Propositional Logic (Part 2)

ECE 650
Methods & Tools for Software Engineering (MTSE)
Fall 2023

Presented by
Dr. Albert Wasef

UNIVERSITY OF WATERLOO

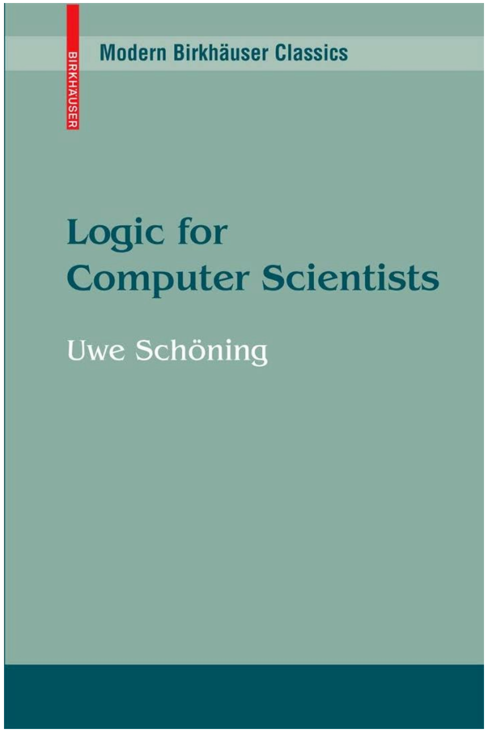
References

Chpater 1 of Logic for Computer Scientists

- https://link.springer.com/chapter/10.1007/978-0-8176-4763-6_2

Library link:

- https://link-springer-com.proxy.lib.uwaterloo.ca/chapter/10.1007/978-0-8176-4763-6_2



Book Example: Secret to long life

"What is the secret of your long life?" a centenarian was asked.

"I strictly follow my diet: If I don't drink beer for dinner, then I always have fish. Any time I have both beer and fish for dinner, then I do without ice cream. If I have ice cream or don't have beer, then I never eat fish."

The questioner found this answer rather confusing. Can you simplify it?

UNIVERSITY OF WATERLOO

Centenarian – a person who lives to or beyond an age 100.

Merriam-Webster SINCE 1828

JOIN MWU | GAMES | BROWSE THESAURUS | WORD OF THE DAY | WORDS AT PLAY

normal form

DICTIONARY THESAURUS

normal form *noun*

Definition of *normal form*

logic

: a canonical or standard fundamental form of a statement to which others can be reduced

especially **:** a compound statement in the propositional calculus consisting of nothing but a conjunction of disjunctions whose disjuncts are either elementary statements or negations thereof

https://www.merriam-webster.com/dictionary/normal%20form

Negation Normal Form (NNF)

A formula F is in Negation Normal Form (NNF) if every occurrence of negation ($\neg$) in F is applied to an atomic sub-formula of F.

For example,

- $\neg a \lor \neg b \lor c$ is in NNF
- $\neg (a \land b \land \neg c)$ is NOT in NNF (why?)

Theorem (NNF): For every formula F there is a semantically equivalent form G in NNF. In symbols

- For every F, exists G in NNF, such that $F \equiv G$

UNIVERSITY OF WATERLOO

Proof of NNF Theorem 1/2

By structural induction on the structure of a formula F

(base case): atomic formulas A and ¬A are in NNF

(IH): Assume that the theorem is true for every sub-formula of F: For every sub-formula H of F there exists J in NNF such that J $\equiv$ H. Show that there exists G in NNF such that G $\equiv$ F

(¬ case): see next slide

(∨ case): Assume $F = H_1 \lor H_2$. Then, $F \equiv J_1 \lor J_2$

(∧ case): Assume $F = H_1 \land H_2$. Then, $F \equiv J_1 \land J_2$

Proof of NNF Theorem 2/2

By structural induction on the structure of a formula F

(base case): atomic formulas A and ¬A are in NNF

(IH): Assume that the theorem is true for every sub-formula of F: For every sub-formula H of F there exists J in NNF such that $J \equiv H$. Show that there exists G in NNF such that $G \equiv F$

(¬ case):

(case 1) $F = \neg(\neg H_0) = \neg\neg H_0 \equiv H_0 \equiv J_0$

(case 2) $F = \neg(H_0 \land H_1) \equiv \neg H_0 \lor \neg H_1 \equiv J_0 \lor J_1$

(case 3) $F = \neg(H_0 \lor H_1) \equiv \neg H_0 \land \neg H_1 \equiv J_0 \land J_1$

