More Smullyan

- On the island of knights and knaves, you meet A and B.

  A states “Both of us are knaves.”

- Which is each?
Smullyanism

- On the island of K&K, you meet E and F.
  
  E states “We are either both knights or both knaves.”

- Can it be determined what either of E or F are?
Truth-Table Analysis

- Let 1 stand for knight, 0 for knave.
- E states “We are either both knights or both knaves.”
- There are 4 combinations to consider:

<table>
<thead>
<tr>
<th>E</th>
<th>F</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>not possible, because E lies</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>possible, because E lies</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>not possible, because E truths</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>possible, if both knights</td>
</tr>
</tbody>
</table>

So F is definitely a knight, but we can’t tell about E.
Satisfaction

- Recall that a valuation is a mapping from proposition symbols to \{0, 1\}. It induces a value of \{0, 1\} on an entire formula.

- A valuation that induces a value of 1 in a formula is said to satisfy it.

- A valuation that induces a value of 0 in a formula is said to falsify it.
Example A

• Formula A: \((p \lor \neg q) \land (q \lor \neg p)\)
• \(v(p) = 1, v(q) = 1\) satisfies A
• \(v(p) = 1, v(q) = 0\) falsifies A
• \(v(p) = 0, v(q) = 1\) __________ A
• \(v(p) = 0, v(q) = 0\) __________ A
Example B

• Formula B: \((p \lor \neg q) \lor (q \lor \neg p)\)

• \(\nu(p) = 1, \nu(q) = 1\)  

• \(\nu(p) = 1, \nu(q) = 0\)

• \(\nu(p) = 0, \nu(q) = 1\)

• \(\nu(p) = 0, \nu(q) = 0\)
Reflection

• \( A = B \) (A and B are equivalent) iff
  A and B are satisfied by the same valuations.
Tautology

- A **tautology** is a formula that is satisfied by *every* valuation.

- Examples:
  - $p \rightarrow p$
  - $\neg p \lor p$
  - $(p \lor \neg q) \lor (q \lor \neg p)$
Contradictions

• *A contradiction* is a formula that is satisfied by *no* valuation.

• Examples:
  • \( \neg p \land p \)
  • \( \bot \)
In-Between

• Some formulas are neither tautologies nor contradictions.

• Example:
  \[ p \rightarrow \neg p \text{ is satisfied by } v(p) = 0. \]
  \[ \neg p \rightarrow p \text{ is satisfied by } v(p) = 1. \]
Satisfiable

- A formula is **satisfiable** if it is satisfied by **some** valuation.

- Every tautology is satisfiable.

- Not every satisfiable formula is a tautology.
Unsatisfiable

• A formula is unsatisfiable if it is satisfied by no valuation.

• Unsatisfiable is the same as being a contradiction.
Examples

• \( \neg p \vee p \)  
  Tautology

• \( \neg p \vee q \)  
  Satisfiable, but not a tautology

• \((p \rightarrow q) \rightarrow (q \rightarrow p)\)

• \((p \rightarrow q) \rightarrow (\neg p \rightarrow \neg q)\)

• \((p \rightarrow q) \lor (q \rightarrow p)\)
Counterexamples

- A *counterexample* for a formula is a valuation that falsifies it.

Example: \( \neg p \lor q \)

A counterexample is \( v(p) = 1, v(q) = 0 \).
Trichotomy

• Every formula is thus one of:
  • Tautology
  • Satisfiable, but not a tautology
  • Unsatisfiable
Observation

• Formula $A$ is a tautology iff
  $\neg A$ is unsatisfiable.

Recall:

• Tautology: $A$ satisfied by all valuations.
• Unsatisfiable: $A$ satisfied by no valuation.
Examples

• $\neg p \land p$  Unsatisfiable
• $\neg(\neg p \land p)$  Therefore Tautology
Unsatisfiable vs. Tautology

- These statements are equivalent for any formula $A$:
  - $A$ is a tautology (“always true”)
  - $v(A) = 1$ for every valuation $v$.
  - $v(\neg A) = \neg v(A) = 0$ for every valuation $v$.
  - $v(\neg A) = 1$ for no valuation $v$.
  - $\neg A$ is unsatisfiable (“never true”)


One-One Correspondence

- There is a contradiction for every tautology and vice-versa.

For any formula $A$, consider $\neg A$. 
Low Density of Tautologies

• Out of $2^n$ equivalence classes of $n$-variable formulas:
  • 1 class consist of tautologies (all 1’s in the truth table)
  • 1 class consists of contradictions
  • $2^n-1$ classes consist of satisfiable formulas
Effort in Characterizing

- To show a formula is a tautology seems to require checking all valuations in the worst case.

- To show a formula is not a tautology only requires finding one counterexample.
A Hot Tip

When informally looking for a counterexample for a formula of the form $A \rightarrow B$, look for $v(A) = 1$, $v(B) = 0$. This is the only form a counterexample can take.
Example Counterexample

• \(((x \land y) \rightarrow z) \rightarrow ((x \rightarrow z) \land (y \rightarrow z))\)

• By the previous tip, we want an evaluation \(v\) such that
  \(v((x \land y) \rightarrow z) = 1\) and
  \(v((x \rightarrow z) \land (y \rightarrow z)) = 0\).

• From the latter, we need \(v(x \rightarrow z) = 0\)
  [or the symmetric case \(v(y \rightarrow z) = 0\)].

