ARGUE: A Manual and Concise Text to Accompany the ARGUE Program for Formal Deduction and Analytic Inquiry

Preston K. Covey
Carnegie Mellon University, dtrollcovey@gmail.com

Follow this and additional works at: http://repository.cmu.edu/philosophy
Part of the Philosophy Commons
ARGUE

A Manual and Concise Text to Accompany the ARGUE Program for
Formal Deduction and Analytic Inquiry

Preston K. Covey, Jr.
Carnegie-Mellon University

(C) Copyright Preston K. Covey, Jr. 1984
# Table of Contents

Preface 3

1. ARGUE's BASIC OPERATIONS 5
   1.1. ARGUE's Formal Functions 5
   1.2. ARGUE's Basic Options 7
   1.3. Running ARGUE 7
   1.4. The INS Command: ARGUE's List of Commands 8

2. Requesting a Problem: the SET and PROblem Commands 9

3. Working a Problem: Key Commands 11
   3.1. The Slash Command '/ ': Justifying the Lines of a Derivation 11
   3.2. The Slash Command '/ ': Commanding ARGUE to Type a Line of Derivation 13
   3.3. The TEXT, ALIST, and ENGLISH Commands 16
   3.4. An Extended Example: The HINT and LIST Commands 18

4. Constructing Your Own Arguments Or Derivations: The ASSIGN Command 24
   4.1. "Enter a Conclusion to Derive. . ." 25
   4.2. "... Or Type 'BEGIN' " 27

Appendix I. What the ARGUE Program Can Help Teach 31

Formal Logic in the Service of Argument Analysis 33
   1.1. Four Tasks of Formal Argument Analysis 33
   1.2. A Sample Argument: Preferential Hiring 41
   1.3. An Initial Reconstruction 42
   1.4. The First Problem: Invalidity and Unstated Premises 43
   1.5. Further Problems: Ambiguity and Vulnerability to Counter Example 46
   1.6. The Game of Formal Argument Reconstruction and Counter Example 50

Appendix II. ARGUE's Rules of Inference and Replacement 55
   II.1. Sentential Rules of Inference 56
   II.2. Sentential Rules of Replacement 57
   II.3. Conditional Proof & Indirect Proof Strategy 58
   II.4. Quantificational Rule Schemas & Summary 59
   II.5. Restrictions on Universal Generalization (UG) 60
   II.6. Restrictions on Existential Instantiation (EI) 61
Preface

The ANALYTICS package of computer-assisted instruction programs for IBM or Apple micro-computers presently consists of the following three programs: SYMBOL, which generates symbolization exercises (with answers) in sentential logic; TRUTH, which generates exercises (and guidance) in the truth-functional analysis of sentential connectives; and ARGUE, which provides guidance in formal derivations and the reconstruction of English arguments in valid deductive form. Each of these programs provides guided practice with tools and techniques of formal logic that are essential not only to learning the apparatus of elementary symbolic logic but also to understanding the formal dimensions of argument reconstruction and analysis. The Appendix of this manual provides a conceptual framework for understanding the curricula of the programs and the applications of what they help teach.

A word of explanation is in order regarding the title of the set of computer-based tutorials, ANALYTICS, for which this manual is a companion. The computer tutorials are designed primarily to exercise you in the use of basic tools and techniques of formal logical analysis. The science of formal logic and the systematic study of what makes reasoning good or bad were invented by Aristotle over twenty centuries ago. Modern logic has made considerable progress on Aristotle's monumental beginnings. But the nature of logical analysis as we pursue it today still owes a great deal to Aristotle's original inspiration and definition. Any history of philosophy or logic will attest to this. The following brief description of Aristotle's basic enterprise from W. T. Jones' History of Western Philosophy (Vol. I, p. 224) is fitting for our course of study:

Aristotle was the inventor of formal logic in the sense that he was the first person to draw up precise rules for distinguishing valid from invalid thinking. Suppose I know that all Greeks are mortal and that Aristotle is a Greek. It follows that Aristotle is mortal. . . . [Now], why would the conclusion that Aristotle is a man not follow from these premises? Of course there are premises from which the latter conclusion could be drawn—for instance, "All Greeks are men" and "Aristotle is a Greek." But even though it is true that Aristotle is a man, this proposition does not follow from the facts that he is a Greek and that all Greeks are mortal.

Thus, as Aristotle saw, we must distinguish between truth and validity. Truth is a characteristic of individual propositions: An individual proposition is true if it correctly classifies things and false if it does not. Thus "Aristotle is a Greek" is true, and "Aristotle is a Turk" is false. Validity is not a characteristic of individual propositions. It is the logical relation between premises . . . and the conclusion that follows from these premises. Thus, although the proposition "Aristotle is a man" is true, it
follows validly from some premises but not from others.

The two chief questions Aristotle set himself to answer . . . were: (1) When we have true propositions, what are the rules of inference by which a conclusion can be validly drawn? (2) How can we know that the premises we start with are true?

In formal logic today we still distinguish (1) the logical FORM and VALIDITY of an argument from (2) the CONTENT or TRUTH of its premises and conclusion. We, like Aristotle, are interested in both of the following questions:

1. What are the rules of inference by which conclusions can be validly drawn from given premises?

2. How can we determine whether the premises of an argument are true or sufficiently plausible to justify assent?

We will also be concerned with how these two questions are inevitably and usefully related: with how valid logical form is usefully related to the pursuit of truth (in philosophy, in particular).

This concern with the distinction and relation between the FORM and CONTENT of reasoning remains very much in the spirit of Aristotle's pioneering study of logic. This interaction is reflected throughout the computer programs in logic and argument analysis. This is why the package of programs is entitled ANALYTICS, reminescent of the title of Aristotle's major treatises on logical analysis, the Prior and Posterior Analytics.

As you probably know and as you will see, you can't argue well about the truth of anything without logic; but logic, while necessary, is hardly sufficient for determining the truth in any dispute. The relation between formal logic and the pursuit of truth in ANALYTICS becomes especially conspicuous in the formal analysis and reconstruction of arguments. For this job we need some special tools. The programs in the ANALYTICS package provide practice with some of the most basic tools of formal logic needed to analyze the elementary logical structure of arguments, to assess their validity, and to reconstruct them in expressly valid form.
1. ARGUE's BASIC OPERATIONS

1.1. ARGUE's Formal Functions

ARGUE is designed to give you guidance in constructing derivations in symbolic notation, analyzing the logical form of English arguments, and constructing arguments in valid logical form.

Practice in doing derivations with ARGUE will exercise your sense of logical form and reinforce your judgments about validity within the very precise constraints of symbolic logic. See Appendix I of your text/manual for illustration of the role formal logic can play in the analysis and reconstruction of arguments in philosophic inquiry.

ARGUE enforces formal logical constraints on the reconstruction of English arguments. It's stored problems present you with English arguments which you must symbolize and prove valid by deriving the conclusion from the stated premises (plus any you need to provide to make the argument valid). ARGUE can also be used simply to present and monitor formal derivation problems in first-order logic.

ARGUE contains a parser and proof-checker that check each line of any derivation you construct for the following crucial logical properties:

1. Well-Formedness: ARGUE will check any symbolic formula you enter in a derivation to see that it is a well-formed formula (WFF). It will give you an error message when any formula is not well formed.

2. Validity: ARGUE will check any line you enter in a derivation to see that it is a valid move according to our given rules of inference and replacement. (See Appendix for summary of given rules.) When you enter any line in a derivation, you must justify it by citing the previous line(s) of the derivation and/or the rule that allows you to derive or enter the new line. ARGUE will check your justification to see:

   Whether you have cited correct or sufficient line numbers to justify the new line.

   Whether you have cited the correct rule to justify the new line.

If you fail to justify any line in a derivation correctly, or if the line you enter is not allowed or valid by the cited rule, ARGUE will give you an error message. ARGUE
thus prompts you to detect and correct your errors immediately.
1.2. ARGUE's Basic Options

ARGUE allows you three different options for working derivations, constructing or reconstructing arguments in valid form:

1. 'Request a Problem': You can work on stored problems from ARGUE's problem sets. These problems are supplied with stored hints and solutions that you can see by using the HINT command. (The present program is supplied with one sample problem set, Set 1, whose problems require only sentential logic. The quantifier rules are not plugged into the proof-checker of this initial version of the program, although it is able to 'read' quantified formulae. Some of the sample problems illustrated in this manual are taken from problem sets not provided in the present package and employ quantificational logic. More extensive problem sets are available from the authors.) When you have successfully derived the conclusion assigned in a given problem, ARGUE will congratulate you (as illustrated in Section 3 below).

2. 'Enter a Conclusion to Derive': You can enter your own conclusion (in symbolic notation) to derive, and thus create your own derivation problems to work on. You can also assign English translations to your logical constants in order to get ARGUE to provide English translations of any arguments you construct (as illustrated in Section 4 below). When you have successfully derived your stated conclusion in valid fashion, ARGUE will congratulate you (as illustrated in Sections 3 and 4 below).

3. 'Or Type 'Begin": You can simply begin a derivation without specifying any target conclusion that you want to derive. ARGUE will allow you to proceed as you will, checking each step for validity. In this 'free-form' mode ARGUE doesn't know what in particular you want to derive, because you have not specified any particular conclusion; so it will not congratulate you at any point in your derivation. It will simply prompt you for successive steps in your derivation until you request a stored problem, ask to start another line of derivation or stop. Section 4 below illustrates this 'free-form' option in ARGUE.

These three options are best illustrated with some sample interactions. Let's take a look at what it's like to run and work with ARGUE.

1.3. Running ARGUE

You run ARGUE by inserting the diskette in the disk drive (usually drive #1) and booting it (usually by turning the computer on). Should you exit ARGUE for any reason, you can re-enter by typing:

ARGUE <RET>

The abbreviation <RET> indicates that you must hit the RETURN key after entering the command or other input.
1.4. The INS Command: ARGUE's List of Commands

When you boot the disk or run ARGUE, the program will be loaded (this takes several seconds) and will announce itself. It will then print out the following invitation:

IF YOU NEED INSTRUCTIONS, TYPE 'INS'; OTHERWISE
ENTER A CONCLUSION TO DERIVE, REQUEST A PROBLEM, OR TYPE 'BEGIN'

In this manual, I use the boldfaced line _ (above) to indicate the flashing cursor that serves as ARGUE's prompt when it's waiting for a command or response from you. At this point (or at any point where ARGUE is prompting you for a response), you may ask for a list of ARGUE's commands by typing INS as follows. (Remember to hit RETURN after you've typed your command or response; you're reminded of this maneuver in the sample interactions below by <RET>. If you make an error in typing, you can backspace and delete by using the left-pointing arrow or the DELete key on your keyboard and then re-typing your input.)

