( c_1..c_4, c_6 )</td>
<td>1</td>
<td>Drop ( c_6 )</td>
</tr>
<tr>
<td>( c_7 )</td>
<td>( c_5, c_7 )</td>
<td>( c_5 )</td>
<td>( c_1..c_4, c_6, c_7 )</td>
<td>0</td>
<td>Keep ( c_7 )</td>
</tr>
</tbody>
</table>

- **MCS**: \( \{c_5, c_7\} \)

Compare with std MCS grow procedure!
- **Deletion**: Check (& remove?) one element at a time
From deletion to progression

- **Deletion**: Check (& remove?) one element at a time
  - Pick set of elements given by **arithmetic** progression
- **Progression**: Check (& remove) exponentially growing set of elements
  - Pick set of elements given by **geometric** progression
Progression algorithm

\[
\begin{align*}
i & \leftarrow 0 \\
n & \leftarrow \min(2^i, |T|) \\
T & = \emptyset \\
G & \leftarrow M \cup T \setminus T_{1..\nu} \\
j & \leftarrow \text{BinS}(M, T, \nu) \\
T & \leftarrow T \setminus T_{1..j} \\
M & \leftarrow M \cup T_{j..\nu} \\
i & \leftarrow 0 \\
P(G) & ?
\end{align*}
\]
Progression algorithm

\[ i \leftarrow 0 \]

\[ \nu \leftarrow \min(2^i, |T|) \]

\[ T = \emptyset? \]

\[ G \leftarrow M \cup T \setminus T_{1..\nu} \]

\[ j \leftarrow \text{BinS}(M, T, \nu) \]

\[ T \leftarrow T \setminus T_{1..j} \]

\[ M \leftarrow M \cup T_{j..j} \]

\[ i \leftarrow 0 \]

\[ P(G)? \]

\[ \mathcal{O}(k \times \log(1 + \frac{m}{k})) \]

predicate tests
Progression – MUS example

<table>
<thead>
<tr>
<th>$c_1$</th>
<th>$c_2$</th>
<th>$c_3$</th>
<th>$c_4$</th>
<th>$c_5$</th>
<th>$c_6$</th>
<th>$c_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\bar{x}_1 \lor \bar{x}_2)$</td>
<td>$(x_1)$</td>
<td>$(x_5 \lor x_6)$</td>
<td>$(\bar{x}_3 \lor \bar{x}_4)$</td>
<td>$(x_2)$</td>
<td>$(x_3)$</td>
<td>$(x_4)$</td>
</tr>
</tbody>
</table>

- **MUS** predicate test: \( \mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu} \), \( P(\mathcal{W}) \triangleq \neg \text{SAT}(\mathcal{W}) \)

\[
i \nu = \min(2^i, |\mathcal{T}|) \quad \mathcal{M} \quad \mathcal{T} \quad \mathcal{T} \setminus \mathcal{T}_{1..\nu} \quad P(\mathcal{W}) \quad \text{BinSearch}
\]
Progression – MUS example

\[
\begin{array}{cccccccc}
c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \\
(\bar{x}_1 \lor \bar{x}_2) & (x_1) & (x_5 \lor x_6) & (\bar{x}_3 \lor \bar{x}_4) & (x_2) & (x_3) & (x_4)
\end{array}
\]

- **MUS predicate test:** \( W \triangleq M \cup T \setminus T_{1..\nu} \), \( P(W) \triangleq \neg \text{SAT}(W) \)

| \( i \) | \( \nu = \min(2^i, |T|) \) | \( M \) | \( T \) | \( T \setminus T_{1..\nu} \) | \( P(W) \) | BinSearch |
|---|---|---|---|---|---|---|
| 0 | 1 | \( \emptyset \) | \( c_1..c_7 \) | \( c_2..c_7 \) | 1 | – |
Progression – MUS example

\[
\begin{array}{cccccccc}
  c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \\
  (\bar{x}_1 \lor \bar{x}_2) & (x_1) & (x_5 \lor x_6) & (\bar{x}_3 \lor \bar{x}_4) & (x_2) & (x_3) & (x_4)
\end{array}
\]

- **MUS predicate test:** \( \mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu} \), \( P(\mathcal{W}) \triangleq \neg \text{SAT}(\mathcal{W}) \)

| \( i \) | \( \nu = \min(2^i, |\mathcal{T}|) \) | \( \mathcal{M} \) | \( \mathcal{T} \) | \( \mathcal{T} \setminus \mathcal{T}_{1..\nu} \) | \( P(\mathcal{W}) \) | BinSearch |
|---|---|---|---|---|---|---|
| 0 | 1 | \( \emptyset \) | \( c_1..c_7 \) | \( c_2..c_7 \) | 1 | – |
| 1 | 2 | \( \emptyset \) | \( c_2..c_7 \) | \( c_4..c_7 \) | 1 | – |
### Progression – MUS example

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>$c_2$</td>
<td>$c_3$</td>
<td>$c_4$</td>
<td>$c_5$</td>
<td>$c_6$</td>
<td>$c_7$</td>
<td></td>
</tr>
<tr>
<td>$(\bar{x}_1 \lor \bar{x}_2)$</td>
<td>$(x_1)$</td>
<td>$(x_5 \lor x_6)$</td>
<td>$(\bar{x}_3 \lor \bar{x}_4)$</td>
<td>$(x_2)$</td>
<td>$(x_3)$</td>
<td>$(x_4)$</td>
<td></td>
</tr>
</tbody>
</table>

- **MUS predicate test:** $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu}$, $P(\mathcal{W}) \triangleq \neg \text{SAT}(\mathcal{W})$

| $i$ | $\nu = \min(2^i, |T|)$ | $\mathcal{M}$ | $\mathcal{T}$ | $\mathcal{T} \setminus \mathcal{T}_{1..\nu}$ | $P(\mathcal{W})$ | BinSearch |
|---|---|---|---|---|---|---|
| 0 | 1 | $\emptyset$ | $c_1..c_7$ | $c_2..c_7$ | 1 | $-$ |
| 1 | 2 | $\emptyset$ | $c_2..c_7$ | $c_4..c_7$ | 1 | $-$ |
| 2 | 4 | $\emptyset$ | $c_4..c_7$ | $\emptyset$ | 0 | $c_4$ |
### Progression – MUS example

| i | $\nu = \min(2^i, |T|)$ | $M$ | $T$ | $T \setminus T_{1..\nu}$ | $P(\mathcal{W})$ | BinSearch |
|---|---|---|---|---|---|---|
| 0 | 1 | $\emptyset$ | $c_1..c_7$ | $c_2..c_7$ | 1 | – |
| 1 | 2 | $\emptyset$ | $c_2..c_7$ | $c_4..c_7$ | 1 | – |
| 2 | 4 | $\emptyset$ | $c_4..c_7$ | $\emptyset$ | 0 | $c_4$ |
| 0 | 1 | $c_4$ | $c_5..c_7$ | $c_6..c_7$ | 1 | – |

