Lecture #13, Continued; Oct. 30

Extending Boolean Logic

Boolean Logic can deal only with the Boolean glue: properties and behaviour.

Can certify tautologies, but it *misses* many other truths as we will see, *like $x = x$ where x stands for a mathematical object like a matrix, string, array, number, etc.*

One of the obvious reasons is that Boolean logic cannot even "see" or "speak" about mathematical objects.

If it cannot see or speak about them, then naturally cannot reason about them either!

E.g, we cannot say inside Boolean logic the sentence "every natural number greater than 1 has a prime factor".

Boolean Logic *does not know* what "every" means or what a "number" is, what "natural" means, what is "1", what "greater" means, what "prime" is, or what "factor" is.

In fact it is worse than not "knowing": It cannot even say any one of the concepts listed above.

Its alphabet and language is extremely limited.

We need a richer language!

0.0.1 Example. Look at these two math statements. The first says that *two sets are equal iff they have the same elements*. The second says that *any object is equal to itself*.

We read "$(\forall x)$" below as "*for all values of x*", usually said MORE SIMPLY as, "*for all x*".

$$(\forall y)(\forall z)\Big((\forall x)(x \in y \equiv x \in z) \to y = z\Big) \tag{1}$$

and

$$x = x \tag{2}$$

Boolean Logic is a *very high level* (= very non-detailed) abstraction of Mathematics.

Since Boolean Logic cannot see object variables x, y, z, cannot see $\forall$ or $=$, *nor can penetrate inside the so-called "scope" of $(\forall z)$* —that is, the big brackets above— it myopically understands each (1) and (2) as atomic statements p and q (not seeing inside the scope it sees no "glue").

Thus Boolean logic, if forced to opine about the above it will say none of the above is a theorem (by soundness).

Yet, (1) is a theorem of *Set Theory* and (2) is an *axiom in ALL mathematics*.

Says: "Every object is equal to itself." □

Enter First-Order Logic or Predicate Logic.

Predicate logic is *the language AND logic* of mathematics and mathematical sciences.

In it we CAN "speak" (1) and (2) above ***and*** *reason about them.*

0.1. The language of First-Order Logic

What symbols are *absolutely necessary* to *include* in the Alphabet, $\mathcal{V}_1$ —the subscript "1" for "1st-order"— of Predicate Logic?

Well, let us enumerate:

0.1.1 Definition. (The 1st-order alphabet; first part)

1. First of all, we are *EXTENDING*, *NOT* discarding, *Boolean Logic*. So we include in $\mathcal{V}_1$ *all of Boolean Logic's symbols* $\mathbf{p}, \bot, \top, (,), \neg, \wedge, \vee, \rightarrow, \equiv$, where $\mathbf{p}$ stands for any of the infinitely many Boolean variables.

2. Then we need *object variables* —that is variables that stand for *mathematical objects*— x, y, z, u, v, w with or without primes or subscripts. So, these are infinitely many.

   Metanotation that stands for any of them will be bold face, but using the same letters with or without primes or subscripts: $\mathbf{x}, \mathbf{x}''_{\mathbf{5}}, \mathbf{y}, \mathbf{w}'''_{\mathbf{123}}$, etc.

3. *Equality* between mathematical objects: $=$

4. *New glue*: $\forall$

   We call this glue *universal quantifier*. It is pronounced "for all".

Is that all? No. *But let's motivate with two examples.* □

0.1.2 Example. (Set theory) The language of set theory needs also a binary relation or *predicate* up in front: Denoted by "$\in$". BUT nothing else.

All else is "*manufactured*" in the theory, that is, introduced by definitions.

The manufactured symbols include *constants* like our familiar $\mathbb{N}$ (the *set of natural numbers*, albeit set theorists often prefer the symbol "ω"), our familiar *constant* "$\emptyset$" (the empty set).

Also include *functions* like $\cup, \cap$ and relations or *predicates* like $\subset, \subseteq$.

So set theory needs no constants or functions up in front to start "operating" (proving theorems, that is). □

0.1.3 Example. (Number theory) The language of Number theory —also called Peano arithmetic— needs —in order to get started:

- A *constant*, the *number zero*: 0
- A *predicate* ("less than"): $<$
- A unary *function*: "S". (This, informally/intuitively is the "successor function" which with input x produces output $x + 1$.)
- Two binary *functions*, "$+, \times$" with the obvious meaning.

All else is "*manufactured*" in the theory, that is, introduced by definitions.

The manufactured symbols include *constants* like our familiar $1, 2, 1000234000785$.

