Overview of Today's Lecture

- Today's Music: *Led Zeppelin*
- **The mid-term is on Thursday, 6/10 (in class).**
  - I've posted (and discussed) a sample mid-term. It has the same structure and complexity as the actual mid-term (good study guide).
- I have posted HW #3, which is due on Thursday @ 4pm in the drop box.
  - It's all chapter 3 problems — truth-table methods for validity-testing.
- I posted revised versions of lecture #6 and my "short method" handout.
- **`MacLogic`** — a useful computer program for natural deduction.
  - You might want to download **`MacLogic`** at this point ...
  - We'll be using it very soon (and for the rest of the term) ...
  - See `http://fitelson.org/maclogic.htm`
- Today: Chapter 3, Finalé and Chapter 4 Intro.

Expressive Completeness: Rewind, and More Extra-Credit

- **Q.** How can we define $\leftrightarrow$ in terms of $|$? **A.** If you naïvely apply the schemes I described last time, then you get a *187 symbol monster*:

  $\ulcorner p \leftrightarrow q \urcorner \mapsto A|A$, where A is given by the following *93 symbol* expression:

  $$(((p|(q|q))|(p|(q|q)))|((p|(q|q))|(p|(q|q))))|(((q|(p|p))|(q|(p|p)))|((q|(p|p))|(q|(p|p))))$$

- There are *simpler* definitions of $\leftrightarrow$ using $|$. *E.g.*, this *43 symbol* answer:

  $$\ulcorner p \leftrightarrow q \urcorner \mapsto ((p|(q|q))|(q|(p|p)))|((p|(q|q))|(q|(p|p)))$$

- I offered E.C. for the shortest solution. Some students have come up with it (the shortest solution is < 25 symbols, counting parens).

- **More E.C.** Find the *shortest possible* definitions of (1) $\ulcorner p \to q \urcorner$, (2) $\ulcorner p \vee q \urcorner$, and (3) $\ulcorner {\sim}p \mathbin{\&} {\sim}q \urcorner$ in terms of p, q, and the NAND operator $|$.

- If you submit EC, please *prove* the correctness of your solution, using a truth-table method. You may submit these E.C. solutions to your GSI.

Presenting Your "Short-Cut" Truth-Table Tests

- In any application of the "short" method, there are two possibilities:

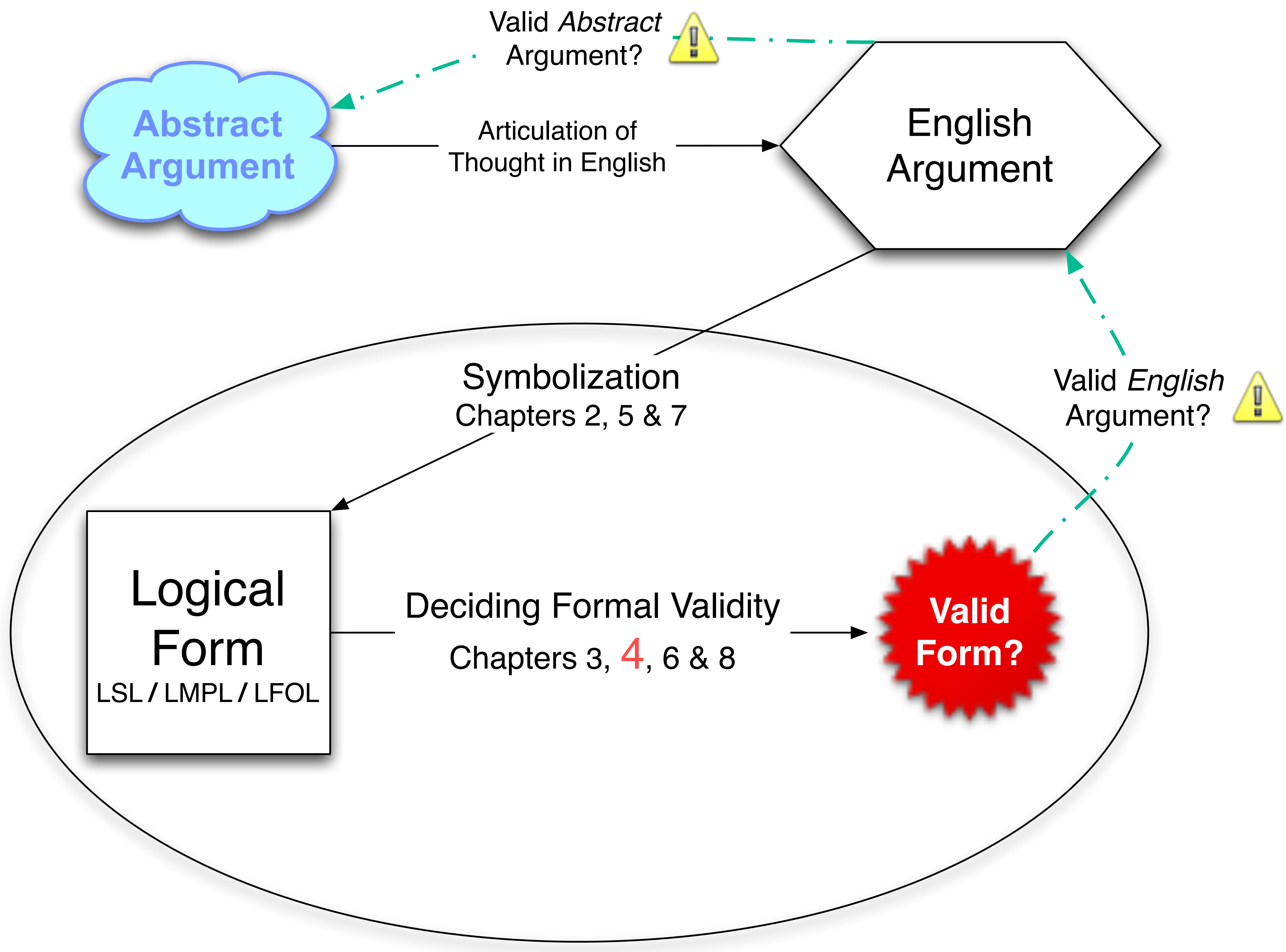
1. You find an interpretation (*i.e.*, a row of the truth-table) on which all the premises $p_1, \ldots, p_n$ of an argument are true and the conclusion q is false. *All you need to do here* is (*i*) write down the relevant row of the truth-table, and (*ii*) say "Here is an interpretation on which $p_1, \ldots, p_n$ are all true and q is false. So, $p_1, \ldots, p_n \therefore q$ is *in*valid."
2. You discover that there is *no possible way* of making $p_1, \ldots, p_n$ true and q false. Here, you need to *explain all of your reasoning* (as I do in lecture, or as Forbes does, or as I do in my handout). It must be clear that you have *exhausted all possible cases*, before concluding that $p_1, \ldots, p_n \therefore q$ is *valid*. This can be rather involved, and should be spelled out in a step-by-step fashion. Each salient case has to be examined.

- Consult my handout and lecture notes for model answers of both kinds.

Some Fun with LSL Symbolization & Semantics: Knights & Knaves

- The island of Knights and Knaves has two types of inhabitants, Knights who always tell the truth, and Knaves who always lie.
- Suppose A is the proposition person a is a knight and suppose a makes a statement S. Then, A is semantically equivalent to S [$A \dashv\vDash S$]. Why?
- So, whenever an inhabitant x makes a claim S, we can infer that $X \leftrightarrow S$. That is, we can infer that x is a knight iff S is true. Simple applications:
- If a says "I am a Knight" then we can infer that $A \leftrightarrow A$. But, since this is always true (a *tautology*), we get no information from this statement.
- Conversely, it *cannot* be the case that a native says "I am a Knave" because we could then conclude $A \leftrightarrow {\sim}A$, which is self-contradictory.
- If a says "I am the same type as b," then we can infer $A \leftrightarrow (A \leftrightarrow B)$ which is equivalent to B [$B \dashv\vDash A \leftrightarrow (A \leftrightarrow B)$] (we proved this above). So, this statement allows us to infer that person b is a Knight!

