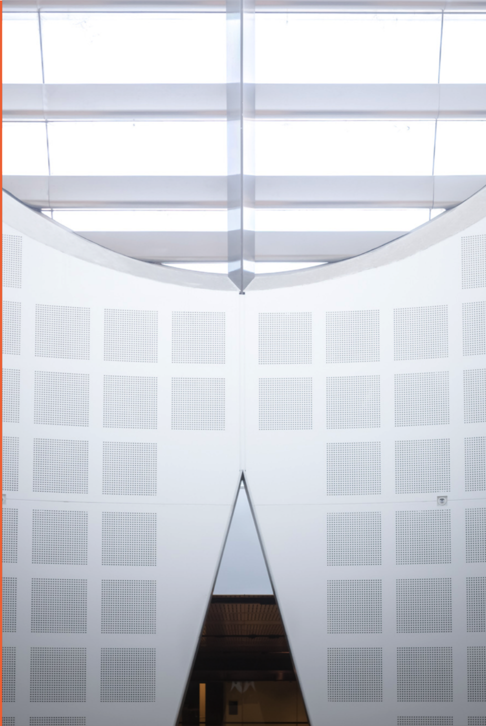
COMP2022|2922 Models of Computation

Propositional Logic: Normal forms and modeling problems in logic

Sasha Rubin

October 17, 2024

THE UNIVERSITY OF SYDNEY



Agenda

1. Normal forms (NNF, CNF)
2. Modeling problems as satisfiability problems

In this lecture we use the original syntax of propositional logic, i.e., the only operators are $\neg, \wedge$ and $\vee$. If you have a formula with any other operator, you should first get rid of it by replacing by equivalent formulas. E.g.,

– replace $(p \to q)$ by $(\neg p \vee q)$

– replace $\perp$ by $(p \wedge \neg p)$

Normal forms: NNF

Definition

A formula F is in negation normal form (NNF) if negations only occur immediately in front of atoms.

Which of the following are in NNF?

1. p
2. $\neg p$
3. $\neg\neg p$
4. $\neg\neg\neg p$
5. $(\neg q \lor p)$
6. $\neg(q \land \neg p)$

Normal forms: NNF

Theorem

For every formula F there is an equivalent formula in NNF.

Algorithm ("push negation inwards")

Here is the algorithm: substitute in F every occurence of a subformula of the form

$\neg\neg G$ by G	Double Negation Law
$\neg(G \wedge H)$ by $(\neg G \vee \neg H)$	de Morgan's Law
$\neg(G \vee H)$ by $(\neg G \wedge \neg H)$	de Morgan's Law

until no such subformulas occur, and return the result.

Why is this algorithm correct?

Normal forms: NNF

Example

Put $\neg(\neg p \land (\neg(r \land s) \lor q))$ into NNF.

Normal forms: CNF

Definition

- A literal is an atomic formula or the negation of an atomic formula.

  i.e., a literal has the form p or $\neg p$; note that $\neg\neg p$ is not a literal.
- A clause is a literal or a disjunction of literals.

  e.g., both $(p \vee \neg q \vee r)$ and $\neg r$ are clauses.
  but $(p \vee q) \wedge r$ is not a clause.
- A formula F is in conjunctive normal form (CNF) if it is a clause or a conjunction of clauses.

Self test

Which of the following are in CNF?

1. $(p \vee \neg q \vee r) \wedge (\neg p \vee r) \wedge q \wedge \neg p$
2. $p \wedge q$
3. $p \vee q$
4. $(p \wedge q) \vee r$

Say we have two variables x, y.

Write a formula that says that "the variables have the same value".

$$(x \wedge y) \vee (\neg x \wedge \neg y)$$

Now write this in CNF.

- Its negation says that x, y have different values, which we can easily write:

$$(x \wedge \neg y) \vee (\neg x \wedge y)$$

- So let's negate to get back to the meaning we want:

$$\begin{aligned}
&\neg((x \wedge \neg y) \vee (\neg x \wedge y)) \\
&\qquad \equiv \neg(x \wedge \neg y) \wedge \neg(\neg x \wedge y) \\
&\qquad \equiv (\neg x \vee \neg\neg y) \wedge (\neg\neg x \vee \neg y) \\
&\qquad \equiv (\neg x \vee y) \wedge (x \vee \neg y)
\end{aligned}$$

Which is in CNF !

