PHIL 303 Midterm Review[1]
Angela Sun
Winter 2020

Contents

1 Symbolization **1**
- 1.1 *SL* . . . 1
- 1.2 Symbolization . . . 2
- 1.3 Practice problems . . . 3

2 Truth-tables **3**
- 2.1 Filling out truth-tables . . . 3
- 2.2 Doing things with truth-tables . . . 5
- 2.3 Truth-tables and truth-functional truth/falsity . . . 5
- 2.4 Truth-tables and truth-functional consistency/inconsistency . . . 6
- 2.5 Truth-tables and truth-functional validity/invalidity . . . 7
- 2.6 Practice problems . . . 7

3 Derivations **7**
- 3.1 Derivation rules . . . 7
- 3.2 Setting up a derivation . . . 7
- 3.3 Strategies for constructing derivations . . . 8
  - 3.3.1 Exploit contradictions . . . 8
  - 3.3.2 Work backward . . . 9
- 3.4 Practice problems . . . 10

4 Definition by Recursion **10**
- 4.1 Practice problems . . . 11

5 Proof by Induction **11**
- 5.1 Practice problems . . . 12

1 Symbolization

1.1 *SL*

SL is a language that allows us to analyze some logically relevant features of sentences. "Symbolization" is the process of translating English to *SL*.

There are two kinds of sentences in *SL*:

1. *Atomic sentences*: capital letters (sentence letters), e.g. A, C, W
2. *Compound sentences*: capital letter(s) and at least one sentential connective, e.g. $\sim A$, $B \equiv E$, $\sim (D \vee C) \supset (D \& F)$

[1] All page references for practice problems are for *The Logic Book*, 6th ed.

There are five sentential connectives in *SL*:

1. Negation ($\sim$): "not", "it is not the case that," etc.

A	$\sim$A
T	F
F	T

2. Disjunction ($\vee$): "or", "unless", "either," "at least one," etc.

A	B	A$\vee$B
T	T	T
T	F	T
F	T	T
F	F	F

3. Conjunction (&): "and", "both", "but", "however", "moreover", etc.

A	B	A&B
T	T	T
T	F	F
F	T	F
F	F	F

4. Material conditional ($\supset$): "if... then...", etc.

A	B	A&B
T	T	T
T	F	F
F	T	T
F	F	T

5. Material biconditional ($\equiv$): "... if and only if...", "... just in case...", etc.

A	B	A$\equiv$B
T	T	T
T	F	F
F	T	F
F	F	T

1.2 Symbolization

When translating English to *SL*, you must:

1. *Paraphrase* the sentence(s) you're translating to make the logical connections clear.
2. Specify a *symbolization key*. The symbols should map every phrase in the text you are translating to its corresponding atomic sentence in *SL*.
3. *Symbolize* the sentence(s) you're translating.

Consider the following argument:

> Angela will go to Panera if and only if Victoria goes. If Victoria goes, then Rebecca will go, too. Unfortunately, Izzi can't make it. So if Victoria goes to Panera, only Rebecca and Angela will be there.

Paraphrase:

> Angela goes to Panera if and only if Victoria goes to Panera.
> If Victoria goes to Panera, then Rebecca goes to Panera.
> It is not the case that Izzi goes to Panera.
>
> ---
>
> If Victoria goes to Panera, then (Rebecca goes to Panera and Angela goes to Panera).

Symbolization key:

- A: Angela goes to Panera
- V: Victoria goes to Panera
- R: Rebecca goes to Panera
- I: Izzi goes to Panera

Symbolization:

> $A \equiv V$
> $V \supset R$
> $\sim I$
>
> ---
>
> $V \supset (R \& A)$

1.3 Practice problems

For practice with simple translations, see pp. 36-37. For more complicated symbolizations, see pp. 55-57. There's a very helpful summary of the logical connectives on p. 54.

2 Truth-tables

Truth-tables allow us to determine the conditions under which sentences of SL are true, sets of sentences of SL are consistent, and arguments of SL are valid.

