Classical Propositional Logic

Peter Baumgartner

http://users.cecs.anu.edu.au/~baumgart/

Ph: 02 6218 3717

Data61/CSIRO and ANU

July 2017
Classical Logic and Reasoning Problems

\( A_1: \) Socrates is a human
\( A_2: \) All humans are mortal

Translation into first-order logic:
\( A_1: \) \( \text{human(socrates)} \)
\( A_2: \) \( \forall X \ (\text{human}(X) \rightarrow \text{mortal}(X)) \)

Reasoning problems
Which of the following statements hold true? (\( \models \) means “entails”)

1. \( \{A_1, A_2\} \models \text{mortal(socrates)} \)
2. \( \{A_1, A_2\} \models \text{mortal(apollo)} \)
3. \( \{A_1, A_2\} \not\models \text{mortal(socrates)} \)
4. \( \{A_1, A_2\} \not\models \text{mortal(apollo)} \)
5. \( \{A_1, A_2\} \models \neg \text{mortal(socrates)} \)
6. \( \{A_1, A_2\} \models \neg \text{mortal(apollo)} \)
Classical Logic and Reasoning Problems

$A_1$: Socrates is a human
$A_2$: All humans are mortal

Translation into first-order logic:
$A_1$: $\text{human(socrates)}$
$A_2$: $\forall X \ (\text{human}(X) \rightarrow \text{mortal}(X))$

Reasoning problems
Which of the following statements hold true? ($\models$ means “entails”)
1. $\{A_1, A_2\} \models \text{mortal(socrates)}$
Classical Logic and Reasoning Problems

$A_1$: Socrates is a human
$A_2$: All humans are mortal

Translation into first-order logic:
$A_1$: \text{human}(socrates)
$A_2$: \forall X \ (\text{human}(X) \rightarrow \text{mortal}(X))$

Reasoning problems
Which of the following statements hold true? ($\models$ means “entails”)

1. $\{A_1, A_2\} \models \text{mortal}(socrates)$
2. $\{A_1, A_2\} \models \text{mortal}(apollo)$
Classical Logic and Reasoning Problems

\( A_1: \) Socrates is a human
\( A_2: \) All humans are mortal

Translation into first-order logic:
\( A_1: \) human(socrates)
\( A_2: \) \( \forall X \left( \text{human}(X) \rightarrow \text{mortal}(X) \right) \)

Reasoning problems
Which of the following statements hold true? (\( \models \) means “entails”)

1. \( \{ A_1, A_2 \} \models \text{mortal}(\text{socrates}) \)
2. \( \{ A_1, A_2 \} \models \text{mortal}(\text{apollo}) \)
3. \( \{ A_1, A_2 \} \not\models \text{mortal}(\text{socrates}) \)
Classical Logic and Reasoning Problems

$A_1$: Socrates is a human
$A_2$: All humans are mortal

Translation into first-order logic:

$A_1$: $\text{human(\text{socrates})}$
$A_2$: $\forall X \ (\text{human}(X) \rightarrow \text{mortal}(X))$

Reasoning problems

Which of the following statements hold true? ($\models$ means “entails”)

1. $\{A_1, A_2\} \models \text{mortal(socrates)}$
2. $\{A_1, A_2\} \models \text{mortal(apollo)}$
3. $\{A_1, A_2\} \not\models \text{mortal(socrates)}$
4. $\{A_1, A_2\} \not\models \text{mortal(apollo)}$

Topics of these lectures

What do these statements exactly mean?

Algorithms/procedures for reasoning problems like the above

Next: some applications
Classical Logic and Reasoning Problems

\( A_1: \) Socrates is a human
\( A_2: \) All humans are mortal

Translation into first-order logic:
\( A_1: \) human(socrates)
\( A_2: \) \( \forall X ( \text{human}(X) \rightarrow \text{mortal}(X)) \)

Reasoning problems
Which of the following statements hold true? (\( \models \) means “entails”)

1. \( \{ A_1, A_2 \} \models \text{mortal}(\text{socrates}) \)
2. \( \{ A_1, A_2 \} \models \text{mortal}(\text{apollo}) \)
3. \( \{ A_1, A_2 \} \not\models \text{mortal}(\text{socrates}) \)
4. \( \{ A_1, A_2 \} \not\models \text{mortal}(\text{apollo}) \)
5. \( \{ A_1, A_2 \} \models \neg \text{mortal}(\text{socrates}) \)
Classical Logic and Reasoning Problems

\( A_1: \) Socrates is a human
\( A_2: \) All humans are mortal

Translation into first-order logic:
\( A_1: \) \text{human}(\text{socrates})
\( A_2: \) \( \forall X (\text{human}(X) \rightarrow \text{mortal}(X)) \)

Reasoning problems

Which of the following statements hold true? (\( \models \) means "entails")

1. \( \{A_1, A_2\} \models \text{mortal}(\text{socrates}) \)
2. \( \{A_1, A_2\} \models \text{mortal}(\text{apollo}) \)
3. \( \{A_1, A_2\} \not\models \text{mortal}(\text{socrates}) \)
4. \( \{A_1, A_2\} \not\models \text{mortal}(\text{apollo}) \)
5. \( \{A_1, A_2\} \models \neg \text{mortal}(\text{socrates}) \)
6. \( \{A_1, A_2\} \models \neg \text{mortal}(\text{apollo}) \)
Classical Logic and Reasoning Problems

$A_1$: Socrates is a human
$A_2$: All humans are mortal

Translation into first-order logic:

$A_1$: $\text{human(socrates)}$
$A_2$: $\forall X (\text{human}(X) \rightarrow \text{mortal}(X))$

Reasoning problems

Which of the following statements hold true? $\models$ means “entails”

1. $\{A_1, A_2\} \models \text{mortal(socrates)}$
2. $\{A_1, A_2\} \models \text{mortal(apollo)}$
3. $\{A_1, A_2\} \not\models \text{mortal(socrates)}$
4. $\{A_1, A_2\} \not\models \text{mortal(apollo)}$
5. $\{A_1, A_2\} \models \neg \text{mortal(socrates)}$
6. $\{A_1, A_2\} \models \neg \text{mortal(apollo)}$

Topics of these lectures

- What do these statements exactly mean?
- Algorithms/procedures for reasoning problems like the above
Classical Logic and Reasoning Problems

A₁: Socrates is a human
A₂: All humans are mortal

Translation into first-order logic:

A₁: human(socrates)
A₂: ∀X (human(X) → mortal(X))

Reasoning problems
Which of the following statements hold true? (|= means “entails”)

1. \{A₁, A₂\} |= mortal(socrates)
2. \{A₁, A₂\} |= mortal(apollo)
3. \{A₁, A₂\} \not|= mortal(socrates)
4. \{A₁, A₂\} \not|= mortal(apollo)
5. \{A₁, A₂\} |= ¬mortal(socrates)
6. \{A₁, A₂\} |= ¬mortal(apollo)

Topics of these lectures

▶ What do these statements exactly mean?
▶ Algorithms/procedures for reasoning problems like the above

Next: some applications
“Application”: Mathematical Theorem Proving

First-Order Logic

Can express (mathematical) structures, e.g. groups

\[
\begin{align*}
\forall x \ 1 \cdot x &= x \\
\forall x \ x^{-1} \cdot x &= 1
\end{align*}
\]

\[
\begin{align*}
\forall x \ x \cdot 1 &= x \\
\forall x \ x \cdot x^{-1} &= 1
\end{align*}
\]  \hspace{1cm} (N)

\[
\forall x, y, z \ (x \cdot y) \cdot z &= x \cdot (y \cdot z)
\]  \hspace{1cm} (A)

Reasoning

Object level: It follows \( \forall x \ (x \cdot x) = 1 \) \( \rightarrow \) \( \forall x, y \ x \cdot y = y \cdot x \)

Meta-level: the word problem for groups is decidable

Automated Reasoning

Computer program to provide the above conclusions
“Application”: Mathematical Theorem Proving

First-Order Logic
Can express (mathematical) structures, e.g. groups

\[ \forall x \ 1 \cdot x = x \quad \forall x \ x \cdot 1 = x \quad (N) \]
\[ \forall x \ x^{-1} \cdot x = 1 \quad \forall x \ x \cdot x^{-1} = 1 \quad (I) \]
\[ \forall x, y, z \ (x \cdot y) \cdot z = x \cdot (y \cdot z) \quad (A) \]

Reasoning

- Object level: It follows \( \forall x \ (x \cdot x) = 1 \rightarrow \forall x, y \ x \cdot y = y \cdot x \)
- Meta-level: the word problem for groups is decidable
“Application”: Mathematical Theorem Proving

First-Order Logic
Can express (mathematical) structures, e.g. groups

\[ \forall x \ 1 \cdot x = x \]
\[ \forall x \ x^{-1} \cdot x = 1 \]

\( \forall x \ x \cdot 1 = x \) \quad (N)
\( \forall x \ x \cdot x^{-1} = 1 \) \quad (I)

\[ \forall x, y, z \ (x \cdot y) \cdot z = x \cdot (y \cdot z) \] \quad (A)

Reasoning

- **Object level:** It follows \( \forall x \ (x \cdot x) = 1 \rightarrow \forall x, y \ x \cdot y = y \cdot x \)
- **Meta-level:** the word problem for groups is decidable

Automated Reasoning

Computer program to provide the above conclusions *automatically*
Application: Compiler Validation

Problem: prove equivalence of source and target program

1: y := 1
2: if z = x*x*x
3: then y := x*x + y
4: endif

1: y := 1
2: R1 := x*x
3: R2 := R1*x
4: jmpNE(z,R2,6)
5: y := R1+1

To prove: (indexes refer to values at line numbers; index 0 = initial values)

From $y_1 = 1 \land z_0 = x_0 \times x_0 \times x_0 \land y_3 = x_0 \times x_0 + y_1$

and $y'_1 = 1 \land R1_2 = x'_0 \times x'_0 \land R2_3 = R1_2 \times x'_0 \land z'_0 = R2_3$

$\land y'_5 = R1_2 + 1 \land x_0 = x'_0 \land y_0 = y'_0 \land z_0 = z'_0$

it follows $y_3 = y'_5$
Application: Constraint Solving

The n-queens problem:

**Given:** An $n \times n$ chessboard

**Question:** Is it possible to place $n$ queens so that no queen attacks any other?

A solution for $n = 8$

$p[1] = 6$
$p[2] = 3$
$p[3] = 5$
$p[4] = 8$
$p[5] = 1$
$p[6] = 4$
$p[7] = 2$
$p[8] = 7$
Application: Constraint Solving

Formalization in sorted first-order logic:

\[ n : \mathbb{Z} \]  
\[ p : \mathbb{Z} \rightarrow \mathbb{Z} \]  
\[ n = 8 \]

\[ \forall i : \mathbb{Z} \ j : \mathbb{Z} \ (1 \leq i \land i \leq n \land i + 1 \leq j \land j < n \Rightarrow \]
\[ p(i) \neq p(j) \land p(i) + i \neq p(j) + j \land p(i) - i \neq p(j) - j \]  
(Queens)

\[ p(1) = 1 \lor p(1) = 2 \lor \cdots \lor p(1) = 8 \quad (p(1) \in \{1, \ldots, n\}) \]

\[ \vdots \]

\[ p(8) = 1 \lor p(8) = 2 \lor \cdots \lor p(8) = 8 \quad (p(n) \in \{1, \ldots, n\}) \]

Logic: Integer arithmetic, quantifiers, “free” symbol \( p \)

Task: Find a satisfying interpretation \( I \) (a model) and evaluate \( I(p(1)), \ldots, I(p(n)) \) to read off the answer
The n-queens has variable symmetry: mapping $p[i] \mapsto p[n + 1 - i]$ preserves solutions, for any $n$.

Therefore, it is justified to add (to the formalization) a constraint $p[1] < p[n]$, for search space pruning.

