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: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$, 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: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$, 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: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$, and parenthesis: $($, $)$
  - Often restricted to Conjunctive Normal Form (CNF)
  - Goal:
    Decide whether formula has a satisfying assignment
- SAT is NP-complete [Coo71]

The CDCL SAT disruption

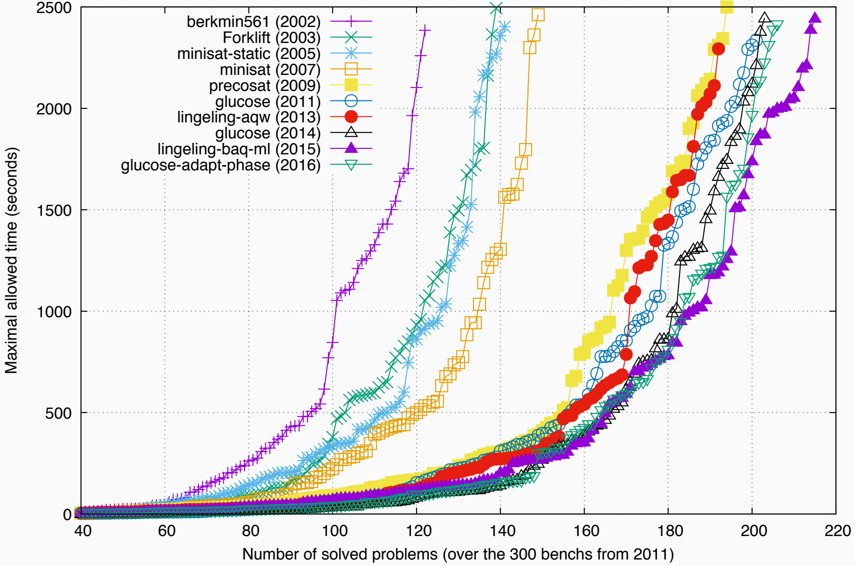
- CDCL SAT solving is a success story of Computer Science

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

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

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

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
```

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!

```
*** 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)

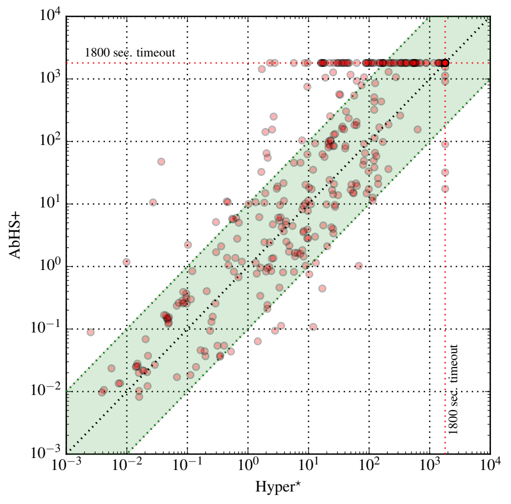
Grasping the search space ...

- 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 ...

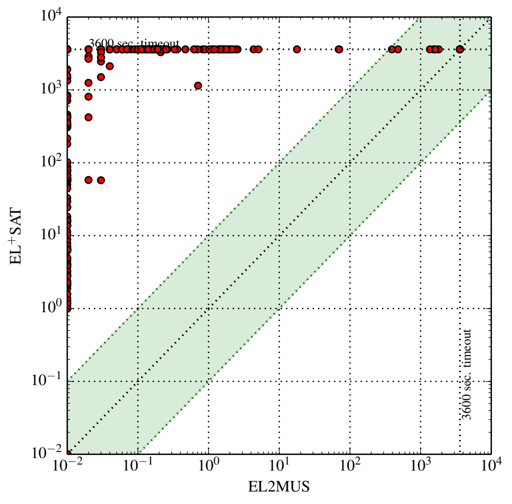
- Number of seconds since the Big Bang: $\approx 10^{17}$
- Number of fundamental particles in observable universe: $\approx 10^{80}$ (or $\approx 10^{85}$)
- Search space with 15775 propositional variables (worst case):
  - # of assignments to 15775 variables: $> 10^{4748}$ **!**
  - **Obs:** SAT solvers 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



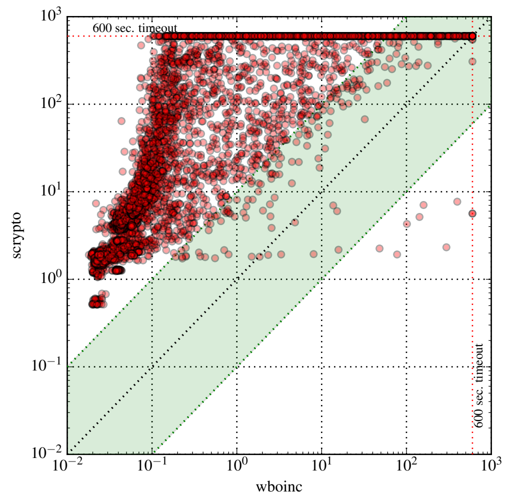
- Propositional abduction instances [IMM16]
  - Implicit hitting set dualization (IHSD)

SAT can make the difference – axiom pinpointing



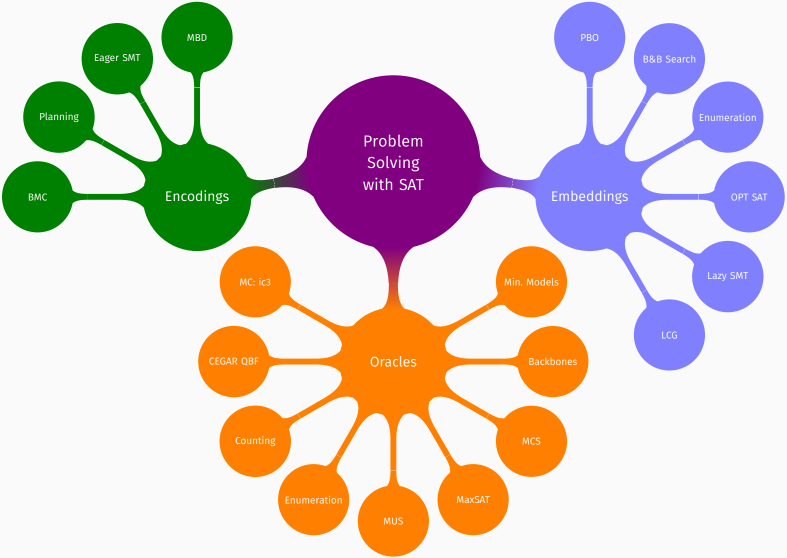
- $\mathcal{EL}^+$ medical ontologies [AMM15]
  - Minimal unsatisfiability (MUSes) & maximal satisfiability (MCSes) & Enumeration

SAT can make the difference – model based diagnosis

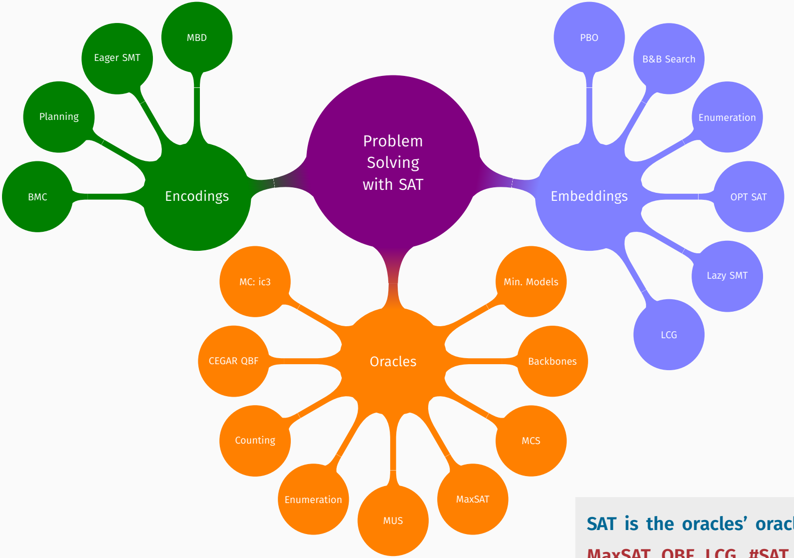


- Model-based diagnosis problem instances [MJIM15]
  - Maximum satisfiability (MaxSAT)

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) — Contact me
  - Minimal Sets over Monotone Predicates (MSMP) — Contact me
  - Enumeration problems
    - MUSes
  - Quantification problems — Contact me
  - (Approximate) counting problems
  - ...

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) Contact me
  - Minimal Sets over Monotone Predicates (MSMP) Contact me
  - Enumeration problems
    - MUSes
  - Quantification problems Contact me
  - (Approximate) counting problems
  - ...
- Part #2: Exploring with SAT oracles
  - Brief introduction to PySAT

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) Contact me
  - Minimal Sets over Monotone Predicates (MSMP) Contact me
  - Enumeration problems
    - MUSes
  - Quantification problems Contact me
  - (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 — A. Biere's talk
  - Clause learning; search restarts; watched literals; VSIDS; ...
- Modeling in propositional logic — Contact me
  - Cardinality constraints; pseudo-boolean constraints; circuits; general constraints; etc.
- Many (high-profile) applications — Contact me
  - Minimal/minimum decision trees/sets [NIPM18, IPNM18]
  - ML model explanations as prime implicants [INMS19]
  - ...

0 Basic Definitions

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

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:

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

  - Example models:

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:

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

  - Example models:
    - $\{r, s, a, b, c, d\}$

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:

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

  - Example models:
    - $\{r, s, a, b, c, d\}$
    - $\{r, s, \bar{x}, y, \bar{w}, z, \bar{a}, b, c, d\}$

Resolution

- Resolution rule: [DP60, Rob65]

$$\frac{(\alpha \vee x) \qquad\qquad (\beta \vee \bar{x})}{(\alpha \vee \beta)}$$

  - Complete proof system for propositional logic

Resolution

- Resolution rule: [DP60, Rob65]

$$\frac{(\alpha \lor x) \qquad\qquad (\beta \lor \bar{x})}{(\alpha \lor \beta)}$$

  - Complete proof system for propositional logic

$$\begin{tikzcd}
(x \lor a) \arrow[d, no head] & (\bar{x} \lor a) \arrow[dl, no head] & (\bar{y} \lor \bar{a}) \arrow[dr, no head] & (y \lor \bar{a}) \arrow[d, no head] \\
(a) \arrow[dr, no head] & & & (\bar{a}) \arrow[dll, no head] \\
& \bot & &
\end{tikzcd}$$

  - Extensively used with (CDCL) SAT solvers

Unit propagation

$$\begin{aligned}\mathcal{F} \quad = \quad & (r) \wedge (\bar{r} \vee s) \wedge \\ & (\bar{w} \vee a) \wedge (\bar{x} \vee \bar{a} \vee b) \wedge \\ & (\bar{y} \vee \bar{z} \vee c) \wedge (\bar{b} \vee \bar{c} \vee d)\end{aligned}$$

Unit propagation

$$\begin{aligned}\mathcal{F} \quad = \quad & (r) \wedge (\bar{r} \vee s) \wedge \\ & (\bar{w} \vee a) \wedge (\bar{x} \vee \bar{a} \vee b) \wedge \\ & (\bar{y} \vee \bar{z} \vee c) \wedge (\bar{b} \vee \bar{c} \vee d)\end{aligned}$$

- Decisions / Variable Branchings:
  $w = 1, x = 1, y = 1, z = 1$

Unit propagation

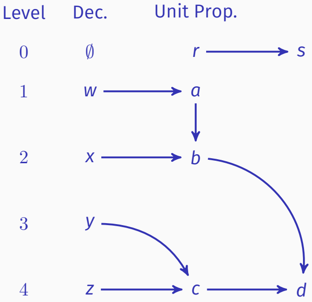
$$\begin{aligned}\mathcal{F} \quad = \quad & (r) \wedge (\bar{r} \vee s) \wedge \\ & (\bar{w} \vee a) \wedge (\bar{x} \vee \bar{a} \vee b) \wedge \\ & (\bar{y} \vee \bar{z} \vee c) \wedge (\bar{b} \vee \bar{c} \vee d)\end{aligned}$$

- 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

$$
\begin{aligned}
\mathcal{F} \quad = \quad & (r) \wedge (\bar{r} \vee s) \wedge \\
& (\bar{w} \vee a) \wedge (\bar{x} \vee \bar{a} \vee b) \wedge \\
& (\bar{y} \vee \bar{z} \vee c) \wedge (\bar{b} \vee \bar{c} \vee d)
\end{aligned}
$$

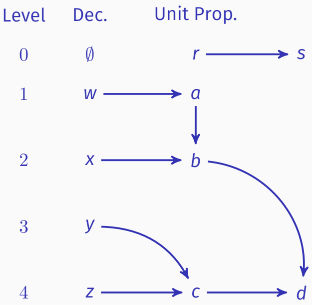


- Decisions / Variable Branchings: $w = 1, x = 1, y = 1, z = 1$

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

Unit propagation

$$\mathcal{F} = (r) \wedge (\bar{r} \vee s) \wedge (\bar{w} \vee a) \wedge (\bar{x} \vee \bar{a} \vee b) \wedge (\bar{y} \vee \bar{z} \vee c) \wedge (\bar{b} \vee \bar{c} \vee d)$$



