Propositional Natural Deduction

Robert Keller
15 November 200

Logic

- In CS 60 we had an introduction to both **proposition-** and **predicate-logic**.
- These were covered from the viewpoint of **meaning** (known as “model theory” to logicians).
- There is another part of the story dealing with the structure of **proofs** (known as “proof theory”).
- We focus on the latter now, and will connect the two eventually.

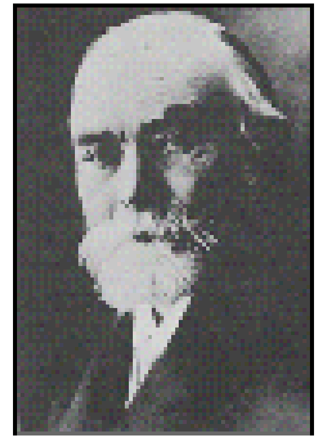
Logic in CS 81

- We have two **objectives** in studying formal logic:
  - To firm up our concept of what forms a proof and how to create proofs.
  - To investigate the connection between computability and provability, such as:
    - The problem of giving an algorithm that will determine whether or not certain kinds of statements can be proved from certain axioms is **unsolvable**.

Formal Systems

- A system of logical proof is a variety of **formal system**, just as grammars and Turing machines are formal systems.
- A formal system tells how to **construct** things, using precise rules, usually as some form of induction.
- “Formal” means that adherence to the rules can be checked **algorithmically**.

Gottlob Frege (1848-1925)



- Created modern logic by introducing the predicate calculus.
- Developed a formalized definition of “proof”.
- Defined the natural numbers in anticipation of Peano’s axiomatization (1889)
- Did not anticipate Russell’s paradox.

Varieties of Logical Proof Systems

- Axiomatic or Hilbert/Ackermann:
  - Basis is a set of **axioms**
  - Rules of inference tell how to derive **theorems** from axioms (in zero or more steps).
  - Relatively few rules of inference
- Natural Deduction, or Gentzen:
  - No axioms
  - Rules of inference tell how to derive **sequents**, which can entail axioms as pre-conditions and theorems as post-conditions.
  - Relatively many rules of inference.
- The two are equivalent; it is a matter of style.

Hilbert/Ackerman and Gentzen

- David Hilbert (1862-1943)

- Wilhelm Ackermann (1896-1962) student of Hilbert (no photo available)
- Gerhard Gentzen (1909-1945)

Natural Deduction

- A natural deduction system derives **sequents**, expressions of the form:

$$\square_1, \square_2, \ldots, \square_n \mid \square \quad \square$$

- Each of the $\square_i$ and $\square$ represents a **logical formula** in an appropriate language (in the sense we have been using the term).

- The interpretation of the sequent is that each $\square_i$ is a **premise** and $\square$ is the **conclusion**.

- The $\square_i$ *could* be **axioms**, then $\square$ would be a **theorem**. However, the word "theorem" is usually reserved for the case that the set of premises is empty.

Truth vs. Derivation

- The **intended interpretation** of the sequent is that $\square$ is a **true** formula provided that each of the $\square_i$ are true.

- Whether or not this is really the case will depend on the rules.

- The definition of "**truth**" will be given later, but you can assume that it is like the one you know.

- Derivations themselves do not rely on notions of truth; they are totally **mechanical**.

Reference

- There are several approaches using sequents and different languages for formulas.
- We will be following the one in Huth & Ryan (HR).

A Typical Propositional Language

- E is the start symbol
- E □ A | // Atom
  (□E) | // Negation (not)
  (E □ E) | // Conjunction (and)
  (E E) | // Disjunction (or)
  (E ‘□ ’ E) | // Implication (implies)
  □ | // Bottom
  T // Top

- A □ ‘p’ | ‘q’ | ‘r’ | ‘s’ | … // Propositions

Bottom and Top?

- Think of bottom ($\bot$) as representing the constant "false".

- Think of top ($\top$) as representing the constant "true".

Precedence

- The language as given fully parenthesizes everything.
- We will allow precedence in lieu of parentheses as an **abbreviation**. The binding order is negation, conjunction, disjunction, implication.

- So

  ((p □ (□q)) ((□r) □ (s □ q)))

  could be abbreviated:

  (p □ □q) (□r □ (s □ q))

Examples of Sequents

- p, (p □ q) |□ q
- (p   q), □p |□ q
- (p □ q), (p   r), □r |□ q
- The first, for example, is interpreted “if p is true and (p □ q) is true, then q is true”.

