Computing with SAT Oracles

Joao Marques-Silva

SAT/SMT/AR 2019 Summer School
IST, Lisbon, Portugal
July 3-6 2019
Computing with SAT Oracles

Joao Marques-Silva

SAT/SMT/AR 2019 Summer School
IST, Lisbon, Portugal
July 3-6 2019
What is SAT?

- **SAT** is the decision problem for propositional logic
  - Well-formed propositional formulas, with variables, logical connectives: \( \neg, \land, \lor, \rightarrow, \leftrightarrow \), and parenthesis: (, )
  - Often restricted to Conjunctive Normal Form (CNF)
What is SAT?

- **SAT** is the *decision problem* for propositional logic
  - Well-formed *propositional formulas*, with variables, logical connectives: $\neg$, $\land$, $\lor$, $\rightarrow$, $\leftrightarrow$, and parenthesis: $(, )$
  - Often restricted to *Conjunctive Normal Form (CNF)*
  - **Goal:**
    
    Decide whether formula has a satisfying assignment
What is SAT?

- **SAT** is the decision problem for propositional logic
  - Well-formed propositional formulas, with variables, logical connectives: $\neg, \land, \lor, \rightarrow, \leftrightarrow$, and parenthesis: (, )
  - Often restricted to Conjunctive Normal Form (CNF)
  - **Goal:** Decide whether formula has a satisfying assignment

- **SAT** is NP-complete

[Coo71]
• **CDCL SAT solving** is a *success story* of Computer Science
The CDCL SAT disruption

- CDCL SAT solving is a success story of Computer Science
  - Conflict-Driven Clause Learning (CDCL)
  - (CDCL) SAT has impacted many different fields
  - Hundreds (thousands?) of practical applications
CDCL SAT solver (continued) improvement  

[Source: Simon 2015]
How good are CDCL SAT solvers?

Demos

1. POSIT: state of the art DPLL SAT solver in 1995
2. GRASP: first CDCL SAT solver, state of the art 1995–2000
3. Minisat: CDCL SAT solver, state of the art until the late 00s
4. Glucose: modern state of the art CDCL SAT solver
5. ...
How good are CDCL SAT solvers?

Demos

• Sample SAT of solvers:
  1. **POSIT**: state of the art **DPLL** SAT solver in 1995
  2. **GRASP**: first **CDCL** SAT solver, state of the art 1995~2000
  3. **Minisat**: **CDCL** SAT solver, state of the art until the late 00s
  4. **Glucose**: modern state of the art **CDCL** SAT solver
  5. ...
How good are CDCL SAT solvers?

Demos

• Sample SAT of solvers:
  1. **POSIT**: state of the art DPLL SAT solver in 1995
  2. **GRASP**: first CDCL SAT solver, state of the art 1995~2000
  3. **Minisat**: CDCL SAT solver, state of the art until the late 00s
  4. **Glucose**: modern state of the art CDCL SAT solver
  5. ...

• **Example 1**: model checking example (from IBM)
• **Example 2**: cooperative path finding (CPF)
How good are SAT solvers? – an example

- **Cooperative pathfinding (CPF)**
  - $N$ agents on some grid/graph
  - **Start** positions
  - **Goal** positions
  - Minimize **makespan**
  - Restricted planning problem

- Note: In the early 90s, SAT solvers could solve formulas with a few hundred variables!
How good are SAT solvers? – an example

• Cooperative pathfinding (CPF)
  • $N$ agents on some grid/graph
  • Start positions
  • Goal positions
  • Minimize makespan
  • Restricted planning problem

• Concrete example
  • Gaming grid
  • 1039 vertices
  • 1928 edges
  • 100 agents

Note: In the early 90s, SAT solvers could solve formulas with a few hundred variables.
How good are SAT solvers? – an example

• Cooperative pathfinding (CPF)
  • $N$ agents on some grid/graph
  • **Start** positions
  • **Goal** positions
  • Minimize **makespan**
  • Restricted planning problem

• Concrete example
  • Gaming grid
  • 1039 vertices
  • 1928 edges
  • 100 agents

---

*** tracker: a pathfinding tool ***

Initialization ... CPU Time: 0.004711
Number of variables: 113315
Tentative makespan 1
Number of variables: 226630
Number of assumptions: 1
c Running SAT solver ... CPU Time: 0.718112
c Done running SAT solver ... CPU Time: 0.830099
No solution for makespan 1
Elapsed CPU Time: 0.830112
Tentative makespan 2
Number of variables: 339945
Number of assumptions: 1
c Running SAT solver ... CPU Time: 1.27113
c Done running SAT solver ... CPU Time: 1.27114
No solution for makespan 2
Elapsed CPU Time: 1.27114
...
Tentative makespan 24
Number of variables: 2832875
Number of assumptions: 1
c Running SAT solver ... CPU Time: 11.8653
c Done running SAT solver ... CPU Time: 11.8653
No solution for makespan 24
Elapsed CPU Time: 11.8653
Tentative makespan 25
Number of variables: 2946190
Number of assumptions: 1
c Running SAT solver ... CPU Time: 12.3491
c Done running SAT solver ... CPU Time: 16.6882
Solution found for makespan 25
Elapsed CPU Time: 16.6995

In the early 90s, SAT solvers could solve formulas with a few hundred variables.
**How good are SAT solvers? – an example**

- **Cooperative pathfinding (CPF)**
  - $N$ agents on some grid/graph
  - **Start** positions
  - **Goal** positions
  - Minimize **makespan**
  - Restricted planning problem

- **Concrete example**
  - Gaming grid
  - 1039 vertices
  - 1928 edges
  - 100 agents
  - Formula w/ 2946190 variables!

---

```plaintext
*** tracker: a pathfinding tool ***

Initialization ... CPU Time: 0.004711
Number of variables: 113315
Tentative makespan 1
Number of variables: 226630
Number of assumptions: 1
c Running SAT solver ... CPU Time: 0.718112
c Done running SAT solver ... CPU Time: 0.830099
No solution for makespan 1
Elapsed CPU Time: 0.830112
Tentative makespan 2
Number of variables: 339945
Number of assumptions: 1
c Running SAT solver ... CPU Time: 1.27113
c Done running SAT solver ... CPU Time: 1.27114
No solution for makespan 2
Elapsed CPU Time: 1.27114
...
...
Tentative makespan 24
Number of variables: 2832875
Number of assumptions: 1
c Running SAT solver ... CPU Time: 11.8653
c Done running SAT solver ... CPU Time: 11.8653
No solution for makespan 24
Elapsed CPU Time: 11.8653
Tentative makespan 25
Number of variables: 2946190
Number of assumptions: 1
c Running SAT solver ... CPU Time: 12.3491
c Done running SAT solver ... CPU Time: 16.6882
Solution found for makespan 25
Elapsed CPU Time: 16.6995
```
How good are SAT solvers? – an example

• Cooperative pathfinding (CPF)
  • $N$ agents on some grid/graph
  • **Start** positions
  • **Goal** positions
  • Minimize **makespan**
  • Restricted planning problem

• Concrete example
  • Gaming grid
  • 1039 vertices
  • 1928 edges
  • 100 agents
  • **Formula w/ 2946190 variables!**

• **Note:** In the early 90s, SAT solvers could solve formulas **with a few hundred variables!**

*** tracker: a pathfinding tool ***

Initialization ... CPU Time: 0.004711
Number of variables: 113315
Tentative makespan 1
Number of variables: 226630
Number of assumptions: 1
c Running SAT solver ... CPU Time: 0.718112
c Done running SAT solver ... CPU Time: 0.830099
No solution for makespan 1
Elapsed CPU Time: 0.830112
Tentative makespan 2
Number of variables: 339945
Number of assumptions: 1
c Running SAT solver ... CPU Time: 1.27113
c Done running SAT solver ... CPU Time: 1.27114
No solution for makespan 2
Elapsed CPU Time: 1.27114
...  
...  
Tentative makespan 24
Number of variables: 2832875
Number of assumptions: 1
c Running SAT solver ... CPU Time: 11.8653
c Done running SAT solver ... CPU Time: 11.8653
No solution for makespan 24
Elapsed CPU Time: 11.8653
Tentative makespan 25
Number of variables: 2946190
Number of assumptions: 1
c Running SAT solver ... CPU Time: 12.3491
c Done running SAT solver ... CPU Time: 16.6882
Solution found for makespan 25
Elapsed CPU Time: 16.6995
Grasping the search space ...

- Number of seconds since the Big Bang: $\approx 10^{17}$
Grasping the search space ...

- Number of seconds since the Big Bang: $\approx 10^{17}$

- Number of fundamental particles in observable universe: $\approx 10^{80}$ (or $\approx 10^{85}$)
Grasping the search space ...

- Number of seconds since the Big Bang: $\approx 10^{17}$

- Number of fundamental particles in observable universe: $\approx 10^{80}$ (or $\approx 10^{85}$)

- Search space with 15775 propositional variables (worst case):
Grasping the search space ...

• Number of seconds since the Big Bang: \( \approx 10^{17} \)

• Number of fundamental particles in observable universe: \( \approx 10^{80} \) (or \( \approx 10^{85} \))

• Search space with 15775 propositional variables (worst case):
  • # of assignments to 15775 variables: \( > 10^{4748} \) !
  • **Obs:** SAT solvers in the late 90s (but formula dependent)
Grasping the search space ...

• Number of seconds since the Big Bang: \( \approx 10^{17} \)

• Number of fundamental particles in observable universe: \( \approx 10^{80} \) (or \( \approx 10^{85} \))

• Search space with 15775 propositional variables (worst case):
  • # of assignments to 15775 variables: \( > 10^{4748} \)!
  • **Obs:** SAT solvers in the late 90s (but formula dependent)

• Search space with 2832875 propositional variables (worst case):
Grasping the search space ...

- Number of seconds since the Big Bang: $\approx 10^{17}$

- Number of fundamental particles in observable universe: $\approx 10^{80}$ (or $\approx 10^{85}$)

- Search space with 15775 propositional variables (worst case):
  - # of assignments to 15775 variables: $> 10^{4748}$
  - **Obs:** SAT solvers in the late 90s (but formula dependent)

- Search space with 2832875 propositional variables (worst case):
  - # of assignments to $> 2.8 \times 10^6$ variables: $\gg 10^{840000}$
  - **Obs:** SAT solvers at present (but formula dependent)
SAT can make the difference – propositional abduction

- Propositional abduction instances
  - Implicit hitting set dualization (IHSD)

[IMM16]
SAT can make the difference – axiom pinpointing

- $\mathcal{EL}^+$ medical ontologies
  - Minimal unsatisfiability (MUSes) & maximal satisfiability (MCSes) & Enumeration

[AMM15]
SAT can make the difference – model based diagnosis

• Model-based diagnosis problem instances
  • Maximum satisfiability (MaxSAT)

[MJIM15]
CDCL SAT is ubiquitous in problem solving
CDCL SAT is ubiquitous in problem solving

SAT is the oracles’ oracle:
MaxSAT, QBF, LCG, #SAT, SMT, ASP, FOL, ...
What this tutorial covers...

- Part #0: Basic definitions & notation
What this tutorial covers ... 

- Part #0: Basic definitions & notation
- Part #1: Problem solving with SAT oracles
  - Minimal unsatisfiability (MUS)
  - Maximum satisfiability (MaxSAT)
  - Maximal satisfiability (MSS/MCS)
  - Minimal Sets over Monotone Predicates (MSMP)
  - Enumeration problems
    - MUSes
  - Quantification problems
  - (Approximate) counting problems
  - ...
- Part #2: Exploring with SAT oracles
  - Brief introduction to PySAT
- Part #3: Research directions
  - Contact me
What this tutorial covers ...

- Part #0: Basic definitions & notation
- Part #1: Problem solving with SAT oracles
  - Minimal unsatisfiability (MUS)
  - Maximum satisfiability (MaxSAT)
  - Maximal satisfiability (MSS/MCS)
  - Minimal Sets over Monotone Predicates (MSMP)
  - Enumeration problems
    - MUSes
  - Quantification problems
  - (Approximate) counting problems
  - ...
- Part #2: Exploring with SAT oracles
  - Brief introduction to PySAT
• Part #0: Basic definitions & notation

• Part #1: Problem solving with SAT oracles
  • Minimal unsatisfiability (MUS)
  • Maximum satisfiability (MaxSAT)
  • Maximal satisfiability (MSS/MCS)
  • Minimal Sets over Monotone Predicates (MSMP)
  • Enumeration problems
    • MUSes
  • Quantification problems
  • (Approximate) counting problems
  • ...

• Part #2: Exploring with SAT oracles
  • Brief introduction to PySAT

• Part #3: Research directions
What this tutorial does not cover ...

- CDCL SAT solvers
  - Clause learning; search restarts; watched literals; VSIDS; ...

- Modeling in propositional logic
  - Cardinality constraints; pseudo-boolean constraints; circuits; general constraints; etc.

- Many (high-profile) applications
  - Minimal/minimum decision trees/sets
  - ML model explanations as prime implicants
  - ...

A. Biere’s talk

Contact me

[14 / 76]
• Variables: $w, x, y, z, a, b, c, \ldots$
• Literals: $w, \bar{x}, \bar{y}, a, \ldots$, but also $\neg w, \neg y, \ldots$
• Clauses: disjunction of literals or set of literals
• Formula: conjunction of clauses or set of clauses
• Model (satisfying assignment): partial/total mapping from variables to $\{0, 1\}$ that satisfies formula
• Each clause can be satisfied, falsified, but also unit, unresolved
• Formula can be SAT/UNSAT
Preliminaries

- Variables: $w, x, y, z, a, b, c, \ldots$
- Literals: $w, \bar{x}, \bar{y}, a, \ldots$, but also $\neg w, \neg y, \ldots$
- Clauses: disjunction of literals or set of literals
- Formula: conjunction of clauses or set of clauses
- Model (satisfying assignment): partial/total mapping from variables to $\{0, 1\}$ that satisfies formula
- Each clause can be satisfied, falsified, but also unit, unresolved
- Formula can be SAT/UNSAT
- Example:

$$\mathcal{F} \triangleq (r) \land (\bar{r} \lor s) \land (\bar{w} \lor a) \land (\bar{x} \lor b) \land (\bar{y} \lor \bar{z} \lor c) \land (\bar{b} \lor \bar{c} \lor d)$$

- Example models:
Preliminaries

- **Variables**: \( w, x, y, z, a, b, c, \ldots \)
- **Literals**: \( w, \neg x, \neg y, a, \ldots \), but also \( \neg w, \neg y, \ldots \)
- **Clauses**: disjunction of literals or set of literals
- **Formula**: conjunction of clauses or set of clauses
- **Model** (satisfying assignment): partial/total mapping from variables to \( \{0, 1\} \) that satisfies formula
- Each clause can be **satisfied**, **falsified**, but also **unit**, **unresolved**
- Formula can be **SAT/UNSAT**
- Example:

\[
F \triangleq (r) \land (\neg r \lor s) \land (\neg w \lor a) \land (\neg x \lor b) \land (\neg y \lor \neg z \lor c) \land (\neg b \lor \neg c \lor d)
\]

- Example models:
  - \( \{r, s, a, b, c, d\} \)
Preliminaries

• Variables: $w, x, y, z, a, b, c, \ldots$
• Literals: $w, \overline{x}, \overline{y}, a, \ldots$, but also $\neg w, \neg y, \ldots$
• Clauses: disjunction of literals or set of literals
• Formula: conjunction of clauses or set of clauses
• Model (satisfying assignment): partial/total mapping from variables to $\{0, 1\}$ that satisfies formula
• Each clause can be satisfied, falsified, but also unit, unresolved
• Formula can be SAT/UNSAT
• Example:

$$\mathcal{F} \equiv (r) \land (\overline{r} \lor s) \land (\overline{w} \lor a) \land (\overline{x} \lor b) \land (\overline{y} \lor \overline{z} \lor c) \land (\overline{b} \lor \overline{c} \lor d)$$

• Example models:
  • $\{r, s, a, b, c, d\}$
  • $\{r, s, \overline{x}, y, \overline{w}, z, \overline{a}, b, c, d\}$
Resolution

• Resolution rule:

\[
\frac{(\alpha \lor x) \quad (\beta \lor \overline{x})}{(\alpha \lor \beta)}
\]

• Complete proof system for propositional logic
**Resolution**

- **Resolution rule:**

\[
\frac{(\alpha \lor x) \quad (\beta \lor \bar{x})}{(\alpha \lor \beta)}
\]

- **Complete proof system for propositional logic**

\[
\begin{align*}
(x \lor a) \\
(\bar{x} \lor a) \\
(\bar{y} \lor a) \\
(y \lor \bar{a})
\end{align*}
\]

- **Extensively used with (CDCL) SAT solvers**
Unit propagation

\[ \mathcal{F} = (r) \land (\bar{r} \lor s) \land \\
(\bar{w} \lor a) \land (\bar{x} \lor \bar{a} \lor b) \land \\
(\bar{y} \lor \bar{z} \lor c) \land (\bar{b} \lor \bar{c} \lor d) \]
Unit propagation

\[ F = (r) \land (\bar{r} \lor s) \land \]
\[ (\bar{w} \lor a) \land (\bar{x} \lor \bar{a} \lor b) \land \]
\[ (\bar{y} \lor \bar{z} \lor c) \land (\bar{b} \lor \bar{c} \lor d) \]

- Decisions / Variable Branchings:
  \[ w = 1, x = 1, y = 1, z = 1 \]
Unit propagation

\[ \mathcal{F} = (r) \land (\bar{r} \lor s) \land (\bar{w} \lor a) \land (\bar{x} \lor \bar{a} \lor b) \land (\bar{y} \lor \bar{z} \lor c) \land (\bar{b} \lor \bar{c} \lor d) \]

