Logic I Course Notes

Spring 2018

James Garson

Logic I Study Guide (to help study for quizzes and final)

Quiz 1 Reading: Ch. 1-2 of Propositional Logic

Concepts: Argument, Premise, Conclusion, Statement, Validity, Soundness, Inductive Logic, Deductive Logic, Modus Ponens, Conditional, Antecedent, Consequent

Skills:

* Distinguish validity of an argument from truth of a statement.
* Give the two definitions of validity.
* Translate conditionals using indicator words (Notes, p.4) to identify antecedents and consequents
* Recognize the conclusion and premises of arguments in English using indicator words, on p. 17, Propositional Logic, (also Notes p. 45) and their synonyms.
* Complete simple proofs using →Out.

Quiz 2 Reading: Ch. 3-4 of Propositional Logic

Concepts: Ampersand, Conjunction, Conjunct, Subproof

Skills:

* Recognize main connectives of complex formulas.
* Complete proofs using →, &, rules.
* Employ correct strategy for creating subproofs for the →In rule.
* Know the rules for boxes found in Notes p. 5

Quiz 3 Reading: Ch. 5-6 of Propositional Logic

Concepts: Indirect Proof, Reductio ad Adsurdum, Standard Contradiction

Skills:

* Accurately diagnose the scope of negation in translating English statements
* Construct proofs using the negation rules (-In and -Out)
* Understand an appropriately apply the strategies for the ↔ Rules.

Quiz 4 Reading: Ch. 7, 8.1, 8.2, of Propositional Logic

Concepts: Disjunction, Disjunct, Exclusive or, Inclusive or, Dilemma, Soundness (Consistency) of Proof Rules, Completeness of Proof Rules, Counterexample

Skills:

* Understand and apply the strategies for the v Rules, especially the vOut strategy. That is, when AvB appears in the proof, and your last goal is G, know to set the "missing conditionals" A→G and B→G as new goals, and derive G by vOut.
* Be able to complete more difficult proofs.

Quiz 5 Reading: Ch. 9.2 Ch. 10.1 Propositional Logic,

Concepts: Derived Rule,

Skills:

* Know how to demonstrate that a rule is derivable.
* Know the Highly Recommended and Recommended derived rules. (Notes p. 9.)
* Appropriately employ the strategies for using the Highly Recommended rules, that is, use AR and DM to simplify sentences with the forms: -(AvB), (A→B), -(A&B), AvB.
* Know how to handle a disjunction as a goal by either converting to → or using the DeMorgan strategy.
* Calculate the truth table for a statement.

* Calculate the truth table for an argument and determine its validity status.

Quiz 6 Reading: "How to Make a Tree", Notes pp. 10-22 Ch. 12-13 of Propositional Logic

Concepts: Valid (TP), Contradiction (TP), Contingent (T), Logical Truth (TP), Tautology (TP), Equivalence (TP), Entailment (TP), Consistency (T).

Skills:

* Know how to apply all the tree rules.
* Construct trees for arguments and diagnose their validity status.
* Know how to test all concepts listed above which are marked with "T" with truth tables and trees. Be able to test all concepts marked "P" with proofs.

Final: Reading: Ch 2-4 of Predicate Logic

Concepts: Instance Explain the relationship between validity and he following notions:

* Counterexample (A valid argument has no counterexample),
* Closed Tree (An argument is valid if the tree formed from the premises and the negated conclusion is closed in every branch).

Skills:

* Be able to distinguish singular from general terms. Especially, be able to diagnose such sentences as "A dog got into the garbage" as general, and "The dog got into the garbage" as singular.
* Be able to translate simple predicate logic sentences with the forms Some A are B; All A are B; No A are B; Not all A are B.
* Complete proofs and trees using the predicate logic rules ∀O, ∃O and QE.
* Order the use of the predicate logic rules properly: QE, ∃O,∀O.
* Understand, and avoid breaking, the restriction on the ∃O rule.
* Be able to construct a tree in predicate logic quickly.

Review for Final: you should know all of the above plus:

Skills

* Remember all expressions on the Translation Charts in Notes p. 4 and 32.
* Recognize main connectives, and apply rules only to main connectives.
* Remember perfectly the 10 primitive rules on p. 272, Propositional Logic, (Notes p. 8) the rules DM, AR, and DN, (Notes p. 9) and the quantifier rules QE, ∃O and ∀O.
* Use the backwards proof finding strategy. Make provisional assumptions (PA) only when appropriate, i.e. only when setting up →IN, -IN or -OUT.
* Know, and be able to use, the strategy suggestions on p. 115, Propositional Logic (or Notes p. 6 and p. 7)
* Remember truth tables perfectly, and be able to calculate truth tables quickly.
* Know the tree rules (Notes p. 21 or Propositional Logic, p. 233-234) and be able to construct trees and counterexamples.

