CS228 Logic for Computer Science 2021

Lecture 7: Conjunctive normal form

Instructor: Ashutosh Gupta

IITB, India

Compile date: 2021-07-31

Normal forms

- Grammar of propositional logic is too complex.
- If one builds a tool, one will prefer to handle fewer connectives and simpler structure
- We transform given formulas into normal forms before handling them.

We will look at the following two normal forms

- Negation normal form (seen in the previous lecture)
- Conjunctive normal forms

Commentary: Building a software for handling formulas with the complexity is undesirable. We aim to reduce the complexity by applying transformations to obtain a normalized form. The normalization results in standardization and interoperability of tool.

Removing $\oplus$, $\Rightarrow$, and $\Leftrightarrow$.

Please note the following equivalences that remove $\oplus$, $\Rightarrow$, and $\Leftrightarrow$ from a formula.

- $(p \Rightarrow q) \equiv (\neg p \vee q)$
- $(p \oplus q) \equiv (p \vee q) \wedge (\neg p \vee \neg q)$
- $(p \Leftrightarrow q) \equiv \neg(p \oplus q)$

For the ease of presentation, we will assume you can remove them at will.

Commentary: Removing $\Rightarrow$ is common and desirable. The removal of $\oplus$ and $\Leftrightarrow$, however, blows up the formula size. Their straight up removal is not desirable. We can avoid the blow up in some contexts. However, in our presentation we will skip the issue.

Topic 7.1

Conjunctive normal form

Some terminology

- Propositional variables are also referred as atoms
- A literal is either an atom or its negation
- A clause is a disjunction of literals.

Since $\vee$ is associative, commutative, and absorbs multiple occurrences, a clause may be referred as a set of literals.

Example 7.1

- *p is an atom but $\neg p$ is not.*
- *$\neg p$ and p both are literals.*
- *$p \vee \neg p \vee p \vee q$ is a clause.*
- *$\{p, \neg p, q\}$ is the same clause.*

Conjunctive normal form(CNF)

Definition 7.1

A formula is in CNF if it is a conjunction of clauses.

Since $\land$ is associative, commutative and absorbs multiple occurrences, a CNF formula may be referred as a set of clauses

Example 7.2

- $\neg p$ *and* p *both are in CNF.*
- $(p \lor \neg q) \land (r \lor \neg q) \land \neg r$ *in CNF.*
- $\{(p \lor \neg q), (r \lor \neg q), \neg r\}$ *is the same CNF formula.*
- $\{\{p, \neg q\}, \{r, \neg q\}, \{\neg r\}\}$ *is the same CNF formula.*

Commentary: A set of formulas is interpreted depending on the context. There is no requirement that we apply conjunction among the elements. A clause is a set of literals. We interpret it as disjunction of literals. A CNF formula is a set of clauses, which is set of sets of literals. We interpret it as conjunction of clauses.

Exercise 7.1

a. Write a formal grammar for CNF

b. How can we represent true and false using CNF formulas?

CNF conversion

Theorem 7.1

For every formula F there is another formula F' in CNF s.t. $F \equiv F'$.

Proof.

Let us suppose we have

- removed $\oplus$, $\Rightarrow$, $\Leftrightarrow$ using the standard equivalences,
- converted the formula in NNF with removal of $\Rightarrow$, $\Leftrightarrow$, and $\oplus$, and
- flattened $\wedge$ and $\vee$.

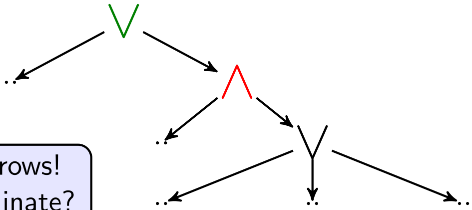
...

Commentary: We will assume flattened $\wedge$ and $\vee$. If we receive a formula that is not flattened, we will ignore the unnecessary parentheses.

CNF conversion (contd.)

Proof(contd.)

Now the formulas have the following form with literals at leaves.



After the push formula size grows!
Why should the method terminate?

Since $\vee$ distributes over $\wedge$, we can push $\vee$ inside $\wedge$. Eventually, we obtain a CNF formula. $\square$

Example 7.3

Are we done?

Conversion to CNF

$(p \Rightarrow (\neg q \wedge r)) \wedge (p \Rightarrow \neg q) \equiv (\neg p \vee (\neg q \wedge r)) \wedge (\neg p \vee \neg q) \equiv (\neg p \vee \neg q) \wedge (\neg p \vee r) \wedge (\neg p \vee \neg q)$

Commentary: The above is a good example of an algorithm that has intuitively clear but formally non-trivial termination argument.