This process, of negating, writing a formula, negating again, and simplifying I call the *duality trick*.

Another way to see why the CNF formula

$$(\neg x \vee y) \wedge (x \vee \neg y)$$

says "the variables x and y take the same values"...

Each clause eliminates an assignment which does not satisfy the specification.

1. $(\neg x \vee y)$ eliminates the assignment where x is true and y is false.
2. $(x \vee \neg y)$ eliminates the assignment where x is false and y is true.

Normal forms: CNF

Theorem

For every formula F there is an equivalent formula in CNF.

Proof

Here is the algorithm:

1. Put F in NNF, call it F'.
2. Substitute in F' each occurrence of a subformula of the form

$$((H \wedge I) \vee G) \text{ or } (G \vee (H \wedge I)) \qquad \text{Commutative Law}$$
$$\text{by } ((G \vee H) \wedge (G \vee I)) \qquad \text{Distributive Law}$$

   until no such subformulas occur, and return the result.

Why is the algorithm correct?

Normal forms

Why is CNF important?

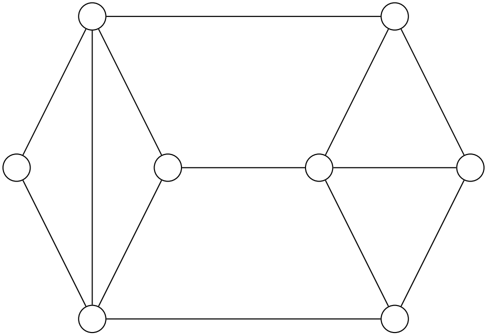
- It allows one to restrict to formulas with a uniform structure.
- It is used in practice... there are many tools for solving the satisfiability problem (SAT solvers) that take CNF formulas as input.

Modeling problems as satisfiability problems

The problem of finding cliques

A clique in a graph is a set of vertices all of which are adjacent to each other.

- in a 'friend' network, a clique is a group of people who are all friends with each other.
- proteins in a highly connected subgraph of a protein interaction network usually share a common function
- ...



The CLIQUE problem

Input: undirected graph (V, E) and an integer $K \geq 1$
Output: "Yes" if the graph has a clique of size K, "No" otherwise

- A clique (aka complete graph) of an undirected graph (V, E) is a set $C \subseteq V$ of vertices such that every two (distinct) vertices in C are adjacent.
- The size of C is the number of vertices in it, written $|C|$.

A naive algorithm would work as follows: for every set S of K vertices, check if there are edges between all the vertices in S.

There are $\binom{|V|}{K}$ such sets, and so this takes time approximately $|V|^K$, which is exponential in the size of the input.

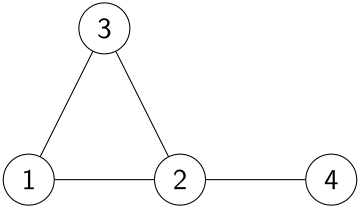
Nonetheless, we will solve it using logic by turning the instance into a formula that we send to a SAT solver!

Solving CLIQUE with logic

Here are the steps:

1. Given input: graph (V, E) and number K
2. Construct a formula $\Phi_{V,E,K}$.
3. Check if the formula $\Phi_{V,E,K}$ is satisfiable.
4. If it is satisfiable, return "Yes, there is a clique of size K in the graph", otherwise return "No".

Running example with $K = 3$:



Solving CLIQUE with logic

Given a graph (V, E) and $K \geq 1$.

Idea. Introduce a variable for each vertex, say v_i for $1 \leq i \leq |V|$, and write a formula expressing "the true variables form a clique in (V, E) of size K."

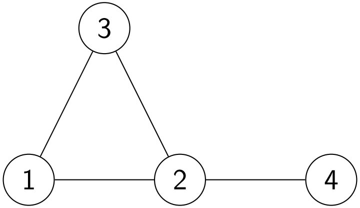
i.e., express in logic that:

1. There are exactly K many true variables.
2. Every pair of true variables correspond to an edge in the graph.

Then: the satisfying assignments of the formula identify the cliques of size K in (V, E)!