- **Decisions / Variable Branchings:**
  \[ w = 1, x = 1, y = 1, z = 1 \]

- **Unit clause rule:** if clause is unit, its sole literal **must** be satisfied
Unit propagation

\[ F = (r) \land (\overline{r} \lor s) \land \\
(\overline{w} \lor a) \land (\overline{x} \lor \overline{a} \lor b) \land \\
(\overline{y} \lor \overline{z} \lor c) \land (\overline{b} \lor \overline{c} \lor d) \]

- **Decisions / Variable Branchings:**
  \[ w = 1, x = 1, y = 1, z = 1 \]

- **Unit clause rule:** if clause is unit, its sole literal **must** be satisfied
Unit propagation

\[ F = (r) \land (\bar{r} \lor s) \land (\bar{w} \lor a) \land (\bar{x} \lor \bar{a} \lor b) \land (\bar{y} \lor \bar{z} \lor c) \land (\bar{b} \lor \bar{c} \lor d) \]

- **Decisions / Variable Branchings:**
  \[ w = 1, x = 1, y = 1, z = 1 \]

- **Unit clause rule:** if clause is unit, its sole literal **must** be satisfied

- **Additional definitions:**
  - **Antecedent (or reason) of an implied assignment**
    - \((\bar{b} \lor \bar{c} \lor d)\) for \(d\)
  - **Associate assignment with decision levels**
    - \(w = 1@1, x = 1@2, y = 1@3, z = 1@4\)
    - \(r = 1@0, d = 1@4, \ldots\)
Resolution proofs

- Refutation of unsatisfiable formula by iterated resolution operations produces resolution proof

- An example:
  \[ \mathcal{F} = (\overline{c}) \land (\overline{b}) \land (\overline{a} \lor c) \land (a \lor b) \land (a \lor \overline{d}) \land (\overline{a} \lor \overline{d}) \]

- Resolution proof:

```
  (a \lor b)   (\overline{a} \lor c)
     \downarrow     \downarrow
  (\overline{c}) (b \lor c)
     \downarrow     \downarrow
  (\overline{b}) (b)  \bot
```

- Modern SAT solvers can generate resolution proofs using clauses learned by the solver

[ZM03]
Unsatisfiable cores & proof traces

• CNF formula:

\[ \mathcal{F} = (\overline{c}) \land (\overline{b}) \land (\overline{a} \lor c) \land (a \lor b) \land (a \lor \overline{d}) \land (\overline{a} \lor \overline{d}) \]

<table>
<thead>
<tr>
<th>Level</th>
<th>Dec.</th>
<th>Unit Prop.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>\emptyset</td>
<td>\overline{b} \rightarrow a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\overline{c} \rightarrow \bot</td>
</tr>
</tbody>
</table>

Implication graph with conflict
Unsatisfiable cores & proof traces

- CNF formula:

\[ \mathcal{F} = (\overline{c}) \land (\overline{b}) \land (\overline{a} \lor c) \land (a \lor b) \land (a \lor \overline{d}) \land (\overline{a} \lor \overline{d}) \]

<table>
<thead>
<tr>
<th>Level</th>
<th>Dec.</th>
<th>Unit Prop.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\emptyset$</td>
<td>$\overline{b} \rightarrow a$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\overline{c} \rightarrow \bot$</td>
</tr>
</tbody>
</table>

Proof trace $\bot$: $(\overline{a} \lor c) (a \lor b) (\overline{c}) (\overline{b})$
Unsatisfiable cores & proof traces

- CNF formula:

\[ \mathcal{F} = (\overline{c}) \land (\overline{b}) \land (\overline{a} \lor c) \land (a \lor b) \land (a \lor \overline{d}) \land (\overline{a} \lor \overline{d}) \]

<table>
<thead>
<tr>
<th>Level</th>
<th>Dec.</th>
<th>Unit Prop.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>\emptyset</td>
<td>\overline{b} \rightarrow a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\overline{c} \rightarrow \bot</td>
</tr>
</tbody>
</table>

Resolution proof follows structure of conflicts
Unsatisfiable cores & proof traces

- CNF formula:

\[ \mathcal{F} = (\overline{c}) \land (\overline{b}) \land (\overline{a} \lor c) \land (a \land b) \land (a \lor d) \land (\overline{a} \land \overline{d}) \]

<table>
<thead>
<tr>
<th>Level</th>
<th>Dec.</th>
<th>Unit Prop.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\emptyset$</td>
<td>$b \rightarrow a$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\overline{c} \rightarrow \bot$</td>
</tr>
</tbody>
</table>

Unsatisfiable subformula (core): $(\overline{c}), (\overline{b}), (\overline{a} \lor c), (a \lor b)$
Problem Solving with SAT Oracles
So what are **SAT oracles**?
So what are **SAT oracles**?
Computing a model

• **Q:** How to solve the FSAT problem?

  **FSAT:** Compute a model of a satisfiable CNF formula $\mathcal{F}$, using an NP oracle
Computing a model

• **Q:** How to solve the **FSAT** problem?

  **FSAT:** Compute a model of a satisfiable CNF formula $\mathcal{F}$, using an NP oracle

  • A possible algorithm:
    1. Analyze each variable $x_i \in \{x_1, \ldots, x_n\} = \text{var}(\mathcal{F})$, in order
    2. $i \leftarrow 1$ and $\mathcal{F}_i \triangleq \mathcal{F}$
    3. Call NP oracle on $\mathcal{F}_i \land (x_i)$
    4. If answer is **yes**, then $\mathcal{F}_{i+1} \leftarrow \mathcal{F}_i \cup (x_i)$
    5. If answer is **no**, then $\mathcal{F}_{i+1} \leftarrow \mathcal{F}_i \cup (\neg x_i)$
    6. $i \leftarrow i + 1$
    7. If $i \leq n$, then repeat from 3.
Computing a model

Q: How to solve the FSAT problem?

FSAT: Compute a model of a satisfiable CNF formula $\mathcal{F}$, using an NP oracle

A possible algorithm:

1. Analyze each variable $x_i \in \{x_1, \ldots, x_n\} = \text{var} (\mathcal{F})$, in order
2. $i \leftarrow 1$ and $\mathcal{F}_i \triangleq \mathcal{F}$
3. Call NP oracle on $\mathcal{F}_i \land (x_i)$
4. If answer is yes, then $\mathcal{F}_{i+1} \leftarrow \mathcal{F}_i \cup (x_i)$
5. If answer is no, then $\mathcal{F}_{i+1} \leftarrow \mathcal{F}_i \cup (\neg x_i)$
6. $i \leftarrow i + 1$
7. If $i \leq n$, then repeat from 3.

Algorithm needs $|\text{var} (\mathcal{F})|$ calls to an NP oracle
Computing a model

• **Q:** How to solve the **FSAT** problem?

  **FSAT:** Compute a model of a satisfiable CNF formula $\mathcal{F}$, using an NP oracle

  • A possible algorithm:
    1. Analyze each variable $x_i \in \{x_1, \ldots, x_n\} = \text{var}(\mathcal{F})$, in order
    2. $i \leftarrow 1$ and $\mathcal{F}_i \triangleq \mathcal{F}$
    3. Call NP oracle on $\mathcal{F}_i \wedge (x_i)$
    4. If answer is **yes**, then $\mathcal{F}_{i+1} \leftarrow \mathcal{F}_i \cup (x_i)$
    5. If answer is **no**, then $\mathcal{F}_{i+1} \leftarrow \mathcal{F}_i \cup (\neg x_i)$
    6. $i \leftarrow i + 1$
    7. If $i \leq n$, then repeat from 3.

  • Algorithm needs $|\text{var}(\mathcal{F})|$ calls to an NP oracle

  • **Note:** Cannot solve FSAT with logarithmic number of NP oracle calls, unless $P = NP$ [GF93]

• FSAT is an example of a function problem
Computing a model

• **Q:** How to solve the **FSAT** problem?

**FSAT:** Compute a model of a satisfiable CNF formula $\mathcal{F}$, using an NP oracle

• A possible algorithm:
  1. Analyze each variable $x_i \in \{x_1, \ldots, x_n\} = \text{var}(\mathcal{F})$, in order
  2. $i \leftarrow 1$ and $\mathcal{F}_i \triangleq \mathcal{F}$
  3. Call NP oracle on $\mathcal{F}_i \land (x_i)$
  4. If answer is **yes**, then $\mathcal{F}_{i+1} \leftarrow \mathcal{F}_i \cup (x_i)$
  5. If answer is **no**, then $\mathcal{F}_{i+1} \leftarrow \mathcal{F}_i \cup (\neg x_i)$
  6. $i \leftarrow i + 1$
  7. If $i \leq n$, then repeat from 3.

• Algorithm needs $|\text{var}(\mathcal{F})|$ calls to an NP oracle

• **Note:** Cannot solve FSAT with logarithmic number of NP oracle calls, unless $P = NP$  

• FSAT is an example of a **function** problem

• **Note:** FSAT can be solved with **one** SAT oracle call
## Beyond decision problems

<table>
<thead>
<tr>
<th>Answer</th>
<th>Problem Type</th>
</tr>
</thead>
</table>

- Decision Problems
  - Some solution
- Function Problems
  - All solutions
- Enumeration Problems
  - # solutions
- Counting Problems
Beyond decision 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>
## Beyond decision 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>Some solution</td>
<td></td>
</tr>
</tbody>
</table>
### Beyond decision 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>Some solution</td>
<td>Function Problems</td>
</tr>
</tbody>
</table>
Beyond decision 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>Some solution</td>
<td>Function Problems</td>
</tr>
<tr>
<td>All solutions</td>
<td></td>
</tr>
<tr>
<td>Answer</td>
<td>Problem Type</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>Yes/No</td>
<td>Decision Problems</td>
</tr>
<tr>
<td>Some solution</td>
<td>Function Problems</td>
</tr>
<tr>
<td>All solutions</td>
<td>Enumeration Problems</td>
</tr>
</tbody>
</table>

Beyond decision problems
## Beyond decision 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>Some solution</td>
<td>Function Problems</td>
</tr>
<tr>
<td>All solutions</td>
<td>Enumeration Problems</td>
</tr>
<tr>
<td># solutions</td>
<td></td>
</tr>
</tbody>
</table>
Beyond decision 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>Some solution</td>
<td>Function Problems</td>
</tr>
<tr>
<td>All solutions</td>
<td>Enumeration Problems</td>
</tr>
<tr>
<td># solutions</td>
<td>Counting Problems</td>
</tr>
</tbody>
</table>
... and beyond NP – decision and function problems

\[
\begin{align*}
\Delta_0^p &= \Sigma_0^p = P = \Pi_0^p = \Delta_1^p \\
NP &= \Sigma_1^p \\
P^\text{NP} &= \Delta_2^p \\
\Sigma_2^p &\quad \Pi_2^p \\
\Delta_3^p &\quad \Pi_3^p \\
FNP &= F\Sigma_1^p \\
FP^\text{NP} &= F\Delta_2^p \\
\Sigma_3^p &\quad \Pi_3^p \\
F\Sigma_3^p &\quad F\Pi_3^p \\
F\Delta_3^p &\quad F\Pi_1^p = \text{coFNP}
\end{align*}
\]
Oracle-based problem solving – simple scenario

Poly-time Algorithm

Yes/No + Witness

Bounded # of calls / queries

Decision Procedure

SAT, SMT, CSP, ...
Solver / Oracle
Oracle-based problem solving – general setting

Poly-time Algorithm

Yes/No + Witness

Decision Procedure

Bounded # of calls / queries

SAT, SMT, CSP, ...
Solver / Oracle
Many problems to solve – within $FP^{NP}$

<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>Some solution</td>
<td>Function Problems</td>
</tr>
<tr>
<td>All solutions</td>
<td>Enumeration Problems</td>
</tr>
</tbody>
</table>
Many problems to solve – within $FP^{NP}$

<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>Some solution</td>
<td>Function Problems</td>
</tr>
<tr>
<td>All solutions</td>
<td>Enumeration Problems</td>
</tr>
</tbody>
</table>

**Function Problems on Propositional Formulas**

- MaxSAT
- PBO
- MinSAT
- WBO
- Minimal Models
- Maximal Models
- Prime Implicates
- Autarkies
- Backbones
- Prime Implicants
- MUSes
- MCSes
- MESes
- Indep. Vars
- MFSes
- MSSes
- MDSes
- Implicates Ext.
- MCFSES
- MNSes
- Implicate Ext.
- ...
Many problems to solve – within $\text{FP}^\text{NP}$

<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>Some solution</td>
<td>Function Problems</td>
</tr>
<tr>
<td>All solutions</td>
<td>Enumeration Problems</td>
</tr>
</tbody>
</table>

Function Problems on Propositional Formulas

- Optimization Problems
  - MaxSAT
  - PBO
  - MinSAT
  - WBO

- Minimal Sets
  - Minimal Models
  - Maximal Models
  - Backbones
  - Prime Implicates
  - Autarkies
  - Prime Implicates
  - Prime Implicates
  - Indep. Vars
  - Implicant Ext.
  - Implicate Ext.
Selection of topics

Problem Solving with SAT

Embeddings
- PBO
- B&B Search
- Enumeration
- OPT SAT
- Lazy SMT
- LCG

Encodings
- Eager SMT
- MBD
- Planning
- BMC

Oracles
- MC: ic3
- Min. Models
- Backbones
- MCS
- MaxSAT
- MUS
- Enumeration
- Counting
- CEGAR QBF

MUS enumeration
MUS extraction
MaxSAT solving
Outline

Minimal Unsatisfiability

MUS Enumeration

Maximum Satisfiability
Analyzing inconsistency – timetabling

<table>
<thead>
<tr>
<th>Subject</th>
<th>Day</th>
<th>Time</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intro Prog</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>6.2.46</td>
</tr>
<tr>
<td>Intro AI</td>
<td>Tue</td>
<td>10:00-11:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>Databases</td>
<td>Tue</td>
<td>11:00-12:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>Linear Alg</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>6.2.46</td>
</tr>
<tr>
<td>Calculus</td>
<td>Tue</td>
<td>10:00-11:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>Adv Calculus</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>8.2.06</td>
</tr>
</tbody>
</table>

... (hundreds of consistent constraints)

- Set of constraints consistent / satisfiable?
Analyzing inconsistency – timetabling

<table>
<thead>
<tr>
<th>Subject</th>
<th>Day</th>
<th>Time</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intro Prog</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>6.2.46</td>
</tr>
<tr>
<td>Intro AI</td>
<td>Tue</td>
<td>10:00-11:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>Databases</td>
<td>Tue</td>
<td>11:00-12:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Linear Alg</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>6.2.46</td>
</tr>
<tr>
<td>Calculus</td>
<td>Tue</td>
<td>10:00-11:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>Adv Calculus</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>8.2.06</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

- Set of constraints consistent / satisfiable? No
Analyzing inconsistency – timetabling

<table>
<thead>
<tr>
<th>Subject</th>
<th>Day</th>
<th>Time</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intro Prog</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>6.2.46</td>
</tr>
<tr>
<td>Intro AI</td>
<td>Tue</td>
<td>10:00-11:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>Databases</td>
<td>Tue</td>
<td>11:00-12:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Linear Alg</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>6.2.46</td>
</tr>
<tr>
<td>Calculus</td>
<td>Tue</td>
<td>10:00-11:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>Adv Calculus</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>8.2.06</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- Set of constraints consistent / satisfiable? No
- Minimal subset of constraints that is inconsistent / unsatisfiable?
## Analyzing inconsistency – timetabling

<table>
<thead>
<tr>
<th>Subject</th>
<th>Day</th>
<th>Time</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intro Prog</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>6.2.46</td>
</tr>
<tr>
<td>Intro AI</td>
<td>Tue</td>
<td>10:00-11:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>Databases</td>
<td>Tue</td>
<td>11:00-12:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>Linear Alg</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>6.2.46</td>
</tr>
<tr>
<td>Calculus</td>
<td>Tue</td>
<td>10:00-11:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>Adv Calculus</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>8.2.06</td>
</tr>
</tbody>
</table>

... (hundreds of consistent constraints)

- Set of constraints consistent / satisfiable? **No**
- **Minimal subset** of constraints that is inconsistent / unsatisfiable?
Analyzing inconsistency – timetabling

<table>
<thead>
<tr>
<th>Subject</th>
<th>Day</th>
<th>Time</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intro Prog</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>6.2.46</td>
</tr>
<tr>
<td>Intro AI</td>
<td>Tue</td>
<td>10:00-11:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>Databases</td>
<td>Tue</td>
<td>11:00-12:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td></td>
<td></td>
<td>... (hundreds of consistent constraints)</td>
<td></td>
</tr>
<tr>
<td>Linear Alg</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>6.2.46</td>
</tr>
<tr>
<td>Calculus</td>
<td>Tue</td>
<td>10:00-11:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>Adv Calculus</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>8.2.06</td>
</tr>
<tr>
<td></td>
<td></td>
<td>... (hundreds of consistent constraints)</td>
<td></td>
</tr>
</tbody>
</table>

- Set of constraints consistent / satisfiable? **No**
- **Minimal subset** of constraints that is inconsistent / unsatisfiable?
- **Minimal subset** of constraints whose removal makes remaining constraints consistent?
Analyzing inconsistency – timetabling

<table>
<thead>
<tr>
<th>Subject</th>
<th>Day</th>
<th>Time</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intro Prog</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>6.2.46</td>
</tr>
<tr>
<td>Intro AI</td>
<td>Tue</td>
<td>10:00-11:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>Databases</td>
<td>Tue</td>
<td>11:00-12:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>Linear Alg</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>6.2.46</td>
</tr>
<tr>
<td>Calculus</td>
<td>Tue</td>
<td>10:00-11:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>Adv Calculus</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>8.2.06</td>
</tr>
</tbody>
</table>

