Adequacy of propositional connectives

Mike Spivey,
Trinity Term 2019

In the lectures, I claimed that every Boolean function of n arguments can be computed by a logic circuit containing AND and OR gates and inverters. We will take this as equivalent to the theorem that for every Boolean function, there is a formula containing the operations $\wedge$, $\vee$ and $\neg$ that expresses the function. Indeed, given any Boolean formula, we can construct an equivalent logic circuit, so any function expressible as a formula is also computable by a circuit. To prove the theorem itself, we will have to pin down precisely what we mean by the terms used in it: what does mean for a formula to *express* a Boolean function? So we must begin with some definitions.

Definition. By a *Boolean function*, we mean a mathematical function

$$f : \mathbf{B}^n \to \mathbf{B},$$

where $\mathbf{B} = \{0, 1\}$. We use the term *assignment* for members of the set $\mathbf{B}^n$ of n-tuples of Booleans. ∎

We can define a particular Boolean function by writing out its truth table: each line lists an assignment of Boolean values to the arguments of the function, and the value of the function on that assignment.

Definition. By the set $\mathcal{F}$ of *formulas*, we mean the smallest set that contains the constant $\mathbf{0}$ and the propositional letters x_i for $1 \le i \le n$, and is closed under the following operations:

- If $\phi \in \mathcal{F}$ then $\neg\phi \in \mathcal{F}$.
- If $\phi, \psi \in \mathcal{F}$ then $\phi \wedge \psi \in \mathcal{F}$.
- If $\phi, \psi \in \mathcal{F}$ then $\phi \vee \psi \in \mathcal{F}$.

We take the operations as forming a kind of tree, so that there is not the ambiguity that would arise if formulas were thought of as strings of symbols. We will use parentheses in writing formulas in order to make it clear which tree is meant. ∎

Definition. Given an assignment a and a formula ϕ, we can determine the *value* $[\![\phi]\!]_a$ of ϕ under a. This is defined recursively as follows:

- $[\![\mathbf{0}]\!]_a = 0$.
- $[\![x_i]\!]_a = a_i$.

- $[\![\neg\phi]\!]_a = 1 - [\![\phi]\!]_a$.
- $[\![\phi \wedge \psi]\!]_a = min([\![\phi]\!]_a, [\![\psi]\!]_a)$.
- $[\![\phi \vee \psi]\!]_a = max([\![\phi]\!]_a, [\![\psi]\!]_a)$. ∎

(In this definition, we have used functions *min*, *max* and $1 - \bullet$ to capture the effect of the connectives as operations on truth values $\{0, 1\}$ – but the gist of this definition is that the logical symbols express the operations we expect.)

The definitions made so far allow us to state precisely the result we want to prove.

Theorem. For each Boolean function f, there exists a formula ϕ that expresses f, in the sense that $[\![\phi]\!]_a = f(a)$ for every assignment a. ∎

We can observe at this point that the operations expressed by $\wedge$ and $\vee$ are associative and commutative and idempotent, so that (for example)

$$[\![(\phi \wedge \psi) \wedge \theta]\!]_a = [\![\phi \wedge (\psi \wedge \theta)]\!]_a$$

for any formulas ϕ, ψ, θ and all assignments a. We will exploit this fact in what follows by talking about the conjunction or disjunction of a *set* of formulas, without troubling to pick out which of the many equivalent possible results we mean. Alternatively, if the elements of the set of formulas have a natural order, we can simply join them together in that order with all the brackets as far as possible to the right, thus picking one result unambiguously.

Specifically, if ϕ_i is a formula for each element i of a non-empty finite set S, we can form the conjunction $\bigwedge_{i\in S} \phi_i$ and the disjunction $\bigvee_{i\in S} \phi_i$. For any assignment a, if $[\![\phi_i]\!]_a = 1$ for all $i \in S$, then $[\![\bigwedge_{i\in S} \phi_i]\!]_a = 1$ also, and if $[\![\phi_j]\!]_a = 0$ for any $j \in S$, it follows that $[\![\bigwedge_{i\in S} \phi_i]\!]_a = 0$. Similarly, to show $[\![\bigvee_{i\in S} \phi_i]\!]_a = 1$, it suffices to show that $[\![\phi_j]\!]_a = 1$ for some j, and we can show $[\![\bigvee_{i\in S} \phi_i]\!]_a = 0$ by showing that $[\![\phi_i]\!]_a = 0$ for all i.

Proof of theorem. Our goal is, given a Boolean function f, to construct a formula ϕ such that $[\![\phi]\!]_a = f(a)$ for every assignment a. We do this by considering the truth table of f, which is to say, the values $f(a)$ for each assignment a.

First, if x is a propositional letter and $t \in \mathbf{B}$, let $\delta(x, t)$ be the formula "x" if $t = 1$, and the formula "$\neg x$" if $t = 0$. It immediately follows that if a is any assignment, then $[\![\delta(x_i, a_i)]\!]_a = 1$. For if $a_i = 1$ then $\delta(x_i, a_i) = x_i$, so

$$[\![\delta(x_i, a_i)]\!]_a = [\![x_i]\!]_a = a_i = 1;$$

and if $a_i = 0$ then $\delta(x_i, a_i) = \neg x_i$, so

$$[\![\delta(x_i, a_i)]\!]_a = [\![\neg x_i]\!]_a = 1 - a_i = 1.$$

Also, if b is any assignment different from a, then for some j we have $[\![\delta(x_j, a_j)]\!]_b = 0$: just pick j such that $a_j \neq b_j$.

