1. Propositional Logic

Huixing Fang

School of Information Engineering
Yangzhou University
Outline

1. Syntax
2. Semantics (meaning)
3. Satisfiability and Validity
4. Equivalence and Implication
5. Substitution
6. Normal Forms
7. Decision Procedures for Satisfiability
Syntax of PL: a set of symbols and rules for combining them to form “sentences” (formulae)

Symbols

1. Truth symbols: $\top$ (“true”), $\bot$ (“false”)
2. Propositional variables: $P$, $Q$, $P_i$, $Q_i$, ...
3. Logical connectives: $\neg$, $\land$, $\lor$, $\rightarrow$, $\leftrightarrow$

Example 1

1. $\neg P$: negation, “not”;
2. $P \land Q$: conjunction, “and”;
3. $P \lor Q$: disjunction, “or”;
4. $P \rightarrow Q$: implication, “implies”;
5. $P \leftrightarrow Q$: iff, “if and only if”.

Huixing Fang (SIE, Yangzhou University)
1 Syntax

### Arity of Logical connectives

1. **unary**: negation (¬) is unary (takes one argument)
2. **binary**: others (∧, ∨, →, ↔) are binary (take two arguments)

### Antecedent/Consequent

The left and right arguments of → are called the antecedent and consequent, respectively.

\[ P \rightarrow Q, \]

in which, \( P \) is antecedent, and \( Q \) is consequent.

### Terminology

- **Atom**: truth symbol \( \top, \bot \) or propositional variable \( P, Q, ... \)
- **Literal**: an atom \( A \) or its negation \( \neg A \).
- **Formula**: a literal or the application of a logical connective to formulae.
1 Syntax

Formula $S$ is a **subformula** of formula $F$ if it occurs syntactically within $F$.

**Example 2 (Subformula)**

1. subformula of $P$ is $P$;
2. subformulae of $\neg F$: $\neg F$ and the subformulae of $F$;
3. subformulae of $F_1 \land F_2$: $F_1 \land F_2$ and the subformulae of $F_1$ and $F_2$.

Notice that every formula is a subformula of itself. The strict subformulae of a formula are all its subformulae except itself.

**Example 3**

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

in which, $P$ and $Q$ are propositional variables. Each instance of $P$ and $Q$ is an atom and a literal. $\neg Q$ is a literal, but not an atom. $F$ has six subformulae: $F$, $P \land Q$, $P \lor \neg Q$, $\neg Q$, $P$, $Q$. 

Huixing Fang (SIE, Yangzhou University)
Relative Precedence

The relative precedence of the logical connectives from highest to lowest:

\[
\neg > \land > \lor > \to > \leftrightarrow
\]

and, \(\to\), \(\leftrightarrow\) associate to the right.

Example 4

\((P \land Q) \to (P \lor \neg Q)\) can be abbreviated to \(P \land Q \to P \lor \neg Q\)

Example 5

\[(P_1 \land ((\neg P_2) \land \top)) \lor ((\neg P_1) \land P_2)\]

can be abbreviated to

\[P_1 \land \neg P_2 \land \top \lor \neg P_1 \land P_2\]
Outline

1. Syntax

2. Semantics (meaning)

3. Satisfiability and Validity

4. Equivalence and Implication

5. Substitution

6. Normal Forms

7. Decision Procedures for Satisfiability
The **semantics** of a logic provides its meaning, and is given by the truth values **true** and **false**.

**Definition 6 (Interpretation)**

An **interpretation** $\mathcal{I}$ assigns to every **propositional variable** exactly one **truth value**.

**Example 7**

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

is an interpretation assigning **true** to $P$ and **false** to $Q$, where ... elides the (countably infinitely many) assignments that are not relevant to us.

Given a PL formula $F$ and an interpretation $\mathcal{I}$, the truth value of $F$ can be computed.
1. Propositional Logic

2. **Semantics (meaning)**
   - Truth Table
   - Inductive Definition

3. Satisfiability and Validity

4. Equivalence and Implication

5. Substitution

6. Normal Forms

7. Decision Procedures for Satisfiability
The simplest manner of computing the truth value of a PL formula $F$ is via a truth table. How to evaluate each logical connective in terms of its arguments?

1. **Negation ($\neg F$).**

<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>

2. **Binary connectives.**

<table>
<thead>
<tr>
<th>$F_1$</th>
<th>$F_2$</th>
<th>$F_1 \land F_2$</th>
<th>$F_1 \lor F_2$</th>
<th>$F_1 \rightarrow F_2$</th>
<th>$F_1 \leftrightarrow F_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</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>
<td>1</td>
</tr>
</tbody>
</table>
Example 8

Consider the formula

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

and the interpretation

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

To evaluate the truth value of \( F \) under \( \mathcal{I} \), construct the following table:

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

The top row is given by the subformulae of \( F \). \( \mathcal{I} \) provides values for the first two columns.
1. Propositional Logic

2. Semantics (meaning)
   - Truth Table
   - Inductive Definition

3. Satisfiability and Validity

4. Equivalence and Implication

5. Substitution

6. Normal Forms

7. Decision Procedures for Satisfiability
2.2 Inductive Definition

Inductive Definition:

1. defines the meaning of basic elements first, i.e. atoms;
2. then defines a more complex element in terms of these elements.

Two symbols:

1. We write $\mathcal{I} \models F$ if $F$ evaluates to true under interpretation $\mathcal{I}$
2. and write $\mathcal{I} \not\models F$ if $F$ evaluates to false.

