PHIL 3something - Logic II

Scott Saunders - 10163541

Winter 2016

Misc. Notation

- The set of positive integers $\{x : x$ is a positive integer $\}$
- The set of positive integers less than3 $\{x : x$ is a positive integer and x is less than 3 $\}. = \{1, 2\}$.
- The empty set: $\emptyset$ or Δ
- Member of: $A \subseteq B$ iff $\forall X(x \in A \implies x \in B)$
- Union of A and B: $A \cup B$ iff $\{x : x \in A \vee x \in B\}$
- Intersection of A and B: $A \cap B$ iff $\{x : x \in A \wedge x \in B\}$
- Difference of A and B: $\{x : x \in A \wedge x \notin B\}$
- For any non-empty sets A, B: Cartesian product: A of B: $A \times B$: $\{< x, y >: x \in A \wedge y \in B\}$ (ALL OF THE POSSIBILITIES)
- TOTAL FUNCTION: Every element in the domain is valid
- PARTIAL FUNCTION: Not every element in the domain is valid.
- for any set of sets A:
  - $\Cup A = \{x : \exists y(y \in A \wedge x \in y)\}$
  - $\Cap A = \{x : \forall y(y \in A \rightarrow x \in y)\}$
- Relations: R is
  - reflexive : $\forall x Rxx$
  - symmetric : $\forall x \forall y(Rxy \implies Ryx)$
  - transitive : $\forall x \forall y \forall z((Rxy \wedge Ryz) \implies Rxz)$
  - Euclidean : $\forall x \forall y \forall z((Rxy \wedge Rxz) \implies Ryz)$
  - a equivalence relation : it's symmetric,reflexive,transitive.
  - a equivalence relation (alt) : it's symmetric, and euclidean.
  - a (partial) function : $\exists x$ and there is at most one y: Rxy : denoted f
  - a (partial) function[1] : $\exists x, \exists y | Rxy$: denoted f.
  - a (total) function: assigns a value to each number of A : denoted f
  - a (total) function[2]: $\forall x, \exists y | Rxy$: denoted f.
- Domain: The set of a functions arguments.
- Range: The set of its values. (Results)

- f is a function from a set A iff the domain of f is included in A
- f is a function to a set B iff its range is included in B.
- f^{-1} is the inverse of the function f from the set A to the set B iff:if for every member $b \in B$, there is exactly one member of $a \in A$ such that $f(a) = b$, then $f^{-1}(b) = a$, otherwise $f^{-1}(b)$ is undefined.
- f is onto B iff B is the range of f (Surjective)
  Alt: (Wikipedia) : $\forall y \in Y, \exists x \in X | y = f(x)$
- f is one-to-one iff $\forall x \forall y (f(x) = f(y) \implies x = y)$ (Injective)
- f is a bijection iff f is onto and one-to-one.
- f is a correspondence iff f is total, one-to-one and onto.
- Sets A and B are equinumerous iff there is a correspondence from A to B.

Equinumerous is transitive. Prove: if A is equinumerous with B and B is equinumerous wit C, then A is equinumerous with C. Proof: Suppose A is equinumerous to B, and B is equinumerous to C. Then: There is a total,one-to-one function f from A onto B, and a total one-to-one function g from B to C. Prove equinumerous via h=g(f), such that h(n)=g(f(n))

- h is total: Let a be a member of A. $h(a) = g(f(a))$. Since f is total there is a member of b of B such that $f(a) = b$). since g is total, there is a member of $c \in C$ such that $g(b) = c$. Hence, h is total.
- h is onto C. WLOG Let c be a member of C, as g is onto, $\exists b \in B$ such that $g(b) = c$. As f is onto, then $\exists a \in A$ such that $f(a) = b$. Hence, the composition of $h = f(g)$ is onto C.
- h is one-to-one: Suppose h is not one-to-one.
  Then there $\exists a_1, a_2 \in A$ such that $h(a_1) = h(a_2), a_1 \neq a_2$.
  Giving $g(f(a_1)) = g(f(a_2)), a_1/not = a_2$
  Since g is one-to-one $g(b_1) = g(b_2)$ iff $b_1 = b_2$.
  So the issue must lie in f. However f is one-to-one $f(a_1) = f(a_2)$ iff $f(a) = f(b)$. Which is a contradiction, giving us that h is one-to-one.

