In this section we define the syntax and semantics of classical propositional logic and examine some of its most basic properties. Our treatment is similar to that of standard mathematical logic texts, except that we will bring in some computational issues.

It is essential to distinguish syntax from semantics. Syntax is concerned with which strings of symbols are to be used in certain roles (e.g., formulas and formal proofs), without regard to their meaning, while semantics is concerned with their meaning.

1 Formulas of Propositional Logic

The formulas of propositional logic are over an alphabet consisting of:

- A countably infinite set of symbols, \( P_1, P_2, \ldots \), called “propositional atoms”;
- A set of “connectives”, namely the symbols \( \land, \lor, \neg \);
- Parentheses: ( and ).

Connectives \( \land, \lor \) and \( \neg \) are read “and”, “or” and “not”, respectively. We will often use \( P, Q, R \) for atoms, and sometimes strings in camel font such as \textit{SantaIsReal}, \textit{ColouredRed}, etc., all of which may be subscripted. On occasion we will use [ and ] as parentheses to improve readability.

**Definition 1.** The set of formulas of propositional logic is defined inductively as follows.

1. each propositional atom is a formula;
2. if \( A \) is a formula then \( \neg A \) is a formula;
3. if $A$ and $B$ are formulas, then $(A \lor B)$ is a formula;

4. if $A$ and $B$ are formulas, then $(A \land B)$ is a formula.

Anything which cannot be constructed by finitely many applications of the rules 1-4 is not a formula.

For the remainder of this section, “formula” means “formula of propositional logic”. We will typically use letters $A$, $B$, $C$, etc., to stand for formulas (which could be atoms, and could be the same formula).

For any two formulas $A$ and $B$, we define $(A \rightarrow B)$ (“$A$ implies $B$” - sometimes written $A \supset B$) to be an abbreviation for $(\neg A \lor B)$, and $(A \leftrightarrow B)$ (“$A$ is equivalent to $B$”) to be an abbreviation for $((A \rightarrow B) \land (B \rightarrow A))$.

**Example 1.** The set of formulas includes: $A$, $(A \lor B)$, $(A \lor A)$, $(P \land (Q \lor \neg Q))$, and $((P \lor Q) \rightarrow (Q \lor P))$. The set of formulas does not contain $()$, $\neg$, $(\lor A B)$, or $P \land Q \land R$ or $(P \leftarrow R)$.

Strictly speaking, $(P)$, $A \land \neg B$, and $P \land Q \land R$ are not formulas. However, in most situations we will allow leaving out parentheses, or writing extra parentheses, as long as the intended formula is clear.

A sub-formula of a formula $A$ is any string which is a sub-string of $A$ and is a formula.

**Example 2.** Let $A$ be $(P \land (Q \lor \neg R))$. The sub-formulas of $A$ are $(P \land (Q \lor \neg R))$, $P$, $(Q \lor \neg R)$, $Q$, $\neg R$ and $R$. The proper sub-formulas exclude $A$ itself.

It is frequently convenient to view a formula $A$ as a labelled tree $T_A$, reflecting its' recursive construction. Each sub-tree of $T_A$ corresponds to a sub-formula of $A$; each node is labelled by the “main connective” of the corresponding sub-formula; the children of a node correspond to the immediate sub-formulas. Leaves of the tree correspond to, and are labelled with, atoms.

**Example 3.** The formula tree for $(\neg A \land (A \lor B))$ is:
2 Semantics of Propositional Logic

A truth assignment for a set \( S \) of atoms is a function mapping each element of \( S \) to \{true, false\}. The satisfaction relation between truth assignments and formulas is defined as follows.

Definition 2. Let \( \tau \) be a truth assignment for set \( S \) of atoms, and \( A \) a formula over the atoms in \( S \). We say that \( \tau \) satisfies \( A \), written \( \tau \models A \), in accordance with:

- if \( A \) is an atom, then \( \tau \models A \) iff \( \tau(A) = \text{true} \);
- if \( A \) is of the form \( \neg B \), then \( \tau \models A \) iff \( \tau \not\models B \);
- if \( A \) is of the form \( (B \lor C) \), then \( \tau \models A \) iff \( \tau \models B \) or \( \tau \models C \);
- if \( A \) is of the form \( (B \land C) \), then \( \tau \models A \) iff \( \tau \models B \) and \( \tau \models C \).

Thus, each truth assignment for \( S \) either satisfies \( A \) or does not. As is convenient, we may equivalently write or say \( \tau \models A \), \( \tau(A) = \text{true} \), “\( \tau \) satisfies \( A \)”, or “\( \tau \) makes \( A \) true”, and similarly \( \tau \not\models A \), \( \tau(A) = \text{false} \), “\( \tau \) does not satisfy \( A \)”, and “\( \tau \) makes \( A \)” false.