The meaning of truth symbols

1. $\mathcal{I} \models \top$
2. $\mathcal{I} \not\models \bot$

Under any interpretation $\mathcal{I}$, $\top$ has value true, and $\bot$ has value false.
2.2 Inductive Definition

The meaning of propositional variables

1. \( \mathcal{I} \models P \) iff \( \mathcal{I}[P] = \text{true} \), \( P \) has value \textbf{true} iff the interpretation \( \mathcal{I} \) assigns \( P \) to have value \textbf{true};

2. \( \mathcal{I} \not\models P \) iff \( \mathcal{I}[P] = \text{false} \),

Assume that formulae \( F, F_1, \) and \( F_2 \) have truth values. From these formulae, evaluate the semantics of more complex formulae:

Semantics of more complex formulae

\[
\begin{align*}
\mathcal{I} & \models \neg F \quad \text{iff} \quad \mathcal{I} \not\models F \\
\mathcal{I} & \models F_1 \land F_2 \quad \text{iff} \quad \mathcal{I} \models F_1 \text{ and } \mathcal{I} \models F_2 \\
\mathcal{I} & \models F_1 \lor F_2 \quad \text{iff} \quad \mathcal{I} \models F_1 \text{ or } \mathcal{I} \models F_2 \\
\mathcal{I} & \models F_1 \rightarrow F_2 \quad \text{iff, if } \mathcal{I} \models F_1 \text{ then } \mathcal{I} \models F_2 \\
\mathcal{I} & \models F_1 \leftrightarrow F_2 \quad \text{iff } \mathcal{I} \models F_1 \text{ and } \mathcal{I} \models F_2, \text{ or } \mathcal{I} \not\models F_1 \text{ and } \mathcal{I} \not\models F_2
\end{align*}
\]
Example 9

Consider the formula $F : P \land Q \rightarrow P \lor \neg Q$ and the interpretation

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

Compute the truth value of $F$ as follows:

1. $\mathcal{I} \models P$ since $\mathcal{I}[P] = \text{true}$
2. $\mathcal{I} \not\models Q$ since $\mathcal{I}[Q] = \text{false}$
3. $\mathcal{I} \models \neg Q$ by 2 and semantics of $\neg$
4. $\mathcal{I} \not\models P \land Q$ by 2 and semantics of $\land$
5. $\mathcal{I} \models P \lor \neg Q$ by 1 and semantics of $\lor$
6. $\mathcal{I} \models F$ by 4 and semantics of $\rightarrow$

We considered the distinct subformulae of $F$ according to the subformula ordering: $F_1$ precedes $F_2$ if $F_1$ is a subformula of $F_2$. 
Outline

1. Syntax
2. Semantics (meaning)
3. Satisfiability and Validity
4. Equivalence and Implication
5. Substitution
6. Normal Forms
7. Decision Procedures for Satisfiability
Determining satisfiability and validity of formulae are important tasks in logic.

**Definition 10 (Satisfiable)**

A formula $F$ is satisfiable iff there exists an interpretation $\mathcal{I}$ such that $\mathcal{I} \models F$.

**Definition 11 (Valid)**

A formula $F$ is valid iff for all interpretations $\mathcal{I} \models F$.

Satisfiability and validity are dual concepts, and switching from one to the other is easy. $F$ is valid iff $\neg F$ is unsatisfiable.
1. Syntax

2. Semantics (meaning)

3. Satisfiability and Validity
   - Truth-table method
   - Semantic Argument Method

4. Equivalence and Implication

5. Substitution

6. Normal Forms

7. Decision Procedures for Satisfiability
### Example 12

Consider the formula $F : P \land Q \rightarrow P \lor \neg Q$. Is it valid?

<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>

### Example 13

Consider the formula $F : P \lor Q \rightarrow P \land Q$. Is it valid?

<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>
1. Syntax

2. Semantics (meaning)

3. Satisfiability and Validity
   - Truth-table method
   - Semantic Argument Method

4. Equivalence and Implication

5. Substitution

6. Normal Forms

7. Decision Procedures for Satisfiability
3.2 Semantic Argument Method

Semantic Argument

1. begins by assuming that the given formula $F$ is invalid: there is a falsifying interpretation $I$ such that $I \nvDash F$;

2. The proof proceeds by applying the semantic definitions of the logical connectives in the form of proof rules.

Proof rule

A proof rule has one or more premises (assumed facts) and one or more deductions (deduced facts).

An application of a proof rule requires matching the premises to facts already existing in the semantic argument and then forming the deductions.
3.2 Semantic Argument Method

Proof rules:

- According to the semantics of negation:
  \[
  \begin{align*}
  \mathcal{I} \models \neg F & \quad \frac{\mathcal{I} \not\models \neg F}{\mathcal{I} \models F} \\
  \mathcal{I} \not\models \neg F & \quad \frac{\mathcal{I} \models F}{\mathcal{I} \not\models \neg F}
  \end{align*}
  \]

- According to the semantics of conjunction:
  \[
  \begin{align*}
  \mathcal{I} \models F \land G & \quad \frac{\mathcal{I} \not\models F \land G}{\mathcal{I} \models F} \\
  \mathcal{I} \not\models F \land G & \quad \frac{\mathcal{I} \models F}{\mathcal{I} \not\models F \land G} \\
  \mathcal{I} \models F & \quad \frac{\mathcal{I} \not\models F \land G}{\mathcal{I} \models G} \\
  \mathcal{I} \not\models F & \quad \frac{\mathcal{I} \not\models F \land G}{\mathcal{I} \not\models G}
  \end{align*}
  \]