• Again using the tip, we restrict to \(v(x) = 1\) and \(v(z) = 0\).

• But in this context, \(v((x \land y) \rightarrow z) = 1\) requires \(v(y) = 0\).

• So our counterexample is \(v(x) = 1, v(y) = v(z) = 0\)
Be Careful with Parsing and Mixing the language and meta-language

• “(¬A) is unsatisfiable”

vs.

• not “A is unsatisfiable” [“not” is meta-]

• They sound alike, but their meaning is different. The first is equivalent to “A is a tautology”, the second to “A is satisfiable”.
Example

• Here $p$ is a proposition symbol.
• “($\neg p$) is unsatisfiable” is false, because $v(p) = 0$ satisfies it.
• vs.
• not “$p$ is unsatisfiable” is true, because $p$ is satisfiable.
= vs. $\leftrightarrow$

- $\leftrightarrow$ is a connective. It is used \textit{within} formulas.

- $=$ is a \textit{meta}-connective. It is used \textit{between} two formulas.

- If $A$ and $B$ are formulas, $A = B$ is the same as saying $v(A) = v(B)$ for every valuation.
Thus

\[ A = B \]

iff

\[ A \leftrightarrow B \text{ is a tautology.} \]
Showing by Refutation

This means the following:

In order to show $A$ is a tautology, we show that $\neg A$ is unsatisfiable.

It is used in various methods, including:
- tableaux
- resolution
Tableaux Systems

- Tableaux systems for propositional logic can be dually viewed as both a proof systems and as an algorithm.

- This is a refutation method: determine whether a formula’s negation is unsatisfiable, in order to determine whether the formula is a tautology.
Learning this Method

• This method is easier to learn by example than it is to explain precisely.
Tableau Proof

- Construct a tree with the negation of the formula to be checked for truth as the root. The tree grows downward.

- Replace formulas higher up in the tree with simpler formulas on the branches below.

- At some point, the satisfiability or unsatisfiability of the root formula will be determined, using reasoning to be explained.
Basic Idea

• If the original formula is satisfiable, then all formulas on some root-to-leaf path are satisfiable.

• Conversely, if there is an obvious contradiction on a root-to-leaf path, then that path cannot contribute to the satisfiability of the root. This is called a closed path.

• The original formula is unsatisfiable iff all complete paths close.
"Using" a Formula on a Path

- When a formula is used, sub-formulas of it are copied to all open paths below, in a manner to be described.

- When the formula is used, it is checked, so as not to use it again.

- When there are no more usable formulas, a path is complete.
Contradiction Indicator

Such a path is called "closed" and marked with $X$.

If all paths close, the tree is **closed** and the root is unsatisfiable. Otherwise the tree is **open**.
Example of a Closed Tableau
(rules still to be explained)

\[ \neg ((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)) \checkmark \]

\[ (p \rightarrow q) \checkmark \]

\[ \neg (\neg q \rightarrow \neg p) \checkmark \]

\[ \neg q \]

\[ \neg \neg p \checkmark \]

\[ p \]

\[ \neg p \]

\[ q \]

\[ X \]

\[ X \]
Example of a Closed Tableau

\(-((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p))\)  
\((p \rightarrow q)\)  
\(-(\neg q \rightarrow \neg p)\)  
\(\neg q\)  
\(\neg \neg p\)  
\(p\)  
\(-p\)  
\(q\)  
\(X\)  
\(X\)

cannot both be true

cannot both be true
Path Construction

- There are two kinds of rules, which I call:
  - **Stacking**: Child nodes are added to the path on which the parent node occurs. These are, in a sense, *and* nodes.
  - **Splitting**: Child nodes split the path into two paths, one child on each. These are, in a sense, *or* nodes.

- The parent node is then checked off as used.
### Stacking Rules

<table>
<thead>
<tr>
<th>Parent</th>
<th>Children</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$\neg (\neg A)$</td>
<td>$A$</td>
<td>(only one)</td>
<td></td>
</tr>
<tr>
<td>$A \land B$</td>
<td>$A$</td>
<td></td>
<td>$B$</td>
</tr>
<tr>
<td>$\neg (A \lor B)$</td>
<td>$\neg A$</td>
<td>$\neg B$</td>
<td></td>
</tr>
<tr>
<td>$\neg (A \rightarrow B)$</td>
<td>$A$</td>
<td></td>
<td>$\neg B$</td>
</tr>
</tbody>
</table>

Rationale: The parent is satisfiable iff both children are (by the same valuation).
## Splitting Rules

<table>
<thead>
<tr>
<th>Parent</th>
<th>Children</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( A \lor B )</td>
<td>A</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>( A \rightarrow B )</td>
<td>( \neg A )</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>( \neg (A \land B) )</td>
<td>( \neg A )</td>
<td>( \neg B )</td>
<td></td>
</tr>
<tr>
<td>( A \leftrightarrow B )</td>
<td>A</td>
<td>B</td>
<td>( \neg A )</td>
</tr>
<tr>
<td>( \neg (A \leftrightarrow B) )</td>
<td>A</td>
<td>( \neg B )</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>( \neg A )</td>
<td>( \neg B )</td>
<td></td>
</tr>
</tbody>
</table>

**Rationale:** The parent is satisfiable iff at least one child is.
Important Note

- Rules are applied only to the outermost connective.