Propositional Logic Translation Chart

A → B
If A, (then) B
B if A
provided that A, B
B provided that A
A only if B
A is sufficient for B
B is necessary for A

A & B
(Both) A and B
A but B
A however B
A although B
A nevertheless B
A moreover B
A yet B
A even though B

Antecedent Indicators:
if _
provided that _
_ is sufficient
should _

Consequent Indicators:
then _
only if _
_ is necessary
results in _
brings about _
leads to _

Premise Indicators:
Because
Since
For

Conclusion Indicators:
Therefore
So
Hence
Thus
Consequently
It follows that

-A
Not A
It is false that A
A is mistaken
A is not the case
Negative Prefixes:
Un (unintelligent)
In (indescribable)
Im (immoral)

A ↔ B
A if and only if B (A iff B)
A just in case B
A exactly when B
A exactly if B
A but only if B
A is a necessary and sufficient condition for B

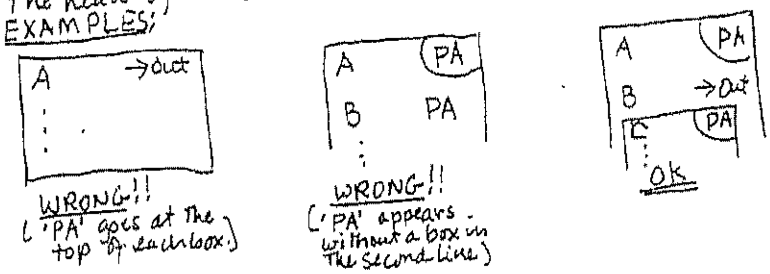
A v B
(Either) A or B
A unless B

-(AvB) or -A&-B
Neither A nor B

Rules for Boxes (Optional)

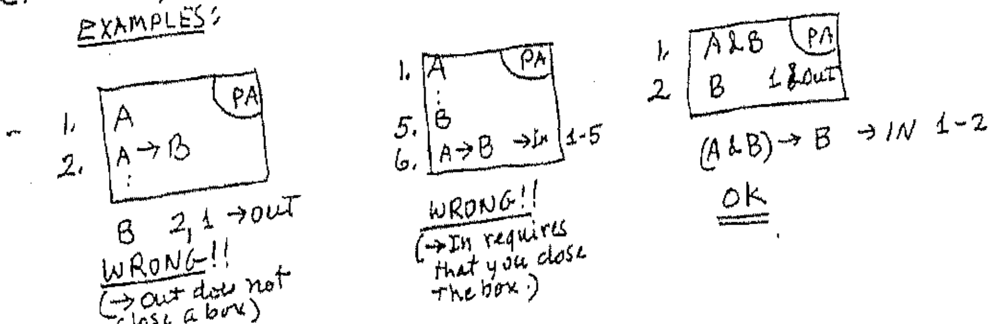
1. Every box is headed by 'PA' and 'PA' is used only at the head of a box.

EXAMPLES:



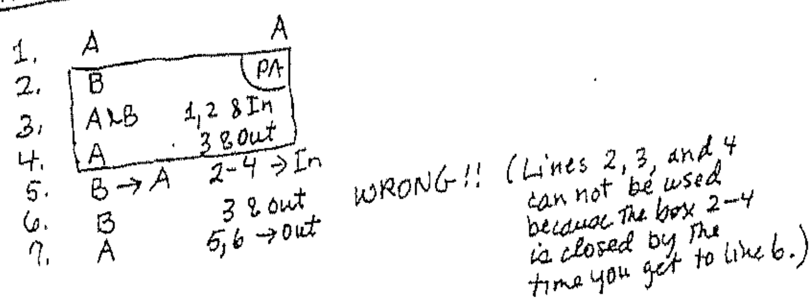
2. →In, ~In and -Out close a box, and no other rules do.

EXAMPLES:



3. When a box has been closed, no line inside the box can be used.

EXAMPLE:



Strategies for Proof Finding

If you see this:	Then Do this:
A → B GOAL	[A (PA) ⋮ B] GOAL A → B ~~GOAL~~ →In
A & B GOAL	A GOAL B GOAL A & B ~~GOAL~~ &IN
A ⇔ B GOAL	A → B GOAL B → A GOAL A ⇔ B ~~GOAL~~ ⇔IN
A ∨ B GOAL Do work elsewhere. If nothing else can be done try these...	A GOAL A ∨ B ~~GOAL~~ ∨In OR B GOAL A ∨ B ~~GOAL~~ ∨In OR ~A → B GOAL A ∨ B ~~GOAL~~ ARRO
A → G (AVAILABLE) ⋮ G GOAL	A → G A GOAL G ~~GOAL~~ →Out
A ∨ B (AVAILABLE) ⋮ G GOAL	A ∨ B A → G GOAL B → G GOAL G ~~GOAL~~ ∨OUT or A ∨ B (PROVEN) ~A → B (Arrow) AR G GOAL
~(A & B)	convert to A → ~B (Arrow) AR
~(A → B)	convert to A & ~B (Arrow) AR
~(A ∨ B)	convert to ~A & ~B (DeMorgan) DM
G GOAL (nothing works)	[~G (PA) ⋮ ?&~?] GOAL G ~~GOAL~~ I.P. (out) Try Indirect Proof. ?&~? stands for a contradiction
?&~? GOAL	How to choose a Contradiction 1. Prefer long negated formulas already available, 2. Avoid formulas already used as contradictions for I.P. 3. Avoid using GOAL for I.P. as part of the contradiction

PROOF STRATEGY

0. **Can I apply DN, &Out, →Out or ↔Out to lines already available?**
Yes? Then do it.

1. **What is my goal?**

2. **Is my goal $\mathcal{A}\to\mathcal{B}$, $\mathcal{A}\&\mathcal{B}$, or $\mathcal{A}\leftrightarrow\mathcal{B}$?**
Yes? Then assume it will come by the corresponding In rule. For example,
if it is $\mathcal{A}\to\mathcal{B}$, make a box with $\mathcal{A}$ a PA, and set $\mathcal{B}$ as a new goal.
If it is $\mathcal{A}\&\mathcal{B}$, set $\mathcal{A}$ and $\mathcal{B}$ as new goals.
If it is $\mathcal{A}\leftrightarrow\mathcal{B}$, set $\mathcal{A}\to\mathcal{B}$ and $\mathcal{B}\to\mathcal{A}$ as new goals.

