Notes on Satisfiability-Based Problem Solving

Propositional Logic

David Mitchell
mitchell@cs.sfu.ca

February 5, 2020

In this section we define the syntax and semantics of classical propositional logic and examine some basic properties. Our treatment is similar to standard mathematical logic texts, with a few computational issues added.

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 $\wedge$, $\vee$, and $\neg$;
- Parentheses: (and).

Connectives $\wedge$, $\vee$ and $\neg$ are read "and", "or" and "not", respectively. We will often use P, Q, and R for atoms, and sometimes strings such as *SantaIsReal*, *ColouredRed*, etc., all of which may be subscripted. We sometime 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 $(A \vee B)$ is a formula;
4. if A and B are formulas, then $(A \wedge B)$ 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).

For any two formulas A and B, we define $(A \to B)$ ("A implies B" - sometimes written $A \supset B$) to be an abbreviation for $(\neg A \vee B)$, and $(A \leftrightarrow B)$ ("A is equivalent to B") to be an abbreviation for $((A \to B) \wedge (B \to A))$.

Example 1. *The set of formulas includes:* A, $(A \vee B)$, $(A \vee A)$, $(P \wedge (Q_1 \vee \neg Q_2))$, *and* $((P \vee Q) \to (Q \vee P))$. *The set of formulas does not contain* $()$, $\neg$, $(\vee\ A\ B)$, $P \wedge Q \wedge$ *or* $(P \leftarrow R)$.

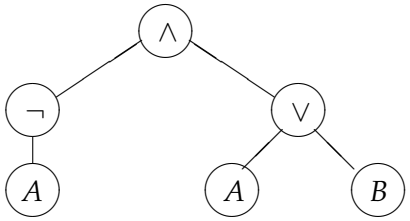
Strictly speaking, (P), $A \wedge (\neg B)$, and $P \wedge Q \wedge R$ are not formulas. However, we will sometimes leave out some parentheses if the intended meaning is clear.

A sub-formula of a formula A is any string which is a sub-string of A and is a formula. The *proper* sub-formulas exclude A itself.

Example 2. *Let* A *be* $(P \wedge (Q \vee \neg R))$. *The sub-formulas of* A *are* $(P \wedge (Q \vee \neg R))$, P, $(Q \vee \neg R)$, Q, $\neg R$ *and* R.

It is often useful to present a formula A as the corresponding "formula tree" T_A, the labelled tree reflecting its syntactic construction. Each sub-tree of T_A corresponds to a sub-formula of A, each node is labelled by the corresponding "main connective" and the children of a node correspond to immediate sub-formulas. Leaves correspond to, and are labelled with, atoms. T_A can be constructed in time $O(|A|)$ (for example with Dijkstra's "shunting yard algorithm").

Example 3. *The formula tree for* $(\neg A \wedge (A \vee B))$ *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 semantics of formulas is specified by the satisfaction relation between truth assignments and formulas, denoted $\models$, which we define next.

Definition 2. Let τ be a truth assignment for set S of atoms, and A a formula over the atoms in S. The satisfaction relation $\models$ on formulas and truth assignments is defined by the following rules:

- if A is an atom, then $\tau \models A$ iff $\tau(A) = true$;
- if A is of the form $\neg B$, then $\tau \models A$ iff $\tau \not\models B$;
- if A is of the form $(B \vee C)$, then $\tau \models A$ iff $\tau \models B$ or $\tau \models C$;
- if A is of the form $(B \wedge C)$, then $\tau \models A$ iff $\tau \models B$ and $\tau \models C$.

If $\tau \models A$ we say that τ satisfies A. Each truth assignment for the atoms of A either satisfies A or does not. In the first case, we may equivalently write or say $\tau \models A$, $\tau(A) = true$ or "τ makes A true". Similarly, if $\tau \not\models A$, we may say or write $\tau(A) = false$, "τ does not satisfy A" or "τ makes A false".

Example 4. *Let τ be a truth assignment with $\tau(P) = true$ and $\tau(Q) = false$. Then $\tau \models (P \vee Q)$ but $\tau \not\models (\neg P \wedge Q)$.*

Definition 3. The **Propositional Logic Model Checking** problem is:

> Given: A formula A and a truth assignment τ for the atoms of A;
> Question: Does $\tau \models A$?

