Notes for Allen & Hand, *Logic Primer*

Notes by Robin Smith (Texas A&M University)

Some Notes on Section 1.1

Arguments, Premises, and Conclusions

Logic is about **ARGUMENTS**. People use arguments in reasoning. Informally, arguments can be seen as offering reasons for believing things. Some reasons are good reasons for believing things, and some are not so good.

Sentences and Statements

The things we give reasons for, and the reasons we give for them, can be expressed in **SENTENCES**. More precisely, they can be expressed in sentences of a certain type: **sentences that are either true or false**.

Some sentences in English (or any other natural language) have the property that it *makes sense* to call them true or false. Consider these examples:

1. Today is Sunday.
2. There is a cockroach in my raspberry frappé
3. Two plus two is four.
4. Two plus two is five.
5. 2 + 2 = 5.
6. It will rain tomorrow.
7. It will rain at 2:00 PM in College Station, Texas, on March 13, 2050.
8. It rained in College Station, Texas, on March 13, 1950.
9. The number of people alive at this moment is an even number.
10. The set of all subsets of an infinite set is of higher cardinality than the set itself.

These are all sentences that it at least *makes sense to call* true or false in a way that it does not make sense to call any of these sentences true or false:

- What time is it?
- I'd like dry white toast and two fried chickens, please.
- Who was the seventh President of the United States?
- I wish this class were over!
- Get out of my room and leave me alone.
- I hope you die a prolonged and miserable death.

Try saying "That's true" or "That's not so" about each of these to see the difference.

How do you tell whether a sentence is a statement?

How do you recognize a sentence? That's part of what it is to understand a language. It's very complex, but children can do a pretty good job of this in their native languages by the age of five or so. Here are some sentences:

- Vita brevis, ars longa.
- Minä puhun Englantia, mutta en puhu paljon Suomea.
- J'aime beaucoup la logique, mais je déteste les logiciens.

Even in your own language, it's sometimes not obvious whether something is a sentence. Try these examples.

- The dog died.
- The dog the cat bit died.
- The dog the cat the possum ate bit died.
- The dog the cat the possum the baby befriended ate bit died.
- The dog the cat the possum the baby the armadillo corrupted befriended ate bit died.

Or even:

- The dog the dog the dog the dog the dog bit bit bit bit died.
- Buffalo buffalo buffalo buffalo buffalo.

Premises and Conclusions

Here are some arguments:

- Hieronymus is a Republican.
- **Therefore**, Hieronymus is a conservative.

- Farquhar has either a cat or a dog.
- Yesterday, I saw him with a dog.
- **Consequently**, he doesn't have a cat.

- My kitchen is ten feet long.
- In addition, the width of my kitchen is thirteen feet.
- **As a result**, the area of my kitchen is 150 square feet.

- Tabby's a cat and cats are mammals, **so** Tabby doesn't have feathers.

- If a year is divisible by four, then it is a leap year unless it is divisible by 100 and not divisible by 400.
- 2000 is divisible by four.
- 2000 is also divisible by 100.
- However, 2000 is divisible by 400.
- **Therefore**, 2000 had 366 days.

- Smith is a philosophy professor.
- Smith is **therefore** an imbecile.
- After all, only an imbecile would be a philosophy professor.

Each of these arguments consists of sentences, and in fact of sentences of the kind that must be true or false. In addition, one sentence in each of them is distinguished in a certain way. One way to describe the distinction is to say that that sentence is what the argument is trying to prove, or the point of the argument, while all the other sentences are offered as support for that sentence, or reasons for accepting it. We will use the term **CONCLUSION** for the sentence that's distinguished in this way, and we will call each of the other sentences a **PREMISE**.

So, how do we tell when an argument is going on, and how do we tell which sentence is the conclusion? Though we'll have a little more to say about that later, we're going to define **ARGUMENT** in an extremely broad way: an argument is just some sentences (the **PREMISES**) and another sentence (the **CONCLUSION**). Formally:

> An **ARGUMENT** is a pair of things:

- a set of sentences, the **PREMISES**
- a sentence, the **CONCLUSION**

On this definition, all of the following are arguments:

- Today is Thursday.
- If today is Thursday, then tomorrow is Friday.
- Therefore, tomorrow is Friday.

- Today is Thursday.
- Therefore, tomorrow is Friday.

(You don't have to have more than one premise)

- Today is Thursday.
- Today is Wednesday.
- Therefore, tomorrow is Friday.

(It doesn't have to be a *good* argument)

- Today is Thursday.
- Today is the sixth of the month.
- Six is a perfect number.
- The population of the United States is approximately 300,000,000.
- Therefore, there is no life on Mars.

(The premises don't have to have anything to do with the conclusion)

- Therefore, two plus two equals four.

(Strictly speaking, you can have an **empty** set of premises)

Valid and Invalid Arguments

Here is the single most important definition in this course:

> An argument is **VALID** if and only if it is necessary that *if* all its premises are true, *then* its conclusion is true.

A valid argument is an argument in which there is a certain **relationship between its premises and its conclusion**. That relationship concerns the **truth values** of the premises and conclusion

"Truth value" is a convenient way of saying "truth or falsehood". Arguments are composed of sentences that are either true or false, so every such sentence has a truth value. Its truth value is "true" if the sentence is true and "false" if the sentence is false (you're not surprised?).

Returning to validity, to say that an argument is valid is to say that the truth values of its premises and its conclusion are related in a certain way: **IF** the premises are **ALL** true, **THEN** the conclusion **MUST** be true.

Since this is easy to misunderstand, let's spend some time on it. First, it does **not** say that **in an argument**, if the premises are true then the conclusion must be true. Instead, it gives the criterion for a **valid** argument.

How can we tell whether the conclusion **must** be true if all the premises are true? Well, what's necessary is what can't possibly be otherwise, so if something **can** possibly be otherwise, then it's not necessary. (That, by the way, was an argument). So, to tell whether an argument is valid, we can:

- First, suppose or imagine that all the premises are true (regardless of whether or not they actually are true).
- Next, see if we can imagine the conclusion being false under those circumstances.
- If we can think of a way for the premises all to be true and the conclusion to be false at the same time, then we know that the argument is **INVALID**.
- If we are certain that there is no way for the premises all to be true at the same time that the conclusion is false, then we know that it is **VALID**.

This sounds like it depends rather a lot on how good we are at thinking up ways that things might be. In fact, we're going to develop some precise ways of doing that for certain arguments as this course proceeds. Let's take a quick look now at how you might proceed, however. Here's an argument:

- In order to pass Professor Abelard's course, Porphyry must either (1) have an average of at least C on each of the four exams and the final or (2) have a passing grade on all the exams and the final and submit an acceptable term paper.
- Porphyry has received grades of D on the first three exams.
- Each exam, including the final, is weighted equally.
- Although Porphyry wrote a term paper that would have been acceptable, a group of brigands has stolen it and will not return it to him unless he pays a ransom of $5000.
- Porphyry does not have $5000, and he does not know anyone who does.
- Being an honorable person, Porphyry does not steal.
- The term paper must be turned in by tomorrow to be acceptable.
- It is impossible for Porphyry to write another term paper by tomorrow.
- Therefore, Porphyry cannot pass Professor Abelard's course.

Sound Arguments

A **SOUND** argument is just a valid argument with true premises, that is:

> An argument is **SOUND** if and only if it is valid and all its premises are true.

What else can you conclude about an argument on the basis of that definition?

Exercise 1.1

All of these can be answered on the basis of the definitions already given.

i*. Every premise of a valid argument is true
> **NO**: Whether an argument is valid depends on **what would happen if the premises WERE all true**, non on whether they actually are all true.

ii*. Every invalid argument has a false conclusion
> **NO**: If the premises of a valid argument are not all true, then nothing follows about whether the conclusion is true or not.

iii*. Every valid argument has exactly two premises
> **NO**: An argument (valid or otherwise) may have any number of premises, including only one (or even including zero)

iv*. Some valid arguments have false conclusions

YES: The only thing that can't happen with a valid argument is having the conclusion false **when the premises are all true.**

v*. Some valid arguments have false conclusions despite having premises that are all true

NO: This almost exactly contradicts the definition of 'valid'.

vi*. A sound argument cannot have a false conclusion

YES: If a valid argument can't have a false conclusion when its premises are all true, and if a sound argument is a valid argument with true premises, then this follows right away.

vii*. Some sound arguments are invalid

NO: Part of the definition of **SOUND** is **VALID ARGUMENT**.

viii*. Some unsound arguments have true premises

YES: can you say which ones?

ix*. Premises of sound arguments entail their conclusions

YES: See the definition of **ENTAILS**.

x*. If an argument has true premises and a true conclusion, then it is sound.

We can talk about this in class

To the Syllabus

On to Exercise 1.2.1

What WFFS Are

and

How to Find Them

Notes for Allen/Hand, *Logic Primer* 1.2

Vocabulary: The symbols of our language are:

- **Sentence Letters:** $A, B, C, ..., A_1, B_1, C_1, ...$
- **Connectives:** $\sim, \&, \vee, \rightarrow, \leftrightarrow$
- **Parentheses:** (and)

Expression: Any string of the symbols of our language

Well-Formed Formula: An expression built according to the rules for well-formed formulas.

There are seven rules, but it's easier to think of them as divided into three groups:

Rule 1: A statement letter alone is a wff

The wffs defined by this rule are all exactly **one character long**:

$$A$$

$$B$$

$$Z_4$$
$$C_0$$

This is the only rule that a one-character wff can satisfy.

Rule 2: If ϕ is a wff, then $\sim\phi$ is a wff

Every wff defined by this rule **begins with the symbol** $\sim$:

$$\sim A$$

$$\sim C$$

$$\sim\sim E$$

This is the only rule that can introduce a '$\sim$' into a wff

If an expression **begins with** $\sim$, then it is a wff if and only if what remains after you remove the initial $\sim$ is a wff.

Rules 3-6: If ϕ and ψ are both wffs, then each of these is a wff

Rule 3: $(\phi \& \psi)$

Rule 4: $(\phi \vee \psi)$

Rule 5: $(\phi \rightarrow \psi)$

Rule 6: $(\phi \leftrightarrow \psi)$

Notice that every wff defined by any of these rules **begins with '(' and ends with ')'.** These are the only rules that introduce parentheses into wffs. Moreover, each rule introduces a matched pair of left and right parentheses:

- If a wff begins with '(', then it must end with ')'.

- If a wff begins with '(', then it must match one of Rules 3-6.

- Every '(' must be matched by one and only one ')'.

Let's combine these points into the beginnings of a method.

To tell whether an expression is a wff:

1. If it begins with '∼', then it must match Rule 2. Remove the '∼' and see whether the remainder of the expression is a wff.

2. If it begins with '(', then it must match one of Rules 3–6. Determine whether it matches one of these rules.

3. If it begins with anything else, then it must match Rule 1. In that case, the only possibility is that it is a sentence letter standing alone.

Note that the only things a wff can begin with are ∼, (, and a sentence letter.

Step 2 here needs to be filled in. Since parentheses always occur in pairs, we can start by looking for the mate of the initial '('. Starting at the beginning of the formula, move to the right one character at a time and count as follows:

- 2a. If the next character is '(', add 1 to the count.

- 2b. If the next character is ')', subtract 1 from the count.

- 2c. If the next character is anything else, leave the count unchanged.

- 2d. Stop when *either* the count is 0 *or* you come to the end of the expression.

When you come to a stop, there will be only two possibilities:

1. The count is 0 and you have stopped at a ')'. You have found the mate of the '(' that you started with.

2. The count is not 0 and you have run out of characters. **The expression is not a wff.**

An illustration of this appears on the next slide.

$$((A \rightarrow B) \vee \sim P)$$

$$\overbrace{(}^{count=1} \quad (A \rightarrow B) \vee \sim P)$$

$$(\quad \overbrace{(}^{count=2} \quad A \rightarrow B) \vee \sim P)$$

$$((\quad \overbrace{A}^{count=2} \quad \rightarrow B) \vee \sim P)$$

$$((A \quad \overbrace{\rightarrow}^{count=2} \quad B) \vee \sim P)$$

$$((A \rightarrow \quad \overbrace{B}^{count=2} \quad) \vee \sim P)$$

$$((A \rightarrow B \quad \overbrace{)}^{count=1} \quad \vee \sim P)$$

$$((A \rightarrow B) \quad \overbrace{\vee}^{count=1} \quad \sim P)$$

$$((A \rightarrow B) \vee \quad \overbrace{\sim}^{count=1} \quad P)$$

$$((A \rightarrow B) \vee \sim \quad \overbrace{P}^{count=1} \quad)$$

$$((A \rightarrow B) \sim P \quad \overbrace{)}^{count=0}$$

We got to the end, and the count was zero. Therefore, we know that the first '(' and the last ')' of $(A \to B) \vee \sim P$ match up.

We need to add a little more to Step 2. Rules 3–6 all give us wffs of the form:

$$(\phi \ connective \ \psi)$$

So, once we have found the matching outside parentheses, we remove them and see whether what is left is of the form:

$$\phi \ connective \ \psi$$

Here is what we should look for, then:

1. Starting at the left, look for a wff at the beginning.

2. The *next symbol after* that wff must be &, $\vee$, $\rightarrow$, or $\leftrightarrow$.

3. The rest of the expression after that symbol must be a wff

Let's try this with our example $((A \rightarrow B) \vee \sim P)$. When we take off the outside parentheses, we have $(A \rightarrow B) \vee \sim P$. The rest of the steps are on the next slide.

$$(A \rightarrow B) \vee \sim P$$

$$\overbrace{(}^{count=1} A \rightarrow B) \vee \sim P$$

$$(\overbrace{A}^{count=1} \rightarrow B) \vee \sim P$$

$$(A \overbrace{\rightarrow}^{count=1} B) \vee \sim P$$

$$(A \rightarrow \overbrace{B}^{count=1}) \vee \sim P$$

$$(A \rightarrow B \overbrace{)}^{count=0} \vee \sim P$$

We stop at the count of zero, and now we have found the first expression:

$$(A \rightarrow B)$$

Is this a wff? We proceed as before, taking off the outside parentheses:

$$A \rightarrow B$$

Let's start at the beginning as before. Here, the first symbol A is neither a '(' nor a '$\sim$'. So, we must use **Rule 1**: it must be a sentence letter standing alone.

Since we've found a wff, the next question is, "Is the next symbol one of &, $\vee$, $\rightarrow$, $\leftrightarrow$?" Here is what we find:

$$\overbrace{A}^{wff} \overbrace{\rightarrow}^{nextsymbol} B$$

The next symbol is indeed one of the four used in Rules 3–6, and so all that remains is to see whether the rest of the formula is a wff:

$$\overbrace{A}^{wff}\ \overbrace{\rightarrow}^{nextsymbol}\ \overbrace{B}^{wff?}$$

B is a wff by Rule 1, so we've satisfied all of the conditions. Summarizing where we are, then, we have found a wff that matches Rule 4:

$$(A \rightarrow B)$$

Going back to our larger formula, this is the wff on the left side. Let's call it ϕ. Our next step is to see whether the very next symbol after it is a connective of the right sort:

$$\overbrace{(A \rightarrow B)}^{\phi}\ \overbrace{\vee}^{nextsymbol}\ \sim P$$

It is: it's the wedge, used in Rule 2. Our expression will satisfy that rule if everything after the wedge is a wff. Let's call that ψ:

$$\overbrace{(A \to B)}^{\phi} \vee \overbrace{\sim P}^{\psi}$$

So, our final question is: is ψ a wff? Since it begins with $\sim$, we need to use Rule 2.

$$\psi = \sim P$$

Since this begins with $\sim$, it can be a wff only if it matches Rule 2. To check this, we remove the $\sim$ and see if the result is a wff:

$$P$$

But it is, so we're done.

Now let's put that back together into our original wff. We start with two wffs, the ones we just found:

$$(A \to B) \qquad\qquad \sim P$$

We then combine them using Rule 4:

$$((A \to B) \vee \sim P)$$

So, the entire expression corresponds to the form given in Rule 4:

$$(\overbrace{(A \to B)}^{\phi} \vee \overbrace{\sim P}^{\psi})$$

The whole method is summarized in the next slide.

1. If it begins with '∼', then it must match Rule 2. Remove the '∼' and see whether the remainder of the expression is a wff.

2. If it begins with '(',

   (a) Count parentheses from left to right as follows:

      i. If the next character is '(', add 1 to the count.

      ii. If the next character is ')', subtract 1 from the count.

      iii. If the next character is anything else, skip it.

      iv. Stop when *either* the count is 0 *or* you come to the end of the expression.

   (b) If you reach 0 before the end of the expression or you reach the end of the expression and the count is not 0, then stop: it's not a wff.

   (c) Otherwise, remove the outside parentheses and:

      i. Find the first wff from the left.

      ii. The next symbol must be &, ∨, →, or ↔. If not, stop: it's not a wff.

      iii. The rest of the expression must be a wff. If not, stop: it's not a wff.

3. If it begins with anything else, then it must be a sentence letter and match Rule 1. If not, stop: it's not a wff.

Recursive Rules

The rules for **constructing** wffs have an important property. They can be applied to the wffs that result from applying them. To use a technical term for this, they are **recursive**.

Since the result of applying a rule to wffs is a new wff, and since this process can be continued indefinitely, the rules for wffs define an **infinite** set of wffs.

Some Notes on Exercise 1.2.1

The rules of formation that define the language are *recursive*: roughly speaking, they can be applied to their own results to yield increasingly complicated formulas. This can be continued indefinitely far. However, each non-atomic wff is still constructed in a unique series of steps from other wffs, and each step involves the application of just one rule. It's the *last* rule used in constructing the wff that defines what kind of wff it is:

- If the last rule is rule 1, then the wff is **atomic**
- If the last rule is rule 2, the rule for '~', then the wff is a **negation**
- If the last rule is rule 3, the rule for '&', then the wff is a **conjunction**
- If the last rule is rule 4, the rule for 'v', then the wff is a **disjunction**
- If the last rule is rule 5, the rule for '→', then the wff is a **conditional**
- If the last rule is rule 6, the rule for '↔', then the wff is a **biconditional**

To tell whether an expression is a wff, you try to apply rules to it in reverse. Here' s Exercise 1.1, with some comments to show how to do this.

i. **A**
: This is a wff by rule 1. It's also an atomic sentence.

ii. **(A**
: This isn't a wff; the only rules that introduce parentheses are 3, 4, 5, and 6, and they all introduce pairs.

iii. **(A)**
: This isn't a wff either; the only rules introducing parentheses are 3, 4, 5, and 6, and they all require parentheses be put around two wffs connected by '&', 'v', '→', or '↔'.

iv. **(A → B)**
: By rule 5, this is a wff if A and B are wffs; by rule 1, A and B are both wffs. Since rule 5 is the only rule used, it's a conditional.

v. **(A→(**
: Not a wff because '→' must always have a wff on each side (among other things), with parentheses around the whole; '(' is not a wff.

vi. **(A→(B→C))**
: This has parentheses around it, so it's a wff if the result of taking off the parentheses , 'A→(B→C)', is two wffs with a connective between. But 'A' is a wff (rule 1); and '(B→C)' is also a wff (rule 5); so the whole thing is a wff (again by rule 5). Since rule 5 is the *last* rule used, it's a conditional.

vii. **((P&Q)→R)**
: As in the last example, this is a wff (by rule 5) if '(P&Q)' and 'R' are wffs. The first is a wff (in fact a conjunction), and the second is a wff (and atomic). So, the entire expression is a wff (and a conditional).

viii. **((A&B) v (C→(D↔G)))**
: This one is more complicated. First, take off the outside pair of parentheses; this gives:
(A&B)
v
(C→(D↔G))
By rule 3, that will be a wff if '(A&B)' and '(C→(D↔G)' are wffs. So, if it's anything, it's a disjunction. Obviously, '(A&B)' is a wff. '(C→(D↔G)' without its outside parentheses becomes 'C' , '→' , and '(D→G)'. The first is atomic and the last is a conditional, so the whole expression is a wff. Therefore, working back to the beginning, since both '(A&B)' and '(C→(D↔G)' are wffs, the original expression is a wff (and a disjunction).

ix. **~(A→B)**
: This consists of '(A→B)' , which is a wff (rule 5), preceded by '~', so it's a wff by rule 2 (and therefore a

negation).

x. **~(P→Q) v ~(Q&R)**

This one can be misleading. Let's start from the left. The only rule for introducing '~' is rule 2, which says that a tilde followed by a wff is a wff. So, if we find a '~', we have to find a wff after it. What follows the first '~'? The entire expression is ' (P→Q) v ~(Q&R)'. That begins and ends with parentheses, so it will be a wff if 'P→Q) v ~(Q&R' is two wffs with a connective between. Unfortunately, it isn't: 'P→Q)' and '(Q&R' aren't wffs. So, whatever the whole expression is, it can't be a negation. However, you may have noticed that '~(P→Q)' by itself *is* a wff, since it is the wff '(P→Q)' preceded by '~'. in the same way, '~(Q&R)' is a wff. The whole expression, then, consists of two wffs with a 'v' in between. Unfortunately, that's not a wff.

xi. **~(A)**

This would be a negation if '(A)' were a wff, but it isn't (see iii above).

xii. **(~A)→B**

Not a wff because: (1) parentheses need to surround the wffs connected by '→'; (2) '(~A)' is not a wff.

xiii. **(~(P&P)&(P↔(Qv~Q)))**

This is a wff, and in fact a conjunction. The following table may help show how it's put together (provided this works in your browser!):

(	~(P&P)				&	(P↔(Qv~Q))					)		
~		(P&P)				(	P	↔	(Qv~Q)				)
	(	P	&	P	)		P	(	Q	v	~Q		)
		P		P							~	Q	

xiv. **(~((BvP)&C)↔((Dv~G)→H))**

Biconditional with left side '~((BvP)&C)', right side '((Dv~G)→H)'. The left side, in turn, is the negation of '((BvP)&C)', and that in turn is a conjunction with conjuncts '(BvP)' and 'C' (and one of those is compound, too). The right side is a conditional with a disjunction for its antecedent and an atomic sentence for its consequent. And you should practice reading that whole description until it makes perfect sense.

xv. **(~(Qv~(B))v(E↔(DvX)))**

Not a wff because of the expression '(B)', which isn't a wff.

To the Syllabus

On to Exercise 1.2.2

About Dropping Parentheses

(You can also get this document as a PDF file)

Although I am not planning to test you on the correct application of the parenthesis-dropping convention (text, pp. 9-10), the exercises in the text often use it. Here's a quick guide to putting back in parentheses that have been dropped.

- **Rule 0:** The outside parentheses on wffs always get dropped, so you can always add them (exception: remember that there are no outside parentheses that accompany ~).
- **Rule 1:** Never add a parenthesis that comes between a ~ and anything else.
- **Rule 2:** You may add parentheses around & or v and the wffs they connect.
- **Rule 3:** After you have finished using Rule 2, if the wff contains both → and ↔ place parentheses around → and the wffs it connects.

These three rules will get you through most of the problematic cases.

Example 1

~P v Q → R

Add parentheses around the v, but don't come between ~ and anything:

(~P v Q) → R

Add a pair on the outside:

((~P v Q) → R)

Example 2

Q & ~R ↔ P v S

Add parentheses around & and v, but don't come between ~ and anything:

(Q & ~R) ↔ (P v S)

And add them on the outside:

((Q & ~R) ↔ (P v S))

Example 3

P → Q & R ↔ P & Q → R

First use Rule 2:

P → (Q & R) ↔ (P & Q) → R

Now, since thre is nothing left to apply Rule 2 to and there is a ↔ in the wff, apply Rule 3:

$$(P \rightarrow (Q \mathbin{\&} R)) \leftrightarrow ((P \mathbin{\&} Q) \rightarrow R)$$

Finally, apply Rule 0:

$$((P \rightarrow (Q \mathbin{\&} R)) \leftrightarrow ((P \mathbin{\&} Q) \rightarrow R))$$

Translating a Sentence to a WFF

There are three main steps to translating a sentence:

1. Take the sentence apart

2. Assign letters to the atomic sentences (that is, choose a **translation scheme**)

3. Put the WFF back together

1. Take the sentence apart

We disassemble the sentence with the following procedure:

1. Find the main connecting word(s)

2. Divide the sentence into the main connecting word and its parts

Note: a 'connecting word' may consist of several words ('only if') and may have parts that are separated in the sentence ('if ... then', 'either ... or', 'both ... and')

Finding the main connecting word

If there are several logical connecting words in a sentence, then you must determine which one of them is the main connecting word. A good way to proceed is to begin by marking all the connecting words. For example, here's a sentence

If either Smith is foggy or Jones is not wobbly, then either today is not Wednesday or, if today is Wednesday, then it is raining only if it is not snowing.

And here it is with all the connecting words marked:

If either Smith is foggy **or** Jones is **not** wobbly, **then either** today is **not** Wednesday **or**, **if** today is Wednesday, **then** it is raining **only if** it is **not** snowing.

The main connecting word is then usually the one that will divide the sentence into one or two pieces each of which is still a sentence:

If (**either** Smith is foggy **or** Jones is **not** wobbly), **then** (**either** today is **not** Wednesday **or**, **if** today is Wednesday, **then** it is raining **only if** it is **not** snowing)

To keep track of how the sentence divides, use paren- theses as above

Keep finding the main connecting words of the parts, using the same procedure

Working within the divisions marked by parentheses, find the main connecting words of the parts:

If (**either** (Smith is foggy) **or** (Jones is **not** wobbly)), **then** (**either** (today is **not** Wednesday) **or** , (**if** (today is Wednesday)**, then (** (it is raining) **only if** (it is **not** snowing))))

We've left 'not' unanalyzed here since it usually gets inserted into the sentence it works on.

Move all the occurrences of ‘not’ out of the sentences they apply to, and indicate the range with square brackets:

If (**either** (Smith is foggy) **or** (**not**[Jones is wobbly])), **then** (**either** (**not**[today is Wednesday]) **or** , (**if** (today is Wednesday)**,** **then** **(** (it is raining) **only if** (**not**[it is snowing]))))

(The reason for using square brackets is that the rules for our symbolic language do not allow parentheses surrounding a negation)

2. Assign a Translation Scheme

*Identify all the atomic sentences, and assign a **unique** letter to each one (this is called a **translation scheme)**:*

If (**either** (Smith is foggy) **or** (**not**[Jones is wobbly])), **then** (**either** (**not**[today is Wednesday]) **or** , (**if** (today is Wednesday**),** **then** **(** (it is raining) **only if** (**not**[it is snowing]))))

Translation scheme:

'Smith is foggy'	= S
'Jones is wobbly'	= J
'Today is Wednesday'	= W
'It is raining'	= R
'It is snowing'	= T

3. Put the WFF back together

We're now ready for the third step: putting the analyzed sentence back together as a WFF. Start by replacing the atomic sentences with their assigned letters:

If (**either** (S) **or** (**not**[J])), **then** (**either** (**not**[W]) **or** , (**if** (W)**, then (** (R) **only if** (**not**[T]))))

At this point, we do have some sentence letters surrounded by parentheses. We'll fix that in a moment.

Working from the inside out, translate the connecting words. This proceeds step by step:

If (**either** (S) **or** ($\sim[J]$)), **then** (**either** ($\sim[W]$) **or** , (**if** (W)**,** **then** **(** (R) **only if** ($\sim[T]$))))

At this point, let's take off the square brackets and all parentheses around negations or atomic sentences:

If (**either** S **or** $\sim J$), **then** (**either** $\sim W$ **or** , (**if** W**,** **then** **(** R **only if** $\sim T$)))

Now we translate the connecting words, using the standard pattern for each word.

'only if': ϕ only if $\psi \Rightarrow (\phi \to \psi)$:

If (**either** S **or** $\sim J$), **then** (**either** $\sim W$ **or** , (**if** W, **then** ($R \to \sim T$)))

'if ... then': if ϕ then $\psi \Rightarrow (\phi \to \psi)$

If (**either** S **or** $\sim J$), **then** (**either** $\sim W$ **or** $(W \to (R \to \sim T))$

(Continuing ...)

'either ... or': either ϕ or $\psi \Rightarrow (\phi \vee \psi)$

If (either S **or** $\sim J$ **), then (** $\sim W \vee$ ($W \rightarrow (R \rightarrow \sim T)$)**)**

'either ... or': either ϕ or $\psi \Rightarrow (\phi \vee \psi)$

If ($S \vee \sim J$ **), then (** $\sim W \vee$ ($W \rightarrow (R \rightarrow \sim T)$)**)**

And now the final 'if ... then': if ϕ *then* $\psi \Rightarrow (\phi \rightarrow \psi)$

$$(S \vee \sim J) \rightarrow (\sim W \vee (W \rightarrow (R \rightarrow \sim T)))$$

There's one more step. We need outside parentheses surrounding the entire WFF (unless it's a negation or atomic):

$$((S \vee \sim J) \rightarrow (\sim W \vee (W \rightarrow (R \rightarrow \sim T))))$$

The Original Sentence:

If either Smith is foggy or Jones is not wobbly, then either today is not Wednesday or, if today is Wednesday, then it is raining only if it is not snowing.

Analyzed into its parts:

If (**either** (Smith is foggy) **or** (Jones is **not** wobbly)), **then** (**either** (today is **not** Wednesday) **or** , (**if** (today is Wednesday)**), then (** (it is raining) **only if** (it is **not** snowing))))

Translation scheme:

'Smith is foggy' = S; 'Jones is wobbly' = J; 'Today is Wednesday' = W; 'It is raining' = R; 'It is snowing' = T

Translation:

$$((S \vee \sim J) \rightarrow (\sim W \vee (W \rightarrow (R \rightarrow \sim T))))$$

Neither Smith nor Jones attended the meeting if both Brown and Green did

Mark connecting words:

Neither Smith **nor** Jones attended the meeting **if both** Brown **and** Green did

Spell out abbreviations:

Neither Smith *attended the meeting* **nor** Jones attended the meeting **if both** Brown *attended the meeting* **and** Green *attended the meeting*

Take it apart:

(**Neither** (Smith *attended the meeting*) **nor** (Jones attended the meeting)) **if** (**both** (Brown *attended the meeting*) **and** (Green *attended the meeting*))

Translation scheme:

S = 'Smith attended the meeting'; J = 'Jones attended the meeting'; B = 'Brown attended the meeting'; G = 'Green attended the meeting'

(**Neither** (S) **nor** (J)) **if** (**both** (B) **and** (G))

Translation:

'neither ... nor': neither ϕ *nor* $\psi \Rightarrow$ $(\phi \vee \psi)$

($\sim$ ($S \vee J$)) **if** (**both** (B) **and** (G))

'both ... and': both ϕ *and* $\psi \Rightarrow$ $(\phi \& \psi)$

($\sim$ ($S \vee J$)) **if** (($B \& G$))

'if': ϕ *if* $\psi \Rightarrow (\psi \rightarrow \phi)$

$((B \& G) \rightarrow \sim (S \vee J))$

If

and

Only if

and

If and only if

Logicians like to translate ‘if’ (sometimes with an optional ‘then’), ‘only if’, and ‘if and only if’ in three different ways. Here’s a summary of the relevant translations:

If P (then) Q	$P \rightarrow Q$
P if Q	$Q \rightarrow P$
P only if Q	$P \rightarrow Q$
P if and only if Q	$P \leftrightarrow Q$

Why do we do this?

Starting with the easiest case, there's a crucial difference between "P if Q" and "If P, Q". Think about the meanings of these three sentences:

1. *If we're in Houston, then we're in Texas.*

2. *We're in Houston if we're in Texas.*

3. *We're in Texas if we're in Houston.*

It's obvious (I hope) that sentence 1 means the same thing as sentence 3 but **not** the same thing as sentence 2.

Now, we use $P \rightarrow Q$ as a translation of "If P then Q". Since "Q if P" means the same thing, we should translate it the same way:

If P (then) Q	$P \rightarrow Q$
Q if P	$P \rightarrow Q$

In each of these sentences, the same sentence comes **after the word 'if'**. So, in general, when we have a conditional, the sentence immediately after 'if' is the **antecedent** of the conditional.