- 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
    - $(\bar{b} \vee \bar{c} \vee d)$ 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:
  $\mathcal{F} = (\bar{c}) \wedge (\bar{b}) \wedge (\bar{a} \vee c) \wedge (a \vee b) \wedge (a \vee \bar{d}) \wedge (\bar{a} \vee \bar{d})$
- Resolution proof:

$$\begin{tikzcd}[column sep=small]
 & & & (a \vee b) \arrow[dr, no head] & & (\bar{a} \vee c) \arrow[dl, no head] \\
 & & (\bar{c}) \arrow[dr, no head] & & (b \vee c) \arrow[dl, no head] & \\
 & (\bar{b}) \arrow[dr, no head] & & (b) \arrow[dl, no head] & & \\
 & & \bot & & &
\end{tikzcd}$$

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

[ZM03]

Unsatisfiable cores & proof traces

- CNF formula:

$$\mathcal{F} \quad = \quad (\bar{c}) \wedge (\bar{b}) \wedge (\bar{a} \vee c) \wedge (a \vee b) \wedge (a \vee \bar{d}) \wedge (\bar{a} \vee \bar{d})$$

Level	Dec.	Unit Prop.
0	$\emptyset$	$\begin{tikzcd} \bar{b} \arrow[r] & a \arrow[d] \\ \bar{c} \arrow[r] & \perp \end{tikzcd}$

Implication graph with conflict

Unsatisfiable cores & proof traces

- CNF formula:

$$\mathcal{F} = (\bar{c}) \wedge (\bar{b}) \wedge (\bar{a} \vee c) \wedge (a \vee b) \wedge (a \vee \bar{d}) \wedge (\bar{a} \vee \bar{d})$$

Level	Dec.	Unit Prop.
0	$\emptyset$	

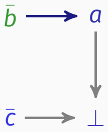
$$\begin{tikzcd} \bar{b} \arrow[r] & a \arrow[d] \\ \bar{c} \arrow[r] & \perp \end{tikzcd}$$

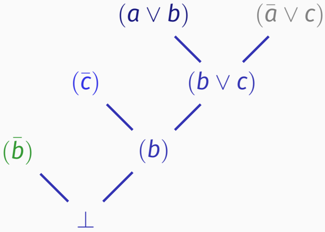
Proof trace $\perp$: $(\bar{a} \vee c)$ $(a \vee b)$ $(\bar{c})$ $(\bar{b})$

Unsatisfiable cores & proof traces

- CNF formula:

$$\mathcal{F} \quad = \quad (\bar{c}) \land (\bar{b}) \land (\bar{a} \lor c) \land (a \lor b) \land (a \lor \bar{d}) \land (\bar{a} \lor \bar{d})$$

Level	Dec.	Unit Prop.
0	$\emptyset$	



Resolution proof follows structure of conflicts

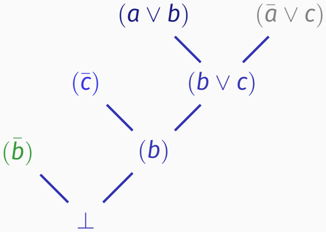
Unsatisfiable cores & proof traces

- CNF formula:

$$\mathcal{F} \quad = \quad (\bar{c}) \wedge (\bar{b}) \wedge (\bar{a} \vee c) \wedge (a \vee b) \wedge (a \vee \bar{d}) \wedge (\bar{a} \vee \bar{d})$$

Level	Dec.	Unit Prop.
0	$\emptyset$	$\bar{b} \to a$, $a \to \bot$, $\bar{c} \to \bot$

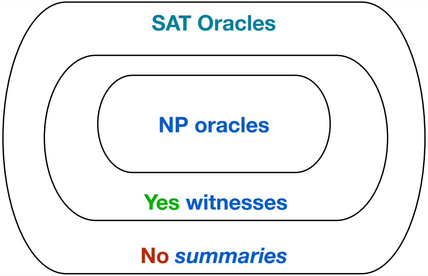
$$\begin{tikzcd} \bar{b} \arrow[r] & a \arrow[d] \\ \bar{c} \arrow[r] & \bot \end{tikzcd}$$



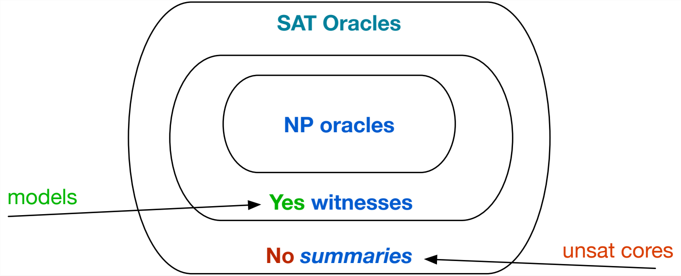
Unsatisfiable subformula (core): $(\bar{c}), (\bar{b}), (\bar{a} \vee c), (a \vee b)$

1 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 $\mathcal{F}_i \wedge (x_i)$
    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.

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

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 $x_i \in \{x_1, \ldots, x_n\} = \mathsf{var}(\mathcal{F})$, 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 $|\mathsf{var}(\mathcal{F})|$ calls to an NP oracle
  - **Note:** Cannot solve FSAT with logarithmic number of NP oracle calls, unless $\mathrm{P} = \mathrm{NP}$ [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 $\mathcal{F}_i \wedge (x_i)$
    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 $\text{P} = \text{NP}$ [GF93]

- FSAT is an example of a function problem
  - **Note:** FSAT can be solved with **one** SAT oracle call

Beyond decision problems

Answer	Problem Type

Beyond decision problems

Answer	Problem Type
Yes/No	Decision Problems

Beyond decision problems

Answer	Problem Type
Yes/No	Decision Problems
Some solution	

Beyond decision problems

Answer	Problem Type
Yes/No	Decision Problems
Some solution	Function Problems

Beyond decision problems

Answer	Problem Type
Yes/No	Decision Problems
Some solution	Function Problems
All solutions	

Beyond decision problems

Answer	Problem Type
Yes/No	Decision Problems
Some solution	Function Problems
All solutions	Enumeration Problems

Beyond decision problems

Answer	Problem Type
Yes/No	Decision Problems
Some solution	Function Problems
All solutions	Enumeration Problems
# solutions	

Beyond decision problems

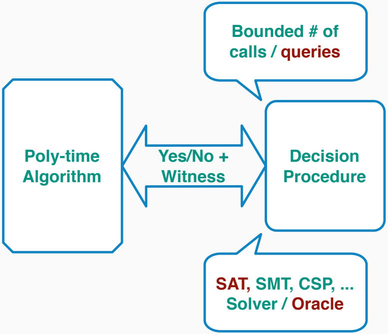
Answer	Problem Type
Yes/No	Decision Problems
Some solution	Function Problems
All solutions	Enumeration Problems
# solutions	Counting Problems

… and beyond NP – decision and function problems

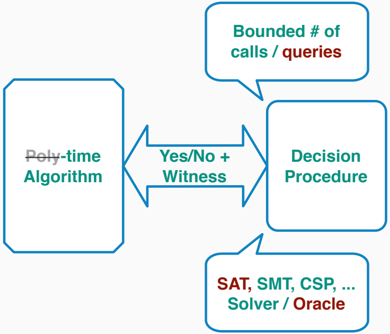
$$
\begin{tikzcd}
& \vdots & \\
\Sigma_3^{\mathsf{p}} & & \Pi_3^{\mathsf{p}} \\
& \Delta_3^{\mathsf{p}} \arrow[ul] \arrow[ur] & \\
\Sigma_2^{\mathsf{p}} \arrow[uu] \arrow[ur] & & \Pi_2^{\mathsf{p}} \arrow[uu] \arrow[ul] \\
& \mathsf{P}^{\mathsf{NP}} = \Delta_2^{\mathsf{p}} \arrow[ul] \arrow[ur] & \\
\mathsf{NP} = \Sigma_1^{\mathsf{p}} \arrow[uu] \arrow[ur] & & \Pi_1^{\mathsf{p}} = \mathsf{coNP} \arrow[uu] \arrow[ul] \\
& \Delta_0^{\mathsf{p}} = \Sigma_0^{\mathsf{p}} = \mathsf{P} = \Pi_0^{\mathsf{p}} = \Delta_1^{\mathsf{p}} \arrow[ul] \arrow[ur] &
\end{tikzcd}
$$

$$
\begin{tikzcd}
& \vdots & \\
\mathsf{F}\Sigma_3^{\mathsf{p}} & & \mathsf{F}\Pi_3^{\mathsf{p}} \\
& \mathsf{F}\Delta_3^{\mathsf{p}} \arrow[ul] \arrow[ur] & \\
\mathsf{F}\Sigma_2^{\mathsf{p}} \arrow[uu] \arrow[ur] & & \mathsf{F}\Pi_2^{\mathsf{p}} \arrow[uu] \arrow[ul] \\
& \mathsf{FP}^{\mathsf{NP}} = \mathsf{F}\Delta_2^{\mathsf{p}} \arrow[ul] \arrow[ur] & \\
\mathsf{FNP} = \mathsf{F}\Sigma_1^{\mathsf{p}} \arrow[uu] \arrow[ur] & & \mathsf{F}\Pi_1^{\mathsf{p}} = \mathsf{coFNP} \arrow[uu] \arrow[ul] \\
& \mathsf{F}\Delta_0^{\mathsf{p}} = \mathsf{F}\Sigma_0^{\mathsf{p}} = \mathsf{FP} = \mathsf{F}\Pi_0^{\mathsf{p}} = \mathsf{F}\Delta_1^{\mathsf{p}} \arrow[ul] \arrow[ur] &
\end{tikzcd}
$$

Oracle-based problem solving – simple scenario



Oracle-based problem solving – general setting



Many problems to solve – within $\mathrm{FP}^{\mathrm{NP}}$

Answer	Problem Type
Yes/No	Decision Problems
Some solution	Function Problems
All solutions	Enumeration Problems

Many problems to solve – within $\mathrm{FP}^{\mathrm{NP}}$

Answer	Problem Type
Yes/No	Decision Problems
Some solution	Function Problems
All solutions	Enumeration Problems

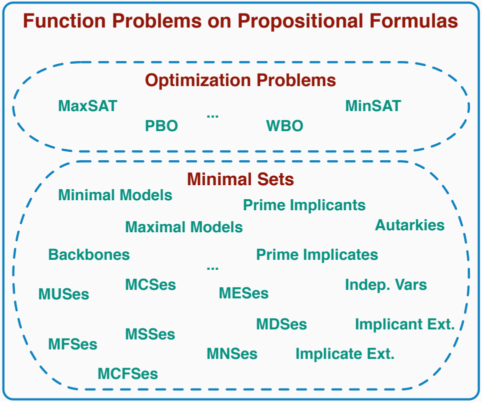
Function Problems on Propositional Formulas

MaxSAT **PBO** **...** **WBO** **MinSAT**

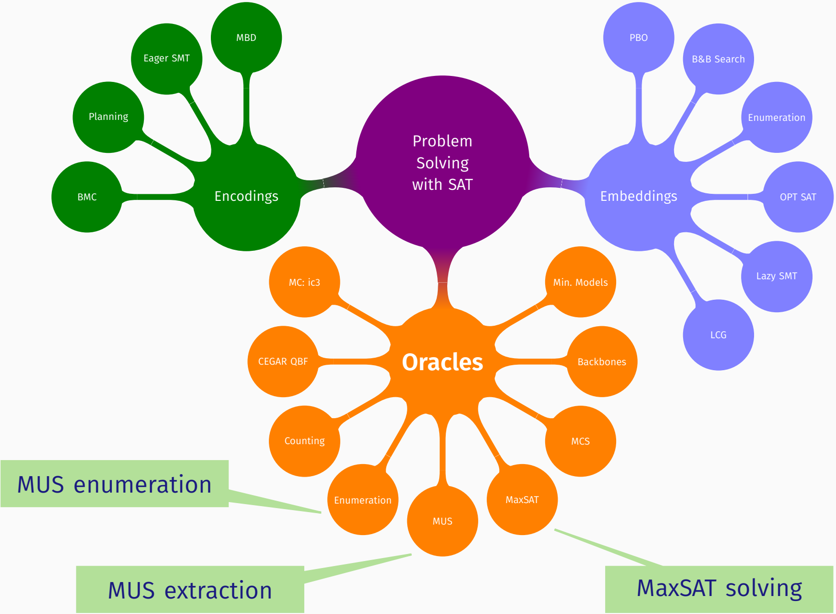
Minimal Models **Prime Implicants** **Maximal Models** **Autarkies** **Backbones** **Prime Implicates** **...** **MUSes** **MCSes** **MESes** **Indep. Vars** **MDSes** **Implicant Ext.** **MFSes** **MSSes** **MNSes** **Implicate Ext.** **MCFSes**

Many problems to solve – within $\mathrm{FP}^{\mathrm{NP}}$

Answer	Problem Type
Yes/No	Decision Problems
Some solution	Function Problems
All solutions	Enumeration Problems



Selection of topics



Outline

Minimal Unsatisfiability

MUS Enumeration

Maximum Satisfiability

Analyzing inconsistency – timetabling

Subject	Day	Time	Room
Intro Prog	Mon	9:00-10:00	6.2.46
Intro AI	Tue	10:00-11:00	8.2.37
Databases	Tue	11:00-12:00	8.2.37
... (hundreds of consistent constraints)			
Linear Alg	Mon	9:00-10:00	6.2.46
Calculus	Tue	10:00-11:00	8.2.37
Adv Calculus	Mon	9:00-10:00	8.2.06
... (hundreds of consistent constraints)			

- Set of constraints consistent / satisfiable?

