1 Truth functional completeness

Remark 6.1.1 The goal in this section is to consider the expressive capacity of propositional language, as well as alternative formulations of propositional logic. It is useful to take a more language independent approach.

Definition 6.1.2 (Boolean functions) Let $k \in \mathbb{N}$. A function $f : BOOL^k \to BOOL$ is called a *Boolean function* of arity k. A 0-ary Boolean function is a constant value, either **T** or **F**.

Remark 6.1.3 The truth table for a proposition α whose support is S of size k is really nothing more than the graph of a k-ary Boolean function. For example, the proposition $\neg(P \wedge Q)$ has the following truth table:

P	Q	$\neg(P \wedge Q)$	$P \uparrow Q$
T	**T**	**F**	**F**
T	**F**	**T**	**T**
F	**T**	**T**	**T**
F	**F**	**T**	**T**

This truth table describes the NAND function, the commonly introduced using the connective $\uparrow$. The table is a set of rules from computing a binary function $\mathcal{H} : BOOL^2 \to BOOL$. For example, $\mathcal{H}(\mathbf{T}, \mathbf{T}) = \mathbf{F}$.

From this point of view, each proposition represents a Boolean function through its truth table. The next definition standardizes this.

Definition 6.1.4 (propositions as functions) Let α be a proposition whose support is contained in the set $\{P_0, \ldots, P_{k-1}\}$. (The support need not be the same as this set, only contained in it.) The Boolean function $B^k_\alpha : BOOL^k \to BOOL$ is defined as follows: for each $b_0, \ldots, b_{k-1} \in BOOL$,

$$\mathcal{H}^k_\alpha(b_0, \ldots, b_{k-1}) = v_{\mathcal{A}}(\alpha),$$

where $\mathcal{A}$ is the assignment $\mathcal{A}(P_i) = b_i$ for each $i < k$ and **F** to every other propositional symbol. (By Lemma 4.1.4 of lecture 4 the values outside the support of α have no impact on the outcome of $v_{\mathcal{A}}(\alpha)$.)

To simplify the notation, in this section we will use the proposition α and the corresponding function $\mathcal{H}^k_\alpha$ interchangeably, when the support of α actually is $\{P_0, \ldots, P_{k-1}$.

Example 6.1.5 Let $\alpha = \neg(P_0 \wedge P_1)$. The corresponding Boolean function $\mathcal{H}^2_\alpha$ was determined by the truth table from the above Remark 6.1.3.

As another example, let $\alpha = \neg P_0$. The Boolean function $\mathcal{H}^2_\alpha$ is a binary Boolean function, even though α has only one variable. The graph of this function is

P_0	P_1	$\neg$	$\mathcal{H}^2_\alpha$
T	**T**	**F**	**F**
T	**F**	**F**	**F**
F	**T**	**T**	**T**
F	**F**	**T**	**T**

$\mathcal{H}^2_\alpha$ is a binary function, but it is *degenerate* in that it ignores its second argument. It "acts" just like the unary negation function on its first argument. There is a corresponding function $\mathcal{H}^2_{\neg\, P_1}$ which ignores its first argument and acts like the unary negation function on its second argument.

Remark 6.1.6 It is natural to ask whether every Boolean function can be realized as the truth function for some proposition α. What we want to determine is whether for every Boolean function $f : BOOL^k \to BOOL$, of any arity $k > 0$, there is a proposition α with $f = \mathcal{H}^k_\alpha$. If so, then we say that the Boolean connectives form a *functionally complete* set.

In Lecture 3 I took this question up in terms of truth tables: given a truth table $\mathscr{T}$ can we find a proposition α whose truth table is $\mathscr{T}$. The advantage of the functional approach is that it eliminates artificial features of a truth table, such as the arrangement of the rows.

