# PL - Normal Forms of Formulas

HE Mingxin, Max

CS104: program07 @ yeah.net

CS108: mxhe1 @ yeah.net

I2ML(H) Spring 2023 (CS104|CS108)

# 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 three normal forms

- Negation normal form (NNF)
- Conjunctive normal forms (CNF)
- Disjunctive normal forms (DNF) -- minor

**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 7B.1

# Negation Normal Form

# Negation Normal Form(NNF)

### Definition 7B.1

*A formula is in NNF if $\neg$ appears only in front of the propositional variables.*

### Theorem 7B.1

*For every formula $F$, there is a formula $F'$ in NNF such that $F \equiv F'$.*

### Proof.

Due to the standard equivalences, we can always push $\neg$ under the logical connectives. □

### Example 7B.1

*Consider* $\neg(q \Rightarrow ((p \vee \neg s) \oplus r))$

$$\equiv q \wedge \neg((p \vee \neg s) \oplus r) \equiv q \wedge (\neg(p \vee \neg s) \oplus r) \equiv q \wedge ((\neg p \wedge \neg\neg s) \oplus r) \equiv q \wedge ((\neg p \wedge s) \oplus r)$$

- There are negations hidden inside $\oplus$, $\Rightarrow$, and $\Leftrightarrow$.
  Sometimes users want to also remove the symbols, while producing NNF.

Often we assume that the formulas are in NNF.

# Exercise : NNF

## Thinking Exercise 7B.1

*Convert the following formulas into NNF*

- $\neg(p \Rightarrow q)$
- $\neg(\neg((s \Rightarrow \neg(p \Leftrightarrow q))) \oplus (\neg q \vee r))$
- $\neg((p \Rightarrow q) \Rightarrow (p \Rightarrow q))$
- $\neg\neg\neg p$

## Thinking Exercise 7B.2

*In the above exercises, remove $\Rightarrow$, $\Leftrightarrow$, and $\oplus$ before turning the above into NNF.*

## Thinking Exercise 7B.3*

*Let us suppose we have access to the parse tree of a formula, which is represented as a directed acyclic graph (DAG) (*not as a tree*). Write an algorithm that produces negation normal form (NNF) of the formula in linear time in the size of the DAG. You may assume the costs of reading from and writing to a map data structure are constant time.*

# Formal Derivation for NNF

## Theorem 7B.2

*Let $F'$ be the NNF 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 transformation.

- Derivations for substitutions.
- Derivations for pushing negations inside connectives.

Therefore, we have the derivations. $\square$

Topic 7B.2

# 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 7B.2

- *$p$ 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 7B.2

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

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

## Example 7B.3

- $\neg p$ *and* $p$ *both are in CNF.*
- $(p \vee \neg q) \wedge (r \vee \neg q) \wedge \neg r$ *in CNF.*
- $\{(p \vee \neg q), (r \vee \neg q), \neg r\}$ *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.

## Thinking Exercise 7B.4

*a. Write a formal grammar for CNF.*

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

# CNF Conversion

### Theorem 7B.3

*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 $\land$ and $\lor$.

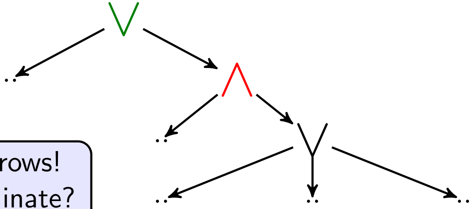
...

**Commentary:** We will assume flattened $\land$ and $\lor$. 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. $\square$

Are we done?

## Example 7B.4

*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.

# CNF Conversion Terminates

## Theorem 7B.4

*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

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

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?) □

### Thinking Exercise 7B.5

*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 $F(G)$.*

### Thinking Exercise 7B.6

*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 $\nu$ and replacing with many subformulas with smaller $\nu$. Therefore, the process will terminate. The formal construction is left for the exercise.

# Formal Derivation for CNF

**Theorem 7B.5**

*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. $\square$

# 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 7B.5

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

## Thinking Exercise 7B.7

*a. Give a linear time algorithm to prove validity of a CNF formula*

*b. What is the interpretation of the empty set of clauses?*

# Topic 7B.3

## 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.

https://zhuanlan.zhihu.com/p/401411901

**Commentary:** Tseitin is also spelled Tseytin.

# Tseitin Encoding : Intuition

## Example 7B.6

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

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

$$(p \lor x) \land (x \Rightarrow (q \land r))$$

*After simplification,*

$$(p \lor x) \land (\neg x \lor q) \land (\neg x \lor r)$$

## Thinking Exercise 7B.8

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

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

c. *Show that transformation from* $(F \lor \neg G)$ *to* $(F \lor \neg x) \land (G \Rightarrow x)$ *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 $G_1 \land \cdots \land G_n$ that is just below an $\lor$ in $F(G_1 \land \cdots \land G_n)$
3. Replace $F(G_1 \land .. \land G_n)$ by $F(p) \land (\neg p \lor G_1) \land .. \land (\neg p \lor G_n)$, where $p$ is a fresh variable
4. goto 2

## Thinking Exercise 7B.9

*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 7B.7

*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.*

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

### Thinking Exercise 7B.10

*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 7B.6

*if* $m \models F(p) \wedge (\neg p \vee G_1) \wedge \cdots \wedge (\neg p \vee G_n)$ *then* $m \models F(G_1 \wedge \cdots \wedge G_n)$

## Proof.

Assume $m \models F(p) \wedge (\neg p \vee G_1) \wedge \cdots \wedge (\neg p \vee G_n)$. We have three cases.

First case $m \models p$:

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

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

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

...

# Tseitin encoding preserves satisfiability(contd.)

## Proof(contd.)

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

- Since $F(G_1 \land \cdots \land G_n)$ 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 $G_i$s, $m[p \mapsto 1] \models G_1 \land \cdots \land G_n$.
- Due to the substitution theorem, $m[p \mapsto 1] \models F(G_1 \land \cdots \land G_n)$
- Therefore, $m \models F(G_1 \land \cdots \land G_n)$. $\square$

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

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

## Thinking Exercise 7B.11

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

Topic 7B.4

# Disjunctive Normal Form

# Disjunctive Normal Form(DNF)

## Definition 7B.3

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

## Theorem 7B.7

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

## Proof.

Proof is similar to CNF. $\square$

## Thinking Exercise 7B.12

*a. Give the formal grammar of DNF.*

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

# Topic 7B.5

## Problems

# Monotonic NNF

### Definition 7B.4

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

### Example 7B.8

$$pos(\neg p_2 \land (p_1 \lor p_2), \{p_1 \mapsto 1, p_2 \mapsto 0\}) = \{p_1, \neg p_2\}$$

### Thinking Exercise 7B.13

*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* $m' \models F$*.*

# CNF and DNF

**Thinking Exercise 7B.14**

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

# CNF

## Thinking Exercise 7B.15

*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

### Thinking Exercise 7B.16

*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**

### Thinking Exercise 7B.17

*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.

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

## Thinking Exercise 7B.18

*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***

### Thinking Exercise 7B.19

*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.*

### Thinking Exercise 7B.20

*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***

### Thinking Exercise 7B.21

*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.

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

## Thinking Exercise 7B.22

*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 7B.6

## Supporting Slides

# König Lemma

## Theorem 7B.8

*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 $v_1, v_2, v_3, \ldots$ 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 $v_1..v_{i-1} \notin G_i$.
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 $G_{i+1} \triangleq G_i|_S$. $\square$

## Thinking Exercise 7B.23

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

End of Lecture 7B