Analyzing inconsistency – timetabling

Subject	Day	Time	Room
Intro Prog	Mon	9:00-10:00	6.2.46
Intro AI	Tue	10:00-11:00	8.2.37
Databases	Tue	11:00-12:00	8.2.37
... (hundreds of consistent constraints)			
Linear Alg	Mon	9:00-10:00	6.2.46
Calculus	Tue	10:00-11:00	8.2.37
Adv Calculus	Mon	9:00-10:00	8.2.06
... (hundreds of consistent constraints)			

- Set of constraints consistent / satisfiable? No

Analyzing inconsistency – timetabling

Subject	Day	Time	Room
Intro Prog	Mon	9:00-10:00	6.2.46
Intro AI	Tue	10:00-11:00	8.2.37
Databases	Tue	11:00-12:00	8.2.37
... (hundreds of consistent constraints)			
Linear Alg	Mon	9:00-10:00	6.2.46
Calculus	Tue	10:00-11:00	8.2.37
Adv Calculus	Mon	9:00-10:00	8.2.06
... (hundreds of consistent constraints)			

- Set of constraints consistent / satisfiable? No
- Minimal subset of constraints that is inconsistent / unsatisfiable?

Analyzing inconsistency – timetabling

Subject	Day	Time	Room
Intro Prog	Mon	9:00-10:00	6.2.46
Intro AI	Tue	10:00-11:00	8.2.37
Databases	Tue	11:00-12:00	8.2.37
... (hundreds of consistent constraints)			
Linear Alg	Mon	9:00-10:00	6.2.46
Calculus	Tue	10:00-11:00	8.2.37
Adv Calculus	Mon	9:00-10:00	8.2.06
... (hundreds of consistent constraints)			

- Set of constraints consistent / satisfiable? No
- Minimal subset of constraints that is inconsistent / unsatisfiable?

Analyzing inconsistency – timetabling

Subject	Day	Time	Room
Intro Prog	Mon	9:00-10:00	6.2.46
Intro AI	Tue	10:00-11:00	8.2.37
Databases	Tue	11:00-12:00	8.2.37
... (hundreds of consistent constraints)			
Linear Alg	Mon	9:00-10:00	6.2.46
Calculus	Tue	10:00-11:00	8.2.37
Adv Calculus	Mon	9:00-10:00	8.2.06
... (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

Subject	Day	Time	Room
Intro Prog	Mon	9:00-10:00	6.2.46
Intro AI	Tue	10:00-11:00	8.2.37
Databases	Tue	11:00-12:00	8.2.37
… (hundreds of consistent constraints)			
Linear Alg	Mon	9:00-10:00	6.2.46
Calculus	Tue	10:00-11:00	8.2.37
Adv Calculus	Mon	9:00-10:00	8.2.06
… (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

Subject	Day	Time	Room
Intro Prog	Mon	9:00-10:00	6.2.46
Intro AI	Tue	10:00-11:00	8.2.37
Databases	Tue	11:00-12:00	8.2.37
... (hundreds of consistent constraints)			
Linear Alg	Mon	9:00-10:00	6.2.46
Calculus	Tue	10:00-11:00	8.2.37
Adv Calculus	Mon	9:00-10:00	8.2.06
... (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

Subject	Day	Time	Room
Intro Prog	Mon	9:00-10:00	6.2.46
Intro AI	Tue	10:00-11:00	8.2.37
Databases	Tue	11:00-12:00	8.2.37
... (hundreds of consistent constraints)			
Linear Alg	Mon	9:00-10:00	6.2.46
Calculus	Tue	10:00-11:00	8.2.37
Adv Calculus	Mon	9:00-10:00	8.2.06
... (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?

Minimality matters!

Unsatisfiable formulas – MUSes & MCSes

- Given $\mathcal{F}$ $(\vDash \perp)$, $\mathcal{M} \subseteq \mathcal{F}$ is a Minimal Unsatisfiable Subset (MUS) iff $\mathcal{M} \vDash \perp$ and $\forall_{\mathcal{M}' \subsetneq \mathcal{M}}, \mathcal{M}' \nvDash \perp$

$$(\neg x_1 \vee \neg x_2) \wedge (x_1) \wedge (x_2) \wedge (\neg x_3 \vee \neg x_4) \wedge (x_3) \wedge (x_4) \wedge (x_5 \vee x_6)$$

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 $\forall_{\mathcal{M}' \subsetneq \mathcal{M}}, \mathcal{M}' \nvDash \bot$

$$(\neg x_1 \vee \neg x_2) \wedge (x_1) \wedge (x_2) \wedge (\neg x_3 \vee \neg x_4) \wedge (x_3) \wedge (x_4) \wedge (x_5 \vee x_6)$$

Unsatisfiable formulas – MUSes & MCSes

- Given $\mathcal{F}$ $(\vDash \perp)$, $\mathcal{M} \subseteq \mathcal{F}$ is a Minimal Unsatisfiable Subset (MUS) iff $\mathcal{M} \vDash \perp$ and $\forall_{\mathcal{M}' \subsetneq \mathcal{M}}, \mathcal{M}' \nvDash \perp$

$$(\neg x_1 \vee \neg x_2) \wedge (x_1) \wedge (x_2) \wedge (\neg x_3 \vee \neg x_4) \wedge (x_3) \wedge (x_4) \wedge (x_5 \vee x_6)$$

- Given $\mathcal{F}$ $(\vDash \perp)$, $\mathcal{C} \subseteq \mathcal{F}$ is a Minimal Correction Subset (MCS) iff $\mathcal{F} \setminus \mathcal{C} \nvDash \perp$ and $\forall_{\mathcal{C}' \subsetneq \mathcal{C}}, \mathcal{F} \setminus \mathcal{C}' \vDash \perp$. $\mathcal{S} = \mathcal{F} \setminus \mathcal{C}$ is MSS

$$(\neg x_1 \vee \neg x_2) \wedge (x_1) \wedge (x_2) \wedge (\neg x_3 \vee \neg x_4) \wedge (x_3) \wedge (x_4) \wedge (x_5 \vee x_6)$$

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 $\forall_{\mathcal{M}' \subsetneq \mathcal{M}}, \mathcal{M}' \nvDash \bot$

$$(\neg x_1 \vee \neg x_2) \wedge (x_1) \wedge (x_2) \wedge (\neg x_3 \vee \neg x_4) \wedge (x_3) \wedge (x_4) \wedge (x_5 \vee x_6)$$

- Given $\mathcal{F}$ $(\vDash \bot)$, $\mathcal{C} \subseteq \mathcal{F}$ is a Minimal Correction Subset (MCS) iff $\mathcal{F} \setminus \mathcal{C} \nvDash \bot$ and $\forall_{\mathcal{C}' \subsetneq \mathcal{C}}, \mathcal{F} \setminus \mathcal{C}' \vDash \bot$. $\mathcal{S} = \mathcal{F} \setminus \mathcal{C}$ is MSS

$$(\neg x_1 \vee \neg x_2) \wedge (x_1) \wedge (x_2) \wedge (\neg x_3 \vee \neg x_4) \wedge (x_3) \wedge (x_4) \wedge (x_5 \vee x_6)$$

Unsatisfiable formulas – MUSes & MCSes

- Given $\mathcal{F}$ $(\vDash \perp)$, $\mathcal{M} \subseteq \mathcal{F}$ is a Minimal Unsatisfiable Subset (MUS) iff $\mathcal{M} \vDash \perp$ and $\forall_{\mathcal{M}' \subsetneq \mathcal{M}}, \mathcal{M}' \nvDash \perp$

$$(\neg x_1 \vee \neg x_2) \wedge (x_1) \wedge (x_2) \wedge (\neg x_3 \vee \neg x_4) \wedge (x_3) \wedge (x_4) \wedge (x_5 \vee x_6)$$

- Given $\mathcal{F}$ $(\vDash \perp)$, $\mathcal{C} \subseteq \mathcal{F}$ is a Minimal Correction Subset (MCS) iff $\mathcal{F} \setminus \mathcal{C} \nvDash \perp$ and $\forall_{\mathcal{C}' \subsetneq \mathcal{C}}, \mathcal{F} \setminus \mathcal{C}' \vDash \perp$. $\mathcal{S} = \mathcal{F} \setminus \mathcal{C}$ is MSS

$$(\neg x_1 \vee \neg x_2) \wedge (x_1) \wedge (x_2) \wedge (\neg x_3 \vee \neg x_4) \wedge (x_3) \wedge (x_4) \wedge (x_5 \vee x_6)$$

- MUSes and MCSes are (subset-)**minimal** sets
- MUSes and minimal hitting sets of MCSes and vice-versa [Rei87, BS05]
  - Easy to see **why**

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 $\forall_{\mathcal{M}' \subsetneq \mathcal{M}}, \mathcal{M}' \nvDash \bot$

$$(\neg x_1 \vee \neg x_2) \wedge (x_1) \wedge (x_2) \wedge (\neg x_3 \vee \neg x_4) \wedge (x_3) \wedge (x_4) \wedge (x_5 \vee x_6)$$

- Given $\mathcal{F}$ $(\vDash \bot)$, $\mathcal{C} \subseteq \mathcal{F}$ is a Minimal Correction Subset (MCS) iff $\mathcal{F} \setminus \mathcal{C} \nvDash \bot$ and $\forall_{\mathcal{C}' \subsetneq \mathcal{C}}, \mathcal{F} \setminus \mathcal{C}' \vDash \bot$. $\mathcal{S} = \mathcal{F} \setminus \mathcal{C}$ is MSS

$$(\neg x_1 \vee \neg x_2) \wedge (x_1) \wedge (x_2) \wedge (\neg x_3 \vee \neg x_4) \wedge (x_3) \wedge (x_4) \wedge (x_5 \vee x_6)$$

- MUSes and MCSes are (subset-)**minimal** sets
- MUSes and minimal hitting sets of MCSes and vice-versa [Rei87, BS05]
  - Easy to see **why**
- How to compute MUSes & MCSes **efficiently** with SAT oracles?

Why it matters?

- Analysis of over-constrained systems
  - Model-based diagnosis [Rei87]
    - 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 [Rei87]
    - 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; ...
  - ...

Enumeration required!

- 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

Deletion-based algorithm

Input : Set $\mathcal{F}$
Output: Minimal subset $\mathcal{M}$
begin
  $\mathcal{M} \leftarrow \mathcal{F}$
  foreach $c \in \mathcal{M}$ **do**
    if $\neg\mathrm{SAT}(\mathcal{M} \setminus \{c\})$ **then**
      $\mathcal{M} \leftarrow \mathcal{M} \setminus \{c\}$ // If $\neg\mathrm{SAT}(\mathcal{M} \setminus \{c\})$, then $c \notin$ MUS
  return $\mathcal{M}$ // Final $\mathcal{M}$ is MUS
end

- Number of oracles calls: $\mathcal{O}(m)$ [CD91, BDTW93]

Deletion-based algorithm

Input : Set $\mathcal{F}$
Output: Minimal subset $\mathcal{M}$
begin
  $\mathcal{M} \leftarrow \mathcal{F}$
  foreach $c \in \mathcal{M}$ **do**
    if $\neg\mathrm{SAT}(\mathcal{M} \setminus \{c\})$ **then**
      $\mathcal{M} \leftarrow \mathcal{M} \setminus \{c\}$ // Remove c from $\mathcal{M}$
  return $\mathcal{M}$ // Final $\mathcal{M}$ is MUS
end

Monotonicity implicit & essential!

- Number of oracles calls: $\mathcal{O}(m)$ [CD91, BDTW93]

Deletion – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\neg x_1 \vee \neg x_2)$	(x_1)	(x_2)	$(\neg x_3 \vee \neg x_4)$	(x_3)	(x_4)	$(x_5 \vee x_6)$

$\mathcal{M}$	$\mathcal{M} \setminus \{c\}$	$\neg\mathrm{SAT}(\mathcal{M} \setminus \{c\})$	Outcome

Deletion – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\neg x_1 \vee \neg x_2)$	(x_1)	(x_2)	$(\neg x_3 \vee \neg x_4)$	(x_3)	(x_4)	$(x_5 \vee x_6)$

$\mathcal{M}$	$\mathcal{M} \setminus \{c\}$	$\neg\mathrm{SAT}(\mathcal{M} \setminus \{c\})$	Outcome
$c_1..c_7$	$c_2..c_7$	1	Drop c_1

Deletion – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\neg x_1 \vee \neg x_2)$	(x_1)	(x_2)	$(\neg x_3 \vee \neg x_4)$	(x_3)	(x_4)	$(x_5 \vee x_6)$

$\mathcal{M}$	$\mathcal{M} \setminus \{c\}$	$\neg\mathrm{SAT}(\mathcal{M} \setminus \{c\})$	Outcome
$c_1..c_7$	$c_2..c_7$	1	Drop c_1
$c_2..c_7$	$c_3..c_7$	1	Drop c_2

Deletion – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\neg x_1 \vee \neg x_2)$	(x_1)	(x_2)	$(\neg x_3 \vee \neg x_4)$	(x_3)	(x_4)	$(x_5 \vee x_6)$

$\mathcal{M}$	$\mathcal{M} \setminus \{c\}$	$\neg\mathrm{SAT}(\mathcal{M} \setminus \{c\})$	Outcome
$c_1..c_7$	$c_2..c_7$	1	Drop c_1
$c_2..c_7$	$c_3..c_7$	1	Drop c_2
$c_3..c_7$	$c_4..c_7$	1	Drop c_3

