Practical SAT Solving

Lecture 1

Carsten Sinz, Tomáš Balyo | April 18, 2016
Events

- Lectures
  - Room 236
  - Every Monday at 14:00
  - Please enroll to the lecture on http://campus.studium.kit.edu

- Exercises
  - Room 301
  - Every second Thursday (starting 28.4.) at 14:00

- Exams
  - 25.7. and 19.9.
  - Oral examination
  - Bonus points for homework improve the grade
Contact

- Carsten Sinz
  - email: carsten.sinz@kit.edu
  - room: 028
- Tomas Balyo
  - email: tomas.balyo@kit.edu
  - room: 210
- Homepage
  - url: http://baldur.iti.kit.edu/sat/
  - Contains all the slides and homework assignments
Goals of this lecture

- How do SAT solvers work
  - Algorithms
- How to make a SAT solver
  - Implementation techniques
- How to use a SAT solver efficiently
  - How to encode stuff into CNF
Propositional Logic

- A **Boolean variable** \( x \) is a variable with two possible values: *True* and *False*.
- A **literal** is a Boolean variable \( x \) (positive literal) or its negation \( \overline{x} \) (negative literal).
- A **clause** is a disjunction (or = \( \lor \)) of literals.
- A **CNF**\(^1\) formula is a conjunction (and = \( \land \)) of clauses.

**Example**

\[
F = (\overline{x}_1 \lor x_2) \land (\overline{x}_1 \lor \overline{x}_2 \lor x_3) \land (x_1)
\]

\[\text{Vars}(F) = \{x_1, x_2, x_3\}\]

\[\text{Lits}(F) = \{x_1, \overline{x}_1, x_2, \overline{x}_2, x_3\}\]

\[\text{Cls}(F) = \{(\overline{x}_1 \lor x_2), (\overline{x}_1 \lor \overline{x}_2 \lor x_3), (x_1)\}\]

---

\(^1\)Conjunctive Normal Form
Satisfiability

- A *truth assignment* $\phi$ assigns a truth value (True or False) to each Boolean variable $x$, i.e., $\phi(x) = True$ or $\phi(x) = False$.

- We say that $\phi$ satisfies
  - a positive literal $x$ if $\phi(x) = True$
  - a negative literal $\overline{x}$ if $\phi(x) = False$
  - a clause if it satisfies at least one of its literals
  - a CNF formula if it satisfies all of its clauses

- If $\phi$ satisfies a CNF $F$ then we call $\phi$ a *satisfying assignment* of $F$.
- A formula $F$ is *satisfiable* if there is $\phi$ that satisfies $F$.
- The *Satisfiability Problem* is to determine whether a given formula is satisfiable. If so, we would also like to see a satisfying assignment.
A truth assignment $\phi$ assigns a truth value (True or False) to each Boolean variable $x$, i.e., $\phi(x) = \text{True}$ or $\phi(x) = \text{False}$.

We say that $\phi$ satisfies
- a positive literal $x$ if $\phi(x) = \text{True}$
- a negative literal $\overline{x}$ if $\phi(x) = \text{False}$
- a clause if it satisfies at least one of its literals
- a CNF formula if it satisfies all of its clauses

If $\phi$ satisfies a CNF $F$ then we call $\phi$ a satisfying assignment of $F$.

A formula $F$ is satisfiable if there is $\phi$ that satisfies $F$.

The Satisfiability Problem is to determine whether a given formula is satisfiable. If so, we would also like to see a satisfying assignment.
Satisfiability - Examples

Satisfiable Formulas

\((x_1)\)
\((x_2 \lor x_8 \lor \overline{x_3})\)
\((\overline{x_1} \lor x_2) \land (\overline{x_1} \lor \overline{x_2} \lor x_3) \land (x_1)\)

Unsatisfiable Formulas

\((x_1) \land (\overline{x_1})\)
\((x_1) \land (\overline{x_1}) \land (x_2 \lor x_8 \lor \overline{x_3})\)
\((x_1 \lor x_2) \land (\overline{x_1} \lor x_2) \land (x_1 \lor \overline{x_2}) \land (\overline{x_1} \lor \overline{x_2})\)
Satisfiability - A Practical Example

Scheduling a meeting consider the following constraints:

- Adam can only meet on Monday or Wednesday
- Bridget cannot meet on Wednesday
- Charles cannot meet on Friday
- Darren can only meet on Thursday or Friday