... (hundreds of consistent constraints)

- Set of constraints consistent / satisfiable? No
- Minimal subset of constraints that is inconsistent / unsatisfiable?
- Minimal subset of constraints whose removal makes remaining constraints consistent?
### Analyzing inconsistency – timetabling

<table>
<thead>
<tr>
<th>Subject</th>
<th>Day</th>
<th>Time</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intro Prog</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>6.2.46</td>
</tr>
<tr>
<td>Intro AI</td>
<td>Tue</td>
<td>10:00-11:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>Databases</td>
<td>Tue</td>
<td>11:00-12:00</td>
<td>8.2.37</td>
</tr>
</tbody>
</table>

... (hundreds of consistent constraints)

<table>
<thead>
<tr>
<th>Subject</th>
<th>Day</th>
<th>Time</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Alg</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>6.2.46</td>
</tr>
<tr>
<td>Calculus</td>
<td>Tue</td>
<td>10:00-11:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>Adv Calculus</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>8.2.06</td>
</tr>
</tbody>
</table>

... (hundreds of consistent constraints)

- Set of constraints consistent / satisfiable? **No**
- Minimal subset of constraints that is inconsistent / unsatisfiable?
- Minimal subset of constraints whose removal makes remaining constraints consistent?
- **How to compute these minimal sets?**
Analyzing inconsistency – timetabling

<table>
<thead>
<tr>
<th>Subject</th>
<th>Day</th>
<th>Time</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intro Prog</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>6.2.46</td>
</tr>
<tr>
<td>Intro AI</td>
<td>Tue</td>
<td>10:00-11:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>Databases</td>
<td>Tue</td>
<td>11:00-12:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>... (hundreds of consistent constraints)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linear Alg</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>6.2.46</td>
</tr>
<tr>
<td>Calculus</td>
<td>Tue</td>
<td>10:00-11:00</td>
<td>8.2.37</td>
</tr>
<tr>
<td>Adv Calculus</td>
<td>Mon</td>
<td>9:00-10:00</td>
<td>8.2.06</td>
</tr>
<tr>
<td>... (hundreds of consistent constraints)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Set of constraints consistent / satisfiable? **No**
- **Minimal subset** of constraints that is inconsistent / unsatisfiable?
- **Minimal subset** of constraints whose removal makes remaining constraints consistent?
- How to compute these **minimal** sets?
Unsatisfiable formulas – MUSes & MCSes

• Given $\mathcal{F} (\vDash \bot)$, $\mathcal{M} \subseteq \mathcal{F}$ is a Minimal Unsatisfiable Subset (MUS) iff $\mathcal{M} \vDash \bot$ and $\forall \mathcal{M}' \subset \mathcal{M}, \mathcal{M}' \not\vDash \bot$

$$\neg x_1 \lor \neg x_2 \land (x_1) \land (x_2) \land (\neg x_3 \lor \neg x_4) \land (x_3) \land (x_4) \land (x_5 \lor x_6)$$
Unsatisfiable formulas – MUSes & MCSes

- Given $\mathcal{F} (\vDash \bot)$, $\mathcal{M} \subseteq \mathcal{F}$ is a Minimal Unsatisfiable Subset (MUS) iff $\mathcal{M} \vDash \bot$ and $\forall \mathcal{M}' \subseteq \mathcal{M}, \mathcal{M}' \not\vDash \bot$.

$$\neg x_1 \lor \neg x_2 \land (x_1) \land (x_2) \land (\neg x_3 \lor \neg x_4) \land (x_3) \land (x_4) \land (x_5 \lor x_6)$$
Unsatisfiable formulas – MUSes & MCSes

- Given $\mathcal{F} (\models \bot)$, $\mathcal{M} \subseteq \mathcal{F}$ is a Minimal Unsatisfiable Subset (MUS) iff $\mathcal{M} \models \bot$ and $\forall \mathcal{M}' \subsetneq \mathcal{M}, \mathcal{M}' \not\models \bot$

  $$(\neg x_1 \lor \neg x_2) \land (x_1) \land (x_2) \land (\neg x_3 \lor \neg x_4) \land (x_3) \land (x_4) \land (x_5 \lor x_6)$$

- Given $\mathcal{F} (\models \bot)$, $\mathcal{C} \subseteq \mathcal{F}$ is a Minimal Correction Subset (MCS) iff $\mathcal{F} \setminus \mathcal{C} \not\models \bot$ and $\forall \mathcal{C}' \subsetneq \mathcal{C}, \mathcal{F} \setminus \mathcal{C}' \models \bot$. $\mathcal{S} = \mathcal{F} \setminus \mathcal{C}$ is MSS

  $$(\neg x_1 \lor \neg x_2) \land (x_1) \land (x_2) \land (\neg x_3 \lor \neg x_4) \land (x_3) \land (x_4) \land (x_5 \lor x_6)$$
• Given $\mathcal{F} (\models \bot)$, $\mathcal{M} \subseteq \mathcal{F}$ is a Minimal Unsatisfiable Subset (MUS) iff $\mathcal{M} \models \bot$ and $\forall \mathcal{M}' \subseteq \mathcal{M}, \mathcal{M}' \not\models \bot$

$$\neg x_1 \lor \neg x_2 \land (x_1) \land (x_2) \land (\neg x_3 \lor \neg x_4) \land (x_3) \land (x_4) \land (x_5 \lor x_6)$$

• Given $\mathcal{F} (\models \bot)$, $\mathcal{C} \subseteq \mathcal{F}$ is a Minimal Correction Subset (MCS) iff $\mathcal{F} \setminus \mathcal{C} \not\models \bot$ and $\forall \mathcal{C}' \subseteq \mathcal{C}, \mathcal{F} \setminus \mathcal{C}' \models \bot$. $\mathcal{S} = \mathcal{F} \setminus \mathcal{C}$ is MSS

$$\neg x_1 \lor \neg x_2 \land (x_1) \land (x_2) \land (\neg x_3 \lor \neg x_4) \land (x_3) \land (x_4) \land (x_5 \lor x_6)$$
• Given $\mathcal{F} (\models \bot)$, $\mathcal{M} \subseteq \mathcal{F}$ is a Minimal Unsatisfiable Subset (MUS) iff $\mathcal{M} \models \bot$ and $\forall \mathcal{M}' \subsetneq \mathcal{M}, \mathcal{M}' \not\models \bot$

\[(\neg x_1 \lor \neg x_2) \land (x_1) \land (x_2) \land (\neg x_3 \lor \neg x_4) \land (x_3) \land (x_4) \land (x_5 \lor x_6)\]

• Given $\mathcal{F} (\models \bot)$, $\mathcal{C} \subseteq \mathcal{F}$ is a Minimal Correction Subset (MCS) iff $\mathcal{F} \setminus \mathcal{C} \not\models \bot$ and $\forall \mathcal{C}' \subsetneq \mathcal{C}, \mathcal{F} \setminus \mathcal{C}' \models \bot$. $S = \mathcal{F} \setminus \mathcal{C}$ is MSS

\[(\neg x_1 \lor \neg x_2) \land (x_1) \land (x_2) \land (\neg x_3 \lor \neg x_4) \land (x_3) \land (x_4) \land (x_5 \lor x_6)\]

• MUSes and MCSes are (subset-)minimal sets

• MUSes and minimal hitting sets of MCSes and vice-versa

• Easy to see why
• Given $\mathcal{F} (\models \bot)$, $\mathcal{M} \subseteq \mathcal{F}$ is a Minimal Unsatisfiable Subset (MUS) iff $\mathcal{M} \models \bot$ and $\forall \mathcal{M}' \subseteq \mathcal{M}, \mathcal{M}' \not\models \bot$

$$(\neg x_1 \lor \neg x_2) \land (x_1) \land (x_2) \land (\neg x_3 \lor \neg x_4) \land (x_3) \land (x_4) \land (x_5 \lor x_6)$$

• Given $\mathcal{F} (\models \bot)$, $\mathcal{C} \subseteq \mathcal{F}$ is a Minimal Correction Subset (MCS) iff $\mathcal{F} \setminus \mathcal{C} \not\models \bot$ and $\forall \mathcal{C}' \subseteq \mathcal{C}, \mathcal{F} \setminus \mathcal{C}' \models \bot$. $S = \mathcal{F} \setminus \mathcal{C}$ is MSS

$$(\neg x_1 \lor \neg x_2) \land (x_1) \land (x_2) \land (\neg x_3 \lor \neg x_4) \land (x_3) \land (x_4) \land (x_5 \lor x_6)$$

• MUSes and MCSes are (subset-)minimal sets

• MUSes and minimal hitting sets of MCSes and vice-versa

  • Easy to see why

• How to compute MUSes & MCSes efficiently with SAT oracles?
Why it matters?

• Analysis of **over-constrained systems**
  • Model-based diagnosis
    • Software fault localization
    • Spreadsheet debugging
    • Debugging relational specifications (e.g. Alloy)
    • Type error debugging
    • Axiom pinpointing in description logics
    • ...
  • Model checking of software & hardware systems
  • Inconsistency measurement
  • Minimal models; MinCost SAT; ...
  • ...

• Find **minimal** relaxations to recover **consistency**
  • But also **minimum** relaxations to recover **consistency**, eg. **MaxSAT**

• **Find minimal explanations of inconsistency**
  • But also **minimum** explanations of **inconsistency**, eg. **Smallest MUS**
Why it matters?

• Analysis of over-constrained systems
  • Model-based diagnosis
    • Software fault localization
    • Spreadsheet debugging
    • Debugging relational specifications (e.g. Alloy)
    • Type error debugging
    • Axiom pinpointing in description logics
    • ...
  • Model checking of software & hardware systems
  • Inconsistency measurement
  • Minimal models; MinCost SAT; ... 
  • ...

• Find minimal relaxations to recover consistency
  • But also minimum relaxations to recover consistency, eg. MaxSAT

• Find minimal explanations of inconsistency
  • But also minimum explanations of inconsistency, eg. Smallest MUS

Enumeration required!
**Deletion-based algorithm**

**Input**: Set $\mathcal{F}$

**Output**: Minimal subset $\mathcal{M}$

begin

\[ M \leftarrow \mathcal{F} \]

\begin{algorithmic}
\State \textbf{foreach} $c \in M$ \textbf{do}
\State \quad \textbf{if } \neg \text{SAT}(M \setminus \{c\}) \text{ then}
\State \quad \quad \text{M} \leftarrow M \setminus \{c\} \quad \quad // \text{If } \neg \text{SAT}(M \setminus \{c\}), \text{ then } c \not\in \text{MUS}
\State \textbf{return } M \quad \quad \quad // \text{Final } M \text{ is MUS}
\end{algorithmic}

end

- Number of oracles calls: $\mathcal{O}(m)$

[CD91, BDTW93]
Deletion-based algorithm

**Input**: Set $\mathcal{F}$

**Output**: Minimal subset $\mathcal{M}$

begin

\[ \mathcal{M} \leftarrow \mathcal{F} \]

\[ \text{foreach } c \in \mathcal{M} \text{ do} \]

\[ \quad \text{if } \neg \text{SAT}(\mathcal{M} \setminus \{c\}) \text{ then} \]

\[ \quad \quad \mathcal{M} \leftarrow \mathcal{M} \setminus \{c\} \]

\[ \text{return } \mathcal{M} \]

end

- Number of oracles calls: $\mathcal{O}(m)$

Monotonicity implicit & essential!

\[ [\text{CD91, BDTW93}] \]
### Deletion – MUS 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>$(\neg x_1 \lor \neg x_2)$</td>
<td>$(x_1)$</td>
<td>$(x_2)$</td>
<td>$(\neg x_3 \lor \neg x_4)$</td>
<td>$(x_3)$</td>
<td>$(x_4)$</td>
<td>$(x_5 \lor x_6)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$\mathcal{M}$</th>
<th>$\mathcal{M} \setminus {c}$</th>
<th>$\neg \text{SAT}(\mathcal{M} \setminus {c})$</th>
<th>Outcome</th>
</tr>
</thead>
</table>
### Deletion – MUS 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>$\neg x_1 \lor \neg x_2$</td>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$\neg x_3 \lor \neg x_4$</td>
<td>$x_3$</td>
<td>$x_4$</td>
<td>$x_5 \lor x_6$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$\mathcal{M}$</th>
<th>$\mathcal{M} \setminus {c}$</th>
<th>$\neg \text{SAT}(\mathcal{M} \setminus {c})$</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$c_1 \ldots c_7$</td>
<td>$c_2 \ldots c_7$</td>
<td>$1$</td>
<td>Drop $c_1$</td>
</tr>
</tbody>
</table>
Deletion – MUS 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>((\neg x_1 \lor \neg x_2))</td>
<td>((x_1))</td>
<td>((x_2))</td>
<td>((\neg x_3 \lor \neg x_4))</td>
<td>((x_3))</td>
<td>((x_4))</td>
<td>((x_5 \lor x_6))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(\mathcal{M})</th>
<th>(\mathcal{M} \setminus {c})</th>
<th>(-\text{SAT}(\mathcal{M} \setminus {c}))</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c_1\ldots c_7)</td>
<td>(c_2\ldots c_7)</td>
<td>1</td>
<td>Drop (c_1)</td>
<td></td>
</tr>
<tr>
<td>(c_2\ldots c_7)</td>
<td>(c_3\ldots c_7)</td>
<td>1</td>
<td>Drop (c_2)</td>
<td></td>
</tr>
</tbody>
</table>
# Deletion – MUS example

<table>
<thead>
<tr>
<th>C₁</th>
<th>C₂</th>
<th>C₃</th>
<th>C₄</th>
<th>C₅</th>
<th>C₆</th>
<th>C₇</th>
</tr>
</thead>
<tbody>
<tr>
<td>(¬x₁ ∨ ¬x₂)</td>
<td>x₁</td>
<td>x₂</td>
<td>(¬x₃ ∨ ¬x₄)</td>
<td>x₃</td>
<td>x₄</td>
<td>(x₅ ∨ x₆)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M</th>
<th>M \ {c}</th>
<th>¬SAT(M \ {c})</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>c₁..c₇</td>
<td>c₂..c₇</td>
<td>1</td>
<td>Drop c₁</td>
</tr>
<tr>
<td>c₂..c₇</td>
<td>c₃..c₇</td>
<td>1</td>
<td>Drop c₂</td>
</tr>
<tr>
<td>c₃..c₇</td>
<td>c₄..c₇</td>
<td>1</td>
<td>Drop c₃</td>
</tr>
</tbody>
</table>
Deletion – MUS example

\[
\begin{array}{cccccccc}
\text{\(c_1\)} & \text{\(c_2\)} & \text{\(c_3\)} & \text{\(c_4\)} & \text{\(c_5\)} & \text{\(c_6\)} & \text{\(c_7\)} \\
(\neg x_1 \lor \neg x_2) & (x_1) & (x_2) & (\neg x_3 \lor \neg x_4) & (x_3) & (x_4) & (x_5 \lor x_6)
\end{array}
\]

<table>
<thead>
<tr>
<th>(\mathcal{M})</th>
<th>(\mathcal{M} \setminus {c})</th>
<th>(-\text{SAT}(\mathcal{M} \setminus {c}))</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<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..c_7)</td>
<td>(c_3..c_7)</td>
<td>1</td>
<td>Drop (c_2)</td>
</tr>
<tr>
<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..c_7)</td>
<td>(c_5..c_7)</td>
<td>0</td>
<td>Keep (c_4)</td>
</tr>
</tbody>
</table>
### Deletion – MUS example

The table below shows the truth values of the formulas $c_1 \ldots c_7$ and the outcomes of dropping them.