- where J_0 is NNF of $\neg H_0$, and J_1 is NNF of $\neg H_1$ by IH

Normal Form: DNF

A *literal* is either an atomic proposition v or its negation ¬v

A *cube* is a conjunction of literals

- e.g., (v1 ∧ ¬ v2 ∧ v3)

A formula F is in *Disjunctive Normal Form* (DNF) if F is a disjunction of conjunctions of literals

$$\bigvee_{i=1}^{n} (\bigwedge_{j=1}^{m_i} L_{i,j})$$

(Fun) Fact: determining whether a DNF formula F is satisfiable is easy

- easy == linear in the size of the formula

Normal Form: CNF

A *literal* is either an atomic proposition v or its negation ¬v

A *clause* is a disjunction of literals

- e.g., (v1 ∨ ¬v2 ∨ v3)

A formula F is in *Conjunctive Normal Form* (CNF) if F is a conjunction of disjunctions of literals

$$\bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m_i} L_{i,j} \right)$$

(Fun) Fact: determining whether a CNF formula F is satisfiable is hard

- hard == NP-complete

Normal Form Theorem

Theorem: For every formula F, there is an equivalent formula F_1 in CNF, and an equivalent formula F_2 in DNF.

That is, CNF and DNF are normal forms:

- Every propositional formula can be converted to CNF and to DNF without affecting its meaning (i.e., semantics)!

Proof: (by induction on the structure of the formula F)

Details are left as an exercise!

Converting a formula to CNF

Given a formula F

1. Substitute in F every occurrence of a sub-formula of the form

   $\neg\neg G$ by G

   $\neg(G \land H)$ by $(\neg G \lor \neg H)$

   $\neg(G \lor H)$ by $(\neg G \land \neg H)$

   The result is a formula in Negation Normal Form (NNF)

2. Substitute in F each occurrence of a sub-formula of the form

   $(F \lor (G \land H))$ by $((F \lor G) \land (F \lor H))$

   $((F \land G) \lor H)$ by $((F \lor H) \land (G \lor H))$

The resulting formula F is in CNF

- the result in CNF might be exponentially bigger than original formula F

Example: From Truth Table to CNF and DNF

DNF

$$(\neg A \wedge \neg B \wedge \neg C) \vee$$
$$(A \wedge \neg B \wedge \neg C) \vee$$
$$(A \wedge \neg B \wedge C)$$

CNF

$$(A \vee B \vee \neg C) \wedge$$
$$(A \vee \neg B \vee C) \wedge$$
$$(A \vee \neg B \vee \neg C) \wedge$$
$$(\neg A \vee \neg B \vee C) \wedge$$
$$(\neg A \vee \neg B \vee \neg C)$$

Truth table

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

see the book for detailed algorithm

From Truth Table to DNF / CNF

From a truth table T to DNF formula F

- For every row i of T with value 1, construct a conjunction F_i that characterizes the row. That is, F_i is a formula that is true **exactly** for the assignment of row I
- Let F = $\vee$ F_i, then F is equivalent to T and is in DNF
- Fact: F has as many disjuncts as rows with value 1 in T
- Question: How big can F be? How small can F be?

From a truth table T to CNF formula G

- For every row i of T with value 0, construct a conjunction F_i that characterizes the row
- Let G_i be NNF of $\neg F_i$
- Let G = $\wedge$ G_i, then G is equivalent to T and is in CNF
- Fact: F has as many disjuncts as rows with value 0 in T
- Question: How big can F be? How small can F be?

UNIVERSITY OF WATERLOO

3-CNF Fragment

A formula F is in 3-CNF iff

- F is in CNF
- every clause of F has at most 3 literals

Theorem: Deciding whether a 3-CNF formula F is satisfiable is at least as hard as deciding satisfiability of an arbitrary CNF formula G

Proof: by effective *reduction* from CNF to 3-CNF

Let G be an arbitrary CNF formula. Replaced every clause of the form

$$(\ell_0 \vee \cdots \vee \ell_n)$$

with 3-literal clauses

$$(\ell_0 \vee b_0) \wedge (\neg b_0 \vee \ell_1 \vee b_1) \wedge \cdots \wedge (\neg b_{n-1} \vee \ell_n)$$

where $\{b_i\}$ are fresh atomic propositions not appearing in F

UNIVERSITY OF WATERLOO

Complexity of 3-CNF Satisfiability

Theorem (Cook-Levin): The Boolean Satisfiability Problem is NP-complete

Consequences

- If a formula F is satisfiable, then there exists a certificate for satisfiability that can be checked in P (polynomial) time.
  - That is, checking solutions is easy
- Any other problem that has polynomial certificates is polynomial reducible to Boolean Satisfiability
  - That is, such problems can be solved by writing a loop-free program, compiling it to a Boolean circuit, and checking whether the circuit ever accepts some input
- **MANY MANY MANY OPTIMIZATION PROBLEMS ARE LIKE THAT**
- Boolean Satisfiability is easy iff P = NP
  - i.e., Boolean satisfiability today is a VERY VERY VERY HARD problem!

