We are working with **natural deduction proofs** $A_1 \ldots A_n \vdash B$ in propositional logic. Deduction rules so far:

→ **Conjunction:**

$$
\begin{align*}
\frac{A_1 \quad A_2}{A_1 \land A_2} & \land i \\
\frac{A_1 \land A_2}{A_1} & \land e_1 \\
\frac{A_1 \land A_2}{A_2} & \land e_2
\end{align*}
$$

→ **Disjunction:**

$$
\begin{align*}
\frac{A_1}{A_1 \lor A_2} & \lor i_1 \\
\frac{A_2}{A_1 \lor A_2} & \lor i_2 \\
\frac{A_1 \lor A_2}{B} & \lor e
\end{align*}
$$

→ **Implication:**

$$
\begin{align*}
\frac{A}{A \rightarrow B} & \rightarrow i \\
\frac{A \rightarrow B}{B} & \rightarrow e \\
\frac{A_1 \rightarrow A_2}{\neg A_1} & \text{MT}
\end{align*}
$$

→ **Negation:**

$$
\begin{align*}
\frac{A}{\bot} & \neg e \\
\frac{\neg A}{\bot} & \neg i \\
\frac{\bot}{A} & \bot e
\end{align*}
$$
Show: \[ \neg A \lor \neg B \vdash \neg (A \land B) \] (De Morgan)
Show: \( \neg A \lor \neg B \vdash \neg(A \land B) \) (De Morgan)

We can prove the left-to-right direction, but we cannot prove \( \neg(A \land B) \vdash \neg A \lor \neg B \).

We are missing a last set of rules.
Double negation
We know that $\text{sem}(\neg\neg A) = \text{sem}(A)$, for any $A$. That is, $\neg\neg A \equiv A$.

Can you derive the following rule?

$$\begin{array}{c}
\neg\neg A \\
\hline
A
\end{array} 
\neg\neg e$$

If our logic does not include $\neg\neg e$ then it is called intuitionistic logic. If our logic does include $\neg\neg e$ then it is called classical logic.
We know that $\text{sem}(\neg\neg A) = \text{sem}(A)$, for any $A$. That is, $\neg\neg A \equiv A$.

Can you derive the following rule?

$$
\begin{array}{c}
\neg\neg A \\
\hline
A
\end{array}
\text{ }\neg\neg e
$$

It turns out the above rule is not derivable and we need to add it as an axiom to the logic. But we can derive the following with the “standard” rules.

$$
\begin{array}{c}
A \\
\hline
\neg\neg A
\end{array}
\text{ }\neg\neg i
$$

If our logic does not include $\neg\neg e$ then it is called intuitionistic logic.

If our logic does include $\neg\neg e$ then it is called classical logic.
Show: \((\neg A \rightarrow \bot) \vdash A\)
Show: \((\neg A \rightarrow \bot) \vdash A\)

Proof by Contradiction (PBC)

\[
\begin{array}{c}
A \\
\hline
\bot
\end{array}

\rightarrow e

\begin{array}{c}
\bot \\
\hline
A
\end{array}

\rightarrow e

\begin{array}{c}
A \\
\hline
B
\end{array}

\rightarrow e

\begin{array}{c}
\neg A \\
A
\end{array}

\rightarrow e
\]
Show: \( \neg(A \land B) \vdash \neg A \lor \neg B \)  

De Morgan
Basic Propositional Logic Rules:

\[
\begin{align*}
\frac{A_1 \quad A_2}{A_1 \land A_2} & \quad \land i \\
\frac{A_1 \land A_2}{A_1} & \quad \land e_1 \\
\frac{A_1 \land A_2}{A_2} & \quad \land e_2 \\
\frac{A_1}{A_1 \lor A_2} & \quad \lor i_1 \\
\frac{A_2}{A_1 \lor A_2} & \quad \lor i_2 \\
\frac{A_1 \lor A_2}{B} & \quad \lor e \\
\frac{A \quad A \rightarrow B}{B} & \quad \rightarrow e \\
\frac{\bot}{\neg A} & \quad \neg e \\
\frac{\bot}{\neg e} & \quad \neg i \\
\frac{\bot}{\neg e} & \quad \neg e \\
\frac{\bot}{\neg e} & \quad \neg e
\end{align*}
\]
Law of Excluded Middle (LEM)

\[ \vdash A \lor \neg A \]