<table>
<thead>
<tr>
<th>M</th>
<th>M \ {c}</th>
<th>¬SAT(M \ {c})</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1 \ldots c_7$</td>
<td>$c_2 \ldots c_7$</td>
<td>1</td>
<td>Drop $c_1$</td>
</tr>
<tr>
<td>$c_2 \ldots c_7$</td>
<td>$c_3 \ldots c_7$</td>
<td>1</td>
<td>Drop $c_2$</td>
</tr>
<tr>
<td>$c_3 \ldots c_7$</td>
<td>$c_4 \ldots c_7$</td>
<td>1</td>
<td>Drop $c_3$</td>
</tr>
<tr>
<td>$c_4 \ldots c_7$</td>
<td>$c_5 \ldots c_7$</td>
<td>0</td>
<td>Keep $c_4$</td>
</tr>
<tr>
<td>$c_4 \ldots c_7$</td>
<td>$c_4 c_6 c_7$</td>
<td>0</td>
<td>Keep $c_5$</td>
</tr>
</tbody>
</table>

The formulas are:

- $c_1$: $(\neg x_1 \lor \neg x_2)$
- $c_2$: $(x_1)$
- $c_3$: $(x_2)$
- $c_4$: $(\neg x_3 \lor \neg x_4)$
- $c_5$: $(x_3)$
- $c_6$: $(x_4)$
- $c_7$: $(x_5 \lor x_6)$
### 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>$(\neg x_1 \lor \neg x_2)$</td>
<td>$(x_1)$</td>
<td>$(x_2)$</td>
<td>$(\neg x_3 \lor \neg x_4)$</td>
<td>$(x_3)$</td>
<td>$(x_4)$</td>
<td>$(x_5 \lor x_6)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\mathcal{M}$</th>
<th>$\mathcal{M} \setminus {c}$</th>
<th>$\neg \text{SAT}(\mathcal{M} \setminus {c})$</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1..c_7$</td>
<td>$c_2..c_7$</td>
<td>1</td>
<td>Drop $c_1$</td>
</tr>
<tr>
<td>$c_2..c_7$</td>
<td>$c_3..c_7$</td>
<td>1</td>
<td>Drop $c_2$</td>
</tr>
<tr>
<td>$c_3..c_7$</td>
<td>$c_4..c_7$</td>
<td>1</td>
<td>Drop $c_3$</td>
</tr>
<tr>
<td>$c_4..c_7$</td>
<td>$c_5..c_7$</td>
<td>0</td>
<td>Keep $c_4$</td>
</tr>
<tr>
<td>$c_4..c_7$</td>
<td>$c_4c_6c_7$</td>
<td>0</td>
<td>Keep $c_5$</td>
</tr>
<tr>
<td>$c_4..c_7$</td>
<td>$c_4c_5c_7$</td>
<td>0</td>
<td>Keep $c_6$</td>
</tr>
</tbody>
</table>
### 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>$(\neg x_1 \lor \neg x_2)$</td>
<td>$(x_1)$</td>
<td>$(x_2)$</td>
<td>$(\neg x_3 \lor \neg x_4)$</td>
<td>$(x_3)$</td>
<td>$(x_4)$</td>
<td>$(x_5 \lor x_6)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\mathcal{M}$</th>
<th>$\mathcal{M} \setminus {c}$</th>
<th>$\neg \text{SAT}(\mathcal{M} \setminus {c})$</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1..c_7$</td>
<td>$c_2..c_7$</td>
<td>1</td>
<td>Drop $c_1$</td>
</tr>
<tr>
<td>$c_2..c_7$</td>
<td>$c_3..c_7$</td>
<td>1</td>
<td>Drop $c_2$</td>
</tr>
<tr>
<td>$c_3..c_7$</td>
<td>$c_4..c_7$</td>
<td>1</td>
<td>Drop $c_3$</td>
</tr>
<tr>
<td>$c_4..c_7$</td>
<td>$c_5..c_7$</td>
<td>0</td>
<td>Keep $c_4$</td>
</tr>
<tr>
<td>$c_4..c_7$</td>
<td>$c_4c_6c_7$</td>
<td>0</td>
<td>Keep $c_5$</td>
</tr>
<tr>
<td>$c_4..c_7$</td>
<td>$c_4c_5c_7$</td>
<td>0</td>
<td>Keep $c_6$</td>
</tr>
<tr>
<td>$c_4..c_7$</td>
<td>$c_4..c_6$</td>
<td>1</td>
<td>Drop $c_7$</td>
</tr>
</tbody>
</table>
## Deletion – MUS 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>$(\neg x_1 \lor \neg x_2)$</td>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$(\neg x_3 \lor \neg x_4)$</td>
<td>$x_3$</td>
<td>$x_4$</td>
<td>$(x_5 \lor x_6)$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\mathcal{M}$</th>
<th>$\mathcal{M} \setminus {c}$</th>
<th>$\neg \text{SAT}(\mathcal{M} \setminus {c})$</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_1..c_7$</td>
<td>$c_2..c_7$</td>
<td>1</td>
<td>Drop $c_1$</td>
</tr>
<tr>
<td>$c_2..c_7$</td>
<td>$c_3..c_7$</td>
<td>1</td>
<td>Drop $c_2$</td>
</tr>
<tr>
<td>$c_3..c_7$</td>
<td>$c_4..c_7$</td>
<td>1</td>
<td>Drop $c_3$</td>
</tr>
<tr>
<td>$c_4..c_7$</td>
<td>$c_5..c_7$</td>
<td>0</td>
<td>Keep $c_4$</td>
</tr>
<tr>
<td>$c_4..c_7$</td>
<td>$c_4c_6c_7$</td>
<td>0</td>
<td>Keep $c_5$</td>
</tr>
<tr>
<td>$c_4..c_7$</td>
<td>$c_4c_5c_7$</td>
<td>0</td>
<td>Keep $c_6$</td>
</tr>
<tr>
<td>$c_4..c_7$</td>
<td>$c_4c_6$</td>
<td>1</td>
<td>Drop $c_7$</td>
</tr>
</tbody>
</table>

- **MUS:** $\{c_4, c_5, c_6\}$
Many MUS algorithms

- Formula $\mathcal{F}$ with $m$ clauses $k$ the size of largest minimal subset

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Oracle Calls</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insertion-based</td>
<td>$O(km)$</td>
<td>[dSNP88, vMW08]</td>
</tr>
<tr>
<td>MCS_MUS</td>
<td>$O(km)$</td>
<td>[BK15]</td>
</tr>
<tr>
<td>Deletion-based</td>
<td>$O(m)$</td>
<td>[CD91, BDTW93]</td>
</tr>
<tr>
<td>Linear insertion</td>
<td>$O(m)$</td>
<td>[MSL11, BLM12]</td>
</tr>
<tr>
<td>Dichotomic</td>
<td>$O(k \log(m))$</td>
<td>[HLSB06]</td>
</tr>
<tr>
<td>QuickXplain</td>
<td>$O(k + k \log(\frac{m}{k}))$</td>
<td>[Jun04]</td>
</tr>
<tr>
<td>Progression</td>
<td>$O(k \log(1 + \frac{m}{k}))$</td>
<td>[MJB13]</td>
</tr>
</tbody>
</table>

- **Note:** Lower bound in $\text{FP}^\text{NP}_{||}$ and upper bound in $\text{FP}^\text{NP}$ [CT95]
- Oracle calls correspond to testing unsatisfiability with SAT solver
- Practical optimizations: clause set trimming; clause set refinement; redundancy removal; (recursive) model rotation
Outline

Minimal Unsatisfiability

MUS Enumeration

Maximum Satisfiability
How to enumerate MUSes?

1. Standard solution:
   - Exploit HS duality between MCSes and MUSes:
     - MCSes are MHSes of MUSes and vice-versa
     - Enumerate all MCSes and then enumerate all MHSes of the MCSes, i.e. compute all the MUSes
     - Problematic if too many MCSes, and we want the MUSes
     - And, often we want to enumerate the MUSes

2. Exploit recent advances in 2QBF

3. Implicit hitting set dualization:
   - Most effective if MUSes provided to user on-demand
1. Standard solution:

   Exploit HS duality between MCSes and MUSes

   **MCSes are MHSes of MUSes and vice-versa**

   - Enumerate *all* MCSes and then enumerate *all* MHSes of the MCSes, i.e. compute *all* the MUSes
   - Problematic if *too* many MCSes, and we want the MUSes
   - And, often *we want to enumerate the MUSes*
How to enumerate MUSes?

1. Standard solution:
   **Exploit HS duality between MCSes and MUSes**

   - MCSes are MHSes of MUSes and vice-versa
   - Enumerate *all* MCSes and then enumerate *all* MHSes of the MCSes, i.e. compute *all* the MUSes
   - Problematic if *too* many MCSes, and we want the MUSes
   - And, often *we want to enumerate the MUSes*

2. Exploit recent advances in 2QBF solving
How to enumerate MUSes?

1. Standard solution:
   **Exploit HS duality between MCSes and MUSes**

   MCSes are MHSes of MUSes and vice-versa

   - Enumerate **all** MCSes and then enumerate **all** MHSes of the MCSes, i.e. compute all the MUSes
   - Problematic if **too** many MCSes, and we want the MUSes
   - And, often we want to enumerate the MUSes

2. Exploit recent advances in **2QBF** solving

3. Implicit hitting set dualization

   - Most effective if MUSes provided to user on-demand
How to enumerate MUSes, preferably?

1. Keep sets representing computed **MUSes** (set $\mathcal{N}$) and **MCSes** (set $\mathcal{P}$)
How to enumerate MUSes, preferably?

1. Keep sets representing computed MUSes (set \( \mathcal{N} \)) and MCSes (set \( \mathcal{P} \))
2. Compute \textbf{minimal hitting set (MHS) } \( H \) of \( \mathcal{N} \), subject to \( \mathcal{P} \)
   - \textbf{Must not} repeat MUSes
   - \textbf{Must not} repeat MCSes
   - Maximize clauses picked, i.e. prefer to check satisfiability on as \textbf{many} clauses as possible
   - If unsatisfiable: \textbf{no more MUSes/MCSes to enumerate}
How to enumerate MUSes, preferably?

1. Keep sets representing computed \textit{MUSes} (set $N$) and \textit{MCSes} (set $P$)
2. Compute \textit{minimal hitting set} (MHS) $H$ of $N$, subject to $P$
   - \textbf{Must not} repeat MUSes
   - \textbf{Must not} repeat MCSes
   - Maximize clauses picked, i.e. prefer to check satisfiability on as \textbf{many} clauses as possible
   - If unsatisfiable: \textbf{no more MUSes/MCSes to enumerate}
3. Target set: $F'$, i.e. $F$ minus clauses from $H$
How to enumerate MUSes, preferably?

1. Keep sets representing computed MUSes (set $\mathcal{N}$) and MCSes (set $\mathcal{P}$)
2. Compute minimal hitting set (MHS) $H$ of $\mathcal{N}$, subject to $\mathcal{P}$
   - Must not repeat MUSes
   - Must not repeat MCSes
   - Maximize clauses picked, i.e. prefer to check satisfiability on as many clauses as possible
   - If unsatisfiable: no more MUSes/MCSes to enumerate
3. Target set: $\mathcal{F}'$, i.e. $\mathcal{F}$ minus clauses from $H$
4. Run SAT oracle on $\mathcal{F}'$
   - If $\mathcal{F}'$ unsatisfiable: extract new MUS
   - Otherwise, $H$ is already an MCS of $\mathcal{F}$
How to enumerate MUSes, preferably?

1. Keep sets representing computed MUSes (set $\mathcal{N}$) and MCSes (set $\mathcal{P}$)
2. Compute minimal hitting set (MHS) $H$ of $\mathcal{N}$, subject to $\mathcal{P}$
   - Must not repeat MUSes
   - Must not repeat MCSes
   - Maximize clauses picked, i.e. prefer to check satisfiability on as many clauses as possible
   - If unsatisfiable: no more MUSes/MCSes to enumerate
3. Target set: $\mathcal{F}'$, i.e. $\mathcal{F}$ minus clauses from $H$
4. Run SAT oracle on $\mathcal{F}'$
   - If $\mathcal{F}'$ unsatisfiable: extract new MUS
   - Otherwise, $H$ is already an MCS of $\mathcal{F}$
5. Repeat loop
**Input:** CNF formula $\mathcal{F}$

```latex
\begin{align*}
\text{begin} & \quad \text{while true do} \\
I & \quad \{p_i \mid c_i \in \mathcal{F}\} \\
(\mathcal{P}, \mathcal{N}) & \quad (\emptyset, \emptyset) \\
\text{while true do} & \quad \text{if not } st \text{ then return} \\
(\mathcal{N}, \mathcal{P}) & \quad \text{MinHittingSet}(\mathcal{N}, \mathcal{P}) \\
\mathcal{F}' & \quad \{c_i \mid p_i \in I \land p_i \notin H\} \\
\text{if not SAT}(\mathcal{F}') & \quad \text{then} \\
\mathcal{M} & \quad \text{ComputeMUS}(\mathcal{F}') \\
\text{ReportMUS } (\mathcal{M}) \\
\mathcal{N} & \quad \mathcal{N} \cup \{\neg p_i \mid c_i \in \mathcal{M}\} \\
\text{else} & \quad \text{end} \\
\mathcal{P} & \quad \mathcal{P} \cup \{p_i \mid p_i \in H\} \\
\text{end}
\end{align*}
```
An example

<table>
<thead>
<tr>
<th>MinHS ((\forall))</th>
<th>(\mathcal{F}')</th>
<th>MUS/MCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>(p_1 p_2 p_3 p_4 p_5 p_6 p_7)</td>
<td>S/U</td>
<td></td>
</tr>
<tr>
<td>1111111</td>
<td>U</td>
<td>(\neg p_1 \lor \neg p_2 \lor \neg p_3)</td>
</tr>
<tr>
<td>0111111</td>
<td>U</td>
<td>(\neg p_6 \lor \neg p_7)</td>
</tr>
<tr>
<td>0111101</td>
<td>S</td>
<td>(p_1 \lor p_6)</td>
</tr>
<tr>
<td>1011101</td>
<td>U</td>
<td>(\neg p_1 \lor \neg p_4 \lor \neg p_5)</td>
</tr>
<tr>
<td>1101010</td>
<td>S</td>
<td>(p_3 \lor p_5 \lor p_7)</td>
</tr>
<tr>
<td>1010110</td>
<td>S</td>
<td>(p_2 \lor p_4 \lor p_7)</td>
</tr>
<tr>
<td>1100101</td>
<td>S</td>
<td>(p_3 \lor p_4 \lor p_6)</td>
</tr>
<tr>
<td>0111110</td>
<td>S</td>
<td>(p_1 \lor p_7)</td>
</tr>
<tr>
<td>1101001</td>
<td>S</td>
<td>(p_3 \lor p_5 \lor p_6)</td>
</tr>
<tr>
<td>1010101</td>
<td>S</td>
<td>(p_2 \lor p_4 \lor p_6)</td>
</tr>
<tr>
<td>1011001</td>
<td>S</td>
<td>(p_2 \lor p_5 \lor p_6)</td>
</tr>
<tr>
<td>1100110</td>
<td>S</td>
<td>(p_3 \lor p_4 \lor p_7)</td>
</tr>
<tr>
<td>1011010</td>
<td>S</td>
<td>(p_2 \lor p_5 \lor p_7)</td>
</tr>
</tbody>
</table>

\(c_1 = p\) \hspace{1cm} c_4 = \neg p \lor q \hspace{1cm} c_5 = \neg q\)

\(c_2 = \neg p \lor r\) \hspace{1cm} c_3 = \neg r\)

\(c_6 = s\) \hspace{1cm} c_7 = \neg s\)
### An example

<table>
<thead>
<tr>
<th>MinHS ((\bigvee))</th>
<th>(\mathcal{F}')</th>
<th>MUS/MCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>(p_1p_2p_3p_4p_5p_6p_7)</td>
<td>S/U</td>
<td>-</td>
</tr>
<tr>
<td>1111111</td>
<td>U</td>
<td>(-p_1 \lor -p_2 \lor -p_3)</td>
</tr>
<tr>
<td>0111111</td>
<td>U</td>
<td>(-p_6 \lor -p_7)</td>
</tr>
<tr>
<td>0111101</td>
<td>S</td>
<td>(p_1 \lor p_6)</td>
</tr>
<tr>
<td>1011101</td>
<td>U</td>
<td>(-p_1 \lor -p_4 \lor -p_5)</td>
</tr>
<tr>
<td>1101010</td>
<td>S</td>
<td>(p_3 \lor p_5 \lor p_7)</td>
</tr>
<tr>
<td>1010110</td>
<td>S</td>
<td>(p_2 \lor p_4 \lor p_7)</td>
</tr>
<tr>
<td>1100101</td>
<td>S</td>
<td>(p_3 \lor p_4 \lor p_6)</td>
</tr>
<tr>
<td>0111110</td>
<td>S</td>
<td>(p_1 \lor p_7)</td>
</tr>
<tr>
<td>1101001</td>
<td>S</td>
<td>(p_3 \lor p_5 \lor p_6)</td>
</tr>
<tr>
<td>1010101</td>
<td>S</td>
<td>(p_2 \lor p_4 \lor p_6)</td>
</tr>
<tr>
<td>1011001</td>
<td>S</td>
<td>(p_2 \lor p_5 \lor p_6)</td>
</tr>
<tr>
<td>1100110</td>
<td>S</td>
<td>(p_3 \lor p_4 \lor p_7)</td>
</tr>
<tr>
<td>1011010</td>
<td>S</td>
<td>(p_2 \lor p_5 \lor p_7)</td>
</tr>
</tbody>
</table>

\[c_1 = p\]
\[c_2 = \neg p \lor r\]
\[c_3 = \neg r\]
\[c_4 = \neg p \lor q\]
\[c_5 = \neg q\]
\[c_6 = s\]
\[c_7 = \neg s\]
An example

<table>
<thead>
<tr>
<th>MinHS ((\land))</th>
<th>(\mathcal{F}')</th>
<th>MUS/MCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>(p_1p_2p_3p_4p_5p_6p_7)</td>
<td>(S/U)</td>
<td></td>
</tr>
<tr>
<td>1111111</td>
<td>U</td>
<td>(\neg p_1 \lor \neg p_2 \lor \neg p_3)</td>
</tr>
<tr>
<td>0111111</td>
<td>U</td>
<td>(\neg p_6 \lor \neg p_7)</td>
</tr>
<tr>
<td>0111101</td>
<td>S</td>
<td>(p_1 \lor p_6)</td>
</tr>
<tr>
<td>1011101</td>
<td>U</td>
<td>(\neg p_1 \lor \neg p_4 \lor \neg p_5)</td>
</tr>
<tr>
<td>1101010</td>
<td>S</td>
<td>(p_3 \lor p_5 \lor p_7)</td>
</tr>
<tr>
<td>1010110</td>
<td>S</td>
<td>(p_2 \lor p_4 \lor p_7)</td>
</tr>
<tr>
<td>1100101</td>
<td>S</td>
<td>(p_3 \lor p_4 \lor p_6)</td>
</tr>
<tr>
<td>0111110</td>
<td>S</td>
<td>(p_1 \lor p_7)</td>
</tr>
<tr>
<td>1101001</td>
<td>S</td>
<td>(p_3 \lor p_5 \lor p_6)</td>
</tr>
<tr>
<td>1010101</td>
<td>S</td>
<td>(p_2 \lor p_4 \lor p_6)</td>
</tr>
<tr>
<td>1011001</td>
<td>S</td>
<td>(p_2 \lor p_5 \lor p_6)</td>
</tr>
<tr>
<td>1100110</td>
<td>S</td>
<td>(p_3 \lor p_4 \lor p_7)</td>
</tr>
<tr>
<td>1011010</td>
<td>S</td>
<td>(p_2 \lor p_5 \lor p_7)</td>
</tr>
</tbody>
</table>