Boolean Satisfiability (CNF-SAT) 1/2

Let V be a set of variables

A *literal* is either a variable v in V or its negation ¬v

A *clause* is a disjunction of literals e.g., (v1 ∨ ¬v2 ∨ v3)

A Boolean formula in *Conjunctive Normal Form* (CNF) is a conjunction of clauses e.g., (v1 ∨ ¬v2) ∧ (v3 ∨ v2)

Boolean Satisfiability (CNF-SAT) 2/2

An *assignment* *s* of Boolean values to variables *satisfies* a clause *c* if it evaluates at least one literal in *c* to true

An assignment *s* *satisfies* a formula *C* in CNF if it satisfies every clause in *C*

Boolean Satisfiability Problem (CNF-SAT): determine whether a given CNF C is satisfiable

Are the following CNFs SAT or UNSAT

Is CNF 1 satisfiable? (3 clauses)

- ¬ b
- ¬ a ∨ ¬b ∨ ¬c
- a
- SAT: s(a) = True; s(b) = False; s(c) = False

Is CNF 2 satisfiable? (4 clauses)

- ¬b
- ¬a ∨ b ∨ ¬c
- a
- ¬a ∨ c
- UNSAT

UNIVERSITY OF WATERLOO

Algorithms for SAT

SAT is NP-complete

- solution can be checked in polynomial time
- no polynomial algorithms for finding a solution are known

DPLL (Davis-Putnam-Logemman-Loveland, ‘60)

- smart enumeration of all possible SAT assignments
- worst-case EXPTIME
- alternate between deciding and propagating variable assignments

CDCL (GRASP ‘96, Chaff ‘01)

- conflict-driven clause learning
- extends DPLL with
  - smart data structures, backjumping, clause learning, heuristics, restarts…
- scales to millions of variables
- N. Een and N. Sörensson, “An Extensible SAT-solver”, in SAT 2013.

(Optional) Background Reading: SAT

http://cacm.acm.org/magazines/2009/8/34498-boolean-satisfiability-from-theoretical-h

Boolean Satisfiability: From ...

Find: currency Previous Next Options

TRUSTED INSIGHTS FOR COMPUTING'S LEADING PROFESSIONALS ACM.org | Join ACM | About Communications | ACM Resources | Alerts & Feeds

SIGN IN

COMMUNICATIONS OF THE ACM

Search

HOME | CURRENT ISSUE | NEWS | BLOGS | OPINION | RESEARCH | PRACTICE | CAREERS | MAGAZINE ARCHIVE

Home / Magazine Archive / August 2009 (Vol. 52, No. 8) / Boolean Satisfiability: From Theoretical Hardness... / Full Text

REVIEW ARTICLES

Boolean Satisfiability: From Theoretical Hardness to Practical Success

By Sharad Malik, Lintao Zhang

Communications of the ACM, Vol. 52 No. 8, Pages 76-82
10.1145/1536616.1536637

Comments

VIEW AS: SHARE:

There are many practical situations where we need to satisfy several potentially conflicting constraints. Simple examples of this abound in daily life, for example, determining a schedule for a series of games that resolves the availability of players and venues, or finding a seating assignment at dinner consistent with various rules the host would like to impose. This also applies to applications in computing, for example, ensuring that a hardware/software system functions correctly with its overall behavior constrained by the behavior of its components and their composition, or finding a plan for a robot to reach a goal that is