- We don’t apply them to sub-formulas.
Tableau Example

Is \((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)\) a tautology?

Negate: \(\neg ((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p))\)
and check for satisfiability.
Tableau Construction

\neg((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p))
Tableau Construction

\[ \neg(((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)) \, \checkmark \, \text{stack using } \rightarrow \text{ rule} \]

\[ (p \rightarrow q) \]
\[ \neg(q \rightarrow \neg p) \]
Tableau Construction

\[ \neg((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)) \checkmark \text{ stack using } \rightarrow \text{ rule} \]

\[
\begin{align*}
(p \rightarrow q) \\
\neg(\neg q \rightarrow \neg p) & \checkmark \text{ stack using } \rightarrow \text{ rule} \\
\neg q \\
\neg \neg p
\end{align*}
\]
Tableau Construction

\[-((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)) \quad \sqrt{\text{stack using } \rightarrow \text{ rule}}\]

\[(p \rightarrow q)\]
\[-(\neg q \rightarrow \neg p) \quad \sqrt{\text{stack using } \rightarrow \text{ rule}}\]
\[\neg q\]
\[-\neg p \quad \sqrt{\text{stack using } \neg \neg \text{ rule}}\]

p
Tableau Construction

\[-((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)) \quad \square \quad \text{stack using } \rightarrow \text{ rule}\]

\[(p \rightarrow q) \quad \square \quad \text{split using } \rightarrow \text{ rule}\]

\[-(\neg q \rightarrow \neg p) \quad \square \quad \text{stack using } \rightarrow \text{ rule}\]

\[-\neg q \quad \square \quad \text{stack using } \neg\neg \text{ rule}\]

\[-\neg p \quad \square \quad \text{stack using } \neg\neg \text{ rule}\]

\[-p \quad \square \quad \text{close left path}\]

\[p \quad \square \quad \text{close right path}\]

\[\neg p \quad \square \quad \text{close left path}\]

\[q \quad \square \quad \text{close right path}\]
Note

- The tableau rules already include certain logical equivalences, such as deMorgan’s rules:
  
  - **Staking** rule:
    
    \[ \neg (A \lor B) = (\neg A) \land (\neg B) \]
    
    includes \( \neg (A \lor B) = (\neg A) \land (\neg B) \)

  - **Splitting** rule:
    
    \[ \neg (A \land B) = (\neg A) \lor (\neg B) \]
    
    includes \( \neg (A \land B) = (\neg A) \lor (\neg B) \)

- This is a good way to remember them.
Tableau Example

Is \((x \land (\neg x \lor y)) \leftrightarrow (x \land y)\) a tautology?

Negate: \(\neg((x \land (\neg x \lor y)) \leftrightarrow (x \land y))\)
and check for satisfiability.
Tableau Example

\neg((x \land (\neg x \lor y)) \iff (x \land y)) ✔ checked

\neg((x \land (\neg x \lor y)) \iff (x \land y))

split

(x \land (\neg x \lor y))

stacked

\neg(x \land y)

\neg(x \land (\neg x \lor y))

(x \land y)
Tableau Example

\[\neg((x \land (\neg x \lor y)) \leftrightarrow (x \land y)) \checkmark\]
Tableau Example

\[-((x \land (\neg x \lor y)) \leftrightarrow (x \land y)) \checkmark\]

\[(x \land (\neg x \lor y)) \checkmark\]

\[-(x \land y) \checkmark\]

\[-x \checkmark\]

\[-y \checkmark\]

\[x \checkmark\]

\[\neg x \checkmark\]

\[\neg y \checkmark\]

\[\neg (x \land (\neg x \lor y))\]

\[(x \land y)\]
Tableau Example

\[ \neg((x \land (\neg x \lor y)) \leftrightarrow (x \land y)) \checkmark \]

When a node splits, the split has to take place on all open paths below. Here there is only one.
Try completing the tableau on this side, to see if this tree closes.
You Have Choices

• Parent nodes can be chosen in any order you wish.

• It is generally preferable to choose stacking nodes over splitting ones, as these do not cause the tree to branch and there will be less duplication of work.
Termination

How would you show that the tableau construction always terminates?
Other Information Gleaned

If some path is not closeable, then the literals (proposition symbols or negations thereof) spell out a valuation that satisfies the formula.
Gleaning Example

Is this a tautology?

\[ ((x \land y) \rightarrow z) \rightarrow ((x \rightarrow z) \land (y \rightarrow z)) \]

If not, what is a counterexample?
Gleaning Example

Negate and check satisfiability

\[ \neg(((x \land y) \rightarrow z) \rightarrow ((x \rightarrow z) \land (y \rightarrow z))) \]
Gleaning Example

Negate and check satisfiability

\[ \neg (((x \land y) \rightarrow z) \rightarrow ((x \rightarrow z) \land (y \rightarrow z))) \checkmark \]

\[ ((x \land y) \rightarrow z) \]

\[ \neg ((x \rightarrow z) \land (y \rightarrow z)) \]
Gleaning Example

Negate and check satisfiability

\[ \neg(((x \land y) \rightarrow z) \rightarrow ((x \rightarrow z) \land (y \rightarrow z))) \checkmark \]

\[ ((x \land y) \rightarrow z) \checkmark \]

\[ \neg((x \rightarrow z) \land (y \rightarrow z)) \]

\[ \neg(x \land y) \rightarrow z \]
Gleaning Example

Negate and check satisfiability

\[ \neg(((x \land y) \rightarrow z) \rightarrow ((x \rightarrow z) \land (y \rightarrow z))) \checkmark \]
\[ ((x \land y) \rightarrow z) \checkmark \]
\[ \neg((x \rightarrow z) \land (y \rightarrow z)) \checkmark \]
\[ \neg(x \land y) \]
\[ \neg(x \rightarrow z) \quad \neg(y \rightarrow z) \]
Gleaning Example