Expressed as SAT:

\[ F = (x_1 \lor x_3) \land (\overline{x_3}) \land (\overline{x_5}) \land (x_4 \lor x_5) \land \text{AtMostOne}(x_1, x_2, x_3, x_4, x_5) \]
Satisifiability - A Practical Example

Scheduling a meeting consider the following constraints

- Adam can only meet on Monday or Wednesday
- Bridget cannot meet on Wednesday
- Charles cannot meet on Friday
- Darren can only meet on Thursday or Friday

Expressed as SAT

\[
F = (x_1 \lor x_3) \land (\overline{x_3}) \land (\overline{x_5}) \land (x_4 \lor x_5) \\
\land (\overline{x_1} \lor \overline{x_2}) \land (x_1 \lor x_3) \land (\overline{x_1} \lor \overline{x_4}) \land (\overline{x_1} \lor \overline{x_5}) \\
\land (\overline{x_2} \lor x_3) \land (\overline{x_2} \lor x_4) \land (x_2 \lor x_5) \\
\land (\overline{x_3} \lor x_4) \land (\overline{x_3} \lor \overline{x_5}) \\
\land (\overline{x_4} \lor \overline{x_5})
\]
Scheduling a meeting consider the following constraints

- Adam can only meet on Monday or Wednesday
- Bridget cannot meet on Wednesday
- Charles cannot meet on Friday
- Darren can only meet on Thursday or Friday

Expressed as SAT

\[ F = (x_1 \lor x_3) \land (\overline{x_3}) \land (\overline{x_5}) \land (x_4 \lor x_5) \land \text{AtMostOne}(x_1, x_2, x_3, x_4, x_5) \]

Solution: Unsatisfiable, i.e., it is impossible to schedule a meeting with these constraints
Satisfiability - Hardness

Satisfiability is NP-Complete [1], proof idea:

- SAT is in NP – easy, checking a solution can be done in linear time
- SAT in NP-hard – encode the run of a non-deterministic Turing machine on an input to a CNF formula.

Consequences:

- We don’t have a polynomial algorithm for SAT (yet) :(  
- If $P \neq NP$ then we won’t have a polynomial algorithm :’(  
- All the known complete algorithms have exponential runtime in the worst case.

Hardness, try it yourself: http://www.cs.utexas.edu/~marijn/game/
Satisfiability - History

- 1960 The first SAT solving algorithm DP (Davis, Putnam) [2]
- 1971 SAT was the first NP-Complete problem [1]
- 1992 The First International SAT Competition, followed by 1993, 1996, since 2002 every year
- 1996 Conflict Driven Clause Learning [5]
- 1996 The First International SAT Conference (Workshop), followed by 1998, since 2000 every year

Early 90’s: 100 variables, 200 clauses, Today: 1,000,000 variables and 5,000,000 clauses.
Applications of SAT solving

- Hardware Model Checking
  - All major hardware companies (Intel, ...) use SAT solver to verify their chip designs

- Software Verification
  - SAT solver based SMT solvers are used to verify Microsoft software products
  - Embedded software in Cars, Airplanes, Refrigerators, ...
  - Unix utilities

- Automated Planning and Scheduling in Artificial Intelligence
  - Still one of the best approaches for optimal planning

- Solving other NP-hard problems (coloring, clique, ...)
SAT Tools – Resolution

The Resolution Rule

\[(l \lor x_1 \lor x_2 \lor \cdots \lor x_n) \land (\overline{l} \lor y_1 \lor y_2 \lor \cdots \lor y_m)\]

\[\overline{(x_1 \lor x_2 \lor \cdots \lor x_n \lor y_1 \lor y_2 \lor \cdots \lor y_m)}\]

The upper two clauses are called Input Clauses the bottom clause is called the Resolvent

Examples

- \[(x_1 \lor x_3 \lor \overline{x_7}) \land (\overline{x_1} \lor x_2) \vdash (x_3 \lor \overline{x_7} \lor x_2)\]
- \[(x_4 \lor x_5) \land (\overline{x_5}) \vdash (x_4)\]
- \[(x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2}) \vdash\]
- \[(x_1) \land (\overline{x_1}) \vdash\]
The Resolution Rule

\[(l \lor x_1 \lor x_2 \lor \cdots \lor x_n) \land (\bar{l} \lor y_1 \lor y_2 \lor \cdots \lor y_m)\]

\[
\frac{(x_1 \lor x_2 \lor \cdots \lor x_n \lor y_1 \lor y_2 \lor \cdots \lor y_m)}{}
\]

The upper two clauses are called *Input Clauses* the bottom clause is called the *Resolvent*