The five propositional connectives are indeed functionally complete. The significance of this is that for any Boolean function f of arity k, we do not enrich the collection of assertions that we can make by adding a new symbol to the language, $\star$, and extending the definition of valuation by defining $\mathcal{H}_\star = f$. There is already a proposition α in the language with symbols $\{\neg, \wedge, \vee, \to, \leftrightarrow\}$ with $f = \mathcal{H}^k_\alpha$. We will do better than this, and show that each of the following sets of connectives is already functionally complete:

$$\{\wedge, \neg\}, \quad \{\vee, \neg\}, \quad \{\to, \neg\} \quad \{\to, \bot\}.$$

We now show that the set of connectives $\{\vee, \wedge, \neg\}$ is functionally complete.

Theorem 6.1.7 For every k-ary truth function f (where $k > 0$), there is a proposition α only using the connectives in the set $\{\vee, \wedge, \neg\}$ such that $f = \mathcal{H}^k_\alpha$.

Proof. The proof is by induction on the arity k. For $k = 1$, there are four truth functions whose truth tables are

P	f_1	f_2	f_3	f_4
T	**T**	**F**	**T**	**F**
F	**T**	**F**	**F**	**T**

The following propositions work (where $A = A_0$)

$$f_1 = \mathcal{H}^1_{P \vee \neg P} \qquad f_2 = \mathcal{H}^1_{P \wedge \neg P} \qquad f_3 = \mathcal{H}^1_P \qquad f_4 = \mathcal{H}^1_{\neg P}.$$

Let f have arity $k+1$ and assume that for every k-ary function g there is a proposition α with $g = \mathcal{H}^k_\alpha$ (inductive hypothesis). Let

$$\begin{aligned} f_1(x_1, \ldots, x_k) &= f(x_1, \ldots, x_k, \mathbf{T}) \qquad \text{and} \\ f_2(x_1, \ldots, x_k) &= f(x_1, \ldots, x_k, \mathbf{F}). \end{aligned}$$

Both these functions are k-ary, so by the inductive hypothesis there are propositions β and γ such that

$$f_1 = \mathcal{H}^k_\beta \qquad \text{and} \qquad f_2 = \mathcal{H}^k_\gamma.$$

Note that by our convention that the propositional symbols in β and γ are included in $\{P_0, \ldots, P_{k-1}\}$, the symbol P_k does not occur in β or γ. Let α by the proposition

$$(P_k \wedge \beta) \vee (\neg P_k \wedge \gamma).$$

It is easy to verify that $f = \mathcal{H}^k_\alpha$. □

Corollary 6.1.8 Each of the following sets of connectives is functionally complete:

$$\{\wedge, \neg\}, \quad \{\vee, \neg\}, \quad \{\to, \neg\}, \{\to, \bot\}$$

Proof. De Morgan's Law from Lecture 5 shows that $\vee$ can be equivalently defined by $\{\wedge, \neg\}$:

$$P \vee Q \simeq \neg(\neg P \wedge \neg Q),$$

so by the substitution of equivalents Lemma 5.1.7 we can replace any disjunction

$$\alpha \vee \beta \qquad \text{by} \qquad \neg(\neg\alpha \wedge \neg\beta),$$

which results in a tautologically equivalent proposition. This allows us to eliminate $\vee$ from the previous theorem.

The other from of De Morgan's Law

$$P \wedge Q \simeq \neg(\neg P \vee \neg Q),$$

allows us to eliminate conjunctions.

We can eliminate disjunction in favor of the conditional and negation by the tautological equivalence:

$$P \to Q \simeq \neg P \to Q.$$

Since $\{\vee, \neg\}$ are complete, so is $\{\to, \neg\}$.

We can eliminate negation as well using $\bot$:

$$\neg P \simeq P \to \bot.$$

which shows that $\{\to, \bot\}$ is complete. □

Example 6.1.9 The proof of Theorem 6.1.7 provides an algorithm for converting arbitrary propositions to tautologically equivalent ones using only the connectives $\{\wedge, \vee, \neg\}$. For example, to convert the proposition $(P \to Q) \to ((Q \to \neg R) \to P)$ we start with its truth table:

P	Q	R	$(P \to Q) \to ((Q \to \neg R) \to P)$
T	**T**	**T**	**T**
T	**T**	**F**	**T**
T	**F**	**T**	**T**
T	**F**	**F**	**T**
F	**T**	**T**	**T**
F	**T**	**F**	**F**
F	**F**	**T**	**F**
F	**F**	**F**	**F**