But how can we know that the problem has symmetries? This is a theorem proving task!
We need two “copies” (Queens\_p) and (Queens\_q) of the constraint:

\[
\begin{align*}
n & : \mathbb{Z} & \text{(Declaration of } n) \\
p, q & : \mathbb{Z} \mapsto \mathbb{Z} & \text{(Declaration of } p, q) \\
perm & : \mathbb{Z} \mapsto \mathbb{Z} & \text{(Declaration of } perm) \\
\forall i : \mathbb{Z} \; j : \mathbb{Z} \; (1 \leq i \land i \leq n \land i + 1 \leq j \land j < n \Rightarrow \\
\quad p(i) \neq p(j) \land p(i) + i \neq p(j) + j \land p(i) - i \neq p(j) - j) & \quad \text{(Queens\_p)} \\
\forall i : \mathbb{Z} \; j : \mathbb{Z} \; (1 \leq i \land i \leq n \land i + 1 \leq j \land j < n \Rightarrow \\
\quad q(i) \neq q(j) \land q(i) + i \neq q(j) + j \land q(i) - i \neq q(j) - j) & \quad \text{(Queens\_q)} \\
\forall i : \mathbb{Z} \; perm(i) = n + 1 - i & \quad \text{(Def. permutation)}
\end{align*}
\]

**Logic:** Integer arithmetic, quantifiers, “free” symbol \( p \)

**Task:** Prove logical consequence

\((\text{Queens}\_p) \land (\forall i : \mathbb{Z} \; q(i) = p(perm(i))) \Rightarrow (\text{Queens}\_q)\)
Issues

- Previous slides gave motivation: *logical analysis of systems*
  System can be “anything that makes sense” and can be described using logic (group theory, computer programs, ...)

- Propositional logic is not very expressive; but it admits *complete* and *terminating* (and sound, and “fast”) reasoning procedures

- First-order logic is expressive but not too expressive; it admits *complete* (and sound, and “reasonably fast”) reasoning procedures

- So, reasoning with it can be automated on computer. BUT
  - How to do it in the first place: suitable calculi?
  - How to do it efficiently: search space control?
  - How to do it optimally: reasoning support for specific theories like equality and arithmetic?

- The lecture will touch on some of these issues and explain basic approaches to their solution
Contents

Lectures 1 – 5: Propositional logic: syntax, semantics, reasoning algorithms, important properties  
(Slides in part thanks to Aaron Bradley)

Lecture 6–10: First-order logic: syntax, semantics, reasoning procedures, important properties
Propositional Logic (PL)

PL Syntax

**Atom**
- Truth symbols \( \top \) ("true") and \( \bot \) ("false")
- Propositional variables \( P, Q, R, P_1, Q_1, R_1, \cdots \)

**Literal**
- Atom \( \alpha \) or its negation \( \neg \alpha \)

**Formula**
- Atom or application of a logical connective to formulae \( F, F_1, F_2 \)
  - \( \neg F \) "not" (negation)
  - \( F_1 \land F_2 \) "and" (conjunction)
  - \( F_1 \lor F_2 \) "or" (disjunction)
  - \( F_1 \rightarrow F_2 \) "implies" (implication)
  - \( F_1 \leftrightarrow F_2 \) "if and only if" (iff)

Speaking formally, formulas are defined *inductively*
Example:

formula $F : (P \land Q) \rightarrow (\top \lor \neg Q)$

atoms: $P, Q, \top$

literal: $\neg Q$

subformulas: $P \land Q, \top \lor \neg Q$

abbreviation (leave parenthesis away)

\[
F : P \land Q \rightarrow \top \lor \neg Q
\]
PL Semantics (meaning)

Formula $F + \text{Interpretation } I = \text{Truth value}$

(true, false)

Interpretation

$I : \{ P \mapsto \text{true, } Q \mapsto \text{false, } \cdots \}$
PL Semantics (meaning)

Formula $F$ + Interpretation $I$ = Truth value
   $(true, false)$

Interpretation

   $I : \{ P \mapsto true, Q \mapsto false, \cdots \}$

Evaluation of $F$ under $I$:

<table>
<thead>
<tr>
<th>$F$</th>
<th>$\neg F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

where 0 corresponds to value false

true
PL Semantics (meaning)

Formula $F + $ Interpretation $I = $ Truth value (true, false)

Interpretation

$I : \{ P \mapsto \text{true}, Q \mapsto \text{false}, \cdots \}$

Evaluation of $F$ under $I$:

\[
\begin{array}{c|c}
F & \neg F \\
\hline
0 & 1 \\
1 & 0 \\
\end{array}
\]

where 0 corresponds to value false

1 true

\[
\begin{array}{c|c|c|c|c|c|c|c}
F_1 & F_2 & F_1 \land F_2 & F_1 \lor F_2 & F_1 \to F_2 & F_1 \leftrightarrow F_2 \\
\hline
0 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 \\
\end{array}
\]
Example:

\[ F : P \land Q \rightarrow P \lor \neg Q \]

\[ I : \{ P \leftrightarrow \text{true}, Q \leftrightarrow \text{false} \} \]

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Q</td>
<td>\neg Q</td>
<td>P \land Q</td>
<td>P \lor \neg Q</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1 = true 0 = false
Example:

\[ F : P \land Q \rightarrow P \lor \neg Q \]
\[ I : \{ P \mapsto \text{true}, Q \mapsto \text{false} \} \]

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Q</td>
<td>\neg Q</td>
<td>P \land Q</td>
<td>P \lor \neg Q</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1 = true \quad 0 = false

\[ F \text{ evaluates to true under } I \]
Inductive Definition of PL’s Semantics

\[ I \models F \] if \( F \) evaluates to \( \text{true} \) under \( I \) ("\( I \) satisfies \( F \)"")

\[ I \not\models F \] false under \( I \) ("\( I \) falsifies \( F \)"")
Inductive Definition of PL’s Semantics

\[ I \models F \quad \text{if } F \text{ evaluates to true under } I \quad ("I \text{ satisfies } F") \]
\[ I \not\models F \quad \text{false under } I \quad ("I \text{ falsifies } F") \]

**Base Case:**
\[ I \models \top \]
\[ I \not\models \bot \]
\[ I \models P \iff I[P] = \text{true} \]
Inductive Definition of PL’s Semantics

\[ I \models F \text{ if } F \text{ evaluates to true under } I \text{ ("} I \text{ satisfies } F \text{"")} \]
\[ I \not\models F \text{ if } F \text{ evaluates to false under } I \text{ ("} I \text{ falsifies } F \text{"")} \]

**Base Case:**
\[ I \models \top \]
\[ I \not\models \bot \]
\[ I \models P \iff I[P] = \text{true} \]
\[ I \not\models P \iff I[P] = \text{false} \]
Inductive Definition of PL’s Semantics

\( I \models F \) if \( F \) evaluates to true under \( I \) (“\( I \) satisfies \( F \)”)

\( I \not\models F \) false under \( I \) (“\( I \) falsifies \( F \)”)

**Base Case:**

\( I \models \top \)

\( I \not\models \bot \)

\( I \models P \) iff \( I[P] = \text{true} \)

\( I \not\models P \) iff \( I[P] = \text{false} \)

**Inductive Case:**

\( I \models \neg F \) iff \( I \not\models F \)

\( I \models F_1 \land F_2 \) iff \( I \models F_1 \) and \( I \models F_2 \)

\( I \models F_1 \lor F_2 \) iff \( I \models F_1 \) or \( I \models F_2 \)

\( I \models F_1 \rightarrow F_2 \) iff, if \( I \models F_1 \) then \( I \models F_2 \)

\( I \models F_1 \leftrightarrow F_2 \) iff, \( I \models F_1 \) and \( I \models F_2 \),

or \( I \not\models F_1 \) and \( I \not\models F_2 \)
Inductive Definition of PL’s Semantics

\[ I \models F \text{ if } F \text{ evaluates to true under } I \text{ ("} I \text{ satisfies } F \text{")} \]
\[ I \not\models F \text{ iffalse under } I \text{ ("} I \text{ falsifies } F \text{"") } \]

Base Case:
\[ I \models \top \]
\[ I \not\models \bot \]
\[ I \models P \text{ iff } I[P] = \text{true} \]
\[ I \not\models P \text{ iff } I[P] = \text{false} \]

Inductive Case:
\[ I \models \neg F \text{ iff } I \not\models F \]
\[ I \models F_1 \land F_2 \text{ iff } I \models F_1 \text{ and } I \models F_2 \]
\[ I \models F_1 \lor F_2 \text{ iff } I \models F_1 \text{ or } I \models F_2 \]
\[ I \models F_1 \rightarrow F_2 \text{ iff, if } I \models F_1 \text{ then } I \models F_2 \]
\[ I \models F_1 \leftrightarrow F_2 \text{ iff, } I \models F_1 \text{ and } I \models F_2, \text{ or } I \not\models F_1 \text{ and } I \not\models F_2 \]

Note:
\[ I \not\models F_1 \rightarrow F_2 \text{ iff } I \models F_1 \text{ and } I \not\models F_2 \]
Example:

\[ F : P \land Q \rightarrow P \lor \neg Q \]

\[ I : \{ P \mapsto \text{true}, \ Q \mapsto \text{false} \} \]
Example:

\[ F : P \land Q \rightarrow P \lor \neg Q \]

\[ I : \{ P \mapsto \text{true}, \; Q \mapsto \text{false} \} \]

1. \( I \models P \) since \( I[P] = \text{true} \)
Example:

\[ F : P \land Q \to P \lor \neg Q \]

\[ I : \{ P \mapsto \text{true}, \ Q \mapsto \text{false} \} \]

1. \( I \models P \) since \( I[P] = \text{true} \)
2. \( I \not\models Q \) since \( I[Q] = \text{false} \)
Example:

\[ F : P \land Q \rightarrow P \lor \neg Q \]

\[ I : \{ P \mapsto \text{true}, \ Q \mapsto \text{false} \} \]

1. \( I \models P \) since \( I[P] = \text{true} \)
2. \( I \not\models Q \) since \( I[Q] = \text{false} \)
3. \( I \models \neg Q \) by 2 and \( \neg \)
Example:

\[ F : P \land Q \rightarrow P \lor \neg Q \]

\[ I : \{ P \mapsto \text{true}, \ Q \mapsto \text{false} \} \]

1. \( I \models P \) since \( I[P] = \text{true} \)
2. \( I \not\models Q \) since \( I[Q] = \text{false} \)
3. \( I \models \neg Q \) by 2 and \( \neg \)
4. \( I \not\models P \land Q \) by 2 and \( \land \)
Example:

\[ F : P \land Q \rightarrow P \lor \neg Q \]

\[ I : \{ P \mapsto \text{true}, \ Q \mapsto \text{false} \} \]

1. \[ I \models P \] since \( I[P] = \text{true} \)
2. \[ I \not\models Q \] since \( I[Q] = \text{false} \)
3. \[ I \models \neg Q \] by 2 and \( \neg \)
4. \[ I \not\models P \land Q \] by 2 and \( \land \)
5. \[ I \models P \lor \neg Q \] by 1 and \( \lor \)
Example:

\[ F : P \land Q \rightarrow P \lor \neg Q \]

\[ I : \{ P \mapsto \text{true}, \ Q \mapsto \text{false} \} \]

1. \( I \models P \) since \( I[P] = \text{true} \)
2. \( I \not\models Q \) since \( I[Q] = \text{false} \)
3. \( I \models \neg Q \) by 2 and \( \neg \)
4. \( I \not\models P \land Q \) by 2 and \( \land \)
5. \( I \models P \lor \neg Q \) by 1 and \( \lor \)
6. \( I \models F \) by 4 and \( \rightarrow \) Why?
Example:

\[ F : P \land Q \rightarrow P \lor \neg Q \]

\[ I : \{ P \mapsto \text{true}, \ Q \mapsto \text{false} \} \]

1. \( I \models P \) since \( I[P] = \text{true} \)
2. \( I \not\models Q \) since \( I[Q] = \text{false} \)
3. \( I \models \neg Q \) by 2 and \( \neg \)
4. \( I \not\models P \land Q \) by 2 and \( \land \)
5. \( I \models P \lor \neg Q \) by 1 and \( \lor \)
6. \( I \models F \) by 4 and \( \rightarrow \) Why?