More Notes on Sequents

- On the left-hand side of |□ in

  $\square_1, \square_2, \ldots \square_n \ |\square \quad \square$

  the formulas are regarded as a **set**:
  - order doesn’t matter
  - repetition doesn’t matter
- Order and repetition does matter **within** a formula. Formulas are just strings.

Sequents and Intuition

- You might be thinking "Why bother with sequents; I can do all of this with my knowledge of tautologies, etc."
- Your knowledge can be used as **intuition** for validating a sequent.
- However, sequents are supposed to express whether certain **deductions** are valid, as they might occur in a mathematical proof.
- Tautologies won't be enough when we introduce predicates and quantifiers.
- In addition to **using** sequents, we intend to **study** the **proof systems** themselves (called meta-logic).

Sequent Meta-Logical Issues

- **Soundness**:
  - Determine whether a sequent derives **only** true formulas from true formulas.
- **Completeness**:
  - Determine whether **every** true formula can be derived from a fixed set of formulas (axioms).

Natural Deduction Rules

- Each rule represents an **allowable step** in deriving a sequent.
- The rules focus on deriving formulas by **introducing** or **eliminating** the various connectives:

  □
  □

  □
- There is one rule for each case (introduction and elimination) for at least each connective, i.e. at least 8 rules. Some rules have multiple sub-rules.

Why “Natural” Deduction?

- “Natural” is a slogan intending to suggest that these rules are ones that might be used in normal proof construction and argumentation.
- Natural deduction also allows an argument to be developed by examining the desired conclusion and working toward assumed premises in a “natural” way.

□-Introduction Rule (□i)

- $\dfrac{□ \quad □}{□□□}$ (□i)

- The reading of this rule is:
  - If □ and □ are any formulas that follow from the premises of a sequent,

    then the formula □□□ also follows from those premises.
- The formulas above the line are called the **antecedents** and the one below the **consequent**.

Rule vs. Sequent

- Every rule immediately creates an infinite number of sequents. For example, the rule

  $$\frac{\square \quad \square}{\square\square\square}$$

  creates sequents **of the form**

  □, □ |□ (□□□)

  for every pair of formulas □ and □.

- The greek letters in the sequent form shown are not **the** formulas; they stand for aribtrary formulas.

- Many sequents require **multiple** rule applications to establish.

Examples of Sequents Derived Using Only the (□i) Rule

- p, (q r) |□ p□(q r) [One rule app.]
- p, (q r) |□ (q r)□p [One rule app.]
- p, (q r), s |□ ((q r)□(p□s)) [Two rule apps.]

Showing Sequent Derivations by Steps

- Derive p, (q r), s |□ ((q r)□(p□s)):

  1. p — Premise
  2. (q r) — Premise
  3. s — Premise
  4. (p□s) — Rule □i applied to formulas 1, 3
  5. ((q r)□(p□s)) — Rule □i applied to formulas 2, 4

1.	p	Premise
2.	(q r)	Premise
3.	s	Premise
4.	(p□s)	Rule □i applied to formulas 1, 3
5.	((q r)□(p□s))	Rule □i applied to formulas 2, 4

- The numbers on the right refer to the **antecedents** used in the rule to obtain the formula on the left, which is the **consequent** of a rule.

Showing Sequent Derivations by DAGs

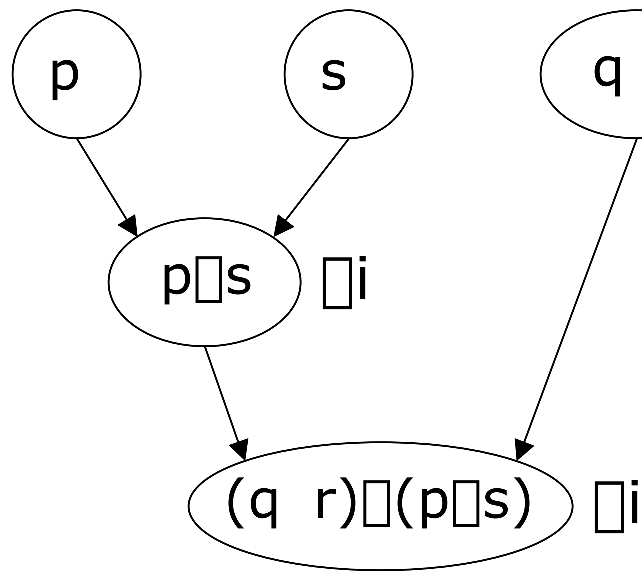
- DAG = “Directed Acyclic Graph”
- The premises are at the leaves of the DAG.

$$\cfrac{\cfrac{p \qquad s}{(p \land s)}\,\land i \qquad (q\ \ r)}{((q\ \ r) \land (p \land s))}\,\land i$$

- Note that (p∧s) is used as the consequent of one rule application and the antecedent of another.