Negate and check satisfiability

\[-(((x \land y) \rightarrow z) \rightarrow ((x \rightarrow z) \land (y \rightarrow z))) \checkmark\]

\[((x \land y) \rightarrow z) \checkmark\]

\[-((x \rightarrow z) \land (y \rightarrow z)) \checkmark\]

\[-(x \land y) \checkmark z\]

\[-(x \rightarrow z) \checkmark - (y \rightarrow z)\]

\[x\]

\[-z\]
Gleaning Example

Negate and check satisfiability

\[ \neg (((x \land y) \rightarrow z) \rightarrow ((x \rightarrow z) \land (y \rightarrow z))) \checkmark \]

\[ ((x \land y) \rightarrow z) \checkmark \]

\[ \neg ((x \rightarrow z) \land (y \rightarrow z)) \checkmark \]

\[ \neg (x \land y) \checkmark \]

\[ \neg (x \rightarrow z) \checkmark \neg (y \rightarrow z) \]

\[ x \]

\[ \neg z \]

Closed \( \neg x \) \( \neg y \) Complete, but open

One satisfying valuation: \( v(x) = 1, v(y) = 0, v(z) = 0 \)
Gleaning Example

So is this a tautology?

$$((x \land y) \rightarrow z) \rightarrow ((x \rightarrow z) \land (y \rightarrow z))$$

No.

Counterexample:

$$v(x) = 1, v(y) = 0, v(z) = 0$$

$$((((1 \land 0) \rightarrow 0) \rightarrow ((1 \rightarrow 0) \land (0 \rightarrow 0))) = 0$$

Any other counterexamples?
Double Turnstile $|= \text{Meta-Symbol}$

- $A_1, A_2, \ldots A_n |= B$ [This is supposed to be one symbol: $|=\text{.}$]
  means that for any valuation $v$ such that $v(A_1) = v(A_2) = \ldots = v(A_n) = 1$ we must also have $v(B) = 1$.

- $A_1, A_2, \ldots A_n$ are called \textit{premises}, and $B$ the \textit{conclusion}.

- To show these by means of a tableau:
  - \textbf{Stack} each of $A_1, A_2, \ldots A_n$ (not negated)
  - \textbf{Stack} the \textit{negation} of $B$.
  - Then proceed as usual.
Example Using $\models$

• $p \rightarrow q, \neg p \rightarrow r, \neg q \rightarrow \neg r \models q$

• Start:

\[
\begin{align*}
p & \rightarrow q \\
\neg p & \rightarrow r \\
\neg q & \rightarrow \neg r \\
\neg q & \text{ negated conclusion}
\end{align*}
\]
Completed Tableau

\[ p \rightarrow q \]
\[ \neg p \rightarrow r \]
\[ \neg q \rightarrow \neg r \]

A. split

\[ \neg \neg p \]
\[ \neg \neg q \]
\[ \neg \neg \neg q \]
\[ \neg r \]
\[ \neg \neg r \]
\[ \neg \neg \neg r \]

X

✓ A. split
✓ B. split
✓ C. split

Diagram: (See accompanying text for logical structure.)
Formal Proof Systems

• A proof system is a formal system using symbol manipulation to derive formulas from other formulas (e.g. “theorems” from “axioms”).

• As with a grammar, a proof system does not rely on a specific algorithm, although algorithmic approaches can be established in some cases.
Why “Formal”? 

An advantage of a formal proof is that the proof can be checked for correctness mechanically, i.e. by a computer program (or by a meticulous, but not necessarily super-intelligent, human).
Why do we need this?

- Can’t we just stick to our truth tables, etc.?

- Truth tables are inadequate for first-order logic and beyond.

- Even for 0-order, they sometimes obscure the reasoning.
Natural Deduction
Gerhard Gentzen (1909-1945)

• Formulas are derived through a series of symbol manipulations involving introduction and elimination rules.

• The rules are organized by the logical connectives.

• Proofs can be shown as trees.

• Natural deduction provides a useful template for meta-logic proofs as well.
Comparison

• Hilbert-Ackermann is more common in early math references.

• Gentzen systems, and others, are more useful in computer science.

• For example, programming language semantics and type systems can be expressed using Gentzen-like ideas.

• Both types of systems prove the same sets of logical formulas overall, so we lose little by focusing on one or the other.
CS 81 Systems

• We will emphasize natural deduction.
• We will also discuss:
  • tableaux systems (Beth, Smullyan)
  • resolution (Prawitz, Robinson)
• We will not emphasize Hilbert-Ackermann systems.
Natural Deduction Proofs

• A natural deduction proof is a tree, usually shown upward from the root.

• The root of the tree is the thing being proved.