Formal derivation for CNF

Theorem 7.2

Let F' be the CNF of F. If we have $\Sigma \vdash F$, then we can derive $\Sigma \vdash F'$.

Proof.

We combine the following pieces of proofs for each step of the transformations.

- Derivations for NNF
- Derivations for substitutions that remove $\Rightarrow$, $\oplus$, and $\Leftrightarrow$
- Derivations for substitutions that flatten $\wedge$ and $\vee$
- Derivations for substitutions that apply distributivity

Therefore, we have the derivations. □

Conjunctive normal form(CNF) more notation

- A unit clause contains only one literal.
- A binary clause contains two literals.
- A ternary clause contains three literals.
- We extend the definition of clauses to the empty set of literals. Say, $\perp$ is the empty clause.

Example 7.4

- $(p \land q \land \neg r)$ *has three unit clauses*
- $(p \lor \neg q \lor \neg s) \land (p \lor q) \land \neg r$ *has a ternary, a binary, and a unit clause*

Exercise 7.2

a. Give a linear time algorithm to prove validity of a CNF formula
b. What is the interpretation of the empty set of clauses?

Clauses as sets and CNF formulas as set of sets

Definition 7.2 (clause redefined)

A clause is a finite set of literals $\{\ell_1, \ldots, \ell_n\}$ *and interpreted as* $\ell_1 \vee .. \vee \ell_n$.

For a clause C and a literal ℓ, we will write $\ell \cup C$ to denote $\{\ell\} \cup C$.

Definition 7.3 (CNF formula redefined)

A CNF formula is a finite set of clauses $\{C_1, \ldots, C_n\}$ *and interpreted as* $C_1 \wedge .. \wedge C_n$.

Topic 7.2

Tseitin encoding

CNF is desirable

- Fewer connectives
- Simple structure
- Many problems naturally encode into CNF.

  We will see this in couple of lectures.

How do we get to CNF?

- The transformation using distributivity explodes the formula
- Is there a way to avoid the explosion?
- Yes! there is a way.

Tseitin encoding

But, with a cost.

Commentary: Tseitin is also spelled Tseytin.

Tseitin encoding : intuition

Example 7.5

Consider formula $p \vee (q \wedge r)$, *which is not in CNF.*

We replace offending $(q \wedge r)$ *by a fresh* x *and add clauses to encode that* x *behaves like* $(q \wedge r)$.

$$(p \vee x) \wedge (x \Rightarrow (q \wedge r))$$

After simplification,

$$(p \vee x) \wedge (\neg x \vee q) \wedge (\neg x \vee r)$$

Exercise 7.3

a. *Ideally, we should have introduced* $(x \Leftrightarrow (q \wedge r))$. *Why is the above with implication correct?*

b. *Show that transformation from* $(F \vee \neg G)$ *to* $(F \vee \neg x) \wedge (x \Rightarrow G)$ *will not preserve satisfiability?*

c. *Show that transformation from* $(F \vee \neg G)$ *to* $(F \vee \neg x) \wedge (G \Rightarrow x)$ *preserves satisfiability?*

Tseitin encoding (Plaisted-Greenbaum optimization included)

By introducing fresh variables, Tseitin encoding can translate every formula into an equisatisfiable CNF formula without exponential explosion.

1. Assume input formula F is NNF without $\oplus$, $\Rightarrow$, and $\Leftrightarrow$.
2. Find a $G_1 \wedge \cdots \wedge G_n$ that is just below an $\vee$ in $F(G_1 \wedge \cdots \wedge G_n)$
3. Replace $F(G_1 \wedge .. \wedge G_n)$ by $F(p) \wedge (\neg p \vee G_1) \wedge .. \wedge (\neg p \vee G_n)$, where p is a fresh variable
4. goto 2

Exercise 7.4

Modify the encoding such that it works without the assumptions at step 1

Commentary: If you read wikipedia about the encoding, you will find that Tseitin encoding adds more clauses. The above translation includes Plaisted-Greenbaum optimization. Solve the above exercise to understand the difference. Hint: Download sat solver $wget http://fmv.jku.at/limboole/limboole1.1.tar.gz look for function tseitin in file limboole.c

