Smoove Proves

Propositions, Structures and the Composition of Truth

Volume I: The Logic of Propositions

by Devon "Smoove D" Belcher

Logos is born for us through the weaving together of Forms.
-Plato, The Sophist

I know that George Bealer used this as the frontispiece to Quality and Concept. But it's so beautiful, and it just seems to me perfect for a logic book, especially this logic book. So there.

This book is dedicated to two teachers that made a difference for me, Sally Aboud and Jerry DeWitt. No doubt Sally Aboud, who thought I was going to study existentialist Russian literature in college, would be shocked to find something as dry as a logic textbook dedicated to her. Perhaps she would be mollified if she considered logic the way Plato did. It's the language of the Demiurge, used to define and create the multiverse!

Many people helped with this book. In particular, I'd like to thank Heidi Metzler and Charles Kern for their help in proofreading.

Table of Contents

Chapter I
The Language of Propositional Logic (PL)

Index

I) Introduction

In this chapter, we'll introduce the language PL (Propositional Logic). Our treatment of the language will be intended to achieve two related goals: First, we want to create a language that allows us to model and understand natural language and some of the important features of natural languages, with an eye towards explaining natural language features like *learnability*; and second, we want a language that will help us to model and explain the central logical concepts, such as truth, validity, and consistency.

One of the central goals of language is to say stuff about stuff. Our most basic sentences are those sentences that say stuff about stuff, or, more precisely, predicate characteristics of subjects. Once we have a sentence that "says stuff about stuff" we have a unit of meaning that can be true or false. A word that simply refers to an object (without saying anything of that object) isn't true or false; a word that simply expresses a property or a characteristic isn't true or false. If I were to walk into the room and confidently assert 'Albert!', you would probably say '. . .Albert *what*"? and wait for me to complete the sentence. The name alone isn't true or false, it simply refers to an individual. But once I say 'Albert is a squirrel', you can either sagely nod in agreement, or vigorously object to what I've said. I will have asserted a sentence that can bear a truth-value. And once I have asserted sentences that have truth-values, we will be able to consider the possible combinations of those truth values, and relations between those truth values - the proper subject matter of logic.

Let us now build up our language PL from the bottom.

II) The Syntax of PL

In what follows, I've made a rather deliberate attempt to describe the various symbols of PL with no reference to their meaning. We are trying to describe the purely structural elements of PL, that is the orthography, and the rules for combining the symbols of the language into more complicated symbols. This is the role of syntax, which gives us only an idea of how to generate strings of symbols, and not how to calculate their meaning or truth. In the next section ('Semantics of PL'), we'll talk about how to assign meaning and truth to these strings, but for now, I think it will be helpful to think of the symbols of PL as being simply strings of symbols formed according to completely arbitrary conventions. I have mixed in some semantics in what follows, but it isn't really necessary to discuss the syntax. It's just there to help the reader see how the syntax of PL really relates to the syntax of languages they are more familiar with.

Individual Terms

The first category of PL is Individual Terms. Within the category of individual terms we have individual constants, and variables. However, we will not deal with variables until chapter 7, so, for all practical purposes, individual constants are the only kind of individual terms right now. I'll also call these *names* quite often.

In PL, individual terms are indicated by lower-case letters. In most logic textbooks, individual terms are single lower-case letters, such as 'a', 'b', 'c', . . . Sometimes these are subscripted, as in 'a_1', 'b_3', 'c_2' . . .The subscripts are a matter of convenience, in case you run out of letters, or perhaps you would like to have two individual terms that you'd like to use the letter 'a' for.

I think that ease of use, intuitiveness, and familiarity are all useful characteristics of a language, however, and thus in PL, I see no reason to limit ourselves to using only single lowercase letters as individual constants. Thus, PL will also count whole words such as 'albert', 'bertha', 'hannah', as individual constants. The central orthographic restriction is that they must all be lowercase. Notice that this is directly opposed to the English convention of capitalizing names; it's just because logicians have to make things difficult so we have something to mark you wrong on.

Individual terms in PL roughly correspond to the natural language categories of *subject* or *noun phrase* (specifically, *proper names*). When translating, there's sometimes a bit of an art in deciding what English phrases are to be translated as individual constants, and what should be treated as predicates; we'll discuss that in a minute. In PL, we will make three simplifying assumptions about the use of individual constants:

1) Every individual constant has a referent; i.e., every individual constant refers to some individual. This might be a concrete individual like a person or a place, or it might be a more abstract individual like a time (*the day of infamy*, or Dec 7, 1942) or a number (7). This convention of PL is not accepted in natural languages; we often use English names such as 'Santa' or 'Sherlock Holmes' that refer to no individual (although they may have some concept associated with them).

2) Every individual constant refers to at most one individual. There is no ambiguity in PL; you may not use the individual constant 'john' to refer to two distinct people (if you wish to use the symbol 'john' in such a way, you must use subscripts to for two distinct constants such as '$john_1$' and '$john_2$'). Arguably, this convention of PL is also not accepted in natural languages; we often (arguably) use a single word 'John' to refer to multiple people in English.[1] Again, we adopt this as a simplification.

3) Every individual may be referred to by more than one individual constant; any object may have many names. This *is* a prominent feature of natural languages; 'Sam Clemens' and 'Mark Twain' refer to the same author, 'Snoop Dogg' and 'Calvin Broadus' refer to the same rapper, and so on. This is because languages grow up out of different customs and uses; people in one area might use one name for the famed rapper, while people in another area create a different symbol to refer to him. Both traditions then become part of English. PL models this; a single individual might be referred to by both the constants 'a' and 'b'.

These three conventions are just that - simplifying conventions. Obviously we'd like logic to be able to model all the features of language, such as arguments about things that don't exist (but that might exist, or will exist in the future, or used to exist). For now, however, I think it will be fruitful to start simply, and try to deal with those complicating features at some later point.

1Kaplan

Predicates

In PL, predicate terms are capital letters, such as ‘P’, ‘Q’, and so on. These may be subscripted, as with individual terms. Again, the usual convention has been to use only capital letters, but once more, I prefer to allow the use of complete words, such as ‘Pirate’, ‘Squirrel’, ‘Runs’, ‘Big’, and so on. Again, if you use complete words, you must capitalize the first letter (and again, that’s sort of the opposite of the English convention).

In natural language, predicates roughly correspond to the categories of *predicate* (duh), *transitive verb / transitive verb phrase*, and *adjective.* We use some simplifying conventions again, namely,

1) Any predicate term expresses at most one property (we’ll talk about this later, in the section on Semantics). Again, there is no ambiguity in PL; homonymous English predicate terms such as ‘Plane’ (flat surface) and ‘Plane’ (flying vehicle) must be symbolized distinctly in PL, such as ‘Plane1' and ‘Plane2'.

2) Any Property might be expressed by more than one predicate term; e.g., ‘Masticates’ and ‘Chews’ express the same action or property

Relation Terms

In PL, relation terms are treated exactly like predicates. In fact, it is ordinary to treat predicates as simply a special case of relation terms. Relation terms are capital letters or words beginning with capital letters, such as ‘R’, ‘S’, ‘Leftof’, ‘Largerthan’, and so on. In natural language, relation terms roughly correspond to relations - words that express relations between objects - or *transitive verbs* (verbs expressing actions that relate multiple individuals, such as ‘hits’ or ‘loves’.)

Relation Terms (and the relations they express) have an *arity*, or a number of individuals they are supposed to relate. The arity of a relation term is essential to it - two very closely related relation terms with different arities are nonetheless wholly distinct relation terms (and the relations they express are treated as wholly distinct relations). We’ll talk about this more in the Semantics section. The arity of the relation term ‘Loves’ in English, for example, is 2 - loving requires a lover and a person loved. The arity of ‘Larger’ in English is 2 - it relates some individual to the individual it is larger than. The arity of ‘Between’ is at least 3 - it requires an individual, and at least two individuals that surround it - you can’t be between just one thing. We’ll call a relation (term) with an arity of two, two-place, or binary; a relation (term) with an arity of three, three-place, or trinary; and so on. We'll occasionally use a superscripted number to indicate the arity of a predicate or relation term, as in F^2, $Loves^2$ or $Between^3$.

In PL, the arity of a relation term is whatever we decide it is - it’s our language, after all, and as we introduce new relation terms into the language, we define them partly by giving their arity. Of course, if you want a relation term of PL to be the translation of some English word, you’d better make sure they have the same arity!

As I suggested, we can treat predicates as a subset of relation terms. They are simply relation terms that have an arity of 1.

The Symbols of PL

Syntactic Category	Term	Predicate	Relation Symbol
NL Analogue	Proper names	Intransitive verbs, adjectives	Relation words, transitive verbs
Orthography	Lowercase letters, lowercase words 'a', 'b', 'albert', 'hannah'	Uppercase letters, words beginning with uppercase letters 'F', 'G', 'Pirate', 'Squirrel'	Uppercase letters, words beginning with uppercase letters 'R', 'S', 'Bigger', 'Pillages'
Refers to or Expresses	Individuals	Properties of individuals or sets of individuals	Relations between individuals

Language and Metalanguage

So far, we have a language in which we can talk about individuals, the properties or characteristics those individuals have, and the relations those individuals stand in (although, we haven’t yet explained how to put any of those together). Let’s call this our *object language*, since it basically lets us talk about objects. We’ll abbreviate it PL_0, when we need to explicitly distinguish it. As useful as an object language is, we will often need a richer language. In logic, we will want to talk about concepts like truth, validity, consistency, and so on. But truth is not ‘part of the world’. Truth is not a property of individuals, or even a property of ‘facts’. Individuals simply *are*, or *are not*; and facts *hold*, or *do not hold*. It would sound silly to say that, for example, Albert was true, or that the fact that Albert is a squirrel was a true fact. Rather, it is something like our representations of the world that are true or false. It is our sentences that are true or false, or perhaps the thoughts embodied by those sentences that are true or false, or perhaps “propositions” about the world that are true or false. And intuitively, our sentences are true or false insofar as they “mirror” the actual facts of the world in a certain way. So when we talk about concepts like truth and validity, we want some way to talk about not just the world, but also a way to talk about our representations of the world (in this case, we want a way to talk about the sentences of PL_0, as well as a way to talk about the facts of the world expressed by the sentences of PL_0). We also want a way to talk about this new relation of “expressing” and “mirroring the facts”. To this end, we will create a new language, one specifically for the purpose of talking about not the “object world”, but a language for talking about language. We can call this a Metalanguage. We will abbreviate this PL_1, when we need to distinguish it.

What are the things that PL_1 talks about? It doesn't talk about ordinary individuals like Albert, or the property of being a pirate. Rather, it talks about words. It talks about the word ‘albert’, or the predicate word ‘Pirate’. In natural languages like English, the way we signal that we are talking about words (as opposed to objects) is to use single quotation marks (double quotation marks are “scare quotes”, used to distinguish some kind of loose usage). So the expressions of the metalanguage PL_1 will be just the expressions of PL_0, but in quotes, such as ‘Pirate’, ‘albert’, ‘LeftOf’, and so on.

In English, this is also known as a *use-mention* distinction. We *use* words to refer to individuals, and the use of a word is just marked by the un-quoted presence of that word. On the other hand, we sometimes *mention* words when we want to talk about the word itself, and not what it represents.

Quotes are used to indicate *mentionings* of words. For example, we say that there are four letters in 'cats', but there are no letters at all in cats (what with cats being composed of flesh and blood, and not alphabetical symbols).

In our language PL_0, we will eventually want some way to talk about individuals in a very general way. We'll want a way to say that every individual has a certain property, or some unknown individual has a certain property. For that, we'll use variables such as 'x' and 'y'. These will be replacements for individual constants. As of now, variables aren't a part of the language PL_0. But we do already need a way to refer not just to this particular sentence or that particular name, but sentences and names in a very general way (for example, 'any sentence that is true is also. . .'). So to that end, we will now introduce variables into the metalanguage PL1, what we'll call *metalanguage* or *metalingusitc variables*.

The first type of metalanguage variables will be variables that range over the individual constants. We will always use lower-case Greek letters for these, such as $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. For convenience, we'll often use subscripts, such as

$\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$.

For example, a particular PL_0 instance of the metalanguage string

$(\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \boldsymbol{\alpha}_3)$

might be the string of terms

(albert, hannah, dave), or perhaps (hannah, dave, sammy).

Notice that these metalanguage strings take as their values ("range over") expressions of PL_0 (i.e., sentences), and not over objects in the world!

It will also be convenient to use yet another variable, a numeric variable n, in conjunction with subscripted variables, to create strings like

α_n.

This will allow us to talk about, for example, strings of exactly n names, by using the notation

$\alpha_1, \ldots \alpha_n$

Which we use to mean something like 'some string of exactly n names, it doesn't matter what they are'. This kind of expression will prove very useful in the next section, in defining how to form "arbitrarily large atomic sentences", i.e., atomic sentences of any size (and eventually, truth for arbitrarily large sentences).

The next type of metalanguage variable we will introduce will be variables ranging over predicates and relation terms. We'll use uppercase Greek letters for this purpose, typically,

φ and ψ

(if we need more, we'll typically just use subscripts to distinguish them such as φ_2, ψ_3 and so on). For example, a potential instance of φ might be the predicate term 'Pirate' (and *not* the property of *being a pirate*!), and another instance might be the relation term 'LeftOf'. Again, we'll use the numeric variable n in a phrase like 'a predicate or relation term φ with an arity of n', to indicate any n-place predicate/ relation term.

The Symbols of PL_1 (PL's Metalanguage)

Category	Term Variables	Predicate/Relation Variables	Punctuation
NL Analogue	None	None	Punctuation
Orthography	Lowercase Greek α, α_1, α_2	Uppercase Greek φ,ψ	Quotes ', '
Role/ Function	Ranges over terms of PL_0	Ranges over predicates and relation terms of PL_0	Creates names for constructions of PL_0

Atomic Sentences

The central idea of a language is that we want to say stuff about stuff. So every sentence should contain stuff that has stuff said about it - a subject- and stuff we say about that stuff - a predicate. Our most basic rule of PL, then, is this:

If φ is a predicate, and α_1 is exactly one name, then
$\varphi(\alpha_1)$
is an atomic sentence.

Really, this is just a fancy way of saying that you can glom a predicate in front of a name, and thus make a sentence. Some sentences of PL might be: 'P(a)' (sometimes we'll say this like "P of a"), 'Hamster(hannah)', 'Sails(albert)', and so on. Pretty simple, eh?

Now let's extend that idea to sentences involving relation terms:

If φ is a relation term with an arity of 2, and α_1 and α_2 are exactly two names, then
$\varphi(\alpha_{1,}\alpha_2)$
is an atomic sentence.

Again, this is just a fancy way of saying that you can glom a two-place relation term on in front of two names and have a sentence. Continuing on,

If φ is a relation term with an arity of 3, and $\alpha_{1,}\alpha_2$ and α_3 are exactly three names, then
$\varphi(\alpha_{1,}\alpha_{2,}\ \alpha_3)$
is an atomic sentence.

You can glom a three-place relation term in front of three names and have a sentence. Obviously this could go on forever, but all these rules are nearly the same, so it's possible to introduce one overall rule to capture all possible atomic sentences:

Atomic sentence formation rule for PL
If φ is a relation term with an arity of n, and $\alpha_1 \ldots \alpha_n$ are exactly *n* names, then
$\varphi(\alpha_{1,}\alpha_{2,}\ldots\alpha_n)$
is an atomic sentence. Nothing else is an atomic sentence.

There, we now have a single rule that encompasses all possible atomic sentences of PL_0! Since we've already decided to treat predicates as one-place sentences, this covers all predicative sentences of PL as well as the relational sentences. A string of symbols that can be built this way is said to be *well-formed* (it's sometimes called a *well-formed formula* or *wff*), and a string of symbols that cannot be generated this way is said to be *ill-formed* or *gibberish*.

Now that we have sentences, we should introduce a way to talk about sentences generally (as opposed to talking about the facts represented by those sentences). In other words, we shall introduce metalanguage variables that range over sentences, or *sentential variables*. We'll use lower-case Greek letters for this, again, such as φ and ψ. (These variables also range over predicates and relation terms, so if there is a risk of confusion, we'll explicitly say something like 'a sentence φ' or 'a relation φ'. Thus, PL_0 instances of φ and ψ might be the sentences:

Pirate(albert), LeftOf(hannah, mary),

and so on. (Note again, that the instances of φ and ψ are sentences, and not the facts referred to by those sentences!)

Sentence Letters

At this point, it will be useful to introduce sentence letters. Sometimes you really don't care what the sentence 'Pirate(albert)' means, because you are interested in what goes on above the subject/ predicate level. For example, think of the (kind of uninteresting) argument:

Albert is a pirate, and Albert is a squirrel

____

Thus Albert is a pirate

The argument will be valid no matter what 'pirates' are, or who the name 'Albert' refers to. So we'll sometimes use capital letters such as 'A', 'B', 'C' and so on to be abbreviations of whole sentences. This is really just a space-saving device, nothing more. How can you tell if 'A' is a sentence letter or a predicate (after all, both use capital letters)? You'll just have to use the context: If you see a string like 'A(a)', you should assume that 'A' is a predicate; if you see a string like 'A v B', which wouldn't makes sense if ‘A’ was a predicate, it’s a sentence letter.

More Symbols of PL_1 (Metalanguage)

Category	Sentential Variables
NL Analogue	None
Orthography	Uppercase Greek φ,ψ
Role/ Function	Ranges over WFFs of PL

Side Note: Predication. Identity, and the ambiguity of 'is'.

When we put together a predicate/ relation term with a name, we are making a predication. In English, we use the copula ‘is’ to do this (or sometimes ‘has’). We say things like ‘Albert is a pirate’, ‘Hannah is left of Sammy’, and so on. We call this the ‘is’ of predication. Of course, we don’t mean that Albert is identical to the property of being a pirate. That’s impossible; after all, Albert is an individual, and the property of being a pirate is an abstract characteristic of things. So the word ‘is’, in English, is ambiguous between the ‘is’ of predication and the ‘is’ of identity. No such ambiguity exists in PL: we indicate predication simply by sticking the predicate term next to the name(s). There is no further symbol like ‘is’ to express the predication relation. If we want to use the ‘is’ of identity, we can simply use the relation term ‘=’, which is just a (special) binary relation.

Connectives

Connectives are symbols used to connect sentences. They roughly correspond to, well, connectives, in natural language. The connectives we will use are:

‘$\rightarrow$’ (the conditional), ‘$\wedge$’ (conjunction), ‘$\vee$’ (disjunction), ‘$\leftrightarrow$’ (biconditional) and ‘$\neg$’(negation).

We’re not going to talk about what those mean at all here, but we’ll tell you how to use them to connect sentences.

More symbols of PL

Syntactic Category	Formula, WFF	Punctuation	Connectives
NL Analogue	Sentence	Punctuation	Connectives
Orthography	Uppercase letters	'(', ')'	'↔', '→', '∧', '¬', '∨'
Refers to or expresses	Truth Values	N/A	N/A (?)

Molecular or Compound Sentences

Suppose you have two sentences, say, 'Pirate(albert)' and 'Hamster(hannah)'. You can connect those two sentences by putting '→', '∧', '∨' or '↔' between them and putting parentheses around the whole thing, e.g.

(Pirate(albert) → Hamster(hannah)) or

(Hamster(hannah) ∨ Pirate(albert))

The atomic sentences don't have to be in any particular order, although the order will often make a big difference in what the sentence means. But either order will be an acceptable sentence of PL.

You can also make a sentence by slapping '¬' in front of another sentence, with no parentheses. For example, ¬Hamster(hannah).

We can once again give a single general rule to cover the formation of compound sentences, as follows:

Compound Sentence formation rule for PL
If φ and ψ are sentences, then

$(\varphi \to \psi)$
$(\varphi \wedge \psi)$
$(\varphi \vee \psi)$
$(\varphi \leftrightarrow \psi)$
$\neg\varphi$

are all sentences of PL. Nothing else is a (Compound) sentence of PL.

The parentheses involved in the first four clauses are important! They will serve to disambiguate many sentences. You'll find that they work much like they work in algebra or arithmetic to indicate the order of operations. Any time two sentences get connected to form a compound sentence, you *must* surround the new sentence with parentheses. The connective is then known as the *central connective* of that sentence. The central connective of the sentence is the connective associated with the outermost pair of parentheses in that sentence. If there is a negation outside of the outermost set of parentheses, it is the central connective of that sentence. A sentence whose central connective is '$\to$' is called a conditional sentence or a conditional. A sentence whose central connective is '$\vee$' is called a disjunctive sentence or a disjunction, and its parts are disjuncts. A sentence whose central connective is '$\wedge$' is called a conjunctive sentence or a conjunction, and its parts are called conjuncts. A sentence whose central connective is '$\leftrightarrow$' is called a biconditional, and a sentence whose central connective is '$\neg$' is called a negation. The sentences connected by a conjunction are called *conjuncts*, and the sentences connected by a disjunction are called *disjuncts*. The first part of a conditional sentence is called the *antecedent*, and the last part is called the *consequent*.

Connectives and Sentences

Sometimes in English we say things like ‘Albert and Hannah are pirates’ or ‘Albert is a cad and a blackguard’. It’s tempting to want to create analogous sentences in PL, such as

Pirate(albert ∧ hannah) and Cad ∧ Blackguard(albert).

However, there is no rule that allows this in PL! Notice that the only rules that mention the connective ∧ (or any other connective, for that matter) specify that you can glom two sentences together with a connective. You can’t glom predicates or names together in this way. If you want to capture the English sentences, you have to be a bit creative, and use the equivalent

Pirate(albert)∧ Pirate(hannah) or Cad(albert) ∧ Blackguard(albert).

This is simply an arbitrary restriction which is in place solely to frustrate you. Actually, no it isn’t, it’s there for a very good reason, which we will discuss when we talk about the semantics of compound sentences.

Figure 1
A Pirate Squirrel

Recursion

You might notice that in our sentence formation rule, the sentences φ and ψ we combine together (the "inputs" to the rule) don't have to be atomic sentences. This is really the key to the whole shebang! Suppose we start with two atomic sentences, 'Pirate(jenny)' and 'Squirrel(albert)'. We'll take advantage of the category of sentence letters to abbreviate them 'P' and 'S', respectively. So, we can form the compounds

'(P → S)', '(S → P)', '(P ∧ S)', '(S ∧ P)','(P ∨ S)', '(S ∨ P)', '(P ↔ S)', '(S ↔ P)', '¬P' and '¬S.

right away.[2]

But then, the rule tells us that these are sentences too, right? So, we can combine these with each other, and with the atomic sentences, to form more complex sentences, such as

'**(**(P → S) **∧** P**)**', '**(**(S ∨ P) **∨** (S ∧ P)**)**', '**¬(**P → S**)**', and so on.

2Moreover, since p and q are variables that range over all possible sentences, there's no reason they can't both take the same value, say 'P', and generate sentences like '(P → P)', '(S → S)', '(P ∧ P)', '(S ∧ S)','(P ∨ P)', '(S ∨ S)', '(P ↔ P)', '(S ↔ S)'. With the large number of possible compounds even at this early stage, you can see why we use abbreviated sentence letters!

I've put the outermost parentheses in boldface, as well as the connective those parentheses are introduced with, just to make them easier to see (I won't continue this). In the case of '¬(P → S)', the outermost parentheses are associated with the conditional, but the central connective is the negation. This sentence is built up by connecting P and S with a conditional, and then negating that conditional. We say the negation has *scope* over the entire sentence. This is distinguished from a sentence like '(¬P → S)', which is the result of applying the negation symbol to the sentence 'P', and forming a conditional out of the sentences '¬P' and 'S'. In this case, the negation applies only to 'P', or has scope over only P, and it doesn't apply to the whole conditional sentence.

In short, every time we connect two sentences with a connective (or negate one with a negation) we have formed a new sentence, and that new sentence can then be used as an 'input' into the sentence formation rule, to form even larger sentences. There is no principled limit to this, and we could go on to form an infinite number of sentences from a very small number of names, predicates, and merely five connectives! Nonetheless, each of these sentences would have a common structure given by our rule.

In this way, we can explain how it is that people, with finite minds, can come to understand how to create an infinite number of sentences. And that really goes a long way towards understanding how it is that humans can learn languages.

III) The Semantics of PL

Alas, all we have done is tell you which strings of symbols are sentences of PL, and which are not; we haven't said what they mean, or how to discover if they are true, or anything interesting. You could now spit out sentences of PL until the cows come home, and have no idea what you were doing, and no understanding of what you'd said. So now let's look at what our words and sentences mean.

Individual Terms (Or, more specifically, individual constants / names).

Names, as we've already said, refer, and they refer to individual things. They refer to concrete things, like people or objects; they refer to more abstract things, like places (Boulder, Colorado) or times (January 23, 2006). They might even refer to very abstract individuals; for example '7' is a symbol that refers to the number seven. There are, of course, lots and lots of individuals we haven't gotten around to naming.

They don't refer by invoking any description of the thing they refer to, they refer directly. For example, the expression 'Prince Charles' might be a proper name. But the fact that he is a prince is not part of the content of this name. From the sentence 'Prince Charles is a dork', one cannot infer that Charles is a prince. This is in accordance with natural language. In the most extreme (and humorous) case, the name 'Holy Roman Empire' refers to a state that is neither Holy, nor Roman, nor an Empire! (Of course, one might also have the predicate 'Prince' or 'Is a Prince', and assert that 'Prince Charles really is a Prince', or 'Prince(princecharles)' in PL).

How can you tell when a word like 'Prince' is part of a proper name, or is a distinct predicate? In English, a genuine predicate usually has some form of the copula 'is' associated with it, as in, 'Charles is a prince'. Often, you have to use your intuitions as a speaker. Sometimes, it's helpful to use a thought-experiment like this: *could* the person named by 'Prince Charles' have failed to be a prince? If so, then the characteristic of being a prince cannot be part of the meaning of the expression ‘Prince Charles’.

Predicates and Relation terms

Predicate and relation terms express properties and relations. Properties[3] are things like characteristics, or aspects of objects, such as Greenness, Largeness, Pirattitude (the property of being a pirate). Predicate terms also express actions that individuals undertake, like Runs, Sleeps, or Sails. Finally, predicates express classes or categories, like Pirate or Squirrel. Predicates thus function much like adjectives and (intransitive) verbs in natural language. They're used to classify or characterize individuals in some way or another. Individuals are sometimes said to *instantiate* or *be instances of* properties

Relation terms express relations (perhaps surprisingly!) Relations are like properties, but instead of classifying individuals, they *hold between* two or more individuals. You can think of them as classifying groups of individuals. Examples of relations are LeftOf (the leftness relation), Plunders, Loves, Between, and so on. Plunders, for example, classifies pairs of individuals (the plunderer and the plunderee). LeftOf classifies pairs of individuals where the first in the pair is left of the second in the pair. The Betweenness relation classifies sets of three (or more) individuals.

Properties (and relations) have some special qualities. More than one individual can have the same property. Both Blackbeard and Jack Sparrow were pirates. And even if a property is so unique that only one individual could ever have it, (say, the property of being sexiest pirate ever), at least in principle it could have been any individual. It might have been Jack Sparrow, and it might have been Captain Blood. But this is not true of individuals. Only one individual, in principle, could have been Captain

3 I don't mean to commit to any particular metaphysics here. In particular, I'm not trying to say logic commits you to any realism about properties, or that properties are universals, or anything like that. If you like, think of predicates as expressing classes or sets of individuals, as opposed to individuals themselves. See Appendix A of this chapter.