3. **Is my goal $\mathcal{A}\vee\mathcal{B}$?**
Yes? Then if either $\mathcal{A}$ or $\mathcal{B}$ is available, use vIn. Otherwise work elsewhere.

4. **Does my goal appear to the right of → in an available line?**
Yes? Then set what is to the left of → as a new goal. When you get that goal, use →Out to get what was on the right of →. (Your old goal.)

5. **Is $\mathcal{A}\vee\mathcal{B}$ avaliable?**
Yes? Then set up vOut. That means set $\mathcal{A}\to G$ and $\mathcal{B}\to G$ as new goals, where G is your old goal.

6. **Is $\sim(\mathcal{A}\vee\mathcal{B})$ available?**
Yes? Then convert to $\sim\mathcal{A}\&\sim\mathcal{B}$ with DM.

7. **Is $\sim(\mathcal{A}\to\mathcal{B})$ or $\sim(\mathcal{A}\&\mathcal{B})$ available?**
Yes? Then use AR to convert this to $\mathcal{A}\&\sim\mathcal{B}$ or $\mathcal{A}\to\sim\mathcal{B}$ respectively.

8. **Are your stuck?**
Yes? Then use IP. Make a box with the opposite of your goal a PA and chose a contradiction as a new goal. In chosing a contradiction look for negative sentences in lines available. Long negations are a good choice. Do not use the goal for IP as part of your contradiction, and do not use contradictions previously used for IP.

A Graphical Summary of the Rules of Propositional Logic

	IN RULES	OUT RULES
$\rightarrow$	$\mathcal{A}$ (PA) : $\mathcal{B}$ ------- $\mathcal{A}\rightarrow\mathcal{B}$ →IN	$\mathcal{A}\rightarrow\mathcal{B}$ $\mathcal{A}$ ----- $\mathcal{B}$ →OUT
&	$\mathcal{A}$ $\mathcal{B}$ ---- $\mathcal{A}$ & $\mathcal{B}$ &IN	$\mathcal{A}$&$\mathcal{B}$ ---- $\mathcal{A}$ &OUT $\mathcal{A}$&$\mathcal{B}$ ---- $\mathcal{B}$ &OUT
$\vee$	$\mathcal{A}$ ---- $\mathcal{A}\vee\mathcal{B}$ ∨IN $\mathcal{A}$ ---- $\mathcal{B}\vee\mathcal{A}$ ∨IN	$\mathcal{A}\vee\mathcal{B}$ $\mathcal{A}\rightarrow\mathcal{C}$ $\mathcal{B}\rightarrow\mathcal{C}$ ----- $\mathcal{C}$ ∨OUT
$\leftrightarrow$	$\mathcal{A}\rightarrow\mathcal{B}$ $\mathcal{B}\rightarrow\mathcal{A}$ ----- $\mathcal{A}\leftrightarrow\mathcal{B}$ ↔IN	$\mathcal{A}\leftrightarrow\mathcal{B}$ ----- $\mathcal{A}\rightarrow\mathcal{B}$ ↔OUT $\mathcal{A}\leftrightarrow\mathcal{B}$ ----- $\mathcal{B}\rightarrow\mathcal{A}$ ↔OUT
~	$\mathcal{A}$ (PA) : ?&~? ------- ~$\mathcal{A}$ ~IN	~$\mathcal{A}$ (PA) : ?&~? ------- $\mathcal{A}$ ~OUT

Chapter 9.

Highly Recommended Derived Rules
(Use TOP-DOWN)

$$\frac{\sim(a \to B)}{a \,\&\, \sim B} \text{ AR} \qquad \frac{\sim(a \,\&\, B)}{a \to \sim B} \text{ AR}$$

$$\frac{\sim(a \vee B)}{\sim a \,\&\, \sim B} \text{ DM} \qquad \frac{\sim\sim a}{a} \text{ DN}$$

Recommended Derived Rules

$$\frac{a \vee B}{\sim a \to B} \text{ AR}$$

Simplify the problem by getting rid of V. (Top-Down)

$$\frac{\sim a \to B \text{ GOAL}}{a \vee B} \text{ AR}$$

In case $a \vee B$ is a goal convert it to $\sim a \to B$ which is likely to be easier to obtain.

$$\frac{\sim a \to \sim B}{B \to a} \text{ CN}$$

You might also use other versions of CN But make sure that after you do CN There is some rule you can apply to the resulting conditional. DO NOT apply CN without a way to use the result.

How to Make a Tree

There is a major defect with the truth table method for determining whether an argument is valid. When there are many different letters in an argument, the number of rows in the truth table can be very large. Since the number of rows doubles with each added letter (for 2 letters - 4 rows, 3 letters - 8 rows, 4 letters - 16 rows, etc.) the table contains more than a thousand rows for an argument with only 10 letters. The tree method is a variation on the truth table method which can vastly shorten validity calculation. True, a tree may take as long as the corresponding table in the worst case, but the chances are very good that it will yield massive savings in effort.