**MUS predicate test:** \[ \mathcal{W} \triangleq M \cup T \setminus T_{1..\nu}, \quad P(\mathcal{W}) \triangleq \neg \text{SAT}(\mathcal{W}) \]
Progression – MUS example

\[
\begin{array}{cccccccc}
  c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \\
  (\bar{x}_1 \lor \bar{x}_2) & (x_1) & (x_5 \lor x_6) & (\bar{x}_3 \lor \bar{x}_4) & (x_2) & (x_3) & (x_4)
\end{array}
\]

- **MUS** predicate test: \[ W \triangleq M \cup T \setminus T_{1..\nu} , \ P(W) \triangleq \neg \text{SAT}(W) \]

| $i$ | $\nu = \min(2^i, |T|)$ | $M$ | $T$ | $T \setminus T_{1..\nu}$ | $P(W)$ | BinSearch |
|-----|-------------------|-----|-----|---------------------|--------|-----------|
| 0   | 1                 | $\emptyset$ | $c_1..c_7$ | $c_2..c_7$ | 1      | –         |
| 1   | 2                 | $\emptyset$ | $c_2..c_7$ | $c_4..c_7$ | 1      | –         |
| 2   | 4                 | $\emptyset$ | $c_4..c_7$ | $\emptyset$ | 0      | $c_4$    |
| 0   | 1                 | $c_4$ | $c_5..c_7$ | $c_6..c_7$ | 1      | –         |
| 1   | 2                 | $c_4$ | $c_6..c_7$ | $\emptyset$ | 0      | $c_6$    |
Progression – MUS example

\[ (\bar{x}_1 \lor \bar{x}_2) \land (x_1) \land (x_5 \lor x_6) \land (\bar{x}_3 \lor \bar{x}_4) \land (x_2) \land (x_3) \land (x_4) \]

- **MUS predicate test:** \( W \triangleq M \cup T \setminus T_{1..\nu} \), \( P(W) \triangleq \neg \text{SAT}(W) \)

| \( i \) | \( \nu = \min(2^i, |T|) \) | \( M \) | \( T \) | \( T \setminus T_{1..\nu} \) | \( P(W) \) | BinSearch |
|------|-------------------|--------|--------|-----------------|-------|--------|
| 0    | 1                 | \( \emptyset \) | \( c_1..c_7 \) | \( c_2..c_7 \) | 1     | –      |
| 1    | 2                 | \( \emptyset \) | \( c_2..c_7 \) | \( c_4..c_7 \) | 1     | –      |
| 2    | 4                 | \( \emptyset \) | \( c_4..c_7 \) | \( \emptyset \) | 0     | \( c_4 \) |
| 0    | 1                 | \( c_4 \)  | \( c_5..c_7 \) | \( c_6..c_7 \) | 1     | –      |
| 1    | 2                 | \( c_4 \)  | \( c_6..c_7 \) | \( \emptyset \) | 0     | \( c_6 \) |
| 0    | 1                 | \( c_4, c_6 \) | \( c_7 \)  | \( \emptyset \) | 0     | \( c_7 \) |
Progression – MUS example

| $i$ | $\nu = \min(2^i, |T|)$ | $\mathcal{M}$ | $\mathcal{T}$ | $\mathcal{T} \setminus \mathcal{T}_{1..\nu}$ | $P(\mathcal{W})$ | BinSearch |
|-----|----------------------|--------------|-------------|-----------------|----------------|-----------|
| 0   | 1                    | $\emptyset$  | $c_1..c_7$  | $c_2..c_7$      | 1              | –         |
| 1   | 2                    | $\emptyset$  | $c_2..c_7$  | $c_4..c_7$      | 1              | –         |
| 2   | 4                    | $\emptyset$  | $c_4..c_7$  | $\emptyset$     | 0              | $c_4$     |
| 0   | 1                    | $c_4$        | $c_5..c_7$  | $c_6..c_7$      | 1              | –         |
| 1   | 2                    | $c_4$        | $c_6..c_7$  | $\emptyset$     | 0              | $c_6$     |
| 0   | 1                    | $c_4, c_6$   | $c_7$       | $\emptyset$     | 0              | $c_7$     |
| 0   | –                    | $c_4, c_6, c_7$ | $\emptyset$ | –               | –              | –         |

- **MUS predicate test:** \[ \mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu}, \quad P(\mathcal{W}) \triangleq \neg \text{SAT}(\mathcal{W}) \]
### Progression – MUS example

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1$</td>
<td>$c_2$</td>
<td>$c_3$</td>
<td>$c_4$</td>
<td>$c_5$</td>
<td>$c_6$</td>
<td>$c_7$</td>
</tr>
<tr>
<td>$(\bar{x}_1 \lor \bar{x}_2)$</td>
<td>$(x_1)$</td>
<td>$(x_5 \lor x_6)$</td>
<td>$(\bar{x}_3 \lor \bar{x}_4)$</td>
<td>$(x_2)$</td>
<td>$(x_3)$</td>
<td>$(x_4)$</td>
</tr>
</tbody>
</table>

- **MUS predicate test:** $\mathcal{W} \triangleq \mathcal{M} \cup \mathcal{T} \setminus \mathcal{T}_{1..\nu}$, $P(\mathcal{W}) \triangleq \neg \text{SAT}(\mathcal{W})$

| $i$ | $\nu = \min(2^i, |\mathcal{T}|)$ | $\mathcal{M}$ | $\mathcal{T}$ | $\mathcal{T} \setminus \mathcal{T}_{1..\nu}$ | $P(\mathcal{W})$ | BinSearch |
|-----|-------------------------------|---------------|---------------|-----------------|----------------|-----------|
| 0   | 1                             | $\emptyset$   | $c_1..c_7$    | $c_2..c_7$      | 1              | –         |
| 1   | 2                             | $\emptyset$   | $c_2..c_7$    | $c_4..c_7$      | 1              | –         |
| 2   | 4                             | $\emptyset$   | $c_4..c_7$    | $\emptyset$     | 0              | $c_4$     |
| 0   | 1                             | $c_4$         | $c_5..c_7$    | $c_6..c_7$      | 1              | –         |
| 1   | 2                             | $c_4$         | $c_6..c_7$    | $\emptyset$     | 0              | $c_6$     |
| 0   | 1                             | $c_4, c_6$    | $c_7$         | $\emptyset$     | 0              | $c_7$     |
| 0   | –                             | $c_4, c_6, c_7$| $\emptyset$  | –               | –              | –         |