(\(\neg A \rightarrow \bot\)) \(\vdash A\)

\(\neg(A_1 \land A_2) \vdash \neg A_1 \lor \neg A_2\)

\(A \rightarrow (B_1 \lor B_2) \vdash (A \rightarrow B_1) \lor B_2\)

\(A \rightarrow B \vdash \neg A \lor B\)

PBC

DeMorgan 1

Material Implication

Note: \(\neg(A_1 \lor A_2) \vdash \neg A_1 \land \neg A_2\) (DeMorgan 2)

does not require a classical proof
ALL PROPOSITIONAL LOGIC RULES
Basic Propositional Logic Rules:

\[
\begin{align*}
\frac{A_1 \quad A_2}{A_1 \land A_2} & \text{ } \land i \\
\frac{A_1 \land A_2}{A_1} & \text{ } \land e_1 \\
\frac{A_1 \land A_2}{A_2} & \text{ } \land e_2 \\
\frac{A_1}{A_1 \lor A_2} & \text{ } \lor i_1 \\
\frac{A_2}{A_1 \lor A_2} & \text{ } \lor i_2 \\
\frac{A_1 \lor A_2}{B} & \text{ } \lor e
\end{align*}
\]
Derived Propositional Logic Rules:\textsuperscript{1}

\[ \frac{}{A} \text{ COPY} \]
\[ \frac{}{\neg\neg A} \text{ \neg\neg i} \]
\[ \frac{A_1 \rightarrow A_2}{\neg A_1} \text{ MT} \]

\[ \begin{array}{c}
\neg A \\
\vdots \\
\bot \\
\hline \\
A \\
\end{array} \quad \text{PBC (proof by contradiction)} \]

\[ \frac{A \lor \neg A}{A \lor \neg A} \text{ LEM (law of excluded middle)} \]

\textsuperscript{1}Prove their validity.
Lecture 7, Part 2:
Meta-theory of propositional logic
→ gives us a syntax to write logical propositions:

$$A ::= p \mid (\neg A) \mid (A \land A) \mid (A \lor A) \mid (A \rightarrow A)$$

→ gives us a method for syntactically proving logical entailment

$$A_1, \ldots, A_n \vdash B$$

by applying natural deduction inference rules

e.g., disjunction:

$$\begin{array}{c}
A_1 \\
A_1 \lor A_2
\end{array} \lor i_1$$

$$\begin{array}{c}
A_2 \\
A_1 \lor A_2
\end{array} \lor i_2$$

$$\begin{array}{c}
A_1 \lor A_2 \\
B
\end{array} \lor e$$

→ The semantics of the logic interpret formulas as functions (truth tables) and give us a way to find equivalent formulas (even with different truth tables):

$$A_1 \land A_2 \rightarrow A_1 \equiv r \lor \neg r$$

means

$$A_1 \land A_2 \rightarrow A_1 \models r \lor \neg r$$

and

$$r \lor \neg r \models A_1 \land A_2 \rightarrow A_1$$
Q: Is every provable statement $A_1, \ldots, A_n \vdash B$ valid according to the semantics of the logic?

In other words is the proof system sound?

**Theorem (Soundness of proof rules)**

*For any provable statement $A_1, \ldots, A_n \vdash B$ it is valid that $A_1, \ldots, A_n \models B$.**
Q: Is every provable statement \( A_1, \ldots, A_n \vdash B \) valid according to the semantics of the logic?

In other words is the proof system sound?

**Theorem (Soundness of proof rules)**

*For any provable statement \( A_1, \ldots, A_n \vdash B \) it is valid that \( A_1, \ldots, A_n \models B \).*

Proof by a form of induction. We will learn more about inductive proofs in the following weeks.

*Soundness is very important: we can’t derive something false from the proof system.*
Q: Do we have enough proof rules so that any valid \(A_1, \ldots, A_n \models B\) we can be proved syntactically as \(A_1, \ldots, A_n \vdash B\)?

In other words is the proof system complete?