• The leaves of the tree are the premises.
Rules of Natural Deduction

- For each connective, there is both an
  - introduction rule: tells how to introduce the connective into a formula
  - elimination rule: tells how to remove the connective from a formula
- Formulas are interpreted as strings.
Formulas

• We’ll use $A, B, C, \ldots$ to stand for formulas.

• Proposition symbols are special cases of formulas, but formulas are not limited to propositions.
Rules

• In addition to being an essential part of a formal proof system,

Natural deduction rules are your friends.

• They will help you learn to construct and present convincing mathematical proofs.
∧ Introduction Rule

\[ \begin{array}{c}
A \\ B \\
\hline
A \land B
\end{array} \quad \land I \]

“A proof of \( A \land B \) follows from proofs of \( A \) and \( B \).”
Example

\[
\begin{array}{c}
A \quad B \\
\hline
A \land B \\
\hline
p \quad q \lor r \\
\hline
p \land (q \lor r)
\end{array}
\]

Identify \( p \) with \( A \)

Identify \( q \lor r \) with \( B \)
Cascaded Example, using \( \land \) Twice

\[
\begin{align*}
\text{p} & \quad q \lor r \\
\text{p} \land (q \lor r) & \quad s \\
(p \land (q \lor r)) & \land s
\end{align*}
\]
\(\land\) Elimination Rules

\[\frac{A \land B}{A}\ \ \ \ ^{\land E_L}\]

\[\frac{A \land B}{B}\ \ \ \ ^{\land E_R}\]

“A proof of A follows from a proof of \(A \land B\).”

“A proof of B follows from a proof of \(A \land B\).”

- Note that these rules “lose information”. This is ok, because form, rather than just content, is important.

- In our presentations, the rule subscripts will be omitted. They can be recovered from context.
Cascaded Example

\[
p \land (q \land r) \\
p \land (q \land r) \quad q \land r \quad p \land (q \land r) \\
p \quad q \\
p \land q \\
(p \land q) \land r
\]
Cascaded Example, with Rules Labeled

\[ p \land (q \land r) \land_{E_R} \]

\[ p \land (q \land r) \land_{E_L} \quad q \land r \land_{E_L} \quad p \land (q \land r) \land_{E_R} \]

\[ p \land q \land_{I} \quad q \land r \land_{E_R} \]

\[ (p \land q) \land r \land_{I} \]
Label Shortcuts

- I often eliminate L and R when it is clear. (You may too.)

- I may sometimes even eliminate the entire rule name, when it is clear. (You probably should not.)
Proof Construction

• Proofs can be constructed by
  • working from the root toward the leaves or
  • the other way around.

• The former is probably preferred, as it is more “goal-driven”.
Turnstile |— Meta-Symbol, Sequents

- $A_1, A_2, \ldots, A_n |— B$ [This is supposed to be one symbol: |—.] means that $B$ is provable from formulas in set $A_1, A_2, \ldots, A_n$.

  i.e. there is a tree with $B$ as root and $A_1, A_2, \ldots, A_n$ as leaves, where each non-leaf node is justified by a rule.

We just showed, for example, (with $n = 1$)

$$p \land (q \land r) |— (p \land q) \land r$$

A meta-expression such as the one above is called a **sequent**. Note: $|—$ appears only once in a sequent.
Box Diagrams
(aka Fitch Diagrams)

• Instead of a tree, a box diagram can be used. The diagram can represent a DAG rather than just a tree, so sometimes is more succinct.

• Numbered formulas are used in place of nodes in the tree.

• Each formula is justified as either:
  • a premise
  • derived from other formulas by a single rule, giving the numbers of those formulas.
## Box Diagram Example

\[
p \land (q \land r) \quad \vdash \quad (p \land q) \land r
\]

<table>
<thead>
<tr>
<th>Number</th>
<th>Formula</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( p \land (q \land r) )</td>
<td>premise</td>
</tr>
<tr>
<td>2</td>
<td>( p )</td>
<td>1, ( \land E_L )</td>
</tr>
<tr>
<td>3</td>
<td>( q \land r )</td>
<td>1, ( \land E_R )</td>
</tr>
<tr>
<td>4</td>
<td>( q )</td>
<td>3, ( \land E_L )</td>
</tr>
<tr>
<td>5</td>
<td>( r )</td>
<td>3, ( \land E_R )</td>
</tr>
<tr>
<td>6</td>
<td>( p \land q )</td>
<td>2, 4, ( \land I )</td>
</tr>
<tr>
<td>7</td>
<td>( (p \land q) \land r )</td>
<td>6, 5, ( \land I )</td>
</tr>
</tbody>
</table>
Comparison: Box vs Tree

\[ p \land (q \land r) \vdash (p \land q) \land r \]

<table>
<thead>
<tr>
<th>Number</th>
<th>Formula</th>
<th>Justification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( p \land (q \land r) )</td>
<td>premise</td>
</tr>
<tr>
<td>2</td>
<td>( p )</td>
<td>1, ( \land E_1 )</td>
</tr>
<tr>
<td>3</td>
<td>( q \land r )</td>
<td>1, ( \land E_2 )</td>
</tr>
<tr>
<td>4</td>
<td>( q )</td>
<td>3, ( \land E_1 )</td>
</tr>
<tr>
<td>5</td>
<td>( r )</td>
<td>3, ( \land E_2 )</td>
</tr>
<tr>
<td>6</td>
<td>( p \land q )</td>
<td>2, 4, ( \land I )</td>
</tr>
<tr>
<td>7</td>
<td>( (p \land q) \land r )</td>
<td>6, 5, ( \land I )</td>
</tr>
</tbody>
</table>

\[ \frac{p \land (q \land r) \quad q \land r}{p \land q} \quad \frac{q \land r}{(p \land q) \land r} \]

\[ \frac{p \land (q \land r) \quad q \land r}{p \land q} \quad \frac{q \land r}{(p \land q) \land r} \]
Comparison