Thus, \( F \) is true under \( I \).
Example:

\[ F : P \land Q \rightarrow P \lor \neg Q \]

\[ I : \{ P \mapsto \text{true}, \ Q \mapsto \text{false} \} \]

1. \( I \models P \) since \( I[P] = \text{true} \)
2. \( I \not\models Q \) since \( I[Q] = \text{false} \)
3. \( I \models \neg Q \) by 2 and \( \neg \)
4. \( I \not\models P \land Q \) by 2 and \( \land \)
5. \( I \models P \lor \neg Q \) by 1 and \( \lor \)
6. \( I \models F \) by 4 and \( \rightarrow \) Why?

Thus, \( F \) is true under \( I \).

Notation

Extend interpretation \( I \) to formulas \( F \):

\[ I[F] = \begin{cases} 
\text{true} & \text{if } I \models F \\
\text{false} & \text{otherwise } (I \not\models F) 
\end{cases} \]
Inductive Proofs

**Induction on the structure of formulas**

To prove that a property $\mathcal{P}$ holds for every formula $F$ it suffices to show the following:

**Induction start:** show that $\mathcal{P}$ holds for every base case formula $A$.

**Induction step:** Assume that $\mathcal{P}$ holds for arbitrary formulas $F_1$ and $F_2$ (*induction hypothesis*).

Show that $\mathcal{P}$ follows for every inductive case formula built with $F_1$ and $F_2$. 

**Example**

Lemma 1

Let $F$ be a formula, and $I$ and $J$ be interpretations. If $I[\mathcal{P}] = J[\mathcal{P}]$ for every propositional variable $\mathcal{P}$ occurring in $F$ then $I[F] = J[F]$ (equivalently: $J|\mathcal{P}=I$ iff $J|\mathcal{P}=I$).
Inductive Proofs

Induction on the structure of formulas
To prove that a property $\mathcal{P}$ holds for every formula $F$ it suffices to show the following:

**Induction start:** show that $\mathcal{P}$ holds for every base case formula $A$

**Induction step:** Assume that $\mathcal{P}$ holds for arbitrary formulas $F_1$ and $F_2$ (*induction hypothesis*).
Show that $\mathcal{P}$ follows for every inductive case formula built with $F_1$ and $F_2$

**Example**

**Lemma 1** Let $F$ be a formula, and $I$ and $J$ be interpretations. If $I[P] = J[P]$ for every propositional variable $P$ occurring in $F$ then $I[F] = J[F]$ (equivalently: $J \models F$ iff $J \models F$).
**Example**

**Lemma 1**  Let $F$ be a formula, and $I$ and $J$ be interpretations. If $I[P] = J[P]$ for every propositional variable $P$ occurring in $F$ then $I[F] = J[F]$.


Induction start
If $F = \top$ or $F = \bot$ then trivially $I[F] = J[F]$.
Example

**Lemma 1**  Let $F$ be a formula, and $I$ and $J$ be interpretations. If $I[P] = J[P]$ for every propositional variable $P$ occurring in $F$ then $I[F] = J[F]$.

Induction step
Case 1: $F = \neg G$ for some formula $G$. 
Example


Induction step
Case 1: $F = \neg G$ for some formula $G$.
If $I[F] = \text{true}$ then

$$I[F] = \text{true} \iff I[\neg G] = \text{true} \iff I[G] = \text{false} \iff J[G] = \text{false} \iff J[\neg G] = \text{true} \iff J[F] = \text{true} \quad (F = \neg G)$$

($F = \neg G$)
(Semantics of $\neg$)
(Induction hypothesis)
(Semantics of $\neg$)
($F = \neg G$)

If $I[F] = \text{false}$: analogously
**Example**

**Lemma 1**  Let $F$ be a formula, and $I$ and $J$ be interpretations. If $I[P] = J[P]$ for every propositional variable $P$ occurring in $F$ then $I[F] = J[F]$.

**Induction step**

**Case 1:** $F = \neg G$ for some formula $G$.

If $I[F] = \text{true}$ then

- $I[F] = \text{true}$ iff
  - $I[\neg G] = \text{true}$ iff $(F = \neg G)$ (Semantics of $\neg$)
  - $I[G] = \text{false}$ iff (Induction hypothesis) (Semantics of $\neg$)
  - $J[\neg G] = \text{true}$ iff $(F = \neg G)$ (Semantics of $\neg$)
  - $J[F] = \text{true}$

If $I[F] = \text{false}$: analogously
Example

**Lemma 1**  Let $F$ be a formula, and $I$ and $J$ be interpretations. If $I[P] = J[P]$ for every propositional variable $P$ occurring in $F$ then $I[F] = J[F]$.

**Induction step**

Case 2: $F = G \land H$ for some formulas $G$ and $H$.

If $I[F] = \text{true}$ then

$$I[F] = \text{true} \iff I[G \land H] = \text{true} \quad (F = G \land H)$$

$$I[G] = \text{true} \text{ and } I[H] = \text{true} \quad \text{(Semantics of } \land)$$

$$J[G] = \text{true} \text{ and } J[H] = \text{true} \quad \text{(Induction hypothesis 2x)}$$

$$J[G \land H] = \text{true} \quad \text{(Semantics of } \land)$$

$$J[F] = \text{true} \quad (F = G \land H)$$

If $I[F] = \text{false}$: analogously

Cases 3, 4 and 5 for $\lor$, $\rightarrow$ and $\leftrightarrow$: analogously
Satisfiability and Validity

$F$ satisfiable iff there exists an interpretation $I$ such that $I \models F$.

$F$ valid iff for all interpretations $I$, $I \models F$.

\[
\begin{array}{ccc}
P & \land & Q \\ 
0 & 0 & 0 \\ 
0 & 1 & 0 \\ 
1 & 0 & 1 \\ 
1 & 1 & 1 \\
\end{array}
\]

Thus $F$ is valid.

\[
\begin{array}{ccc}
P \land Q & & \neg Q \\ 
P \lor \neg Q & & \\
0 & 0 & 1 \\ 
0 & 1 & 0 \\ 
1 & 0 & 1 \\ 
1 & 1 & 1 \\
\end{array}
\]

$F$ is valid iff \(\neg F\) is unsatisfiable.
Satisfiability and Validity

F satisfiable iff there exists an interpretation I such that I \models F.

F valid iff for all interpretations I, I \models F.

F is valid iff \neg F is unsatisfiable

Method 1: Truth Tables

Example

\[ F : P \land Q \rightarrow P \lor \neg Q \]

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P \land Q</th>
<th>\neg Q</th>
<th>P \lor \neg Q</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Satisfiability and Validity

$F$ satisfiable iff there exists an interpretation $I$ such that $I \models F$.

$F$ valid iff for all interpretations $I$, $I \models F$.

$$F \text{ is valid iff } \neg F \text{ is unsatisfiable}$$

Method 1: Truth Tables

Example: $F : P \land Q \rightarrow P \lor \neg Q$

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$P \land Q$</th>
<th>$\neg Q$</th>
<th>$P \lor \neg Q$</th>
<th>$F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Thus $F$ is valid.
Example

$F : P \lor Q \rightarrow P \land Q$

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$P \lor Q$</th>
<th>$P \land Q$</th>
<th>$F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
### Example

$F : P \lor Q \rightarrow P \land Q$

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$P \lor Q$</th>
<th>$P \land Q$</th>
<th>$F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Thus $F$ is satisfiable, but invalid.
Examples

Which of the following formulas is satisfiable, which is valid?

1. $F_1 : P \land Q$

2. $F_2 : \neg(P \land Q)$

3. $F_3 : P \lor \neg P$

4. $F_4 : \neg(P \lor \neg P)$

5. $F_5 : (P \to Q) \land (P \lor Q) \land \neg Q$
Examples

Which of the following formulas is satisfiable, which is valid?

1. $F_1 : P \land Q$
   satisfiable, not valid

2. $F_2 : \neg(P \land Q)$

3. $F_3 : P \lor \neg P$

4. $F_4 : \neg(P \lor \neg P)$

5. $F_5 : (P \rightarrow Q) \land (P \lor Q) \land \neg Q$
Examples

Which of the following formulas is satisfiable, which is valid?

1. \( F_1 : P \land Q \)
   satisfiable, not valid

2. \( F_2 : \neg(P \land Q) \)
   satisfiable, not valid

3. \( F_3 : P \lor \neg P \)

4. \( F_4 : \neg(P \lor \neg P) \)

5. \( F_5 : (P \rightarrow Q) \land (P \lor Q) \land \neg Q \)
Examples

Which of the following formulas is satisfiable, which is valid?

1. $F_1 : P \land Q$
   satisfiable, not valid
2. $F_2 : \neg(P \land Q)$
   satisfiable, not valid
3. $F_3 : P \lor \neg P$
   satisfiable, valid
4. $F_4 : \neg(P \lor \neg P)$
5. $F_5 : (P \rightarrow Q) \land (P \lor Q) \land \neg Q$
Examples

Which of the following formulas is satisfiable, which is valid?

1. $F_1 : P \land Q$
   satisfiable, not valid

2. $F_2 : \neg(P \land Q)$
   satisfiable, not valid

3. $F_3 : P \lor \neg P$
   satisfiable, valid

4. $F_4 : \neg(P \lor \neg P)$
   unsatisfiable, not valid

5. $F_5 : (P \rightarrow Q) \land (P \lor Q) \land \neg Q$
Examples

Which of the following formulas is satisfiable, which is valid?

1. $F_1 : P \land Q$
   satisfiable, not valid

2. $F_2 : \neg (P \land Q)$
   satisfiable, not valid

3. $F_3 : P \lor \neg P$
   satisfiable, valid

4. $F_4 : \neg (P \lor \neg P)$
   unsatisfiable, not valid

5. $F_5 : (P \rightarrow Q) \land (P \lor Q) \land \neg Q$
   unsatisfiable, not valid
Method 2: Semantic Argument ("Tableau Calculus")

Proof rules

\[
\frac{I \models \neg F}{I \not\models F}
\]

\[
\frac{I \not\models \neg F}{I \models F}
\]

\[
\frac{I \models F \land G}{I \models F \quad I \models G}
\]

\[
\frac{I \not\models F \land G}{I \not\models F \quad I \not\models G}
\]

\[
\frac{I \models F \lor G}{I \models F \quad I \models G}
\]

\[
\frac{I \not\models F \lor G}{I \not\models F \quad I \not\models G}
\]

\[
\frac{I \models F \rightarrow G}{I \not\models F \quad I \models G}
\]

\[
\frac{I \not\models F \rightarrow G}{I \not\models F \quad I \models G}
\]

\[
\frac{I \models F \leftrightarrow G}{I \models F \land G \quad I \not\models F \lor G}
\]

\[
\frac{I \not\models F \leftrightarrow G}{I \models F \land \neg G \quad I \not\models \neg F \lor G}
\]

\[
\frac{I \models F}{I \not\models F}
\]

\[
\frac{I \not\models F}{I \not\models \bot}
\]
Example 1: Prove

\[ F : P \land Q \rightarrow P \lor \neg Q \] is valid.

Let’s assume that \( F \) is not valid and that \( I \) is a falsifying interpretation.

1. \( I \not\models P \land Q \rightarrow P \lor \neg Q \) assumption
Example 1: Prove

\[ F : P \land Q \rightarrow P \lor \neg Q \] is valid.

Let’s assume that \( F \) is not valid and that \( I \) is a falsifying interpretation.

1. \( I \not| = P \land Q \rightarrow P \lor \neg Q \) assumption
2. \( I | = P \land Q \) 1 and \( \land \rightarrow \)
3. \( I \not| = P \lor \neg Q \) 1 and \( \lor \rightarrow \)
4. \( I | = \bot \) 4 and 5 are contradictory

Thus \( F \) is valid.
Example 1: Prove

F: \( P \land Q \rightarrow P \lor \neg Q \) is valid.

Let’s assume that \( F \) is not valid and that \( I \) is a falsifying interpretation.

1. \( I \nmodels P \land Q \rightarrow P \lor \neg Q \) assumption
2. \( I \models P \land Q \) 1 and \( \rightarrow \)
3. \( I \nmodels P \lor \neg Q \) 1 and \( \rightarrow \)
Example 1: Prove

\[ F : P \land Q \rightarrow P \lor \neg Q \] is valid.