Theorem (Completeness of proof rules)

For any valid sequent \(A_1, \ldots, A_n \models B\) it is provable that \(A_1, \ldots, A_n \vdash B\).

Proof: see book 1.4.4.

*Completeness means we can prove any valid propositional logic theorem, using only the syntactic proof system. This is a very strong statement, not true for many other logics.
Q: Do we have enough proof rules so that any valid $A_1, \ldots, A_n \models B$ we can be proved syntactically as $A_1, \ldots, A_n \vdash B$?
In other words is the proof system complete?

Theorem (Completeness of proof rules)
For any valid sequent $A_1, \ldots, A_n \models B$ it is provable that $A_1, \ldots, A_n \vdash B$.
Proof: see book 1.4.4.

*Completeness means we can prove any valid propositional logic theorem, using only the syntactic proof system. This is a very strong statement, not true for many other logics.

**Natural deduction is not the only sound and complete system for doing propositional proofs. Exercise 1.2.6 in the book shows another: the sequent calculus, a system of rules to transform valid sequents to other valid sequents.
Q: is it possible to write an algorithm that decides whether $A_1, \ldots, A_n \vdash B$ is a valid sequent?
Q: is it possible to write an algorithm that decides whether $A_1, \ldots, A_n \vdash B$ is a valid sequent?

We only need an algorithm to decide whether $\models A$:

$A_1, \ldots, A_n \vdash B$ by a theorem, is equivalent to
$\vdash A_1 \rightarrow \ldots \rightarrow A_n \rightarrow B$ by soundness and completeness, is equivalent to
$\models A_1 \rightarrow \ldots \rightarrow A_n \rightarrow B$
Q: is it possible to write an algorithm that decides whether $A_1, \ldots, A_n \vdash B$ is a valid sequent?

We only need an algorithm to decide whether $\models A$:

$$A_1, \ldots, A_n \vdash B$$ by a theorem, is equivalent to

$$\vdash A_1 \rightarrow \ldots \rightarrow A_n \rightarrow B$$ by soundness and completeness, is equivalent to

$$\models A_1 \rightarrow \ldots \rightarrow A_n \rightarrow B$$

There are many ways to do this. One is to turn formulas into Conjunctive Normal Form (CNF).
CNF is a formula which has the following structure:

- It contains **literals** $L$ which are either atoms (e.g., $p$) or their negation (e.g., $\neg p$)
- It composes literals into **clauses** using disjunction ($\lor$)
- It composes clauses into a **formula** using conjunction ($\land$)
CNF

CNF is a formula which has the following structure:

→ It contains **literals** $L$ which are either atoms (e.g., $p$) or their negation (e.g., $\neg p$)
→ It composes literals into **clauses** using disjunction ($\lor$)
→ It composes clauses into a **formula** using conjunction ($\land$)

example:

$$(q \lor p \lor r) \land (\neg p \lor s \lor p) \land (\neg s)$$
CNF is a formula which has the following structure:

→ It contains **literals** \( L \) which are either atoms (e.g., \( p \)) or their negation (e.g., \( \neg p \))

→ It composes literals into **clauses** using disjunction (\( \lor \))

→ It composes clauses into a **formula** using conjunction (\( \land \))

example:

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

CNF formulas **do not contain**:

→ double negation

→ implication
A CNF formula is valid iff every clause contains a literal and its negation. (why?)

Valid formulas:

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

Not valid formulas:

\[p\]
\[(P \lor q)\]
\[(q \lor p \lor r \lor \neg q) \land (\neg p \lor s \lor p) \land (s)\]
A CNF formula is valid iff every clause contains a literal and its negation. (why?)

