0.0.1 Lecture #3 —continued

Recall

0.0.1 Definition. (Boolean formulas or wff) A *string* (or *expression*) A over the alphabet of Boolean symbols $\mathcal{V}$ is called a *Boolean formula* or a *Boolean well-formed formula* (in short *wff*) ***iff*** it occurs in *some* formula construction.

The set of all wff we denote by the all-capitals **WFF**.

The wff that are either propositional variables $p, q, p'', r_{123}, \ldots$ or $\bot$ or $\top$, in short, *variables* or *constants*, we call *Atomic* wff. □

Notation. We often want to say things such as "... bla-bla ... *for all variables* p ...".

Well this is not exactly right! **There is only ONE variable p!**

We get around this difficulty by having *informal names* (in the *metatheory* as we say) for Boolean variables: $\mathbf{p}, \mathbf{q}, \mathbf{r}'$, etc.

Any such bold face informal variable can stand for *any* actual variable of our alphabet $\mathcal{V}$ whatsoever.

So "all **p**" means "any of the actual variables $p, q, r_{1110001}, \ldots$ that **p** may stand for" while "all p" is *meaningless*!

We can give a definition of formulas that is *independent* from formula constructions: OK, the above Definition 0.0.1 says that A is a wff iff it appears in a construction as

1. Atomic: $\bot, \top, \mathbf{p}$
2. A *negation* $(\neg B)$, where B appeared earlier in the construction
3. An expression $(B \wedge C)$ or $(B \vee C)$ or $(B \to C)$ or $(B \equiv C)$, where B *and* C appeared earlier in the construction and

We can stop referring to *constructions* by changing —by Definition 0.0.1— the blue and red statements 2 and 3 above to read instead "B is a wff" and "B *and* C are each a wff" respectively.

Thus we have

0.0.2 Definition. (The Inductive Definition of wff) An expression A over $\mathcal{V}$ is a wff just in case A is:

(1) Atomic $(\mathbf{p}, \bot, \top)$

or one of

(2) $(\neg B)$, $(B \wedge C)$, $(B \vee C)$, $(B \to C)$, $(B \equiv C)$, where B and C are wff. □

0.0.3 Remark. The formulas $(\neg A)$, $(A \wedge B)$, $(A \vee B)$, $(A \to B)$, $(A \equiv B)$ are read, from left to right, "not A", "A and B", "A or B", "if A then B" (but also " A implies B"), "A is equivalent to B".

The above wff have the same names with their "last glue" , namely, negation, conjunction, disjunction. implication and equivalence.

Pause. Why "LAST" glue?

□

0.0.4 Example. Using 0.0.1 let us verify that $((p \vee q) \vee r)$ is a wff.
Well, here is a formula construction written with annotations:

(1)	p	⟨atomic⟩
(2)	q	⟨atomic⟩
(3)	r	⟨atomic⟩
(4)	$(p \vee q)$	⟨1 + 2 + ∨-glue⟩
(5)	$((p \vee q) \vee r)$	⟨4 + 3 + ∨-glue⟩

Do we have to write down *all* the atomic wff at the very beginning? *Not really*, but it is important to write them BEFORE they are needed! □

Intuitively, immediate predecessors of a wff are the formulas we used to apply the last glue.

0.0.5 Definition. (Immediate predecessors (i.p.)) No atomic formula has immediate predecessors.

Any of the following wff $(A \wedge B), (A \vee B), (A \to B), (A \equiv B)$ has as i.p. A *and* B.

A is an i.p. of $(\neg A)$. □

0.0.6 Example.

- The i.p. of $((p \vee q) \vee r)$ are $(p \vee q)$ and r
- The i.p. of $(p \vee q)$ are p and q
- The only i.p. of $(\neg\top)$ is $\top$

□

0.0.7 Remark. (Priorities of glue (connectives)) The *priorities* of glue form left to right go from strongest to weakest according to the sequence

$$\neg, \wedge, \vee, \rightarrow, \equiv \tag{1}$$

□

Why do we care? What does "priority" do?

Well, *suppose* we do not want to always write wff down with all the brackets that definition 0.0.1 and 0.0.2 require.

Why wouldn't we? For better readability!

Thus we **agree** to judiciously omit brackets in a manner that we can reinsert them correctly if we are required to!

That is, we **agree** on how to write formulas **sloppily** and get away with it!
Is there any other way to do this?

Yes, BUT: As with any agreement between any two people, there can be ONLY one agreement.