The latter deduction results in a fork in the proof; each case must be considered separately.
3.2 Semantic Argument Method

Proof rules:

- According to the semantics of disjunction:

\[ \mathcal{I} \models F \lor G \quad \Rightarrow \quad \mathcal{I} \models F \mid \mathcal{I} \models G \]

\[ \mathcal{I} \not\models F \lor G \quad \Rightarrow \quad \mathcal{I} \not\models F \mid \mathcal{I} \not\models G \]

- According to the semantics of implication:

\[ \mathcal{I} \models F \rightarrow G \quad \Rightarrow \quad \mathcal{I} \not\models F \mid \mathcal{I} \models G \]

\[ \mathcal{I} \not\models F \rightarrow G \quad \Rightarrow \quad \mathcal{I} \models F \mid \mathcal{I} \not\models G \]
3.2 Semantic Argument Method

Proof rules:

- According to the semantics of iff:

\[
\begin{align*}
\mathcal{I} &\models F \leftrightarrow G \\
\mathcal{I} &\not\models F \lor G
\end{align*}
\]

\[
\begin{align*}
\mathcal{I} &\not\models F \leftrightarrow G \\
\mathcal{I} &\models F \land \neg G \\
\mathcal{I} &\models \neg F \land G
\end{align*}
\]

- Contradiction occurs when an interpretation \( \mathcal{I} \) both satisfies \( F \) and does not satisfy \( F \):

\[
\begin{align*}
\mathcal{I} &\models F \\
\mathcal{I} &\not\models F \\
\mathcal{I} &\models \bot
\end{align*}
\]
Example 14

To prove that the formula

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

is valid, assume that it is invalid and derive a contradiction.

1. \( \mathcal{I} \not\models P \land Q \rightarrow P \lor \neg Q \) assumption

2. \( \mathcal{I} \models P \land Q \) by 1 and semantics of \( \rightarrow \)

3. \( \mathcal{I} \not\models P \lor \neg Q \) by 1 and semantics of \( \rightarrow \)

4. \( \mathcal{I} \models P \) by 2 and semantics of \( \land \)

5. \( \mathcal{I} \not\models P \) by 3 and semantics of \( \land \)

6. \( \mathcal{I} \models \bot \) 4 and 5 contradictory

The contradiction indicates that our assumption must be wrong.
Example 15

To prove that the formula

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

is valid, assume otherwise and derive a contradiction:

1. \( \mathcal{I} \not\models F \) \hspace{1cm} \text{assumption}
2. \( \mathcal{I} \models (P \rightarrow Q) \land (Q \rightarrow R) \) \hspace{1cm} \text{by 1 and semantics of } \rightarrow
3. \( \mathcal{I} \not\models (P \rightarrow R) \) \hspace{1cm} \text{by 1 and semantics of } \rightarrow
4. \( \mathcal{I} \models P \) \hspace{1cm} \text{by 3 and semantics of } \rightarrow
5. \( \mathcal{I} \not\models R \) \hspace{1cm} \text{by 3 and semantics of } \rightarrow
6. \( \mathcal{I} \models P \rightarrow Q \) \hspace{1cm} \text{by 2 and semantics of } \land
7. \( \mathcal{I} \models Q \rightarrow R \) \hspace{1cm} \text{by 2 and semantics of } \land
3.2 Semantic Argument Method

There are two cases to consider from 6. In the first case,

\[ 8a. \mathcal{I} \not\models P \quad \text{by 6 and semantics of } \rightarrow \]
\[ 9a. \mathcal{I} \models \bot \quad \text{4 and } 8a \text{ are contradictory} \]

In the second case,

\[ 8b. \mathcal{I} \models Q \quad \text{by 6 and semantics of } \rightarrow \]

Now there are two more cases from 7. In the first case,

\[ 9ba. \mathcal{I} \not\models Q \quad \text{by 7 and semantics of } \rightarrow \]
\[ 10ba. \mathcal{I} \models \bot \quad \text{8b and } 9ba \text{ are contradictory} \]

In the second case,

\[ 9bb. \mathcal{I} \models R \quad \text{by 7 and semantics of } \rightarrow \]
\[ 10bb. \mathcal{I} \models \bot \quad \text{5 and } 9bb \text{ are contradictory} \]
Example 16

The derived rule of modus ponens simplifies the proof of Example 15.

\[
\begin{array}{c}
1. \not \models F \\
2. \models (P \rightarrow Q) \land (Q \rightarrow R) \\
3. \not \models (P \rightarrow R) \\
4. \models P \\
5. \not \models R \\
6. \models P \rightarrow Q \\
7. \models Q \rightarrow R \\
8. \models Q \\
9. \models R \\
10. \not \models \bot
\end{array}
\]

assumption
by 1 and semantics of \( \rightarrow \)
by 1 and semantics of \( \rightarrow \)
by 1 and semantics of \( \rightarrow \)
by 3 and semantics of \( \rightarrow \)
by 3 and semantics of \( \rightarrow \)
by 3 and semantics of \( \rightarrow \)
by 2 and semantics of \( \land \)
by 2 and semantics of \( \land \)
by 4, 6, and *modus ponens*
by 8, 7, and *modus ponens*
5 and 9 are contradictory
Outline

1. Syntax
2. Semantics (meaning)
3. Satisfiability and Validity
4. Equivalence and Implication
5. Substitution
6. Normal Forms
7. Decision Procedures for Satisfiability
Two formulae $F_1$ and $F_2$ are **equivalent** if they evaluate to the same truth value under all interpretations $\mathcal{I}$.