IF YOU NEED INSTRUCTIONS, TYPE 'INS'; OTHERWISE
ENTER A CONCLUSION TO DERIVE, REQUEST A PROBLEM, OR TYPE 'BEGIN'
_ INS <RET>

ARGUE is a computer program that will guide you in constructing formal derivations or reconstructing arguments in valid deductive form.

ARGUE accepts the following commands:

SET N OPENS PROBLEM SET N
PROBLEM N ASKS ARGUE TO GIVE YOU PROBLEM NUMBER N
HINT REQUESTS A HINT ABOUT SOLVING THE PROBLEM
NEW ALLOWS YOU TO BEGIN ANOTHER PROBLEM OR DERIVATION
TEXT PRINTS THE TEXT OF THE CURRENT PROBLEM
LIST LISTS YOUR DERIVATION (TO GET A CLEAN COPY)
LIST M-N LISTS LINES M THROUGH N OF YOUR DERIVATION
ENG PRINTS AN ENGLISH TRANSLATION OF YOUR DERIVATION
ENG M-N PRINTS TRANSLATION OF LINES M THROUGH N
ALIST LISTS ENGLISH ASSIGNMENTS TO CONSTANTS
ASSIGN ALLOWS YOU TO ASSIGN TRANSLATIONS TO NEW CONSTANTS
DELETE N ERASES YOUR DERIVATION FROM LINE N TO THE LAST LINE
REMOVE N ERASES LINE N ONLY (DERIVATION IS RENUMBERED; USE LIST)
RULES PRINTS RULES OF INFERENCE AND REPLACEMENT
STOP STOPS/EXITS THE PROGRAM
INS PRINTS THIS LIST OF COMMANDS
/ TYPE THE SLASH '/' TO JUSTIFY EACH LINE IN YOUR DERIVATION, FOLLOWED BY THE RULE AND LINE NUMBERS APPEALED TO

ENTER A CONCLUSION TO DERIVE, REQUEST A PROBLEM, OR TYPE 'BEGIN'
You can use any of these commands at any point in a problem. You can use most of them at any point in the program at all. (An obvious exception is that you can't get a hint when you're not actually working a stored problem.) It's useful to keep in mind that you can ask for a different problem or set in the midst of working any problem, or exit that problem by typing 'NEW.' The first three letters of any command will suffice. You may type any command in any combination of upper and lower case. (CTRL-A acts as a CAPS-lock toggle switch with the M&R SUP'R TERM board.)

The interaction above illustrates (1) the ARGUE program's opening response, (2) the program's prompt, the flashing cursor _, in response to which you would type a command (or a line in a derivation), (3) the commands available in ARGUE (printed in response to the INS command), and (4) the three options allowed you for working in ARGUE: 'Enter a conclusion to derive, request a problem, or type 'Begin'. I will illustrate each of these options in the sections that follow, and highlight the commands you should be familiar with at each stage. I will begin with the middle option, requesting a problem, since it is the one you are most apt to elect in your initial work with ARGUE.

2. Requesting a Problem: the SET and PROblem Commands

When the ARGUE program prompts you to select one of its three options for practicing derivations and you wish to work on some stored problem(s), you need first to tell ARGUE which of its problem sets to open by using the SET command.

Only one problem set (Set 1) is supplied with the present program. Several problem sets would normally be available. These might contain only derivation problems in symbolic notation (like the examples in this section), logical 'word problems' (similar to algebra word problems) requiring symbolization and derivation for their solution, or they may contain English arguments to be reconstructed and proven valid by derivation. (Examples of the latter, in both sentential and quantificational logic, are illustrated in other sections of this manual.)

You request a problem set N in response to ARGUE's prompt _, by typing the command SET N, where in place of the N you would type the number of the set you want. To get ARGUE to open problem Set 1, you would type, in response to the prompt _ : SET 1. (You may type any commands or other input to ARGUE in either lower or upper case, in any combination.)
ARGUE will tell you how many problems are contained in the requested set and then again give you its three basic options: 'Enter a conclusion to derive, request a problem, or type 'Begin'. After opening a given set N, you may pursue any of these options and whenever you request a problem by number, say, problem M, ARGUE will give you problem M from set N. Whenever you request a problem, ARGUE will assume that you want a problem from the problem set last opened. (You can open a different problem set by using the SET command at any time that ARGUE has prompted you with the flashing cursor _, even in the middle of doing a given problem.)

To request a given problem, once you have opened a problem set, you use the Problem command. Notice (in the illustration that follows) that the Problem command can be abbreviated to Pro. For example, to request the first problem in a set you could type, in response to ARGUE's prompt ': either Problem 1, or Prob 1, or Pro 1. In general, ARGUE's commands (and rule names) can be abbreviated to their first three or four letters (whatever is sufficient to unambiguously distinguish that command or rule from others).

In the illustrative interaction that follows, I will request a problem (without first opening a problem set, to show you how ARGUE responds), then open a problem set, request a problem, and (in the middle of the problem) request that ARGUE open a different problem set and give me a problem from that set -- all to show you how easily you can flip in and out of different problem sets with the SET command.

(Again, what I type -- or what you would type -- is represented in boldface. ARGUE's responses are in normal typeface. Commentary which is not part of the input/output will be between square brackets [...]. I will often dispense with the <RET> to remind you to hit the Return bar after typing your input in the illustrations provided below.)

ENTER A CONCLUSION TO DERIVE, REQUEST A PROBLEM, OR TYPE 'BEGIN'
_ Pro 4 <RET>

YOU MUST FIRST OPEN A PROBLEM SET WITH THE 'SET' COMMAND

ENTER A CONCLUSION TO DERIVE, REQUEST A PROBLEM, OR TYPE 'BEGIN'
_ Set 2
THERE ARE 13 PROBLEMS IN THIS SET

ENTER A CONCLUSION TO DERIVE, REQUEST A PROBLEM, OR TYPE 'BEGIN'
_ Pro 4
DERIVE: \(-G\)

1  \(F \Rightarrow -G\)  / PREMISE
2  \(-L \Rightarrow -F\)  / PREMISE
3  \(G \Rightarrow F\)  / PREMISE
4  \(-L\)  / PREMISE
5  _ Set 3 <RET>

[Notice that after setting up the problem by giving me a conclusion to derive and four premises from which to derive it, ARGUE then prompts me for in-put with the flashing cursor _ at line 5.

At this point I have elected not to do the problem but rather to open Problem Set 3. ARGUE then responds as follows:]

THERE ARE 15 PROBLEMS IN THIS SET

ENTER A CONCLUSION TO DERIVE, REQUEST A PROBLEM, OR TYPE 'BEGIN'
_ Pro 2 <RET>

DERIVE: \(-(L \& K)\)

1  \(-L V -K\)  / PREMISE
2  _ Set 1 <RET>

THERE ARE 26 PROBLEMS IN THIS SET

ENTER A CONCLUSION TO DERIVE, REQUEST A PROBLEM, OR TYPE 'BEGIN'
_ Pro 3 <RET>

DERIVE: \(Q \& P\)

1  \(P \& Q\)  / PREMISE
2  _

3. Working a Problem: Key Commands

The illustration above shows you that you can request a new problem or problem set at any point while working a given problem where ARGUE has prompted you for in-put with the flashing cursor _. The illustrations below show you the effects of certain key commands in the course of actually working through a problem.

3.1. The Slash Command '/': Justifying the Lines of a Derivation

Whenever you enter a symbolic formula on a line of a derivation, you must give a justification for that line by citing the rule that warrants the line and the numbers of any previous line(s) to which you appeal.
You indicate to ARGUE that you are going to give a justification for a line of derivation that you have just entered by typing the slash / after entering your formula followed by the appropriate rule and line number(s). For example, given the problem below, you could proceed as follows (where what you type is in boldface):

**DERIVE: P & R**

1. \( P \land Q \) / PREMISE
2. \( S \land R \) / PREMISE
3. \( P \) / SIMPL 1
4. \( R \) / 2 SIMPR
5. \( P \land R \) / 3 4 CONJ

**CONGRATULATIONS! YOUR DERIVATION IS COMPLETE.**

**LIST YOUR DERIVATION, ASK FOR A PROBLEM, OR TYPE 'NEW'.**

When you have succeeded in deriving the assigned conclusion, ARGUE will congratulate you (as above) and give you the options of getting a clean listing of your derivation, requesting another problem, or starting anew. Typing the **NEW** command at this juncture, or at any point where ARGUE prompts you with the flashing cursor _, will simply get you ARGUE's original three basic options: Enter a conclusion to derive, request a problem, or type 'Begin'. More on LISTing your derivation later. For now you should take note of the following:

1. Whenever you enter a formula, like \( P \), on a line of derivation, like line 3 above, you must also enter a justification preceded by the slash / on that line.

2. The justification of a line of derivation, following the slash /, consists of a rule citation and, where previous lines are appealed to, line numbers.

3. Some rules, like the **PREMISE** introduction rule, do not require the citation of previous lines because they allow the introduction of lines that do not depend logically on previous lines. Notice that in introducing the two premises in the problem above, ARGUE cited only the **PREMISE** rule preceded by the slash / . The other rules that do not require the citation of previous lines are **Excluded Middle Introduction (EMI)**, **Hypothesis (HYP)**, and **Identity Introduction (II)**. When you come across these rules in your logic text, it will be clear why they allow you to introduce lines in a derivation that are not logically dependent on previous lines.

4. You may cite the rule either before the line numbers or vice versa (compare lines 3 and 4 in the derivation above) -- the order does not matter.
5. You need not put any spaces between any of the entries in a line of derivation except between line numbers (see line 5 above) . . .

6. You must put either a space or a comma between line numbers when more than one line number is cited in a justification: For example, on line 5 above, had we typed 34 instead of 3 4, ARGUE would have thought we were referring to line 34 rather than lines 3 and 4 (and would have given us an error message to the effect that the line number was too large and too few lines were appealed to for the CONJunction rule).

7. It does not matter how many spaces you put between entries in a line of derivation, so long as you put no spaces in the name or abbreviation of the cited rule.

8. You may use abbreviations for the rules of inference and replacement. (The first three or four letters of the rule name or the initials of the rule will do; legal abbreviations are given on your rule summaries (in Appendix) and in ARGUE's rule file, which you can have ARGUE print out by typing RULES in response to the prompt '__').