Let’s assume that \( F \) is not valid and that \( I \) is a falsifying interpretation.

1. \( I \not\models P \land Q \rightarrow P \lor \neg Q \)  assumption
2. \( I \models P \land Q \)  1 and \( \rightarrow \)
3. \( I \not\models P \lor \neg Q \)  1 and \( \rightarrow \)
4. \( I \models P \)  2 and \( \land \)
Example 1: Prove

\[ F : P \land Q \rightarrow P \lor \neg Q \] is valid.

Let’s assume that \( F \) is not valid and that \( I \) is a falsifying interpretation.

1. \( I \not\models P \land Q \rightarrow P \lor \neg Q \) assumption
2. \( I \models P \land Q \) 1 and \( \rightarrow \)
3. \( I \not\models P \lor \neg Q \) 1 and \( \rightarrow \)
4. \( I \models P \) 2 and \( \land \)
5. \( I \not\models P \) 3 and \( \lor \)
Example 1: Prove

\[ F : P \land Q \to P \lor \neg Q \] is valid.

Let’s assume that \( F \) is not valid and that \( I \) is a falsifying interpretation.

1. \( I \not\models P \land Q \to P \lor \neg Q \) assumption
2. \( I \models P \land Q \) 1 and \( \to \)
3. \( I \models P \lor \neg Q \) 1 and \( \to \)
4. \( I \models P \) 2 and \( \land \)
5. \( I \not\models P \) 3 and \( \lor \)
6. \( I \models \bot \) 4 and 5 are contradictory
Example 1: Prove

\[ F : P \land Q \rightarrow P \lor \neg Q \]  is valid.

Let’s assume that \( F \) is not valid and that \( I \) is a falsifying interpretation.

1. \( I \not|= P \land Q \rightarrow P \lor \neg Q \) \text{ assumption}
2. \( I |= P \land Q \) \hspace{1cm} 1 \text{ and } \rightarrow
3. \( I \not|= P \lor \neg Q \) \hspace{1cm} 1 \text{ and } \rightarrow
4. \( I |= P \) \hspace{1cm} 2 \text{ and } \land
5. \( I \not|= P \) \hspace{1cm} 3 \text{ and } \lor
6. \( I |= \bot \) \hspace{1cm} 4 \text{ and } 5 \text{ are contradictory}

Thus \( F \) is valid.
Example 2: Prove

\[ F : (P \rightarrow Q) \land (Q \rightarrow R) \rightarrow (P \rightarrow R) \] is valid.

Let’s assume that \( F \) is not valid.

1. \( I \not\models F \) 
   assumption
Example 2: Prove

\[ F : (P \rightarrow Q) \land (Q \rightarrow R) \rightarrow (P \rightarrow R) \] is valid.

Let’s assume that \( F \) is not valid.

1. \( I \not\models F \) 
2. \( I \models (P \rightarrow Q) \land (Q \rightarrow R) \)  
   
   \[ 1 \text{ and } \rightarrow \]
Example 2: Prove

\[ F : (P \rightarrow Q) \land (Q \rightarrow R) \rightarrow (P \rightarrow R) \] is valid.

Let’s assume that \( F \) is not valid.

1. \( I \nmid F \) \hspace{1cm} \text{assumption}
2. \( I \models (P \rightarrow Q) \land (Q \rightarrow R) \) \hspace{1cm} 1 and \( \rightarrow \)
3. \( I \nmid P \rightarrow R \) \hspace{1cm} 1 and \( \rightarrow \)
Example 2: Prove

\[ F : (P \to Q) \land (Q \to R) \to (P \to R) \] is valid.

Let’s assume that \( F \) is not valid.

1. \( I \not \models F \)  
   assumption
2. \( I \models (P \to Q) \land (Q \to R) \)  
   1 and \( \to \)
3. \( I \not \models P \to R \)  
   1 and \( \to \)
4. \( I \models P \)  
   3 and \( \to \)
Example 2: Prove

\[ F : (P \to Q) \land (Q \to R) \to (P \to R) \] is valid.

Let’s assume that \( F \) is not valid.

1. \( I \nvdash F \) \hspace{2cm} assumption
2. \( I \models (P \to Q) \land (Q \to R) \) \hspace{1cm} 1 and \( \to \)
3. \( I \nvdash P \to R \) \hspace{1cm} 1 and \( \to \)
4. \( I \models P \) \hspace{2cm} 3 and \( \to \)
5. \( I \nvdash R \) \hspace{2cm} 3 and \( \to \)
Example 2: Prove

\[ F : \ (P \rightarrow Q) \land (Q \rightarrow R) \rightarrow (P \rightarrow R) \] is valid.

Let’s assume that \( F \) is not valid.

1. \( I \not\models F \) assumption
2. \( I \models (P \rightarrow Q) \land (Q \rightarrow R) \) 1 and \( \rightarrow \)
3. \( I \not\models P \rightarrow R \) 1 and \( \rightarrow \)
4. \( I \models P \) 3 and \( \rightarrow \)
5. \( I \not\models R \) 3 and \( \rightarrow \)
6. \( I \models P \rightarrow Q \) 2 and of \( \land \)
Example 2: Prove

\[ F : (P \rightarrow Q) \land (Q \rightarrow R) \rightarrow (P \rightarrow R) \] is valid.

Let’s assume that \( F \) is not valid.

1. \( I \not\models F \) assumption
2. \( I \models (P \rightarrow Q) \land (Q \rightarrow R) \) 1 and \( \rightarrow \)
3. \( I \not\models P \rightarrow R \) 1 and \( \rightarrow \)
4. \( I \models P \) 3 and \( \rightarrow \)
5. \( I \not\models R \) 3 and \( \rightarrow \)
6. \( I \models P \rightarrow Q \) 2 and of \( \land \)
7. \( I \models Q \rightarrow R \) 2 and of \( \land \)
Two cases from 6

\begin{align*}
8a. & \quad I \not\models P \quad 6 \text{ and } \rightarrow \\
9a. & \quad I \models \bot \quad 4 \text{ and } 8a \text{ are contradictory}
\end{align*}
Two cases from 6

8a. $I \not\models P$  6 and $\rightarrow$

9a. $I \models \bot$  4 and 8a are contradictory

and

8b. $I \models Q$  6 and $\rightarrow$

Our assumption is incorrect in all cases — $F$ is valid.
Two cases from 6

\[8a. \; l \nmid P \quad 6 \text{ and } \rightarrow\]
\[9a. \; l \mid \bot \quad 4 \text{ and } 8a \text{ are contradictory}\]

and

\[8b. \; l \mid Q \quad 6 \text{ and } \rightarrow\]

Two cases from 7

\[9ba. \; l \nmid Q \quad 7 \text{ and } \rightarrow\]
\[10ba. \; l \mid \bot \quad 8b \text{ and } 9ba \text{ are contradictory}\]
Two cases from 6

8a. $I \not \models P$ 6 and $\rightarrow$
9a. $I \models \bot$ 4 and 8a are contradictory

and

8b. $I \models Q$ 6 and $\rightarrow$

Two cases from 7

9ba. $I \not \models Q$ 7 and $\rightarrow$
10ba. $I \models \bot$ 8b and 9ba are contradictory

and

9bb. $I \models R$ 7 and $\rightarrow$
10bb. $I \models \bot$ 5 and 9bb are contradictory
Two cases from 6

8a. \( I \nvdash P \) 6 and \( \rightarrow \)

9a. \( I \models \bot \) 4 and 8a are contradictory

and

8b. \( I \models Q \) 6 and \( \rightarrow \)

Two cases from 7

9ba. \( I \nvdash Q \) 7 and \( \rightarrow \)

10ba. \( I \models \bot \) 8b and 9ba are contradictory

and

9bb. \( I \models R \) 7 and \( \rightarrow \)

10bb. \( I \models \bot \) 5 and 9bb are contradictory

Our assumption is incorrect in all cases — \( F \) is valid.
Example 3: Is

\[ F : P \lor Q \rightarrow P \land Q \] valid?

Let’s assume that \( F \) is not valid.

We have to derive a contradiction in both cases for \( F \) to be valid.

Falsifying interpretation:

\[ I_1 : \{ P \mapsto \text{true}, \, Q \mapsto \text{false} \} \]

\[ I_2 : \{ Q \mapsto \text{true}, \, P \mapsto \text{false} \} \]
Example 3: Is
\[ F : P \lor Q \rightarrow P \land Q \text{ valid?} \]

Let’s assume that \( F \) is not valid.

1. \( I \not\models P \lor Q \rightarrow P \land Q \) assumption
Example 3: Is

\[ F : P \lor Q \rightarrow P \land Q \] valid?

Let’s assume that \( F \) is not valid.

1. \( I \not\models P \lor Q \rightarrow P \land Q \) assumption
2. \( I \models P \lor Q \) 1 and \( \rightarrow \)
Example 3: Is

\[ F : P \lor Q \rightarrow P \land Q \] valid?

Let’s assume that \( F \) is not valid.

1. \( I \not\models P \lor Q \rightarrow P \land Q \) assumption
2. \( I \models P \lor Q \) 1 and \( \rightarrow \)
3. \( I \not\models P \land Q \) 1 and \( \rightarrow \)
Example 3: Is 

\[ F : P \lor Q \rightarrow P \land Q \] 

valid?

Let’s assume that \( F \) is not valid.

1. \( I \not\models P \lor Q \rightarrow P \land Q \) assumption
2. \( I \models P \lor Q \) 1 and \( \rightarrow \)
3. \( I \not\models P \land Q \) 1 and \( \rightarrow \)

Two options

4a. \( I \models P \) 2 or
5a. \( I \not\models Q \) 3
Example 3: Is

\[ F : P \lor Q \rightarrow P \land Q \] valid?

Let’s assume that \( F \) is not valid.

1. \( I \not\models P \lor Q \rightarrow P \land Q \) assumption
2. \( I \models P \lor Q \) 1 and \( \rightarrow \)
3. \( I \not\models P \land Q \) 1 and \( \rightarrow \)

Two options

4a. \( I \models P \) 2 or 4b. \( I \models Q \) 2

5a. \( I \not\models Q \) 3 5b. \( I \not\models P \) 3

We cannot derive a contradiction.

\( F \) is not valid.

Falsifying interpretation:

\[ I_1 : \{ P \mapsto \text{true}, Q \mapsto \text{false} \} \]

\[ I_2 : \{ Q \mapsto \text{true}, P \mapsto \text{false} \} \]

We have to derive a contradiction in both cases for \( F \) to be valid.
Example 3: Is

\[ F : P \lor Q \rightarrow P \land Q \] valid?

Let’s assume that \( F \) is not valid.

1. \( I \not\models P \lor Q \rightarrow P \land Q \) assumption
2. \( I \models P \lor Q \) 1 and \( \rightarrow \)
3. \( I \not\models P \land Q \) 1 and \( \rightarrow \)

Two options

4a. \( I \models P \) 2 or 4b. \( I \models Q \) 2
5a. \( I \not\models Q \) 3 5b. \( I \not\models P \) 3

We cannot derive a contradiction. \( F \) is not valid.

Falsifying interpretation:

\( I_1 : \{ P \leftrightarrow \text{true}, \ Q \leftrightarrow \text{false} \} \quad I_2 : \{ Q \leftrightarrow \text{true}, \ P \leftrightarrow \text{false} \} \)

We have to derive a contradiction in both cases for \( F \) to be valid.
Equivalence

$F_1$ and $F_2$ are equivalent ($F_1 \iff F_2$)

iff for all interpretations $I$, $I \models F_1 \iff F_2$

To prove $F_1 \iff F_2$ show $F_1 \iff F_2$ is valid.
Equivalence

$F_1$ and $F_2$ are equivalent ($F_1 \iff F_2$)
iff for all interpretations $I$, $I \models F_1 \iff F_2$

To prove $F_1 \iff F_2$ show $F_1 \iff F_2$ is valid.

$F_1$ implies $F_2$ ($F_1 \Rightarrow F_2$)
iff for all interpretations $I$, $I \models F_1 \Rightarrow F_2$
Equivalence

$F_1$ and $F_2$ are equivalent ($F_1 \iff F_2$)

iff for all interpretations $I$, $I \models F_1 \iff F_2$

To prove $F_1 \iff F_2$ show $F_1 \iff F_2$ is valid.