Deletion – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\neg x_1 \vee \neg x_2)$	(x_1)	(x_2)	$(\neg x_3 \vee \neg x_4)$	(x_3)	(x_4)	$(x_5 \vee x_6)$

$\mathcal{M}$	$\mathcal{M} \setminus \{c\}$	$\neg\mathrm{SAT}(\mathcal{M} \setminus \{c\})$	Outcome
$c_1..c_7$	$c_2..c_7$	1	Drop c_1
$c_2..c_7$	$c_3..c_7$	1	Drop c_2
$c_3..c_7$	$c_4..c_7$	1	Drop c_3
$c_4..c_7$	$c_5..c_7$	0	Keep c_4

Deletion – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\neg x_1 \vee \neg x_2)$	(x_1)	(x_2)	$(\neg x_3 \vee \neg x_4)$	(x_3)	(x_4)	$(x_5 \vee x_6)$

$\mathcal{M}$	$\mathcal{M} \setminus \{c\}$	$\neg\mathrm{SAT}(\mathcal{M} \setminus \{c\})$	Outcome
$c_1..c_7$	$c_2..c_7$	1	Drop c_1
$c_2..c_7$	$c_3..c_7$	1	Drop c_2
$c_3..c_7$	$c_4..c_7$	1	Drop c_3
$c_4..c_7$	$c_5..c_7$	0	Keep c_4
$c_4..c_7$	$c_4 c_6 c_7$	0	Keep c_5

Deletion – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\neg x_1 \vee \neg x_2)$	(x_1)	(x_2)	$(\neg x_3 \vee \neg x_4)$	(x_3)	(x_4)	$(x_5 \vee x_6)$

$\mathcal{M}$	$\mathcal{M} \setminus \{c\}$	$\neg\mathrm{SAT}(\mathcal{M} \setminus \{c\})$	Outcome
$c_1..c_7$	$c_2..c_7$	1	Drop c_1
$c_2..c_7$	$c_3..c_7$	1	Drop c_2
$c_3..c_7$	$c_4..c_7$	1	Drop c_3
$c_4..c_7$	$c_5..c_7$	0	Keep c_4
$c_4..c_7$	$c_4 c_6 c_7$	0	Keep c_5
$c_4..c_7$	$c_4 c_5 c_7$	0	Keep c_6

Deletion – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\neg x_1 \vee \neg x_2)$	(x_1)	(x_2)	$(\neg x_3 \vee \neg x_4)$	(x_3)	(x_4)	$(x_5 \vee x_6)$

$\mathcal{M}$	$\mathcal{M} \setminus \{c\}$	$\neg\mathrm{SAT}(\mathcal{M} \setminus \{c\})$	Outcome
$c_1..c_7$	$c_2..c_7$	1	Drop c_1
$c_2..c_7$	$c_3..c_7$	1	Drop c_2
$c_3..c_7$	$c_4..c_7$	1	Drop c_3
$c_4..c_7$	$c_5..c_7$	0	Keep c_4
$c_4..c_7$	$c_4c_6c_7$	0	Keep c_5
$c_4..c_7$	$c_4c_5c_7$	0	Keep c_6
$c_4..c_7$	$c_4..c_6$	1	Drop c_7

Deletion – MUS example

c_1	c_2	c_3	c_4	c_5	c_6	c_7
$(\neg x_1 \vee \neg x_2)$	(x_1)	(x_2)	$(\neg x_3 \vee \neg x_4)$	(x_3)	(x_4)	$(x_5 \vee x_6)$

$\mathcal{M}$	$\mathcal{M} \setminus \{c\}$	$\neg\mathrm{SAT}(\mathcal{M} \setminus \{c\})$	Outcome
$c_1..c_7$	$c_2..c_7$	1	Drop c_1
$c_2..c_7$	$c_3..c_7$	1	Drop c_2
$c_3..c_7$	$c_4..c_7$	1	Drop c_3
$c_4..c_7$	$c_5..c_7$	0	Keep c_4
$c_4..c_7$	$c_4 c_6 c_7$	0	Keep c_5
$c_4..c_7$	$c_4 c_5 c_7$	0	Keep c_6
$c_4..c_7$	$c_4..c_6$	1	Drop c_7

- MUS: $\{c_4, c_5, c_6\}$

Many MUS algorithms

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

Algorithm	Oracle Calls	Reference
Insertion-based	$\mathcal{O}(k\,m)$	[dSNP88, vMW08]
MCS_MUS	$\mathcal{O}(k\,m)$	[BK15]
Deletion-based	$\mathcal{O}(m)$	[CD91, BDTW93]
Linear insertion	$\mathcal{O}(m)$	[MSL11, BLM12]
Dichotomic	$\mathcal{O}(k\ \log(m))$	[HLSB06]
QuickXplain	$\mathcal{O}(k + k\ \log(\frac{m}{k}))$	[Jun04]
Progression	$\mathcal{O}(k\ \log(1 + \frac{m}{k}))$	[MJB13]

- **Note:** Lower bound in $\mathrm{FP}_{\|}^{\mathrm{NP}}$ and upper bound in $\mathrm{FP}^{\mathrm{NP}}$ [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?

How to enumerate MUSes?

1. Standard solution:
   Exploit HS duality between MCSes and MUSes [Rei87, LS08]

   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 [Rei87, LS08]

   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 [Rei87, LS08]

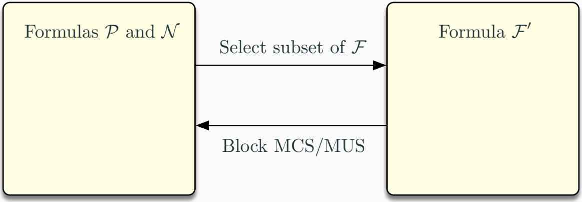
   MCSes are MHSes of MUSes and vice-versa

   - 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

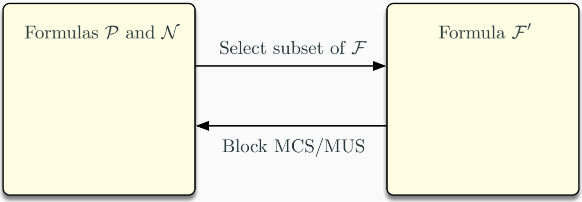
3. Implicit hitting set dualization [LPMM16]
   - Most effective if MUSes provided to user on-demand

How to enumerate MUSes, preferably?



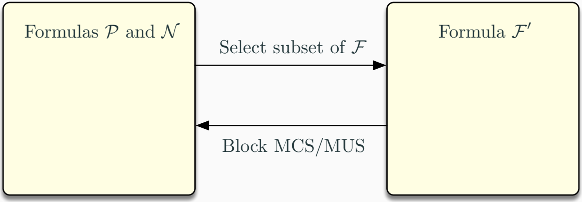
1. Keep sets representing computed MUSes (set $\mathcal{N}$) and MCSes (set $\mathcal{P}$)

How to enumerate MUSes, preferably?



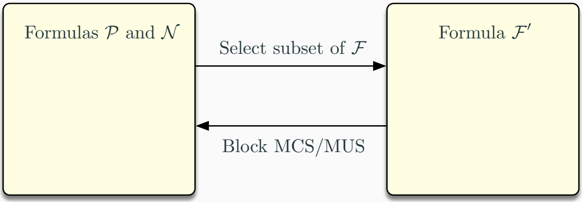
1. Keep sets representing computed MUSes (set $\mathcal{N}$) and MCSes (set $\mathcal{P}$)
2. Compute minimal hitting set (MHS) H of $\mathcal{N}$, subject to $\mathcal{P}$
   - Must **not** repeat MUSes
   - Must **not** repeat MCSes
   - Maximize clauses picked, i.e. prefer to check satisfiability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

How to enumerate MUSes, preferably?



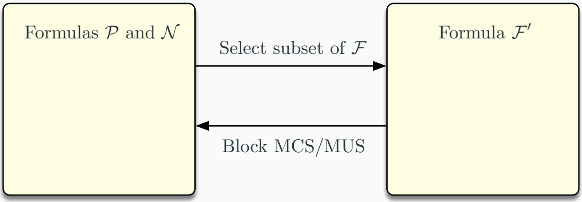
1. Keep sets representing computed MUSes (set $\mathcal{N}$) and MCSes (set $\mathcal{P}$)
2. Compute minimal hitting set (MHS) H of $\mathcal{N}$, subject to $\mathcal{P}$
   - Must **not** repeat MUSes
   - Must **not** repeat MCSes
   - Maximize clauses picked, i.e. prefer to check satisfiability 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

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

MARCO/eMUS algorithm

Input: CNF formula $\mathcal{F}$

begin

  $I \leftarrow \{p_i \mid c_i \in \mathcal{F}\}$

  $(\mathcal{P}, \mathcal{N}) \leftarrow (\emptyset, \emptyset)$

  while true **do**

    $(st, H) \leftarrow \text{MinHittingSet}(\mathcal{N}, \mathcal{P})$

    if not st **then return**

    $\mathcal{F}' \leftarrow \{c_i \mid p_i \in I \land p_i \notin H\}$

    if not $\text{SAT}(\mathcal{F}')$ **then**

      $\mathcal{M} \leftarrow \text{ComputeMUS}(\mathcal{F}')$

      $\text{ReportMUS}\ (\mathcal{M})$

      $\mathcal{N} \leftarrow \mathcal{N} \cup \{\neg p_i \mid c_i \in \mathcal{M}\}$

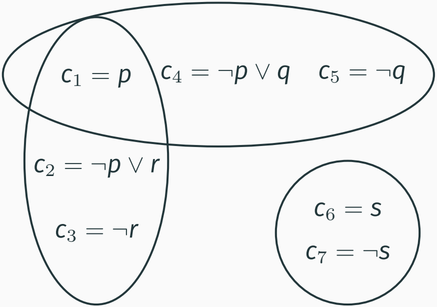
    else

      $\mathcal{P} \leftarrow \mathcal{P} \cup \{p_i \mid p_i \in H\}$

end

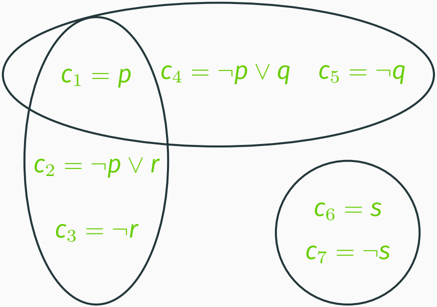
An example

MinHS ($\mathcal{N}$)	$\mathcal{F}'$	MUS/MCS
$p_1p_2p_3p_4p_5p_6p_7$	S/U	
1111111	U	$\neg p_1 \vee \neg p_2 \vee \neg p_3$
0111111	U	$\neg p_6 \vee \neg p_7$
0111101	S	$p_1 \vee p_6$
1011101	U	$\neg p_1 \vee \neg p_4 \vee \neg p_5$
1101010	S	$p_3 \vee p_5 \vee p_7$
1010110	S	$p_2 \vee p_4 \vee p_7$
1100101	S	$p_3 \vee p_4 \vee p_6$
0111110	S	$p_1 \vee p_7$
1101001	S	$p_3 \vee p_5 \vee p_6$
1010101	S	$p_2 \vee p_4 \vee p_6$
1011001	S	$p_2 \vee p_5 \vee p_6$
1100110	S	$p_3 \vee p_4 \vee p_7$
1011010	S	$p_2 \vee p_5 \vee p_7$



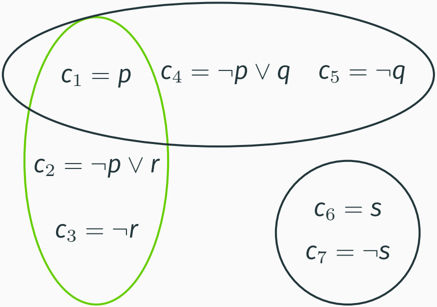
An example

MinHS ($\mathcal{N}$) $p_1p_2p_3p_4p_5p_6p_7$	$\mathcal{F}'$ S/U	MUS/MCS
1111111	U	$\neg p_1 \vee \neg p_2 \vee \neg p_3$
0111111	U	$\neg p_6 \vee \neg p_7$
0111101	S	$p_1 \vee p_6$
1011101	U	$\neg p_1 \vee \neg p_4 \vee \neg p_5$
1101010	S	$p_3 \vee p_5 \vee p_7$
1010110	S	$p_2 \vee p_4 \vee p_7$
1100101	S	$p_3 \vee p_4 \vee p_6$
0111110	S	$p_1 \vee p_7$
1101001	S	$p_3 \vee p_5 \vee p_6$
1010101	S	$p_2 \vee p_4 \vee p_6$
1011001	S	$p_2 \vee p_5 \vee p_6$
1100110	S	$p_3 \vee p_4 \vee p_7$
1011010	S	$p_2 \vee p_5 \vee p_7$



