Predicate Logic, Normal Forms (CNF, DNF)

Michael L. Littman

CS 22 2020

March 2, 2020

Overview

Arguing either way

(A) If I eat a chocolate donut, I will be hyper for the rest of the day. P IMPLIES Q.

(B) If I am not hyper for the rest of the day, I didn't eat a chocolate donut. NOT(Q) IMPLIES NOT(P).

Same? Let's check.

P	Q	P IMPLIES Q	NOT(Q) IMPLIES NOT(P)
F	**F**	**T**	**T**
F	**T**	**T**	**F**
T	**F**	**F**	**T**
T	**T**	**T**	**F**

Arguing either way

(A) If I eat a chocolate donut, I will be hyper for the rest of the day. P IMPLIES Q.

(B) If I am not hyper for the rest of the day, I didn't eat a chocolate donut. NOT(Q) IMPLIES NOT(P).

Same? Let's check.

P	Q	P IMPLIES Q	NOT(Q)	IMPLIES	NOT(P)
F	**F**	**T**	**T**	**T**	**T**
F	**T**	**T**	**F**	**T**	**T**
T	**F**	**F**	**T**	**F**	**F**
T	**T**	**T**	**F**	**T**	**F**

The *implication* (A) and its *contrapositive* (B) are equivalent.

Converse

(A) If I eat a chocolate donut, I will be hyper for the rest of the day. P IMPLIES Q.

(C) If I am hyper for the rest of the day, I ate a chocolate donut. Q IMPLIES P.

Same? Let's check.

P	Q	P IMPLIES Q	Q IMPLIES P
F	**F**	**T**	**F**
F	**T**	**T**	**F**
T	**F**	**F**	**T**
T	**T**	**T**	**T**

Can't argue both ways

(A) If I eat a chocolate donut, I will be hyper for the rest of the day. P IMPLIES Q.

(C) If I am hyper for the rest of the day, I ate a chocolate donut. Q IMPLIES P.

Same? Let's check.

P	Q	P IMPLIES Q	Q	IMPLIES	P
F	**F**	**T**	**F**	**T**	**F**
F	**T**	**T**	**T**	**F**	**F**
T	**F**	**F**	**F**	**T**	**T**
T	**T**	**T**	**T**	**T**	**T**

The implication (A) and its *converse* (C) are not equivalent.

Both boths

The conjunction of an implication and its converse is an if and only if.

(A) If I eat a chocolate donut, I will be hyper for the rest of the day. P IMPLIES Q.

(C) If I am hyper for the rest of the day, I ate a chocolate donut. Q IMPLIES P.

(D) I am hyper for the rest of the day if and only if I ate a chocolate donut. Q IFF P.

Equivalence and validity: Definitions

A formula can be thought of as a function mapping variable assignments to truth values. Each row of the truth table shows one input and its corresponding output.

Definition: Two formulas over the same set of variables are *equivalent* if they evaluate to the same value regardless of variable assignment.

Definition: A formula is *valid* if it is always true regardless of variable assignment.

Example: P OR NOT(P).

P	P	OR	NOT(P)
F	**F**	**T**	**T**
T	**T**	**T**	**F**

Equivalence and validity

A formula is valid iff it is equivalent to **T**.

Two formulas α and β are equivalent iff α IFF β is valid.

Example: Prove "P" is equivalent to "NOT(NOT(P))".

P	P IFF NOT(NOT(P))
F	
T	

Equivalence and validity

A formula is valid iff it is equivalent to **T**.

Two formulas α and β are equivalent iff α IFF β is valid.

Example: Prove "P" is equivalent to "NOT(NOT(P))".

P	P IFF NOT(NOT(P))
F	**T**
T	**F**

Equivalence and validity

A formula is valid iff it is equivalent to **T**.

Two formulas α and β are equivalent iff α IFF β is valid.

Example: Prove "P" is equivalent to "NOT(NOT(P))".

P	P IFF NOT(NOT(P))
F	**F** **F** **T**
T	**T** **T** **F**

Equivalence and validity

A formula is valid iff it is equivalent to **T**.

Two formulas α and β are equivalent iff α IFF β is valid.