$F_1$ implies $F_2$ ($F_1 \Rightarrow F_2$)

iff for all interpretations $I$, $I \models F_1 \rightarrow F_2$

$F_1 \iff F_2$ and $F_1 \Rightarrow F_2$ are not formulae!
Proposition 1 (Substitution Theorem)

Assume $F_1 \iff F_2$. If $F$ is a formula with at least one occurrence of $F_1$ as a subformula then $F \iff F'$, where $F'$ is obtained from $F$ by replacing some occurrence of $F_1$ in $F$ by $F_2$. 

Proof.
(Sketch) By induction on the formula structure. For the induction start, if $F = F_1$ then $F' = F_2$, and $F \iff F'$ follows from $F_1 \iff F_2$.

The proof of the induction step is similar to the proof of Lemma 1.

Proposition 1 is relevant for conversion of formulas into normal form, which requires replacing subformulas by equivalent ones.
Proposition 1 (Substitution Theorem)

Assume $F_1 \iff F_2$. If $F$ is a formula with at least one occurrence of $F_1$ as a subformula then $F \iff F'$, where $F'$ is obtained from $F$ by replacing some occurrence of $F_1$ in $F$ by $F_2$.

Proof.
(Sketch) By induction on the formula structure. For the induction start, if $F = F_1$ then $F' = F_2$, and $F \iff F'$ follows from $F_1 \iff F_2$. The proof of the induction step is similar to the proof of Lemma 1.

Proposition 1 is relevant for conversion of formulas into normal form, which requires replacing subformulas by equivalent ones.
Normal Forms

1. Negation Normal Form (NNF)

Negations appear only in literals. (only ¬, ∧, ∨)

To transform $F$ to equivalent $F'$ in NNF use recursively the following template equivalences (left-to-right):

$$
\neg \neg F_1 \iff F_1 \quad \neg \top \iff \bot \quad \neg \bot \iff \top
$$

$$
\neg (F_1 \land F_2) \iff \neg F_1 \lor \neg F_2
$$

$$
\neg (F_1 \lor F_2) \iff \neg F_1 \land \neg F_2
$$

\{De Morgan’s Law\}

$$
F_1 \rightarrow F_2 \iff \neg F_1 \lor F_2
$$

$$
F_1 \iff F_2 \iff (F_1 \rightarrow F_2) \land (F_2 \rightarrow F_1)
$$
Normal Forms

1. **Negation Normal Form (NNF)**

   Negations appear only in literals. (only $\neg$, $\land$, $\lor$)

   To transform $F$ to equivalent $F'$ in NNF use recursively the following template equivalences (left-to-right):

   $$
   \neg\neg F_1 \iff F_1 \quad \neg \top \iff \bot \quad \neg \bot \iff \top \\
   \neg (F_1 \land F_2) \iff \neg F_1 \lor \neg F_2 \\
   \neg (F_1 \lor F_2) \iff \neg F_1 \land \neg F_2 \\
   F_1 \rightarrow F_2 \iff \neg F_1 \lor F_2 \\
   F_1 \leftrightarrow F_2 \iff (F_1 \rightarrow F_2) \land (F_2 \rightarrow F_1)
   $$

   **De Morgan’s Law**

   $$
   \neg (F_1 \land F_2) \iff \neg F_1 \lor \neg F_2 \\
   \neg (F_1 \lor F_2) \iff \neg F_1 \land \neg F_2
   $$

   **Example**: Convert $F : \neg(P \rightarrow \neg(P \land Q))$ to NNF

   $F' : \neg(\neg P \lor \neg(P \land Q)) \rightarrow \lor$

   $F'' : \neg\neg P \land \neg\neg(P \land Q) \quad \text{De Morgan’s Law}$

   $F''' : P \land P \land Q \quad \neg\neg$

   $F'''$ is equivalent to $F$ ($F''' \iff F$) and is in NNF
2. **Disjunctive Normal Form (DNF)**

Disjunction of conjunctions of literals

\[ \bigvee_i \bigwedge_j \ell_{i,j} \quad \text{for literals } \ell_{i,j} \]

To convert \( F \) into equivalent \( F' \) in DNF, transform \( F \) into NNF and then use the following template equivalences (left-to-right):

\[
\begin{align*}
(F_1 \lor F_2) \land F_3 & \iff (F_1 \land F_3) \lor (F_2 \land F_3) \\
F_1 \land (F_2 \lor F_3) & \iff (F_1 \land F_2) \lor (F_1 \land F_3)
\end{align*}
\]

\( \dist \)
2. **Disjunctive Normal Form (DNF)**

Disjunction of conjunctions of literals

\[ \bigvee_i \bigwedge_j \ell_{i,j} \quad \text{for literals } \ell_{i,j} \]

To convert \( F \) into equivalent \( F' \) in DNF, transform \( F \) into NNF and then use the following template equivalences (left-to-right):

\[
\begin{align*}
(F_1 \lor F_2) \land F_3 & \iff (F_1 \land F_3) \lor (F_2 \land F_3) \\
F_1 \land (F_2 \lor F_3) & \iff (F_1 \land F_2) \lor (F_1 \land F_3)
\end{align*}
\]

**Example:** Convert

\( F : (Q_1 \lor \neg\neg Q_2) \land (\neg R_1 \rightarrow R_2) \) into DNF

\( F' : (Q_1 \lor Q_2) \land (R_1 \lor R_2) \) \quad \text{in NNF}

\( F'' : (Q_1 \land (R_1 \lor R_2)) \lor (Q_2 \land (R_1 \lor R_2)) \) \quad \text{dist}

\( F''' : (Q_1 \land R_1) \lor (Q_1 \land R_2) \lor (Q_2 \land R_1) \lor (Q_2 \land R_2) \) \quad \text{dist}

\( F''' \) is equivalent to \( F \) \( (F''' \iff F) \) and is in DNF
3. **Conjunctive Normal Form (CNF)**

Conjunction of disjunctions of literals

\[ \bigwedge_i \bigvee_j \ell_{i,j} \quad \text{for literals } \ell_{i,j} \]

To convert \( F \) into equivalent \( F' \) in CNF, transform \( F \) into NNF and then use the following template equivalences (left-to-right):

\[
(F_1 \land F_2) \lor F_3 \iff (F_1 \lor F_3) \land (F_2 \lor F_3)
\]

\[
F_1 \lor (F_2 \land F_3) \iff (F_1 \lor F_2) \land (F_1 \lor F_3)
\]

**Relevance:** DPLL and Resolution both work with CNF
Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

Decides the satisfiability of PL formulae in CNF, or clause sets.
Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

Decides the satisfiability of PL formulae in CNF, or clause sets

Clause
A (propositional) clause is a disjunction of literals

Convention
A formula in CNF is taken as a set of clauses. Example:

\[(A \lor B) \land (C \lor \neg A) \land (D \lor \neg C \lor \neg A) \land (\neg D \lor \neg B)\]

Clause Set

Typical Application: Proof by Refutation

To prove the validity of

\[Axiom_1 \land \cdots \land Axiom_n \implies Conjecture\]

it suffices to prove that the CNF of

\[Axiom_1 \land \cdots \land Axiom_n \land \neg Conjecture\]

is unsatisfiable.
Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

Decides the satisfiability of PL formulae in CNF, or clause sets

Clause
A (propositional) clause is a disjunction of literals

Convention
A formula in CNF is taken as a set of clauses. Example:

\[(A \lor B) \land (C \lor \neg A) \land (D \lor \neg C \lor \neg A) \land (\neg D \lor \neg B)\] CNF

\[\{A \lor B, C \lor \neg A, D \lor \neg C \lor \neg A, \neg D \lor \neg B\}\] Clause Set
Davis-Putnam-Logemann-Loveland (DPLL) Algorithm

Decides the satisfiability of PL formulae in CNF, or clause sets

Clause
A (propositional) clause is a disjunction of literals

Convention
A formula in CNF is taken as a set of clauses. Example:

\[(A \lor B) \land (C \lor \neg A) \land (D \lor \neg C \lor \neg A) \land (\neg D \lor \neg B)\] CNF

\[\{A \lor B, C \lor \neg A, D \lor \neg C \lor \neg A, \neg D \lor \neg B}\] Clause Set

Typical Application: Proof by Refutation
To prove the validity of

\[Axiom_1 \land \cdots \land Axiom_n \Rightarrow Conjecture\]

it suffices to prove that the CNF of

\[Axiom_1 \land \cdots \land Axiom_n \land \neg Conjecture\]

is unsatisfiable
DPLL Interpretations

DPLL works with trees whose nodes are labelled with literals

Consistency
No branch contains the labels A and ¬A, for no A

Every branch in a tree is taken as a (consistent) set of its literals

A consistent set of literals S is taken as an interpretation:

- if A ∈ S then (A ↦ true) ∈ I
- if ¬A ∈ S then (A ↦ false) ∈ I
- if A /∈ S and ¬A /∈ S then (A ↦ false) ∈ I
DPLL Interpretations

DPLL works with trees whose nodes are labelled with literals

Consistency
No branch contains the labels $A$ and $\neg A$, for no $A$

Every branch in a tree is taken as a (consistent) set of its literals

A consistent set of literals $S$ is taken as an interpretation:

- if $A \in S$ then $(A \mapsto \text{true}) \in I$
- if $\neg A \in S$ then $(A \mapsto \text{false}) \in I$
- if $A \notin S$ and $\neg A \notin S$ then $(A \mapsto \text{false}) \in I$

Example

$\{A, \neg B, D\}$ stands for

$I : \{A \mapsto \text{true}, \ B \mapsto \text{false}, \ C \mapsto \text{false}, \ D \mapsto \text{true}\}$

Model

A model for a clause set $N$ is an interpretation $I$ such that $I \models N$. 
DPLL as a Semantic Tree Method

(1) $A \lor B$   (2) $C \lor \neg A$   (3) $D \lor \neg C \lor \neg A$   (4) $\neg D \lor \neg B$

$\langle$empty tree$\rangle$

- A Branch stands for an interpretation
- *Purpose of splitting*: satisfy a clause that is currently falsified
- Close branch if some clause is plainly falsified by it ($\star$)
DPLL as a Semantic Tree Method

(1) $A \lor B$  (2) $C \lor \neg A$  (3) $D \lor \neg C \lor \neg A$  (4) $\neg D \lor \neg B$

$\{A\} \models A \lor B$
$\{A\} \not\models C \lor \neg A$
$\{A\} \models D \lor \neg C \lor \neg A$
$\{A\} \models \neg D \lor \neg B$

- A Branch stands for an interpretation
- *Purpose of splitting:* satisfy a clause that is currently falsified
- Close branch if some clause is plainly falsified by it (⋆)
DPLL as a Semantic Tree Method

(1) $A \lor B$  
(2) $C \lor \neg A$  
(3) $D \lor \neg C \lor \neg A$  
(4) $\neg D \lor \neg B$

\{A, C\} $\models$ $A \lor B$
\{A, C\} $\models$ $C \lor \neg A$
\{A, C\} $\not\models$ $D \lor \neg C \lor \neg A$
\{A, C\} $\models$ $\neg D \lor \neg B$

- A Branch stands for an interpretation
- **Purpose of splitting**: satisfy a clause that is currently falsified
- Close branch if some clause is plainly falsified by it ($\star$)
DPLL as a Semantic Tree Method

(1) $A \lor B$  (2) $C \lor \neg A$  (3) $D \lor \neg C \lor \neg A$  (4) $\neg D \lor \neg B$

\[
\{A, C, D\} \models A \lor B \\
\{A, C, D\} \models C \lor \neg A \\
\{A, C, D\} \models D \lor \neg C \lor \neg A \\
\{A, C, D\} \models \neg D \lor \neg B
\]

Model $\{A, C, D\}$ found.