The main idea behind the tree method is to avoid calculating any parts of the table which do not contribute to the main objective in the validity test, namely to find a row where all the premises are T and the conclusion is F. If there is such a row, the argument is invalid, because validity requires that it is impossible for (all) the premises to be T and the conclusion F. Let us call a row with (all) T premises and a F conclusion a counterexample. The idea behind tables is to examine all possible combinations of truth values to see whether there is a counterexample. If there is, the argument is invalid, and if not, it is valid. The tree method is more economical than tables because it calculates only those portions of the table where a counterexample can be. To help illustrate the point, consider this table for the following invalid argument: $Q\rightarrow\sim P \vdash P\rightarrow Q$

P	Q	$Q\rightarrow\sim P$	$\vdash P\rightarrow Q$	
T	T	F	T	
F	T	T	T	
T	F	T	F	← Counterexample
F	F	T	T	

Note that the third row is a counterexample to the argument - it shows that the argument is invalid because it is possible for the premise to be T and the conclusion F. If we had some way to know to calculate only this third row, and to ignore the other three, we would have the answer (invalid) with only 1/4 the effort.

So let us see whether we can find a method to locate rows where counterexamples will be. Suppose that the argument $Q\rightarrow\sim P \vdash P\rightarrow Q$ does have a counterexample. Then there must be a row where $Q\rightarrow\sim P$ is T and $P\rightarrow Q$ is F. But there is only one way for a conditional like $P\rightarrow Q$ to be F,

$$P \to \sim Q$$
$$\sim(Q \to \sim P)$$

Note that to indicate $Q \to \sim P = F$, we have written $\sim(Q \to \sim P)$ instead. Now if $\sim(Q \to \sim P)$ is to be T, i.e. $Q \to \sim P = F$, then there is only one possibility: $Q = T$ and $\sim P = F$. Entering this information using our convention, we have the following.

$$\begin{array}{rl} & P \to \sim Q \\ ① & \sim(Q \to \sim P) \\ & Q \\ & \sim\sim P \end{array}$$

The circled 1 is there to indicate that we have worked on the second line - the numbers will indicate the order in which we worked. We have carried out the tree rule for negative arrows. In general, the rule has the form:

$$\begin{array}{c} \sim(\mathcal{A} \to \mathcal{B}) \\ \mathcal{A} \\ \sim\mathcal{B} \end{array}$$

This indicates that whenever we have $\sim(\mathcal{A} \to \mathcal{B})$ in a tree, we should then write down $\mathcal{A}$ and $\sim\mathcal{B}$ below it. This principle is strongly related to the AR (Arrow) rule. You probably remember the following useful move: $\sim(\mathcal{A} \to \mathcal{B}) \vdash \mathcal{A} \& \sim\mathcal{B}$. Notice that the tree rule for a negative arrow amounts to almost the same thing, for from $\sim(\mathcal{A} \to \mathcal{B})$, we obtain $\mathcal{A} \& \sim\mathcal{B}$ by AR, from which $\mathcal{A}$ and $\sim\mathcal{B}$ follow by &Out.

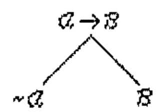
Notice that $\sim\sim P$ now appears on the tree. Using the double negation rule:

$$\begin{array}{c} \sim\sim\mathcal{A} \\ \mathcal{A} \end{array}$$

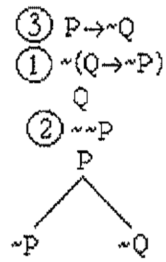
we may simplify this to P:

$$\begin{array}{rl} & P \to \sim Q \\ ① & \sim(Q \to \sim P) \\ & Q \\ ② & \sim\sim P \\ & P \end{array}$$

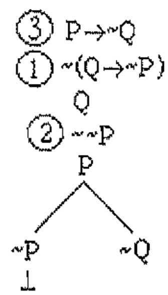
The next step in building our tree corresponds to checking to see whether $P \to \sim Q$ can be T. To do so, we must introduce the tree rule for positive conditionals. In general the rule has the following form:



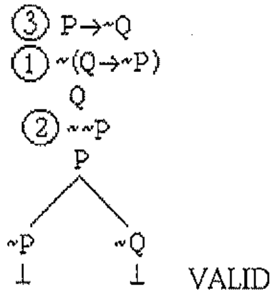
The branch in the rule indicates that for $\mathcal{A} \to \mathcal{B}$ to be true, there are two (and only two) possibilities: either $\mathcal{A}$ must be F or $\mathcal{B}$ must be T. Review the truth table for → to convince yourself that this is correct. Another way to understand (and remember) this rule is to think of it as a variation on the AR rule for converting from v to →: $\mathcal{A} \to \mathcal{B} \vdash \sim \mathcal{A} \vee \mathcal{B}$. When we apply this principle to our tree, we have the following:



The circled numbers show that we have worked on every complex line of our tree. It is now time to evaluate the argument. The tree shows that if there is to be a counterexample to our argument, there are two possibilities. The first, is recorded by the left branch, which contains the simple sentences Q, P and ~P. This indicates (according to our convention) that Q must be T (Q), P must be T (P) and P must be F (~P). But it is impossible for P to be both T and F on the same row, so this possibility can be ruled out. To indicate that this situation is impossible, we indicate that this branch is "dead" by adding the contradiction sign:



Notice that the right hand branch also contains a contradiction, for the counterexample it describes requires that Q be both F and T. So we mark that branch with ⊥ as well:



Let us review what this tree tells us about validity. It provides a complete record of all the possible ways in which a counterexample can be constructed for P→~Q ⊢ Q→~P. There were two options (described by the left and right branches), but both of these turned out to be impossible. The verdict: there cannot be a counterexample to this argument. To put it another way, the argument is valid.