Now, consider the difference between **if** and **only if**. Compare these two sentences:

1. The patient survived if he had the operation.

2. The patient survived only if he had the operation.

Do these mean the same thing? Compare what happens if we add a little more to each:

1. The patient survived if he had the operation, and even then he didn't survive if the the surgeon was not unusually skillful.*

2. The patient survived only if he had the operation, and even then he didn't survive if the surgeon was not unusually skillful.*

Sentence 1* seems to be self-contradictory: it says that the patient **will survive** if he has the operation, but then it takes that away by saying that under certain circumstances he **will not** survive if he has the operation. Sentence 2*, on the other hand, seems to be perfectly consistent.

But consider these two sentences:

3. **If** *the patient survived,* **then** *he had the operation.*

4. The patient survived **only if** *he had the operation.*

Do these mean the same thing? If so, then we should translate them the same way:

The patient survived $\longrightarrow$ *he had the operation.*

Finally, consider '**if and only if**'. If 'if' and 'only if' mean different things, then 'if and only if' can't mean the same thing as each of them. In fact, it means the combination of both of them (as you might guess).

"The patient survived **if and only if** *he had the operation"*

=

"The patient survived **if** *he had the operation,*

and

the patient survived **only if** *he had the operation"*

You might think of translating this as follows:

(The patient had the operation $\rightarrow$ *he survived)*

&

(the patient survived $\rightarrow$ *he had the operation)*

However, logicians have a special fondness for this kind of sentence, and so they have a special connective for just such a case:

The patient survived $\leftrightarrow$ *he had the operation*

Some Notes on Exercise 1.3

Solutions for all of these are given in Allen/Hand. Here are a few notes on how those solutions are arrived at.

FINDING THE MAIN CONNECTIVE. When English sentences contain more than one connective, the trick to translating them into our formal language is determining which connective is the *main* connective, that is, the one that operates on the largest pieces (or piece) of the sentence. Our formal language is so defined that *every* non-atomic sentence has one and only one main connective. In English, it's sometimes not quite so nice. However, if you can succeed in finding the main one, all you have to do is translate it, then translate the components it operates on. If those components are atomic, then all you need to to is assign them letters. If they're non-atomic, then you translate them in exactly the same way. To use a word I've used before, this process is *recursive*: the process of looking for the main connective in one of the components is just like the process of looking for it in the whole sentence. It's also guaranteed to end, eventually, since every time you translate a connective, you wind up with smaller components; eventually, you will reach components with no connectives at all, at which point you're done.

Here's an illustration. Consider the sentence

If it's raining but it's not pouring, then unless it's Thursday, this is Harlingen.

A good first step in translating this is to go through and mark all the connectives:

> ***If*** it's raining ***but*** it's ***not*** pouring, ***then unless*** it's Thursday, this is Harlingen.

Now, which one of these is the main connective? A test that often works is this: if you take out anything that's *not* the main connective and divide the sentence at that point, the pieces left behind are not (both) complete sentences. To illustrate, here's what you get if you split the above example at '***but***':

> ***If*** it's raining

> it's ***not*** pouring, ***then unless*** it's Thursday, this is Harlingen.

Neither of these is a coherent sentence, and so 'but' is not the main connective of the example. What about 'unless'? We get:

> ***If*** it's raining ***but*** it's ***not*** pouring, ***then***

and

> it's Thursday, this is Harlingen.

The first of these obviously isn't a coherent sentence, and the second is at least a little rocky. What, then, about '**if**'?

Before we look at the result, here's a little note about '**if**'. Frequently, '**if**' is associated with '**then**' to form a two-piece connective. You need to regard the 'then' as part of an '**if...then**', not as a separate connective, in such cases. To apply our test, what we do is delete the entire '**if...then**' pair, to get:

> it's raining ***but*** it's ***not*** pouring,

and

unless it's Thursday, this is Harlingen.

Notice that these are two perfectly coherent sentences? That tells you that the main connective here is '**if...then**'. Now, 'if A then B' translates into 'A→B'. Let's do that much translation here:

(it's raining ***but*** it's ***not*** pouring) → (***unless*** it's Thursday, this is Harlingen)

To continue translating this, we translate each of the parts in the same way.

(it's raining ***but*** it's ***not*** pouring)

becomes

(it's raining) **&** (it's ***not*** pouring)

And

(it's ***not*** pouring)

becomes

~(it's pouring)

So, the whole left side is

(it's raining **&** **~**(it's pouring)

Now, let's turn to the right hand side. This has '**unless**' out front. It's frequently easier if we rearrange the sentence to put the second component after '**unless**' in front of it:

(This is Harlingen***unless*** it's Thursday)

And we can translate that as

(This is Harlingen) **v** (it's Thursday)

So, the whole sentence looks like this:

(it's raining **&** **~**(it's pouring)→((this is Harlingen) **v** (it's Thursday))

Just turn the atomic sentences into letters and this becomes:

((P&~Q)→(RvS))

(Of course, you can drop unnecessary parentheses from this.)

Some notes on Exercise 1.3

1. *John is dancing but Mary is not dancing.*
   P&~Q. This is quite straightforward (but notice that the
2. *If John does not dance, then Mary will not be happy.*
   ~P→~T. Note that the 'if' has a 'then' with it.
3. *John's dancing is sufficient to make Mary happy.*
   P→T. Also straightforward.

4. *John's dancing is necessary to make Mary happy.*
   T→P. Watch this one: the order of antecedent and consequent is **vital**.
5. *John will not dance unless Mary is happy.*
   ~TvP. Also straightforward.
6. *If John's dancing is necessary for Mary to be happy, Bill will be happy.*
   (T→P)→~U). Finding the main connective is a little tricky here. Note that you could add 'then' where the comma is without changing the meaning of the sentence. Rewrite it like that, and analyzing is straightforward.
7. *If Mary dances although John is not happy, Bill will dance.*
   (Q&~S)→R. Same note as before.
8. *If neither John nor Bill is dancing, Mary is not happy.*
   ~(PvR)→~T. Notice that 'neither...nor' also functions as a single unit. This is an important help in finding the main connective. Also notice that this is partly abbreviated: 'neither John nor Bill is dancing' is a short way of saying 'neither John is dancing nor Bill is dancing.'
9. *Mary is not happy unless either John or Bill is dancing.*
   ~Tv(PvR). Notice again that 'either...or' is a single unit.
10. *Mary will be happy if both John and Bill dance.*
   (P&R)→T. Abbreviation again: 'both John and Bill dance' = 'Both John dances and Bill dances.'
11. *Although neither John nor Bill is dancing, Mary is happy.*
   T&~(PvR). Here, the parts were rearranged to put 'although' in the middle.
12. *If Bill dances, then if Mary dances John will too.*
   R→(Q→P). The structure is 'If (Bill dances), then (if Mary dances John will too)'. An 'if' with a 'then' often provides clues about what the main connective is. Notice also that 'John will too' is shorthand for John will dance.'
13. *Mary will be happy only if Bill is happy.*
   T→U. Straightforward. Notice that 'only if' translates into '→' and keeps the antecedent and consequent in the same order. **AND REMEMBER THAT 'ONLY IF' IS NOT THE SAME THING AS 'IF'.**
14. *Neither John nor Bill will dance if Mary is not happy.*
   ~T→~(PvR). The structure is '(Neither John nor Bill will dance) if (Mary is not happy),' and that's translatable as '(Mary is not happy)→(neither John nor Bill will dance)'. **AS I WAS JUST SAYING, 'IF' AND 'ONLY IF' ARE NOT THE SAME CREATURE.**
15. *If Mary dances only if Bill dances and John dances only if Mary dances, then John dances only if Bill dances.*
   (Q→R)&(P→Q)→(P→R). This is a real exercise for finding the main connective. It helps to add back some unnecessary parentheses: ((Q→R)&(P→Q))→(P→R). The main connective is 'if...then': the first 'if' goes with the 'then'.
16. *Mary will dance if John or Bill but not both dance.*
   (PvR)&~(P&R)→Q. Well, I don't know whether I think this example is quite English. I would have preferred something like 'Mary will dance if John or Bill dances, but not both.' However, the main connective is 'if': (Mary will dance) if (John or Bill but not both dance) is the structure. So, that becomes: (John or Bill but not both dance)→(Mary will dance). The cumbersome 'John or Bill but not both dance' might be rewritten as '(John or Bill dances) but (John and Bill do not both dance).' John or Bill dances' = 'John dances or Bill dances.' 'John and Bill do not both dance' = 'not (John dances and Bill dances)'; the 'not both' construction is a way of indicating that the 'not' governs the *entire conjunction*.
17. *If John dances and so does Mary, but Bill does not, then Mary will not be happy but John and Bill will.*
   (P&Q)&~R→(~T&(S&U)). Here's a really good example on which to try out tests for the main connective. Adding a word or two to spell out abbreviations, and marking off the main connective and its components, we get:

   If (John dances and Mary dances, but Bill does not dance), then (Mary will not be happy but John will be happy and Bill will be happy)

That is:

(John dances and Mary dances, but Bill does not dance)→(Mary will not be happy but John will be happy and Bill will be happy)

From here on out, it's easy.

18. *Mary will be happy if and only if John is happy.*
    T↔S. That's what double-arrows are for. **AND NOTICE THAT 'IF AND ONLY IF' IS NOT THE SAME THING EITHER AS 'IF' OR AS 'ONLY IF'.**
19. Provided that Bill is unhappy, John will not dance unless Mary is dancing.
    ~U→(~PvQ). Just be careful here to notice the order of antecedent and consequent.
20. If John dances on the condition that if he dances Mary dances, then he dances.
    ((P→Q)→P)→P. Sounds hard to follow, but the rules suggested for finding the main connective work very well. The connectives are as marked here:

    If John dances **on the condition that if** he dances Mary dances, **then** he dances

    The only trick is telling which 'if' goes with the 'then.' But if you take out the second 'if' and the 'then,' you get these three pieces

    If John dances **on the condition that**

    he dances Mary dances

    and

    he dances

    This is gibberish. However, if you take out the first 'if' and the 'then', you get:

    John dances **on the condition that if** he dances Mary dances,

    and

    he dances

    And those are perfectly coherent sentences. So, the structure is

    If (John dances **on the condition that if** he dances Mary dances,) **then** (he dances)

    which is:

    (John dances **on the condition that if** he dances Mary dances,) → (he dances)

    Analyzing the left side, we get:

    (**if** (he dances) (Mary dances))→(John dances)

    That is, ((P→Q)→P)
    So, the whole thing is ((P→Q)→P)→P.
21. If a purpose of punishment is deterrence and capital punishment is an effective deterrent, then capital punishment should be continued.
    Structure:

    If ((a purpose of punishment is deterrence) **and** (capital punishment is an effective deterrent)), **then** (capital punishment should be continued).

    Adding connectives:

((a purpose of punishment is deterrence) & (capital punishment is an effective deterrent)) → (capital punishment should be continued).

Replacing atomic constituents and deleting one pair of parentheses:

(P&Q) → R

22. Capital punishment is not an effective deterrent although it is used in the United States.

Structure (and spelling out a little abbreviation):

(Capital punishment is **not** an effective deterrent) **although** (capital punishment is used in the United States).

Substituting connectives:

~(Capital punishment is an effective deterrent) & (capital punishment is used in the United States).

What we get:

~Q&S

23. Capital punishment should not be continued if it is not an effective deterrent, unless deterrence is not a purpose of punishment.

Structure:

((Capital punishment should **not** be continued) **if** (capital punishment is **not** an effective deterrent), **unless** (deterrence is **not** a purpose of punishment).

Part of the way there:

(~(Capital punishment should be continued) **if** ~(capital punishment is an effective deterrent), v ~(deterrence is a purpose of punishment).

That is to say:

(~(capital punishment is an effective deterrent) → ~(Capital punishment should be continued) v ~(deterrence is a purpose of punishment).

NOTE THE ORDER HERE!!

(~Q → ~R) v ~P

24. If retribution is a purpose of punishment but deterrence isn't, then capital punishment should not be continued.

Structure:

If ((retribution is a purpose of punishment) **but** (deterrence is**n't** a purpose of punishment)), **then** (capital punishment should **not** be continued).

((retribution is a purpose of punishment) & ~(deterrence is a purpose of punishment)) → ~(capital punishment should be continued).

(T&~P) → ~R

25. Capital punishment should be continued even though capital punishment is not an effective deterrent, provided that a purpose of punishment is retribution in addition to deterrence.

Structure, and a little rewriting into something not quite English:

((Capital punishment should be continued) **even though** (capital punishment is **not** an effective deterrent)), **provided that** ((a purpose of punishment is retribution) **in addition to** (a purpose of punishment is deterrence)).

((Capital punishment should be continued) & ~(capital punishment is an effective deterrent)), **provided that** ((a purpose of punishment is retribution) & (a purpose of punishment is deterrence)).

So (watching the order again--careful with 'provided that'!):

(T&P) → (R & ~Q)

To the Syllabus

On to Exercise 1.4.1

Back to Exercise 1.2.2

Arguments and Proofs

For the next section of this course, we will study PROOFS. A proof can be thought of as the formal representation of a process of reasoning. Proofs are comparable to arguments, since they can be seen as having premises and conclusions. That is only part of the story, however.

Let's begin by returning to the definition of an argument. An argument is two things:

1. A set of sentences (the **premises**)

2. A sentence (the **conclusion**)

Arguments Made of WFFs

Our language for sentential logic also contains sentences, so this definition can be applied to it. A sentence in the formal language is a **well-formed formula** (**wff**). So, we can also say that an argument **in sentential logic** is:

1. A set of wffs. (the **premises**)

2. A wff. (the **conclusion**)

An example would be:

Premises: $(P \vee \sim S)$
$\sim(Q \leftrightarrow (R \& S))$
$(P \rightarrow (Q \& R))$

Conclusion: $(Q \vee (P \leftrightarrow \sim R))$

Sequents and Arguments

For convenience, and just because it's what we want to do, we will write arguments in another way, and when we do that we will call them by another name: **SEQUENTS**. A sequent is just an argument written on one line, with the premises first (separated by commas) and the conclusion last (separated from the premises by the symbol '$\vdash$'). So, the argument on the previous page, written as a sequent, looks like this:

$$(P \vee \sim S), \sim(Q \leftrightarrow (R \& S)), (P \rightarrow (Q \& R)) \vdash (Q \vee (P \leftrightarrow \sim R))$$

The symbol '$\vdash$' is called the **TURNSTILE**.

A Small Confession

In general, I have not required that you learn the details of the parenthesis-dropping convention. However, in these notes the **outside pair of parentheses** for wffs will often be dropped. That is, you will find such forms as $P \rightarrow \sim(Q\&R)$ instead of the strictly correct $(P \rightarrow \sim(Q\&R))$.

Valid Arguments and Reasoning

An argument is a set of premises and a conclusion. If the premises and conclusion are related in such a way that it's impossible for the conclusion to be false if the premises are all true, then we call it **valid**. How do we tell if an argument is valid?

One way, and one that we often use, is to see if we can **reason deductively** from the premises to the conclusion. It's best to explain this with an example. Consider the following argument:

$P1$	My kitchen is ten feet long and twelve feet wide.
$P2$	George's kitchen is eight feet long and fourteen feet wide.
C	Therefore, my kitchen is bigger than George's kitchen.

Why is it valid? You might defend this by reasoning as follows:

$P1$	My kitchen is ten feet long and twelve feet wide.
$P2$	George's kitchen is eight feet long and fourteen feet wide.
$R3$	*If my kitchen is ten feet long and twelve feet wide, then its area is one hundred twenty square feet*
$R4$	*So, the area of my kitchen is one hundred twenty square feet*
$R5$	*If George's kitchen is eight feet long and fourteen feet wide, then its area is one hundred twelve square feet*
$R6$	*So, the area of George's kitchen is one hundred twelve square feet*
$R7$	*One hundred twenty square feet is a larger area than one hundred twelve square feet*
C	Therefore, my kitchen is bigger than George's kitchen.

The additional sentences in red are a series of steps that start with the premises of the argument and eventually reach the conclusion. Each step follows necessarily from previous steps, and the last step is the conclusion. Therefore, these additional sentences **show that this argument is valid**. Let's call these intermediate steps **REASONING**.

Reasoning and Proof

A **proof** is like an argument with reasoning added to show that it is valid. However, proofs are in the formal language of sentental logic, not English. We use proofs to model the process of reasoning.

In the example of reasoning above, I didn't explain how we know that the reasoning actually does show that the argument is valid. Instead, I just relied on "intuition": it seems obvious that this reasoning works. But what makes it seem obvious?

One of the purposes of logic is to give an answer to that question. We will give a precise definition of a **proof** in the language of sentential logic that corresponds to the intuitive idea of a valid argument with reasoning added to show that it is valid.

Just as we defined the sentences of sentential logic exactly with a list of basic vocabulary and a set of rules, we'll also define a proof using a precise set of rules. These rules will be easier to understand if we start by looking at a proof.

A Proof

Here is a **proof of the sequent** $P \to Q, Q \to R \vdash P \to R$:

1	(1)	$P \to Q$	A
2	(2)	$Q \to R$	A
3	(3)	P	A
1,3	(4)	Q	1,3 $\to$E
1,2,3	(5)	R	2,4 $\to$E
1,2	(6)	$P \to R$	5 $\to$I (3)

Notice that this proof is made up of **lines**. Each line has a certain structure. It begins with one or more numbers, separated by commas. After that, there is a number in parentheses. Next comes a wff. Finally, at the right side of the line, there is a mysterious and very short string. Each of these parts has a name:

$$\overbrace{1,2,3}^{Assumption\ set} \quad \overbrace{(5)}^{Line\ no.} \quad \overbrace{R}^{Sentence} \quad \overbrace{2,4 \to \mathrm{E}}^{Annotation}$$

Each of these different parts has a function:

Assumption set: The line numbers of all the lines on which this line depends.

Line number: Which line this is (lines are numbered in order, starting with one).

Sentence: The contents of this line.

Annotation: The rule that allows us to add the sentence, and the number(s) of the line(s) we applied that rule to.

Whenever we add a line to a proof, we add a **sentence** by applying a **rule** to **previous lines**. The **assumption set** we put at the front of the line is determined by the assumption set(s) of the line(s) we used and the particular rule we used.

Primitive Rules of Proof

There are ten basic rules of proof which we call **primitive rules** because they are the fundamental rules in terms of which proofs are defined. (Later on, we'll see how we can add more rules to the system.) These include:

1. The **assumption** rule (the simplest of them all)

2. For each of the connectives $\&$, $\vee$, $\rightarrow$, $\leftrightarrow$, two rules: an **intro** rule and an **elim** rule.

3. The rule called **reductio ad absurdum**

Assumption

assumption	Assume any sentence

This is the simplest rule of all, and it is also the rule with which a proof usually starts. You may assume any sentence you like, no restrictions. This can correspond to taking something as a premise in an argument, but there are also cases where you will want to make assumptions for other purposes.

When you add a line using **assumption**, you give it **an assumption set containing just its own line number**. An assumption doesn't depend on anything else, but other things may depend on it.

When you add a line using **assumption**, you add an **annotation** consisting simply of the letter **A**.

Ampersand-Intro (&I)

ampersand-intro	Given two sentences (at lines *m* and *n*, conclude a conjunction of them.

What this rule permits you to do is **add a conjunction**, that is, add a sentence with & as its main connective. The two lines can be any previous lines of the proof.

.	.	.	.
1,2,3	(*m*)	ϕ	
.	.	.	
2,4	(*n*)	ψ	
1,2,3,4	(*k*)	$(\phi \& \psi)$	m, n&I

The assumption set for the line we add includes all the numbers that are in either of the lines we used. Notice that we include each number only once. (By the way, the numbers in this example are just made up.)

A Closer Look at Annotations

The annotation for &I has certain form:

.	.	.	.	
1,2,3	(m)	ϕ		
.	.	.		
2,4	(n)	ψ		
1,2,3,4	(k)	$(\phi \& \psi)$	m,n	&I
			line numbers	**Rule name**

The annotation begins with **the numbers (or number) of the lines (or line) to which the rule was applied**, separated by a comma (if there is more than one number). Following that is **the name of the rule**. We will see this same structure for the annotations in several rules: &E, $\vee$I, $\vee$E, $\rightarrow$E, $\leftrightarrow$I, and $\leftrightarrow$E. Some of these rules apply to two lines at a time: &I, $\vee$E, $\rightarrow$E, $\leftrightarrow$I, $\leftrightarrow$E. The rest apply to only one line: &E, $\vee$I.

A closer Look at Assumption Sets

Every rule specifies how to determine the **assumption set** for the line it adds, based on the assumptions set(s) of the line(s) it is applied to. **Getting this part right is critical**. In the case of &E, the assumption set is *the union of the assumption sets* of the lines to which the rule is applied. 'Union' is a term from set theory: the union of two sets is the set of all the things that are in either of them. In other words, the assumption set here contains **all the numbers that are in either of the assumption sets** of the lines that the rule applies to:

.	.	.	.
1,2,3	(*m*)	ϕ	
.	.	.	
2,4	(*n*)	ψ	
1,2,3,4	(*k*)	$(\phi \& \psi)$	m,n &I

Ampersand-Elim (&E)

ampersand-elim	Given a sentence that is a conjunction (at line *m*), conclude either conjunct.

What this rule permits you to do is **get either part of a conjunction**, that is, add either the left or the right conjunct of a conjuction as a line.

.	.	.	.
2,4	(*m*)	$(\phi \& \psi)$	
.	.	.	
2,4	(*k*)	ϕ	m&E

We could equally well have added ψ.

The assumption set here is the same as the assumption set in front of the line containing the conjunction we used.

Wedge-Intro ($\vee$I)

wedge-intro	Given a sentence (at line *m*), conclude any sentence having it as a disjunct.

What this rule permits you to do is **take any line you like and add a line that contains the sentence you get by combining that line with any other sentence whatsoever, using the wedge**. The sentence you combine it with can be anything: it doesn't have to occur already in the proof as a line.

.	.	.	.
1,2,3	(*m*)	ϕ	
.	.	.	
1,2,3	(*k*)	$(\phi \vee \psi)$	$m \vee$I

We could equally well have added $(\psi \vee \psi)$. The sentence ψ can be anything

The assumption set for the line we add is the same as the assumption set for the line we used, that is, *m*.

Wedge-Elim ($\vee$E)

wedge-elim Given a sentence (at line *m*) that is a disjunction and another sentence (at line *n*) that is a denial of one of its disjuncts, conclude the other disjunct.

What this rule permits you to do is **get one of the disjuncts of a disjunction**. For the definition of a **denial**, see p. 7 of the text. Briefly, if there are two sentences, one of which is the other sentence with a $\sim$ in front of it, then the two sentences are denials of each other.

.	.	.	.
1,2,3	(*m*)	$(\phi \vee \psi)$	
.	.	.	
2,4	(*n*)	$\sim \phi$	
.	.	.	
1,2,3,4	(*k*)	ψ	$m, n\vee$E

The assumption set here includes all the numbers on either of the lines *m* or *n*.

Arrow-Elim (→E)

arrow-elim	Given a conditional sentence (at line *m*) and another sentence that is its antecedent (at line *n*), conclude the consequent of the conditional.

What this rule permits you to do is **get the consequent (right side) of a conditional**. To apply it, you need the conditional on one line and its **antecedent** (left side) on the other. **NOTE: there is no rule that lets you get the antecedent from a conditional and its consequent**.

.	.	.	.
1,2,3	(*m*)	$(\phi \to \psi)$	
.	.	.	
2,4	(*n*)	ϕ	
.	.	.	
1,2,3,4	(*k*)	ψ	m, n →E

The assumption set here includes all the numbers on either of the lines *m* or *n*.

Proofs

Before we go on to $\rightarrow$I and RAA, which introduce a new feature of rules, let's pause to consider a proof that uses many of the rules we've introduced. We can start by defining **PROOF** precisely (see p. 17 of the text):

> A **PROOF** is a sequence of lines containing sentences. Each sentence is either an assumption or the result of applying a rule of proof to earlier sentences in the sequence.

The meaning of this should now be clear. Notice that it adds one important requirement: **every proof starts with one or more assumptions**. (An assumption, in a proof, is a line that was introduced by the rule **assumption**).

Proofs for Arguments

> A **PROOF FOR A GIVEN ARGUMENT** is a proof whose last sentence is the argument's conclusion depending on nothing other than the argument's premises.

To rephrase this, here is how we construct a proof for a given argument:

1. Begin by **adding all of the premises of the argument**, each on one line, using the rule **assumption**.

2. Add more lines to the proof, using the rules, until you reach **a line containing the conclusion** of the argument.

3. If the **assumption set** of the line containing the conclusion includes **only the numbers of lines containing premises of the argument**, then then entire proof is a proof of the given argument.

Working through an example will help. Here is an argument (sequent):

$$P\& \sim Q, (P \vee S) \rightarrow (Q \vee R) \vdash P\&R$$

We start by assuming each of the premises:

1	(1)	$P\& \sim Q$	A
2	(2)	$(P \vee S) \rightarrow (Q \vee R)$	A

Our goal in constructing a proof is then to add lines in accordance with the rules until we arrive at a line that meets two criteria:

1. Its sentence is the conclusion, $P\&R$

2. Its assumption set includes only the assumption numbers of the premises (1 and 2).

What do we do next? We will disucss strategies for completing proofs a little later, but for the present let's concentrate on what a proof is. Here are two steps to get our proof started:

1	(1)	$P \& \sim Q$	A
2	(2)	$(P \vee S) \rightarrow (Q \vee R)$	A
1	(3)	P	1 &E
1	(4)	$P \vee S$	3 $\vee$I

The first step (line 3) was to add P by applying &E to line 1. Next, we used $\vee$I to add $P \vee S$ as line 4. In each case, the assumption set includes only 1 (why?).

1	(1)	$P\& \sim Q$	A
2	(2)	$(P \vee S) \rightarrow (Q \vee R)$	A
1	(3)	P	1 &E
1	(4)	$P \vee S$	3 $\vee$I
1,2	(5)	$Q \vee R$	2,4 $\rightarrow$E
1	(6)	$\sim Q$	1 &E

Next, we added line 5 using $\rightarrow$E. Notice that the assumption set includes the assumption sets from lines 2 and 4. We **don't** put down 4 as part of the assumption set (line 4 is not an assumption); instead, we include the **assumption set of** line 4.

Line 6 is exactly like line 3 except that here we added the right conjunct of line 1, not the left one.

1	(1)	$P\& \sim Q$	A
2	(2)	$(P \vee S) \rightarrow (Q \vee R)$	A
1	(3)	P	1 &E
1	(4)	$P \vee S$	3 $\vee$I
1,2	(5)	$Q \vee R$	2,4 $\rightarrow$E
1	(6)	$\sim Q$	1 &E
1,2	(7)	R	5,6 $\vee$E
1,2	(8)	$P\&R$	3,7 &I

LIne 7 applies $\vee$E to lines 5 and 6. Notice that the sentence on line 6, $\sim Q$, is the denial of one of the disjuncts of line 5.

Line 8 applies &I to lines 3 and 7. Just to make a point clear, notice that this is the second time we applied a rule to line 3. You can apply rules to a line or lines as often as you need to, in a proof: they don't get used up once you've used them.

Line 8 is the conclusion of the argument. Its assumption set only includes the numbers of the premises of the argument (1 and 2). Therefore, this is a proof of the sequent in question.

Rules for Biconditionals: $\leftrightarrow$I and $\leftrightarrow$E

Double-Arrow-Elim ($\leftrightarrow$E)

double-arrow-elim	Given a biconditional sentence $(\phi \leftrightarrow \psi)$ (at line *m*), conclude either $(\phi \rightarrow \psi)$ or $(\psi \rightarrow \phi)$.

What this rule permits you to do is **get a conditional** having one of the constitutents of a biconditional as antecedent and the other as consequent.

.	.	.	.
7,8	(*m*)	$(\phi \leftrightarrow \psi)$	
.	.	.	
7,8	(*k*)	$(\phi \rightarrow \psi)$	m $\leftrightarrow$E

We could equally well have added $\psi \rightarrow \phi$.

The assumption set here is the same as the assumption set in front of the line containing the biconditional we used.

Double-Arrow-Intro ($\leftrightarrow$I)

double-arrow-intro	Given two conditionals having the forms $(\phi \to \psi)$ and $(\psi \to \phi)$ (at lines *m* and *n*), conclude a biconditional with ϕ on one side and ψ on the other.

What this rule permits you to do is **get a conditional** having one of the constitutents of a biconditional as antecedent and the other as consequent.

5	(*m*)	$(\phi \to \psi)$	
.	.	.	
7	(*n*)	$(\psi \to \phi)$	
.	.	.	
5,7	(*k*)	$(\phi \leftrightarrow \psi)$	$m, n \leftrightarrow$I

We could equally well have added $(\psi \leftrightarrow \phi)$.

The assumption set includes all the numbers in the assumption sets for the two conditionals used.

Assumption Rules: $\rightarrow$I and RAA

There are two remaining rules, $\rightarrow$I and RAA. These two rules introduce a new feature: each of them **discharges an assumption**. The critical point about these rules is that **an assumption number is dropped from the assumption set** on the line added by either of these rules.

The rule **assumption** . Although we have used it up to now for assuming the premises of an argument, the rule itself is broader in scope: you can assume anything you like at any time. Of course, if you use that assumption to get other lines, then its assumption number will be carried through to those lines. Consequently, you cannot get a proof of a sequent just by adding some extra premises to get the conclusion, since you'll have numbers in the assumption set that don't correspond to premises—.

—except under certain circumstances. One way to argue for a conditional conclusion is to **assume its antecedent** and then see if you can deduce its consequent. If you can, then what that shows is this: **if** you assume the antecedent, **then** you get the consequent. In other words, you have proved the conditional *If (the antecedent) then (the consequent)*.

Another strategy in argument is to assume **the negation of what you want to prove** and then try to get an inconsistent pair of sentences using that assumption. If you can, then what you've shown is this: *If (the assumption) were true, then something impossible would follow*. Since impossible things can't happen, the assumption must be false, and we can conclude its negation.

The rules $\rightarrow$I and RAA represent these lines of reasoning formally.

Arrow-Intro (→I)

arrow-intro Given an assumption (at line *m*) and a sentence (at line *n*), conclude the conditional having the assumption as its antecedent and the sentence as its consequent.

What this rule permits you to do is **get a conditional if you have already got its consequent and have assumed its antecedent**. The antecedent **must be an assumption**: it cannot be a line obtained using any other rule.

The assumption set here includes all the numbers on *n* **except for the line number of the assumption**. This last part of the rule is critical:

m	(*m*)	ϕ	
.	.	.	
1,3,*m*	(*n*)	ψ	
↑ ↑ ↑	.	.	
1,3	(*k*)	$\phi \rightarrow \psi$	n →I (*m*)
↑ ↑			

Getting the annotation and the assumption set right here is crucial. Notice that we only cite one line, the line *n* on which the **consequent** of the conditional is found. However, we also cite the line number of the **antecedent** after the rule, and in parentheses.

An example will show how this works.

Sequent: $S \rightarrow P, P \rightarrow Q \vdash S \rightarrow Q$

Proof:

1	(1)	$S \rightarrow P$	A
2	(2)	$P \rightarrow Q$	A
3	(3)	S	A
1,3	(4)	P	1,3 →E
1,2,3	(5)	Q	2,4 →E
1,2	(6)	$S \rightarrow Q$	5 →I (3)

Notice three things: (1) line 3 is an assumption; (2) the line number of line 3 does **not** appear in the assumption set of line 6; (3) in the annotation for line 6, the line number of line 3 is placed **after the name of the rule, in parentheses**.

Discharging assumptions

When we use $\rightarrow$I in this way to get a line that does not contain the number of an assumption in its assumption set, we say that we have **discharged** that assumption. This terminology is just a little misleading, since it implies that we can only do this one time with a given assumption. In fact, there's no such limitation. Here is an example:

Sequent:

$P \rightarrow (Q \& S), P \rightarrow R, R \rightarrow T, S \vee \sim Q, \vdash (P \rightarrow S) \& (P \rightarrow T)$

1	(1)	$P \rightarrow Q$	A
2	(2)	$P \rightarrow R$	A
3	(3)	$R \rightarrow T$	A
4	(4)	$S \vee \sim Q$	A
5	(5)	P	A
1,5	(6)	Q	1,5 $\rightarrow$E
2,5	(7)	R	2,5 $\rightarrow$E
1,4,5	(8)	S	4,6 $\vee$E
1,4	(9)	$P \rightarrow S$	8 $\rightarrow$I (5)
2,3,5	(10)	T	3,7 $\rightarrow$E
2,3	(11)	$P \rightarrow T$	10 $\rightarrow$I (5)
1,2,3,4	(12)	$(P \rightarrow S) \& (P \rightarrow T)$	9,11 &I

Reductio ad Absurdum (RAA)

Reductio ad absurdum	Given an assumption (at line *k*) and two sentences (at lines *m* and *n*) one of which is a negation of the other, conclude the negation of the assumption.