We write $F_1 \iff F_2$ when $F_1$ and $F_2$ are equivalent.

$F_1 \iff F_2$ is **not** a formula; it simply abbreviates the statement "$F_1$ and $F_2$ are equivalent."

---

**Example 17**

To prove that

$$P \iff \neg\neg P,$$

we prove that

$$P \iff \neg\neg P$$

is valid via a truth table:

<table>
<thead>
<tr>
<th>$P$</th>
<th>$\neg P$</th>
<th>$\neg\neg P$</th>
<th>$P \iff \neg\neg P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
4 Equivalence and Implication

- Formula $F_1$ implies formula $F_2$ if $\mathcal{I} \models F_2$ for every interpretation $\mathcal{I}$ such that $\mathcal{I} \models F_1$.

- Another way to state that $F_1$ implies $F_2$ is to assert the validity of the formula $F_1 \rightarrow F_2$.

- We write $F_1 \Rightarrow F_2$ when $F_1$ implies $F_2$. The implication $F_1 \Rightarrow F_2$ is not a formula.

**Example 18**

To prove that

$$R \land (\neg R \lor P) \Rightarrow P,$$

we prove that

$$F : R \land (\neg R \lor P) \rightarrow P$$

is valid via a semantic argument.
4 Equivalence and Implication

\[ F : R \land (\neg R \lor P) \rightarrow P \]

Suppose \( F \) is not valid; then there exists an interpretation \( I \) such that \( I \notmodels F \).

1. \( I \notmodels F \)  
   assumption
2. \( I \models R \land (\neg R \lor P) \)  
   by 1 and semantics of \( \rightarrow \)
3. \( I \notmodels P \)  
   by 1 and semantics of \( \rightarrow \)
4. \( I \models R \)  
   by 2 and semantics of \( \land \)
5. \( I \models \neg R \lor P \)  
   by 2 and semantics of \( \land \)

There are two cases to consider. In the first case,

6a. \( I \models \neg R \)  
   by 5 and semantics of \( \lor \)
7a. \( I \models \bot \)  
   4 and 6a are contradictory

What is the second case?
Outline

1. Syntax
2. Semantics (meaning)
3. Satisfiability and Validity
4. Equivalence and Implication
5. Substitution
6. Normal Forms
7. Decision Procedures for Satisfiability
5 Substitution

- Substitution is a **syntactic operation** on formulae with significant semantic consequences.
- It allows us to **prove** the validity of entire sets of formulae via **formula templates**.

**Definition 19 (5 Substitution)**

A **substitution** \( \sigma \) is a mapping from formulae to formulae:

\[
\sigma : \{ F_1 \mapsto G_1, \ldots, F_n \mapsto G_n \}.
\]

The **domain** of \( \sigma \), \( \text{domain}(\sigma) \), is

\[
\text{domain}(\sigma) : \{ F_1, \ldots, F_n \},
\]

while the **range** \( \text{range}(\sigma) \) is

\[
\text{range}(\sigma) : \{ G_1, \ldots, G_n \}.
\]
5 Substitution

- The **application** of a substitution $\sigma$ to a formula $F$, $F\sigma$, **replaces** each occurrence of a formula $F_i$ in $\text{domain}(\sigma)$ with its corresponding formula $G_i$ in $\text{range}(\sigma)$.
- When both subformulae $F_j$ and $F_k$ are in $\text{domain}(\sigma)$, and $F_k$ is a strict subformula of $F_j$, then $F_j$ is replaced by the corresponding formula $G_j$.

**Example 20**

Consider the formula

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

and the substitution

$$\sigma : \{ P \leftrightarrow R, P \land Q \leftrightarrow P \rightarrow Q \}.$$  

Then

$$F\sigma : (P \rightarrow Q) \rightarrow R \lor \neg Q.$$
5 Substitution

- A **variable substitution** is a substitution in which the **domain** consists **only of propositional variables**.
- Useful notation: \( F[F_1, \ldots, F_n] \), we mean that formula \( F \) can have formulae \( F_i, \ i = 1, \ldots, n \), as subformulae.
- Two semantic consequences can be derived from substitution.

**Proposition 21 (Substitution of Equivalent Formulae)**

Consider substitution \( \sigma : \{ F_1 \leftrightarrow G_1, \ldots, F_n \leftrightarrow G_n \} \) such that for each \( i \), \( F_i \leftrightarrow G_i \). Then \( F \leftrightarrow G \).

**Proposition 22 (Valid Template)**

If \( F \) is valid and \( G = F\sigma \) for some variable substitution \( \sigma \), then \( G \) is valid.
Consider applying substitution

\[ \sigma : \{ P \to Q \mapsto \neg P \lor Q \} \]

to

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

Since \( P \to Q \iff \neg P \lor Q \), the formula

\[ F_\sigma : (\neg P \lor Q) \to R \]

is equivalent to \( F \).
5 Substitution

Composition of substitutions

Given substitutions $\sigma_1$ and $\sigma_2$, compute substitution $\sigma$ such that $F\sigma_1\sigma_2 = F\sigma$ for any $F$:

1. apply $\sigma_2$ to each formula of the range of $\sigma_1$, and add the results to $\sigma$;
2. if $F_i$ of $F_i \mapsto G_i$ appears in the domain of $\sigma_2$ but not in the domain of $\sigma_1$, then add $F_i \mapsto G_i$ to $\sigma$.