- **MUS:** $\{c_4, c_6, c_7\}$
Progression – MUS example

| $i$ | $\nu = \min(2^i, |T|)$ | $M$  | $T$  | $T \setminus T_{1..\nu}$ | $P(\mathcal{W})$ | BinSearch |
|-----|-----------------|------|------|---------------------|-----------------|-----------|
| 0   | 1               | $\emptyset$ | $c_1..c_7$ | $c_2..c_7$ | 1              | $-$       |
| 1   | 2               | $\emptyset$ | $c_2..c_7$ | $c_4..c_7$ | 1              | $-$       |
| 2   | 4               | $\emptyset$ | $c_4..c_7$ | $\emptyset$ | 0              | $c_4$     |
| 0   | 1               | $c_4$  | $c_5..c_7$ | $c_6..c_7$ | 1              | $-$       |
| 1   | 2               | $c_4$  | $c_6..c_7$ | $\emptyset$ | 0              | $c_6$     |
| 0   | 1               | $c_4, c_6$ | $c_7$  | $\emptyset$ | 0              | $c_7$     |
| 0   | $-$             | $c_4, c_6, c_7$ | $\emptyset$ | $-$       | $-$             | $-$       |

- **MUS** predicate test: $\mathcal{W} \triangleq M \cup T \setminus T_{1..\nu}$, $P(\mathcal{W}) \triangleq \neg\text{SAT}(\mathcal{W})$

- **MUS:** $\{c_4, c_6, c_7\}$

BinSearch gets elements of $M$
Outline

Optimization Problems

Minimal Sets

Query Complexity

Conclusion
Disclaimer: Initial ideas; comments are welcome!
Oracles and query complexity

- NP oracles vs. witness oracles
  - Given instance:

<table>
<thead>
<tr>
<th></th>
<th>NP oracle</th>
<th>witness oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accepts((Y)) / Rejects((N))</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Returns poly-size (Y) witness</td>
<td>✗</td>
<td>✔</td>
</tr>
</tbody>
</table>
Oracles and query complexity

- NP oracles vs. witness oracles
  - Given instance:

<table>
<thead>
<tr>
<th></th>
<th>NP oracle</th>
<th>witness oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accepts((Y)) / Rejects((N))</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Returns poly-size (Y) witness</td>
<td>×</td>
<td>✓</td>
</tr>
</tbody>
</table>

- SAT solvers produce **witnesses** for \(Y\) outcomes
  - SAT solvers correspond to witness oracles, i.e. SAT oracles
Oracles and query complexity

- NP oracles vs. witness oracles
  - Given instance:

<table>
<thead>
<tr>
<th></th>
<th>NP oracle</th>
<th>witness oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accepts((Y)) / Rejects((N))</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Returns poly-size (Y) witness</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

- SAT solvers produce witnesses for \(Y\) outcomes
  - SAT solvers correspond to witness oracles, i.e. SAT oracles

- Some complexity classes for function problems:

<table>
<thead>
<tr>
<th>NP oracles</th>
<th>(\text{FP}^{\text{NP}[\log n]} \subseteq \text{FP}^{\text{NP}} \subseteq \text{FP}^{\text{NP}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>witness oracles</td>
<td>(\text{FP}^{\text{NP}[\text{wit}, \log n]})</td>
</tr>
</tbody>
</table>
Oracles and query complexity

- **NP oracles vs. witness oracles**
  - Given instance:

<table>
<thead>
<tr>
<th></th>
<th>NP oracle</th>
<th>witness oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accepts(Y) / Rejects(N)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Returns poly-size Y witness</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

- **SAT solvers produce witnesses for **Y** outcomes**
  - SAT solvers correspond to witness oracles, i.e. **SAT oracles**

- **Some complexity classes for function problems:**

<table>
<thead>
<tr>
<th>NP oracles</th>
<th>FP&lt;sup&gt;NP&lt;/sup&gt; [log n] ( \subseteq ) FP&lt;sup&gt;NP&lt;/sup&gt;</th>
<th>FP&lt;sup&gt;NP&lt;/sup&gt; ( \parallel ) ( \not\subseteq ) FP&lt;sup&gt;NP&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>witness oracles</td>
<td>FP&lt;sup&gt;NP&lt;/sup&gt; [wit, log n]</td>
<td></td>
</tr>
</tbody>
</table>
Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:

  - Backbones: literals common to all models of $F$ \cite{MZKST99,MSJL10,ZWSM11}
  - Assume reference model
  - Algorithm: use one query to check each literal

- MUS #1: compute MUS for formulas with exactly 1 MUS

- Why?

  - Best: $O(V - B)$ \cite{ZWSM11}
  - Best: $O(|F|)$?
Preliminary results

• # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  – Backbones: literals common to all models of $\mathcal{F}^{[MZKST99,MSJL10,ZWSM11]}
    ▶ Assume reference model
    ▶ Algorithm: use one query to check each literal

  • MUS
    #1: compute MUS for formulas with exactly 1 MUS

  • Why?
    Best: $O(V-B)$ calls [ZWSM11]
    Best $O(|F|)$?
Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - **Backbones:** literals common to all models of \( \mathcal{F} \) \cite{MZKST99,MSJL10,ZWSM11}
    - Assume reference model
    - **Algorithm:** use one query to check each literal
  - **MUS**
    - #1: compute MUS for formulas with exactly 1 MUS
      - Why?
        - Best: \( O(V - B) \) calls \cite{ZWSM11}
Preliminary results

• # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  – **Backbones**: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    ▶ Assume reference model
    ▶ **Algorithm**: use one query to check each literal
  – **MUS$\#_1$**: compute MUS for formulas with *exactly* 1 MUS
Preliminary results

• # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  – **Backbones**: literals common to all models of $\mathcal{F}$ \cite{MZKST99,MSJL10,ZWSM11}
    ▶ Assume reference model
    ▶ **Algorithm**: use one query to check each literal
  – **MUS\#1**: compute MUS for formulas with exactly 1 MUS

Best $O(|\mathcal{F}|)$?
Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - **Backbones**: literals common to all models of $\mathcal{F}$ \cite{MZKST99,MSJL10,ZWSM11}
    - Assume reference model
    - Algorithm: use one query to check each literal
  - **MUS\#1**: compute MUS for formulas with exactly 1 MUS