- A Branch stands for an interpretation
- **Purpose of splitting:** satisfy a clause that is currently falsified
- Close branch if some clause is plainly falsified by it ($\star$)
DPLL as a Semantic Tree Method

(1) \( A \lor B \)  
(2) \( C \lor \neg A \)  
(3) \( D \lor \neg C \lor \neg A \)  
(4) \( \neg D \lor \neg B \)

\{B\} \models A \lor B  
\{B\} \models C \lor \neg A  
\{B\} \models D \lor \neg C \lor \neg A  
\{B\} \models \neg D \lor \neg B

Model \{B\} found.

- A Branch stands for an interpretation
- *Purpose of splitting*: satisfy a clause that is currently falsified
- Close branch if some clause is plainly falsified by it (*)
DPLL Pseudocode

```
function DPLL(N)
    %% N is a set of clauses
    %% returns true if N satisfiable, false otherwise
    while N contains a unit clause \{L\\}
        N := simplify(N, L)
    if N = {} then return true
    if ⊥ ∈ N then return false
    L := choose-literal(N) %% any literal that occurs in N - ”decision literal”
    if DPLL(simplify(N, L))
        then return true
    else return DPLL(simplify(N, ¬L));

function simplify(N, L) %% also called unit propagation
    remove all clauses from N that contain L
    delete ¬L from all remaining clauses %% possibly get empty clause ⊥
    return the resulting clause set
```
Simplify Examples

function simplify(N, L) \%
also called unit propagation
remove all clauses from N that contain L
delete \neg L from all remaining clauses \%
possibly get empty clause \bot
return the resulting clause set

simplify({A \lor \neg B, C \lor \neg A, D \lor \neg C \lor \neg A, \neg D \lor \neg B}, A) =

Simplify Examples

1. \textbf{function} simplify\((N, \ L)\) \ %% also called \textit{unit propagation}\n2. remove all clauses from \textit{N} that contain \textit{L}\n3. delete \textit{¬L} from all remaining clauses \ %% possibly get empty clause \(⊥\) \n4. return the resulting clause set

\[
\text{simplify}\left(\{A \lor ¬B, \ C \lor ¬A, \ D \lor ¬C \lor ¬A, \ ¬D \lor ¬B\}, \ A\right) = \{C, \ D \lor ¬C, \ ¬D \lor ¬B\}
\]
Simplify Examples

function simplify(N, L)  \text{ also called unit propagation} \\
remove all clauses from N that contain L \\
delete \neg L from all remaining clauses  \text{ possibly get empty clause } \bot \\
return the resulting clause set

\begin{align*}
simplify(\{A \lor \neg B, \ C \lor \neg A, \ D \lor \neg C \lor \neg A, \ \neg D \lor \neg B\}, \ A) &= \{ \ C \ , \ D \lor \neg C \ , \ \neg D \lor \neg B \} \\
\simplify(\{ \ C \ , \ D \lor \neg C \ , \ \neg D \lor \neg B \}, \ C) &= \{ \neg B \}\end{align*}
Simplify Examples

1  **function** simplify($N$, $L$)  
   %%% also called *unit propagation*
2       remove all clauses from $N$ that contain $L$
3       delete $\neg L$ from all remaining clauses  
   %%% possibly get empty clause $\bot$
4       return the resulting clause set

\[
simplify(\{A \lor \neg B, \ C \lor \neg A, \ D \lor \neg C \lor \neg A, \ \neg D \lor \neg B\}, \ A) \\
= \{ \ C \ , \ D \lor \neg C \ , \ \neg D \lor \neg B \} \\
\]

\[
simplify(\{\ C \ , \ D \lor \neg C \ , \ \neg D \lor \neg B\}, \ C) \\
= \{ \ D \ , \ \neg D \lor \neg B \} \\
\]
Simplify Examples

1. **function** simplify$(N, L)$ \(\text{%% also called unit propagation}\)
   - remove all clauses from $N$ that contain $L$
   - delete $\neg L$ from all remaining clauses \(\text{%% possibly get empty clause} \bot\)
   - return the resulting clause set

\[
\text{simplify}([A \lor \neg B, C \lor \neg A, D \lor \neg C \lor \neg A, \neg D \lor \neg B], A) = \{C, D \lor \neg C, \neg D \lor \neg B\}
\]

\[
\text{simplify}([C, D \lor \neg C, \neg D \lor \neg B], C) = \{D, \neg D \lor \neg B\}
\]

\[
\text{simplify}([D, \neg D \lor \neg B], D) = \{\neg B\}
\]
Simplify Examples

1. **function** simplify($N$, $L$) \(\text{%% also called } \text{unit propagation}\)
2. remove all clauses from $N$ that contain $L$
3. delete $\neg L$ from all remaining clauses \(\text{%% possibly get empty clause } \bot\)
4. return the resulting clause set

simplify($\{A \lor \neg B, C \lor \neg A, D \lor \neg C \lor \neg A, \neg D \lor \neg B\}$, $A$)  
\[= \{C, D \lor \neg C, \neg D \lor \neg B\}\]

simplify($\{C, D \lor \neg C, \neg D \lor \neg B\}$, $C$)  
\[= \{D, \neg D \lor \neg B\}\]

simplify($\{D, \neg D \lor \neg B\}$, $D$)  
\[= \{\neg B\}\]
Making DPLL Fast – Overview

Conflict Driven Clause Learning (CDCL) solvers extend DPLL

**Lemma learning:** add new clauses to the clause set as branches get closed (“conflict driven”)

Goal: reuse information that is obtained in one branch for subsequent derivation steps.

**Backtracking:** replace chronological backtracking by “dependency-directed backtracking”, aka “backjumping”: on backtracking, skip splits that are not necessary to close a branch

**Randomized restarts:** every now and then start over, with learned clauses

**Variable selection heuristics:** what literal to split on. E.g., use literals that occur often

**Make unit-propagation fast:** 2-watched literal technique
Making DPLL Fast

2-watched literal technique
A technique to implement unit propagation efficiently

- In each clause, select two (currently undefined) “watched” literals.
- For each variable \( A \), keep a list of all clauses in which \( A \) is watched and a list of all clauses in which \( \neg A \) is watched.
- If an undefined variable is set to false (or to true), check all clauses in which \( A \) (or \( \neg A \)) is watched and watch another literal (that is true or undefined) in this clause if possible.
- As long as there are two watched literals in a \( n \)-literal clause, this clause cannot be used for unit propagation, because \( n - 1 \) of its literals have to be false to provide a unit conclusion.
- Important: Watched literal information need not be restored upon backtracking.
2-Watched Literals Example

In an $n$-literal clause, $n - 1$ literals must be assigned false before it can unit-propagate. Defer unit propagation until this is the case.

*Invariant*: if clause is not satisfied, watched literals are undefined. Only clauses violating the invariant can unit-propagate

Clause $\neg A \lor \neg B \lor \neg C \lor \neg D \lor E$  (watched literals underlined)
2-Watched Literals Example

In an $n$-literal clause, $n - 1$ literals must be assigned false before it can unit-propagate. Defer unit propagation until this is the case.

*Invariant:* if clause is not satisfied, watched literals are undefined. Only clauses violating the invariant can unit-propagate.

Clause $\neg A \lor \neg B \lor \neg C \lor \neg D \lor E$ (watched literals *underlined*)

1. Assignments developed in this order $C$
2-Watched Literals Example

In an $n$-literal clause, $n - 1$ literals must be assigned false before it can unit-propagate. Defer unit propagation until this is the case.

**Invariant:** if clause is not satisfied, watched literals are undefined. Only clauses violating the invariant can unit-propagate

Clause $\neg A \lor \neg B \lor \neg C \lor \neg D \lor E$ (watched literals underlined)

1. Assignments developed in this order $C \rightarrow D$

2. Watched literal $\neg A$ is false $\Rightarrow$ find another literal to watch

3. Extend with decision literal $C \rightarrow D \rightarrow A \rightarrow B$

4. Impossible to watch two literals now $\Rightarrow E$ is unit-propagated

5. Now have $C \rightarrow D \rightarrow A \rightarrow B \rightarrow E$ Maintains invariant

Invariant is maintained in case of backtracking to $\neg B$: Then have $C \rightarrow D \rightarrow A \rightarrow \neg B$
2-Watched Literals Example

In an \( n \)-literal clause, \( n - 1 \) literals must be assigned false before it can unit-propagate. Defer unit propagation until this is the case.

**Invariant**: if clause is not satisfied, watched literals are undefined. Only clauses violating the invariant can unit-propagate

Clause \( \neg A \lor \neg B \lor \neg C \lor \neg D \lor E \)  \( \text{ (watched literals underlined) } \)

1. Assignments developed in this order \( C \rightarrow D \rightarrow A \)

\[ \neg A \lor \neg B \lor \neg C \lor \neg D \lor E \]
2-Watched Literals Example

In an \( n \)-literal clause, \( n - 1 \) literals must be assigned false before it can unit-propagate. Defer unit propagation until this is the case.

**Invariant:** if clause is not satisfied, watched literals are undefined. Only clauses violating the invariant can unit-propagate

Clause \( \neg A \lor \neg B \lor \neg C \lor \neg D \lor E \) (watched literals underlined)

1. Assignments developed in this order \( C \quad D \quad A \)
2. Watched literal \( \neg A \) is false \( \Rightarrow \) find another literal to watch
2-Watched Literals Example

In an $n$-literal clause, $n - 1$ literals must be assigned false before it can unit-propagate. Defer unit propagation until this is the case.

*Invariant:* if clause is not satisfied, watched literals are undefined. Only clauses violating the invariant can unit-propagate

Clause $\neg A \lor \neg B \lor \neg C \lor \neg D \lor E$  \hspace{10mm} (watched literals underlined)

1. Assignments developed in this order $C \rightarrow D \rightarrow A$
2. Watched literal $\neg A$ is false $\Rightarrow$ find another literal to watch

Clause $\neg A \lor \underline{\neg B} \lor \neg C \lor \neg D \lor E$
In an \( n \)-literal clause, \( n - 1 \) literals must be assigned false before it can unit-propagate. Defer unit propagation until this is the case.

**Invariant:** if clause is not satisfied, watched literals are undefined. Only clauses violating the invariant can unit-propagate.

\[
\neg A \lor \neg B \lor \neg C \lor \neg D \lor E \quad \text{(watched literals underlined)}
\]

1. Assignments developed in this order \( C \rightarrow D \rightarrow A \)
2. Watched literal \( \neg A \) is false \( \leadsto \) find another literal to watch

\[
\neg A \lor \neg B \lor \neg C \lor \neg D \lor E
\]

3. Extend with decision literal \( C \rightarrow D \rightarrow A \rightarrow B \)
2-Watched Literals Example

In an $n$-literal clause, $n - 1$ literals must be assigned false before it can unit-propagate. Defer unit propagation until this is the case.

*Invariant*: if clause is not satisfied, watched literals are undefined. Only clauses violating the invariant can unit-propagate

Clause $\neg A \lor \neg B \lor \neg C \lor \neg D \lor E$ (watched literals underlined)

1. Assignments developed in this order $C \rightarrow D \rightarrow A$
2. Watched literal $\neg A$ is false $\Rightarrow$ find another literal to watch

Clause $\neg A \lor \underline{\neg B} \lor \neg C \lor \neg D \lor E$

3. Extend with decision literal $C \rightarrow D \rightarrow A \rightarrow B$
4. Impossible to watch two literals now $\Rightarrow E$ is unit-propagated
2-Watched Literals Example

In an \( n \)-literal clause, \( n - 1 \) literals must be assigned false before it can unit-propagate. Defer unit propagation until this is the case.

*Invariant*: if clause is not satisfied, watched literals are undefined. Only clauses violating the invariant can unit-propagate

Clause \( \neg A \lor \neg B \lor \neg C \lor \neg D \lor E \) (watched literals underlined)

1. Assignments developed in this order \( C \rightarrow D \rightarrow A \)
2. Watched literal \( \neg A \) is false \( \Rightarrow \) find another literal to watch

Clause \( \neg A \lor \neg B \lor \neg C \lor \neg D \lor E \)

3. Extend with decision literal \( C \rightarrow D \rightarrow A \rightarrow B \)
4. Impossible to watch two literals now \( \Rightarrow E \) is unit-propagated
5. Now have \( C \rightarrow D \rightarrow A \rightarrow B \rightarrow E \)
   Maintains invariant
2-Watched Literals Example

In an \( n \)-literal clause, \( n - 1 \) literals must be assigned false before it can unit-propagate. Defer unit propagation until this is the case.