DAG made more evident

p   s □i
(p□s)   (q r) □i
((q r)□(p□s))



Steps vs. DAGs

- Steps correspond to the way that an argument might be presented in a math text or paper.

- DAGs allow for better visulization of what is used for what.

- Either representation can be constructed from the other.

□-Elimination Rule (□e$_1$, □e$_2$)

- $\frac{□□□}{□}$ (□e$_1$)

- $\frac{□□□}{□}$ (□e$_2$)

- Two sub-rules are needed because **order matters** within a formula. This rule eliminates one side of the □ or the other.

A Step Derivation Using ∧e and ∧i

- Derive $p \wedge (q \wedge r) \vdash (p \wedge q) \wedge r$:

1. $p \wedge (q \wedge r)$ — Premise
2. p — $\wedge e_1$ 1
3. $q \wedge r$ — $\wedge e_2$ 1
4. q — $\wedge e_1$ 3
5. r — $\wedge e_1$ 3
6. $p \wedge q$ — $\wedge i$ 2, 4
7. $(p \wedge q) \wedge r$ — $\wedge i$ 6, 5

A DAG Derivation Using ∧e and ∧i

- Derive $p \wedge (q \wedge r) \vdash (p \wedge q) \wedge r$:

```
p ∧ (q ∧ r)                                      ∧e1     ∧e2
_____________________________________
      p              (q ∧ r)                     ∧e1     ∧e2
                     _____________
                       q       r      ∧i
      __________________
         (p ∧ q)                                 ∧i
         ___________________________
              (p ∧ q) ∧ r
```

This shows that the DAG is not generally a "tree", as some antecedents are used multiple times.

DAG made more evident

- Derive $p \wedge (q \wedge r) \vdash (p \wedge q) \wedge r$:

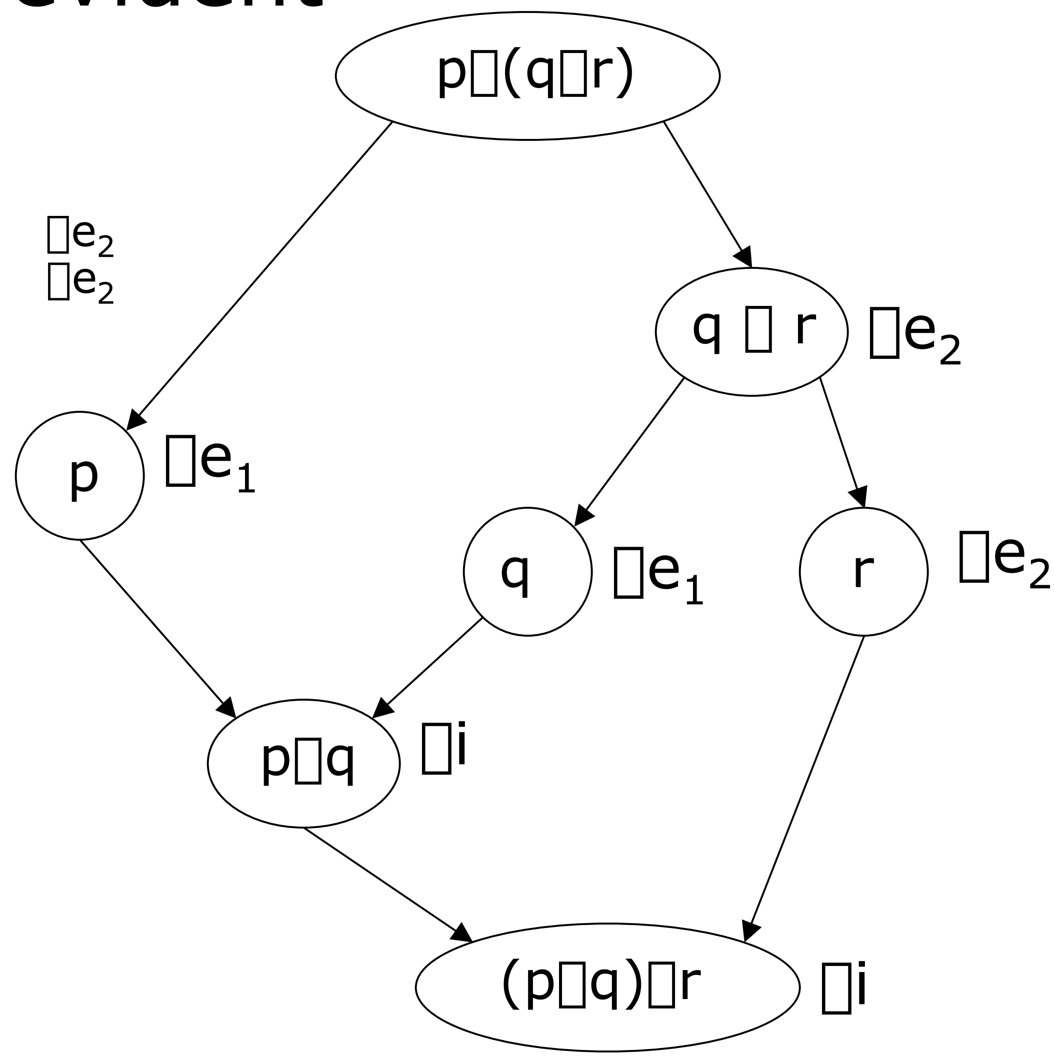
$p \wedge (q \wedge r)$ $\wedge e_1$ $\wedge e_2$

