Propositional Logic

Definition 1 *Let $X = \{x_1, x_2, \dots\}$ be a countably infinite set of propositional variables. Formulae of propositional logic are inductively defined as follows:*
- true *and* false *are formulae.*
- *Every propositional variable x_i is a formula.*
- *For every formula F, $\neg F$ is a formula.*
- *For all formulae F and G, $F \wedge G$ and $F \vee G$ are formulae.*

We derive the further connectives of implication, bi-implication, xor, etc. in precisely the expected way. Further, we note the set of formulae on variables X is $\mathcal{F}(X)$.

Definition 2 *An assignment is a function $\mathcal{A} : X \to \{0,1\}$ that induces an assignment $\mathcal{A} : \mathcal{F}(X) \to \{0,1\}$ as follows:*
- $\mathcal{A}(\text{false}) = 0$, $\mathcal{A}(true) = 1$.
- *For every $x \in X$, $\mathcal{A}(x) := \mathcal{A}(x)$.*
- $\mathcal{A}(\neg F) := 1 - \mathcal{A}(F)$.
- *...*

Definition 3 *Let $F \in \mathcal{F}(X)$ and $\mathcal{A} : X \to \{0,1\}$ be an assignment. If $\mathcal{A}(F) = 1$ then we write $\mathcal{A} \vDash F$. If F has at least one model then it is satisfiable, otherwise it is unsatisfiable.*

A formula G is entailed by a set of formulae $\mathcal{S}$ if every assignment that satisfies each formula in $\mathcal{F}$ also satisfies G. We write this $\mathcal{S} \vDash G$. Further we say that two formulae are logically equivalent if $\mathcal{A}(F) = \mathcal{A}(G)$ for every assignment $\mathcal{A}$. We write this $F \equiv G$.

A formula is in conjunctive normal form if it is the conjunction of disjunctions of literals. It is in disjunctive normal form if it is a disjunction of conjunctions of literals. Every formula is equivalent to one in CNF or DNF.

By convention we say that true is CNF without any clauses, and false is DNF with no clauses.

We can use an algorithm known as a SAT solver to determine the set of variable assignments for which a formula is assigned truth. As an example of the use of this, take a graph $G = (V, E)$, and we want to find a hamiltonian path (that is, one which goes through every vertex without intersecting). Where x_{ij} represents whether the vertex from i to j is included in the path, we can represent this problem by the below formula:

$$F_G := \bigwedge_{i=1}^{n} \bigvee_{j=1}^{n} x_{ij} \wedge \bigwedge_{j=1}^{n} \bigvee_{i=1}^{n} x_{ij} \wedge \bigwedge_{(i,i') \notin E} \bigwedge_{j=1}^{n-1} \neg(x_{ij} \wedge x_{i'(j+1)})$$

Check this again - this formula isn't quite complete.

Where n is the length of the formula, we can solve SAT in $O(2^n)$, but no sub-exponential algorithm is known. We can do better for special formula classes however.

Definition 4 *A CNF forula is a Horn formula if each clause contains at most one positive literal.*

This form allows them to be written as conjunctions of implications. Consequently, we can use an algorithm that begins by setting every variable to 0, then while the assignment doesn't satisfy the formula we look through the unsatisfied clauses. If the consequent is a variable then assign it true, and if it is false then return that the formula is unsatisfiable. The argument here is that for $P \to$ false to be unsatisfied then an element of P has been made true. Provided we only make things true once we're sure they have to be, we know we have unsatisfiability. In a similar way, if $P \to p$ is unsatisfied, then p is false and an element of P is true, so provided that element had to be made true, p must also be true. Thus we have that each step is logically valid.

Definition 5 *A 2-CNF formula, or Krom formula is a CNF formula F such that every clause has at most two literals.*

We write an implication graph $G = (V, E)$ where $V := \{x_1, x_2, \dots\} \cup \{\neg x_1, \neg x_2, \dots\}$, and E is defined using $a \vee b \equiv \neg a \to b$ to allow us to assign an edge between two variables where there is such an implication. Consequently we can say that a 2-CNF formula is satisfiable iff its implication graph has no $p \in X$ such that there is a path from p to $\neg p$ and $\neg p$ to p

In the forwards direction this is obvious, because if there was such a path then we get a contradiction. In the reverse, suppose the graph is consistent. We can get an assignment by looping over the graph vertices. Pick an unassigned variable with no path from itself to its complement, set it to 1, **finish proof**.

Theorem 1 *Given an arbitrary formula F, we can compute an equisatisfiable 3-CNF formula G in polynomial time.*

The proof of this is via taking subformulas $F_i = p_i$, for $i \in \{1, \dots, m\}$, and the rest of the subformulas $F_{m+1}, \dots, F_n$. Introduce new variables $p_{m+1}, \dots, p_n$, and if $F_i = F_j \vee F_k$, then $G_i := p_i \leftrightarrow (p \vee p_k)$. Then take $G = G_{m+1} \wedge \dots \wedge G_n \wedge p_n$.

Walk-SAT is a SAT algorithm which works randomly on CNF formulae. We input a CNF formula with n variables and a repetition parameter r. Pick a random assignment of the n variables, and r times we pick an unsatisfied clause, flip a randomly selected literal, and at each point check if F is now satisfied.

We find that we have a bound that the probability of failure is $\leq 2^{-m}$ where $r = 2mn^2$, so setting m reasonably high gives a good bound on the rate of error.

Resolution

Resolution is a proof calculus for propositional logic. It is sound and complete, so anything proved is valid, and anything valid can be proved.