Duplicated derivation in tree

\begin{align*}
\p & \rightarrow \q & \rightarrow \r \\
\p \land (\q \land \r) & \rightarrow \\
\q & \rightarrow \\
\q \land \r & \rightarrow \\
(p \land q) \land r & \rightarrow
\end{align*}

\begin{tabular}{|c|c|c|}
\hline
Number & Formula & Justification \\
\hline
1 & p \land (q \land r) & premise \\
2 & p & 1, \land E \\
3 & q \land r & 1, \land E \\
4 & q & 3, \land E \\
5 & r & 3, \land E \\
6 & p \land q & 2, 4, \land I \\
7 & (p \land q) \land r & 6, 5, \land I \\
\hline
\end{tabular}

Derived only once
\[ \text{\textbf{\lor} Introduction Rules} \]

\[
\[
\frac{A}{A \lor B} \quad \lor I_R
\]

\[
\frac{B}{A \lor B} \quad \lor I_L
\]

A proof of \( A \lor B \) can be derived from either a proof of \( A \) or a proof of \( B \).

The \( \lor I \) rules generally lose information.

It will rarely be the very last line of a proof. It is used in rule matching however.
→ Elimination Rule

\[
\begin{array}{c}
F  \\
F \rightarrow G  \\
\hline
G
\end{array}
\rightarrow E
\]

A proof of $G$ can be constructed from a proof of $F$ and a proof of $F \rightarrow G$.

This rule is also called by its Latin name: modus ponens (MP)
Example

• Sequent: \( p, \ r, \ p \rightarrow q \mid \neg \neg q \land r \)

• Proof

\[
\begin{array}{c}
p \\
p \rightarrow q \\
q \\
r
\hline
q \land r
\end{array}
\]

\( q \land r \)
Thinking inside the box

- Boxes can be nested arbitrarily, much as blocks in a program can.

- The inner boxes are used for sub-proofs.
Introduction Rule, Sub-Proofs

- To introduce $\rightarrow$, as in $A \rightarrow B$, we make an **temporary assumption** of $A$ and show that $B$ is derivable from it.

- To emphasize that $A$ is an assumption, we put it and the derivation inside a box, while $A \rightarrow B$ is outside.

$$
\begin{array}{c c c c c c c c}
A & \text{assumption} \\
\cdot & \cdot & \cdot \\
B & \rightarrow I \\
\hline
A \rightarrow B
\end{array}
$$

**Note:** Formulas outside the box can be used inside, but not vice-versa.

It’s very similar to scoping rules in some programming languages.
Example of $\rightarrow$I

$p \rightarrow q, \ q \rightarrow r \mid \vdash \ p \rightarrow r$

1. $p \rightarrow q$ \hspace{2cm} premise
2. $q \rightarrow r$ \hspace{2cm} premise
3. $p$ \hspace{2cm} assumption
4. $q$ \hspace{2cm} $1, 3, \rightarrow E$
5. $r$ \hspace{2cm} $4, 2, \rightarrow E$
6. $p \rightarrow r$ \hspace{2cm} $3-5, \rightarrow I$

Note: 4 uses 1, which is outside.
Note: 5 uses 2, which is outside.

Inner box

Take-away from box
Tree Version of $\rightarrow$I: Tricky

• In a tree proof, we show temporary assumptions in brackets [...].

• The bracketed formula is discharged (or cancelled) with the rule application.

• When confusion could result, a number is attached to the bracket formula and the corresponding discharging rule.
Tree Version of $\rightarrow$I Example

- Sequent: $p \rightarrow q, \quad q \rightarrow r \quad \vdash \quad p \rightarrow r$

\[
\begin{array}{c}
p \rightarrow q \quad [p]_1 \\
\hline
q \\
\hline
r \\
\hline
p \rightarrow r
\end{array}
\]

assumption 1

$\rightarrow$E

$\rightarrow$E

$\rightarrow$I$_1$

Note: A tree proof is not complete unless all assumptions are discharged.
\( \lor \) Elimination Rule

- \( \lor \) elimination requires two sub-proofs

\[
\begin{array}{c|c}
A & B \\
\cdot & \cdot \\
\cdot & \cdot \\
A \lor B & C \\
\end{array}
\]

\( \lor E \)

One rule application discharges two assumptions.

Note carefully which symbols recur in this rule and where.
\** Elimination Example**

(Justification is on same line as antecedents.)

