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:

\[ \square_1, \square_2, \ldots, \square_n \models \square \]

• 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 $\phi$ is a **true** formula provided that each of the $\phi_i$ 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 \ni A \ni A
  (\neg E) \ni // Negation (not)
  (E \land E) \ni // Conjunction (and)
  (E \lor E) \ni // Disjunction (or)
  (E \rightarrow E) \ni // Implication (implies)
  \bot \ni // Bottom
  T \ni // Top

- A \ni 'p' | 'q' | 'r' | 's' | ... // Propositions
Bottom and Top?

- Think of bottom (⊥) as representing the constant “false”.

- Think of 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
\[((p \land (\neg q)) \lor ((\neg r) \land (s \lor q)))\]

could be abbreviated:
\[(p \land \neg q) \lor (\neg r\land (s \lor q))\]
Examples of Sequents

- \( p, (p \implies q) \rightarrow q \)
- \( (p \implies q), \neg p \rightarrow q \)
- \( (p \implies q), (p \implies r), \neg r \rightarrow q \)

- The first, for example, is interpreted “if \( p \) is true and \( (p \implies q) \) is true, then \( q \) is true”. 
More Notes on Sequents

- On the left-hand side of $\Box$ in $\Box_1, \Box_2, \ldots \Box_n \Box \Box$

  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)

- \( i \) (\( \overline{i} \))

- The reading of this rule is:
  - If \( i \) and \( \overline{i} \) are any formulas that follow from the premises of a sequent,
    then the formula \( i \overline{i} \) 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
  \[ \frac{\varphi}{\varphi, \psi} \]
  creates sequents of the form
  \[ \varphi, \psi \mid \varphi, \psi (\varphi, \psi) \]
  for every pair of formulas \( \varphi \) and \( \psi \).

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

- Many sequents require multiple rule applications to establish.
Examples of Sequents Derived Using Only the (→i) Rule

- \( p, (q \rightarrow r) \vdash p \rightarrow (q \rightarrow r) \) [One rule app.]
- \( p, (q \rightarrow r) \vdash (q \rightarrow r) \rightarrow p \) [One rule app.]
- \( p, (q \rightarrow r), s \vdash ((q \rightarrow r) \rightarrow (p \rightarrow s)) \) [Two rule apps.]
Showing Sequent Derivations by Steps

- Derive $p, (q \land r), s \vdash ((q \land r) \land (p \land s))$:

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

\[
\begin{array}{c}
p & s & \supset i \\
(p \supset s) & (q \supset r) & \supset i \\
((q \supset r) \supset (p \supset s))
\end{array}
\]

- Note that \((p \supset s)\) is used as the consequent of one rule application and the antecedent of another.
DAG made more evident

\[
p \rightarrow s \leftarrow i \\
(p \bowtie s) \rightarrow (q \bowtie r) \leftarrow i \\
((q \bowtie r) \bowtie (p \bowtie s))
\]
Steps vs. DAGs

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

• DAGs allow for better visualization of what is used for what.

• Either representation can be constructed from the other.
-Elimination Rule (e₁, e₂)

- \( \frac{\frac{\bar{\gamma}}{\bar{\gamma}}}{\bar{\gamma}} (e₁) \)

- \( \frac{\frac{\bar{\gamma}}{\bar{\gamma}}}{\bar{\gamma}} (e₂) \)

- Two sub-rules are needed because order matters within a formula. This rule eliminates one side of the \( \bar{\gamma} \) or the other.
A Step Derivation Using $\Box e$ and $\Box i$

- Derive $p \Box (q \Box r) \Box (p \Box q) \Box r$:

1. $p \Box (q \Box r)$  
   Premise
2. $p$  
   $\Box e_1$ 1
3. $q \Box r$  
   $\Box e_2$ 1
4. $q$  
   $\Box e_1$ 3
5. $r$  
   $\Box e_1$ 3
6. $p \Box q$  
   $\Box i$ 2, 4
7. $(p \Box q) \Box r$  
   $\Box i$ 6, 5
A DAG Derivation Using $\vee$ and $\wedge$

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

```
\begin{array}{ccc}
\, & \, & \\wedge e_1 \quad \wedge e_2 \\
\, & \, & \wedge e_1 \quad \wedge e_2 \\
\, & \, & \wedge i \\
\wedge e_1 \quad \wedge e_2 \\
\wedge i \\
\wedge i \\
\wedge i \\
\wedge i \\
\end{array}
```