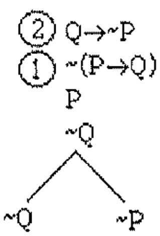
Now we will work out the tree for Q→~P ⊢ P→Q to see what happens with an invalid argument. We begin by writing down the premise and the <u>negative</u> of the conclusion:

$$\begin{array}{c} Q\to\sim P \\ \sim(P\to Q) \end{array}$$

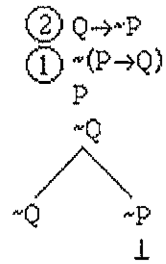
Applying the rule for negative conditionals we obtain:

$$\begin{array}{lc} & Q\to\sim P \\ ① & \sim(P\to Q) \\ & P \\ & \sim Q \end{array}$$

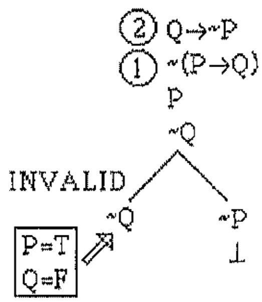
Now we apply the rule for positive conditionals to Q→~P:



Note that the right-hand branch contains a contradiction, so we mark it closed.



However, the left hand branch is still open. Since we have worked on every complex sentence, we have a full account of the possibilities for counterexamples for this argument. The left hand branch reports that there is a counterexample, namely one with P, ~Q, and ~Q all true. That amounts to saying that there is a counterexample with P = T and Q = F. We list this counterexample by putting the values in a box:



Here is a review of the tree construction process.

1. Write down the premises and the <u>negative</u> of the conclusion.
2. Apply tree rules to every complex sentence.
3. Mark every branch that contains a contradiction with ⊥.
4. If all branches are closed (contain ⊥), the argument is valid; if any branch is open (no ⊥ on it), then this branch describes a counterexample and the argument is invalid.

Let's review the process with a more complex argument: (P&Q)→R, P ⊢ Q→R. First enter the premises and the negative of the conclusion:

```
(P&Q)→R
 P
~(Q→R)
```

Now apply the negative conditional rule to the last line:

```
    (P&Q)→R
    P
(1) ~(Q→R)
    Q
    ~R
```

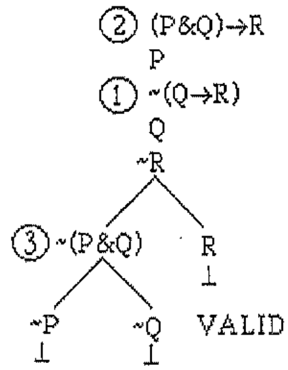
Now apply the positive conditional rule to the first premise:

```
        (2) (P&Q)→R
            P
        (1) ~(Q→R)
            Q
            ~R
           /  \
    ~(P&Q)      R
                ⊥
```

Notice that the right hand branch is closed because it contains ~R and R, so we have marked it with ⊥. Now we must apply a rule to ~(P&Q). The rule for negative conjunctions is similar to De Morgan's Law: ~($\mathcal{A}$&$\mathcal{B}$) ⊢ ~ $\mathcal{A}$ ∨~$\mathcal{B}$. If $\mathcal{A}$&$\mathcal{B}$ is F then either $\mathcal{A}$ is false or $\mathcal{B}$ is false:

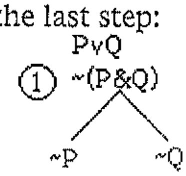
```
  ~(𝒜&ℬ)
   /  \
 ~𝒜    ~ℬ
```

Applying this rule to ~(P&Q), we obtain:

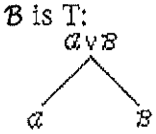


Since all branches close, the argument is valid.

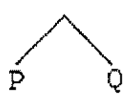
There are still a few details about the tree method that need explaining. To illustrate, we will work out the tree for PvQ ⊢ P&Q. We begin entering the premise and negative conclusion as usual, and we have already applied the negative & rule to the last step:



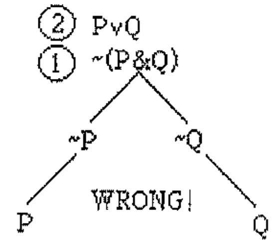
Now we must work on PvQ. It is easy enough to see what rule we want here. If $\mathcal{A}$v$\mathcal{B}$ is T then either $\mathcal{A}$ is T or $\mathcal{B}$ is T:



The problem is where to put the result of applying this rule. We have two open branches in our tree, one ending with ~P and the other ending with ~Q, and neither one is closed. So where do we place the new branch generated by PvQ;



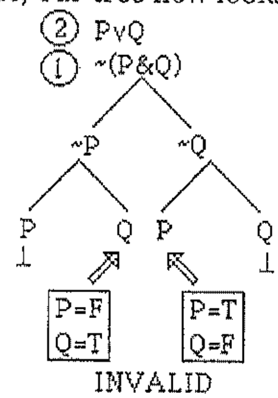
on the left (below ~P) or on the right (below ~Q)? It would be wrong to do either one, and it would also be wrong to divide the results of applying the rule between the two sides like this:



To see what we should do, it helps to remember what the tree represents. The two branches ending with ~P and ~Q represent two alternatives: for there to be a counterexample, either P = F or Q = F. However, the fact that PvQ is T means that <u>on top of</u> these two alternatives there are two more, either P = T or Q = T. So the correct thing to do is represent a total of <u>four</u> alternatives. Beneath the alternative ~P, we need to record the alternative P or Q, and beneath ~Q we must do so as well. This illustrates a basic principle about building a tree when there is more than one open branch.