2.1 Filling out truth-tables

To fill out a truth-table:

1. List the atomic sentence(s) in columns the very left side of the table in alphabetical order.
2. Fill out the columns with the atomic sentences systematically.
   - Say that you have n letters, so that there are 2^n rows to the truth-table. Assign the first letter a block of 2^{n-1} T's and then a block of 2^{n-1} F's. Then assign the next letter a block of 2^{n-2} T's, then 2^{n-2} F's, 2^{n-2} T's and then finally 2^{n-2} F's... And the last letter will get alternating T's and F's.
3. List the compound sentence(s) you're evaluating in columns to the right of the atomic sentences.
4. Copy the truth values of the atomic sentences in the columns they appear in in the compound sentence(s).
5. Fill in the columns under the logical connectives. Fill in the main logical connective last, working your way there by filling in the more immediate logical connectives first.

The truth values in the column with the main logical connective represent the truth value of the sentence with the corresponding truth-value assignments to atomic sentences.

Suppose you want to construct a truth-table for sentence $(A\& \sim B) \supset C$. Start by listing the atomic sentences in alphabetical order on the left side of the table and filling out the truth values using the algorithm above:

A	B	C	$(A$	&	$\sim$	B)	$\supset$	C
T	T	T						
T	T	F						
T	F	T						
T	F	F						
F	T	T						
F	T	F						
F	F	T						
F	F	F						

Next, copy the truth values of the atomic sentences in the columns they appear in in the compound sentence:

A	B	C	$(A$	&	$\sim$	B)	$\supset$	C
T	T	T	T			T		T
T	T	F	T			T		F
T	F	T	T			F		T
T	F	F	T			F		F
F	T	T	F			T		T
F	T	F	F			T		F
F	F	T	F			F		T
F	F	F	F			F		F

Now we want to fill in the columns under the logical connectives. *Fill in the main logical connective last*, working your way there by filling in the more immediate logical connectives first. The first connective you should fill out in this table is the $\sim$:

A	B	C	$(A$	&	$\sim$	B)	$\supset$	C
T	T	T	T		F	T		T
T	T	F	T		F	T		F
T	F	T	T		T	F		T
T	F	F	T		T	F		F
F	T	T	F		F	T		T
F	T	F	F		F	T		F
F	F	T	F		T	F		T
F	F	F	F		T	F		F

Next fill in the column with the & and finally the column with the $\supset$, completing the truth-table:

A	B	C	$(A$	&	$\sim$	B)	$\supset$	C
T	T	T	T	F	F	T	**T**	T
T	T	F	T	F	F	T	**T**	F
T	F	T	T	T	T	F	**T**	T
T	F	F	T	T	T	F	**F**	F
F	T	T	F	F	F	T	**T**	T
F	T	F	F	F	F	T	**T**	F
F	F	T	F	F	T	F	**T**	T
F	F	F	F	F	T	F	**T**	F

2.2 Doing things with truth-tables

We can use truth-tables to say the following things...:

- About *sentences* of *SL*:
  - A sentence of *SL* is *truth-functionally true* iff it is true on every truth-value assignment.
  - A sentence of *SL* is *truth-functionally false* iff it is false on every truth-value assignment.
  - A sentence of *SL* is *truth-functionally indeterminate* iff it true on at least one truth-value assignment and false on at least one truth-value assignment.
  - Two sentences of *SL* are *truth-functionally equivalent* iff they are true on exactly the same truth-value assignments.
- About *sets of sentences* of *SL*:
  - A set of sentences of *SL* is *truth-functionally consistent* iff there is at least one truth-value assignment that makes all of the sentences in the set true.
  - A set of sentences of *SL* is *truth-functionally inconsistent* iff there is no truth-value assignment that makes all of the sentences in the set true.
- About *arguments* with sentences of *SL*:
  - An argument with sentences of *SL* is *truth-functionally valid* iff there is no truth-value assignment that makes all of the premises true and the conclusion false.
  - An argument of *SL* is *truth-functionally invalid* iff there is a truth-value assignment that makes all of the premises true and the conclusion false.

2.3 Truth-tables and truth-functional truth/falsity

You can use truth-tables to show that a sentence is truth-functionally true by showing that the column under the main logical connective of the sentence contains only T's.