So: checking if the formula is satisfiable will check if there is a clique of size K in (V, E).

Running example



- Introduce 4 variables x_1, x_2, x_3, x_4.
- How do we express "There are exactly $K = 3$ many true variables" in logic?

The idea is to express that there is a subset S of $\{1, 2, 3, 4\}$ of size 3 such that the variables in S are true and the variables out of S are false.

S can be $\{1, 2, 3\}$, $\{1, 2, 4\}$, $\{1, 3, 4\}$, $\{2, 3, 4\}$, so the formula is:

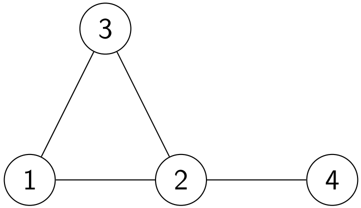
$$\boxed{\begin{gathered}(x_1 \wedge x_2 \wedge x_3 \wedge \neg x_4) \vee (x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4) \vee \\ (x_1 \wedge \neg x_2 \wedge x_3 \wedge x_4) \vee (\neg x_1 \wedge x_2 \wedge x_3 \wedge x_4)\end{gathered}}$$

In general, to express "there are exactly K many true variables":

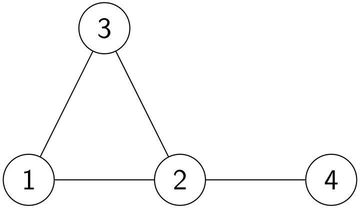
For a set $S \subseteq \{1, 2, \cdots, |V|\}$ of indices:

- $\bigwedge_{i \in S} x_i$ says that all the variables indexed by S are true.
- $\bigwedge_{i \notin S} \neg x_i$ says that all the variables not indexed by S are false.
- So the following says what we want:

$$\boxed{\bigvee_{|S|=K} (\wedge_{i \in S} x_i) \wedge (\wedge_{i \notin S} \neg x_i)}$$



How do we express "Every pair of true variables correspond to an edge in the graph"?



How do we express "Every pair of true variables correspond to an edge in the graph"?

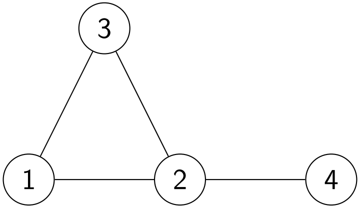
1. $(x_1 \wedge x_2) \wedge (x_1 \wedge x_3) \wedge (x_2 \wedge x_3) \wedge (x_2 \wedge x_4)$?
2. $\neg(x_1 \wedge x_4) \wedge \neg(x_3 \wedge x_4)$?
3. Some other formula?

In general, to express "Every pair of true variables correspond to an edge in the graph":

$$\boxed{\bigwedge_{(i,j)\notin E} \neg(x_i \wedge x_j)}$$

Putting this together... Given input (V, E) and size K, build the formula $\Phi_{V,E,K}$:

$$\boxed{\begin{array}{c} \displaystyle\bigvee_{|S|=K} (\wedge_{i \in S} x_i) \wedge (\wedge_{i \notin S} \neg x_i) \\ \\ \wedge \\ \\ \displaystyle\bigwedge_{(i,j) \notin E} \neg(x_i \wedge x_j) \end{array}}$$



$$\begin{aligned}
&\big((x_1 \wedge x_2 \wedge x_3 \wedge \neg x_4) \vee (x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4) \vee \\
&(x_1 \wedge \neg x_2 \wedge x_3 \wedge x_4) \vee (\neg x_1 \wedge x_2 \wedge x_3 \wedge x_4)\big) \\
\wedge& \\
&\big(\neg(x_1 \wedge x_4) \wedge \neg(x_3 \wedge x_4)\big)
\end{aligned}$$

What are the satisfying assignments?

How big is this formula? $O(|V|^K + |V|^2)$, which is exponential in the size of the input.

In the tutorial you will explore a different encoding which is polynomial.