Ma 6c: Intro to mathematical logic

Garrett Ervin
gervin@caltech.edu
Office: 204 Kellogg
OH: TBA

TA: Tal Hershko
OH: TBA

Assessment: 8-10 HWs

Grades: not worse than standard cutoffs: 90-100 = A
80-89 = B etc.

Propositions

Def'n. A proposition is a statement that is either true (T) or false (F)
i.e. a statement w/ a truth value

Propositional logic (PL) is the mathematical study of propositions

- can have simple prop'ns: e.g.:
  - "My name is Garrett" (T)
  - "My name begins w/ G" (T)
  - "My name begins w/ J" (F)

call this prop'n p

call this q

call this r

... or connected prop'ns: e.g.:

"p and r"	(F)	(conjunction)
"p or r"	(T)	(disjunction)
"if p then q"	(T)	(conditional)
"if p then r"	(F)	
"if r then p"	(T but counterintuitive conditioning on false premise → vacuously true)	

Syntax of PL

- to formalize study of prop'ns, first need to formalize how to write prop'ns, i.e. our syntax.

Def'n: the language of PL consists of:

① propositional variables: $p_1, p_2, \ldots$ (may also use $q, r, \ldots$)

② connectives: $\neg$ ("not"), $\wedge$ ("and"), $\vee$ ("or"), $\Rightarrow$ ("implies"), $\Leftrightarrow$ ("iff")

③ parentheses (,)

Defn: a string or word is a sequence of symbols in this language

- e.g. $pq \wedge ((\ ,\ \Rightarrow qr$, and $(p_1 \Leftrightarrow p_2)$ are strings.
- we also consider the empty word a string
- write $|A|$ for length of string A
- e.g. $|pq \wedge (((| = 5$
  $|\phi| = 0$

- intuition: prop'nal vars $p_1, p_2, \ldots$ stand for simple prop'ns
- next: we formally define which strings will stand for connected prop'ns

Def'n The well-formed formulas (wffs) are defined recursively:

① every variable p_i is a wff
② if A is wff so is $\neg A$
③ if A, B are wffs so are
$(A \wedge B)$, $(A \vee B)$, $(A \Rightarrow B)$, $(A \Leftrightarrow B)$

- so: a string A is a wff iff it has a parsing sequence, i.e. a sequence
$A_1, A_2, \ldots, A_n$
s.t. $A_n = A$ and for $i \leq n$, either A_i is (i) a variable, (ii) $\neg A_j$ for some $j <$

or (iii) $(A_j * A_k)$ for some $j, k < i$ and $* \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$

Ex (a) $(p \Leftrightarrow q)$ is a wff w/ parsing sequence $p, q, (p \Leftrightarrow q)$

(b) $((p \Rightarrow q) \Leftrightarrow (\neg p \vee q))$ is a wff w/ sequence $p, q, (p \Rightarrow q), \neg p, (\neg p \vee q), ((p \Rightarrow q) \Leftrightarrow (\neg p \vee q))$

Note: $p, q, \neg p, (\neg p \vee q), (p \Rightarrow q), ((p \Rightarrow q) \Leftrightarrow (\neg p \vee q))$ is also a parsing sequence (so parsing sequences are not unique)

(c) $pq\wedge(($ and $\Rightarrow qr$ are not wffs
↳ how to prove? Hm!

Induction on formulas

- Sps $P(A)$ is a property of strings
- To prove P is true of all wffs A, sufficient (why?) to show:

"Induction on construction of wffs"

(i) (Base case): $P(p)$ is true for all variables p

(ii) (Inductive step): if $P(A)$ and $P(B)$ are true, then so are $P(\neg A)$ and $P((A * B))$ for $* \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$

Ex. Sps P(A) asserts "A has the same number of left and right parentheses"

Claim P(A) is true of all wffs A

Pf. (BC) any var. p has the same number of ('s and)'s (none)

(IS) if both A, B have some # of ('s and)'s then so does ¬A and (A * B) for * ∈ {∧, ∨, ⇒, ⇔} ✓

↳ follows from ex: pq∧((not a wff (two ('s, no)'s)

↳ but not enough to show pq∧() not a wff.

Def'n an initial segment of a string $s_1 s_2 \dots s_n$ is a string of the form $s_1 s_2 \dots s_m$ for some $0 \le m \le n$

↳ if $m < n$ say $s_1 \dots s_m$ is a strict initial seg. of $s_1 \dots s_n$

Prop'n no strict initial seg. of a wff A is itself a wff.

Pf. induction on construction of wffs (you try!)

↳ follows from prop'n: pq∧() is not a wff: p is a strict initial seg that is also a wff.

Unique readability of wffs

- want to guarantee that wffs can be decomposed into simpler wffs ("parsed") in a unique way
  - e.g. $(p \Rightarrow (q \wedge r))$ breaks into p and $(q \wedge r)$, which breaks into q, r
- if we allowed finlays like $p \Rightarrow q \wedge r$ to be wffs, would have ambiguity:
  - does this mean $p \Rightarrow (q \wedge r)$ or $(p \Rightarrow q) \wedge r$???

Thm (unique readability thm): every wff is in exactly one of the following forms: (i) p, for a unique variable p
(ii) $\neg A$, for a unique wff A
(iii) $(A * B)$, for unique wffs A, B and a unique $* \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$

Pf: - easy to show (by induction) every wff in one of these forms (you try!)
- also easy: every wff in at most one of these forms

- ... what about uniqueness?

(i) obvious: two diff. variables are diff.