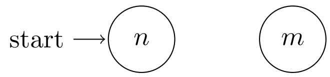
□

A^n : the nth Cartesian product of A with itself.
Suppose that the set of real Numbers $r, r < r < 1$, is enumerable. Then $L_r : r_1, r_2, r_3....$ written in a notation of $0.n_1n_2n_3.(nbeingnaturalnumbers)$
The set of functions form the set of positive integers to positive integers is not enumerable.

Proof. Suppose S is enumerable.
Then there is a list L_s of the members of S.
$L_s = \{s_1, s_2, s_3, \cdots\}$
Let $\forall n \in \mathbb{N}, n \in k \iff n \notin S_n$
k is a set of positive integers.
so There is a number $j$such that $k = s_j$. So $j \in S_j \iff j \not inS_j$
Hence S is not enumerable.

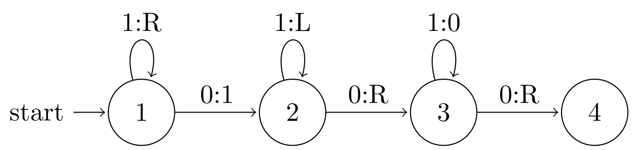
□

The set of total nomadic functions from the set of positive integers, F^1, is not enumerable.
It's a Proof by contradiction.
Turing machines are in the following form: $q_n, S_{1/0}, S_{1/0}/R/L, q_m$ where q_n is our current state, and you see $S_{1/0}$, perform function $S_{1/0}/R/$ and move to state q_m. If there is no operation specified on the current state for a scan, then it halts. (Also Called the Turing Alphabet)

Example with notation: 

ex: (These are the same)

$$Q_1S_1RQ_1, Q_1S_0S_1Q_2, Q_2S_1LQ_2, Q_2S_0RQ_3, Q_3S_1S_0Q_3, Q_3S_0RQ_4$$



Remark (Turing Machines)**.**

- Each Turing machine is a finite set of Turing instructions.
- Each instruction is a 4 letter word of the Turing Alphabet.
- The set of Turing machine is enumerable. (Proof: exercise)

Definition (Standard inital configuration)**.** A Turing machine is in a standard Initial configuration $\iff$

- for some positive integer k, there are k blocks of 1's on the tape.
- separated by a blank,
- and the rest of the tape is blank.
- the machine is scanning the left-most 1 on the tape.
- the machine is in it's lowest numbered state.

ex: $\cdots 0010110111000 \cdots$ is a SIC. (if it's in lowest state) ex: $\cdots 00010000 \cdots$ is a SIC.

Definition (Standard final configuration)**.** A Turing machine is in a standard final configuration $\iff$

- there is a single block of k 1's
- and the rest of the tape is blank.
- the machine is scanning the left-most 1 on the tape.

ex: $\cdots 00111111000 \cdots$ is a SIC. (if it's in lowest state) ex: $\cdots 00010000 \cdots$ is a SIC.

Definition (Computes a one-place function f^1)**.** A Turing M computes a one-place function f^1:
if M is started in a SIC with a single block of k 1's and

- if f^1 is defined for the argument k, then M eventually halts in a SFC
- or if f^1 is not defined for the argument k, then either M never halts or it halts in a non-standard final configuration.

Remark. Every Turing machine computes exactly one function of two arguments.

Remark. For any n, each Turing Machine computes exactly one function of n arguments.
The set of one-place Turing computable functions is enumerable

$\vdots$

The set of Turing computable functions is enumerable.

Definition (The halting problem)**.** The problem of finding an effective method to determine whether a Turing macine will eventually ahlt or not after it is started with some input.