**Examples**

- \((x_1 \lor x_3 \lor \bar{x}_7) \land (\bar{x}_1 \lor x_2) \vdash (x_3 \lor \bar{x}_7 \lor x_2)\)
- \((x_4 \lor x_5) \land (\bar{x}_5) \vdash (x_4)\)
- \((x_1 \lor x_2) \land (\bar{x}_1 \lor \bar{x}_2) \vdash\)
- \((x_1) \land (\bar{x}_1) \vdash\)
The Resolution Rule

\[(l \lor x_1 \lor x_2 \lor \cdots \lor x_n) \land (\bar{l} \lor y_1 \lor y_2 \lor \cdots \lor y_m) \quad \Rightarrow \quad (x_1 \lor x_2 \lor \cdots \lor x_n \lor y_1 \lor y_2 \lor \cdots \lor y_m)\]

The upper two clauses are called *Input Clauses* the bottom clause is called the *Resolvent*

**Examples**

- \((x_1 \lor x_3 \lor \bar{x}_7) \land (\bar{x}_1 \lor x_2) \vdash (x_3 \lor \bar{x}_7 \lor x_2)\)
- \((x_4 \lor x_5) \land (\bar{x}_5) \vdash (x_4)\)
- \((x_1 \lor x_2) \land (\bar{x}_1 \lor \bar{x}_2) \vdash\)
- \((x_1) \land (\bar{x}_1) \vdash\)
The Resolution Rule

\[(l \lor x_1 \lor x_2 \lor \cdots \lor x_n) \land (\bar{l} \lor y_1 \lor y_2 \lor \cdots \lor y_m)\]

\[(x_1 \lor x_2 \lor \cdots \lor x_n \lor y_1 \lor y_2 \lor \cdots \lor y_m)\]

The upper two clauses are called *Input Clauses* the bottom clause is called the *Resolvent*

**Examples**

- \[(x_1 \lor x_3 \lor \overline{x_7}) \land (\overline{x_1} \lor x_2) \vdash (x_3 \lor \overline{x_7} \lor x_2)\]
- \[(x_4 \lor x_5) \land (\overline{x_5}) \vdash (x_4)\]
- \[(x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2}) \vdash\]
- \[(x_1) \land (\overline{x_1}) \vdash\]
The Resolution Rule

\[ \frac{(l \lor x_1 \lor x_2 \lor \cdots \lor x_n) \land (\bar{l} \lor y_1 \lor y_2 \lor \cdots \lor y_m)}{(x_1 \lor x_2 \lor \cdots \lor x_n \lor y_1 \lor y_2 \lor \cdots \lor y_m)} \]

The upper two clauses are called *Input Clauses* the bottom clause is called the *Resolvent*

**Examples**

- \((x_1 \lor x_3 \lor \overline{x_7}) \land (\overline{x_1} \lor x_2) \vdash (x_3 \lor \overline{x_7} \lor x_2)\)
- \((x_4 \lor x_5) \land (\overline{x_5}) \vdash (x_4)\)
- \((x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2}) \vdash \)
- \((x_1) \land (\overline{x_1}) \vdash \)
The Resolution Rule

\[
(l \lor x_1 \lor x_2 \lor \cdots \lor x_n) \land (\overline{l} \lor y_1 \lor y_2 \lor \cdots \lor y_m) \\
\overline{(x_1 \lor x_2 \lor \cdots \lor x_n \lor y_1 \lor y_2 \lor \cdots \lor y_m)}
\]

Special Cases

- **Tautological Resolvent** \((x_1 \lor x_2) \land (\overline{x_1} \lor \overline{x_2}) \vdash (x_1 \lor \overline{x_1})\)
  - Usually forbidden, does no harm, will be useful later

- **Empty Clause** \((x_1) \land (\overline{x_1}) \vdash \bot\)
  - The empty clause a.k.a conflict clause a.k.a ”\(\bot\)” is unsatisfiable

Notation

- \(R((x_1 \lor x_2), (\overline{x_1} \lor x_3)) = (x_2 \lor x_3)\)
Theorem: Resolution maintains satisfiability

Let $F$ be a CNF formula and $C_1$ and $C_2$ two of its clauses with a pair complementary literals. Then $F$ is satisfiable if and only if $F \land R(C_1, C_2)$ is satisfiable.

