Roadmap

Previously
- Course overview
- Program synthesis overview

Today
- Review of propositional logic
- Normal forms for propositional logic
# Propositional logic (PL) syntax

<table>
<thead>
<tr>
<th>Atom</th>
<th>truth symbols</th>
<th>$\top$ (&quot;true&quot;) and $\bot$ (&quot;false&quot;)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>propositional variables</td>
<td>$p, q, r, p_1, q_1$</td>
</tr>
<tr>
<td>Literal</td>
<td>atom $\alpha$ or its negation $\neg\alpha$</td>
<td></td>
</tr>
<tr>
<td>Formula</td>
<td>literal or application of a logical connective to $F, F_1, F_2$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\neg F$</td>
<td>&quot;not&quot; (negation)</td>
</tr>
<tr>
<td></td>
<td>$F_1 \lor F_2$</td>
<td>&quot;or&quot; (disjunction)</td>
</tr>
<tr>
<td></td>
<td>$F_1 \land F_2$</td>
<td>&quot;and&quot; (conjunction)</td>
</tr>
<tr>
<td></td>
<td>$F_1 \to F_2$</td>
<td>&quot;implies&quot; (implication)</td>
</tr>
<tr>
<td></td>
<td>$F_1 \leftrightarrow F_2$</td>
<td>&quot;if and only if&quot; (iff)</td>
</tr>
</tbody>
</table>
Interpretation $I$: mapping of each propositional variable to a truth value

$I: \{ p \mapsto \top, q \mapsto \bot, \ldots \}$

Satisfying interpretation: $F$ evaluates to $\top$ under $I$, written $I \models F$

Falsifying interpretation: $F$ evaluates to $\bot$ under $I$, written $I \not\models F$
PL semantics: inductive definition

Base Cases:

\( I \models T \)
\( I \not\models \bot \)
\( I \models p \iff I[p] = T \)
\( I \not\models p \iff I[p] = \bot \)

Inductive Cases:

\( I \models \neg F \iff I \not\models F \)
\( I \models F_1 \lor F_2 \iff I \models F_1 \text{ or } I \models F_2 \)
\( I \models F_1 \land F_2 \iff I \models F_1 \text{ and } I \models F_2 \)
\( I \models F_1 \rightarrow F_2 \iff I \not\models F_1 \text{ or } I \models F_2 \)
\( I \models F_1 \leftrightarrow F_2 \iff I \models F_1 \text{ and } I \models F_2 \text{, or, } \)
\( I \not\models F_1 \text{ and } I \not\models F_2 \)
\( p \land q \Rightarrow (p \lor \neg q) \)

I: \( p \Rightarrow T, q \Rightarrow F \)

1. \( I \models p \quad I \models [p] = T \)
2. \( I \models q \quad I \models [q] = F \)
3. \( I \models \neg q \quad \text{2, and } \neg \)
4. \( I \models p \lor q \quad \text{2 and } \lor \)
5. \( I \models p \lor \neg q \quad 3/1, \text{and } \lor \)
6. \( I \models \neg \phi \quad 4, 5, \to \)
Satisfiability and Validity

$F$ is **satisfiable** iff there exists $I : I \models F$

$F$ is **valid** iff for all $I : I \models F$

**Duality:**

$F$ is valid iff $\neg F$ is unsatisfiable

Procedure for deciding satisfiability *or* validity suffices!
Deciding satisfiability/validity

- SAT solvers! (next lecture)

- Basic techniques
  - Truth table method: search-based
  - Semantic argument method: deductive technique

- SAT solvers combine search and deduction
Truth table method

1. Enumerate all interpretations
2. Search for satisfying interpretation

Brute-force!
Impractical (2^n interpretations)
Can’t be used if domain is not finite, e.g., for first-order logic

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

<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>⊥</td>
<td>⊥</td>
<td>⊥</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>⊥</td>
<td>T</td>
<td>⊥</td>
<td>⊥</td>
<td>⊥</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>⊥</td>
<td>⊥</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>⊥</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Semantic argument (decide validity)

Proof by contradiction:
1. Assume $F$ is not valid
2. Apply proof rules
3. Contradiction (i.e., $\bot$) along every branch of proof tree $\Rightarrow F$ is valid
4. Otherwise, $F$ is not valid

A bit of an overhead for PL
Applicable to first-order logic
\[ F : (p \land q) \rightarrow (p \lor \neg q) \]

Assume \( F \) is not valid, \( I \not\models F \)

1. \( I \not\models (p \land q) \rightarrow (p \lor \neg q) \) ass.