\(c_1 = p\)
\(c_2 = \neg p \lor r\)
\(c_3 = \neg r\)
\(c_4 = \neg p \lor q\)
\(c_5 = \neg q\)
\(c_6 = s\)
\(c_7 = \neg s\)
An example

<table>
<thead>
<tr>
<th>MinHS (∨)</th>
<th>$\mathcal{F}'$</th>
<th>MUS/MCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1p_2p_3p_4p_5p_6p_7$</td>
<td>S/U</td>
<td>$\neg p_1 \lor \neg p_2 \lor \neg p_3$</td>
</tr>
<tr>
<td>1111111</td>
<td>U</td>
<td>$\neg p_1 \lor \neg p_2 \lor \neg p_3$</td>
</tr>
<tr>
<td>0111111</td>
<td>U</td>
<td>$\neg p_6 \lor \neg p_7$</td>
</tr>
<tr>
<td>0111101</td>
<td>S</td>
<td>$p_1 \lor p_6$</td>
</tr>
<tr>
<td>1011101</td>
<td>U</td>
<td>$\neg p_1 \lor \neg p_4 \lor \neg p_5$</td>
</tr>
<tr>
<td>1101010</td>
<td>S</td>
<td>$p_3 \lor p_5 \lor p_7$</td>
</tr>
<tr>
<td>1010110</td>
<td>S</td>
<td>$p_2 \lor p_4 \lor p_7$</td>
</tr>
<tr>
<td>1100101</td>
<td>S</td>
<td>$p_3 \lor p_4 \lor p_6$</td>
</tr>
<tr>
<td>0111110</td>
<td>S</td>
<td>$p_1 \lor p_7$</td>
</tr>
<tr>
<td>1101001</td>
<td>S</td>
<td>$p_3 \lor p_5 \lor p_6$</td>
</tr>
<tr>
<td>1010101</td>
<td>S</td>
<td>$p_2 \lor p_4 \lor p_6$</td>
</tr>
<tr>
<td>1011001</td>
<td>S</td>
<td>$p_2 \lor p_5 \lor p_6$</td>
</tr>
<tr>
<td>1100110</td>
<td>S</td>
<td>$p_3 \lor p_4 \lor p_7$</td>
</tr>
<tr>
<td>1011010</td>
<td>S</td>
<td>$p_2 \lor p_5 \lor p_7$</td>
</tr>
</tbody>
</table>

$c_1 = p$ \hspace{1cm} $c_4 = \neg p \lor q$ \hspace{1cm} $c_5 = \neg q$

$c_2 = \neg p \lor r$ \hspace{1cm} $c_3 = \neg r$ \hspace{1cm} $c_6 = s$ \hspace{1cm} $c_7 = \neg s$
### An example

<table>
<thead>
<tr>
<th>$\text{MinHS} (\land)$</th>
<th>$\mathcal{F}'$</th>
<th>$\text{MUS/MCS}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1 p_2 p_3 p_4 p_5 p_6 p_7$</td>
<td>S/U</td>
<td></td>
</tr>
<tr>
<td>1111111</td>
<td>U</td>
<td>$\lnot p_1 \lor \lnot p_2 \lor \lnot p_3$</td>
</tr>
<tr>
<td>0111111</td>
<td>U</td>
<td>$\lnot p_6 \lor \lnot p_7$</td>
</tr>
<tr>
<td>0111101</td>
<td>S</td>
<td>$p_1 \lor p_6$</td>
</tr>
<tr>
<td>1011101</td>
<td>U</td>
<td>$\lnot p_1 \lor \lnot p_4 \lor \lnot p_5$</td>
</tr>
<tr>
<td>1101010</td>
<td>S</td>
<td>$p_3 \lor p_5 \lor p_7$</td>
</tr>
<tr>
<td>1010110</td>
<td>S</td>
<td>$p_2 \lor p_4 \lor p_7$</td>
</tr>
<tr>
<td>1100101</td>
<td>S</td>
<td>$p_3 \lor p_4 \lor p_6$</td>
</tr>
<tr>
<td>0111110</td>
<td>S</td>
<td>$p_1 \lor p_7$</td>
</tr>
<tr>
<td>1101001</td>
<td>S</td>
<td>$p_3 \lor p_5 \lor p_6$</td>
</tr>
<tr>
<td>1010101</td>
<td>S</td>
<td>$p_2 \lor p_4 \lor p_6$</td>
</tr>
<tr>
<td>1011001</td>
<td>S</td>
<td>$p_2 \lor p_5 \lor p_6$</td>
</tr>
<tr>
<td>1100110</td>
<td>S</td>
<td>$p_3 \lor p_4 \lor p_7$</td>
</tr>
<tr>
<td>1011010</td>
<td>S</td>
<td>$p_2 \lor p_5 \lor p_7$</td>
</tr>
</tbody>
</table>

$c_1 = p \quad c_4 = \lnot p \lor q \quad c_5 = \lnot q$

$c_2 = \lnot p \lor r$

$c_3 = \lnot r$

$c_6 = s \quad c_7 = \lnot s$
An example

<table>
<thead>
<tr>
<th>MinHS ($\forall$)</th>
<th>$\mathcal{F}'$</th>
<th>MUS/MCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1 p_2 p_3 p_4 p_5 p_6 p_7$</td>
<td>$S/U$</td>
<td></td>
</tr>
<tr>
<td>1111111</td>
<td>U</td>
<td>$\neg p_1 \lor \neg p_2 \lor \neg p_3$</td>
</tr>
<tr>
<td>0111111</td>
<td>U</td>
<td>$\neg p_6 \lor \neg p_7$</td>
</tr>
<tr>
<td>0111101</td>
<td>S</td>
<td>$p_1 \lor p_6$</td>
</tr>
<tr>
<td>1011101</td>
<td>U</td>
<td>$\neg p_1 \lor \neg p_4 \lor \neg p_5$</td>
</tr>
<tr>
<td>1101010</td>
<td>S</td>
<td>$p_3 \lor p_5 \lor p_7$</td>
</tr>
<tr>
<td>1010110</td>
<td>S</td>
<td>$p_2 \lor p_4 \lor p_7$</td>
</tr>
<tr>
<td>1100101</td>
<td>S</td>
<td>$p_3 \lor p_4 \lor p_6$</td>
</tr>
<tr>
<td>0111110</td>
<td>S</td>
<td>$p_1 \lor p_7$</td>
</tr>
<tr>
<td>1101001</td>
<td>S</td>
<td>$p_3 \lor p_5 \lor p_6$</td>
</tr>
<tr>
<td>1010101</td>
<td>S</td>
<td>$p_2 \lor p_4 \lor p_6$</td>
</tr>
<tr>
<td>1011001</td>
<td>S</td>
<td>$p_2 \lor p_5 \lor p_6$</td>
</tr>
<tr>
<td>1100110</td>
<td>S</td>
<td>$p_3 \lor p_4 \lor p_7$</td>
</tr>
<tr>
<td>1011010</td>
<td>S</td>
<td>$p_2 \lor p_5 \lor p_7$</td>
</tr>
</tbody>
</table>

$c_1 = p$
$c_2 = \neg p \lor r$
$c_3 = \neg r$
$c_4 = \neg p \lor q$
$c_5 = \neg q$
$c_6 = s$
$c_7 = \neg s$
An example

<table>
<thead>
<tr>
<th>MinHS ($\lor$)</th>
<th>$\mathcal{F}'$</th>
<th>MUS/MCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1p_2p_3p_4p_5p_6p_7$</td>
<td>S/U</td>
<td></td>
</tr>
<tr>
<td>1111111</td>
<td>U</td>
<td>$\neg p_1 \lor \neg p_2 \lor \neg p_3$</td>
</tr>
<tr>
<td>0111111</td>
<td>U</td>
<td>$\neg p_6 \lor \neg p_7$</td>
</tr>
<tr>
<td>0111101</td>
<td>S</td>
<td>$p_1 \lor p_6$</td>
</tr>
<tr>
<td>1011101</td>
<td>U</td>
<td>$\neg p_1 \lor \neg p_4 \lor \neg p_5$</td>
</tr>
<tr>
<td>1101010</td>
<td>S</td>
<td>$p_3 \lor p_5 \lor p_7$</td>
</tr>
<tr>
<td>1010110</td>
<td>S</td>
<td>$p_2 \lor p_4 \lor p_7$</td>
</tr>
<tr>
<td>1100101</td>
<td>S</td>
<td>$p_3 \lor p_4 \lor p_6$</td>
</tr>
<tr>
<td>0111110</td>
<td>S</td>
<td>$p_1 \lor p_7$</td>
</tr>
<tr>
<td>1101001</td>
<td>S</td>
<td>$p_3 \lor p_5 \lor p_6$</td>
</tr>
<tr>
<td>1010101</td>
<td>S</td>
<td>$p_2 \lor p_4 \lor p_6$</td>
</tr>
<tr>
<td>1011001</td>
<td>S</td>
<td>$p_2 \lor p_5 \lor p_6$</td>
</tr>
<tr>
<td>1100110</td>
<td>S</td>
<td>$p_3 \lor p_4 \lor p_7$</td>
</tr>
<tr>
<td>1011010</td>
<td>S</td>
<td>$p_2 \lor p_5 \lor p_7$</td>
</tr>
</tbody>
</table>

$c_1 = p$
$c_2 = \neg p \lor r$
$c_3 = \neg r$
$c_4 = \neg p \lor q$
$c_5 = \neg q$

$c_6 = S$
$c_7 = \neg S$
Outline

Minimal Unsatisfiability

MUS Enumeration

Maximum Satisfiability
• Given **unsatisfiable** formula, find **largest** subset of clauses that is satisfiable
Recap MaxSAT

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

- 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
Recap MaxSAT

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.

- Note: Clauses can have weights & there can be hard clauses.
Recap MaxSAT

- 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 cost** MCSes.
  - **Note:** Clauses can have weights & there can be hard clauses.
Recap MaxSAT

- 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 cost MCSes
  - **Note**: Clauses can have weights & there can be hard clauses
- Many practical applications

\[ \begin{align*}
x_6 \lor x_2 & \quad \lnot x_6 \lor x_2 & \quad \lnot x_2 \lor x_1 & \quad \lnot x_1 \\
\lnot x_6 \lor x_8 & \quad x_6 \lor \lnot x_8 & \quad x_2 \lor x_4 & \quad \lnot x_4 \lor x_5 \\
x_7 \lor x_5 & \quad \lnot x_7 \lor x_5 & \quad \lnot x_5 \lor x_3 & \quad \lnot x_3
d\end{align*} \]
<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>Weights?</td>
<td>Hard Clauses?</td>
</tr>
<tr>
<td>---------</td>
<td>--------------</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Plain</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Partial</td>
</tr>
<tr>
<td></td>
<td>Weighted</td>
</tr>
<tr>
<td></td>
<td>Weighted Partial</td>
</tr>
</tbody>
</table>
## MaxSAT problem(s)

<table>
<thead>
<tr>
<th>Weights?</th>
<th>Hard Clauses?</th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>No</td>
<td>Plain</td>
<td>Partial</td>
</tr>
<tr>
<td>Yes</td>
<td></td>
<td>Weighted</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)
### MaxSAT Problem(s)

<table>
<thead>
<tr>
<th>Weights?</th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Plain</td>
<td>Partial</td>
</tr>
<tr>
<td>Yes</td>
<td>Weighted</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!
Issues with MaxSAT

- Unit propagation is unsound for MaxSAT
• **Unit propagation is unsound for MaxSAT**
  
  • Formula with all clauses soft:

\[
\{ (x) , (\neg x \lor y_1), (\neg x \lor y_2), (\neg y_1 \lor \neg z), (\neg y_2 \lor \neg z), (z) \}
\]

• After unit propagation:

\[
\{ (x) , (\neg x \lor y_1), (\neg x \lor y_2), (\neg y_1 \lor \neg z), (\neg y_2 \lor \neg z), (z) \}
\]

• Is \(2\) the MaxSAT solution??

• No!

• Enough to either falsify \( (x) \) or \( (z) \)

• Cannot use unit propagation

• Cannot learn clauses (using unit propagation)

• Need to solve MaxSAT using different techniques
Issues with MaxSAT

- **Unit propagation is unsound for MaxSAT**
  - Formula with all clauses soft:

    \[
    \{ (x), (\neg x \lor y_1), (\neg x \lor y_2), (\neg y_1 \lor \neg z), (\neg y_2 \lor \neg z), (z) \}\n    \]

  - After unit propagation:

    \[
    \{ (x), (\neg x \lor y_1), (\neg x \lor y_2), (\neg y_1 \lor \neg z), (\neg y_2 \lor \neg z), (z) \}\n    \]
• **Unit propagation is unsound for MaxSAT**
  
  • Formula with all clauses soft:

  \[
  \{(x), (\neg x \vee y_1), (\neg x \vee y_2), (\neg y_1 \vee \neg z), (\neg y_2 \vee \neg z), (z)\}
  \]

  • After unit propagation:

  \[
  \{(x), (\neg x \vee y_1), (\neg x \vee y_2), (\neg y_1 \vee \neg z), (\neg y_2 \vee \neg z), (z)\}
  \]

  • Is 2 the MaxSAT solution??
## Issues with MaxSAT

- **Unit propagation is unsound for MaxSAT**
  - Formula with all clauses soft:
    \[
    \{(x), (\neg x \lor y_1), (\neg x \lor y_2), (\neg y_1 \lor \neg z), (\neg y_2 \lor \neg z), (z)\}
    \]
  - After unit propagation:
    \[
    \{(x), (\neg x \lor y_1), (\neg x \lor y_2), (\neg y_1 \lor \neg z), (\neg y_2 \lor \neg z), (z)\}
    \]
  - Is 2 the MaxSAT solution??
  - **No!** Enough to either falsify \(x\) or \(z\)
• **Unit propagation is unsound for MaxSAT**
  • Formula with all clauses soft:
    \[
    \{(x), (\neg x \lor y_1), (\neg x \lor y_2), (\neg y_1 \lor \neg z), (\neg y_2 \lor \neg z), (z)\}
    \]
  • After unit propagation:
    \[
    \{(x), (\neg x \lor y_1), (\neg x \lor y_2), (\neg y_1 \lor \neg z), (\neg y_2 \lor \neg z), (z)\}
    \]
  • Is 2 the MaxSAT solution??
  • **No!** Enough to either falsify \((x)\) or \((z)\)
  • **Cannot** use unit propagation
Issues with MaxSAT

- **Unit propagation is unsound for MaxSAT**
  - Formula with all clauses soft:
    \[
    \{(x), (\neg x \lor y_1), (\neg x \lor y_2), (\neg y_1 \lor \neg z), (\neg y_2 \lor \neg z), (z)\}
    \]
  
  - After unit propagation:
    \[
    \{(x), (\neg x \lor y_1), (\neg x \lor y_2), (\neg y_1 \lor \neg z), (\neg y_2 \lor \neg z), (z)\}
    \]

  - Is 2 the MaxSAT solution??
  - **No!** Enough to either falsify \((x)\) or \((z)\)

- **Cannot** use unit propagation
- **Cannot** learn clauses (using unit propagation)
Issues with MaxSAT

• **Unit propagation is unsound for MaxSAT**
  • Formula with all clauses soft:
    \[
    \{(x), (\neg x \lor y_1), (\neg x \lor y_2), (\neg y_1 \lor \neg z), (\neg y_2 \lor \neg z), (z)\}
    \]
  • After unit propagation:
    \[
    \{(x), (\neg x \lor y_1), (\neg x \lor y_2), (\neg y_1 \lor \neg z), (\neg y_2 \lor \neg z), (z)\}
    \]
  • Is 2 the MaxSAT solution??
    • **No!** Enough to either falsify \((x)\) or \((z)\)
  • **Cannot** use unit propagation
  • **Cannot** learn clauses (using unit propagation)
  • Need to solve MaxSAT using different techniques
Many MaxSAT approaches

MaxSAT Algorithms

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

- For practical (**industrial**) instances: **core-guided** & **iterative MHS** approaches are the most effective

[MaxSAT14]
Core-guided solver performance – partial

Number \( x \) of instances solved in \( y \) seconds

CPU time in seconds

Number of instances

Open-WBO-In
QMaxSAT2-mt-13
QMaxSat-g2-12
QMaxSat0.4-11
QMaxSat-10

Source: [MaxSAT 2014 organizers]
Core-guided solver performance – weighted partial

Number x of instances solved in y seconds

CPU time in seconds

Number of instances

Eva500a
WPM1-2013
WPM1-11
pwbo2.1-12
wbo-1.4a-wcnf-10

Source: [MaxSAT 2014 organizers]
Outline

Minimal Unsatisfiability

MUS Enumeration

Maximum Satisfiability

Iterative SAT Solving

Core-Guided Algorithms

Minimum Hitting Sets
### Basic MaxSAT with iterative SAT solving

<table>
<thead>
<tr>
<th>( x_6 \lor x_2 )</th>
<th>( \neg x_6 \lor x_2 )</th>
<th>( \neg x_2 \lor x_1 )</th>
<th>( \neg x_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \neg x_6 \lor x_8 )</td>
<td>( x_6 \lor \neg x_8 )</td>
<td>( x_2 \lor x_4 )</td>
<td>( \neg x_4 \lor x_5 )</td>
</tr>
<tr>
<td>( x_7 \lor x_5 )</td>
<td>( \neg x_7 \lor x_5 )</td>
<td>( \neg x_5 \lor x_3 )</td>
<td>( \neg x_3 )</td>
</tr>
</tbody>
</table>

