Introduction to the theory of computation
week 9 (Course Notes, chapters 5)

Propositional formulas

Last time we defined a set of well-formed propositional formulas,[1] in two steps:

1. Let $PV = \{v_1, v_2, \ldots\}$ be the (possibly infinite) set of propositional variables, which are strings over some alphabet. These stand for the most primitive propositions that we'll be considering, and they are assigned truth values independently of each other.

2. Our set of propositional formulas, $\mathcal{F}_{PV}$ is the smallest set such that:

   Basis: Any propositional variable in PV belongs to $\mathcal{F}_{PV}$.

   Induction step: If p_1 and p_2 are in $\mathcal{F}_{PV}$, then so are $\neg p_1$, $(p_1 \leftrightarrow p_2)$, $(p_1 \rightarrow p_2)$, $(p_1 \wedge p_2)$, and $(p_1 \vee p_2)$.

The Unique Readability Theorem (Course Notes) guarantees that there is exactly one way of parsing each propositional formula, but the cost is that we have to write a lot of parentheses. We add some rules of precedence to allow us to, informally, leave out a few parentheses without sacrificing clarity:

1. Remove the outer parentheses.

2. $\neg$ has higher precedence than the other connectives.

3. $\wedge$ and $\vee$ have higher precedence than $\rightarrow$ or $\leftrightarrow$, and $\wedge$ has higher precedence than $\vee$.

4. If you have several operators of the same type in a row, group to the right, so $p_1 \rightarrow p_2 \rightarrow p_3$ is parsed as $p_1 \rightarrow (p_2 \rightarrow p_3)$.

Applying the precedence rules:

$$(((\neg x \wedge (y \vee z))) \rightarrow (\neg((x \wedge z) \vee y) \rightarrow (x \vee z))) \iff$$
$$\neg x \wedge (y \vee z) \rightarrow \neg(x \wedge z \vee y) \rightarrow x \vee z$$

Now we know which propositional formulas are legal, but they are meaningless unless we assign truth values to the basic elements, the propositional variables. You may think of assigning truth values as setting up the current state of the universe — what's true, what's false.

Tabulating truth

Suppose you have propositional formulas, v_1 and v_2. There are four possible TRUTH ASSIGNMENTS to these two formulas, and each of these truth assignments extends to new formulas formed using the five connectives:

v_1	v_2	$\neg v_1$	$\neg v_2$	$(v_1 \wedge v_2)$	$(v_1 \vee v_2)$	$(v_1 \rightarrow v_2)$	$(v_1 \leftrightarrow v_2)$
0	0	1	1	0	0	1	1
0	1	1	0	0	1	1	0
1	0	0	1	0	1	0	0
1	1	0	0	1	1	1	1

How do v_1 and v_2 get their truth assignments? If they are propositional variables, then there is an arbitrary truth assignment, τ, corresponding to a particular row of any truth table that includes v_1 and v_2, and then τ extends to a truth assignment for any formulas built from v_1 and v_2. If v_1 and v_2 are not propositional variables, then their truth assignment depends on the truth assignment of the propositional variables they are formed from.

This tells us how to evaluate any formula, from the inside out, once you know the truth assignment of its propositional formulas. Here's another example, starting from propositional variables x, y, and z:

x	y	z	$(x \vee y)$	$\neg x$	$(\neg x \wedge z)$	$((x \vee y) \rightarrow (\neg x \wedge z))$
0	0	0	0	1	0	1
0	0	1	0	1	1	1
0	1	0	1	1	0	0
0	1	1	1	1	1	1
1	0	0	1	0	0	0
1	0	1	1	0	0	0
1	1	0	1	0	0	0
1	1	1	1	0	0	0

Rows 1, 2 and 4 have a "1" in the column under the formula we are evaluating — those are the truth assignments that SATISFY our formula. In rows 3, 5, 6, 7 and 8 a "0" appears

under the formula we are evaluating — those are the truth assignments that FALSIFY our formula. A TAUTOLOGY is satisfied by every truth assignment, a CONTRADICTION can't get any satisfaction.

Normal forms

We can do things the other way around. Start with a column in a truth table, and then derive a formula. In general, there will be more than one way to do this. Our formula above is satisfied by the truth assignment in rows 1, 2, and 4 and no others. We could match our truth assignment to a conjunction of either propositional variables or negations of propositional variables (these two categories are called LITERALS), and this conjunction will match exactly one row:

x	y	z	$((x \vee y) \rightarrow (\neg x \wedge z))$	
0	0	0	1	$\neg x \wedge \neg y \wedge \neg z$
0	0	1	1	$\neg x \wedge \neg y \wedge z$
0	1	0	0	
0	1	1	1	$\neg x \wedge y \wedge z$
1	0	0	0	
1	0	1	0	
1	1	0	0	
1	1	1	0	

Taking the disjunction of the three conjunctions in the last column gives us an equivalent formula:

$$(\neg x \wedge \neg y \wedge \neg z) \vee (\neg x \wedge \neg y \wedge z) \vee (\neg x \wedge y \wedge z)$$

This propositional formula is satisfied by exactly the same truth assignments as our original formula. Conjunctions of one or more propositional variables or their negations (LITERALS) are called MINTERMS. Formulas that are disjunctions of one or more minterms are in Disjunctive Normal Form (DNF).