SIGN IN for Full Access

User Name

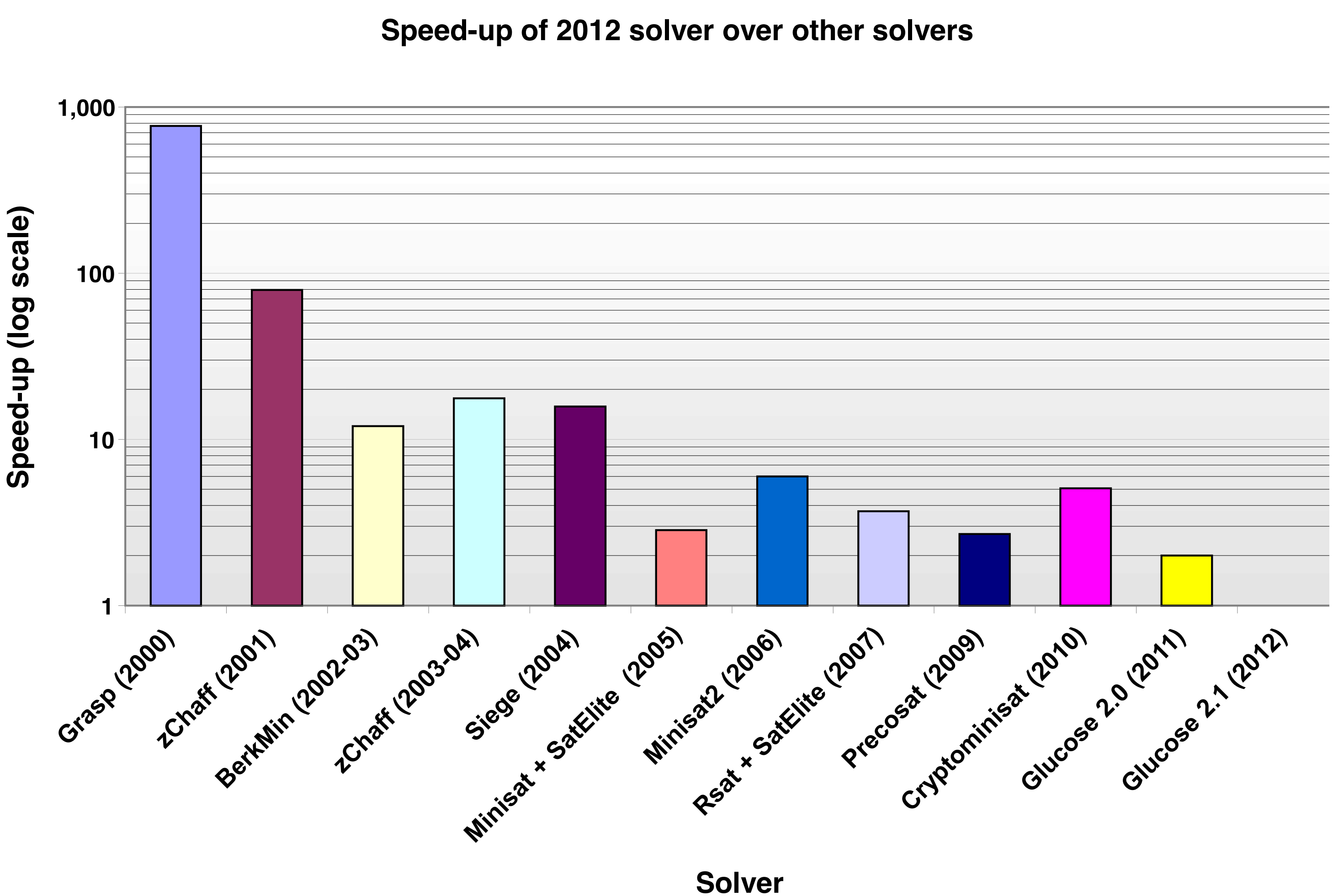
Password

» Forgot Password?
» Create an ACM Web Account

SIGN IN

ARTICLE CONTENTS:

Some Experience with SAT Solving



from M. Vardi, https://www.cs.rice.edu/~vardi/papers/highlights15.pdf

UNIVERSITY OF WATERLOO

SAT - Milestones

Problems impossible 10 years ago are trivial today

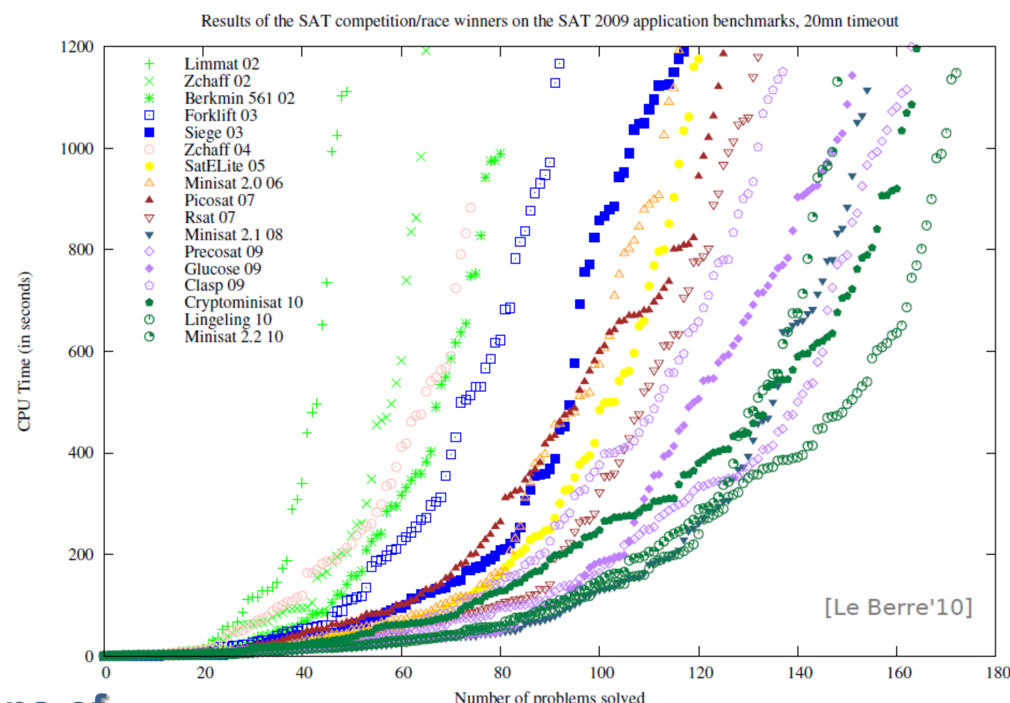
year	Milestone
1960	Davis-Putnam procedure
1962	Davis-Logeman-Loveland
1984	Binary Decision Diagrams
1992	DIMACS SAT challenge
1994	SATO: clause indexing
1997	GRASP: conflict clause learning
1998	Search Restarts
2001	zChaff: 2-watch literal, VSIDS
2005	Preprocessing techniques
2007	Phase caching
2008	Cache optimized indexing
2009	In-processing, clause management
2010	Blocked clause elimination

Concept

Millions of variables from HW designs

2002

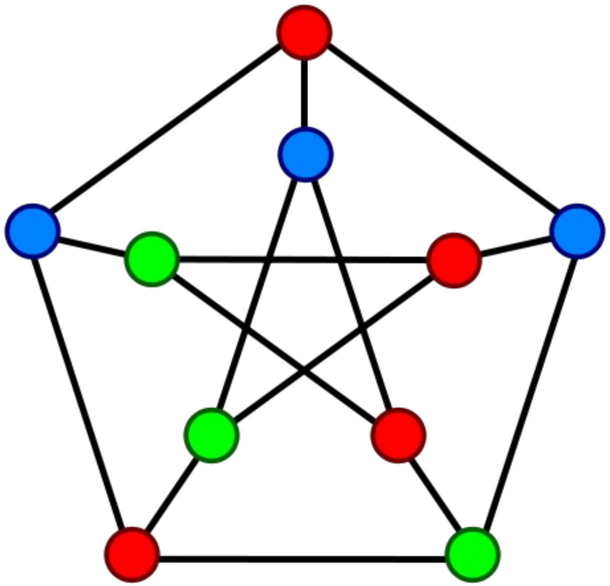
2010



Courtesy Daniel le Berre

ENCODING PROBLEMS TO SAT

Graph k-Coloring



Given a graph *G = (V, E)*, and a natural number *k > 0* is it possible to assign colors to vertices of *G* such that no two adjacent vertices have the same color.

Formally:

- does there exists a function f : V → [0..k) such that
- for every edge (u, v) in E, f(u) != f(v)

Graph coloring for k > 2 is NP-complete

Problem: Encode k-coloring of G into CNF

- construct CNF *C* such that C is SAT iff G is k-colorable

https://en.wikipedia.org/wiki/Graph_coloring

k-coloring as CNF

Let a Boolean variable $f_{v,i}$ denote that vertex *v* has color *i*

- if $f_{v,i}$ is true if and only if f(v) = i

Every vertex has at least one color

$$\bigvee_{0 \leq i < k} f_{v,i} \qquad (v \in V)$$

No vertex is assigned two colors

$$\bigwedge_{0 \leq i < j < k} (\neg f_{v,i} \vee \neg f_{v,j}) \qquad (v \in V)$$

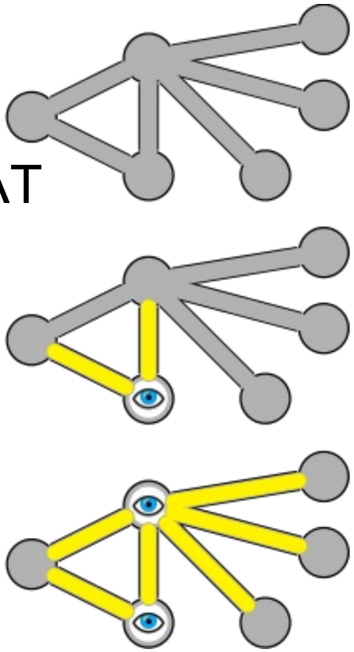
No two adjacent vertices have the same color

$$\bigwedge_{0 \leq i < k} (\neg f_{v,i} \vee \neg f_{u,i}) \qquad ((v, u) \in E)$$