<table>
<thead>
<tr>
<th>Problem</th>
<th>NP Oracles</th>
<th>SAT Oracles</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - **Backbones**: literals common to all models of $\mathcal{F}$ \cite{MZKST99,MSJL10,ZWSM11}
    - Assume reference model
    - **Algorithm**: use one query to check each literal
  - **MUS\#1**: compute MUS for formulas with exactly 1 MUS

<table>
<thead>
<tr>
<th>Problem</th>
<th>NP Oracles</th>
<th>SAT Oracles</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS</td>
<td>$O(n)$</td>
<td></td>
</tr>
</tbody>
</table>
Preliminary results

• # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  
  − **Backbones**: literals common to all models of $\mathcal{F}$ \cite{MZKST99,MSJL10,ZWSM11} 
    
      ▶ Assume reference model
      ▶ **Algorithm**: use one query to check each literal
  
  − **MUS\#1**: compute MUS for formulas with **exactly** 1 MUS

<table>
<thead>
<tr>
<th>Problem</th>
<th>NP Oracles</th>
<th>SAT Oracles</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS</td>
<td>$O(n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>MUS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Preliminary results

• # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  – **Backbones**: literals common to all models of $\mathcal{F}$ [MZKST99, MSJL10, ZWSM11]
    ▶ Assume reference model
    ▶ **Algorithm**: use one query to check each literal
  – **MUS$_{\#1}$**: compute MUS for formulas with exactly 1 MUS

<table>
<thead>
<tr>
<th>Problem</th>
<th>NP Oracles</th>
<th>SAT Oracles</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS</td>
<td>$\mathcal{O}(n)$</td>
<td>$\mathcal{O}(\log n)$</td>
</tr>
<tr>
<td>MUS</td>
<td>$\mathcal{O}(n)$</td>
<td></td>
</tr>
</tbody>
</table>
Preliminary results

• # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - **Backbones**: literals common to all models of $\mathcal{F}$ [MZKST99, MSJL10, ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - **MUS$_1$**: compute MUS for formulas with exactly 1 MUS

<table>
<thead>
<tr>
<th>Problem</th>
<th>NP Oracles</th>
<th>SAT Oracles</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS</td>
<td>$O(n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>MUS</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
</tbody>
</table>
Preliminary results

- The number of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
- **Backbones**: literals common to all models of $\mathcal{F}$ [MZKST99, MSJL10, ZWSM11]
  - Assume reference model
  - Algorithm: use one query to check each literal
- **MUS#1**: compute MUS for formulas with exactly 1 MUS

<table>
<thead>
<tr>
<th>Problem</th>
<th>NP Oracles</th>
<th>SAT Oracles</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS</td>
<td>$O(n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>MUS</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
</tbody>
</table>

- Best $O(|\mathcal{F}|)$?
Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - **Backbones**: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - **MUS#1**: compute MUS for formulas with **exactly** 1 MUS

<table>
<thead>
<tr>
<th>Problem</th>
<th>NP Oracles</th>
<th>SAT Oracles</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS</td>
<td>$O(n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>MUS</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Backbones</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Preliminary results

- Number of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - **Backbones**: literals common to all models of \( \mathcal{F} \) [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - **Algorithm**: use one query to check each literal
  - **MUS\#1**: compute MUS for formulas with **exactly** 1 MUS

<table>
<thead>
<tr>
<th>Problem</th>
<th>NP Oracles</th>
<th>SAT Oracles</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS</td>
<td>( O(n) )</td>
<td>( O(\log n) )</td>
</tr>
<tr>
<td>MUS</td>
<td>( O(n) )</td>
<td>( O(n) )</td>
</tr>
<tr>
<td>Backbones</td>
<td>( O(n), \parallel )</td>
<td>( O(n) )</td>
</tr>
</tbody>
</table>
Preliminary results

• # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - **Backbones**: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - **MUS#1**: compute MUS for formulas with exactly 1 MUS

<table>
<thead>
<tr>
<th>Problem</th>
<th>NP Oracles</th>
<th>SAT Oracles</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS</td>
<td>$O(n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>MUS</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Backbones</td>
<td>$O(n)$, $\parallel$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>MUS#1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Preliminary results

• # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  – Backbones: literals common to all models of $\mathcal{F}$ \cite{MZKST99,MSJL10,ZWSM11}
    - Assume reference model
    - Algorithm: use one query to check each literal
  – MUS$_{#1}$: compute MUS for formulas with exactly 1 MUS

<table>
<thead>
<tr>
<th>Problem</th>
<th>NP Oracles</th>
<th>SAT Oracles</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS</td>
<td>$O(n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>MUS</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Backbones</td>
<td>$O(n)$, $\parallel$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>MUS$_{#1}$</td>
<td>$O(n)$, $\parallel$</td>
<td></td>
</tr>
</tbody>
</table>
Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - **Backbones**: literals common to all models of $\mathcal{F}$ [MZKST99,MSJL10,ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - **MUS\#1**: compute MUS for formulas with exactly 1 MUS

<table>
<thead>
<tr>
<th>Problem</th>
<th>NP Oracles</th>
<th>SAT Oracles</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS</td>
<td>$O(n)$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>MUS</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Backbones</td>
<td>$O(n)$, $|$</td>
<td>$O(\log n)$</td>
</tr>
<tr>
<td>MUS#1</td>
<td>$O(n)$, $|$</td>
<td>$O(\log n)$</td>
</tr>
</tbody>
</table>

- **Why?**
Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - **Backbones**: literals common to all models of $\mathcal{F}$ [MZKST99, MSJL10, ZWSM11]
    - Assume reference model
    - Algorithm: use one query to check each literal
  - **MUS*1**: compute MUS for formulas with exactly 1 MUS

<table>
<thead>
<tr>
<th>Problem</th>
<th>NP Oracles</th>
<th>SAT Oracles</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS</td>
<td>$FP^{NP}$</td>
<td>$FP^{NP}$ [wit, log n]</td>
</tr>
<tr>
<td>MUS</td>
<td>$FP^{NP}$</td>
<td>$FP^{NP}$</td>
</tr>
<tr>
<td>Backbones</td>
<td>$FP^{NP}$</td>
<td>$FP^{NP}$ [wit, log n]</td>
</tr>
<tr>
<td>MUS*1</td>
<td>$FP^{NP}$</td>
<td>$FP^{NP}$ [wit, log n]</td>
</tr>
</tbody>
</table>