Example: linear cost of Tseitin encoding

Example 7.6

Consider formula $(p_1 \wedge \cdots \wedge p_n) \vee (q_1 \wedge \cdots \wedge q_m)$
Using distributivity, we obtain the following CNF containing mn *clauses.*

$$\bigwedge_{i \in 1..n,\ j \in 1..m} (p_i \vee q_j)$$

Using Tseitin encoding, we obtain the following CNF containing $m + n + 1$ *clauses, where* x *and* y *are the fresh Boolean variables.*

$$(x \vee y) \wedge \bigwedge_{i \in 1..n} (\neg x \vee p_i) \wedge \bigwedge_{j \in 1..m} (\neg y \vee q_j)$$

Exercise 7.5

Give a model to the original formula that is not a model of the transformed formula

Tseitin encoding preserves satisfiability

Let us prove one direction of the equisatisfiability.

Theorem 7.3

if $m \models F(p) \land (\neg p \lor G_1) \land \dots \land (\neg p \lor G_n)$ *then* $m \models F(G_1 \land \dots \land G_n)$

Proof.

Assume $m \models F(p) \land (\neg p \lor G_1) \land \dots \land (\neg p \lor G_n)$. We have three cases.

First case $m \models p$:

- Therefore, $m \models G_i$ for all $i \in 1..n$.
- Therefore, $m \models G_1 \land \dots \land G_n$.
- Due to the substitution theorem, $m \models F(G_1 \land \dots \land G_n)$.

Second case $m \not\models p$ and $m \not\models G_1 \land \dots \land G_n$:

- Due to the substitution theorem, $m \models F(G_1 \land \dots \land G_n)$

...

Tseitin encoding preserves satisfiability(contd.)

Proof(contd.)

Third case $m \not\models p$ and $m \models G_1 \wedge \cdots \wedge G_n$:

- Since $F(G_1 \wedge \cdots \wedge G_n)$ is in NNF, p occurs only positively in $F(p)$.
- Therefore, $m[p \mapsto 1] \models F(p)$(why?).
- Since p does not occur in G_is, $m[p \mapsto 1] \models G_1 \wedge \cdots \wedge G_n$.
- Due to the substitution theorem, $m[p \mapsto 1] \models F(G_1 \wedge \cdots \wedge G_n)$
- Therefore, $m \models F(G_1 \wedge \cdots \wedge G_n)$. $\square$

Commentary: We have introduced p, which is replacing $G_1 \wedge \ldots \wedge G_n$. Since the formula is in NNF, the negation symbols are only on variables. Therefore, they cannot be above $G_1 \wedge \ldots \wedge G_n$ in $F(G_1 \wedge \ldots \wedge G_n)$. Therefore in $F(p)$, p occurs positively.

We leave the other direction of equisatisfiability as the following exercise.

Exercise 7.6

Show if $\not\models F(p) \wedge (\neg p \vee G_1) \wedge .. \wedge (\neg p \vee G_n)$ *then* $\not\models F(G_1 \wedge .. \wedge G_n)$

Topic 7.3

k-sat

k-sat

Definition 7.4

A k*-sat* *formula is a CNF formula and has at most* k *literals in each of its clauses*

Example 7.7

- $(p \wedge q \wedge \neg r)$ *is 1-SAT*
- $(p \vee \neg p) \wedge (p \vee q)$ *is 2-SAT*
- $(p \vee \neg q \vee \neg s) \wedge (p \vee q) \wedge \neg r$ *is 3-SAT*

3-SAT satisfiablity

Theorem 7.4

For each k-SAT formula F there is a 3-SAT formula F' with linear blow up such that F and F' are equisatisfiable.

Proof.

Consider F a k-SAT formula with $k \geq 4$.
Consider a clause $G = (\ell_1 \vee \cdots \vee \ell_k)$ in F, where ℓ_i are literals.

Let $x_2, \ldots, x_{k-2}$ be variables that are not in $Vars(F)$.
Let G' be the following set of clauses

$$(\ell_1 \vee \ell_2 \vee x_2) \wedge \bigwedge_{i \in 2..k-3} (\neg x_i \vee x_{i+1} \vee \ell_{i+1}) \wedge (\neg x_{k-2} \vee \ell_{k-1} \vee \ell_k).$$

We show F is sat iff $(F - \{G\}) \cup G'$ is sat. ...

Exercise 7.7

Convert $(p \vee \neg q \vee s \vee \neg t) \wedge (\neg q \vee x \vee \neg y \vee z)$ into a 3-SAT formula

3-SAT satisfiability(cont. I)

Proof(contd. from last slide).