p $(q \wedge r)$ $\wedge e_1$ $\wedge e_2$

q r $\wedge i$

$(p \wedge q)$ $\wedge i$

$(p \wedge q) \wedge r$



Constructing Proofs by Working Backward

- If the conclusion is a premise, there is nothing to do.
- Otherwise, the outermost logical connective may suggest what rule could be used:
  - Derive p (q r) | (p q) r
  - The outermost connective in the conclusion is therefore use i as the last step:
    - (p q) r   i   6, 5
  - The use of i will require derivation of two new formulas:
    - (p q)   r
  - Apply this approach recursively.

Choices

- Often the rule choice is not unique.
- Make a choice, but be prepared to backtrack (crossing off what you have done) and try a different one.

Constructing Proofs by Working Forward

- If a premise is the conclusion, there is nothing to do.
- Otherwise, synthesize a formula from existing formulas using available rules.
- Working forward might entail many choices of a formula to be synthesized, not all of which will be useable in deriving the conclusion.

Constructing Proofs by Working Both Directions Simultaneously

- Blend together working backward with working forward until the two “meet in the middle”.
- Don’t overlook the DAG model as a means of arriving at proofs.
- Consider converting the DAG to steps for final clarity.

-Introduction Rule (i_1, i_2)

- $\dfrac{\square}{\square \quad \square}$ (i_1)

- $\dfrac{\square}{\square \quad \square}$ (i_2)

□ -Elimination Rule, Modus Ponens

- $\dfrac{□,\ □□\ □}{□}$ (□ e)

- Its latin name ***modus ponens*** (**MP**) is often used for this rule.

Example using □ -Elimination Rule

- Derive p, (p □ q), (q □ r) |□ r

1. p — Premise
2. p □ q — Premise
3. q □ r — Premise
4. q — □ e 1, 2
5. r — □ e 4, 3

- With this example, you can start to see how deriving a sequent might actually be easier (and more "natural") than establishing a tautology.

Another form of □ -Elimination Rule, Modus Tollens

- A related **macro** or “derived rule” is ***modus tollens*** (**MT**):

- □□ □, □□ (MT)
  □□

- “macro” means that this rule is a convenience and can be treated as an abbreviation for the application of other rules.
- We will elaborate on this later.

Example using MT

- Derive $\neg r, (p \rightarrow q), (q \rightarrow r) \vdash \neg p$

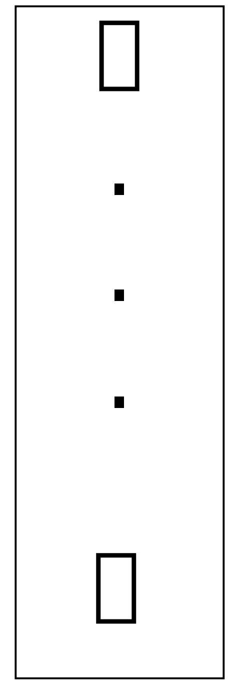
1.	$\neg r$	Premise
2.	$p \rightarrow q$	Premise
3.	$q \rightarrow r$	Premise
4.	$\neg q$	MT 3, 1
5.	$\neg p$	MT 2, 4

□□-Elimination and Introduction Rules

- $\dfrac{\square\square\square}{\square}$ (□□e)
- $\dfrac{\square}{\square\square\square}$ (□□i) (This rule is “derived”.)