- **Why?**
Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - **Backbones**: literals common to all models of \( \mathcal{F} \) [MZKST99, MSJL10, ZWSM11]
    - Assume reference model
    - **Algorithm**: use one query to check each literal
  - **MUS\#1**: compute MUS for formulas with exactly 1 MUS

<table>
<thead>
<tr>
<th>Problem</th>
<th>NP Oracles</th>
<th>SAT Oracles</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS</td>
<td>( \text{FP}^{\text{NP}} )</td>
<td>( \text{FP}^{\text{NP}}[\text{wit, log } n] )</td>
</tr>
<tr>
<td>MUS</td>
<td>( \text{FP}^{\text{NP}} )</td>
<td>( \text{FP}^{\text{NP}} )</td>
</tr>
<tr>
<td>Backbones</td>
<td>( \text{FP}^{\parallel} )</td>
<td>( \text{FP}^{\text{NP}}[\text{wit, log } n] )</td>
</tr>
<tr>
<td>MUS#1</td>
<td>( \text{FP}^{\parallel} )</td>
<td>( \text{FP}^{\text{NP}}[\text{wit, log } n] )</td>
</tr>
</tbody>
</table>

- **Why?**

\( \text{FP}^{\text{NP}}[\log n] \) if goal is number
Preliminary results

- # of queries to SAT/NP oracle for solving selected (possibly restricted) function problems:
  - **Backbones**: literals common to all models of $\mathcal{F}$ [MZKST99, MSJL10, ZWSM11]
    - **Assume reference model**
    - **Algorithm**: use one query to check each literal
  - **MUS\#1**: compute MUS for formulas with exactly 1 MUS

<table>
<thead>
<tr>
<th>Problem</th>
<th>NP Oracles</th>
<th>SAT Oracles</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCS</td>
<td>FP$^\text{NP}$</td>
<td>FP$^\text{NP}$[wit, log $n$]</td>
</tr>
<tr>
<td>MUS</td>
<td>FP$^\text{NP}$</td>
<td>FP$^\text{NP}$</td>
</tr>
<tr>
<td>Backbones</td>
<td>FP$^\text{NP}$</td>
<td>FP$^\text{NP}$[wit, log $n$]</td>
</tr>
<tr>
<td>MUS#1</td>
<td>FP$^\text{NP}$</td>
<td>FP$^\text{NP}$[wit, log $n$]</td>
</tr>
</tbody>
</table>

- **Why?**
  - “Easier” than computing SAT witness!
Outline

Optimization Problems

Minimal Sets

Query Complexity

Conclusion
Conclusions

- Significant progress in SAT-based (function) problem solving
  - MUSes, MCSes, MaxSAT, MinSAT, backbones, autarkies, minimal models, prime implicants & implicates
  - But also, MESes, MFSes, etc.

- Categorized function problems on Boolean formulas:
  - Optimization problems
  - Computation of minimal sets

- Introduced the MSMP problem
  - Framework for reasoning about (many) minimal sets problems

- Overviewed algorithms for optimization problems and for minimal set computation
  - E.g. refine UB, refine LB, binary search, core-guided, etc.
  - Insertion, Deletion, Dichotomic, QuickXplain, Progression

- Developed some preliminary query complexity results with witness oracles
  - MCSes, Backbones, MUS #1
Research directions

- New minimal set problems?
  - And new optimization problems?
- New algorithms?
- New pruning techniques?
- New implementation techniques?
  - How about preprocessing?
  - How about parallelization?
- Query complexity results?
  - Also, FPT reductions to SAT?
- How about enumeration problems?
  - MUSes, MCSes, MaxSAT, etc.
- ...

Research directions

- New minimal set problems?
  - And new optimization problems?
- New algorithms?
- New pruning techniques?
- New implementation techniques?
  - How about preprocessing?
  - How about parallelization?
- Query complexity results?
  - Also, FPT reductions to SAT?
- How about enumeration problems?
  - MUSes, MCSes, MaxSAT, etc.
- ...
Thank You

Thanks to the researchers at UCD, INESC-ID & UofS

A. Belov, F. Heras, A. Ignatiev,
M. Janota, I. Lynce, V. Manquinho,
A. Morgado, J. Planes, A. Previti,
and many others
References

ABGL12  C. Ansotegui, M. Bonet, J. Gabas, J. Levy: Improving SAT-Based Weighted MaxSAT Solvers. CP 2012: 86-101

ABGL13  C. Ansotegui, M. Bonet, J. Gabas and J. Levy: Improving WPM2 for (Weighted) Partial MaxSAT. CP 2013

ABL09a  C. Ansotegui, M. Bonet, J. Levy: Solving (Weighted) Partial MaxSAT through Satisfiability Testing. SAT 2009: 427-440


ABL10  C. Ansotegui, M. Bonet, J. Levy: A New Algorithm for Weighted Partial MaxSAT. AAAI 2010


<table>
<thead>
<tr>
<th>Code</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM07</td>
<td>A. Bradley, Z. Manna: Checking Safety by Inductive Generalization of Counterexamples to Induction. FMCAD 2007: 173-180</td>
</tr>
<tr>
<td>BMS11</td>
<td>A. Belov, J. Marques-Silva: Accelerating MUS extraction with recursive model rotation. FMCAD 2011: 37-40</td>
</tr>
<tr>
<td>BS05</td>
<td>J. Bailey, P. Stuckey: Discovery of Minimal Unsatisfiable Subsets of Constraints Using Hitting Set Dualization. PADL 2005: 174-186</td>
</tr>
<tr>
<td>DB11</td>
<td>J. Davies, F. Bacchus: Solving MAXSAT by Solving a Sequence of Simpler SAT Instances. CP 2011: 225-239</td>
</tr>
<tr>
<td>DB13a</td>
<td>J. Davies, F. Bacchus: Exploiting the Power of MIP Solvers in MaxSAT. SAT 2013: 166-181</td>
</tr>
<tr>
<td>DB13b</td>
<td>J. Davies and F. Bacchus: Postponing Optimization to Speed Up MaxSAT Solving. CP 2013</td>
</tr>
<tr>
<td>Year</td>
<td>Authors</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
</tr>
<tr>
<td>DHN06</td>
<td>N. Dershowitz, Z. Hanna, A. Nadel</td>
</tr>
<tr>
<td>FM06</td>
<td>Z. Fu, S. Malik</td>
</tr>
<tr>
<td>HJ90</td>
<td>P. Hansen, B. Jaumard</td>
</tr>
<tr>
<td>HLSB06</td>
<td>F. Hemery, C. Lecoutre, L. Sais, F. Boussemart</td>
</tr>
<tr>
<td>Ref</td>
<td>Author(s)</td>
</tr>
<tr>
<td>-----</td>
<td>----------</td>
</tr>
<tr>
<td>KBK09</td>
<td>Hans Kleine Buning, Oliver Kullmann</td>
</tr>
<tr>
<td>KK01</td>
<td>A. Kaiser, W. Kuchlin</td>
</tr>
<tr>
<td>Reference</td>
<td>Author(s)</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MHMS12</td>
<td>A. Morgado, F. Heras, J. Marques-Silva</td>
</tr>
<tr>
<td>MSJB13</td>
<td>J. Marques-Silva, M. Janota, A. Belov</td>
</tr>
</tbody>
</table>
References