What this rule lets you do is conclude **the negation of an assumption** if you are able to get a **pair** of lines, one of which is the negation of the other.

Like $\rightarrow$I, this rule discharges an assumption: the assumption set for the line added includes all the numbers in the assumption sets of the pair of lines, **except for** the line number of the assumption.

.	.	.	.
k	(*k*)	ϕ	
.	.	.	
1,2,*k*	(*m*)	ψ	
.	.	.	
2,3	(*n*)	$\sim\psi$	
.	.	.	
1,2,3	(*x*)	$\sim\phi$	*m*,*n* RAA (*k*)

In this example, we've shown line *m* as depending on assumption *k* but line *n* not depending on it. There is actually no requirement that either of the lines containing the contradictory pair contain the assumption number of the assumption in its assumption set: both may, or one may, or neither may.

RAA corresponds to a familiar kind of argument: we prove that something is **not** so by assuming that it is and deducing an impossibility from that. The particular form of impossibility is a pair consisting of a sentence and its negation. Such a pair is often called a **contradiction**, and the pair of sentences are said to be **inconsistent**.

As with $\rightarrow$I, it is critical to get the assumption set right when using **RAA**.

Here's an example.

Sequent:

$$P \rightarrow Q, Q \rightarrow \sim R, R \vdash \sim P$$

1	(1)	$P \rightarrow Q$	A
2	(2)	$Q \rightarrow \sim R$	A
3	(3)	R	A
4	(4)	P	A
1,4	(5)	Q	1,4 $\rightarrow$E
1,2,4	(6)	$\sim R$	2,5 $\rightarrow$E
1,2,3	(8)	$\sim P$	3,6 RAA (4)

Primitive Rules of Inference

Assumption — **A**

n	(n)	Φ	A

Φ can be any wff whatever.

This space intentionally left void of any useful information

Ampersand-Elimination — **&E**

$i_1, ... i_x$	(m)	$(\Phi \& \Psi)$	...
$i_1, ... i_x$	(n)	Φ	m &E

Can also conclude Ψ.

Ampersand-Introduction — **&I**

$i_1, ... i_x$	(k)	Φ	...
$j_1, ... j_y$	(m)	Ψ	...
$i_1, ... i_x, j_1, ... j_y$	(n)	$(\Phi \& \Psi)$	k, m &I

Wedge-Elimination — **$\vee$E**

$i_1, ... i_x$	(k)	$(\Phi \vee \Psi)$	...
$j_1, ... j_x$	(m)	$\sim\Phi$	...
$i_1, ... i_x, j_1, ... j_y$	(n)	Ψ	k, m $\vee$E

If line m is $\sim\Psi$, conclude Φ.
Also if line $k = (\sim\Phi \vee \Psi)$, line $m = \Phi$, etc.

Wedge-Introduction — **$\vee$I**

$i_1, ... i_x$	(m)	Φ	...
$i_1, ... i_x$	(n)	$(\Phi \vee \Psi)$	m $\vee$I

Can also conclude $(\Psi \vee \Phi)$.

Double-Arrow-Elimination — **$\leftrightarrow$E**

$i_1, ... i_x$	(m)	$(\Phi \leftrightarrow \Psi)$	...
$i_1, ... i_x$	(n)	$(\Phi \rightarrow \Psi)$	m $\leftrightarrow$E

Can also conclude $(\Psi \rightarrow \Phi)$.

Double-Arrow-Introduction — **$\leftrightarrow$I**

$i_1, ... i_x$	(m)	$(\Phi \rightarrow \Psi)$	...
$j_1, ... j_y$	(n)	$(\Psi \rightarrow \Phi)$	...
$i_1, ... i_x, j_1, ... j_y$	(n)	$(\Phi \leftrightarrow \Psi)$	m, n $\leftrightarrow$I

Order does not matter: can also conclude $(\Psi \leftrightarrow \Phi)$.

Arrow-Elimination — **$\rightarrow$E**

$i_1, ... i_x$	(k)	$(\Phi \rightarrow \Psi)$	...
$j_1, ... j_x$	(m)	Φ	...
$i_1, ... i_x, j_1, ... j_y$	(n)	Ψ	k, m $\rightarrow$E

You CANNOT conclude Φ if line $m = \Psi$.

Arrow-Introduction — **$\rightarrow$I**

k	(k)	Φ	A
$i_1, ... i_x, k$	(m)	Ψ	...
$i_1, ... i_x$	(n)	$(\Phi \rightarrow \Psi)$	m $\rightarrow$I(k)

Line k MUST be an assumption.
k is dropped from the assumption set for line n.
k does not have to be in the assumption set for line m, but in practice it usually is.

In all rules, there may be other lines between the lines indicated.

Assumption sets are indicated by strings like $i_1, ... i_x$ for convenience. In an actual proof, this will be a string of actual numbers, e.g. $1, 2, 4, 7$. $i_1, ... i_x, j_1, ... j_y$ means "the string of numbers that includes everything in $i_1, ... i_x$ and $j_1, ... j_y$"

Reductio ad Absurdum — **RAA**

k	(k)	Φ	A
$i_1, ... i_x, k$	(l)	Ψ	...
$j_1, ... j_y, k$	(m)	$\sim\Psi$	...
$i_1, ... i_x, j_1, ... j_y$	(n)	$\sim\Phi$	l, m RAA(k)

Line k MUST be an assumption.
Can also assume $\sim\Phi$ and conclude Φ.
k is dropped from the assumption set for line n.
k does not have to be in the assumption set for line l or line m, but in practice it usually is.

Strategies for Proofs

Working Backwards

Take the conclusion as your goal and then use an appropriate strategy to see what you need in order to get there. This will give you a new goal. Apply a strategy to that goal in turn, and continue until what you need is something that you already have. You can then reverse direction and complete the proof.

Taking things apart (Elimination rules)

Use one of these strategies when you see the wff that you are trying to get is already present as a constituent of a line that you have already added. They tell you how to get it out of that line and what else you need in order to do that (if it's possible).

If your goal is ...	and you have ...	then the rule to use is ...	and you also need ...
Φ	$\Phi \& \Psi$ or $\Psi \& \Phi$	&E	nothing else
Φ	$\Phi \vee \Psi$	$\vee$E	$\sim\Psi$
$\Phi \rightarrow \Psi$	$\Phi \leftrightarrow \Psi$ or $\Psi \leftrightarrow \Phi$	$\leftrightarrow$E	nothing else
Φ	$\Psi \rightarrow \Phi$	$\rightarrow$E	Ψ
Φ	$\Phi \rightarrow \Psi$	another strategy	help

Building things up (Introduction rules)

Try one of these strategies if the wff that you are trying to get is not already present as a constituent of another line you have already added. They tell you how to build it up and what else you need in order to do that.

If your goal is ...	then the rule to use is ...	and you need ...	and ...
$\Phi \& \Psi$	&I	Φ	Ψ
$\Phi \vee \Psi$	$\vee$I	Φ or Ψ	(nothing else)
$\Phi \leftrightarrow \Psi$	$\leftrightarrow$I	$\Phi \rightarrow \Psi$	$\Psi \rightarrow \Phi$
$\Phi \rightarrow \Psi$	$\rightarrow$I	to assume Φ	to deduce Ψ
$\sim\Phi$	RAA	to assume Φ	to deduce Ψ and $\sim\Psi$

Exercise 1.4.2

S1. P v ~R, ~R → S, ~P |- S

Our goal is to get the atomic statement S. If we look over the premises, we find S as the right side of the conditional ~R → S. If we had the left side of this conditional, ~R, then we could get S by →E. So, the next step is to get ~R. In fact, ~R occurs in the premise P v ~R. Since this is a disjunction, we could get ~R by vE if we had a denial of the other disjunct, P. But we do have this as the third premise. So, reversing this, we first get ~R by vE, then get S by →E:

1	(1) P v ~R	A
2	(2) ~R → S	A
3	(3) ~P	A
1,3	(4) ~R	1,3 vE
1,2,3	(5) S	2,4 →E

S2. P v ~R, ~R → S, ~P |- S & ~R

This has the same premises as the previous examle, but now our goal is a conjunction: S & ~R. We can get a conjunction by &I if we can get both conjunctions. The previous proof shows how to get S, so we can use that again here. But in addition, we got ~R as one of the lines of that previous proof. So, we can get what we want by just adding one further line to the last proof:

1	(1) P v ~R	A
2	(2) ~R → S	A
3	(3) ~P	A
1,3	(4) ~R	1,3 vE
1,2,3	(5) S	2,4 →E
1,2,3	(6) S & ~R	4,5 &I

S3. P → ~Q, ~Q v R → ~S, P & T |- ~S

Since the conclusion we want, ~S, appears as the consequent of one of the premises, ~Q v R → ~S, we can get it if we can get the antecedent of that premise. The antecedent ~Q v R is a disjunction, and we can get a disjunction by vI if we can get either disjunct. R doesn't occur anywhere else in the premises, but ~Q is the consequent of P →~Q; so, if we can get P, we can get what we want. P is one of the conjuncts of P & T, and we can get it gy &E. So, reversing this reasoning:

1	(1) P → ~Q	A
2	(2) ~Q v R → ~S	A
3	(3) P & T	A
3	(4) P	3 &E
1,3	(5) ~Q	1,4 →E
1,3	(6) ~Q v R	5 vI
1,2,3	(7) ~S	2,6 →E

S4. P & (Q & R), P & R → ~S, S v T |- T

We want to get T, and we have it as one disjunct of S v T, so we should try to get a denial of the other disjunct. We have that (~S) as the consequent of another premise (P & R → ~S), so we need to get the antecedent of this, which is P & R. For that, we will need to get both P and R. We have P as one conjunct of P & (Q & R), so we can get it by &E; R is one conjunct of the *other* conjunct of P & (Q & R), so we can get it with two applications of &E.

1	(1) P & (Q & R)	
2	(2) P & R → ~S	
3	(3) S v T	
1	(4) P	1 &E
1	(5) Q &R	1 &E
1	(6) R	5 &E
1	(7) P & R	4,6 &I
1,2	(8) ~S	2,7 →E
1,2,3	(9) T	3,8 vE

S5. P → Q, P → R, P |- Q & R

To get Q & R, we need to get Q, get R, and then use &I. Both Q and R appear as consequents of conditionals that have P as antecedent, and we already have P. So:

1	(1) P → Q	A
2	(2) P → R	A
3	(3) P	A
1,3	(4) Q	1,3 >E
2,3	(5) R	2,3 >E
1,2,3	(6) Q & R	4,5 &I

S6. P, Q v R, ~R v S, ~Q |- P & S

To get P & S, we need to get P and S and use &I. In this case, we already have P, so all we need to work for is S. S is a disjunct of ~R v S, so we need to get a denial of ~R (for instance, R). Since R appears as a disjunct of Q v R, what we need is a denial of Q: but we have that.

1	(1) P	A
2	(2) Q v R	A
3	(3) ~R v S	A
4	(4) ~Q	A
2,4	(5) R	2,4 vE
2,3,4	(6) S	3,5 vE
1,2,3,4	(7) P &S	1,6 &I

S7. ~P, R v ~P ↔ P v Q, |- Q

This proof involves the rules for ↔. Basically, ↔ lets us get a conditional from a biconditional: it can have either the left side of the biconditional as antecendent and the right side as consequent, or the reverse. Here, we need Q, and we have Q only as a part of one of the parts of a biconditional (P v Q). If we could get that out, then we could get Q using vE if we had ~P. But we're in luck, since we do have ~P. Now all we need is a way to get P v Q out of the biconditional. The only way to do that would be to use ↔ to get the conditional with P v Q as consequent and then try to get its antecedent. The conditional in question is R v ~P → P v Q. How do we get its antecedent, R v ~P? We could do that if we had either one of its disjuncts. Here, we can use ~P again and get R v ~P using vI. The proof goes as follows:

1	(1) ~P	A
2	(2) R v ~P ↔ P v Q	A
1	(3) R v ~P	1 vI
2	(4) R v ~P → P v Q	2 ↔E
1,2	(5) P v Q	3,4 →E
1,2	(6) Q	5 vE

S8. (P ↔ Q) → R, P → Q, Q → P |- R

We have R as the consequent of a conditional, so in order to get it we need the antecedent, P ↔ Q. Since that's a biconditional, we need the two conditionals P → Q, Q → P in order to get it. But those are just our second and third premises, so we're done:

1	(1) (P ↔ Q) → R	A
2	(2) P → Q	A
3	(3) Q → P	A
2,3	(4) P ↔ Q	2,3 ↔I
1,2,3	(5) R	1,4 →E

S9. ~P → Q & R, ~P v S → ~T, U & ~P |- (U & R) & ~T

We've been asked to get a conjunction that has a conjunction as one of its conjuncts. To prove this, we need to get all three of the parts: the right conjunct ~T and the two conjuncts of the left conjunct, U and R. So, we have three tasks. We can get U right away from the third premise by &E. We could get ~T from the second premise if we had ~P v S, and we could get that if we had either ~P or S, using vI. But we can get ~P from the third premise as well, so that's solved. All that remains is getting R. We could get it by &E from Q & R, which appears as the consequent of the first premise, if we had the antecedent of that premise, which is ~P. But we already needed ~P for the previous step, so that's been solved. Here's all of this put together:

1	(1)	~P → Q & R	A
2	(2)	~P v S → ~T	A
3	(3)	U & ~P	A
3	(4)	U	3 &E
3	(5)	~P	3 &E
1,3	(6)	Q & R	1,5 →E
1,3	(7)	R	6 &E
3	(8)	~P v S	5 vI
2,3	(9)	~T	2,8 →E
1,3	(10)	U & R	4,7 &I
1,2,3	(11)	(U & R) & ~T	9,10 &I

S10. (Q v R) & ~S → T, Q & U, ~S v ~U |- T &U

We need to get T and get U, then use &I. Getting U is easy: use &E on Q & U. All we need to work on then is getting T. We have T as the consequent of a conditional, (Q v R) & ~S → T, so we need to get that conditional's antecedent, (Q v R) & ~S. Since this is a conjunction, we need to get both conuuncts. ~S appears in ~S v ~U, so we could get it if we had a denial of ~U, and we do (we already saw how to get U). Q v R is a disjunction, and getting it is thus a matter of getting either of its disjuncts and using vI. But here again, we already have U. So:

1	(1)	(Q v R) & ~S → T	A
2	(2)	Q & U	A
3	(3)	~S v ~U	A
2	(4)	Q	2 &E
2	(5)	U	2 &E
2	(6)	Q v R	4 vI
2,3	(7)	~S	3,5 vE
2,3	(8)	(Q v R) & ~S	6,7 &I
1,2,3	(9)	T	1,8 →E
1,2,3	(10)	T & U	5,9 &I

To the Syllabus

Back to Exercise 1.4.1

On to Exercise 1.5.1

Last modified: Mon Feb 4 15:41:48 CST 2002

PROOFS WITHOUT TEARS

A Student's Handy Guide to Proving Sequents in the system of Allen/Hand, *Logic Primer*

The rules of any logical system define *what a proof is*. They do not, however, tell you *how to build one*. Here are some procedures that will help you with the latter.

What are you trying to prove?

A proof is a trip from a given destination from a given starting point. The destination is the *conclusion*; the starting point is the set of *premisses*. Your task is to discover *a* route--not *the* route, since there are many different ones.

A general strategy for getting to where you want to go from where you already are is to work backwards from your destination. If you need to get to Galveston, and you know how to get to Galveston from Houston, then all you have to figure out is how to get from where you are to Houston. If you know how to get from Waller to Houston (and Houston to Galveston), then all you have to do is figure out how to get from where you are to Waller. At each step of this process, you reduce your problem of how to get from where you are to Galveston to the problem of how to get from where you are to somewhere else--hopefully, somewhere closer to where you are. If your strategy works, eventually you reduce your problem to one you already know how to solve: I can get from College Station to Galveston if I can get to Houston, and I can get from College Station to Houston if I can get to Waller, and I can get from College Station to Waller if I can get to Hempstead, and I can get from College Station to Hempstead if I can get to Navasota--but, behold, I know how to get from College Station to Navasota. So, my problem is solved.

This kind of strategy works well in trying to discover proofs. Start with where you want to end up, that is, with the conclusion, and try to figure out how you could get there from *something*; keep working backwards from the intermediate steps until you find that what you need is something you know how to get.

On What Everything Is

A very, very important point to remember is that, in our formal system, there are just exactly six kinds of thing, and everything is exactly one of those and not two of those. Every wff is:

1. **Atomic** (a letter all by itself, or
2. a **Negation** (main connective ~)
3. a **Conjunction** (main connective &), or
4. a **Disjunction** (main connective v), or
5. a **Conditional** (main connective →), or
6. a **Biconditional** (main connective ↔)

This is the first thing to notice about any wff you're working with, since it determines both the strategies for *proving* it and the strategies for *using* it in proving other things.

Some important questions: the Direct Approach

There are several fundamental questions that will help in constructing a strategy. We'll call the strategy they rest on 'the direct approach'.

1. Do I already have it as part of something else?
   The wff you need may already be present as a constituent part of something you have.
2. If I do, how do I take that thing apart and get out what I need?
   This depends on what kind of wff it is. In general, the Elimination rules are useful here (see below).
3. If I don't, then how do I build what I need?
   This depends on what kind of wff it is. In general, the Introduction rules are useful here.

More about Question 2: how to take things apart

If what you need is	**then the rule to use is...**	**and what else you need is...**
The P in PvQ	vE	~Q
The Q in PvQ	vE	~P
The P in P&Q	&E	nothing
The Q in P&Q	&E	nothing
The Q in P→Q	→E	P
P in P→Q	(You just can't get there from here)	Another strategy

More about Question 3: How to Build New Things

If what you need to build is ...	**Then the rule to use is ...**	**and what you need is ...**	**... and**
P&Q	&I	P	Q
PvQ	vI	either P or Q	(nothing)
P→Q	→I	...to assume P	...and deduce Q
~P	RAA	...to assume P	...and deduce both X and ~X (for any X you can)
P↔Q	↔I	P→Q	O→P

The Indirect Approach

Sometimes, the direct approach doesn't work. In that case (or at other times), you can always resort to 'the indirect approach': assume a denial of what you want to get, then try to get a contradictory pair so that you can use RAA.

Some exercises from Allen/Hand, Section 1.5.1

Here is an example of working backwards, in detail.

S12: P→Q, ~Q |- ~P

Since the Middle Ages, this rule has been called 'Modus Tollendo Tollens,' or just 'Modus Tollens', or just 'MT'

Here, what we want is not a constituent of any premiss. So, we need to find a way to build it. ~P is a negation, so we need to introduce a '~'. The rule to do that with is RAA:

1 (1) P→Q A

2	(2)	~Q	A
	.	.	
1,2		~P	?, ? RAA (?)

Next, we need to figure out what assumption to make (and discharge), and also how to deduce a statement and its denial. A good strategy is just to assume the denial of what we want:

1	(1)	P→Q	A
2	(2)	~Q	A
3	(3)	P	A
	.	.	
1,2	( )	~P	?, ? RAA (3)

Now, all we need is to get a pair of contradictories. Since we already have ~Q, one way to do this would be to get Q:

1	(1)	P→Q	A
2	(2)	~Q	A
3	(3)	P	A
	.	.	
	( )	Q	?
1,2	( )	~P	?, ? RAA (3)

In fact, we do already have Q as part of premise 1. So, the next question is: how do we get it? The rule →E lets us get the consequent of a conditional if we have its antecedent, which in this case is P. So, we could get Q if we could get P. Here, we're in luck: we *do* have P:

1	(1)	P→Q	A
2	(2)	~Q	A
3	(3)	P	A
1,3	(4)	Q	1,3 →E
1,2	(5)	~P	2,4 RAA (3)

S13-15: These can be done with exactly the same strategy as S2

S13-S15 are variant forms of Modus Tollens

S16: P→Q, Q→R |- P→R

This rule has the name 'hypothetical syllogism', or 'HS'

1	(1)	P→Q	A
2	(2)	Q→R	A
	.	.	
1,2	( )	P→R	?

What we need to get is a conditional, and it doesn't already appear in the premises. So, since we need to introduce an arrow, let's try →I:

1	(1)	P→Q	A
2	(2)	Q→R	A
3	(3)	P	A
	.	.	
	( )	R	?
1,2	()	P→R	? →I (3)

That is: if we can get R after assuming P (line 3), then we can get what we want with →I and discharge the assumption. Our problem, then, reduces to this: how do we get R? R is found in premise (2) as the consequent of a conditional, so we could get it if we had the antecedent, Q. Once again, then, we've reduced our problem to another:

1	(1)	P→Q	A
2	(2)	Q→R	A
3	(3)	P	A
	.	.	
	(*n*-1)	Q	?
	(*n*)	R	2, (*n*-1) →E
1,2	(*n*+1)	P→R	(*n*) →I (3)

So now, we need to get Q. We have it on the right side of (1), so we could get it by →E if we had the left side. But we do: it's (3). So, we're done:

1	(1)	P→Q	A
2	(2)	Q→R	A
3	(3)	P	A
1,3	(4)	Q	1,3 →E
1,2,3	(5)	R	2,4 →E
1,2	(6)	P→R	5 →I (3)

S17: P |- Q→P

What we need to get is a conditional, and all that we have is an atomic sentence. So, we'll have to build the conclusion, and to do that we will need →I:

1	(1)	P	A
2	(2)	Q	A
	.	.	
	(*n*)	P	?
1	(*n*+1)	Q→P	*n* →I (2)

All that remains, then, is to figure out how to get P, having assumed Q. But we already *have* P: it's the first premise. So, we're done:

1	(1)	P	A
2	(2)	Q	A
1	(3)	P	A
1	(4)	Q→P	3 →I (2)

S18: ~P |- P→Q

This rule is known as 'False Antecedent' (FA). A nice medieval Latin name is 'ex falso quodlibet'

The conclusion is a conditional, and it's not already present as part of any premise, so we should probably try to use →I:

1 (1) ~P A
. .
. .
1 () P→Q ? →I

Of course, to use →I, we'll have to assume the antecedent of the conditional we want (that is, P), then try to deduce its consequent (that is, Q). So, our proof will need to have this overall structure:

1 (1) ~P A
2 (2) P A
. .
() Q ?
1 () P→Q ? →I (2)

So, how can we get Q? There's an important point of strategy we can notice here. Q does not occur at all in the premises. Thus, we have to find a way to introduce a *new atomic sentence* in the proof. There are only two rules that allow us to get a conclusion containing a statement letter not found among the premises: vI and RAA. We're not trying to get a disjunction, so vI is not likely to be of much use here. Let's try RAA. The strategy would be to assume a denial of Q, then try to get a contradictory pair:

1 (1) ~P A
2 (2) P A
3 (3) ~Q A
. .
() Q ? RAA (3)
1 () P→Q ? →I (2)

But wait! we've already *got* a sentence and its denial (lines (1) and (2)). So, we can fill in the details for RAA and for the rest of the proof:

1 (1) ~P A
2 (2) P A
3 (3) ~Q A
1,2 (4) Q 1,2 RAA (3)
1 (5) P→Q 4 →I (2)

S22: PvQ -||- ~P→Q

Among the names of this is 'Wedge-Arrow', or 'v→'.

This is a double turnstile, so we need to prove two sequents: PvQ |- ~P→Q and ~P→Q |- PvQ. The first is easy: since we need to build a conditional, we use →I.

1 (1) PvQ A

2 (2) ~P A

1,2 (3) Q 1,2 vE

1 (4) ~P→Q 3 →I (2)

The second is a little harder. We might think of using vI, provided that we could get either P or Q. But how will we get P or Q? We could get Q if we could get ~P; so, if we can get ~P, we will have a solution. Since ~P is a negation, the strategy to try is RAA, with P as the assumption:

1 (1) ~P→Q A

2 (2) ~P A

1,2 (3) Q 1,2 →E

1,2 (4) PvQ 3 vI

Unfortunately, this isn't yet a proof, since we have too many assumptions in the assumption set of line (4). We need to get rid of the '2', and there's no obvious way to do that. However, we have learned something from this blind alley: if we *did* assume ~P, then we *would* get PvQ. We can use this for a strategy in the indirect approach: if we assume a denial of PvQ and also assume ~P, then we can get PvQ, in which case we will have both a sentence and its denial:

1 (1) ~P→Q A

2 (2) ~(PvQ) A

3 (3) ~P A

1,3 (4) Q 1,3 →E

1,3 (5) PvQ 4 vI

We now have contradictory sentences at lines (2) and (5), so we can discharge any assumption we've made with RAA. This would let us get rid of the assumption of ~P with RAA and conclude PvQ by vI:

1 (1) ~P→Q A

2 (2) ~(PvQ) A

3 (3) ~P A

1,3 (4) Q 1,3 →E

1,3 (5) PvQ 4 vI

1,2 (6) P 2,5 RAA (3)

1,2 (7) PvQ 6 vI

This still isn't what we want, unfortunately, since line (7) still has too many assumptions in its assumption set. We need to get rid of assumption (2). But wait! As a matter of fact, assumption (2) just happens to be a denial of line (7). So, we have, at line (7), both a sentence and its denial (lines (7) and (2)), and therefore we can discharge any assumption, including assumption (2):

1 (1) ~P→Q A

2 (2) ~(PvQ) A

3 (3) ~P A

1,3 (4) Q 1,3 →E

1,3 (5) PvQ 4 vI

1,2 (6) P 2,5 RAA (3)

1,2 (7) PvQ 6 vI

1 (8) PvQ 2,7 RAA (2)

This may appear to be sneaky, but it is in fact a common strategy: if we can deduce the negation of an assumption using that assumption, then we can conclude the negation of that assumption.

S25: PvQ, P→R, Q→R |- R

The venerable name for this is 'Simple Dilemma'. Please notice how many times 'm' and 'n' respectively do and do not occur in the word 'dilemma'.

1 (1) PvQ A
2 (2) P→R A
3 (3) Q→R A
. .
( ) R ?

We need to get R. Looking over the premises, we see that we could get R if we could get P, and we could also get R if we could get Q (each time using →E). Which shall we try? Let's start by trying P:

1	(1)	PvQ	A
2	(2)	P→R	A
3	(3)	Q→R	A
	.	.	
	(*n*-1)	P	?
1,2,3	(*n*)	R	2, *n*-1 →E

So, the problem is how to get P. There's no rule that lets us get the left side of a conditional, but we could get it from line (1) if we had ~Q, using vE:

1	(1)	PvQ	A
2	(2)	P→R	A
3	(3)	Q→R	A
	.	.	
	(*n*-2)	~Q	?
	(*n*-1)	P	1,*n*-2 vE
1,2,3	(*n*)	R	2, *n*-1 →E

So, how could we get ~Q? It's not part of any premise, so the best strategy would be to try RAA. We would then assume Q and try to deduce a pair of contradictory formulas. Before we start down that path, however, let's look where it's leading us. If we look over the premises, the only thing we can get from them in one step with the additional assumption Q is R. There are no negations anywhere in the premises, so we might suspect that it's going to be hard to come up with our contradictory pair. We will probably have to make one or more assumptions in addition to Q.

At this point, we're approaching the condition of aimless floundering. Aimless floundering is, in fact, a strategy of sorts, rather like guessing on a multiple-choice quiz or buying lottery tickets: sometimes, those strategies win. However, when you find that you don't really know what to do next, that's usually a sign that it's time to give up on the direct approach and try the indirect approach. That's particularly apt when the conclusion you want is atomic, as it is in this case. So, let's start over:

1	(1)	PvQ	A
2	(2)	P→R	A
3	(3)	Q→R	A
4	(4)	~R	A

	.	.	
1,2,3	( )	R	?, ? RAA (4)

So, we need to get a contradictory pair. Notice right away that if we assume either P or Q, we'll get R. This is promising, so let's try it:

1	(1)	PvQ	A
2	(2)	P→R	A
3	(3)	Q→R	A
4	(4)	~R	A
5	(5)	P	A
2,5	(6)	R	2,5 →E
	.	.	
1,2,3	( )	R	?, ? RAA (4)

Now we have a contradiction (lines (4 and 6). But which assumption do we want to discharge? If we discharge (4), we won't be done, because we'll still have (5) in the assumption set. However, we can discharge (5):

1	(1)	PvQ	A
2	(2)	P→R	A
3	(3)	Q→R	A
4	(4)	~R	A
5	(5)	P	A
2,5	(6)	R	2,5 →E
2,4	(7)	~P	6,4 RAA (5)
	.	.	
1,2,3	( )	R	?, ? RAA (4)

This takes us a little farther. With ~P, we can get Q (by vE, from (1)), and with Q we can get R (by →E, from (3)). But R is a denial of ~R, so we have what we need.

1	(1)	PvQ	A
2	(2)	P→R	A
3	(3)	Q→R	A
4	(4)	~R	A
5	(5)	P	A
2,5	(6)	R	2,5 →E
2,4	(7)	~P	6,4 RAA (5)
1,2,4	(8)	Q	1,7 vE
1,2,3,4	(9)	R	3,8 →E
1,2,3	(10)	R	4,9 RAA (4)

Notice that we couldn't stop at (9), since there's something besides the premises in the assumption set. However, using RAA, we can get rid of that assumption. Line (10), remember, is the denial of *line (4)*: it just *happens* to be the same thing as line (9)

S28: ~(PvQ) -||- ~P & ~Q

This is one of the rules known as 'DeMorgan's Law' (after 19th-century logician Augustus DeMorgan).

Left to right: we need to build a conjunction, so we'll use &I. To do that, we need to get *both* conjuncts, that is, ~P and ~Q. those are both negations, so we will use RAA for each.

1	(1)	~(PvQ)	A
2	(2)	P	A
2	(3)	PvQ	2 vI
1	(4)	~P	1,3 RAA (2)
5	(5)	Q	A
5	(6)	PvQ	5 vI
1	(7)	~Q	1,6 RAA (5)
1	(8)	~P&~Q	4,7 &I

Right to left: we need to build a negation, so it's best to try RAA at once. This turns out to be easy:

1	(1)	~P&~Q	A
2	(2)	PvQ	A
1	(3)	~P	1 &E
1,2	(4)	Q	2,3 vE
1	(5)	~Q	1 &E
1	(6)	~(PvQ)	4,5 RAA(2)

S39: P→Q -||- ~Q→~P

For fairly obvious reasons, this is known as 'Transposition'.

For both directions: usual strategy for building a condition, that is, assume the antecedent, try to get the consequent, then use →I. In each case, the way to get the consequent turns out to be RAA:

1	(1)	P→Q	A
2	(2)	~Q	A
3	(3)	P	A
1,3	(4)	Q	1,3 →E
1,2	(5)	~P	2,4 RAA (3)
1	(6)	~Q→~P	5 →I (2)
1	(1)	~Q→~P	A
2	(2)	P	A
3	(3)	~Q	A
1,3	(4)	~P	1,3 →E
1,2	(5)	Q	2,4 RAA (3)
1	(6)	P→Q	5 →I (2)

You might notice that these two proofs are *very* similar.

S44: P→(Q→R) -||- (P&Q) → R

This is called 'exportation' going left to right and 'importation' right to left. I replaced a redundant pair of parentheses for

clarity.

Right to left: we need to build a conditional, so we assume the antecedent. To get R from the premise, we need to get Q→R out first. Again, this is straightforward:

1	(1)	P→(Q→R)	A
2	(2)	P&Q	A
2	(3)	P	2 &E
1,2	(4)	Q→R	1,3 →E
2	(5)	Q	2 &E
1,2	(6)	R	4,5 →E
1	(7)	(P&Q)→R	→I (2)

Right to left: a little more subtle. We need to build a conditional, but one side of it is a conditional. What we need to do, then, is assume P and try to get Q→R. Since that's also a conditional, we then assume Q and try to get R. Then, we discharge the two assumptions in reverse order:

1	(1)	(P&Q)→R	A
2	(2)	P	A
3	(3)	Q	A
2,3	(4)	P&Q	2,3 &I
1,2,3	(5)	R	1,4 →E
1,2	(6)	Q→R	5 →I (3)
1	(7)	P→(Q→R)	6→I (2)

To keep this file from getting any longer, examples will be continued here. If there's nothing there, try again later.