So please *do* follow (1) above and the clarifications that follow below. Anything else will be wrong.

The "algorithm" is that whenever two pieces of glue compete for a variable as in

$$\ldots \vee p \wedge \ldots$$

then $\wedge$ wins (higher priority) and "gets" the p. This means brackets were intended —and hence are reinserted— this way:

$$\ldots \vee (p \wedge \ldots$$

What if we have the situation

$$\ldots \vee p \vee \ldots \tag{2}$$

i.e., same glue left and right of p?

We have the agreement that all glue is right-associative, that is, in a chain like (2) *the glue on the right wins*! We insert brackets this way:

$$\ldots \vee (p \vee \ldots$$

In particular

$$\neg\neg\neg p$$

means

$$\big(\neg\big(\neg(\neg p)\big)\big)$$

0.0.8 Definition. (Complexity of a wff) The *complexity* of a wff is *the number of occurrences* of connectives (glue) in it. Counting occurrences means that *multiplicity matters* and counts! □

0.0.9 Example. Clearly we can compute complexity correctly whether we wrote a formula with all its brackets or not.

For example, the complexity of $p \to \bot \to r$ is 2 whether we wrote it with no brackets or wrote it as Definitions 0.0.1 and 0.0.2 want: $(p \to (\bot \to r))$.

Directly from the definition above, every atomic formula has complexity zero. □

All the theorems (and their corollaries) in this section are **ABOUT** formulas of Boolean logic, and their *FORM*.

They are not theorems OF Boolean logic. This concept we have not defined yet!!

Theorems that are ABOUT logic we call METAtheorems.

0.0.10 Theorem. *Every formula A has equal numbers of left and right brackets.*

Proof. Induction on the complexity, let's call it n, of A.

1. *Basis.* $n = 0$. Then A has no glue, so it is atomic. But an atomic formula has no left or right brackets!

   Since 0=0 we are good!

2. *Induction Hypothesis*, in short "I.H." Fix an n and assume the statement for all A of complexity $\leq n$.

3. *Induction Step*, in short "I.S.", is for any A of complexity $n + 1$. As $n + 1 > 0$, A is NOT atomic THEREFORE it has one of *TWO* forms:

   (a) A is $(\neg B)$. By I.H. —applicable since A has complexity $n + 1$ hence the complexity of B is $\leq n$— B has equal number of left and right brackets. Forming A we added one of each type of bracket. So, total left=total right for A.

(b) A is $(B \circ C)$, where we wrote "$\circ$" as a *metasymbol* that stands for any *binary glue* among $\wedge, \vee, \to, \equiv$.

How you glue TWO formulas, with what glue, will *not change* the number of brackets you will insert!

Now by the I.H. (since *each* of B and C have complexity $\leq n$) we have that the *total* number of *left* brackets from B and C equal their total number of *right* brackets.

A has ONE extra left, and ONE extra right brackets that the totals above. So A has the property! *DONE*! $\square$

Lecture #4 Sept. 18.

IMPORTANT! You will note that the induction for the formula A above essentially went like this:

- Prove the property for the atomic formulas $\mathbf{p}, \bot, \top$

  We assumed the I.H. that all the i.p. of A have the property.

  Then *we proved* (I.S.)

- If A is $(\neg B)$, then A has the property *since the i.p. B does* (**WHY B does?**).

- If A is $(B \circ C)$, then A has the property *since the two i.p. B and C do.*

The technique above is called **Induction on** (the *shape of*) **formulas** and **does not need the concept of complexity.**

This is how we will do it *in our inductions going forward*.

0.0.11 Corollary. *Every* nonempty proper prefix *of a wff A has an excess of left (compared to right) brackets.*

Proof. We do induction on the formula A.

- *Basis.* A is atomic. Then we are done since A has NO nonempty proper prefix!

  People also say "**then there is nothing to prove**" or "**the statement is vacuously satisfied**".

What just happened here?! Well, I am claiming "the statement is true" and suppose that you are claiming "**the statement is false**".

It is for you to give me a *counterexample* to what I said in order to show that you are right: Namely,

You must produce a *nonempty proper prefix* of A that fails the property.

BUT there is no way! There is NO *nonempty proper prefix* of A!

So I win!

- *Assume* the I.H. that *all the i.p. of A have the property.*

- For the I.S. we examine *ALL possible forms* of nonempty proper prefixes. These are:

  1. Case where A is $(\neg B)$. A nonempty proper prefix of A has one of the four **forms** below:

     (a) (Then clearly we have an excess of "(" The I.H. was NOT needed.