Also include *functions* like x^y, $\lfloor x/y \rfloor$ and more relations or *predicates* like $\leq$. □

We will do logic for the user, that is, we are aiming to teach the USE of logic.

But will do so *without having to do set theory or number theory or any specific mathematical theory* (geometry, algenra, etc.).

So equipped with our observations from the examples above, we note that various theories start up with *DIFFERENT* sets of constants, functions and predicates.

So we will complete the Definition 0.1.1 in a way that *APPLIES TO ANY AREA OF MATHEMATICAL APPLICATION*.

0.1.4 Definition. (The 1st-order alphabet; part 2) Our 1st-order alphabet also includes the following symbols

(1) Symbols for zero or more *constants*. *Generically*, we use a, b, c, d with or without primes or subscripts for constants.

(2) Symbols for zero or more *functions*. *Generically* we use f, g, h with or without primes or subscripts for functions.

Each such symbol will have the need for a certain number of arguments, this number called the function's "*arity*" (must be ≥ 1). For example, S has arity 1; it is unary. Each of $+, \times$ have arity two; they are binary.

(3) Symbols for zero or more *predicates*, *generically* denoted as ϕ, ψ, with or without primes or subscripts.

Each predicate symbol will have the need for a certain number of arguments called it "*arity*" (must be ≥ 1). For example, $<$ has arity 2. $\square$

The first-order *LANGUAGE* is a set of strings of two types —*terms* and *formulas*— over the *alphabet* 0.1.1 – 0.1.4.

By now we should feel comfortable with *first-order inductive definitions.*

In fact we gave inductive definitions of *first-order Boolean formulas* and used it quite a bit, but also more recently gave an inductive definition of Boolean *proofs*.

Thus we introduce first-order Terms, that denote objects, and first-order formulas, that denote statements, inductively in two separate definitions.

First terms:

0.1.5 Definition. (Terms)
A term is a string over the alphabet $\mathcal{V}_1$ that satisfies one of:

(1) It is just an *object variable* $\mathbf{x}$ (recall that $\mathbf{x}$ is metanotation and stands for *any* object variable).

BTW, we drop the qualifier "object" from "object variable" from now on, but *RETAIN* the qualifier "Boolean" in "Boolean variable".

(2) An *object constant* a (this stands for any constant —generically).

BTW, we ALSO drop the qualifier "object" from "object constant" from now on, but *RETAIN* the qualifier "Boolean" in "Boolean constant".

(3) General case. It is a string of the form $ft_1t_2 \ldots t_n$ where the function symbol f has *arity* n.

We will denote arbitrary terms generically by the metasymbols t, s *with or without primes or subscripts.* □

We will often abuse notation and write "$f(t_1, t_2, \ldots, t_n)$" for "$ft_1t_2 \ldots t_n$".

This is one (rare) case where *the human eye prefers extra brackets*! Be sure to note that the comma "," is not in our alphabet!

Examples from number theory.
$x, 0$ are terms. $x + 0$ is a term (abuse of the actual "$+x0$" notation).

$(x + y) \times z$ is a term (abuse of the actual $\times + xyz$).

With the concept of terms out of the way we now define 1st-order formulas:

First the Atomic Case:

0.1.6 Definition. (1st-order Atomic formulas) The following are the *atomic formulas of 1st-order logic*:

(i) Any *Boolean* atomic formula.

(ii) The *expression* (*string*) "$t = s$", for any choice of t and s (probably, the t and s name the same term).

(iii) For any predicate ϕ of arity n, and any n terms $t_1, t_2, \ldots, t_n$, the string "$\phi t_1 t_2 \ldots t_n$".

We denote *the set of all atomic formulas* here defined **AF**. □

0.1.7 Remark.
(1) As in the case of "complex" terms $f t_1 t_2 \ldots t_n$, we often abuse notation using "$\phi(t_1, t_2, \ldots, t_n)$" in place of the correctly written "$\phi t_1 t_2 \ldots t_n$".

(2) The symbol "$=$" is a binary predicate and is always written as it is here (never "ϕ, ψ").

(3) We absolutely NEVER confuse "$=$" with the "glue" "$\equiv$".

They are more different than apples and oranges! □

0.1.8 Definition. (1st-order formulas) A first-order formula A —or **wff** A— is one of

We let context fend for us as to *what formulas we have in mind when we say "wff"*.

From here on it is 1st-order ones!

If we want to talk about Boolean wff we *WILL USE* the qualifier "Boolean"!

(1) A *member of 1st-order* **AF** *set* —in particular it could be a *Boolean* atomic wff!

(2) $(\neg B)$ if B is a wff.

