Propositional Logic

Syntax

Let $x, x_1, \ldots, y, y_1, \ldots \in \mathcal{X}$, where $\mathcal{X}$ is a set of *propositional variables*. We assume $\mathcal{X}$ to be denumerable. In the following, we will use the symbol $\equiv$ to mean syntactic identity (i.e., *identical* letter for letter, symbol for symbol, node for node).

Definition: Propositions *Prop* are inductively defined, using the following abstract grammar:

$$p, p_1, p_2 \ldots \in Prop ::= \mathrm{T} \,|\, \mathrm{F} \,|\, x \,|\, \neg p_1 \,|\, p_1 \wedge p_2 \,|\, p_1 \vee p_2 \,|\, p_1 \rightarrow p_2 \,|\, p_1 \leftrightarrow p_2$$

Coding propositions in OCaml: Define a data type!

```
type prop = T | F   (* propositional constants *)
    | Atom of string (* propositional variables *)
    | Not of prop    (* negation *)
    | And of prop * prop  (* conjunction *)
    | Or of prop * prop    (* disjunction *)
    | Imply of prop * prop   (* Implication *)
    | Iff of prop * prop (* Bi-implication or Iff or
equivalence *)
;;
```

Exercise: Write a recursive function `ht: prop -> int`, which returns the height of a proposition (seen as a syntax tree).

Exercise: Write a recursive function `size: prop -> int`, which returns the number of nodes in a proposition (seen as a syntax tree).

Exercise: Write a recursive function `atoms: prop -> string set`, which returns the *set* of propositional variables that appear in a proposition (seen as a syntax tree).

Definition: A proposition p is in *negation normal form* (NNF) if in its syntax tree, no connectives appear below a negation symbol $(\neg)$, i.e., only propositional variables can appear below a negation, and not even constants T and F or another negation symbol $(\neg)$.

Semantics

Definition: A truth assignment is a (total function) $\rho \in [\mathcal{X} \rightarrow \mathbb{B}]$, from propositional variables to the booleans.

The meaning of a proposition $p \in Prop$, is given with respect to a truth assignment $\rho \in [\mathcal{X} \rightarrow \mathbb{B}]$. We can directly define a semantics-defining function $truth[\![p]\!]\rho$ as follows:

$$
\begin{aligned}
truth[\![\mathrm{T}]\!]\rho &= true \\
truth[\![\mathrm{F}]\!]\rho &= false \\
truth[\![x]\!]\rho &= \rho(x) \\
truth[\![\neg p_1]\!]\rho &= not\ (truth[\![p_1]\!]\rho) \\
truth[\![p_1 \wedge p_2]\!]\rho &= (truth[\![p_1]\!]\rho)\ and\ (truth[\![p_2]\!]\rho) \\
truth[\![p_1 \vee p_2]\!]\rho &= (truth[\![p_1]\!]\rho)\ or\ (truth[\![p_2]\!]\rho) \\
truth[\![p_1 \to p_2]\!]\rho &= (not\ (truth[\![p_1]\!]\rho))\ or\ (truth[\![p_2]\!]\rho) \\
truth[\![p_1 \leftrightarrow p_2]\!]\rho &= (truth[\![p_1]\!]\rho)\ iff\ (truth[\![p_2]\!]\rho)
\end{aligned}
$$

It can be seen as the unique homomorphic extension of $\rho \in [\mathcal{X} \to \mathbb{B}]$ to $\hat{\rho} \in [Prop \to \mathbb{B}]$ where the propositional constants and connectives are interpreted as their corresponding logical functions on the booleans (under the standard interpretation of those symbols, given by the truth tables).

Exercise: Write a recursive function `truth: prop -> (string -> bool) -> bool`, which evaluates a proposition with respect to a given truth assignment to the propositional variables.

Definition:
A proposition p is called a tautology if for *every* truth assignment $\rho \in [\mathcal{X} \to \mathbb{B}]$, $truth[\![p]\!]\rho = true$.
A proposition p is called a contradiction if for *every* truth assignment $\rho \in [\mathcal{X} \to \mathbb{B}]$, $truth[\![p]\!]\rho = false$.
Otherwise, if for some truth assignments $\rho \in [\mathcal{X} \to \mathbb{B}]$, $truth[\![p]\!]\rho = true$, and for other truth assignments $\rho \in [\mathcal{X} \to \mathbb{B}]$, $truth[\![p]\!]\rho = false$, then p is called a contingency.
A truth assignment $\rho \in [\mathcal{X} \to \mathbb{B}]$ is said to satisfy p if $truth[\![p]\!]\rho = true$.
A truth assignment $\rho \in [\mathcal{X} \to \mathbb{B}]$ is said to falsify p if $truth[\![p]\!]\rho = false$.

Exercise If a proposition p has n distinct propositional variables, then what is the (worst-case) complexity of finding a truth assignment that satisfies p?

Exercise If a proposition p has n distinct propositional variables, then what is the (worst-case) complexity of determining if p is a tautology?

