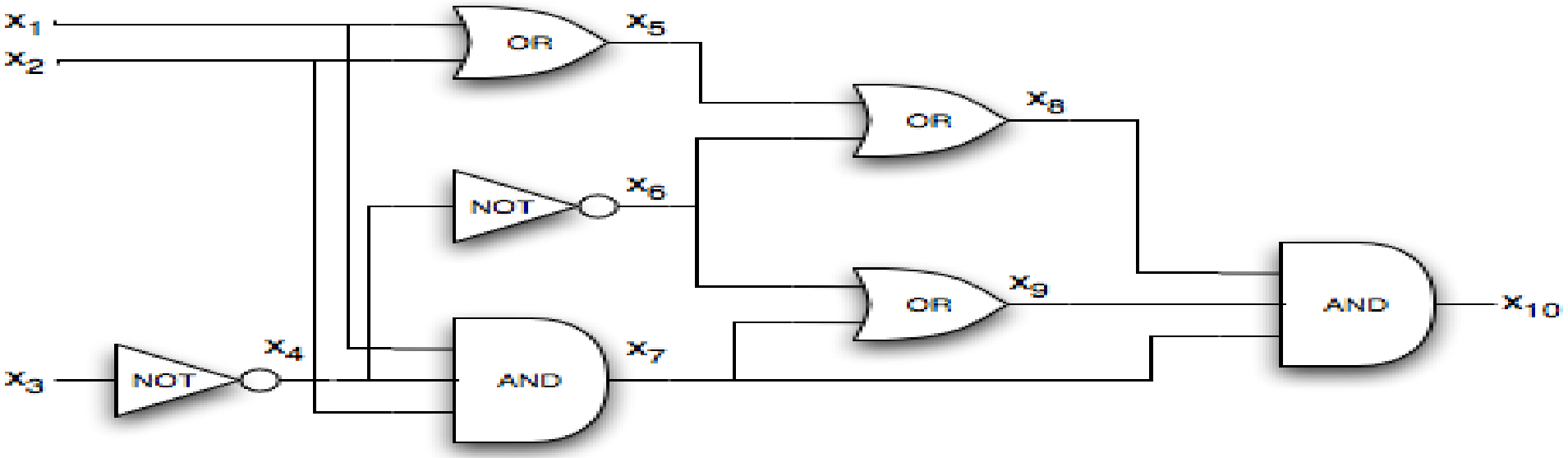
Karlsruhe Institute of Technology

Practical SAT Solving

Lecture 1

Carsten Sinz, Tomáš Balyo | April 18, 2016

NSTITUTE FOR THEORETICAL COMPUTER SCIENCE



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 = $\vee$) of literals.
- A *CNF*[1] *formula* is a conjunction (and = $\wedge$) of clauses.

Example

$F = (\overline{x_1} \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (x_1)$

$Vars(F) = \{x_1, x_2, x_3\}$

$Lits(F) = \{x_1, \overline{x_1}, x_2, \overline{x_2}, x_3\}$

$Cls(F) = \{(\overline{x_1} \vee x_2), (\overline{x_1} \vee \overline{x_2} \vee x_3), (x_1)\}$

[1] Conjunctive Normal Form

Satisfiability

- A *truth assignment* ϕ assigns a truth value (True or False) to each Boolean variable x, i.e., $\phi(x) = \mathit{True}$ or $\phi(x) = \mathit{False}$.
- We say that ϕ satisfies
  - a positive literal x if $\phi(x) = \mathit{True}$
  - a negative literal $\overline{x}$ if $\phi(x) = \mathit{False}$
  - a clause if it satisfies at least one of its literals
  - a CNF formula if it satisfies all of its clauses
- If ϕ satisfies a CNF F then we call ϕ a *satisfying assignment* of F.
- A formula F is *satisfiable* if there is ϕ 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