Blood[4]. Only one individual could have been Ronald Reagan. You might think that someone else could have had the *properties* that Reagan had, but they wouldn't thereby *be* Reagan.

There's one final note. As we mentioned before, the arity of a predicate or relation term is essential to it, and the arity of a property or relation is also essential to it. So suppose you had the relation of Loving^2, that held between two individuals, and the relation of Loving^3, that held between three individuals. No matter how similar the love seems, these are not the same relation! Why should this be? Well, the dominant tradition in semantics has been to treat predicates and relations as *functions* (see appendix 3), and functions are best described as having their arity essentially. This shouldn't be an issue for us, but it does suggest further areas of inquiry in formal semantics. Perhaps there is the seed of an argument here, against treating properties as functions. At any rate, this won't be a concern for us in this course.

Sentences

The predicates, relation terms and names or individual terms of PL can be combined together, as we saw, to form sentences (and formulas). What do sentences refer to? It's tempting to say that sentences refer to something like states of affairs or things like that. Or perhaps it's more natural to think that sentences denote or refer to *what they mean* - things like "propositions". But we're primarily interested in truth, and the relationships between truth values, so in formal logic it has been agreed that sentences refer to, or denote, the two truth values - true and false. What's important and interesting about this is the way in which the elements of PL combine to produce these sentences and their truth values. This is the topic of the next section.

---

4 Things may be a bit complicated by the fact that these are fictional characters. Johnny Depp *portrayed* Jack Sparrow, but it could have been (ugh!) Orlando Bloom. Portraying is not the same as Being, though.

Section IV) Truth

What is Truth? asks Pontius Pilate. As a wealthy and educated Roman, Pilate would probably have studied Aristotle, and so his question must surely have been rhetorical. He was simply awaiting the response "Truth is saying of what is, that it is; and of what is not, that it is not". This is hardly mind-blowing! There's some hidden content there, though. Truth seems to be a sort of property that things (things that are said, like sentences or maybe propositions) have when they stand in a certain relationship to the world. Names or individuals cannot be true, and nor can predicates or relation terms. If I simply asserted 'Albert!' or 'Squirrel!', that assertion couldn't be true or false. It's not even clear that it counts as an assertion. After all, what have I conveyed? So if truth is a sort of property that sentences or propositions have, it falls to us to give the conditions under which sentences or propositions have that property. Henceforth, I'll just talk about sentences as being true; but again, I don't mean to commit to the age - old question about the primary bearers of truth. We could just as easily talk about the truth of properties, and then say that certain sentences were derivatively true, insofar as they expressed true propositions.

We should also be careful in talking about truth as a property. It certainly seems like a property of sentences - after all, many sentences can be true, and a sentence can be true at one point and false later - but treating Truth exactly like a property can lead us to paradox later. Still, for now, this is a good way to metaphorically understand truth.

What, then, are the circumstances under which a sentence is true? What we'll do now is to consider all the possible sentences in our language, and give truth conditions for each. Although this sounds like an impossible task, since there are an infinite number of sentences, there are only a finite (and small!) number of sentence forms. A sentence of PL is either an atomic sentence, or a compound; and a compound is either a conjunction, disjunction, conditional, biconditional or a negation.

Truth rule for atomic sentences

Let's start with the simplest possible sentence - the result of putting a predicate in front of an individual term. Say, Pirate(albert). What would it take for this to be true? Don't knock yourself out - Albert would have to be a pirate.

This actually obscures the fact that we're talking about two different things here - in Pirate(albert) we're talking about a sentence, and in the truth conditions, we're talking about actual pirates. So let's try to be more explicit about that in a proto-definition:

A sentence of the form 'Pirate(albert)' is true just in case Albert is a pirate

This seems perhaps adequate for that sentence, but it doesn't seem to be very generalizable. Moreover, it's only true in PL. There could be other languages where 'Pirate' expressed the property of being a merchantman. And so the sentence 'Pirate(albert)' would be false in that language! So the above truth conditions can't be right for that other language. If truth is a property of sentences, then truth conditions must be relative to the language. So it'd be better to say:

A sentence of the form 'Pirate(albert)' is true-in-PL just in case Albert is a pirate, or even
A sentence of the form 'Pirate(albert)' is true_{PL} just in case Albert is a pirate

We can also be more explicit by talking about the parts of the sentence, and what they refer to or express. For example:

A sentence of the form 'Pirate(albert)' is true_{PL} just in case the individual referred to by the individual term 'albert' has the property expressed by the predicate 'Pirate'

Notice that now we've characterized the truth conditions of the sentences 'Pirate(albert)' completely independently of what the individual words mean! If, for example, 'Pirate' had expressed the property of being a squirrel, then the sentence 'Pirate(albert)' would have been true just in case albert had the property expressed by the predicate 'Pirate' - namely the property of being a squirrel! But this is intuitive - if we used words differently, we'd say different things, right? And if we said different things, they might not have the same truth value as they actually do. If by 'they will throw rose petals at our feet', Bush had meant 'We will be stuck in an intractable quagmire', he would have uttered a truth, for example.

The above truth condition takes a step towards defining truth solely in terms of other semantic relations (*expressing* and *referring*). We can use this, together with our metalanguage, to define truth for *any* subject-predicate sentence of PL:

> A sentence of the form $\varphi(\alpha)$ is true_{PL} iff the individual referred to by α has the property expressed by φ.

What about atomic sentences involving relations? Consider the sentence 'Albert plunders Tortuga', or, in PL, 'Plunders(a. t)'. By our same process of reasoning, it's true just in case Albert plunders Tortuga, or the thing referred to by 'albert' stands in the relation expressed by the relation-term 'Plunders' to the thing referred to by the name 'tortuga' (in that order). Or:

> A sentence of the form $\varphi\ (\alpha_1, \alpha_2)$ is true_{PL} iff the individual referred to by α_1 stands in the relation expressed by φ, to the individual referred to by α_2.

What about 3-place relations? That really goes the same way:

> A sentence of the form $\varphi\ (\alpha_1, \alpha_2, \alpha_3)$ is true_{PL} iff the individual referred to by α_1 stands in the relation expressed by φ, to the individuals referred to by α_2 and α_3.

In fact, given what we said about arity earlier, we can wrap all of these definitions up into one definition:

> Truth Rule for Atomic Sentences of PL
>
> A sentence of the form $\varphi^n\ (\alpha_1, \alpha_2, \ldots \alpha_n)$ is true_{PL} iff the individual referred to by α_1 stands in the relation expressed by φ^n, to the individuals referred to by $\alpha_2 \ldots \alpha_3$.

Notice how closely this mirrors the syntactic rules for atomic sentences of PL! This is what ensures that we have a truth definition for every single possible atomic sentence of PL! Also notice that from our definition of truth, we cannot tell what sentences are true. But this is natural. To know a language is to know how to use it, that is, to know the conditions under which sentences can be truthfully asserted (maybe it takes more to know a language, but at least it takes this much). Suppose I said 'Albert is a pirate', and you asked, 'wait a minute, what kind of circumstances would make that sentence true'? If I had no answer, you'd say 'Well, you don't know what you're talking about, literally'! On the other hand, you can know what the sentence says, without knowing whether Albert really *is* a pirate. Knowing a language isn't the same as knowing the extralinguistic facts that the language talks about. I know full well what it means to say 'The Flash is faster than Superman', but I have no idea if it's true. What we've given is a definition of Truth in general, and not a list of truths. You can understand what truth is, without knowing the truths, just like you can know what Love is, without knowing who loves you. Alas.

Truth for compound sentences

Just like we gave truth rules for atomic sentence forms, we'll try to give truth rules for compound sentence forms. And we'll note all the same general points - our definition will not tell us whether (most) sentences *are* true or false, but simply give the conditions under which they are true or false.

Our rules will be compositional. That is, the truth of the whole will always be composed of the truth of the parts and nothing more. Ultimately those parts will be atomic sentences, and we've already described the truth of those parts in terms of their semantic properties - the referents of the parts.

Conjunctions '∧'

We've already hinted that '∧', conjunction, is supposed to be like the English 'and'. Under what circumstances are conjunctions true? Consider a sentence like 'Albert is a pirate and Albert is a squirrel'. What if Albert was a pirate, but not a squirrel? Then I lied! 'Albert is a pirate and Albert is a squirrel' is false! What if he wasn't a squirrel? Then I also lied! If I assert some sentence of the form 'A ∧ B', then I'm asserting that both conjuncts, A and B, are true.

So we can give a truth definition for PL conjunctions:

> Truth for Conjunction
>
> A sentence of the form $\varphi \wedge \psi$ is true_{PL} iff φ is true_{PL} and ψ is true_{PL}.

Disjunctions '∨'

We've already hinted that '∨', disjunction, is supposed to be like the English 'or'. But there's a bit of unclarity in the English, so we'll have to be more precise. Suppose we have a sentence like 'Reagan is a Republican or a Democrat', 'Repub(reagan) ∨ Dem(reagan)'. Just as long as he's one or the other, the sentence is true. The only way it could be false would be if he wasn't a Republican *and* he wasn't a Democrat. (Bull Moose, maybe?). So if both disjuncts are false, the disjunction is false. And if one or the other disjunct is true, the disjunction is true.

But political lines have been blurred lately. What about Clinton? Many say he (or she!) is a Democrat in name only, and in truth either of them seems as much a Republican as a Democrat. Perhaps you find this controversial. But consider Joe Lieberman. He's not a member of the Democratic party, he's an independent. Still, he considers himself a Republican in some speeches. Suppose he's both a Democrat and a Republican.

Or perhaps you think even this is controversial, and the two political affiliations preclude each other. Then consider this: 'Albert is either a buccaneer or a freebooter'. Surely, he might be *both*! What then of the disjunction? Would you respond, "Hah, you're wrong! He's both a buccaneer *and* a freebooter!' That would make no sense! Still, it seems like in *some* circumstances, a disjunction excludes the possibility that both disjunctions are true. We'll call this *exclusive* disjunction. For example, when the waitress asks if you want soup *or* salad, it's probably intended as an exclusive disjunction - you don't get the option of both. Sometimes we use inflection to indicate we mean this, or other contextual features of our conversation. But for our purposes, the wedge - '$\vee$' - will express *inclusive* disjunction. Inclusive disjunction includes the possibility that both disjuncts are true. So a disjunction will be true if one or both disjuncts are true. So our definition will be:

> **Truth for Disjunction**
> A sentence of the form $\varphi \vee \psi$ is true_{PL} iff either φ is true_{PL}, or ψ is true_{PL}, or both are true.

Negation

The Wayne and Garth operator

The Tilde, '$\neg$', is like a big 'NOT!' appended to the beginning of a sentence. Much like when Wayne and Garth of 'Wayne's World' say 'That Kenny G album totally shreds. . .NOT!', the negation operator has the effect of forming a sentence with the opposite truth value from the sentence it negates. Even though it's practically unthinkable that Kenny G has a shredding album, if he did, the above sentence would be false. The main syntactic difference is that '$\neg$' goes in front of a sentence and 'NOT!' follows it, of course. If some sentence P is true, then $\neg P$ is false, and vice versa. So:

Truth for negation

A sentence of the form $\neg\varphi$ is $true_{PL}$ iff φ is $False_{PL}$.

Conditionals

The conditional '$\rightarrow$' mimics the phrase 'if. . .then'. But there are some important differences!

A conditional like

Pirate(albert) $\rightarrow$ Bloodthirsty(albert)

says that if Albert is a pirate, then he is bloodthirsty. It doesn't say anything about temporal ordering; i.e., it doesn't say that his being a pirate happens before his being bloodthirsty. And it doesn't say that his being a Pirate *causes* him to be bloodthirsty, so the conditional isn't meant to be any sort of causal relation. Another example that illustrates this point is:

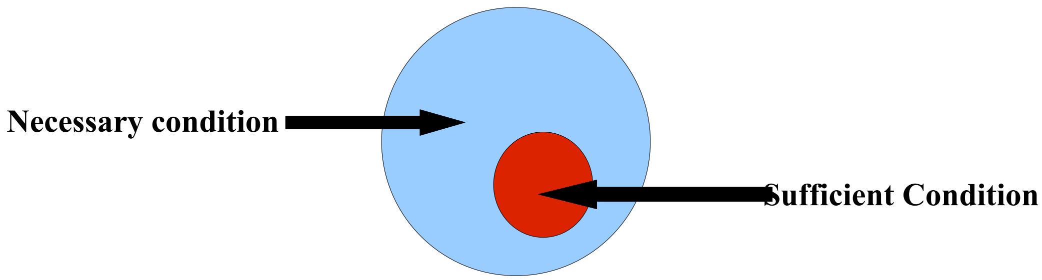
Spots(albert) $\rightarrow$ Measles(albert)

This says that if Albert has spots, he has measles. The spots surely don't cause the measles! They are an indicator of the measles. Knowing that Albert has spots is sufficient for knowing that he has measles. In fact, we call the antecedent of the conditional the *sufficient condition*: the truth of the antecedent is sufficient for (is enough for) the truth of the consequent.

The consequent, on the other hand, is a necessary result of the antecedent, or the *necessary condition*. If the antecedent is true, the consequent comes from it. Consider:

If object s is a square, then it has four sides (or, [S(s) $\rightarrow$ 4-sided(s)])

If something is square, it necessarily follows that it has four sides, or, being square is enough to ensure (sufficient for) being four-sided. Does the converse follow? If something is four-sided, is it then a square? No! it could be a rectangle, or a rhombus, or some other sort of four-sided object! The direction of the arrow in the conditional is meant to indicate this "one-way" flow of truth. Consider the below diagram: If a sentence "fits into" the space of the sufficient condition, it also fits into the necessary condition. But something can fit into the necessary condition without fitting the sufficient condition!



The terms *antecedent* and *consequent* are purely syntactic - they indicate a property about the form of a sentence (before and after the arrow). The terms *necessary* and *sufficient condition* are semantic terms - they indicate properties of the sentence's truth or meaning. In this case, the syntax is used to represent the semantics - the antecedent is always the sufficient condition, and the consequent is always the necessary condition. We'll come back to this in a minute.

The truth condition is a bit complex. Consider some conditional P → Q. Now suppose the sufficient condition is true. Then the necessary condition must also be true, or the conditional was a lie (false): P is *not* sufficient for Q. So far:

if P is true and Q is false, (P → Q) is false.

if P is true and Q is true, (P → Q) is true.

What happens when the antecedent is false? This is where things get a bit tricky. A conditional like (P → Q) only makes a claim about what follows from P, i.e., it only makes claims about what happens if P is true. If P isn't true, you have no counterexample to my claim. Suppose I return to the example about squares: if object s is a square, then it has four sides (or, [Square(s) → 4-sided(s)]). What if you responded: 'AHA! You liar! S is not square at all'! Wouldn't we all be bemused by this? After all, I only made a claim about what would be the case if S *was* square, or what follows from S's squareness. To rebut that claim, you need a positive counterexample: a case where the antecedent is true, but the consequent false. So to conclude:

if P is false and Q is false, $(P \rightarrow Q)$ is true.

if P is false and Q is true, $(P \rightarrow Q)$ is true.

This analysis has a major benefit: one of the central logical terms has been defined truth-functionally, that is, you can tell what the truth of a conditional is solely by looking at the truth of its parts. And because of the way it's been defined, you are guaranteed to never be led from truth to falsity. There is one small drawback: it gives some unnatural results. For example, the following conditionals are all (counterintuitively) true:

If Tom Cruise could act, he'd be made of cheese

If the Sun goes out right now, the earth will be fine for at least another year

If Bill Gates founded Apple, I am a hamster.

For now, we'll ignore these issues. The study of conditionals like these has led to the development of more complex logics dealing with *strict conditionals* and *subjunctive conditionals*. This will take us far too far afield, though.

Perhaps these examples show that the English "if-then" construction is not captured by the conditional (or, as we'll sometimes call it, the *material* or *truth-functional* conditional). No matter: the material conditional of PL is still defined by this truth rule.

Truth for conditionals

A sentence of the form $\varphi \rightarrow \psi$ is true_{PL} iff φ is false_{PL}, or if ψ is true_{PL}.

Before we move on, let's talk about the translation of some English constructions related to conditionals. If P, then Q is synonymous to

If P, Q

Q follows from P

P, conditional on Q

Q if P

These all translate as P $\rightarrow$ Q. In English, the overall position in the sentence doesn't matter; the antecedent is the statement immediately following the term 'if'. So what matters is not the overall position of the antecedent but its position relative to the key word.. In PL, the antecedent always comes first. Much easier!

We also sometimes use the term *only if*. This indicates a necessary condition (not a sufficient one!) Consider: 'Joe will live only if he gets the operation'. If joe is alive, what can you tell? He must have gotten the operation. If Joe has gotten the operation, does it follow that he will be alive? No! He might die from complications, or be hit by a steamroller as he leaps for joy in the street. So the phrase 'P only if Q' is synonymous with 'only if Q, P' and 'only if Q, then P', and these all get translated as P $\rightarrow$ Q. Just as before the statement immediately following the English construction 'only if' is the necessary condition, no matter where it occurs in the sentence. The basic rule of thumb when translating these is to think about what is a necessary result of what, and what is a sufficient circumstance for what.

Biconditionals

A biconditional is simply a two-way conditional, as the symbol indicates. So a biconditional means that the antecedent is sufficient; but since an arrow leads back to the antecedent from the consequent, the antecedent is also necessary. And the same goes for the consequent. The two parts of a biconditional are necessary and sufficient for each other. Biconditionals often appear in things like definitions. When you define a concept P, you want to give some other concept that contains only things that P contains, and doesn't contain anything that P doesn't contain. The definition of a square must contain all the squares, and no mere rectangles or rhombii.

With this in mind, we can tell that a biconditional states that the antecedent and the consequent must have the same truth value. They're either both true, or both false. So:

Truth for biconditionals

A sentence of the form $\varphi \rightarrow \psi$ is true$_{PL}$ iff φ and ψ are both true$_{PL,}$, or if, or if φ and ψ are both false$_{PL}$. (i.e., φ and ψ share the same truth value)

This gives the basic truth rules for all the sentences of PL. Even though there's only a handful of rules, and an infinite number of PL sentences, we've still got the tools to evaluate them all. How is this?

Recursion and complex sentences

Notice that all the truth rules are formulated in a wholly general way. We've used sentence variables such as φ and ψ and never made any claims about whether φ and ψ were atomic or complex! And in fact, we know from our discussion of recursion in syntax that a sentence like:

$(P \vee Q) \rightarrow (S \wedge R)$

is in fact a sentence of the form

$\varphi \rightarrow \psi$

where φ stands in for '$(P \vee Q)$' and ψ stands in for '$(S \wedge R)$'. So we know to use the truth rule for conditionals to evaluate this sentence. But the truth rule for the conditionals requires that we have the truth value of its parts, and we don't have that yet. So we put the conditional - evaluation off to last, and we evaluate each part: $(P \vee Q)$ and $(S \wedge R)$. To evaluate *those*, we know we need to use the disjunction rule and the conjunction rule, respectively. Start with the disjunction. To evaluate the disjunction, we must evaluate its parts. The parts of *this* disjunction are atomic, so we go to the truth rule for atomics.

In this case, I haven't told you what P and Q stand for (in terms of subjects and predicates), so we have to stop there. And the same follows for the conjunction (S ∧ R). Pretend you have the truth values for the atomics, though. Now we can build back up, determining the truth of (P ∨ Q) and (S ∧ R), and using those truth values to evaluate the entire conditional (P ∨ Q) → (S ∧ R).

More graphically:

(P ∨ Q) → (S ∧ R)

find the central connective that gives the sentence's form

(P ∨ Q) → (S ∧ R)

evaluate its parts immediately if you can, or move on to:

(P ∨ Q) → (S ∧ R)

find the connective that gives the forms of *those* sentences, and evaluate *their* parts

(P ∨ Q) → (S ∧ R)

evaluate those parts immediately if you can (using atomic truth rule here)

(P ∨ Q) → (S ∧ R)

move back "out" to evaluate the complex sentences they are parts of

(P ∨ Q) → (S ∧ R)

move back out to evaluate the whole sentence.

As you can see, evaluating the truth of any sort of a complex sentence is a bit of a tricky affair. It's easy to forget where you are at any given point, and it's easy to go wrong. In the next chapter, we'll introduce a simple, mechanical procedure for evaluating truth, that will allow us to "compose" the truth conditions of any complicated sentence in an intuitive and systematic way. This is the method of *truth tables*.

Summary of tables, definitions and rules

The Symbols of PL

Syntactic Category	Term	Predicate	Relation Symbol
NL Analogue	Proper names	Intransitive verbs, adjectives	Relation words, transitive verbs
Orthography	Lowercase letters, lowercase words 'a', 'b', 'albert', 'hanna'	Uppercase letters, words beginning with uppercase letters 'F', 'G', 'Pirate', 'Squirrel'	Uppercase letters, words beginning with uppercase letters 'R', 'S', 'Bigger', 'Pillages'
Refers to or Expresses	Individuals	Properties of individuals or sets of individuals	Relations between individuals

More Symbols of PL

Syntactic Category	Formula, WFF	Punctuation	Connectives
NL Analogue	Sentence	Punctuation	Connectives
Orthography	Uppercase letters	'(', ')'	'↔', '→', '∧, '¬, '∨'
Refers to or expresses	Truth Values	N/A	N/A (?)

The Symbols of PL_1 (PL's Metalanguage)

Category	Term Variables	Predicate/Relation Variables	Punctuation
NL Analogue	None	None	Punctuation
Orthography	Lowercase Greek α, α_1, α_2	Uppercase Greek φ,ψ	Quotes ', '
Role/ Function	Ranges over terms of PL	Ranges over predicates and relation terms of PL	Creates names for constructions of PL_0

More Symbols of PL_1 (Metalanguage)

Category	Sentential Variables
NL Analogue	None
Orthography	Uppercase Greek φ,ψ
Role/ Function	Ranges over WFFs of PL

Sentence Formation and Truth rules

Atomic sentence formation rule for PL
If φ is a relation term with an arity of n, and $\alpha_1 \ldots \alpha_n$ are exactly *n* names, then
$\varphi(\alpha_1, \alpha_2, \alpha_n)$
is an atomic sentence. Nothing else is an atomic sentence.

Compound Sentence formation rule for PL
If φ and ψ are sentences, then

$(\varphi \rightarrow \psi)$
$(\varphi \wedge \psi)$
$(\varphi \vee \psi)$
$(\varphi \leftrightarrow \psi)$
$\neg\varphi$

are all sentences of PL. Nothing else is a (Compound) sentence of PL.

Truth Rule for Atomic Sentences of PL

A sentence of the form φ^n ($\alpha_1, \alpha_2, \ldots \alpha_n$) is true$_{PL}$ iff the individual referred to by α_1 stands in the relation expressed by φ^n, to the individuals referred to by $\alpha_2 \ldots \alpha_3$.

Truth for Conjunction

A sentence of the form $\varphi \wedge \psi$ is true$_{PL}$ iff φ is true$_{PL}$ and ψ is true$_{PL}$.

Truth for Disjunction

A sentence of the form $\varphi \vee \psi$ is true$_{PL}$ iff either φ is true$_{PL}$, or ψ is true$_{PL}$, or both are true.

Truth for negation

A sentence of the form $\neg\varphi$ is true$_{PL}$ iff φ is false$_{PL}$.

Truth for conditionals

A sentence of the form $\varphi \rightarrow \psi$ is true$_{PL}$ iff φ is false$_{PL}$, or if ψ is true$_{PL}$.

Truth for biconditionals

A sentence of the form $\varphi \rightarrow \psi$ is $true_{PL}$ iff φ and ψ are both $true_{PL,}$, or if, or if φ and ψ are both $false_{PL}$. (i.e., φ and ψ share the same truth value)

Glossary

Arity

Antecedent -

Conjunct -

Consequent -

Disjunct -

Iff -

Abbreviation for If and Only If. P iff Q means that P is jointly necessary and sufficient for Q (and vice versa); P and Q are equivalent, P and Q always happen together.

Semantics -

Syntax -

Chapter II

Truth Tables and the Logical Concepts

Chapter Index

I) Introduction

II) The logical concepts, informally presented

A) Arguments

B) Equivalence

C) Validity

D) Entailment

F) Logical Truth

E) Contingency

G) Consistency

III) The Method of Truth Tables

A) Overview

B) Conjunctions

C) Disjunctions

D) Conditionals

E) Biconditionals

F) Negations

IV) Complex truth tables

A) More complex sentences

B) Short cuts

V) The logical concepts, formally defined

A) Overview

B) Logical Truth

C) Contingency

D) Logical Falsity

I Introduction

Now that we have a language, or at least a language fragment, it would be nice to use that language to explain the central concepts of logic – argument, proof, validity, entailment, and so on. Of course, we already have an informal understanding of those concepts. What formal logic does is to make those concepts precise and rigorous. This way, we can use the formal concepts to provide a systematic and mechanical means for applying them – that is, a completely automatic and easily-verified means of determining whether any argument is valid, any sentences are consistent, and so on. Human beings are both fallible and lazy (well, I'm lazy, anyway), and having an automatic way of determining validity helps us to overcome our fallibility. Doing things the systematic way helps us lazy folk get answers without having to work too hard at it.

In this chapter, we will discuss the central logical concepts in an informal manner. We'll then build on the analysis of truth in the last chapter to show how we can mechanically determine the truth conditions of a sentence (or set of sentences) in any possible circumstance. This is the method of truth tables, which allows us to build the truth of a sentence from the truth value of its parts. Then, we'll give formal definitions of the central logical concepts in terms of these truth tables, and show how to apply these concepts to sentences of PL.

When this is over, we'll have a means of proving validity, consistency, and so on for any sentences or arguments made in PL. Now, obviously, PL is a pretty small language. There are a lot of facts and / or thoughts we can't represent in PL. Well, we can represent them, after all, representation is cheap (we could easily stipulate that 'A' represents the proposition that *All men are mortal*, for example). But we can't represent these facts in such a way as to show what structures they have in common, what structures allow us to make valid arguments with them, and so on. So we can think of logic not as a "finished product" but as an ongoing project to discover the relevant structures of propositions. But of course, the journey of a thousand miles begins with but a single step, and the first task is to show how the logical concepts interact with the mast basic sort of language that we've already created.

II) The logical concepts, informally presented

Language is concerned with meaning, truth, and communication, but logic is concerned with – so it appears – arguments, validity, entailment, consistency, and similar ideas – mostly relations between sentences. It isn't immediately clear how we can connect the two up. The first task is to give the natural language meanings of these concepts. After all, we would like our formal definitions in PL to serve as explications of those natural language concepts!

A) Arguments

An argument is a set of sentences. Some of those sentences are *premises*, some (usually just one) is the conclusion. But it's not just any set of sentences – the premise sentences are *supposed* to force you (somehow) into believing the conclusion. Still, there are such things as bad arguments, very bad arguments, and even arguments where it's completely unclear what the premises have to do with the conclusions. Continental philosophy provides a good example of the latter sort of arguments. Nonetheless, they are all arguments, regardless of whether or not they succeed or even could succeed in forcing someone who accepts the premises to also accept the conclusion. All that is required is an *intention* that the premises lead to the conclusion.

What keeps the above characterization from being a formal definition is its reliance on unexplained concepts like 'force you to accept', 'lead to', and so on. And that will in general be what keeps the rest of these informal definitions from being final definitions. Nonetheless, they'll serve as a guide to our formal definition.