(3) $(B \circ C)$ if B *and* C are wff, and $\circ$ is one of $\wedge, \vee, \rightarrow, \equiv$.

(4) $((\forall \mathbf{x})B)$, where B is a wff and $\mathbf{x}$ any variable.

TWO things: (1) we already agreed that "variable" means *object variable* otherwise I'd say "Boolean variable". (2) Nowhere in the definition is required that $\mathbf{x}$ occurs in B as a substring.

We call "$\forall$" the *universal quantifier*.

The configuration $(\forall \mathbf{x})$ is pronounced "for all $\mathbf{x}$" —intuitively meaning "*for all values of* $\mathbf{x}$" rather than "for all variables $x, y'', z'''_{1234009}, \ldots$ that $\mathbf{x}$ may stand for".

We say that the part of A between the two red brackets is the scope of $(\forall \mathbf{x})$.

Thus the $\mathbf{x}$ in $(\forall \mathbf{x})$ and the entire B are in this scope. □

The "in particular" observation in (1) and the cases (2) and (3) make it clear that every Boolean wff is also a (1st-order) wff.

0.1.9 Example. $x = y$ and p are wff. *The second one is also a Boolean wff.*

$((\forall x)((\forall y)(\neg x = y)))$ is a wff. Note that $\neg$ in $(\neg x = y)$ *applies to* $x = y$ NOT to x!

Glue cannot apply to an object like x. Must apply to a *statement* (a wff)!

$((\forall y)((\neg x = y) \wedge p))$ and $(((\forall y)(\neg x = y)) \wedge p)$ are also formulas.

BTW, in the two last examples: *p is in the scope of $(\forall y)$ in the first*, but not so in the second. □

0.1.10 Definition. (Existential quantifier)

It is convenient —but NOT NECESSARY— to introduce the "*existential quantifier*", ∃.

This is only a *metatheoretical* abbreviation symbol that we introduce by this *Definition*, that is, by a "*naming*"

For any wff A, we define $((\exists \mathbf{x})A)$ to be *short for*

$$\Big(\neg\Big((\forall \mathbf{x})(\neg A)\Big)\Big) \tag{1}$$

We pronounce $((\exists \mathbf{x})A)$ "for some (value of) $\mathbf{x}$, A holds".

The intuition behind this $((\exists \mathbf{x})A)$ *naming* is captured by the diagram below

$$\Bigg(\overbrace{\neg}^{\textit{it is not the case that}} \Big(\underbrace{(\forall \mathbf{x})}_{\textit{all values of } x} \overbrace{(\neg A)}^{\textit{make } A \textit{ false}} \Big)\Bigg)$$

The *scope of* $(\exists \mathbf{x})$ in

$$\Big((\exists \mathbf{x})A\Big) \tag{2}$$

is the area between the two red brackets.

In particular, the leftmost $\mathbf{x}$ in (2) is in the scope. □

Priorities Revisited

We *augment* our priorities *table*, *from highest to lowest*:

$$\overbrace{\forall, \exists, \neg}^{\textit{equal priorities}}, \wedge, \vee, \rightarrow, \equiv$$

Associativities *remain right*! Thus, $\neg(\forall x)\neg A$ is *a short form* of (1) in 0.1.10.

Another example: $(u = v \rightarrow (((\forall x)x = a) \wedge p))$ simplifies into

$$u = v \rightarrow (\forall x)x = a \wedge p$$

More examples:

(2) Instead of $((\forall z)(\neg x = y))$ we write

$$(\forall z)\neg x = y$$

(3) Instead of $((\forall x)((\forall x)x = y))$ we write

$$(\forall x)(\forall x)x = y$$

BOUND vs FREE.

0.1.11 Definition. A variable $\mathbf{x}$ *occurs free* in a wff A iff *it is NOT in the scope of a* $(\forall \mathbf{x})$ *or* $(\exists \mathbf{x})$.

A bound variable $\mathbf{x}$ in $(\forall \mathbf{x})A$ other than the one in the displayed $(\forall \mathbf{x})$, belongs to the displayed leftmost "$(\forall \mathbf{x})$" iff $\mathbf{x}$ *occurs free in A*.

We apply this criterion to *subformulas* of A of the form $(\forall \mathbf{x})(\ldots)$ to determine where various bound $\mathbf{x}$ found inside A belong. □

0.1.12 Example. Consider

$$(\forall x)\overbrace{(x = y \to (\forall x)x = z)}^{A}$$

Here the red x in A belongs to the red $\forall x$. The black x belongs to the black $\forall x$. □