The halting problem is unsolvable. Ex: $L_M : M_1, \dots$
$h(m,n) =$
1 if M_m eventually ahlts after starting with input n
2 if M_m never halts after starting with input n
The halting problem is solvable iff h is computable. Show: h is not Turing computable.
Let C be a copying machine.
Let F be $\frac{1}{2}$ flipper.
Suppose h is Turing Computable.
Let H be a Turing machine that computes h.
If h is a Turing computable, then H exists.
If H exists, then $D(C-H-F)$ exists.
Let $D = M_k$, for some k. $M_k \in L_M$.

Start D with input k. The C-part of D will produce a copy of k, Then the $H-$part will do its job:

- If M_k will eventually halt after starting with input k, then H will produce output 1.
- If M_k will never halt after starting with input k, then H will produce output 2.

Then the $F-$part will do it's job.

- If output from H is 1, F will never halt.
- If output from H is 2, F will eventually halt.

Giving us:

- If M_k will eventually halt after starting with input k, then D will never halt ater starting with input k.
- If M_k will never halt ater starting with input k, then D will eventually halt after starting with input k.

So M_k will halt, after starting with input k, $\iff$ D will nver halt after starting with input k.
Then M_k isn't identical with D, which is a contradiction! Hence$D$ doesn't exist. So H does not exists. So h is not turing computable. $\square$

Another halting problem..? $L_M : M_1, \cdots$ $L_F : F_1, \cdots$
$g(n) = 1$, if $f_n(n) = 2$
$g(n) = 2$, otherwise.
$g \neq f_k \forall k$
$h(m,n) = 1$if M_m eventuall halys after starting with input n.
$h(m,n) = 2$ if M_m never halts after starting with input n.
$s(m) = 1$, if M_m eventually halts after starting with input m.
$s(m) = 2$, if M_m never halts after starting with input m.

1. The halting problem is solvable iff h is computable.
2. If h is computable, then s is computable.
3. If s is computable, and TT's is true, then g is computable.
4. g is not Turing computable.
5. Turing Thesis is true (Whatever is not Turing Computable is not computable)

6. The halting problem is not solvable.

3. Suppose S is computable and TT is true.
Then: There is a Turing machine $S*$ that computes s.
Suppose that we are to calculate $g(n)$, for some n.
Start $S*$ with input n.

- Case 1: S* eventually halts with output 1
  We know that M_n will eventually ahlt after it is started with input n Start M_n with input n, when it halts, inspect the tape.
  - Case 1.1: Halted in SFC $f_n(n) = 2$ $g(n) = 1$
  - Case 1.2: Halted in non-SFC: f_n is undefined. $f_n(n) \neq 2$

  And then Blake broke it:
  As it's a halting problem to figure out if it's in SFC?
  $g(n) = 2$
- Case 2: S* eventually halts with output 2 We know that M_n will never halt after it is started with input n.
  So we know that f_n is undefined for the argument n.
  So we know that $g(n) = 2$

□

□

0.1 Sentental logic

Some symbols n things:

- (
-)
- Successor : '
- Not: -
- And: $\wedge$ (Conjunction)
- Or: $\vee$ (Disjunction)
- Exists: $\exists$
- Forall: $\forall$
- Variables: $v_1, v_2, v_3, \ldots$
- Equality: $=$
- Predicates: $\begin{matrix} A_1^1 & A_2^1 & \ldots \\ \vdots & \vdots & \ddots \\ A_1^n & A_2^n & \ldots \end{matrix}$
- Constant names: $a_1, \ldots$

- Functions: $\begin{matrix} f_1^1 & f_2^1 & \dots \\ \vdots & \vdots & \ddots \\ f_1^n & f_2^n & \dots \end{matrix}$

Definition (Term)**.**
- Every variable is a(n atomic) (open) term.
- Every constant is a(n atomic) (closed) term.
- If $t_1, \dots, t_n$ are terms, then $f^n(t_1, \dots, t_n)$ is a term.
- Nothing else is a term.

Definition (Formula)**.**
- $A^n(t_1, \dots, t_n)$ is a formula where A^n is an $n-$place predcate and t_i are terms. (This is an atomic formula).
- If F is a formula then $-F$ is a formula
- If F and G are fomulas then $(F \wedge G)$ is a formula.
- If F and G are fomulas then $F(\vee G)$ is a formula.
- If F is a formula, then $\exists vF$ is a formula.
- If F is a formula then $\forall vF$ is a formula.
- NOTHING ELSE IS A FORMULA.