Here's a different approach to an equivalent formula for our truth table. For each of the rows 3, 5, 6, 7, and 8 (where there is a zero in the column under our formula in the truth table) write a disjunction of propositional variables or their negations, that is NOT satisfied by that truth assignment, but IS satisfied by all others:

x	y	z	$((x \vee y) \rightarrow (\neg x \wedge z))$	
0	0	0	1	
0	0	1	1	
0	1	0	0	$x \vee \neg y \vee z$
0	1	1	1	
1	0	0	0	$\neg x \vee y \vee z$
1	0	1	0	$\neg x \vee y \vee z$
1	1	0	0	$\neg x \vee \neg y \vee z$
1	1	1	0	$\neg x \vee \neg y \vee \neg z$

A formula that is falsified by any one of these formulas, but no others, is the conjunction:

$$(x \vee \neg y \vee z) \wedge (\neg x \vee y \vee z) \wedge (\neg x \vee y \vee \neg z) \wedge (\neg x \vee \neg y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$$

Disjunctions of one or more propositional variables, or their negations, are called MAXTERMS. Formulas that are conjunctions of 1 or more maxterms are in CONJUNCTIVE NORMAL FORM (CNF).

By the recipe just given, you can construct DNF or CNF formulas to match any truth table. Your choice of which to construct might be influenced by the number of zeros or ones in the column under the formula you are devising an equivalent formula for.

LOGICAL EQUIVALENCE

In the previous section we defined three different formulas with the same entries in a truth table — every truth assignment that satisfied one satisfied the other two. We should be able to save ourself some work by working out some standard equivalences.

Here are some "laws" that state special cases of equivalent formulas — any truth assignment that satisfies one satisfies the other. You can take my word for these laws, or write out the appropriate truth table to check them (none of the tables will have more than 8 rows). These formulas can be generalized, since anywhere you see formula P, Q, or R, you can substitute any propositional formula you want, and the equivalence will be maintained so long as you apply the equivalence law. Also, the LEQV relation (logically equivalent) is symmetrical, so you can read each law from left to right, or from right to left.

DOUBLE NEGATION LAW: $\neg\neg P$ LEQV P.

DE MORGAN'S LAWS: $\neg(P \wedge Q)$ LEQV $\neg P \vee \neg Q$. Symmetrically, $\neg(P \vee Q)$ LEQV $\neg P \wedge \neg Q$.

COMMUTATIVE LAWS: $P \wedge Q$ LEQV $Q \wedge P$. Symmetrically, $P \vee Q$ LEQV $Q \vee P$.

ASSOCIATIVE LAWS: $P \wedge (Q \wedge R)$ LEQV $(P \wedge Q) \wedge R$. Symmetrically $P \vee (Q \vee R)$ LEQV $(P \vee Q) \vee R$.

DISTRIBUTIVE LAWS: $P \wedge (Q \vee R)$ LEQV $(P \wedge Q) \vee (P \wedge R)$. Symmetrically (!) $P \vee (Q \wedge R)$ LEQV $(P \vee Q) \wedge (P \vee R)$.

IDENTITY LAWS: P LEQV $P \wedge (Q \vee \neg Q)$. Symmetrically, P LEQV $P \vee (Q \wedge \neg Q)$.

IDEMPOTENCY LAWS: P LEQV $P \vee P$. Symmetrically, P LEQV $P \wedge P$.

$\rightarrow$ LAW: $P \rightarrow Q$ LEQV $\neg P \vee Q$.

$\leftrightarrow$ LAW: $P \leftrightarrow Q$ LEQV $(P \wedge Q) \vee (\neg P \wedge \neg Q)$.

Here's an example (Exercise 3c from the Course Notes) of using these laws to prove two formulas are logically equivalent. Notice how De Morgan's law is used in both directions.

$$\begin{array}{rll} x \wedge \neg y \rightarrow \neg z & \text{LEQV} & \neg(x \wedge \neg y) \vee \neg z \quad [\rightarrow \text{law}] \\ & \text{LEQV} & (\neg x \vee y) \vee \neg z \quad [\text{De Morgan's law}] \\ & \text{LEQV} & \neg x \vee (y \vee \neg z) \quad [\text{Associativity}] \\ & \text{LEQV} & \neg x \vee (\neg z \vee y) \quad [\text{Commutativity}] \\ & \text{LEQV} & (\neg x \vee \neg z) \vee y \quad [\text{Associativity}] \\ & \text{LEQV} & \neg(x \wedge z) \vee y \quad [\text{De Morgan's law}] \\ & \text{LEQV} & x \wedge z \rightarrow y \quad [\rightarrow \text{law}] \end{array}$$

Proving that two propositional formulas are NOT equivalent has a completely different flavour. In this case it is necessary (and sufficient) to exhibit a truth assignment that satisfies one formula and falsifies the other.