- Given this set-up, use truth-tables (complete or "shortened") to justify your answers to the following questions about Knights and Knaves. These are good exercises that combine LSL translation, LSL semantics, and truth-table methods (*i.e.*, chapters 2 and 3 of our textbook).
  1. It is rumored that there is gold buried on the island (G). You ask one of the natives, a, whether there is gold on the island. He makes the following response: "There is gold on this island if and only if I am a Knight." Is there gold buried on the island? [Answer: Yes.]
  2. Inhabitant a says "Either I am a Knave or b is a Knight." What can we infer about a and b? [Answer: a and b are both Knights.]
  3. Three of the inhabitants — a, b and c — were standing together in the garden. A stranger passed by and asked a, "Are you a Knight or a Knave?". a answered, but rather indistinctly, so the stranger could not make out what he said. The stranger then asked b, "What did a say?". b replied, "a said that he is a Knave." At this point the third man, c, said "Don't believe b; he's lying!". What are b and c? [Answer: b is a Knave and c is a Knight.]


Valid *Abstract* Argument?
Abstract Argument
Articulation of Thought in English
English Argument
Symbolization
Chapters 2, 5 & 7
Valid *English* Argument?
Logical Form
LSL / LMPL / LFOL
Deciding Formal Validity
Chapters 3, 4, 6 & 8
Valid Form?

Truth *vs* Proof ($\vDash$ *vs* $\vdash$)

- Recall: $p \vDash q$ iff it is impossible for p to be true while q is false.
- We have methods (truth-tables) for establishing $\vDash$ and $\nvDash$ claims. These methods are especially good for $\nvDash$ claims, but they get very complex for $\vDash$ claims. Is there another more "natural" way to prove $\vDash$'s? Yes!
- In Chapter 4, we will learn a *natural deduction system* for LSL. This is a system of *rules of inference* that will allow us to prove all valid LSL arguments in a purely syntactical way (no appeal to semantics).
- The notation $p \vdash q$ means that *there exists a natural deduction proof of q from p* in our natural deduction system for sentential logic.
- $\ulcorner p \vdash q \urcorner$ is short for $\ulcorner p$ *deductively* entails $q \urcorner$.
- While $\vDash$ has to do with *truth*, $\vdash$ does *not*. $\vdash$ has only to do with what can be *deduced*, using a *fixed set* of formal, natural deduction rules.

- Happily, our system of natural deduction rules is *sound* and *complete*:
  - **Soundness**. If $p \vdash q$, then $p \vDash q$. [no proofs of *in*validities!]
  - **Completeness**. If $p \vDash q$, then $p \vdash q$. [proofs of *all* validities!]
- We will not prove the soundness and completeness of our system of natural deduction rules. I will say a few things about soundness as we go along, but completeness is much harder to establish (140A!).
- We'll have rules that permit the *elimination* or *introduction* of each of the connectives &, $\rightarrow$, $\vee$, $\sim$, $\leftrightarrow$ within natural deductions. These rules will make sense, from the point of view of the semantics.
- A *proof* of q from p is a sequence of LSL formulas, beginning with p and ending with q, where each formula in the sequence is *deduced* from previous lines, *via* a correct application of one of the *rules*.
- Generally, we will be talking about deductions of formulas q from sets of premises $p_1, \ldots, p_n$. We call these $\ulcorner p_1, \ldots, p_n \vdash q \urcorner$'s *sequents*.

An Example of a Natural Deduction Involving & and →

- The following is a valid LSL argument form:

$A \,\&\, B$

$C \,\&\, D$

$(A \,\&\, D) \rightarrow H$

$\therefore H$

- Here's a (7-line) natural deduction proof of the sequent corresponding to this argument: $A \,\&\, B, C \,\&\, D, (A \,\&\, D) \rightarrow H \vdash H$.

1	(1)	$A \,\&\, B$	Premise
2	(2)	$C \,\&\, D$	Premise
3	(3)	$(A \,\&\, D) \rightarrow H$	Premise
1	(4)	A	1 &E
2	(5)	D	2 &E
1, 2	(6)	$A \,\&\, D$	4, 5 &I
1, 2, 3	(7)	H	3, 6 →E ✦

The Rule of Assumptions (Preliminary Version)

- **Rule of Assumptions** (preliminary version): The premises of an argument-form are listed at the start of a proof in the order in which they are given, each labeled 'Premise' on the right and numbered with its own line number on the left. Schematically:

j	(j)	p	Premise

- We can see that our example proof begins, as it should, with the three premises of the argument-form, written as follows:

1	(1)	$A \,\&\, B$	Premise
2	(2)	$C \,\&\, D$	Premise
3	(3)	$(A \,\&\, D) \rightarrow H$	Premise

The Rule of &-Elimination (&E)

- **Rule of &-Elimination**: If a conjunction ⌜p & q⌝ occurs at line j, then at any *later* line k one may infer either conjunct, labeling the line 'j &E' and writing on the left all the numbers which appear on the left of line j.