Definition (Bound)**.** An occurrence of variable x is bound if it is part of a subformula beginning $\forall x$ or $\exists x$, in which case the quantifier $\forall$ or $\exists$ in question is said to bind that occurrence of the variable x, and otherwise the occurrence of the variable x is free.
Ex: $Fx \rightarrow \forall xFx$: The first x is free, and the second is bound.
Ex: $x < y \wedge -\exists z(x < z \wedge z < y)$: all occurrences of x and y are free, and all the occurrences of z are bound.

Definition (Model)**.** A model M (interpretation) of a language L is $\{|M|v\}$ Where $|M|$ is a non-empty set and v is a valuation function that assigns values (extensions/denotations) to the mebers of L in such a way that

- $\forall v(a) \in |M|$
- $v(A^n) \subseteq$ the nth caresian product of $|M|$ with itself: $|M| \times \dots |M|$.
- $v(f^n)$ is a total function from $|M| \times \dots |M|$ to $|M|$.

Definition (Truth)**.**
- $M \vDash F^n(t_1, \dots, t_n)$ iff $< M(t_1), \dots, M(t_n) >\in M(F^n)$.
- $M \vDash -SiffM \neq S$.
- $M \vDash (K \wedge L)iffM \vDash KandM \vDash L$
- $M \vDash \forall xF$
- $M \vDash \exists xF(c)iff$ there is an object $o \in |M|$ and given a name c (that is not interpreted by M), $M :\vDash F(c)$

Definition. of the denoteizan/extension of a closed term in a model M. If T is a name $M(t) = v(t)$ If t is $f^n(t_1, \dots, t_n)$ then
$M(f^n(t_1, \dots, t_n)$ $M(f^n)(M(t_1), \dots, M(t_n))$.

Validy = Satisfiability = Implication.

Misc-crap:

- $A \vDash B$ is $-(A \wedge -B)$

Lemma 1. *Extensionality Lemma*

- *Let M be a model of a language L.*
- *Let S be a sentence of L.*
- *Let L^+ be an extension of L. $L \subseteq L^+$*
- *Let M^+ be a model of L^+*
- *So: M^+ is an extension of M.*
- *$M \vDash S$ iff $M^+ \vDash S$*

Example:
If $A \vDash B$ and $B \vDash C$, then $A \vdash C$. Suppose $A \vDash B$, and $B \vDash C$.
In every interpretation of A and B in which A is true, B is true, In everry interpretation of B and C in which B is true, C is true. Shows: In every interpretation of A and C in which A is true, C is true.
Let M be an interpretation of A and C such that $M \vDash A$.

- Case 1: M is an interpretation of B.
  Then $M \vDash B$
  So $M \vDash C$.

- M is not an interpretation of B.
  Then there is an extension M^+ that interprets B as well as A and C.
  so: $M^+ \vDash B$
  So: $M^+ \vDash C$
  So $M \vdash C$ (By the ext, llemma)

Lemma 2 (Undecibality)**.** *If the decision problem (for implication) is solvable, then the halting problem is solvable. There is an effective methoid for specifiing for any Turing machine M and any input N a finite set of setences Δ and a sentence H such that $\Delta \vDash H$ iff M eventually halts after starting with input n.*
$\Delta \vDash H$ iff M eventualy alts after start with input n.

Define the one place predicate $Q_i j$ as: At time j, M is in state i. Define the two place preidcate $@js$ as At time j, M, is scanning square s. Define the two place predicate Mjs as: At time j, square s is marked with a 1.
A description D for a start state could then be: $D : [Q_1 0 \wedge @_{0,0} \wedge M_{0,0} \wedge M_{0,1} \wedge M_{0,2} \wedge \forall y((y \neq 0 \wedge y \neq 1 \wedge y \neq 2) \implies -M_{0,y}]$ Time = 0, $[\ldots 01110 \ldots]$ Square #, $[\cdots - 10123 \ldots]$

