COMPSCI 230: Discrete Mathematics for Computer Science	January 23, 2019
Lecture 4	
Lecturer: Debmalya Panigrahi	*Scribe: Kevin Sun*

1 Overview

In this lecture, we give an introduction to propositional logic, which is the mathematical formalization of logical relationships. We also discuss the disjunctive and conjunctive normal forms, how to convert formulas to each form, and conclude with a fundamental problem in computer science known as the satisfiability problem.

2 Propositional Logic

Until now, we have seen examples of different proofs by primarily drawing from simple statements in number theory. Now we will establish the fundamentals of writing formal proofs by reasoning about logical statements.

Throughout this section, we will maintain a running analogy of propositional logic with the system of arithmetic with which we are already familiar. In general, capital letters (e.g., A, B, C) represent propositional variables, while lowercase letters (e.g., x, y, z) represent arithmetic variables.

But what is a propositional variable? A propositional variable is the basic unit of propositional logic. Recall that in arithmetic, the variable x is a placeholder for any real number, such as 7, 0.5, or -3. In propositional logic, the variable A is a placeholder for any truth value. While x has (infinitely) many possible values, there are only two possible values of A: True (denoted **T**) and False (denoted **F**). Since propositional variables can only take one of two possible values, they are also known as *Boolean* variables, named after their inventor George Boole.

By letting x denote any real number, we can make claims like "$x^2 - 1 = (x + 1)(x - 1)$" regardless of the numerical value of x. Similarly, as we shall see, letting A, B, C, etc. denote Boolean variables allows us to make claims regarding these variables regardless of their truth values.

2.1 Combining Propositions

In arithmetic, the basic operators are addition ($+$) and multiplication ($\times$). Using these operators, we can define new variables such as "$z = x^2 + y$." Similarly, in propositional logic, the basic operators are AND (denoted by $\wedge$), OR ($\vee$), and NOT ($\neg$). This allows us to combine propositions to create new, more complicated propositions.

Furthermore, arithmetic contains shorthand that enables us to write longer formulas more concisely. For example, x^4 is shorthand for $x \times x \times x \times x$. Similarly, in propositional logic, we have notational shorthand, e.g.:

- $A \to B$ is shorthand for $\neg A \vee B$,
- $A \leftrightarrow B$ is shorthand for $(A \to B) \wedge (B \to A)$.

(Note that the "$\neg$" in "$\neg A \vee B$" modifies "A", not "$A \vee B$". In the latter case, we would write "$\neg(A \vee B)$". We will elaborate on this order of operations in the following section.)

In our example above, if we define z as $x^2 + y$, then if $x = 2$ and $y = 1$, we must have $z = 5$ in order to obey the rules of addition and multiplication. Similarly, there are rules that must be obeyed when creating propositions using $\wedge$, $\vee$, and $\neg$. These are given below, in the form of truth tables:

A	B	$A \wedge B$	$A \vee B$
T	**T**	**T**	**T**
T	**F**	**F**	**T**
F	**T**	**F**	**T**
F	**F**	**F**	**F**

A	$\neg A$
T	**F**
F	**T**

Note that the truth tables given above match our understanding of the corresponding words in English: $A \wedge B$ is true if both A and B are true, $A \vee B$ is true if either A or B (or both) is true, and $\neg A$ is simply the opposite of A.

Note: Although the mathematical definitions align with the English meaning in this case, in general, it is not a good idea to rely on intuition when reasoning about logical statements. In fact, the whole point of propositional logic is to remove our dependence on English, which is full of logical ambiguities. For example, what do "Do you want cake or ice cream?" and "We will go golfing unless it rains" mean, exactly?