#### Example CNF formula
Basic MaxSAT with iterative SAT solving

\[\begin{align*}
x_6 \lor x_2 \lor r_1 & \quad \neg x_6 \lor x_2 \lor r_2 & \quad \neg x_2 \lor x_1 \lor r_3 & \quad \neg x_1 \lor r_4 \\
\neg x_6 \lor x_8 \lor r_5 & \quad x_6 \lor \neg x_8 \lor r_6 & \quad x_2 \lor x_4 \lor r_7 & \quad \neg x_4 \lor x_5 \lor r_8 \\
x_7 \lor x_5 \lor r_9 & \quad \neg x_7 \lor x_5 \lor r_{10} & \quad \neg x_5 \lor x_3 \lor r_{11} & \quad \neg x_3 \lor r_{12}
\end{align*}\]

\[\sum_{i=1}^{12} r_i \leq 12\]

Relax all clauses; Set UB = 12 + 1
Basic MaxSAT with iterative SAT solving

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

Formula is SAT; E.g. all \( x_i = 0 \) and \( r_1 = r_7 = r_9 = 1 \) (i.e. cost = 3)
Basic MaxSAT with iterative SAT solving

\[ x_6 \lor x_2 \lor r_1 \quad \neg x_6 \lor x_2 \lor r_2 \quad \neg x_2 \lor x_1 \lor r_3 \quad \neg x_1 \lor r_4 \]

\[ \neg x_6 \lor x_8 \lor r_5 \quad x_6 \lor \neg x_8 \lor r_6 \quad x_2 \lor x_4 \lor r_7 \quad \neg x_4 \lor x_5 \lor r_8 \]

\[ x_7 \lor x_5 \lor r_9 \quad \neg x_7 \lor x_5 \lor r_{10} \quad \neg x_5 \lor x_3 \lor r_{11} \quad \neg x_3 \lor r_{12} \]

\[ \sum_{i=1}^{12} r_i \leq 2 \]

Refine \( UB = 3 \)
Basic MaxSAT with iterative SAT solving

\[
\begin{align*}
    x_6 \lor x_2 \lor r_1 & \quad \neg x_6 \lor x_2 \lor r_2 & \quad \neg x_2 \lor x_1 \lor r_3 & \quad \neg x_1 \lor r_4 \\
    \neg x_6 \lor x_8 \lor r_5 & \quad x_6 \lor \neg x_8 \lor r_6 & \quad x_2 \lor x_4 \lor r_7 & \quad \neg x_4 \lor x_5 \lor r_8 \\
    x_7 \lor x_5 \lor r_9 & \quad \neg x_7 \lor x_5 \lor r_{10} & \quad \neg x_5 \lor x_3 \lor r_{11} & \quad \neg x_3 \lor r_{12} \\
    \sum_{i=1}^{12} r_i & \leq 2
\end{align*}
\]

Formula is SAT; E.g. \( x_1 = x_2 = 1; x_3 = \ldots = x_8 = 0 \) and \( r_4 = r_9 = 1 \) (i.e. cost = 2)
Basic MaxSAT with iterative SAT solving

\[ x_6 \lor x_2 \lor r_1 \quad \neg x_6 \lor x_2 \lor r_2 \quad \neg x_2 \lor x_1 \lor r_3 \quad \neg x_1 \lor r_4 \]

\[ \neg x_6 \lor x_8 \lor r_5 \quad x_6 \lor \neg x_8 \lor r_6 \quad x_2 \lor x_4 \lor r_7 \quad \neg x_4 \lor x_5 \lor r_8 \]

\[ x_7 \lor x_5 \lor r_9 \quad \neg x_7 \lor x_5 \lor r_{10} \quad \neg x_5 \lor x_3 \lor r_{11} \quad \neg x_3 \lor r_{12} \]

\[ \sum_{i=1}^{12} r_i \leq 1 \]

Refine \( UB = 2 \)
Basic MaxSAT with iterative SAT solving

\[
x_6 \lor x_2 \lor r_1 \\
\neg x_6 \lor x_2 \lor r_2 \\
\neg x_2 \lor x_1 \lor r_3 \\
\neg x_1 \lor r_4 \\
\neg x_6 \lor x_8 \lor r_5 \\
x_6 \lor \neg x_8 \lor r_6 \\
x_2 \lor x_4 \lor r_7 \\
\neg x_4 \lor x_5 \lor r_8 \\
x_7 \lor x_5 \lor r_9 \\
\neg x_7 \lor x_5 \lor r_{10} \\
\neg x_5 \lor x_3 \lor r_{11} \\
\neg x_3 \lor r_{12} \\
\sum_{i=1}^{12} r_i \leq 1
\]

Formula is UNSAT; terminate
Basic MaxSAT with iterative SAT solving

\[
x_6 \lor x_2 \lor r_1 \quad \neg x_6 \lor x_2 \lor r_2 \quad \neg x_2 \lor x_1 \lor r_3 \quad \neg x_1 \lor r_4
\]
\[
\neg x_6 \lor x_8 \lor r_5 \quad x_6 \lor \neg x_8 \lor r_6 \quad x_2 \lor x_4 \lor r_7 \quad \neg x_4 \lor x_5 \lor r_8
\]
\[
x_7 \lor x_5 \lor r_9 \quad \neg x_7 \lor x_5 \lor r_{10} \quad \neg x_5 \lor x_3 \lor r_{11} \quad \neg x_3 \lor r_{12}
\]
\[
\sum_{i=1}^{12} r_i \leq 1
\]

MaxSAT solution is last satisfied UB: \( UB = 2 \)
Basic MaxSAT with iterative SAT solving

\[
x_6 \lor x_2 \lor r_1 \\
\neg x_6 \lor x_2 \lor r_2 \\
\neg x_2 \lor x_1 \lor r_3 \\
\neg x_1 \lor r_4 \\
\neg x_6 \lor x_8 \lor r_5 \\
x_6 \lor \neg x_8 \lor r_6 \\
x_2 \lor x_4 \lor r_7 \\
\neg x_4 \lor x_5 \lor r_8 \\
x_7 \lor x_5 \lor r_9 \\
\neg x_7 \lor x_5 \lor r_{10} \\
\neg x_5 \lor x_3 \lor r_{11} \\
\neg x_3 \lor r_{12}
\]

\[\sum_{i=1}^{12} r_i \leq 1\]

MaxSAT solution is last satisfied UB: \(UB = 2\)

AtMostk/PB constraints over all relaxation variables

All (possibly many) soft clauses relaxed
Outline

- Minimal Unsatisfiability
- MUS Enumeration
- Maximum Satisfiability
  - Iterative SAT Solving
  - Core-Guided Algorithms
- Minimum Hitting Sets
Example CNF formula
MSU3 core-guided algorithm

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

Formula is \textbf{UNSAT}; \textbf{OPT} \leq |\varphi| - 1; Get unsat core
MSU3 core-guided algorithm

\[
\begin{align*}
x_6 \lor x_2 & \quad \neg x_6 \lor x_2 & \quad \neg x_2 \lor x_1 \lor r_1 & \quad \neg x_1 \lor r_2 \\
\neg x_6 \lor x_8 & \quad x_6 \lor \neg x_8 & \quad x_2 \lor x_4 \lor r_3 & \quad \neg x_4 \lor x_5 \lor r_4 \\
x_7 \lor x_5 & \quad \neg x_7 \lor x_5 & \quad \neg x_5 \lor x_3 \lor r_5 & \quad \neg x_3 \lor r_6 \\
\sum_{i=1}^{6} r_i \leq 1
\end{align*}
\]

Add relaxation variables and AtMost\(k\), \(k = 1\), constraint
Formula is (again) **UNSAT**; $\text{OPT} \leq |\varphi| - 2$; Get unsat core
MSU3 core-guided algorithm

\[ x_6 \lor x_2 \lor r_7 \quad \neg x_6 \lor x_2 \lor r_8 \quad \neg x_2 \lor x_1 \lor r_1 \quad \neg x_1 \lor r_2 \]

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

\[ x_7 \lor x_5 \lor r_9 \quad \neg x_7 \lor x_5 \lor r_{10} \quad \neg x_5 \lor x_3 \lor r_5 \quad \neg x_3 \lor r_6 \]

\[ \sum_{i=1}^{10} r_i \leq 2 \]

Add new relaxation variables and update AtMost\( k \), \( k=2 \), constraint
Instance is now SAT
MaxSAT solution is $|\varphi| - I = 12 - 2 = 10$
MSU3 core-guided algorithm

\[
\begin{align*}
x_6 \lor x_2 \lor r_7 & \quad \neg x_6 \lor x_2 \lor r_8 & \quad \neg x_2 \lor x_1 \lor r_1 & \quad \neg x_1 \lor r_2 \\
\neg x_6 \lor x_8 & \quad x_6 \lor \neg x_8 & \quad x_2 \lor x_4 \lor r_3 & \quad \neg x_4 \lor x_5 \lor r_4 \\
x_7 \lor x_5 \lor r_9 & \quad \neg x_7 \lor x_5 \lor r_{10} & \quad \neg x_5 \lor x_3 \lor r_5 & \quad \neg x_3 \lor r_6 \\
\sum_{i=1}^{10} r_i & \leq 2
\end{align*}
\]

MaxSAT solution is \(|\varphi| - I = 12 - 2 = 10\)

AtMostk/PB constraints used

Relaxed soft clauses become hard
MSU3 core-guided algorithm

\[ x_6 \lor x_2 \lor r_7 \quad \neg x_6 \lor x_2 \lor r_8 \quad \neg x_2 \lor x_1 \lor r_1 \quad \neg x_1 \lor r_2 \]

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

\[ x_7 \lor x_5 \lor r_9 \quad \neg x_7 \lor x_5 \lor r_{10} \quad \neg x_5 \lor x_3 \lor r_5 \quad \neg x_3 \lor r_6 \]

\[ \sum_{i=1}^{10} r_i \leq 2 \]

MaxSAT solution is \(|\varphi| - I = 12 - 2 = 10\)

AtMostk/PB constraints used

Some clauses not relaxed

Relaxed soft clauses become **hard**
Outline

Minimal Unsatisfiability

MUS Enumeration

Maximum Satisfiability
  Iterative SAT Solving
  Core-Guided Algorithms

Minimum Hitting Sets
MHS approach for MaxSAT

\[ \begin{align*}
  c_1 &= x_6 \lor x_2 &
  c_2 &= \neg x_6 \lor x_2 &
  c_3 &= \neg x_2 \lor x_1 &
  c_4 &= \neg x_1 \\
  c_5 &= \neg x_6 \lor x_8 &
  c_6 &= x_6 \lor \neg x_8 &
  c_7 &= x_2 \lor x_4 &
  c_8 &= \neg x_4 \lor x_5 \\
  c_9 &= x_7 \lor x_5 &
  c_{10} &= \neg x_7 \lor x_5 &
  c_{11} &= \neg x_5 \lor x_3 &
  c_{12} &= \neg x_3
\end{align*} \]

\[ \mathcal{K} = \emptyset \]

- Find MHS of \( \mathcal{K} \):
MHS approach for MaxSAT

\[
\begin{align*}
c_1 &= x_6 \lor x_2 \\
c_2 &= \neg x_6 \lor x_2 \\
c_3 &= \neg x_2 \lor x_1 \\
c_4 &= \neg x_1 \\
c_5 &= \neg x_6 \lor x_8 \\
c_6 &= x_6 \lor \neg x_8 \\
c_7 &= x_2 \lor x_4 \\
c_8 &= \neg x_4 \lor x_5 \\
c_9 &= x_7 \lor x_5 \\
c_{10} &= \neg x_7 \lor x_5 \\
c_{11} &= \neg x_5 \lor x_3 \\
c_{12} &= \neg x_3
\end{align*}
\]

\[\mathcal{K} = \emptyset\]

- Find MHS of \(\mathcal{K}\): \(\emptyset\)
MHS approach for MaxSAT

\[ c_1 = x_6 \lor x_2 \quad c_2 = \neg x_6 \lor x_2 \quad c_3 = \neg x_2 \lor x_1 \quad c_4 = \neg x_1 \]

\[ c_5 = \neg x_6 \lor x_8 \quad c_6 = x_6 \lor \neg x_8 \quad c_7 = x_2 \lor x_4 \quad c_8 = \neg x_4 \lor x_5 \]

\[ c_9 = x_7 \lor x_5 \quad c_{10} = \neg x_7 \lor x_5 \quad c_{11} = \neg x_5 \lor x_3 \quad c_{12} = \neg x_3 \]

\[ \mathcal{K} = \emptyset \]

- Find MHS of \( \mathcal{K} \): \( \emptyset \)
- \( \text{SAT}(F \setminus \emptyset) \)?
MHS approach for MaxSAT

\[c_1 = x_6 \lor x_2 \quad c_2 = \neg x_6 \lor x_2 \quad c_3 = \neg x_2 \lor x_1 \quad c_4 = \neg x_1 \]
\[c_5 = \neg x_6 \lor x_8 \quad c_6 = x_6 \lor \neg x_8 \quad c_7 = x_2 \lor x_4 \quad c_8 = \neg x_4 \lor x_5 \]
\[c_9 = x_7 \lor x_5 \quad c_{10} = \neg x_7 \lor x_5 \quad c_{11} = \neg x_5 \lor x_3 \quad c_{12} = \neg x_3 \]

\[\mathcal{K} = \emptyset\]

- Find MHS of \( \mathcal{K} \): \( \emptyset \)
- \(\text{SAT}(\mathcal{F} \setminus \emptyset)\)? No
MHS approach for MaxSAT

\[ \begin{align*}
\mathcal{C}_1 &= x_6 \lor x_2 & \mathcal{C}_2 &= \neg x_6 \lor x_2 & \mathcal{C}_3 &= \neg x_2 \lor x_1 & \mathcal{C}_4 &= \neg x_1 \\
\mathcal{C}_5 &= \neg x_6 \lor x_8 & \mathcal{C}_6 &= x_6 \lor \neg x_8 & \mathcal{C}_7 &= x_2 \lor x_4 & \mathcal{C}_8 &= \neg x_4 \lor x_5 \\
\mathcal{C}_9 &= x_7 \lor x_5 & \mathcal{C}_{10} &= \neg x_7 \lor x_5 & \mathcal{C}_{11} &= \neg x_5 \lor x_3 & \mathcal{C}_{12} &= \neg x_3
\end{align*} \]

\[ \mathcal{K} = \emptyset \]

- Find MHS of \( \mathcal{K} \): \( \emptyset \)
- \( \text{SAT}(\mathcal{F} \setminus \emptyset) \)? No
- Core of \( \mathcal{F} \): \( \{c_1, c_2, c_3, c_4\} \)
MHS approach for MaxSAT

\[ \begin{align*}
    c_1 &= x_6 \lor x_2 & c_2 &= \neg x_6 \lor x_2 & c_3 &= \neg x_2 \lor x_1 & c_4 &= \neg x_1 \\
    c_5 &= \neg x_6 \lor x_8 & c_6 &= x_6 \lor \neg x_8 & c_7 &= x_2 \lor x_4 & c_8 &= \neg x_4 \lor x_5 \\
    c_9 &= x_7 \lor x_5 & c_{10} &= \neg x_7 \lor x_5 & c_{11} &= \neg x_5 \lor x_3 & c_{12} &= \neg x_3 
\end{align*} \]

\[ \mathcal{K} = \{\{c_1, c_2, c_3, c_4\}\} \]

- Find MHS of \( \mathcal{K} \): \( \emptyset \)
- \( \text{SAT}(\mathcal{F} \setminus \emptyset) \)? No
- Core of \( \mathcal{F} \): \( \{c_1, c_2, c_3, c_4\} \). Update \( \mathcal{K} \)
MHS approach for MaxSAT

\[ c_1 = x_6 \lor x_2 \quad c_2 = \neg x_6 \lor x_2 \quad c_3 = \neg x_2 \lor x_1 \quad c_4 = \neg x_1 \]
\[ c_5 = \neg x_6 \lor x_8 \quad c_6 = x_6 \lor \neg x_8 \quad c_7 = x_2 \lor x_4 \quad c_8 = \neg x_4 \lor x_5 \]
\[ c_9 = x_7 \lor x_5 \quad c_{10} = \neg x_7 \lor x_5 \quad c_{11} = \neg x_5 \lor x_3 \quad c_{12} = \neg x_3 \]

\[ \mathcal{K} = \{\{c_1, c_2, c_3, c_4\}\} \]

- Find MHS of \( \mathcal{K} \):
MHS approach for MaxSAT

\[ c_1 = x_6 \lor x_2 \quad c_2 = \neg x_6 \lor x_2 \quad c_3 = \neg x_2 \lor x_1 \quad c_4 = \neg x_1 \]

\[ c_5 = \neg x_6 \lor x_8 \quad c_6 = x_6 \lor \neg x_8 \quad c_7 = x_2 \lor x_4 \quad c_8 = \neg x_4 \lor x_5 \]

\[ c_9 = x_7 \lor x_5 \quad c_{10} = \neg x_7 \lor x_5 \quad c_{11} = \neg x_5 \lor x_3 \quad c_{12} = \neg x_3 \]

\[ \mathcal{K} = \{\{c_1, c_2, c_3, c_4\}\} \]