Definition A proposition p_1 is *logically equivalent* to another proposition p_2, written $p_1 \Leftrightarrow p_2$, if for *every* truth assignment $\rho \in [\mathcal{X} \to \mathbb{B}]$, $truth[\![p_1]\!]\rho = truth[\![p_2]\!]\rho$.
That is, *every* way of assigning truth values to the propositional variables makes p_1 and p_2 get the same truth value, *i.e.*, each truth assignment $\rho \in [\mathcal{X} \to \mathbb{B}]$ either satisfies both p_1 and p_2 or falsifies both p_1 and p_2.

Exercise*: Show that in all the equational laws of Boolean Logic the two propositions are logically equivalent.

Exercise: Write a recursive function `nnf: prop -> prop`, which converts a given proposition to a logically equivalent proposition in NNF. *Prove that your* *program preserves the truth of the input proposition with respect to any truth assignment.*

Theorem (Expressive complete subset of connectives) For any proposition p, there exists a logically equivalent proposition that uses only the propositional variables, and the set of connectives $\{\mathtt{F}, \neg, \wedge, \vee\}$.

That is, all boolean functions on $n \geq 0$ boolean variables can be expressed using NOT, AND and OR gates.

Proof Hint: How do you construct a Sum of Products form?

Definition: A proposition p using only propositional variables and the connectives $\{\mathtt{F}, \neg, \wedge, \vee\}$ is in *disjunctive normal form* (DNF) if (i) it is in NNF and (ii) in its syntax tree, no disjunction symbol $(\vee)$ appears below a conjunction symbol $(\wedge)$.

Exercise: Write a recursive function `dnf: prop -> prop,` which converts a given proposition to a logically equivalent proposition in DNF (also called Sum of Products or SoP form). *Prove that your program preserves the truth of the input proposition with respect to any truth assignment.*

Exercise: Show that the set of connectives $\{\mathtt{T}, \neg, \rightarrow\}$ is expressively complete.

Exercise: Show that there is an expressively complete singleton set of connectives.

Definition: A proposition p using only propositional variables and the connectives $\{\mathtt{F}, \neg, \wedge, \vee\}$ is in *conjunctive normal form* (CNF) if (i) it is in NNF and (ii) in its syntax tree, no conjunction symbol $(\wedge)$ appears below a disjunction symbol $(\vee)$.

Exercise: Write a recursive function `cnf: prop -> prop,` which converts a given proposition to a logically equivalent proposition in CNF (also called Product of Sums or PoS form). *Prove that your program preserves the truth of the input proposition with respect to any truth assignment.*

Definition A proposition p_2 is a *logical consequence* of another proposition p_1, written $p_1 \Rightarrow p_2$, if for *every* truth assignment $\rho \in [\mathscr{X} \rightarrow \mathbb{B}]$, *whenever* $truth[\![p_1]\!]\rho = true$, then $truth[\![p_2]\!]\rho = true$ as well. That is, *every* way of assigning truth values to the propositional variables that satisfies p_1 also satisfies p_2.

Definition The definition of logical consequence generalises in the following way. Let Γ be a set of propositions. A proposition p is a *logical consequence* of the set of propositions Γ, written $\Gamma \vDash p$, *if* for *every* truth assignment $\rho \in [\mathscr{X} \rightarrow \mathbb{B}]$ such that for each proposition $p_i \in \Gamma$, $truth[\![p_i]\!]\rho = true$, *then* $truth[\![p]\!]\rho = true$ as well. That is, *every* way of assigning truth values to the propositional variables that satisfies *each* proposition in Γ also satisfies p.

Definition A <u>*model of a proposition*</u> p is the *set* of truth assignments $\rho \in [\mathscr{X} \to \mathbb{B}]$ which make p true. That is, $\mathscr{M}(p) = \{\rho \mid truth[\![p]\!]\rho = true\}$.

Exercise: What is $\mathscr{M}(p)$ if p is a tautology? if p is a contradiction?

Definition: The <u>*model of a set of propositions*</u> Γ is the set of truth assignments which make each $p' \in \Gamma$ true. That is, $\mathscr{M}(\Gamma) = \bigcap_{p' \in \Gamma} \{\rho \mid truth[\![p']\!]\rho = true\}$.

Exercise: Show that $p_1 \Leftrightarrow p_2$ if and only if $\mathscr{M}(p_1) = \mathscr{M}(p_2)$.

Exercise: Show that $p_1 \Rightarrow p_2$ if and only if $\mathscr{M}(p_1) \subseteq \mathscr{M}(p_2)$.

Exercise: Show that $\Gamma \models p$ if and only if $\mathscr{M}(\Gamma) \subseteq \mathscr{M}(p)$.

Substitution

Consider a *syntactic* operation of <u>substituting</u> a proposition q <u>for</u> each occurrence of a propositional variable x in proposition p. That is, replace *each* leaf labelled x in the syntax tree of p by the tree corresponding to q. We shall write $p[q/x]$ to denote this operation. Note that the propositional variable x can appear in the resultant tree $p[q/x]$, e.g., if x appears in q.