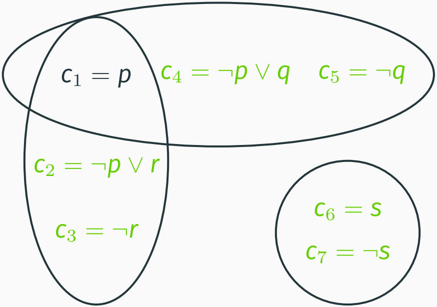
An example

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



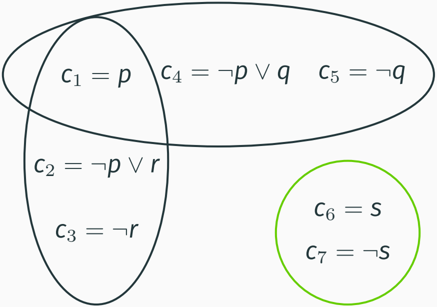
An example

MinHS ($\mathcal{N}$) $p_1p_2p_3p_4p_5p_6p_7$	$\mathcal{F}'$ S/U	MUS/MCS
1111111	U	$\neg p_1 \vee \neg p_2 \vee \neg p_3$
0111111	U	$\neg p_6 \vee \neg p_7$
0111101	S	$p_1 \vee p_6$
1011101	U	$\neg p_1 \vee \neg p_4 \vee \neg p_5$
1101010	S	$p_3 \vee p_5 \vee p_7$
1010110	S	$p_2 \vee p_4 \vee p_7$
1100101	S	$p_3 \vee p_4 \vee p_6$
0111110	S	$p_1 \vee p_7$
1101001	S	$p_3 \vee p_5 \vee p_6$
1010101	S	$p_2 \vee p_4 \vee p_6$
1011001	S	$p_2 \vee p_5 \vee p_6$
1100110	S	$p_3 \vee p_4 \vee p_7$
1011010	S	$p_2 \vee p_5 \vee p_7$



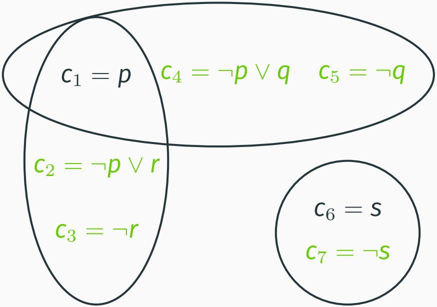
An example

MinHS ($\mathcal{N}$)	$\mathcal{F}'$	MUS/MCS
$p_1p_2p_3p_4p_5p_6p_7$	S/U	
1111111	U	$\neg p_1 \vee \neg p_2 \vee \neg p_3$
0111111	U	$\neg p_6 \vee \neg p_7$
0111101	S	$p_1 \vee p_6$
1011101	U	$\neg p_1 \vee \neg p_4 \vee \neg p_5$
1101010	S	$p_3 \vee p_5 \vee p_7$
1010110	S	$p_2 \vee p_4 \vee p_7$
1100101	S	$p_3 \vee p_4 \vee p_6$
0111110	S	$p_1 \vee p_7$
1101001	S	$p_3 \vee p_5 \vee p_6$
1010101	S	$p_2 \vee p_4 \vee p_6$
1011001	S	$p_2 \vee p_5 \vee p_6$
1100110	S	$p_3 \vee p_4 \vee p_7$
1011010	S	$p_2 \vee p_5 \vee p_7$



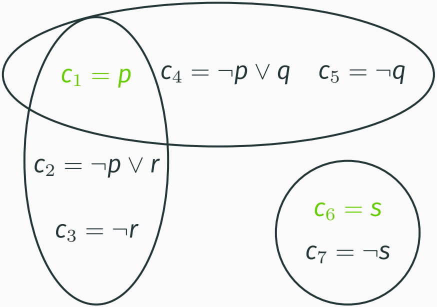
An example

MinHS ($\mathcal{N}$) $p_1p_2p_3p_4p_5p_6p_7$	$\mathcal{F}'$ S/U	MUS/MCS
1111111	U	$\neg p_1 \vee \neg p_2 \vee \neg p_3$
0111111	U	$\neg p_6 \vee \neg p_7$
0111101	S	$p_1 \vee p_6$
1011101	U	$\neg p_1 \vee \neg p_4 \vee \neg p_5$
1101010	S	$p_3 \vee p_5 \vee p_7$
1010110	S	$p_2 \vee p_4 \vee p_7$
1100101	S	$p_3 \vee p_4 \vee p_6$
0111110	S	$p_1 \vee p_7$
1101001	S	$p_3 \vee p_5 \vee p_6$
1010101	S	$p_2 \vee p_4 \vee p_6$
1011001	S	$p_2 \vee p_5 \vee p_6$
1100110	S	$p_3 \vee p_4 \vee p_7$
1011010	S	$p_2 \vee p_5 \vee p_7$



An example

MinHS ($\mathcal{N}$) $p_1p_2p_3p_4p_5p_6p_7$	$\mathcal{F}'$ S/U	MUS/MCS
1111111	U	$\neg p_1 \vee \neg p_2 \vee \neg p_3$
0111111	U	$\neg p_6 \vee \neg p_7$
0111101	S	$p_1 \vee p_6$
1011101	U	$\neg p_1 \vee \neg p_4 \vee \neg p_5$
1101010	S	$p_3 \vee p_5 \vee p_7$
1010110	S	$p_2 \vee p_4 \vee p_7$
1100101	S	$p_3 \vee p_4 \vee p_6$
0111110	S	$p_1 \vee p_7$
1101001	S	$p_3 \vee p_5 \vee p_6$
1010101	S	$p_2 \vee p_4 \vee p_6$
1011001	S	$p_2 \vee p_5 \vee p_6$
1100110	S	$p_3 \vee p_4 \vee p_7$
1011010	S	$p_2 \vee p_5 \vee p_7$



Outline

Minimal Unsatisfiability

MUS Enumeration

Maximum Satisfiability

Recap MaxSAT

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

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

Recap MaxSAT

$\boxed{x_6 \vee x_2}$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\boxed{\neg x_4 \vee x_5}$
$\boxed{x_7 \vee x_5}$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

- 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

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\boxed{\neg x_1}$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$\boxed{x_7 \vee x_5}$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

- 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

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

- 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

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

- 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

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

- 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 [SZGN17]

MaxSAT problem(s)

		Hard Clauses?	
		No	Yes
Weights?	No		
	Yes		

MaxSAT problem(s)

		Hard Clauses?	
		No	Yes
Weights?	No	Plain	Partial
	Yes	Weighted	Weighted Partial

MaxSAT problem(s)

		Hard Clauses? No	Hard Clauses? Yes
Weights?	No	Plain	Partial
	Yes	Weighted	Weighted Partial

- 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)

		Hard Clauses?	
		No	Yes
Weights?	No	Plain	Partial
	Yes	Weighted	Weighted Partial

- 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**

Issues with MaxSAT

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

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

Issues with MaxSAT

- **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:

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

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??

Issues with MaxSAT

- **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??
  - **No!** Enough to either falsify (x) or (z)

Issues with MaxSAT

- **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??
  - **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:

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

  - 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??
  - **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:

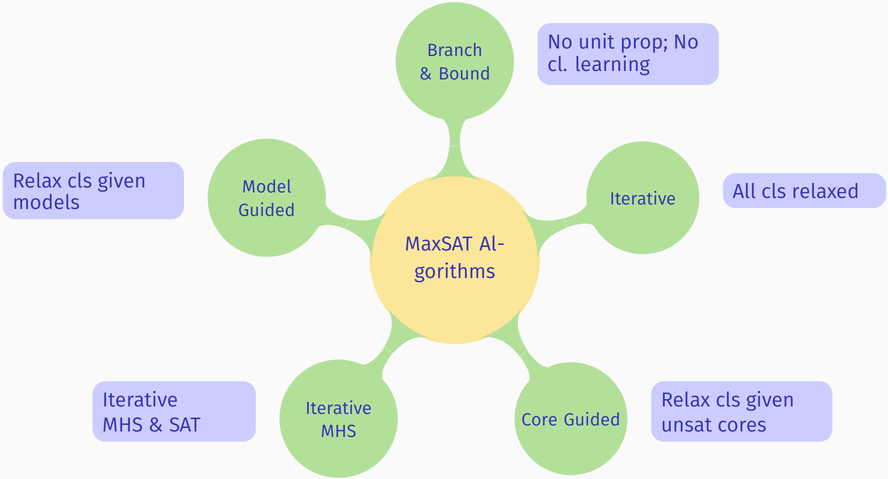
$$\{(x), (\neg x \vee y_1), (\neg x \vee y_2), (\neg y_1 \vee \neg z), (\neg y_2 \vee \neg z), (z)\}$$

  - After unit propagation:

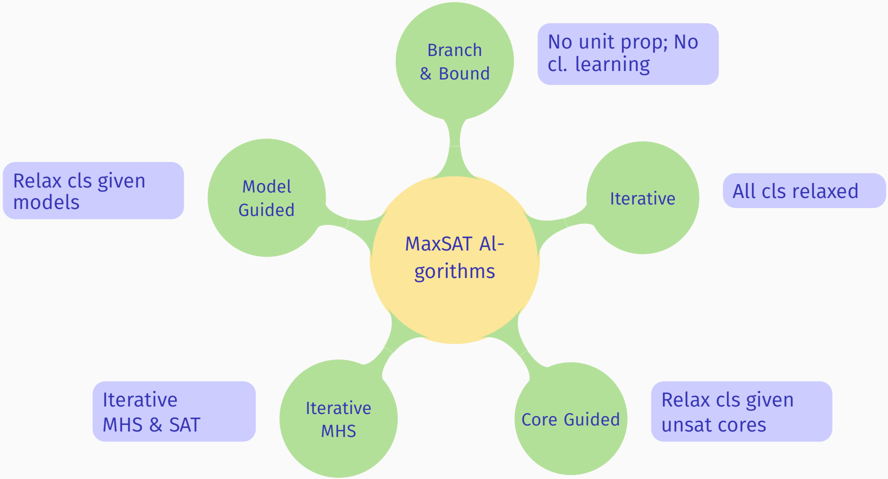
$$\{(x), (\neg x \vee y_1), (\neg x \vee y_2), (\neg y_1 \vee \neg z), (\neg y_2 \vee \neg z), (z)\}$$

  - Is 2 the MaxSAT solution??
  - **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

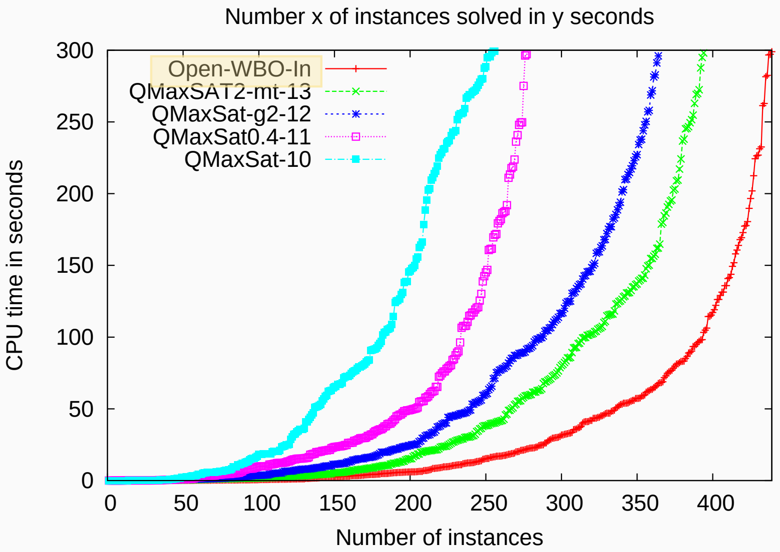


Many MaxSAT approaches



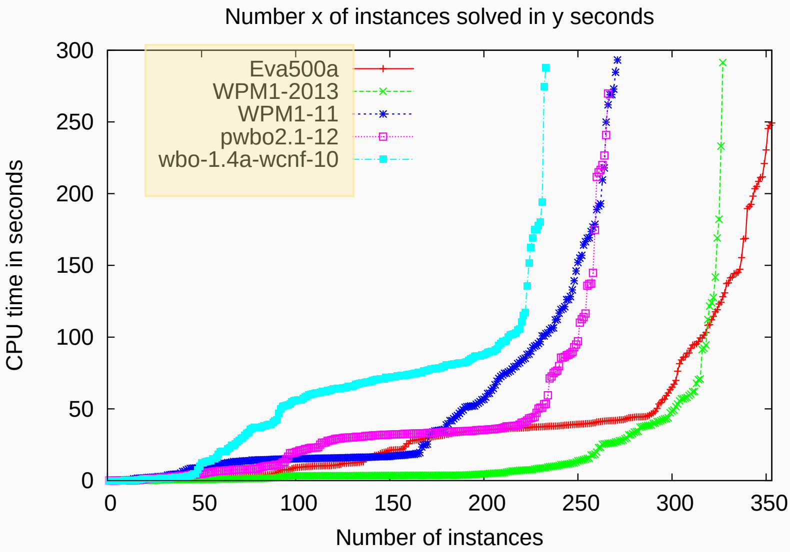
- For practical (**industrial**) instances: **core-guided** & **iterative MHS** approaches are the most effective [MaxSAT14]

Core-guided solver performance – partial



Source: [MaxSAT 2014 organizers]

Core-guided solver performance – weighted partial



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

$x_6 \lor x_2$	$\neg x_6 \lor x_2$	$\neg x_2 \lor x_1$	$\neg x_1$
$\neg x_6 \lor x_8$	$x_6 \lor \neg x_8$	$x_2 \lor x_4$	$\neg x_4 \lor x_5$
$x_7 \lor x_5$	$\neg x_7 \lor x_5$	$\neg x_5 \lor x_3$	$\neg x_3$