- A *truth assignment* ϕ assigns a truth value (True or False) to each Boolean variable x, i.e., $\phi(x) = \mathit{True}$ or $\phi(x) = \mathit{False}$.
- We say that ϕ satisfies
  - a positive literal x if $\phi(x) = \mathit{True}$
  - a negative literal $\overline{x}$ if $\phi(x) = \mathit{False}$
  - a clause if it satisfies at least one of its literals
  - a CNF formula if it satisfies all of its clauses
- If ϕ satisfies a CNF F then we call ϕ a *satisfying assignment* of F.
- A formula F is *satisfiable* if there is ϕ 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)

$(x_2 \vee x_8 \vee \overline{x_3})$

$(\overline{x_1} \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (x_1)$

Unsatisfiable Formulas

$(x_1) \wedge (\overline{x_1})$

$(x_1) \wedge (\overline{x_1}) \wedge (x_2 \vee x_8 \vee \overline{x_3})$

$(x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2})$

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

$$F = (x_1 \vee x_3) \wedge (\overline{x_3}) \wedge (\overline{x_5}) \wedge (x_4 \vee x_5) \wedge \text{AtMostOne}(x_1, x_2, x_3, x_4, x_5)$$

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

$$F = (x_1 \vee x_3) \wedge (\overline{x_3}) \wedge (\overline{x_5}) \wedge (x_4 \vee x_5)$$
$$\wedge(\overline{x_1} \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_5})$$
$$\wedge(\overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_2} \vee \overline{x_5})$$
$$\wedge(\overline{x_3} \vee \overline{x_4}) \wedge (\overline{x_3} \vee \overline{x_5})$$
$$\wedge(\overline{x_4} \vee \overline{x_5})$$

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

$$F = (x_1 \vee x_3) \wedge (\overline{x_3}) \wedge (\overline{x_5}) \wedge (x_4 \vee x_5) \wedge \text{AtMostOne}(x_1, x_2, x_3, x_4, x_5)$$

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]
- 1962 An improved version of DP – DPLL [3]
- 1971 SAT was the first NP-Complete problem [1]
- 1992 Local Search SAT solving [4]
- 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.

SAT Conference 2015

Applications of SAT solving

- Hardware Model Checking
  - All major hardware companies (Intel, ...) use SAT sovler to verify their chip desgins
- Software Verification
  - SAT solver based SMT solvers are used to verify Microsoft software products
  - Embedded software in Cars, Aiplanes, 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

$$\frac{(l \vee x_1 \vee x_2 \vee \cdots \vee x_n) \wedge (\bar{l} \vee y_1 \vee y_2 \vee \cdots \vee y_m)}{(x_1 \vee x_2 \vee \cdots \vee x_n \vee y_1 \vee y_2 \vee \cdots \vee y_m)}$$

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

Examples

- $(x_1 \vee x_3 \vee \overline{x_7}) \wedge (\overline{x_1} \vee x_2) \vdash (x_3 \vee \overline{x_7} \vee x_2)$
- $(x_4 \vee x_5) \wedge (\overline{x_5}) \vdash (x_4)$
- $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2}) \vdash$
- $(x_1) \wedge (\overline{x_1}) \vdash$

SAT Tools – Resolution

The Resolution Rule

$$\frac{(l \vee x_1 \vee x_2 \vee \cdots \vee x_n) \wedge (\bar{l} \vee y_1 \vee y_2 \vee \cdots \vee y_m)}{(x_1 \vee x_2 \vee \cdots \vee x_n \vee y_1 \vee y_2 \vee \cdots \vee y_m)}$$

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

Examples

- $(x_1 \vee x_3 \vee \overline{x_7}) \wedge (\overline{x_1} \vee x_2) \vdash (x_3 \vee \overline{x_7} \vee x_2)$
- $(x_4 \vee x_5) \wedge (\overline{x_5}) \vdash (x_4)$
- $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2}) \vdash$
- $(x_1) \wedge (\overline{x_1}) \vdash$

SAT Tools – Resolution

The Resolution Rule

$$\frac{(l \vee x_1 \vee x_2 \vee \cdots \vee x_n) \wedge (\bar{l} \vee y_1 \vee y_2 \vee \cdots \vee y_m)}{(x_1 \vee x_2 \vee \cdots \vee x_n \vee y_1 \vee y_2 \vee \cdots \vee y_m)}$$