9. You may type your input to ARGUE in any combination of upper and lower case.

10. You must put the slash / between the formula you enter as a line of derivation and its justification — unless you use the slash command to have ARGUE type the line for you (see the next section below).

3.2. The Slash Command '/ ': Commanding ARGUE to Type a Line of Derivation

It is often possible to get ARGUE to type a desired line of a derivation for you (and thereby save yourself both effort and the risk of making a typographical error) by using the slash / as a command. When prompted by ARGUE's prompt _ to enter a line in a derivation, you might type the slash / plus the rule you want to apply plus the line numbers you want the rule applied to. Then hit the carriage return. This will have the effect of telling ARGUE: 'Give me what I can get by the cited rule from the cited lines.' If, and only if, there is a valid consequence that can be directly derived (in one step) from the cited lines by the rule you cite, then ARGUE will print out that result for you with its justification. If there is no valid consequence that can be obtained by the cited rule from the cited lines in one step, ARGUE will of course give you an appropriate error message to that effect. Were we to use the slash command in this way on the problem in the previous illustration, the results would look like this (where what we type is printed in boldface):

DERIVE: P & R
When the formulae that you want to derive are longer and more complex than the
above, this feature of ARGUE's slash command is especially helpful. In using this
convenience, however, you should note carefully the following:

1. The slash command cannot be used with all rules to get ARGUE to print
the line you want. It can only be used when the result of applying the
cited rule to the cited lines would be unique or unambiguous. For
example, were you to type \texttt{/ PREMISE} in response to ARGUE's prompt \_
, ARGUE could not print out a premise for you because it could not
possibly know which of the infinite possible premises you might want.

2. In certain cases where the application of a rule to certain lines might
allow more than one valid result, the ambiguity can be resolved by
specifying an appropriate rule variation. Certain rules (like SIMPlification)
that warrant some basic logical move (like deriving one conjunct from a
conjunction) but that could have more than one result (like the derivation
of either the left or the right conjunct), have different variations to get
different results. Look at lines 3 and 4 in the illustration above: the rule
specification SIMPL gave us the left conjunct from line 1; the rule
specification SIMPR gave us the right conjunct from line 2. The
SIMPlification rule thus has two specifications to allow unambiguous
application of the slash command. The case is similar with DSL and DSR,
CDI and CDII (see your rule summary, in your logic text or Appendix to
the ARGUE manual).

3. In cases where a replacement rule could be applied to more than one
connective in a formula, ARGUE will always apply the rule to the major
connective. Thus, for example, if you use the slash command to apply
the COMmutation rule to the formula \( P \lor (Q \land R) \), ARGUE will commute
the disjunctive part of the formula (since the \( \lor \) is the major connective)
as follows: \( (Q \land R) \lor P \).

4. When using the slash command to get ARGUE to type formulae for you
and when there is a choice about the order in which the components (say, the conjuncts in a conjunction) occur, you should type the line numbers of the components (conjuncts) in the order in which you want them combined (conjoined). Look at line 5 in the illustration above. I typed / 3 4 CONJ and ARGUE formed the resultant conjunction with the conjunct from line 3 first and the conjunct from line 4 second. Had I typed / 4 3 CONJ ARGUE would have formed the conjunction in the reverse order: R & P rather than P & R.

ARGUE's rule set is given in the Appendix. Special ARGUE commands that allow work with English argument texts are given in the following section below.
3.3. The TEXT, ALIST, and ENGLISH Commands

When you are working on a stored problem that presents an English argument for reconstruction, at any point in your work you may review (1) the original text of the argument and (2) the English assignments for the logical constants used to symbolize the argument. Or, you may wish to get a logically perspicuous translation of the symbolized steps of the argument as far as you’ve worked.

The following interaction illustrates the use of the TEXT, ALIST, and ENGLISH commands for these purposes. In the problem below, an argument is given, the task is posed, and (in this case) the conclusion and stated premises are already symbolized and set up for us. The following commands are variously useful for reminding yourself of what the argument you’re reconstructing is all about; this is essential when your task is to supply an unstated/missing premise.

Persons have a right to life, allright! And it's morally wrong to kill what's got a right to life. So, it's certainly morally wrong to kill a human fetus!

Supply the UNSTATED GENERAL PREMISE required to make this argument valid and derive the conclusion.

DERIVE: (x)(Fx => Mx)

1  (x)(Px => Rx) / PREMISE
2  (x)(Rx => Mx) / PREMISE
3 _ TEXT <RET>

Persons have a right to life, allright! And it's morally wrong to kill what's got a right to life. So, it's certainly morally wrong to kill a human fetus!

3 _ ALIST <RET>

Px: x IS A PERSON
Rx: x HAS A RIGHT TO LIFE
Mx: IT'S MORALLY WRONG TO KILL x
Fx: x IS A HUMAN FETUS

3 _ ENG <RET>

DERIVE: (x)(Fx => Mx)

FOR ALL x, IF x IS A HUMAN FETUS THEN IT'S MORALLY WRONG TO KILL x
1. \((x)(Px \Rightarrow Rx)\)  
   For all \(x\), if \(x\) is a person then \(x\) has a right to life  

2. \((x)(Rx \Rightarrow Mx)\)  
   For all \(x\), if \(x\) has a right to life then it's morally wrong to kill \(x\)
3.4. An Extended Example: The HINT and LIST Commands

The HINT Command

Stored problems can be provided with hints that give you advice about the tasks of symbolization, reconstruction, and derivation. Each time you type HINT, ARGUE will give you the next available hint in sequence. (ARGUE will tell you how many hints remain.) You can also request hints by number (if you keep track and wish to review a previous hint) by typing HINT N (where N is the number of the hint that you want). You are well advised to execute each hint at the time that you get it. In the illustration below, however, I will ask for all the hints so you can see how they form a continuous discussion of how to solve the problem.

The LIST Command

In doing any problem you may find that the screen becomes cluttered with hints or other distracting matter. You can get a clean listing of your derivation as far as you've gone at any point by typing LIST. You can get a selective listing of certain lines M through N of your derivation by typing the line numbers after list separated by a dash: LIST M-N. (ENGLISH works similarly, to give you English translations selectively of any lines M through N, when you type ENG M-N.)

In the extended interaction illustrated below, notice also the use of the slash '/' as a command to get ARGUE to type lines of the derivation, and the use of the TEXT command to review the original statement of the argument and problem.

I will first review the hints and then execute what they advised.

Notice that after each hint ARGUE prompts me again for the next line of the derivation (in this case, line 3).

Enter a conclusion to derive, request a problem, or type 'BEGIN'
  _ SET 2 <RET>
There are 5 problems in this set.

Enter a conclusion to derive, request a problem, or type 'BEGIN'
  _ PROBLEM 2 <RET>

Persons have a right to life, allright! And it's morally wrong to kill
what's got a right to life. So, it's certainly morally wrong to kill a human fetus!

Supply the UNSTATED GENERAL PREMISE required to make this argument valid and derive the conclusion.

Let:  \( P_x = \text{'x is a person'} \)
     \( R_x = \text{'x has a right to life'} \)
     \( M_x = \text{'It's morally wrong to kill x'} \)
     \( F_x = \text{'x is a human fetus'} \)
     \( t / u = \text{'Arbitrary thing t / u'} \)

DERIVE:  \( (x)(F_x \Rightarrow M_x) \)

1  \( (x)(P_x \Rightarrow R_x) \)  / PREMISE
2  \( (x)(R_x \Rightarrow M_x) \)  / PREMISE

3  _ HINT <RET>

REMEMBER TO USE THE ENG COMMAND IN ORDER TO DISPLAY THE ENGLISH VERSION OF THE ARGUMENT ONCE THE ORIGINAL STATEMENT OF THE ARGUMENT HAS SCROLLED OFF THE SCREEN.

5 Hints remaining.

3  _ ENG <RET>

DERIVE:  \( (x)(F_x \Rightarrow M_x) \)

FOR ALL \( x \), IF \( x \) IS A HUMAN FETUS THEN IT'S MORALLY WRONG TO KILL \( x \)

1  \( (x)(P_x \Rightarrow R_x) \)

FOR ALL \( x \): IF \( x \) IS A PERSON THEN \( x \) HAS A RIGHT TO LIFE

2  \( (x)(R_x \Rightarrow M_x) \)

FOR ALL \( x \): IF \( x \) HAS A RIGHT TO LIFE THEN IT'S MORALLY WRONG TO KILL \( x \)

3  _ HINT

You must supply the tacit assumption needed to make the argument valid as an EXPLICIT PREMISE. Isn't it obvious? Either of these will do:

(a) Human fetuses are persons
(b) Human fetuses have a right to life

But while either will make the argument VALID, (b) makes premise (1) superfluous whereas (a) posits a connection between persons and rights.

4 Hints remaining.

3  _ HINT

Since the original argument posits a connection between being a person and having a right to life, your reconstruction of the argument should preserve this connection: the argument seems tacitly to assume that it is ON THE BASIS OF BEING PERSONS that human fetuses have a right to life.
The argument tacitly assumes that HUMAN FETUSES ARE PERSONS. So, state this!
3 Hints remaining.

3 _ HINT

Use UNIVERSAL INSTANTIATION (UI) on the premises in order to display their underlying sentential structure. You must instantiate to pseudo-names (t) in order to later derive the given conclusion by UNIVERSAL GENERALIZATION: what goes for EVERYTHING x goes for ARBITRARY THING t [by UI]; and what goes for ANY ARBITRARY THING t goes for EVERYTHING x [by UG].
2 Hints remaining.

3 _ HINT

Take 'Human fetuses are persons' (x) (Fx => Px) as an explicit premise

Get:
Pt => Rt by UI
Rt => Mt by UI
Pt => Mt by HS
Ft => Pt by UI and Ft => Mt by HS
1 Hints remaining.

3 _ HINT

Given: IF ANY ARBITRARY THING t is a human fetus, it's wrong to kill t
Then: For EVERYTHING x: If x is a human fetus, it's wrong to kill x

Thus, you can get (x) (Fx => Mx) from Ft => Mt by UG
where Ft => Mt was previously derived using UI
0 Hints remaining.

[Here it's time to get a clean listing of the problem]

3 _ LIST

DERIVE: (x)(Fx => Mx)

1 (x)(Px => Rx) / PREMISE
2 (x)(Rx => Mx) / PREMISE
3 _ TEXT

Persons have a right to life, allright! And it's morally wrong to kill what's got a right to life. So, it's certainly morally wrong to kill a human fetus!

Supply the UNSTATED GENERAL PREMISE required to make this argument valid and derive the conclusion.