This shows that the DAG is not generally a “tree”, as some antecedents are used multiple times.
DAG made more evident

- Derive $p \uparrow (q \uparrow r) \uparrow (p \uparrow q) \uparrow r$:

\[\begin{align*}
&\quad p \uparrow (q \uparrow r) \\
&\quad (p \uparrow q) \uparrow r \\
&\quad (p \uparrow q) \uparrow r
\end{align*}\]
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 \( p \rightarrow (q \rightarrow r) \rightarrow (p \rightarrow q) \rightarrow r \)
  - The outermost connective in the conclusion is \( \rightarrow \) therefore use \( \rightarrow i \) as the last step:
    - \( (p \rightarrow q) \rightarrow r \rightarrow i \ 6, 5 \)
  - The use of \( \rightarrow i \) will require derivation of two new formulas:
    - \( (p \rightarrow q) \rightarrow r \)
  - 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 \((i_1, i_2)\)

- \(i_1\) 
  - \(j\) 

- \(i_2\) 
  - \(j\)
¬-Elimination Rule, Modus Ponens

- \( \neg p, p \rightarrow q \quad (\neg e) \)

- Its latin name *modus ponens* (MP) is often used for this rule.
Example using $\exists$ -Elimination Rule

- Derive $p, (p \exists q), (q \exists r) \vdash r$

1. $p$ Premise
2. $p \exists q$ Premise
3. $q \exists r$ Premise
4. $q \quad \exists e \ 1, 2$
5. $r \quad \exists e \ 4, 3$

- 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 \( \Box \)-Elimination Rule, Modus Tollens

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

\[
\begin{array}{c}
\Box \\
\hline
\end{array}
\]

- “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 \( r, (p \implies q), (q \implies r) \mid p \)

1. \( r \)  
   Premise
2. \( p \implies q \)  
   Premise
3. \( q \implies r \)  
   Premise
4. \( q \)  
   MT 3, 1
5. \( p \)  
   MT 2, 4
Elimination and Introduction Rules

- \( \frac{\text{\times}}{\text{\checkmark}} \)  (\( \text{\checkmark \text{-e}} \))

- \( \frac{\text{\times}}{\text{\checkmark \checkmark \checkmark}} \)  (\( \text{\checkmark \text{-i}} \))  (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.

- Here to derive we use \( \square \) as an assumption and get \( \square \) as a conclusion using a sub-derivation.

- The sub-derivation is in a box because \( \square \) is not useable outside.
Example Using Sub-Derivation

- Derive \((p \implies q), (q \implies r) \models (p \implies r)\)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>(p \implies q)</td>
<td>Premise</td>
</tr>
<tr>
<td>2.</td>
<td>(q \implies r)</td>
<td>Premise</td>
</tr>
<tr>
<td>3.</td>
<td>(p)</td>
<td>Assumption</td>
</tr>
<tr>
<td>4.</td>
<td>(q)</td>
<td>(\vDash e) 1, 2</td>
</tr>
<tr>
<td>5.</td>
<td>(r)</td>
<td>(\vDash e) 2, 4</td>
</tr>
<tr>
<td>6.</td>
<td>(p \implies r)</td>
<td>(\vDash i) 2-5</td>
</tr>
</tbody>
</table>
Another Example Using Sub-Derivation

- Derive ($\neg p \lor \neg q \lor (q \lor p)$):

1. $\neg p \lor \neg q$  
   Premise
2. $q$  
   Assumption
3. $\neg q$  
   $\neg i$  2
4. $\neg p$  
   MT  1, 3
5. $p$  
   $\neg e$  4
6. $q \lor p$  
   $\neg i$  2-5

- Pattern matching:
  $\neg p \lor \neg q \lor (q \lor p)$ ($\lor$ is $\lor p$, $\lor$ is $\lor q$, $\lor$ is $\lor p$, $\lor$ is $\lor q$)
A Sub-Derivation can be Trivial

- Derive $\vdash (p \supset p)$ (Set of premises is empty):

1. $p$ Assumption
2. $p \supset p$ $\supset i$ 1, 1

- Pattern matching:

Both $\supset$ and $\supset$ are $p$. 
Sub-Derivations can be Nested

- Derive $(p \implies q) \implies r \implies p \implies (q \implies r)$

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>$(p \implies q) \implies r$</td>
<td>Premise</td>
</tr>
<tr>
<td>2.</td>
<td>$p$</td>
<td>Assumption</td>
</tr>
<tr>
<td>3.</td>
<td>$q$</td>
<td>Assumption</td>
</tr>
<tr>
<td>4.</td>
<td>$p \implies q$</td>
<td>$\because i 2,3$</td>
</tr>
<tr>
<td>5.</td>
<td>$r$</td>
<td>$\because e 1, 4$</td>
</tr>
<tr>
<td>6.</td>
<td>$q \implies r$</td>
<td>$\because i 3-5$</td>
</tr>
<tr>
<td>7.</td>
<td>$p \implies (q \implies r)$</td>
<td>$\because i 2-6$</td>
</tr>
</tbody>
</table>
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 \((p \land q) \rightarrow r \equiv p \rightarrow q \equiv r\)

```
p
  q
  ______
(p \land q)    Assumption (/ denotes discharged)

  (p \land q) \equiv r      Assumption
  r                         Premise
__________
  q \equiv r
  p \equiv (q \equiv r)     \equiv i
                              \equiv 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 \( \vdash \models \) rather than \( \vdash \models \)?

  \[
  \begin{array}{c}
  \hline
  \vdash \\
  \hline
  \end{array}
  \]

  \[
  \begin{array}{c}
  \vdash \\
  \vdash \\
  \vdash \\
  \hline
  \end{array}
  \]

- The answer probably lies in the fact that sub-derivations can make use of formulas \textbf{outside} the box, and we’d have to repeat those formulas as premises of the sequent.
□-Introduction Rule

This rule introduces □ through “contradiction”:

□ □
□ □...
□ □...
□ □...
□ □ (□i)
\(-\)-Elimination Rule

\[ \text{Diagram: Elimination Rule} \]
$\
eg\neg$-Elimination Rule

$(\neg\neg e)$

- If we can derive $\neg\neg e$ then we can derive anything. Consequently, the things we derive won’t have much information value. So being able to derive $\neg\neg e$ 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

- \( \overline{\text{MT}} \) derived from other rules

1. Premise
2. Premise
3. Assumption
4. \( \overline{e} \) 1, 3
5. \( \overline{e} \) 4, 2
6. \( \overline{i} \) 3-5

Every use of MT could thus be replaced with this sequence, which uses 3 rules: \( \overline{e} \), \( \overline{e} \), \( \overline{i} \).
$i$ as a Macro derived from other rules

- \[ (\text{Premise}) \]

1. [ ] \hspace{2cm} Premise
2. [ ] \hspace{2cm} Assumption
3. [ ] \hspace{2cm} \text{Assumption}
4. [ ] \hspace{2cm} $i$ 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 $\prod$:
RAA as a Macro derived from other rules

1. \[ \ldots \]
   Premise

2. \[ \ldots \]
   \[ i \quad 1 \]

3. \[ \] Assumption
4. \[ \] \[ e \quad 2,3 \]
5. \[ \] \[ i \quad 3-4 \]
6. \[ \] \[ e \quad 5 \]
LEM (Law of the Excluded Middle)

- ____ (No antecedent)

<table>
<thead>
<tr>
<th>Step</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Assumption</td>
</tr>
<tr>
<td>2.</td>
<td>Assumption</td>
</tr>
<tr>
<td>3.</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td></td>
</tr>
</tbody>
</table>
Summary of Non-Derived Rules

<table>
<thead>
<tr>
<th>Connective</th>
<th>Introduction</th>
<th>Elimination</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image16.png" alt="Image" /></td>
<td><img src="image17.png" alt="Image" /></td>
<td><img src="image18.png" alt="Image" /></td>
</tr>
</tbody>
</table>
Summary of Derived Rules So Far

- MT (Modus Tollens)
- RAA (Reductio ad Absurdum)
- LEM (Law of the Excluded Middle)
- ⁿ⁻¹
Validity vs. Provability

- \( \varphi_1, \ldots, \varphi_n \models \psi \) means \( \psi \) is **provable** from \( \varphi_1, \ldots, \varphi_n \).

- \( \varphi_1, \ldots, \varphi_n \models \psi \) means roughly the following:

  If each of \( \varphi_i \) is true, then \( \psi \) is true.

- In other words, \( \psi \) is a **valid** conclusion from \( \varphi_1, \ldots, \varphi_n \).

- We need a definition of **truth** to make this precise.