PROOFS WITH EVEN FEWER TEARS

Being the Continuation of a Student's Handy Guide to Proving Sequents in the system of Allen/Hand, *Logic Primer*

Some more exercises from Allen/Hand, Section 1.5.1

S29. ~(P&Q) -||- ~Pv~Q

Left to right: No point repeating the argument to show that you can't get this one by vI, even though that *is* the first strategy to think of with disjunctions. (In this case, it just won't work). Instead, we have to resort to the indirect approach, which turns out not to be as bad as you might expect. Assume ~(~Pv~Q), then look for a contradictory pair. One nice contradictory pair to look for would be P&Q and ~(P&Q), since we already have ~(P&Q). In order to get P&Q, we need to get P and to get Q. The proof below does each of those indirectly: assume ~P (which at once gives you ~Pv~Q, the denial of (2)), then discharge with RAA; then do the analogous thing with ~Q. This gives P&Q, which is the denial of the premise, and so we can use RAA to discharge the assumption at (2).

1	(1)	~(P&Q)	A
2	(2)	~(~Pv~Q)	A
3	(3)	~P	A
3	(4)	~Pv~Q	3 vI
2	(5)	P	2,4 RAA (3)
6	(6)	~Q	A
6	(7)	~Pv~Q	6 vI
2	(8)	Q	2,7 RAA (6)
2	(9)	P&Q	5,8 &I
1	(10)	~Pv~Q	1,9 RAA (2)

Moral: it's often a good idea to try the indirect approach when proving disjunctions.

Right to left:

1	(1)	~Pv~Q	A
2	(2)	P&Q	A
2	(3)	P	2 &E
2	(4)	Q	2 &E
1,2	(5)	~Q	1,3 vE
1	(6)	~(P&Q)	4,5 RAA (2)

S27. P -> Q, ~P -> Q |- Q

Your text names this 'Special Dilemma', but another medieval name for it was 'Consequentia Mirabilis': 'the amazing consequence.'

This requires a little subtlety. To get Q out of (1), we'd need P, and there's no likely way to get it. So, best to try the indirect approach: we assume ~Q, then try to get a contradiction. Here is where it becomes subtle.

There are two likely contradictory pairs we could aim for: Q and ~Q or P and ~P. Since we've assumed ~Q, it might be easiest to try to get Q. We could get Q if we had *either* P (from (1)) *or* ~P (from (2)). So, we could try to get one of those. I will choose ~P and try to get it with the indirect approach: assume P and look for a contradictory pair. In fact, we can get Q right away, giving us a contradictory pair with (3); ~P then gives us Q, so we have a contradictory pair after discharging the assumption of P, and we can then discharge the first assumption (~Q):

1	(1) P->Q	A (premise)
2	(2) ~P->Q	A (premise)
3	(3) ~Q	A
4	(4) P	A
1,4	(5) Q	1,4 ->E
1,3	(6) ~P	3,5 RAA (4)
1,2,3	(7) Q	2,6 ->E
1,2	(8) Q	3,7 RAA (3)

Notice that we can't stop at line (7), since its assumption set still includes the undischarged assumption (3). However, since we have contradictory lines (namely, it and that very assumption), we can discharge (3) with RAA.

S24. P -> Q -||- ~(P & ~Q)

Left to right: use the indirect approach. After you assume P&~Q, it's relatively quick to get both Q and ~Q, then use RAA. Right to left: what you want is a conditional, so use ->I (assume P and try to get Q). There's no rules for getting at the parts of negated conjunctions, so assume ~Q and look for a contradictory pair (the easiest one to get is ~(P&~Q) and P&~Q).

Left to right:

1	(1) P->Q	A (premise)
2	(2) P&~Q	A
2	(3) P	2 &E
1,2	(4) Q	1,3 ->E
2	(5) ~Q	2 &E
1	(6) ~(P&~Q)	4,5 RAA (2)

Right to left:

1	(1) ~(P&~Q)	A (premise)
2	(2) P	A
3	(3) ~Q	A
2,3	(4) P&~Q	2,3 &I
1.2	(5) Q	1,4 RAA (3)
1	(6) P->Q	5 ->I (2)

S36. P & Q -||- Q & P

This is easy, but it's important to realize that it *does* require proof. Here's the left-to-right part:

1	(1) P&Q	A (premise)
1	(2) P	1 &E

1	(3)	Q	1 &E
1	(4)	Q&P	2,3 &I

Right-to-left bears an extraordinary similarity to this.

S37. P v Q -||- Q v P

Compare this with the previous proof. It's not quite as easy. The only choice here is the indirect approach. After assuming ~(QvP), we need to get a contradictory pair. We can aim for the denial of that very assumption, that is, the thing we want to prove: QvP. To get that, we'd need either P or Q. However, we can get either one of those from (1) if we have the denial of the other. So, we assume one of them (in this case, I chose ~P):

1	(1)	PvQ	A (premise)
2	(2)	~(QvP)	A
3	(3)	~P	A
1,3	(4)	Q	1,3 vE
1,3	(5)	QvP	4 vI
1,2	(6)	P	2,5 RAA (3)
1,2	(7)	QvP	6 vI
1	(8)	QvP	2,7 RAA (2)

S51. ~(P<->Q) -||- P<->~Q

Strategy here is important. We'll begin with the left-to-right proof. Since what we want to conclude is a biconditional, we will need to use <->I after first deducing both P->~Q and ~Q->P. We also know that we need to get each of these with "1" (the number of our first assumption) as the entire assumption set, since that's what we want on the last line. So, we know this much:

1	(1)	~(P<->Q)	A (premise)
.	.	.	.
1	(m)	P->~Q	
.	.	.	.
1	(n)	~Q->P	
1	( )	P<->~Q	m,n <->I

Since each of the things we need to deduce is a conditional, the strategy we should use in each case is to assume the antecedent and try to get the consequent, then use ;->I. We can therefore fill in a little more of our proof:

1	(1)	~(P<->Q)	A (premise)
2	(2)	P	A
.	.	.	.
	(m-1)	~Q	
1	(m)	P->~Q	->I (2)
m+1	(m+1)	~Q	A
.	.	.	.
	(n-1)	P	
1	(n)	~Q->P	->I (m+1)

1	(n+1)	P<->~Q	m,n <->I

Now all we need to do is fill in the proof between (2) and (m-1) and the proof between (m+1) and (n-1). The first of these is going to require a use of RAA, since we need to get ~Q and don't currently have it as part of anything. So, we will make yet another assumption, this time of the denial of ~Q (i.e. Q), and try to get a contradiction. So, our proof will look like this:

1	(1)	~(P<->Q)	A (premise)
2	(2)	P	A
3	(3)	Q	A
.	.	.	.
	(m-1)	~Q	?, ?, RAA(3)
1	(m)	P->~Q	->I (2)
m+1	(m+1)	~Q	A
.	.	.	.
	(n-1)	P	
1	(n)	~Q->P	->I (m+1)
1	(n+1)	P<->~Q	m,n <->I

How will we then get a contradiction? At this point, we need to think about the first premise, which we haven't made any use of at all. This is a negation, so if we could manage to get P<->Q, we would have a contradiction. To get that, in turn, we would need to get P->Q and Q->P and use <->I:

1	(1)	~(P<->Q)	A (premise)
2	(2)	P	A
3	(3)	Q	A
.	.	.	.
	(m-4)	P->Q	
	(m-3)	Q->P	
	(m-2)	P<->Q	m-3, m-4 <->I
	(m-1)	~Q	1,m-2 RAA(3)
1	(m)	P->~Q	->I (2)
m+1	(m+1)	~Q	A
.	.	.	.
	(n-1)	P	
1	(n)	~Q->P	->I (m+1)
1	(n+1)	P<->~Q	m,n <->I

So, all that remains is to get lines (m-4) and (m-3). These are conditionals, so in each case we will assume the antecedent and try to get the consequent. But notice that we have, in each case, *already* assumed the antecedent, and we've *already got* the consequent (as an assumption, in each case. So, we can do this:

1	(1)	~(P<->Q)	A (premise)
2	(2)	P	A
3	(3)	Q	A
3	(4)	P->Q	3 ->I (2)
2	(5)	Q->P	2 ->I (3)
2,3	(6)	P<->Q	4, 5 <->I
1,2	(7)	~Q	1,6 RAA(3)

1	(8)	P->~Q	->I (2)
9	(9)	~Q	A
.	.	.	.
	(n-1)	P	
1	(n)	~Q->P	->I (m+1)
1	(n+1)	P<->~Q	m,n <->I

Now we're halfway done. All we have to do is the reverse. Sort of. Let's try to use the same strategy as before. We'll try to get P at line (n-1) by assuming ~P and deducing a contradiction, and as before we'll try to get that contradiction by first getting P-Q and Q-P and then deducing P-Q, the denial of (1):

1	(1)	~(P<->Q)	A (premise)
2	(2)	P	A
3	(3)	Q	A
3	(4)	P->Q	3 ->I (2)
2	(5)	Q->P	2 ->I (3)
2,3	(6)	P<->Q	4, 5 <->I
1,2	(7)	~Q	1,6 RAA(3)
1	(8)	P->~Q	->I (2)
9	(9)	~Q	A
10	(10)	~P	A
.	.	.	.
	(n-4)	P->Q	
	(n-3)	Q->P	
	(n-2)	P<->Q	n-4,n-3 <->I
	(n-1)	P	1, n-2, RAA (10)
1	(n)	~Q->P	->I (9)
1	(n+1)	P<->~Q	m,n <->I

Now, we might think we could just do what we did before in lines (4) and (5). However, we need to remember that we need the right assumption sets in front of the lines. At line (n), we're using ->I to get rid of assumption number (9), so it's a good bet that line (n-1) should have 9 in its assumption set. For a similar reason, line (n-2) is probably going to have (10) in its assumption set. And if (n-1) has 9 in its assumption set, then the only place it could get it would be from (n-2), so we can assume it has a 9 as well:

1	(1)	~(P<->Q)	A (premise)
2	(2)	P	A
3	(3)	Q	A
3	(4)	P->Q	3 ->I (2)
2	(5)	Q->P	2 ->I (3)
2,3	(6)	P<->Q	4, 5 <->I
1,2	(7)	~Q	1,6 RAA(3)
1	(8)	P->~Q	->I (2)
9	(9)	~Q	A
10	(10)	~P	A
.	.	.	.
	(n-4)	P->Q	

	(n-3)	Q->P	
9,10	(n-2)	P<->Q	n-4,n-3 <->I
9	(n-1)	P	1, n-2, RAA (10)
1	(n)	~Q->P	->I (9)
1	(n+1)	P<->~Q	m,n <->I

These are clues to how we will need to fill in the rest of the proof. The problem with just repeating lines (4) and (3) to get our contradiction is that we get the wrong assumption set: we would have 2,3 instead of 9,10. What we need, then, is a way of getting P that doesn't have a new assumption number attached. There's a way to do this. We have both P (line (2)) and ~P (line (10)), so we can use this contradiction to infer the denial of *any previous assumption*: and ~Q (at (9)) is an assumption, so we can use RAA to get Q *as the denial of line (9)*. Since the assumption we are discharging is (9), that gives (2,10) as the assumption set; we then get rid of 2 by using ->>I to deduce line (n-4). Changing this a little, we use the contradictory pair (9) and (3) to get P by RAA, then infer line (n-3) by ->I. This gives us the finished proof, so we can now fill in all the actual line numbers:

1	(1)	~(P<->Q)	A (premise)
2	(2)	P	A
3	(3)	Q	A
3	(4)	P->Q	3 ->I (2)
2	(5)	Q->P	2 ->I (3)
2,3	(6)	P<->Q	4, 5 <->I
1,2	(7)	~Q	1,6 RAA(3)
1	(8)	P->~Q	->I (2)
9	(9)	~Q	A
10	(10)	~P	A
2,10	(11)	Q	2,10 RAA (9)
3,9	(12)	P	3,9 RAA (10)
10	(13)	P->Q	11 ->I (2)
9	(14)	Q->P	12 ->I (3)
9,10	(15)	P<->Q	13,14 <->I
9	(16)	P	1, 15, RAA (10)
1	(17)	~Q->P	16 ->I (9)
1	(18)	P<->~Q	8,17 <->I

Astonishingly, the whole thing actually works out.

There's a lot of other exercises in here you could do...

Some exercises from Allen/Hand, Section 1.5.2

S54. P-> Q & R, R v ~Q -> S & T, T <-> U |- P -> U

Overall strategy: assume P, get U, use ->I. U occurs on the right side of a biconditional, so we can get it if we can get the other side (use <->E, then ->E), which is T. To get T, we need to get the right side of (2), for which we need its left side Rv~Q. Since that's a disjunction, we can get it if we can get either of its disjuncts R and ~Q. We can get R from the right side of (1) by &E; to get that, we'll need P. But we've already assumed P, so we've found the proof:

1	(1)	P->Q&R	A (premise)
2	(2)	Rv~Q->S&T	A (premise)
3	(3)	T<->U	A (premise)
4	(4)	P	A
1,4	(5)	Q&R	1,4 ->E
1,4	(6)	R	5 &E
1,4	(7)	Rv~Q	6 vI
1,2,4	(8)	S&T	2,7 ->E
1,2,4	(9)	T	8 &E
3	(10)	T->U	3 <->E
1,2,3,4	(11)	U	9,10 ->E
1,2,3	(12)	P->U	11 ->I (4)

S55. (~P v Q) & R, Q -> S |- P -> (R -> S)

The conclusion is a conditional, so assume P and try to deduce R->S. Since that's also a conditional, assume R and try to get S. To get S we need Q, and we can get that from the left side of (1) and vE if we can get P; but we've already assumed P. So, we can discharge the two assumptions of R and P, in that order, to get the conclusion.

1	(1)	(~PvQ)&R	A (premise)
2	(2)	Q->S	A (premise)
3	(3)	P	A
4	(4)	R	A
1	(5)	~PvQ	1 &E
1,3	(6)	Q	3,5 vE
1,2,3	(7)	S	2,6 ->E
1,2,3	(8)	R->S	7 ->I (4)
1,2	(9)	P->(R->S)	8 ->I (3)

Notice that this is just a mite sneaky. We didn't actually use the assumption (4) to get anything at all: we got S without using it, then discharged it to get R->S. Sneaky, but legitimate.

S56. Q & R, Q -> P v S, ~(S & R) |- P

Use the indirect approach. Having assumed P, what shall we try to contradict? Since it's hard to think of anything else to do with ~(S&R), let's try for that. We can get its denial, S&R, if we can get both S and R. R is easy (line (1) by &E), so we're halfway there. We could get S if we could get the right side of (2) and use vE (we already have assumed ~P). But Q comes from (1) by &E, so we're home:

1	(1)	Q&R	A (premise)
2	(2)	Q->PvS	A (premise)
3	(3)	~(S&R)	A (premise)
4	(4)	~P	A
1	(5)	Q	1 &E
1,2	(6)	PvS	2,5 ->E

1,2,4	(7)	S	4,6 vE
1	(8)	R	1 &E
1,2,4	(9)	S&R	7,8 &I
1,2,3	(10)	P	3,9 RAA (4)

S57. P -> R & Q, S -> ~R v ~Q |- S & P -> T

1	(1)	P->R&Q	A (premise)
2	(2)	S->~Rv~Q	A (premise)
3	(3)	S&P	A
4	(4)	~T	A
3	(5)	S	3 &E
2,3	(6)	~Rv~Q	2,5 ->E
3	(7)	P	3 &E
1,3	(8)	R&Q	1,7 ->E
1,3	(9)	R	8 &E
1,2,3	(10)	~Q	6,9 vE
1,3	(11)	Q	8 &E
1,2,3	(12)	T	10,11 RAA (4)
1,2	(13)	S&P->T	12 ->I (3)

S58. R & P, R -> (S v Q), ~(Q & P) |- S

S is a disjunct of the right side of (2), so if we can get that disjunct out (using ->E, requires R) we can get S with vE (this will require ~Q). R we can get from (1) with &E. ~Q is less obvious; we'll need to use the indirect approach. However, notice that if we assume Q and can get P, we can get Q&P, which is the denial of (3): so, there's our contradictory pair. Here's the proof:

1	(1)	R & P	A (premise)
2	(2)	R -> (S v Q)	A (premise)
3	(3)	~(Q & P)	A (premise)
4	(4)	Q	A
1	(5)	P	1 &E
1,4	(6)	Q&P	4,5 &I
1,3	(7)	~Q	3,6 RAA (4)
1	(8)	R	1 &E
1,2	(9)	SvQ	2,8 ->E
1,2,3	(10)	S	7,9 vE

S59. P & Q, R & ~S, Q -> (P -> T), T -> (R -> S v W) |- W

We could get W using vE, if we could get SvW out of line (4) and get ~S. ~S is available from (2) by &E. However, to get to SvW, we need T and R. R comes from line (2); T would come from the right side of (3), if we had P and Q. But we can get both those from (1), so we're set.

1	(1)	P & Q	A (premise)

2	(2)	R & ~S	A (premise)
3	(3)	Q -> (P -> T)	A (premise)
4	(4)	T -> (R -> S v W)	A (premise)
1	(5)	P	1 &E
1	(6)	Q	1 &E
1,3	(7)	P->T	3,6 ->E
1,3	(8)	T	5,7 ->E
1,3,4	(9)	R->SvW	4,8 ->E
2	(10)	R	2 &E
1,2,3,4	(11)	SvW	9,10 ->E
2	(12)	~S	2 &E
1,2,3,4	(13)	W	11,12 vE

S60. R -> ~P, Q, Q -> (P v ~S) |- S -> ~R

Main strategy: Assume S and try to get ~R, then use ->E. To get ~R, assume R and aim for a contradictory pair, then use RAA. There's an S and a ~S in the premises, so the contradictory pair to try for is S and ~S. We already have S, so it's just a matter of getting ~S. To get that, we need first to get the right side of (3) , for which we need Q; but we already have Q. Then, we need ~P, for which we need R; but we've already assumed that. So, we can do the proof:

1	(1)	R -> ~P	A
2	(2)	Q	A
3	(3)	Q -> (P v ~S)	A
4	(4)	S	A
5	(5)	R	A
1,5	(6)	~P	1,5 ->E
2,3	(7)	Pv~S	2,3 ->E
1,2,3,5	(8)	~S	6,7 vE
1,2,3,4	(9)	~R	4,8 RAA (5)
1,2,3	(10)	S->~R	9, ->I (4)

S61. P -> Q, P -> R, P -> S, T -> (U -> (~V -> ~S)), Q -> T, R -> (W -> U), V -> ~W, W |- ~P

This looks hideous, but we can approach it systematically. We want to assume P and aim or a contradictory pair, then use RAA. We have several possibilities hidden in the premises: w and ~W, S and ~S, V and ~V. Since W all by itself is a premise, let's try to get ~W. For that, we need V. We can get V (maybe) if we can get out the right side of the right side of (4), then use ->E twice (for which we will need T and U). We can get T from (5) if we can get Q, and we can get Q from (1) if we have P (but we've assumed it at (9). As for U, we can get that if we can get R and W; W we already have, and R we can get from (2) if we have P (and once again, we've assumed it at (9)). So, we can get ~V->~S. What we needed was V; so, we assume ~V and look for a contradictory pair, which is easy to find (get ~S from ~V->~S, S from (3) and (9)). Or, in its full glory:

1	(1)	P -> Q	A (premise)
2	(2)	P -> R	A (premise)
3	(3)	P -> S	A (premise)

4	(4)	T -> (U -> (~V -> ~S))	A (premise)
5	(5)	Q -> T	A (premise)
6	(6)	R -> (W -> U)	A (premise)
7	(7)	V -> ~W	A (premise)
8	(8)	W	A (premise)
9	(9)	P	A
1,9	(10)	Q	1,9 ->E
2,9	(11)	R	2,9 ->E
3,9	(12)	S	3,9 ->E
1,5,9	(13)	T	5,10 ->E
1,4,5,9	(14)	U -> (~V -> ~S)	4,13 ->E
2,6,9	(15)	W -> U	6,11 ->E
2,6,8,9	(16)	U	8,15 ->E
1,2,4,5,6,8,9	(17)	~V -> ~S	14,16 ->E
18	(18)	V	A
7,18	(19)	~W	7,18 ->E
7,8	(20)	~V	8,19 RAA (18)
1,2,4,5,6,7,8,9	(21)	~S	17,20 ->E
1,2,3,4,5,6,7,8	(22)	~P	12,21 RAA (9)

S62. P <-> ~Q & S, P & (~T -> ~S) |- ~Q & T

We need to build a conjunction, so we need to get each of its conjuncts. ~Q is available by &E if we can get the right hand side of (1), so we'll want to use <->E and then look for P to use with the resulting conditional and ->E. To get T, we need first to get at the right hand side of P (use &E), then try assuming T for an RAA. The latter is relatively straightforward: we get ~S from ~T->~S, then get S from ~Q&S. Here (after a little tidying up) is the proof:

1	(1)	P <-> ~Q & S	A (premise)
2	(2)	P & (~T -> ~S)	A (premise)
1	(3)	P->~Q&S	1, <->E
2	(4)	P	2 &E
1,2	(5)	~Q&S	3,4 ->E
1,2	(6)	S	5 &E
2	(7)	~T->~S	2 &E
8	(8)	~T	A
2,8	(9)	~S	7,8 ->E
1,2	(10)	T	6,9 RAA (8)
1,2	(11)	~Q	5 &E
1,2	(12)	~Q & T	10,11 &I

S63. P v Q <-> P & Q -||- P <-> Q

This is lengthy but relatively straightforward. Since each side is a biconditional, each of the two proofs will consist of *two* subproofs, each of which is the proof of a conditional. For left-to-right, we assume P and get Q, then assume Q and get P, then get the biconditional by <->I. In each case, we need only the left-right conditional from the premise.

1	(1)	P v Q <-> P & Q	A (premise)
1	(2)	PvQ -> P&Q	1 <->E
3	(3)	P	A
3	(4)	PvQ	3 vI
1,3	(5)	P&Q	2,4 ->E
1,3	(6)	Q	5 &E
1	(7)	P->Q	6 ->I (3)
8	(8)	Q	A
8	(9)	PvQ	8 vI
1,8	(10)	P&Q	2,9 ->E
1,8	(11)	P	10 &E
1	(12)	Q->P	11 ->I (8)
1	(13)	P<->Q	7,12 <->I

1	(1)	P<->Q	A (premise)
1	(2)	P->Q	1 <->E
1	(3)	Q->P	1 <->E
4	(4)	PvQ	A
5	(5)	~P	A
4,5	(6)	Q	4,5 vE
1,4,5	(7)	P	3,6 ->E
1,4	(8)	P	5,7 RAA (5)
1,4	(9)	Q	2,8 ->E
1,4	(10)	P&Q	8,9 &I
1	(11)	PvQ->P&Q	10 ->I (4)
12	(12)	P&Q	A
12	(13)	P	12 &E
12	(14)	PvQ	13 vI
	(15)	P&Q->PvQ	14 ->I (12)
1	(16)	PvQ<->P&Q	11,15 <->I

Notice that the assumption set for line (15) is empty. That's not an error: line 15 is a theorem which can be proved from no assumptions at all.

PROOFS WITH EVEN MORE RULES

In Which Theorems are Proved using Rules Derived as Well as Primitive

A theorem is simply a conclusion that can be proved from the empty set of premises. The most common technique for proving a theorem is to start with its denial as an assumption and construct a proof for it that discharges that assumption with RAA. If the theorem is a conditional, it may also be possible to prove it by assuming the antecedent, deducing the consequent, and using ->I. Here are some examples.

Exercise 1.6.1

T1 (Identity): |- P->P

1	(1)	P	A
	(2)	P->P	2 ->I (1)

This is about as short as a proof gets!

T2 (Excluded Middle): |- Pv~P

1	(1)	~(Pv~P)	A
2	(2)	P	A
2	(3)	Pv~P	2 vI
1	(4)	~P	1,3 RAA (2)
1	(5)	Pv~P	4 vI
	(6)	Pv~P	1,5 RAA (1)

T3 (Non-Contradiction): |- ~(P&~P)

1	(1)	P&~P	A
1	(2)	P	1 &E
1	(3)	~P	1 &E
	(4)	~(P&~P)	2,3 RAA (1)

T4 (Weakening): |- P->(Q->P)

1	(1)	P	A
2	(2)	Q	A
3	(3)	~P	A
1	(4)	P	1,3 RAA (3)
1	(5)	Q->P	4 ->I (2)
	(6)	P->(Q->P)	5 ->I (1)

T5 (Paradox of Material Implication): |- (P->Q)v(Q->P)

1	(1)	~((P->Q)v(Q->P))	A
2	(2)	P	A
3	(3)	Q	A
3	(4)	P->Q	3 ->I (2)

3	(5)	(P->Q)v(Q->P)	4 vI
1	(6)	~Q	1,5 RAA (3)
7	(7)	Q	A
8	(8)	~P	A
1,7	(9)	P	6,7 RAA (8)
1	(10)	Q->P	9 ->I (7)
1	(11)	(P->Q)v(Q->P)	10 vI
	(12)	(P->Q)v(Q->P)	1,11 RAA(1)

T6 (Double Negation): |- P <-> ~~P

1	(1)	P	A
2	(2)	~P	A
1	(3)	~~P	1,2 RAA (2)
	(4)	P->~~P	3 ->I (1)
	(5)	~~P->P	4 Trans
	(6)	P<->~~P	4,5 <->I

T7: |- (P<->Q)<->(Q<->P)

With derived rules, this is *very* easy:

1	(1)	P<->Q	A
1	(2)	Q<->P	1 <-> Comm
	(3)	(P<->Q)->(Q<->P)	2 ->I (1)
4	(4)	Q<->P	A
4	(5)	P<->Q	4 <-> Comm
	(6)	(Q<->P)->(P<->Q)	5 ->I (4)
	(7)	(P<->Q)<->(Q<->P)	3,6 <->I

T8: |- ~(P<->Q) <-> (~P<->Q)

1	(1)	~(P<->Q)	A
2	(2)	P	A
3	(3)	Q	A
3	(4)	P->Q	3 ->I(2)
2	(5)	Q->P	2 ->I(3)
2,3	(6)	P<->Q	4,5 <->I
1,3	(7)	~P	1,6 RAA(2)
1	(8)	Q->~P	7 ->I(3)
9	(9)	~P	A
10	(10)	~Q	A
3,10	(11)	P	3,10 RAA(9)
10	(12)	Q->P	11 ->I(3)
2,9	(13)	Q	2,9 RAA(10)
9	(14)	P->Q	13 ->I(2)

9,10	(15) P<->Q	12,14 <->I
1,9	(16) Q	1,15 RAA(10)
1	(17) ~P->Q	16 ->I(9)
1	(18) ~P<->Q	8,17 <->I
	(19) ~(P<->Q) -> (~P<->Q)	18 ->I(1)
20	(20) ~P<->Q	A
21	(21) P<->Q	A
20	(22) ~P->Q	20 <->E
20	(23) Q->~P	20 <->E
21	(24) P->Q	21 <->E
21	(25) Q->P	21 <->E
26	(26) P	A
21,26	(27) Q	24,26 ->E
20,21,26	(28) ~P	23,27 ->E
20,21	(29) ~P	26,28 RAA(26)
20,21	(30) Q	22,29 ->E
20,21	(31) P	25,30 ->E
20	(32) ~(P<->Q)	29,31 RAA(21)
	(33) (~P<->Q) -> ~(P<->Q)	32 ->I(20)
	(34) (~P<->Q) <-> ~(P<->Q)	19,33 <->I

T9 (Peirce's Law): |- ((P -> Q) -> P) ->P

Since this is a conditional, the straightforward approach is to assume the antecedent and try to deduce the consequent. We need RAA to finish the proof, however.

1	(1)	(P -> Q) -> P	A
2	(2)	~P	A
3	(3)	P -> Q	A
1,3	(4)	P	1,3 ->E
1,2	(5)	~(P -> Q)	2,4 RAA(3)
2	(6)	~P v Q	2 vI
7	(7)	P	A
2,7	(8)	Q	6,7 vE
2	(9)	P->Q	8 ->I(7)
1	(10)	P	5,9 RAA(2)
	(11)	((P -> Q) -> P) ->P	10 ->I(1)

T10: |- (P->Q)v(Q ->R)

We assume the denial of what we want and use RAA. With disjunctions, this is always harder than you expect. This can be shortened considerably with derived rules, obviously.

1	(1)	~((P->Q)v(Q ->R))	A
2	(2)	P->Q	A
2	(3)	(P->Q)v(Q ->R)	2 vI

1	(4)	~(P->Q)	1,3 RAA(2)
5	(5)	~P	A
6	(6)	P	A
6	(7)	P v Q	6 vI
5,6	(8)	Q	5,7 VE
5	(9)	P->Q	8 ->I(6)
1	(10)	P	4,9 RAA(5)
11	(11)	Q	A
11	(12)	P ->Q	11 ->I(6)
1	(13)	~Q	4,12 RAA(11)
1	(14)	~Q v R	13 vI
1,11	(15)	R	11,14 vE
1	(16)	Q -> R	15 ->I(11)
1	(17)	(P->Q)v(Q ->R)	16 vI
	(18)	(P->Q)v(Q ->R)	1,17 RAA(1)

T11: |- (P <-> Q) <-> (~P <-> ~Q)

1	(1)	P <-> Q	A
2	(2)	~P	A
3	(3)	~Q	A
1	(4)	P->Q	1 <->E
1	(5)	Q->P	1 <->E
6	(6)	P	A
1,6	(7)	Q	4,6 ->E
1,3	(8)	~P	3,7 RAA(6)
1	(9)	~Q ->~P	8 ->I(3)
10	(10)	Q	A
1,10	(11)	P	5,10 ->E
1,2	(12)	~Q	2,11 RAA(10)
1	(13)	~P->~Q	12 ->I(2)
1	(14)	~P<->~Q	9,13 <->I
	(15)	(P <-> Q) -> (~P <-> ~Q)	14 ->I(1)

...and that's halfway there. But the reverse direction is very similar: just interchange ~P and P, ~Q and Q in the first 15 lines. You need a final step of <->I, of course.

T12: |-

1	(1)	A
2	(2)	A
1	(3)	
	(4)	
	(5)	
	(6)	
	(7)	

	(8)		
	(9)		
	(10)		
	(11)		
	(12)		

T13 (& Idempotence): |- P <-> (P&P)

This is relatively simple.

1	(1)	P	A
1	(2)	P&P	1,1 &I
	(3)	P->(P&P)	2 ->I(1)
4	(4)	P&P	A
4	(5)	P	4 &E
	(6)	(P&P)->P	5 ->I(4)
	(7)	P <-> (P&P)	3,6 <->I

T14 (v Idempotence): |- P <-> (P v P)

1	(1)		A
2	(2)		A
1	(3)		
	(4)		
	(5)		
	(6)		

T15: |- (P<->Q)&(R<->S) ->((P->R)<->(Q->S))

Assume the antecedent, then get the consequent by assuming each side in turn, deducing the other side, using ->I, and then combining with <->I. The final step is then discharging the original assumption with ->I.