2. \( I \models p \land q \)

3. \( I \not\models p \lor \neg q \)

4. \( I \models p \)

5. \( I \not\models p \)

6. \( I \models \bot \)
Semantic judgements

\[ F_1 \text{ and } F_2 \text{ are equivalent } (F_1 \iff F_2) \text{ iff for all } I, I \models F_1 \iff F_2 \]

\[ F_1 \text{ implies } F_2 \ (F_1 \implies F_2) \text{ iff for all } I, I \models F_1 \implies F_2 \]

A procedure for deciding satisfiability can decide equivalence and implication!
A **normal form** for a logic is a syntactical restriction such that for every formula in the logic, there is an equivalent formula in the normal form.

Three useful normal forms for propositional logic:

- Negation Normal Form (NNF)
- Disjunctive Normal Form (DNF)
- Conjunctive Normal Form (CNF)
Negation Normal Form (NNF)

<table>
<thead>
<tr>
<th>Atom</th>
<th>(T, \bot), propositional variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Literal</td>
<td>(\text{Atom} \mid \neg\text{Atom})</td>
</tr>
<tr>
<td>Formula</td>
<td>(\text{Literal} \mid \text{Formula op Formula})</td>
</tr>
<tr>
<td>\textbf{op}</td>
<td>(\lor \mid \land)</td>
</tr>
</tbody>
</table>

The only logical connectives are \(\neg, \land, \lor\)

Negations appear only in literals

Conversion to NNF:

Eliminate \(\rightarrow\) and \(\leftrightarrow\)

“Push negations in” using **DeMorgan’s Laws**:

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

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

\(\forall\phi \lor \neg \eta \equiv \neg (\phi \land \eta)\)
Disjunctive Normal Form (DNF)

- **Atom**: $T$, $\bot$, propositional variables
- **Literal**: $\text{Atom} \mid \neg \text{Atom}$
- **Disjunct**: $\text{Literal} \land \text{Disjunct}$
- **Formula**: $\text{Disjunct} \lor \text{Formula}$

**Conversion to DNF:**

1. First convert to NNF
2. Distribute $\land$ over $\lor$

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

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

Deciding satisfiability of DNF formulas is trivial.

Why not convert all PL formulas to DNF for SAT solving?

Exponential blow-up of formula size in DNF conversion!
## Conjunctive Normal Form (CNF)

<table>
<thead>
<tr>
<th>Atom</th>
<th>T, ⊥, propositional variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Literal</td>
<td>Atom</td>
</tr>
<tr>
<td>Clause</td>
<td>Literal ∨ Clause</td>
</tr>
<tr>
<td>Formula</td>
<td>Clause ∧ Formula</td>
</tr>
</tbody>
</table>

### Conversion to CNF:

First convert to NNF

Distribute ∨ over ∧

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

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

Deciding satisfiability of CNF formulas is not trivial.

CNF conversion must also exhibit an exponential blow-up of formula size.

Yet, almost all SAT solvers convert to CNF first before solving. Why?
Equisatisfiability and Tseitin’s Transformation

Two formulas $F_1$ and $F_2$ are **equisatisfiable** iff: $F_1$ is satisfiable iff $F_2$ is satisfiable

**Tseitin’s transformation** converts any PL formula $F_1$ to equisatisfiable formula $F_2$ in CNF with only a **linear** increase in size

Note that equisatisfiability is a much weaker notion than equivalence, but is adequate for checking satisfiability.
Tseitin’s Transformation

1. Introduce an auxiliary variable \( \text{rep}(G) \) for each subformula \( G = G_1 \ op \ G_2 \) of formula \( F_1 \)

2. Constrain auxiliary variable to be equivalent to subformula: \( \text{rep}(G) \leftrightarrow \text{rep}(G_1) \ op \ \text{rep}(G_2) \)

3. Convert equivalence constraint to CNF: \( \text{CNF}(\text{rep}(G) \leftrightarrow \text{rep}(G_1) \ op \ \text{rep}(G_2)) \)

4. Let \( F_2 \) be \( \text{rep}(F) \land \bigwedge_G \text{CNF}(\text{rep}(G) \leftrightarrow \text{rep}(G_1) \ op \ \text{rep}(G_2)) \). Check if \( F_2 \) is satisfiable.

\( F_1 \) and \( F_2 \) are equisatisfiable!

Size of each equivalence constraint is bounded by a constant

This restricts the size of \( F_2 \) to be linear in the size of \( F_1 \): \( |F_2| = 30 \cdot |F_1| + 2 \)
Summary

Today
- Review of propositional logic
- Normal forms for propositional logic

Next
- SAT Solving