  Schematically:

$a_1, \ldots, a_n$	(j)	p & q			$a_1, \ldots, a_n$	(j)	p & q	
	⋮			OR		⋮		
$a_1, \ldots, a_n$	(k)	p	j &E		$a_1, \ldots, a_n$	(k)	q	j &E

- We can see that our example deduction continues, in lines (4) and (5), with two correct applications of the &-Elimination Rule:

1	(4)	A	1 &E
2	(5)	D	2 &E

The Rule of &-Introduction (&I)

- **Rule of &-Introduction**: For any formulae p and q, if p occurs at line j and q occurs at line k then the formula ⌜$p \,\&\, q$⌝ may be inferred at line m, labeling the line 'j, k &I' and writing on the left all numbers which appear on the left of line j *and* all which appear on the left of line k. [Note: we may have j < k, j > k, *or* j = k. *Why*?]

$a_1, \ldots, a_n$	(j)	p	
	⋮		
$b_1, \ldots, b_u$	(k)	q	
	⋮		
$a_1, \ldots, a_n, b_1, \ldots, b_u$	(m)	$p \,\&\, q$	j, k &I

- We can see that our example deduction continues, in lines (6), with a correct application of the &-Introduction Rule:

1, 2	(6)	$A \,\&\, D$	4, 5 &I

The Rule of →-Elimination (→E)

- **Rule of →-Elimination**: For any formulae p and q, if ⌜$p \to q$⌝ occurs at a line j and p occurs at a line k, then q may be inferred at line m, labeling the line 'j, k →E' and writing on the left all numbers which appear on the left of line j *and* all numbers which appear on the left of line k. [Note: We may have either j < k or j > k.]

$$
\begin{array}{rcll}
a_1, \ldots, a_n & (j) & p \to q & \\
 & \vdots & & \\
b_1, \ldots, b_u & (k) & p & \\
 & \vdots & & \\
a_1, \ldots, a_n, b_1, \ldots, b_u & (m) & q & \text{j, k} \to \text{E}
\end{array}
$$

- Our example deduction *concludes* (we indicate the end of a proof with a '✦'), in line (7), with a correct application of the →-Elimination Rule:

$$
1, 2, 3 \quad (7) \quad H \quad 3, 6 \to \text{E} ✦
$$

Deduction #2 Using the Rules &E and &I

- Consider the valid LSL argument form:

  $A \,\&\, (B \,\&\, C)$

  $\therefore\, C \,\&\, (B \,\&\, A)$

- Let's do a deduction of this argument form:

1	(1)	$A \,\&\, (B \,\&\, C)$	Premise
1	(2)	A	1 &E
1	(3)	$B \,\&\, C$	1 &E
1	(4)	B	3 &E
1	(5)	C	3 &E
1	(6)	$B \,\&\, A$	4, 2 &I
1	(7)	$C \,\&\, (B \,\&\, A)$	5, 6 &I ✦

- NOTE: &E can *only* be applied to formulas whose *main* connective is '&', and &E *must* be *applied to that particular* connective.

Deduction #3 Using the Rules &E, &I, and →E

- Let's do a deduction of:

$$A \to (B \to (C \to D))$$
$$C \,\&\, (A \,\&\, B)$$
$$\therefore D$$

1	(1)	$A \to (B \to (C \to D))$	Premise
2	(2)	$C \,\&\, (A \,\&\, B)$	Premise
2	(3)	$A \,\&\, B$	2 &E
2	(4)	A	3 &E
1, 2	(5)	$B \to (C \to D)$	1, 4 →E
2	(6)	B	3 &E
1, 2	(7)	$C \to D$	5, 6 →E
1	(8)	C	2 &E
1, 2	(9)	D	7, 8 →E ✦

Note on -E Rules — Avoiding a Common Error

- As with &E, →E can *only* be applied to the *main* → of a conditional — *not* to any *other* →'s which may be in a formula.
- So, the step from (3) to (4) in the following is *incorrect*.

