CS228 Logic for Computer Science 2020

Lecture 7: Conjunctive Normal Form

Instructor: Ashutosh Gupta

IITB, India

Compile date: 2020-02-01

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

We have seen equivalences that remove $\oplus$, $\Rightarrow$, and $\Leftrightarrow$ from a formula.

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

In the lecture, we will assume you can remove them at will.

Commentary: Note that removal of $\oplus$ and $\Leftrightarrow$ blows up the formula size. Their straight up removal is not desirable.

Topic 7.1

Negation normal form

Negation normal form(NNF)

Definition 7.1

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

Theorem 7.1

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

Proof.

Due to the equivalences, we can always push $\neg$ under the connectives $\square$

- Often we assume that the formulas are in NNF.
- However, there are negations hidden inside $\oplus$, $\Rightarrow$, and $\Leftrightarrow$. Sometimes, the symbols are also expected to be removed while producing NNF

Exercise 7.1

Write an efficient algorithm to convert a propositional formula to NNF?

Commentary: In our context, we will not ask one to remove e $\oplus$, $\Rightarrow$, and $\Leftrightarrow$ during conversion to NNF.

Example :NNF

Example 7.1

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

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

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

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

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

Exercise 7.2

Convert the following formulas into NNF

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

Exercise 7.3

Remove $\Rightarrow$, $\Leftrightarrow$, *and* $\oplus$ *before turning the above into NNF.*

Exercise 7.4

Are there any added difficulties if the formula is given as a DAG not as a tree?

Formal derivation for NNF

Theorem 7.2

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 7.2

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 7.2

- *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 7.2

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 7.3

- $\neg p$ *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.*

Exercise 7.5

Write a formal grammar for CNF

Attendance quiz

Which of the following formulas are in CNF?

p, $\neg p$, $p \vee \neg p$, $p \vee q$, $p \wedge q$, $\neg p \wedge q$, $(p \vee q) \wedge p$, $(p \vee q) \wedge (\neg p \wedge q)$, $(p \vee q) \wedge (\neg p \vee q)$, $(p \wedge q) \wedge (\neg p \wedge q)$ $(p \vee q) \vee (\neg p \vee q)$,

$\neg(p \vee q)$, $p \oplus q$, $p \Rightarrow q$, $p \Leftrightarrow q$, $(p \wedge q) \vee r$, $(p \wedge q) \vee (\neg p \vee q)$, $(p \vee q) \vee (\neg p \wedge r)$, $(p \wedge q) \vee (\neg p \wedge r)$, $\neg(p \wedge q)$,

CNF conversion

Theorem 7.3

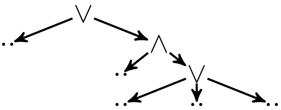
For every formula F there is another formula F' in CNF s.t. $F \equiv F'$.

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 $\wedge$ and $\vee$.

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



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

Since $\vee$ distributes over $\wedge$, we can always push $\vee$ inside $\wedge$.

Are we done?

Eventually, we will obtain a formula that is CNF. □

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

CNF conversion terminates

Theorem 7.4

The procedure of converting a formula in 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(s) are $\wedge$
- G_{ij} is either a literal or an $\vee$ formula

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

CNF conversion terminates (contd.)

(contd.)

Due to Köing lemma, the procedure terminates.(why?) □

Exercise 7.6

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 Köing lemma, show that the process always terminates.

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

CNF examples

Example 7.4

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

Exercise 7.7

Convert the following formulas into CNF

1. $\neg((p \Rightarrow q) \Rightarrow ((q \Rightarrow r) \Rightarrow (p \Rightarrow r)))$
2. $(p \Rightarrow (\neg q \Rightarrow r)) \wedge (p \Rightarrow \neg q) \Rightarrow (p \Rightarrow r)$

Formal derivation for CNF

Theorem 7.5

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 removes $\Rightarrow$, $\oplus$, and $\Leftrightarrow$
- Derivations for substitutions that flattens $\wedge$ and $\vee$
- Derivations for substitutions that applies 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 naturally extend definition of the clauses to the empty set of literals. We refer to $\perp$ as empty clause.

Example 7.5

- $(p \wedge q \wedge \neg r)$ *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*

Exercise 7.8

a. *Show F' obtained from the procedure may be exponentially larger than F*
b. *Give a linear time algorithm to prove validity of a CNF formula*
c. *What is the interpretation of empty set of clauses?*

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's encoding

But, with a cost.

Tseitin's encoding (Plaisted-Greenbaum optimization included)

We can translate every formula into CNF without exponential explosion using Tseitin's encoding by introducing fresh variables.

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 a $\vee$ 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 7.9

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

Commentary: 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's 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's 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 7.10

Convert the following formulas into CNF using Tseitin's encoding

1. $(p \Rightarrow (\neg q \wedge r)) \wedge (p \Rightarrow \neg q)$
2. $(p \Rightarrow q) \vee (q \Rightarrow \neg r) \vee (r \Rightarrow q) \Rightarrow \neg(\neg(q \Rightarrow p) \Rightarrow (q \Leftrightarrow r))$

Tseitin's encoding preserves satisfiability

Theorem 7.6

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

Proof.

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

First case $m \models p$:

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

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

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

...

Tseitin's encoding preserves satisfiability(contd.)

Proof(contd.)

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

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

Exercise 7.11

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

Wisdom: any transformation that introduces a fresh symbols most likely looses either equisatisfiability or equivalidity.

Topic 7.3

Disjunctive normal form

Disjunctive normal form(DNF)

Definition 7.3

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

Theorem 7.7

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

Proof.

Proof is similar to CNF. □

Exercise 7.12

a. Give the formal grammar of DNF

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

Topic 7.4

Problems

CNF and DNF

Exercise 7.13

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

CNF

Exercise 7.14

Convert the following formulas into CNF

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

CNF vs. DNF

Exercise 7.15

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 7.16

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.

P=NP argument

Exercise 7.17

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

Tseitin's 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.

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 7.18

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.

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.

Validity

Exercise 7.21

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

Exercise: linear NNF transformation

Exercise 7.22

*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 terms of the size of the DAG. You may assume the cost of reading from and writing to a map data structure is constant time.*

Topic 7.5

Supporting slides

Köing's Lemma

Theorem 7.8

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 $v_1, v_2, v_3, ...$ as follows.

base case:

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

induction step:

1. Assume we have a 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 $G_{i+1} \triangleq G_i|_S$. □

Exercise 7.23

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

End of Lecture 7