Example CNF formula

Basic MaxSAT with iterative SAT solving

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

$\sum_{i=1}^{12} r_i \leq 12$

Relax all clauses; Set $UB = 12 + 1$

Basic MaxSAT with iterative SAT solving

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

$\sum_{i=1}^{12} r_i \leq 12$

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

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

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

Refine $UB = 3$

Basic MaxSAT with iterative SAT solving

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

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

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

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

$\sum_{i=1}^{12} r_i \leq 1$

Refine $UB = 2$

Basic MaxSAT with iterative SAT solving

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

$\sum_{i=1}^{12} r_i \leq 1$

Formula is UNSAT; terminate

Basic MaxSAT with iterative SAT solving

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

$\sum_{i=1}^{12} r_i \leq 1$

MaxSAT solution is last satisfied UB: $UB = 2$

Basic MaxSAT with iterative SAT solving

$x_6 \vee x_2 \vee r_1$ $\quad \neg x_6 \vee x_2 \vee r_2$ $\quad \neg x_2 \vee x_1 \vee r_3$ $\quad \neg x_1 \vee r_4$

$\neg x_6 \vee x_8 \vee r_5$ $\quad x_6 \vee \neg x_8 \vee r_6$ $\quad x_2 \vee x_4 \vee r_7$ $\quad \neg x_4 \vee x_5 \vee r_8$

$x_7 \vee x_5 \vee r_9$ $\quad \neg x_7 \vee x_5 \vee r_{10}$ $\quad \neg x_5 \vee x_3 \vee r_{11}$ $\quad \neg x_3 \vee r_{12}$

$\sum_{i=1}^{12} r_i \leq 1$

MaxSAT solution is last satisfied UB: $UB = 2$

AtMost$k$/PB constraints over **all** relaxation variables

All (possibly many) soft clauses relaxed

Outline

MSU3 core-guided algorithm

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

Example CNF formula

MSU3 core-guided algorithm

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1$	$\neg x_1$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4$	$\neg x_4 \vee x_5$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3$	$\neg x_3$

Formula is UNSAT; OPT $\leq |\varphi| - 1$; Get unsat core

MSU3 core-guided algorithm

$x_6 \vee x_2$ $\neg x_6 \vee x_2$ $\neg x_2 \vee x_1 \vee r_1$ $\neg x_1 \vee r_2$

$\neg x_6 \vee x_8$ $x_6 \vee \neg x_8$ $x_2 \vee x_4 \vee r_3$ $\neg x_4 \vee x_5 \vee r_4$

$x_7 \vee x_5$ $\neg x_7 \vee x_5$ $\neg x_5 \vee x_3 \vee r_5$ $\neg x_3 \vee r_6$

$\sum_{i=1}^{6} r_i \leq 1$

Add relaxation variables and AtMost$k$, $k = 1$, constraint

MSU3 core-guided algorithm

$x_6 \vee x_2$	$\neg x_6 \vee x_2$	$\neg x_2 \vee x_1 \vee r_1$	$\neg x_1 \vee r_2$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5$	$\neg x_7 \vee x_5$	$\neg x_5 \vee x_3 \vee r_5$	$\neg x_3 \vee r_6$

$\sum_{i=1}^{6} r_i \leq 1$

Formula is (again) UNSAT; OPT $\leq |\varphi| - 2$; Get unsat core

MSU3 core-guided algorithm

$x_6 \vee x_2 \vee r_7$	$\neg x_6 \vee x_2 \vee r_8$	$\neg x_2 \vee x_1 \vee r_1$	$\neg x_1 \vee r_2$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5 \vee r_9$	$\neg x_7 \vee x_5 \vee r_{10}$	$\neg x_5 \vee x_3 \vee r_5$	$\neg x_3 \vee r_6$

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

Add new relaxation variables and update AtMost$k$, k=2, constraint

MSU3 core-guided algorithm

$x_6 \vee x_2 \vee r_7$	$\neg x_6 \vee x_2 \vee r_8$	$\neg x_2 \vee x_1 \vee r_1$	$\neg x_1 \vee r_2$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5 \vee r_9$	$\neg x_7 \vee x_5 \vee r_{10}$	$\neg x_5 \vee x_3 \vee r_5$	$\neg x_3 \vee r_6$

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

Instance is now SAT

MSU3 core-guided algorithm

$x_6 \vee x_2 \vee r_7$	$\neg x_6 \vee x_2 \vee r_8$	$\neg x_2 \vee x_1 \vee r_1$	$\neg x_1 \vee r_2$
$\neg x_6 \vee x_8$	$x_6 \vee \neg x_8$	$x_2 \vee x_4 \vee r_3$	$\neg x_4 \vee x_5 \vee r_4$
$x_7 \vee x_5 \vee r_9$	$\neg x_7 \vee x_5 \vee r_{10}$	$\neg x_5 \vee x_3 \vee r_5$	$\neg x_3 \vee r_6$

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

MaxSAT solution is $|\varphi| - \mathcal{I} = 12 - 2 = 10$

MSU3 core-guided algorithm

$x_6 \vee x_2 \vee r_7$ $\quad \neg x_6 \vee x_2 \vee r_8$ $\quad \neg x_2 \vee x_1 \vee r_1$ $\quad \neg x_1 \vee r_2$

$\neg x_6 \vee x_8$ $\quad x_6 \vee \neg x_8$ $\quad x_2 \vee x_4 \vee r_3$ $\quad \neg x_4 \vee x_5 \vee r_4$

$x_7 \vee x_5 \vee r_9$ $\quad \neg x_7 \vee x_5 \vee r_{10}$ $\quad \neg x_5 \vee x_3 \vee r_5$ $\quad \neg x_3 \vee r_6$

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

MaxSAT solution is $|\varphi| - \mathcal{I} = 12 - 2 = 10$

AtMost$k$/PB constraints used

Relaxed soft clauses become **hard**

MSU3 core-guided algorithm

$x_6 \vee x_2 \vee r_7$ $\quad \neg x_6 \vee x_2 \vee r_8$ $\quad \neg x_2 \vee x_1 \vee r_1$ $\quad \neg x_1 \vee r_2$

$\boxed{\neg x_6 \vee x_8 \quad x_6 \vee \neg x_8}$ $\quad x_2 \vee x_4 \vee r_3$ $\quad \neg x_4 \vee x_5 \vee r_4$

$x_7 \vee x_5 \vee r_9$ $\quad \neg x_7 \vee x_5 \vee r_{10}$ $\quad \neg x_5 \vee x_3 \vee r_5$ $\quad \neg x_3 \vee r_6$

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

MaxSAT solution is $|\varphi| - \mathcal{I} = 12 - 2 = 10$

AtMost$k$/PB constraints used

Some clauses not relaxed

Relaxed soft clauses become **hard**

Outline

MHS approach for MaxSAT

$$c_1 = x_6 \vee x_2 \qquad c_2 = \neg x_6 \vee x_2 \qquad c_3 = \neg x_2 \vee x_1 \qquad c_4 = \neg x_1$$

$$c_5 = \neg x_6 \vee x_8 \qquad c_6 = x_6 \vee \neg x_8 \qquad c_7 = x_2 \vee x_4 \qquad c_8 = \neg x_4 \vee x_5$$

$$c_9 = x_7 \vee x_5 \qquad c_{10} = \neg x_7 \vee x_5 \qquad c_{11} = \neg x_5 \vee x_3 \qquad c_{12} = \neg x_3$$

$$\mathcal{K} = \emptyset$$

- Find MHS of $\mathcal{K}$:

MHS approach for MaxSAT

$$c_1 = x_6 \vee x_2 \qquad c_2 = \neg x_6 \vee x_2 \qquad c_3 = \neg x_2 \vee x_1 \qquad c_4 = \neg x_1$$

$$c_5 = \neg x_6 \vee x_8 \qquad c_6 = x_6 \vee \neg x_8 \qquad c_7 = x_2 \vee x_4 \qquad c_8 = \neg x_4 \vee x_5$$

$$c_9 = x_7 \vee x_5 \qquad c_{10} = \neg x_7 \vee x_5 \qquad c_{11} = \neg x_5 \vee x_3 \qquad c_{12} = \neg x_3$$

$$\mathcal{K} = \emptyset$$

- Find MHS of $\mathcal{K}$: $\emptyset$

MHS approach for MaxSAT

$$c_1 = x_6 \vee x_2 \qquad c_2 = \neg x_6 \vee x_2 \qquad c_3 = \neg x_2 \vee x_1 \qquad c_4 = \neg x_1$$

$$c_5 = \neg x_6 \vee x_8 \qquad c_6 = x_6 \vee \neg x_8 \qquad c_7 = x_2 \vee x_4 \qquad c_8 = \neg x_4 \vee x_5$$

$$c_9 = x_7 \vee x_5 \qquad c_{10} = \neg x_7 \vee x_5 \qquad c_{11} = \neg x_5 \vee x_3 \qquad c_{12} = \neg x_3$$

$$\mathcal{K} = \emptyset$$

- Find MHS of $\mathcal{K}$: $\emptyset$
- SAT($\mathcal{F} \setminus \emptyset$)?

MHS approach for MaxSAT

$$c_1 = x_6 \vee x_2 \qquad c_2 = \neg x_6 \vee x_2 \qquad c_3 = \neg x_2 \vee x_1 \qquad c_4 = \neg x_1$$

$$c_5 = \neg x_6 \vee x_8 \qquad c_6 = x_6 \vee \neg x_8 \qquad c_7 = x_2 \vee x_4 \qquad c_8 = \neg x_4 \vee x_5$$

$$c_9 = x_7 \vee x_5 \qquad c_{10} = \neg x_7 \vee x_5 \qquad c_{11} = \neg x_5 \vee x_3 \qquad c_{12} = \neg x_3$$

$$\mathcal{K} = \emptyset$$

- Find MHS of $\mathcal{K}$: $\emptyset$
- SAT($\mathcal{F} \setminus \emptyset$)? No

MHS approach for MaxSAT

$c_1 = x_6 \vee x_2$ $\quad c_2 = \neg x_6 \vee x_2$ $\quad c_3 = \neg x_2 \vee x_1$ $\quad c_4 = \neg x_1$

$c_5 = \neg x_6 \vee x_8$ $\quad c_6 = x_6 \vee \neg x_8$ $\quad c_7 = x_2 \vee x_4$ $\quad c_8 = \neg x_4 \vee x_5$

$c_9 = x_7 \vee x_5$ $\quad c_{10} = \neg x_7 \vee x_5$ $\quad c_{11} = \neg x_5 \vee x_3$ $\quad c_{12} = \neg x_3$

$$\mathcal{K} = \emptyset$$

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

MHS approach for MaxSAT

$$c_1 = x_6 \lor x_2 \qquad c_2 = \neg x_6 \lor x_2 \qquad c_3 = \neg x_2 \lor x_1 \qquad c_4 = \neg x_1$$

$$c_5 = \neg x_6 \lor x_8 \qquad c_6 = x_6 \lor \neg x_8 \qquad c_7 = x_2 \lor x_4 \qquad c_8 = \neg x_4 \lor x_5$$

$$c_9 = x_7 \lor x_5 \qquad c_{10} = \neg x_7 \lor x_5 \qquad c_{11} = \neg x_5 \lor x_3 \qquad c_{12} = \neg x_3$$

$$\mathcal{K} = \{\{c_1, c_2, c_3, c_4\}\}$$

- Find MHS of $\mathcal{K}$: $\emptyset$
- 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

$$c_1 = x_6 \vee x_2 \qquad c_2 = \neg x_6 \vee x_2 \qquad c_3 = \neg x_2 \vee x_1 \qquad c_4 = \neg x_1$$

$$c_5 = \neg x_6 \vee x_8 \qquad c_6 = x_6 \vee \neg x_8 \qquad c_7 = x_2 \vee x_4 \qquad c_8 = \neg x_4 \vee x_5$$

$$c_9 = x_7 \vee x_5 \qquad c_{10} = \neg x_7 \vee x_5 \qquad c_{11} = \neg x_5 \vee x_3 \qquad c_{12} = \neg x_3$$

$$\mathcal{K} = \{\{c_1, c_2, c_3, c_4\}\}$$

- Find MHS of $\mathcal{K}$:

MHS approach for MaxSAT

$$c_1 = x_6 \vee x_2 \qquad c_2 = \neg x_6 \vee x_2 \qquad c_3 = \neg x_2 \vee x_1 \qquad c_4 = \neg x_1$$

$$c_5 = \neg x_6 \vee x_8 \qquad c_6 = x_6 \vee \neg x_8 \qquad c_7 = x_2 \vee x_4 \qquad c_8 = \neg x_4 \vee x_5$$

$$c_9 = x_7 \vee x_5 \qquad c_{10} = \neg x_7 \vee x_5 \qquad c_{11} = \neg x_5 \vee x_3 \qquad c_{12} = \neg x_3$$

$$\mathcal{K} = \{\{c_1, c_2, c_3, c_4\}\}$$

- Find MHS of $\mathcal{K}$: E.g. $\{c_1\}$

MHS approach for MaxSAT

$$c_1 = x_6 \vee x_2 \qquad c_2 = \neg x_6 \vee x_2 \qquad c_3 = \neg x_2 \vee x_1 \qquad c_4 = \neg x_1$$

$$c_5 = \neg x_6 \vee x_8 \qquad c_6 = x_6 \vee \neg x_8 \qquad c_7 = x_2 \vee x_4 \qquad c_8 = \neg x_4 \vee x_5$$

$$c_9 = x_7 \vee x_5 \qquad c_{10} = \neg x_7 \vee x_5 \qquad c_{11} = \neg x_5 \vee x_3 \qquad c_{12} = \neg x_3$$

$$\mathcal{K} = \{\{c_1, c_2, c_3, c_4\}\}$$

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

MHS approach for MaxSAT

$$c_1 = x_6 \vee x_2 \qquad c_2 = \neg x_6 \vee x_2 \qquad c_3 = \neg x_2 \vee x_1 \qquad c_4 = \neg x_1$$

$$c_5 = \neg x_6 \vee x_8 \qquad c_6 = x_6 \vee \neg x_8 \qquad c_7 = x_2 \vee x_4 \qquad c_8 = \neg x_4 \vee x_5$$

$$c_9 = x_7 \vee x_5 \qquad c_{10} = \neg x_7 \vee x_5 \qquad c_{11} = \neg x_5 \vee x_3 \qquad c_{12} = \neg x_3$$

$$\mathcal{K} = \{\{c_1, c_2, c_3, c_4\}\}$$

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

MHS approach for MaxSAT

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

$$\mathcal{K} = \{\{c_1, c_2, c_3, c_4\}\}$$

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

MHS approach for MaxSAT

$$c_1 = x_6 \vee x_2 \qquad c_2 = \neg x_6 \vee x_2 \qquad c_3 = \neg x_2 \vee x_1 \qquad c_4 = \neg x_1$$

$$c_5 = \neg x_6 \vee x_8 \qquad c_6 = x_6 \vee \neg x_8 \qquad c_7 = x_2 \vee x_4 \qquad c_8 = \neg x_4 \vee x_5$$

$$c_9 = x_7 \vee x_5 \qquad c_{10} = \neg x_7 \vee x_5 \qquad c_{11} = \neg x_5 \vee x_3 \qquad c_{12} = \neg x_3$$

