Propositional logic
as a formalisms for declarative knowledge processing

We start with classical propositional logic, the simplest possible logic

Recall that, for each formalism, we study:

1. Background and motivation
2. The syntax and semantics of the formalism
3. Main reasoning problems
4. Reasoning techniques and algorithms
5. Computational complexity and implementation of reasoners
Outline

1. Introduction

2. Syntax

3. Semantics

4. Reasoning Problems
   4.1 SAT, the satisfiability problem
   4.2 Other reasoning problems

5. Reasoning in Propositional Logic
   5.1 SAT Algorithms and their complexity
   5.2 Complexity of Reasoning

6. SAT in the real world
Modeling problems in Propositional Logic

- Propositional logic is the simples and most basic logic.

- We use propositional symbols (aka Boolean variables) to represent propositions, i.e., possible facts that are either true or false.

<table>
<thead>
<tr>
<th>$p$</th>
<th>Paul is outside</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r$</td>
<td>It is raining</td>
</tr>
<tr>
<td>$u$</td>
<td>Paul has an umbrella</td>
</tr>
<tr>
<td>$w$</td>
<td>Paul is wet</td>
</tr>
</tbody>
</table>

- $overheat1$ Engine 1 is overheated
- $valveOpen1$ The safety valve of engine 1 is open
- $alarm$ The alarm is activated
Modeling problems in Propositional Logic

- Boolean combinations of these symbols allow us to build more complex statements

\[ p \land r \land \neg u, \quad p \land r \land \neg u \rightarrow w \]
\[ overheat1 \land (\neg valveOpen1 \lor \neg alarm) \]

- We can infer properties of the described domain by reasoning about these statements

Is Paul wet?
Is the system safe?

Recall:
- \( p \): Paul is outside
- \( u \): Paul has an umbrella
- \( r \): it is raining
- \( w \): Paul is wet
Outline

1. Introduction

2. Syntax

3. Semantics

4. Reasoning Problems
   4.1 SAT, the satisfiability problem
   4.2 Other reasoning problems

5. Reasoning in Propositional Logic
   5.1 SAT Algorithms and their complexity
   5.2 Complexity of Reasoning

6. SAT in the real world
Syntax of Propositional Logic

- We start from a countable set $V$ of *propositional variables* (or simply *propositions*).

- Propositional formulas are defined inductively:
  - Every $p \in V$ is a formula.
  - $\top$ (verum, top) and $\bot$ (falsum, bottom) are formulas.
  - If $\phi_1$ and $\phi_2$ are formulas, then so are $(\neg \phi_1)$ (negation), $(\phi_1 \land \phi_2)$ (conjunction), and $(\phi_1 \lor \phi_2)$ (disjunction).

We can use $(\phi_1 \rightarrow \phi_2)$ as a shortcut for $(\neg \phi_1 \lor \phi_2)$ and $(\phi_1 \leftrightarrow \phi_2)$ as a shortcut for $(\phi_1 \rightarrow \phi_2) \land (\phi_2 \rightarrow \phi_1)$. 
3. Semantics

4. Reasoning Problems
4.1 SAT, the satisfiability problem
4.2 Other reasoning problems

5. Reasoning in Propositional Logic
5.1 SAT Algorithms and their complexity
5.2 Complexity of Reasoning

6. SAT in the real world
Interpretations

- We consider two truth values: true and false

- A interpretation (aka truth assignment) is a mapping $\mathcal{I}$ that assigns either true or false to each propositional variable, i.e.,

$$\mathcal{I}: \mathcal{V} \mapsto \{\text{true, false}\},$$

and such that $\mathcal{I}(\top) = \text{true}$ and $\mathcal{I}(\bot) = \text{false}$.

- An interpretation $\mathcal{I}$ is inductively extended to all formulas (next slide)

Once we fix an interpretation (i.e., values for the propositional variables), the value of all formulas is unambiguously determined.
Interpretations (ctd.)