     (b) (¬ Then clearly we have an excess of "(" The I.H. again was NOT needed.

     (c) $(\neg D$, where D is an nonempty proper prefix of B. D already has an excess of "(" by the I.H. that applies since B is an i.p. of A.
     So, adding to them the leading red "(" does no harm!

     (d) $(\neg B$ Now (0.0.10) B has equal number of lefts and rights. The leading (red)"(" contributes an excess. The I.H. again was NOT needed.

2. A is $(B \circ C)$. A nonempty proper prefix of A has one of the six **forms** below:

   (a) (Then clearly we have an excess of "(" The I.H. was NOT needed.

   (b) $(B'$, where B' is an nonempty proper prefix of B. B' already has an excess of "(" by the I.H. that applies since B is an i.p. of A. So, adding to them the leading "(" does no harm!

   (c) $(B$ B has balanced bracket numbers by 0.0.10, thus the leading "(" creates a majority of "(".

   (d) $(B\circ$ As $\circ$ adds no brackets we are done by the previous case.

   (e) $(B\circ C'$ Here B is a formula so it contributes 0 excess. C' is *a nonempty proper prefix of* C and the I.H. applies to the latter as it is an i.p. of A.

   So C' has an excess of "(" and the leading "(" of A helps too.

   (f) $(B \circ C$ Neither B nor C contribute an excess of "(" as both are formulas. The leading red "(" breaks the balance in favour of "(". □

This is easy:

0.0.12 Theorem. *Every formula A begins with an atomic wff, or with a "(".*

Proof. By 0.0.2, A is one of

- Atomic $\mathbf{p}, \bot, \top$
- $(\neg B)$
- $(B \circ C)$ where $\circ \in \{\wedge, \vee, \to, \equiv\}$

So, in the first case A begins with an atomic wff, and in the other two begins with an "(".

No Induction was used or needed! □

0.0.13 Theorem. (Unique Readability) *The i.p. of any formula A are unique.*

So we can "deconstruct" or "parse" a formula in a unique way: It is **exactly one of** *atomic*, a *negation*, a *disjunction*, a *conjunction*, an *implication*, an *equivalence*.

Proof.

- Clearly *no atomic formula can be read also as one of* a *negation*, a *disjunction*, a *conjunction*, an *implication*, an *equivalence* since it *has no glue*, but the all the others do.

- Can we read a formula A as two *distinct* negations? That is, *using here "=" as equality of strings*, *can we have*
$$A = (\neg B) = (\neg C)?$$
No, since $(\neg B) = (\neg C)$ implies that after we match the first two symbols (left to right) then we will continue matching all symbols —by position— until we match all of B with C and finally match the rightmost ")".

- Can we read a formula A as a *negation* **and** as a *disjunction*, or a *conjunction*, or an *implication*, or an *equivalence*? That is, can I have
$$A = (\neg B) = (C \circ D)?$$
No, since if we have $(\neg B) = (C \circ D)$, then from left to right the first position is OK (match) but the 2nd is NOT: *C cannot begin with "¬"* (see 0.0.12).

- Can we read a formula A as a $(B \circ C)$ and also as a $(D \diamond Q)$, *where $\diamond$ stands for any binary glue?*

  Let's **assume** *that* we can *and get a* **contradiction**.

  Well, note first that if $(B \circ C) = (D \diamond Q)$ then if we have $B = D$ then this forces $\circ = \diamond$ and hence also that $C) = Q)$ which trivially (remove the ending ")") leads to $C = Q$.

  BUT this is not the case that we are looking at.

  So, assume that $B \neq D$. There are **two cases**.

Case 1. B is a nonempty proper prefix of D. Then, by 0.0.11, B has an excess of left brackets. But being a wff it also has balanced numbers of left/right brackets. **Contradiction**!

Case 2. D is a nonempty proper prefix of B. Then, by 0.0.11, D has an excess of left brackets. But being a wff it also has balanced numbers of left/right brackets. **Contradiction**!

□

Why do we care about unique readability?

Well, there is an old programming language called "PL/1" (from "Programming Language 1").

The language defines "*statement*" to be *any instruction*.

It has two kinds of **if-statements**, namely

- **IF** Con **THEN** St

  and

- **IF** Con **THEN** St_1 **ELSE** St_2

  where "*Con*" stands for any *condition* and "*St*", "St_1" and "St_2" can be any *statements*.