Rules with Sub-Derivations

- Certain rules have **sub-derivations**, rather than simply formulas, in their **antecedents**.
- A sub-derivation may incorporate **assumptions** that behave as **premises** but are not premises of the sequent being proved.
- These assumptions must be treated carefully to avoid confusion with regular premises.
- Accordingly, sub-derivations are shown inside a **box**.
- **Assumptions introduced inside the box cannot be used as premises outside the box.**
- **However**, sub-derivations **may** use formulas derived earlier **outside** the box.



□ -Introduction Rule

- This is an example of a rule using a sub-derivation.

□
.
.
.
□

(□ i)

□□ □

- Here to derive we use □ as an assumption and get □ as a conclusion using a sub-derivation.
- The sub-derivation is in a box because □ is not useable outside.

Example Using Sub-Derivation

- Derive (p □ q), (q □ r) |□ (p □ r)

1.	p □ q	Premise	
2.	q □ r	Premise	
3.	p	Assumption	
4.	q	□ e	1, 2
5.	r	□ e	2, 4
6.	p □ r	□ i	2-5

Another Example Using Sub-Derivation

- Derive (□p □ □q) |□ (q □ p) :

1.	□p □ □q	Premise	
2.	q	Assumption	
3.	□□q	□□i	2
4.	□□p	MT	1, 3
5.	p	□□e	4
6.	q □ p	□i	2-5

- Pattern matching:
  □□ □, □□ (MT) □ is □p, □ is □q,
  □□ □□ is □□p, □□ is □□q

A Sub-Derivation can be Trivial

- Derive |□ (p □ p) (Set of premises is empty):

1.	p	Assumption	
2.	p □ p	□ i	1, 1

- Pattern matching:

□
.
.
.
□

———————— (□ i)

□□ □

Both □ and □ are p.

Sub-Derivations can be Nested

- Derive $(p \land q) \rightarrow r \vdash p \rightarrow (q \rightarrow r)$

1.	$(p \land q) \rightarrow r$	Premise
2.	p	Assumption
3.	q	Assumption
4.	$p \land q$	$\land$i 2,3
5.	r	$\rightarrow$e 1, 4
6.	$q \rightarrow r$	$\rightarrow$i 3-5
7.	$p \rightarrow (q \rightarrow r)$	$\rightarrow$i 2-6

Sub-Derivations and DAGs

- It is unclear how to show sub-derivations in the DAG model.
- The customary way is to introduce the sub-derivation and **discharge** (**cross-out**) the assumptions so that they cannot be used outside the sub-derivation.
- The steps model is clearer in this regard, because nesting shows the order of discharge.

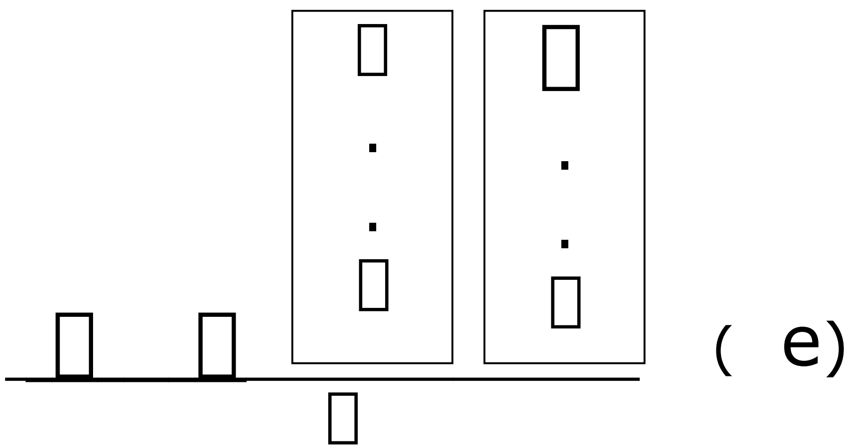
Sub-Derivations in the DAG model

- Derive $(p \wedge q) \rightarrow r \vdash p \rightarrow (q \rightarrow r)$

$\not p$	Assumption (/ denotes discharged)
$\not q$	Assumption
$(p \wedge q)$	$\wedge$i
$(p \wedge q) \rightarrow r$	Premise
r	$\rightarrow$ e
$q \rightarrow r$	$\rightarrow$ i
$p \rightarrow (q \rightarrow r)$	$\rightarrow$ i

-Elimination Rule

- This rule uses two sub-derivations:



- The interpretation is that if we want to “get rid of” a disjunction, we can derive a common formula from the two disjuncts.