MSP08  J. Marques-Silva, Jordi Planes: Algorithms for Maximum Satisfiability using Unsatifiable Cores. DATE 2008: 408-413


OOF05  B. O’Callaghan, B. O’Sullivan, E. Freuder: Generating Corrective Explanations for Interactive Constraint Satisfaction. CP 2005: 445-459

References


SW01  J. Slaney, T. Walsh: Backbones in Optimization and Approximation. IJCAI 2001: 254-259


Backbones — proof sketch

- $\nu(x_i)$: truth assignment given to $x_i$ in given reference model (optional, but simpler)

  - $F_{\{X/Y_i\}}$: formula with fresh set of variables $Y_i$, associated with each $x_i$

    - Introduce new variable $z_i \leftrightarrow (F_{\{X/Y_i\}} \land (y_i \leftrightarrow \neg \nu(x_i)))$ – $z_i = 1$ iff $F_{\{X/Y_i\}}$ satisfied with $y_i = \neg \nu(x_i)$

      - $\Rightarrow z_i = 1$ iff $x_i$ is not a backbone variable

    - Construct formula: $\forall i (z_i \leftrightarrow (F_{\{X/Y_i\}} \land (y_i \leftrightarrow \neg \nu(x_i))))$

    - Any $z_i$ that can take value 1 represents a non-backbone variable

      - Goal is to maximize the number of $z_i$ variables with value 1

      - Can be modeled with soft clauses: $(z_i)\\land F$

        - This is a (unweighted) partial MaxSAT problem

        - Can find solution with $O(\log n)$ calls to a SAT oracle

        - $\therefore$ Backbone is in $\text{FP}[\text{wit}, \log n]$
Backbones — proof sketch

- \( \nu(x_i) \): truth assignment given to \( x_i \) in given reference model (optional, but simpler)
- \( \mathcal{F}[X/Y_i] \): formula with fresh set of variables \( Y_i \), associated with each \( x_i \)

- Introduce new variable \( z_i \) \( \leftrightarrow (\mathcal{F}[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i))) \)
  - \( z_i = 1 \) iff \( \mathcal{F}[X/Y_i] \) satisfied with \( y_i = \neg \nu(x_i) \)
  - \( \Rightarrow \) \( z_i = 1 \) iff \( x_i \) is not a backbone variable

- Construct formula:
  \[
  \forall z_i \left( \text{var}(\mathcal{F}) \land z_i \leftrightarrow (\mathcal{F}[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i))) \right)
  \]

- Any \( z_i \) that can take value 1 represents a non-backbone variable
  - Goal is to maximize the number of \( z_i \) variables with value 1
  - Can be modeled with soft clauses: \( (z_i) \)

- This is a(n unweighted) partial MaxSAT problem
  - Can find solution with \( O(\log n) \) calls to a SAT oracle

- \( \therefore \) Backbone is in \( \text{FP}_{\text{NP}}[\text{wit},\log n] \)
Backbones — proof sketch

- $\nu(x_i)$: truth assignment given to $x_i$ in given reference model (optional, but simpler)
- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables $Y_i$, associated with each $x_i$
- Introduce new variable $z_i \leftrightarrow (\mathcal{F}[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i)))$
Backbones — proof sketch

• \( \nu(x_i) \): truth assignment given to \( x_i \) in given reference model (optional, but simpler)

• \( \mathcal{F}[X/Y_i] \): formula with fresh set of variables \( Y_i \), associated with each \( x_i \)

• Introduce new variable \( z_i \leftrightarrow (\mathcal{F}[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i))) \)
  
  \( - z_i = 1 \) iff \( \mathcal{F}[X/Y_i] \) satisfied with \( y_i = \neg \nu(x_i) \)
Backbones — proof sketch

- $\nu(x_i)$: truth assignment given to $x_i$ in given reference model (optional, but simpler)
- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables $Y_i$, associated with each $x_i$
- Introduce new variable $z_i \leftrightarrow (\mathcal{F}[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i)))$
  - $z_i = 1$ iff $\mathcal{F}[X/Y_i]$ satisfied with $y_i = \neg \nu(x_i)$
    - i.e. $z_i = 1$ iff $x_i$ is not a backbone variable
Backbones — proof sketch

- $\nu(x_i)$: truth assignment given to $x_i$ in given reference model (optional, but simpler)
- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables $Y_i$, associated with each $x_i$
- Introduce new variable $z_i \leftrightarrow (\mathcal{F}[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i)))$
  - $z_i = 1$ iff $\mathcal{F}[X/Y_i]$ satisfied with $y_i = \neg \nu(x_i)$
  - i.e. $z_i = 1$ iff $x_i$ is not a backbone variable
- Construct formula:
  $$\bigwedge_{i=1}^{\text{var}(\mathcal{F})} (z_i \leftrightarrow (\mathcal{F}[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i))))$$

Any $z_i$ that can take value 1 represents a non-backbone variable – Goal is to maximize the number of $z_i$ variables with value 1

- Can be modeled with soft clauses: $(z_i)$
- This is a(n unweighted) partial MaxSAT problem
- Can find solution with $O(\log n)$ calls to a SAT oracle
- $\therefore$ Backbone is in $\text{FP} \cap \text{NP}$[wit, log $n$]
Backbones — proof sketch

- $\nu(x_i)$: truth assignment given to $x_i$ in given reference model (optional, but simpler)
- $F[X/Y_i]$: formula with fresh set of variables $Y_i$, associated with each $x_i$
- Introduce new variable $z_i \leftrightarrow (F[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i)))$
  - $z_i = 1$ iff $F[X/Y_i]$ satisfied with $y_i = \neg \nu(x_i)$
  - i.e. $z_i = 1$ iff $x_i$ is not a backbone variable
- Construct formula:
  $$\bigwedge_{i=1}^{\text{var}(F)} (z_i \leftrightarrow (F[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i))))$$
- Any $z_i$ that can take value 1 represents a non-backbone variable
Backbones — proof sketch