B) Equivalence

Two sentences are (roughly) equivalent when they always pick out the same state of affairs in the world (in the parlance of appendix 1 to the last chapter, they always have the same extension). 'Neither A nor B are true' and 'A and B are both false' are two different ways of stating the same thing; they're logically equivalent.

There are several sorts of equivalence. The broadest sort is what we'll call necessary equivalence. In this chapter, we'll discuss logical equivalence, but in the appendix, we'll talk about at least two other kinds.

C) Validity

An argument is valid when it's good. Better yet: an argument is valid when accepting the premises should force you to accept the conclusion. When should premises force you to accept a conclusion? When it's impossible for the premises to be true and the conclusion to be false.

A valid argument does not require the premises to be true! The following argument is valid:

Clinton is a Democrat
<u>All Democrats love cheeseburgers</u>
Thus, Clinton loves cheeseburgers.

It's valid because it is impossible for the premises to be true and the conclusion false. At least one premise is false, but *if* it were true, the conclusion would also necessarily be true. So valid arguments can in fact have false premises, and thus false conclusions. They cannot have all true premises and a false conclusion, though. So validity is a property of arguments as a whole.

Obviously, as arguers, we want something more than merely valid arguments. Since we want reasons to believe we have true conclusions, we want valid arguments *and* true premises! An argument like that is called a *sound* argument. Alas, logic cannot really tell you when an argument is sound (well, except under very narrow circumstances). Reason alone cannot tell you if a sentence like 'Albert is a pirate' is true, and logic alone cannot tell you if a sentence like 'Pirate(albert)' is true. These require empirical confirmation – you have to go out into the world and see if Albert really *is* a pirate.[5]

D) Entailment

Entailment is a relation between sentences. For one sentence to entail another is for the truth of the

5 In some cases, the premises will be true no matter what, and logic alone can tell you that. For example, you need no empirical confirmation to tell that 'Albert is either a pirate or he isn't' is true. If all the premises are like that, then logic alone can tell you whether or not the argument is sound, but those cases are few and far between.

first sentence to lead to the truth of the second, to force the second sentence to be true. So entailment is just the relation that holds between the premises and conclusion in a valid argument (and vive versa). In English, we use 'implication' as a synonym for 'entailment': we say 'the premises entail the conclusion', or 'the premises imply the conclusion'.

E) Contingency

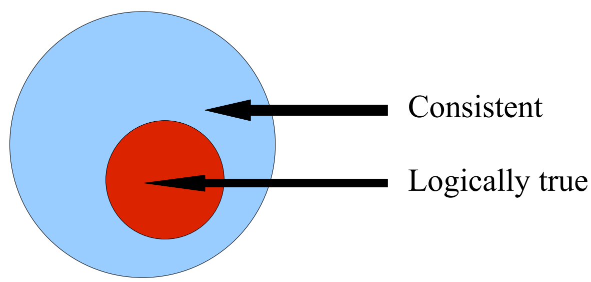
Most sentences are contingent – that is, whether or not they are true is contingent on the way the world turns out. 'Albert is a squirrel' is a true sentence just in case the individual named by 'Albert' has the property expressed by the phrase 'is a squirrel', i.e., if Albert has the property of being a squirrel. And of course, the world might have turned out so that Albert was a squirrel, and it might have turned out that he was a capybara. So Albert might have been, or might not have been a squirrel, depending on the way the world turned out. It is a contingent truth.

F) Logical Truth

Some sentences, however, turn out to be true no matter how the world turns out. They are "made true" not by any particular world, but by the constraints on what any possible world must be like – they are made true by logic alone. In this sense we call them truths of logic, a particular kind of necessary truth. For example, 'Albert is a squirrel' is a contingent truth, because although he is in fact a squirrel, the world *could* have turned out in such a way that he wasn't a squirrel/ On the other hand, the sentence 'Albert is either a squirrel or a non-squirrel' is not contingent. There's no way the world could turn out such that Albert failed to be a squirrel *and* failed to be a non-squirrel at the same time. The classes of squirrel and non-squirrel divide up all the possibilities for any world. . If logic gives the structure of possibilities, then 'Albert is either a squirrel or a non-squirrel' is true in virtue not of the world, but in virtue of the structure of possibilities. That is, it is a truth of logic or a logical truth. it is not contingent on any way the world happens to turn out.

G) Consistency

Sometimes we say things that are false, but at least the might be true. Sometimes we just get the facts wrong. Maybe Albert isn't a pirate, but we were fooled by his cutlass and eyepatch, and we say he is a pirate. Our utterance 'Albert is a pirate' is false, but there's nothing about the utterance itself that makes it false no matter how the world turns out. So the sentence is internally consistent; there's no sort of internal conflict in the sentence that necessarily prohibits it from being true. In this way, an internally consistent sentence or a self-consistent sentence is much like a contingent sentence. The difference is that every logically true sentence is also a consistent sentence – after all, if a sentence is always true, then there can't be any internal conflict in the sentence that makes it false. But no logically true sentence is a contingent sentence – because a contingent sentence could turn out either way, and logical truths can only turn out true. So contingent sentences are those that could turn out either true or false, consistent sentences are those that could turn out true, and logically true sentences could only turn out true.



The blue shaded area (excluding the red) is the space of contingent sentences

Sentences can also be consistent with each other. We say that two claims are consistent if they don't rule each other out. Think of the sentences 'Albert is a pirate' and 'Albert is an Englishman'. It might be that both claims are false, but neither claim precludes the other or rules it out. There are English pirates, or at least there could be, so nothing about either claim means the other has to be false. Some sentences are inconsistent with each other: For example, 'Albert is a squirrel' and 'Albert is a human'. Either claim might be true, but they can't both be true together, because being a squirrel precludes being a human. Inconsistent sentences are sentences that do preclude each other. One might be true, but they can't both be true. Of course, we can extend this idea to sets of many sentences, and ask whether a group of five or six sentences are consistent with each other or not.

H) Logical Falsity

Just as some sentences must turn out true, there are some sentences that must turn out false. Their falsity isn't a matter of how the would turns out, but rather, a matter of logic. The structure of possibilities rules these sentences out. For example, 'Albert is both a squirrel and a non-squirrel.' You don't even have to look at the way the world is to see that it could never be true. It's *logically false*.

In general, the negation of a logically true sentence will be logically false. Sometimes it's difficult to tell which sentences are logically true and logically false, especially with long and complex sentences. And it's difficult to tell which sets of sentences are consistent, or contingent, or valid arguments, or when any of the other concepts apply. The task now falls to us to explain these concepts in terms of the central concepts of truth and truth conditions that we gave in the last chapter. To do that, we'll demonstrate a method for mechanically determining the truth conditions of any possible sentence of PL, using the semantic principles we've already discussed – the rules defining the connectives for building sentences.

III) The Method of Truth Tables

A) Overview

In the last chapter, we demonstrated that every sentence of PL was either an atomic sentence - an "atom" - or a compound "molecule" built up out of atoms with a very small number of connectives. So we ought to be able to mechanically determine the truth value of any given molecule, once we know the truth values of its parts, by using the semantic rules for its connectives. That is, we can determine the truth conditions for the compound sentences by thinking about the rules used to build or compose that compound sentence. What we'll do now is present a simple, tabular method for doing this systematically.

What our rules will do is tell us what the truth value of any sentence is by looking at the truth values of its most basic parts. So what our table needs to first represent is the range of truth values of those basic parts. We'll call this the *reference columns.* Think of a simple conjunctive sentence like 'Pirate(albert) ∧ Squirrel(albert)'. The atoms of this sentence are P(a) and S(a) (I'm abbreviating here just to keep things simple). Each atom can be true or false. So we can represent the possible combinations in a table like so:

reference columns | Compound sentence

P(a)	S(a)	P(a) ∧ S(a)
T	T	
T	F	
F	T	
F	F	

Under each compound sentence, we'll calculate the truth values of that sentence *given the truth values of the compound parts shown to its left.* We'll adopt a few conventions just to make life easier. The reference columns are always along the left side of the table. Every table must have as many sentences in the reference columns as there are atomic parts in the compound sentence(s) on the table. And each atomic sentence should have two possible values (T and F). So for a sentence with n atomic parts, there will be 2^n possible combinations of truth values. Since each row is supposed to represent a possible way the world could have turned out with respect to the sentence, this means that the table for a sentence with n atomic parts will itself have 2^n rows. If a sentence has three atomic parts, its corresponding table will have eight rows (2 x 2 x 2); if a sentence has four atomic parts, there will be sixteen rows, and so on.

Let's adopt a convention for writing down the truth values in the reference columns. We'll start with the right-most reference column, and alternate T and F every other row, like so:

P(a)	S(a)	some sentence with P(a) and S(a)
	T	
	F	
	T	
	F	
	T	
	F	
	:	

Of course, if P(a) and S(a) are the only atomics, then we could have stopped after the fourth row. We continue this pattern until we've filled in all the rows needed in the table. Now the convention will be to move one row *left* and double the alternating pattern, so we get two trues, two falses, two trues, and so on:

P(a)	S(a)	some sentence with P(a) and S(a)
T	T	
T	F	
F	T	
F	F	
T	T	
T	F	
:	:	

If there's another reference column, we'll double the alternation pattern yet again - so we'd get four trues, four falses, four trues, four falses, and so on. if there's a fourth reference column, that column will have eight trues, eight falses, and so on.

Now, this is just a convention. You don't *have* to do it that way. You could write them down at random and then check to make sure you got every possible combination. But this convention guarantees that you'll get every possible combination of truth values relevant to the truth of your compound sentence. It also makes things easier on your grader when everyone follows the same convention. When things are easy on the grader, the grader is happy and not irritated. You want your grader in a good mood when he grades you, right?

That's how to fill out the reference columns, but we haven't yet shown how to calculate the truth value of the compound sentences. We'll do that now.

The limiting case for truth tables is not a compound sentence at all but rather an atomic sentence. It's such a limiting case that you'll never actually see it! But it would go like this:

P(a)	P(a)
T	T
F	F

The reference column (on the far left) gives the possible truth values for all the parts (i.e., one part) of the sentence. The rightmost column gives the truth value of the sentence given the values of its parts. If P(a) is true, then P(a) is true (Duh!), and if P(a) is false, then P(a) is false. See why I called this a limiting case, and said you'd never see it?

What we'll do now is to give the truth tables for the most basic sorts of compound sentences - conjunctions, disjunctions, conditionals, biconditionals and negations. But these will simply be the semantic rules for those connectives (that we gave in chapter I), expressed as a table.

B) Conjunctions

Let's return to our pirate squirrel from a few pages back.

P(a)	S(a)	P(a) ∧ S(a)
T	T	
T	F	
F	T	
F	F	

Recall the semantic rule for ∧: a conjunction is true iff both conjuncts are true (otherwise it's false.)

So, the truth value of (P(a) ∧ S(a)) in row one is *True*. Let's represent this as:

P(a)	S(a)	P(a) ∧ S(a)
T	T	T
T	F	
F	T	
F	F	

Notice that we write down the truth value of the sentence directly under its central connective. Now, given the semantic rule for , what is the truth value in row 2? A conjunction is true iff both conjuncts are true, but in row two, S(a) is false. So the truth value of (P(a) ∧ S(a)) in row two is itself false. We show this below:

P(a)	S(a)	P(a) ∧ S(a)
T	T	T
T	F	F
F	T	
F	F	

In fact, the only row where both conjuncts are true is in line one. Lines three and four each have at least one false conjunct. So the truth value of (P(a) ∧ S(a)) in lines three and four is also false:

P(a)	S(a)	P(a) ∧ S(a)
T	T	T
T	F	F
F	T	F
F	F	F

There! We've shown how to determine the truth value of any conjunctive sentence, for every possible combination of truth values of its parts! There's no other way the world could turn out with respect to the only things that matter for the sentence, namely, the truth of P(a) and of S(a). Conjunctive sentences are thus truth functional. The truth value of the whole is a function of the truth value of the parts.

Just to be all formal, and have a truth table that applies to *any* conjunctive sentence, let's use metalanguage variables instead of particular sentences. Our general truth table for conjunctions looks like:

φ	ψ	$\varphi \wedge \psi$
T	T	T
T	F	F
F	T	F
F	F	F

Let's think about this table for a minute. We can think of each row as representing a possible circumstance, or a possible way the world could have turned out. So what we've really done in this table is show what the truth of a conjunction (like (P(a) ∧ S(a))) is in any possible circumstance that could arise. We've determined not its truth value, (because we haven't determined which row of the table is the 'actual circumstance'), but its truth conditions. And isn't that a big part of showing what the meaning of a sentence is - showing what its truth value would be in any possible circumstance?

C) Disjunctions

Now that we've explained how truth tables work, the rest of these basic tables will go quickly. Consider a random disjunction (φ ∨ ψ). The semantic rule for disjunctions tells us that a disjunction is true whenever one *or both* of its disjuncts is true, and false otherwise (i.e., when both disjuncts are false). So we can represent this as:

φ	ψ	φ ∨ ψ
T	T	T
T	F	T
F	T	T
F	F	F

I've highlighted the one row where the disjunction is false. Pretty simple, no? I'll give the rest of the basic tables here, for reference, even though you can certainly figure them out yourself quite easily.

D) Conditionals

In a conditional like ($\varphi \rightarrow \psi$), the conditional is only false when the antecedent is true but the consequent is false (Yes, odd, I know, but we talked about that last chapter, so let's forget about that.) So the truth table for conditionals looks like:

φ	ψ	$\varphi \rightarrow \psi$
T	T	T
T	F	F
F	T	T
F	F	T

Once again, I've highlighted the only false outcome, just to be clear.

E) Biconditionals

Recall that a biconditional like (φ ↔ ψ) claims that two sentences, φ and ψ, have the same truth value. So the biconditional is true just in case both parts have the same truth value: i.e., they're both true, or they're both false. Otherwise, the biconditional is false.

φ	ψ	φ ↔ ψ
T	T	T
T	F	F
F	T	F
F	F	T

F) Negations

Negations are the odd man out here. They aren't used to connect sentences to form a bigger sentence, but rather to operate on a single sentence to form a bigger sentence. Sometimes negation is called an *operator* rather than a connective. We can think of a negation as a 'flipper': it flips the truth value of the sentences it operates on. The negation of a sentence such as φ is $\neg\varphi$. $\neg\varphi$ is false if φ is true, and $\neg\varphi$ is true if is φ false. That's just what 'not' means. So we can represent this as follows:

φ	$\neg\varphi$
T	F
F	T

These tables exhaust every possible sentence that can be made in PL. What we'll do know is to show how to use these basic tables to give the truth conditions for much, much more complex compound sentences.

IV) Complex truth tables

A) More complex sentences

Obviously, you're going to want to evaluate sentences a bit more complex than these basic two - part sentences. Luckily, it's quite simple to extend the truth table mechanism to deal with sentences of any complexity! What we need to do is to find the central connective (as defined in CH I) for each sentence. This central connective (or operator) will connect two sentences of arbitrary complexity (or operate on one sentence, in the case of negation). So what we'll then do is hold off on the main sentence, and find the central connective for its immediate subsentences. That central connective, in turn, will connect either compound or atomic sentences. We repeat this process until we get to a connective that connects atomic sentences. Then we use our basic truth tables to work out the truth values of these subsentences. Once we've done that, we use the values of *those* subsentences to determine the value of the sentences they form, until we've worked our way back "out" to the central connective of the whole sentence. Sounds like a mess! But it's really quite simple. Consider this nasty sentence:

$\neg[(P \rightarrow Q) \wedge R] \leftrightarrow [S \rightarrow (R \vee \neg P)]$

The central connective is the biconditional. It connects the two highlighted subsentences. We'll look at the first subsentence now.

$\neg[(P \rightarrow Q) \wedge R]$

It's central connective is a negation. It negates the highlighted sentence. So we now consider it:

$[(P \rightarrow Q) \wedge R]$

This sentence is a conjunction. It conjoins the two highlighted sentences. One of those sentences is atomic, so when we do the truth table, we'll just copy over the value from the reference column, and write those values in the column under R. Now we go to the other conjunct:

$(P \rightarrow Q)$

This is a conditional, that connects the highlighted parts. But those parts are atomic. So we can just use the truth table for conditionals to determine the values of $(P \rightarrow Q)$. When we do the table, we'll just write those values in the column under the $\rightarrow$.

Now we'll be in a position to work out the values for $[(P \rightarrow Q) \wedge R]$, since we have the values of each of its conjuncts. We'll use the table for conjunctions, and write the values for the conjunction in the column under the $\wedge$.

At *this* point, we'll be able to work out the values for the negation $\neg[(P \rightarrow Q) \wedge R]$, since we just worked out the value of the subsentence being negated. We'll just use the truth table for negations to 'flip' the value of the negated sentence, namely, $[(P \rightarrow Q) \wedge R]$, and we'll write that down in the column under the $\neg$.

Now we'll go over to the other main subsentence, namely, $[S \rightarrow (R \vee \neg P)]$. The central connective here is the $\rightarrow$. So we'll work all the way "in" just like we did before, and ultimately use the truth table for conditionals to write down the value in the column under the $\rightarrow$.

The final step will be to use the truth table for biconditionals to match up the truth values for the two main subsentences (these will be given in the columns for $\neg$ and $\rightarrow$, respectively):

$\neg[(P \rightarrow Q) \wedge R] \leftrightarrow [S \rightarrow (R \vee \neg P)]$

central connective

Let's pick a slightly simpler sentence, and show how we build the truth table in this way. Let's start with just the sentence ¬[(P → Q) ∧ R].

The sentence has three atomic parts, so we'll need 2^3, or eight, rows. And we'll fill in the reference columns according to the convention we just described.

P	Q	R	¬[(P → Q) ∧ R]
T	T	T	
T	T	F	
T	F	T	
T	F	F	
F	T	T	
F	T	F	
F	F	T	
F	F	F	

The central connective is the negation, which I've highlighted below. We'll give it its own column. We'll also put in a column for each of the subparts.

P	Q	R	¬	[(P → Q)	∧	R]
T	T	T	?			
T	T	F	?			
T	F	T	?			
T	F	F	?			
F	T	T	?			
F	T	F	?			
F	F	T	?			
F	F	F	?			

We will ultimately get the values for the central connective - the negation - by flipping the value for the sentence it operates on. That sentence is the conjunction, highlighted below:

P	Q	R	¬	[(P → Q)	∧	R]
T	T	T			?	
T	T	F			?	
T	F	T			?	
T	F	F			?	
F	T	T			?	
F	T	F			?	
F	F	T			?	
F	F	F			?	

This conjunction in turn conjoins the two sentences (P → Q) and R. The value of (P → Q) can be determined by using the truth table for conditionals. Recall that a conditional is always true *unless* both the antecedent is true and the consequent is false. We get that information from the reference columns of P and Q.

P	Q	R	¬	[(P → Q)	∧	R]
T	T	T		T	?	
T	T	F		T	?	
T	F	T		F	?	
T	F	F		F	?	
F	T	T		T	?	
F	T	F		T	?	
F	F	T		T	?	
F	F	F		T	?	

We can also get the value for R simply by copying it over from the reference column:

P	Q	R	¬	[(P → Q)	∧	R]
T	T	T		T	?	T
T	T	F		T	?	F
T	F	T		F	?	T
T	F	F		F	?	F
F	T	T		T	?	T
F	T	F		T	?	F
F	F	T		T	?	T
F	F	F		T	?	F

Now we get the value of the conjunction just by using the truth table for conjunctions and matching up the truth values of the sentences that it conjoins (namely, (P → Q) and R). . Recall that a conjunction is only true where both conjuncts are true. So that's just lines 1, 5 and 7.

P	Q	R	¬	[(P → Q)	∧	R]
T	T	T	?	T	T	T
T	T	F	?	T	F	F
T	F	T	?	F	F	T
T	F	F	?	F	F	F
F	T	T	?	T	T	T
F	T	F	?	T	F	F
F	F	T	?	T	T	T
F	F	F	?	T	F	F

All that remains now is to get the value of the sentence as a whole, i.e, the negation. But that will be simple: the truth table for negation just tells us to "flip" the value of the negated sentence, and we already have that value (it's highlighted in the above table). So our completed truth table looks like this:

P	Q	R	¬	[(P → Q)	∧	R]
T	T	T	F	T	T	T
T	T	F	T	T	F	F
T	F	T	T	F	F	T
T	F	F	T	F	F	F
F	T	T	F	T	T	T
F	T	F	T	T	F	F
F	F	T	F	T	T	T
F	F	F	T	T	F	F

There! Done! When you turn in your truth tables, you'll want to indicate the central connective by highlighting that column, or circling it, or something like that. But that's all there is to it. Any sentence you can imagine is going to be the result of connecting two subsentences. So just find the smallest subsentences, use the basic truth tables to determine their values, and use those values in turn to determine the values of the bigger sentences that they comprise. Work your way back out until you get the values of the whole sentence, and BANG, you're done. Start from the smallest subsentences, and work your way out to the sentence defined by the central connective.

B) Short cuts

There's a few points to make that might speed things up considerably. Let's think about the semantics of our basic connectives a minute.

If you've got a conjunction, and one of the conjuncts is false, you don't even need to know the value of the other conjunct. The conjunction as a whole will be false, no matter what. So there's no need to fill in or determine the value of the second conjunct! This might be very helpful when the false conjunct is atomic, and the other conjunct is some hairy complicated sentence. In half the cases (i.e. where the atomic sentence is false), you can tell right away what the value of the conjunction is without messing with the tough part.

If you've got a disjunction, and one of the disjuncts is true, you don't even need to know the value of the other disjunct. The disjunction as a whole will be true, no matter what. So there's no need to fill in or determine the value of the second disjunct! This might be very helpful when the true disjunct is atomic, and the other disjunct is some hairy complicated sentence. In half the cases (i.e. where the atomic sentence is false), you can tell right away what the value of the disjunction is without messing with the tough part.

If you've got a conditional, and the antecedent is false, the conditional will be true no matter what. So you can ignore the consequent, which will be especially nice if the consequent is some hairy complex sentence. On the other hand, if the consequent is true, the conditional will also be true no matter what.

Basically, if you've got a complex sentence, focus on the smaller part first. Once you've determined the values of the smaller part, you'll be able to ignore the larger part in many rows! let's give an example of how this works.

Consider a Complex conditional: If Albert is a pirate squirrel, then he plunders Tortuga. We can represent this as [(Pirate(a) $\wedge$ Squirrel(a)) $\rightarrow$ Plunders (a, t)]. The subject-predicate structure is irrelevant here, but I think its a lot more fun to talk about pirates and plundering than it is to talk about Ps and Qs. Anyway, let's set up a table for this.

Pirate(a)	Squirrel(a)	Plunders (a, t)	(Pirate(a) $\wedge$ Squirrel(a)) $\rightarrow$ Plunders (a, t)
T	T	T	
T	T	F	
T	F	T	
T	F	F	
F	T	T	
F	T	F	
F	F	T	
F	F	F	

This is just an eight-row table, so it's not too much work. But the sooner we're done, the sooner we can drink a tasty Arrogant Bastard Ale. So I bet we can cut our work by nearly two thirds. Remember what we said about starting with the smaller part of the sentence - in this case, the consequent Plunders(a,t). We'll set up a column for the central connective (the conditional) and for the antecedent and consequent:

Pirate(a)	Squirrel(a)	Plunders (a, t)	(Pirate(a) $\wedge$ Squirrel(a))	$\rightarrow$	Plunders (a, t)
T	T	T			
T	T	F			
T	F	T			
T	F	F			
F	T	T			
F	T	F			
F	F	T			
F	F	F			

Now we can just copy over the values from the reference column for Plunders(a,t):

Pirate(a)	Squirrel(a)	Plunders (a, t)	(Pirate(a) ∧ Squirrel(a))	→	Plunders (a, t)
T	T	T			T
T	T	F			F
T	F	T			T
T	F	F			F
F	T	T			T
F	T	F			F
F	F	T			T
F	F	F			F

Right away, we can tell that the conditional will be automatically true in every row that the consequent is true. In other words, every other row. BAM! That's something like half the work, done (depending on how you count work).

Pirate(a)	Squirrel(a)	Plunders (a, t)	(Pirate(a) ∧ Squirrel(a))	→	Plunders (a, t)
T	T	T	------	T	T
T	T	F		?	F
T	F	T	------	T	T
T	F	F		?	F
F	T	T	------	T	T
F	T	F		?	F
F	F	T	------	T	T
F	F	F		?	F

The question marks are there to show that these cells could go either way, and the ------ is to show that we don't need to calculate this. I bet we can slack off on even more work, though. Let's add some new columns for the conjunction:

Pirate(a)	Squirrel(a)	Plunders (a, t)	(Pirate(a)	∧	Squirrel(a))	→	Plunders (a, t)
T	T	T	----	----	----	T	T
T	T	F				?	F
T	F	T	----	----	----	T	T
T	F	F				?	F
F	T	T	----	----	----	T	T
F	T	F				?	F
F	F	T	----	----	----	T	T
F	F	F				?	F

I t really doesn't matter if we start with Pirate(a) or Squirrel(a). I'll start here with Pirate(a). We just copy the values for Pirate(a) down from the reference column:

Pirate(a)	Squirrel(a)	Plunders (a, t)	(Pirate(a)	∧	Squirrel(a))	→	Plunders (a, t)
T	T	T	T	----		T	T
T	T	F	T			?	F
T	F	T	T	----		T	T
T	F	F	T			?	F
F	T	T	F	----		T	T
F	T	F	F			?	F
F	F	T	F	----		T	T
F	F	F	F			?	F

Recall that if one conjunct is false, the whole conjunction is false. BAM! That's the last four rows, done!

Pirate(a)	Squirrel(a)	Plunders (a, t)	(Pirate(a)	∧	Squirrel(a))	→	Plunders (a, t)
T	T	T	T	----	----	T	T
T	T	F	T	?		?	F
T	F	T	T	----	----	T	T
T	F	F	T	?		?	F
F	T	T	F	----	----	T	T
F	T	F	F	F	----	?	F
F	F	T	F	----	----	T	T
F	F	F	F	F	----	?	F

All that's really left is to finish off lines 2 and 4. We'll copy the values for Squirrel(a) from the reference column:

Pirate(a)	Squirrel(a)	Plunders (a, t)	(Pirate(a)	∧	Squirrel(a))	→	Plunders (a, t)
T	T	T	T	----	----	T	T
T	T	F	T	?	T	?	F
T	F	T	T	----	----	T	T
T	F	F	T	?	F	?	F
F	T	T	F	----	----	T	T
F	T	F	F	F	----	?	F
F	F	T	F	----	----	T	T
F	F	F	F	F	----	?	F

Now we calculate the value of the conjunction in lines 2 and 4:

Pirate(a)	Squirrel(a)	Plunders (a, t)	(Pirate(a)	∧	Squirrel(a))	→	Plunders (a, t)
T	T	T	T	----	----	T	T
T	T	F	T	T	T	?	F
T	F	T	T	----	----	T	T
T	F	F	T	F	F	?	F
F	T	T	F	----	----	T	T
F	T	F	F	F	----	?	F
F	F	T	F	----	----	T	T
F	F	F	F	F	----	?	F

And then we finish off the table, by calculating the values of the central conditional:

Pirate(a)	Squirrel(a)	Plunders (a, t)	(Pirate(a)	∧	Squirrel(a))	→	Plunders (a, t)
T	T	T	T	----	----	T	T
T	T	F	T	T	T	F	F
T	F	T	T	----	----	T	T
T	F	F	T	F	F	T	F
F	T	T	F	----	----	T	T
F	T	F	F	F	----	T	F
F	F	T	F	----	----	T	T
F	F	F	F	F	----	T	F

There! A small amount of thought in setting up the problem actually spared us quite a bit of work!