Let: Px = 'x is a person'
Rx = 'x has a right to life'
Mx = 'It's morally wrong to kill x'
Fx = 'x is a human fetus'
t / u = 'Arbitrary thing t / u'

3 _ LIST
DERIVE: (x)(Fx => Mx)

1 (x)(Px => Rx) / PREMISE
2 (x)(Rx => Mx) / PREMISE

[I will now begin my derivation: first, by entering the 'missing' premise; then by using the slash '/' command to get ARGUE to type the lines I want.]

3 _ (x)(Fx => P) / Premise
4 _ / 1 UI xt [Note: Slash / command can be used to make substitution of t for x with UI.]

4 Pt => Rt / 1 UI
5 _ / 2 UI xt
6 Rt => Mt / 2 UI
7 _ / 3 UI xt
8 Ft => Pt / 3 UI
9 _ / 4 5 HS
10 Pt => Mt / 4, 5 HS
11 _ / 6 7 HS
12 Ft => Mt / 6, 7 HS
13 _ / 8 UG xt
14 (x) (Fx => Mx) / 8 UG

CONGRATULATIONS! YOUR DERIVATION IS COMPLETE.

List your derivation, ask for a problem, or type 'NEW'.

LIST

DERIVE: (x)(Fx => Mx)

1 (x)(Px => Rx) / PREMISE
2 (x)(Rx => Mx) / PREMISE
3 (x)(Fx => Px) / PREMISE
4 Pt => Rt / 1 UI
5 Rt => Mt / 2 UI
6 Ft => Pt / 3 UI
7 Pt => Mt / 4, 5 HS
8 Ft => Mt / 6, 7 HS
9 (x)(Fx => Mx) / 8 UG

List your derivation, ask for a problem, or type 'NEW'.

SET 51

There are 9 problems in this set.

Enter a conclusion to derive, request a problem, or type 'BEGIN'
[This is an example of how easily one can peruse problems in different problem sets.]

[Problem 1:]

Is preventing serious hurt to other people the only legitimate ground for justifying coercion or prohibition by law? Is HURT the only form of HARM?

Well, advertising the pleasures and techniques of sodomy on a large billboard in public is surely offensive even if it's not seriously hurtful to people. But if preventing serious hurt to others is the only legitimate ground for justifying prohibition by law, then (i) ONLY what is seriously hurtful may legitimately be prohibited and (ii) what is offensive but not seriously hurtful may NOT be prohibited. Therefore, serious hurt to others is not the only ground for justifying prohibition by law.

Be sure all the stated and additional required premises are symbolized. Then derive the conclusion. [Use Indirect Proof: CP + REDUCTIO.]

Let: 
\[ a = \text{the act of advertising sodomy in public} \]
\[ d = \text{the act of preventing actual hurt to other people} \]
\[ G_x = x \text{ is the only legitimate ground for justifying prohibition} \]
\[ L_x = x \text{ may legitimately be prohibited} \]
\[ H_x = x \text{ is seriously hurtful to other people}; \quad O_x = x \text{ is offensive} \]

Derive: \[ \neg G_d \]

1. \[ O_a \land \neg H_a \quad / \text{PREMISE} \]
2. \[ \_ \quad \text{PROBLEM 4} \]

[Problem 4:]

About my skinning my own dog alive -- when it's done secretly, say, in the privacy of my basement, it is neither hurtful to people nor offensive to people. So, although it is morally objectionable, my skinning my dog alive may NOT legitimately be prohibited so long as it's done secretly.

Symbolize the stated premise of this argument below, and supply the unstated premises required to obtain the conclusion. One of the missing premises is a tacitly assumed principle of legitimate prohibition: you must construct the conditions it alleges to be NECESSARY and/or SUFFICIENT for justifying coercive prohibition.

Let: \[ L_x = x \text{ may legitimately be prohibited} \]
Hx = x is hurtful to persons
Ox = x is offensive to persons
Sx = x is done secretly
Mx = x is morally objectionable
a = my skinning my dog alive

DERIVE: Ma & (Sa => -La)

1  _ STOP <RET>

RIGHTO! GOODBYE!
ARGUE allows you to enter your own derivation problems or arguments to reconstruct in valid form. When constructing your own argument in ARGUE it's possible to assign English translations to the logical constants of the argument so that ARGUE can give you translations of any steps of the argument. You enter symbolized premises or target conclusions and use the ASSIGN command to make the English assignments (as illustrated in the sections that follow).

The **ASSIGN** Command

This command allows you to assign English translations to variables and constants so that ARGUE can print out a rough English translation of any line(s) of your argument. When you make any assignment, ARGUE continues to prompt you for further assignments. When you are finished making assignments, simply hit RETURN in response to ARGUE's prompt and ARGUE will return you to the last line of your derivation.

It is important to remember the following symbolic conventions:

1. Let F through S stand for either sentences or predicates. (In ARGUE we reserve other letters for other logical constants and operators.)

2. Indicate a one or two-place predicate by one or two subscripts -- For example: Fx is a one-place, Fxy a two-place predicate.

3. Let x, y, z, or w be individual variables.

4. Let a, b, c, or d be individual constants.

Remember the **ALIST** Command.

This command will print out a list of the variable/constant assignments you make.

Remember the **ENG** Command.

This command will print your derivation with English, if you have entered English translations using ASSIGN. ENG N gives the English for line N. ENG N-M gives the English for lines N through M. ENG (alone) gives the English for all lines of your argument/derivation as far as you've gone.
4.1. "Enter a Conclusion to Derive..."

In the sample interactions that follow, notice that in the first example I have entered my own conclusion to derive along with my own premise. ARGUE then congratulates me when I have derived the designated conclusion, just as with a stored problem. Notice that after I have completed my derivation, ARGUE still allows me to LIST it or get ENGLISH translations. Notice also my use of the ASSIGN command.

Type 'INS' for instructions; otherwise,

Enter a conclusion to derive, request a problem, or type 'BEGIN'

BEGIN YOUR DERIVATION:

1  _  -(G => P) / PREMISE
2  _  ASSIGN

Variable or constant: _ G

Now enter an English assignment: _ God has committed a grievous crime

Variable or constant: _ P

Now enter an English assignment: _ God will be punished

2  _  /1 IMPL
2  _  -(G V P) / 1 IMPL
3  _  /2 DEM
3  _  G & -P / 2 DEM

CONGRATULATIONS! YOUR DERIVATION IS COMPLETE.

List your derivation, ask for a problem, or type 'NEW'.

LIST

DERIVE: G & -P

1  _  -(G => P) / PREMISE
2  _  -(G V P) / 1 IMPL
3  _  G & -P / 2 DEM

List your derivation, ask for a problem, or type 'NEW'.

ENG 1
1  -(G => P)  
   NOT (IF God has committed a grievous crime, 
   THEN God will be punished)

List your derivation, ask for a problem, or type 'NEW'.

   ENG 3

3  G & -P  
   God has committed a grievous crime 
   AND it's NOT the case that God will be punished

List your derivation, ask for a problem, or type 'NEW'.

   NEW

After listing my last derivation I typed 'NEW' to begin anew on something else. In
response, ARGUE prompts me as usual:

Enter a conclusion to derive, request a problem, or type 'BEGIN'
4.2. "... Or Type 'BEGIN' "

Notice in the example below that when I type 'BEGIN' in response to ARGUE's prompt, the program allows me to pursue any line of derivation: because ARGUE doesn't know in this case what my objective might be, it will let me continue until I type 'NEW' to get a new set of options (or 'STOP' in order to exit the program).

Since it does not know my objective, ARGUE will not at any point congratulate me on my derivation; it will only inform me if and when I make an error.

This "Begin" mode is useful for exploring what follows logically from some premise(s) without having to specify any particular conclusion. Here's an example where what we can derive—what is logically implied— is perhaps a little surprising.

Suppose I tell you: It's not the case that if you fail the final exam you will not pass the course.

This statement is intentionally a little confounding. What does this imply? Does it follow, for example, that you will fail the final but you will nonetheless pass the course? Let's see:

Notice that I did NOT say: If you fail the final you will not pass the course. Rather, I said: It's NOT the case that if you fail the final you will not pass the course. What?

Let's explore what we can derive from this statement in ARGUE; this will also serve to illustrate ARGUE's facility for assigning English translations to sentential symbols that you enter into the program.

Enter a conclusion to derive, request a problem, or type 'BEGIN'
  _ BEGIN

Begin your derivation:
  1 _ -(F => -P) / PREMISE
  2 _ ASSIGN
Variable or constant: _ F
Now enter an English assignment: _ You fail the final exam
Variable or constant: _ P
Now enter an English assignment: _ You pass the course
Variable or constant: _ <RET>

2 _ ALIST
F: You will fail the final exam
P: You will pass the course

2 _ LIST

1 -(F => -P) / PREMISE
2 /1 IMPL
2 -(~F V ~P) / 1 IMPL
3 /2 DEM
3 F & P / 2 DEM

4 _ LIST

1 -(F => -P) / PREMISE
2 -(~F V ~P) / 1 IMPL
3 F & P / 2 DEM
4 _ ENG 1
1 -(F => -P) It's NOT the case that
if you fail the final you will not the course

4 _ ENG 3
4 F & P You'll fail the final but you'll pass the course

4 _ NEW
Enter a conclusion to derive, request a problem, or type 'BEGIN'

You have shown that
You will fail the final but you will pass the course nonetheless
can be derived from my statement

It's not the case that if you fail the final you won't pass the course.

-- a perhaps surprising result, This illustrates one exploratory use of the "Begin"
mode in ARGUE, as well as use of the ASSIGN command for getting translations of formulae that you enter.

At this point you could STOP and exit the program, enter your own conclusion to derive, request a problem from a problem set, or begin again as above.

Remember: Whatever English assignments you make within a derivation are forgotten by ARGUE as soon as you go on to a NEW derivation.
Appendix I
What the ARGUE Program Can Help Teach
Formal Logic in the Service of Argument Analysis

1.1. Four Tasks of Formal Argument Analysis

This section contains brief illustrations of four basic tasks in the analysis of arguments, particularly philosophic arguments about normative issues. Normative issues are issues about the norms, rules or principles which are the logical bases for our arguments and judgments about what is right and what is wrong.