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

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

MHS approach for MaxSAT

$$c_1 = x_6 \vee x_2 \qquad c_2 = \neg x_6 \vee x_2 \qquad c_3 = \neg x_2 \vee x_1 \qquad c_4 = \neg x_1$$

$$c_5 = \neg x_6 \vee x_8 \qquad c_6 = x_6 \vee \neg x_8 \qquad c_7 = x_2 \vee x_4 \qquad c_8 = \neg x_4 \vee x_5$$

$$c_9 = x_7 \vee x_5 \qquad c_{10} = \neg x_7 \vee x_5 \qquad c_{11} = \neg x_5 \vee x_3 \qquad c_{12} = \neg x_3$$

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

- Find MHS of $\mathcal{K}$:

MHS approach for MaxSAT

$$c_1 = x_6 \vee x_2 \qquad c_2 = \neg x_6 \vee x_2 \qquad c_3 = \neg x_2 \vee x_1 \qquad c_4 = \neg x_1$$

$$c_5 = \neg x_6 \vee x_8 \qquad c_6 = x_6 \vee \neg x_8 \qquad c_7 = x_2 \vee x_4 \qquad c_8 = \neg x_4 \vee x_5$$

$$c_9 = x_7 \vee x_5 \qquad c_{10} = \neg x_7 \vee x_5 \qquad c_{11} = \neg x_5 \vee x_3 \qquad c_{12} = \neg x_3$$

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

- Find MHS of $\mathcal{K}$: E.g. $\{c_1, c_9\}$

MHS approach for MaxSAT

$$c_1 = x_6 \vee x_2 \qquad c_2 = \neg x_6 \vee x_2 \qquad c_3 = \neg x_2 \vee x_1 \qquad c_4 = \neg x_1$$

$$c_5 = \neg x_6 \vee x_8 \qquad c_6 = x_6 \vee \neg x_8 \qquad c_7 = x_2 \vee x_4 \qquad c_8 = \neg x_4 \vee x_5$$

$$c_9 = x_7 \vee x_5 \qquad c_{10} = \neg x_7 \vee x_5 \qquad c_{11} = \neg x_5 \vee x_3 \qquad c_{12} = \neg x_3$$

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

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

MHS approach for MaxSAT

$c_1 = x_6 \vee x_2$ $\quad c_2 = \neg x_6 \vee x_2$ $\quad c_3 = \neg x_2 \vee x_1$ $\quad c_4 = \neg x_1$

$c_5 = \neg x_6 \vee x_8$ $\quad c_6 = x_6 \vee \neg x_8$ $\quad c_7 = x_2 \vee x_4$ $\quad c_8 = \neg x_4 \vee x_5$

$c_9 = x_7 \vee x_5$ $\quad c_{10} = \neg x_7 \vee x_5$ $\quad c_{11} = \neg x_5 \vee x_3$ $\quad c_{12} = \neg x_3$

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

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

MHS approach for MaxSAT

$$c_1 = x_6 \vee x_2 \qquad c_2 = \neg x_6 \vee x_2 \qquad c_3 = \neg x_2 \vee x_1 \qquad c_4 = \neg x_1$$

$$c_5 = \neg x_6 \vee x_8 \qquad c_6 = x_6 \vee \neg x_8 \qquad c_7 = x_2 \vee x_4 \qquad c_8 = \neg x_4 \vee x_5$$

$$c_9 = x_7 \vee x_5 \qquad c_{10} = \neg x_7 \vee x_5 \qquad c_{11} = \neg x_5 \vee x_3 \qquad c_{12} = \neg x_3$$

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

- Find MHS of $\mathcal{K}$: E.g. $\{c_1, c_9\}$
- 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

$$c_1 = x_6 \vee x_2 \qquad c_2 = \neg x_6 \vee x_2 \qquad c_3 = \neg x_2 \vee x_1 \qquad c_4 = \neg x_1$$

$$c_5 = \neg x_6 \vee x_8 \qquad c_6 = x_6 \vee \neg x_8 \qquad c_7 = x_2 \vee x_4 \qquad c_8 = \neg x_4 \vee x_5$$

$$c_9 = x_7 \vee x_5 \qquad c_{10} = \neg x_7 \vee x_5 \qquad c_{11} = \neg x_5 \vee x_3 \qquad c_{12} = \neg x_3$$

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

- Find MHS of $\mathcal{K}$: E.g. $\{c_1, c_9\}$
- 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

$$c_1 = x_6 \vee x_2 \qquad c_2 = \neg x_6 \vee x_2 \qquad c_3 = \neg x_2 \vee x_1 \qquad c_4 = \neg x_1$$

$$c_5 = \neg x_6 \vee x_8 \qquad c_6 = x_6 \vee \neg x_8 \qquad c_7 = x_2 \vee x_4 \qquad c_8 = \neg x_4 \vee x_5$$

$$c_9 = x_7 \vee x_5 \qquad c_{10} = \neg x_7 \vee x_5 \qquad c_{11} = \neg x_5 \vee x_3 \qquad c_{12} = \neg x_3$$

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

- Find MHS of $\mathcal{K}$:

MHS approach for MaxSAT

$$c_1 = x_6 \vee x_2 \qquad c_2 = \neg x_6 \vee x_2 \qquad c_3 = \neg x_2 \vee x_1 \qquad c_4 = \neg x_1$$

$$c_5 = \neg x_6 \vee x_8 \qquad c_6 = x_6 \vee \neg x_8 \qquad c_7 = x_2 \vee x_4 \qquad c_8 = \neg x_4 \vee x_5$$

$$c_9 = x_7 \vee x_5 \qquad c_{10} = \neg x_7 \vee x_5 \qquad c_{11} = \neg x_5 \vee x_3 \qquad c_{12} = \neg x_3$$

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

- Find MHS of $\mathcal{K}$: E.g. $\{c_4, c_9\}$

MHS approach for MaxSAT

$$c_1 = x_6 \vee x_2 \qquad c_2 = \neg x_6 \vee x_2 \qquad c_3 = \neg x_2 \vee x_1 \qquad c_4 = \neg x_1$$

$$c_5 = \neg x_6 \vee x_8 \qquad c_6 = x_6 \vee \neg x_8 \qquad c_7 = x_2 \vee x_4 \qquad c_8 = \neg x_4 \vee x_5$$

$$c_9 = x_7 \vee x_5 \qquad c_{10} = \neg x_7 \vee x_5 \qquad c_{11} = \neg x_5 \vee x_3 \qquad c_{12} = \neg x_3$$

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

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

MHS approach for MaxSAT

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

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

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

MHS approach for MaxSAT

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

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

- Find MHS of $\mathcal{K}$: E.g. $\{c_4, c_9\}$
- SAT($\mathcal{F} \setminus \{c_4, c_9\}$)? Yes
- Terminate & return 2

MaxSAT solving with SAT oracles – a sample

- A sample of recent algorithms:

Algorithm	# Oracle Queries	Reference
Linear search SU	Exponential***	[BP10]
Binary search	Linear*	[FM06]
FM/WMSU1/WPM1	Exponential**	[FM06, MP08, MMSP09, ABL09, ABGL12]
WPM2	Exponential**	[ABL10, ABL13]
Bin-Core-Dis	Linear	[HMM11, MHM12]
Iterative MHS	Exponential	[DB11, DB13a, DB13b]

* $\mathcal{O}(\log m)$ 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 [IMM+14]
  - Soft cardinality constraints (OLL) [MDM14, MIM14]
    - Recent implementation (RC2, using PySAT) won 2018 MaxSAT Evaluation
  - MaxSAT resolution [NB14]
  - ...

2 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: [ES03]
  - Use activation/selector/indicator variables

Given clause	Added to SAT solver
$\mathfrak{c}_i$	$\mathfrak{c}_i \vee \overline{s}_i$

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: [ES03]
  - Use activation/selector/indicator variables

Given clause	Added to SAT solver
$\mathfrak{c}_i$	$\mathfrak{c}_i \vee \overline{s_i}$

  - 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 activation/selector/indicator variables

Given clause	Added to SAT solver
$\mathfrak{c}_i$	$\mathfrak{c}_i \vee \overline{s_i}$

  - 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: [ES03]
  - Use activation/selector/indicator variables

Given clause	Added to SAT solver
$\mathfrak{c}_i$	$\mathfrak{c}_i \vee \overline{s_i}$

  - 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: [ES03]
  - Use activation/selector/indicator variables

Given clause	Added to SAT solver
$\mathfrak{c}_i$	$\mathfrak{c}_i \vee \overline{s_i}$

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

An example

$$\mathcal{B} = \{(\bar{a} \vee b), (\bar{a} \vee c)\}$$
$$\mathcal{S} = \{(a \vee \overline{s_1}), (\bar{b} \vee \bar{c} \vee \overline{s_2}), (a \vee \bar{c} \vee \overline{s_3}), (a \vee \bar{b} \vee \overline{s_4})\}$$

- Background knowledge $\mathcal{B}$: final clauses, i.e. no indicator variables
- Soft clauses $\mathcal{S}$: add indicator variables $\{s_1, s_2, s_3, s_4\}$

An example

$$\mathcal{B} = \{(\bar{a} \vee b), (\bar{a} \vee c)\}$$
$$\mathcal{S} = \{(a \vee \overline{s_1}), (\bar{b} \vee \bar{c} \vee \overline{s_2}), (a \vee \bar{c} \vee \overline{s_3}), (a \vee \bar{b} \vee \overline{s_4})\}$$

- Background knowledge $\mathcal{B}$: final clauses, i.e. no indicator variables
- Soft clauses $\mathcal{S}$: 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:

$$\mathcal{F} = \{(\bar{a} \vee b), (\bar{a} \vee c), (a), (a \vee \bar{b})\}$$

which is satisfiable

Quiz – what happens in this case?

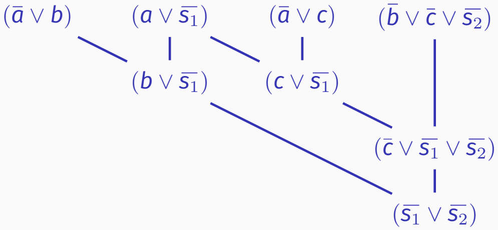
$$\mathcal{B} = \{(\bar{a} \vee b), (\bar{a} \vee c)\}$$
$$\mathcal{S} = \{(a \vee \overline{s_1}), (\bar{b} \vee \bar{c} \vee \overline{s_2}), (a \vee \bar{c} \vee \overline{s_3}), (a \vee \bar{b} \vee \overline{s_4})\}$$

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

Quiz – what happens in this case?

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

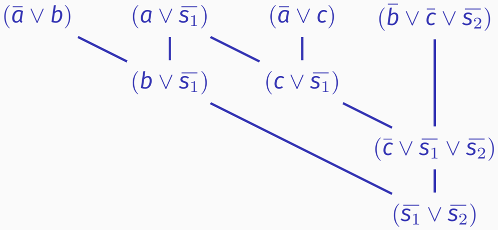
- Given assumptions $\{s_1 = 1, s_2 = 1, s_3 = 1, s_4 = 1\}$?



Quiz – what happens in this case?

$$\mathcal{B} = \{(\bar{a} \vee b), (\bar{a} \vee c)\}$$
$$\mathcal{S} = \{(a \vee \overline{s_1}), (\bar{b} \vee \bar{c} \vee \overline{s_2}), (a \vee \bar{c} \vee \overline{s_3}), (a \vee \bar{b} \vee \overline{s_4})\}$$

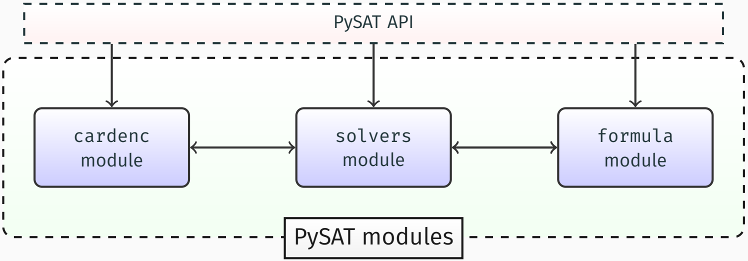
- Given assumptions $\{s_1 = 1, s_2 = 1, s_3 = 1, s_4 = 1\}$?