Proof:

- If $F$ is not satisfiable then $F \land C$ for any $C$ is also not satisfiable.
- If $F$ is satisfiable and $\phi$ is a satisfying assignment of $F$ then we show that $\phi$ also satisfies $R(C_1, C_2)$.
  - If $C_1 = (l \lor P_1)$ and $C_2 = (\overline{l} \lor P_2)$ then $R(C_1, C_2) = (P_1 \lor P_2)$
  - Since $\phi$ satisfies both $C_1$ and $C_2$ it must satisfy at least one of the literals in $P_1$ or $P_2$.
    - if $\phi$ satisfies $l$ then it satisfies some literal in $P_2$
    - if $\phi$ satisfies $\overline{l}$ then it satisfies some literal in $P_1$
Theorem: Resolution maintains satisfiability

Let $F$ be a CNF formula and $C_1$ and $C_2$ two of its clauses with a pair complementary literals. Then $F$ is satisfiable if and only if $F \land R(C_1, C_2)$ is satisfiable.

Proof:

- If $F$ is not satisfiable then $F \land C$ for any $C$ is also not satisfiable.
- If $F$ is satisfiable and $\phi$ is a satisfying assignment of $F$ then we show that $\phi$ also satisfies $R(C_1, C_2)$.
  - If $C_1 = (l \lor P_1)$ and $C_2 = (\bar{l} \lor P_2)$ then $R(C_1, C_2) = (P_1 \lor P_2)$
  - Since $\phi$ satisfies both $C_1$ and $C_2$ it must satisfy at least one of the literals in $P_1$ or $P_2$.
    - if $\phi$ satisfies $l$ then it satisfies some literal in $P_2$
    - if $\phi$ satisfies $\bar{l}$ then it satisfies some literal in $P_1$
Theorem: Resolution maintains satisfiability

Let $F$ be a CNF formula and $C_1$ and $C_2$ two of it's clauses with a pair complementary literals. Then $F$ is satisfiable if and only if $F \land R(C_1, C_2)$ is satisfiable.

Consequences

- If we manage to resolve the empty clause ($\bot$) the original formula is unsatisfiable.

Usage

- Proof of unsatisfiability – Resolution Proof
  - A resolution proof is a sequence of clauses such that each clause is either a clause of the original formula or a resolvent of two previous clauses ending with $\bot$.

Example:

$$(x_1 \lor x_2), (\overline{x_1} \lor x_2), (x_1 \lor \overline{x_2}), (\overline{x_1} \lor \overline{x_2}), (x_2), (\overline{x_2}), \bot$$
Theorem: Resolution maintains satisfiability

Let $F$ be a CNF formula and $C_1$ and $C_2$ two of its clauses with a pair complementary literals. Then $F$ is satisfiable if and only if $F \land R(C_1, C_2)$ is satisfiable.

Consequences

- If we manage to resolve the empty clause ($\perp$) the original formula is unsatisfiable

Usage

- Proof of unsatisfiability – Resolution Proof
  - A resolution proof is a sequence of clauses such that each clause is either a clause of the original formula or a resolvent of two previous clauses ending with $\perp$.

Example: $(x_1 \lor x_2), (\overline{x_1} \lor x_2), (x_1 \lor \overline{x_2}), (\overline{x_1} \lor \overline{x_2}), (x_2), (\overline{x_2}), \perp$
Theorem: Resolution maintains satisfiability

Let $F$ be a CNF formula and $C_1$ and $C_2$ two of its clauses with a pair complementary literals. Then $F$ is satisfiable if and only if $F \land R(C_1, C_2)$ is satisfiable.

Consequences

- If we manage to resolve the empty clause ($\bot$) the original formula is unsatisfiable.

Usage

- Proof of unsatisfiability – Resolution Proof
  
  A resolution proof is a sequence of clauses such that each clause is either a clause of the original formula or a resolvent of two previous clauses ending with $\bot$.

Example: $(x_1 \lor x_2), (\overline{x_1} \lor x_2), (x_1 \lor \overline{x_2}), (\overline{x_1} \lor \overline{x_2}), (x_2), (\overline{x_2}), \bot$
SAT Tools – Resolution

Theorem: Resolution maintains satisfiability

Let $F$ be a CNF formula and $C_1$ and $C_2$ two of its clauses with a pair complementary literals. Then $F$ is satisfiable if and only if $F \land R(C_1, C_2)$ is satisfiable.

Consequences

- If we manage to resolve the empty clause ($\bot$) the original formula is unsatisfiable

Usage

- Proof of unsatisfiability – Resolution Proof
  - A resolution proof is a sequence of clauses such that each clause is either a clause of the original formula or a resolvent of two previous clauses ending with $\bot$.