Given an interpretation $\mathcal{I}$, the semantics of formulas is inductively defined as follows:

$$\mathcal{I}(\neg \phi) = \begin{cases} 
  \text{true} & \text{if } \mathcal{I}(\phi) = \text{false}, \\
  \text{false} & \text{otherwise.}
\end{cases}$$

$$\mathcal{I}(\phi \land \psi) = \begin{cases} 
  \text{true} & \text{if } \mathcal{I}(\phi) = \text{true} \text{ and } \mathcal{I}(\psi) = \text{true}, \\
  \text{false} & \text{otherwise.}
\end{cases}$$

$$\mathcal{I}(\phi \lor \psi) = \begin{cases} 
  \text{true} & \text{if } \mathcal{I}(\phi) = \text{true} \text{ or } \mathcal{I}(\psi) = \text{true}, \\
  \text{false} & \text{otherwise.}
\end{cases}$$

Questions:

- What is the meaning of the defined connectives $\rightarrow$ and $\leftrightarrow$?
- Given a formula $\phi$, does the value of $\phi$ in $\mathcal{I}$ depend on all the propositional symbols in $\mathcal{V}$? Why?

Exercises: You should be able to determine the semantics (truth value) of any given formula in any given interpretation.
Models, Logical Implication

- An interpretation $\mathcal{I}$ is a model of a formula $\phi$ if $\mathcal{I}(\phi) = \text{true}$. In this case we write
  \[ \mathcal{I} \models \phi \]
- For a set of formulas $\Gamma$ (aka theory), we say $\mathcal{I}$ is a model of $\Gamma$ if
  \[ \mathcal{I}(\phi) = \text{true} \text{ for every } \phi \in \Gamma \]
- We say that a formula $\phi$ follows from a theory $\Gamma$, if every model of $\Gamma$ is a model of $\phi$. In this case we write
  \[ \Gamma \models \phi \]

**Questions:**
- What can we say about $\Gamma \models \phi$ if $\phi \in \Gamma$?
- What is the intuitive meaning of $\emptyset \models \phi$?

- We write $\models \phi$ instead of $\emptyset \models \phi$, and say that $\phi$ is valid (true in all interpretations)
Outline

1. Introduction
2. Syntax
3. Semantics
4. Reasoning Problems
   4.1 SAT, the satisfiability problem
   4.2 Other reasoning problems
5. Reasoning in Propositional Logic
   5.1 SAT Algorithms and their complexity
   5.2 Complexity of Reasoning
6. SAT in the real world
SAT, the satisfiability problem

The most important reasoning problem in Propositional Logic is the **satisfiability problem**

---

**Formula satisfiability**

A formula $\phi$ is satisfiable if there exists a model for $\phi$

---

**SAT, the satisfiability problem**

Input: A formula $\phi$

Question: Is $\phi$ satisfiable?

- SAT is a **decision problem**, i.e., the answer is either *yes* or *no*

- We call UNSAT the complementary problem, i.e., to decide if $\phi$ is not satisfiable
Other Reasoning Problems

**Validity**

Input: A formula $\phi$
Question: Does $\models \phi$ hold?

**Logical Consequence**

Input: A theory $\Gamma$, a formula $\phi$
Question: Does $\Gamma \models \phi$ hold?

**Modelhood (aka satisfaction)**

Input: An interpretation $\mathcal{I}$, a formula $\phi$
Question: Does $\mathcal{I} \models \phi$ hold?

*Note:* These are all decision problems, but there are other reasoning problems that are not decision problems (finding a model, counting models, etc.)
5. Reasoning in Propositional Logic

4. Reasoning Problems
4.1 SAT, the satisfiability problem
4.2 Other reasoning problems

5. Reasoning in Propositional Logic
5.1 SAT Algorithms and their complexity
5.2 Complexity of Reasoning

6. SAT in the real world
### Solving different Reasoning Problems

We have mentioned different reasoning problems:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Input</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modelhood</td>
<td>$\mathcal{I}, \phi$</td>
<td>Does $\mathcal{I} \models \phi$?</td>
</tr>
<tr>
<td>Satisfiability</td>
<td>$\phi$</td>
<td>Is there some $\mathcal{I}$ such that $\mathcal{I} \models \phi$?</td>
</tr>
<tr>
<td>Validity</td>
<td>$\phi$</td>
<td>Does $\mathcal{I} \models \phi$ for every $\mathcal{I}$?</td>
</tr>
<tr>
<td>Logical Consequence</td>
<td>$\Gamma, \phi$</td>
<td>Does $\mathcal{I} \models \phi$ for every model $\mathcal{I}$ of $\Gamma$?</td>
</tr>
</tbody>
</table>

**Questions:**
- How do we solve modelhood? (it’s very easy!)
- Why do the other problems seem to be harder?
- Do we need an algorithm for each problem?

Validity and logical consequence can be reduced to UNSAT.

Hence, an algorithm for (UN)SAT suffices.
A non-deterministic algorithm for SAT

Assume we have an algorithm $models(I, \phi)$ that decides modelhood.