Suppose you are asked to show that $(y \rightarrow x) \wedge (z \rightarrow x)$ is NOT logically equivalent to $(y \wedge z) \rightarrow x$. You may be able to see a truth assignment that satisfies one but not the other (and then you'd be done), or you could try manipulating the first formula a bit:

$$\begin{array}{rll} (y \rightarrow x) \wedge (z \rightarrow x) & \text{LEQV} & (\neg y \vee x) \wedge (\neg z \vee x) \quad [\rightarrow \text{law}] \\ & \text{LEQV} & (\neg y \wedge \neg z) \vee x \quad [\text{Distributive law}] \\ & \text{LEQV} & \neg(y \vee z) \vee y \quad [\text{De Morgan's law}] \\ & \text{LEQV} & (y \vee z) \rightarrow x \quad [\rightarrow \text{law}] \end{array}$$

Now you need to find a truth assignment that satisfies exactly one of $(y \wedge z) \rightarrow x$ or $(y \vee z) \rightarrow x$. Certainly $\tau(x, y, z) = (0, 1, 0)$ satisfies the first but not the second, so you've shown that they are not equivalent.

Boolean functions

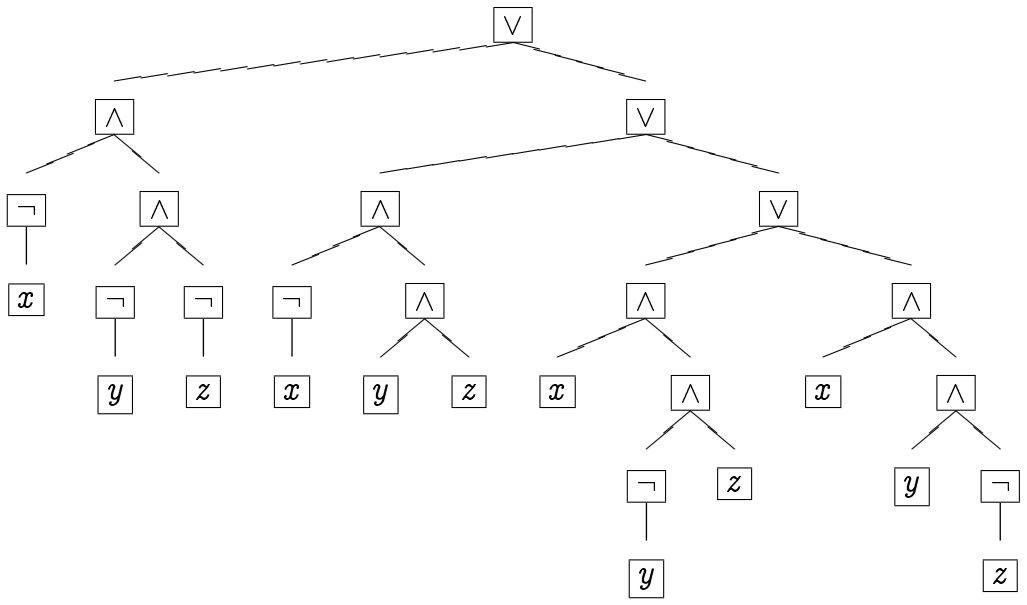
A function that takes arguments from $\{0,1\}^n$ and returns values in $\{0,1\}$ is called a BOOLEAN FUNCTION. If n is not too big, you can specify a boolean function with a truth table. Here's one for $n = 3$, together with a DNF formula:

x	y	z	$f(x,y,z)$	
0	0	0	1	$\neg x \wedge \neg y \wedge \neg z$
0	0	1	0	
0	1	0	0	
0	1	1	1	$\neg x \wedge y \wedge z$
1	0	0	0	
1	0	1	1	$x \wedge \neg y \wedge z$
1	1	0	1	$x \wedge y \wedge \neg z$
1	1	1	0	

Here's the DNF formula:

$$(\neg x \wedge \neg y \wedge \neg z) \vee (\neg x \wedge y \wedge z) \vee (x \wedge \neg y \wedge z) \vee (x \wedge y \wedge \neg z).$$

This DNF formula represents the boolean function specified in the truth table. The DNF formula can also be represented as a tree, where the leaves are variables x, y, or z and the interior nodes are symbols $\wedge$, $\vee$, or $\neg$.



Now the parse tree can be turned into a circuit diagram, by replacing $\neg$, $\vee$, and $\wedge$ by the symbols for an inverter, or-gate, and and-gate, and then drawing input lines to tie together all the x, y, and z inputs.

Since every boolean function can be represented by a DNF formula (we have a recipe for doing so), this means that every boolean function can be represented by a propositional formula containing only $\wedge$, $\vee$, and $\neg$ symbols, and implemented by a circuit containing only AND OR and NOT gates. We say that $\{\wedge, \vee, \neg\}$ is a COMPLETE SET of connectives (every boolean function can be represented using these connectives). In the Course Notes you can see that $\{\wedge, \neg\}$ is a complete set of connectives, but $\{\wedge, \vee\}$ is not.

Notes

[1] AKA formulae — it depends on whether you use the Latin or Germanic formula for forming plurals.