Example 24

Compute the composition of substitutions

$$\sigma_1\sigma_2 : \{P \mapsto R, P \land Q \mapsto P \rightarrow Q\}\{P \mapsto S, S \mapsto Q\}$$

as follows:

$$\sigma = \{P \mapsto R\sigma_2, P \land Q \mapsto (P \rightarrow Q)\sigma_2, S \mapsto Q\}$$

$$= \{P \mapsto R, P \land Q \mapsto S \rightarrow Q, S \mapsto Q\}$$
1. Syntax

2. Semantics (meaning)

3. Satisfiability and Validity

4. Equivalence and Implication

5. Substitution

6. Normal Forms

7. Decision Procedures for Satisfiability
A normal form of formulae is a syntactic restriction such that for every formula of the logic, there is an equivalent formula in the normal form.

Three normal forms are particularly important for PL:

1. **NNF**: Negation normal form;
2. **DNF**: Disjunctive normal form;
3. **CNF**: Conjunctive normal form.
1. Syntax
2. Semantics (meaning)
3. Satisfiability and Validity
4. Equivalence and Implication
5. Substitution
6. Normal Forms
   - NNF
   - DNF
   - CNF
7. Decision Procedures for Satisfiability
Negation Normal Form (NNF)

NNF requires that $\neg$, $\land$, and $\lor$ be the only connectives and that negations appear only in literals.

Transforming a formula $F$ to equivalent formula $F'$ in NNF can be computed recursively using the following list of template equivalences:

1. $\neg\neg F_1 \iff F_1$
2. $\neg \top \iff \bot$
3. $\neg \bot \iff \top$
4. $\neg (F_1 \land F_2) \iff \neg F_1 \lor \neg F_2$
5. $\neg (F_1 \lor F_2) \iff \neg F_1 \land \neg F_2$
6. $F_1 \to F_2 \iff \neg F_1 \lor F_2$
7. $F_1 \iff F_2 \iff (F_1 \to F_2) \land (F_2 \to F_1)$

The equivalences (4) and (5) are known as De Morgan’s Law.
Example 25

To convert the formula $F : \neg(P \rightarrow \neg(P \land Q))$ into NNF, apply the template equivalence

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

to produce $F' : \neg(\neg P \lor \neg(P \land Q))$, apply De Morgan’s law

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

to produce $F'' : \neg \neg P \land \neg \neg(P \land Q)$, apply

$$\neg \neg F_1 \iff F_1$$

twice to produce

$$F''' : P \land P \land Q,$$

which is in NNF and equivalent to $F$. 
1. Propositional Logic

1. Syntax
2. Semantics (meaning)
3. Satisfiability and Validity
4. Equivalence and Implication
5. Substitution
6. Normal Forms
   - NNF
   - DNF
   - CNF
7. Decision Procedures for Satisfiability
Disjunctive Normal Form (DNF)

A formula is in disjunctive normal form if it is a disjunction of conjunctions of literals:

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

To convert a formula \( F \) into an equivalent formula in DNF, transform \( F \) into NNF and then use the following table of template equivalences:

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

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

When implementing the transformation, the equivalences should be applied left-to-right. The equivalences simply say that conjunction distributes over disjunction.
Example 26

To convert

\[ F : (Q_1 \lor \neg\neg Q_2) \land (\neg R_1 \rightarrow R_2) \]

into DNF, first transform it into NNF

\[ F' : (Q_1 \lor Q_2) \land (R_1 \lor R_2) \]

and then apply distributivity to obtain

\[ F'' : (Q_1 \land (R_1 \lor R_2)) \lor (Q_2 \land (R_1 \lor R_2)) \]

and then distributivity twice again to produce

\[ F''' : (Q_1 \land R_1) \lor (Q_1 \land R_2) \lor (Q_2 \land R_1) \lor (Q_2 \land R_2) \]

\[ F''' \] is in DNF and is equivalent to \[ F \].
1. Syntax

2. Semantics (meaning)

3. Satisfiability and Validity

4. Equivalence and Implication

5. Substitution

6. Normal Forms
   - NNF
   - DNF
   - CNF

7. Decision Procedures for Satisfiability

Huixing Fang  (SIE, Yangzhou University)  1. Propositional Logic
Conjunctive Normal Form (CNF)

A formula is in conjunctive normal form if it is a conjunction of disjunctions of literals:

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

Each inner block of disjunctions is called a clause.

To convert a formula \( F \) into an equivalent formula in CNF, \textbf{transform \( F \) into NNF} and then use the following table of \textbf{template equivalences}:

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

\[
F_1 \lor (F_2 \land F_3) \iff (F_1 \lor F_2) \land (F_1 \lor F_3)
\]
Example 27

To convert

\[ F : (Q_1 \land \neg\neg Q_2) \lor (\neg R_1 \rightarrow R_2) \]

into CNF, first transform \( F \) into NNF:

\[ F' : (Q_1 \land Q_2) \lor (R_1 \lor R_2). \]

Then apply distributivity to obtain

\[ F'' : (Q_1 \lor R_1 \lor R_2) \land (Q_2 \lor R_1 \lor R_2) \]

which is in CNF and equivalent to \( F \).
Outline

1. Syntax
2. Semantics (meaning)
3. Satisfiability and Validity
4. Equivalence and Implication
5. Substitution
6. Normal Forms
7. Decision Procedures for Satisfiability
A decision procedure for satisfiability of PL formulae reports, after some finite amount of computation, whether a given PL formula $F$ is satisfiable.