Example: Prove "P" is equivalent to "NOT(NOT(P))".

P	P IFF NOT(NOT(P))
F	**F T F T**
T	**T T T F**

Satisfiability

Definition: A formula is *satisfiable* if at least one assignment evaluates to true.

A formula is satisfiable iff its negation is not valid. (DeMorgan's law in another form.)

Validity is kind of like "$\forall$".

Satisfiability is kind of like "$\exists$".

Determining whether a formula is satisfiable, efficiently, is a core problem in computer science. Examples: Solving puzzles, finding successful plans, arranging items in space, factoring, finding paths in graphs...

Checking satisfiability

Easy if few variables. Just write out the truth table!

P	Q	NOT(Q)	AND	(Q	OR	NOT(P))
F	**F**	**T**	**T**	**F**	**T**	**T**
F	**T**	**F**	**F**	**T**	**T**	**T**
T	**F**	**T**	**F**	**F**	**F**	**F**
T	**T**	**F**	**F**	**T**	**T**	**F**

Blows up as the number of variables gets large. Need another way.

Disjunctive normal form

Definition: A formula in *disjunctive normal form* is an OR of terms, where each term is an AND of variables or negations of variables.

(A AND B AND NOT(C)) OR (NOT(B) AND C)

A	B	C	value
F	**F**	**F**	**F**
F	**F**	**T**	**T**
F	**T**	**F**	**F**
F	**T**	**T**	**F**
T	**F**	**F**	**F**
T	**F**	**T**	**T**
T	**T**	**F**	**T**
T	**T**	**T**	**F**

Disjunctive normal form is universal

Theorem: All formulas can be written in disjunctive normal form.

Proof: You can read the terms off of the truth table, turning each "true" row into a conjunction of literals. QED.

A	B	C	value	
F	**F**	**F**	**F**	
F	**F**	**T**	**T**	$\leftarrow$ NOT(A) AND NOT(B) AND C
F	**T**	**F**	**F**	
F	**T**	**T**	**F**	
T	**F**	**F**	**F**	
T	**F**	**T**	**T**	$\leftarrow$ A AND NOT(B) AND C
T	**T**	**F**	**T**	$\leftarrow$ A AND B AND NOT(C)
T	**T**	**T**	**F**	

Properties of disjunctive normal form

How big could the disjunctive normal form get? Big. 2^n terms if there are n variables.

Definition: If every variable appears in every term in a disjunctive normal form expression, then it is in *full disjunctive normal form*.

Book	Wikipedia/me
disjunctive form	disjunctive normal form
disjunctive normal form	full disjunctive normal form

Given a formula in DNF (disjunctive normal form), can we determine whether it is satisfiable? Valid? Satisfiability is easy—a single term tells us the a satisfying assignment. Validity is not obvious—a given term might exclude an assignment, but perhaps another picks it up? What if it's in full DNF? Then, we'd need a term for each "true" row in the truth table: valid iff 2^n rows.

DeMorgan's Laws

1. DeMorgan: NOT(A AND B) = NOT(A) OR NOT(B)
2. Double negation: NOT(NOT(A)) = A
3. DeMorgan: NOT(A OR B) = NOT(A) AND NOT(B)

Proof of DeMorgan 3 from DeMorgan 1 and double negation:

NOT(A OR B)	
= NOT(NOT(NOT(A)) OR NOT(NOT(B)))	double double negation
= NOT(NOT(NOT(A) AND NOT(B)))	DeMorgan 1
= NOT(A) AND NOT(B)	double negation

Conjunctive normal form

Definition: A formula in *conjunctive normal form* is an AND of clauses, where each clause is an OR of variables or negations of variables.

(NOT(A) OR NOT(B) OR C) AND (B OR NOT(C))

A	B	C	value
F	**F**	**F**	**T**
F	**F**	**T**	**F**
F	**T**	**F**	**T**
F	**T**	**T**	**T**
T	**F**	**F**	**T**
T	**F**	**T**	**F**
T	**T**	**F**	**F**
T	**T**	**T**	**T**

Conjunctive normal form is universal

Theorem: All formulas can be written in conjunctive normal form.