UNIVERSITY OF WATERLOO

Vertex Cover

Given a graph G=(V,E). A vertex cover of G is a subset C of vertices in V such that every edge in E is incident to at least one vertex in C



see a4_encoding.pdf for details of reduction to CNF-SAT

- will be given together with assignment 4

https://en.wikipedia.org/wiki/Vertex_cover

USING A SAT SOLVER

DIMACS interface to a SAT Solver

Input:

- a CNF in DIMACS format

Output:

- SAT/UNSAT + satisfying assignment

We will use a SAT solver called MiniSAT

- available at https://git.uwaterloo.ca/ece650-f23/minisat
- written in C++
- use as a library in Assignment 4
- use via DIMACS interface today in class
- MiniSat examples:
  - https://git.uwaterloo.ca/ece650-f23/minisat-example

DIMACS CNF File Format

Textual format to represent CNF-SAT problems

```
c start with comments
c
c
p cnf 5 3
 1 -5 4 0
-1  5 3 4 0
-3 -4 0
```

Details

- comments start with `c`
- header line: `p cnf nbvar nbclauses`
  - `nbvar` is # of variables, `nbclauses` is # of clauses
- each clause is a sequence of distinct numbers terminating with 0
  - positive numbers are variables, negative numbers are negations

MiniSat

MiniSat is one of the most famous modern SAT-solvers

- written in C++
- designed to be easily understandable and customizable
- many new SAT-solvers use MiniSAT as their base

Web page: http://minisat.se/

We will use a slightly updated version from:
https://git.uwaterloo.ca/ece650-f23/minisat

Good references for understanding SAT solving details

- MiniSat architecture: http://minisat.se/downloads/MiniSat.pdf
- Donald Knuth's SAT13 (also based on MiniSat)
  - http://www-cs-faculty.stanford.edu/~knuth/programs/sat13.w

https://git.uwaterloo.ca/ece650-f23/minisat-example

MINISAT EXAMPLES

PROPOSITIONAL RESOLUTION

From CNF to database of literals

Assume that all propositional formulas are converted to CNF

Each clause is determined by the set of literals

- e.g., $(a \lor b \lor \neg c)$ is same as $\{a, b, \neg c\}$

A CNF is a database (a set) of clauses

- $(a \lor b \lor \neg c) \land (c) \land (\neg b \land d)$ is represented as
- $\{ \{a, b, \neg c\}, \{c\}, \{\neg b, d\} \}$

Propositional Resolution

Let A be a clause of the form $C \lor p$

Let B be a clause of the form $D \lor \neg p$

Propositional Resolution:

A clause $(C \lor D)$ is a resolvent of A and B on pivot p

Propositional Resolution In Symbols

$$\mathrm{Res}(\{C, p\}, \{D, \neg p\}) = \{C, D\}$$

Given two clauses $\{C, p\}$ and $\{D, \neg p\}$ that contain a literal p of different polarity, create a new clause by taking the union of literals in C and D

Notation: Inference Rule

Premise

Side-condition

$$\frac{F_1 \quad \ldots \quad F_n}{G} \quad H$$

Conclusion

Inference Rules

We express the evaluation rules as inference rules for our judgments.

The rules are also called evaluation rules.

An inference rule $$\frac{F_1 \ldots F_n}{G} \quad \text{where } H$$

defines a relation between judgments $F_1,\ldots,F_n$ and G.

- The judgments $F_1,\ldots,F_n$ are the premises of the rule;
- The judgments G is the conclusion of the rule;
- The formula H is called the side condition of the rule.

If n=0 the rule is called an axiom. In this case, the line separating premises and conclusion may be omitted.

Propositional Resolution Inferenc

Pivot

$$\frac{C \vee p \qquad\qquad D \vee \neg p}{C \vee D}$$

Resolvent

Res({C, p}, {D, ¬p}) = {C, D}

Given two clauses {C, p} and {D, ¬p} that contain a literal p of different polarity, create a new clause by taking the union of literals in C and D

Resolution Lemma

Let F be a CNF formula.

Let R be a resolvent with pivot p of two clauses X and Y in F

Then, $F \cup \{ R \}$ is equivalent to F.

- That is, R is implied by F and adding it to F does not change the meaning of F

Proof:

Show that for any assignment M, $M \models F$ if and only if $M \models F \cup \{ R \}$

If $M \models F \cup \{ R \}$ then $M \models F$ is trivial.

Show that if $M \models \{X, Y\}$ then $M \models R$.