- Sequent: \( p \lor q \vdash q \lor p \)
- Tree proof

\[
\begin{align*}
\hline
[p]_1 \lor I & [q]_2 \lor I \\
\hline
p \lor q & q \lor p & q \lor p & \lor E_{1, 2} \\
q \lor p & \\
\hline
\end{align*}
\]
\( v \) Elimination Example

1. \( p \lor q \) premise
2. \( p \) assumption
3. \( q \lor p \) 2, \( \lor I_L \)
4. \( q \) assumption
5. \( q \lor p \) 4, \( \lor I_R \)
6. \( q \lor p \) 1, 2-3, 4-5, \( \lor E \)
Proof Exercises

• \( p \land q \vdash q \land p \)

• \( p \lor (q \land r) \vdash (p \lor q) \land (p \lor r) \)
Rules for $\neg$

- One way to think of $\neg$: $\neg A$ is an abbreviation for $A \rightarrow \bot$.

- We then get $\neg E$ and $\neg I$ rules from the corresponding rules for $\rightarrow$. 
¬Elimination Rule

\[
\frac{A \quad A \to \bot}{\bot \quad \bot} \quad \rightarrow E \text{ instance}
\]

becomes

\[
\frac{A \quad \neg A}{\bot \quad \bot} \quad \neg E
\]
Introduce Rule

\[
\frac{A \rightarrow \bot}{A \rightarrow \bot} \quad \text{instance}
\]

becomes

\[
\frac{A \rightarrow \bot}{\neg A} \quad \text{I assumption}
\]
-Introduction Example (nested boxes)

1. $p \rightarrow q$  premise
2. $\neg q$  assumption
3. $p$  assumption
4. $q$  $3, 1, \rightarrow E$
5. $\bot$  $4, 2, \neg E$
6. $\neg p$  $3-5, \neg I$
7. $\neg q \rightarrow \neg p$  $2-6, \rightarrow I$
Introduction Example as a tree
(Justification is on same line as antecedents.)

\[
[p]_1 \quad p \rightarrow q \quad \rightarrow E
\]
\[
q \quad \quad \quad [\neg q]_2 \quad \neg E
\]
\[
\bot \quad \neg I_1
\]
\[
\neg p \quad \rightarrow I_2
\]
\[
\neg q \rightarrow \neg p
\]
Constructive Contradiction Rule
(aka $\bot$ Elimination)

- $\bot$  $\bot E$
  
  where $A$ is any formula

- This rule is mostly applied inside a box, because that is where $\bot$ is usually derived.
Try Proving These

• \( p \lor q, \neg p \vdash q \)

• \( \neg(p \land q), p \vdash \neg q \)
**Derived Rules vs. Sequents**

**Substitution Theorem:** For any sequent, it is clear that we could produce a derivation with proposition symbols replaced with formulas. For example, instead of

\[ p \rightarrow q \mid\mid \neg q \rightarrow \neg p \]

we could just as well derive

\[ A \rightarrow B \mid\mid \neg B \rightarrow \neg A \]

where \( A \) and \( B \) are any formulas.

We could treat the latter as a derived rule:

\[
\frac{A \rightarrow B}{\neg B \rightarrow \neg A}
\]

sometimes also called a Lemma.
Example of a Derived Rule: Modus Tollens (MT)

\[
\begin{align*}
A \rightarrow B & \quad \neg B \\
\hline
\neg A & \quad \text{MT}
\end{align*}
\]

Proof:

\[
\begin{align*}
[A]_1 & \quad A \rightarrow B \\
& \quad \rightarrow E
\end{align*}
\]

\[
\begin{align*}
B & \quad \neg B \\
\hline
\neg I_1 & \quad \neg A
\end{align*}
\]

1. \(A \rightarrow B\) \quad Premise
2. \(\neg B\) \quad Premise
3. \(A\) \quad Assumption
4. \(B\) \quad 1, 3, \(\rightarrow E\)
5. \(\bot\) \quad 2, 4, \(\neg E\)
6. \(\neg A\) \quad 3-5, \(\neg I\)
Classical Contradiction Rule
(aka RAA Rule, *not* $\neg$-Introduction)

\[
\begin{array}{c}
\neg A \\
\vdots \\
\bot \\
\hline \\
A
\end{array}
\]

*RAA (reductio ad absurdum)*

For contrast:

\[
\begin{array}{c}
A \\
\vdots \\
\bot \\
\hline \\
\neg A
\end{array}
\]

These look similar, but are *not* the same. (Recall we are dealing with strings.)

RAA is our first example of a "classical" ("non-constructive") rule.

Constructive (aka "Intuitionistic") rules are preferred, but classical rules are necessary to derive all tautologies.
Example Using RAA

1. \(-p \rightarrow -q\) premise
2. \(q\) assumption
3. \(-p\) assumption
4. \(-q\) 3, 1, \(\rightarrow E\)
5. \(\bot\) 2, 4, \(-E\)
6. \(p\) 3-5, RAA
7. \(q \rightarrow p\) 2-6, \(\rightarrow I\)
Law of the Excluded Middle (LEM)

\[ A \lor \neg A \]

This rule can be derived from the classical contra, but not from constructive contra.
Why LEM is not Constructive

Recall that, in a constructive context,

\[ \top \]

\[ A \lor \neg A \]

means there is a proof of \( A \) or a proof of \( \neg A \).

But for an \textit{arbitrary} \( A \), we cannot say there is a proof of either.
LEM

• Many (most?) mathematicians take LEM as self-evident. Others, called constructivists, do not.

• Their reasoning goes back to the notion of proving $A \lor B$. For this we need either a proof of $A$ or a proof of $B$.