Now we fill out the truth table for $A \to B$, also known as "A implies B," "If A, then B," and "A is sufficient for B." Recall this expression is defined as $\neg A \vee B$, so if A is **F** then $A \to B$ is **T**, and if A is **T** then $A \to B$ is the value of B. This yields the following truth table:

A	B	$A \to B$
T	**T**	**T**
T	**F**	**F**
F	**T**	**T**
F	**F**	**T**

Similarly, we can reason about the truth table for $A \leftrightarrow B$, defined as $(A \to B) \wedge (B \to A)$. The expression "$A \leftrightarrow B$" is also known as "$A$ if and only if B", "A iff B," "A is equivalent to B,", and "$A = B$." It is not too difficult to see that if A and B have the same truth value, then $A \leftrightarrow B$ is **T**, and otherwise, $A \leftrightarrow B$ is **F**. Again, this corresponds to the intuitive idea of "equivalence" in English. The truth table is below:

A	B	$A \leftrightarrow B$
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**T**

If the above truth table is not apparent, it is a good idea to fill in the details by first writing the truth table for $C = (A \to B)$ and $D = (B \to A)$, and then considering $C \wedge D$. The following section contains more opportunities to practice filling out truth tables.

2.2 Axiomatic Rules

In this section, we present rules that allow us to manipulate logical expressions while maintaining logical equivalence. Most, but not all of the rules have analogous counterparts in arithmetic, and we will point these out as they appear. Most rules also have reasonable interpretations in English, but as noted above, we should not rely on semantics in the English language when reasoning about mathematics.

Note: Just like in arithmetic (e.g., $3 + 2 \times 5$ is 13, not 25), logical symbols follow a standard order of operations. In decreasing order of precedence, this is: $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$. For clarity, the statement of the rules below contain parentheses, some of which are unnecessary. Also, as noted above, we often use $A = B$ and $A \leftrightarrow B$ interchangeably.

1. Commutativity: The $\wedge$ and $\vee$ operators are commutative, which means

$$A \wedge B = B \wedge A \quad \text{and} \quad A \vee B = B \vee A.$$

   (In arithmetic, we know $2 + 3 = 3 + 2$ and $2 \times 3 = 3 \times 2$.)

2. Associativity: The $\wedge$ and $\vee$ operators are associative, which means

$$\Big((A \wedge B) \wedge C\Big) = \Big(A \wedge (B \wedge C)\Big) \quad \text{and} \quad \Big((A \vee B) \vee C\Big) = \Big(A \vee (B \vee C)\Big).$$

   (In arithmetic, we know $(3 + 4) + 2 = 3 + (4 + 2)$ and $(3 \times 4) \times 2 = 3 \times (4 \times 2)$.) Together, commutativity and associativity essentially mean "order doesn't matter."

3. Idempotence: The $\wedge$ and $\vee$ operators are idempotent, which means

$$A \wedge A = A \text{ and } A \vee A = A.$$

   (This rule has no analogy in arithmetic: $3 + 3$ is not 3, and neither is 3×3.)

4. Identity: The truth value **T** is the identity for $\wedge$ and **F** is the identity for $\vee$. In other words,

$$(A \wedge \mathbf{T}) = A, \quad \text{and} (A \vee \mathbf{F}) = A.$$

   Moreover, **F** is the "zero" for $\wedge$ and **T** makes $\vee$ valid:

$$(A \wedge \mathbf{F}) = \mathbf{F}, \quad (A \vee \mathbf{T}) = \mathbf{T}.$$

   (We know $3 + 0 = 3$, $3 \times 1 = 3$, and $3 \times 0 = 0$. There is no counterpart of the last rule.)

5. Distributivity: The operators $\wedge$ and $\vee$ distribute over each other, which means

$$A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C) \quad \text{and} \quad A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C).$$

   (In arithmetic, the analogy only applies in one case: $3 \times (2 + 5) = 3 \times 2 + 3 \times 5$, but the statement is not true if we swap the $\times$ and $+$ symbols.)

6. Negations: The $\neg$ operator cancels itself, which means

$$\neg(\neg A) = A.$$

(In arithmetic, we know $-(-5) = 5$.)

7. Tautologies and Contradictions: A tautology is a logical expression that always has truth value **T**, such as $A \vee \neg A$. A contradiction is a logical expression that always has truth value **F**, such as $A \wedge \neg A$. (There is no illustrative analogy in arithmetic.)

8. De Morgan's Laws: Named after Augustus De Morgan, these rules allow us to distribute the $\neg$ operator over $\wedge$ and $\vee$. More specifically, we have

$$\neg(A \vee B) = \neg A \wedge \neg B \quad \text{and} \quad \neg(A \wedge B) = \neg A \vee \neg B.$$

In other words, the $\neg$ gets distributed and the corresponding operator "flips." (There is no illustrative example in arithmetic.)