Two cases: (case 1) $M \models p$, (case 2) $M \models \neg p$

Resolution Theorem

Let F be a set of clauses

$Res(F) = F \cup \{R \mid R \text{ is a resolvent of two clauses in } F\}$

Define Res^n recursively as follows:

$$Res^0(F) = F$$

$$Res^{n+1}(F) = Res(Res^n(F)), \text{ for } n \geq 0$$

$$Res^*(F) = \bigcup_{n \geq 0} Res^n(F)$$

Resolution Theorem:

A CNF F is UNSAT iff Res*(F) contains an empty clause

Exercise from LCS

For the following set of clauses determine Res^n for n=0, 1, 2

$$A \vee \neg B \vee C$$

$$B \vee C$$

$$\neg A \vee C$$

$$B \vee \neg C$$

$$\neg C$$

Proof of the Resolution Theorem 1/3

(Soundness) By Resolution Lemma, F is equivalent to $Res^i(F)$ for any i.

Let n be such that $Res^{n+1}(F)$ contains an empty clause, but $Res^n(F)$ does not

- such n must exist because an empty clause was added at some point

Then, $Res^n(F)$ must contain two unit clauses L and $\neg L$

- because the only way to construct an empty clause is to resolve two unit clauses

Hence, F is UNSAT

- every clause added by resolution is implied (entailed) by F
- hence, $F \rightarrow L$ and $F \rightarrow \neg L$
- Therefore, $F \rightarrow (L \land \neg L)$, and $F \rightarrow False$

Proof of the Resolution Theorem 2/3

(Completeness) By **induction** on the number of different atomic propositions in F.

(base case) if F has 0 atomic propositions and has a clause, then F contains an empty clause

- empty clause is the only clause without any atomic propositions

Proof of the Resolution Theorem 3/3

(inductive case):

Assume F is UNSAT and F has atomic propositions $A_1, \ldots A_{n+1}$

Let F_0 be the result of replacing atomic proposition A_{n+1} by 0

Let F_1 be the result of replacing atomic proposition A_{n+1} by 1

Since F is UNSAT, so are F_0 and F_1

- e.g., if F_0 is SAT with assignment M, then extend M to $A_{n+1} \rightarrow 0, \ldots$

By IH, both F_0 and F_1 derive an empty clause

- Hence, Res*(F) contains (A_{n+1}) (or empty clause) and Res*(F) contains ($\neg A_{n+1}$) (or empty clause)

Therefore, Res*(F) contains an empty clause!

Example for the last step of Pf of Res Theorem

$F = (a) \land (\neg a \lor b) \land (\neg b \lor c) \land (\neg c)$

$F_0 = (a) \land (\neg a) \land (\neg c)$

- $Res^*(F_0)$ contains an empty clause
- By following the same resolution steps in F, we show that $Res^*(F)$ contains the clause (b)

$F_1 = (a) \land (c) \land (\neg c)$

- $Res^*(F_1)$ contains an empty clause
- By following the same resolution steps in F, we show that $Res^*(F)$ contains the clause $(\neg b)$

Therefore, $Res^*(F)$ contains an empty clause!

END OF LECTURE 8

Proof System

$$P_1, \ldots, P_n \vdash C$$

An inference rule is a tuple $(P_1, \ldots, P_n, C)$

- where, $P_1, \ldots, P_n$, C are formulas
- P_i are called premises and C is called a conclusion
- intuitively, the rules says that the conclusion is true if the premises are

A proof system P is a collection of inference rules

A proof in a proof system P is a tree (or a DAG) such that

- nodes are labeled by formulas
- for each node *n*, the tuple (parents(n), n) is an inference rule in P