1	(1)	(P<->Q)&(R<->S)	A
2	(2)	P->R	A
3	(3)	Q	A
1	(4)	P<->Q	1 &E
1	(5)	R<->S	1 &E
1	(6)	P->Q	4 <->E
1	(7)	Q->P	4 <->E
1	(8)	R->S	5 <->E
1	(9)	S->R	5 <->E
1,3	(10)	P	3,7 ->E
1,2,3	(11)	R	2,10 ->E
1,2,3	(12)	S	8,11 ->E
1,2	(13)	Q->S	12 ->I(3)
1	(14)	(P->R)->(Q->S)	13 ->I(2)
15	(15)	Q->S	A

16	(16)	P	A
1,16	(17)	Q	6,16 ->E
1,15,16	(18)	S	15,17 ->E
1,15,16	(19)	R	9,18 ->E
1,15	(20)	P->R	19 ->I(16)
1	(21)	(Q->S)->(P->R)	20 ->I(15)
1	(22)	(P->R)<->(Q->S)	14,21 <->I
	(23)	(P<->Q)&(R<->S) -> ((P->R)<->(Q->S))	2 ->I(1)

T20: |- (P<->Q)->(R&P <-> R&Q)

1	(1)	P<->Q	A
2	(2)	R&P	A
1	(3)	P->Q	1 <->E
1	(4)	Q->P	1 <->E
2	(5)	R	2 &E
2	(6)	P	2 &E
1,2	(7)	Q	3,6 ->E
1,2	(8)	R&Q	5,7 &I
1	(9)	R&P -> R&Q	8 ->I(2)
10	(10)	R&Q	A
10	(11)	R	10 &E
10	(12)	Q	10 &E
1,10	(13)	P	4,12 ->E
1,10	(14)	R&P	11,13 &I
1	(15)	R&Q -> R&P	14 ->I(10)
	(16)	R&P <-> R&Q	9,15 <->I
	(17)	(P<->Q)->(R&P <-> R&Q)	16 ->I(1)

T23: |- P & (Q<->R) -> (P&Q <-> R)

1	(1)	P & (Q<->R)	A
2	(2)	P&Q	A
1	(3)	P	1 &E
1	(4)	Q<->R	1 &E
1	(5)	Q->R	4 <->E
2	(6)	Q	2 &E
1,2	(7)	R	5,6 ->E
1	(8)	P&Q -> R	7 ->I(2)
9	(9)	R	A
1	(10)	R->Q	4 <->E
1,9	(11)	Q	9,10 ->E
1,9	(12)	P&Q	3,11 &I
1	(13)	R -> P&Q	12 ->I(9)

1	(14) P&Q <-> R	8,13 <->I
	(15) P & (Q<->R) -> (P&Q <-> R)	14 ->I(1)

T25: \|- P -> (Q ->R) <-> Q -> (P->R)

Standard strategy for a biconditional: assume each side in turn and deduce the other, use ->I twice, then use <->I. Notice that steps 9-16 are exactly like steps 1-8 with P and Q interchaged. See if you can shorten this proof by using some assumptions more than once.

1	(1)	P -> (Q ->R)	A
2	(2)	Q	A
3	(3)	P	A
1,3	(4)	Q->R	1,3 ->E
1,2,3	(5)	R	2,4 ->E
1,2	(6)	P->R	5 ->I(3)
1	(7)	Q -> (P->R)	6 ->I(2)
	(8)	(P -> (Q->R)) -> (Q -> (P->R))	7 ->I(1)
9	(9)	Q -> (P->R)	A
10	(10)	P	A
11	(11)	Q	A
9,11	(12)	P->R	9,11 ->E
9,10,11	(13)	R	10,12 ->E
9,10	(14)	Q->R	13 ->I(11)
9	(15)	P -> (Q->R)	14 ->I(10)
	(16)	(Q -> (P->R)) -> (P -> (Q->R))	15 ->I(9)
	(17)	P -> (Q ->R) <-> Q -> (P->R)	8,16 <->I

Note that on lines 8 and 16 you can't drop the parentheses around the antecedent and consequent, even though you can drop the analogous parentheses in the conclusion.

T29: \|- ~P -> P <-> P

Notice that this is (~P -> P) <-> P (<-> binds more weakly than .>). This is an interesting result: what might an actual sentence of the form ~P ->P be like? The proof might be considered excessively clever.

1	(1) ~P -> P	A
2	(2) ~P	A
1,2	(3) P	1,2 ->E
1	(4) P	2,3 RAA(2)
	(5) (~P->P) ->P	4 ->I(1)
6	(6) P	A
6	(7) ~P->P	6 ->I(2)
	(8) P->(~P->P)	7 ->I(6)
	(9) ~P -> P <-> P	5,8 <->I

T33: \|- (Pv~P) & Q <-> Q

As it happens, Pv~P is a theorem (EM). We can use this in constructing a quick proof, if we jump ahead just a little (see pp. 37-38). Could you generalize this to cases involving theorems other than Pv~P?

Incidentally, the Logic Daemon doesn't know the names of theorems and will flag an error if you try to use this procedure.

1	(1)	(Pv~P) & Q	A
1	(2)	Q	1 &E
	(3)	(Pv~P) & Q -> Q	2 ->I(1)
4	(4)	Q	A
	(5)	Pv~P	EM
4	(6)	(Pv~P) & Q	4,5 &I
	(7)	Q-> (Pv~P) & Q	6 ->I(4)
	(8)	(Pv~P) & Q <-> Q	3,7 <->I

T: |-

1	(1)		A
2	(2)		A
1	(3)		
	(4)		
	(5)		
	(6)		

T8 : |-

1	(1)		A
2	(2)		A
1	(3)		
	(4)		
	(5)		
	(6)		

T39: |- (P->Q&R)->(P&Q<->P&R)

1	(1)	P->Q&R	A
2	(2)	P&Q	A
2	(3)	P	2 &E
1,2	(4)	Q&R	1,3 ->E
1,2	(5)	R	4 &E
1,2	(6)	P&R	3,5 &I
1	(7)	P&Q->P&R	6 ->I(2)
8	(8)	P&R	A
8	(9)	P	8 &E
1,8	(10)	Q&R	1,9 ->E
1,8	(11)	Q	10 &E
1,8	(12)	P&Q	9,11 &I
1	(13)	P&R->P&Q	12 ->I(8)

1	(14)	P&Q<->P&R	7,13 <->I
	(15)	(P->Q&R)->(P&Q<->P&R)	14 ->I(1)

To the Syllabus

Back to Exercise 1.5.2

On to Exercise 2.1

Truth Table Basics

Our logical theory so far consists of a **vocabulary** of basic symbols, rules defining how to combine symbols into **wffs**, and rules defining how to construct **proofs** from wffs. All of this only concerns **manipulating symbols.** We now need to give these symbols some **meanings**.

We are going to give them just a little meaning. For the **sentence letters**, all that we are actually going to notice is that each of them must be either true or false. That's as far as we will go.

For the **connectives**, we will develop more of a theory. Each of them has a meaning that is defined in terms of how it affects the meanings of sentences that contain it.

Since a wff represents a sentence, it must be either true or false. We will call this its **truth value: the truth value of a wff is "true" if the wff is true and "false" if the wff is false.**

The truth values of atomic sentences are determined by whatever those sentences mean and what the world is like. For example, the truth value of "It is raining" is determined by what it means and whether or not it is raining. Likewise, the truth value of "Austin is the largest city in Texas" is determined by what it means and what the facts are about cities in Texas. These two sentences are about the weather and geography, respectively. Since this is not a course in meteorology or geography, we won't have anything else to say about the truth values of atomic sentences except that they have them.

For compound sentences, however, we do have a theory. Some compound sentences are **truth functions** of their constituents. Take the simple sentence "It's cold and it's snowing." Is it true or false? We can't tell without knowing something about the weather, but we *can* say how its truth value depends on the truth values of the two atomic sentences in it:

It's cold	It's snowing	It's cold and it's snowing
True	True	True
True	False	False
False	True	False
False	False	False

All that you need to know to determine whether or not *"It's cold and it's snowing"* is true or false is whether each of its constitutents is true or false. We describe this by saying that *"It's cold and it's snowing"* is a **truth function** of its constituents.

Notice that this sentence works like it does **because of the meaning of the word *"and"***. This word combines two sentences into a new sentence that has a truth value determined in a certain way as a function of the truth values of those two sentences. So, *and* is a **truth functional connective**.

We define each of the four connections using a table like the one above that shows, schematically, how the truth value of a **wff** made with that connective depends on the truth values of its constituents.

For the ampersand:

Φ	Ψ	Φ & Ψ
T	T	T
T	F	F

F	T	F
F	F	F

For the wedge:

Φ	**Ψ**	**Φ v Ψ**
T	T	T
T	F	T
F	T	T
F	F	F

For the arrow:

Φ	**Ψ**	**Φ → Ψ**
T	T	T
T	F	F
F	T	T
F	F	T

For the double arrow:

Φ	**Ψ**	**Φ ↔ Ψ**
T	T	T
T	F	F
F	T	F
F	F	T

And the simplest of all, for the tilde:

Φ	**~Φ**
T	F
F	T

These rules also define the meanings of more complex sentences. Consider this sentence:

~P → Q

This is a conditional (main connective →), but the antecedent of the conditional is a negation. To construct its truth table, we might do this:

~P	Q	~P → Q
T	T	T
T	F	F
F	T	T
F	F	T

However, ~P is also a truth function of P. So, to get a more complete truth table, we should consider the truth values of the atomic constituents.

P	~P	Q	~P → Q
T	F	T	T
T	F	F	T
F	T	T	T
F	T	F	F

A still more complicated example is the truth table for (P→Q)&(Q→P).

P	Q	P→Q	Q → P	(P → Q) & (Q → P)
T	T	T	T	T
T	F	F	T	F
F	T	T	F	F
F	F	T	T	T

How is this table constructed? It will help to go through it step by step.

1. Determine the Columns for the Truth Table

The first step is to **determine the columns of our truth table.** To do that, we **take the wff apart into its constituents until we reach sentence letters.** As we do that, we add a column for each constituent.

This is a step-by-step process as well. The steps are these:

1. Find the main connective of the wff we are working on.
2. Determine the main constituents that go with this connective.
3. Add new columns to the left for each constituent.
4. Repeat for each new constituent.

To continue with the example(P→Q)&(Q→P), the first step is to set up a truth table with this statement as its its only column:

(P→Q) & (Q→P)

Next, we identify the **main connective** of this wff:

(P→Q) & (Q→P)

Now we identify the **main constituents** that go with this connective. To make it clear that these are part of a single step, they are identified with a "1" to indicate that this is the first step:

(P→Q) & (Q→P)

Next, we **add columns under the constituents and the main connective**:

<table>
<tr><th colspan="3">(P→Q) & (Q→P)</th></tr>
<tr><td>(P→Q)</td><td>&</td><td>(Q→P)</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td></tr>
</table>

We now **repeat the process** with the constituents we have just found, working down below each constituent. We start with P→Q:

<table>
<tr><td colspan="5">(P→Q) & (Q→P)</td></tr>
<tr><td colspan="3">(P→Q)</td><td>&</td><td>(Q→P)</td></tr>
<tr><td>P</td><td>→</td><td>Q</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td colspan="2"></td></tr>
</table>

We then proceed to the constituents of P→Q:

<table>
<tr><td colspan="5">(P→Q) & (Q→P)</td></tr>
<tr><td colspan="3">(P→Q)</td><td>&</td><td>(Q→P)</td></tr>
<tr><td>P</td><td>→</td><td>Q</td><td></td><td></td></tr>
<tr><td>P</td><td></td><td>Q</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td colspan="2"></td></tr>
</table>

Next, Q→P

<table>
<tr><td colspan="7">(P→Q) & (Q→P)</td></tr>
<tr><td colspan="3">(P→Q)</td><td>&</td><td colspan="3">(Q→P)</td></tr>
<tr><td>P</td><td>→</td><td>Q</td><td></td><td>Q</td><td>→</td><td>P</td></tr>
<tr><td>P</td><td></td><td>Q</td><td></td><td>Q</td><td></td><td>P</td></tr>
<tr><td></td><td></td><td></td><td></td><td colspan="3"></td></tr>
</table>

We've now reached sentence letters under each of the constituents. So, the next step is to add columns to the left for each sentence letter:

<table>
<tr><td rowspan="4">P</td><td rowspan="4">Q</td><td colspan="7">(P→Q) & (Q→P)</td></tr>
<tr><td colspan="3">(P→Q)</td><td>&</td><td colspan="3">(Q→P)</td></tr>
<tr><td>P</td><td>→</td><td>Q</td><td></td><td>Q</td><td>→</td><td>P</td></tr>
<tr><td>P</td><td></td><td>Q</td><td></td><td>Q</td><td></td><td>P</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="3"></td></tr>
</table>

<table>
<tr><td rowspan="4">P</td><td rowspan="4">Q</td><td colspan="7">(P→Q) & (Q→P)</td></tr>
<tr><td colspan="3">(P→Q)</td><td>&</td><td colspan="3">(Q→P)</td></tr>
<tr><td>P</td><td>→</td><td>Q</td><td></td><td>Q</td><td>→</td><td>P</td></tr>
<tr><td>P</td><td></td><td>Q</td><td></td><td>Q</td><td></td><td>P</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td colspan="3"></td></tr>
</table>

2. Determine the Rows of the Truth Table

What we are trying to construct is a table that shows what the truth value of the main wff is for any combination of truth values of its constituents. We will do this by constructing **one row for each possible combination of truth values.**

All that we have to consider is the combinations of truth values of the **sentence letters**, since everything else is determined by these. So, we want to include one row in our truth table for each combination of truth values of the sentence letters. In this case, there are two sentence letters, P and Q. What are the possible combinations of truth values for P and Q? Think about it this way:

- There are two possibilities, **T** and **F**, for P.
- For each of these cases, there are two possibilities: Q = **T** and Q = **F**.
- Therefore, there are 2 × 2 = 4 possibilities altogether.

An easy way to write these down is to begin by adding four rows to our truth table, since we know that there are four combinations:

P	Q	(P→Q) & (Q→P)						
		(P→Q)			&	(Q→P)		
		P	→	Q		Q	→	P
		P		Q		Q		P

Half of these will have P = **T** and half will have P = **F**:

P	Q	(P→Q) & (Q→P)						
		(P→Q)			&	(Q→P)		
		P	→	Q		Q	→	P
		P		Q		Q		P
T								
T								
F								
F								

For each of these halves, one will have Q = **T** and one will have Q = **F**:

P	Q	(P→Q) & (Q→P)						
		(P→Q)			&	(Q→P)		
		P	→	Q		Q	→	P
		P		Q		Q		P
T	**T**							
T	**F**							
F	**T**							
F	**F**							

3. Calculate the Truth Values for Each Row

The last step is to work across each row from left to right, **calculating the truth value for each column based on the truth values of wffs to the left and the connective used in that column**. So, we start with the first row and work across.

For each column in that row, we need to ask:

1. What is the **main connective** of the wff at the top of the column?
2. What **previous column(s)** are the main constituents in?

For the first column, the main connective is → and the previous columns are the first two columns:

P ↑	Q ↑	(P→Q) ↑	(Q → P)	(P→Q) & (Q→P)
T	**T**			
T	**F**			
F	**T**			
F	**F**			

Next, look at the **truth value combination** we find in those previous columns:

P	Q	(P→Q)	(Q → P)	(P→Q) & (Q→P)
T ↑	**T** ↑			
T	**F**			
F	**T**			
F	**F**			

Now, substitute that combination of truth values for the constituents in the column we're working on and **look up the value they produce using the truth table for the main connective**. In this case, we want to use the combination P = **T**, Q = **T** in the wff (P→Q).

P	Q	(P→Q)	(Q → P)	(P→Q) & (Q→P)
T	**T**	***(T→T)***		
T	**F**			
F	**T**			
F	**F**			

Now we need to look up the appropriate combination in the truth table for the arrow:

Φ	**Ψ**	**Φ → Ψ**
T	T	T ↑
T	F	F
F	T	T
F	F	T

And we substitute this into the cell we are working on in our truth table:

P	Q	(P→Q)	(Q → P)	(P→Q) & (Q→P)
T	**T**	**T**		
T	**F**			

F	**T**			
F	**F**			

That's one! We go on to the next column, headed by (Q→P). This depends on the same two columns as the previous column did, **but not in the same order**: here, Q is the antecedent and P is the consequent.

P	Q	(P→Q)	(Q → P)	(P→Q) & (Q→P)
T	**T**	**T**	*(**T**→**T**)*	
T	**F**			
F	**T**			
F	**F**			

We can then substitute the value from the table for →:

P	Q	(P→Q)	(Q → P)	(P→Q) & (Q→P)
T	**T**	**T**	**T**	
T	**F**			
F	**T**			
F	**F**			

Going on to the last column, we have a wff that is a conjunction (main connective &), with constituents (P → Q) and (Q → P):

P	Q	(P → Q)	(Q → P)	(P→Q) & (Q→P)
T	**T**	**T**	**T**	
T	**F**			
F	**T**			
F	**F**			

We need to evaluate this combination:

P	Q	(P → Q)	(Q → P)	(P→Q) & (Q→P)
T	**T**	**T**	**T**	(**T** & **T**)
T	**F**			
F	**T**			
F	**F**			

That corresponds to this row of the truth table for the ampersand:

Φ	Ψ	Φ & Ψ
T	T	T ↑
T	F	F
F	T	F
F	F	F

So, we complete the first row as follows:

P	Q	(P → Q)	(Q → P)	(P→Q) & (Q→P)
T	**T**	**T**	**T**	**T**
T	**F**			
F	**T**			
F	**F**			

Here's the next row. Notice that the values under (P → Q) and (Q → P) are not the same. Why?

P	Q	(P → Q)	(Q → P)	(P→Q) & (Q→P)
T	**T**	**T**	**T**	**T**
T	**F**	**F**	**T**	**F**
F	**T**			
F	**F**			

Finally, here is the full truth table. Notice that what this shows, overall, is **what the truth value of (P → Q) & (Q → P) is for each combination of truth values of its atomic constituents (sentence letters)**.

P	Q	(P → Q)	(Q → P)	(P→Q) & (Q→P)
T	**T**	**T**	**T**	**T**
T	**F**	**F**	**T**	**F**
F	**T**	**T**	**F**	**F**
F	**F**	**T**	**T**	**T**

A More Economical Way with Columns

To the syllabusMore about truth tables

Last modified: Wed Oct 4 09:18:16 CDT 2006

Truth Tables for Sequents

We can also construct a truth table for an entire sequent. The procedure is exactly like what we've just done, except that we start with several columns to the right: one for each premise and one for the conclusion. Here is a simple example:

Sequent:

(P v Q) → R, ~R |- ~Q

Begin with columns for premises and conclusion:

(P v Q) → R	~R	|- ~Q

Decompose the wffs:

Q	(P v Q) → R	~R	|- ~Q

Q	R	(P v Q) → R	~R	|- ~Q

Q	R	(P v Q)	(P v Q) → R	~R	|- ~Q

P	Q	R	(P v Q)	(P v Q) → R	~R	|- ~Q

Figure out the combinations (there are three sentence letters, so we will need 2 × 2 × 2 = 8 rows)

P	Q	R	(P v Q)	(P v Q) → R	~R	|- ~Q
T	T	T				
T	T	F				
T	F	T				
T	F	F				
F	T	T				
F	T	F				
F	F	T				
F	F	F				

Do the calculations:

P	Q	R	(P v Q)	(P v Q) → R	~R	|- ~Q
T	T	T	T	T	F	F
T	T	F	T	F	T	F
T	F	T	T	T	F	T

T	**F**	**F**	**T**	**F**	**T**	**T**
F	**T**	**T**	**T**	**T**	**F**	**F**
F	**T**	**F**	**T**	**F**	**T**	**F**
F	**F**	**T**	**F**	**T**	**F**	**T**
F	**F**	**F**	**F**	**T**	**T**	**T**

Now, examine the truth values for the two premises and the conclusion in each row. Is there any row in which the conclusion is false and the premises both true?

				Premise 1	**Premise 2**	**Conclusion**
P	**Q**	**R**	**(P v Q)**	**(P v Q) → R**	**~R**	**|- ~Q**
T	**T**	**T**	**T**	**T**	**F**	**F**
T	**T**	**F**	**T**	**F**	**T**	**F**
T	**F**	**T**	**T**	**T**	**F**	**T**
T	**F**	**F**	**T**	**F**	**T**	**T**
F	**T**	**T**	**T**	**T**	**F**	**F**
F	**T**	**F**	**T**	**F**	**T**	**F**
F	**F**	**T**	**F**	**T**	**F**	**T**
F	**F**	**F**	**F**	**T**	**T**	**T**

In fact, there isn't. There are four rows in which the conclusion is false (marked in red), but in each case at least one premise is also false. There is one case in which both premises are both true (marked in yellow), but in that case the conclusion is also true. So, we can say that in this argument, there is no way for the premises to be true and the conclusion also false at the same time. Therefore, **this argument is valid.**

Now, let's consider an argument that is not valid:

P v Q, Q → S, T |- S & T

P	**Q**	**S**	**T**	**P v Q**	**Q → S**	**T**	**|- S & T**
T	T	T	T	T	T	T	T
T	T	T	F	T	T	F	F
T	T	F	T	T	F	T	F
T	T	F	F	T	F	F	F
T	F	T	T	T	T	T	T
T	F	T	F	T	T	F	F
T	F	F	T	T	T	T	F
T	F	F	F	T	T	F	F
F	T	T	T	T	T	T	T
F	T	T	F	T	T	F	F
F	T	F	T	T	F	T	F
F	T	F	F	T	F	F	F
F	F	T	T	F	T	T	T
F	F	T	F	F	T	F	F
F	F	F	T	F	T	T	F
F	F	F	F	F	T	F	F

There is one rows in which the premises are all true and the conclusion false. Therefore, this argument is not valid. It does not matter how many such rows there are as long as there is at least one: if there is **at least one row in which the premises are true and the conclusion false**, then the argument is invalid.

Invalidating Assignments

The truth values assigned to the atomic wffs (sentence letters) in a row of a truth table for a sequent in which the premises are true and the conclusion is false are called an **invalidating assignment:**

> An **INVALIDATING ASSIGNMENT** for a sequent is an assignment of truth and falsity to its sentence letters that makes the premises true and the conclusion false.

If a sequent has an invalidating assignment, then it is invalid (do you see why?). Therefore, a valid sequent has no invalidating assignments. You could in fact define a valid sequent as one for which no assignment is an invalidating assignment.

As soon as we have found a row in which the premises are true and the conclusion false, we can stop: we know at that point that the argument is invalid, and filling in further rows will not add anything to this.

P	Q	S	T	P v Q	Q → S	T	|- S & T
T	T	T	T	T	T	T	T
T	T	T	F	T	T	F	F
T	T	F	T	T	F	T	F
T	T	F	F	T	F	F	F
T	F	T	T	T	T	T	T
T	F	T	F	T	T	F	F
T	F	F	T	T	T	T	F
T	F	F	F				
F	T	T	T				
F	T	T	F				
F	T	F	T				
F	T	F	F				
F	F	T	T				
F	F	T	F				
F	F	F	T				
F	F	F	F				

We can use this to develop an abbreviated truth-table test by trying to work backwards from the assumption that an argument is invalid. Taking the same example, suppose that it did have true premises and a false conclusion. We can represent this by starting out a "truth table" with the **right** side filled in first:

P	Q	S	T	P v Q	Q → S	T	|- S & T
			T	T	T	T	F

What can we add to this? First, if S & T is false and T is true, then S must be false:

P	Q	S	T	P v Q	Q → S	T	|- S & T
		F	T	T	T	T	F

Next, if Q → S is true and S is false, then Q must be false:

P	Q	S	T	P v Q	Q → S	T	|- S & T
	F	F	T	T	T	T	F

But if Q is false and P v Q is true, then P must be true:

P	Q	S	T	P v Q	Q → S	T	|- S & T
T	F	F	T	T	T	T	F

In this case, we have figured out the only possible combination of truth values for the sentence letters in these wffs that makes the conclusion false and the premises true: P = **T**, Q = **F**, S = **F**, and T = **T**.

What would happen if we tried this method on a valid argument? First, let's take note of a difference between what it takes to show that an argument is valid and what it takes to show it is invalid:

> To show that an argument is invalid, we only need to find **one row of its truth table in which the premises are true and the conclusion false.**
>
> To show that an argument is valid, we need to show that **there is no row of its truth table in which the premises are true and the conclusion false.**

The important difference is that once we have found a single row with true premises and a false conclusion, we can stop (since we know that the argument is invalid), but in order to prove that it is valid we will have to **check every row**.

As a side note, you may think that the reason proving an argument is valid requires more work than proving it is invalid is that "it is hard to prove a negative." The real reason, however, is that proving validity requires proving something **universal**: it requires proving, for **every possible combination of truth values**, that that combination does not make the premises true and the conclusion false. Consider this sentences:

> Every member of Congress is either a Democrat or a Republican.

There is nothing negative about this sentence, but in order to prove it you will need to determine, for each and every member of Congress, whether or not that person is either a Democrat or a Republican. On the other hand, to prove it false, all you need to do is find one member of Congress who is neither a Democrat nor a Republican.

Back to 'Truth Table Basics'
On to 2.2
To the syllabus
Last modified: Mon Oct 2 10:00:49 CDT 2006

Tautologies, Inconsistent Sentences, and Contingent Sentences

Tautologies

Truth tables can be used for other purposes. One is to test statements for certain logical properties. Some sentences have the property that they cannot be false under any circumstances. An example is P v ~P:

P	~P	P v ~P
T	F	T
F	T	T

A sentence with this property is called a **tautology.** Another example:

(P → Q) ↔ ~(P &~Q)

P	Q	(P→Q)↔ ~(P &~Q)								
		P→Q			↔	~(P&~Q)				
		P	→	Q		~(P&~Q)				
						~	(P&~Q)			
						~	P	&	~Q	
									~	Q
T	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**F**	**F**	**T**
T	**F**	**T**	**F**	**F**	**T**	**F**	**T**	**T**	**T**	**F**
F	**T**	**T**	**T**	**T**	**T**	**T**	**F**	**F**	**F**	**T**
F	**F**	**T**	**T**	**F**	**T**	**T**	**F**	**F**	**T**	**F**

If there are sentences that are always true, then there are sentences that are always false. Such sentences are called **INCONSISTENT**. One example:

P	~P	P & ~P
T	F	F
F	T	F

We defined an argument as "a **set** of sentences (the premises) and a sentence (the conclusion)." That definition does not actually say that an argument must have *premises*, only that it must have a *set of premises*. However, sets can be empty, that is, they can have no members. We can also have an argument with an empty set of premises. As a sequent, such an argument would be written like this:

|- Φ

If a sequent has an empty set of premises, can it be valid? Yes: if it is a tautology, then it is valid. The definition applies to it because it is impossible for it to have true premises and a false conclusion (since it is impossible for it to have a false conclusion at all).

All other sentences--that is, all those that are neither tautologies nor inconsistent--are called **CONTINGENT**. A contingent sentence is one that is neither always true nor always false. Equivalently, a contingent sentence is one that is true for at least one combination of truth values and false for at least one combination of truth values.

Here are a few provable truths:

- Every sentence is a tautology, or inconsistent, or contingent, and no sentence is more than one of these.
- The denial of a tautology is inconsistent.
- The denial of an inconsistent sentence is a tautology.
- The denial of a contingent sentence is a contingent sentence.

Incompatible Premises

We can extend the notion of inconsistency to sets of sentences. Even if two sentences are both contingent, it may be impossible for them both to be true at the same time. Example:

P→Q and P&~Q

To see this, look at the truth table above for (P → Q) ↔ ~(P &~Q), but note the columns for P→Q and P&~Q:

P	**Q**	**(P→Q)↔ ~(P &~Q)**								
		P→Q			↔	~(P&~Q)				
		P	→	Q		~(P&~Q)				
						~	(P&~Q)			
							P	&	~Q	
									~	Q
T	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**F**	**F**	**T**
T	**F**	**T**	**F**	**F**	**T**	**F**	**T**	**T**	**T**	**F**
F	**T**	**T**	**T**	**T**	**T**	**T**	**F**	**F**	**F**	**T**
F	**F**	**T**	**T**	**F**	**T**	**T**	**F**	**F**	**T**	**F**

There is no row in which these two wffs have the same truth value. Therefore, it is not possible for them to be true at the same time. We call these two sentences **INCOMPATIBLE**.

If the premises of an argument are incompatible, then it is impossible for all the premises to be true. If it is impossible for the premises all to be true, then it is impossible for the premises all to be true and the conclusion false. So, **an argument with incompatible premises is valid.**

Last modified: Mon Oct 2 07:23:49 CDT 2006

Indirect Truth Tables

Earlier, we observed that as soon as we have found a row in which the premises are true and the conclusion false, we can stop: we know at that point that the argument is invalid, and filling in further rows will not add anything to this. For example:

P	Q	S	T	P v Q	Q → S	T	|- S & T
T	T	T	T	T	T	T	T
T	T	T	F	T	T	F	F
T	T	F	T	T	F	T	F
T	T	F	F	T	F	F	F
T	F	T	T	T	T	T	T
T	F	T	F	T	T	F	F
T	F	F	T	T	T	T	F
T	F	F	F				
F	T	T	T				
F	T	T	F				
F	T	F	T				
F	T	F	F				
F	F	T	T				
F	F	T	F				
F	F	F	T				
F	F	F	F				

We can use this to develop an abbreviated truth-table test by trying to work backwards from the assumption that an argument is invalid. Taking the same example, suppose that it did have true premises and a false conclusion. We can represent this by starting out a "truth table" with the **right** side filled in first:

P	Q	S	T	P v Q	Q → S	T	|- S & T
			T	T	T	T	F

What can we add to this? First, if S & T is false and T is true, then S must be false:

P	Q	S	T	P v Q	Q → S	T	|- S & T
		F	T	T	T	T	F

Next, if Q → S is true and S is false, then Q must be false:

P	Q	S	T	P v Q	Q → S	T	|- S & T
	F	F	T	T	T	T	F

But if Q is false and P v Q is true, then P must be true:

P	Q	S	T	P v Q	Q → S	T	|- S & T
T	F	F	T	T	T	T	F

In this case, we have figured out the only possible combination of truth values for the sentence letters in these wffs that makes the conclusion false and the premises true: P = **T**, Q = **F**, S = **F**, and T = **T**.

What would happen if we tried this method on a valid argument? First, let's take note of a difference between what it takes to show that an argument is valid and what it takes to show it is invalid:

> To show that an argument is invalid, we only need to find **one row of its truth table in which the premises are true and the conclusion false.**
>
> To show that an argument is valid, we need to show that **there is no row of its truth table in which the premises are true and the conclusion false.**

The important difference is that once we have found a single row with true premises and a false conclusion, we can stop (since we know that the argument is invalid), but in order to prove that it is valid we will have to **check every row**.

As a side note, you may think that the reason proving an argument is valid requires more work than proving it is invalid is that "it is hard to prove a negative." The real reason, however, is that proving validity requires proving something **universal**: it requires proving, for **every possible combination of truth values**, that that combination does not make the premises true and the conclusion false. Consider this sentences:

> Every member of Congress is either a Democrat or a Republican.

There is nothing negative about this sentence, but in order to prove it you will need to determine, for each and every member of Congress, whether or not that person is either a Democrat or a Republican. On the other hand, to prove it false, all you need to do is find one member of Congress who is neither a Democrat nor a Republican.

Last modified: Mon Oct 2 07:36:31 CDT 2006

PREDICATE LOGIC I:

Being a Set of Supplementary Note on Predicate Logic, to accompany Allen/Hand, *Logic Primer*

Course syllabus
Part 2 | Part 3 | Part 4
Versions for browsers that can't display quantifier symbols:
Part 1-images | Part 2-images | Part 3-images | Part 4-images

Names, Predicate Expressions, and Simple Predications

Sentential logic is really the theory of the sentential connectives. Compound sentences are built up from other sentences (and ultimately from atomic sentences) by combining them with connectives. Each primitive rule of inference concerns *one* of the connectives and gives a rule for inferring a new sentence in which that connective has been either added or deleted. Truth tables for sentences and arguments are constructed using the basic truth table definitions for the connectives.

So, from the standpoint of sentential logic, atomic statements are really atomic, that is, indivisible (that's what the word 'atomic' originally meant). Sentential logic can't see any structure inside atomic statements. From this standpoint, all that an atomic statement is is a bearer of a truth value: there's nothing more that sentential logic can say about the meaning of an atomic statement.

What makes this significant for logic is that some arguments composed entirely of atomic sentences, from the standpoint of sentential logic, are still clearly valid. For instance:

> Rover is a dog
>
> All dogs are carnivores
>
> Therefore, Rover is a carnivore

This is obviously valid. However, if we translate it into the language of sentential logic, it becomes:

> P, Q |- R

And this is, equally obviously, an *invalid* sequent. What's wrong?

The problem is simply that our system of sentential logic can only catch *part* of the story of validity. Some arguments are valid because of the way they are structured with sentential connectives. Sentential logic explains why these are valid. Other arguments, such as our last example, are not valid for that reason, but they are nevertheless valid. Predicate logic is an extension of sentential logic which studies why those additional arguments are valid.