Proof: Negate the truth table. Compute DNF. Negate formula via DeMorgan's law. QED.

A	B	C	value	
F	**F**	**F**	**T**	
F	**F**	**T**	**F**	$\leftarrow A$ OR B OR NOT(C)
F	**T**	**F**	**T**	
F	**T**	**T**	**T**	
T	**F**	**F**	**T**	
T	**F**	**T**	**F**	$\leftarrow$ NOT(A) OR B OR NOT(C)
T	**T**	**F**	**F**	$\leftarrow$ NOT(A) OR NOT(B) OR C
T	**T**	**T**	**T**	

Properties of conjunctive normal form

How big could the conjunctive normal form get? Big. 2^n clauses if there are n variables.

Definition: If every variable appears in every clause in a conjunctive normal form expression, then it is in *full conjunctive normal form*.

Book	Wikipedia/me
conjunctive form	conjunctive normal form
conjunctive normal form	full conjunctive normal form

Given a formula in CNF (conjunctive normal form), can we determine whether it is satisfiable? Valid? Satisfiability is not so clear—each clause knocks out some assignments, but not clear if the set of clauses miss anything. Validity is easy now—a single clause throws out an assignment, so a single clause makes the formula not valid.

Some algebraic rewrite rules

- Commutativity: A AND $B = B$ AND A, A OR $B = B$ OR A.
- Associativity: $(A$ AND $B)$ AND $C = A$ AND $(B$ AND $C)$, $(A$ OR $B)$ OR $C = A$ OR $(B$ OR $C)$.
- Identity: **T** AND $A = A$, **F** OR $A = A$.
- Zero: **F** AND $A =$ **F**, **T** OR $A =$ **T**.
- Idempotence: A AND $A = A$, A OR $A = A$.
- Complements: A AND NOT$(A) =$ **F**, A OR NOT$(A) =$ **T**.
- Distributivity: A AND $(B$ OR $C) = (A$ AND $B)$ OR $(A$ AND $C)$.

Converting to disjunctive normal form

$$
\begin{aligned}
&\text{NOT}((A \text{ AND } B) \text{ OR } (A \text{ AND } C)) \\
&= \text{NOT}(A \text{ AND } B) \text{ AND NOT}(A \text{ AND } C) && \text{DeMorgan} \\
&= (\text{NOT}(A) \text{ OR NOT}(B)) \\
&\quad \text{AND } (\text{NOT}(A) \text{ OR NOT}(C)) && \text{DeMorgan} \\
&= (\text{NOT}(A) \text{ OR NOT}(B)) \text{ AND NOT}(A) \\
&\quad \text{OR } (\text{NOT}(A) \text{ OR NOT}(B)) \text{ AND NOT}(C)) && \text{Distributive} \\
&= (\text{NOT}(A) \text{ AND NOT}(A)) \\
&\quad \text{OR } (\text{NOT}(B) \text{ AND NOT}(A)) \\
&\quad \text{OR } (\text{NOT}(A) \text{ AND NOT}(C)) \\
&\quad \text{OR } (\text{NOT}(B) \text{ AND NOT}(C)) && \text{Distributive}
\end{aligned}
$$

We converted the formula into disjunctive normal form. In fact, it's a general strategy for doing so: (1) use DeMorgan to push the "NOT"s all the way down the the propositions. (2) use distributive to "multiply out" all the clauses.

Completeness: Rewrite rule sufficiency

Theorem: Any propositional formula can be transformed into disjunctive normal form or a conjunctive normal form using the rewrite rules discussed.

Proof sketch: DNF is achieved using DeMorgan and distributive law. The idea for conjunctive normal form is that we can convert the *negation* to DNF, then negate the result with DeMorgan's law to get CNF.

Theorem: Two propositional formula are equivalent iff they can be proved equivalent using the rewrite rules.

Proof sketch: Since the rewrite rules are correct, if they say we have equivalence, we have equivalence. If two formulas are equivalent, we can show them to be so by converting both to DNF and checking that they match.

Caveats

Rewrite rules can lead to much shorter proofs than using the truth table method.

But, there is no guarantee.

In fact, both can end up being extremely tedious.