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

Examples

- $(x_1 \vee x_3 \vee \overline{x_7}) \wedge (\overline{x_1} \vee x_2) \vdash (x_3 \vee \overline{x_7} \vee x_2)$
- $(x_4 \vee x_5) \wedge (\overline{x_5}) \vdash (x_4)$
- $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2}) \vdash$
- $(x_1) \wedge (\overline{x_1}) \vdash$

SAT Tools – Resolution

The Resolution Rule

$$\frac{(l \vee x_1 \vee x_2 \vee \cdots \vee x_n) \wedge (\bar{l} \vee y_1 \vee y_2 \vee \cdots \vee y_m)}{(x_1 \vee x_2 \vee \cdots \vee x_n \vee y_1 \vee y_2 \vee \cdots \vee y_m)}$$

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

Examples

- $(x_1 \vee x_3 \vee \overline{x_7}) \wedge (\overline{x_1} \vee x_2) \vdash (x_3 \vee \overline{x_7} \vee x_2)$
- $(x_4 \vee x_5) \wedge (\overline{x_5}) \vdash (x_4)$
- $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2}) \vdash$
- $(x_1) \wedge (\overline{x_1}) \vdash$

SAT Tools – Resolution

The Resolution Rule

$$\frac{(l \vee x_1 \vee x_2 \vee \cdots \vee x_n) \wedge (\bar{l} \vee y_1 \vee y_2 \vee \cdots \vee y_m)}{(x_1 \vee x_2 \vee \cdots \vee x_n \vee y_1 \vee y_2 \vee \cdots \vee y_m)}$$

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

Examples

- $(x_1 \vee x_3 \vee \overline{x_7}) \wedge (\overline{x_1} \vee x_2) \vdash (x_3 \vee \overline{x_7} \vee x_2)$
- $(x_4 \vee x_5) \wedge (\overline{x_5}) \vdash (x_4)$
- $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2}) \vdash$
- $(x_1) \wedge (\overline{x_1}) \vdash$

SAT Tools – Resolution

The Resolution Rule

$$\frac{(l \vee x_1 \vee x_2 \vee \cdots \vee x_n) \wedge (\bar{l} \vee y_1 \vee y_2 \vee \cdots \vee y_m)}{(x_1 \vee x_2 \vee \cdots \vee x_n \vee y_1 \vee y_2 \vee \cdots \vee y_m)}$$

Special Cases

- Tautological Resolvent $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2}) \vdash (x_1 \vee \overline{x_1})$
  - Usually forbidden, does no harm, will be useful later
- Empty Clause $(x_1) \wedge (\overline{x_1}) \vdash \bot$
  - The empty clause a.k.a conflict clause a.k.a "$\bot$" is unsatisfiable

Notation

- $\mathrm{R}((x_1 \vee x_2), (\overline{x_1} \vee x_3)) = (x_2 \vee x_3)$

SAT Tools – Resolution

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 $F \wedge \mathrm{R}(C_1, C_2)$ is satisfiable.

Proof:

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

SAT Tools – Resolution

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 $F \wedge \mathrm{R}(C_1, C_2)$ is satisfiable.

Proof:

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

SAT Tools – Resolution

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 $F \wedge \mathrm{R}(C_1, C_2)$ 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: $(x_1 \vee x_2), (\overline{x_1} \vee x_2), (x_1 \vee \overline{x_2}), (\overline{x_1} \vee \overline{x_2}), (x_2), (\overline{x_2}), \bot$

SAT Tools – Resolution

Theorem: Resolution maintains satisfiability

Let F be a CNF formula and C_1 and C_2 two of it's clauses with a pair complementary literals. Then F is satisfiable if and only if $F \wedge \mathrm{R}(C_1, C_2)$ 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: $(x_1 \vee x_2), (\overline{x_1} \vee x_2), (x_1 \vee \overline{x_2}), (\overline{x_1} \vee \overline{x_2}), (x_2), (\overline{x_2}), \bot$