The English sentences that we treat as atomic in sentential logic obviously *do* have some internal structure. 'John likes mangoes', 'Mary took a long vacation in Flush, Kansas', 'Every single one of the many cockroaches in my kitchen is a disgusting abomination' are all atomic, and so we'd translate them as 'P', 'Q', 'R'; nevertheless, these sentences are clearly built up of smaller parts.

In fact, different atomic sentences can share some of those smaller parts. Consider these examples:

- John likes mangoes
- Mary likes mangoes
- Mary likes manatees
- John likes Mary
- John hates manatees
- Mary hates John

The words 'John', 'Mary', 'likes', 'hates', 'mangoes', 'manatees' appear in different places in these various examples. Our goal in predicate logic is to understand how these combinations are constructed and how their relationships can make an argument valid.

Names and Predicate Expressions

Here are some very simple sentences:

- Fido is a dog
- Rover is a dog
- Piewacket is a cat
- Rover is a Schnauzer
- Snidely was a rat
- Godzilla exploded

These sentences all share a common structure. Each consists of a *name*, such as 'Fido' or 'Godzilla', combined with a certain kind of phrase. As the examples show, two sentences might combine the same phrase with different names, or the same name with different phrases. We can picture this structure as follows:

'Fido'	+	'___ is a dog'	=	'Fido is a dog'
'Rover'	+	'___ is a dog'	=	'Rover is a dog'
'Rover'	+	'___ is a Schnauzer'	=	'Rover is a Schnauzer'

The expression '___ is a dog' is an expression with a blank in it. That blank is a place in which a name may be substituted to produce a sentence. We will call expressions like this, with blanks in them where names can be filled in, *predicate expressions*. A predicate expression is a sentence with a name (or names) that has been replaced by a blank (or blanks) in this way. You might put it this way: a predicate expression is something that becomes a sentence when its blank space (or spaces) is (or are) filled in with names; a name is a word that can be used to fill in a blank in a predicate expression. Finally, the sentence that results when the blank(s) in a predicate expression are filled with names is called a *simple predication*

As the last paragraph suggests, there are predicate expressions that have more than one blank space. Here are some examples:

- Rover ate Roderick
- Roderick ate Rover
- Fido loves Murgatroyd
- Godzilla will destroy North Zulch
- John knows Marcia
- John knows John
- John knows himself
- John stole Rover from Marcia

The sentence 'Rover chased Fido' contains two names, 'Rover' and 'Fido', and the predicate expression '___ chased ___'. It's important to note that there are *two* blanks in this expression. As the first two examples illustrate, it makes a difference where each name is substituted for the blank: the ingredients that go into 'Rover ate Roderick' and 'Roderick ate Rover' are the same, but the sentences are very different in meaning.

There are also predicate expressions with three blanks for names. Here are some examples:

1. George introduced Alice to Margaret
2. Achilles dragged Hector around Troy
3. Antony took Cleopatra to Rome
4. Wilfred sat between Winifred and Winston
5. Austin is between San Antonio and Waco

Predicate expressions with four or more blanks are harder to find in a natural langauge like English, but examples do exist:

1. Seven is to three as fourteen is to six
2. Godzilla divided Brooklyn between Gamela and Rodan

Translating Simple Predications

We can expand our symbolic language for logic to accommodate the structures of simple predications. We will need two new classes of symbols, corresponding to the two new types of expression we've introduced (names and predicate expressions).

Table 1: Translating Names and Predicate Expressions

To English...	*...(for example)...*	*...there correspond*	*...which look like this:*
Names	Alice, Bob, Chicago	NAMES	a,b,c, ... a_1, b_1, c_1, ... a_2, b_2, c_2, ...
Predicate Expressions	__ is hideous ___ ate ___ ___ is between ___ and ___	PREDICATE LETTERS	A, B, C, ...

We translate *names* with *lower-case letters from the beginning of the alphabet*. Just to make sure we never run out of names, we can add subscripts to these letters as much as we need.

We translate *predicate expressions* with *upper-case letters*. As before, we will allow ourselves to add subscripts to get as many as we need. We call these translations of predicate expressions *predicate letters*.

But this isn't yet quite enough. Each predicate expression has a certain number of blank places in it where names can be substituted. Where are the blanks in our symbolic translations? The answer is that we will adopt the convention that they all occur in a sequence, one after the other, to the right of the letter. (It's perhaps just a little unnatural, but that's the way logicians have got used to doing it in the last century or so.) That still leaves one question unanswered: how do we tell how many blanks go with a given predicate letter? In English, we can identify the blanks in a predicate expression just by looking. However, in our symbolic langauge, we will use a *superscript number* on a predicate letter to indicate how many blank places go with it. More formally, there are:

One-place predicate letters	A^1, B^1, C^1, D^1 ...
Two-place predicate letters	A^2, B^2, C^2, D^2 ...
Three-place predicate letters	A^3, B^3, C^3, D^3 ...
....	
n-place predicate letters	A^n, B^n, C^n, D^n ...

We'll suppose that we have predicate letters with any (finite) number of places whatever. Then, we will use a one-place predicate letter to translate a predicate expression with one blank, a two-place predicate letter to translate a predicate expression with two blanks, etc. So,

Table 2: Translating Names and Predicate Expressions (The Whole Story)

To English...	*...(for example)...*	*...there correspond*	*...which look like this:*
Names	Alice, Bob, Chicago	NAMES	a, b, c, ... a_1, b_1, c_1, ... a_2, b_2, c_2, ...
One-blank Predicate Expressions	__ is hideous	ONE-PLACE PREDICATE LETTERS	A^1, B^1, C^1, D^1 ...
Two-blank Predicate Expressions	___ ate ___	TWO-PLACE PREDICATE LETTERS	A^2, B^2, C^2, D^2 ...
Three-blank Predicate Expressions	___ is between ___ and ___	THREE-PLACE PREDICATE LETTERS	A^3, B^3, C^3, D^3 ...
...	...	...	...

In passing, we should note that there are also *zero*-place predicate letters: these are just the propositional variables we used in sentential logic.

Now we just need to explain how to combine these into translations. For one-place predicates, that's easy: we translate names with names, predicate expressions with predicate letters, and simple predications as a predicate letter followed by a name. Using these translations,

A^1	___ is abominable
B^1	___ blew up
C^1	___ cried mournfully
a	Godzilla
b	Bonzo
c	Mothra

we will get the following:

Godzilla blew up	B^1a

Bonzo is abominable	A^1b
Godzilla is abominable	A^1a
Mothra cried mournfully	C^1c
Bonzo cried mournfully	C^1b
Mothra is abominable	A^1c
Bonzo blew up	B^1b

Sentential Compounds of Simple Predications

We can combine simple predications with sentential connectives, just as we learned to do in studying sentential logic:

Mothra blew up and Godzilla cried mournfully	B^1c & C^1a
Bonzo isn't abominable	$\sim A^1b$
If Godzilla blew up, then so did Mothra	$B^1a \rightarrow B^1c$

As the last of these examples illustrates, when different components of a compound sentence share a name or a predicate expression, we often abbreviate the second occurrence: 'so did Mothra' is short for 'Mothra blew up'. In translating, we always spell out such abbreviations in full:

Abbreviated	*Spelled out in full*	*Translation*
John and Sally are aardvarks	John is an aardvark and Sally is an aardvark	A^1j & A^1s
Cyril is a snake, but Cyrus isn't	Cyril is a snake, but Cyrus isn't a snake	S^1a & $\sim S^1b$
Either Horace or Hortense did it	Either Horace did it or Hortense did it	D^1a v D^1b

Many-Place Predicates

In many-place predications (that is, those with more than one place), the different places are not equivalent, as explained above: 'Godzilla ate Brooklyn' and 'Brooklyn ate Godzilla' are not at all the same. We need a way to keep track of this difference in translating. Suppose, for example, that we use the following translations:

g	Godzilla
b	Brooklyn
A^2	___ ate ___

How will we know whether 'A^2gb' translates 'Godzilla ate Brooklyn' or 'Brooklyn ate Godzilla? We don't, until we decide which of the two places following the predicate letter corresponds to each of the blanks in the predicate expression. The easiest way to do this is to use the letters x, y, z (and others as necessary) to indicate these connections, as in the following example:

Predicate Expression	*Translation*
x ate y	A^2xy

What this means is that, in translating '___ ate ___', we will associate the first blank place in the predicate expression with the first place following the predicate letter (note where 'x' occurs) and the second blank place with the second place following the letter (note where 'y' occurs). From now on, these notes will always specify the translation of a predicate expression in this way. Here are some examples:

Predicate Expression	*Translation*
x ate y	A^2xy
x is hideous	H^1x
x gave y to z	G^3xyz
x is between y and z	B^3xyz

It's important to note that, in deciding on a translation for a predicate expression, you can assign *any order you please* to the places following the predicate letter. However, once you've done so, you have to stick with that assignment.

On Superscripts: That They May Be Omitted

We added superscripts to our predicate letters to indicate how many places each one had. That's important in defining a wff in predicate logic: an *n*-place predicate letter followed by exactly *n* names is a wff, but if it is followed by more or fewer than *n* names it's not a wff. However, in translating sentences, we can follow the practice of common among logicians of omitting the superscripts. What we do, in effect, is to assume that every predicate letter we write always has the right number of places filled after it.

Since we're going to do this, it becomes a very good idea *not* to use two predicate letters in the same translation that look just alike except for their superscripts, e.g. A^1 and A^2.

Course syllabus
Part 2 | Part 3 | Part 4

Note on Browser Compatibility

This web page uses some symbols that may not be rendered correctly by all web browsers. If you do not see what the table below indicates, use this images-only version instead.

Symbol	What you should see
→	A right arrow
↔	A double arrow
∀	An upside-down "A"
∃	A backwards "E"

Course syllabus
Part 1 | Part 3 | Part 4
Versions for browsers that can't display quantifier symbols:
Part 1-images | Part 2-images | Part 3-images | Part 4-images

The Quantifiers

In addition to names, predicate letters, and the connectives of sentential logic, our symbolic language for predicate logic contains two other types of symbol: *quantifiers* and *variables*. These always go together: in a wff, every quantifier must be associated with a variable, and every occurrence of a variable must be associated with a quantifier.

Variables are the last six letters of the alphabet, with or without subscripts (we allow ourselves to have as many of these as we need:

VARIABLES	u, v, w, x, y, z, u_1, v_1, w_1, x_1, y_1, z_1, u_2, v_2, w_2, x_2, y_2, z_2, ...

There are two quantifiers, each of which has its own name:

The Universal Quantifier	∀	(Upside-down A)
The Existential Quantifier	∃	(Backwards E)

Don't confuse variables with *names*: variables are lower-case letters from the *end* of the alphabet (u, v, w, x, y, z), while names are lower case letters from the *beginning* of the alphabet (a, b, c, d).

In a wff:

- Variables can occur wherever names can occur
- A predicate letter must *always* be followed by an appropriate number of names and/or variables
- A quantifier must *always* be followed by a variable
- A variable must *always* occur within the scope of a quantifier associated with that same variable

Some Recipes for Translating English into WFFs

The process of turning English sentences into symbolic notation is considerably more complicated than the process of translating into sentential-logic formulas. One reason the process is sometimes confusing is that there are usually a number of equally correct ways of translating the same English sentence. What I mean by "equally correct" is that the different translations are actually logically equivalent to one another, even though they may look very different. With lots of possibilities to choose from, it's sometimes hard to know where to begin.

So, here are several recipes for translating a variety of English sentence patterns into wffs. Like recipes, they usually work but sometimes fail, and if you have some experience in translating you can often find simpler translations. However, they have the advantage of being methodical.

In order to explain these recipes, I need to introduce another notion. Previously, we associated predicate letters with English predicate expressions, which are sentences in which occurrences of names have been replaced by blank places:'___ is a dog', '___ ate ___'. However, there are many uses of predicate expressions in English in which this form is not so obvious. Consider these two sentences:

1. Rover is a dog
2. Some dog is barking

In the first sentence, we can see clearly the predicate expression '___ is a dog,' with its bnalk space filled in with the name 'Rover'. However, in the second example, we have the word 'dog' but no obvious blank spaces associated with it. We explain this as follows: in sentence 2, the predicate expression '___ is a dog' is still present, but implicitly. What the sentence says is actually two things: (1) something is a dog and (2) it's barking. To put this more precisely, the sentence says:

> Of some x: x **is a** dog **and** x is barking

This rewriting of the sentence *does* contain the predicate expression '___ is a dog', in addition to the predicate expression '___ is barking.' Therefore, we can regard it as also present implicitly in the original sentence 'Some dog is barking.' In what follows, we'll do a great deal of this sort of rewriting, moving between fully-expressed predicate expressions like '___ is a dog' and more abbreviated but equivalent ones like 'dog'.

For convenience, we'll use the term *predicate* for any English-language expression that can be regarded as a predicate expression. This includes both '___ is a dog' and the shorter form 'dog.' We will also treat 'dog' and '___is a dog' as just two different forms of the *same* predicate. We also regard '___ is a dog' as its standard form, and we turn other forms into this in translation: 'dog', 'dogs', etc.

Recipe 1: Simple Existential Quantifications

FORM:	Some (predicate$_1$) (predicate$_2$)
FIRST REWRITE AS:	Some predicate$_1$=x: (x (is) predicate$_2$)
THEN REWRITE AS:	Some x: (x (is) (predicate$_1$) **and** x (is) (predicate$_2$))
TRANSLATION SCHEME:	F=predicate$_1$, G=predicate$_2$
TRANSLATION:	$\exists x(Fx\&Gx)$

At this stage, the first rewrite may seem unnecessary. Later on, we'll see that it is an important step in dealing with complex cases.

Some examples:

EXAMPLE:	Some dog is barking
FIRST REWRITE AS:	Some dog=x: (x is barking)
REWRITE AS:	Some x: (x is a dog **and** x is barking)
TRANSLATION SCHEME:	Dx=x is a dog, Gx=x is barking
TRANSLATION:	∃x(Dx&Bx)

Other forms using this recipe:

Plural:	Some dogs are barking
Indefinite article:	A dog is barking

We can also include here the quantifier words 'someone,' somebody'.

Recipe 2: Simple Universal Quantifications

FORM:	Every (predicate_1) (predicate_2)
FIRST REWRITE AS:	Every predicate_1=x: (x predicate_2
THEN REWRITE AS:	Every x: (**if** x (is) (predicate_1) **then** x (is) (predicate_2))
TRANSLATION SCHEME:	F=predicate_1, G=predicate_2
TRANSLATION:	∀x(Fx→Gx)

Some examples:

EXAMPLE:	Every dog pants
	Every dog=x: (x pants)
REWRITE AS:	**Every x: (if** x is a dog **then** x pants
TRANSLATION SCHEME:	Dx=x is a dog, Px=x pants
TRANSLATION:	∀x(Dx→Px)

Some other equivalent patterns using this recipe:

All:	All dogs pant
Each:	Each dog pants
Plurals alone:	Dogs pant (*Note: these are often ambiguous)*
Any:	Any dog pants

'Everyone,' 'anyone', 'everybody', anybody' function in the same way.

Recipe 3: Negative Universal Quantifications

FORM:	No ($predicate_1$) ($predicate_2$)
FIRST REWRITE AS:	Every $predicate_1$=x: not (x $predicate_2$)
REWRITE AS:	Every x: (**if** x (is) ($predicate_1$) **then not** x (is) ($predicate_2$))
TRANSLATION SCHEME:	F=$predicate_1$, G=$predicate_2$
TRANSLATION:	∀x(Fx→~Gx)

Here., the first rewrite lets us put the sentence into the same pattern as a simple universal quantification, and the rest of the translation proceeds as with 'every.' This shows a little bit of the point of rewriting in two stages, but its real value will become evident later.

Some examples:

EXAMPLE:	No dog sweats
FIRST REWRITE AS:	Every dog = x: not (x sweats)
THEN REWRITE AS:	Every x: (**if** x is a dog **then not** (x sweats))
TRANSLATION SCHEME:	Dx=x is a dog, Sx=x sweats
TRANSLATION:	∀x(Dx→~Sx)

Some other equivalent patterns using this recipe:

Plurals with negation: Dogs don't sweat

'Nobody' and 'no one' can be translated similarly.

Quantification and Sentential Connectives

Sentences often contain quantifiers and sentential connectives in combination. Sometimes, this is straightforward: quantifications are sentences too, and they can be combined with connectives.

One example:

Some dogs are ugly, but Fido isn't
(Some dogs are ugly) **and not**(Fido is ugly)
(**Some dog=x:(x** is ugly) and not (Fido is ugly)
(**Some x: (x is a dog and** x is ugly and not (Fido is ugly)
∃x(Dx&Ux)&~Uf
Translation scheme: Dx=x is a dog, Ux= x is ugly, f=Fido

Another example:

Some dogs are ugly, but all dogs are loveable

(Some dogs are ugly) **but** (all dogs are loveable)
(**Some dog=x: x** is ugly) **but** (**Every dog=x: x** is loveable)
(**Some x: x is a dog and** x is ugly) **but** (**Every x: if x is a dog then** x is loveable)
∃x(Dx&Ux)&∀x(Dx→Lx)

In the last example, notice that the 'x' in the left conjunct ∃(Dx&Ux) has nothing to do with the 'x' in the right conjunct ∀(Dx→Lx): these are two separate wffs. However, although it's perfectly correct to use the same variable in connection with different quantifiers in a case like this, it might be clearer to pick different variables:

∃y(Dy&Uy)&∀x(Dx→Lx)

Negated Quantifications

Since quantifications are sentences, they can be denied, just like any other sentence. With simple universal quantifications, one way to do this is particularly clear: put a 'not' at the beginning:

Not every dog is vicious
not (every dog is vicious)
~(every x: if x is a dog then x is vicious)
~∀x(Dx→Vx)

'Not all' works the same way as 'not every'. Notice that there are no parentheses around ∀x(Dx→Vx): we only need parentheses around the open form Dx→Vx.

You'll remember from sentential logic that in English, the word that makes a sentence a negation is usually somewhere in the middle of it, not at the beginning: 'John is not tall', 'Fido doesn't have fleas.' With quantifications, however, these combinations usually take on a different meaning. Consider this sentence:

Some dogs aren't ugly

You might think this is simply the negation of 'Some dogs are ugly' and translate it ~∃x(Dx&Ux). However, if these two sentences are denials of each other, then the sentence 'Some dogs are ugly and some dogs aren't ugly' should be self-contradictory, just like 'Fido is ugly and Fido isn't ugly.' But it obviously isn't: the truth is, some dogs *are* ugly and some dogs *aren't*. What went wrong? The answer is that the negation in 'Some dogs aren't ugly' is actually attached only to the second predicate. This comes out nicely in our two-stage method of rewriting:

Some dogs aren't ugly
Some dog=x: (x isn't ugly)
Some x: (x is a dog and x isn't ugly)
∃x(Dx&~Ux)

This same form of translation will help us deal with complicated predicates:

Some dogs are ugly, mean, and nasty
Some dog=x: (x is ugly, mean, and nasty)
Some dog=x: (x is ugly and x is mean and x is nasty)
Some x: (x is a dog and (x is ugly and x is mean and x is nasty))
∃x(Dx&(Ux&(Mx&Nx)))

Every dog is slimy, icky, or awful

Every dog=x:(x is slimy, icky, or awful)
Every dog=x:(x is slimy or x is icky or x is awful)
Every x: (if x is a dog then (x is slimy or x is icky or x is awful))
$\forall x(Dx \rightarrow (Sx \vee (Ix \vee Ax)))$

Sometimes, we find sentential connectives in the first predicate, as well. In these cases, the best strategy is usually to take these connectives as having larger scope than any quantifiers:

Some dogs and cats are wimps
(Some dogs are wimps) and (some cats are wimps)
(Some dog=x: (x is a wimp)) and (some cat=y: (y is a wimp))
(Some x: (x is a dog and x is a wimp) and (some y: (y is a cat and y is a wimp))
$\exists x(Dx \& Wx) \& \exists y(Cy \& Wy)$

All dogs and cats are mammals
(All dogs are mammals) and (all cats are mammals)
Every dog=x:(x is a mammal) and (every cat=y: (y is a mammal))
(every x: (if x is a dog then x is a mammal)) and (every y: (if y is a cat then y is a mammal))
$\forall x(Dx \rightarrow Mx) \& \forall y(Cy \rightarrow My)$

Be careful, however, about cases in which 'and' really indicates a two-place predicate:

All dogs and cats are enemies

This really contains the two-place predicate expression '___ and ___ are enemies.' Compare 'Fred and Ferd are brothers': this doesn't (usually) mean 'Fred is a brother and Ferd is a brother', but rather that Fred is Ferd's brother (and conversely). We'll return to these later.

A Few Words about 'Any'

Above, we listed 'any' along with 'every' and 'all' as one of the words used in forming simple universal quantifications. In combination with sentential connectives, however, 'any' and 'every' don't behave alike. Compare these two sentences:

Fido didn't eat every cat

Fido didn't eat any cat

The first sentence leaves room for a lot of cat-eating on Fido's part ("Yes, Ms. Jones, I realize Fido's been eating your cats again today, but he's getting better: after all, he didn't eat *every* cat"), but the second says that he's totally innocent. But the only difference is that the first contains 'every' where the second contains 'any'. How does that work? See here.

Course syllabus
Part 1 | Part 3 | Part 4

Course syllabus
Part 1 | Part 2 | Part 4
Versions for browsers that can't display quantifier symbols:
Part 1-images | Part 2-images | Part 3-images | Part 4-images

How to Handle Anything and Anybody

The difference between 'any' and 'every' can often be understood as a matter of *scope*. In general, 'any' tends to have *wider scope* than sentential connectives in the same sentence, while 'every' tends to have '*narrower scope*. An example will help explain. In

Fido didn't eat every cat

the sentential connective 'not' operates on the whole sentence 'Fido ate every cat'. Our analysis reflects this by analyzing that connective first, then analyzing the rest of the sentence:

Not (Fido ate every cat)
(that takes care of the 'not'; now we analyze 'Fido ate every cat')
Not (Every cat=x: (Fido ate x))
Not (Every x: (if x is a cat then Fido ate x))
$\sim \forall x(Cx \to Afx)$

However, in

Fido didn't eat any cat

the quantifier 'any' has the entire sentence in its scope, and so we analyze 'any cat' *first*, then 'not':

Any cat=x: (Fido didn't eat x)
(Now we continue with the analysis of 'any cat':)
Any x: (if x is a cat then Fido didn't eat x)
$\forall x(Cx \to \sim Afx)$

This example shows why our two-stage rewrite procedure is important: it's at the first rewrite stage that we decide whether to analyze the quantifier or the connective first. Here is a recipe rule:

When there's a choice, analyze 'any' quantifiers first

This process also works when 'any' and 'every' are combined with conditionals. These sentences clearly don't mean the same thing:

If any professor is an idiot, then Smith is an idiot
If every professor is an idiot, then Smith is an idiot

Let's apply the recipe for 'any' and 'every' to show why they're different:

If any professor is an idiot, then Smith is an idiot
Any professor=x: (if x is an idiot, then Smith is an idiot)
Any x: (if x is a professor then (if x is an idiot then Smith is an idiot))
$\forall x(Px \to (Ix \to Is))$

If every professor is an idiot, then Smith is an idiot
If (every professor is an idiot) then (Smith is an idiot)
If (every professor=x: (x is an idiot)) then (Smith is an idiot)
(just to make it clearer, let's translate the main connective at this point:)
(every professor=x: (x is an idiot))→(Smith is an idiot)
(translating the left and right sides)
(every x: if x is a professor then x is an idiot)→(Smith is an idiot)
$\forall x(Px \to Ix) \to Is$

What to Do with 'Only'

'Only' usually indicates universal quantification, but the translation is different in an important way: the predicates in the original sentence wind up in different places in the final translation than they do with 'every.' Consider this example:

Only an idiot would do that

For simplicity's sake, let's treat '___ would do that' as a one-place predicate expression here. So, we can rewrite our sentence as:

Only an idiot=x: (x would do that)

A simple way to translate this is to turn 'only' into a universal quantifier 'every' followed by the sentential connective 'only if' placed **after** the bit of sentence we're translating, like this:

Every x: (x would do that **only if** x is an idiot)

As you'll remember from sentential logic, 'P only if Q' is just P→Q. So, using the translation scheme Ix= 'x is an idiot', Dx = 'x would do that', we can then translate this as:

$\forall x(Dx \to Ix)$

Here, then, is the recipe:

FORM:	Only (predicate$_1$) (predicate$_2$)
FIRST REWRITE AS:	Only predicate$_1$=x: (x (is) predicate$_2$)
THEN REWRITE AS:	Every x: (x (is) (predicate$_2$) **only if** x (is) (predicate$_1$))
	Note the order of the predicates here!
TRANSLATION SCHEME:	F=predicate$_1$, G=predicate$_2$
TRANSLATION:	$\forall x(Gx \to Fx)$
	Note the order of the predicates here!

Compound Predicates

When several predicates occur together, treat them as a single predicate in the first rewrite and then break them up and connect with 'and':

> Some **ugly dog** is on the porch
> Some **ugly dog**=x: (x is on the porch)
> Some x: ((x **is an ugly dog**) and x is on the porch)
> Some x: ((x **is ugly and x is a dog**) and x is on the porch)
> ∃x((Ux&Dx)&Px)
>
> Every hairy cat is a loathsome nuisance
> Every hairy cat=x: (x **is a loathsome nuisance**)
> Every hairy cat=x: (x **is loathsome and x is a nuisance**)
> Every x: (if **x is a hairy cat** then (x is loathsome and x is a nuisance))
> Every x: (if (**x is hairy and x is a cat**) then (x is loathsome and x is a nuisance))
> ∀x((Hx&Cx)→(Lx&Nx))

Relative Clauses: Who, Which, What, That

Predicates with relative clauses (involving 'which', 'that', 'who', etc., can be treated in the same way as compound predicates.

> Every house that Jack built was overpriced
>
> > Every house that Jack built=x: (x was overpriced)
> >
> > Every x: (if x is a house that Jack built then x was overpriced

Once you reach this point in the analysis, just turn the relative into 'and', then put 'x' (or whatever the variable is) in the appropriate place in the predicate expression. Now, in this case, the predicate expression is really '___ built ___': the first place is filled in with the name 'Jack', but the second is open. So, we rewrite as:

> Every x: (if (x is a house **and Jack built x**) then x was overpriced

The translation, with scheme Hx='x is a house', Bxy='x built y', Ox='x was overpriced', and a='Jack', is:

> ∀x((Hx&Bax)→Ox)

Many-Place Predicates and Multiple Quantification

We've already introduced many-place predicates (predicates with more than one place), and a few previous examples contained such predicates (e.g. '___ ate ___'). With a many-place predicate, a quantifier can be attached to any place. With only a single quantifier involved, this is easy to handle: just make sure the quantifier's variable is in the right place. For these purposes, it's absolutely critical to be sure what translation scheme is being used, since only in that way can we tell which place following the predicate letter corresponds to which place in the English predicate expression. In the following two examples, we use the same translation scheme, namely

Translation scheme: Axy='x ate y', g='Godzilla'

> Godzilla ate something
> Something=x: Godzilla ate x
> Some x: Godzilla ate x

$\exists xAgx$

Something ate Godzilla
Something=x: x ate Godzilla
Some x: x ate Godzilla
$\exists xAxg$

Some further examples (Cx='x is a cat', Mx='x is a monster', Axy='x ate y', g='Godzilla'):

Godzilla ate some cats
Some cat=x: (Godzilla ate x)
Some x: (x is a cat and Godzilla ate x)
$\exists x(Cx \& Agx)$

Some monster ate Godzilla
Some monster=x: (x ate Godzilla)
Some x: (x is a monster and x ate Godzilla)
$\exists x(Mx \& Axg)$

Translation scheme for these examples: Bxy='x bit y', Cx='x is a cat', f='Fido'

No cat bit Fido
No cat=x: (x bit Fido)
Every cat=x: not (x bit Fido)
Every x: (if x is a cat then not (x bit Fido))
$\forall x(Cx \rightarrow \sim Bxf)$

Fido bit no cat
No cat=x: (Fido bit x)
Every cat=x: not (Fido bit x)
Every x: (if x is a cat then not (Fido bit x))
$\forall x(Cx \rightarrow \sim Bfx)$

Things get more complicated when we add a second quantifier. The basic rule is to analyze quantifiers one at a time, in order from left to right:

Some monster ate all the cats
Some monster=x: (x ate all the cats)
Some x: (x is a monster and (x ate all the cats))
Some x: (x is a monster and (every cat=y: (x ate y)))
(Notice what happened here: we analyzed 'x ate all the cats' *right where it is.*)
Some x: (x is a monster and (every y: (if y is a cat then x ate y)))
$\exists x(Mx \& \forall y(Cy \rightarrow Axy))$

Compare this to:

Some cat ate all the monsters
Some cat=x: (x ate all the monsters)
Some x: (x is a cat and (x ate all the monsters))
Some x: (x is a cat and (every monster=y: (x ate y)))
Some x: (x is a cat and (every y: (if y is a monster then x ate y)))
$\exists x(Cx \& \forall y(My \rightarrow Axy))$

And also to:

> Every monster ate some cats
> Every monster=x: (x ate some cats)
> Every x: (if x is a monster then (x ate some cats))
> Every x: (if x is a monster then (some cat=y: (x ate y)))
> (Compare this with the same stage in the previous two examples.)
> Every x: (if x is a monster then (some y: (y is a cat and x ate y)))
> $\forall x(Mx \rightarrow \exists y(Cy \& Axy))$

Course syllabus
Part 1 | Part 2 | Part 4

Course syllabus
Part 1 | Part 2 | Part 3
Versions for browsers that can't display quantifier symbols:
Part 1-images | Part 2-images | Part 3-images | Part 4-images

Combining Recipes for Complicated cases

Relative clauses with quantifiers

We can combine the previous methods to translate relative clauses containing quantifiers.

> Everyone who loves everyone loves George

The subject of this sentence is "everyone who loves everyone"; it contains the relative clause "who loves everyone". We can apply the preceding recipes step by step:

> Of everyone who loves everyone=x: x loves George
>
> Of every **x: if x is a person who loves everyone then** x loves George
>
> Of every x: if (x is a person **and x** loves everyone) then x loves George
>
> Of every x: if (x is a person and **of everyone=y: x loves y**) then x loves George
>
> Of every x: if (x is a person and of every **y: if y is a person then** x loves y) then x loves George
>
> ∀x((Px& ∀y(Py→Lxy))→Lxa)

Further examples:

> A dog that bit Fluffy bit me
> Of some dog that bit Fluffy=x: x bit me
> Of some x: x is a dog that bit Fluffy and x bit me
> Of some x: x is a dog and x bit Fluffy and x bit me
> ∃x(Dx&Bxa&Bxb)
>
> No dog that bit Fluffy bit me
> Of no dog that bit Fluffy=x: x bit me
> Of every dog that bit Fluffy=x: not(x bit me)
> Of every x: if (x is a dog that bit Fluffy) then not (x bit me)
> Of every x: if (x is a dog and x bit Fluffy) then not (x bit me)
> ∀x((Dx&Bxa)→~Bxb)
>
> I bite every dog that bites me
> Of every dog that bites me=x: I bite x
> Of every x: if (x is a dog that bites me) then I bite x
> Of every x: if (x is a dog and x bites me) then I bite x
> ∀x((Dx&Bxb)→Bbx)
>
> Every dog that bit a cat was bitten by a rat
> Of every dog that bit a cat=x: x was bitten by a rat

Of every x: if x is a dog that bit a cat then x was bitten by a rat
Of every x: if (x is a dog and of some (a) cat=y: x bit y) then x was bitten by a rat
Of every x: if (x is a dog and (of some y: y is a cat and x bit y)) then x was bitten by a rat
Of every x: if (x is a dog and (of some y: y is a cat and x bit y)) then of some rat = z: x was bitten by z
Of every x: if (x is a dog and (of some y: y is a cat and x bit y)) then (of some z: z is a rat and x was bitten by z)
$\forall x ((Dx\&\exists y (Cy\&Bxy))\rightarrow \exists z (Rz\&Bzx))$
(*Note that Bxy="x bit y" and also "y was bitten by x"*)

A dog that bit a cat bit a man that wore a hat
Of some dog that bit a cat =x: x bit a man that wore a hat
Of some x: x is a dog that bit a cat and x bit a man that wore a hat
Of some x: (x is a dog and x bit a cat) and (x bit a man that wore a hat)
Of some x: (x is a dog and of some cat=y: x bit y) and (x bit a man that wore a hat)
Of some x: (x is a dog and of some y: y is a cat and x bit y) and (x bit a man that wore a hat)
Of some x: (x is a dog and of some y: y is a cat and x bit y) and (of some man that wore a hat=z: x bit z)
Of some x: (x is a dog and of some y: y is a cat and x bit y) and (of some z: z is a man that wore a hat and x bit z)
Of some x: (x is a dog and of some y: y is a cat and x bit y) and (of some z: z is a man and z wore a hat and x bit z)
Of some x: (x is a dog and of some y: y is a cat and x bit y) and (of some z: z is a man and (of some hat=w: z wore w) and x bit z)
Of some x: (x is a dog and of some y: y is a cat and x bit y) and (of some z: z is a man and (of some w: w is a hat and z wore w) and x bit z)
$\exists x((Dx\& \exists y(Cy\& Bxy))\& \exists z(Mz\& \exists w((Hw\&Wzw)\&Bxz)))$

More Horrifying Examples

In English, it's possible (and pretty common) to drop the relative pronoun or adjective in a relative clause: we can say "I saw the dog John bought" instead of "I saw the dog that John bought", for instance. When you see these, it may help to supply the appropriate relative pronoun or adjective ("that" will often do).