- Find MHS of \( \mathcal{K} \): E.g. \( \{c_1\} \)
MHS approach for MaxSAT

\[ c_1 = x_6 \lor x_2 \quad c_2 = \neg x_6 \lor x_2 \quad c_3 = \neg x_2 \lor x_1 \quad c_4 = \neg x_1 \]

\[ c_5 = \neg x_6 \lor x_8 \quad c_6 = x_6 \lor \neg x_8 \quad c_7 = x_2 \lor x_4 \quad c_8 = \neg x_4 \lor x_5 \]

\[ c_9 = x_7 \lor x_5 \quad c_{10} = \neg x_7 \lor x_5 \quad c_{11} = \neg x_5 \lor x_3 \quad c_{12} = \neg x_3 \]

\[ \mathcal{K} = \{c_1, c_2, c_3, c_4\} \]

- Find MHS of \( \mathcal{K} \): E.g. \( \{c_1\} \)
- \( \text{SAT}(\mathcal{F} \setminus \{c_1\}) \)?
MHS approach for MaxSAT

\[ \begin{align*}
    c_1 &= x_6 \lor x_2 \\
    c_2 &= \neg x_6 \lor x_2 \\
    c_3 &= \neg x_2 \lor x_1 \\
    c_4 &= \neg x_1 \\
    c_5 &= \neg x_6 \lor x_8 \\
    c_6 &= x_6 \lor \neg x_8 \\
    c_7 &= x_2 \lor x_4 \\
    c_8 &= \neg x_4 \lor x_5 \\
    c_9 &= x_7 \lor x_5 \\
    c_{10} &= \neg x_7 \lor x_5 \\
    c_{11} &= \neg x_5 \lor x_3 \\
    c_{12} &= \neg x_3
\end{align*} \]

\[ \mathcal{K} = \{ \{c_1, c_2, c_3, c_4\} \} \]

- Find MHS of \( \mathcal{K} \): E.g. \( \{c_1\} \)
- \( \text{SAT}(\mathcal{F} \setminus \{c_1\})? \) No
MHS approach for MaxSAT

\[ c_1 = x_6 \lor x_2 \quad c_2 = \neg x_6 \lor x_2 \quad c_3 = \neg x_2 \lor x_1 \quad c_4 = \neg x_1 \]
\[ c_5 = \neg x_6 \lor x_8 \quad c_6 = x_6 \lor \neg x_8 \quad c_7 = x_2 \lor x_4 \quad c_8 = \neg x_4 \lor x_5 \]
\[ c_9 = x_7 \lor x_5 \quad c_{10} = \neg x_7 \lor x_5 \quad c_{11} = \neg x_5 \lor x_3 \quad c_{12} = \neg x_3 \]

\[ \mathcal{K} = \{c_1, c_2, c_3, c_4\} \]

- Find MHS of \( \mathcal{K} \): E.g. \( \{c_1\}\)
- \( \text{SAT}(\mathcal{F} \setminus \{c_1\})? \) No
- Core of \( \mathcal{F} \): \( \{c_9, c_{10}, c_{11}, c_{12}\} \)
MHS approach for MaxSAT

\[ c_1 = x_6 \lor x_2 \quad c_2 = \neg x_6 \lor x_2 \quad c_3 = \neg x_2 \lor x_1 \quad c_4 = \neg x_1 \]

\[ c_5 = \neg x_6 \lor x_8 \quad c_6 = x_6 \lor \neg x_8 \quad c_7 = x_2 \lor x_4 \quad c_8 = \neg x_4 \lor x_5 \]

\[ c_9 = x_7 \lor x_5 \quad c_{10} = \neg x_7 \lor x_5 \quad c_{11} = \neg x_5 \lor x_3 \quad c_{12} = \neg x_3 \]

\[ \mathcal{K} = \{\{c_1, c_2, c_3, c_4\}, \{c_9, c_{10}, c_{11}, c_{12}\}\} \]

- Find MHS of \( \mathcal{K} \): E.g. \( \{c_1\} \)
- \( \text{SAT}(\mathcal{F}\setminus\{c_1\})? \text{ No} \)
- Core of \( \mathcal{F} \): \( \{c_9, c_{10}, c_{11}, c_{12}\} \). Update \( \mathcal{K} \)
MHS approach for MaxSAT

\begin{align*}
  c_1 &= x_6 \lor x_2 & c_2 &= \neg x_6 \lor x_2 & c_3 &= \neg x_2 \lor x_1 & c_4 &= \neg x_1 \\
  c_5 &= \neg x_6 \lor x_8 & c_6 &= x_6 \lor \neg x_8 & c_7 &= x_2 \lor x_4 & c_8 &= \neg x_4 \lor x_5 \\
  c_9 &= x_7 \lor x_5 & c_{10} &= \neg x_7 \lor x_5 & c_{11} &= \neg x_5 \lor x_3 & c_{12} &= \neg x_3
\end{align*}

\[ \mathcal{K} = \{\{c_1, c_2, c_3, c_4\}, \{c_9, c_{10}, c_{11}, c_{12}\}\} \]

• Find MHS of \( \mathcal{K} \):
MHS approach for MaxSAT

\[
\begin{align*}
  c_1 &= x_6 \lor x_2 &  c_2 &= \neg x_6 \lor x_2 &  c_3 &= \neg x_2 \lor x_1 &  c_4 &= \neg x_1 \\
  c_5 &= \neg x_6 \lor x_8 &  c_6 &= x_6 \lor \neg x_8 &  c_7 &= x_2 \lor x_4 &  c_8 &= \neg x_4 \lor x_5 \\
  c_9 &= x_7 \lor x_5 &  c_{10} &= \neg x_7 \lor x_5 &  c_{11} &= \neg x_5 \lor x_3 &  c_{12} &= \neg x_3
\end{align*}
\]

\[
\mathcal{K} = \{\{c_1, c_2, c_3, c_4\}, \{c_9, c_{10}, c_{11}, c_{12}\}\}
\]

- Find MHS of \(\mathcal{K}\): E.g. \(\{c_1, c_9\}\)
MHS approach for MaxSAT

\[
\begin{align*}
  c_1 &= x_6 \lor x_2 & c_2 &= \neg x_6 \lor x_2 & c_3 &= \neg x_2 \lor x_1 & c_4 &= \neg x_1 \\
  c_5 &= \neg x_6 \lor x_8 & c_6 &= x_6 \lor \neg x_8 & c_7 &= x_2 \lor x_4 & c_8 &= \neg x_4 \lor x_5 \\
  c_9 &= x_7 \lor x_5 & c_{10} &= \neg x_7 \lor x_5 & c_{11} &= \neg x_5 \lor x_3 & c_{12} &= \neg x_3
\end{align*}
\]

\[\mathcal{K} = \{\{c_1, c_2, c_3, c_4\}, \{c_9, c_{10}, c_{11}, c_{12}\}\}\]

- Find MHS of \(\mathcal{K}\): E.g. \(\{c_1, c_9\}\)
- \(\text{SAT}(\mathcal{F} \setminus \{c_1, c_9\})?\)
MHS approach for MaxSAT

\[
\begin{align*}
    c_1 &= x_6 \lor x_2 & c_2 &= \neg x_6 \lor x_2 & c_3 &= \neg x_2 \lor x_1 & c_4 &= \neg x_1 \\
    c_5 &= \neg x_6 \lor x_8 & c_6 &= x_6 \lor \neg x_8 & c_7 &= x_2 \lor x_4 & c_8 &= \neg x_4 \lor x_5 \\
    c_9 &= x_7 \lor x_5 & c_{10} &= \neg x_7 \lor x_5 & c_{11} &= \neg x_5 \lor x_3 & c_{12} &= \neg x_3
\end{align*}
\]

\[\mathcal{K} = \{\{c_1, c_2, c_3, c_4\}, \{c_9, c_{10}, c_{11}, c_{12}\}\}\]

- Find MHS of \(\mathcal{K}\): E.g. \(\{c_1, c_9\}\)
- \(\text{SAT}(\mathcal{F} \setminus \{c_1, c_9\})\)? No
MHS approach for MaxSAT

\[
\begin{align*}
    c_1 &= x_6 \lor x_2 &
    c_2 &= \neg x_6 \lor x_2 &
    c_3 &= \neg x_2 \lor x_1 &
    c_4 &= \neg x_1 \\
    c_5 &= \neg x_6 \lor x_8 &
    c_6 &= x_6 \lor \neg x_8 &
    c_7 &= x_2 \lor x_4 &
    c_8 &= \neg x_4 \lor x_5 \\
    c_9 &= x_7 \lor x_5 &
    c_{10} &= \neg x_7 \lor x_5 &
    c_{11} &= \neg x_5 \lor x_3 &
    c_{12} &= \neg x_3
\end{align*}
\]

\[\mathcal{K} = \{\{c_1, c_2, c_3, c_4\}, \{c_9, c_{10}, c_{11}, c_{12}\}\}\]

- Find MHS of \(\mathcal{K}\): E.g. \(\{c_1, c_9\}\)
- \(\text{SAT}(\mathcal{F} \setminus \{c_1, c_9\})\)? No
- Core of \(\mathcal{F}\): \(\{c_3, c_4, c_7, c_8, c_{11}, c_{12}\}\)
MHS approach for MaxSAT

\[\begin{align*}
  c_1 &= x_6 \lor x_2 \\
  c_2 &= \neg x_6 \lor x_2 \\
  c_3 &= \neg x_2 \lor x_1 \\
  c_4 &= \neg x_1 \\
  c_5 &= \neg x_6 \lor x_8 \\
  c_6 &= x_6 \lor \neg x_8 \\
  c_7 &= x_2 \lor x_4 \\
  c_8 &= \neg x_4 \lor x_5 \\
  c_9 &= x_7 \lor x_5 \\
  c_{10} &= \neg x_7 \lor x_5 \\
  c_{11} &= \neg x_5 \lor x_3 \\
  c_{12} &= \neg x_3
\end{align*}\]

\[\mathcal{K} = \{\{c_1, c_2, c_3, c_4\}, \{c_9, c_{10}, c_{11}, c_{12}\}, \{c_3, c_4, c_7, c_8, c_{11}, c_{12}\}\}\]

- Find MHS of \(\mathcal{K}\): E.g. \(\{c_1, c_9\}\)
- \(\text{SAT}(\mathcal{F} \setminus \{c_1, c_9\})\)? No
- Core of \(\mathcal{F}\): \(\{c_3, c_4, c_7, c_8, c_{11}, c_{12}\}\). Update \(\mathcal{K}\)
MHS approach for MaxSAT

\[ c_1 = x_6 \lor x_2 \quad c_2 = \neg x_6 \lor x_2 \quad c_3 = \neg x_2 \lor x_1 \quad c_4 = \neg x_1 \]
\[ c_5 = \neg x_6 \lor x_8 \quad c_6 = x_6 \lor \neg x_8 \quad c_7 = x_2 \lor x_4 \quad c_8 = \neg x_4 \lor x_5 \]
\[ c_9 = x_7 \lor x_5 \quad c_{10} = \neg x_7 \lor x_5 \quad c_{11} = \neg x_5 \lor x_3 \quad c_{12} = \neg x_3 \]

\[ \mathcal{K} = \{\{c_1, c_2, c_3, c_4\}, \{c_9, c_{10}, c_{11}, c_{12}\}, \{c_3, c_4, c_7, c_8, c_{11}, c_{12}\}\} \]

- Find MHS of \( \mathcal{K} \):
MHS approach for MaxSAT

\[ c_1 = x_6 \lor x_2 \quad c_2 = \neg x_6 \lor x_2 \quad c_3 = \neg x_2 \lor x_1 \quad c_4 = \neg x_1 \]

\[ c_5 = \neg x_6 \lor x_8 \quad c_6 = x_6 \lor \neg x_8 \quad c_7 = x_2 \lor x_4 \quad c_8 = \neg x_4 \lor x_5 \]

\[ c_9 = x_7 \lor x_5 \quad c_{10} = \neg x_7 \lor x_5 \quad c_{11} = \neg x_5 \lor x_3 \quad c_{12} = \neg x_3 \]

\[ \mathcal{K} = \{\{c_1, c_2, c_3, c_4\}, \{c_9, c_{10}, c_{11}, c_{12}\}, \{c_3, c_4, c_7, c_8, c_{11}, c_{12}\}\} \]

- Find MHS of \( \mathcal{K} \): E.g. \( \{c_4, c_9\} \)
MHS approach for MaxSAT

\[ c_1 = x_6 \lor x_2 \quad c_2 = \neg x_6 \lor x_2 \quad c_3 = \neg x_2 \lor x_1 \quad c_4 = \neg x_1 \]
\[ c_5 = \neg x_6 \lor x_8 \quad c_6 = x_6 \lor \neg x_8 \quad c_7 = x_2 \lor x_4 \quad c_8 = \neg x_4 \lor x_5 \]
\[ c_9 = x_7 \lor x_5 \quad c_{10} = \neg x_7 \lor x_5 \quad c_{11} = \neg x_5 \lor x_3 \quad c_{12} = \neg x_3 \]

\[ \mathcal{K} = \{ \{ c_1, c_2, c_3, c_4 \}, \{ c_9, c_{10}, c_{11}, c_{12} \}, \{ c_3, c_4, c_7, c_8, c_{11}, c_{12} \} \} \]

- Find MHS of \( \mathcal{K} \): E.g. \( \{ c_4, c_9 \} \)
- \( \text{SAT}(\mathcal{F} \setminus \{ c_4, c_9 \}) \)?
MHS approach for MaxSAT

\[ c_1 = x_6 \lor x_2 \quad c_2 = \neg x_6 \lor x_2 \quad c_3 = \neg x_2 \lor x_1 \quad c_4 = \neg x_1 \]

\[ c_5 = \neg x_6 \lor x_8 \quad c_6 = x_6 \lor \neg x_8 \quad c_7 = x_2 \lor x_4 \quad c_8 = \neg x_4 \lor x_5 \]

\[ c_9 = x_7 \lor x_5 \quad c_{10} = \neg x_7 \lor x_5 \quad c_{11} = \neg x_5 \lor x_3 \quad c_{12} = \neg x_3 \]

\[ \mathcal{K} = \{\{c_1, c_2, c_3, c_4\}, \{c_9, c_{10}, c_{11}, c_{12}\}, \{c_3, c_4, c_7, c_8, c_{11}, c_{12}\}\} \]

- Find MHS of \( \mathcal{K} \): E.g. \( \{c_4, c_9\} \)
- \( \text{SAT}(\mathcal{F} \setminus \{c_4, c_9\}) \)? Yes
MHS approach for MaxSAT

\[
\begin{align*}
    c_1 &= x_6 \lor x_2 & c_2 &= \neg x_6 \lor x_2 & c_3 &= \neg x_2 \lor x_1 & c_4 &= \neg x_1 \\
    c_5 &= \neg x_6 \lor x_8 & c_6 &= x_6 \lor \neg x_8 & c_7 &= x_2 \lor x_4 & c_8 &= \neg x_4 \lor x_5 \\
    c_9 &= x_7 \lor x_5 & c_{10} &= \neg x_7 \lor x_5 & c_{11} &= \neg x_5 \lor x_3 & c_{12} &= \neg x_3
\end{align*}
\]

\[\mathcal{K} = \{\{c_1, c_2, c_3, c_4\}, \{c_9, c_{10}, c_{11}, c_{12}\}, \{c_3, c_4, c_7, c_8, c_{11}, c_{12}\}\}\]

- Find MHS of \(\mathcal{K}\): E.g. \(\{c_4, c_9\}\)
- SAT\((\mathcal{F} \setminus \{c_4, c_9\})? Yes\)
- Terminate & return 2
MaxSAT solving with SAT oracles – a sample

- 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>[BP10]</td>
</tr>
<tr>
<td>Binary search</td>
<td>Linear*</td>
<td>[FM06]</td>
</tr>
<tr>
<td>FM/WMSU1/WPM1</td>
<td>Exponential**</td>
<td>[FM06, MP08, MMSP09, ABL09, ABGL12]</td>
</tr>
<tr>
<td>WPM2</td>
<td>Exponential**</td>
<td>[ABL10, ABL13]</td>
</tr>
<tr>
<td>Bin-Core-Dis</td>
<td>Linear</td>
<td>[HMM11, MHM12]</td>
</tr>
<tr>
<td>Iterative MHS</td>
<td>Exponential</td>
<td>[DB11, DB13a, DB13b]</td>
</tr>
</tbody>
</table>

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

- But also additional recent work:
  - Progression
  - Soft cardinality constraints (OLL)
    - Recent implementation (RC2, using PySAT) won 2018 MaxSAT Evaluation
  - MaxSAT resolution
  - ...
Exploring With SAT Oracles
Incremental SAT solving

- SAT solver often called *multiple* times on related formulas
- It helps to make *incremental* changes & remember already *learned* clauses (that still hold)
Incremental SAT solving

- SAT solver often called *multiple* times on related formulas
- It helps to make *incremental* changes & remember already *learned* clauses (that still hold)
- Most often used solution:

[ES03]
Incremental SAT solving

- SAT solver often called **multiple** times on related formulas

- It helps to make **incremental** changes & remember already **learned** clauses (that still hold)

- Most often used solution:
  - Use **activation(selector/indicator)** variables
    
    | Given clause | Added to SAT solver |
    |--------------|---------------------|
    | $c_i$        | $c_i \lor \overline{s_i}$ |

[ES03]
Incremental SAT solving

- SAT solver often called **multiple** times on related formulas

- It helps to make **incremental** changes & remember already **learned** clauses (that still hold)

- Most often used solution:
  - Use **activation/selector/indicator** variables
  
<table>
<thead>
<tr>
<th>Given clause</th>
<th>Added to SAT solver</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_i$</td>
<td>$c_i \lor \overline{s_i}$</td>
</tr>
</tbody>
</table>