Algorithm 1: Decision procedure based on the truth-table method

```plaintext
let rec SAT F =
  if $F = \top$ then true
  else if $F = \bot$ then false
  else
    let $P = \text{CHOOSE vars}(F)$ in
    (SAT $F\{P \mapsto \top\}$) \lor (SAT $F\{P \mapsto \bot\}$)
```

SAT is a recursive function that takes one argument. This algorithm returns true immediately upon finding a satisfying interpretation.
Consider the formula

\[ F : (P \rightarrow Q) \land P \land \neg Q. \]

To compute \textbf{SAT} \( F \), choose a variable, say \( P \), and recurse on the first case,

\[ F\{P \mapsto \top\} : (\top \rightarrow Q) \land \top \land \neg Q, \]

which simplifies to

\[ F_1 : Q \land \neg Q. \]

Now try each of

\[ F_1\{Q \mapsto \top\} \quad \text{and} \quad F_1\{Q \mapsto \bot\}. \]

Both simplifies to \( \bot \), so this branch ends without finding a satisfying interpretation.
Now try the other branch for $P$ in $F$:

$$F\{ P \mapsto \bot \} : (\bot \rightarrow Q) \land \bot \land \neg Q,$$

which simplifies to $\bot$. This branch also ends without finding a satisfying interpretation. Thus, $F$ is unsatisfiable.

**Figure:** Visualizing runs of SAT
Example 29

Consider the formula

\[ F : (P \rightarrow Q) \land \neg P. \]

To compute \textbf{SAT} \( F \), choose a variable, say \( P \), and recurse on the first case,

\[ F\{P \rightarrow \top\} : (\top \rightarrow Q) \land \neg \top, \]

which simplifies to \( \bot \). Therefore, try

\[ F\{P \rightarrow \bot\} : (\bot \rightarrow Q) \land \neg \bot, \]

instead, which simplifies to \( \top \). Arbitrarily assigning a value to \( Q \) produces the following satisfying interpretation:

\[ \mathcal{I} : \{P \mapsto \text{false}, Q \mapsto \text{true}\}. \]
1. Syntax
2. Semantics (meaning)
3. Satisfiability and Validity
4. Equivalence and Implication
5. Substitution
6. Normal Forms

7. Decision Procedures for Satisfiability
   - Conversion
   - The Resolution Procedure
   - DPLL
The transformation (Section 6) produces an equivalent formula that can be exponentially larger: e.g., DNF to CNF.

To decide the satisfiability of $F$, we need only examine an equisatisfiable formula $F'$.

$F$ and $F'$ are equisatisfiable when $F$ is satisfiable iff $F'$ is satisfiable.

**Conversion (at most constant factor larger)**

1. introduce new propositional variables to represent the subformulae of formula $F$;
2. add extra clauses (in $F'$) that assert that these new variables are equivalent to the subformulae that they represent.
7.1 PL formulae to Equisatisfiable CNF Formulae

**Representative Function (Rep)**

\[
\text{Rep} : \text{PL} \rightarrow \mathcal{V} \cup \{\top, \bot\} \\
F \mapsto P_F
\]

PL is the set of PL formulae, \(\mathcal{V}\) represents the set of propositional variables. \(P_F\) provides a compact way of referring to \(F\).

**Encoding Function (En)**

\[
\text{En} : \text{PL} \rightarrow \text{PL} \\
F \mapsto F'
\]

intended to map a PL formula \(F\) to a PL formula \(F'\) in CNF that asserts that \(F\)'s representative, \(P_F\), is equivalent to \(F\): \(\text{Rep}(F) \leftrightarrow F\).
### 7.1 PL formulae to Equisatisfiable CNF Formulae

#### Base cases for defining \( \text{Rep} \) and \( \text{En} \)

On \( \top, \bot \) and propositional variables \( P \):

<table>
<thead>
<tr>
<th>Formula</th>
<th>( \text{Rep} ) Value</th>
<th>( \text{En} ) Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \top )</td>
<td>( \top )</td>
<td>( \text{Rep}(\top) \iff \top = \top \iff \top = \top )</td>
</tr>
<tr>
<td>( \bot )</td>
<td>( \bot )</td>
<td>( \text{Rep}(\bot) \iff \bot = \bot \iff \bot = \bot )</td>
</tr>
<tr>
<td>( P )</td>
<td>( P )</td>
<td>( \text{Rep}(P) \iff P = P \iff P = \top )</td>
</tr>
</tbody>
</table>

For the inductive case, \( F \) is a formula other than an atom:

\[
\text{Rep}(F) = P_F.
\]

\( \text{En} \) then asserts the equivalence of \( F \) and \( P_F \) as a CNF formula.
On Conjunction

On conjunction, \( F_1 \land F_2 \):

\[
\text{En}(F_1 \land F_2) = \\
\text{let } P = \text{Rep}(F_1 \land F_2) \text{ in} \\
(\neg P \lor \text{Rep}(F_1)) \land (\neg P \lor \text{Rep}(F_2)) \land (\neg \text{Rep}(F_1) \lor \neg \text{Rep}(F_2) \lor P)
\]

The returned formula

\[
(\neg P \lor \text{Rep}(F_1)) \land (\neg P \lor \text{Rep}(F_2)) \land (\neg \text{Rep}(F_1) \lor \neg \text{Rep}(F_2) \lor P)
\]

is in CNF and is equivalent to

\[
\text{Rep}(F_1 \land F_2) \leftrightarrow \text{Rep}(F_1) \land \text{Rep}(F_2).
\]