*Invariant*: if clause is not satisfied, watched literals are undefined. Only clauses violating the invariant can unit-propagate

Clause  \( \neg A \lor \neg B \lor \neg C \lor \neg D \lor E \)  (watched literals underlined)

1. Assignments developed in this order  \( C \rightarrow D \rightarrow A \)
2. Watched literal \( \neg A \) is false \( \Rightarrow \) find another literal to watch

Clause  \( \neg A \lor \neg B \lor \neg C \lor \neg D \lor E \)

3. Extend with decision literal  \( C \rightarrow D \rightarrow A \rightarrow B \)
4. Impossible to watch two literals now \( \Rightarrow \) \( E \) is unit-propagated
5. Now have  \( C \rightarrow D \rightarrow A \rightarrow B \rightarrow E \)
   Maintains invariant

*Invariant is maintained in case of backtracking to \( \neg B \):*

Then have  \( C \rightarrow D \rightarrow A \rightarrow \neg B \)
"Avoid making the same mistake twice"

\[
\begin{align*}
\cdots \\
B \lor \neg A & \quad (1) \\
D \lor \neg C & \quad (2) \\
\neg D \lor \neg B \lor \neg C & \quad (3)
\end{align*}
\]
Lemma Learning

"Avoid making the same mistake twice"

\[ B \lor \neg A \quad (1) \]
\[ D \lor \neg C \quad (2) \]
\[ \neg D \lor \neg B \lor \neg C \quad (3) \]

w/o Lemma

```
\begin{align*}
A & \quad \neg A \\
(1) & \quad \\quad B \\
(2) & \quad C \quad \neg C \\
(3) & \quad \\quad D
\end{align*}
```

Diagram:

```
            A
           /  \
          /    \
         /      \
        /        \
       /          \
      /            \
     /              \
    /                \
   /                  \
  /                    \
 /                      \
/                        \

\text{w/o Lemma}
```

Diagram:

```
            A
           /  \
          /    \
         /      \
        /        \
       /          \
      /            \
     /              \
    /                \
   /                    \
  /                        \
 /                            \

\begin{align*}
B & \lor \neg A \quad (1) \\
D & \lor \neg C \quad (2) \\
\neg D & \lor \neg B \lor \neg C \quad (3)
\end{align*}
```
Lemma Learning

"Avoid making the same mistake twice"

\[
\begin{align*}
\ldots \\
B \lor \neg A & \quad (1) \\
D \lor \neg C & \quad (2) \\
\neg D \lor \neg B \lor \neg C & \quad (3)
\end{align*}
\]

w/o Lemma

\[
\begin{tikzpicture}
  \node (A) at (0,0) {$A$};
  \node (B) at (-1,-1) {$B$};
  \node (C) at (1,-1) {$C$};
  \node (D) at (0,-2) {$D$};
  \node (A') at (2,0) {$\neg A$};
  \node (C') at (1,-2) {$\neg C$};
  \node (D') at (0,-3) {$*$};
  \draw (A) -- (B);
  \draw (A) -- (C);
  \draw (B) -- (D);
  \draw (C) -- (D);
  \draw (B) -- (A');
  \draw (C) -- (A');
  \draw (D) -- (D');
\end{tikzpicture}
\]
Lemma Learning

"Avoid making the same mistake twice"

...  
\[ B \lor \neg A \] (1)  
\[ D \lor \neg C \] (2)  
\[ \neg D \lor \neg B \lor \neg C \] (3)

Lemma Candidates by Resolution:

\[ \neg D \lor \neg B \lor \neg C \]

w/o Lemma

\( A \)
\( \neg A \)

\( B \)
\( \land \)
\( C \)
\( \neg C \)

\( D \)
\( \land \)
\( * \)
\( \land \)
\( \land \)
Lemma Learning

"Avoid making the same mistake twice"

\[\begin{align*}
B \lor \neg A & \quad \text{(1)} \\
D \lor \neg C & \quad \text{(2)} \\
\neg D \lor \neg B \lor \neg C & \quad \text{(3)}
\end{align*}\]

Lemma Candidates by Resolution:

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

w/o Lemma

\[\begin{align*}
A & \\
\neg A & \text{(1)} \\
B & \\
\neg C & \text{(2)} \\
C & \\
\neg D & \\
D & \text{(3)} \\
\neg C & \text{(*)}
\end{align*}\]
Lemma Learning

"Avoid making the same mistake twice"

Lemma Candidates by Resolution:

\[ \neg D \lor \neg B \lor \neg C \]
\[ D \lor \neg C \]
\[ \neg D \lor \neg B \lor \neg C \]

w/o Lemma

\[ A \]
\[ \neg A \]
\[ B \]
\[ C \]
\[ \neg C \]
\[ D \]
\[ \neg C \]
\[ B \lor \neg A \]

\[ \neg C \lor \neg A \]
Lemma Learning

"Avoid making the same mistake twice"

Lemma Candidates by Resolution:

\[
\begin{align*}
\neg D \lor \neg B \lor \neg C & \quad (1) \\
D \lor \neg C & \quad (2) \\
\neg D \lor \neg B \lor \neg C & \quad (3)
\end{align*}
\]

With Lemma!

\[
\begin{align*}
A & \quad \neg A \\
\neg D \lor \neg B \lor \neg C & \quad (2) \\
D \lor \neg C & \quad (3)
\end{align*}
\]

w/o Lemma

\[
\begin{align*}
B & \lor \neg A (1) \\
D & \lor \neg C (2) \\
\neg D & \lor \neg B \lor \neg C (3)
\end{align*}
\]
Lemma Learning

"Avoid making the same mistake twice"

\[ B \lor \neg A \]  (1)
\[ D \lor \neg C \]  (2)
\[ \neg D \lor \neg B \lor \neg C \]  (3)

**Lemma Candidates by Resolution:**

\[ \neg D \lor \neg B \lor \neg C \]
\[ D \lor \neg C \]

\[ \neg B \lor \neg C \]
\[ B \lor \neg A \]

\[ \neg C \lor \neg A \]

w/o Lemma

With Lemma

(1) \[ A \]
(2) \[ B \]
(3) \[ C \]

* \[ D \]

* \[ A \]

\[ \neg A \]

\[ \neg A \]

\[ \neg C \]
Lemma Learning

"Avoid making the same mistake twice"

Lemma Candidates by Resolution:

\[\neg D \lor \neg B \lor \neg C \quad (3)\]

\[\neg B \lor \neg C \quad (2)\]

\[\neg D \lor \neg C \quad (1)\]

With Lemma

\[\neg C \lor \neg A\]

w/o Lemma

\[\neg D \lor \neg B \lor \neg C \quad (3)\]

\[\neg B \lor \neg C \quad (2)\]

\[\neg D \lor \neg C \quad (1)\]
Further Information

The ideas described so far have been implemented in the SAT checker zChaff:


Other Overviews


DPLL and the refined CDCL algorithm are the practically best methods for PL.

The resolution calculus (Robinson 1969) has been introduced as a basis for automated theorem proving in first-order logic. We will see it in detail in the first-order logic part of this lecture.

Refined versions are still the practically best methods for first-order logic.

The resolution calculus is best introduced first for propositional logic.
The Propositional Resolution Calculus

Propositional resolution inference rule

\[
\begin{array}{c}
C \lor A \\
\neg A \lor D \\
\hline
C \lor D
\end{array}
\]

Terminology: \( C \lor D \): resolvent; \( A \): resolved atom
The Propositional Resolution Calculus

Propositional resolution inference rule

\[
\begin{array}{c}
C \lor A \\
\sim A \lor D \\
\hline
C \lor D
\end{array}
\]

Terminology: \( C \lor D \): resolvent; \( A \): resolved atom

Propositional (positive) factoring inference rule

\[
\begin{array}{c}
C \lor A \lor A \\
\hline
C \lor A
\end{array}
\]

Terminology: \( C \lor A \): factor
The Propositional Resolution Calculus

**Propositional resolution inference rule**

\[ \frac{C \lor A \quad \neg A \lor D}{C \lor D} \]

Terminology: \( C \lor D \): resolvent; \( A \): resolved atom

**Propositional (positive) factoring inference rule**

\[ \frac{C \lor A \lor A}{C \lor A} \]

Terminology: \( C \lor A \): factor

These are schematic inference rules:
- \( C \) and \( D \) – propositional clauses
- \( A \) – propositional atom
- “\( \lor \)” is considered associative and commutative
Derivations

Let $N = \{ C_1, \ldots, C_k \}$ be a set of input clauses. A derivation (from $N$) is a sequence of the form

$$C_1, \ldots, C_k, C_{k+1}, \ldots, C_n, \ldots$$

such that for every $n \geq k + 1$

- $C_n$ is a resolvent of $C_i$ and $C_j$, for some $1 \leq i, j < n$, or
- $C_n$ is a factor of $C_i$, for some $1 \leq i < n$. 
Derivations

Let $N = \{C_1, \ldots, C_k\}$ be a set of input clauses

A derivation (from $N$) is a sequence of the form

$$C_1, \ldots, C_k, C_{k+1}, \ldots, C_n, \ldots$$

such that for every $n \geq k + 1$

- $C_n$ is a resolvent of $C_i$ and $C_j$, for some $1 \leq i, j < n$, or
- $C_n$ is a factor of $C_i$, for some $1 \leq i < n$.

The empty disjunction, or empty clause, is written as $\Box$

A refutation (of $N$) is a derivation from $N$ that contains $\Box$
Sample Refutation

1. \( \neg A \lor \neg A \lor B \) (given)
2. \( A \lor B \) (given)
3. \( \neg C \lor \neg B \) (given)
4. \( C \) (given)

5. \( \neg A \lor B \lor B \) (Res. 2. into 1.)
6. \( \neg A \lor B \) (Fact. 5.)
7. \( B \lor B \) (Res. 2. into 6.)
8. \( B \) (Fact. 7.)
9. \( \neg C \) (Res. 8. into 3.)
10. \( \) (Res. 4. into 9.)
Sample Refutation

1. \( \neg A \lor \neg A \lor B \)  (given)
2. \( A \lor B \)  (given)
3. \( \neg C \lor \neg B \)  (given)
4. \( C \)  (given)
5. \( \neg A \lor B \lor B \)  (Res. 2. into 1.)
Sample Refutation

1. \( \neg A \lor \neg A \lor B \) \hspace{1cm} (given)
2. \( A \lor B \) \hspace{1cm} (given)
3. \( \neg C \lor \neg B \) \hspace{1cm} (given)
4. \( C \) \hspace{1cm} (given)
5. \( \neg A \lor B \lor B \) \hspace{1cm} (Res. 2. into 1.)
6. \( \neg A \lor B \) \hspace{1cm} (Fact. 5.)
Sample Refutation

1. ¬A ∨ ¬A ∨ B (given)
2. A ∨ B (given)
3. ¬C ∨ ¬B (given)
4. C (given)
5. ¬A ∨ B ∨ B (Res. 2. into 1.)
6. ¬A ∨ B (Fact. 5.)
7. B ∨ B (Res. 2. into 6.)
Sample Refutation

1. $\neg A \lor \neg A \lor B$ (given)
2. $A \lor B$ (given)
3. $\neg C \lor \neg B$ (given)
4. $C$ (given)
5. $\neg A \lor B \lor B$ (Res. 2. into 1.)
6. $\neg A \lor B$ (Fact. 5.)
7. $B \lor B$ (Res. 2. into 6.)
8. $B$ (Fact. 7.)
Sample Refutation

1. \( \neg A \lor \neg A \lor B \)  
   (given)
2. \( A \lor B \)  
   (given)
3. \( \neg C \lor \neg B \)  
   (given)
4. \( C \)  
   (given)
5. \( \neg A \lor B \lor B \)  
   (Res. 2. into 1.)
6. \( \neg A \lor B \)  
   (Fact. 5.)
7. \( B \lor B \)  
   (Res. 2. into 6.)
8. \( B \)  
   (Fact. 7.)
9. \( \neg C \)  
   (Res. 8. into 3.)
Sample Refutation