It's useful to be able to determine truth conditions for sentences, but what we're really after is a way to check for very particular sorts of truth conditions. This is what we'll do in the next section - use truth tables to check for validity, logical truth, and so on.

V) The logical concepts, formally defined

A) Overview

In the beginning of this chapter, we informally introduced several concepts central to logic. Of these, probably the most important is the concept of validity. After all, we usually think of logic as the science of argument, and we're especially concerned with how to evaluate arguments. Do the premises really force us to accept the conclusion? And it's often extremely difficult to tell, just by looking at an argument, if it's valid or not. In this section we'll extend the method of truth tables to enable us to mechanically evaluate the validity of arguments; moreover, we'll use this method to check sentences for consistency, logical truth, and so on.

The first step will be to define each of these concepts in terms of the truth values of sentences alone.

B) Logical Truth

Earlier, we informally characterized logical truths (or logically true sentences) as those sentences that were true no matter what, solely in virtue of their logical structure. Well, the logical structure of a sentence is determined by its connectives, so we should be able to very easily implement this definition in a formal setting:

Logical Truth:
φ is logically true (a logical truth) iff φ is true on every row of its truth table.

Let's take a look at a classic logical truth, an instance of the Law of the Excluded Middle - P(a) ∨ ¬P(a). There's only one atomic part, namely P(a), so this will be a very small table (2 lines):

P(a)	P(a) ∨ ¬P(a)
T	
F	

We'll do this the long way, just to give a fully detailed example of working with truth tables. First we'll isolate the central connective - the disjunction - and make a column for it:

P(a)	P(a)	∨	¬P(a))
T		?	
F		?	

To get the value of the disjunction, we'll need to compare the values of the two disjuncts, using the truth table for disjunctions. The first disjunct is atomic, so we can simply copy it over from the reference column:

P(a)	P(a)	∨	¬P(a))
T	T	?	
F	F	?	

The second disjunct is the negation of an atomic sentence, so we'll just "flip" the value of the atomic sentence as given in the reference column:

P(a)	P(a)	∨	¬P(a))
T	T	?	F
F	F	?	T

The truth table for disjunctions tells us that a disjunction is true iff at least one disjunct is true. Each row of our table has at least one true disjunct. So the disjunction itself is true on each row:

P(a)	P(a)	∨	¬P(a))
T	T	T	F
F	F	T	T

This disjunction is true in every possible circumstance that could be relevant! So it's a logical truth. A logical truth in this sense is sometimes called a *Tautology*.

C) Contingency

Most sentences aren't like this, however. Most sentences are true or false contingent on the way the world turns out. They could turn out true, and they could turn out false. These are the contingent sentences. This turns out to be very simple to define formally, using the ideas already presented:

Contingency:

φ is contingent (a contingent sentence) iff φ is true on at least one row of its truth table, and false on at least one row.

When you check for contingency, be sure to circle, highlight, or otherwise mark the true row and the false row.

D) Logical Falsity

Earlier, we informally characterized logical falsehoods (or logically false sentences) as those sentences that were false no matter what, solely in virtue of their logical structure. So the definition of logically false sentences will just be the opposite of the definition for logically true sentences.

> **Logical Falsity:**
> φ is logically false (a logical falsehood) iff φ is false on every row of its truth table.

Sometimes we'll refer to a logical falsity as a self-inconsistent sentence.

E) Logical Equivalence

Informally, we stated that two sentences were logically equivalent when they picked out the same state of affairs. We can make this more exact by talking about the truth conditions for the sentences. If the two sentences really do pick out the same state of affairs, then the two sentences will be true in all and only the same circumstances, right? So the formal definition of logical equivalence is:

> **Logical Equivalence:**
>
> φ and ψ are logically equivalent iff they share the same truth value on every row of a single truth table.

We haven't discussed truth tables with multiple sentences yet, so let's demonstrate. Suppose we have two sentences, ¬(P ∨ Q) and (¬P ∧ ¬Q), and we want to determine if they're logically equivalent. We create a single table, with a reference column for each of the atomics in all the sentences. Then we'll have a separate column for each main sentence:

P	Q	¬(P ∨ Q)	(¬P ∧ ¬Q)
T	T		
T	F		
F	T		
F	F		

Now we'll create a column for each of the central connectives:

P	Q	¬	(P ∨ Q)	(¬P	∧	¬Q)
T	T					
T	F					
F	T					
F	F					

It'd be nice if I could get some kind of very solid line to separate the two sentences. Oh well, we work with the word processor we have, not with the word processor we wish we had. There! That's a truth table for multiple sentences. And to check whether the two sentences are logically equivalent, you just complete the table and look to see if the values in the highlighted rows are the same. Let's do that now.

P	Q	¬	(P ∨ Q)	(¬P	∧	¬Q)
T	T			F		F
T	F			F		T
F	T			T		F
F	F			T		T

We'll start with the second sentence. Each conjunct is a negation, so we just look at the value of the atomic sentences P and Q from the reference columns, and flip it. Then we compare those two to get the value of the conjunction:

P	Q	¬	(P ∨ Q)	(¬P	∧	¬Q)
T	T			F	F	F
T	F			F	F	T
F	T			T	F	F
F	F			T	T	T

Now let's do the first sentence. First we'll do the values for the innermost sentence, the disjunction. Disjunctions are true iff at least one disjunct is true, and we can find the values of the disjuncts from the reference column:

P	Q	¬	(P ∨ Q)	(¬P	∧	¬Q)
T	T		T	F	F	F
T	F		T	F	F	T
F	T		T	T	F	F
F	F		F	T	T	T

Then, to get the value of the negation, we just flip the value of the sentence being negated:

P	Q	¬	(P ∨ Q)	(¬P	∧	¬Q)
T	T	F	T	F	F	F
T	F	F	T	F	F	T
F	T	F	T	T	F	F
F	F	T	F	T	T	T

Now, we simply compare the values of the two central connectives:

P	Q	¬	(P ∨ Q)	(¬P	∧	¬Q)
T	T	F	T	F	F	F
T	F	F	T	F	F	T
F	T	F	T	T	F	F
F	F	T	F	T	T	T

The four rows we have exhaust the possible circumstances that could be relevant to the truth of the two sentences. And we can see that the two sentences share the same truth value in all and only the same circumstances! So they are in fact logically equivalent.

There's one oddity to point out about our definition of logical equivalence. Suppose two sentences are each logically true, that is, true in every possible circumstance. Then they'll share the same truth conditions, even if they aren't "about" the same things at all. For example, $P(a) \lor \neg P(a)$ is true in every possible circumstance. But so is $\neg(S(b) \land \neg S(b))$. So they're *ipso facto* logically equivalent, even though they seem to pick out totally different states of affairs.

Perhaps our formal concept is only roughly an analysis of the informal concept. When we analyze concepts, we sometimes change our minds about some of the aspects of the concept we're analyzing. So maybe, after analysis, we decide that L.E. sentences aren't really sentences that pick out the same state of affairs, but instead, they're something closely related (and more "deep") - such as, sentences that share the same truth conditions.

Another way to think of this is that logical truths like $P(a) \lor \neg P(a)$ don't really pick out a state of affairs. Rather, logical truths describe some facet of the logical structure of *every* state of affairs. If you think of logical truths that way, then every logical truth describes this logical structure from a different angle, and so in a way, every logical truth does pick out the same thing.

F) Consistency

Recall that both single sentences can be consistent (with themselves), and sets of sentences can be consistent. For a sentence to be self-consistent, it has to be that the sentence doesn't somehow rule out its own truth. So a self-consistent sentence is naturally a sentence that could be true.

> **Consistency:**
> A sentence φ is consistent (with itself) iff it is true on at least one row of its truth table.

Self-consistent sentences are much like contingent sentences, except they include all the logical truths as well as the contingent truths.

A *set* of sentences is consistent just in case they could all be true together. To show this, we create a single large truth table that includes all the sentences in the set, and all their atomic parts in the reference column.

> **Consistency:**
> A set of sentences {φ, ψ,. . .π} is consistent iff there is at least one row of its truth table where every sentence in {φ, ψ,. . .π} is true.

Let's demonstrate this with a simple set of three sentences: P, (P → Q) and ¬Q. There's only two atomic sentences here, so this will be a small (four row) table.

P	Q	P	P → Q	¬Q
T	T			
T	F			
F	T			
F	F			

We can easily fill in the first and last sentences. We just copy the value for P over from the reference column, and we flip the value of Q in the reference column to get ¬Q:

P	Q	P	P → Q	¬Q
T	T	T		F
T	F	T		T
F	T	F		F
F	F	F		T

Actually, we didn't even need to do that much work. Since P is false in rows three and four, we already know that those rows won't be rows where all the sentences are true. So we didn't even need to calculate ¬Q. At any rate, next we calculate the value of P → Q. The truth table for conditionals tells us that P → Q is true *iff* there's no counterexample; i.e., it's true on every row except when P is true and Q is false.

P	Q	P	P → Q	¬Q
T	T	T	T	F
T	F	T	F	T
F	T	F	T	F
F	F	F	T	T

Now we look at each row. Is there a row where all three sentences are assigned T? No! These three sentences are *not* consistent! But that only makes sense: when you think about what the sentences mean, you notice that if the first two sentences are true, then Q must be true. But if Q is true, ¬Q can't be true. So they can't all be true at the same time. Each sentence is consistent with itself, but they aren't consistent with each other.

If the sentences are consistent, show this by highlighting or circling the single row on which they are all true.

G) Validity

Validity is a property of arguments, not of individual sentences. So, like consistency and equivalence, validity is a relation between sentences. Specifically, its the relationship between the sentences in an argument.

Remember our original, informal definition of validity: an argument is valid if the truth of the premises forces you to accept the truth of the conclusion. We can get rid of the unclear term "forces" and characterize validity just in terms of truth. In a nutshell, an argument is valid just in case, every time the premises are true, the conclusion is also true. In other words, the truth of the premises guarantees that the conclusion is true. What about when the premises are false? That doesn't matter. In a valid argument, *whenever* the premises are true, so is the conclusion.

Validity:
An argument with the premises {φ, ψ,. . .π} and the conclusion σ is valid iff, on *every* row where *all* the premises {φ, ψ,. . .π} are true, the conclusion σ is also true.

If there's any row with all true premises, but a false conclusion, that row is said to be a counterexample. So another way to think of a valid argument is that its an argument with no possible counterexamples. If an argument has a single counterexample, it's not valid. It doesn't matter if there are 317 "good" rows and only one measly counterexample. Validity is an ironclad guarantee. In a valid argument, the premises are said to *Entail* the conclusion.

Let's illustrate with an example.

Consider the argument

1) (A ∨ B) → C

2)¬ D → B

3) ¬ E

∴ D ∧ ¬A

To evaluate this, we'll need a single truth table with *five* atomic sentences (A, B, C, D and E), and four sentence columns (the three premises and the conclusion). Five atomic sentences means - eek! - 32 rows! This may take a while.

A	B	C	D	E	(A ∨ B) → C	¬ D → B	¬ E	D ∧ ¬A
T	T	T	T	T				
T	T	T	T	F				
T	T	T	F	T				
T	T	T	F	F				
T	T	F	T	T				
T	T	F	T	F				
T	T	F	F	T				
T	T	F	F	F				
T	F	T	T	T				
T	F	T	T	F				
T	F	T	F	T				
T	F	T	F	F				
T	F	F	T	T				
T	F	F	T	F				
T	F	F	F	T				
T	F	F	F	F				
F	T	T	T	T				
F	T	T	T	F				
F	T	T	F	T				
F	T	T	F	F				
F	T	F	T	T				
F	T	F	T	F				
F	T	F	F	T				
F	T	F	F	F				
F	F	T	T	T				
F	F	T	T	F				
F	F	T	F	T				
F	F	T	F	F				
F	F	F	T	T				
F	F	F	T	F				
F	F	F	F	T				
F	F	F	F	F				

Ugh! Since this is so long, let's help ourselves to a shortcut. Remember that in checking for validity, we're looking for counterexamples. And any row with even a *single* false premise *cannot be* a counterexample, because a counterexample is a row with *all* true premises and a false conclusion. So let's start with our smallest premise, which would be premise three, ¬E. We can get the value of ¬ E simply by looking at the value of E in the reference column, and "flipping" that value. Then, on any row where ¬ E is false, we know it won't be a counterexample, so we can stop work on the rest of that row! Nifty, eh?

A	B	C	D	E	$(A \vee B) \rightarrow C$	$\neg D \rightarrow B$	$\neg E$	$D \wedge \neg A$
T	T	T	T	T	---	---	F	---
T	T	T	T	F			T	
T	T	T	F	T	---	---	F	---
T	T	T	F	F			T	
T	T	F	T	T	---	---	F	---
T	T	F	T	F			T	
T	T	F	F	T	---	---	F	---
T	T	F	F	F			T	
T	F	T	T	T	---	---	F	---
T	F	T	T	F			T	
T	F	T	F	T	---	---	F	---
T	F	T	F	F			T	
T	F	F	T	T	---	---	F	---
T	F	F	T	F			T	
T	F	F	F	T	---	---	F	---
T	F	F	F	F			T	
F	T	T	T	T	---	---	F	---
F	T	T	T	F			T	
F	T	T	F	T	---	---	F	---
F	T	T	F	F			T	
F	T	F	T	T	---	---	F	---
F	T	F	T	F			T	
F	T	F	F	T	---	---	F	---
F	T	F	F	F			T	
F	F	T	T	T	---	---	F	---
F	F	T	T	F			T	
F	F	T	F	T	---	---	F	---
F	F	T	F	F			T	
F	F	F	T	T	---	---	F	---
F	F	F	T	F			T	
F	F	F	F	T	---	---	F	---
F	F	F	F	F			T	

Now let's look at the conclusion. Any row on which the conclusion is true will be fail to be a counterexample, by the same reasoning we used earlier. The conclusion is a conjunction, that conjunction will be true on any row where D is true and A is false. We'll "cross off" any row where the conclusion is true:

A	B	C	D	E	$(A \vee B) \rightarrow C$	$\neg D \rightarrow B$	$\neg E$	$D \wedge \neg A$
T	T	T	T	T	---	---	F	---
T	T	T	T	F			T	F
T	T	T	F	T	---	---	F	---
T	T	T	F	F			T	F
T	T	F	T	T	---	---	F	---
T	T	F	T	F			T	F
T	T	F	F	T	---	---	F	---
T	T	F	F	F			T	F
T	F	T	T	T	---	---	F	---
T	F	T	T	F			T	F
T	F	T	F	T	---	---	F	---
T	F	T	F	F			T	F
T	F	F	T	T	---	---	F	---
T	F	F	T	F			T	F
T	F	F	F	T	---	---	F	---
T	F	F	F	F			T	F
F	T	T	T	T	---	---	F	---
F	T	T	T	F	---	---	T	T
F	T	T	F	T	---	---	F	---
F	T	T	F	F			T	F
F	T	F	T	T	---	---	F	---
F	T	F	T	F	---	---	T	T
F	T	F	F	T	---	---	F	---
F	T	F	F	F			T	F
F	F	T	T	T	---	---	F	---
F	F	T	T	F	---	---	T	T
F	F	T	F	T	---	---	F	---
F	F	T	F	F			T	F
F	F	F	T	T	---	---	F	---
F	F	F	T	F	---	---	T	T
F	F	F	F	T	---	---	F	---
F	F	F	F	F			T	F

Let's move on to the simplest sentence remaining, premise two: $\neg D \rightarrow B$. This will be true in every row *except* where D is false and B is false. So let's do that. And wherever is false, the row can't be a counterexample, so we'll "cross off" the rest of that row:

A	B	C	D	E	(A ∨ B) → C	¬ D → B	¬ E	D ∧ ¬A
T	T	T	T	T	---	---	F	---
T	T	T	T	F		T	T	F
T	T	T	F	T	---	---	F	---
T	T	T	F	F		T	T	F
T	T	F	T	T	---	---	F	---
T	T	F	T	F		T	T	F
T	T	F	F	T	---	---	F	---
T	T	F	F	F		T	T	F
T	F	T	T	T	---	---	F	---
T	F	T	T	F		T	T	F
T	F	T	F	T	---	---	F	---
T	F	T	F	F	---	F	T	F
T	F	F	T	T	---	---	F	---
T	F	F	T	F		T	T	F
T	F	F	F	T	---	---	F	---
T	F	F	F	F	---	F	T	F
F	T	T	T	T	---	---	F	---
F	T	T	T	F	---	---	T	T
F	T	T	F	T	---	---	F	---
F	T	T	F	F		T	T	F
F	T	F	T	T	---	---	F	---
F	T	F	T	F	---	---	T	T
F	T	F	F	T	---	---	F	---
F	T	F	F	F		T	T	F
F	F	T	T	T	---	---	F	---
F	F	T	T	F	---	---	T	T
F	F	T	F	T	---	---	F	---
F	F	T	F	F	---	F	T	F
F	F	F	T	T	---	---	F	---
F	F	F	T	F	---	---	T	T
F	F	F	F	T	---	---	F	---
F	F	F	F	F	---	F	T	F

We'll wind up by calculating the values for premise 1, $(A \vee B) \rightarrow C$. Because of the truth table for conditionals, this will be true whenever C is true, and true whenever the disjunction $(A \vee B)$ is true. It'll be false otherwise.

A	B	C	D	E	$(A \vee B) \to C$	$\neg D \to B$	$\neg E$	$D \wedge \neg A$
T	T	T	T	T	---	---	F	---
T	T	T	T	F	T	T	T	F
T	T	T	F	T	---	---	F	---
T	T	T	F	F		T	T	F
T	T	F	T	T	---	---	F	---
T	T	F	T	F		T	T	F
T	T	F	F	T	---	---	F	---
T	T	F	F	F		T	T	F
T	F	T	T	T	---	---	F	---
T	F	T	T	F		T	T	F
T	F	T	F	T	---	---	F	---
T	F	T	F	F	---	F	T	F
T	F	F	T	T	---	---	F	---
T	F	F	T	F		T	T	F
T	F	F	F	T	---	---	F	---
T	F	F	F	F	---	F	T	F
F	T	T	T	T	---	---	F	---
F	T	T	T	F	---	---	T	T
F	T	T	F	T	---	---	F	---
F	T	T	F	F		T	T	F
F	T	F	T	T	---	---	F	---
F	T	F	T	F	---	---	T	T
F	T	F	F	T	---	---	F	---
F	T	F	F	F		T	T	F
F	F	T	T	T	---	---	F	---
F	F	T	T	F	---	---	T	T
F	F	T	F	T	---	---	F	---
F	F	T	F	F	---	F	T	F
F	F	F	T	T	---	---	F	---
F	F	F	T	F	---	---	T	T
F	F	F	F	T	---	---	F	---
F	F	F	F	F	---	F	T	F

Whoa! We immediately find a counterexample at line two! This argument is *invalid.* There's no need to go any further! Just highlight or circle the counterexample, and write 'INVALID' next to it.

That's all there is to using truth tables to evaluate arguments for validity. Pain in the butt, isn't it? Once you get the idea, you can do it in your sleep, which is a good thing, since these are so tedious that they will *put* you to sleep. Let's find a better way.

Appendix I: Syntax and semantics

What we've done here is give a purely semantic analysis of the logical concepts, that is, an analysis in terms of the *referents* of the sentences - their truth values. Validity, for example, has been defined just in terms of patterns of truth values. We haven't really shown how to *derive* one sentence from another in terms of manipulating its structure or otherwise transforming it. We'll refer to what we've talked about here as *semantic validity* or *semantic entailment,* and we'll use the single turnstile symbol '$\vdash$' to represent semantic entailment. To say that a set of sentences $\{\varphi, \psi,. . .\pi\}$ semantically entails a conclusion σ, we'll write $\{\varphi, \psi,. . .\pi\} \vdash \sigma$. This doesn't tell us anything about the sentences involved except their truth value distributions: It tells us that if all the sentences in $\{\varphi, \psi,. . .\pi\}$ are true, then so is σ.

In the next chapter, we'll work with the idea that you can manipulate sentences, breaking them down and building them up, to transform them into other sentences which are guaranteed to have the same truth value as the original sentence. That is, we'll show how to *derive* some sentences from other sentences, in a way that preserves the truth of the original sentences. But this is a *syntactic* notion - it involves manipulating and transforming the syntactic structure of sentences or propositions, to derive one sentence from another. We'll call this relation *syntactic derivability* or *provability*, and we'll use the double turnstile symbol '$\models$' to indicate it. $\{\varphi, \psi,. . .\pi\} \models \sigma$ means that the set of sentences $\{\varphi, \psi,. . .\pi\}$ derives the sentence σ, or that theres a proof leading from $\{\varphi, \psi,. . .\pi\}$ to σ. Of course, this will be relative to a particular system of proof rules, such as we'll describe for PL in the next chapter. So really, you can't simply evaluate claims like $\{\varphi, \psi,. . .\pi\} \vdash \sigma$ and$\{\varphi, \psi,. . .\pi\} \models \sigma$ on their own, but only within a system such as PL. Theres no such thing as derivability or entailment simpliciter, but only derivability and entailment within a system. So we shall subscript the turnstiles with some notation indicating what system we mean: for example, $\vdash_{PL}$ and $\models_{PL}$ mean entails-in-PL and derives-in-PL respectively.

I bring this up because it's of great interest to logicians whether the syntactic and semantic notions of validity and derivability actually coincide within any given system. We'll discuss this - the completeness and soundness of a logical system - in a later chapter.

Appendix II: Logical truth and Analytic truth

Summary of tables and definitions

Conjunction

φ	ψ	φ ∧ ψ
T	T	T
T	F	F
F	T	F
F	F	F

Disjunction

φ	ψ	φ ∨ ψ
T	T	T
T	F	T
F	T	T
F	F	F

Conditional

φ	ψ	φ → ψ
T	T	T
T	F	F
F	T	T
F	F	T

Biconditional

φ	ψ	φ ↔ ψ
T	T	T
T	F	F
F	T	F
F	F	T

Negation

φ	$\neg\varphi$
T	F
F	T

Logical Truth:

φ is logically true (a logical truth) iff φ is true on every row of its truth table.

Contingency:

φ is contingent (a contingent sentence) iff φ is true on at least one row of its truth table, and false on at least one row.

Logical Falsity:

φ is logically false (a logical falsehood) iff φ is false on every row of its truth table.

Logical Equivalence:

φ and ψ are logically equivalent iff they share the same truth value on every row of a single truth table.

Consistency:

A sentence φ is consistent (with itself) iff it is true on at least one row of its truth table.

Consistency:

A set of sentences $\{\varphi, \psi, \ldots \pi\}$ is consistent iff there is at least one row of its truth table where every sentence in $\{\varphi, \psi, \ldots \pi\}$ is true.

Validity:

A argument with the premises $\{\varphi, \psi, \ldots \pi\}$ and the conclusion σ is valid iff, on *every* row where *all* the premises $\{\varphi, \psi, \ldots \pi\}$ are true, the conclusion σ is also true.

Glossary

Chapter 3
Derivation and Proof

Index

I) Introduction

The truth-table method presented in the last chapter is powerful and simple. Unfortunately it's, also quite cumbersome and tedious. Just try, for example, to decide whether a sentence like

$(A \wedge B) \leftrightarrow R) \vee (T \wedge (D \rightarrow C))$[6]

is a tautology! That's 2^6, or 64, truth-table rows! We can use shortcuts to eliminate a large number of rows, but even still, it's pretty tedious just to set the thing up.

Moreover, the table method is kind of "unnatural", isn't it? Although it's useful for checking results, it really doesn't shed much light on how it is that thinking, speaking, information-processing creatures (like us) actually reason about the information we have. It doesn't seem to illuminate the "connections" between sentences in any particular way. There has *got* to be a better way than this!

Luckily, there is a better way. This is the method of syntactic derivation. This is a method that roughly mimics the step-by-step reasoning process we ordinarily use in everyday argument and reasoning. And this is one thing that should be clear: proofs or derivations aren't some new "magic" sort of "intuition" or other mysterious process. The method of proof simply codifies our ordinary reasoning processes and allows us to write down our reasoning in a standardized form. In this form, we can use the principles *that we already know are good inferences*, and use the standard form to keep track of assumptions we've made and what conclusions follow from which assumptions. For we'll see that in most reasoning, errors arise not from the use of flawed principles, but because we've either misremembered what we've already proven or we've forgotten that some information was based on a contentious premise (one we hadn't yet demonstrated).

6 Recall that a sentence like this should technically have another set of surrounding parentheses, although our convention is to drop the outermost pair. On occasion, I'll leave the outermost pair on, usually to help set the PL sentence off from the surrounding text. Nothing "logical" is intended by the choice to leave or drop these parentheses.

II) Arguments, proofs and derivations

The nature of derivations

Let's think of a *derivation* as simply a set of sentences arranged from top to bottom. Of course, we want this list to have a certain structure; it's supposed to represent some information you start with (we'll call these the premises), information you hope to get to (the conclusion), and the steps you take along the way, Each of these elements will be represented in our derivations, along with the evidence that these were legitimate steps. And for convenience, we'll assign every sentence a number, so we can easily refer back to that sentence by its number.

A formal derivation will arrange the list with the premises at the start. Under the premises will be a horizontal line separating them from all the other sentences. For example,

1) P(a)

2) (P(a) $\rightarrow$ R(a, b) $\vee$ P(b))

A list like this is part of a derivation, but it isn't *deriving* anything! There's no conclusion. Alas, this is the human condition, isn't it? In the everyday practice of most philosophy or science, we don't know beforehand what conclusion we are supposed to draw. All we have is bunch of information, in the form of experimental data, or (for example) ethical intuitions. The world doesn't tell us what the conclusion is supposed to be. After all - if we already knew what the conclusions were, why would we need logic? Most of the time, we don't even know what possible conclusions there are - or even that there *is* a conclusion!

It's not our purpose to throw you into the deep end like that, though. All the derivations we discuss will have conclusions, a set goal for you to arrive at. These will be the final sentence in the list.

Sometimes they'll be indicated with the sign ∴, or perhaps the phrase 'Q.E.D' ('Quod Erat Demonstrandum', which is Latin for 'I told you so!') So a derivation will look like:

1) P(a)	Premises
2) (P(a) → R(a, b) ∨ P(b))	Premises
3) S(a)	conclusion

Now, you've probably looked at this derivation and quickly noticed something. Where the heck did S(a) come from? The premises don't even mention squirrels, or anything even related to them! There's no way that those premises entail that conclusion! (A quick truth table will verify this). So it's a bad derivation, or an invalid derivation, as we discussed last chapter. But it's still a derivation. Let's formally define a derivation as *a set of sentence in which the last sentence is intended to be derived from the earlier sentences.* We'll use the term 'proof' to refer to a valid derivation.

The above is a derivation in the technical sense. But it's missing so much of what we want. Even if it was valid, it doesn't tell us why it's valid. This derivation doesn't say exactly how we got from the premises to the conclusion; i.e., it offers no justification for itself or its parts. In this example, such justification would probably have been pretty simple. But in many other arguments, there might be many steps in between premises and conclusions, and each step would require a justification. What we'll do now is add a justification column to our derivations. A derivation will look like:

1) P(a)	Premise
2) P(a) → (R(a, b) ∨ P(b)	Premise
3) S(a)	"Because I said so"

The column on the far right is our justification column. Every line requires a justification. In the case of our premises, the justification is simply "it's the premise", or "Premise", for short. The rest of this chapter will discuss what the legitimate justifications are. Obviously the rule of BISS used in line 3 will not be one of them! (It's just a placeholder for now.)