The following sections offer an extended illustration of these four basic tasks of argument analysis applied to a sample argument about social policy, the policy of preferential hiring. These illustrations demonstrate how the formal logic you are learning can be applied in the systematic reconstruction of arguments. Careful reconstruction of arguments is a prerequisite for their analysis; and it provides a guiding framework for philosophic analysis, the analysis of the normative principles that are crucial premises of our arguments about what is right and what is wrong. These illustrations will demonstrate how the formal reconstruction of arguments is relevant to philosophic analysis and will introduce you to basic tools and techniques of philosophic analysis, in particular, the concept of plausibility and the use of counter-examples in testing the plausibility of normative principles. In the sections that follow you will begin to see how formal validity can serve as a guide in the pursuit of truth and justified belief in philosophy.

1. ANALYZING LOGICAL FORM: CHECKING FOR VALIDITY

One way to get clear about a given line of reasoning, about the assumptions that must be made for a conclusion to follow logically, is to reconstruct the line of reasoning in the precise form of a deductively valid argument. This enables one to identify key assumptions, to see the precise logical form these must take in order to support a given conclusion, to uncover tacit assumptions (unstated premises needed to make the argument valid), and to single out inessential assumptions (logically superfluous premises, premises not needed for the argument to be valid).

In many of the arguments we encounter, especially in philosophic disputes, there will be tacit premises (assumptions that are not stated but that must be made explicit for the argument to be valid). Before we can adequately evaluate any argument, we must make any tacit premises (unstated but necessary assumptions) explicit. The first step in our reconstruction of any argument will be to analyze and represent the
logical form of the argument’s stated premises and conclusion: this provides an important clue to the form and content that any tacit premises must have in order to make the argument valid.

Consider the following argument.

As the first step in reconstruction, I've represented the logical form of the argument symbolically to its right:

If you have ambition, you're in for a lot of frustration. And, you're apt to be miserable if you're often frustrated.

So, your life will take on purpose only if you're apt to be miserable.

Once the logical form of the argument is represented as above we can tell by inspection that the argument is invalid: There's no way to derive the conclusion from the given premises. But, in this case, it should also be clear from inspection of the argument's logical form that a premise of the following logical form would be sufficient to make the argument valid:

\[
\begin{align*}
1 & \quad H \Rightarrow F \\
2 & \quad F \Rightarrow M \\
3 & \quad P \Rightarrow H \\
4 & \quad H \Rightarrow M \\
5 & \quad P \Rightarrow M
\end{align*}
\]

The derivation above shows that a premise of the form

\[
P \Rightarrow H \quad \text{Your life will take on purpose only if you have ambition}
\]

is sufficient to make the argument valid.

Once you've depicted the logical form of an argument in some clear and explicit (symbolic) way, it's easier to inspect the argument for validity and to see what forms of additional premise(s) will make it valid. This is why our first task in reconstructing and analyzing an argument will be to analyze and represent (symbolize) its logical form: to check the argument for validity and unstated premises needed for validity.

This first step in the analysis of arguments (ANALYZING LOGICAL FORM and
CHECKING FOR VALIDITY) will help us with one of the most crucial tasks of PHILOSOPHIC ANALYSIS, which is the second basic task of argument analysis: identifying underlying principles and 'hidden' or tacit (unstated but necessary) premises, illustrated below.

2. IDENTIFYING UNDERLYING PRINCIPLES AND TACIT PREMISES

To identify the general principle behind a position on some issue, it helps to try various formulations of the likely candidate principles as explicit premises in a valid deductive argument whose conclusion is the position in question. Consider, for illustration, the following widely held position:

[P] We owe it to future generations to control population size.

A likely principle behind this position [P] would be the following:

[0] We have an obligation to future generations to bequeath to them the means for the best possible life.

An additional tacit premise in support of [P] might be:

[0] => [P] IF we have an obligation to future generations to bequeath them the means for the best possible life,

THEN we owe it to future generations to control population size.

We can see by inspection, or prove by the derivation on the right below, that the resulting argument form is valid:

\[ O \quad 1. \ O \quad \text{/ Premise [Unstated Principle]} \]

\[ O => P \quad 2. \ O => P \quad \text{/ Premise [Unstated]} \]

\[ P \quad 3. P \quad \text{/ 1, 2 MP} \]
3. SORTING OUT AMBIGUITIES IN PRINCIPLES AND PREMISES

There is usually more than one way of construing the general principle behind a given position. We want to beware, in particular, of ambiguous principles, principles that can be taken to mean either or both of two different things. Once we've identified a likely candidate principle (like [O]) behind a position (like [P]) and reconstructed an argument for that position with the likely principle and other likely assumptions needed for validity stated as explicit premises (as above), we need to be aware of other possible formulations of the principle in question and, in particular, to beware of any ambiguity that may be lurking in the principle we've chosen in reconstructing the argument. This is the next task in argument analysis: sorting out ambiguities in underlying principles and premises.

Can you think of at least two different things that the principle [O] (Premise 1 below) might be taken to mean?

1. O We have an obligation to future generations to bequeath them the means for the best possible life

2. O => P IF so, THEN we owe it to future generations to control population size

3. P So, we owe it to future generations to control population size

Here are three things Premise 1 might be taken to mean:

[01] We have an obligation to future generations to ensure that the future population, however it be constituted, has the means for the best possible life of which it is capable

[02] We have an obligation to future generations to ensure that the future population is so constituted as to be capable of enjoying the best possible life

[01 & 02] We have BOTH these obligations to future generations
WHICH of the above versions of the original principle [O] we incorporate into the argument for the call for population control [P] will make a decided difference to the SOUNDNESS or PLAUSIBILITY of the argument, as follows.

Using [O2] as a premise (or the conjunction of [O2] and [O1]), it’s pretty easy to construct an argument with [P] as a conclusion that is valid. But it is not so easy—it may well be impossible—to construct an argument using [O2] that BOTH (1) is VALID and (2) has all true or PLAUSIBLE premises, for this reason: [O2] (or any premise containing [O2]) is objectionable/IMPLAUSIBLE because it has objectionable logical consequences: Not only can [O2] be used to support a policy controlling the SIZE of any future population, but it can also be used to support a number of other quite objectionable policies, such as policies of genetic manipulation, selective breeding and sterilization of humans, consumer preference conditioning, and the like.

On the other hand, while [O1] by itself may be perfectly plausible (and have no untoward consequences if adopted), [O1] by itself (without additional premises) will not constitute a VALID argument in support of the control of population SIZE [P]. It will be difficult—if not impossible—to use [O1] along with additional PLAUSIBLE premises to construct a VALID argument in support of the control of population SIZE [P].

TRY IT! Try to construct an argument for [P] using [O1] (but not [O2]) that BOTH (1) is VALID and (2) all of whose premises are true or PLAUSIBLE.

How, actually can we assess the PLAUSIBILITY or [TRUTH] of the premises (normative or factual) of philosophic arguments. This is the next crucial task in argument analysis, and a basic task in all philosophic analysis: testing the truth or plausibility of general principles and premises. An important tool for this task is the counter-example, illustrated in the next sections.
4. TESTING THE PLAUIBILITY OF PRINCIPLES AND PREMISES
THE USE OF COUNTER-EXAMPLES IN PHILOSOPHIC ARGUMENT

Scientists refute empirical hypotheses by citing counter-evidence—instances in which the hypotheses are false. Somewhat similarly, philosophers standardly refute putative definitions of important concepts (justice, knowledge, etc.), as well as general normative principles, by citing counter-examples. Arriving at a satisfactory definition by a series of successive formulations and counter-examples is the most characteristically philosophical of reasoning techniques. Attributed to Socrates, it is an important feature of philosophic analysis.

To illustrate the philosophic use of counter-examples, consider the distinction commonly cited by medical personnel between killing a patient and letting him die. Some try to explain it thus:

To let someone die, as opposed to killing him, is to be in a position to save his life but deliberately to refrain from doing so.

This formulation may be shown to be incorrect by the following counter-example. Suppose someone smothers you by pressing a pillow to your face for a period of several minutes. Once the pillow is in place, he is in a position to do something that would save your life—viz., lift the pillow—but he deliberately refrains from doing so. Yet, contrary to the proposed explanation, it would be natural to say that such a person killed you, not that he merely let you die.

Sometimes it is said that the distinction between killing and letting die is an instance of the 'active/passive' distinction. Putative counter-examples to this proposal are ready to hand. Removing a respirator from a critically ill patient is surely 'active' rather than 'passive.' Yet such an action could well be described as 'letting die' rather than 'killing.' Or suppose an anesthesiologist deliberately fails to make the necessary adjustments in certain life-support and monitoring systems attached to a patient undergoing surgery, thereby deliberately ensuring that the patient dies. Although such failures to act are 'passive,' it would be natural to accuse the anesthesiologist not merely of letting the patient die but of killing him.
The four basic tasks illustrated in the analysis of the sample argument below are:

1. The Reconstruction of an Argument in VALID Deductive Form.

2. The Explication or Revision of the Underlying Normative Principles that are Crucial Premises of the Argument.

3. The Analysis of any Ambiguity in the Crucial Premises of the Argument.

4. Testing the Plausibility or Truth of the Crucial Premises of the Argument Against Putative Counter-Example

These tasks are not exhaustive of what-all is involved in the analysis of arguments, but they are basic and important. Tasks (3)-(4) often require a reformulation of an argument, which in turn requires the repetition of tasks (1) and (2) in order to preserve the validity of the argument and the plausibility of its crucial premises.

The analysis of arguments requires the coordination of logical analysis and philosophic analysis.

Logical analysis primarily concerns the logical form the premises must take in order to support conclusion and maintain validity.

One function of philosophic analysis is to test the plausibility of the normative principles underlying our judgments and arguments; thereby, to articulate and develop the various issues underlying our arguments by adducing pertinent objections to the crucial premises of an argument (counter-examples or problem-cases), possible replies to those objections, possible rejoinders to the replies. The procedure, in rough outline, is: (1) to try to capture in explicit premises the 'intuitions' (and tacit principles) to which we appeal in our particular judgments by (2) abstracting those principles from clear 'paradigm' cases (cases, precedents or common practice where we are especially confident in our judgments); then (3) to adduce the logical consequences of our principles and test our premises against conflicting intuitions about putative counter-examples and (4) to reformulate these premises in order to overcome refutation by counter-example, to adjudicate or explain away problem-cases; and, so, (5) to repeat this process of careful formulation, testing and reformulation . . . until, ideally, we have made our arguments clearly valid and rendered our principles (a) perfectly explicit, (b) logically consistent among themselves, (c) evidently immune to further refutation by counter-example and (d) sufficient for adjudicating or explaining away problem-cases.
The analysis of the sample argument that follows is an illustration of logical and philosophical analysis coordinated in the service of the four basic tasks outlined above.
I.2. A Sample Argument: Preferential Hiring

Justice surely demands that someone unjustly deprived of something to which he had a right be compensated. Of course, normally, it's wrong to discriminate among job applicants on the basis of racial or sexual characteristics. But there are exceptions (as to any general rule). Blacks and women, for example, have a right to equal opportunity for advancement in education and employment. Yet both have been unjustly discriminated against in these areas. Not only does justice require that victims of such discrimination and right-violation be compensated, but by hiring blacks and women in preference to white males we do not thereby discriminate in a morally objectionable way. We rather compensate the victims of job discrimination as justice demands.