So what does the following do?

$$\textbf{IF } \text{Con}_1 \textbf{ THEN IF } \text{Con}_2 \textbf{ THEN } \text{St}_1 \textbf{ ELSE } \text{St}_2 \tag{1}$$

We *DON'T KNOW!* The above is *ambiguous*! (No *unique* way to go backwards to figure out what it says)

To fix the context, say Con_1 evaluates as *false*.

Now, *one* meaning of (1) is

$$\textbf{IF } \text{Con}_1 \textbf{ THEN } \left\{ \textbf{IF } \text{Con}_2 \textbf{ THEN } \text{St}_1 \textbf{ ELSE } \text{St}_2 \right\} \tag{2}$$

which does *nothing* (*skips all*) and control goes to the next statement, whatever that is.

Another meaning of (1) is

$$\textbf{IF } \text{Con}_1 \textbf{ THEN } \left\{ \textbf{IF } \text{Con}_2 \textbf{ THEN } \text{St}_1 \right\} \textbf{ ELSE } \text{St}_2 \tag{3}$$

which causes the execution of St_2.

POSTSCRIPT The PL/1 language *was not redefined to remove the ambiguity*! Rather, the **Compiler** was programmed to "believe" that (2)) above was meant (*rule* "ELSE matches the closest IF")

Boolean Semantics

Boolean Logic is about the ***behaviour of glue****. That is, we use it to find out how glue leads to (computes) the value of a formula, if values are arbitrarily assigned to the atomic formulas.*

What values do we have in mind?

The so-called truth-values, *true* and *false*.

These values are OUTSIDE Boolean Logic.

Did *you* see them in the alphabet $\mathcal{V}$? **Nor did I**!!
They are in the metatheory of Boolean Logic, that is in the domain were we are speaking about the logic, rather than using it.

0.0.14 Definition. A *state* v (or s) is a function *that assigns the value* $\mathbf{f}$ (*false*) or $\mathbf{t}$ (*true*) to *every* Boolean variable, while the *constants* $\bot$ and $\top$, *necessarily, always* get the values $\mathbf{f}$ and $\mathbf{t}$ respectively.

None of these symbols —$v, s, \mathbf{t}, \mathbf{f}$— are in the Boolean logic alphabet $\mathcal{V}$. They are *metasymbols* in the *meta*theory.

The $\mathbf{f}$ and $\mathbf{t}$ we call *truth values*.

On paper or on the chalk board one usually *underlines* rather than *bolds* —as bolding is cumbersome— so one denotes $\mathbf{f}$ as $\underline{\mathrm{f}}$ and $\mathbf{t}$ as $\underline{\mathrm{t}}$ respectively.
The fact that v gives (assigns) the value $\mathbf{f}$ to the variable q'' is denoted by $v(q'') = \mathbf{f}$. □

Therefore a state v is (think of MATH 1019/1028 here!) an *infinite* input/output *table* like the one below

input	output
$\bot$	**f**
$\top$	**t**
p	**t**
q	**f**
$\vdots$	$\vdots$

where *no two rows can have the same input but different outputs*.

Why an *infinite* table?

Because our *Boolean logic language* has *infinitely many variables* and a state, by definition, assigns a value to *each of them*.

0.0.15 Definition. (Truth tables) In the *metatheory* of Boolean logic there are five *operations* we are interested in applied on the members of the set of *truth values* $\{\mathbf{t}, \mathbf{f}\}$.

Each operation takes its input(s) from the above set, and its outputs are also in this set.

We have one operation for each connective (glue) and in order to keep track of which is which we use the generic letter F (for "function") subscripted by the name of the corresponding glue.

These functions of the metatheory are called Boolean functions and are the following.

$$F_{\neg}(x), F_{\vee}(x,y), F_{\wedge}(x,y), F_{\rightarrow}(x,y), F_{\equiv}(x,y)$$

The behaviour of these functions —input/output behaviour, that is— is fully described by the following table that goes by the nickname "*truth table*".

x	y	$F_{\neg}(x)$	$F_{\vee}(x,y)$	$F_{\wedge}(x,y)$	$F_{\rightarrow}(x,y)$	$F_{\equiv}(x,y)$
f	**f**	**t**	**f**	**f**	**t**	**t**
f	**t**	**t**	**t**	**f**	**t**	**f**
t	**f**	**f**	**t**	**f**	**f**	**f**
t	**t**	**f**	**t**	**t**	**t**	**t**

□