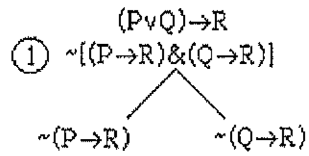
The results of applying a rule to a line must be entered on <u>every open branch</u> below that line.

Following this principle, our tree now looks like this:

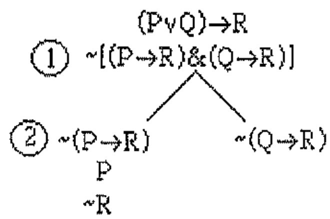


Note that contradictions appeared on two of the branches, so we marked them closed: Since we have worked on every step, the tree is complete and we can evaluate the argument. Note that there are two open branches indicating two different counterexamples to our argument. So the argument is invalid. It is a good idea to calculate the truth table for this argument to verify that it has these two counterexamples. One nice thing about trees is that they always give a full account of all the counterexamples to an argument.

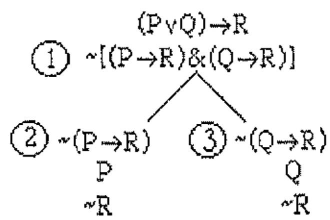
Our next project will be to show that (PvQ)→R ⊢ (P→R)&(Q→R) is valid using trees. We begin by negating the conclusion, and applying the negative & rule:



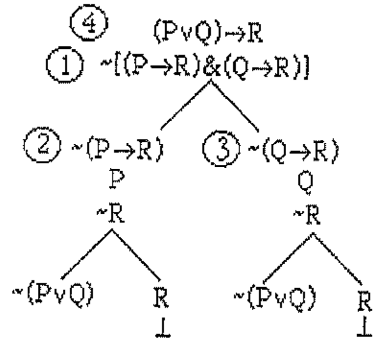
Next we will work on the ~(P→R) on the left hand branch. According to the negative → rule we enter P and ~R below ~(P→R). **It is not necessary to place P and ~R below the right hand branch.** Remember the rule is to put the results of applying a rule on every open branch <u>below the formula you are working on</u>.



After working on ~(Q→R) on the right branch, we obtain.



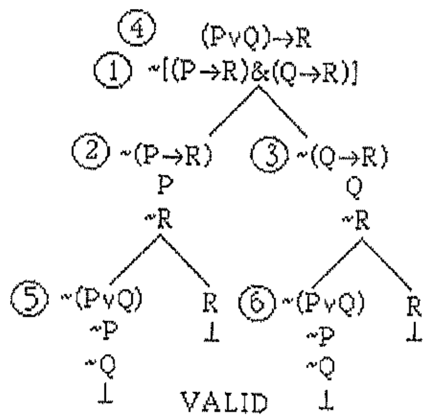
Now we will work on the first line: (PvQ)→R. In this case, we must place the result of applying the rule on all open branches below this line. So the result goes on both branches:



All that remains is to work on the ~(PvQ) on each branch. The rule for a negative v is as follows:

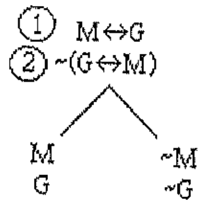
$$\sim(\mathcal{A} \vee \mathcal{B})$$
$$\sim\mathcal{A}$$
$$\sim\mathcal{B}$$

This rule is related to DeMorgan's Law. The idea is that when $\mathcal{A} \vee \mathcal{B}$ is F, then both $\mathcal{A}$ and $\mathcal{B}$ are F. Applying this rule on each branch we complete the tree:

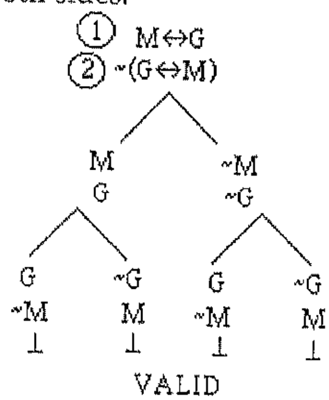


There is a final detail about tree construction worth mentioning. You may have been curious about the order in which I have carried out the steps in the trees. You might want to experiment with the problems we have already done to see what happens if you carry out the steps in a different order. In each case I have ordered the steps to keep the tree as simple as possible. There is a simple recipe to help economize in tree construction. The branching rules tend to create open branches, and subsequent steps must be copied onto all these open branches. For this reason, it is a good idea to **do the non-branching steps first.** Then when branching steps are finally carried out, there is a better chance that the branches created will close.

As a last note, we must discuss the ↔ rules. The rules may appear complex. But there is a simple way to remember them. In case $\mathcal{A}$↔ $\mathcal{B}$ is T, the values of $\mathcal{A}$ and $\mathcal{B}$ match. So either $\mathcal{A}$ and $\mathcal{B}$ are both T or $\mathcal{A}$ and $\mathcal{B}$ are both F. So the positive ↔ rule indicates these two possibilities on two branches. In case $\mathcal{A}$↔ $\mathcal{B}$ is F, then the values of $\mathcal{A}$ and $\mathcal{B}$ disagree. So either $\mathcal{A}$ is T and $\mathcal{B}$ is F or $\mathcal{A}$ is F and $\mathcal{B}$ is T. This is why the negative ↔ rule shows two alternatives, one containing $\mathcal{A}$ and ~$\mathcal{B}$ and the other containing ~$\mathcal{A}$ and $\mathcal{B}$. Here is a tree for the argument M↔ G ⊢ G↔ M to help you practice these rules.