Example 4. Let \( \tau \) be a truth assignment with \( \tau(P) = \text{true} \) and \( \tau(Q) = \text{false} \). Then \( \tau \models (P \lor Q) \) but \( \tau \not\models (\neg P \land Q) \).

Definition 3. The Boolean Formula Value Problem (FVP) is:

Given: A formula \( A \) and a truth assignment \( \tau \) for the atoms of \( A \);
Question: Does \( \tau \models A \)?

It is easy to see FVP is polynomial time solvable. One method is a recursive procedure which computes the value, under \( \tau \), for each sub-formula \( B \) in terms of the immediate sub-formulas of \( B \), in accordance with the rules of Definition 2. We can think of this procedure labelling the nodes of the formula tree, bottom-up (starting from the leaves) with the values \( \tau \) gives to the sub-formulas corresponding to each sub-tree, using the rules of the semantic definition. The label on the root is the value given to \( A \) by \( \tau \). It is obvious that this can be done in polynomial time. (Determining the exact complexity of FVP is not so simple.)
3 Satisfiability and Validity

Definition 4. We say that formula $A$ is:

- Satisfiable, if there is some truth assignment for the atoms of $A$ that satisfies $A$;
- Unsatisfiable, if no truth assignment satisfies $A$;
- Valid (or is a tautology) if every truth assignment satisfies $A$;

Definition 4 defines a tri-partition of the formulas into: the set of valid formulas; the set of unsatisfiable formulas; and the set of formulas which are satisfiable but not valid (sometimes called “contingent”).

Example 5. For any atom $P$, $(P \lor \neg P)$ is valid, $(P \land \neg P)$ is unsatisfiable, and $\neg P$ is neither.

Proposition 1. Formula $A$ is unsatisfiable if and only if $\neg A$ is a tautology.

Exercise 1. Identify each of the following formulas as being unsatisfiable, valid, or satisfiable but not valid. For each formula which is satisfiable, give a satisfying assignment, and for each formula which is not valid, give a truth assignment under which it is false.

1. $(A \rightarrow A)$.
2. $((A \rightarrow B) \land (B \rightarrow C)) \rightarrow (A \rightarrow C)$.
3. $\neg A \lor ((A \land B) \lor (A \land \neg B))$
4. $(\neg(P \rightarrow R) \land ((\neg Q \lor R) \land \neg(P \land \neg Q)))$
5. $(\neg(\neg(P \lor Q) \lor (\neg(Q \lor R) \lor (\neg P \lor R)))$

We generalize the notion of satisfiability to sets of formulas. Let $S$ be a set of atoms and $\Gamma = \{A_1, A_2, \ldots\}$ a set of formulas over the atoms in $S$. We say a truth assignment $\tau$ for $S$ satisfies $\Gamma$ iff $\tau$ satisfies every formula in $\Gamma$. Then $\Gamma$ is satisfiable if there is a truth assignment which satisfies it, and is unsatisfiable otherwise.

Often it is useful to talk about the set of satisfying assignments for a formula or set of formulas. Satisfying assignments for a formula are sometimes called models of the formula. Thus, if $X$ is a formula or a set of formulas, we write $Mod X$ for the set of assignments to the atoms of $X$ which satisfy $X$.

Remark 1. The Formula Value Problem is also called the "Model Checking Problem", i.e., that of checking if a given truth assignment is a model of a given formula.
Definition 5. The Propositional Formula Satisfiability Problem (Propositional Satisfiability) is:

Given: A propositional formula $A$.
Question: Is $A$ satisfiable?

Let $n$ be the number of atoms in formula $A$, and $l$ the length of $A$ (i.e., total number of symbols). Propositional Satisfiability can be straightforwardly solved in time $O(l^2 2^n)$ by the method of truth tables: Construct a table of all truth assignments for the atoms of $A$, and check the value of $A$ under each truth assignment. Because there are $2^n$ rows in the truth table, this method could be used in practice only for very small formulas.

Example 6. With $n = 100$, there are $2^{100}$ (which is more than $10^{30}$), truth assignments to consider. A 10 GHz processor generating and evaluating one truth assignment every cycle would require more than $10^{13}$ years to check the full truth table.

If we want to do useful, real problem solving based on Propositional Satisfiability, we will need methods which are much more efficient than this. Unfortunately, all known algorithms require time $\Omega(2^n)$ in the worst case. (There are many special cases with algorithms better than this, a few of which we will look at in later sections.) Moreover, the problem is NP-Complete, so the existence of a polynomial time algorithm would imply equivalence of the complexity classes P and NP. The positive side of this fact is that we can represent a wide range of very complex problems – all the problems in NP – reasonably efficiently using this very simple language. For practical problem solving, we are interested in algorithms which, despite very poor asymptotic worst case performance, perform well enough to be useful in practice.