Again, every one of these equivalences can be formally verified by completing the corresponding truth tables for two equivalent expressions. Each entry of these tables can be derived from the basic principles given in Section 2.1.

3 Disjunctive and Conjunctive Normal Forms

The rules given in the previous section (such as De Morgan's) allow us to transform propositional formulas (expressions) to equivalent formulas. In this lecture, we will see two "standard forms" that can represent every propositional formula.

Definition 1. *A propositional formula is in* disjunctive normal form (DNF) *if it is the disjunction of conjunctions of literals.*

On its own, this definition may seem somewhat cryptic, so we explain each term below:

- A *literal* is a Boolean variable, or the negation of a Boolean variable (e.g., P, $\neg P$).
- A *conjunction* is a logical formula that is the AND ($\wedge$) of (smaller) formulas (e.g., $P \wedge \neg Q \wedge R$).
- A *disjunction* is a logical formula that is the OR ($\vee$) of (smaller) formulas (e.g., $P \vee \neg Q \vee R$).

With this in mind, the meaning of DNF should be clearer: a DNF formula is an OR of AND's. For example,

$$X = (A \wedge \neg B \wedge \neg D) \vee (B \wedge C) \vee (C \wedge \neg D \wedge E)$$

is a DNF formula: the literals of X are $A, \neg B, \neg D, B, C, \neg D$, and E; the conjunctions are $A \wedge \neg B \wedge \neg D$, $B \wedge C$, and $C \wedge \neg D \wedge E$; and X is the disjunction (OR) of these three conjunctions. On the other hand,

$$Y = (A \vee \neg B) \wedge (B \vee C \vee \neg D)$$

is not a DNF formula, because the structure of the operators is inverted. But this formula *is* in the other "standard" form (CNF) for propositional formulas.

Definition 2. *A propositional formula is in* conjunctive normal form (CNF) *if it is the conjunction of disjunctions of literals.*

As noted above, Y is a CNF formula because it is an AND of OR's. The literals of Y are $A, \neg B, B, C$, and $\neg D$; the disjunctions are $(A \vee \neg B)$ and $(B \vee C \vee \neg D)$; and Y is the conjunction (AND) of the two disjunctions.

3.1 Converting to DNF

We now describe a procedure that converts any propositional formula X into disjunctive normal form. Note that the resulting DNF is not necessarily the shortest expression equivalent to X (or even the shortest DNF equivalent to X), but the result is guaranteed to be a DNF formula.

We illustrate the procedure on the following example:

$$X = \neg\Big((A \vee B) \wedge (A \vee C)\Big).$$

The first step is to write the truth table of X:

A	B	C	X
T	**T**	**T**	**F**
T	**T**	**F**	**F**
T	**F**	**T**	**F**
T	**F**	**F**	**F**
F	**T**	**T**	**F**
F	**T**	**F**	**T**
F	**F**	**T**	**T**
F	**F**	**F**	**T**

(Note that if $A = \mathbf{T}$, then the $\wedge$ "succeeds", so $X = \mathbf{F}$. Otherwise, the $\wedge$ "succeeds" only if $B = C = \mathbf{T}$, and so $X = \mathbf{F}$ in this case and $X = \mathbf{T}$ in all remaining rows.)

The second step is to identify the rows that contain $X = \mathbf{T}$; in our case, these are rows 6, 7, and 8. Finally, we encode each row as a conjunction of the corresponding literals, and the final result is the disjunction of these conjunctions. Intuitively, we can think of the DNF we're constructing as the following expression:

$$(\text{row } 6) \vee (\text{row } 7) \vee (\text{row } 8).$$

In our example, we have the following encoding:

$$\begin{aligned}
&\text{row } 6 : (\neg A \wedge B \wedge \neg C) \\
&\text{row } 7 : (\neg A \wedge \neg B \wedge C) \\
&\text{row } 8 : (\neg A \wedge \neg B \wedge \neg C).
\end{aligned}$$

Taking the disjunction of these conjunctions yields the following DNF formula for X:

$$(\neg A \wedge B \wedge \neg C) \vee (\neg A \wedge \neg B \wedge C) \vee (\neg A \wedge \neg B \wedge \neg C).$$

3.2 Converting to CNF

Similarly, we can convert any propositional formula into conjunctive normal form. We will illustrate the process on the same propositional formula X from the previous subsection. In this case, the intuition is the following: $X = \mathbf{T}$ if the truth assignment does not belong to any of the rows that evaluate to **F**.