It is easy to see propositional model checking can be done in polynomial time. For example, we can evaluate A by recursively evaluating all of its sub-formulas, determining the value of each by using the semantics of its main connective, as given by Definition 2. The recursion bottoms out on atoms. We can think of this procedure as labelling the nodes of the formula tree, bottom-up (starting from the leaves) with the values τ gives to the sub-formulas corresponding to each sub-tree. The label on the root is the value given to A by τ. With a little care this can be done in time $O(|A| \log n)$, where n is the number of distinct atoms.

Boolean Formula Value Problem (FVP), is the problem of evaluating a formula where

each atom has been labelled with its truth value, so is essentially the same problem as propositional model checking.

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, $(P \vee \neg P)$ is valid, $(P \wedge \neg P)$ 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 which makes it false.*

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

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 τ for S satisfies Γ iff τ satisfies every formula in Γ. Then Γ is satisfiable if there is a truth assignment which satisfies it, and is unsatisfiable otherwise.

Definition 5. The Propositional Satisfiability Problem is:

Given: A formula A of propositional logic.

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 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.) For practical problem solving, we are interested in algorithms which, despite very poor asymptotic worst case performance, perform well enough on problem instances we encounter in practice.

Propositional Satisfiability is NP-Complete, so the existence of a polynomial time algorithm would imply equivalence of the complexity classes P and NP. NP-completeness sounds like a negative result, because it suggests we cannot find an efficient algorithm. It also can be viewed as a positive result: we can represent a wide range of very complex problems – all the problems in NP – reasonably efficiently using this very simple language.

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 Γ or in B. B is a logical consequence of Γ (or, Γ logically entails B), written as $\Gamma \models B$, iff every truth assignment for S which satisfies Γ 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 formulas on the left.

Example 7. $P \models (P \vee Q)$, *but* $(P \vee Q) \not\models P$.

If Γ 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* Γ *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 τ (for all the atoms contained in Γ and B). We need to show τ does not satisfy $\Gamma \cup \{\neg B\}$. There are two cases. If τ does not satisfy Γ, then it makes some formula in Γ false, and thus does not satisfy $\Gamma \cup \{\neg B\}$. But if τ satisfies Γ, 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 Γ is unsatisfiable, then trivially $\Gamma \models B$. Otherwise, consider any truth assignment τ that satisfies Γ. By assumption, τ 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 (that is, in the manner of our proof of Proposition 2):*

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

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

Propositions 1 through 3 show that an algorithm for deciding Satisfiability is sufficient to solve the problems of validity, logical implication (theorem proving), and logical equivalence.

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

1. $(A \vee B) \equiv (B \vee A)$.
2. $(A \wedge B) \equiv (B \wedge A)$.

3. $\neg\neg A \equiv A$.
4. $((A \vee B) \vee C) \equiv (A \vee (B \vee C))$.
5. $((A \wedge B) \wedge C) \equiv (A \wedge (B \wedge C))$.
6. $((A \wedge B) \vee C) \equiv ((A \vee C) \wedge (B \vee C))$.
7. $((A \vee B) \wedge C) \equiv ((A \wedge C) \vee (B \wedge C))$.
8. $\neg(A \vee B) \equiv (\neg A \wedge \neg B)$.
9. $\neg(A \wedge B) \equiv (\neg A \vee \neg B)$.
10. $(A \rightarrow B) \equiv (\neg B \rightarrow \neg A)$.
11. $(A \vee A) \equiv A$.
12. $(A \wedge A) \equiv A$.

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. $(P \rightarrow (P \wedge Q))$;
2. $((P \rightarrow Q) \vee Q))$;
3. $((P \vee Q) \leftrightarrow (Q \vee R)) \rightarrow \neg Q$;
4. $\neg((P \wedge Q) \leftrightarrow (P \wedge (Q \vee R)))$;
5. $((P \wedge Q) \rightarrow R) \leftrightarrow (P \vee (Q \wedge R))$;
6. $((P \rightarrow Q) \rightarrow (Q \rightarrow R)) \rightarrow (P \rightarrow R)$.

Hint: *The minimum equivalent formula problem is very hard - much harder than satisfiability - so don't be surprised if these are challenging. Try different methods: re-write the formulas in simpler form, look for related sub formulas, construct the truth table, etc.*

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 may write *Mod* X for the set of assignments to the atoms of X which satisfy X.

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