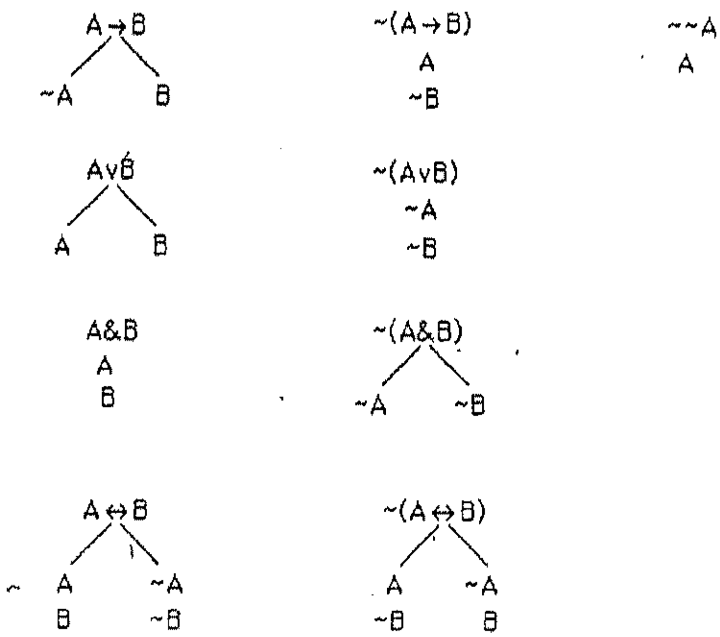


Since there are two open branches, the result of applying the negative ↔ rule to ~(M↔ G) goes on both sides:

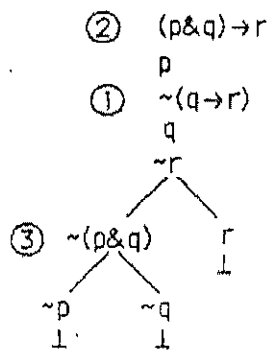


Note that all branches are closed. It is not necessary for there to be a contradiction for M and a contradiction for G to close a branch. **One contradiction on a branch is enough to close it.**

Tree Rules for Propositional Logic



Example: To show the argument (p&q)→r, p ⊢ q→r is valid, negate the conclusion, and then show that there is no assignment that makes the premises and the negated conclusion all T.

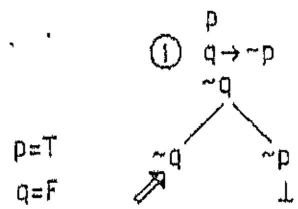


EXERCISES.

1) p→(q→r) ⊢ q→(p→r)
2) p→(p→q) ⊢ p→q
3) p ⊢ qv(r→p)
4) p ⊢ (p→r)↔r
5) pvq, p→r, q→s ⊢ rvs

Finding Counterexamples with Trees

To find a counterexample to this invalid argument: $p, q \to \sim p \vdash q$
first build its tree:

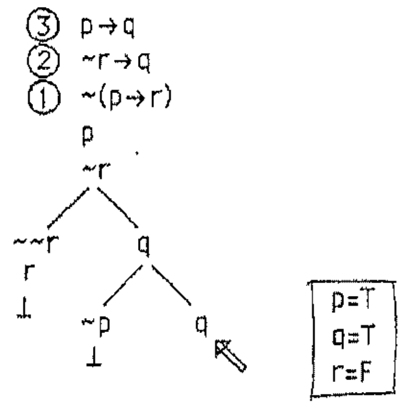


Definition

A *counterexample* is an assignment of truth values to the letters of an argument that makes all premises T and the conclusion F.

The left branch is open (marked with ⇗). Now construct the open branch assignment, by assigning T to all letters, that appear unnegated in the open branch, and F to those that appear negated. In this case we have p=T and q=F. Calculate the values of the sentences in this argument, and you will find that the premises are T and the conclusion is F in this case.

Here is tree for the argument $p \to q, \sim r \to q \vdash p \to r$.



Here is your counterexample for the open branch.

The open branch assignment is p=T, q=T, r=F. Calculate the values of the sentences in this argument to show that the premises are T and the conclusion is F.

EXERCISES.
1) $p \to q \vdash q \to p$
2) $p \to q, p \to r \vdash \sim r \to q$
3) $p \to q \vdash (p \vee r) \to q$
4) $(p \& r) \to q \vdash p \to q$

TOS' to	Logical Truth	Contradiction	Co ngent	a entails B	a equivalent to B	[illegible] consistent
T. Table	Table for a has all T a: T T T ⋮	Table for a has all F a: F F F ⋮	Table for a has an F and a T a: T ⋮ F ⋮	Table for a, B has no T, F row (Argument a ⊢ B is valid)	Tables for a and B are the same a: T F T T ⋮ / B: T F T T ⋮ (MATCH)	Table for a, B has a T, T row a B: T T
Tree	Tree for ~a closes ~a ⊥⊥⊥⊥	Tree for a closes a ⊥⊥⊥	Tree for a open and Tree for ~a open a ~a	Tree from a, ~B closes a ~B ⊥⊥⊥⊥	Trees for both a, ~B and B, ~a close a ~B ⊥⊥⊥⊥ / B ~a ⊥⊥⊥	Tree for a, B is open a B
Proof	Prove a Here's how it looks: ⋮ a Goal	Prove ~a Here's how it looks: ⋮ ~a Goal	No TEST	Assume a, prove B Here's how it looks: a A ⋮ B Goal	Assume a, prove B then assume B, and prove A. Here's how it looks: a A ⋮ B Goal B A ⋮ a Goal	No TEST

Predicate Logic Translation Chart

Singular:

(Use a lower case letter s for the subject and a upper case letter P for the predicate and write: Ps.)