More specifically, we first write the truth table, as before. We then identify the rows that contain $X = \mathbf{F}$; in our case, these are rows 1, 2, 3, 4, and 5. Finally, we "negate" each row and return the conjunction over these "negated" rows. Intuitively, we can think of the CNF we're constructing as the following expression:

$$(\text{NOT row } 1) \wedge (\text{NOT row } 2) \wedge (\text{NOT row } 3) \wedge (\text{NOT row } 4) \wedge (\text{NOT row } 5).$$

Again, the intuition is the following: if the truth assignment for A, B, C does not correspond to row 1, nor 2, nor 3, nor 4, nor 5, then $X = \mathbf{T}$. Our encoding of the rows is given below:

$$\begin{aligned}
\text{NOT row } 1 &: \neg(A \wedge B \wedge C) = \neg A \vee \neg B \vee \neg C \\
\text{NOT row } 2 &: \neg(A \wedge B \wedge \neg C) = \neg A \vee \neg B \vee C \\
\text{NOT row } 3 &: \neg(A \wedge \neg B \wedge C) = \neg A \vee B \vee \neg C \\
\text{NOT row } 4 &: \neg(A \wedge \neg B \wedge \neg C) = \neg A \vee B \vee C \\
\text{NOT row } 5 &: \neg(\neg A \wedge B \wedge C) = A \vee \neg B \vee \neg C.
\end{aligned}$$

Notice that we've applied De Morgan's Law and the double negation rule to produce a disjunction. The final CNF is the conjunction of these disjunctions:

$$(\neg A \vee \neg B \vee \neg C) \wedge (\neg A \vee \neg B \vee C) \wedge (\neg A \vee B \vee \neg C) \wedge (\neg A \vee B \vee C) \wedge (A \vee \neg B \vee \neg C).$$

3.3 Proving Equivalence of Formulas

A typical problem is the following: given two propositional formulas X and Y, are X and Y equivalent? That is, what is the truth value of the proposition $X \leftrightarrow Y$ (denoted by $X = Y$)?

Roughly speaking, there are three ways to prove that X and Y are equivalent:

1. Use a truth table to check if the columns corresponding to X and Y are identical.
2. Apply a sequence of rules to X until Y appears. (We can also "work backwards" from Y and meet X at some formula "in the middle.")
3. Convert X and Y to DNF (or CNF) using the procedure described above, and check if the resulting formulas are identical. This is essentially a special case of Method 2.

In general, Method 1 is the most straightforward, but also the most tedious. Method 2 often yields a more succinct proof that $X = Y$, but it is sometimes unclear which rules to apply. In this section, we explore Method 3; in particular, we will convert one propositional formula to DNF, and then to CNF. However, instead of writing the truth table and applying the procedures described above, we will apply the rules we learned in the previous lecture.

The propositional formula that we will consider is the following:

$$X = \neg\Big((A \vee \neg B) \wedge (\neg A \vee C)\Big).$$

As promised, we first convert X to DNF:

$$\begin{aligned} X &= \neg\Big((A \vee \neg B) \wedge (\neg A \vee C)\Big) \\ &= \neg(A \vee \neg B) \vee \neg(\neg A \wedge C) && \text{(De Morgan's for } \wedge) \\ &= (\neg A \wedge \neg\neg B) \vee (\neg\neg A \vee \neg C) && \text{(De Morgan's for } \vee \text{ and } \wedge) \\ &= (\neg A \wedge B) \vee (A \wedge \neg C). && \text{(double negation)} \end{aligned}$$

Note that this final formula is the OR of AND's, so it is indeed in DNF. Instead of writing the entire truth table of X, we simply applied De Morgan's Laws and the double negation rule a few times.