Justification in general

Justification in PL usually means citing some legitimate principle of inference (not BISS!) Sometimes all you have to do is cite whichever principle you're using, such as in the "Premise" rule. Or, for example, if you have a sentence which is a fundamental principle of PL, such as "$P \vee \neg P$", you can write it down any time and justify it by noting that it's a fundamental principle (in this case, the law of the excluded middle). This would look like:

:	
5) $P \vee \neg P$	Excluded Middle
:	

(Since the excluded middle is a fundamental principle, it needs no further justification. Which theorems are fundamental principles? We'll get into that later in this chapter).

If these were the only means of justification, we'd be might pretty limited. (After all, there's only a finite (and small) number of fundamental principles, although that actually turns out to not be so much of a problem.) On the other hand, a process of simply writing down theorems would surely obscure what we really think we're up to when we prove, or derive, or argue: we are trying to extract some information from the information we already have. So what we really want are *principles of inference* that allow us to show how that information is being extracted, and what it's being extracted from. Such justifications will typically mention both a principle of inference, and the line numbers of the sentences that we are inferring from. For example:

1) P	
2) $P \to Q$	
3) Q	$\to$ elim, 1, 2

What we're saying in the justification for line 3 is that the information (Q) is already present in lines 1 and 2, and that we're use the principle or *rule* of $\to$ elim to extract it at 3. (We'll discuss $\to$ elim in a few paragraphs). Check for yourself with truth tables: this is a valid inference.

Summary

A proof is a set of sentences. The first sentences are premises, separated from the rest by a divider line. The last sentence is the conclusion. Every sentence after the premises requires a justification. In a valid proof, every justification must be either a legitimate fundamental theorem of PL, or a legitimate rule of inference applied to some information that is, in turn, legitimately justified. Ultimately, this further information is justified by being directly inferred from the premises.

III) Rules of Inference

A) Overview

There are many different (yet equivalent) systems of rules of inference (henceforth *ROI*s). Some systems seek to have an absolute minimal number of 'fundamental' rules, and derive all the other rules from this foundation. Some systems feel that certain rules are intuitively more "fundamental" or "basic" than others. For our part, I'll present a system of rules in which each of the basic connectives ($\rightarrow$, $\vee$, $\neg$, etc.) has a rule for *eliminating* it, or breaking up sentences involving the connective, and a rule for *introducing* it, or constructing sentences involving the connective.

As it turns out, this isn't quite the most efficient way to construct a derivation system for PL. But I do think it's a very intuitive way, and a very language - oriented way, of understanding logic. Just remember: every sentence, no matter how complex, is defined by its central connective. And every connective has rules for breaking it down, and for building it up.

B) **Rules for Conjunction**

Conjunction-junction, what's your function?

These are about the simplest rules of all. So they make a good starting point! Our first rule allows us to eliminate or break apart conjunctions.

Conjunction Elimination (∧ elim)

Suppose you know that Albert is a pirate and a squirrel, or, in PL, (P(a) ∧ S(a)). We can break apart the conjunction to isolate each of its parts. You don't have to write them both down, and it doesn't matter which conjunct you pick. You can only write one down per new line. For each one you extract, cite the line number of the conjunction you extracted it from. For example:

:	
7) P(a) ∧ S(a)	BISS
8) S(a)	∧ elim 7
:	

In this example, we're just interested in line 8, so it doesn't matter where 7 came from (hence its justification in terms of BISS). The justification for line 8 tells us that we extracted S(a) by eliminating the conjunction in line 7. This rule is guaranteed to always preserve truth - if the information in line 7 is true, the result of applying the ROI will also always be true. See for yourself by using a truth table.[7]

You could also derive the further line:

7 Technically, it will take more than a truth table to really prove this, but that sort of meta-logical proof is a topic for a much later chapter.

:
9) S(a) ∧ elim 7

There's also no reason a line has to be justified by the immediately preceding line. Lines 7 and 8 could have been separated by twenty other lines, it wouldn't matter. If Albert is a pirate squirrel, then he's going to be a squirrel at any point in the proof. After all, our arguments aren't supposed to *change* the information, merely to extract what information is already there!

In our metalanguage, we can state this rule generally as:

Conjunction elimination

m) φ ∧ ψ

n) φ (alternatively, ψ) ∧ elim n

If a sentence of the form **φ ∧ ψ** is on line *m*, then you may derive a sentence of the form **φ** (or, alternately, of the form **ψ**) on any later line *n* within the scope of *m*, citing '∧ elim m' as your justification.

This may seem like a lot of muhey for such a simple and obvious principle. But it will be useful to see how the formal rules work in these simple situations, since they really work the same way even in the most complicated cases as well. We'll discuss the point about scope in the next chapter; for now, every line is within the scope of every other line, so it isn't an issue.

Conjunction Introduction (∧ intro)

Suppose you know that Albert is a pirate, and also that Albert is a squirrel. You can "glom" these two bits of information together into one complex claim: Albert is a pirate and Albert is a squirrel[8]. Technically, for any two sentences you already have, you can form a new sentence that simply

8 In colloquial English, we'd say 'Albert is a pirate squirrel', but see Ch I.

"conjoins" the smaller sentences with a ∧ symbol. For example:

:	
7) P(a)	BISS
8) S(a)	BISS
9) P(a) ∧ S(a)	∧ intro 7, 8
:	

Again, it doesn't matter if 9 immediately follows 7 and 8; and it doesn't matter if 7 and 8 are right "next to" each other. So the following would be fine:

7)	P(a)	BISS
:		
24)	S(a)	BISS
:		
1,317)	P(a) ∧ S(a)	∧ intro 7, 24

(You won't get any 1, 317 line proofs, though!) However, you must cite *both* sentences that are being glommed together!

Again, we can state this rule with full generality in our metalanguage:

Conjunction introduction

m) φ	
n) ψ	
o) $\varphi \wedge \psi$	$\wedge$ intro m, n

If a sentence of the form $\boldsymbol{\varphi}$ is on line *m*, and a sentence of the form $\boldsymbol{\psi}$ is on some later line *n* within the scope of *m*, then you may derive a sentence of the form $\boldsymbol{\varphi \wedge \psi}$ on any later line o within the scope of *m* and *n*, citing '$\wedge$ intro m, n' as your justification.

The way we've formulated this rule, the order of the sentences is important. For example, the following is not a legitimate application of $\wedge$ intro:

:	
7) P(a)	BISS
8) S(a)	BISS
9) S(a) $\wedge$ P(a)	$\wedge$ intro 7, 8 **INVALID!**
:	

However, it certainly looks like the order shouldn't make any difference (and truth tables will confirm this). One of your exercises will be to demonstrate that you can indirectly get this result.

Reiteration

Suppose you know that Albert is a pirate. As we said, derivation doesn't change any information, so at any later point of the proof, Albert is still a pirate. Merely proving things about him will not make him change his piratical ways! So if Albert is a pirate on line 2, you can restate or *reiterate* this fact on any later line. For example:

2) P(a)	BISS
:	
16) P(a)	Reit 2

This might seem like a pointless, dumb, rule. Sometimes, however, it can make a proof much easier to "see", by placing relevant information that came from different parts of the proof next to each other.

In fully general, metalinguistic form:

Reiteration

m) φ	
n) φ	Reit m

If *m* is a sentence of the form φ, you may state φ at any later line *n* within the scope of m, citing as your justification 'Reit m'

This exhausts the basic rules governing conjunction! There will be some derived rules we'll discuss later, but all they do is speed proofs up slightly. They aren't at all necessary, given these basic rules.

Side Note: two points about ROIs

The first point - and this is important! - is that ROIs may *only* be applied to the *central connective* of a sentence. They work on sentences as a whole, not on parts of sentences. Why is this?

Suppose you knew that Reagan was either a Republican or a liberal Democrat:

$R(r) \vee (L(r) \wedge D(r))$

It would surely not follow from this proposition that Reagan was a liberal, would it? After all, it's not even certain from this proposition that he's a liberal Democrat. In fact, he's a Republican, and not a liberal Democrat (or any other sort of Democrat), and that's completely consistent with this proposition. (Consider the central disjunction) So we can't simply proceed with:

:	
7) $R(r) \vee (L(r) \wedge D(r))$	
8) $L(r)$	$\wedge$ elim 7
:	

That would be a *terrible* inference. This sort of inference is prevented by our rule that ROIs may only be applied to the central connective of a sentence.

In a sense, line 7 has *less* information than its right disjunct $(L(r) \wedge D(r))$. $(L(r) \wedge D(r))$ tells you that Reagan is definitely a liberal Democrat, but line 7 only says he might be a liberal Democrat. Just because some proposition follows from *part* of a sentence doesn't mean it follows from the *whole* sentence. As a limiting (and obvious) case, notice that A obviously follows from A. But even though A is a constituent of the sentence $\neg$ A, it surely does not follow from $\neg$ A!

The second point is about complexity. In our examples, we've used very basic conjunctions - conjunctions consisting of just two atomic conjuncts. But the rules can be applied to the central connectives of any arbitrarily complex conjunction (the fully general formulations reflect this - since they apply to *any* sentence of the form **φ ∧ ψ** , no matter how complicated **φ** and **ψ** are themselves.) So we could easily apply ∧ elim in the following way:

:	
7) (C(r) → F(r)) ∧ (L(r) → D(r))	Premise
8) (L(r) → D(r))	∧ elim 7
:	

After all, our rule says that we can take any sentence that's a conjunction, and break it up into its conjuncts. In this case, the conjuncts are themselves complex (they're conditionals). And we could easily reverse that reasoning to apply ∧ intro in the following way:

:	
7) C(r) → F(r)	BISS
8) L(r) → D(r)	BISS
9) (C(r) → F(r)) ∧ (L(r) → D(r))	∧ intro 7,8
:	

After all, the rule ∧ intro tells us that we can take any two sentences - no matter what their internal form is - and glom them together to form a conjunction of the two sentences. And if the original sentences are true, the conjunction itself will be guaranteed to be true.

C) Rules for Disjunctions

Disjunction Introduction ($\vee$ intro)

Suppose you knew that Albert was a squirrel. Then wouldn't it follow that he was either a squirrel or a hamster? In fact, wouldn't it follow that either Albert was a squirrel or . . .your mother is a hamster? In fact, as long as he's a squirrel, it'll be true that either he's a squirrel, or (insert anything here). So, the following inference is valid:

:		
7)	P(a)	
8)	P(a) $\vee$ H(m)	$\vee$ intro 7
:		

H(m) may be whatever you want! (or as we say: 'The inference is valid, for all values of Hamster'.) I specifically chose H(m) to be a totally arbitrary sentence. Just as with our conjunction rules, the sentence you add on with the disjunction may be arbitrarily complex; and of course the sentence you are adding on to may be arbitrarily complex. And just as with our conjunction rules, there's no reason that the two sentences you put together with $\vee$ intro must be right next to each other. In our fully general formulation:

Disjunction introduction

m)	φ	
n)	$\varphi \vee \psi$	$\vee$ intro m

If a sentence of the form **φ** is on line *m*, then you may derive a sentence of the form **$\varphi \vee \psi$** on any later line *n* within the scope of *m*, citing '$\vee$ intro m' as your justification.

This may seem like a strange rule. In a sense, you've taken some definite information that you had (that Albert is a squirrel) and "weakened" it to something less definite. From S(a) you can tell that Albert is a squirrel, but from (S(a) ∨ P(a)) you can't really tell whether or not he's a squirrel. . .or a pirate. He might be a squirrel, or instead a pirate, or even both! That's why this rule is sometimes called "weakening" - you "weaken" the knowledge that you have.

Why would we ever use it? Why would we ever want to, in this way, "lose" definite information that we had? Well, sometimes you want to show that certain kinds of sentences are equivalent to certain other kinds, and you need to "force" sentences into a certain kind of structure. In these cases, ∨ intro can be very useful. We'll demonstrate this later.

Disjunction Elimination

There is no disjunction elimination! Don't try it. Well, actually, there sort of is, but we're going to call it something else, and it doesn't work anything like how it sounds. It doesn't work anything like conj elim, for starters. After all, suppose you know that Reagan was either a Democrat or a Republican (this is true, given the nature of the U.S. political space, right? Actually, it's true, simply given that Reagan was a Republican). From merely knowing that he belonged to one of two parties, you surely can't deduce which party it is! So the following is INVALID:

:		
m)	R(r) ∨ D(r)	
n)	R(r)	∨ elim m **INVALID!**

In this case, even though R(r) is true, it surely doesn't follow from line n! I'm going to the trouble of even mentioning this because it's such a common fallacy. Logic students try to do this all the time. Part of that is because we encourage students to not "over-think" the proofs and to simply apply the rules mechanically. But if you stop very briefly and think about what a disjunction is, you'll see why this is a bad inference.

That exhausts the basic rules for disjunction. Later we'll introduce some derived rules for disjunctions that will speed proofs up, but these rules are all we need for now.

D) Rules for Negation

Double Negation Elimination (2NE)

Suppose, as the old blues song goes, you ain't misbehavin'. This is just a roundabout way of saying that you are behavin', right? (Right.) In PL, double negatives "cancel each other out" in this way. So the following inference is valid:

:		
m)	$\neg \neg B(a)$	
n)	$B(a)$	2NElim (or 2NE), m
:		

Double Negation Introduction (2NI)

The same pattern of reasoning holds in reverse: If you're behavin', then you ain't misbehavin'. So the reverse pattern is also justified:

:		
m)	φ	
n)	$\neg \neg \varphi$	2NIntro (or 2NI), m
:		

Formally, this can be stated in full generality as:

Double Negation Elimination (2NE)

m)	$\neg \neg \varphi$	
n)	φ	2NE, m

If line *m* is a sentence of the form **$\neg \neg \varphi$,** you may derive a sentence of the form **φ** on any later line *n* within the scope of *m*. Cite as your justification '2NE m'.

and:

Double Negation Introduction (2NI)
m) φ
n) ¬ ¬φ 2NI, m

If line *m* is a sentence of the form **φ,** you may derive a sentence of the form **¬ ¬φ** on any later line *n* within the scope of *m*. Cite as your justification '2NI m'.

Keep in mind that the negation rules are ROIs just like the others, and the general points apply to them as well. They can apply where the sentence φ is arbitrarily complex or atomic. One application of this that doesn't automatically occur to many is that you could use DNE to "strip off" two out of three negations in the following way:

:

m) ¬ ¬ ¬φ

n) ¬ φ 2NE m

Also, 2NE and 2NI may (for now) only be applied to the sentence as a whole, and not to subsentences, just like our other ROIs.

Single Negations

There actually is a single negation introduction rule. But, much like the earlier discussion of disjunction elimination, *it doesn't work the way you think it works!* And it certainly doesn't work like the conjunction rules work. You certainly are *never* allowed to infer thusly:

:

m) ¬φ

n) φ ¬elim m **INVALID!**

Again, I've brought this up because logic students sometimes get a bit carried away by the apparent symmetry of the elimination and introduction rules. But the symmetry is only apparent - a moment's reflection will convince you that this inference is about as bad as an inference could be. Not only is it invalid, in that it is not guaranteed to take you from truth to truth, but it *is* guaranteed to take you from truth to falsity!

These are all the negation rules we need for now.

Side Note: logic and language

Natural languages are a rich phenomena. There's an old story where a philosopher named Sidney Morganbesser was attending a linguistics lecture, sitting in the back and cracking wise, as philosophers are wont to do. The linguist giving the lecture explained: "In many languages, such as English, a double negative is really a positive. In some other languages, such as (he inserted some strange tongue here), a double negative is an emphatic negative. But there is no language in which a double positive is a negative!"

Morganbesser sardonically replied: "Yeah, right."

The linguist understated the case: even in English, sometimes a double negative is an emphatic negative. And there's all sorts of subtle vocal intonations or word combinations ("yeah, yeah", or "yeah, right") that can influence what propositions are actually expressed by our words. Contextual features of our conversations also affect what propositions our words express. But in PL, we abstract away from all this. In PL, a double negative is a positive, and that's the end of the story.

Does this mean PL is simply a useless and rarefied abstraction, that has little to do with what language users do? Not at all. First, think of meaning as being a combination of many different factors. Logic (PL) helps us understand the way that words and rules for combining words contribute to meaning. Other studies, such as formal pragmatics, help us understand the way contextual features interact with this "literal meaning". Second, think of logic as the study of reasoning. What PL does is help us understand the relationships between propositions, after we've determined what propositions have *actually* been expressed by our sentences.

E) Rules for conditionals

Conditional elimination (→ elim, or *Modus Ponens*)

Suppose we know that Albert is a pirate. What do we know about pirates? They're all bloodthirsty! So we know that if Albert is a pirate, then Albert is bloodthirsty. Let's express these two things we know about Albert as premises:

1)	P(a)	Premise
2)	P(a) → B(a)	Premise

It seems pretty obvious that we can conclude that Albert is bloodthirsty. But this pattern of reasoning doesn't have to involve the same individual on both sides of the conditional. Suppose we know that Donald is a hamster. And suppose we also know that if Donald is a hamster, then Marie smells of elderberries:

1)	H(d)	Premise
2)	H(d) → E(m)	Premise

Clearly, we can conclude that Marie smells of elderberries, right? Think of a conditional as a locked door. The antecedent is the keyhole. So if you've got a sentence that "fits" in the keyhole, you can open the door and get to the other side (the consequent). That's why this rule is sometimes called detachment - it allows you to detach the consequent. Classically, it's also called 'Modus Ponens', the 'Mode of Putting'. I have no idea why it's the mode of putting, but Latin sounds cool anyways.

We can state this rule in full generality as:

Conditional elimination

m)	φ	
n)	$\varphi \rightarrow \psi$	
o)	ψ	$\rightarrow$ elim m, n

If line *m* is a sentence of the form $\boldsymbol{\varphi}$, and line *n* is of the form $\boldsymbol{\varphi \rightarrow \psi}$ within the scope of *m*, you may derive a sentence of the form $\boldsymbol{\psi}$ on any later line *o* within the scope of *m* and *n*.

Some points to note about this: First, the order of lines m and n doesn't matter. Second, just as with all other ROIs, it only works on conditional sentences - not on subsentences! And it only works where the antecedent of the conditional is "matched" by a complete sentence, not simply by subsentences! So the following are invalid:

1)	H(d) $\vee$ P(a)	Premise
2)	H(d) $\rightarrow$ E(m)	Premise
3)	E(m)	$\rightarrow$ elim 1, 2 **INVALID!**

1)	H(d)	Premise
2)	(H(d) $\rightarrow$ E(m)) $\vee$ P(a)	Premise
3)	E(m)	$\rightarrow$ elim 1, 2 **INVALID!**

And just like the other ROIs, it applies perfectly well where the individual sentences φ and ψ are themselves complex. For example, the following is perfectly *valid*:

1)	$H(d) \vee P(a)$	Premise
2)	$(H(d) \vee P(a)) \rightarrow E(m)$	Premise
3)	$E(m)$	$\rightarrow$ elim 1, 2

Conditional Introduction (→ intro)

This is a key rule. But it works in a very special way that will be better to explain in the next chapter.

F) Rules for biconditionals

Biconditional elimination (↔ elim)

A biconditional, remember, is really just shorthand for a two-way conditional. It simply says that two propositions have exactly the same truth conditions: one is true if, and only if, the other is true. So a biconditional is really just a pair of conditionals. A biconditional such as 'I know P if and only if I have a true justified belief that P' is really just a way of saying 'If I know P, then I have a true justified belief that P; and, if I have a true justified belief that P, then I know P'. Thus, a biconditional of the form $(\varphi \leftrightarrow \psi)$ is equivalent to $[(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)]$. Fully generally, we say:

Biconditional elimination
m) $(\varphi \leftrightarrow \psi)$
n) $[(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)]$ ↔ elim m

---

If line *m* is a sentence of the form $\boldsymbol{(\varphi \leftrightarrow \psi)}$, you may derive a sentence of the form $\boldsymbol{[(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)]}$on any later line *n* within the scope of *m*.

Obviously, this kind of inference is a two way street! Since the biconditional is just a shorthand for a pair of conditionals, we can also use the rule of:

Biconditional Introduction (↔ intro)

We've pretty much already justified this rule. If it's true that 'If I know P, then I have a true justified belief that P; and, if I have a true justified belief that P, then I know P', then that's really just a longhand way of saying that knowledge is equivalent to true justified belief. Formulated generally:

Biconditional introduction

m) $[(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)]$

n) $(\varphi \leftrightarrow \psi)$ $\leftrightarrow$ **intro** m

---

If line *m* is a sentence of the form $\mathbf{[(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)]}$**,** you may derive a sentence of the form $\mathbf{(\varphi \leftrightarrow \psi)}$ on any later line *n* within the scope of *m*.

The way we've formulated this rule, the order of the sentences in the biconditional is important. For example, the following is *not* a legitimate application of $\leftrightarrow$ intro:

:

n) $(P(a)\rightarrow F(a)) \wedge (F(a) \rightarrow P(a))$,

m) $(F(a) \leftrightarrow P(a))$ $\leftrightarrow$ intro n **INVALID!**

:

Look closely at the rule and you'll see why. However, it certainly looks like the order shouldn't make any difference (and truth tables will confirm this). A similar point holds for $\leftrightarrow$ elim. One of your exercises will be to demonstrate that, in fact, you can use $\leftrightarrow$ intro to derive a biconditional with any order of 'antecedent" or "consequent". Another will be to demonstrate that you can use $\leftrightarrow$ elim to derive a conjunction with any order of conjuncts.

G) The Excluded Middle

A proposition is either true or false. There's no 'maybe' or waffling. If it's unambiguous, what middle ground could there be? This is a basic fact of PL. It's so basic, it can't be derived from anything else. Thus, at any given point, we can state of any proposition that either it or its negation is true:

:		
n)	$S \vee \neg S$	**XM**
:		

S doesn't even have to appear beforehand! It could be a "Hamster" proposition, for all we cared.

Whatever S is, we know that $S \vee \neg S$ will be true. So in full generality:

Excluded Middle (XM)

:

m) $\varphi \vee \neg\varphi$ XM

:

On any line *m* you may write a sentence of the form $\boldsymbol{\varphi \vee \neg\varphi}$**,** for any value of $\boldsymbol{\varphi}$**.** As justification, simply cite XM.

Conclusion

There are all the rules we need to get the ball rolling. In the next chapter we'll add a powerful new technique dealing with reasoning via assumptions. There will be three more basic rules that use this technique - Negation introduction (or *Reductio ad absurdum*), Conditional Introduction, and Argument by Cases. More importantly, we'll introduce general proof strategy. After all, we'll have a bazillion rules, but what's really important is knowing when to apply which rule, and how to start and structure a derivation.

Appendix A:

Basic and derived ROIs

As we hinted at earlier, the notion of a basic *vs.* a *derived* rule is wholly relative. In fact, if a system of basic ROIs for PL is complete (there's enough rules to prove all the theorems of PL) and consistent (they don't prove anything that *isn't* a theorem of PL), there will be numerous alternative sets of ROIs that do exactly the same thing. For example, we've chosen to use AC as a basic rule, and the conditional equivalence rule will be a derived rule. In Benson Mates' classic logic textbook[9], this is done the other way around.

How do you decide which set of rules to accept as basic? There's many considerations. A pure logician might want to accept the smallest possible basic set of rules. Quine's system *NF*[10], for example, uses only three basic rules. Another system uses a connective called the *Sheffer Stroke* (equivalent to 'not. . .or. . .'), and gets away with only two basic rules. It's an interesting exercise to see just how few rules you can get away with.

Such parsimony isn't really a primary concern of mine here. I've chosen a system of basic rules that I think is intuitive and easy to remember. I think this system lends itself well to categorizing some basic operations we perform on sentences. One could, of course, choose fewer, or different rules - but this would be a different (equivalent) system.

---

9 See Mates, Elementary Logic, 1972
10 Quine, "New Foundations for Mathematical Logic", 1937

Summary of Rules

Conjunction elimination

m) $\varphi \wedge \psi$

n) φ (alternatively, ψ) $\wedge$ elim n

If a sentence of the form $\boldsymbol{\varphi \wedge \psi}$ is on line *m*, then you may derive a sentence of the form $\boldsymbol{\varphi}$ (or, alternately, of the form $\boldsymbol{\psi}$) on any later line *n* within the scope of *m*, citing '$\wedge$ elim m' as your justification.

Conjunction introduction

m) φ

n) ψ

o) $\varphi \wedge \psi$ $\wedge$ intro m, n

If a sentence of the form $\boldsymbol{\varphi}$ is on line *m*, and a sentence of the form $\boldsymbol{\psi}$ is on some later line *n* within the scope of *m*, then you may derive a sentence of the form $\boldsymbol{\varphi \wedge \psi}$ on any later line o within the scope of *m* and *n*, citing '$\wedge$ intro m, n' as your justification.

Reiteration

m) φ

n) φ Reit m

If *m* is a sentence of the form φ, you may state φ at any later line *n* within the scope of m, citing as your justification 'Reit m'

Disjunction introduction

m) φ

n) $\varphi \vee \psi$ $\vee$ intro m

If a sentence of the form $\boldsymbol{\varphi}$ is on line *m*, then you may derive a sentence of the form $\boldsymbol{\varphi \vee \psi}$ on any later line *n* within the scope of *m*, citing '$\vee$ intro m' as your justification.

Double Negation Elimination (2NE)

m) $\neg\neg\varphi$

n) φ 2NE, m

If line *m* is a sentence of the form **$\neg\neg\varphi$,** you may derive a sentence of the form **φ** on any later line *n* within the scope of *m*. Cite as your justification '2NE m'.

Double Negation Introduction (2NI)

m) φ

n) $\neg\neg\varphi$ 2NI, m

If line *m* is a sentence of the form **φ,** you may derive a sentence of the form **$\neg\neg\varphi$** on any later line *n* within the scope of *m*. Cite as your justification '2NI m'.

Conditional elimination

m) φ

n) $\varphi \rightarrow \psi$

o) ψ $\rightarrow$ elim m, n

If line *m* is a sentence of the form **φ,** and line *n* is of the form **$\varphi \rightarrow \psi$** within the scope of *m*, you may derive a sentence of the form **ψ** on any later line *o* within the scope of *m* and *n*.

Biconditional elimination

m) $(\varphi \leftrightarrow \psi)$

n) $[(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)]$ $\leftrightarrow$ elim m

If line *n* is a sentence of the form **$(\varphi \leftrightarrow \psi)$,** you may derive a sentence of the form **$[(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)]$**on any later line *n* within the scope of *m*.

Biconditional introduction

m) $[(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)]$

n) $(\varphi \leftrightarrow \psi)$ $\leftrightarrow$ **intro** m

---

If line *m* is a sentence of the form $\mathbf{[(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)]}$, you may derive a sentence of the form $\mathbf{(\varphi \leftrightarrow \psi)}$ on any later line *n* within the scope of *m*.

Excluded Middle (XM)

:

m) $\varphi \vee \neg\varphi$ XM

:

On any line *m* you may write a sentence of the form $\boldsymbol{\varphi \vee \neg\varphi}$, for any value of $\boldsymbol{\varphi}$. As justification, simply cite XM.

Glossary

Derivation: A set of sentences in which the first sentences are designated as the premises and the last sentences are designated as the conclusion. The conclusion is intended to follow from the premises.

Muhey: Malarkey, Confusion, Mayhem.

Proof: A valid derivation.