Let $\phi$ be a given propositional formula, and let $\{p_1, \ldots, p_n\}$ be the set of all propositional variables in $\phi$. Then $SAT(\phi)$ decides the satisfiability of $\phi$:

**Algorithm $SAT(\phi)$**

1. guess an assignment $I: \{p_1, \ldots, p_n\} \mapsto \{\text{true}, \text{false}\}$;

2. return $models(I, \phi)$;

- **Soundness** is trivial: if $SAT(\phi)$ says yes, then $\phi$ has a model $I$.

- **Completeness** is also trivial: if there is a model for $\phi$, then there is a way to guess an interpretation $I$ such that $models(I, \phi)$ says yes, and hence $SAT(\phi)$ says yes.
A non-deterministic algorithm for SAT - complexity

- It is not hard to show that $\text{models}(I, \phi)$ can be run in time polynomial (linear) in the size of $\phi$

- Storing a guessed interpretation $I$ needs only polynomial (linear) space

- Hence, $\text{SAT}(\phi)$ needs only polynomial time

However, in practice, we can not really guess the right interpretation!

If $n$ propositional variables occur in $\phi$, then there are $2^n$ possible interpretations that have to be tested!
A deterministic algorithm for SAT

Assume we have an algorithm $selectProp(\phi)$ that returns a propositional variable occurring in $\phi$.

We denote by $\phi[p/\top]$ (resp. $\phi[p/\bot]$) the result of replacing in $\phi$ each instance of the propositional letter $p$ by $\top$ (resp. $\bot$) and simplifying the resulting formula.

Algorithm $SAT(\phi)$

1. if $\phi == \top$ then return true;
2. if $\phi == \bot$ then return false;
3. $p := selectProp(\phi)$;
4. return $(SAT(\phi[p/\top]) \text{ OR } SAT(\phi[p/\bot]))$;

Exercises: Can you explain why the algorithm is sound and complete? Why does it terminate? How much time and space does it require?
Complexity of SAT

From the non-deterministic algorithm we discussed, we can infer:

**Theorem.** SAT is in NP

Proof sketch: there is a non-deterministic algorithm for SAT that runs in polynomial time

Is this optimal, or can we solve SAT more efficiently?

**Theorem [Cook 70, Levin 71].** SAT is NP-hard

Proof sketch: For any non-deterministic polynomial time Turing machine $M$ and input word $w$, we can build in polynomial time a formula $\phi_{M,w}$ such $\phi_{M,w}$ is satisfiable iff $M$ has an accepting run on $w$. 
Complexity of reasoning in propositional logic

From the two theorems above, it follows that:

**Corollary.** SAT is in NP-complete

We can also easily infer that:

- The **validity** problem is coNP-complete
- The **logical implication** problem is coNP-complete

**Question:**
- What can you say about the complexity of the modelhood problem?
Outline

1. Introduction
2. Syntax
3. Semantics
4. Reasoning Problems
   4.1 SAT, the satisfiability problem
   4.2 Other reasoning problems
5. Reasoning in Propositional Logic
   5.1 SAT Algorithms and their complexity
   5.2 Complexity of Reasoning
6. SAT in the real world
SAT Solvers and applications

- Nowadays there are many SAT solvers available e.g., MiniSAT, Chaff, HyperSAT, WalkSAT, precosat, LySAT, SATzilla, ManySAT . . .

- Although they all need exponential time in the worst case (why?), some of them can efficiently handle huge formulas with thousands of propositional variables

- Many of them work on formulas in clausal form, aka Conjunctive Normal Form (CNF)

- Many SAT solvers employ optimized versions of the famous Davis-Putnam-Logemann-Loveland algorithm (DPLL)

- There is an international SAT Competition and a SAT Race every year, see http://www.satcompetition.org/
SAT Solvers and applications (ctd.)

- SAT solvers can be applied for solving any NP-complete problem
- They are employed in a huge range of fields
  
  See Marques-Silva, Practical Applications of Boolean Satisfiability, WODES’08:
  
  - Noise prediction in integrated circuits
  - Termination analysis in term-rewriting systems
  - Model-checking of finite state systems
  - Design debugging
  - AI planning
  - Package management in software distributions
  - Software model-checking and testing
  - Haplotype inference in bioinformatics

- SAT solving is an extremely active and large research field, with hundreds of researchers, thousands of publications, own conferences and journals
  