Now we continue where we left off, and convert X into CNF:

$$\begin{aligned} X &= (\neg A \wedge B) \vee (A \wedge \neg C) \\ &= \Big(\neg A \vee (A \wedge \neg C)\Big) \wedge \Big(B \vee (A \wedge \neg C)\Big) && \text{(distribution of } \vee \text{ over } \wedge) \\ &= \Big((\neg A \vee A) \wedge (\neg A \vee \neg C)\Big) \wedge \Big((B \vee A) \wedge (B \vee \neg C)\Big) && \text{(distribution of } \vee \text{ over } \wedge) \end{aligned}$$

Technically, this final expression is in CNF: it is the AND of four OR's, each of which has two literals. But recall that

$$(\neg P \vee P) = \mathbf{T} \quad \text{and} \quad \mathbf{T} \wedge Q = Q.$$

These two facts allow us to simplify our CNF for X to a shorter CNF:

$$X = (A \vee B) \wedge (\neg A \vee \neg C) \wedge (B \vee \neg C).$$

Note that for stylistic purposes, we have alphabetized the order of variable appearances, which we are allowed to do because $\wedge$ and $\vee$ are both commutative and associative.

3.4 The Satisfiability Problem

We now present one of the most well-known problems in computer science: the satisfiability problem, which we denote by SAT. In SAT, we are given a propositional formula X in CNF and must decide whether or not there exists a truth assignment to the variables of X that makes X evaluate to TRUE. If there is such an assignment, we return "Yes," and if there is no satisfying assignment, then we return "No."

For example, let's consider the propositional formula from the previous section:

$$X = (A \vee B) \wedge (\neg A \vee \neg C) \wedge (B \vee \neg C).$$

If we set $A = \mathbf{T}, B = \mathbf{T}, C = \mathbf{F}$, then $X = \mathbf{T}$, which means X is satisfiable, so we return "Yes." Since X only has a total of 6 literals, finding this assignment shouldn't take too long. But in general, how can we solve this problem?

The simplest solution is the following: given X, write the truth table for X, and if at any point we see a $\mathbf{T}$ in the column belonging to X, return "Yes." If every row has $X = \mathbf{F}$, then return "No." This algorithm clearly works, but if X has n variables, then its truth table has 2^n rows. In the worst case, we would check every row, so informally, the worst-case running time is roughly 2^n "steps."

But 2^n can be quite large: if $n = 100$, then 2^n has 31 digits! Computer scientists consider a solution to be *efficient* if its worst-case running time has a polynomial (e.g., $2n, n^2, 5n^3$) number of steps (or less). Since 2^n is not a polynomial, the "truth table" solution is not efficient.

Solving SAT efficiently is one formulation of the "P versus NP" problem, which is perhaps the most well-known unsolved problem in computer science. The first person to give an efficient algorithm for SAT (or prove that no such algorithm exists) will receive $1,000,000 from the Clay Mathematics Institute. An efficient algorithm for SAT would likely have significant consequences: online financial transactions would no longer be secure, and many modern-day security systems would crumble!

Before we conclude this section, let's consider the SAT problem with a slight modification: instead of the input being in CNF, assume we are given X in DNF. Then there's an efficient solution: for each conjunction of X, assign the variables of the conjunction with their appropriate truth values. If any conjunction evaluates to TRUE, then return "Yes," and otherwise, return "No."

For example, suppose we are given the same propositional formula X in DNF:

$$X = (\neg A \wedge B) \vee (A \wedge \neg C).$$

Looking at the first conjunction, if we simply set $A = \mathbf{F}$ and $B = \mathbf{T}$, then that conjunction is satisfied, so $X = \mathbf{T}$ and we can return "Yes." (If that assignment didn't work, we would try the next conjunction.) Since this algorithm essentially "scans" the input just once, it is efficient, so this version of SAT is easy. However, the original version of SAT (given a CNF) is still difficult, because the conversion process from a CNF to a DNF can require roughly 2^n steps.

4 Summary

In this lecture, we studied propositional logic, discussed the basic axiomatic rules for propositional logic, and learned how to verify these rules using truth tables. We also learned how to convert every propositional formula to DNF and CNF, and took a look at the SAT problem.