Chapter IV

Subproofs and Assumptions

Index

I) Introduction

In this chapter, we'll introduce three new rules. These rules are fundamentally different from the other rules. They involve a technique that is at the heart of all deeper reasoning and thinking: the technique of hypothetical reasoning.

Hypothetical reasoning is where imagination and creativity enter the picture in logic. Just because logic involves very strict rules doesn't mean it is rote or purely mechanical. A logician has to be creative in determining an overall proof strategy and in deciding where to start. And most of all a logician has to be creative and imaginative in deciding what new information he should temporarily assume.

'Wait', you say. '. . .You can just assume things, when you're trying to do proofs? What sorts of things can you assume?'

Whatever you want. You can assume whatever you want! 'But wait,' you say, incredulous. 'Why don't you just assume the conclusion?' Sure, you can assume the conclusion, if you like. But that's not likely to convince anyone who didn't already believe the conclusion! This is because the conclusion will now *depend* on the disputed assumption. So the goal is to either 1) only make assumptions everyone agrees upon, or 2) only make assumptions that *nothing depends on*.

In philosophical argument, we import uncontroversial assumptions all the time. Hardly an ethics debate goes by without using the assumption that the Holocaust was evil, or that slavery is wrong. And the cases are similar in metaphysics or any other part of philosophy. But even in these cases, it's important to remember that the conclusions still depend upon these assumptions. In a sense, the rule of excluded middle that we briefly discussed in CH III is an example of importing a certain logical assumption - namely the assumption that propositions are true or false, and there is no "middle" truth value. (Still, we won't call XM an assumption, and we will instead treat it as being specifically validated by a ROI.)

But this isn't what we'll do in PL. We will not simply import assumptions unannounced, no matter how uncontroversial or obvious they are. We're interested in the connections between propositions; specifically, the entailment relation. We want to demonstrate what propositions depend on what other propositions. And when we want to derive some conclusion R from premises P and Q, we want to demonstrate that R depends *only* on P and Q, and no other assumptions. How can we do this if we are assuming some new propositions to use in our derivations?

What we'll do in the rest of this chapter is to talk about formal ways of using assumptions. We'll talk about both making assumptions, and clearly marking what further claims rest on those assumptions. Finally we'll show how to "get rid of" those assumptions by showing that nothing really depends on them.

II) Everyday informal uses of assumptions

Before we discuss the use of assumptions in PL, let's think about how we use assumptions in everyday informal reasoning. Sometimes we argue like this:

> 'Jason makes several claims. He claims that you are in some financial situation. He claims that there are only two decisions one can make in this circumstance. He also claims that decision one (D1) will lead to poverty. He also claims that decision two will lead to defaulting on a loan. But defaulting on a loan will lead to poverty. So no matter what, (according to Jason) you are going to be poor'.

How do we get to the conclusion? You sort of try to imagine that you make decision one. That is, you *assume* for the sake of argument that you make D1. But then you know that that D1 leads to poverty, so *based on the assumption* that you make D1, you are poor. But of course you don't have to make D1. So then we imagine that we've made D2. Based on that assumption, you sort of informally apply $\rightarrow$ elim to derive the fact that you will be poor. But of course, *that* poverty depends on the assumption that you made D2, and you didn't have to make D2. Still, since you've got to make D1 or D2, it doesn't really matter which decision you make, right? So your poverty doesn't *depend* on either the assumption that you make D1, or the assumption that you make D2.

Or consider this sort of argument:

> 'Jason and I both agree on what propositions are asserted in the Bible. Jason goes on to claim that the Bible is 100% literally true. Now, I can't go and check the facts to test every single proposition of the Bible. But one of the things the Bible asserts is that no book is 100% true. So Jason's claim cannot be true'.

What went on here? First, we treat the propositions of the Bible as premises. Then, we reason like this: I disagree that the Bible is 100% true. But, *just for the sake of argument*, I'll pretend (or *assume*) that Jason's claim is true, just to see what follows. Well, if we pretend it's true, then it rather immediately follows that no book - including the Bible - is 100% true. Therefore, the assumption that Bible is 100% true contains a contradiction, and it must be false.

Did the conclusion *depend* on the assumption that the Bible is 100% true? Not really. We just pretended it was true, to show that it contained a contradiction. We use this kind of argument strategy all the time: Assume your opponent's claim just for the sake of argument, in order to show that the claim leads to contradiction. Obviously philosophical arguments are rarely as simple as above. Usually we have to go through quite a few steps to show that the assumption leads to a contradiction. The point remains, however: the conclusion, in these cases, doesn't depend on the assumption made for the sake of argument. After all, the only other possible assumption is that the claim is false - so no matter what assumption you make, the claim must be false.

These strategies lie all over reasoning and argumentation. The crucial, tricky part is keeping track of which propositions depend on which other propositions. In a nutshell, all PL really does for us is give us a very handy way of keeping track of these dependency relations, so that we can easily tell that our conclusions don't depend on any new assumptions. What we'll do now is to lay out a formal method for doing this.

III) Scope Lines (or dependency lines) and subproofs

A) Assumptions and subproofs

The first thing we should notice is that in any proof or derivation, the conclusion depends on the premises[11]. After all, that's what it means to *derive* the conclusion *from* the premises. So, in order to represent this, we'll draw what we will call a *scope line*.

1) P	Premise
2) P → Q	Premise
3) Q → R	Premise
4) Q	→ elim 1, 2
5) R	→ elim 3, 4

The line we just drew is the scope line. It indicates that everything below the horizontal line is under the scope of the things above it. That is, everything under the horizontal line depends on the things above it. In this case, the conclusion depends on the premises.

Suppose that we were kind of lame, and we didn't think we could get to line 5 just given the information we had (premises 1 - 3). Maybe we thought that in order to get to the conclusion, we had to assume that Jason's mother was a hamster. I don't know, some people seem obsessed with hamsters. As we said, you can assume whatever you want, but in a valid proof, the conclusion doesn't depend on anything but the premises. So let's represent this hamster assumption as follows:

11 If there are any - In Chapter VI, we'll demonstrate some very special proofs where the conclusion rests on no premises at all.

1) P	Premise
2) $P \rightarrow Q$	Premise
3) $Q \rightarrow R$	Premise
4) H(m)	Assumption
5)	?
6)	?

At 4, we've introduced our new assumption, that Jason's mom (m) is a hamster. And we've introduced it with a new scope line of its own, to show that everything after this depends on that assumption (in addition to depending on the premises). I have no idea where we could possibly go with this in lines 5 and 6, hence the question marks. The problem is, even if we did conclude R at line 6, it would depend on the assumption at (4). So let's introduce a new principle for doing proofs:

> In a valid proof of a conclusion ψ from premises $\{\varphi_1, \varphi_2, . . . \varphi_n\}$, the conclusion ψ must be on the same scope line as $\{\varphi_1, \varphi_2, . . . \varphi_n\}$; i.e. the "first" or "outermost" scope line, i.e., the line furthest to the left.

And let's introduce new rule, the *Assumption* rule:

> Assumption: You may write down any sentence φ whatsoever on any line, as long as it is written down inside a new scope line, and everything that follows remains within that scope line until the assumption is "discharged" or the subproof is ended. No information inside the scope line may be reiterated out of the subproof or otherwise brought out of the subproof except as specifically allowed by the rules. The justification is "Assume".

And while we're at it, a general principle about "ending" assumptions:

End Assumption: You may end an assumption at any point. The next line of the subproof is written down on the next scope line outside the "ended" assumption. No information from the ended subproof is available outside that subproof if the assumption is ended in this way.

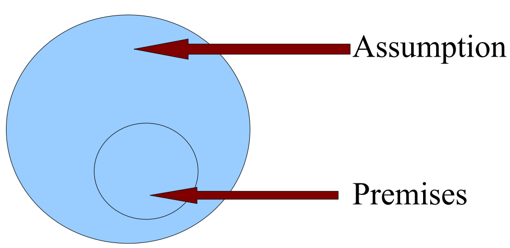
We can see right away that these two principles together limit the use of the assumption rule. If every assumption causes us to create a new scope line, but a valid proof requires the conclusion to be out on the first scope line, then how do we get information "out" of the scope of the new assumptions? In our "hamster" proof of R, even if we can now derive R, how are we going to get it back on the outermost scope line? Here's another way of looking at it: when we introduce an assumption, and prove that some conclusion follows from that assumption, we've created a *subproof*. Inside of any proof might be several subproofs, and our goal will always be to get information "out" of those subproofs and back into the main proof.

The subproofs are like little imaginary hypothetical worlds. When we make some assumption, we are imagining another possible world, and we're trying to see what would necessarily follow from our assumption about that possible world. And the outermost scope line- the one with the premises - is analogous to the actual world. When we want to prove that C holds in the actual world, it's no good to simply show that it holds in some other possible worlds that might or might not be like our world (that is, it's no good to show simply that C holds in the possible world we define with our new assumptions). What we really want to do is to somehow use our subproofs to show that C holds in *every* possible world consistent with the premises - or, in other words, that it doesn't matter which possible world or assumption we pick. If we can do that, it'll be tantamount to showing that C doesn't depend on our assumptions.

The key lies in the highlighted part of the assumption rule: You may write down any sentence **φ** whatsoever on any line, as long as it is written down inside a new scope line, and **everything that follows is within that scope line until the assumption is "discharged".** Discharging an assumption means to show that nothing depended on that assumption. The trick to using assumptions is in discharging them. As it turns out, there are very few ways to discharge an assumption. In fact, in PL, there's only three ways to discharge an assumption. So one should be very careful about what assumptions one makes, and in fact, you should *never make an assumption unless you know how you're going to discharge it.* Before we discuss discharging assumptions, lets stop for a moment and think about how information 'moves' across assumptions. Then we'll detail the rules for discharging assumptions now.

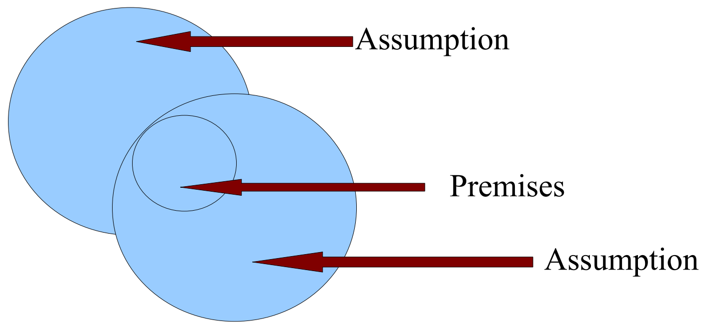
B) Subproofs and the flow of information.

As we've set up the assumption rules, information can only "move" in a certain way. When we make assumptions, we're *adding* those assumptions to the information we already have (e.g. the premises, and everything that follows from the premises). This means the information in the premises is always available inside the new assumptions. But the converse doesn't automatically hold! When we assume some new information consistent with the premises, that information (and the information that follows from it) doesn't automatically hold outside the assumption. Here's a Venn diagram:



Think of the spheres as representing information. All the information available in the premises is available within any assumption. But the assumption may be consistent with the premises, while adding new information. This new information wouldn't be available if you were considering only what information was contained in the premises.

Moreover, when we make concurrent assumptions, they might be completely independent of each other. We do this, for example, in the argument by cases technique. That would look like so:



This means that information can "move" over scope lines only in very particular ways. That is, it can be imported into or exported out of subproofs only in certain ways. If you follow the rules correctly, you'll ensure this. But it's sometimes helpful to see a visual representation of this, such as:

The green arrows represent the legitimate flow of information. Red arrows represent restricted flow.

In short, information may move down and to the right completely freely. It may not "hop" subproofs on the same scope line at all. And it can only move down and to the left (back "out" of the subproof) under very special circumstances, when the assumptions are discharged.

C) Scope and Available Information

Let's introduce some terminology here to talk about information. Subproofs are like those annoying Russian dolls that nest inside each other. How can we state his formally?

A subproof X is within the scope of an assumption (or premise) A *iff*: The subproof X begins on a line of some subproof (or proof) that itself begins with A, or, the subproof X begins on a line of some subproof that is itself within the scope of a subproof (or proof) that begins with A. We might also say that the subproof X is within the scope of the other subproof (the subproof beginning with A) as well.

Lines *within* a subproof are also said to be within the scope of whatever premises / assumptions their subproof is within. Sometimes we also say 'under the scope of'. The reverse relationship is 'having scope over'. An assumption A has scope over another subproof just in case that subproof is within the scope of the assumption A.

Example:

```
| 1) A
|-------
|    | 2) B
|    |------
|    |    | 3) C
|    |    |------
|    |    | 4) :
|    |    | 5) D
|    | 6) E
| 7) F
```

The subproof 3 - 5 is within the scope of the assumption B. Since B (and the corresponding subproof 2 - 6) is also within the scope of the premise A, the subproof 3 - 5 is also within the scope of A. Line 5, the proposition D, is also within the scope of the assumptions A and B.

So, A has scope over everything. B has scope over subproof 3 - 5, as well as line 6. C has scope over lines 4 and 5.

Another example:

```
| 1) A
|----
|     | 2) B
|     |----
|     | 3) C
|     | 4) D
|
|     | 5) E
|     |----
|     |     | 6) F
|     |     |----
|     |     | 7) G
|     |     | 8) H
|     |
|     | 9) J
|
| 10) K
```

In the above example, the subproof 2-4 is within the scope of A. Lines 3-4 are within the scope of B. But nothing else is within the scope of B or the subproof 2-4. Specifically, subproof 5-9 is *not* within the scope of B. Subproof 7-8 is within the scope of E, and since E is within the scope of A, subproof 6-8 is within the scope of A.

We've been talking about scope because it's of key importance in using information within a proof. Scope determines what information can be used at any given point. At any given line, you'll have a certain amount of information available to you. The information available at any given line *n* is the set of propositions that both

A) precede *n*, and

B) are in a subproof (or proof) that has scope over n.

Consider our second example above. Suppose you're on line 8 and trying to determine what information you can use to go on. You can use anything before line 8 as long as it's in some subproof that's got scope over 8. The subproof beginning with 6 has scope over 8, so you have the propositions F and G available to you. The subproof beginning with 5 has scope over subproof 6 - 8, so it's got scope over 8. So we have the proposition on line 5 (F) available to us. And of course the main proof - the one beginning with the premise at line 1 - has scope over subproof 5 - 9, and subproof 5 - 9 has scope over subproof 6 - 8. So line 1 has scope over 8, thus we have the proposition A available to us.

The first subproof, 2 - 4, *does not* have scope over 5 - 9 or 6 - 8. So *nothing* within 2 - 4 is available to us at line 8! At line 8, propositions B, C and D are off limits!

Why is some information "off limits" in this way? Well, it's because of what subproofs are supposed to be - imagined alternatives. When you imagine some possible world, and then you imagine a further more specific part of that world, everything from the first imaging holds in the more specific imagining. Suppose you imagine that you're a pirate. Then suppose you make the further assumption that you're also a squirrel (i.e., within the scope of the pirate assumption). All the pirate facts still hold, right? But suppose you're reasoning with (say) Argument by Cases. Suppose you're at a crossroads, with two roads A and B. You look on the map and you imagine taking road A, to see where it will go. You mentally trace road A all the way to its destination. Now you stop, and you imagine taking road B. This isn't a more specific version of road A, it's totally different! So as you trace your possible route on road B, none of the things that happened in your imagining of road B apply! In other words, the information within the imagined road A isn't available to you when you're imagining taking road B.

I realize the above examples and definitions are a bit hard to follow as you read. But this is why I've opted for a derivation system that uses scope lines. The scope lines allow you to easily and visually represent these concepts within a proof in a way that's very natural and easy to see. They allow you to very quickly see what lines are within the scope of what other lines, and consequently, they allow to to very easily and rapidly determine what information you have available at any given point.

Let's talk about the rules for discharging assumptions now.

III) ROIs involving subproofs

A) Argument by Cases (AC)

Think about that informal proof we did a few pages ago - the proof involving poverty. Lets take some steps to formalizing it. Our premises are these: 1) You must make either Decision 1 or Decision 2. 2) Decision 1 leads to poverty. 3) Decision 2 leads to defaulting on a loan. 4) Defaulting on a loan leads to poverty. And our conclusion is that you will be poor. So:

1) $D1 \vee D2$	Premise
2) $D1 \rightarrow P$	Premise
3) $D2 \rightarrow L$	Premise
4) $L \rightarrow P$	Premise
5) P	Conclusion

We don't need any subject-predicate structure in this derivation, so our lexicon is pretty obvious. Now, consider how we reasoned before. First we imagined that we picked D1, and concluded that if we did take D1, we'd be poor. Then we imagined that we picked D2, and we also concluded that we'd be poor if we made that choice, as well. So let's represent that thusly:

1) D1 $\vee$ D2	Premise
2) D1 $\rightarrow$ P	Premise
3) D2 $\rightarrow$ L	Premise
4) L $\rightarrow$ P	Premise
5) D1	Assume
6) P	$\rightarrow$ elim 2,5
7) D2	Assume
8) L	$\rightarrow$ elim 3,7
9) P	$\rightarrow$ elim 4,8
5) P	?

You can see that we've introduced two subproofs, the 5-6 subproof, and the 7-9 subproof. After 6 and after 9, we simply use the "end assumption" rule to, well, end our assumptions. The subproofs themselves are fairly straightforwards affairs involving good 'ol $\rightarrow$ elim. So how do we "get rid of" or *discharge* the assumptions made at 5 and 7?

The key in this piece of reasoning is premise 1: D1 $\vee$ D2. This tells us that we *have* to make one of the two decisions; i.e., that *at least one of* D1 or D2 is true. Well, if at least one is true, and both D1 and D2 both lead to P, then it doesn't matter which choice we make - we're going to be poor! So the conclusion, P, does not depend on either assumption. And we know this because in the two subproofs, we've run through all the possible cases. That's why this is called Argument by Cases - if you have some finite number of possible cases, and you can show that by assuming every possible case, you arrive at the same conclusion C, then you can discharge all the assumed cases and conclude that P. And as our justification we want to show that 1) there's only 2 options (cite the disjunction that gives those two options) and 2) that we've demonstrated (in subproofs) that each option leads to the same result (cite the two subproofs). We can state this wholly generally in our metalanguage:

Argument by Cases (AC)

m) $\varphi \vee \psi$	
:	
n) φ	Assume
:	
o) π	
:	
p) ψ	Assume
:	
q) π	
r) π	AC m, n-o, p-q

If:
A) Line *m* is some disjunction $\varphi \vee \psi$,
B) Lines *n-o* is a subsequent subproof directly in the scope of line *m,* beginning by assuming the first disjunct (φ) and ending in some subconclusion π,
C) Lines *p-q* is is a subsequent subproof *on the same scope level as the subproof in n-o*, beginning by assuming the subsequent disjunct (ψ) and ending in the same proposition π from subproof *n-o*,
Then you may discharge the assumptions φ and ψ, and write down π on the same scope line as the original disjunction. For justification, cite AC m, n-o, p-q.

B) Conditional Introduction (→ Intro)

Suppose you didn't want to prove anything about what state a particular object was in, but rather, you just wanted to show that certain states were connected. This happens in science all the time: Scientists prove laws that govern the way objects behave when they're in certain states, but scientists rarely care what states particular objects happen to be in. For example, a scientist might demonstrate that *if* you run a current through some object, that object will radiate a lot of heat, or transmit the current very well, or whatever. And that's more important than showing that the object is in fact radiating a lot of heat. So the moral of the story is that sometimes (often!) you want to prove a conditional fact: that if one thing holds, another also holds. In PL, we often want to prove statements of the form P → Q, rather than simply demonstrating Q.

Now, if you wanted to prove that one thing (P) led to another (Q), wouldn't it make sense to imagine that P was true? Or, in other words, to assume P? Then suppose we've assumed P, and we somehow, with the help of our other premises, derive Q. Well, we haven't then shown that Q is really true, but we have shown that *if* P is true, Q is also true. Q isn't true on it's own, but it depends on P. But if that's the case, haven't we then shown, independently of our assumption that P, that P → Q? Let's illustrate with something simple.

Suppose we know that if if you push Albert, he'll move. And suppose we know that if Albert is in motion, then he will continue that motion unless acted upon by another force. That's Newton's second law, as applied to our squirrelly friend, right? It ought to then follow that if we push Albert, he'll remain in motion unless acted upon by some other force. We can start to formalize this as follows:

1) $P(a) \to M(a)$	Premise
2) $M(a) \to R(a)$	Premise
3) $P(a) \to R(a)$	Conclusion

We'll let P(a) represent *a is pushed,* M(a) represent *a is in motion*, and R(a) as *a remains in motion unless acted upon by another force*. [12]

In our informal presentation, we suggested that the way to prove some conditional statement like $P \to R$ was to assume the antecedent, P, and try to show that R follows from it (along with the premises). That suggests the following strategy:

1) $P(a) \to M(a)$	Premise
2) $M(a) \to R(a)$	Premise
3) $P(a)$	Assume
:	
5) $R(a)$	?
6) $P(a) \to R(a)$	?

If we could only fill in the gap from 3 to 5, we would have shown that the assumption P(a), together with the premises, led to R(a). But isn't that the same as showing that the premises alone entail the conditional fact $P(a) \to R(a)$? (Yes, it is). And as it turns out, the move from P(a) to R(a) is pretty easy. We'll just "daisy-chain" along using $\to$ Elim:

12 Clearly we could get fancier here: recall the translation schema for φ *unless* ψ, and you'll see that premise (2) actually has the more detailed structure
$M(a) \to (\neg A(a) \to R(a))$,
where A(a) is something like *a is acted upon by another force*. And we could probably represent *pushing* as P(x, y), or a two-place relation between the pusher and the pushed, for starters. But that's not necessary here.

1) $P(a) \to M(a)$	Premise
2) $M(a) \to R(a)$	Premise
3) $P(a)$	Assume
4) $M(a)$	$\to$ Elim 1, 3
5) $R(a)$	$\to$ Elim 2, 4
6) $P(a) \to R(a)$	?

All that remains is to discharge the premise in line 3. So to mirror our informal reasoning process above, let us adopt the rule of Conditional Introduction ($\to$ Intro). In order to prove some conditional such as $P \to Q$, begin by starting a subproof that begins with the antecedent P. If you can derive Q within that subproof, then you can end the subproof and state the conditional $P \to Q$ one scope line outside the original assumption of P. As justification, cite the subproof that proves that P leads to Q.

And now for the usual "technical formulation":

Conditional Intro ($\to$ Intro) :

:	
m) φ	Assume
:	
n) ψ	
o) $\varphi \to \psi$	$\to$ Intro m-n

If lines *m* - *n* comprise a valid subproof beginning at *m* with the assumption φ and ending at *n* with the proposition ψ, you may end the subproof, discharging the assumption φ and writing the conditional $\varphi \to \psi$ on a following line *o*, one scope line outside the subproof *m* - *n*. Cite $\to$ Intro m-n as your justification.

C) Contradiction Introduction (⊥ Intro)

This next rule deals with contradictions quite a bit. So it may be useful for us to introduce a sort of shorthand for noting that we have a contradiction. This rule doesn't really do any logical 'work", it simply makes some proofs a bit cleaner and neater.

Recall that the contradiction symbol is the upside-down "T"; i.e., ⊥. None of our derivation rules really give us a way of talking about this yet. The natural thing to do would be to say that whenever you have two propositions of the form φ and ¬φ, *one within the scope of the other* , you can simply write down on the next line that you've got a contradiction ⊥. As your justification, all you would need would be the line numbers of the two contradictory propositions φ and ¬φ, right?

Contradiction Introduction (⊥ Intro)

:	
m) φ	
n) ¬φ	
o) ⊥	⊥ Intro m, n

If lines *m* and *n* are contradictory propositions (i.e., of the form φ and ¬φ) and n is within the scope of m, ⊥ on any subsequent line *o* within the scope of both *m* and *n*. Cite as your justification the rule ⊥ Intro and the line numbers of the contradictory propositions, i.e., *m* and *n.*

D) Negation Intro (¬Intro, or RAA)

This rule is really more of a proof technique or strategy than just a rule. Think of it as "giving your opponent enough rope to hang himself". The central idea is just this: Contradictions are always false. So, if we can take some proposition P and together with the premises, derive a contradiction, then the contradiction must have been somewhere in the premises and P. But in a derivation, you're supposed to assume the premises are true. The premises aren't "up for grabs" the way the later assumptions are. So, if the premises are true, the contradiction must have been within P. If that's the case, then P must have been false; but if P was false, ¬P is true.[13]

We use this strategy all the time. Consider our earlier informal argument involving the 100% true book. Ordinarily, what happens is not that we've proven ¬P, but rather we've proven that P is inconsistent with our premises. In that case, *something* has to go: either we deny P, or we re-examine our premises. Sometimes that's the real value of philosophical argument - not that it necessarily convinces someone of a particular proposition, but that it forces them to re-examine their presuppositions and background beliefs. Or, if the argument technique is used in a cheesy legal thriller, the clever, heroic attorney uses it to show that the witness on the stand has uttered at least one falsehood (and is therefore not to be trusted).

> 'Mr. Ripper, you say that you were not at the scene of the crime at 12 because you were at the bar at 12, drinking Arrogant Bastard Ale. But you also say you were giving blood at 1. And this would mean you could not have been drinking beer at 12, because the doctors would not have let you give blood with alcohol in your system. So which claim are you lying about, Mr. Ripper? The Bar? Giving blood? Or. . .dare I suggest it. . .are you lying about *everything*"?[14]

In pure logic, however, we're interested in the connections and relationships between propositions. So

13 Of course, we might start from contradictory premises, that *couldn't* both be true together. Still, even in these cases, you'll wind up proving that *if* the premises were true, some further assumption leads to contradiction. But since the contradiction was already there in the premises, *any* further assumption will lead to contradiction. So a*nything* can be proven from contradictory premises. We'll discuss this later.

14 From the upcoming major motion picture, Albert the Christmas Squirrel: Defender of Justice.

if you're asked to derive some proposition P from a set of premises, those premises aren't revisable. If you can show that P's truth is inconsistent with the premises, then the truth of the premises guarantees the falsity of P - or rather, the truth of the premises guarantees the truth of ¬P. That doesn't mean the premises are true, of course - just that if they *were* true, ¬P *would* also be true.

Let's consider an example much like the crime drama example just given. Suppose we have the following premises: Jack D. Ripper gave blood at 1. If Jack was at the bar at 12, then Jack was drinking at 12. If Jack was drinking at 12, he did not give blood at 1. From this, we'll prove that Jack wasn't at the bar at 12.

1) Bar(j) → Drink(j)	Premise
2) Drink(j)→ ¬Blood(j)	Premise
3) Blood (j)	Premise
4) ¬Bar(j)	Conclusion

There's a very simple way to do this with the derived rule of *Modus Tolens*. But we don't have that rule yet, so we'll do this the longer way. Actually the very same strategy we use here will be used in proving the derived rule.

Our reasoning process goes like this: The bold prosecuting attorney says,

> "Well, assume you *were* in fact at the bar at 12. And you say that if you were in the bar, you were drinking. And if you were drinking, you wouldn't have been able to donate blood at 1. But you did donate blood at 1. So your assertion, if we assume it is true, that you were in the bar, leads to contradiction! Thus your assertion is false. You were *not* in the bar at 12!"

We can represent this like so:

1) Bar(j) → Drink(j)	Premise
2) Drink(j)→ ¬Blood(j)	Premise
3) Blood(j)	Premise
4) Bar(j)	Assume
5) Drink (j)	→ elim 1, 4
6) ¬Blood(j)	→ elim 2, 5
¬Bar(j)	?