We break this table into the case where $P = \mathbf{T}$ and $P = \mathbf{F}$:

Q	R	$(P \to Q) \to ((Q \to \neg R) \to P)$
T	**T**	**T**
T	**F**	**T**
F	**T**	**T**
F	**F**	**T**

Q	R	$(P \to Q) \to ((Q \to \neg R) \to P)$
T	**T**	**T**
T	**F**	**F**
F	**T**	**F**
F	**F**	**F**

For the first table we need to consider the cases where $Q = \mathbf{T}$ and $Q = \mathbf{F}$:

$$(P \wedge ((Q \wedge (R \vee \neg R)) \vee (\neg Q \wedge (R \vee \neg R))))$$

When P is true, the proposition is true, regardless of whether Q is true or false. Regardless of whether Q is true or false, the proposition is true. For the second table, we again consider the cases where $Q = \mathbf{T}$ and $Q = \mathbf{F}$:

$$(\neg P \wedge ((Q \wedge R) \vee (\neg Q \wedge (R \wedge \neg R))))$$

When P is false and Q is true, the proposition has the same truth value as R; when P is false and Q is false, the proposition is always false. $(P \to Q) \to ((Q \to \neg R) \to P)$ has the same truth value as the disjunction of the two propositions above:

$$(P \wedge ((Q \wedge (R \vee \neg R)) \vee (\neg Q \wedge (R \vee \neg R)))) \vee (\neg P \wedge ((Q \wedge R) \vee (\neg Q \wedge (R \wedge \neg R))))$$

We will look at several other ways of constructing equivalent propositions using only $\{\wedge, \vee, \neg\}$ in the next lectures.

It is always harder to show that a set of connectives is not functionally complete. Here is one example:

Lemma 6.1.10 The set of connectives $\{\top, \vee, \wedge, \to, \leftrightarrow\}$ is not functionally complete.

Proof. We will show that negation, the unary function $\mathcal{H}_\neg$ cannot be represented by a proposition built from the connectives $\{\vee, \wedge, \to, \leftrightarrow\}$ and the propositional symbol P_0. The key property we will exploit is that $\mathcal{H}_\neg(\mathbf{T}) = \mathbf{F}$, but for any proposition α built from from the above four connectives together with P_0, the unary Boolean function $\mathcal{H}^1_\alpha(\mathbf{T}) = \mathbf{T}$. This is clear by the inspecting the first line of the truth tables for $\mathcal{H}_\vee, \mathcal{H}_\wedge, \mathcal{H}_\to, \mathcal{H}_\leftrightarrow$.

More formally, the proof is by induction on α. The only possible atoms are $\top$ and P_0, however $\mathcal{H}^1_\top(\mathbf{T}) = \mathbf{T}$ and $\mathcal{H}^1_{P_0}(\mathbf{T}) = \mathbf{T}$. For the induction step, suppose $\mathcal{H}^1_\alpha(\mathbf{T}) = \mathbf{T}$ and $\mathcal{H}^1_\beta(\mathbf{T} = \mathbf{T}$. Then for $\diamond \in \{\wedge, \vee, \to, \leftrightarrow\}$,

$$\mathcal{H}^1_{\alpha\diamond\beta}(\mathbf{T}, \mathbf{T}) = \mathcal{H}_\diamond(\mathcal{H}^1_\alpha(\mathbf{T}), \mathcal{H}^1_\beta(\mathbf{T})) = \mathcal{H}_\diamond(\mathbf{T}, \mathbf{T}) = \mathbf{T}.$$

The third equality uses the inductive hypothesis and the final equality is by inspection of the truth table of these connectives. □

Remark 6.1.11 It is now apparent that no single connective in the set $\{\wedge, \vee, \to, \leftrightarrow, \neg\}$ is functionally complete, and any functionally complete subset must contain either $\neg$ or $\bot$. We leave it as an exercise that $\{\neg, \leftrightarrow\}$ and $\{\bot, \leftrightarrow\}$ are not functionally complete.