Sub-Derivations vs. Sequents?

- Aren’t the boxed sub-derivations essentially sequents themselves?
- If so, why don’t we use the notation □ |□ □ rather than

□
.
.
.
□

- The answer probably lies in the fact that sub-derivations can make use of formulas **outside** the box, and we’d have to repeat those formulas as premises of the sequent.

□-Introduction Rule

This rule introduces □ through “contradiction”:

□

.

.

.

□

□□ (□i)

[illegible]-Elimination Rule

$$\frac{[illegible]\ [illegible]\ [illegible]\ [illegible]}{[illegible]} \quad ([illegible]\text{e})$$

⊥-Elimination Rule

$$\frac{\bot}{\phi}\ (\bot e)$$

- If we can derive ⊥ then we can derive anything. Consequently, the things we derive won’t have much information value. So being able to derive ⊥ is undesirable, except in a sub-derivation.

Macro or Derived Rules

- Earlier MT was mentioned as a “macro” rule.
- The name “macro” alludes to programming language macros.
- While superficially similar to a subroutine, a macro is a text substitution done before a source is compiled or interpreted.
- In our case, it is a rule that could be replaced with a sequence of uses of other rules.

MT as a Macro derived from other rules

- □□ □, □□ (MT)
  □□

1. □□ □ Premise
2. □□ Premise
3. □ Assumption
4. □ □ e 1, 3
5. □ □e 4, 2
6. □□ □i 3-5

- Every use of MT could thus be replaced with this sequence, which uses 3 rules: □ e, □e, □i.

□□i as a Macro derived from other rules

- $\dfrac{\square}{\square\square\square}$ (□□i)

1.	□	Premise
2.	□ □	Assumption
3.	□	□e 1, 2
4.	□□□	□i 2-3

Macro vs. Sequent

- Why isn't a macro rule just another sequent?

RAA (Reductio ad absurdum) Rule

- This rule has a similarity to □i:

□□
.
.
.
□

―――――――――― (RAA)

□

RAA as a Macro derived from other rules

1. □□ . . . □ — Premise
2. □□ □ □ — □ i 1
3. □□ — Assumption
4. □ — □e 2,3
5. □□□ — □i 3-4
6. □ — □□e 5

LEM (Law of the Excluded Middle)

- $$\frac{}{\varphi \vee \neg\varphi}$$ (No antecedent)

1.	$\neg(\varphi \vee \neg\varphi)$	Assumption
2.	φ	Assumption
3.	$\varphi \vee \neg\varphi$	$\vee i_1$ 2
4.	$\bot$	$\neg e$ 3, 1
5.	$\neg\varphi$	$\neg i$ 2-4
6.	$\varphi \vee \neg\varphi$	$\vee i_2$ 2
7.	$\bot$	$\neg e$ 6, 1
8.	$\neg\neg(\varphi \vee \neg\varphi)$	$\neg i$ 1, 7
9.	$\varphi \vee \neg\varphi$	$\neg\neg e$ 8

Summary of Non-Derived Rules

Connective	Introduction	Elimination
□	□i	□e_1, □e_2
	i_1, i_2	e
□	□ i	□ e
□	□i	□e
□	(none)	□e
□□	(derived)	□□e

Summary of Derived Rules So Far

- MT (Modus Tollens)
- RAA (Reductio ad Absurdum)
- LEM (Law of the Excluded Middle)
- $\neg\neg$i

Validity vs. Provability

- $\square_1, \ldots, \square_n \mid\square \;\; \square$ means $\square$ is **provable** from $\square_1, \ldots, \square_n$
- $\square_1, \ldots, \square_n \models \square$ means roughly the following:

  If each of $\square_i$ is true, then $\square$ is true.

- In other words, $\square$ is a **valid** conclusion from $\square_1, \ldots, \square_n$.
- We need a definition of ***truth*** to make this precise.