The following truth-table demonstrates that the sentence $\sim (A\& \sim A)$ is truth-functionally true. The main logical connective of the sentence is the first $\sim$, and the column under that symbol shows only T's:

A	$\sim$	(A	&	$\sim$	A)
T	T	T	F	F	T
F	T	F	F	T	F

Likewise, you can use truth-tables to show that a sentence is truth-functionally false by showing that the column under the main logical connective of the sentence contains only F's. The following truth-table demonstrates that the sentence $\sim (A \vee \sim A)$ is truth-functionally false:

A	$\sim$	(A	$\vee$	$\sim$	A)
T	F	T	T	F	T
F	F	F	T	T	F

2.4 Truth-tables and truth-functional consistency/inconsistency

To show that a set of sentences of *SL* is consistent, you can show a *shortened truth-table* that demonstrates that there is one truth-value assignment that makes all of the sentences in the set true:

1. Begin by listing the atomic sentences in columns on the left side of the table and the sentences in the set on the right side of the table.
2. Mark the main logical connectives with a T.
3. Work backwards to find a truth-value assignment to atomic sentences that makes the all the T assignments compatible.

Suppose you want to demonstrate that $\{(A \equiv B) \equiv A, B \& \sim A\}$ is truth-functionally consistent. Begin by setting up the table and marking the main logical connectives with a T:

A	B	$(A$	$\equiv$	B)	$\equiv$	A	B	&	$\sim$	A
					T			T		

Next, work backward. Look for places where you are constrained in what truth values you can assign. For instance, for & to be T, the two conjuncts must be T:

A	B	$(A$	$\equiv$	B)	$\equiv$	A	B	&	$\sim$	A
					T		T	T	T	

Now you know that A is F and B is T.

A	B	$(A$	$\equiv$	B)	$\equiv$	A	B	&	$\sim$	A
F	T	F		T	T	F	T	T	T	F

And now you can complete the rest of the table:

A	B	$(A$	$\equiv$	B)	$\equiv$	A	B	&	$\sim$	A
F	T	F	F	T	T	F	T	T	T	F

To show that a set of sentences of *SL* is inconsistent, you must show a full truth-table that demonstrates that there is no truth-assignment on which all the sentences in the set are true.

2.5 Truth-tables and truth-functional validity/invalidity

To show that an argument of *SL* is valid, you must show a full truth-table that demonstrates that there is no truth-value assignment on which all the premises are true and the conclusion is false.

To show that an argument of *SL* is invalid, you can show a shortened truth-table that demonstrates that there is one truth-value assignment that makes all the premises true but the conclusion false. Use the same method for constructing shortened truth-tables as you did before for consistency, but instead of marking *all* the main logical connectives true, mark the main logical connectives of the premises true and the main logical connective of the conclusion false.

2.6 Practice problems

For practice with constructing truth-tables, see p. 76. For practice with truth-functional truth, falsity, and indeterminacy, see pp. 85-87. For practice with truth-functional equivalence, see pp. 91-92. For practice with truth-functional consistency/inconsistency, see pp. 94-95.

3 Derivations

Derivations provide a way of mimicking the kind of logical reasoning we use in everyday discourse. They use *syntactical* techniques; derivations are not intended to reveal whether or not there is a truth-value assignment of a certain sort that makes the argument valid, but whether the form or structure of the sentences of *SL* makes the argument valid.

3.1 Derivation rules

We use the derivation system *SD*. There are 11 derivation rules in *SD*: one introduction rule for each logical connective, one elimination rule for each logical connective, and reiteration. **No other rules may be used.**

To view the derivation rules, see this handout, which can also be found on the inner cover of *the Logic Book* and on the 303 Canvas page. Here is a copy of the same handout, but with additional instructions for writing line justifications.

The key to getting good at derivations is getting lots of practice. Do as many practice problems from the textbook as you can to gain confidence in applying the derivation rules.

3.2 Setting up a derivation

Derivations should be set up so that the premises of an argument (if there are any) are listed at the top, above a horizontal line, and the conclusion is written at the bottom. Each line should be numbered, and each line should have a justification. The justification can either be an assumption or point to an application of one of the derivation rules.