Example: $(x_1 \lor x_2), (\overline{x_1} \lor x_2), (x_1 \lor \overline{x_2}), (\overline{x_1} \lor \overline{x_2}), (x_2), (\overline{x_2}), \bot$
Saturation Algorithm

- INPUT: CNF formula \( F \)
- OUTPUT: \( \{ SAT, UNSAT \} \)

while (true) do
  \( R = \text{resolveAll}(F) \)
  if \( (R \cap F \neq R) \) then \( F = F \cup R \)
  else break

if \( (\bot \in F) \) then return UNSAT else return SAT

Properties of the saturation algorithm:
- it is sound and complete – always terminates and answers correctly
- has exponential time and space complexity
Saturation Algorithm

**INPUT:** CNF formula $F$

**OUTPUT:** \{SAT, UNSAT\}

while (true) do
  $R = \text{resolveAll}(F)$
  if ($R \cap F \neq R$) then $F = F \cup R$
  else break
  if ($\bot \in F$) then return UNSAT else return SAT

Properties of the saturation algorithm:

- it is sound and complete – always terminates and answers correctly
- has exponential time and space complexity
SAT Tools – Unit Propagation

Unit Resolution

= at least one of the resolved clauses is unit (has one literal).

Example:

\[ R\left((x_1 \lor x_7 \lor \overline{x_2} \lor x_4), (x_2)\right) = (x_1 \lor x_7 \lor x_4) \]

Unit Propagation

= a process of applying unit resolution as long as we get new clauses.

Example:

\[ \begin{align*}
(x_1) \land (x_7 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3) \\
(x_1) \land (x_7 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3) \land (x_3) \\
(x_1) \land (x_7 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3) \land (x_3) \land (x_7 \lor x_2)
\end{align*} \]
SAT Tools – Unit Propagation

Unit Resolution

= at least one of the resolved clauses is unit (has one literal).

Example:
\[ R((x_1 \lor x_7 \lor \overline{x_2} \lor x_4), (x_2)) = (x_1 \lor x_7 \lor x_4) \]

Unit Propagation

= a process of applying unit resolution as long as we get new clauses.

Example:
\[ (x_1) \land (x_7 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3) \]
\[ (x_1) \land (x_7 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3) \land (x_3) \]
\[ (x_1) \land (x_7 \lor x_2 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3) \land (x_3) \land (x_7 \lor x_2) \]
Easy Cases

SAT is not always hard, in the following cases it is polynomially solvable

- 2-SAT
- Horn-SAT
- Hidden Horn-SAT
- SLUR
2-SAT

2-SAT Formula

= each clause has exactly 2 literals.

Example:

- \((x_1 \lor x_3) \land (x_7 \lor \overline{x_3}) \land (\overline{x_1} \lor x_3)\)
- \((x_1 \lor x_2) \land (\overline{x_1} \lor x_2) \land (x_1 \lor \overline{x_2}) \land (\overline{x_1} \lor \overline{x_2})\)

Also called Binary SAT or Quadratic SAT

How to solve 2-SAT?
How to solve 2-SAT?

Saturation Algorithm

The resolution saturation algorithm is polynomial for 2-SAT

Proof:
- Only 2-literal resolvents are possible
- There are only $O(n^2)$ 2-literal clauses on $n$ variables

Complexity:
- Both time and space $O(n^2)$
- There exists a linear algorithm! [6]
Implication Graph

Implication graph of a formula $F$ is an oriented graph that has:

- a vertex for each literal of $F$
- 2 edges for each clause $(l_1 \lor l_2)$
  - $\overline{l}_1 \rightarrow l_2$
  - $\overline{l}_2 \rightarrow l_1$

Example:

$$(\overline{x}_1 \lor x_2) \land (\overline{x}_2 \lor x_3) \land (\overline{x}_3 \lor x_1) \land (x_2 \lor x_4) \land (x_3 \lor x_4) \land (x_1 \lor x_3)$$
The next step is to analyse the *Strongly Connected Components* of the implication graphs

- SCC = there is a path in each direction between each pair
- Tarjan’s algorithm finds SCCs in $O(|V| + |E|)$
- If any $x$ and $\overline{x}$ literal pair is in the same SCC then the formula is UNSAT
  - All the literals in an SCC must be all True or all False

Implication Graph

Organization Definition
Carsten Sinz, Tomáš Balyo – SAT Solving
April 18, 2016
How to find the solution?