1. \( \neg A \lor \neg A \lor B \) \hspace{1cm} (given)
2. \( A \lor B \) \hspace{1cm} (given)
3. \( \neg C \lor \neg B \) \hspace{1cm} (given)
4. \( C \) \hspace{1cm} (given)
5. \( \neg A \lor B \lor B \) \hspace{1cm} (Res. 2. into 1.)
6. \( \neg A \lor B \) \hspace{1cm} (Fact. 5.)
7. \( B \lor B \) \hspace{1cm} (Res. 2. into 6.)
8. \( B \) \hspace{1cm} (Fact. 7.)
9. \( \neg C \) \hspace{1cm} (Res. 8. into 3.)
10. \( \square \) \hspace{1cm} (Res. 4. into 9.)
Soundness and Completeness

Important properties a calculus may or may not have:

**Soundness:** if there is a refutation of $N$ then $N$ is unsatisfiable

**Deduction completeness:**
if $N$ is valid then there is a derivation of $N$

**Refutational completeness:**
if $N$ is unsatisfiable then there is a refutation of $N$
Soundness and Completeness

Important properties a calculus may or may not have:

**Soundness:** if there is a refutation of $N$ then $N$ is unsatisfiable

**Deduction completeness:**

if $N$ is valid then there is a derivation of $N$

**Refutational completeness:**

if $N$ is unsatisfiable then there is a refutation of $N$

The resolution calculus is sound and refutationally complete, but not deduction complete
Soundness of Propositional Resolution

Theorem 2

Propositional resolution is sound

Proof.
Let $I$ be an interpretation. To be shown:

1. for resolution:
   
   
   \[ I \models \neg A \lor C \lor D \lor \neg A \Rightarrow I \models C \lor D \]

2. for factoring:
   
   \[ I \models C \lor A \lor A \Rightarrow I \models C \lor A \]

Ad (1): Assume premises are valid in $I$. Two cases need to be considered:

(a) $A$ is valid in $I$, or (b) $\neg A$ is valid in $I$.

a) $I \models \neg A \Rightarrow I \models C \Rightarrow I \models C \lor D$

b) $I \models A \Rightarrow I \models C \Rightarrow I \models C \lor D$
Soundness of Propositional Resolution

Theorem 2

*Propositional resolution is sound*

Proof.

Let $I$ be an interpretation. To be shown:

1. for resolution: $I \models C \lor A, I \models D \lor \neg A \Rightarrow I \models C \lor D$

Ad (1): Assume premises are valid in $I$. Two cases need to be considered:

(a) $A$ is valid in $I$, or (b) $\neg A$ is valid in $I$.

a) $I \models A \Rightarrow I \models D \Rightarrow I \models C \lor D$

b) $I \models \neg A \Rightarrow I \models C \Rightarrow I \models C \lor D$

Ad (2): even simpler
Soundness of Propositional Resolution

Theorem 2

*Propositional resolution is sound*

**Proof.**

Let \( I \) be an interpretation. To be shown:

1. for resolution: \( I \models C \lor A \), \( I \models D \lor \neg A \) \( \Rightarrow \) \( I \models C \lor D \)
2. for factoring: \( I \models C \lor A \lor A \) \( \Rightarrow \) \( I \models C \lor A \)

Ad (1): Assume premises are valid in \( I \). Two cases need to be considered:

(a) \( A \) is valid in \( I \), or (b) \( \neg A \) is valid in \( I \).

\[ I \models A \Rightarrow I \models D \Rightarrow I \models C \lor D \]
\[ I \models \neg A \Rightarrow I \models C \Rightarrow I \models C \lor D \]

Ad (2): even simpler

\[ I \models C \lor A \lor A \Rightarrow I \models C \lor A \]
Soundness of Propositional Resolution

**Theorem 2**

*Propositional resolution is sound*

**Proof.**

Let \( I \) be an interpretation. To be shown:

1. for resolution: \( I \models C \lor A, \ I \models D \lor \neg A \Rightarrow I \models C \lor D \)
2. for factoring: \( I \models C \lor A \lor A \Rightarrow I \models C \lor A \)

Ad (1): Assume premises are valid in \( I \). Two cases need to be considered:
(a) \( A \) is valid in \( I \), or (b) \( \neg A \) is valid in \( I \).
Soundness of Propositional Resolution

**Theorem 2**

*Propositional resolution is sound*

**Proof.**

Let $I$ be an interpretation. To be shown:

1. for resolution: $I \models C \lor A$, $I \models D \lor \neg A \Rightarrow I \models C \lor D$

2. for factoring: $I \models C \lor A \lor A \Rightarrow I \models C \lor A$

Ad (1): Assume premises are valid in $I$. Two cases need to be considered:

(a) $A$ is valid in $I$, or (b) $\neg A$ is valid in $I$.

a) $I \models A \Rightarrow I \models D \Rightarrow I \models C \lor D$
Soundness of Propositional Resolution

**Theorem 2**

*Propositional resolution is sound*

**Proof.**

Let $I$ be an interpretation. To be shown:

1. for resolution: $I \models C \lor A$, $I \models D \lor \neg A \Rightarrow I \models C \lor D$
2. for factoring: $I \models C \lor A \lor A \Rightarrow I \models C \lor A$

Ad (1): Assume premises are valid in $I$. Two cases need to be considered:

(a) $A$ is valid in $I$, or (b) $\neg A$ is valid in $I$.

a) $I \models A \Rightarrow I \models D \Rightarrow I \models C \lor D$

b) $I \models \neg A \Rightarrow I \models C \Rightarrow I \models C \lor D$
Soundness of Propositional Resolution

Theorem 2

*Propositional resolution is sound*

Proof.

Let \( I \) be an interpretation. To be shown:

1. for resolution: \( I \models C \lor A, I \models D \lor \neg A \Rightarrow I \models C \lor D \)
2. for factoring: \( I \models C \lor A \lor A \Rightarrow I \models C \lor A \)

Ad (1): Assume premises are valid in \( I \). Two cases need to be considered:

(a) \( A \) is valid in \( I \), or (b) \( \neg A \) is valid in \( I \).

\[ \text{a)} \quad I \models A \Rightarrow I \models D \Rightarrow I \models C \lor D \]

\[ \text{b)} \quad I \models \neg A \Rightarrow I \models C \Rightarrow I \models C \lor D \]

Ad (2): even simpler
Completeness of Propositional Resolution

Theorem 3

Propositional Resolution is refutationally complete

- That is, if a propositional clause set is unsatisfiable, then Resolution will derive the empty clause $\square$ eventually
- More precisely: If a clause set is unsatisfiable and closed under the application of the Resolution and Factoring inference rules, then it contains the empty clause $\square$
- Perhaps easiest proof: semantic tree proof technique (see whiteboard)
- This result can be considerably strengthened, some strengthenings come for free from the proof
Semantic Trees
(Robinson 1968, Kowalski and Hayes 1969)

Semantic trees are a convenient device to represent interpretations for possibly infinitely many atoms

Applications

- To prove the completeness of the propositional resolution calculus
- Characterizes a specific, refined resolution calculus
- To prove the compactness theorem of propositional logic. Application: completeness proof of first-order logic Resolution.
Trees

A tree

- is an acyclic, connected, directed graph, where
- every node has at most one incoming edge

A rooted tree has a dedicated node, called root that has no incoming edge

A tree is finite iff it has finitely many vertices (and edges) only

In a finitely branching tree every node has only finitely many edges

A binary tree every node has at most two outgoing edges. It is complete iff every node has either no or two outgoing edges
A path $\mathcal{P}$ in a rooted tree is a possibly infinite sequence of nodes $\mathcal{P} = (N_0, N_1, \ldots)$, where $N_0$ is the root, and $N_i$ is a direct successor of $N_{i-1}$, for all $i = 1, \ldots, n$.

A path to a node $N$ is a finite path of the form $(N_0, N_1, \ldots, N_n)$ such that $N = N_n$; the value $n$ is the length of the path.

The node $N_{n-1}$ is called the immediate predecessor of $N$.

Every node $N_0, N_1, \ldots, N_{n-1}$ is called a predecessor of $N$.

A (node-)labelled tree is a tree together with a labelling function $\lambda$ that maps each of its nodes to an element in a given set.

Let $L$ be a literal. The complement of $L$ is the literal

$$
\overline{L} := \begin{cases} 
\neg A & \text{if } L \text{ is the atom } A \\
A & \text{if } L \text{ is the negated atom } \neg A.
\end{cases}
$$
Semantic Trees

A semantic tree $B$ (for a set of atoms $D$) is a labelled, complete, rooted, binary tree such that

1. the root is labelled by the symbol $\top$

2. for every inner node $N$, one successor of $N$ is labeled with the literal $A$, and the other successor is labeled with the literal $\neg A$, for some $A \in D$

3. for every node $N$, there is no literal $L$ such that $L \in I(N)$ and $\overline{L} \in I(N)$, where

$$I(N) = \{ \lambda(N_i) \mid N_0, N_1, \ldots, (N_n = N) \text{ is a path to } N \text{ and } 1 \leq i \leq n \}$$
**Semantic Trees**

**Atom Set**
For a clause set $N$ let the atom set (of $N$) be the set of atoms occurring in clauses in $N$.

A semantic tree for $N$ is a semantic tree for the atom set of $N$.

**Path Semantics**
For a path $P = (N_0, N_1, \ldots)$ let

$$\mathcal{I}(P) = \{ \lambda(N_i) \mid i \geq 0 \}$$

be the set of all literals along $P$.

**Complete Semantic Tree**
A semantic tree for $D$ is complete iff for every $A \in D$ and every branch $P$ it holds that

$$A \in \mathcal{I}(P) \text{ or } \neg A \in \mathcal{I}(P)$$
Interpretation Induced by a Semantic Tree

Every path $\mathcal{P}$ in a complete semantic tree for $D$ induces an interpretation $\mathcal{I}_\mathcal{P}$ as follows:

$$\mathcal{I}_\mathcal{P}[A] = \begin{cases} 
\text{true} & \text{if } A \in \mathcal{I}_\mathcal{P} \\
\text{false} & \text{if } \neg A \in \mathcal{I}_\mathcal{P}
\end{cases}$$

A complete semantic tree can be seen as an enumeration of all possible interpretations for $N$ (it holds $\mathcal{I}_\mathcal{P} \neq \mathcal{I}_{\mathcal{P}'}$ whenever $\mathcal{P} \neq \mathcal{P}'$)
Failure Node

If a clause set $N$ is unsatisfiable (not satisfiable) then, by definition, every interpretation $\mathcal{I}$ falsifies some clause in $N$, i.e., $\mathcal{I} \not\models C$ for some $C \in N$.

This motivates the following definition:

**Failure Node**

A node $N'$ in a semantic tree for $N$ is a failure node, if

1. there is a clause $C \in N$ such that $\mathcal{I}_N \not\models C$, and
2. for every predecessor $N''$ of $N'$ it holds:
   - there is no clause $C \in N$ such that $\mathcal{I}_{N''} \not\models C$
Open, Closed

A path $\mathcal{P}$ in a semantic tree for $N$ is closed iff $\mathcal{P}$ contains a failure node, otherwise it is open.

A semantic tree $\mathcal{B}$ for $M$ is closed iff every path is closed, otherwise $\mathcal{B}$ is open.

Every closed semantic tree can be turned into a finite closed one by removing all subtrees below all failure nodes.

Remark
The construction of a (closed or open) finite semantic tree is the core of the propositional DPLL procedure above. Our main application now, however, is to prove compactness of propositional clause logic.
Compactness

Theorem 4
A (possibly infinite) clause set \( N \) is unsatisfiable iff there is a closed semantic tree for \( N \)

Proof.
See whiteboard

Corollary 5 (Compactness)
A (possibly infinite) clause set \( N \) is unsatisfiable iff some finite subset of \( N \) is unsatisfiable

Proof.
The if-direction is trivial. For the only-if direction, Theorem 4 gives us a finite unsatisfiable subset of \( N \) as identified by the finitely many failure nodes in the semantic tree.