1	(1)	$A \to (B \to (C \to D))$	Premise
2	(2)	$C \,\&\, (A \,\&\, B)$	Premise
2	(3)	C	2 &E
1, 2	(4)	D	1, 3 →E (NO!)

- *The elimination rule for a connective c can* ***only*** *be applied to a line if that line has an occurrence of c as its* ***main*** *connective, and the rule* ***must*** *be applied to* ***that*** *occurrence of c.*

How to Deduce a Conditional: I

- To deduce a conditional, we *assume* its antecedent and try to deduce its consequent from this assumption. If we are able to deduce the consequent from our assumption of the antecedent, then we *discharge* our assumption, and infer the conditional.
- To implement the $\to$I rule, we will first need a refined Rule of Assumptions that will allow us to assume arbitrary formulas "for the sake of argument", later to be discharged after making desired deductions. Here's the refined rule of Assumptions:
- **Rule of Assumptions** (final version): At any line j in a proof, any formula p may be entered and labeled as an assumption (or premise, where appropriate). The number j should then be written on the left. Schematically:

j (j) p Assumption (or: Premise)

How to Deduce a Conditional: II — The →I Rule

- Now, we need a formal Introduction Rule for the →, which captures the intuitive idea sketched above (*i.e.*, assuming the antecedent, *etc.*):
- **Rule of →-Introduction**: For any formulae p and q, if q has been inferred at a line k in a proof and p is an assumption or premise occurring at line j, then at line m we may infer ⌜$p \to q$⌝, labeling the line 'j, k →I' and writing on the left the same assumption numbers which appear on the left of line k, except that we *delete* j if it is one of these numbers. Note: we may have j < k, j > k, *or* j = k (*why*?). Schematically:

j	(j)	p	Assumption (or: Premise)
	⋮		
$a_1, \ldots, a_n$	(k)	q	
	⋮		
$\{a_1, \ldots, a_n\}/j$	(m)	$p \to q$	j, k →I

Examples Involving &E, &I, →E, and →I

- Can you deduce the following, using &E, &I, →E, and →I?

(a) $A \to B$, $A \to C$ $\therefore A \to (B \,\&\, C)$

(b) $(A \,\&\, B) \to C$ $\therefore A \to (B \to C)$

(c) $B \,\&\, C$ $\therefore (A \to B) \,\&\, (A \to C)$

(d) $A \to B$ $\therefore (A \,\&\, C) \to (B \,\&\, C)$

(e) $A \,\&\, (B \,\&\, C)$ $\therefore A \to (B \to C)$

(f) $A \to B$ $\therefore A \to (C \to B)$

Important Tips For Using the →I Rule

- Use →I only when you wish to *derive* a conditional ⌜$p \rightarrow q$⌝.
- To derive ⌜$p \rightarrow q$⌝ using →I, assume the antecedent p and try to prove the consequent q. Always assume the *whole* of p, not just a part of it (like one of the conjuncts of a conjunction).
- When a conditional ⌜$p \rightarrow q$⌝ is derived by →I, the antecedent p must always be a formula which you have assumed at a previous line: it cannot be a formula that you have derived from other things. This is because it must be *discharged*.
- When you apply →I, remember to *discharge* the assumption by dropping the assumption number on the left.
- Check that the last line of your proof does not depend on any extra assumptions you have made besides your premises.

Proofs by Contradiction and the Rules for $\sim$

- If assuming p leads us to a contradiction, then we may infer $\ulcorner\sim p\urcorner$. [Note: This was implicit in our "short" truth-table method.]
- This style of proof is called *proof by contradiction* (or *reductio ad absurdum*). It is a very powerful technique that we'll see often.
- In our natural deduction system, the introduction and elimination rules for negation ($\sim$I and $\sim$E) allow us to perform *reductio*s.
- We use the symbol '$\curlywedge$' to indicate that a contradiction has been deduced (*i.e.*, that p and $\ulcorner\sim p\urcorner$ have been deduced, for some p). We call '$\curlywedge$' the *absurdity symbol* (an *atom*, added to the lexicon of LSL).
- With these preliminaries out of the way, we're ready to see what the negation rules look like, and how they work…