Fluffy ate every mouse Snookums didn't eat
(*making it more explicit:*) Fluffy ate every mouse **that**Snookums didn't eat
Of every mouse **that**Snookums didn't eat=x: Fluffy ate x
Of every x: if x is a mouse **that**Snookums didn't eat then Fluffy ate x
Of every x: if x is a mouse and Snookums didn't eat x then Fluffy ate x
$\forall x ((Mx\&\sim Aax)\rightarrow Abx)$

Sentences built like this can get remarkably involved:

A dog bit a cat: $\exists x(Dx\& \exists y(Cy\&Bxy))$
A dog a rat bit bit a cat: $\exists x(Dx\& \exists y((Ry\&Byx)\& \exists z(Cz\&Bxz)))$
A dog a rat a snake bit bit bit a cat: $\exists x(Dx\& \exists y(Ry\& \exists z((Sz\&Bzy)\&Byx)\& \exists w(Cw\&Bxw)))$
A dog a rat a cat a baby bit bit bit expired

Plurals in constructions like these often function as universal quantifiers:

People like dogs

People dogs like like dogs

Conjunction and disjunction inside predication

Some peculiar things can happen with conjunction and disjunction in combination with predication and quantifiers. First, we need to notice that "and" isn't always a conjunction: sometimes, it is used to hook together the two names that go with a two-place predicate. Compare these two sentences:

1. Fido and Rover are schnauzers
2. Fido and Rover are littermates

The first sentence is a conjunction. You can see this by noting that it means exactly the same thing as

Fido is a schnauzer and Rover is a schnauzer

The second sentence, however, is just a simple predication with a two-place predicate "__and__are littermates". If you rewrite it as

Fido is a littermate and Rover is a littermate,

the result no longer means the same thing as the original. Other examples:

John and Mary are married [i.e. to each other]
Alice and Anita are sisters
Godzilla and Mothra were engaged in mortal combat
Bush and Clinton were opponents

This gives us a useful test: if we can rewrite a sentence containing a compound subject (two names connected with "and") as an explicit conjunction without changing its meaning, then the original sentence was a conjunction; if rewriting it changes its meaning, then the original sentence involved a two-place predicate.

When we apply this to sentences with quantifiers, the same test will continue to work. First, an example involving conjunction:

All dogs and cats are mammals
=All dogs are mammals and all cats are mammals
=$\forall x(Dx \rightarrow Mx) \& \forall x(Cx \rightarrow Mx)$

Next, an example involving a two-place predicate:

All dogs and cats are enemies
=All dogs are enemies and all cats are enemies? NO

The rewriting test shows that this sentence is not a conjunction. But how do we translate it? We need to introduce a new technique. The subject of this sentence, logically speaking, is actually *all dog-cat pairs*, that is, every pair xy where x is a dog and y is a cat. Let's express this as follows:

Of every dog and cat=xy: x and y are enemies
Of (every dog=x, every cat=y): x and y are enemies

We can then finish translating this using our usual techniques:

Of every x: if x is a dog then of every y: if y is a cat then x and y are enemies

$\forall x(Dx \rightarrow \forall y(Cy \rightarrow Exy))$

We can avail ourselves of a little logical magic to make this simpler. It just happens to be the case that

$\forall x(Dx \rightarrow \forall y(Cy \rightarrow Exy))$

is equivalent to

$\forall x \forall y(Dx \rightarrow (Cy \rightarrow Exy))$

which in turn is equivalent to

$\forall x \forall y(Dx \& Cy \rightarrow Exy)$

which you could also write as

$\forall xy(Dx \& Cy \rightarrow Exy)$

So, we have a recipe for translating sentences of the form "All Fs and Gs are Hs" when 'H' is really a two-place predicate.

Similar methods work with existential quantifiers:

Some dogs and cats are house pets
=Some dogs are house pets and some cats are house pets
of some x(x is a dog and x is a house pet) and of some y(y is a cat and y is a house pet)
$\exists x(Dx \& Hx) \& \exists y(Cy \& Hy)$

Some dogs and cats are friends
=Some dogs are friends and some cats are friends? NO
Of some dog and cat=xy: x and y are friends
Of some x: x is a dog and of some y: y is a cat and x and y are friends
$\exists x(Dx \& \exists y(Cy \& Fxy))$

This also can be made a little neater with some logical tricks: it's equivalent to

$\exists xy(Dx \& Cy \& Fxy)$

Some Logician's Magic: Getting Quantifiers Out Front

The recipes presented in these pages are intended to be practical ways to solve the problem of translating English sentences into predicate logic. They are methodical, and so they are somewhat mechanical. They also may sometimes produce somewhat cumbersome results. On the assumption that a plodding but reliable method is safer to follow than the method of relying on flashes of brilliant insight (which are just fine until you discover that you can't have them at will), I recommend them. I also think they allow you to get a better understanding of how quantifiers work.

However, most logicians will give rather different translations from the more complex ones here. Often, they're just as long but have a difference in order (for one thing, the quantifiers are likely to come in strings at the beginning); in some cases, they may be much shorter. These different translations will be equivalent to the ones reached methodically here. Generally, it is a complex business to determine whether two formulas are equivalent: to prove equivalence, you would need to show that each could be deduced from the other.

However, it might be useful to take note here of some equivalences that hold generally. In these examples, Φ and Ψ are wffs, not open forms.

This...	**is equivalent to...**
∀x(Fx→Hx)&∀x(Gx→Hx)	∀x((Fx v Gx)→Hx)
∀(Fx→∀y(Gy→Hxy))	∀xy((Fx & Gy)→Hxy)
∀x(Fx→Φ)	∃xFx →Φ
∃x(Fx→Φ)	∀xFx →Φ

Course syllabus
Part 1 | Part 2 | Part 3

Exercise 3.2: Translations into Predicate Logic WFFS

For a version of this page with images in place of the characters for the quantifiers, go here

These translations follow the processes suggested here, here, and here; answers will not always look like those in Allen/Hand, but they are equivalent.

1. All dogs are mammals

SENTENCE:	All dogs are mammals
FIRST REWRITE AS:	All dogs=*x*: (*x* is a mammal)
REWRITE AS:	Every *x*: (if *x* is a dog then *x* is a mammal)
TRANSLATION SCHEME:	D*x*=*x* is a dog, M*x*=*x* is a mammal
TRANSLATION:	$\forall x(Dx \rightarrow Mx)$

2. Some sharks are ovoviviparous

SENTENCE:	Some sharks are ovoviviparous
FIRST REWRITE AS:	Some sharks=*x*: (*x* is ovoviviparous)
REWRITE AS:	Some *x*: (*x* is a shark and *x* is ovoviviparous)
TRANSLATION SCHEME:	S*x*=*x* is a shark, O*x*=*x* is ovoviviparous
TRANSLATION:	$\exists x(Sx \& Ox)$

3. No fishes are endothermic

SENTENCE:	No fishes are endothermic
FIRST REWRITE AS:	No fishes=*x*: (*x* is endothermic)
REWRITE AS:	Every fish=*x*: not (*x* is endothermic)
REWRITE AS:	Every *x*: (if *x* is a fish then not (*x* is endothermic))
TRANSLATION SCHEME:	F*x*=*x* is a fish, E*x*=*x* is endothermic
TRANSLATION:	$\forall x(Fx \rightarrow \sim Ex)$

4. Not all fishes are pelagic

In this sentence, note that the sentential connective 'not' has (as usual) wider scope than the 'all': so, we analyze it first.

EXERCISE 4:	Not all fishes are pelagic

Sentential connective first:	Not (all fishes are pelagic)
REWRITE AS:	Not (all fishes=*x*: (*x* is pelagic)
REWRITE AS:	Not (Every *x*: (if *x* is a fish then *x* is pelagic))
TRANSLATION SCHEME:	F*x*=*x* is a fish, P*x*=*x* is pelagic
TRANSLATION:	~∀*x*(F*x*→P*x*)

5. Reptiles and amphibians are not endothermic

This sentence is sententially compound. Once we spell out the two conjuncts,we have a second problem: how do we interpret 'Reptiles are not endothermic'? The text translates it as equivalent to 'No reptiles are endothermic'; this is one possibility.

SENTENCE:	Reptiles and amphibians are not endothermic
Sentential connective first:	Reptiles are not endothermic and amphibians are not endothermic
REWRITE (TWO AT ONCE):	(Reptiles=*x*: (*x* is not endothermic)) and (amphibians=*y*: (*y* is not endothermic))
REWRITE (taking plurals as universals):	(Every *x*: (if *x* is a reptile then *x* is not endothermic)) and (every *y*: (if *y* is an amphibian then *y* is not endothermic))
TRANSLATION SCHEME:	R*x*=*x* is a reptile, A*x*=*x* is an amphibian, E*x*=*x* is endothermic
TRANSLATION:	∀*x*(R*x*→~E*x*)&∀*y*(A*y*→~E*y*)

Note that we could have used '*x*' as the variable in both sides; however, it's a good idea always to introduce a new variable to avoid possible mistakes.

6. Some primates and rodents are arboreal

This involves a sentential connective as well, but here the quantifier is explicit:

SENTENCE:	Some primates and rodents are arboreal
Sentential connective first:	(Some primates are arboreal) and (some rodents are arboreal)
REWRITE (TWO AT ONCE):	(Some primates=*x*: (*x* is arboreal)) and (some rodents=*y*: (*y* is arboreal))
REWRITE:	((Some *x*: (*x* is a primate and (*x* is arboreal))) and (some *y*: (*y* is a rodent and (*y* is arboreal)))
TRANSLATION SCHEME:	P*x*=*x* is a primate, R*x*=*x* is a rodent, A*x*=*x* is arboreal
TRANSLATION:	∃*x*(P*x*&A*x*)&∃*x*(R*x*&A*x*)

7. Only lagomorphs gnaw

The next example involves a new English expression: 'only'. We need a new recipe for handling this. To get one, notice that 'Only Fs are Gs' can be rewritten as something like 'It's a G only if it's an F'; we can then translate the 'only if' as '→'. In more detail, and with this example:

SENTENCE:	Only lagomorphs gnaw
FIRST REWRITE AS:	Only lagomorphs=*x*: (*x* gnaws)
REWRITE AS:	**Every *x***: (*x* gnaws **only if** *x* is a lagomorph)
Notice how this works: we turn 'only' into 'every' and then put 'only if' + (the first predicate) ***at the end*** *of the translation.*	
TRANSLATION SCHEME:	G*x*=*x* gnaws, L*x*=*x* is a lagomorph
TRANSLATION:	$\forall x(Gx \rightarrow Lx)$

8. Among spiders, only tarantulas and black widows are poisonous

The next example adds two separate complications to the last one. We could translate 'Only tarantulas are poisonous' in the same way as the last example. 'Only tarantulas and black widows are poisonous', however, is more difficult. You might think this is sententially compound, that is:

> Only tarantulas are poisonous and only black widows are poisonous

But that's obviously not what the sentence means. So, you might think that the 'and' simply forms part of the first predicate expression in this manner:

> It's poisonous only if it's a tarantula and it's a black widow

Unfortunately, that can't be right, since it would mean that the only poisonous things are those things which are simultaneously tarantulas and black widows, and nothing is (tarantulas aren't black widows, and black widows aren't tarantulas). So what does it mean?

The answer, alas, is that in this case we have to rewrite 'and' as 'or', in the following way:

> It's poisonous only if it's a tarantula or it's a black widow

So, rewriting this to make the quantifier explicit, our sentence is:

> Every *x*: (*x* is poisonous only if *x* is a tarantula or *x* is a black widow)

Taking care to put the parentheses in the right place, this becomes:

> $\forall x(Px \rightarrow (Tx \vee Bx))$

But that's still not the original sentence: number 8 begins with the phrase 'among spiders'. How do we handle this? Take a simpler case:

> Among spiders, only tarantulas eat gerbils

(For simplicity's sake, let's treat '___ eats gerbils' as a one-place predicate here.) The phrase 'among spiders' here has the effect of saying 'we're talking about (all) spiders'. We can express that by treating it just like 'every spider'. We can do that in two stages, first analyzing 'only;

Among spiders, (every *x*: (*x* eats gerbils only if *x* is a tarantula))

Then we translate 'among spiders' to 'if *x* is a spider then' and put this **after** the quantifier:

Every *x*: (if *x* is a spider then (*x* eats gerbils only if *x* is a tarantula))

Now let's try to combine these recipes for dealing with the original sentence:

Among spiders, only tarantulas and black widows are poisonous
Among spiders, (only (tarantulas or black widows)=*x*: (*x* is poisonous))
Among spiders, (every *x*: (*x* is poisonous only if (*x* is a tarantula or black widow)))
Among spiders, (every *x*: (*x* is poisonous only if (*x* is a tarantula or *x* is a black widow)))
Every *x*: (if *x* is a spider then (*x* is poisonous only if (*x* is a tarantula or *x* is a black widow)))
$\forall x(Sx \rightarrow (Px \rightarrow (Tx \lor Bx)))$

(By the way, it's not true: tarantulas actually make nice pets, and anyone who's been bitten by a brown recluse knows there are other poisonous spiders.)

9. All and only marsupials have pouches

This actually can be handled using the recipes we've seen so far. The 'and' is a sentential connective, so this is

All marsupials have pouches and only marsupials have pouches

So, doing the translations left and right, we have

(Every *x*: (if *x* is a marsupial then *x* has a pouch)) and (every *x*: (*x* has a pouch only if *x* is a marsupial))

And so:

$\forall x(Mx \rightarrow Px)$ & $\forall x(Px \rightarrow Mx)$

Some of you are already complaining that this violates the rule proposed earlier of using different variables for different quantifiers. However, here we have a point to make. If we were to combine the open forms '$Mx \rightarrow Px$' and '$Px \rightarrow Mx$' with '&', the end result would be equivalent to '$Mx \leftrightarrow Px$'. It follows from that (and a little more logic) that our last translation is equivalent to:

$\forall x(Mx \leftrightarrow Px)$

So, we could also use this as a recipe for translating 'all and only'.

10. No fish have wings unless they belong to the family *Exocoetidae*

The first thing to notice about this is that it is *not* sententially compound: if you divide it at the connective 'unless', then the right half,

they belong to the family *Exocoetidae*,

isn't actually a sentence because it contains 'they' (with no indication, in the sentence, as to who or what 'they' are). This is an extension of the test we used in sentential logic to determine what the main connective is. Note that 'they' is obviously connected to the quantifiying expression 'no fish'. When we're checking to see whether a sentence is sententially compound, we cannot separate a pronoun from the quantifying expression it's connected with (the result would be the English equivalent of an open formula).

Unfortunately, there's another problem in this sentence: 'unless'. We might try to translate the sentence this way:

> No fish=*x*: (*x* has wings unless *x* belong to the family *Exocoetidae*)

When we apply the recipe for 'no' to this, you might think we would get this:

> Every fish=*x*: not (*x* has wings unless *x* belong to the family *Exocoetidae*)

And that isn't what we want (if you finish the translation, you'll see that this comes out as equivalent to 'No fish has wings and no fish belongs to the family *Exocoetidae*'). But a small change will make it work: the 'not' only attaches to the part of the sentence *before* the connective 'unless':

> No fish=*x*: (*x* has wings) unless *x* belongs to the family *Exocoetidae*
> Every fish=*x*: (not (*x* has wings) unless *x* belongs to the family *Exocoetidae*)
> Every *x*: (if *x* is a fish then (not (*x* has wings) unless *x* belongs to the family *Exocoetidae*)))
> ∀*x*(F*x*→(~W*x* v E*x*))

11. Some organisms are chordates and some organisms are molluscs, but nothing is both a chordate and a mollusc

This is long but straightforward. The only mildly novel point is the compound predicate expression '__ is both a chordate and a mollusc', which we disassemble into two predicate expressions and 'and'. First, the sentential-logic structure:

> ((Some organisms are chordates) and (some organisms are molluscs,)) but (nothing is both a chordate and a mollusc)

The two on the left are easy, so let's get them out of the way in one step:

> ((Some *x*: (*x* is an organism and *x* is a chordate)) and (some y: (*y* is an organism and *y* is a mollusc))) but (nothing is both a chordate and a mollusc)

The right hand side, then, is analyzed as:

> Nothing=*x*: (*x* is both a chordate and a mollusc)

But the complex predicate becomes

> Nothing=*x*: (both *x* is a chordate and *x* is a mollusc)

And so this is:

> Every *x*: not (both *x* is a chordate and *x* is a mollusc)

Combining the two, we have:

> ((Some *x*: (*x* is an organism and *x* is a chordate)) and (some *y*: (*y* is an organism and *y* is a mollusc))) but (every z: not (both z is a chordate and z is a mollusc))
>
> (∃*x*(O*x*&C*x*)&∃*y*(O*y*&M*y*))&∀z~(Cz&Mz)

12. None but phylogenists are intelligent

Treat 'none but' just like 'only':

> None but phylogenists=*x*: (*x* is intelligent)
> Only phylogenists=*x*: (*x* is intelligent)
> Every *x*: (*x* is intelligent only if *x* is a phylogenist)
> $\forall x(Ix \to Px)$

13. Animals behave normally if not watched

Notice that this is in effect an abbreviation: it's equivalent to

> Animals behave normally if they are not watched

If we think about the meaning of this sentence, it's clear that it is a universal quantification. We can rewrite it as:

> Every animal behaves normally if it is not watched

The 'it' is connected with 'every animal', so this is not sententially compound. We can translate by substituting *x* for *both* places in the original sentence:

> Every animal=*x*: (*x* behaves normally if *x* is not watched)

The rest of the translation is straightforward:

> Every *x*: (if *x* is an animal then (*x* behaves normally if *x* is not watched))
> $\forall x(Ax \to ({\sim}Wx \to Nx))$

14. Animals behave normally only if not watched

Just as before:

> Animals behave normally only if they are not watched
> Every animal behaves normally only if it is not watched
> Every animal=*x*: (*x* behaves normally only if *x* is not watched)
> Every *x*: (if *x* is an animal then (*x* behaves normally only if *x* is not watched))
> $\forall x(Ax \to (Nx \to {\sim}Wx))$

15. Some sharks are pelagic fish, but not all pelagic fish are sharks

Note that '___ is a pelagic fish' = '___ is pelagic and ___ is a fish':

> (Some sharks are pelagic fish,) but (not all pelagic fish are sharks)
> (Some shark=*x*: (*x* is pelagic and *x* is a fish,)) but (not (all pelagic fish=y: *y* is a shark))

At this point, the right side still contains 'pelagic fish'. However, we just apply the recipes: first analyze the 'all', then analyze the compound predicate:

> every *y*: (if *y* is a pelagic fish then (*y* is a shark))
> every *y*: (if *y* is pelagic and *y* is a fish) then (*y* is a shark))

So, putting this back into the rewrite of the whole sentence:

> (Some shark=*x*: (*x* is pelagic and *x* is a fish,)) but (not (every *y*: (if *y* is pelagic and *y* is a fish) then

(*y* is a shark)))

∃*x*(S*x*&(P*x*&F*x*))&~∀*y*((P*x*&F*x*)→S*x*))

16. If Shamu is a whale and all whales are mammals, then Shamu is a mammal

Sentential structure first:

If (Shamu is a whale and all whales are mammals), then (Shamu is a mammal)

With the scheme a = 'Shamu', W*x*='*x* is a whale', M*x*='*x* is a mammal', the three parts of this are:

Shamu is a whale: Wa
All whales are mammals: ∀*x*(W*x*→M*x*)
Shamu is a mammal: Ma

Putting these back into the original:

If (Wa and ∀*x*(W*x*→M*x*)) then Ma
(Wa & ∀*x*(W*x*→M*x*))→ Ma

17. No sparrow builds a nest unless it has a mate

Just like example 10:

No sparrow=*x*: (*x* builds a nest unless *x* has a mate)
Every sparrow=*x*: (not (*x* builds a nest) unless *x* has a mate))
Every *x*: (if *x* is a sparrow then (not (*x* builds a nest) unless *x* has a mate))
∀*x*(S*x*→(~N*x* v M*x*))

18. No organism that is edentulous is a predator

To do this one, we'll need to use the recipe for relative clauses:

No organism that is edentulous=*x*: (*x* is a predator)
Every organism that is edentulous=*x*: not (*x* is a predator)
Every *x*: (if (*x* is an organism that is edentulous) then not (*x* is a predator))
Every *x*: (if (*x* is an organism and *x* is edentulous) then not (*x* is a predator))
∀*x*((O*x*&E*x*)→~P*x*)

19. All predators are not herbivorous

This has two possible interpretations, according to whether you take the 'not' to have wider or narrower scope than the 'all'. On the first interpetation (which I think is more natural), it is:

Not (all predators are herbivorous)

And that is:

~∀*x*(P*x*→H*x*)

On the other interpretation, it could be rewritten thus:

All predators=*x*: (*x* is not herbivorous)

That is,

∀*x*(P*x*→~H*x*)

20. Not all predators are carnivorous

This one isn't ambiguous at all:

Not (all predators are carnivorous
Not (all predators=*x*:(*x* is carnivorous))
Not (every *x*: (if *x* is a predator then *x* is carnivorous))
~∀*x*(P*x*→H*x*)

21. A mammal with wings is a bat

22. A mammal with wings is flying

These two examples illustrate another source of ambiguity: predicates with nothing but an indefinite article attached can be either universal or existential. Number 21 is a universal case which we could rewrite as:

Every mammal with wings is a bat
Every mammal with wings=*x*: (*x* is a bat)
Every *x*: (if *x* is a mammal with wings then *x* is a bat)

'With wings' amounts to 'has wings', so we analyze the first predicate as compound:

Every *x*: (if (*x* is a mammal and *x* has wings) then *x* is a bat)
∀*x*((M*x*&W*x*)→B*x*)

Number 22, by contrast, is really an existential quantification:

Some mammal with wings is flying
Some mammal with wings=*x*: (*x* is flying)
Some *x*: (*x* is a mammal with wings and (*x* is flying))
Some *x*: ((*x* is a mammal and *x* has wings) and (*x* is flying))
∃*x*((M*x*&W*x*)&F*x*)

Exercise 3.2, 23-47 | Exercise 3.2, 48-57

Allen/Hand, Section 3.3: Instances, Open Formulas and Quantifier Scopes

This page uses html math characters that may not render properly in some browsers. For a version that uses graphic images instead, go here instead.

Proofs in Predicate Logic

We want to extend our system for proofs to include predicate logic sequents. To do that, we will add two pairs of rules: Elimination and Introduction rules for each of the quantifiers. These require some explanation in advance. First, we need a definition of an **open formula**:

> An **OPEN FORMULA** is the result of replacing at least one occurrence of a name in a wff by a new variable (one not already occurring in the wff)

Examples:

WFF	Replace...	Open Formula
Ab	b with x	Ax
~Fcd	d with y	~Fcy
(Aac&Bad)	a with x, once	(Axc &Bad)
(∀xFx →Gab)	b with y	(∀xFx →Gay)
∃y(Fy↔Hya)	a with x	∃y(Fy↔Hyx)

Note that **an open formula is not a wff**.

Now we need to define precisely the **scope** of an occurrence of a quantifier. Intuitively, the scope of a quantifier is the part of the sentence that the quantifier works on. Quantifiers always take the **smallest possible scope**. To put this formally:

> The **SCOPE** of a quantifier in a given formula is the shortest open formula to the right of the quantifier.

Some examples:

WFF	Scope of indicated quantifier
(∀xFx→Gab)	Fx
(Fa&∀xFx)	Fx
∃x(Fx→Gab)	(Fx↔Gab)

Instances of Universal and Existential Quantifications

> An **INSTANCE** of a universal or existential quantification is a wff that results by performing the following two steps on it:

1. Remove the initial quantifier
2. Replace all occurrences of the free variable in the resulting open formula with a name (using the same name for all instances)

Note these points:

- Only universal and existential quantifications have instances.
- You must substitute the same name for all the occurences of the free variable.
- You **do not** need to choose a name that does not already occur in the formula: any name can be used.

Universalizations and Existentializations of wffs

A **UNIVERSALIZATION** of a wff is the result of performing the following two steps on it:

1. Replace one or more occurrences of a name in it by a new variable (that is, a variable not already occurring in it).
2. Prefix a universal quantifier containing that variable to the resulting open formula.

An **EXISTENTIALIZATION** of a wff is the result of performing the following two steps on it:

1. Replace one or more occurrences of a name in it by a new variable (that is, a variable not already occurring in it).
2. Prefix an existential quantifier containing that variable to the resulting open formula.

Points to note:

-

Introduction and Elimination Rules for Quantifiers

Our system will have rules of **Universal Elimination (∀E), Universal Introduction (∀I)**, **Existential Elmination (∃E)**, and **Existential Introduction (∃I)**. As it happens,

one rule of each pair is quite simple to understand and to use, while the other member of each pair is more difficult. Let's begin with the two easy ones, which ar ∀E and ∃I.

Universal Elimination

UNIVERSAL ELIM (∀E): Given a universally quantified sentence (at line *m*), conclude any instance of it.
ASSUMPTION SET: the same as the line containing the universal quantification.

If ∀x;Φ is a universally quantified wff, then Φ is an open formula with x as an unbound variable. If α is a name, then let's use Φ(α) to mean "the result of replacing x with the name α in Φ". Then, we can summarize this as:

k_1 ... k_i	(*m*)	∀xΦ		
	(.)	.	.	
k_1 ... k_i	(*n*)	Φ(x=α)	*m*	∀E

Examples:

1	(1)	∀x(Fx→Gx)

(.)	.	.
1 (*n*)	(Fb→Gb)	1 ∀E

In this example, Φ = (Fx→Gx) and α = b

The next example is a proof of the sequent Fc, ∀y(Fy→Gy) ⊦ Gc :

1	(1) Fc	A
2	(2) ∀x(Fx→Gx)	A
2	(3) (Fc→Gc)	2 ∀E
1,2	(4) Gc	1,3 →E

Note these points:

1. This rule can only be applied to universal quantifications, that is, wffs that begin with a universal quantifier having the entire remainder of the wff as its scope.
2. You can use any name whatever in producing the instance: it does not have to be a name that already occurs somewhere in the proof.
3. You can apply ∀E repeatedly to the same wff: universal quantifications do not get "used up" or discharged when you use them once.

THINGS NOT TO DO. The following are all mistaken applications of ∀E:

1 (1) ∀xFx↔∀yGy		
1 (2) Fa↔∀yGy	2 ∀E	WRONG: line 1 is not a universal quantification
1 (1) ∀x(Fx&Gx)		
1 (2) Fa&Gb	2 ∀E	WRONG: the same name must be substituted for all occurrences of the unbound variable

Existential Introduction

> **EXISTENTIAL INTRO (∃I)**: Given a sentence (at line *m* containing any occurrence of a name, conclude any existentialization of that sentence with respect to that name.
> **ASSUMPTION SET:** the same as the original line.

We will use Φ(α) to mean 'a wff that contains the name α'. We can then use Φ(x) to mean 'the universalization of Φ(α) with respect to α'. Note that x has to be a variable that does not already occur in Φ(α).

$k_1 \ldots k_i$ (*m*)	Φ(α)	
(.)	.	.
$k_1 \ldots k_i$ (*n*)	∃xΦ(x)	*m* ∃I

Example: a proof of the sequent (Ha ∨ ~Ga) ⊦ ∃x(Hx ∨ ~Gx) :

1 (1) Ha ∨ ~Ga	A

1	(2)	∃x(Hx ∨ ~Gx)	1 ∃I

Example: a proof of the sequent (Ha ∨ ~Ga) ⊦ ∃x(Hx ∨ ~Ga) :

1	(1)	Ha ∨ ~Ga	A
1	(2)	∃x(Hx ∨ ~Ga)	1 ∃I

These are both correct: nothing says that we have to replace **every** occurrence of the name with a variable in making an existentialization of it.

Example using both ∀E and ∃I: a proof of the sequent Fa, ∃xFx→∀yHy ⊦ Hb :

1	(1)	Fa	A
2	(2)	∃xFx→∀yHy	A
1	(3)	∃xFx	1 ∃I
1,2	(4)	∀yHy	2,3 →E
1,2	(5)	Hb	4 ∀E

Universal Introduction

∀I is a simple rule, with a catch: it has a restriction on when it can be used. The restriction is absolutely critical:

> **UNIVERSAL INTRO (∀i)**: Given a sentence (at line *m*) containing at least one occurrence of a name, conclude a universalization of the with respect to the name
> **ASSUMPTION SET:** the same as the original line.
> **RESTRICTION:** the name α must not occur in any assumption in the assumption set for line *m*.

The crucial part of this rule is the restriction. We can explain it as follows. If we are able to get, as a line of a proof, a sentence containing the name α from premises that do not contain α, then it must not be anything pecular about α that made this possible: we could have done the same proof with any other name besides α. In that case, it's reasonable to say that we could have come to the same conclusion about any name whatsoever. And that, in effect, is what a universal quantifier means.

Existential Elimination

This rule, unfortunately, is considerably more complicated to state than the other quantifier rules. Like ∀I, it also has a restriction. However, in this case the restriction is rather complicated as well.

> **EXISTENTIAL ELIM (∃E)**Given the following:
>
> 1. a sentence (at line *m*),
> 2. an existentially quantified sentence (at line *k*), and
> 3. an assumption (at line *i*) that is an instance of the sentence at line *k*,
>
> conclude the sentence at line *m* again.
> **ASSUMPTION SET:** all the assumptions on line *m* and *k*, **with the exception of** *i*.
> **RESTRICTION:** the instantial name at line *i* does not occur in any of the following:

1. the sentence at line *m*,
2. the sentence at line *k*, or
3. any sentence in the assumption sets of lines *m* or *k* except for line *i*.

This is much more complicated than any of the other rules. To explain it, we can first consider an argument in English:

> Someone stole a huge bag of cash from the bank yesterday. We don't know who it was, so let's call that person "Rob". So, Rob stole a huge bag of cash from the bank yesterday. Now, unfortunately for Rob, this huge bag of cash was booby-trapped with a can of malodorous, fluorescent dye designed to explode when the bag is opened and cover whoever opened it with evil-smelling fluorescent dye. Rob is bound to have opened the bag, so Rob is now covered with stinky day-glow stuff. Therefore, someone is now covered with bad-smelling fluorescent dye.

In this argument, "Rob" functions as a made-up name with no other purpose than allowing me to reason about whoever it was that stole a huge bag of cash from the bank yesterday. The conclusion of this argument doesn't mention Rob and only talks about *someone*. Intuitively, this looks like a valid argument. We don't draw any conclusions about "Rob" except those that follow from the assumption that he stole a huge bag of cash, etc.