Propositional Resolution as an Inference Rule

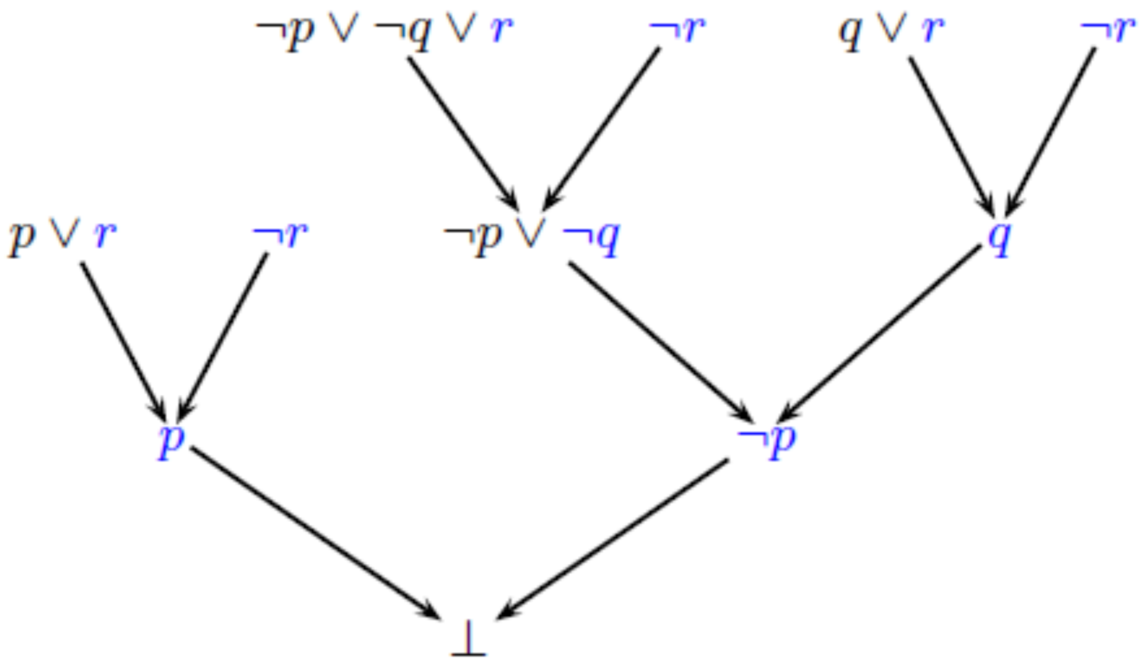
$$\frac{C \vee p \qquad\qquad D \vee \neg p}{C \vee D}$$

Propositional resolution is a sound inference rule

Proposition resolution proof system consists of a single propositional resolution rule

A Resolution Proof Example

A refutation of $\neg p \vee \neg q \vee r$, $p \vee r$, $q \vee r$, $\neg r$:



Another Resolution Pf Example

Show by resolution that the following CNF is UNSAT

$$\neg b \wedge (\neg a \vee b \vee \neg c) \wedge a \wedge (\neg a \vee c)$$

$$\dfrac{\dfrac{\dfrac{\neg a \vee b \vee \neg c \qquad a}{b \vee \neg c} \qquad b}{\neg c} \qquad \dfrac{a \qquad \neg a \vee c}{c}}{\bot}$$

Book: Exercise 33

Using resolution show that

$$A \wedge B \wedge C$$

is a consequence of

$$\neg A \vee B$$

$$\neg B \vee C$$

$$A \vee \neg C$$

$$A \vee B \vee C$$

Entailment and Derivation

A set of formulas F entails a set of formulas G iff every model of F and is a model of G

$$F \models G$$

A formula G is derivable from a formula F by a proof system P if there exists a proof whose leaves are labeled by formulas in F and the root is labeled by G

$$F \vdash_P G$$

UNIVERSITY OF WATERLOO

Book: Exercise 33

Using resolution show that

$$A \wedge B \wedge C$$

is a consequence of

$$\neg A \vee B$$

$$\neg B \vee C$$

$$A \vee \neg C$$

$$A \vee B \vee C$$

Soundness and Completeness

A proof system P is sound iff

$$(F \vdash_P G) \implies (F \models G)$$

A proof system P is complete iff

$$(F \models G) \implies (F \vdash_P G)$$

PR: Soundness and Completeness

Theorem: Propositional resolution is sound and complete for propositional logic

Proof:

Follows immediately from the Resolution Theorem!

Exercise 34

Show using resolution that F is valid

$$F = (\neg B \wedge \neg C \wedge D) \vee (\neg B \wedge \neg D) \vee (C \wedge D) \vee B$$

$$\neg F = (B \vee C \vee \neg D) \wedge (B \vee D) \wedge (\neg C \vee \neg D) \wedge \neg B$$

Compactness Theorem

Theorem:

A (possibly infinite) set M of propositional formulas is satisfiable iff every finite subset of M is satisfiable.

Corollary:

A (possibly infinite) set M of propositional formulas is unsatisfiable iff there exists a finite subset U of M such that U is unsatisfiable

Proof:

- Section 1.4 in Logic for Computer Scientists by Uwe Schoning

Satisfiability and Unsatisfiability

Let F be a propositional formula (large)

Assume that F is satisfiable. What is a short proof / certificate to establish satisfiability without a doubt?

- provide a model. The model is linear in the size of the formula

Now, assume that F is unsatisfiable. What is a short proof / certificate to establish UNSATISFIABILITY without a doubt?

Is the following formula SAT or UNSAT? How do you explain your answer?

$$\neg b \wedge (\neg a \vee b \vee \neg c) \wedge a \wedge (\neg a \vee c)$$