- Construct the *Condensation* of the implication graph
  - contract each SCC into one vertex
- Topologically order the vertices of the condensation
- In reverse topological order, if the variables do not already have truth assignments, set all the terms to true.

Example: \( x_1 = x_2 = x_3 = \text{True}, x_4 = \text{True}, \) the rest is already assigned.
How to solve 2-SAT?

Linear Algorithm

- Construct the Implication Graph
- Find all the SCCs
- Check if any SCC contains a complementary pair
- Construct a condensation of the implication graph
- Run topological sort on the condensation
- Construct the solution

Complexity:

- All the steps can be done in linear time
Horn Formula

A CNF formula is a *Horn formula* is each of its clauses contains at most one positive literal.

Example: \((\overline{x}_1 \lor \overline{x}_7 \lor x_3) \land (\overline{x}_2 \lor \overline{x}_4) \land (x_1)\)

Solving Horn Formulas

- Perform unit propagation on the input formula
- If you resolve \(\bot\) then the formula is UNSAT otherwise it is SAT
- Get the solution:
  - Assign the variables in unit clauses to satisfy them
  - Set the rest of the variables to False
Horn Formula

A CNF formula is a *Horn formula* is each of its clauses contains at most one positive literal.

Example: \((\overline{x}_1 \lor \overline{x}_7 \lor x_3) \land (\overline{x}_2 \lor \overline{x}_4) \land (x_1)\)

Solving Horn Formulas

- Perform unit propagation on the input formula
- If you resolve \(\bot\) then the formula is UNSAT otherwise it is SAT
- Get the solution:
  - Assign the variables in unit clauses to satisfy them
  - Set the rest of the variables to False
Mixed Horn

Mixed Horn Formula

A CNF formula is Mixed Horn if it contains only quadratic and Horn clauses.

Example: $(\overline{x}_1 \lor \overline{x}_7 \lor x_3) \land (\overline{x}_2 \lor \overline{x}_4) \land (x_1 \lor x_5) \land (x_3)$

Questions:

- How to solve a Mixed Horn formula?
- How to hard is it to solve a Mixed Horn formula?
Mixed Horn Formula

Mixed Horn SAT solving is NP-complete

Proof:

- We will reduce SAT to Mixed Horn SAT
- For each non-Horn clause \( C = (l_1 \lor l_2 \lor \ldots) \) do
  - for each but one positive \( l_i \in C \) introduce a new variable \( l'_i \)
  - replace \( l_i \) in \( C \) by \( l'_i \)
  - add \((l'_i \lor l_i) \land (l'_i \lor \overline{l_i})\) to establish \( l_i = \overline{l'_i} \)

Example:

\[
(x_1 \lor \overline{x}_7 \lor x_3) \land (\overline{x}_2 \lor \overline{x}_4) \land (x_1 \lor x_5) \Rightarrow
\Rightarrow (x'_1 \lor \overline{x}_7 \lor x_3) \land (\overline{x}_2 \lor \overline{x}_4) \land (x'_1 \lor \overline{x}_5) \land (x'_1 \lor x_1) \land (x'_1 \lor \overline{x}_1)
\]
Hidden/Renamable/Disguised Horn

Hidden Horn Formulas

A CNF formula is *Hidden Horn* if it can be made Horn by renaming some of its variables.

Example:

\[(x_1 \lor x_2 \lor x_4) \land (x_2 \lor \overline{x_4}) \land (x_1) \leadsto (\overline{x_1} \lor \overline{x_2} \lor x_4) \land (\overline{x_2} \lor \overline{x_4}) \land (\overline{x_1})\]

Questions:

- How to recognize a Hidden Horn formula?
- How to hard is it to recognize and solve a Hidden Horn formula?
Recognizing Hidden Horn Formulas

Translate into 2-SAT

Let $F$ be original formula, $R_F$ contains the clause $(l_1 \lor l_2)$ if and only if there is a clause $C \in F$ such that $l_1 \in C$ and $l_2 \in C$.

Example: $F = (x_1 \lor x_2 \lor x_4) \land (x_2 \lor \overline{x}_4) \land (x_1)$  
$R_F = (x_1 \lor x_2) \land (x_1 \lor x_4) \land (x_2 \lor x_4) \land (x_2 \lor \overline{x}_4)$

Recognize Hidden Horn

If $R_F$ is satisfiable, then $F$ is a hidden Horn formula. Furthermore, the satisfying assignment $\phi$ of $R_F$ identifies the variables to be renamed.

- if $x_i = True$ in $\phi$ then $x_i$ needs to be renamed to $\overline{x}_i$