• If we don’t have either a proof of $A$ or a proof $\neg A$, we have no right to assert $A \lor \neg A$. 
Non-Constructive Proof Example

• Prove: “There are two irrational numbers a, b such that $a^b$ is rational.”

• We can prove that $\sqrt{2}$ is irrational.

• Now consider $c=\sqrt{2}^\sqrt{2}$. If c is rational, then let $a = b = \sqrt{2}$.

• If c is irrational, then $c^\sqrt{2} = (\sqrt{2}^\sqrt{2})^\sqrt{2} = (\sqrt{2})^2 = 2$, so let $a = c$ and $b = \sqrt{2}$.

• But we still don’t know what two such numbers a and b are.
Middle Ground on LEM

• You don’t have to take a stand.

• You just want to know whether you are making essential use of LEM (or RAA) or not.

• It is better not to use it when you don’t have to.
DNE (or $\neg
\neg$ E)

Double-Not Elimination

This is another non-constructive rule.

$\neg\neg A$  DNE

A  where A is any formula

RAA, LEM, and DNE are equivalent in that each can be proved from the other + constructive rules.
\neg \neg A \text{ vs. } A

- Obviously these are not identical, although they are classically equivalent.
- We can easily prove $A \vdash \neg \neg A$ constructively.
- We cannot prove $\neg \neg A \vdash A$ constructively. This is DNE.
Constructive vs. Classical DeMorgan’s Rules

• **Constructive (easy):**
  \[(\neg A \lor \neg B) \vdash \neg (A \land B)\]

• **Classical (harder):**
  \[\neg (A \land B) \vdash (\neg A \lor \neg B)\]
\[(\neg A \lor \neg B) \vdash \neg (A \land B)\]

1. \(\neg A \lor \neg B\)  
   *Premise*

2. \(A \land B\)  
   *Assumption*

3. \(A\)  
   \(2, \land E_L\)

4. \(B\)  
   \(2, \land E_R\)

5. \(\neg A\)  
   *Assumption*

6. \(\bot\)  
   \(3, 5, \neg E\)

7. \(\neg B\)  
   *Assumption*

8. \(\bot\)  
   \(4, 7, \neg E\)

9. \(\bot\)  
   \(1, 5-6, 7-8, \lor E\)

10. \(\neg (A \land B)\)  
    \(2-9, \neg I\)
\[ \neg(A \land B) \quad \vdash \quad (\neg A \lor \neg B) \]

1. \( \neg(A \land B) \)  
   Premise

Try it!

\[ \neg A \lor \neg B \]
\[-(A \land B) \vdash (\neg A \lor \neg B)\]

1. \(\neg(A \land B)\) 
   Premise
2. \(\neg(\neg A \lor \neg B)\) 
   Assumption
3. \(\neg A\) 
   Assumption
4. \(\neg A \lor \neg B\) 
   3, \(\lor I_{R}\)
5. \(\bot\) 
   2, 5, \(\neg E\)
6. \(A\) 
   \(3-5, RAA\)
7. \(\neg B\) 
   Assumption
8. \(\neg A \lor \neg B\) 
   7, \(\lor I_{R}\)
9. \(\bot\) 
   2, 8, \(\neg E\)
10. \(B\) 
    \(7-9, RAA\)
11. \(A \land B\) 
    6, 10, \(\land I\)
12. \(\bot\) 
    2, 11, \(\neg E\)
13. \(\neg A \lor \neg B\) 
    2-12, \(RAA\)
Try Proving This Derived Rule

\[\begin{align*}
A \lor B & \quad \neg A \lor C \\
\hline
B \lor C
\end{align*}\]

Resolution

Can it be done constructively?
Origin of Constructive Logic

- Brouwer’s (1891-1966) Program of Intuitionism
- Gentzen: Different systems of logical calculii
- Also see: van Dalen, Logic and Structure
Glivenko’s Theorem

- A propositional formula $A$ is provable classically iff

$$\neg \neg A$$

is provable constructively.

In other words, we don’t “lose too much” by adhering to constructive logic, in the propositional case.
Example: Constructive proof of
$\neg\neg(B \lor \neg B)$

1. $\neg(B \lor \neg B)$ Assumption
2. $B$ Assumption
3. $B \lor \neg B$ 2, $\lor I_R$
4. $\bot$ 1, 3, $\neg E$
5. $\neg B$ 2-4, $\neg I$
6. $B \lor \neg B$ 5, $\lor I_L$
7. $\bot$ 1, 6, $\neg E$
8. $\neg\neg(B \lor \neg B)$ 1-7, $\neg I$
Constructive Logic in CS (aside)

Curry-Howard Correspondence:

To every proof in constructive logic there corresponds a program in the typed lambda-calculus, and vice-versa.
Hilbert-Ackermann System
(for reference only)

- *Principles of Mathematical Logic*, 1938
- Formulas are derived through a series of symbol manipulations.
- There is one rule of derivation (Modus Ponens- see later).
- There are three logical axiom schemata.
- A proof is a sequence, wherein each element is either an axiom or is derived from earlier elements using the rule.
Hilbert-Ackermann Axiom Schemata

• $A \rightarrow (B \rightarrow A)$

• $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$

• $(\neg B \rightarrow \neg A) \rightarrow (A \rightarrow B)$
  where $A$, $B$, and $C$ are any formulas.

• These may have seemed intuitive and natural to Hilbert and Ackerman.