- $\nu(x_i)$: truth assignment given to $x_i$ in given reference model (optional, but simpler)
- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables $Y_i$, associated with each $x_i$
- Introduce new variable $z_i \leftrightarrow (\mathcal{F}[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i)))$
  - $z_i = 1$ iff $\mathcal{F}[X/Y_i]$ satisfied with $y_i = \neg \nu(x_i)$
    - i.e. $z_i = 1$ iff $x_i$ is not a backbone variable
- Construct formula:
  $$\bigwedge_{i=1}^{\text{var}(\mathcal{F})} (z_i \leftrightarrow (\mathcal{F}[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i))))$$
- Any $z_i$ that can take value 1 represents a non-backbone variable
  - Goal is to maximize the number of $z_i$ variables with value 1
Backbones — proof sketch

- $\nu(x_i)$: truth assignment given to $x_i$ in given reference model (optional, but simpler)
- $F[X/Y_i]$: formula with fresh set of variables $Y_i$, associated with each $x_i$
- Introduce new variable $z_i \leftrightarrow (F[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i)))$
  - $z_i = 1$ iff $F[X/Y_i]$ satisfied with $y_i = \neg \nu(x_i)$
    - i.e. $z_i = 1$ iff $x_i$ is not a backbone variable
- Construct formula:
  $$\bigwedge_{i=1}^{\text{var}(F)} (z_i \leftrightarrow (F[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i))))$$
- Any $z_i$ that can take value 1 represents a non-backbone variable
  - Goal is to maximize the number of $z_i$ variables with value 1
  - Can be modeled with soft clauses: $(z_i)$
Backbones — proof sketch

- $\nu(x_i)$: truth assignment given to $x_i$ in given reference model (optional, but simpler)
- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables $Y_i$, associated with each $x_i$
  - Introduce new variable $z_i \leftrightarrow (\mathcal{F}[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i)))$
    - $z_i = 1$ iff $\mathcal{F}[X/Y_i]$ satisfied with $y_i = \neg \nu(x_i)$
      - i.e. $z_i = 1$ iff $x_i$ is not a backbone variable
- Construct formula:
  $$\bigwedge_{i=1}^{\text{var}(\mathcal{F})} (z_i \leftrightarrow (\mathcal{F}[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i))))$$
- Any $z_i$ that can take value 1 represents a non-backbone variable
  - Goal is to maximize the number of $z_i$ variables with value 1
  - Can be modeled with soft clauses: $(z_i)$
- This is a(n unweighted) partial MaxSAT problem

• $\nu(x_i)$: truth assignment given to $x_i$ in given reference model (optional, but simpler)
• $\mathcal{F}[X/Y_i]$: formula with fresh set of variables $Y_i$, associated with each $x_i$
  - Introduce new variable $z_i \leftrightarrow (\mathcal{F}[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i)))$
    - $z_i = 1$ iff $\mathcal{F}[X/Y_i]$ satisfied with $y_i = \neg \nu(x_i)$
      - i.e. $z_i = 1$ iff $x_i$ is not a backbone variable
• Construct formula:
  $$\bigwedge_{i=1}^{\text{var}(\mathcal{F})} (z_i \leftrightarrow (\mathcal{F}[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i))))$$
• Any $z_i$ that can take value 1 represents a non-backbone variable
  - Goal is to maximize the number of $z_i$ variables with value 1
  - Can be modeled with soft clauses: $(z_i)$
• This is a(n unweighted) partial MaxSAT problem
Backbones — proof sketch

• \( \nu(x_i) \): truth assignment given to \( x_i \) in given reference model (optional, but simpler)

• \( \mathcal{F}[X/Y_i] \): formula with fresh set of variables \( Y_i \), associated with each \( x_i \)

• Introduce new variable \( z_i \leftrightarrow (\mathcal{F}[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i))) \)
  - \( z_i = 1 \) iff \( \mathcal{F}[X/Y_i] \) satisfied with \( y_i = \neg \nu(x_i) \)
    ▶ i.e. \( z_i = 1 \) iff \( x_i \) is not a backbone variable

• Construct formula:
  \[
  \bigwedge_{i=1}^{\text{var}(\mathcal{F})} (z_i \leftrightarrow (\mathcal{F}[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i))))
  \]

• Any \( z_i \) that can take value 1 represents a non-backbone variable
  - Goal is to maximize the number of \( z_i \) variables with value 1
  - Can be modeled with soft clauses: \( (z_i) \)

• This is a(n unweighted) partial MaxSAT problem
  - Can find solution with \( O(\log n) \) calls to a SAT oracle

\[ \therefore \text{Backbone is in } \text{FP[} \text{NP[wit, log} n] \]
Backbones — proof sketch

- \( \nu(x_i) \): truth assignment given to \( x_i \) in given reference model (optional, but simpler)
- \( \mathcal{F}[X/Y_i] \): formula with fresh set of variables \( Y_i \), associated with each \( x_i \)
- Introduce new variable \( z_i \leftrightarrow (\mathcal{F}[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i))) \)
  - \( z_i = 1 \) iff \( \mathcal{F}[X/Y_i] \) satisfied with \( y_i = \neg \nu(x_i) \)
    - i.e. \( z_i = 1 \) iff \( x_i \) is not a backbone variable
- Construct formula:
  \[
  \bigwedge_{i=1}^{\text{var}(\mathcal{F})} (z_i \leftrightarrow (\mathcal{F}[X/Y_i] \land (y_i \leftrightarrow \neg \nu(x_i))))
  \]
- Any \( z_i \) that can take value 1 represents a non-backbone variable
  - Goal is to maximize the number of \( z_i \) variables with value 1
  - Can be modeled with soft clauses: \( (z_i) \)
- This is a(n unweighted) partial MaxSAT problem
  - Can find solution with \( \mathcal{O}(\log n) \) calls to a SAT oracle
- \( \therefore \) Backbone is in \( \text{FP}^{\text{NP}} \) [wit, log \( n \)]
MUS$_1$ — proof sketch

- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables $Y_i$, associated with each $c_i$

- Introduce new variable $z_i \leftrightarrow (\mathcal{F}[c_i])_{X/Y_i}$

  - $z_i = 1$ iff $(\mathcal{F}[c_i])_{X/Y_i}$ satisfied

  - i.e. $z_i = 1$ iff $c_i$ is in MUS, since there is exactly one MUS

- Construct formula:
  