  - To **activate** clause: add assumption $s_i = 1$
Incremental SAT solving

- SAT solver often called **multiple** times on related formulas

- It helps to make **incremental** changes & remember already **learned** clauses (that still hold)

- Most often used solution: [ES03]
  - Use **activationselectorindicator** variables
    - Given clause
    - Added to SAT solver
    - $c_i$
    - $c_i \lor \overline{s}_i$
  - To **activate** clause: add assumption $s_i = 1$
  - To **deactivate** clause: add assumption $s_i = 0$ (optional)
Incremental SAT solving

- SAT solver often called *multiple* times on related formulas
- It helps to make *incremental* changes & remember already *learned* clauses (that still hold)

- Most often used solution:
  - Use *activation(selector/indicator)* variables
    - **Given clause** | **Added to SAT solver**
      | $c_i$ | $c_i \lor \overline{s_i}$
    - To *activate* clause: add assumption $s_i = 1$
    - To *deactivate* clause: add assumption $s_i = 0$ (optional)
    - To *remove* clause: add unit ($\overline{s_i}$)
Incremental SAT solving

• SAT solver often called *multiple* times on related formulas

• It helps to make *incremental* changes & remember already *learned* clauses (that still hold)

• Most often used solution:
  
  • Use *activation(selector/indicator)* variables

<table>
<thead>
<tr>
<th>Given clause</th>
<th>Added to SAT solver</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_i$</td>
<td>$c_i \lor \overline{s_i}$</td>
</tr>
</tbody>
</table>

  • To *activate* clause: add assumption $s_i = 1$
  • To *deactivate* clause: add assumption $s_i = 0$ (optional)
  • To *remove* clause: add unit ($\overline{s_i}$)
  • *Any* learned clause contains explanation given working assumptions (more next)
An example

\[ B = \{ (\bar{a} \lor b), (\bar{a} \lor c) \} \]
\[ S = \{ (a \lor \bar{s}_1), (b \lor \bar{c} \lor \bar{s}_2), (a \lor \bar{c} \lor \bar{s}_3), (a \lor \bar{b} \lor \bar{s}_4) \} \]

- Background knowledge \( B \): **final** clauses, i.e. no indicator variables
- Soft clauses \( S \): add indicator variables \( \{ s_1, s_2, s_3, s_4 \} \)
An example

\[ B = \{ (\overline{a} \lor b), (\overline{a} \lor c) \} \]
\[ S = \{ (a \lor s_1), (\overline{b} \lor \overline{c} \lor s_2), (a \lor \overline{c} \lor s_3), (a \lor \overline{b} \lor s_4) \} \]

- Background knowledge \( B \): final clauses, i.e. no indicator variables
- Soft clauses \( S \): add indicator variables \( \{s_1, s_2, s_3, s_4\} \)
- E.g. given assumptions \( \{s_1 = 1, s_2 = 0, s_3 = 0, s_4 = 1\} \), SAT solver handles formula:

\[ \mathcal{F} = \{ (\overline{a} \lor b), (\overline{a} \lor c), (a), (a \lor \overline{b}) \} \]

which is satisfiable
Quiz – what happens in this case?

\[ B = \{(\bar{a} \lor b), (\bar{a} \lor c)\} \]
\[ S = \{(a \lor \bar{s_1}), (b \lor \bar{c} \lor \bar{s_2}), (a \lor \bar{c} \lor \bar{s_3}), (a \lor \bar{b} \lor \bar{s_4})\} \]

• Given assumptions \{s_1 = 1, s_2 = 1, s_3 = 1, s_4 = 1\}?
Quiz – what happens in this case?

\[
\mathcal{B} = \{ (\bar{a} \lor b), (\bar{a} \lor c) \}
\]

\[
\mathcal{S} = \{ (a \lor \bar{s}_1), (\bar{b} \lor \bar{c} \lor \bar{s}_2), (a \lor \bar{c} \lor \bar{s}_3), (a \lor \bar{b} \lor \bar{s}_4) \}
\]

- Given assumptions \( \{ s_1 = 1, s_2 = 1, s_3 = 1, s_4 = 1 \} \)?
Quiz – what happens in this case?

\[ \mathcal{B} = \{ (\bar{a} \lor b), (\bar{a} \lor c) \} \]
\[ \mathcal{S} = \{ (a \lor \bar{s_1}), (\bar{b} \lor \bar{c} \lor \bar{s_2}), (a \lor \bar{c} \lor \bar{s_3}), (a \lor \bar{b} \lor \bar{s_4}) \} \]

- Given assumptions \( \{ s_1 = 1, s_2 = 1, s_3 = 1, s_4 = 1 \} \)?

- Unsatisfiable core: 1\textsuperscript{st} and 2\textsuperscript{nd} clauses of \( \mathcal{S} \), given \( \mathcal{B} \)
Overview of PySAT

PySAT modules
- solvers module
- cardenc module
- formula module

PySAT API

- Open source, available on github
- Comprehensive list of SAT solvers
- Comprehensive list of cardinality encodings
- Fairly comprehensive documentation
- Several use cases
Overview of PySAT

- Open source, available on github
Overview of PySAT

- **Open source**, available on **github**
- Comprehensive list of **SAT solvers**
- Comprehensive list of **cardinality encodings**
- Fairly comprehensive documentation
- Several use cases
Available solvers

<table>
<thead>
<tr>
<th>Solver</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glucose</td>
<td>3.0</td>
</tr>
<tr>
<td>Glucose</td>
<td>4.1</td>
</tr>
<tr>
<td>Lingeling</td>
<td>bbc-9230380-160707</td>
</tr>
<tr>
<td>Minicard</td>
<td>1.2</td>
</tr>
<tr>
<td>Minisat</td>
<td>2.2 release</td>
</tr>
<tr>
<td>Minisat</td>
<td>GitHub version</td>
</tr>
<tr>
<td>MapleCM</td>
<td>SAT competition 2018</td>
</tr>
<tr>
<td>Maplesat</td>
<td>MapleCOMSPS_LRB</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

• Solvers can either be used incrementally or non-incrementally
• Tools can use multiple solvers, e.g. for hitting set dualization or CEGAR-based QBF solving

• URL: https://pysathq.github.io/docs/html/api/solvers.html
Formula manipulation

Features

- CNF & Weighted CNF (WCNF)
- Read formulas from file/string
- Write formulas to file
- Append clauses to formula
- Negate CNF formulas
- Translate between CNF and WCNF
- ID manager

**URL**: https://pysathq.github.io/docs/html/api/formula.html
Available cardinality encodings

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>pairwise</td>
<td>AtMost1</td>
</tr>
<tr>
<td>bitwise</td>
<td>AtMost1</td>
</tr>
<tr>
<td>ladder</td>
<td>AtMost1</td>
</tr>
<tr>
<td>sequential counter</td>
<td>AtMost$k$</td>
</tr>
<tr>
<td>sorting network</td>
<td>AtMost$k$</td>
</tr>
<tr>
<td>cardinality network</td>
<td>AtMost$k$</td>
</tr>
<tr>
<td>totalizer</td>
<td>AtMost$k$</td>
</tr>
<tr>
<td>mtotalizer</td>
<td>AtMost$k$</td>
</tr>
<tr>
<td>kmtotalizer</td>
<td>AtMost$k$</td>
</tr>
</tbody>
</table>

- Also AtLeast$K$ and Equals$K$ constraints

- **URL:**
  https://pysathq.github.io/docs/html/api/card.html
Installation & info

• Installation:
  $ [sudo] pip2|pip3 install python-sat

• Website: https://pysathq.github.io/
Basic interface – Python3 shell

```python
>>> from pysat.card import *
>>> am1 = CardEnc.atmost(lits=[1, -2, 3], encoding=EncType.pairwise)
>>> print(am1.clauses)
[[-1, 2], [-1, -3], [2, -3]]
>>> from pysat.solvers import Solver
>>> with Solver(name='m22', bootstrap_with=am1.clauses) as s:
...     if s.solve(assumptions=[1, 2, 3]) == False:
...         print(s.get_core())
[3, 1]
```
Basic interface – Python3 script

```python
#!/usr/local/bin/python3
from sys import argv

from pysat.formula import CNF
from pysat.solvers import Glucose3, Solver

def main():
    formula = CNF()
    formula.append([-1, 2, 4])
    formula.append([1, -2, 5])
    formula.append([-1, -2, 6])
    formula.append([1, 2, 7])

    g = Glucose3(bootstrap_with=formula.clauses)
    if g.solve(assumptions=[-4, -5, -6, -7]) == False:
        print("Core: ", g.get_core())
```

Example: naive (deletion) MUS extraction

**Input** : Set $\mathcal{F}$

**Output**: Minimal subset $\mathcal{M}$

begin

\[
\mathcal{M} \leftarrow \mathcal{F}
\]

foreach $c \in \mathcal{M}$ do

if $\neg$SAT($\mathcal{M} \setminus \{c\}$) then

\[
\mathcal{M} \leftarrow \mathcal{M} \setminus \{c\}
\]

end

return $\mathcal{M}$

end

- Number of predicate tests: $\mathcal{O}(m)$

[CD91, BDTW93]
def main():
    cnf = CNF(from_file=argv[1])  # create a CNF object from file
    (rnv, assumps) = add_assumps(cnf)

    oracle = Solver(name='g3', bootstrap_with=cnf.clauses)

    mus = find_mus(assumps, oracle)
    mus = [ref - rnv for ref in mus]
    print("MUS: ", mus)

if __name__ == '__main__':
    main()
def add_assups(cnf):
    rnv = topv = cnf.nv
    assumps = []  # list of assumptions to use
    for i in range(len(cnf.claus):
        topv += 1
        assumps.append(topv)  # register literal
        cnf.claus[i].append(−topv)  # extend clause with literal
    cnf.nv = cnf.nv + len(assumps)  # update # of vars
    return rnv, assumps

def main():
    cnf = CNF(from_file=argv[1])  # create a CNF object from file
    (rnv, assumps) = add_assups(cnf)

    oracle = Solver(name='g3', bootstrap_with=cnf.claus)

    mus = find_mus(assumps, oracle)
    mus = [ref − rnv for ref in mus]
    print("MUS: ", mus)

if __name__ == "__main__":
    main()
from sys import argv
from pysat.formula import CNF
from pysat.solvers import Solver

def find_mus(assmp, oracle):
    i = 0
    while i < len(assmp):
        ts = assmp[:i] + assmp[(i+1):]
        if not oracle.solve(assumptions=ts):
            assmp = ts
        else:
            i += 1
    return assmp
```python
from sys import argv
from pysat.formula import CNF
from pysat.solvers import Solver

def find_mus(assmp, oracle):
    i = 0
    while i < len(assmp):
        ts = assmp[:i] + assmp[(i+1):]
        if not oracle.solve(assumptions=ts):
            assmp = ts
        else:
            i += 1
    return assmp

Demo
```
A less naive MUS extractor

def clset_refine(assmp, oracle):
    assmp = sorted(assmp)
    while True:
        oracle.solve(assumptions=assmp)
        ts = sorted(oracle.get_core())
        if ts == assmp:
            break
        assmp = ts
    return assmp

def main():
    cnf = CNF(from_file=argv[1])  # create a CNF object from file
    (rnv, assumps) = add_assumps(cnf)

    oracle = Solver(name='g3', bootstrap_with=cnf.clauses)

    assumps = clset_refine(assumps, oracle)
    mus = find_mus(assumps, oracle)
    mus = [ref - rnv for ref in mus]
    print("MUS: ", mus)

if __name__ == "__main__":
    main()
A Glimpse of the Future
What next?

- Oracle-based computing
  - Problems beyond NP: optimization, quantification, enumeration, (approximate) counting, decision

- Arms race for proof systems stronger than resolution/clause learning
  - Extended Resolution (and equivalent)
  - Cutting Planes (CP)
  - MaxSAT-inspired proof systems [IMM17, BBI+18]

- Verification of ML models with SAT/SMT
- Scalable explainable AI/ML
  - Deep NNs operate as black-boxes
  - Often important to provide small/intuitive explanations for predictions made
What next?

• Oracle-based computing
  • Problems beyond NP: optimization, quantification, enumeration, (approximate) counting, decision

• Arms race for proof systems stronger than resolution/clause learning
  • Extended Resolution (and equivalent)
  • Cutting Planes (CP)
  • MaxSAT-inspired proof systems

[IMM17, BBQ+ 18]
What next?

- Oracle-based computing
  - Problems beyond NP: optimization, quantification, enumeration, (approximate) counting, decision

- Arms race for proof systems stronger than resolution/clause learning
  - Extended Resolution (and equivalent)
  - Cutting Planes (CP)
  - MaxSAT-inspired proof systems

- Verification of ML models with SAT/SMT
What next?

• Oracle-based computing
  • Problems beyond NP: optimization, quantification, enumeration, (approximate) counting, decision

• Arms race for proof systems stronger than resolution/clause learning
  • Extended Resolution (and equivalent)
  • Cutting Planes (CP)
  • MaxSAT-inspired proof systems

• Verification of ML models with SAT/SMT

• Scalable explainable AI/ML
  • Deep NNs operate as black-boxes
  • Often important to provide small/intuitive explanations for predictions made
What next?

• Oracle-based computing
  - Problems beyond NP: optimization, quantification, enumeration, (approximate) counting, decision

• Arms race for proof systems stronger than resolution/clause learning
  - Extended Resolution (and equivalent)
  - Cutting Planes (CP)
  - MaxSAT-inspired proof systems

• Verification of ML models with SAT/SMT

• Scalable explainable AI/ML
  - Deep NNs operate as black-boxes
  - Often important to provide small/intuitive explanations for predictions made

• ...

[IMM17, BBI+18]
Some final notes

• SAT is a low-level, but very powerful problem solving paradigm
  • PySAT suggests a way to tackle this drawback, but there are others

• There is an ongoing revolution on problem solving with SAT (and SMT) oracles
  • E.g. QBF, model-based diagnosis, explainability, theorem proving, program synthesis, ...

• The use of SAT oracles is impacting problem solving for many different complexity classes
  • With well-known representative problems, e.g. QBF, #SAT, etc.
Some final notes

• SAT is a low-level, but very powerful problem solving paradigm
  • PySAT suggests a way to tackle this drawback, but there are others

• There is an ongoing revolution on problem solving with SAT (and SMT) oracles
  • E.g. QBF, model-based diagnosis, explainability, theorem proving, program synthesis, ...

• The use of SAT oracles is impacting problem solving for many different complexity classes
  • With well-known representative problems, e.g. QBF, #SAT, etc.

• Many fascinating research topics out there!
  • Connections with ML seem unavoidable
Sample of tools

- PySAT
- SAT solvers:
  - MiniSat
  - Glucose
- MaxSAT solvers:
  - RC2
  - MSCG
  - OpenWBO
  - MaxHS
- MUS extractors:
  - MUSer
- MCS extractors:
  - mcsXL
  - LBX
  - MCSls
- Many other tools available from the ReasonLab server
Questions?


[BBI+18] Maria Luisa Bonet, Sam Buss, Alexey Ignatiev, Joao Marques-Silva, and António Morgado.

MaxSAT resolution with the dual rail encoding.


Diagnosing and solving over-determined constraint satisfaction problems.


Using minimal correction sets to more efficiently compute minimal unsatisfiable sets.
In CAV (2), volume 9207 of Lecture Notes in Computer Science, pages 70–86. Springer, 2015.


Towards efficient MUS extraction.


The Sat4j library, release 2.2.
**Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization.**  

**Locating minimal infeasible constraint sets in linear programs.**  

[Coo71] Stephen A. Cook.  
**The complexity of theorem-proving procedures.**  

[CT95] Zhi-Zhong Chen and Seinosuke Toda.  
**The complexity of selecting maximal solutions.**  

**Solving MAXSAT by solving a sequence of simpler SAT instances.**  
**Exploiting the power of MIP solvers in MAXSAT.**  

**Postponing optimization to speed up MAXSAT solving.**  

[DP60] Martin Davis and Hilary Putnam.  
**A computing procedure for quantification theory.**  

**Explanation-based generalisation of failures.**  

[ES03] Niklas Eén and Niklas Sörensson.  
**An extensible SAT-solver.**  


References


*Improvements to core-guided binary search for MaxSAT.*

*MSCG: robust core-guided MaxSAT solving.*

*Minimal sets over monotone predicates in boolean formulae.*

*Efficient model based diagnosis with maximum satisfiability.*

[MMSP09] Vasco M. Manquinho, Joao Marques-Silva, and Jordi Planes. 
*Algorithms for weighted boolean optimization.*
[MP08] Joao Marques-Silva and Jordi Planes.  
*Algorithms for maximum satisfiability using unsatisfiable cores.*  

*On improving MUS extraction algorithms.*  

[NB14] Nina Narodytska and Fahiem Bacchus.  
*Maximum satisfiability using core-guided maxsat resolution.*  

[NIPM18] Nina Narodytska, Alexey Ignatiev, Filipe Pereira, and Joao Marques-Silva.  
*Learning optimal decision trees with SAT.*  

[Rei87] Raymond Reiter.  
*A theory of diagnosis from first principles.*  
References

[Rob65] John Alan Robinson.  
A machine-oriented logic based on the resolution principle.  

[SZGN17] Xujie Si, Xin Zhang, Radu Grigore, and Mayur Naik.  
Maximum satisfiability in software analysis: Applications and techniques.  

Finding guaranteed MUSes fast.  

[ZM03] Lintao Zhang and Sharad Malik.  
Validating SAT solvers using an independent resolution-based checker:  
Practical implementations and other applications.  