Here's a sample derivation of $A \supset H$ from the premises $A \supset G$ and $G \supset H$:

1	$A \supset G$	Assumption
2	$G \supset H$	Assumption
3	$\quad A$	A / $\supset I$
4	$\quad G$	1,3 $\supset E$
5	$\quad H$	2,4 $\supset E$
6	$A \supset H$	3-5 $\supset I$

3.3 Strategies for constructing derivations

Here are a couple of strategies that may be helpful when you're doing derivations. **These strategies will not work in every derivation**; sometimes you'll need to think more carefully about how to approach a derivation before you begin. Nevertheless, these strategies will generally be helpful at some point in every derivation you do.

§5.3 (pp. 179-214) of *The Logic Book* goes into more detail on some strategies for construction derivations in *SD*. I recommend reading through it.

3.3.1 Exploit contradictions

Once you have a contradiction your proof, you can get whatever you want *in just three lines*! To do this, assume the opposite of what you want, reiterate the contradiction, and then discharge the thing that you want using either $\sim$ introduction or $\sim$ elimination.

Suppose that you want to derive $B \equiv (C\&D)$ from the premises $A \vee B$ and $\sim (A \vee B)$

$A \vee B$	Assumption
$\sim (A \vee B)$	Assumption
$\vdots$	
$B \equiv (C\&D)$	

There is a contradiction available for you to use in the premises, so you know you'll be able to get to the conclusion in just three lines by assuming the negation of the sentence you want to derive and using $\sim$ elimination:

1	$A \vee B$	Assumption
2	$\sim (A \vee B)$	Assumption
3	$\quad \sim (B \equiv (C\&D))$	A / $\sim E$
4	$\quad A \vee B$	1R
5	$\quad \sim (A \vee B)$	2R
6	$B \equiv (C\&D)$	3-5 $\sim E$

This completes the proof!

3.3.2 Work backward

In general, it's not a good idea to work from the top of the derivation to the bottom. Instead, you should *work backward* by identifying the main logical connective of the conclusion and using the corresponding introduction rule.

Suppose you want to derive $A \supset\sim B$ from the premises $A \supset (B \supset C)$ and $\sim C$:

$A \supset (B \supset C)$	Assumption
$\sim C$	Assumption
$\vdots$	
$A \supset\sim B$	

The main logical connective of the conclusion is $\supset$. So you know that you will use $\supset$ introduction to get there. You can set up your proof accordingly:

$A \supset (B \supset C)$	Assumption
$\sim C$	Assumption
$\quad A$	A / $\supset I$
$\quad \vdots$	
$\quad \sim B$	
$A \supset\sim B$	$\supset I$

Your new goal is to get to $\sim B$ in the subderivation. Once you get to $\sim B$, then you are free to use $\supset$ introduction to get $A \supset\sim B$.

Now that your new goal is $\sim B$, you can look at the sentence, see that the main logical connective is a $\sim$, and use the corresponding introduction rule to get it:

$A \supset (B \supset C)$	Assumption
$\sim C$	Assumption
$\quad A$	A / $\supset I$
$\quad\quad B$	A / $\sim I$
$\quad\quad \vdots$	
$\quad \sim B$	$\sim I$
$A \supset\sim B$	$\supset I$

Now your new goal is to get a contradiction in the subderivation. Once you get that, then you'll be done the derivation! To get the contradiction, you can use conditional elimination and reiteration:

1	$A \supset (B \supset C)$	Assumption
2	$\sim C$	Assumption
3	A	A / $\supset I$
4	B	A / $\sim I$
5	$B \supset C$	1,3 $\supset E$
6	C	4,5 $\supset E$
7	$\sim C$	2R
8	$\sim B$	4-7 $\sim I$
9	$A \supset \sim B$	3-8 $\supset I$

This completes the proof.

By working from bottom to top—identifying the main logical connective of the conclusion and using the corresponding introduction rule—rather than top to bottom, we were able to efficiently complete the derivation. The strategy of working backward is even more important when you encounter longer, more complicated derivations because it allows you to break down those derivations into smaller and smaller sub-problems.

3.4 Practice problems

The only way to learn to do derivations well is to get plenty of practice. Derivations can get quite complex and you need to gain confidence in applying the derivation rules.