The foregoing is an argument from alleged requirements of justice. People often appeal to considerations of compensatory justice in defense of preferential hiring. One point we want to make as explicit and precise as possible by constraining the argument in valid form is exactly what justice is supposed to require.

Reconstructing this line of argument in deductively valid form will not produce a single argument or principle of justice: many valid reconstructions are possible employing any one of several possible formulations of the alleged requirements of justice.

But by constraining the argument in deductively valid form we force ourselves to specify some principle explicitly connected to the policy in question. This begins the dialectical program of successive reconstructions of the principle to take account of objections to it, and successive reconstructions of the argument providing the logical connection between the principle and the policy of preferential hiring.
1.3. An Initial Reconstruction

Any simple, plausible formulation of the argument will do for starters. Whatever formulation we begin with, it can be made progressively more precise under the fire of counter-examples and within the constraint of deductively valid form.

Consider the following generalized reconstruction of the argument. Replacing 'Xs' by 'blacks' or 'women' and 'Ys' by 'white males' will render the intended conclusion of the original argument. In brackets I will assign a variable letter to each statement in the argument so that its sentential-logical form can be readily depicted.

\[( A ) \ ( 1 ) \] If \([0]\) Xs have been unjustly deprived of something (e.g., equal employment opportunity) to which they had a right, then \([J]\) justice demands that Xs be compensated

\[(2)\] \([0]\) Xs have been unjustly deprived of something (i.e., equal employment opportunity) to which they had a right

\[(3)\] \([M]\) Preferential hiring of Xs over Ys is on balance morally permissible if \([S]\) preferential hiring of Xs serves to compensate them as victims of job discrimination

Therefore: \([P]\) Justice demands preferential hiring of Xs over Ys

There are at least three problems with the argument as stated that come out in the course of reconstruction. First, whatever quarrel one might have with the accuracy of the initial reconstruction (A) (anyone may try his own), one can see that the logical form of the argument is, in any case, not manifestly valid: the conclusion does not follow. There are important unstated assumptions. Sentential logic will suffice, for starters, to find and fill the gross logical gaps in the argument. Second, the argument is rife with ambiguity. Third, once one has sorted out some of the ambiguity, it turns out to be remarkably difficult to reconstruct the argument so that it both is valid and has all true or plausible premises, premises at least immune to obvious counter-example. From these lessons of reconstruction philosophic lessons are also to be learned. I will deal with them in turn.
1.4. The First Problem: Invalidity and Unstated Premises

The argument as it stands is invalid. This is easily seen by inspection of its logical form, abstracted symbolically:

\[(A') (1') \text{If } O \text{ then } J \quad (A'') (1'') \quad O \Rightarrow J\]
\[(2') O \quad (2'') O \]
\[(3') M \text{ if } S \quad (3'') S \Rightarrow M\]

Therefore: P

One advantage of being able to depict the logical form of an argument in abbreviated notation is analogous to the advantage of having an x-ray device: it allows us to look at the bare skeletal structure apparently supporting the conclusion, to detect distinctly structural flaws underneath the enveloping verbal flesh and musculature. From our x-ray of argument (A) it’s clear that the conclusion P is in no way explicitly connected to any of the stated premises. Nor is any explicit connection between premises (1) and (2) and premise (3) yet apparent: from (1) and (2) we can conclude J; but what connections are presumed to exist among J, premise (3) and the conclusion P? These connections, in some form, must be made explicit, so as to make explicit use of the stated premises of the argument and also render it valid. Here we need to consider the content as well as the form of the argument.

Symbolic logical form and validity serve, respectively, as clues and guiding constraints in the search for tacit premises; but they are not sufficient grounds for generating sensible additional premises, or for deciding among competing premises where any number might make an argument valid. It is necessary to introduce other guiding constraints in the reconstruction of an argument. Validity remains a powerful minimal condition of the enterprise nonetheless: insisting on manifest validity keeps us honest about what exactly is or must be assumed and exactly what follows from what. In this case it requires us to produce some further assumptions on which the conclusion tacitly rests. Once laid out explicitly, these assumptions are open to question; and they may force us to change the shape of the argument or even abandon it. One obvious tacit assumption is:

[R] Ys (white males) have received undue preferential treatment over Xs (blacks or women) in hiring practice.

Without assuming at least some such condition it would make no sense to assert that it is morally permissible to compensate Xs at the expense of Ys. Moreover, without the addition of some such condition as R to premise (3), this premise is open to obvious counter-example and, so, is false. That is, the truth of S is not
always a sufficient condition for the truth of M, for surely the following interpretation of premise (3) is false:

If preferential hiring of Mexican-Americans (X's) compensates them . . . then preferential hiring of Mexican-Americans (X's) over blacks (Y's) is morally permissible.

The logical form of our further reconstruction now looks like this:

(A')  (1') If 0 then J  
(2')  0  
________________________  
Therefore:  J  
(3')* If S and R, then M  
(4')  R  
________________________  
Therefore:  P

Explicitly assuming the condition S as an additional premise

(5')  S  

we may draw the further intermediary conclusion

Therefore:  M

from (3'), (4') and (5'). We have now gotten so far as to conclude that [J] justice demands compensation and that [M] preferential hiring is on balance a morally permissible way to compensate. We have yet explicitly to complete the connection to the ultimate conclusion [P] to the effect that justice in turn demands preferential hiring as the mode of compensation. Any of the following additional premises connecting the conclusion to the foregoing results would render the argument valid:

(6')  (a) If J and M, then P
           
           (b) If J then P
           
           (c) If M then P

Both (b) and (c) are objectionable, on similar grounds. That [J] justice demands compensation is not sufficient grounds for asserting that [P] justice demands that compensation take a particular form, namely, preferential hiring. That [M] preferential hiring (or anything else, say, singing in the shower) is on balance morally permissible

is not sufficient grounds for holding that [P] justice requires it. So, (b) and (c) are
implausible or false. Moreover, their addition to the argument, while making it valid, would be to cast adrift other presumably relevant premises as logically superfluous.

(6') (a) seems the best of the three alternatives. Choosing it has been an exercise in the reconstruction of a normative principle, an attempt to specify sufficient grounds on which justice would require and, so, justify a particular policy. The reconstruction of principles and the reconstruction of arguments go hand-in-hand in the moral-philosophic forum, because general normative principles are always among the (stated or tacit) assumptions of a moral-philosophic argument. Hence, the reconstruction of arguments can play a heuristic role in the explication and analysis of the normative principles underlying our reasonings.

Once the argument, with its tacit underlying principles, has been reconstructed in valid form, we are at least assured that if the premises are acceptable, so must be the conclusion. But are they?
1.5. Further Problems: Ambiguity and Vulnerability to Counter Example

For purposes of illustration, I will focus on the first premise of the argument only. Where ambiguities are discerned or counter-examples found premises must be reformulated, jettisoned, or added. Premise revision involves further, alternative reconstructions of the argument to preserve its validity. This may be difficult, but to that extent instructive.

Consider: The principle of compensatory justice to which argument (A) appeals, premise (A-1), can of course be applied quite generally. So, the original line of reasoning and policy based on this principle can be applied quite generally. How generally? To whoever can be counted among the X's. Who might be counted among the X's? On grounds of premise (A-1), anyone who has ever been deprived of something to which she had a right, say, 'equal' employment opportunity. (He could well be a highly competent white bank executive from a wealthy family who has been denied 'equal' consideration for jobs many times because of his religious or political views.)

There is a serious ambiguity in premise (A-1). How are we to interpret the demand for compensation of Xs? There are at least two possibilities where Xs are members of some identifiable group:

A distributive interpretation: a person is to be compensated if he is an X (black, woman, atheist . . .) and he has himself been unjustly discriminated against . . .

A collective interpretation: a person is to be compensated if he is an X and Xs (blacks, women, Irish-Catholics, communists . . .) have in general been unjustly discriminated against . . .

The collective interpretation of the demand for compensation for Xs does not require that any given X have been unjustly discriminated against, but rather that other Xs as a group have been unjustly deprived: under this interpretation Xs as such are to be compensated.

A wealthy Jewish or Irish Catholic businessman who had never himself been deprived of anything could qualify under the collective interpretation for compensation where X's were Irish Catholics or Jews. A wealthy white male who had himself been unjustly discriminated against because of his atheism could qualify for compensation under a distributive interpretation. Presumably the purpose of the preferential hiring policy in question is neither to compensate wealthy people nor to
compensate just anybody for any unjust deprivation she may have suffered. Under either interpretation the general demand for compensation could be applied to practically anybody; whereas the specific demand for compensatory preferential hiring is on behalf of certain presently and unfairly disadvantaged groups, namely, certain racial minorities and women.

We need to specify the conditions of premise (A-1) so as to justify compensatory treatment in the form of preferential hiring for all and only those whom the policy is meant to compensate. For a sense of these two possibly conflicting constraints—the justice and purpose of the policy in question—we need appeal to our intuitions, our tacit conceptions of both, as yet imperfectly captured in premise (A-1) (and yet to be tested against limiting counter-examples).

We need first to specify more precisely the grounds on which justice demands compensation. We will consider five candidate criteria. These will be sufficient to delineate some of the major ambiguities of our original principle of compensatory justice.

(a) Membership in a group whose members have been widely and unjustly discriminated against and thereby deprived of something to which they had a right.

This criterion would qualify blacks and women, but would it qualify all and only those actually deserving compensatory treatment? As already suggested, it would not. The criterion is too inclusive. Who would not qualify for compensation? Consider the cases of well-to-do Catholics, Protestants or Jews who have never been unjustly deprived of anything but who are members of groups which have (somewhere) suffered great injustice. Does justice require that they be compensated?

It is evidently not mere membership in some identifiable class of persons many of whose members have suffered injustice at some time in the past that recommends a given member for compensation. Yet the policy we are seeking to justify on the basis of the requirements of justice designates its beneficiaries according to racial or sexual characteristics.

Perhaps it is rather the likelihood of having herself suffered injustice, given effective, recent and widespread prejudice and discrimination against Xs as such, that recommends any given X for compensatory treatment.