Next, for each a, let $\Delta(a)$ be the formula

$$\Delta(a) = \bigwedge_{1\leq i\leq n} \delta(x_i, a_i).$$

We find first that $[\![\Delta(a)]\!]_a = 1$, and second that if $b \neq a$ then $[\![\Delta(a)]\!]_b = 0$. The first claim follows from the fact that $[\![\delta(x_i, a_i)]\!]_a = 1$ for all i, and the second claim follows because we can pick j such that $[\![\delta(x_j, a_j)]\!]_b = 0$.

Finally, define ϕ by saying that if $f(a) = 0$ for all a, then $\phi = \mathbf{0}$; and otherwise, define

$$\phi = \bigvee_{a\,:\,f(a)=1} \Delta(a),$$

using a disjunction over the (finite) set of assignments a such that $f(a) = 1$. In the first case, the result follows trivially. In the second case, we claim that for any b, it holds that $[\![\phi]\!]_b = f(b)$. For if $f(b) = 1$, then $\Delta(b)$ appears as a disjunct on the right-hand side, and we have $[\![\Delta(b)]\!]_b = 1$ and so $[\![\phi]\!]_b = 1$. And if $f(b) = 0$ then all the disjuncts $\Delta(a)$ have $a \neq b$, and so $[\![\Delta(a)]\!]_b = 0$; it follows that $[\![\phi]\!]_b = 0$ also. In both cases, f and ϕ agree as desired. ∎

By way of contrast, it is claimed on a problem sheet that the operations $\oplus$ (exclusive OR) and $\neg$ do not form an adequate set. To show this formally, we need consider only the case $n = 2$, and let $\mathcal{G}$ be the set of formulas containing x_1 and x_2 and closed under the formation of formulas $\neg\phi$ and $\phi \oplus \psi$. Now the relevant definition of the value of a formula has

- $[\![\neg\phi]\!]_a = 1 - [\![\phi]\!]_a$.
- $[\![\phi \oplus \psi]\!]_a = ([\![\phi]\!]_a + [\![\psi]\!]_a) \bmod 2$.

Now let S be a set of eight Boolean functions defined by

$$S = \{0000, 0011, 0101, 0110, 1111, 1100, 1010, 1001\},$$

where $uvxy$ denotes the function f such that $f(0, 0) = u$, $f(0, 1) = v$, $f(1, 0) = x$, and $f(1, 1) = y$ – so that the truth table for f reads u, v, x, y from top to bottom. The set S contains only half of the 16 Boolean functions of two variables, and in particular it does not contain the AND function 0001. We claim, however, that for any formula $\phi \in \mathcal{G}$, if a function f is defined by $f(a) = [\![\phi]\!]_a$, then $f \in S$, and so there are some functions, including the AND function, that are not expressed by any formula $\phi \in \mathcal{G}$.

To prove this, we use structural induction on ϕ.

- If ϕ is a propositional letter, then it expresses one of the functions $x_1 = 0011$ or $x_2 = 0101$, both members of S.
- If ϕ expresses a function in S, then so does $\neg\phi$, because for each function $f \in S$, the function g defined by $g(a, b) = 1 - f(a, b)$ is also in S: in fact, this operation maps each function to the one that appears four places to its right or left in the list given above.
- If ϕ and ψ both express functions in S, then so does $\phi \oplus \psi$. Showing this is more tedious, but amounts to the fact that any two of the functions listed above, if combined by element-wise addition modulo 2, gives another of the functions listed.

It follows by induction that if $f(a) = [\![\phi]\!]_a$ for some formula $\phi \in \mathcal{G}$, then $f \in S$, and the set of connectives $\{\oplus, \neg\}$ is not adequate. ∎

For general n, we can characterise and count the functions expressible with these connectives by finding a *normal form*. Among the algebraic properties of the connectives are that $\oplus$ is associative and commutative and has $\mathbf{0}$ as an identity element. Each function is self-inverse: $\phi \oplus \phi \equiv \mathbf{0}$; and $\neg$ distributes over $\oplus$ in that $(\neg\phi) \oplus \psi \equiv \phi \oplus (\neg\psi) = \neg(\phi \oplus \psi)$. The consequence is that any expressible function can be put in a form such as $\neg(x_1 \oplus x_4 \oplus x_6)$, the

$\oplus$-sum of distinct variables, possibly with a negation sign at the front. We might capture the normal form symbolically as

$$\pm \sum_{1 \le i \le n} a_i x_i,$$

where $\pm$ denotes a possible negation, $\sum$ combines multiple terms with $\oplus$, and the coefficients a_i are each 0 or 1. There are 2^{n+1} such expressions, but 2^{2^n} Boolean functions in total, so necessarily some Boolean functions are not expressible.

Another property of the expressible functions is that they all take value 1 for an *even* number of assignments. This too can be proved by structural induction. The propositional letters are each true in precisely 2^{n-1} assignments. If ϕ is true in $2a$ assignments, then $\neg\phi$ is true in $2^n - 2a$ assignments, an even number. If ϕ and ψ are true in $2a$ and $2b$ assignments respectively, and there are c assignments where both are true, then $\phi \oplus \psi$ is true in those assignment where one or other but not both of ϕ and ψ are true; the number of such assignments is $(2a - c) + (2b - c) = 2a + 2b - 2c$, also an even number.

Boolean functions that are true in an even number of assignments can be put in one-to-one correspondence with those that are true in an odd number of assignments by negating the value for a single chosen assignment, such as the one where all letters are false. So exactly half of all Boolean functions are even, and there are 2^{2^n-1} such functions. In summary, there are 2^{n+1} functions expressible with $\{\oplus, \neg\}$, there are 2^{2^n-1} functions true in an even number of assignemts, and there are 2^{2^n} Boolean functions altogether, and $2^{n+1} < 2^{2^n-1} < 2^{2^n}$ whenever $n \ge 3$.