We notice that line 3 contradicts line 6. Let's make that explicit by reiterating line 3 inside of the subproof, and then using the ⊥ intro rule. This is strictly speaking a bit more work than is logically required, but it makes the proof a bit easier to follow at a glance. Proofs that are easy to follow make your grader happy. Proofs that are hard to follow make your grader irritable. You want your grader to be happy and not irritable, right?

1) Bar(j) → Drink(j)	Premise
2) Drink(j)→ ¬Blood(j)	Premise
3) Blood(j)	Premise
4) Bar(j)	Assume
5) Drink (j)	→ elim 1, 4
6) ¬Blood(j)	→ elim 1, 4
7) Blood(j)	Reit 3
7) ⊥	⊥ Intro6, 7
¬Bar(j)	¬Intro 4-7

The final step of this proof is simply to note that the assumption Bar(j) at 4 leads to contradiction, so the negation of the the assumption, i.e., ¬Bar(j), must be true. We introduce a negation into the assumption, in other words.

Here's the full rule:

Negation Introduction (¬Intro or RAA)

:
  m) φ
  :
  :
  n) $\bot$
o) $\neg\varphi$ ¬Intro m-n

If you have a subproof starting with some proposition φ on line *m*, and ending with a contradiction symbol $\bot$ on line *n*, you may end the subproof, discharging the assumption φ and writing $\neg\varphi$ the on the subsequent line *o*.

These are all the rules needed to prove anything that can be represented in PL. But having the rules is one thing, and knowing when and how to implement them is another. The next section will describe a wholly general strategery for thinking about proofs. We'll also demonstrate this strategery by deriving several useful new rules that we'll go on to use.

Summary of Rules

In a valid proof of a conclusion ψ from premises $\{\varphi_1, \varphi_2, \ldots \varphi_n\}$, the conclusion ψ must be on the same scope line as $\{\varphi_1, \varphi_2, \ldots \varphi_n\}$; i.e. the "first" or "outermost" scope line.

Assumption: You may write down any sentence φ whatsoever on any line, as long as it is written down inside a new scope line, and everything that follows is remains within that scope line until the assumption is "discharged" or the subproof is ended. No information inside the scope line may be reiterated out of the subproof or otherwise brought out of the subproof except as specifically allowed by the rules. The justification is "Assume".

End Assumption: You may end an assumption at any point. The next line of the subproof is written down on the next scope line outside the "ended" assumption. No information from the ended subproof is available outside that subproof if the assumption is ended in this way.

Argument by Cases (AC)

m) $\varphi \vee \psi$
:
n) φ Assume
:
o) π
:
p) ψ Assume
:
q) π

r) π AC m, n-o, p-q

If:
A) Line *m* is some disjunction $\varphi \vee \psi$,
B) Lines *n-o* is a subsequent subproof directly in the scope of line *m,* beginning by assuming the first disjunct (φ) and ending in some subconclusion π,
C) Lines *p-q* is is a subsequent subproof *on the same scope level as the subproof in n-o*, beginning by assuming the subsequent disjunct (ψ) and ending in the same proposition π from subproof *n-o*,
Then you may discharge the assumptions φ and ψ, and write down π on the same scope line as the original disjunction. For justification, cite AC m, n-o, p-q.

Conditional Intro ($\rightarrow$ Intro) :

:
m) φ Assume
:
n) ψ
o) $\varphi \rightarrow \psi$ $\rightarrow$ Intro m-n

If lines *m* - *n* comprise a valid subproof beginning at *m* with the assumption φ and ending at *n* with the proposition ψ, you may end the subproof, discharging the assumption φ and writing the conditional $\varphi \rightarrow \psi$ on a following line *o*, one scope line outside the subproof *m* - *n.* Cite $\rightarrow$ Intro m-n as your justification.

Contradiction Introduction (⊥ Intro)

:

m) φ

n) ¬φ

o) ⊥ ⊥ Intro m, n

If lines *m* and *n* are contradictory propositions (i.e., of the form φ and ¬φ) and n is within the scope of m, ⊥ on any subsequent line *o* within the scope of both *m* and *n*. Cite as your justification the rule ⊥ Intro and the line numbers of the contradictory propositions, i.e., *m* and *n.*

Negation Introduction (¬Intro or RAA)

:

m) φ

:

:

n) ⊥

o) ¬φ ¬Intro m-n

If you have a subproof starting with some proposition φ on line *m*, and ending with a contradiction symbol ⊥ on line *n*, you may end the subproof, discharging the assumption φ and writing ¬φ the on the subsequent line *o*.

Glossary

Assumption -
Scope Line -
Subproof -

Chapter V

Proof Strategery and Derived Rules

Index

I) Introduction

II) The Tripartite Strategery

A) Phase 1: work backwards from the conclusion

Side note: the analytic method

B) Phase 2: Tear 'em down, build 'em up.

Phase III: The Hard Stuff (subproofs).

I) Introduction

So now we have a bunch of rules. Weee. It should seem pretty obvious that each of the rules makes intuitive sense, and there really shouldn't be much confusion about how to apply them. It should also seem pretty obvious that we haven't really proved anything interesting (in some sense of 'interesting') with these rules. The complicated part will be combining rules and techniques. And when we do this, we'll quickly see that the real trick is knowing how to start proofs, when to use any given rule or technique, and what order to do them in.

In this chapter we'll introduce the tripartite strategery for doing proofs. Think of the strategery as a meta-rule that tells you when to apply rules. The strategery is nearly wholly mechanical - you can follow it without really "thinking" or even knowing what you're doing. Obviously it's better if you understand what you're doing - because then you'll be able to broaden extend the strategery to reasoning in non-formal circumstances. The strategery has been specifically designed to have this fully general applicability. Just like the ROIs, the strategery is not any sort of magical new way of thinking, but rather a codification of reasoning techniques we already use, just made explicit.

In showing the strategery, we'll kill two birds with one stone. First, of course, we'll lay out and then demonstrate the strategery. To demonstrate the strategery, we'll prove several theorems - useful theorems that we'll wind up using in the future. These theorems are a set of *derived rules*. Since they're derived from rules we already have, they're never strictly necessary to doing any proofs. Any time we use a derived rule, we could have instead used the combination of basic rules we used to prove the derived rule. But once we've proved the derived rules, we'll be able to use them to speed up and simplify future proofs.

II) The Tripartite Strategery

A) Phase 1: work backwards from the conclusion

'Every inquiry is a seeking. And every seeking gets guided beforehand by what is sought'.[15] And with these words, this becomes the first (and only) formal logic textbook to quote Heidegger.

Ordinarily, in Philosophy, you don't know beforehand what you want to conclude. Actually, you always have some inkling - you have a gut feeling that a certain argument is unsound, or that a certain proposition *must* be true, or whatever. That doesn't mean you always wind up proving what you set out to prove - sometimes, along the way, you see that your initial instincts were wrong. But at least they provided you a starting point.

In the context of doing formal proofs, you have it easier. We're always going to give you a conclusion and tell you to prove that it follows from certain premises. Logic doesn't always work that way - formal logicians seeking to prove new theorems don't have any logic oracles to tell them what they should be looking for. But that's a task for a further day.

You should always begin by looking at the form of the conclusion and letting it guide you through the proof. The simplest form of this is to look for conditionals in the conclusion. If the central connective is a conditional, then you should straight away set up a conditional introduction subproof, where the first line of the subproof is the antecedent of the conditional, and the last line is the consequent of the conditional. After all, if you need to prove a conditional statement, conditional intro is about the only rule we have. Once you've set up the subproof, you know that the justification for the conclusion will be $\rightarrow$ intro, so write that down as the justification for the final line. Give as many line numbers as you can - obviously you won't be able to give the last line of the subproof yet, since you don't know how many steps the subproof will have. Call the last line of the subproof '*' or something,

15 Heidegger, Being and Time, Section 5 (p 24)

and cite it instead. That way, when the proof is over, all you need to do is look back at what line number * wound up with, and replace s* in the conclusion with that number. This is important - if you don't write this stuff down now, then when you finish the proof, you'll need to "re-think your way through it" in order to remember what rule you meant to use at this step. We can do all this because we know that $\rightarrow$ intro will be our final step in the proof. So we know that if we can get to the last line of the subproof, we'll be home free. So the last line of our subproof becomes our new target - let's call it the "subconclusion"

After we've done this, we need to prove the subconclusion. So we start all over with our strategery at phase I. Is the subconclusion itself a conditional? (it would be if, for example, the original conclusion had had been something like $[P \rightarrow (Q \rightarrow R)]$). If it is a conditional, then apply phase I all over again to it, creating a *new* subproof. This will be a subproof that's *nested* inside the previous subproof.

If the new subproof isn't a conditional, then move on th phase II of the strategery. Let's illustrate with an example phase I.

Prove: $(P \wedge Q) \rightarrow R$ $P \rightarrow (Q \rightarrow R)$

Phase I of the strategery tells us to look at our conclusion. Is it a conditional? Yes, the central connective is the first in the proposition. So phase I directs us to set up a subproof, where the first line is the antecedent of the conditional we wish to prove (i.e., *P*) and the last line is the consequent of said conditional. To wit:

1) $(P \wedge Q) \rightarrow R$	Premise
2) P	Assume
3) :	?
4) :	?
(etc)	
*) $(Q \rightarrow R)$	? (new subconclusion)
C) $P \rightarrow (Q \rightarrow R)$	$\rightarrow$ intro, 2 - *

See? The strategery has already given us the shell of a proof. it's also given us some new (hypothetical) information o work with: P, in line 2. And we already know how we're going to discharge that assumption. Remember, *never* begin a subproof or make an assumption without knowing what rule you're going to use to discharge that assumption! Now, how are we going to get to line *? Who knows? Let's use the strategery again!

Now we have new premises / assumptions, and a new subconclusion. So let's go all the way back and start with phase I all over again, taking this new information into account. Phase I tells us to look for conditionals in the (sub)conclusion. Lo and behold, * is in fact a conditional. So we should set up yet another subproof starting at line 3, as follows:

1) $(P \wedge Q) \to R$	Premise
2) P	Assume
3) Q	Assume
4) :	?
**) R	?
*) $(Q \to R)$	$\to$ intro, 3 - **
C) $P \to (Q \to R)$	$\to$ intro, 2 - *

We don't know what line number R will be on, so we'll call that line "**" for now. And we don't yet know how we'll prove **, but we know we have to, and we know that once we do, we've proven $(Q \to R)$, by using $\to$ intro on the subproof that begins at 3. This means that we are "in" two subproofs deep by now, which may seem a bit intimidating. But we already know how we're getting out of both subproofs, because of our strategery. So it really isn't as difficult as it first looks. All that *really* remains for us to do know is derive R from our stock of premises and assumptions. So, let's start over with phase I. Look at our new (sub)conclusion - is it a conditional? In this case, no. It's just an atomic sentence - not really much to work with. So we should move on to Phase 2.

Side Note: the analytic method

You should stop for a minute and notice what the strategery has done for us so far. It's allowed us to break down what appeared to be a complicated problem - i.e., prove $P \to (Q \to R)$ - into some much smaller problems. That is, first we want to prove $(Q \to R)$ from the premises plus a new assumption (*P*), and then we want to prove R from the premises plus assumptions P and Q. Little problems are better than big problems, and there are no big problems, just large collections of little problems. Think of this as the core of the analytic method. What PL (and the strategery) really allows us to do is break down the big problems into groups of small problems, and keep track of the order we solve them in so we can put them back together into the complicated solution (to the big problem). And this is explicitly designed into the first two phases of the strategery. Well, it's designed into the strategery as a whole, but some animals are more equal than others, right?

B) Phase 2: Tear 'em down, build 'em up.

In phase II of the proof, we want to break down as much of the premises as possible (usually using our connective elimination rules). Ideally, we break the information in the premises down to just atomic statements. Then, we use our connective introduction rules to build the atomic propositions up into the conclusion (or, if we've used phase I to break things down into smaller problems, we build our information up into the subconclusions).

The key to this step is really just blind pattern matching. Look at the information available to you. Do you have a conditional? Then look for the antecedent so you can use $\rightarrow$ elim. Do you have a conjunction? Then break it apart with $\wedge$ elim. Do you have a disjunction? Well, you're SOL for now, but we'll derive some rules for working with disjunctions in a bit. Do you have a Biconditional? Break it up into conditionals with $\leftrightarrow$ elim. Basically, look at the connectives you have, and try to use the appropriate connective elimination rules on them. Here's a partial list:

Phase II: Break down information
Conditionals: look for the antecedent, and use $\rightarrow$ elim
Conjunctions: use $\wedge$ elim
Biconditionals: $\leftrightarrow$ elim
Double negations: DNE

That list isn't very long right now, because we have only a very few rules. We'll use these rules to generate some other useful rules or phase II, and we'll revisit the expanded strategery at the end of the chapter.

Remember that this procedure can be applied recursively just like phase I. That is, as you break up information, you generate *new* information. Take this new information and re-apply phase II to it. For example, perhaps you have a proposition like

4) P ∧ (P → Q)

First, use ∧ elim to break apart the central connective. This yields:

4) P ∧ (P → Q)	
5) P	∧ elim 4
6) (P → Q)	∧ elim 4

Then, start back over with Phase II on lines 5 and 6. Notice that 5 matches the antecedent of 6, and use → elim to break 6 down into:

4) P ∧ (P → Q)	
5) P	∧ elim 4
6) (P → Q)	∧ elim 4
7) Q	→ elim 5, 6

Most of this phase is simple pattern recognition. Look for antecedents of conditionals, in particular. The further rules we'll add will extend the kinds of patterns you'll be looking for, but the general idea will remain the same. For every kind of connective, there are rules for breaking it down. So look at each connective, and think about the different possibilities. The hardest part of doing a proof is knowing how to proceed, or where to start. But if you reflect a bit on the information available to you, you'll see that each proposition really only gives you a small number of ways to proceed.

After you've broken down all the information available, you need to build it up into the conclusion (or subconclusion). Typically, this will involve the connective introduction rules. But there's some back-and-forth here - you may find yourself breaking down, building up, breaking down and so on. Sometimes it's obvious how to build up your available information into your goal proposition. But sometimes it's not at all apparent. But you'll always have a pretty powerful clue - namely, the goal proposition (the conclusion or subconclusion) As in phase I, you should let the structure of the conclusion guide your strategery.

Often, the best way to implement this is by working backwards from your goal. Are you trying to prove a conjunction? Then the odds are you need to prove each conjunct separately and build them into the conjunction using ∧ intro. So, create two lines prior to the goal conjunction, and write a conjunct on each. Then write ∧ into on the conjunction line:

11) (Q → S) ∧ (R ∨ P)	?

Need to prove a conjunction?

→

9 (Q → S)	?
10) (R ∨ P)	?
11) (Q → S) ∧ (R ∨ P)	∧ intro 9, 10

Write down the conjuncts beforehand!

Many people find this counterintuitive. How can you rationally assert something you haven't justified yet? But really, that's not what's going on here. All we are *really* asserting is that *in order to get to* 11, we'll first need to get to these individual conjuncts (9 and 10). And *once we get those* conjuncts separately, we'll be justified in conjoining them in 7. But we aren't yet asserting that we've gotten to 11, because we aren't claiming to have gotten the individual conjuncts.

Recall what we said about the analytic method earlier. When we work backwards from our conclusion, we break apart our problem into smaller subproblems. And specifically, we think a little bit about what the overall problem is in order to decide what subproblems we should tackle.

We can go on with this method in the little mini - proof we have above. Notice that 9 is a conditional. We should now go back to phase one of the strategery, and set up the appropriate → intro subproof. This highlights the recursive nature of the strategery: every time you add new information, whether you derive new information from the premises, or you break down the problem into smaller problems and hence add new subconclusions, go back to phase I of the strategery and start over.

5) :	
6) Q	Assume
7) :	
8) S	?
9) $(Q \rightarrow S)$	→ intro 6-8
10) $(R \vee P)$	?
11) $(Q \rightarrow S) \wedge (R \vee P)$	∧ intro 10, 11

Set up the subproof to prove line 9 with → intro

Next look at the subconclusion in 10. How do you prove a disjunction? Usually that's easy. All you need is one half of the disjunction - in this case, either R or P - and you get the other half "free" by using ∨ intro. So look around and see if you have either of those available to you in the premises. For example, if you don't see P anywhere in your premises, it's a pretty good bet you'll need to get get R and tack P on with ∨ intro.

There's a bit of an art to this, but it will become easier as you work with proofs. For now, let's summarize the "buildup" part of phase II:

Phase II: Build up information

Disjunctions: Look for one disjunct and use $\vee$ intro.

Conjunctions: Look for both conjuncts and use $\wedge$ intro.

Biconditionals: Look for "backwards" and "forwards versions of the same conditional, and use $\leftrightarrow$ intro.

Double negations: DNI.

Let's return now to our original proof, that we started the chapter with:

Prove:

$(P \wedge Q) \rightarrow R$

$P \rightarrow (Q \rightarrow R)$

You'll recall we'd gotten quite some way into this proof. We stopped at:

1) $(P \wedge Q) \rightarrow R$	Premise
2) P	Assume
3) Q	Assume
4) :	?
**) R	?
*) $(Q \rightarrow R)$	$\rightarrow$ intro, 3 - **
C) $P \rightarrow (Q \rightarrow R)$	$\rightarrow$ intro, 2 - *

All we really need to do now is focus on the subproof from 3 - **. (Recall, we aren't sure how long the subproofs will take, so we're using * and ** for line numbers right now). What information is available to us in this subproof? We' have every proposition from lines 1 through 3 available. What's our goal proposition? The subconclusion at line **, namely, R.

Now you might be able to look at the proof so far and just have your pattern-matching sensors go off. I know some logicians that simply glance at a proof, and see past the marks on paper to the pure heaven of the forms underneath, and the proof does itself for them as the angels sing. but for the rest of us, sometimes we need a more brainless strategy. So let's look at our conclusion, R. Where do we see R in the premises? Pretty much only in line 1. This suggests that the only real "road" to R is via line 1. Line 1 is a conditional, so the only way we have of getting at the consequent is via $\rightarrow$ elim. If we're going to use $\rightarrow$ elim, we also need to get the antecedent $(P \wedge Q)$. And we don't have that yet. But if we *did* have it, we could just do $\rightarrow$ elim and get our subconclusion. So lets work backwards that way:

1) $(P \wedge Q) \rightarrow R$	Premise
2) P	Assume
3) Q	Assume
4) :	?
***) $(P \wedge Q)$	?
**) R	$\rightarrow$ elim, 1, ***
*) $(Q \rightarrow R)$	$\rightarrow$ intro, 3 - **
C) $P \rightarrow (Q \rightarrow R)$	$\rightarrow$ intro, 2 - *

So now we look at our new subconclusion, $(P \wedge Q)$ at line ***. This is a conjunction. We generally build up conjunctions by finding each of the conjuncts in our pool of available information and applying ∧ intro. Ordinarily, I'd write down P and Q separately on the lines immediately preceding line ***. Then I'd think about how I was going to derive those. But in this case, I quickly look at my available information, and see that I already have P and Q, at lines 2 and 3 respectively. Yay! Completing a proof is often like this: you work alternatively from the end and from the beginning, and suddenly, to your surprise, both ends meet! We'll just go back and fill in all the * lines with the correct line numbers at this point, to finish it off:

1) $(P \wedge Q) \rightarrow R$	Premise
2) P	Assume
3) Q	Assume
4)) $(P \wedge Q)$	∧ intro 2, 3
5) R	→ elim, 1, 4
6) $(Q \rightarrow R)$	→ intro, 3 - 5
C) $P \rightarrow (Q \rightarrow R)$	→ intro, 2 - 6

Phase III: The Hard Stuff (subproofs).

I am a lazy bastard. if I can get away without doing hard work, I will, and then crack open an Arrogant Bastard ale while everyone else is still sweating. This holds doubly for proofs. If I can get away with doing a proof without mucking around with subproofs or assumptions or discharging stuff, I will. Let's face it, except for the subproofs, everything o far is pretty straightforward. So notice that phases I and II use subproofs minimally. I save the hard stuff for the very last option. And with any luck, I won't even have to use them!

Sometimes, though . . .you work all the way through phases I and II, and you're still stuck. Maybe you've got a disjunction in your premises you just can't break up. Or maybe it's something else you've broken stuff down as far as you can, and set up all your subproofs, and you still can't make each "end" of the proof meet. At this point, we break out the last two weapons in our arsenal: ∨ intro and AC.

∨ intro is used when you just can't do anything else, and you've got a stubborn disjunction that you can't break up. At this point, about the only option you have left is to break up the disjunction with intro, by showing that ultimately both disjuncts lead to the same thing. Well, that's all fine and dandy, but what proposition do you want to derive from the disjunction? After all, there's probably a multitude of propositions that can be derived. Many of them are probably pretty uninteresting. And many more are probably useless to your goal.

The short answer here is: once more, let your subconclusion guide you. To break up a disjunction, use ∨ intro to start two subproofs, each one beginning by assuming one of the disjuncts. And remember that in order to *discharge* those assumptions, you need each subproof to end with the same proposition. That is, to demonstrate that all roads lead to Rome, you have to take each road separately and show that each road takes you to Rome. So what is your Rome? What proposition do you want each subproof to end with? The natural place to start is . . .your goal proposition. Prove that all roads lead to Bakersfield, CA? (ugh!) make a subproof for each road, each road ending (unfortunately) Bakersfield. To illustrate:

5) :

6) $S \vee P$

7) :

8) :

9) $(Q \rightarrow S) \wedge (R \vee P)$

10) :

11):

In this mini-proof-sketch, we imagine that we've gotten to line 6 by using some combination of phase I and phase II. We also imagine that somehow we've worked our way backwards from the conclusion to a subgoal in line 9, also using some combination of phase I and II. We can't get rid of the disjunction in line 6. So a natural move here would be to go on to phase III, and try to set up an AC in the following way :

5) :	
6) S ∨ P	
7) S	Assume
8) :	?
9) :	?
10) (Q → S) ∧ (R ∨ P)	?
11) P	Assume
12) :	?
13) :	?
14) (Q → S) ∧ (R ∨ P)	?
15) (Q → S) ∧ (R ∨ P)	
10) :	
11):	

Notice that what was line 9 is now line 15. This happens a lot, as you realize your proof might take longer than expected. That's why I generally don't fill in all the line numbers until I'm done. All we've done here is set up a subproof for each of the disjuncts in line 5, each subproof terminating in the same proposition - namely, our goal proposition (15). Then we cite the rule for AC to discharge our assumptions and get back out of the subproofs.

Okay, so suppose you've been through Phases I and II, and broken up all your disjunctions with AC, but you're still stuck. It's time to break out the big guns. This is the method of Reductio Ad Absurdum, or ¬ intro. This is the most powerful tool available. Anything that can be done in PL, can be done with

¬ intro. So the last ditch is: When you find yourself completely stumped, make a new subproof by assuming the negation of what you want to prove, and deriving some contradiction.

To demonstrate this, we'll prove some very useful derived rules. First, we will prove the rule of Modus Tolens. In natural language, it goes like this: If you are run over by a steamroller, you die. (Just accept this premise for now. If you'd like to argue the point, a physical demonstration could probably be arranged). Suppose we run across Albert, dead in the road. Can we conclude he was run over by a steamroller? *NO!* Alas, life as a squirrel is fraught with peril, and there are many possible ways Albert could have died. All we know is that steamrollers are *sufficient*, not *necessary*, for death. But suppose instead we see Albert, hopping about happily, gathering nuts and manifestly *not* dead. What can we conclude from this? Well, whatever else may be, at least we know he wasn't hit by a steamroller. Because if he was, he'd be a squirrelly pancake (i.e., dead). This inference pattern is known as Modus Tolens, the mode of taking. I have even less of an idea why it's called this than I had with Modus Ponens. Oh well. It looks like this:

m) $\varphi \rightarrow \psi$	
n) $\neg\psi$	
o) $\neg\varphi$	MT m, n

As you can see, this rule itself is pretty straightforwards. No assumptions, no subproofs, nothing. It's just sort of a backwards → elim with some negations. And it makes total intuitive sense. What's more, it already follows from the rules we have - that is, we can derive it from those rules. The rest of the derived rules are like this - not only do they make intuitive sense, but they're already 'contained within' the other, more basic rules. In this way, PL isn't simply a bunch of handy rules, but it's a unified system whose parts all cohere with each other and help explain each other. Once we've proven MT, you'll be allowed to use it in your further proofs (unless specifically prohibited).

Let's use our steamroller example to prove it. It would be fully general if we only used metalanguage variables like the Greek letters, but fie on that. You can do it that way yourself if you want to. We'll start with the premises $S \rightarrow D$ and $\neg D$, and derive $\neg S$:

1) $S \rightarrow D$	
2) $\neg D$	
C) $\neg S$	?

Now we look at our options. Phase I says to look for conditionals in the conclusion. No love there, however. So we move on to phase II: Break things down. Well, line 2 is as broken down as it could be. Line 1 is a conditional, so we look around for the antecedent, namely, S. Do we find S anywhere? (not as a component of some sentence, but all on its own, as an atomic sentence). Nope. If only we had MT, we could use it to break down line 1 - but we're trying to *prove* MT, so it would be pretty illegitimate to use it in its own proof! So we've pretty much exhausted all the options in Phase II. Moving on to Phase III, we note that we don't have any disjunctions. So AC is out. That leaves only one possibility - Negation Introduction. Ugh. So, lets set it up.

1) S → D	
2) ¬D	
3) S	Assume
4) :	
*) ⊥	⊥ intro ?, ?
C) ¬S	¬Intro, 3 - *

Just as we described in the previous chapter, we start by taking the opposite of the thing we want to prove. That is, we assume the goal sentence *minus one negation* ¬ . The goal sentence is ¬S, so we assume S. If We can show that S leads to contradiction, we will show it must be false, so we can discharge the assumption and add a negation to the assumption *outside the subproof*.

What negation are we going to derive? At this point, we don't know. Any contradiction will do. maybe it's a very complex one, or maybe it's just an atomic sentence and its negation. We'll just continue on with the strategery and see what happens.

Now we've added some new information available to us. We now have the proposition S to work with, at least within the subproof from 3 - *. And all the information in the premises is also available to us. To break down 1, we'd need to use → elim, which would require that we had the antecedent S. We do have it now! So that gets us ¬D on line 4:

1) $S \to D$	
2) $\neg D$	
3) S	Assume
4) D	$\to$ intro 1, 3
*) $\bot$	$\bot$ intro ?, ?
C) $\neg S$	$\neg$Intro, 3 - *

Now we just need to find a contradiction. We've broken everything down about as far as it will go, so it's time to look around for a pair of sentence like φ and ¬φ. Luckily these sentences practically leap off the age at us: D and ¬D, at lines 2 and 4. Whoa! The proof is done! Practically by itself! We'll just clean it up by filling in the appropriate line numbers:

1) $S \to D$	
2) $\neg D$	
3) S	Assume
4) D	$\to$ intro 1, 3
5) $\bot$	$\bot$ intro 2, 4
6) $\neg S$	$\neg$Intro, 3 - 5

And while we're at it, we'll give a fully general presentation of the rule Modus Tolens (MT):

Modus Tolens (MT):

m) $\varphi \to \psi$

n) $\neg\psi$

o) $\neg\varphi$ MT m, n

If you have a sentence of the form $\boldsymbol{\varphi \to \psi}$ on line *m*, and a sentence of the form $\boldsymbol{\neg\psi}$ on line *n*(the order of m and n are unimportant), you may derive a sentence of the form $\boldsymbol{\neg\varphi}$ on any subsequent line *o*. Cite as your justification MT m, n.