Convention 6.1.12 From now on we will use the slightly reduced connective set $\{\bot\, \top, \wedge, \vee, \to, \neg\}$. Then,

$$(P \leftrightarrow Q) \qquad \text{abbreviates} \qquad ((P \to Q) \wedge (Q \to P)).$$

This will allow us to provide a unifying notation for the truth conditions of the remaining connectives in the next lecture.

2 Other Boolean connectives

The material in this section was presented earlier in Lecture 3, but it more naturally fits in this lecture. We consider other possible sets of connectives.

Definition 6.2.1 (Boolean connectives) A k-ary connective is a function $\sigma : \mathbf{PROP} \to \mathbf{PROP}$ which assigns a proposition $\sigma(\alpha_1, \ldots, \alpha_k)$ for every k-tuple of propositions $\alpha_1, \ldots, \alpha_k$. We will also insist that the the support of $\sigma(\alpha_1, \ldots, \alpha_k)$ is just the union of the support of $\alpha_1, \ldots, \alpha_k$.

A *k-ary Boolean connective* is a k-ary connective in which the truth value of $\sigma(\alpha_1, \ldots, \alpha_k)$ depends solely on the truth values of $\alpha_1, \ldots, \alpha_k$. That is, for any valuation v and propositions $\alpha_1, \ldots, \alpha_k$ and $\beta_1, \ldots, \beta_k$, if

$$v(\alpha_1) = v(\beta_1) \ldots v(\beta_k) = v(\alpha_k),$$

then $v\big(\sigma(\alpha_1, \ldots, \alpha_k)\big) = v\big(\beta_1, \ldots, \beta_k\big)$ If σ is a k-ary Boolean connective, then its meaning is given by a k-entry truth table.

Example 6.2.2 We introduced the Boolean connective $\uparrow$ (NAND) above as

$$\alpha \uparrow \beta = \neg(\alpha \wedge \beta),$$

together with its corresponding truth table. Another important Boolean connective is $\downarrow$ (NOR),

$$\alpha \downarrow \beta = \neg(\alpha \vee \beta).$$

Remark 6.2.3 (Unary Boolean connectives) There are four possible unary Boolean connectives. Their truth tables are:

P_1	①P_1	②P_1	③P_1	④P_1
T	**T**	**F**	**T**	**F**
F	**T**	**F**	**F**	**T**

The first three connectives are trivial, only the fourth is interesting and this is the table for negation $\neg$. We can easily construct propositions using only P and our five connectives which have the same truth table as each of these unary connectives:

P_1	$\top$	$\bot$	P_1	$\neg P_1$
T	**T**	**F**	**T**	**F**
F	**T**	**F**	**F**	**T**

Remark 6.2.4 (binary Boolean connectives) There are 16 binary Boolean connectives, we have only included four. Two of these are constantly true and constantly false. Two ignore the second input, so that they behave like unary connectives. Two ignore the second input. This leaves ten noteworthy binary connectives. Besides our four, the remaining six are

P_1	P_2	$P_1 \oplus P_2$	$P_1 \uparrow P_2$	$P_1 \downarrow P_2$	$P_1 < P_2$	$P_1 > P_2$	$P_1 \leftarrow P_2$
T	**T**	**F**	**F**	**F**	**F**	**F**	**T**
T	**F**	**T**	**T**	**F**	**F**	**T**	**T**
F	**T**	**T**	**T**	**F**	**T**	**F**	**F**
F	**F**	**F**	**T**	**T**	**F**	**F**	**T**

The connective $\oplus$ has the meaning of *exclusive* disjunction, or parity in the computer science literature. If you think of **T** and **F** as 1 and 0 (bit values in computer science), then $\oplus$ corresponds to bitwise addition modulo 2. Similarly, $<$ and $>$ correspond to the natural ordering $0 < 1$. You can also read $\rightarrow$ as $\leq$ and $\leftarrow$ as $\geq$. We can capture $\oplus$ by:

P_1	P_2	$P_1 \oplus P_2$	$(P_1 \vee P_2) \wedge (\neg(P_1 \wedge P_2))$
T	**T**	**F**	**F**
T	**F**	**T**	**T**
F	**T**	**T**	**T**
F	**F**	**F**	**F**