In detail, the first two clauses together assert \( P \leftrightarrow \text{Rep}(F_1) \land \text{Rep}(F_2) \), the last clause asserts \( \text{Rep}(F_1) \land \text{Rep}(F_2) \rightarrow P \).
7.1 PL formulae to Equisatisfiable CNF Formulae

On Negation

$\text{En}(\neg F)$ returns a formula equivalent to $\text{Rep}(\neg F) \leftrightarrow \neg \text{Rep}(F)$:

$$\text{En}(\neg F) =$$

let $P = \text{Rep}(\neg F)$ in

$$(\neg P \lor \neg \text{Rep}(F)) \land (P \lor \text{Rep}(F))$$

On Disjunction

$$\text{En}(F_1 \lor F_2) =$$

let $P = \text{Rep}(F_1 \lor F_2)$ in

$$(\neg P \lor \text{Rep}(F_1) \lor \text{Rep}(F_2)) \land (\neg \text{Rep}(F_1) \lor P) \land (\neg \text{Rep}(F_2) \lor P)$$
7.1 PL formulae to Equisatisfiable CNF Formulae

On Implication

\[ \text{Rep}(F_1 \rightarrow F_2) = \]
\[ \text{let } P = \text{Rep}(F_1 \rightarrow F_2) \text{ in} \]
\[ (\neg P \lor \neg \text{Rep}(F_1) \lor \text{Rep}(F_2)) \land (\text{Rep}(F_1) \lor P) \land (\neg \text{Rep}(F_2) \lor P) \]

On IIF

\[ \text{En}(F_1 \leftrightarrow F_2) = \]
\[ \text{let } P = \text{Rep}(F_1 \leftrightarrow F_2) \text{ in} \]
\[ (\neg P \lor \neg \text{Rep}(F_1) \lor \text{Rep}(F_2)) \land (\neg P \lor \text{Rep}(F_1) \lor \neg \text{Rep}(F_2)) \land (P \lor \neg \text{Rep}(F_1) \lor \neg \text{Rep}(F_2)) \land (P \lor \text{Rep}(F_1) \lor \text{Rep}(F_2)) \]
If $S_F$ is the set of all subformulae of $F$ (including $F$ itself), then

$$F' : \text{Rep}(F) \land \bigwedge_{G \in S_F} \text{En}(G)$$

is in CNF and is equisatisfiable to $F$.

**Example 30**

Consider formula

$$F : (Q_1 \land Q_2) \lor (R_1 \land R_2),$$

which is in DNF. To convert it to CNF, we collect its subformulae

$$S_F : \{Q_1, Q_2, Q_1 \land Q_2, R_1, R_2, R_1 \land R_2, F\}$$
7.1 PL formulae to Equisatisfiable CNF Formulae

Continue Example 30, compute

\[\begin{align*}
  \text{En}(Q_1) &= \top, \quad \text{En}(Q_2) = \top, \quad \text{En}(R_1) = \top, \quad \text{En}(R_2) = \top \\
  \text{En}(Q_1 \land Q_2) &= (\neg P_{(Q_1 \land Q_2)} \lor Q_1) \land (\neg P_{(Q_1 \land Q_2)} \lor Q_2) \\
  &\quad \land (\neg Q_1 \lor \neg Q_2 \lor P_{(Q_1 \land Q_2)}) \\
  \text{En}(R_1 \land R_2) &= (\neg P_{(R_1 \land R_2)} \lor R_1) \land (\neg P_{(R_1 \land R_2)} \lor R_2) \\
  &\quad \land (\neg R_1 \lor \neg R_2 \lor P_{(R_1 \land R_2)}) \\
  \text{En}(F) &= (\neg P_{(F)} \lor P_{(Q_1 \land Q_2)} \lor P_{(R_1 \land R_2)}) \\
  &\quad \land (\neg P_{(Q_1 \land Q_2)} \lor P_{(F)}) \land (\neg P_{(R_1 \land R_2)} \lor P_{(F)})
\end{align*}\]

Then

\[F' = P_{(F)} \land \bigwedge_{G \in S_F} \text{En}(G)\]

is equisatisfiable to \( F \) and is in CNF.
1. Syntax

2. Semantics (meaning)

3. Satisfiability and Validity

4. Equivalence and Implication

5. Substitution

6. Normal Forms

7. Decision Procedures for Satisfiability
   - Conversion
   - The Resolution Procedure
   - DPLL
7.2 The Resolution Procedure

Resolution follows from the following observation of any PL formula \( F \) in CNF:

- To satisfy clauses \( C_1[P] \) and \( C_2[\neg P] \) that share variable \( P \) but disagree on its value, either the rest of \( C_1 \) or the rest of \( C_2 \) must be satisfied;
- If \( P \) is \textbf{true}, then a literal other than \( \neg P \) in \( C_2 \) must be satisfied;
- If \( P \) is \textbf{false}, then a literal other than \( P \) in \( C_1 \) must be satisfied.

