Comp487/587 - Boolean Formulas

1 Logic and SAT

1.1 What is a Boolean Formula

- Logic is a way through which we can analyze and reason about simple or complicated events.
- In particular, we are interested in Boolean logic in which we simplify the events to be either 0 or 1, true or false. This strong simplification allows us to actually reason about the events, things that we cannot do with more complicated logics.
- One of the formal way to do this is to take atomic events, or propositions, that can each be true or false. From these we can construct a more complicated formula through operations like “and” or “or”.

1.2 Syntax of Boolean Formulas

Symbolically, a Boolean formula is a finite string which is constructed from:

- Variables: From a set **Vars** of variable symbols, e.g. $x_1, x_2 \cdots$
- Boolean operators: $\neg$ (negation), $\vee$ (disjunction, or), $\wedge$ (conjunction, and), $\rightarrow$ (implication), $\leftrightarrow$ (equivalence)
- Parenthesis: (,).

The definition of Boolean formulas is given recursively, as follows:

Definition 1. The set of Boolean formulas **Form** is the smallest set satisfying the following two properties:

- Every variable is a Boolean formula. That is, **Vars** $\subseteq$ **Form**.
- If ψ_1, ψ_2 are in **Form**, then so are $(\neg\psi_1)$, $(\psi_1 \vee \psi_2)$, $(\psi_1 \wedge \psi_2)$, $(\psi_1 \rightarrow \psi_2)$, and $(\psi_1 \leftrightarrow \psi_2)$.

Boolean formulas of the second type are called compound formulas, and the ψ_1 and ψ_2 are called the immediate subformulas.

Example 1. $\varphi = ((x_1 \wedge (\neg x_2)) \rightarrow ((\neg x_3) \wedge x_2))$ is a Boolean formula.

Sometimes we denote φ as $\varphi(x_1, x_2, x_3)$ to notate the (free/unquantified) variables in it. We will talk later on quantified variables.

1.3 Semantics of Boolean Formulas

A *truth assignment* is an assignment of either 1 (**true**) or 0 (**false**) to each variable. We will often use T to denote 1 and F to denote 0. Formally, a truth assignment is a function $\tau : \text{Vars} \rightarrow \{0, 1\}$. Notice that there are 2^n possible truth assignments over n variables.

We can use a particular truth assignment τ to evaluate a Boolean formula to be either **true** or **false**. In this way, a Boolean formula represents a function from the set of all possible truth assignments $\{\tau : \text{Vars} \rightarrow \{0, 1\}\}$ to the set $\{0, 1\}$. We give the computation of this function recursively, along the lines of the definition above:

- If φ is a variable, to determine the value of φ we can look directly at the truth assignment. That is, to determine $\varphi(\tau)$ we consider φ as a variable and use it to lookup into τ. Thus $\varphi(\tau) = \tau(\varphi)$.
- If φ is a compound formula (with immediate subformulas ψ_1 and ψ_2), then:
  - If $\varphi = (\neg\psi_1)$ then $\varphi(\tau) = 1$ iff $\psi_1(\tau) = 0$.
  - If $\varphi = (\psi_1 \vee \psi_2)$ then $\varphi(\tau) = 1$ iff $\psi_1(\tau) = 1$ or $\psi_2(\tau) = 1$.
  - If $\varphi = (\psi_1 \wedge \psi_2)$ then $\varphi(\tau) = 1$ iff $\psi_1(\tau) = 1$ and $\psi_2(\tau) = 1$.
  - If $\varphi = (\psi_1 \rightarrow \psi_2)$ then $\varphi(\tau) = 1$ iff $\psi_1(\tau) = 0$ or $\psi_2(\tau) = 1$.
  - If $\varphi = (\psi_1 \leftrightarrow \psi_2)$ then $\varphi(\tau) = 1$ iff either $(\psi_1(\tau) = 1$ and $\psi_2(\tau) = 1)$ or $(\psi_1(\tau) = 0$ and $\psi_2(\tau) = 0)$.

Example 2. Let $\varphi = ((x_1 \land (\neg x_2)) \to ((\neg x_3) \land x_2))$. Let $\tau = \{x_1 \mapsto 1, x_2 \mapsto 0, x_3 \mapsto 1\}$ i.e. the truth assignment which sets x_1 and x_3 to **true** and x_2 to **false**. Then $\varphi(\tau) = 0$.

In many cases is it easy to ditch the parenthesis (unless we really need them). If we do, we give precedence to $\neg$ over other the Boolean operators. Obviously $((x_1 \land x_2) \land x_3)$ is the same as $(x_1 \land (x_2 \land x_3))$ (and similarly for $\lor$) so we just write $(x_1 \land x_2 \land x_3)$.