Here's another example:

> There is no largest prime number. For suppose that there is a largest prime number. Call this number *N*. Now consider the number *N!*, which is the product of all the numbers up to *N*: *N! = 2 × 3 × ... × (N - 1) × N*. Add one to this number. Either *N!+1* is prime or it is not. If it is prime, then it is a prime number larger than *N*, and so *N* is not the largest prime number. So, suppose that *N* is not a prime number. Then it must be divisible by some number *K*. Now, for any number *M* not larger than *N*, *N!+1* is one greater than a multiple of *M*. But then *M* is not a divisor or *N! + 1*. So, *K* must be different from every number less than *N*. In that case, *K* is greater than *N*, and again *N* is not the greatest prime number. Therefore, there is no greatest prime number

In the second example, *N* is used as an arbitrary name for 'the largest prime number.' In fact, the entire argument shows that there is no such number. However, within the context of the argument, once we have assumed (for the sake of *RAA*, you could say) that there is one, we can then give it a name and draw some conclusions about it, using only the assumption that it is the largest prime number.

Now, here is how the rule of ∃E embodies this line of reasoning. Suppose that we have, as a line of a proof, an existential sentence ∃xΦ(x). This sentence says, in effect, that *for some name α*, the instance we get from ∃xΦ(x) by instantiating with α is true. So, we introduce a procedure that lets us do the following:

1. Assume an instance of an existential sentence that we already have as a line of the proof
2. Draw further inferences from that assumption
3. Conclude a sentence that does not contain the instantial name of the assumption any more (usually by using ∃I).
4. Add another line that contains the same sentence just concluded, but with the assumption set changed so that the number of the assumption we made is replaced with the assumption set of the existential sentence

Here is an example using both ∃I and ∃E: a proof of the sequent ∃xFx, ∀x(Fx→Gx) ⊢ ∃xGx:

1	(1) ∃xFx	A
2	(2) ∀x(Fx→Gx)	A

3	(3)	Fa	A
2	(4)	Fa→Ga	2 ∀E
2,3	(5)	Ga	3,4 →E
2,3	(6)	∃xGx	5 ∃I
1,2	(7)	∃xGx	1,6 ∃E(3)

Exercise 3.3.2

S87. ∃x(Gx & ~Fx), ∀x(Gx → Hx) ⊦ ∃x(Hx & ~Fx)

1	(1)	∃x(Gx & ~Fx)	A
2	(2)	∀x(Gx → Hx)	A
3	(3)	Ga & ~Fa	A
3	(4)	Ga	3 &E
3	(5)	~Fa	3 &E
2	(6)	Ga → Ha	2 ∀E
2,3	(7)	Ha	4,6 →E
2,3	(8)	Ha & ~Fa	5,7 &I
2,3	(9)	∃x(Hx & ~Fx)	8 ∃I
1,2	(10)	∃x(Hx & ~Fx)	1,9 ∃E(3)

S88. ∃x(Gx & Fx), ∀x(Fx → ~Hx) ⊦ ∃x~Hx

1	(1)	∃x(Gx & Fx)	A
2	(2)	∀x(Fx → ~Hx)	A
3	(3)	Ga & Fa	A
3	(4)	Fa	3 &E
2	(5)	Fa → ~Ha	2 ∀E
2,3	(6)	~Ha	4,5 →E
2,3	(7)	∃x~Hx	6 ∃I
1,2	(8)	∃x~Hx	1,7 ∃E(3)

S89. ∀x(Gx → ~Fx), ∀x(~Fx → Hx) ⊦ ∀x(Gx →~Hx

1	(1)	∀x(Gx → ~Fx)	A
2	(2)	∀x(~Fx → ~Hx)	A
3	(3)	Ga	A
1	(4)	Ga → ~Fa	1 ∀E
2	(5)	~Fa → ~Ha	2 ∀E
1,3	(6)	~Fa	3,4 →E
1,2,3	(7)	~Ha	5,6 →E
1,2	(8)	Ga → ~Ha	7 →I(3)
1,2	(9)	∀x(Gx →~Hx)	8 ∀I

S90. ∃x(Fx & Ga), ∀x(Fx → Hx) ⊦ Ga & ∃x(Fx & Hx)

1	(1)	∃x(Fx & Ga)	A
2	(2)	∀x(Fx → Hx)	A
3	(3)	Fb & Ga	A
3	(4)	Fb	3 &E
3	(5)	Ga	3 &E
2	(6)	Fb → Hb	2 ∀E
2,3	(7)	Hb	4,6 →E
2,3	(8)	Fb & Hb	4,7 &I
2,3	(9)	∃x(Fx & Hx)	8 ∃I
2,3	(10)	Ga & ∃x(Fx & Hx)	5,9 &I
1,2	(11)	Ga & ∃x(Fx & Hx)	1,10 ∃E(3)

S91. ∀x(Gx → ∃y(Fy & Hy)) ⊦ ∀x~Fx → ~∃zGz

1	(1)	∀x(Gx → ∃y(Fy & Hy))	A
2	(2)	∀x~Fx	A
3	(3)	∃zGz	A
4	(4)	Ga	A
1	(5)	Ga → ∃y(Fy & Hy)	1 ∀E/td>
1,4	(6)	∃y(Fy & Hy)	4,5 →E
7	(7)	Fb & Hb	A
7	(8)	Fb	7 &E
2	(9)	~Fb	2 ∀E
2,7	(10)	~∃zGz	8,9 RAA(3)
1,2,4	(11)	~∃zGz	6,10 ∃E(7)
1,2,3	(12)	~∃zGz	3,11 ∃E(4)
1,2	(13)	~∃zGz	3,12 RAA(3)
1	(14)	∀x~Fx → ~∃zGz	13 →I(2)

S92. ∀x(Gx → (Hx & Jx)), ∀x((Fx ∨ ~Jx) → Gx) ⊦ ∀x(Fx & Hx)

1	(1)	∀x(Gx → (Hx & Jx))	A
2	(2)	∀x((Fx ∨ ~Jx) → Gx)	A
1	(3)	Ga →(Ha & Ja)	1 ∀E
2	(4)	(Fa ∨ ~Ja) → Ga	2 ∀E
5	(5)	Fa	A
5	(6)	Fa ∨ ~Ja	5 ∨I
2,5	(7)	Ga	4,6 →E
1,2,5	(8)	Ha & Ja	3,7 →E
1,2,5	(9)	Ha	8 &E
1,2	(10)	Fa → Ha	9 →I(3)

1,2	(11)	∀x(Fx & Hx)	10 ∀I

S93. ∀x((Gx & Kx) ↔ Hx), ~∃x(Fx & Gx) ⊦ ∀x~(Fx & Hx)

1	(1)	∀x((Gx & Kx) ↔ Hx)	A
2	(2)	~∃x(Fx & Gx)	A
3	(3)	Fa & Ha	A
3	(4)	Fa	3 &E
3	(5)	Ha	3 &E
1	(6)	(Ga & Ka) ↔ Ha	1 ∀E
1	(7)	Ha → (Ga & Ka)	6 ↔E
1,3	(8)	Ga & Ka	5, 7 →E
1,3	(9)	Ga	8 &E
1,3	(10)	Fa & Ga	4,9 &I
1,3	(11)	∃x(Fx & Gx)	10 ∃I
1,2	(12)	~(Fa & Ha)	2,11 RAA(3)
1,2	(13)	∀x~(Fx & Hx)	12 ∀I

S94. ∀x(Gx → Hx), ∃x((Fx & Gx) & Mx) ⊦ ∃x(Fx & (Hx & Mx))

1	(1)	∀x(Gx → Hx)	A
2	(2)	∃x((Fx & Gx) & Mx)	A
3	(3)	(Fa & Ga) & Ma	A
3	(4)	Fa & Ga	3 &E
3	(5)	Fa	4 &E
3	(6)	Ga	4 &E
3	(7)	Ma	3 &E
1	(8)	Ga → Ha	1 ∀E
1,3	(9)	Ha	6,8 →E
1,3	(10)	Ha & Ma	7,9 &I
1,3	(11)	Fa & (Ha & Ma)	5,10 &I
1,3	(12)	∃x(Fx & (Hx & Mx))	11 ∃I
1,2	(12)	∃x(Fx & (Hx & Mx))	2,11 ∃E(3)

S95. ∃x(Gx & ~Fx), ∀x(Gx → Hx) ⊦ ∃x(Hx & ~Fx)

1	(1)	∃x(Gx & ~Fx)	A
2	(2)	∀x(Gx → Hx)	A
3	(3)	Ga&~Fa	A
2	(4)	Ga → Ha	2 ∀E
3	(5)	Ga	3 &E
2,3	(6)	Ha	4,5 →E
3	(7)	~Fa	3 &E

2,3	(8)	Ha & ~Fa	6,7 &I
2,3	(9)	∃x(Hx & ~Fx)	8 ∃I
1,2	(10)	∃x(Hx & ~Fx)	1,9 ∃E(3)

S96. ∃x(Gx & ~Fx), ∀x(Gx → Hx) ⊦ ∃x(Hx & ~Fx)

1	(1)	∃x(Gx & ~Fx)	A
2	(2)	∀x(Gx → Hx)	A
3	(3)	Ga&~Fa	A
3	(4)	Ga	3 &E
2	(5)	Ga→Ha	2 ∀E
2,3	(6)	Ha	4,5 &E
3	(7)	~Fa	
2,3	(8)	Ha&~Fa	
2,3	(9)	∃x(Hx&~Fx)	
1,2	(10)	∃x(Hx&~Fx)	

S97. ∀x~(Gx & Hx), ∃x(Fx & Gx) ⊦ ∃x(Fx & ~Hx)

1	(1)	∀x~(Gx & Hx)	A
2	(2)	∃x(Fx & Gx)	A
3	(3)	Fa & Ga	A
3	(4)	Fa	3 &E
3	(5)	Ga	3 &E
1	(6)	~(Ga & Ha)	1 ∀E
7	(7)	Ha	A
3,7	(8)	Ga & Ha	5,7 &I
1,3	(9)	~Ha	6,8 RAA(7)
1,3	(10)	Fa & ~Ha	4,9 &I
1,3	(11)	∃x(Fx & ~Hx)	10 ∃I
1,2	(12)	∃x(Fx & ~Hx)	2,11 ∃E(3)

S98. ∃x(Fx & ~Hx), ~∃x(Fx & ~Gx) ⊦ ~∀x(Gx → Hx)

1	(1)	∃x(Fx & ~Hx)	A
2	(2)	~∃x(Fx & ~Gx)	A
3	(3)	∀x(Gx → Hx)	A
4	(4)	Fa & ~Ha	A
3	(5)	Ga → Ha	3 ∀E
6	(6)	Ga	A
3,6	(7)	Ha	5,6 →E
4	(8)	~Ha	4 &E
3,4	(9)	~Ga	7,8 RAA(6)

4	(10)	Fa	4 &E
3,4	(11)	Fa & ~Ga	9,10 &I
3,4	(12)	∃x(Fx & ~Gx)	11 ∃I
2,4	(13)	~∀x(Gx → Hx)	2,12 RAA(3)
1,2	(14)	~∀x(Gx → Hx)	1,13 ∃E(4)

S99. ∀x(Hx → (Hx & Gx)), ∃x(~Gx & Fx) ⊦ ∃x(Fx & ~Hx)

1	(1)	∀x(Hx → (Hx & Gx))	A
2	(2)	∃x(~Gx & Fx)	A
3	(3)	~Ga & Fa	A
1	(4)	Ha → (Ha & Ga)	1 ∀E
5	(5)	Ha	A
1,5	(6)	Ha & Ga	4,5 →E
1,5	(7)	Ga	6 &E
3	(8)	~Ga	3 &E
1,3	(9)	~Ha	7,8 RAA(5)
3	(10)	Fa	3 &E
1,3	(11)	Fa & ~Ha	9,10 &I
1,3	(12)	∃x(Fx & ~Hx)	11 ∃I
1,2	(13)	∃x(Fx & ~Hx)	2,12 ∃E(3)

S100. ∀x(Hx → ~Gx), ~∃x(Fx & ~Gx) ⊦ ∀x~(Fx & Hx)

1	(1)	∀x(Hx → ~Gx)	A
2	(2)	~∃x(Fx & ~Gx)	A
3	(3)	Fa & Ha	A
1	(4)	Ha → ~Ga	1 ∀E
3	(5)	Ha	3 &E
1,3	(6)	~Ga	4,5 →E
3	(7)	Fa	3 &E
1,3	(8)	Fa & ~Ga	6,7 &I
1,3	(9)	∃x(Fx & ~Gx)	8 ∃I
1,2	(10)	~(Fa & ~Ha)	2,9 RAA(3)
1,2	(11)	∀x~(Fx & Hx)	10 ∀I

S101. ∀x(Fx ↔ Gx) ⊦ ∀xFx → ∀xGx

1	(1)	∀x(Fx ↔ Gx)	A
2	(2)	∀xFx	A
2	(3)	Fa	2 ∀E
1	(4)	Fa↔Ga	1 ∀E
1	(5)	Fa→Ga	3,4 ↔E

1,2	(6)	Ga	3,5 →E
1,2	(7)	∀xGx	6 ∀I
1	(8)	∀xFx→∀xGx	7 →I(2)

S102. ∃xFx→∀y(Gy→Hy), ∃xJx→∃xGx ⊦ ∃x(Fx&Jx)→∃zHz

1	(1)	∃xFx→∀y(Gy→Hy)	A
2	(2)	∃xJx→∃xGx	A
3	(3)	∃x(Fx&Jx)	A
4	(4)	Fa&Ja	A
4	(5)	Fa	4 &E
4	(6)	∃xFx	5 ∃I
1,4	(7)	∀y(Gy→Hy)	1,6 →E
4	(8)	Ja	4 &E
4	(9)	∃xJx	8 ∃I
2,4	(10)	∃xGx	2,9 →E
11	(11)	Gb	A
1,4	(12)	Gb→Hb	7 ∀E
1,4,11	(13)	Hb	11,12 →E
1,4,11	(14)	∃zHz	13 ∃I
1,2,4	(15)	∃zHz	10, 14 ∃E(11)
1,2,3	(16)	∃zHz	3,15 ∃E(4)
1,2	(17)	∃x(Fx&Jx)→∃zHz	16 →I(3)

S103. ∃xFx ∨ ∃xGx, ∀x(Fx→Gx) ⊦ ∃xGx

1	(1)	∃xFx ∨ ∃xGx	A
2	(2)	∀x(Fx → Gx)	A
3	(3)	~∃xGx	A
1,3	(4)	∃xFx	1,3 ∨E
5	(5)	Fa	A
2	(6)	Fa → Ga	2 ∀E
2,5	(7)	Ga	5,6 →E
2,5	(8)	∃xGx	7 ∃I
2,5	(9)	∃xGx	3,8 RAA(3)
1,2,3	(10)	∃xGx	4,9 ∃E(5)
1,2	(11)	∃xGx	3,10 RAA(3)

S104. ∀x(Fx → ~Gx) ⊦ ~∃x(Fx & Gx)

1	(1)	∀x(Fx → ~Gx)	A
2	(2)	∃x(Fx & Gx)	A
3	(3)	Fa & Ga	A

3	(4)	Fa	3 &E
1	(5)	Fa → ~Ga	1 ∀E
1,3	(6)	~Ga	4,5 →E
3	(7)	Ga	3 &E
1,3	(8)	~∃x(Fx & Gx)	6,7 RAA(2)
1,2	(9)	~∃x(Fx & Gx)	2,8 ∃E(3)
1	(10)	~∃x(Fx & Gx)	2,9 RAA(2)

S105. ∀x((Fx ∨ Hx) →(Gx&Kx)), ~∀x(Kx & Gx) ⊦ ∃x~Hx

1	(1)	∀x((Fx ∨ Hx) →(Gx&Kx))	A
2	(2)	~∀x(Kx & Gx)	A
3	(3)	~∃x~Hx	A
4	(4)	~Ha	A
4	(5)	∃x~Hx	4 ∃I
3	(6)	Ha	3,5 RAA(4)
1	(7)	(Fa ∨ Ha) →(Ga&Ka)	1 ∀E
3	(8)	Fa ∨ Ha	6 ∨I
1,3	(9)	Ga&Ka	7,8 →E
1,3	(10)	Ka	9 &E
1,3	(11)	Ga	9 &E
1,3	(12)	Ka&Ga	10,11 &I
1,3	(13)	∀x(Kx&Gx)	12 ∀I
1,2	(14)	∃x~Hx	2,13 RAA(3)

S106. ∃x((Fx & Gx)→ Hx), Ga & ∀xFx ⊦ Fa & Ha

1	(1)	∃x((Fx & Gx)→ Hx)	A
2	(2)	Ga & ∀xFx	A
2	(3)	Ga	2 &E
2	(4)	∀xFx	2 &E
2	(5)	Fa	4 ∀E
2	(6)	Fa & Ga	3,5 &I
1	(7)	(Fa & Ga) → Ha	1 ∀E
1,2	(8)	Ha	6,7 →E
1,2	(9)	Fa & Ha	5,8 &I

S107. ∀x(Fx ↔∀yGy) ⊦ ∀xFx ∨ ∀x~Fx

This one is something of a trick. Although ∀yGy, which is a constituent of the premise, is a universal sentence, it actually doesn't make any difference *what* it is. The proof below begins with a deduction of a theorem of the form 'P ∨ ~P' and then uses this in the remainder of the proof. Can you guess why this proof works?

1	(1)	∀x(Fx ↔∀yGy)	A
2	(2)	~(∀xFx ∨ ∀x~Fx)	A
3	(3)	~(∀yGy ∨ ~∀yGy)	A
4	(4)	∀yGy	A
4	(5)	∀yGy ∨ ~∀yGy	4 ∨I
3	(6)	~∀yGy	3,5 RAA(4)
3	(7)	∀yGy ∨ ~∀yGy	6 ∨I
	(8)	∀yGy ∨ ~∀yGy	3,7 RAA(4)
1	(9)	Fa↔∀yGy	1 ∀E
1	(10)	∀yGy→Fa	9 ↔E
1,4	(11)	Fa	4,10 →E
1,4	(12)	∀xFx	11 ∀I
1,4	(13)	∀xFx ∨ ∀x~Fx	12 ∨I
1,2	(14)	~∀yGy	2,13 RAA(4)
1	(15)	Fa→∀yGy	9 ↔E
16	(16)	Fa	A
1,16	(17)	∀yGy	15,16 →E
1,2	(18)	~Fa	14,16 RAA(16)
1,2	(19)	∀x~Fx	18 ∀I
1,2	(20)	∀xFx ∨ ∀x~Fx	19 ∨I
1	(20)	∀xFx ∨ ∀x~Fx	2,20 RAA(2)

S108. ∀y(Fa → (∃xGx → Gy)), ∀x(Gx → Hx), ∀x(~Jx → ~Hx) ⊦ ∃x~Jx → (~Fa ∨ ∀x~Gx)

This is frightfully long, but nevertheless it's fairly straightforward. The strategy: (a) assume all the premises and the antecedent of the conclusion; (2) assume the negation of the consequent of the conclusion (for an RAA); (c) assume an instance of the existential sentence at (4) (for ∃E); (d) using (2) and (3), get ~Gb; (e) get Fa from assumption (5) and use that and an instance of (1) to get (eventually) ~∃xGx; (f) use that in turn to get another RAA with (5); (g) discharge the last two assumptions, (4) and (5), with RAA and →I. Note the use of ∃E at line 21 to discharge assumption (6). You can make this a *lot* shorter if you use various derived rules.

1	(1)	∀y(Fa → (∃xGx → Gy))	A
2	(2)	∀x(Gx → Hx)	A
3	(3)	∀x(~Jx → ~Hx)	A
4	(4)	∃x~Jx	A
5	(5)	~(~Fa ∨ ∀x~Gx)	A
6	(6)	~Jb	A
3	(7)	~Jb→~Hb	3 ∀E
3,6	(8)	~Hb	6,7 →E
2	(9)	Gb→Hb	2 ∀E
10	(10)	Gb	A
2,10	(11)	Hb	9,10 →E

2,3,6	(12) ~Gb	8,11 RAA(10)
13	(13) ~Fa	A
13	(14) ~Fa ∨ ∀x~Gx	13 ∨I
5	(15) Fa	5,14 RAA(13)
1	(16) Fa→(∃xGx→Gb)	1 ∀I
1,5	(17) ∃xGx→Gb	15,16 →E
18	(18) ∃xGx	A
1,5,18	(19) Gb	17,18 →E
1,2,3,5,6	(20) ~∃xGx	12,19 RAA(18)
1,2,3,4,5	(21) ~∃xGx	4,20 ∃E(6)
22	(22) Gc	A
22	(23) ∃xGx	22 ∃I
1,2,3,4,5	(24) ~Gc	21,23 RAA(23)
1,2,3,4,5	(25) ∀x~Gx	24 ∀I
1,2,3,4,5	(26) ~Fa ∨ ∀x~Gx	25 ∨I
1,2,3,4	(27) ~Fa ∨ ∀x~Gx	5,26 RAA(5)
1,2,3	(28) ∃x~Jx → (~Fa ∨ ∀x~Gx)	27 →I(4)

S109. ∀x(Dx → Fx) ⊦ ∀z(Dz → (∀y(Fy→ Gy)→Gz))

1	(1)	∀x(Dx→Fx)	A
2	(2)	Da	A
3	(3)	∀y(Fy→Gy)	A
1	(4)	Da→Fa	1 ∀E
3	(5)	Fa→Ga	3 ∀E
1,2	(6)	Fa	2,4 →E
1,2,3	(7)	Ga	5,6 →E
1,2	(8)	∀y(Fy→Gy)→Ga	7 →I(3)
1	(9)	Da→(∀y(Fy→Gy)→Ga)	8 →I(2)
1	(10)	∀z(Dz→(∀y(Fy→Gy)→Gz))	9 ∀I

S110. ∃xFx ↔ ∀y((Fy ∨ Gy) → Hy), ∃xHx, ~∀z~Fz ⊦ ∃x(Fx & Hx)

1	(1)	∃xFx ↔ ∀y((Fy ∨ Gy) → Hy)	A
2	(2)	∃xHx	A
3	(3)	~∀z~Fz	A
4	(4)	~∃xFx	A
5	(5)	Fa	A
5	(6)	∃xFx	5 ∃I
4	(7)	~Fa	4,6 RAA(5)
4	(8)	∀z~Fz	7 ∀I
3	(9)	∃xFx	3,8 RAA(4)

1	(10)	∃xFx → ∀y((Fy ∨ Gy) → Hy)	1 ↔E
1,3	(11)	∀y((Fy ∨ Gy) → Hy)	9,10 →E
5	(12)	Fa ∨ Ga	5 ∨I
1,3	(13)	(Fa ∨ Ga) → Ha	11 ∀E
1,3,5	(14)	Ha	12, 13 →E
1,3,5	(15)	Fa & Ha	5,14 &I
1,3,5	(16)	∃x(Fx & Hx)	15 ∃I
1,3	(17)	∃x(Fx & Hx)	9,16 ∃E(5)
	(16)		

S111. ∀xFx ⊦ ~∃xGx ↔ ~(∃x(Fx&Gx) & ∀y(Gy→Fy))

Although this is a relatively long proof, it's not especially complex: it just takes time because in order to use ↔I, you first need to give separate proofs for each of two conditionals. For the first of these, ~∃xGx → ~(∃x(Fx&Gx) & ∀y(Gy→Fy)), notice that there are *no* assumptions at line (10). That's not a mistake: line (10) is a theorem.

1	(1)	∀xFx	A
2	(2)	~∃xGx	A
3	(3)	∃x(Fx&Gx) & ∀y(Gy→Fy)	A
3	(4)	∃x(Fx&Gx)	3 &E
5	(5)	Fa&Ga	A
5	(6)	Ga	5 &E
5	(7)	∃xGx	6 ∃I
3	(8)	∃xGx	4,7 ∃E(5)
2	(9)	~(∃x(Fx&Gx) & ∀y(Gy→Fy))	2,8 RAA(3)
	(10)	~∃xGx → ~(∃x(Fx&Gx) & ∀y(Gy→Fy))	9 →I(2)
11	(11)	~(∃x(Fx&Gx) & ∀y(Gy→Fy))	A
12	(12)	∃xGx	A
13	(13)	Ga	A
1	(14)	Fa	1 ∀E
1,13	(15)	Fa&Ga	13,14 &I
1,13	(16)	∃x(Fx&Gx)	15 ∃I
1,12	(17)	∃x(Fx&Gx)	12,16 ∃E(13)
20	(18)	Gb	A
1	(19)	Fb	1 ∀E
1	(20)	Gb→Fb	19 →I(18)
1	(21)	∀y(Gy→Fy)	20 ∀I
1,12	(22)	∃x(Fx&Gx) & ∀y(Gy→Fy)	17,21 &I
1,11	(23)	~∃xGx	11,22 RAA(12)
1	(24)	~(∃x(Fx&Gx) & ∀y(Gy→Fy)) → ~∃xGx	23 →I(11)
1	(25)	~∃xGx ↔ ~(∃x(Fx&Gx) & ∀y(Gy→Fy))	10,24 ↔I

S112. ∀x(∃yFyx→∀zFxz) ⊦ ∀yx(Fyx→Fxy)

1	(1)	∀x(∃yFyx→∀zFxz)	A
1	(2)	∃yFya→∀zFaz	1 ∀E
3	(3)	Fba	A
3	(4)	∃yFya	3 ∃I
1,3	(5)	∀zFaz	2,4 →E
1,3	(6)	Fab	5 ∀E
1	(7)	Fba→Fab	6 →I(3)
1	(8)	∀x(Fbx→Fxb)	7 ∀I
1	(9)	∀yx(Fyx→Fxy)	8 ∀I

S113. ∃x(Fx & ∀yGxy), ∀xy(Gxy → Gyx) ⊦ ∃x(Fx & ∀yGyx)

1	(1)	∃x(Fx & ∀yGxy)	A
2	(2)	∀xy(Gxy → Gyx)	A
3	(3)	Fa & ∀yGay	A
3	(4)	Fa	3 &E
3	(5)	∀yGay	3 &E
2	(6)	∀y(Gay → Gya)	2 ∀E
3	(7)	Gab	5 ∀E
2	(8)	Gab→Gba	6 ∀E
2,3	(9)	Gba	7,8 →E
2,3	(10)	∀yGya	9 ∀I
2,3	(11)	Fa & ∀yGya	3,10 &I
2,3	(12)	∃x(Fx & ∀yGyx)	11 ∃I
1,2	(12)	∃x(Fx & ∀yGyx)	1,12 ∃E(3)

S114. ∃x~∀y(Gxy → Gyx) ⊦ ∃x∃y(Gxy & ~Gyx)

1	(1)	∃x~∀y(Gxy → Gyx)	A
2	(2)	~∀y(Gay → Gya)	A
3	(3)	~∃y~(Gay → Gya)	A
4	(4)	~(Gab → Gba)	A
4	(5)	∃y~(Gay → Gya)	4 ∃I
3	(6)	Gab → Gba	3,5 RAA(4)
3	(7)	∀y(Gay → Gya)	6 ∀I
2	(8)	∃y~(Gay → Gya)	2,7 RAA(3)
9	(9)	~(Gab → Gba)	A
10	(10)	Gab	A
11	(11)	Gba	A
11	(12)	Gab → Gba	11 →I(10)
9,11	(13)	~Gba	9,12 RAA(10)

9	(14) ~Gba	11,13 RAA(11)
15	(15) ~Gab	A
15	(16) ~Gab ∨ Gba	A
10,15	(17) Gba	10,16 ∨E
15	(18) Gab → Gba	17 →I(10)
9	(19) Gab	9,18 RAA(15)
9	(20) Gab & ~Gba	14,19 &I
9	(21) ∃y(Gay & ~Gya)	20 ∃I
9	(22) ∃x∃y(Gxy & ~Gyx)	21 ∃I
2	(23) ∃x∃y(Gxy & ~Gyx)	8,22 ∃E(9)
1	(22) ∃x∃y(Gxy & ~Gyx)	1,23 ∃E(2)

S115. ∀x(Gx → ∀y(Fy → Hxy)), ∃x(Fx & ∀z~Hxz) ⊦ ~∀xGx

1	(1)	∀x(Gx → ∀y(Fy → Hxy))	A
2	(2)	∃x(Fx & ∀z~Hxz)	A
3	(3)	∀xGx	A
4	(4)	Fa & ∀z~Haz	A
4	(5)	Fa	4 &E
4	(6)	∀z~Haz	4 &E
4	(7)	~Haa	6 ∀E
3	(8)	Ga	3 ∀E
1	(9)	Ga → ∀y(Fy → Hay)	1 ∀E
1,3	(10)	∀y(Fy → Hay)	8,9 →E
1,3	(11)	Fa → Haa	10 ∀E
1,3,4	(12)	Haa	5,11 →E
1,4	(13)	~∀xGx	7,12 RAA(3)
1,2	(14)	~∀xGx	2,13 ∃E(4)

S116. ∀xy(Fxy → Gxy) ⊦ ∀x(Fxx → ∃y(Gxy & Fyx))

1	(1) ∀xy(Fxy → Gxy)	A
2	(2) Faa	A
1	(3) ∀y(Fay → Gay)	1 ∀E
1	(4) Faa → Gaa	3 ∀E
1,2	(5) Gaa	2,4 →E
1,2	(6) Gaa & Faa	2,5 &I
1,2	(7) ∃y(Gay & Fya)	6 ∃I
1	(8) Faa → ∃y(Gay & Fya)	7 →I(2)
1	(9) ∀x(Fxx → ∃y(Gxy & Fyx))	8 ∀I

S117. ∀xy(Fxy→~Fyx) ⊦ ~∃xFxx

One odd feature of this proof is that assumption 2 is discharged twice. First, it's discharged by RAA in line 7.

Then, it's brought back in so as to discharge assumption 3 by ∃E. Finally, it's discharged again by RAA. In this case, it's hard to avoid that double discharge.

1	(1)	∀xy(Fxy→~Fyx)	A
2	(2)	∃xFxx	A
3	(3)	Faa	A
1	(4)	∀y(Fay→~Fya)	1 ∀E
1	(5)	Faa→~Faa	4 ∀E
1,3	(6)	~Faa	3,5 →E
1,3	(7)	~∃xFxx	3,6 RAA(2)
1,2	(8)	~∃xFxx	2,7 ∃E(3)
1	(9)	~∃xFxx	2,8 RAA(2)

S118. ∀x∃y(Fxy & ~Fyx) ⊦ ∃x~∀yFyx

1	(1)	∀x∃y(Fxy&~Fyx)	A
1	(2)	∃y(Fay&~Fya)	1 ∀E
3	(3)	Fab&~Fba	A
4	(4)	∀yFya	A
4	(5)	Fba	4 ∀E
3	(6)	~Fba	3 &E
3	(7)	~∀yFya	5,6 RAA(4)
1	(8)	~∀yFya	2,7 ∃E(3)
1	(9)	∃x~∀yFyx	8 ∀I

S119. ∀y(∃x~Fxy→~Fyy) ⊦ ∀x(Fxx → ∀yFyx)

Notice that line 9 below does not have line 7 in its assumption set. Therefore, we can use ∀I at row 10 **on b**. We could not have used it on a, since line 9 still has 3 in its assumption set, and line 3 contains a.

1	(1)	∀y(∃x~Fxy→~Fyy)	A
1	(2)	∃x~Fxa→~Faa	1 ∀E
3	(3)	Faa	A
4	(4)	∃x~Fxa	A
1,4	(5)	~Faa	2,4 →E
1,3	(6)	~∃x~Fxa	3,5 RAA(4)
7	(7)	~Fba	A
7	(8)	∃x~Fxa	7 ∃I
1,3	(9)	Fba	6,8 RAA(7)
1,3	(10)	∀Fya	9 ∀I
1	(11)	Faa→yFya	10 →I(3)
1	(12)	∀x(Fxx → ∀yFyx)	11 ∀I

S120. ∃xFxx→∀xyFxy ⊦ ∀x(Fxx → ∀yFxy)

1	(1)	∃xFxx→∀xyFxy	A
2	(2)	Faa	A
2	(3)	∃xFxx	2 ∃I
1,2	(4)	∀xyFxy	1,3 →E
1,2	(5)	∀yFay	4 ∀E
1	(6)	Faa→∀yFay	5 →I(2)
1	(7)	∀x(Fxx → ∀yFxy)	6 ∀I