SAT Tools – Resolution

Theorem: Resolution maintains satisfiability

Let F be a CNF formula and C_1 and C_2 two of it's clauses with a pair complementary literals. Then F is satisfiable if and only if $F \wedge \mathrm{R}(C_1, C_2)$ 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: $(x_1 \vee x_2), (\overline{x_1} \vee x_2), (x_1 \vee \overline{x_2}), (\overline{x_1} \vee \overline{x_2})$, $(x_2), (\overline{x_2}), \bot$

SAT Tools – Resolution

Theorem: Resolution maintains satisfiability

Let F be a CNF formula and C_1 and C_2 two of it's clauses with a pair complementary literals. Then F is satisfiable if and only if $F \wedge \mathrm{R}(C_1, C_2)$ 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: $(x_1 \vee x_2), (\overline{x_1} \vee x_2), (x_1 \vee \overline{x_2}), (\overline{x_1} \vee \overline{x_2}), (x_2), (\overline{x_2}), \bot$

SAT Tools – Saturation Algorithm

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 – Saturation Algorithm

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:

- $\mathrm{R}((x_1 \vee x_7 \vee \overline{x_2} \vee x_4), (x_2)) = (x_1 \vee x_7 \vee x_4)$

Unit Propagation

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

Example:

- $(x_1) \wedge (x_7 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_3)$
- $(x_1) \wedge (x_7 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_3) \wedge (x_3)$
- $(x_1) \wedge (x_7 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_3) \wedge (x_3) \wedge (x_7 \vee x_2)$

SAT Tools – Unit Propagation

Unit Resolution

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

Example:

- $\mathrm{R}((x_1 \vee x_7 \vee \overline{x_2} \vee x_4), (x_2)) = (x_1 \vee x_7 \vee x_4)$

Unit Propagation

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

Example:

- $(x_1) \wedge (x_7 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_3)$
- $(x_1) \wedge (x_7 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_3) \wedge (x_3)$
- $(x_1) \wedge (x_7 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_3) \wedge (x_3) \wedge (x_7 \vee x_2)$

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:

- $(x_1 \vee x_3) \wedge (x_7 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_3)$
- $(x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2})$

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 $\mathcal{O}(n^2)$ 2-literal clauses on n variables

Complexity:

- Both time and space $\mathcal{O}(n^2)$
- There exists a linear algorithm! [6]

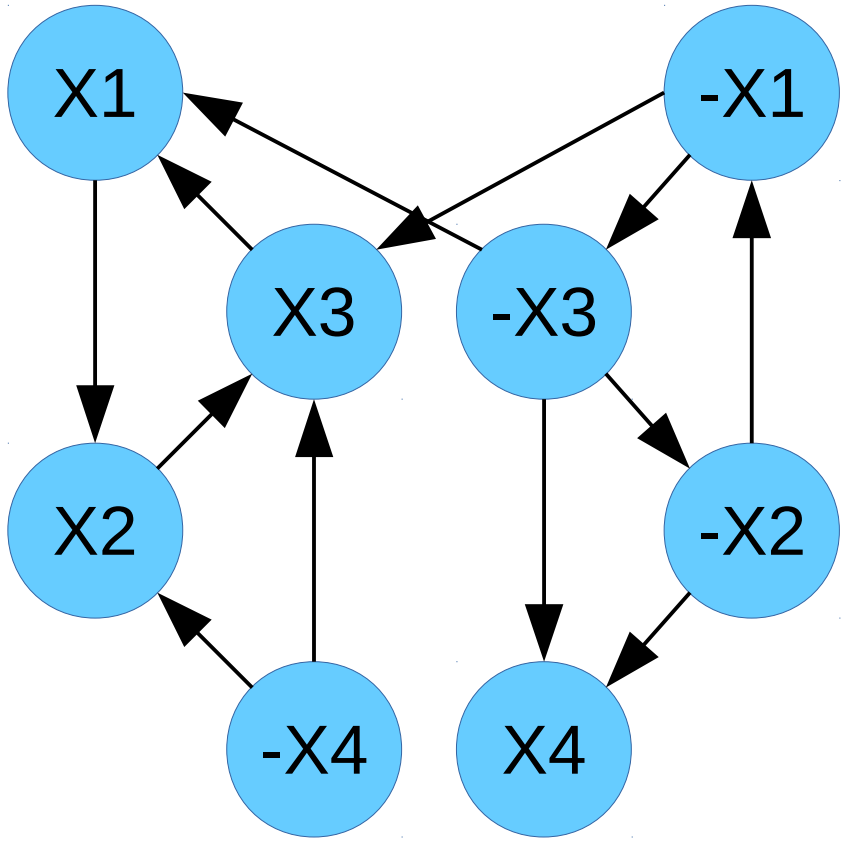
Implication Graph

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

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

Example:

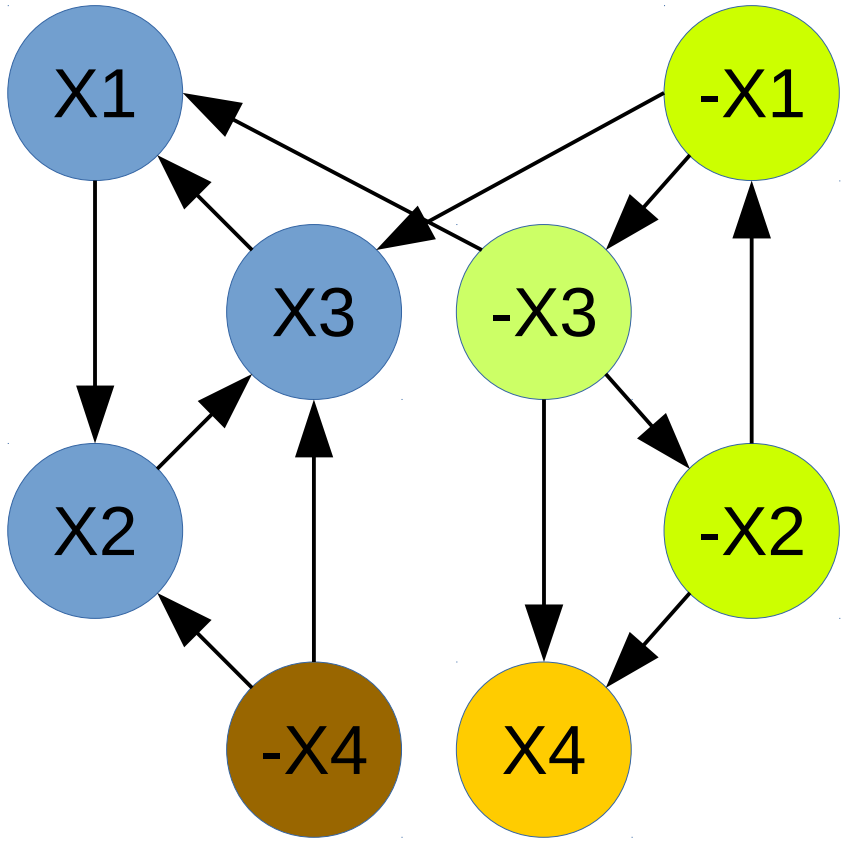
$(\overline{x}_1 \vee x_2) \wedge (\overline{x}_2 \vee x_3) \wedge (\overline{x}_3 \vee x_1) \wedge$
$(x_2 \vee x_4) \wedge (x_3 \vee x_4) \wedge (x_1 \vee x_3)$