(b) Likelihood of having suffered unjust deprivation oneself because of membership in a group whose members have been
recently and widely discriminated against.

This criterion would include blacks and women and exclude consideration of white Catholics or Jews for compensation. But against this suggestion stands the case of any well-to-do black woman who has never been deprived of anything. Does justice demand that a person who has never suffered any injustice be compensated? What is it for which she would be compensated? Analogously, should courts award compensatory damages to a person on the likelihood that he suffered defamation of character when in fact he hasn’t—or because it is established that he was actually wronged and harmed in a way penalizable by law? The latter case is more problem-case than counter-example. Consider then:

(c) Having in fact been unjustly discriminated against and thereby deprived of something to which one had a right.

Whatever justice demands in the way of compensation, it would seem that justice demands it only for persons who have themselves been unjustly harmed or deprived, not for persons who haven’t in fact been wronged but who happen to have certain characteristics (e.g., race) in common with others who have.

But while actually having been wronged oneself may be a necessary condition for claiming compensation on grounds of justice, this requirement of justice would not justify preferential hiring of blacks or women as such. On the other hand, a principle stipulating criterion (c) as a sufficient condition for compensatory treatment would not justify preferential hiring of all and only those (certain minority groups and women) whom the policy seems intended to benefit. Such a principle would justify compensatory treatment of a person irrespective of her race, sex or socio-economic status. A policy of preferential hiring based on such a criterion, designating beneficiaries according to their personal histories rather than race or sex, would seem impracticable.

There are further ambiguities in the position regarding the qualifications for compensatory treatment. Some of these are made explicit in the multiple-choice reconstruction, argument (E), below.

One instructive difficulty with the line of argument under consideration has clearly emerged in our reconstructive effort: the problem of fitting (logically connecting) the desired policy (compensatory preferential hiring of certain racial minorities and women) to the requirements of justice that seemed initially to demand such a policy.
The grounds on which justice might demand and distribute compensation are not obviously the grounds on which the policy in question would distribute preferential treatment.

A fairly superficial examination of the ambiguities of our initial formulation of what justice requires has produced a fair array of questions. Never mind objections to other premises for now. We find that the alleged requirements of justice are themselves clearly questionable. We can map out the issues and strategic options confronting us by reformulating the original deductive argument to take account of the ambiguities and objections raised. We might consider this endeavor a kind of game, a game of argumentative reconstruction and counter-example.

Attempting to reconstruct the argument in expressly valid form, while taking account of ambiguities and counter-examples, makes the philosophic problem of fitting the desired policy to the demands of justice more acute. While this reconstructive exercise may well make the argument less persuasive; the exercise is nonetheless instructive regarding some of the philosophic issues underlying the policy in question. This is one educational objective of the task of reconstructing arguments in deductively valid form, and one rationale for the CAI programs in argument construction and reconstruction, which enforce the constraint of validity while allowing the student to view and manipulate both the logical form and the content of an argument side-by-side.
I.6. The Game of Formal Argument Reconstruction and Counter Example

The object of this game is to construct an evidently valid and plausible argument supporting the policy of preferential hiring in question from the requirements of justice.

The first phase of the game is to construct a deductive argument whose conclusion is the position on the policy in question by selecting those premises required to make manifest the validity of the argument.

The second phase of the game is to test the plausibility of the selected premises, by explicating ambiguities and adducing putative counter-examples or problem-cases. A given premise remains plausible only so far as it is at least immune to obvious counter-examples.

Our two concerns are the logical connection between premises and conclusion, a matter of logical form, and the plausibility of the premises, a function of their actual content and the argumentative context. As we shall see, these concerns are not unrelated in the game of argumentative strategy that follows. It is often very difficult to satisfy both constraints at once. In this respect deductive validity, a matter of logical form, is indeed related to the pursuit of truth. Where the task is apparently impossible, we have good reason to abandon an argument and seek alternative strategies.

Consider now a multiple-choice reformulation of the original argument from justice, argument (B), below. Premises may be constructed by selecting one (or more) of the lettered options (and providing suitable logical connectives). The options (a)-(e) given under each premise are intended to take account of the ambiguities already detected in the principle of justice employed in the original argument (A). At this stage of reconstruction it is useful to have recourse to quantificational logic. The logical form of each premise is symbolized to its right so that validity can be readily assessed by inspection or derivation.
(1) If a person

(a) is a member of a group whose members have been widely and unjustly discriminated against and thereby deprived of something to which they had a right

(b) is likely himself to have been unjustly discriminated against and thereby deprived of something to which he had a right

(c) has in fact himself been unjustly deprived of something to which he had a right

(d) has himself suffered harm or serious disadvantage as a result of having been unjustly discriminated against

(e) presently is suffering harm or serious disadvantage as a result of having been unjustly discriminated against

THEN justice demands that person be compensated.

(2) ALL women

(a) are members of some group whose members have been widely and unjustly discriminated against and thereby deprived . . .

(b) are likely to have been unjustly deprived . . .

(c) have in fact been unjustly discriminated against and thereby deprived . . .

(d) have suffered harm or serious disadvantage as a result of having been unjustly discriminated against

(e) are presently suffering harm or serious disadvantage as a result of having been unjustly discriminated against

Therefore: Justice demands that women be compensated.
We are considering now just the first stage of the original argument, to the first intermediate conclusion that justice demands compensation for, say, women. If we can't construct a valid and plausible argument to this intermediate conclusion, we can hardly justify the policy in question on the basis of premise (A-1).

There are at least five different sets of premises, consisting of alternative versions of premises (A-1) and (A-2), that will each provide a valid argument to the first conclusion. They are: (A-1a), (A-2a); (A-1b), (A-2b); (A-1c), (A-2c); (A-1d), (A-2d); (A-1e), (A-2e). The list could be lengthened by including sets of compound premises, such as: (A-1e or d), (A-2e). However, nothing would be gained thereby. Keeping in mind the questions raised previously regarding criteria (a) through (e), consider the arguments resulting from these sets of premises. Which of the optional arguments is most evidently sound? As it happens, none is sound (i.e., both is valid and has all its premises immune to obvious counter-example).

Conditions (1c), (1d) and (1e) seem to provide the most plausible grounds for justice to demand compensation, namely: that a person herself has suffered an injustice, or harm as a result of injustice, in order that she actually have something to be compensated for. The difference between (1c) and (1d) or (1e) concerns whether we wish to compensate persons who were in fact treated unjustly but who suffered no harm or disadvantage on that account. Is it the mere fact of injustice or rather the resultant harm that demands compensation? If the latter, is it present or past harm?

By contrast, condition (1a) seems an implausible basis for justice to demand anything, let alone compensation. Mere membership in a group does not suffice to establish that a person suffered any injustice. If a person suffered no injustice, what is she to be compensated for? Some further condition seems necessary to establish an evidentiary connection between group membership and injustice. (A-1b) makes this condition explicit, asserting a probable connection. The implausibility of (A-1a) can be shown by an appeal to the untoward consequence that would result from its general application: cases where justice demanded compensation but where there was no victim of injustice to be compensated. The same objection could be lodged against (A-1b), with counter-examples.

Whereas premises (A-1c), (A-1d) and (A-1e) provide plausible grounds for justice to demand compensation, the factual assumptions respectively required to entail the desired conclusion are very likely false. On the other hand the factual assumptions (A-2a) and (A-2b) while true, seem to provide insufficient grounds for justice to
demand compensation, as shown by counter-examples to (A-1a) and (A-1b) above.

If we revise this stage of the argument in order to demand compensation, discriminately, only for those (blacks, women or Xs) who qualify on conditions (1c), (1d) or (1e), we would vitiate the validity of the argument to the final conclusion (which calls for compensation for ALL blacks or women as such). If we revise the final conclusion, to preserve validity, and thereby discriminately demand preferential hiring for only those who qualify on conditions (1c), (1d), or (1e), we cannot justify compensatory hiring of blacks and women as such. We would then be arguing for a very different, and probably impracticable, policy. The difficulty is to provide a manifestly valid argument with plausible premises to the specified conclusion. This difficulty would be compounded if we were to examine other premises in the argument. More subtle refinement or reformulation of the premises will not eliminate the basic difficulty.

We have reached an apparent impasse in our game of argument reconstruction. There are some lessons of argumentative strategy to be gained at this impasse.

It is not obvious that an appeal to the requirements of compensatory justice is, after all, viable in behalf of preferential hiring of blacks, women or other minority members as such. This has been shown by making certain alternative connections between the policy and the presumed requirements of justice explicit in valid deductive form. Making the supposed logical connections between policies and principles explicit in this way forces us to clarify our normative assumptions and provides us with clear departure points for further dialectical analysis of policy issues and matters of principle.

We also get clearer on exactly what position we want or need to hold on the policy in question. When we reach an impasse such as we have in our appeal to compensatory justice, we would be well-advised at least to consider other lines of argument. Perhaps mere compatibility with the requirements of justice and an appeal to ‘social utility’ would suffice to support the policy in question. Perhaps the correction of certain social ills (disproportionate poverty, unfair competitive disadvantage or unemployment among certain minorities) or the provision of certain social benefits (positive role models and career incentives) is the proper aim of the policy in question. Perhaps these ends, if achievable with negligible infractions of justice, would justify preferential hiring of blacks or women as such. Perhaps. This, in any case, is a strategy different from the one with which we began. To get clear
on exactly what would have to be true in the way of both normative and factual assumptions in order to support preferential hiring along these lines we would do well to make those assumptions explicit within the frame of a valid deductive argument.

And the game of argument reconstruction and counter-example would resume. A game that is, after all, a serious form of philosophic inquiry within strict logical constraints. The ARGUE program enforces the constraint of validity, while hints in the stored problems provide suggestions regarding the plausibility of alternative premises, in order to rehearse you in the game of argument reconstruction.
Appendix II
ARGUE's Rules of Inference and Replacement

CONTENTS

SENTENTIAL RULES OF INFERENCE

SENTENTIAL RULES OF REPLACEMENT

CONDITIONAL PROOF RULE & INDIRECT PROOF STRATEGY

QUANTIFICATIONAL RULE SCHEMAS

RESTRICTIONS ON THE USE OF UNIVERSAL GENERALIZATION

RESTRICTIONS ON THE USE OFEXISTENTIAL INSTANTIATION
II.1. Sentential Rules of Inference

Premise (PREM or P)
A premise may be introduced on any line of a derivation, except within Conditional Proof.

Excluded-Middle Introduction (E-MI or EMI): \( P \lor \neg P \)
At any point in a derivation one may introduce a sentence of the above form.