4 Logical Consequence and Logical Equivalence

Definition 6. Let $\Gamma = \{A_1, A_2, \ldots\}$ be a set of formulas, $B$ a formula, and $S$ a set of atoms containing every atom which appears in $\Gamma$ or in $B$. $B$ is a logical consequence of $\Gamma$ (or, $\Gamma$ logically entails $B$), written as $\Gamma \models B$, iff every truth assignment for $S$ which satisfies $\Gamma$ also satisfies $B$.

We have now overloaded the symbol $\models$. The meanings are closely related, but in one case we have a truth assignment on the left, and in the other we have a formula or set of
Example 7. $P \models (P \lor Q)$, but $(P \lor Q) \not\models P$.

If $\Gamma$ is a singleton set $\{A\}$, we may write $A \models B$ instead of $\{A\} \models B$, and if $\emptyset \models B$ (i.e., $B$ is valid), we may write $\models B$.

Proposition 2. Let $\Gamma$ be a set of formulas, and $B$ a formula. $\Gamma \models B$ iff $\Gamma \cup \{\neg B\}$ is unsatisfiable.

Proof. 
$\Rightarrow$ Assume $\Gamma \models B$, and consider an arbitrary truth assignment $\tau$ (for all the atoms contained in $\Gamma$ and $B$). There are two cases. If $\tau$ does not satisfy $\Gamma$, then it makes some formula in $\Gamma$ false, and thus does not satisfy $\Gamma \cup \{\neg B\}$. But if $\tau$ satisfies $\Gamma$, by assumption it also satisfies $B$. Thus, it makes $\neg B$ false, so does not satisfy $\Gamma \cup \{\neg B\}$.

$\Leftarrow$ Assume $\Gamma \cup \{\neg B\}$ is unsatisfiable. If $\Gamma$ is unsatisfiable, then trivially $\Gamma \models B$. Otherwise, consider any truth assignment $\tau$ that satisfies $\Gamma$. By assumption, $\tau$ makes $\neg B$ false, and therefore makes $B$ true. So $\Gamma \models B$.

Proposition 3. $A \models B$ if and only if $(A \rightarrow B)$ is a tautology.

Proposition 4. If $\Gamma \models A$ and $\Gamma \cup \{A\} \models B$ then $\Gamma \models B$.

Exercise 2. Prove, using careful semantic arguments:

1. Proposition 3.
2. The transitivity of logical consequence (Proposition 4).

Propositions 3 and 4 show that an algorithm for Satisfiability is sufficient to solve the problems of validity, logical implication (theorem proving), and logical equivalence (and similarly, that we might not expect to find an algorithm for these problem which is efficient in the worst case).

Definition 7. Formulas $A$ and $B$ are logically equivalent iff $A \models B$ and $B \models A$, in which case we write $A \equiv B$.

Proposition 5. Let $A$, $B$ and $C$ be arbitrary formulas. Then:

1. $(A \lor B) \equiv (B \lor A)$. 

6
2. \((A \land B) \equiv (B \land A)\).
3. \(\neg\neg A \equiv A\).
4. \(((A \lor B) \lor C) \equiv (A \lor (B \lor C))\).
5. \(((A \land B) \land C) \equiv (A \land (B \land C))\).
6. \(((A \land B) \lor C) \equiv ((A \lor C) \land (B \lor C))\).
7. \(((A \lor B) \land C) \equiv ((A \land C) \lor (B \land C))\).
8. \(\neg(A \lor B) \equiv (\neg A \land \neg B)\).
9. \(\neg(A \land B) \equiv (\neg A \lor \neg B)\).
10. \((A \rightarrow B) \equiv (\neg B \rightarrow \neg A)\).
11. \((A \lor A) \equiv A\).
12. \((A \land A) \equiv A\).

**Exercise 3.** Use truth tables or a careful semantic argument to verify each equivalence of Proposition 5.

**Exercise 4.** For each of the following, write the smallest logically equivalent formula you can find.

1. \((P \rightarrow (P \land Q))\);
2. \(((P \rightarrow Q) \lor Q))\);
3. \(((P \lor Q) \leftrightarrow (Q \lor R)) \rightarrow \neg Q\);
4. \(\neg((P \land Q) \leftrightarrow (P \land (Q \lor R)))\);
5. \(((P \land Q) \rightarrow R) \leftrightarrow (P \lor (Q \land R))\);
6. \(((P \rightarrow Q) \rightarrow (Q \rightarrow R)) \rightarrow (P \rightarrow R)\);

**Proposition 6.** Let \(A\) and \(B\) be formulas, both over the same set \(S\) of atoms. Then \(A \equiv B\) iff \(\text{Mod } A = \text{Mod } B\).

**Exercise 5.** The conclusion in Proposition 6 does not necessarily hold if \(A\) and \(B\) are not over the same set of atoms. Why?

---

Changes from version of Sept. 6, 2013: • Some re-ordering and re-wording; • Added the second direction to the proof of Proposition 2; • Added exercises on satisfiability and equivalence;