  $|\mathcal{F}| \wedge \bigwedge_{i=1}^n (z_i \leftrightarrow (\mathcal{F}[c_i])_{X/Y_i})$

- Any $z_i$ that can take value 1 represents an MUS clause

  - Goal is to maximize the number of $z_i$ variables with value 1

  - Can be modeled with soft clauses: $(z_i)$

- This is a(n unweighted) partial MaxSAT problem

  - Can find solution with $O(\log n)$ calls to a SAT oracle

- $\therefore$ MUS$_1$ is in $\text{FP}^{\text{NP}[\text{wit}, \log n]}$
MUS\#1 — proof sketch

- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables $Y_i$, associated with each $c_i$
- Introduce new variable $z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i]$
**MUS#1 — proof sketch**

- \( \mathcal{F}[X/Y_i] \): formula with fresh set of variables \( Y_i \), associated with each \( c_i \)
- Introduce new variable \( z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i] \)
  - \( z_i = 1 \) iff \( (\mathcal{F} \setminus \{c_i\})[X/Y_i] \) satisfied
MUS\#1 — proof sketch

- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables $Y_i$, associated with each $c_i$
- Introduce new variable $z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i]$
  - $z_i = 1$ iff $(\mathcal{F} \setminus \{c_i\})[X/Y_i]$ satisfied
    - i.e. $z_i = 1$ iff $c_i$ is in MUS, since there is exactly one MUS
MUS$_{#1}$ — proof sketch

- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables $Y_i$, associated with each $c_i$
- Introduce new variable $z_i \leftrightarrow (\mathcal{F}\setminus c_i)[X/Y_i]$
  - $z_i = 1$ iff $(\mathcal{F}\setminus \{c_i\})[X/Y_i]$ satisfied
    - i.e. $z_i = 1$ iff $c_i$ is in MUS, since there is exactly one MUS
- Construct formula:

$$\bigwedge_{i=1}^{\vert \mathcal{F} \vert} (z_i \leftrightarrow (\mathcal{F}\setminus c_i)[X/Y_i])$$
MUS \#1 — proof sketch

- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables $Y_i$, associated with each $c_i$
- Introduce new variable $z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i]$
  - $z_i = 1$ iff $(\mathcal{F} \setminus \{c_i\})[X/Y_i]$ satisfied
    - i.e. $z_i = 1$ iff $c_i$ is in MUS, since there is exactly one MUS
- Construct formula:
  $$|\mathcal{F}| \bigwedge_{i=1}^{n} (z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i])$$
- Any $z_i$ that can take value 1 represents an MUS clause
MUS\#1 — proof sketch

• $\mathcal{F}[X/Y_i]$: formula with fresh set of variables $Y_i$, associated with each $c_i$

• Introduce new variable $z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i]$
  - $z_i = 1$ iff $(\mathcal{F} \setminus \{c_i\})[X/Y_i]$ satisfied
    ▶ i.e. $z_i = 1$ iff $c_i$ is in MUS, since there is exactly one MUS

• Construct formula:

$$\left| \mathcal{F} \right| \bigwedge_{i=1}^{n} (z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i])$$

• Any $z_i$ that can take value 1 represents an MUS clause
  - Goal is to maximize the number of $z_i$ variables with value 1
MUS\#1 — proof sketch

- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables $Y_i$, associated with each $c_i$
- Introduce new variable $z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i]$
  - $z_i = 1$ iff $(\mathcal{F} \setminus \{c_i\})[X/Y_i]$ satisfied
    - i.e. $z_i = 1$ iff $c_i$ is in MUS, since there is exactly one MUS
- Construct formula:
  $$\bigwedge_{i=1}^{\lvert \mathcal{F} \rvert} (z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i])$$
- Any $z_i$ that can take value 1 represents an MUS clause
  - Goal is to maximize the number of $z_i$ variables with value 1
  - Can be modeled with soft clauses: $(z_i)$
MUS\#1 — proof sketch

- \(\mathcal{F}[X/Y_i]\): formula with fresh set of variables \(Y_i\), associated with each \(c_i\)
- Introduce new variable \(z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i]\)
  - \(z_i = 1\) iff \((\mathcal{F} \setminus \{c_i\})[X/Y_i]\) satisfied
    - i.e. \(z_i = 1\) iff \(c_i\) is in MUS, since there is exactly one MUS
- Construct formula:
  \[
  \bigwedge_{i=1}^{\mathcal{F}} (z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i])
  \]
- Any \(z_i\) that can take value 1 represents an MUS clause
  - Goal is to maximize the number of \(z_i\) variables with value 1
  - Can be modeled with soft clauses: \((z_i)\)
- This is a(n unweighted) partial MaxSAT problem
MUS \#1 — proof sketch

- \( \mathcal{F}[X/Y_i] \): formula with fresh set of variables \( Y_i \), associated with each \( c_i \)
- Introduce new variable \( z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i] \)
  - \( z_i = 1 \) iff \( (\mathcal{F} \setminus \{c_i\})[X/Y_i] \) satisfied
    - i.e. \( z_i = 1 \) iff \( c_i \) is in MUS, since there is exactly one MUS
- Construct formula:
  \[
  \bigwedge_{i=1}^{\mathcal{F}} (z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i])
  \]
- Any \( z_i \) that can take value 1 represents an MUS clause
  - Goal is to maximize the number of \( z_i \) variables with value 1
  - Can be modeled with soft clauses: \( (z_i) \)
- This is a(n unweighted) partial MaxSAT problem
  - Can find solution with \( \mathcal{O}(\log n) \) calls to a SAT oracle
MUS$_1$ — proof sketch

- $\mathcal{F}[X/Y_i]$: formula with fresh set of variables $Y_i$, associated with each $c_i$
- Introduce new variable $z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i]$
  - $z_i = 1$ iff $(\mathcal{F} \setminus \{c_i\})[X/Y_i]$ satisfied
    - i.e. $z_i = 1$ iff $c_i$ is in MUS, since there is exactly one MUS
- Construct formula:
  \[
  |\mathcal{F}| \land \bigwedge_{i=1}^{n} (z_i \leftrightarrow (\mathcal{F} \setminus c_i)[X/Y_i])
  \]
  - Any $z_i$ that can take value 1 represents an MUS clause
    - Goal is to maximize the number of $z_i$ variables with value 1
    - Can be modeled with soft clauses: $(z_i)$
- This is a(n unweighted) partial MaxSAT problem
  - Can find solution with $O(\log n)$ calls to a SAT oracle
- $\therefore$ MUS$_1$ is in FP$^N$P$^N$[wit, log $n$]