For each instructon of a TM, we may write the instruction as a setence:
$Q_{i1} S_1 R Q_{i2}$: Move right seeing 1 in state:
$\forall x \forall t((Q_{i,1} \wedge @_{t,x} \wedge M_{t,x}) \implies (Q_{i2,(t+1)} \wedge @_{(t+1),(x+1)} \wedge \forall y((M_{t,y} \implies M_{(t+1),y}) \wedge (-M_{t,y} \implies -M_{(t+1),y}))$

Misc-crap that's on the board for some reason:

$$\begin{array}{rr}
\Delta & \\
\mathbb{Q}_2^1: & \forall x \forall y \forall z((Sxy \wedge Sxz) \implies y = z) \\
@^2: & \forall x \forall y \forall z((Sxz \wedge Syz) \implies x = y) \\
M^2: & \forall x \forall y(Sxy \implies x < y) \\
0: & \forall x \forall y \forall z((x < y \wedge y < z) \implies x < z) \\
S^2: & -\exists x, x < x{+}1 \\
<^2: & \text{lessthan}
\end{array}$$

Some thingelse now too:
$\forall x \forall t((Q_1 t \wedge @tx \wedge Mtx) \implies \exists u(s_1(t,u) \wedge Q_2(u) \wedge @(u,v)) \wedge \forall y((M(t,y) \implies M(u,y)) \wedge (-M(t,y) \implies -M(u,y)))))$
$\exists x \exists t(Q_m t \wedge @tx \wedge Mtx)$

Proof. Something about biconditonal:
$\Delta \vDash H$ iff M halts after starting with input n.

1. if $\Delta \vDash H$, then M halts.
2. if M halts, then $\Delta \vDash H$.

1. if $\Delta \vDash H$, then M halts - Proof.
   Suppuse $\Delta \vDash H$
   All members of Δ are true in the standard interpretation I. H is true in I. So: M halts.
2. if M halts, then $\Delta \vDash H$ - Proof.
   Suppose M halts. (Show $\Delta \vDash H$).
   There is a time $\underline{t}$ M halts at t.
   There is a state q_i, M halts at t in state q_i.
   There is a square x, M halts at t in state q_i, scanning square x which is Marked / $-$ Makred
   1: $Q_i(t) \wedge @(t,x) \wedge M(t,x)$.
   1 is a cojunct of the description of time t, $\mathbb{D}(t)$
   $\mathbb{D}(t) \vDash (i)$

   2: $\exists x \exists t(Q_1(t) \wedge @(t,x) \wedge (t,x))$.

   $(i) \vDash (ii)$.
   (ii) is disjunct of H.
   $(ii) \vDash H$.
   So: $\mathbb{D}(t) \vDash H$.

   Δ implies a description of everytime before which M did not halt.
   $\forall n($ if M has not halted before time n, then $\Delta \vDash \mathbb{D}(n)$

□

First-Order Logic Revamp: Some definitions for first-order logic:

- Logical Symbols:
  - Negation: : 'not'
  - Conjunction: &, $\wedge$: 'and'
  - Disjunction: $\vee$: 'or'
  - Conditional: $\rightarrow$: 'if ... then ... '
  - Biconditional: $\leftrightarrow$: 'if and only if'
  - Universal quantification: $\forall x$: 'for every x'
  - Existential quantification: $\exists x$: 'for some x'
  - Identity symbol: $=$: '... is (the very same thing as) ... '
  - Variables: x, y, z
  - Punctuation: '(', ')' , ','
- Nonlogical Symbols:
  - Constants or Individual symbols (a,b,c ..)
  - Predicates or Relation symbols : Have a fixed positive number of places
  - Function symbols: : Have a fixed positive number of places.
- Other-Definitions:
  - language: an enumerable set of nonlogical symbols. Denoted: L
    Empty Language: $L_{\varnothing}$, The language with no logical symbols.
- Closure:
  - Closed Function: they make a complete statement capable of being true or false
  - Closed Term: Has no variables.
- Interpretation: An interpretation M for a language L consists of two components.
  - A nonempty set $|M|$, called the domain or universe of discourse of the set of things M.
  - For each non-logical symbol, a denotation assigned to it. Noted by x^M.