- Unsatisfiable core: 1^{st} and 2^{nd} clauses of $\mathcal{S}$, given $\mathcal{B}$

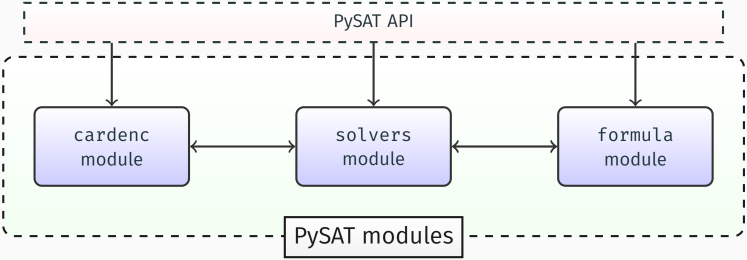
Overview of PySAT

[IMM18]



Overview of PySAT

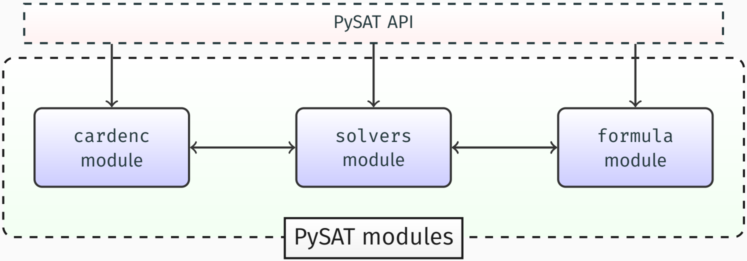
[IMM18]



- Open source, available on github

Overview of PySAT

[IMM18]



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

Available solvers

Solver	Version
Glucose	3.0
Glucose	4.1
Lingeling	bbc-9230380-160707
Minicard	1.2
Minisat	2.2 release
Minisat	GitHub version
MapleCM	SAT competition 2018
Maplesat	MapleCOMSPS_LRB
...	...

- 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

Name	Type
pairwise	AtMost1
bitwise	AtMost1
ladder	AtMost1
sequential counter	AtMost$k$
sorting network	AtMost$k$
cardinality network	AtMost$k$
totalizer	AtMost$k$
mtotalizer	AtMost$k$
kmtotalizer	AtMost$k$

- 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

```
>>> 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

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

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

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 $\neg\mathrm{SAT}(\mathcal{M} \setminus \{c\})$ **then**
      $\mathcal{M} \leftarrow \mathcal{M} \setminus \{c\}$ // If $\neg\mathrm{SAT}(\mathcal{M} \setminus \{c\})$, then $c \notin$ MUS
  return $\mathcal{M}$ // Final $\mathcal{M}$ is MUS
end

- Number of predicate tests: $\mathcal{O}(m)$ [CD91, BDTW93]

Naive MUS extraction I

```
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()
```

Naive MUS extraction II

```python
def add_assumps(cnf):
    rnv = topv = cnf.nv
    assumps = []                        # list of assumptions to use
    for i in range(len(cnf.clauses)):
        topv += 1
        assumps.append(topv)            # register literal
        cnf.clauses[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_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()
```

Naive MUS extraction III

```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
```

Naive MUS extraction III

```
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()
```

3 A Glimpse of the Future

What next?

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

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]

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

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, 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
- ...

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.

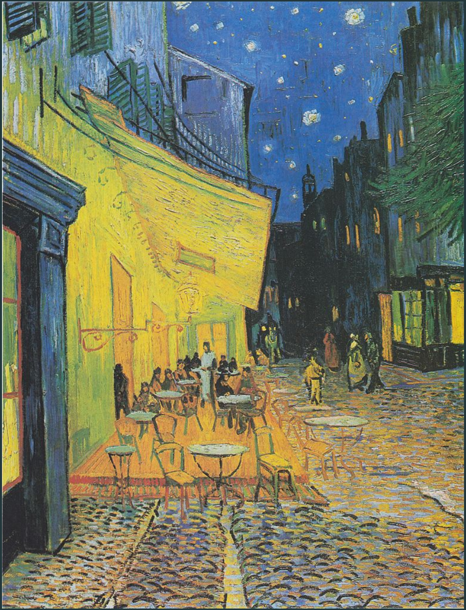
Some final notes

- 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?



References i

[ABGL12] Carlos Ansótegui, Maria Luisa Bonet, Joel Gabàs, and Jordi Levy.
Improving SAT-based weighted MaxSAT solvers.
In *CP*, pages 86–101, 2012.

[ABL09] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy.
Solving (weighted) partial MaxSAT through satisfiability testing.
In *SAT*, pages 427–440, 2009.

[ABL10] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy.
A new algorithm for weighted partial MaxSAT.
In *AAAI*, 2010.

[ABL13] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy.
SAT-based MaxSAT algorithms.
Artif. Intell., 196:77–105, 2013.

[AMM15] M. Fareed Arif, Carlos Mencía, and Joao Marques-Silva.
Efficient MUS enumeration of horn formulae with applications to axiom pinpointing.
In *SAT*, volume 9340 of *Lecture Notes in Computer Science*, pages 324–342. Springer, 2015.

References ii

[BBI$^+$18] Maria Luisa Bonet, Sam Buss, Alexey Ignatiev, Joao Marques-Silva, and António Morgado.
MaxSAT resolution with the dual rail encoding.
In *AAAI*, pages 6565–6572. AAAI Press, 2018.

[BDTW93] R. R. Bakker, F. Dikker, F. Tempelman, and P. M. Wognum.
Diagnosing and solving over-determined constraint satisfaction problems.
In *IJCAI*, pages 276–281, 1993.

[BK15] Fahiem Bacchus and George Katsirelos.
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.

[BLM12] Anton Belov, Inês Lynce, and Joao Marques-Silva.
Towards efficient MUS extraction.
AI Commun., 25(2):97–116, 2012.

[BP10] Daniel Le Berre and Anne Parrain.
The Sat4j library, release 2.2.
JSAT, 7(2-3):59–6, 2010.

References iii

[BS05] James Bailey and Peter J. Stuckey.
Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization.
In *PADL*, pages 174–186, 2005.

[CD91] John W. Chinneck and Erik W. Dravnieks.
Locating minimal infeasible constraint sets in linear programs.
INFORMS Journal on Computing, 3(2):157–168, 1991.

[Coo71] Stephen A. Cook.
The complexity of theorem-proving procedures.
In *STOC*, pages 151–158. ACM, 1971.

[CT95] Zhi-Zhong Chen and Seinosuke Toda.
The complexity of selecting maximal solutions.
Inf. Comput., 119(2):231–239, 1995.

[DB11] Jessica Davies and Fahiem Bacchus.
Solving MAXSAT by solving a sequence of simpler SAT instances.
In *CP*, pages 225–239, 2011.

[DB13a] Jessica Davies and Fahiem Bacchus.
Exploiting the power of MIP solvers in MAXSAT.
In *SAT*, pages 166–181, 2013.

[DB13b] Jessica Davies and Fahiem Bacchus.
Postponing optimization to speed up MAXSAT solving.
In *CP*, pages 247–262, 2013.

[DP60] Martin Davis and Hilary Putnam.
A computing procedure for quantification theory.
J. ACM, 7(3):201–215, 1960.

[dSNP88] J. L. de Siqueira N. and Jean-Francois Puget.
Explanation-based generalisation of failures.
In *ECAI*, pages 339–344, 1988.

[ES03] Niklas Eén and Niklas Sörensson.
An extensible SAT-solver.
In *SAT*, pages 502–518, 2003.

[FM06] Zhaohui Fu and Sharad Malik.
On solving the partial MAX-SAT problem.
In *SAT*, volume 4121 of *Lecture Notes in Computer Science*, pages 252–265. Springer, 2006.

[GF93] Georg Gottlob and Christian G. Fermüller.
Removing redundancy from a clause.
Artif. Intell., 61(2):263–289, 1993.

[HLSB06] Fred Hemery, Christophe Lecoutre, Lakhdar Sais, and Frédéric Boussemart.
Extracting MUCs from constraint networks.
In *ECAI*, pages 113–117, 2006.

[HMM11] Federico Heras, António Morgado, and Joao Marques-Silva.
Core-guided binary search algorithms for maximum satisfiability.
In *AAAI*. AAAI Press, 2011.

[IMM+14] Alexey Ignatiev, António Morgado, Vasco M. Manquinho, Inês Lynce, and Joao Marques-Silva.
Progression in maximum satisfiability.
In *ECAI*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 453–458. IOS Press, 2014.

References vi

[IMM16] Alexey Ignatiev, António Morgado, and Joao Marques-Silva.
Propositional abduction with implicit hitting sets.
In *ECAI*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, pages 1327–1335. IOS Press, 2016.

[IMM17] Alexey Ignatiev, António Morgado, and Joao Marques-Silva.
On tackling the limits of resolution in SAT solving.
In *SAT*, volume 10491 of *Lecture Notes in Computer Science*, pages 164–183. Springer, 2017.

[IMM18] Alexey Ignatiev, António Morgado, and Joao Marques-Silva.
PySAT: A python toolkit for prototyping with SAT oracles.
In *SAT*, volume 10929 of *Lecture Notes in Computer Science*, pages 428–437. Springer, 2018.

[INMS19] Alexey Ignatiev, Nina Narodytska, and Joao Marques-Silva.
Abduction-based explanations for machine learning models.
In *AAAI*, 2019.

[IPNM18] Alexey Ignatiev, Filipe Pereira, Nina Narodytska, and João Marques-Silva.
A SAT-based approach to learn explainable decision sets.
In *IJCAR*, volume 10900 of *Lecture Notes in Computer Science*, pages 627–645. Springer, 2018.

[Jun04] Ulrich Junker.
QUICKXPLAIN: preferred explanations and relaxations for over-constrained problems.
In *AAAI*, pages 167–172, 2004.

[LPMM16] Mark H. Liffiton, Alessandro Previti, Ammar Malik, and Joao Marques-Silva.
Fast, flexible MUS enumeration.
Constraints, 21(2):223–250, 2016.

[LS08] Mark H. Liffiton and Karem A. Sakallah.
Algorithms for computing minimal unsatisfiable subsets of constraints.
J. Autom. Reasoning, 40(1):1–33, 2008.

[MDM14] António Morgado, Carmine Dodaro, and Joao Marques-Silva.
Core-guided MaxSAT with soft cardinality constraints.
In *CP*, volume 8656 of *Lecture Notes in Computer Science*, pages 564–573. Springer, 2014.

[MHM12] António Morgado, Federico Heras, and João Marques-Silva.
Improvements to core-guided binary search for MaxSAT.
In *SAT*, volume 7317 of *Lecture Notes in Computer Science*, pages 284–297. Springer, 2012.

[MIM14] António Morgado, Alexey Ignatiev, and João Marques-Silva.
MSCG: robust core-guided MaxSAT solving.
JSAT, 9:129–134, 2014.

[MJB13] Joao Marques-Silva, Mikolás Janota, and Anton Belov.
Minimal sets over monotone predicates in boolean formulae.
In *CAV*, volume 8044 of *Lecture Notes in Computer Science*, pages 592–607. Springer, 2013.

[MJIM15] Joao Marques-Silva, Mikolás Janota, Alexey Ignatiev, and António Morgado.
Efficient model based diagnosis with maximum satisfiability.
In *IJCAI*, pages 1966–1972. AAAI Press, 2015.

[MMSP09] Vasco M. Manquinho, Joao Marques-Silva, and Jordi Planes.
Algorithms for weighted boolean optimization.
In *SAT*, volume 5584 of *Lecture Notes in Computer Science*, pages 495–508. Springer, 2009.

[MP08] Joao Marques-Silva and Jordi Planes.
Algorithms for maximum satisfiability using unsatisfiable cores.
In *DATE*, pages 408–413. ACM, 2008.

[MSL11] Joao Marques-Silva and Inês Lynce.
On improving MUS extraction algorithms.
In *SAT*, volume 6695 of *Lecture Notes in Computer Science*, pages 159–173. Springer, 2011.

[NB14] Nina Narodytska and Fahiem Bacchus.
Maximum satisfiability using core-guided maxsat resolution.
In *AAAI*, pages 2717–2723. AAAI Press, 2014.

[NIPM18] Nina Narodytska, Alexey Ignatiev, Filipe Pereira, and Joao Marques-Silva.
Learning optimal decision trees with SAT.
In *IJCAI*, pages 1362–1368, 2018.

[Rei87] Raymond Reiter.
A theory of diagnosis from first principles.
Artif. Intell., 32(1):57–95, 1987.

References x

[Rob65] John Alan Robinson.
A machine-oriented logic based on the resolution principle.
J. ACM, 12(1):23–41, 1965.

[SZGN17] Xujie Si, Xin Zhang, Radu Grigore, and Mayur Naik.
Maximum satisfiability in software analysis: Applications and techniques.
In *CAV*, pages 68–94, 2017.

[vMW08] Hans van Maaren and Siert Wieringa.
Finding guaranteed MUSes fast.
In *SAT*, pages 291–304, 2008.

[ZM03] Lintao Zhang and Sharad Malik.
Validating SAT solvers using an independent resolution-based checker: Practical implementations and other applications.
In *DATE*, pages 10880–10885. IEEE Computer Society, 2003.