Examples: John, David's cat, he*, she*, it*, that*, the king*

* on occasion, these may be general

General:

A. $\forall x(Ax \rightarrow Bx)$

All As are Bs
Every A is a B
As are Bs
Any A is a B
Each A is a B
An A is a B*
Only Bs are As
None but Bs are As

* Usually 'a' means some, but it may mean all. It is NEVER singular.

E. $\sim\exists x(Ax \& Bx)$ or $\forall x(Ax \rightarrow \sim Bx)$

No A is a B
None of the As are Bs
Not any A is a B

I. $\exists x(Ax \& Bx)$

Some As are Bs
Some A is a B
There are As that are Bs
A least one A is a B
As are sometimes Bs
An A is a B*
Many A are B
There exist As that are Bs

O. $\exists x(Ax \& \sim Bx)$ or $\sim\forall x(Ax \rightarrow Bx)$

Some As are not Bs
Some As are non-Bs
Not all As are Bs

Trees for Predicate Logic

It is a simple matter to combine the rules you learned for the quantifiers with the (single sided) tree rules to create a method for testing for validity in predicate logic. The tree rules for the quantifiers are exactly the same as those you learned for proofs, namely the following:

$\forall$Out	$\exists$Out	QE	
$\underline{\forall xAx}$	$\underline{\exists xAx}$	$\underline{-\forall xAx}$	$\underline{-\exists xAx}$
An	An	$\exists x\text{-}Ax$	$\forall x\text{-}Ax$
	n is new		

Here is a tree that shows that the translation of the first argument presented in this class is valid, namely the UH students are Nazis Argument: $\forall x(Ux \rightarrow Nx), \forall x(Nx \rightarrow Bx) \vdash \forall x(Ux \rightarrow Bx)$. We start by negating the conclusion.

$\forall x(Ux \rightarrow Nx)$
$\forall x(Nx \rightarrow Bx)$
$-\forall x(Ux \rightarrow Bx)$

Then we apply the predicate logic rules in the familiar order, first we do QE.

$\forall x(Ux \rightarrow Nx)$
$\forall x(Nx \rightarrow Bx)$
(1) $-\forall x(Ux \rightarrow Bx)$
$\exists x\text{-}(Ux \rightarrow Bx)$

Then $\exists$Out:

$\forall x(Ux \rightarrow Nx)$
$\forall x(Nx \rightarrow Bx)$
(1) $-\forall x(Ux \rightarrow Bx)$
(2) $\exists x\text{-}(Ux \rightarrow Bx)$
$-(Ua \rightarrow Ba)$ a is new

Note that we do $\exists$Out first so that we can avoid violating the restriction that the name must be new. Now we do $\forall$Out on the top two steps.

```
(3)   ∀x(Ux→Nx)
(4)   ∀x(Nx→Bx)
(1)   -∀x(Ux→Bx)
(2)   ∃x-(Ux→Bx)
      -(Ua→Ba)
      Ua→Na
      Na→Ba
```

All the quantifier steps are now done, so all that remains is to apply the propositional logic tree rules to the last three lines.

```
(3)   ∀x(Ux→Nx)
(4)   ∀x(Nx→Bx)
(1)   -∀x(Ux→Bx)
(2)   ∃x-(Ux→Bx)
(5)   -(Ua→Ba)
(6)   Ua→Na
(7)   Na→Ba
      Ua
      -Ba
      / \
  -Ua    Na
   *     / \
      -Na   Ba
       *    *
```

Since all branches are closed, the argument is valid, which is what we claimed in the first week of class.

We also made the point then that the Canaries are Pets Argument was invalid. Here is a tree that verifies that. The argument is ∀x(Cx→Fx), ∃x(Px&Fx) ⊢ ∃x(Px&Cx). We negate the conclusion and apply QE to the last step:

```
      ∀x(Cx→Fx)
      ∃x(Px&Fx)
(1)   -∃x(Px&Cx)
      ∀x-(Px&Cx)
```

Now we take off the quantifiers making sure to do ∃Out first so as not to violate the restriction on that rule:

```
(3)   ∀x(Cx→Fx)
(2)   ∃x(Px&Fx)
(1)   -∃x(Px&Cx)
(4)   ∀x-(Px&Cx)
      Pa&Fa
      Ca→Fa
      -(Pa&Ca)
```

Now we apply the propositional rules to the remaining three steps:

```
(3)   ∀x(Cx→Fx)
(2)   ∃x(Px&Fx)
(1)   -∃x(Px&Cx)
(4)   ∀x-(Px&Cx)
(5)   Pa&Fa
(7)   Ca→Fa
(6)   -(Pa&Ca)
      Pa
      Fa
      / \
  -Pa     -Ca
   *      / \
       -Ca   Fa
        o     o
```

This time we got open branches on the right, so the argument is invalid as we claimed. (One open branch would be enough to show invalidity.)