The Elimination Rule for ∼

Rule of ∼-Elimination: For any formula q, if ⌜∼q⌝ has been inferred at a line j in a proof and q at line k (j < k or j > k) then we may infer '⅄' at line m, labeling the line 'j, k ∼E' and writing on its left the numbers on the left at j and on the left at k. Schematically (with j < k):

$a_1, \ldots, a_n$	(j)	∼q	
	⋮		
$b_1, \ldots, b_u$	(k)	q	
	⋮		
$a_1, \ldots, a_n, b_1, \ldots, b_u$	(m)	⅄	j, k ∼E

- Note: we have *added* the symbol '⅄' to the language of LSL. It is treated as if it were an *atomic sentence* of LSL. We can now use it in compound sentences (*e.g.*, '$A \rightarrow$ ⅄', '∼∼⅄', *etc.*).

The Introduction Rule for ~

Rule of ~-Introduction: If '⋏' has been inferred at line k in a proof and $\{a_1, \ldots, a_n\}$ are the assumption and premise numbers '⋏' depends upon, then if p is an assumption (or premise) at line j, ⌜$\sim p$⌝ may be inferred at line m, labeling the line 'j, k ~I' and writing on its left the numbers in the set $\{a_1, \ldots, a_n\}$/j.

j	(j)	p	Assumption
	⋮		
$a_1, \ldots, a_n$	(k)	⋏	
	⋮		
$\{a_1, \ldots, a_n\}$/j	(m)	$\sim p$	j, k ~I

- ~I is used (typically *with* ~E) to deduce ⌜$\sim p$⌝ *via reductio ad absurdum*, by (*i*) *assuming* p, (*ii*) deducing '⋏', and (*iii*) *discharging* the assumption.

Using The Rules ∼I and ∼E: Example #2

- Let's construct a proof of: $\sim(A \,\&\, B) \vdash (A \rightarrow \sim B)$.

1	(1)	$\sim(A \,\&\, B)$	Premise
2	(2)	A	Assumption
3	(3)	B	Assumption
2, 3	(4)	$A \,\&\, B$	2, 3 &I
1, 2, 3	(5)	$\curlywedge$	1, 4 ∼E
1, 2	(6)	$\sim B$	3, 5 ∼I
1	(7)	$A \rightarrow \sim B$	2, 6 →I ✦

The Rule of Double Negation (DN)

- Negation is an odd connective in our system. It not only has an introduction rule and an elimination rule, but it also has an additional rule called the *double negation* (DN) rule.
- The DN rule says that we may infer p from $\ulcorner{\sim}{\sim}p\urcorner$. Without this DN rule, we would not be able to prove certain valid LSL argument forms — *e.g.*, ${\sim}(A \,\&\, {\sim}B) \;\therefore\; (A \rightarrow B)$.

Rule of Double Negation: For any formula p, if $\ulcorner{\sim}{\sim}p\urcorner$ has been inferred at a line j in a proof, then at line k we may infer p, labeling the line 'j ' and writing on its left the numbers to the left of j.

$a_1, \ldots, a_n$	(j)	${\sim}{\sim}p$	
$a_1, \ldots, a_n$	(k)	p	j DN

An Example which *Requires* DN: I

- Consider the valid LSL form $\sim(A \mathbin{\&} \sim B)\ \therefore\ (A \rightarrow B)$. If we try to prove this without using DN, we'll quickly get "stuck".
- We would begin by (*i*) writing down '$\sim(A \mathbin{\&} \sim B)$' as our only Premise, then (*ii*) assuming 'A' and trying to deduce 'B'.
- But, since 'B' has no main connective, it's not clear how in the world we could possibly prove it. Without a main connective to introduce using an -I rule, we have no way to derive 'B'.
- But, '$\sim\sim B$' *does* have a main connective ('$\sim$'). So, we could use $\sim$I to prove '$\sim\sim B$', and then use DN to infer 'B'.
- In fact, this is the *only* strategy that will work!
- Let's prove $\sim(A \mathbin{\&} \sim B) \vdash (A \rightarrow B)$.