Recall

$$G' = (\ell_1 \vee \ell_2 \vee x_2) \wedge \bigwedge_{i \in 2..k-3} (\neg x_i \vee x_{i+1} \vee \ell_{i+1}) \wedge (\neg x_{k-2} \vee \ell_{k-1} \vee \ell_k).$$

Assume $m \models (F - \{G\}) \cup G'$:

Assume for each $i \in 1..k$, $m(\ell_i) = 0$.

Due to the first clause $m(x_2) = 1$.

Due to ith clause, if $m(x_i) = 1$ then $m(x_{i+1}) = 1$.

Due to induction, $m(x_{k-2}) = 1$.

Due to the last clause of G', $m(x_{k-2}) = 0$. Contradiction.

Therefore, there is $i \in 1..k$ such that $m(\ell_i) = 1$. Therefore $m \models G$. Therefore, $m \models F$

...

3-SAT satisfiability(cont. II)

Proof(contd. from last slide).

Recall $$G' = (\ell_1 \vee \ell_2 \vee x_2) \wedge \bigwedge_{i \in 2..k-3} (\neg x_i \vee x_{i+1} \vee \ell_{i+1}) \wedge (\neg x_{k-2} \vee \ell_{k-1} \vee \ell_k).$$

Assume $m \models F$:

Therefore, $m \models G$.

There is a $m(\ell_i) = 1$.

Let $m' \triangleq m[x_2 \mapsto 1, .., x_{i-1} \mapsto 1, x_i \mapsto 0, \ldots, x_{k-2} \mapsto 0]$.

Therefore, $m' \models F - \{G\}$ and $m' \models G'_{\text{(why?)}}$. Therefore, $m' \models (F - \{G\}) \cup G'$.

G' contains $3(k-2)$ literals. In the worst case, the formula size will increase 3 times. □

Commentary: Tseitin is a trick. We can apply it anywhere. Can we can convert 3-SAT to 2-SAT? All the usual tricks including Tseitin do not work. Unfortunately, We do not have a proof that 3-SAT cannot be converted to 2-SAT with in polynomial blow up. This transformation is the heart of the question P=?=NP.

Exercise 7.8

a. Complete the above argument.

b. Show a 3-SAT cannot be converted into a 2-SAT via Tseitin encoding.

c. When is the worst case?

Topic 7.4

Problems

Convert into CNF

Exercise 7.9

Give a CNF formula equivalent to $p \oplus q \oplus \neg r$.

Monotonic NNF

Definition 7.5

Let $pos(m, F)$ be the set of literals that are true in m and appear in F.

Example 7.8

$pos(\neg p_2 \wedge (p_1 \vee p_2), \{p_1 \mapsto 1, p_2 \mapsto 0\}) = \{p_1, \neg p_2\}$

Exercise 7.10

Let F be in NNF and does not contain $\oplus$, $\Rightarrow$, and $\Leftrightarrow$. Show that if $m \models F$ and $pos(m, F) \subseteq pos(m', F)$ then $m' \models F$.

Exercise: disjunctive normal form(DNF)

Definition 7.6

A formula is in DNF if it is a disjunction of conjunctions of literals.

Exercise 7.11

a. *Prove: for every formula F there is another formula F' in DNF such that $F \equiv F'$.*

b. *Give the formal grammar of DNF*

c. *Give a linear time algorithm to prove satisfiability of a DNF formula*

CNF and DNF

Exercise 7.12

Give an example of a non-trivial formula that is both CNF and DNF

CNF

Exercise 7.13

Convert the following formulas into CNF with/without introducing fresh variables

1. $\neg((p \Rightarrow q) \Rightarrow ((q \Rightarrow r) \Rightarrow (p \Rightarrow r)))$
2. $(p \Rightarrow (\neg q \Rightarrow r)) \land (p \Rightarrow \neg q) \Rightarrow (p \Rightarrow r)$
3. $(p \Rightarrow q) \lor (q \Rightarrow \neg r) \lor (r \Rightarrow q) \Rightarrow \neg(\neg(q \Rightarrow p) \Rightarrow (q \Leftrightarrow r))$

P=NP argument

Exercise 7.14

What is wrong with the following proof of P=NP? Give counterexample.

Tseitin encoding does not explode and proving validity of CNF formulas has a linear time algorithm. Therefore, we can convert every formula into CNF in polynomial time and check validity in linear time. As a consequence, we can check satisfiability of F in linear time by checking validity of $\neg F$ in linear time.