Valid formulas:

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

Not valid formulas:

\[p\]
\[(P \lor q)\]
\[(q \lor p \lor r \lor \neg q) \land (\neg p \lor s \lor p) \land (s)\]

The above gives an efficient algorithm to check validity of CNF formulas \(O(n)\) to the size of the formula).
Every formula can be transformed to an equivalent CNF formula by the following method:

1. replace implication using the theorem: $A \rightarrow B \equiv \neg A \lor B$
2. push all negations inwards using De Morgan laws:
   
   \[ \neg (A_1 \land A_2) \equiv \neg A_1 \lor \neg A_2 \]
   \[ \neg (A_1 \lor A_2) \equiv \neg A_1 \land \neg A_2 \]
3. remove double negations: $\neg \neg A \equiv A$
4. distribute and over or: $(A_1 \land A_2) \lor B \equiv (A_1 \lor B) \land (A_2 \lor B)$
Every formula can be transformed to an equivalent CNF formula by the following method:

1. replace implication using the theorem: $A \rightarrow B \equiv \neg A \lor B$
2. push all negations inwards using De Morgan laws:
   $$\neg(A_1 \land A_2) \equiv \neg A_1 \lor \neg A_2 \quad \neg(A_1 \lor A_2) \equiv \neg A_1 \land \neg A_2$$
3. remove double negations: $\neg \neg A \equiv A$
4. distribute **and** over **or**: $(A_1 \land A_2) \lor B \equiv (A_1 \lor B) \land (A_2 \lor B)$

The above conversion outputs in the worst case an exponentially large formula ($O(2^n)$) to the size of the input formula).
Convert to CNF and check the validity of the formulas:

\[ \rightarrow \neg p \land q \rightarrow p \land (r \rightarrow q) \]
\[ \rightarrow p \rightarrow q \rightarrow r \]
\[ \rightarrow (p \rightarrow q \rightarrow r) \rightarrow (p \land q \rightarrow r) \]
\[ \rightarrow \bot \rightarrow p \]
\[ \rightarrow p \rightarrow \top \]
Satisfiability: Given $A$, is there a model which makes $A$ true?

Q: Can we decide satisfiability?
Satisfiability: Given $A$, is there a model which makes $A$ true?
Q: Can we decide satisfiability?

Theorem
*The satisfiability problem is decidable, and NP-complete*

So there are known algorithms but they are not efficient in the worst case.
Satisfiability: Given $A$, is there a model which makes $A$ true?

Q: Can we decide satisfiability?

Theorem
The satisfiability problem is decidable, and NP-complete

So there are known algorithms but they are not efficient in the worst case.

But there are efficient algorithms for a some CNF formulas: Horn clauses
A CNF formula is a horn formula if all its clauses have at most one positive literal.

\[ \neg p \lor \neg q \lor r \quad \text{becomes} \quad p \land q \rightarrow r \]
\[ \neg p \lor \neg q \quad \text{becomes} \quad p \land q \rightarrow \bot \]
\[ p \quad \text{becomes} \quad \top \rightarrow p \]
Algorithm: Inputs a Horn formula and maintains a list of literals, $\bot$, and $\top$ in the formula. It marks the literals in this list as follows:

→ it marks $\top$ if it exists in the list

→ If there is a conjunct

$$L_1 \land \ldots L_n \rightarrow L'$$

and all $L_1 \land \ldots L_n$ are marked then mark $L'$. Repeat (2) until no more such conjuncts.

→ if $\bot$ marked then output “unsatisfiable” and stop

→ else output “satisfiable” and stop
Algorithm: Inputs a Horn formula and maintains a list of literals, ⊥, and ⊤ in the formula.
It marks the literals in this list as follows:

→ it marks ⊤ if it exists in the list
→ If there is a conjunct

\[ L_1 \land \ldots L_n \rightarrow L' \]

and all \( L_1 \land \ldots L_n \) are marked then mark \( L' \). Repeat (2) until no more such conjuncts.
→ if ⊥ marked then output “unsatisfiable” and stop
→ else output “satisfiable” and stop

This is a \( O(n) \) algorithm.
EXAMPLES:

\[ \rightarrow (p \land q \land s \rightarrow p) \land (q \land r \rightarrow p) \land (p \land s \rightarrow s) \]
\[ \rightarrow (p \land q \land s \rightarrow \bot) \land (q \land r \rightarrow p) \land (\top \rightarrow s) \]
\[ \rightarrow (p \land q \land s \rightarrow \bot) \land (p \land s \rightarrow q) \land (s \rightarrow p) \land (\top \rightarrow s) \]
\[ \rightarrow (p \land q \land s \rightarrow \bot) \land (s \rightarrow p) \land (\top \rightarrow s) \]