You can think of a Boolean formula as a way to compactly represent a set of truth assignments, namely the set of truth assignments τ that make the formula true. We will often find two formulas that are composed of different symbols but represent the same set of truth assignments; this motivates the following definition:

Definition 2. Two Boolean formulas are called *equivalent* if they have the same value under every possible truth assignment. That is, we say φ and ψ are equivalent (denoted $\varphi \equiv \psi$) if for all $\tau : \text{Vars} \to \{0,1\}$ we have that $\varphi(\tau) = \psi(\tau)$.

Example 3. $\varphi_1 = ((x_1 \to x_2) \land x_1)$ is equivalent to $\varphi_2 = (x_1 \land x_2)$.

1.4 Truth tables

Another way to see the evaluation of the Boolean formula is by truth tables. A truth table is a table in which we see the evaluation of a formula under all possible truth assignments.

Example 4. The truth table for $\varphi = ((x_1 \land (\neg x_2)) \to ((\neg x_3) \land x_2))$ is:

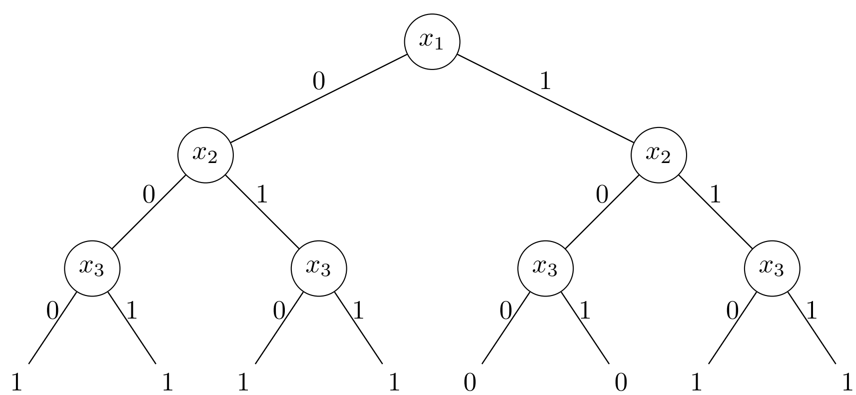
x_1	x_2	x_3	φ
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Notice that the size of the truth table is exponential in the number of variables. When we want to reason about larger Boolean formulas a truth table will quickly become cumbersome.

1.5 Binary Decision Tree

Another way to describe a Boolean formula is by a Binary Decision Tree. This is a binary tree in which every layer represents a fresh variable and every node has two children: Left (to set the variable to 0) and Right (to set the variable to 1). Then each path in the tree represents a possible assignment for the variables; the corresponding evaluation of the formula is the leaf at the end of the path.

Example 5. The truth table given above for $\varphi = ((x_1 \land (\neg x_2)) \to ((\neg x_3) \land x_2))$ becomes:



Note that this description of a Boolean formula is also exponential in the number of variables.

1.6 Satisfiability, Unsatisfiability, and Validity

Definition 3. Let φ be a Boolean formula.

- φ is called *satisfiable* if it is **true** under at least one truth assignment of the variables, i.e., if $\varphi(\tau) = 1$ for some truth assignment τ. We say that the assignment τ *satisfies* the formula.
- φ is called *unsatisfiable* if φ is **false** under every truth assignment, i.e., if $\varphi(\tau) = 0$ for all truth assignments τ.
- φ is called *valid* if it is **true** under every truth assignment, i.e., if $\varphi(\tau) = 1$ for all truth assignments τ.

Example 6. Let $\varphi = (x_1 \vee x_2)$. Then $\{x_1 \mapsto 1, x_2 \mapsto 0\}$ satisfies φ but $\{x_1 \mapsto 0, x_2 \mapsto 0\}$ does not satisfy φ. Thus φ is satisfiable but not valid.

Example 7. $\varphi = (x \vee \neg x)$ is valid.

1.7 CNF, *k*-CNF

We will shortly be interested in determining the complexity of checking satisfiability of a Boolean formula. It turns out that we do not always have to consider every possible Boolean formula. Instead, it is sufficient to consider only a certain subset of Boolean formulas. A common subset is known as CNF.

Definition 4. – A *literal* is either a variable (x) or the negation of a variable ($\neg x$).
- A [*CNF-*]*clause* is the disjunction (or) of literals.
- A Boolean formula is in *Conjunctive Normal Form* (*CNF*) if it is the conjunction of disjunctions of literals.

That is, φ is in CNF if

$$\varphi = \bigwedge(\bigvee \ell_i) = (\ell_1 \vee \ell_2 \vee \cdots \vee \ell_{a_1}) \wedge (\ell_{a_1+1} \vee \ell_{a_1+2} \vee \cdots \vee \ell_{a_2}) \wedge \cdots \wedge (\ell_{a_{b-1}+1} \vee \ell_{a_{b-1}+2} \vee \cdots \vee \ell_{a_b})$$

where each ℓ_i is a literal (either a variable or the negation of a variable). For a truth assignment to satisfy a CNF formula it must satisfy at least one literal from every clause. Note that the concatenation of two CNF formulas is still a CNF formula.