For practice problems, see pp. 209-214. See also pp. 222-223, but use only SD and not $SD+$ to complete your derivations.

4 Definition by Recursion

A recursive definition is used to define elements in a set in terms of other elements in a set. A recursive definition has three parts:

1. *Base clause*: define the most basic case.
2. *Recursion clause*: stipulate that if you take something that already counts and then apply an operation to it, then you get something else that counts.
3. *Closure close*: say that nothing else counts.

Consider the recursive definition of natural numbers:

1. *Base clause*: 1 is a natural number.
2. *Recursion clause*: if x is a natural number, then $x + 1$ is a natural number.
3. *Closure clause*: nothing else is a natural number.

And the recursive definition of SL:

1. *Base clause*: every sentence letter is a sentence of SL.

2. *Recursion clause*:
    - If **P** is a sentence of SL, then $\sim$**P** is a sentence of SL
    - If **P** and **Q** are sentences of SL, then (**P**&**Q**) is a sentence of SL.
    - If **P** and **Q** are sentences of SL, then (**P**$\vee$**Q**) is a sentence of SL.
    - If **P** and **Q** are sentences of SL, then (**P**$\supset$**Q**) is a sentence of SL.
    - If **P** and **Q** are sentences of SL, then (**P**$\equiv$**Q**) is a sentence of SL.
3. *Closure clause*: nothing else is a sentence of SL.

4.1 Practice problems

Define the following things recursively (if you'd like to check your answers, send me an email!):

1. The even numbers
2. The Fibonacci sequence (a sequence of numbers beginning with 0 and 1, with each subsequent number being the sum of the previous two)
3. Say that the language IFF consists of all the sentences of SL satisfying these conditions: (i) the only sentence letter that can occurin the sentence is A. (ii) The only connective in the sentence is $\equiv$. Give a recursive definition of "Sentence of IFF".

5 Proof by Induction

Mathematical induction is a proof technique that allows you to show that if a property holds for some base case, then it will hold for every natural number greater than the base case.

A proof by induction for some claim C has three parts:

1. *Base case*: state that C holds for some basic case, like 0 or 1.
2. *Induction hypothesis*: assume that C holds for n.
3. *Induction step*: given the induction hypothesis, prove that C holds for $n+1$.

Here is an example of a proof by induction. Suppose that there is a language called "AND" that consists in just sentence letters and the logical connective &. Prove that every sentence S in AND has the same number of right and left parentheses.[2]

1. *Base case*: When $n = 1$, S is a sentence letter. It therefore has the same number of right as left parentheses, namely 0.
2. *Induction hypothesis*: assume that for every S with n symbols such that $1 < n < k$, S has the same number of right as left parentheses.
3. *Induction step*: prove that S of length k has the same number of right as left parentheses.
    - S is not a sentence letter because its length is greater than 1, so it must be of the form $(S_1 \& S_2)$, with S_1 and S_2 being sentences of AND.
    - S_1 and S_2 both have fewer than k symbols, so by the induction hypothesis, they have the same number of left and right parentheses.

[2] This example is adapted from lecture 6, pp. 30-33.

- So, the total number of right parentheses in S is $1+$ (the number of right parentheses in S_1) $+$ (the number of right parentheses in S_2), and the total number of left parentheses in S is $1+$ (the number of left parentheses in S_1) $+$ (the number of left parentheses in S_2).
- These two numbers must be equal, since they are sums of three numbers, each number equal to its counterpart in the other sum.
- This completes the induction step, and so the proof.

5.1 Practice problems

Again, let me know if you'd like to go over any answers!

1. Prove that for all natural numbers n,

$$\sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{n}{n+1}$$

2. Prove that if **Q** is a sentence of IFF (as defined in §4.1 of this document) with an even number of occurrences of "≡", it will be assigned F when A is assigned F. You can use the following two facts without proof:
   (a) If **P** is any sentence of IFF with an *odd* number of occurrences of "≡" and the sentence letter A is assigned F, then **P** is assigned T.
   (b) If a sentence **Q** of IFF is of the form **P**≡**R**, and **Q** has an even number of occurrences of "≡", then one of **P** and **R** has an even number of occurrences of "≡" and the other one has an odd number of occurrences of "≡".