<table>
<thead>
<tr>
<th>Conjunction (CONJ)</th>
<th>Addition (ADD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P )</td>
<td>( P )</td>
</tr>
<tr>
<td>( Q )</td>
<td>( P \lor Q )</td>
</tr>
<tr>
<td>( P \land Q )</td>
<td>( P )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Simplification (SIMPL)</th>
<th>Simplification (SIMPR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P \land Q )</td>
<td>( P )</td>
</tr>
<tr>
<td>( P )</td>
<td>( P )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Disjunctive Syllogism (DSL)</th>
<th>Disjunctive Syllogism (DSR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P \lor Q )</td>
<td>( P \lor Q )</td>
</tr>
<tr>
<td>( \neg Q )</td>
<td>( \neg P )</td>
</tr>
<tr>
<td>( P )</td>
<td>( Q )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Modus Ponens (MP)</th>
<th>Modus Tollens (MT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P \Rightarrow Q )</td>
<td>( P \Rightarrow Q )</td>
</tr>
<tr>
<td>( P )</td>
<td>( \neg Q )</td>
</tr>
<tr>
<td>( Q )</td>
<td>( \neg P )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constructive Dilemma (CDI)</th>
<th>Constructive Dilemma (CDII)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P \lor Q )</td>
<td>( P \lor Q )</td>
</tr>
<tr>
<td>( P \Rightarrow R )</td>
<td>( P \Rightarrow R )</td>
</tr>
<tr>
<td>( Q \Rightarrow R )</td>
<td>( Q \Rightarrow S )</td>
</tr>
<tr>
<td>( R )</td>
<td>( R \lor S )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hypothetical Syllogism (HS)</th>
<th>Reductio Ad Absurdum (RED)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P \Rightarrow Q )</td>
<td>( P \Rightarrow (Q \land \neg Q) )</td>
</tr>
<tr>
<td>( Q \Rightarrow R )</td>
<td>( \neg P )</td>
</tr>
<tr>
<td>( P \Rightarrow R )</td>
<td>( \neg P )</td>
</tr>
</tbody>
</table>
II.2. Sentential Rules of Replacement

Note: The dual colon `::` indicates that the respective formulae are logically equivalent — i.e., that either formula may be replaced by the other in any line of a derivation, OR that either may be derived from the other in a derivation.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double Negation (DN)</td>
<td>$P :: \neg P$</td>
</tr>
<tr>
<td>Commutation (COM)</td>
<td>$P \land Q :: Q \land P$</td>
</tr>
<tr>
<td></td>
<td>$P \lor Q :: Q \lor P$</td>
</tr>
<tr>
<td></td>
<td>$P \iff Q :: Q \iff P$</td>
</tr>
<tr>
<td>Transposition (TRANS)</td>
<td>$P \implies Q :: \neg Q \implies \neg P$</td>
</tr>
<tr>
<td>Equivalence (EQUIV)</td>
<td>$P \iff Q :: (P \implies Q) \land (Q \implies P)$</td>
</tr>
<tr>
<td></td>
<td>$P \iff Q :: (P \land Q) \lor (\neg P \land \neg Q)$</td>
</tr>
<tr>
<td>Implication (IMPL)</td>
<td>$P \implies Q :: \neg P \lor Q$</td>
</tr>
<tr>
<td>De Morgan (DEM)</td>
<td>$\neg P \land \neg Q :: \neg (P \lor Q)$</td>
</tr>
<tr>
<td></td>
<td>$\neg P \lor \neg Q :: \neg (P \land Q)$</td>
</tr>
<tr>
<td></td>
<td>$P \land Q :: \neg (\neg P \lor \neg Q)$</td>
</tr>
<tr>
<td></td>
<td>$P \lor Q :: \neg (\neg P \land \neg Q)$</td>
</tr>
<tr>
<td>Exportation (EXP)</td>
<td>$(P \land Q) \implies R :: P \implies (Q \implies R)$</td>
</tr>
<tr>
<td>Tautology (TAUT)</td>
<td>$P :: P \lor P$</td>
</tr>
<tr>
<td></td>
<td>$P :: P \land P$</td>
</tr>
</tbody>
</table>
II.3. Conditional Proof & Indirect Proof Strategy

Conditional Proof: HYPOTHESIS + CP

With conditional proof your aim is to derive a conditional, say: P => Q

1. Assume the ANTECEDENT of the conditional as a HYPOTHESIS.
   [Be sure to enter any premises BEFORE you enter your hypothesis!]

2. Derive the CONSEQUENT of the conditional you wish to derive.

3. DISCHARGE your hypothesis by conditionalization:
   Form the desired conditional (P => Q), citing the CP rule and the line of your hypothesis (P) through the line of the consequent (Q):

   THUS: 1 P / HYPOTHESIS
   : 2 Q / [Cite rules/lines by which derived]
   3 P => Q / CP 1-2 [Note use of dash '-']

Indirect Proof: CP + REDUCTIO

Where you wish to derive some sentence, say: P

1. Assume as a HYPOTHESIS the NEGATION of the sentence to be derived:
   1 -P / HYPOTHESIS

2. Derive a CONTRADICTION:
   2 Q & -Q / [Rule/line cit.]

3. Apply CP:
   3 -P => (Q & -Q) / CP 1-2

4. Apply the REDUCTIO rule:
   4 P / REDUCTIO, 3

   THUS: 1 -P / HYPOTHESIS
   : 2 Q & -Q / [Cite rules/lines by which derived]
   3 -P => (Q & -Q) / CP 1-2
   4 P / REDUCTIO, 3
II.4. Quantificational Rule Schemas & Summary

UI To constants: 

\[ (x)Px \quad \rightarrow \quad (x)(Px \Rightarrow Qx) \]
\[ Pa \quad \rightarrow \quad Pa \Rightarrow Qa \]

or pseudo-names: 

\[ (x)Px \quad \rightarrow \quad (x)(Px \Rightarrow Qx) \]
\[ Pt \quad \rightarrow \quad Pt \Rightarrow Qt \]

UG ONLY from pseudo-names NOT occurring in lines obtained by UI:

\[ Pt \quad \rightarrow \quad Pt \Rightarrow Qt \]
\[ (x)Px \quad \rightarrow \quad (x)(Px \Rightarrow Qx) \]

QN 

\begin{align*}
(x)Px & : : - (Ex) - Px \\
(x) - Px & : : - (Ex) Px \\
-(x)Px & : : (Ex) - Px \\
-(x)-Px & : : (Ex)Px
\end{align*}

\begin{align*}
(x)(Px & \& Qx) : : -(Ex)-(Px & Qx) \\
(x) - (Px & Qx) & : : - (Ex)(Px & Qx)
\end{align*}

Steps: 1. Change quantifier: 

\[ (x)Px \quad \rightarrow \rightarrow \quad (Ex)-Px \]

2. Negate quantifier: 

\[ -(Ex)-Px \]

3. Negate after quantifier: 

\[ -(Ex)--Px \]

4. Drop any double negation: 

\[ -(Ex)Px \]

EG From constants: 

\[ Pa \quad \rightarrow \quad Pa & Qa \]
\[ (Ex)Px \quad \rightarrow \quad (Ex)(Px & Qx) \]

or pseudo-names: 

\[ Pt \quad \rightarrow \quad Pt & Qt \]
\[ (Ex)Px \quad \rightarrow \quad (Ex)(Px & Qx) \]

EI ONLY to pseudo-names not used in a previous line:

\[ (Ex)Px \quad \rightarrow \quad (Ex)-(Px & Qx) \]
\[ Pu \quad \rightarrow \quad -(Pu & Qu) \]
II.5. Restrictions on Universal Generalization (UG)

1. You may NOT universally generalize from constants

You may universally generalize only from pseudo-names, never from constants. Generalizing from an individual case is obviously INVALID by commonsense, and proven to be so by the following argument schema and interpretation, whereby the premise (1) is obviously true but the conclusion (2) is obviously false:

Let: \( P_x = x \) was president of the U.S.; \( a = \text{Abe Lincoln} \)

1. \( Pa \) Abe Lincoln was president of the U.S.

INVALID: 2. \( (x)Px \) Everything was president of the U.S.

2. You may NOT UG from pseudo-names that occur in a line obtained by El

When a given pseudo-name (say, \( t \)) ever occurs in a line that is obtained by El (as \( t \) does in line 3 below), you may not universally generalize from it — even if the pseudo-name was itself originally obtained by Ul (as \( t \) was at line 2 below). The following derivation and interpretation shows that this move can lead from truth (Everyone has a parent) to falsity (Someone is everyone’s parent). Therefore, it is INVALID:

Let: The Domain = people; \( P_{xy} = x \) is a parent of \( y \)

1. \( (x)(Ey)Pyx \) Everyone has a parent
2. \( (Ey)Pyt \) / UI, 1
3. \( Put \) / EI, 2 [Cannot UG from \( t \)]

INVALID >> 4. \( (x)Pux \) Some person \( u \) is everyone’s parent
5. \( (Ey)(x)Pyx \) / EG, 4 Someone’s everyone’s parent
II.6. Restrictions on Existential Instantiation (EI)

1. You may NOT existentially instantiate to constants

You may existentially instantiate only to pseudo-names, never to constants: the following argument schema, existentially instantiating to a constant, is shown to be INVALID by the following interpretation, whereby the premise (1) is obviously true but the conclusion (2) is obviously false.

Let: \( Px = x \) is president of the U.S.; \( d = \) Princess Diana

\[
1 \ (Ex)Px \quad \text{Someone is president of the U.S.} \\
2 \ Pd \quad \text{Princess Diana is president of the U.S.}
\]

2. You may NOT EI to a pseudo-name already introduced in the derivation

When a given pseudo-name (say, \( t \)) has been previously introduced in a derivation (say, by UI at line 2 in the example below), you must existentially instantiate to a different pseudo-name (say, \( u \)). The following interpretation shows that existentially instantiating to a pseudo-name already introduced can lead from truth (Everyone has a parent) to falsity (Someone is his own parent). The INVALID move is at line 3; line 4 is legal by EG.

Let: The Domain = people; \( Pxy = x \) is a parent of \( y \)

\[
1 \ (x)(Ey)Pyx \quad \text{Everyone has a parent} \\
2 \ (Ey)Pyt \quad / \ UI, 1 \\
3 \ Ptt \quad \text{Someone} \ t^* \ \text{is his own parent} \\
4 \ (Ex)Pxx \quad \text{There is someone who's his own parent}
\]

* Notice that the pseudo-name \( t \) in question is in boldface above.