Validity**

Exercise 7.15

Give a procedure like Tseitin encoding that converts a formula into another equi-valid DNF formula. Prove correctness of your transformation.

Algebraic normal form(ANF)**

ANF formulas are defined using the following grammar.

$$
\begin{aligned}
A &::= \top \mid \bot \mid p \\
C &::= A \wedge C | A \\
ANF &::= C \oplus ANF | C
\end{aligned}
$$

Exercise 7.16

a. Give an efficient algorithm to covert any formula into equivalent ANF formula.

b. Give an efficient algorithm to covert any formula into equisatisfiable ANF formula.

CNF vs. DNF***

Exercise 7.17
Give a class of Boolean functions that can be represented using linear size DNF formula but can only be represented by an exponential size CNF formula.

Exercise 7.18
Give a class of Boolean functions that can be represented using linear size CNF formula but can only be represented by an exponential size DNF formula.

Probability of satisfiability***

Exercise 7.19

a. What is the probability that the conjunction of a random multiset of literals of size k over n Boolean variables is unsatisfiable?

b. What is the probability that the conjunction of a random set of literals of size k over n Boolean variables is unsatisfiable?

And invertor graphs (AIG)**

AIG formulas are defined using the following grammar.

$$A ::= A \wedge A | \neg A | p$$

Exercise 7.20

Give heuristics to minimize the number of inverters in an AIG formula without increasing the size of the formula.

Commentary: Example of such heuristics: Local Two-Level And-Inverter Graph Minimization without Blowup. Robert Brummayer and Armin Biere, 2006.

Topic 7.5

Extra slides: termination of CNF conversion

CNF conversion terminates

Theorem 7.5

The procedure of converting a formula into CNF terminates.

Proof.

For a formula F, let $\nu(F) \triangleq$ the maximum height of $\vee$ to $\wedge$ alternations in F.

Consider a formula $F(G)$ such that

$$G = \bigvee_{i=0}^{m} \bigwedge_{j=0}^{n_i} G_{ij}.$$

After the push we obtain $F(G')$, where

$$G' = \bigwedge_{j_1=0}^{n_1} \dots \bigwedge_{j_m=0}^{n_m} \underbrace{\bigvee_{i=0}^{m} G_{ij_i}}_{\nu(\quad) < \nu(G)}$$

Observations

- G' is either the top formula or the parent connective is $\wedge$
- G_{ij} is either a literal or an $\vee$ formula

Commentary: $\vee$ formula means that the top symbol in the formula is $\vee$

...

We need to apply flattening to keep $F(G')$ in the form(like the previous proof).

CNF conversion terminates (contd.)

(contd.)

Due to König lemma, the procedure terminates.[König lemma slides are at the end.](why?) □

Exercise 7.21

Consider a set of balls that are labelled with positive numbers. We can replace a k labelled ball with any number of balls with labels less than k. Using König lemma, show that the process always terminates.

Hint: in the above exercise, the bag is the subformulas of $F(G)$.

Exercise 7.22

Show F' obtained from the procedure may be exponentially larger than F.

Commentary: It is slightly involved to see the application of König lemma on our theorem. We can understand the application via the above exercise. In the above exercise, we are removing balls with large labels and replacing with balls that have smaller labels. This process will eventually hit label 1 for all balls. Once the balls with label 1 are removed, we can not add any more balls. In a similar way in our theorem, we are removing subformulas with larger ν and replacing with many subformulas with smaller ν. Therefore, the process will terminate. The formal construction is left for the exercise.

König Lemma

Theorem 7.6

For an infinite connected graph G, if degree of each node is finite then there is an infinite simple path in G from each node.

Proof.

We construct an infinite simple path $v_1, v_2, v_3, ...$ as follows.

base case:

Choose any $v_1 \in G$.Let $G_1 \triangleq G$.

induction step:

1. Assume path $v_1, .., v_i$ and an infinite connected graph G_i such that $v_i \in G_i$ and $v_1..v_{i-1} \not\in G_i$.
2. In G_i, there is a neighbour $v_{i+1} \in G_i$ of v_i such that infinite nodes are reachable from v_{i+1} without visiting v_i.(why?)
3. Let S be the reachable nodes. Let $G_{i+1} \triangleq G_i|_S$. □

Exercise 7.23

Prove that any finitely-branching infinite tree must have an infinite branch.

End of Lecture 7