Resolution

Clausal resolution is stated as the following proof rule:

\[
\frac{C_1[P] \quad C_2[\neg P]}{C_1[\bot] \lor C_2[\bot]}
\]

In which, \( P \leftrightarrow \bot \) for \( C_1 \), and \( \neg P \leftrightarrow \bot \) in \( C_2 \). From the two clauses of the premise, deduce the new clause, called the \textbf{resolvent}. The resolvent represents that other literals (excludes \( P \) and \( \neg P \)) have to be satisfied.
The CNF of \((P \rightarrow Q) \land P \land \neg Q\) is the following:

\[
F : (\neg P \lor Q) \land P \land \neg Q.
\]

From resolution

\[
\frac{(\neg P \lor Q) \quad P}{Q},
\]

construct

\[
F_1 : (\neg P \lor Q) \land P \land \neg Q \land Q.
\]

From resolution

\[
\frac{\neg Q \quad Q}{\perp},
\]

deduce that \(F\) is unsatisfiable.
Example 32

Consider the formula

\[ F : (\neg P \lor Q) \land \neg Q. \]

The one possible resolution

\[
\frac{\neg P \lor Q}{\neg P},
\]

yields

\[ F_1 : (\neg P \lor Q) \land \neg Q \land \neg P. \]

Since no further resolutions are possible, \( F \) is satisfiable. Indeed,

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

is a satisfying interpretation.
1 Syntax

2 Semantics (meaning)

3 Satisfiability and Validity

4 Equivalence and Implication

5 Substitution

6 Normal Forms

7 Decision Procedures for Satisfiability
   - Conversion
   - The Resolution Procedure
   - DPLL
Modern satisfiability procedures for propositional logic are based on the \textbf{D}avis-\textbf{P}utnam-\textbf{L}ogemann-\textbf{L}oveland algorithm (DPLL)

- DPLL operates on PL formulae in CNF
- DPLL applies a restricted form of resolution: Boolean constraint propagation (BCP)
- BCP is based on unit resolution

\section*{Unit Resolution}

Unit resolution operates on two clauses. One clause, called the \textbf{unit clause}, consists of a single literal $\ell$. The second clause contains the negation of $\ell$: $C[\neg \ell]:$

$$
\begin{array}{c}
\ell \\
C[\ell] \\
\hline \\
C[\bot]
\end{array}
$$
Example 33

In the formula $F : (P) \land (\neg P \lor Q) \land (R \lor \neg Q \lor S)$, $(P)$ is a unit clause. Therefore applying unit resolution

$$
\frac{P \quad (\neg P \lor Q)}{Q}
$$

(partial interpretation $\{P \mapsto \top\}$)

produces $F' : (Q) \land (R \lor \neg Q \lor S)$. Applying unit resolution again

$$
\frac{Q \quad (R \lor \neg Q \lor S)}{R \lor S}
$$

(partial interpretation $\{Q \mapsto \top\}$)

produces

$$F'' : (R \lor S),$$

ending this round of BCP.
The implementation of DPLL is structurally similar to SAT, except that it begins by applying BCP:

Algorithm 2: Basic DPLL

```plaintext
let rec DPLL F =
    let F' = BCP F in
    if F' = ⊤ then true
    else if F' = ⊥ then false
    else
        let P = CHOOSE vars(F') in
        (DPLL F'{P ↦ ⊤}) ∨ (DPLL F'{P ↦ ⊥})
```

As in SAT, intermediate formulae are simplified according to the template equivalences.
Consider the formula

$$F : (P) \land (\neg P \lor Q) \land (R \lor \neg Q \lor S).$$

On the first level of recursion, \textbf{DPLL} recognizes the unit clause \((P)\) and applies the BCP steps from Example 33, resulting in the formula

$$F'' : R \lor S.$$

The unit resolutions correspond to the partial interpretation

$$\{P \mapsto \text{true}, Q \mapsto \text{true}\}.$$

Only positively occurring variables remain, so \(F\) is satisfiable. In particular,

$$\{P \mapsto \text{true}, Q \mapsto \text{true}, R \mapsto \text{true}, S \mapsto \text{true}\}$$

is a satisfyng interpretation of \(F\).
Example 35

Consider the formula

\[ F : (\neg P \lor Q \lor R) \land (\neg Q \lor R) \land (\neg Q \lor \neg R) \land (P \lor \neg Q \lor \neg R). \]

Branching on \( Q \) or \( R \) will result in unit clauses; choose \( Q \). Then

\[ F\{Q \mapsto \top\} : (R) \land (\neg R) \land (P \lor \neg R). \]

The unit resolution

\[
\begin{array}{c|c}
  R & (\neg R) \\
  \hline
  \top & \\
\end{array}
\]

finishes this branch. On the other branch,

\[ F\{Q \mapsto \bot\} : (\neg P \lor R). \]

\( P \) appears only negatively, and \( R \) appears only positively, so the formula is satisfiable. \( F \) is satisfied by \( \mathcal{I} : \{P \mapsto \text{false}, Q \mapsto \text{false}, R \mapsto \text{true}\}. \)
7.3 DPLL

Figure: Visualizing of Example 35
Summary

1 **Syntax.** How one constructs a PL formula. Propositional variables, atoms, literals, logical connectives

2 **Semantics.** What a PL formula means. Truth values true and false. Interpretations. Truth-table definition, inductive definition.

3 **Satisfiability and validity.** Whether a PL formula evaluates to true under any or all interpretations. Duality of satisfiability and validity, truth-table method, semantic argument method.

4 **Equivalence and implication.** Whether two formulae always evaluate to the same truth value under every interpretation. Whether under any interpretation, if one formula evaluates to true, the other also evaluates to true. Reduction to validity.

5 **Substitution,** which is a tool for manipulating formulae and making general claims. Substitution of equivalent formulae. Valid templates.

6 **Normal forms and Decision procedures for satisfiability.**