Implication Graph

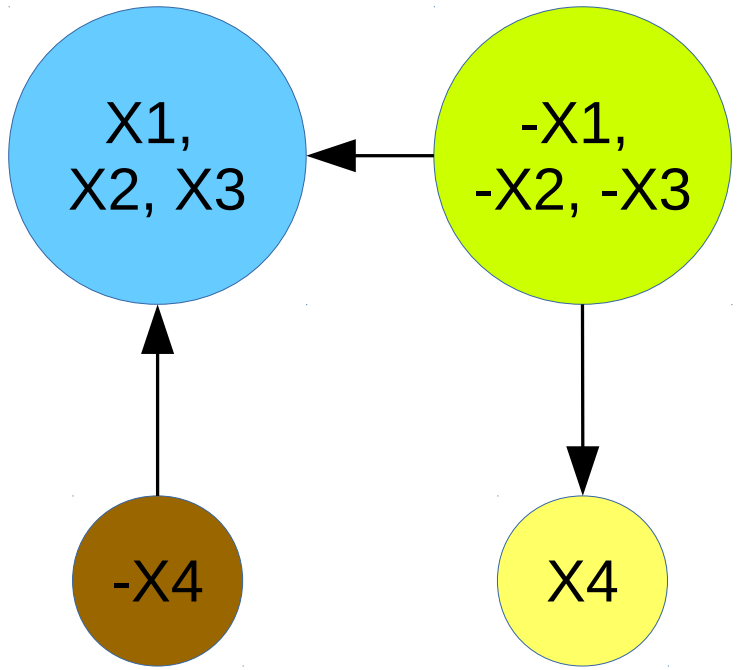
The next step is to analyse the *Strongly Connected Components* of the implication graphs

- SCC = there is a path in each direction between each pair
- Tarjan's algorithm finds SCCs in $\mathcal{O}(|V| + |E|)$
- 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



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: $x_1 = x_2 = x_3 = \mathit{True}$, $x_4 = \mathit{True}$, 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

HornSAT

Horn Formula

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

Example: $(\overline{x}_1 \vee \overline{x}_7 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_4) \wedge (x_1)$

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

HornSAT

Horn Formula

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

Example: $(\overline{x}_1 \vee \overline{x}_7 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_4) \wedge (x_1)$

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

has at least one negative literal in every clause so should work

Mixed Horn

Mixed Horn Formula

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

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

Questions:

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

Mixed Horn Formula

Mixed Horn Complexity

Mixed Horn SAT solving is NP-complete

Proof:

- We will reduce SAT to Mixed Horn SAT
- For each non-Horn clause $C = (l_1 \vee l_2 \vee \dots)$ do
  - for each but one positive $l_i \in C$ intruduce a new variable l_i'
  - replace l_i in C by $\overline{l_i'}$
  - add $(l_i' \vee l_i) \wedge (\overline{l_i'} \vee \overline{l_i})$ to establish $l_i = \overline{l_i'}$

Example: $(x_1 \vee \overline{x_7} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_4}) \wedge (x_1 \vee x_5) \rightsquigarrow$
$\rightsquigarrow (\overline{x_1'} \vee \overline{x_7} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_4}) \wedge (\overline{x_1'} \vee x_5) \wedge (x_1' \vee x_1) \wedge (\overline{x_1'} \vee \overline{x_1})$

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:

$$(x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee \overline{x}_4) \wedge (x_1) \rightsquigarrow (\overline{x}_1 \vee \overline{x}_2 \vee x_4) \wedge (\overline{x}_2 \vee \overline{x}_4) \wedge (\overline{x}_1)$$

Questions:

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

Recognizing Hidden Horn Formulas

Transalte into 2-SAT

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

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

Recognize Hidden Horn

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

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

References I


S. A. Cook, The complexity of theorem-proving procedures, in: Proceedings of the third annual ACM symposium on Theory of computing, ACM, 1971, pp. 151–158.


M. Davis, H. Putnam, A computing procedure for quantification theory, J. ACM 7 (3) (1960) 201–215.
doi:10.1145/321033.321034.
URL http://doi.acm.org/10.1145/321033.321034


M. Davis, G. Logemann, D. Loveland, A machine program for theorem-proving, Commun. ACM 5 (7) (1962) 394–397.
doi:10.1145/368273.368557.
URL http://doi.acm.org/10.1145/368273.368557

References II

- B. Selman, H. J. Levesque, D. G. Mitchell, et al., A new method for solving hard satisfiability problems., in: AAAI, Vol. 92, 1992, pp. 440–446.
- J. P. Marques-Silva, K. A. Sakallah, Grasp: A search algorithm for propositional satisfiability, Computers, IEEE Transactions on 48 (5) (1999) 506–521.
- B. Aspvall, M. F. Plass, R. E. Tarjan, A linear-time algorithm for testing the truth of certain quantified boolean formulas, Information Processing Letters 8 (3) (1979) 121–123.