Now let's try another derived rule, DeMorgan's Rule. This is an equivalence that we demonstrated way back in chapter II, on Truth Tables. Suppose you have a conjunction of negations, say, $(\neg A \land \neg B)$.

Since that essentially says 'Not A, and Not B', ' it really is equivalent to 'Neither A nor B', or $\neg(A \lor B)$

Makes sense, right? Let us prove it follows from our other rules.

1) ¬ A ∧ ¬ B	Premise
C) ¬ (A ∨ B)	

DeMorgan's Law (version 1)

The first thing to notice is that there is not conditional in the conclusion. So we can pass right through phase I. Next, we notice that the premise is a conjunction, which is easily broken apart with the ∧ elim rule.

1) ¬ A ∧ ¬ B	Premise
2) ¬ A	∧ elim 1
3) ¬ B	∧ elim 1
:	
C) ¬ (A ∨ B)	?

At this point, we've broken our premises down as far as they will go, and our conclusion isn't providing much help. So we've exhausted phases I and II of the strategery. It's on to the hard part, alas. But our options are already limited: there are no disjunctions in our available information at line 3, so the only thing we *can* do is set up a ¬intro on the conclusion! (Incidentally, if the conclusion is the negation of some complex proposition, that's often a good sign that you'll want to use ¬intro. Especially if the conclusion is a conjunction: because then your assumption for ¬intro will essentially help you to a fairly strong claim, namely, a conjunction. And the stronger the claim, the more likely you are to be able to contradict it!)

1) ¬ A ∧ ¬ B	Premise
2) ¬ A	∧ elim 1
3) ¬ B	∧ elim 1
4) (A ∨ B)	Assume
:	
*)⊥	⊥ intro ?
C) ¬ (A ∨ B)	¬intro 4-*

Setting up ¬intro

We don't know what line our contradiction will be, and we don't know what propositions will contradict each other. So we call the contradiction line * for now, and we leave a '?' in its justification.

Now, we've added some new information to our stock of available information, i.e., the disjunction at line 4. We need to break this down. We still don't know exactly what our subconclusion is going to be at line *, so we move past phase I. The only new information is a disjunction, and phase II doesn't have any way to break down disjunctions, so it's straight on to phase III. It's clearly time to try an AC to break down line 4. So we'll set up two subproofs, one starting with A, the other starting with B, and each ending in the same proposition. What proposition should that be? How about the subconclusion we're currently after - namely, ⊥.

1) $\neg A \wedge \neg B$	Premise
2) $\neg A$	$\wedge$ elim 1
3) $\neg B$	$\wedge$ elim 1
4) $(A \vee B)$	Assume
5) A	Assume
:	
**)	$\bot$ intro ?
***) B	Assume
:	
****)	$\bot$ intro ?
*)$\bot$	$\bot$ intro ?
C) $\neg (A \vee B)$	$\neg$intro 4-*

We're now three subproofs deep, which is often a cause for alarm. How are we going to discharge all those assumptions? But if you've used the strategery correctly (as we have), every time you make an assumption, you include the "exit plan". You don't make any assumptions unless you know what technique you will use to discharge them. So if we can just fill in the missing lines (indicated with ":") with valid steps, we'll be guaranteed to be able to discharge the assumptions. In fact, by citing the rule for discharging the assumptions after the close of each subproof, we're already halfway there! All we need to do is find some contradictions. We've got a lot of atomic statements, so at lines 5 and ***, we should start trying to find contradictions in our available information.

The 5 - ** subproof is pretty easy: The assumption A at line 5contradicts the information derived from the premise in line 2, namely, ¬ A. So we don't even need that first ":" line. Then, in line ***, we look around. Notice that none of the information in subproof 5 - ** is "live" right now, in the sense that none of it is available within subproof *** - ****. That's because these two subproofs are independent of each either - neither is in the scope of the other, and specifically, *** - **** is not within the scope of 5 - **. Again, this is easier to see using the scope lines than it is using asterisks and text. But this isn't a problem - we can find a contradiction in the same way we found it in the previous subproof. The proposition B at line *** contradicts the proposition ¬ B at line 3.

Just like the earlier proof, we're suddenly done! The information in the premises has "met up with" the subconclusions we've established. All that remains is to fill in the line numbers:

1) $\neg A \wedge \neg B$	Premise
2) $\neg A$	$\wedge$ elim 1
3) $\neg B$	$\wedge$ elim 1
4) $(A \vee B)$	Assume
5) A	Assume
6) $\perp$	$\perp$ intro 2, 5
7) B	Assume
8) $\perp$	$\perp$ intro 3,7
9) $\perp$	AC 4, 5-6, 7-8
C) $\neg (A \vee B)$	$\neg$ intro 4 - 9

As a final note, notice that we had to change the justification in line 9. That contradiction was introduced inside the AC subproofs, and then we discharged the assumptions using AC to get it on the appropriate scope line in line 9. This kind of pattern emerges quite frequently in PL.

This completes our overview of general proof strategery. All that remains is to extend the strategery slightly, by adding in some useful derived rules, and showing how to apply the strategery to proofs from no premises. The exercises in this chapter will require you to prove several derived rules. The next chapter, Chapter VI, will simply be a handy list of those derived rules, and the final chapter about PL will explore some special kinds of rules and proof techniques.

Summary of the Tripartite Strategery

Phase 1: work backwards from the conclusion

Look for conditionals (→) as the central connective of the conclusion. If there are any, set up an → intro subproof, with the antecedent as the first line of the subproof, and the consequent as the last line. Repeat Phase I on the new subconclusion. If there is no conditional conclusion (or subconclusion, go onto phase II..

Phase 2: Tear 'em down, build 'em up.

Break down the premises and the information derived from the premises:

Conditionals:

Look for the antecedent, and use → elim.

Look for the negation of the consequent, and use MT

Conjunctions:

Use ∧ elim

Biconditionals:

Use ↔ elim

Double negations:

Use 2NE

As you derive new information, start this part all over again, breaking down the new information. When you've broken the premises as far down as you can, start to build up the information derived from the premises, into the subconclusions. Try working backwards to think about how to build up to your subconclusions.

Disjunctions: Look for one disjunct and use ∨ intro.

Conjunctions: Look for both conjuncts and use ∧ intro.

Biconditionals: Look for "backwards" and "forwards versions of the same conditional, and use ↔ intro.

Double negations: DNI.

As you generate new subconclusions, return to phase I. When you've exhausted phase I and II, and you're still not done, move on to phase III.

Phase III: The Hard Stuff (subproofs).

If you've got a disjunction in your premises, and you can't break it up, set up an Argument by Cases, with two subproofs. Each subproof will begin with one of the disjuncts, and end in the subconclusion that is your immediate goal.

If you're still totally stumped, set up a negation introduction subproof to prove your subconclusion. The subproof will begin by assuming the opposite of the subconclusion you're trying to prove, and it will end with some contradiction. Go back to Phase I at this point.

Glossary

Within the scope of

Under the scope of

Has scope over

Scope

Chapter VI
Derived Rules

Chapter Index

A) Introduction

In this exciting chapter, we cover more rules for PL. In a sense, this chapter, while dramatic and thrilling, is superfluous. That's because every rule in this chapter is unnecessary. They can all be recreated 'on the fly' from the previous 'basic' rules. Still, having them around can sometimes speed up proofs quite a bit. It's the nature of human progress - we can refer back to knowledge previously gained without having to re-invent the wheel every time we build a new car or prove a new theorem.

In the final part of the chapter (and volume), we'll look at a new technique for proofs - proofs involving no premises. Such proofs are required in proving logical truths. That's because logical truths, by definition, don't depend on the truth of any other propositions whatsoever; they are true solely in virtue of the principles of PL. Although these proofs seem intimidating, in fact they are often even simpler than the ordinary proofs. That's because you need *some* propositions to work with, and there's nly a very few ways of introducing new information (usually via assumptions) into a proof. So the first, and hardest, part of the derivation is already settled - you almost always will know how to start the proof.

Without further ado, let's look at the rules.

B) Distribution Rules

The DeMorgan Equivalences (DeM)

Recall from our discussion of truth tables and semantics that you cannot simply distribute a negation through a disjunction, conjunction or anything else. For example,

$\neg(P \vee Q)$ does not entail

$(\neg P \vee \neg Q)$.

For some reason, people think this means that PL is not 'mathematical' or systematic. But this is silly! After all, not every mathematical operation distributes straightforwardly! Consider simple addition. It doesn't distribute over multiplication! For example,

+1 (2 * 5) surely doesn't equal or even entail

+1(2) * +1(5)

You can surely find a host of other, similar examples. Still, the negation does distribute in a more complicated way. Think of the intuitive reading of 'not' and 'and'. If

$P \vee Q$

is false, then neither of them can be true. And conversely, if

P is false ($\neg P$ is true) and Q is false, i.e.,$(\neg P \wedge \neg Q)$

Then it can't be the case that either one or the other is true, i.e.,

$\neg(P \vee Q)$

So our first pair of new ROIs is:

DeMorgan's Law (DeM)

m) $\neg(\varphi \vee \psi)$

n) $\neg\varphi \wedge \neg\psi$ DeM m

If you have a sentence of the form $\boldsymbol{\neg(\varphi \vee \psi)}$ on line ***m***, you may write a sentence of the form $\boldsymbol{\neg\varphi \wedge \neg\psi}$ on any subsequent line ***n*** within the scope of ***m***.

DeMorgan's Law (DeM)

m) $\neg\varphi \wedge \neg\psi$

n) $\neg(\varphi \vee \psi)$ DeM m

If you have a sentence of the form $\boldsymbol{\neg\varphi \wedge \neg\psi}$ on line ***m***, you may write a sentence of the form $\boldsymbol{\neg(\varphi \vee \psi)}$ on any subsequent line ***n*** within the scope of ***m***.

In practice, you almost always distribute the negation through, but the reverse is valid as well. The two sentences not only entail each other, they are logically equivalent. In crude terms, DeMorgans allows you to drive the negation inside the parentheses, and flip the connective over.

The same holds for distribution over conjunction:

DeMorgan's Law (DeM)

m) $\neg(\varphi \wedge \psi)$

n) $\neg\varphi \vee \neg\psi$ DeM m

If you have a sentence of the form **$\neg(\varphi \wedge \psi)$** on line ***m*****,** you may write a sentence of the form **$\neg\varphi \vee \neg\psi$** on any subsequent line ***n*** within the scope of ***m***.

DeMorgan's Law (DeM)

m) $\neg\varphi \vee \neg\psi$

n) $\neg(\varphi \wedge \psi)$ DeM m

If you have a sentence of the form **$\neg\varphi \vee \neg\psi$** on line ***m***, you may write a sentence of the form **$\neg(\varphi \wedge \psi)$** on any subsequent line ***n*** within the scope of ***m***.

It's a good idea to go ahead with using DeMorgan's any time you see a negated disjunction or conjunction in your premises. Just like in algebra, the sentences are far easier to work with after you've distributed the negation.

Negated Conditional Rule (LL)

Similarly, you can't just distribute a negation over a conditional. It's not the case that masturbation leads to blindness - alas, that certainly doesn't mean that abstinence prevents blindness (leads to non-blindness).

Still, if $\neg(M \rightarrow B)$ is true, then $(M \rightarrow B)$ is false. Now think of the truth conditions for a conditional like $(M \rightarrow B)$. The only time it's false is when M is true and B is false. So if $M \rightarrow B$ is false, then M is true and nB is true. In other words,

Negated Conditional (LL)

m) $\neg(\varphi \rightarrow \psi)$

n) $\varphi \wedge \neg \psi$ LL m

If you have a sentence of the form $\boldsymbol{\neg(\varphi \rightarrow \psi)}$ on line ***m***, you may write a sentence of the form $\boldsymbol{\varphi \wedge \neg \psi}$ on any subsequent line ***n*** within the scope of ***m***.

I don't really know of a proper name for this one. It doesn't get used very often in PL, because it really only saves two steps (it combines CEQ and DeM). You can call it NC (Negated Conditional) if you want. I call it **LL** is for Lena's Lemma, after a student named Lena Rabinowich who proved it in some homework, and then kept referring back to it. Maybe if you're clever, you too can have a rule named after you like this!

C) Conditional Rules

Hypothetical Syllogism

This one's quite easy. I call it the daisy-chain rule. If P leads to Q, and Q leads to R, then you can cut out the middleman and just state that P leads (ultimately) to R.

Negated Conditional (LL)	
m) $\varphi \rightarrow \psi$	
n) $\psi \rightarrow \pi$	
o) $\varphi \rightarrow \varphi$	HS m,n

If you have a sentence of the form $\boldsymbol{\varphi \rightarrow \psi}$ on line m and a sentence of the form $\boldsymbol{\psi \rightarrow \pi}$ on some subsequent line ***n*** within the scope of ***m***, then you may write a sentence of the form $\boldsymbol{\varphi \rightarrow \varphi}$ on any subsequent line ***o*** within the scope of ***m*** and ***n***.

Modus Tolens

Suppose being hit with a steamroller - I mean really hit, not some weaselly glancing blow, but full-on hit - led to death. Suppose we then come across a dead squirrel (!) in the road. Can we tell that he was hit by a steamroller? No! Life as a squirrel is perilous and he could have died many ways. (the pancake shape might be a giveaway, though). So ($S\rightarrow$ D) and D together do not entail S - that's a common fallacy. It's so common that it's got a name - 'affirming the consequent', or some such name that pedantic sorts love to compile. But there's a related inference that's perfectly legitimate. Suppose we come across a squirrel frolicking among the trees, and very manifestly not dead. What then, can we infer? Well, whatever else may have happened to him, he must not have been hit by a steamroller! So ($S \rightarrow D$) and $\neg$D together *do* entail $\neg$S. This is called Modus Tolens, the *mode of taking*. I have no idea why. It's pretty useful though.

Modus Tolens (MT)

m) $\varphi \rightarrow \psi$

n) $\neg\psi$

o) $\neg\varphi$ MT m,n

If you have a sentence of the form $\boldsymbol{\varphi \rightarrow \psi}$ on line m and a sentence of the form $\boldsymbol{\neg\psi}$ on some subsequent line ***n*** within the scope of ***m***, then you may write a sentence of the form $\boldsymbol{\neg\varphi}$ on **a**ny subsequent line ***o*** within the scope of ***m*** and ***n***.

Notice that the way the rule is stated, you have to have the negation of the consequent in order to do MT. So the following is (technically) INVALID:

7)	$P \rightarrow \neg Q$	
8)	Q	
9)	$\neg P$	MT 7,8 INVALID!

But look. All you'd really need to do would be to perform a 2NI on line 8, to get $\neg\neg Q$, and that would be the negation of the consequent in line 7. Then you could do MT. Instead of forcing you do do this, let's just adopt a convention:

Modus Tolens (MT) (Simplifying convention)

m) $\varphi \rightarrow \psi$

n) (opposite of ψ)

o) $\neg\varphi$ MT m,n

If you have a sentence of the form $\boldsymbol{\varphi \rightarrow \psi}$ on line m and a sentence **that is the opposite of** $\boldsymbol{\psi}$ on some subsequent line ***n*** within the scope of ***m***, then you may write a sentence of the form $\boldsymbol{\neg\varphi}$ on **a**ny subsequent line ***o*** within the scope of ***m*** and ***n***.

That is, if you have $\varphi \rightarrow \psi$, then you need a sentence of the form $\neg\psi$, and if you have $\varphi \rightarrow \neg\psi$, you need a sentence of the form ψ. All this really does is streamline proofs a bit.

Contraposition

Another way of stating this logical relationship is this: If P leads to Q, then the negation of Q leads to the negation of P (and vice versa). If being hit by a steamroller impleas death, then life (*not-death*, not *undeath*!) implies non-steamroller-being-ht-by. Er, being alive implies that you weren't hit by a steamroller.

Contraposition

m) $\varphi \rightarrow \psi$

n) $\neg\psi \rightarrow \neg\varphi$ CEQ m

If you have a sentence of the form $\boldsymbol{\varphi \rightarrow \psi}$ on line ***m***, you may write a sentence of the form $\boldsymbol{\neg\psi \rightarrow \neg\varphi}$ on any subsequent line ***n*** within the scope of ***m***.

Contraposition

m) $\neg\psi \rightarrow \neg\varphi$

n) $\varphi \rightarrow \psi$ CEQ m

If you have a sentence of the form $\neg\psi \rightarrow \neg\varphi$ on line ***m***, you may write a sentence of the form $\varphi \rightarrow \psi$ on any subsequent line ***n*** within the scope of ***m***.

Once again, let's adopt the same simplifying convention we used in MT: instead of being strict about the symbol string $\neg\varphi$ and $\neg\psi$ here, let's just use 'the opposite of φ', and so on. So the (conventionally simplified) rule of contraposition basically says that you can take a conditional, and switch the antecedent and consequent around as long as you turn them into their opposites.

Conditional equivalence

Suppose again that being hit by a steamroller leads to death (not much of a supposition, really, is it?). Because of the somewhat odd way we defined the conditional $\rightarrow$, this means that either something is not hit by a steamroller, or it's dead. And the converse holds: Suppose you either haven't been hit by a steamroller, or you're dead. So one of $\neg S$ or D is true ($\neg S \vee D$). But then, if the first disjunct is false, and the whole disjunct is true, the second disjunct must be true: $\neg\neg S \rightarrow D$. But $\neg\neg S$ is just S, so $S \rightarrow D$. In other words, a conditional is really just equivalent to a fancy disjunction:

Conditional Equivalence (CEQ)

m) $\varphi \rightarrow \psi$

n) $\neg\varphi \vee \psi$ CEQ m

If you have a sentence of the form $\boldsymbol{\varphi \rightarrow \psi}$ on line ***m***, you may write a sentence of the form $\boldsymbol{\neg\varphi \vee \psi}$ on any subsequent line ***n*** within the scope of ***m***.

Conditional Equivalence (CEQ)

m) $\neg\varphi \vee \psi$

n) $\varphi \rightarrow \psi$ CEQ m

If you have a sentence of the form $\boldsymbol{\varphi \rightarrow \psi}$ on line ***m***, you may write a sentence of the form $\boldsymbol{\neg\varphi \vee \psi}$ on any subsequent line ***n*** within the scope of ***m***.

D) Miscellaneous Rules

Disjunctive Syllogism

Just because they're miscellaneous doesn't make them less useful! This one is very useful indeed. And it's pretty obvious to boot. Suppose the butler did it, or the maid did it. Then suppose you definitively clear the butler. That leaves only the maid.

You might think, 'Wait, couldn't it be some third party'? But the original supposition was that it was either the butler or the maid! If that's true, it *must* be one of them. Now, perhaps in practice you are convinced of the maid's innocence. In that case, the *only* recourse you have is to re-evaluate the premise that either the butler or maid did it. That's always an option in philosophy as well as criminology. But if you're asked to assume a premise, you don't get to re-evaluate.

Disjunctive Syllogism (DS)

m)	$\varphi \vee \psi$	
n)	$\neg\varphi$	
o)	ψ	DS m,n

If you have a sentence of the form **$\varphi \vee \psi$** on line m and a sentence of the form **$\neg\varphi$** on some subsequent line ***n*** within the scope of ***m***, then you may write a sentence of the form **ψ** on **a**ny subsequent line ***o*** within the scope of ***m*** and ***n***.

Again, let's use the simplifying convention that we want the *opposite* of one of the disjuncts, not necessarily the negation of the disjunct. So we'll allow the following, even though it's not technically right:

7)	P ∨ ¬Q	
8)	Q	
9)	P	DS 7, 8

Contradiction Elimination (Elim, or XFQ)

In the semantics, you may have noticed that anything follows from a contradiction ("ex falso quodlibet"). Let's show that this is mirrored in the derivation system

1)	P ∧ ¬P	Premise
2)	P	∧ elim 1
3)	¬P	∧ elim 1
4)	P ∨ Hamster(m)	∨ intro 2
7)	Hamster(m)	DS 3,4

Simple enough! And obviously, the conclusion, that your mother is a hamster, could have been absolutely anything. And it would have clearly followed from any contradiction P ∧ ¬P. So the simplified rule is:

Contradiction Elimination (⊥ elim)		
m)	⊥	
n)	ψ	⊥ elim m

If you have a sentence of the form ⊥ on line ***m***, then you may write any sentence whatsoever on a subsequent line ***n*** within the scope of ***m***.

Excluded Middle (XM)

We mentioned this way back in the start of chapter 3, I think, but we've never gotten around to it. The law a of the excluded middle - that a proposition is either true or false - is a constitutive principle of PL. It's what separates PL from some of the so-called 'deviant logics', where there are extra truth values or even truth-value gaps. The study of these systems is interesting in itself, but in PL, we accept the law of the excluded middle. It's not really a derived rule, but it's worth stating it here anyway. Since it's true, you can write it down on any line, anywhere in a proof. (you might use it, for example, to start off an argument by cases proof).

Excluded Middle (XM)

m) $\varphi \vee \neg\varphi$ ΞM

You may write a sentence of the form $\boldsymbol{\varphi \vee \neg\varphi}$ on any line ***m.***

Hopefully this section has provided enough rules to make your derivations run more smoothly. Remember that there's always a trade-off between memory and cleverness: if you're clever, you don't need to remember derived rules, because you can always re-prove them in the middle of a proof. And if you've got a good memory, and can remember all these rules, you can speed up proofs and avoid having to be clever and re-prove these subsidiary rules. It's up to you how how to proceed.

E) Proofs with no premises

There remains one last proof technique. Actually, there's nothing really new here, but people sometimes find it a shock to do proofs with no premises, and become unsure about how to proceed. How could anything be proven without any starting material? All of our rules are rules for manipulating some sentences and turning it into another sentence, so if there's no sentence to work on, what can you do?

First, consider what kinds of sentences could possibly be proved from no premises. Only a logically true sentence could possibly be proven in this way - for only a logically true sentence depends on no other truths. Logically true sentences depend only on the laws of logic. So we'd better be able to demonstrate that they follow from no premises!

Second, think about this. If you're given no premises at all, you need *something* to work with. So you really only have three starting points. First, you can simply assert the law of the excluded middle (XM). This allows you to have some disjunction as your starting point. Which disjunction should you choose? Try to make it look as much like your conclusion as possible. If you do this, the next natural step is to either set up an argument by cases, or to turn that disjunction into a conditional.

The other possibilities involve making an assumption just so you have something to work with. But don't assume any old thing - only make an assumption if you have some idea how you'll discharge that assumption. So really, there's only two options. You can either set up a subproof for a conditional intro proof, or you can assume the opposite of the conclusion, and try to derive a contradiction (in other words, set up an RAA using the negated conclusion). That pretty much exhausts your possibilities. And it's even simpler: if your conclusion is a conditional, it makes good sense to try to proceed by setting up conditional intro (asume the antecedent, derive the consequent). If it's not a conditional, then you're simply left with RAA as your only option.

Let's try a sample proof. $(P \rightarrow (Q \rightarrow P))$ is a theorem of PL. It's actually a kind of interesting one. it asserts that if P is true, then no new information (Q) will change that. If P is true, then if Q is true, P is still true. Contrast this with statistical reasoning. Suppose we used the property of being *highly probable* instead of true. And suppose we used the arrow to represent some sort of relationship of statistical support rather than the material conditional: so $P \rightarrow Q$ meant P makes it highly likely that Q. Call this system SL, for Statistical Logic (obviously we'd need far more to make an interesting system). In SL, $(P \rightarrow (Q \rightarrow P))$ is not a theorem! That's because if P is highly probable, we can always add information that makes it less probable. Suppose P is *Albert is a pirate*. Q might be some information that makes this much less likely - such as, *Albert has never set foot off land*.

Well, that's just a little side excursion into a theorem that captures one of the key properties of deductive rather than inductive reasoning - it's monotonic, you cannot add new information that changes the truth of the old information. How to prove it?

First, we note that it's a conditional. So the old strategery applies: phase one tells us to set up a conditional intro proof.

1)	| P	Assume
	| :	
	| :	
o)	| $(Q \rightarrow P)$	?
?)	$P \rightarrow (Q \rightarrow P)$	$\rightarrow$ Intro 1-o

At this point, we don't know how many lines the proof will be, so we just call the last line '?' and the penultimate line 'o'. When we finish, we'll go through and fix the line numbers. Really, the proof now proceeds just like any ordinary PL proof. Look at our goal sentence in line o: it too is a conditional. SO we need to set up a nested subproof for another conditional intro:

1)	P	Assume
2)	Q	Assume
	:	
	:	
n)	P	?
o)	$(Q \rightarrow P)$	$\rightarrow$ Intro 2-n
?)	$P \rightarrow (Q \rightarrow P)$	$\rightarrow$ Intro 1-o

Again, note the use of the dummy line number 'n'. Now we've set up about as much as we can, and we just need to get from Q in line 2 to P. How the hell are we going to get P? . . .Oh, we already have it in line 1. So just reiterate! While we're at it, we'll adjust the line numbers appropriately

1)	P	Assume
2)	Q	Assume
3)	P	?
4)	$(Q \rightarrow P)$	$\rightarrow$ Intro 2-3
5)	$P \rightarrow (Q \rightarrow P)$	$\rightarrow$ Intro 1-4

F) Conclusion

This concludes our survey of PL. There are some interesting remaining topics, such as logic with a third truth-value, the completeness and consistency of PL (i.e., a demonstration that there are enough rules in the derivation system to derive everything the semantics allows, and no combination of rules that will allow us to derive anything unsanctioned by the semantics). There's also some interesting philosophical issues - what are the bearers of truth, sentences or abstract propositions? What is analytic truth? What is necessity? How many metalanguages are there? We'll save these questions for another class, or maybe another book.

In the next volume, we'll look at the logic of quantification, such as 'some' and 'all'. We'll also investigate the logic of the subject-predicate structure that we've abstracted away from in this volume.

Revised (uphanced) Strategery

Phase 1: work backwards from the conclusion

Look for conditionals (→) as the central connective of the conclusion. If there are any, set up an → intro subproof, with the antecedent as the first line of the subproof, and the consequent as the last line. Repeat Phase I on the new subconclusion. If there is no conditional conclusion (or subconclusion, go onto phase II.

Phase 2: Tear 'em down, build 'em up.

Break down the premises and the information derived from the premises:

Conditionals:

Look for the antecedent, and use → elim.

Look for the negation of the consequent, and use MT

Conjunctions:

Use ∧ elim

Disjunctions:

Look for the opposite of one disjunct and break the disjunction up with DS

Biconditionals:

Use ↔ elim

Negations:

Distribute them using deMorgans and LL

Double negations:

Use 2NE

As you derive new information, start this part all over again, breaking down the new information. When you've broken the premises as far down as you can, start to build up the information derived from the premises, into the subconclusions. Try working backwards to think about how to build up to your subconclusions. Use the following rules:

Disjunctions:

Look for one disjunct and use ∨ intro.

Conjunctions:

Look for both conjuncts and use ∧ intro.

Biconditionals:

Look for "backwards" and "forwards versions of the same conditional, and use ↔ intro.

Double negations:

2NI.

Phase III: The Hard Stuff (subproofs).

If you've got a disjunction in your premises, and you can't break it up, set up an Argument by Cases, with two subproofs. Each subproof will begin with one of the disjuncts, and end in the subconclusion that is your immediate goal.

If you're still totally stumped, set up a negation introduction subproof to prove your subconclusion. The subproof will begin by assuming the opposite of the subconclusion you're trying to prove, and it will end with some contradiction. Go back to Phase I at this point.