An Example which *Requires* DN: II

1	(1)	$\sim(A \,\&\, \sim B)$	Premise
2	(2)	A	Assumption
3	(3)	$\sim B$	Assumption
2, 3	(4)	$A \,\&\, \sim B$	2, 3 &I
1, 2, 3	(5)	$\curlywedge$	1, 4 ∼E
1, 2	(6)	$\sim\sim B$	3, 5 ∼I
1, 2	(7)	B	6 DN
1	(8)	$A \rightarrow B$	2, 7 →I ✦

Another Example Requiring DN: Using

- Here is a (generated) proof of: $B, \sim B \vdash A$.

1	(1)	B	Premise
2	(2)	~B	Premise
3	(3)	~A	Assumption
1,2	(4)	⋏	2,1 ~E
1,2	(5)	~~A	3,4 ~I
1,2	(6)	A	5 DN

Cautionary Remarks about *Reductio* Proofs

- Once you have deduced a contradiction (⅄) in the course of a proof, you can subsequently deduce *any* formula p *via* ~I and DN.
- But, such a deduction may depend on various assumptions, which means *they won't be proofs from the premises alone*. From last time:

1	(1)	$\sim(A \& B)$	Premise
2	(2)	A	Assumption
3	(3)	B	Assumption
2, 3	(4)	$A \& B$	2, 3 &I
1, 2, 3	(5)	⅄	1, 4 ~E
1, 2	(6)	$\sim B$	3, 5 ~I
1	(7)	$A \rightarrow \sim B$	2, 6 →I ✦

- You might be tempted to think that you could prove $A \rightarrow \sim B$ *via* $\sim$I and DN after step (5). You *can* deduce it in this way, *but* you get:

1	(1)	$\sim(A \,\&\, B)$	Premise
2	(2)	A	Assumption
3	(3)	B	Assumption
2, 3	(4)	$A \,\&\, B$	2, 3 &I
1, 2, 3	(5)	$\curlywedge$	1, 4 $\sim$E
6	(6)	$\sim(A \rightarrow \sim B)$	Assumption
1, 2, 3	(7)	$\sim\sim(A \rightarrow \sim B)$	6, 5 $\sim$I
1, 2, 3	(8)	$A \rightarrow \sim B$	7 DN

- *This does not help*. We need to prove $A \rightarrow \sim B$ from (1) *alone*, *not* from (1), (2), and (3). [Note: (1)–(3) is an *inconsistent* set!]

- Lesson: A strategy for proving the conclusion *from the premises alone* requires *discharging* all assumptions that are not premises.

Important Tips For Using the Negation Rules

- If you are trying to derive a formula with '∼' as its main connective, use ∼I to obtain it. I.e., assume the formula within the scope of the '∼' and try to derive ⅄ using ∼E.
- When you apply ∼I, the formula which you infer must be the negation of a premise or an assumption. It cannot be the negation of a formula which has been deduced.
- If one of your premises or assumptions has '∼' as its main connective, it is likely that its role in the proof will be to be one of a pair of contradictory formulae in an application of ∼E. You should therefore consider trying to derive the formula within the scope of the '∼' to get the other member of the contradictory pair.
- If you are trying to deduce a sentence-letter and there is no obvious way to do it, consider trying to derive its double-negation and then use DN. Last Resort!
- At this point, *you should only assume a formula p if you are trying to deduce its negation or trying to deduce a conditional with p as antecedent*. Only make an assumption when you've figured how you're going to use ∼I or →I to *discharge* it.

General Strategy — Working in Both Directions

- Begin by writing the premises (if any) at the top of your scratch paper area, using the Rule of Assumptions.
- Then, write the conclusion (the *main* goal formula) you're trying to derive at the bottom of your scratch area.
- Next, determine what the main connective (if any) of your conclusion is, then apply the introduction rule for that connective.
- This will yield *sub-goal* formula(s). Write the sub-goal formula(s) directly above your conclusion. Then try to figure-out how to prove the sub-goal formula(s) from your premises.
- This will yield sub-sub-goal formula(s). And so on ...
- Repeat this process until you have worked your way all the way back up to your premises/assumptions (if a formula is resisting proof, you might try to prove its *double-negation* using ~I with ~E).