Remark 6.2.5 (Ternary Boolean connectives) There are $2^8 = 256$ ternary Boolean connectives. Of these 2 are essentially constants, $6 = 2 \cdot \binom{3}{1}$ are essentially unary and $30 = 10 \cdot \binom{3}{2}$ are essentially binary. This leave 218 unaccounted. For example, the *majority* connective, $\sharp\alpha\beta\gamma$ assigns the truth value of the majority. This connective can be expressed using our connectives as

$$(\alpha \wedge \beta) \vee (\alpha \wedge \gamma) \vee (\beta \wedge \gamma).$$

Another ternary connective is $\oplus^3$ which is ternary addition modulo two, when we view **T**and **F**as 1 and 0. Do you see how to represent $\oplus^3\alpha\beta\gamma$ using our connectives? One possible approach using the fact that

$$\oplus^3\alpha\beta\gamma \text{ has the same truth table as } (\alpha \oplus \beta) \oplus \gamma$$

and that we have already shown $\oplus$ can be represented.

Definition 6.2.6 If S is a set of connectives, we write $\overline{S}$ for the set smallest set of propositions meeting the following conditions:

1. All propositional symbols are in $\overline{S}$. $\bot$ and $\top$ are in $\overline{S}$ only if they are in S.
2. If $\sigma \in S$ is k-ary and $\alpha_1, \ldots, \alpha_k \in \overline{S}$, then $\sigma(\alpha_1, \ldots, \alpha_k) \in \overline{S}$.

A set S of Boolean connectives is *truth-functionally complete* if for each k-ary Boolean function $f : BOOL^k \to BOOL$, there is a proposition $\alpha \in \overline{S}$ with $\mathcal{H}^k_\alpha = f$.

Lemma 6.2.7 $\{\uparrow\}$ and $\{\downarrow\}$ are truth functionally complete. They are the only binary connectives which are truth functionally complete.

Proof. You can verify that $\{\uparrow\}$ is functionally complete by defining $\wedge$ and $\neg$ as follows:

$$P \uparrow P \simeq \neg P \qquad (P \uparrow Q) \uparrow (P \uparrow Q) \simeq P \wedge Q$$

(This was Exercise 3 from Homework 1.) It is left as an exercise to show that $\{\downarrow\}$ is truth functionally complete. These are the only two binary connectives that are truth functionally complete. First, any truth functionally complete binary connective must be a truth function $\mathcal{H}$ for which $\mathcal{H}(\mathbf{T}, \mathbf{T}) = \mathbf{F}$ and $\mathcal{H}(\mathbf{F}, \mathbf{F}) = \mathbf{T}$. The reason is exactly as argued in Lemma 6.1.10: If $\mathcal{H}(\mathbf{T}, \mathbf{T}) = \mathbf{T}$ or $\mathcal{H}(\mathbf{F}, \mathbf{F}) = \mathbf{F}$. then it is impossible to represent negation. This rules out all but $\uparrow$ and $\downarrow$ among the nontrivial connectives given in Remark 6.2.4 (as well as our four binary connectives). The only remaining connectives are the ones that are essentially unary (they ignore at least one argument). The functions can never distinguish between the possibilities $(\mathbf{T}, \mathbf{F})$ and $(\mathbf{F}, \mathbf{T})$, so cannot represent any binary connective like $\to$ which does assign different values. □

Remark 6.2.8 (Choice of connectives) The five connectives in our set were chosen because they occur naturally in mathematics and they are complete. If we our goal was to capture the uses in logic programming we might add $>$, or $\leftarrow$ as it is more commonly used in Prolog. If we were interested in logic circuits we might want NOR and NAND.

It is noteworthy that both $\{\uparrow\}$ and $\{\downarrow\}$ are functionally complete set. These are the only two binary Boolean functions which are complete. The connective $\downarrow$, or $|$ in the logic community, is known as the Sheffer stroke Henry Sheffer who discovered in 1913 that it alone is complete. Actually, Charles Sanders Pierce recognized both $\uparrow$ and $\downarrow$ are complete in 1880, but his work was not published for another fifty years.