(ii) if $\neg A = \neg B$ for wffs A, B, then clearly $A = B$

(iii) if $(A * B) = (C \circ D)$ for wffs A, B, C, D and $*, \circ \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$ then (by removing the (at the beginning) must have either:

- A is initial seg of C, or
- vice versa

- Sps the former (without loss of generality)
- if A were a strict init. seg. of C, wouldn't be wff by prev.
- hence $A = C$
- hence $* B) = \circ D)$
- follows $* = \circ$ and $B = D$ ✓

Recursion on formulas:

- we often want to define a function or operation recursively using construction of wffs
- by unique readability, such definitions yield well-defined and unique operations.

Ex ① Define a function C on the class of wffs by:

$C(p) = 0$ (for a variable p)

$C(\neg A) = C(A) + 1$

$C((A * B)) = C(A) + C(B) + 1$ for $* \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$

- e.g. $C((p \Rightarrow \neg(q \Rightarrow r))) = 3$
- in general: $C(A) = \#$ of connectives in A

② Given p a fixed propositional variable and P a wff we define $A[p/P]$ for a wff A by:

if $A = p$, $A[p/P] = P$

if $A = q$, $A[p/P] = q \neq A$

if $A = \neg B$ $A[p/P] = \neg B[p/P]$

if $A = (B * C)$, $A[p/P] = (B[p/P] * C[p/P])$

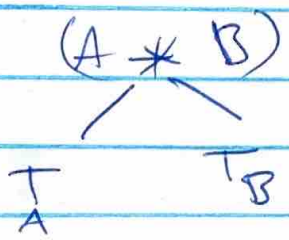
- i.e. $A[p/P]$ is obtained by replacing all instances of the var. p in A by P
- e.g. if $P = (s \vee t)$ and $A = (p \Rightarrow (q \Rightarrow r))$ then $A[p/P] = ((s \vee t) \Rightarrow (q \Rightarrow r))$

③ the parse tree T_A of a wff A is defined recursively by:

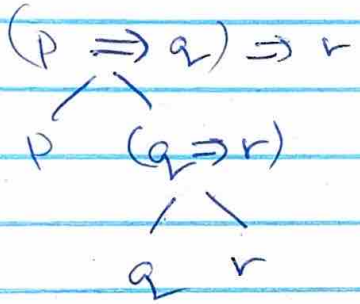
T_p is p

$T_{\neg A}$ is $\neg$ over T_A

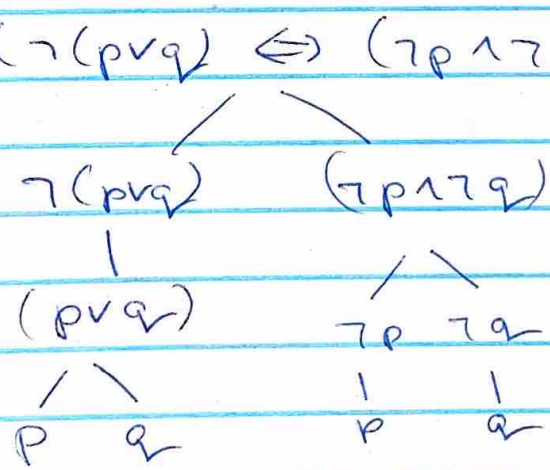
$T_{(A*B)}$ is



- e.g. if A is $(p \Rightarrow (q \Rightarrow r))$, T_A is:



- if A is $(\neg(p \vee q) \Leftrightarrow (\neg p \wedge \neg q))$, T_A is:



Def a formula P is a **subformula** of A if P appears in T_A

- e.g. $(p \vee q)$ is a subformula of $(\neg(p \vee q) \Leftrightarrow (\neg p \wedge \neg q))$

An algorithm recognizing wff's

- We give an algorithm that tests whether a given string (in lang of PL) is a wff.

Def'n a **substring** of a string S is a consecutive sequence of symbols in S
↳ e.g. $pq \Rightarrow$ is a substring of $)) pq \Rightarrow r \wedge (v$

Recognition algorithm

(i) - Given string S, choose a var. x not appearing in S
- define $S(x)$ = string obtained by replacing every var. in S w/ x

(ii) - For a string T whose only variable is x:
If T has a substring of the form $\neg x$, or $(x * x)$ for $* \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$
let T' = string obtained by replacing leftmost such substring by x
= (if no such substring: let $T' = T$)

(iii) The algorithm:

- Given string S, recursively define:

$$S_0 = S(x)$$
$$S_{n+1} = S_n'$$

- STOP at first n_0 s.t. $S_{n_0+1} = S_{n_0}$
(i.e. $S_{n_0}' = S_{n_0}$)

Note: algorithm must terminate since for each n either $S_{n+1} = S_n$ (hence STOP) or S_{n+1} is strictly shorter than S_n.

Claim: S is a wff iff $S_{n_0} = x$

↳ before proving claim: some examples

Ex's ① Say $S = ((p \Rightarrow (q \vee \neg p) \Rightarrow (r \vee (\neg s \Rightarrow t))))$

Algorithm: $((x \Rightarrow (x \vee \neg x) \Rightarrow (x \vee (\neg x \Rightarrow x)))) = S_0$
$((x \Rightarrow (x \vee x) \Rightarrow (x \vee (\neg x \Rightarrow x)))) = S_1$
$((x \Rightarrow x \Rightarrow (x \vee (\neg x \Rightarrow x)))) = S_2$
$((x \Rightarrow x \Rightarrow (x \vee (x \Rightarrow x)))) = S_3$
$((x \Rightarrow x \Rightarrow (x \vee x))) = S_4$
$((x \Rightarrow x \Rightarrow x)) = S_5 = S_6$ STOP

Alg says: S is not a wff.