Example 8. $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee x_5 \vee x_4) \wedge (x_4 \vee \neg x_2)$ is in CNF.

It is often interesting to consider only CNF formulas with the same number of literals, k, in each clause. This defines a subset of CNF formulas known as k-CNF formulas:

Definition 5. Let $k \geq 2$ be an integer. A formula ϕ is in k-CNF if it is in CNF and there are exactly k literals in each clause.

Example 9. $(x_1 \vee \neg x_2 \vee x_3) \wedge (x_3 \vee x_5 \vee x_4) \wedge (x_4 \vee \neg x_2 \vee x_1)$ is in 3-CNF.

1.8 SAT, *k*-SAT

Checking the satisfiability of Boolean formulas is a famous important problem. It turns out that checking the satisfiability of general Boolean formulas is equivalent to checking the satisfiability of Boolean formulas in CNF, because of the following theorem:

Theorem 1. *There is a polynomial time algorithm that, given a Boolean formula φ, will construct a CNF formula which is satisfiable iff φ is satisfiable.*

Proof. The proof of this theorem will be given as a Homework Exercise

The formula constructed in the theorem above may introduce a linear fraction of new variables. It is also possible to construct a CNF formula equivalent (not just equi-satisfiable) to a given Boolean formula *without* introducing new variables, but the new formula may be exponentially larger in the worst-case.

We can define two decision problems, SAT and k-SAT, that check the satisfiability of a formula in either CNF or k-CNF:

SAT:
Input: a Boolean formula φ in CNF
Output: Yes if φ is satisfiable, No otherwise.

k-SAT:
Input: a Boolean formula φ in k-CNF
Output: Yes if φ is satisfiable, No otherwise.

We will see later that these two decision problems are closely related. In fact, both SAT and k-SAT (for $k \geq 3$) are NP-complete.

2 Exercises

2.1 Problem 1

Which of the following is a Boolean formula?

1. $(x_1 \neg x_2 \wedge x_3) \rightarrow (\neg(x_2)$
2. (x_2)
3. $(x_1 \ \vee x_2) \wedge \neg((x_2 \leftrightarrow (\neg x_3)))$

2.2 Problem 2

Prove the De-Morgan laws. That is, for every pair of Boolean formulas ψ_1, ψ_2 we have:

1. $(\neg(\psi_1 \vee \psi_2)) \equiv (\neg\psi_1 \wedge \neg\psi 2)$
2. $(\neg(\psi_1 \wedge \psi_2)) \equiv (\neg\psi_1 \vee \neg\psi 2)$
3. $\neg(\neg\psi_1) \equiv \psi_1$

2.3 Problem 3

Prove the distributive and associative properties. That is, for every pair of Boolean formulas ψ_1, ψ_2, ψ_3 we have:

1. $(\psi_1 \vee \psi_2) \vee \psi_3 = \psi_1 \vee (\psi_2 \vee \psi_3)$
2. $(\psi_1 \wedge \psi_2) \wedge \psi_3 = \psi_1 \wedge (\psi_2 \wedge \psi_3)$
3. $(\psi_1 \vee \psi_2) \wedge \psi_3 = (\psi_1 \wedge \psi_3) \vee (\psi_2 \wedge \psi_3)$
4. $(\psi_1 \wedge \psi_2) \vee \psi_3 = (\psi_1 \vee \psi_3) \wedge (\psi_2 \vee \psi_3)$
5. Is it also true that $(\psi_1 \wedge \psi_2) \vee \psi_3 = \psi_1 \wedge (\psi_2 \vee \psi_3)$?

2.4 Problem 4

Prove all of the following. Let φ be a Boolean formula:

1. φ is valid iff $\neg(\varphi)$ is not satisfiable.
2. φ is no valid iff $\neg(\varphi)$ is satisfiable.
3. φ is satisfiable iff $\neg(\varphi)$ is not valid.

2.5 Problem 5

Let $\varphi = (x_1 \wedge x_2) \rightarrow ((x_2 \vee \neg x_3) \wedge (x_1 \wedge x_3 \wedge \neg x_2))$.

1. Draw the truth table and Binary Decision Tree of φ.
2. Is φ satisfiable/valid/not-satisfiable? If satisfiable, which assignments satisfy φ?
3. Let $\psi \equiv (x_1 \wedge \neg x_2 \wedge x_3)$ Prove or refute that φ and ψ are equivalent.

2.6 Problem 6

We learned about formulas in CNF. A formula is in *Disjunctive Normal Form* (*DNF*) if if it is the disjunction of conjunctions of literals. For example, $(x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3 \wedge x_5 \wedge x_4) \vee (x_4 \wedge \neg x_2)$ is a formula in DNF. Prove the following:

1. φ is a DNF formula iff $\neg\varphi$ is a CNF formula.
2. $L = \{\varphi \mid \varphi \text{ is a DNF formula}\}$ is in $PTIME$.