Definition $p[q/x]$ is defined as follows

- If $p \equiv x$, then $p[q/x] \equiv q$
- If $p \equiv y \ (y \neq x)$, then $p[q/x] \equiv p$
- If $p \equiv \mathtt{T}$ or $p \equiv \mathtt{F}$, then $p[q/x] \equiv p$
- If $p \equiv \neg p_1$, then $p[q/x] \equiv \neg(p_1[q/x])$
- If $p \equiv p_1 \wedge p_2$, then $p[q/x] \equiv (p_1[q/x])) \wedge (p_2[q/x]))$
- If $p \equiv p_1 \vee p_2$, then $p[q/x] \equiv (p_1[q/x])) \vee (p_2[q/x]))$
- If $p \equiv p_1 \to p_2$, then $p[q/x] \equiv (p_1[q/x])) \to (p_2[q/x]))$
- If $p \equiv p_1 \leftrightarrow p_2$, then $p[q/x] \equiv (p_1[q/x])) \leftrightarrow (p_2[q/x]))$

Exercise: Show that $p[q/x]$ is the Unique Homomorphic Extension of the function in $\mathscr{X} \to Prop$ which replaces variable x by proposition q and leaves *every* other variable unchanged.

Exercise: Show that $ht(p[q/x]) \leq ht(p) + ht(q) - 1$.

Exercise: Generalise the definition of substitution to $p[q_1, \ldots, q_n/x_1, \ldots, x_n]$ which denotes the <u>simultaneous substitution</u> of n propositions $q_1, \ldots, q_n$ respectively for n distinct propositional variables $x_1, \ldots, x_n$ that may appear in a given proposition p.

Exercise: Write a recursive function `subst: prop -> (string -> prop) -> prop`, which applies a given (simultaneous) substitution of propositions for corresponding propositional variables in a given proposition. [Hint: Follow the structure of *truth*.]

Substitution Lemma: Suppose p, q are propositions and x is a propositional variable, and let $\rho \in [X \to \mathbb{B}]$ be any truth assignment. Let $\rho[x \mapsto b] \in [\mathcal{X} \to \mathbb{B}]$ denote the truth assignment that is identical to ρ at all propositional variables except at x, where it is assigned boolean value b. Suppose $truth[\![q]\!]\rho = b'$.
Then $truth[\![p[q/x]]\!]\rho = truth[\![p]\!](\rho[x \mapsto b'])$

That is, computing the truth value of the proposition $p[q/x]$ with respect to truth assignment ρ is the same as computing the truth value of the proposition p, using the truth assignment that is identical to ρ at all propositional variables except at x, where we plug in the value of q with respect to ρ.

Proof is by induction on the structure (or height) of p, and uses the definitions of *truth* and substitution.
Base cases (*ht* p = 0)
There are three sub-cases
- $p \equiv x$. So $truth[\![p[q/x]]\!]\rho = truth[\![q]\!]\rho = b' = truth[\![x]\!](\rho[x \mapsto b'])$
- $p \equiv y$ $(y \neq x)$. So $truth[\![p[q/x]]\!]\rho = truth[\![y]\!]\rho = truth[\![y]\!](\rho[x \mapsto b'])$
- $p \equiv \mathtt{T}$ or $p \equiv \mathtt{F}$. So $truth[\![p[q/x]]\!]\rho = truth[\![p]\!]\rho = truth[\![p]\!](\rho[x \mapsto b'])$

Induction Hypothesis: Assume that for all $p' \in Prop$ such that $ht(p')) \leq n$
$truth[\![p'[q/x]]\!]\rho = truth[\![p']\!](\rho[x \mapsto b'])$

Induction Step (*ht* p = n+1)
- $p \equiv \neg p_1$. So $truth[\![p[q/x]]\!]\rho = truth[\![\neg p_1[q/x]]\!]\rho = not(truth[\![p_1[q/x]]\!]\rho) = not(truth[\![p_1]\!](\rho[x \mapsto b'])) = truth[\![\neg p_1]\!](\rho[x \mapsto b'])$.
- $p \equiv p_1 \wedge p_2$. So $truth[\![p[q/x]]\!]\rho = truth[\![(p_1 \wedge p_2)[q/x]]\!]\rho =$
$(truth[\![p_1[q/x]]\!]\rho)$ and $(truth[\![p_2[q/x]]\!]\rho) =$
$(truth[\![p_1]\!](\rho[x \mapsto b']))$ and $(truth[\![p_2]\!](\rho[x \mapsto b'])) =$
$truth[\![p_1 \wedge p_2]\!](\rho[x \mapsto b'])$.
- Other sub-cases are left as exercises.

Exercise: Complete the proof of the Substitution Lemma for the following cases:
- $p \equiv p_1 \vee p_2$
- $p \equiv p_1 \to p_2$
- $p \equiv p_1 \leftrightarrow p_2$