Definition 6 *Let C_1 and C_2 be clauses. A clause R is called a resolvent of C_1 and C_2 if there are complementary literals $L \in C_1$ and $\overline{L} \in C_2$ such that*

$$R = (C_1 \setminus \{L\}) \cup (C_2 \setminus \{\overline{L}\}).$$

The main point of this is that $C_1 \wedge C_2 \vDash R$. Consequently this allows us to reduce CNF formulae, and either refute (if this reduces them to the empty clause) or find a satisfaction.

A derivation of a clause C from a set of clauses F is a sequence $C_1, C_2, \dots, C_m$ of clauses where $C_m = C$ and for each $i \in \{1, 2, \dots, m\}$ either $C_i \in F$ or C_i is a resolvent of C_j, C_k for some $j, k < i$. A derivation of the empty clause $\square$ from a formula F is called a refutation of F.

Lemma 2 *Let F be a CNF formula represented as a set of clauses. If R is a resolvent of clauses C_1 and C_2 of F then $F \equiv F \cup \{R\}$.*

Let F be a CNF formula represented as a set of clauses. If R is a resolvent of clauses C_1 and C_2 of F, then $F \equiv F \cup \{R\}$. Either $\mathcal{A} \vDash L$ or $\mathcal{A} \vDash$. In the former case $\mathcal{A} \vDash C_2$ so $\mathcal{A} \vDash C_2 \setminus \{\overline{L}\}$, in the latter case $\mathcal{A} \vDash C_1$ so $\mathcal{A} \vDash C_2 \setminus \{L\}$ and thus in both cases $\mathcal{A} \vDash R$.

Thus we get both soundness, that if there is a derivation of $\square$ from F then F is unsatisfiable, and completeness, that if F is unsatisfiable there is a derivation of $\square$ from F. The former proof is from repeatedly applying the resolution lemma, and the latter via induction.

The DPLL algorithm

The DPLL algorithm combines search and deduction to decide satisfiability of CNF-formulae. The idea is to use a depth-first search – at every unsuccessful leaf of the search tree (called a conflict), use resolution to compute a conflict clause. Add this clause to the formula we're deciding about.

We initiaise $\mathcal{A}$ as the empty assignment. While there is a unit clause $\{L\}$ in $F\big|_{\mathcal{A}}$, update $\mathcal{A} \mapsto \mathcal{A}_{[L \mapsto 1]}$. If $F\big|_{\mathcal{A}}$ contains no clauses, stop and output $\mathcal{A}$. If this contains empty, determine a new clause C to add to F by a learning procedure. If this is the empty clause, then F is unsatisfiable – otherwise backtrack to the highest level where C is the unit clause and loop.

Lower Bounds for Resolution

Proof of the pigeonhole principle:

Fix $n \in \mathbb{N}$ and for $i, j \in \{1, \dots, n\}$ let x_{ij} denote that pigeon i is in box j. Constructing formulas as follows:

$$P_i := \bigvee_{i=1}^{n-1} x_{ij}$$

$$F_1 = \bigwedge_{j=1}^{n-1} \bigwedge_{1 \leq i < i' \leq n} (\neg x_{ij} \vee \neg x_{i'j})$$

$$F_2 := \bigwedge_{j=1}^{n-1} \bigvee_{i=1}^{n} x_{ij}$$

$$F_3 = \bigwedge_{1 \leq j < j' \leq n-1} (\neg x_{ij} \vee \neg x_{ij'})$$

These combined state that each pigeon is in a single box, every box has a pigeon, and no box contains two different pigeons. Note that there are n pigeons for $n-1$ boxes.

Define $\text{CRIT}_n = F_1 \wedge F_2 \wedge F_3$. A valuation satisfying CRIT_n is said to be critical. The pigeonhole principle is equivalent to the assertion that $F_1 \wedge \bigwedge_{i=1}^{n} P_i$ is unsatisfiable.

A pseudo refutation of $\text{PHP}_n := \text{CRIT}_n \wedge \bigwedge_{i=1}^{n} P_i$ is a sequence of monotone clauses (clauses with only positive literals) $C_1, \dots, C_m = \square$ such that for all $1 \leq i \leq m$ either $\text{CRIT}_n \wedge P_j \vDash C_i$ for some $j \in \{1, \dots, n\}$, or $\text{CRIT}_n \wedge C_j \wedge C_k \vDash C_i$ for some $j, k < i$.

Proof incomplete

Compactness

Theorem 3 *A set of formulas $\mathcal{S}$ is satisfiable if and only if every finite subset of $\mathcal{S}$ is satisfiable.*

A partial assignment is a function $\mathcal{A} : D \to \{0,1\}$, whose domain $D \subseteq \{p_1, p_2, \dots\}$ is a set of variables $\text{dom}(\mathcal{A})$. The central idea of this proof is to construct a sequence of good partial assignments $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2, \dots$ such that $\text{dom}(\mathcal{A}_n) = \{p_1, \dots, p_n\}$ and $\mathcal{A}_{n+1}$ extends $\mathcal{A}_n$ for each n. We maintain the induction hypothesis that there are infinitely many good partial assignments that extend $\mathcal{A}_n$.

Suppose we have $\mathcal{A}_0, \dots, \mathcal{A}_n$ such that $\mathcal{A}_n$ has infinitely many good extensions. We only have two extensions which include p_{n+1}, and so one of them must have infinitely many good extensions, and we can just choose that one.

First-Order Predicate Logic

First-Order logic introduces the notion of quantifiers, as well as both predicate symbols, and constant symbols.

Syntactically, we have a signature σ consisting of constant symbols $c_0, c_1, c_2, \dots$, function symbols $f_1, f_2, \dots$, and predicate symbols $p_1, p_2, \dots$. Every variable x or constant symbol is a σ-term, and further if $t_1, \dots, t_k$ are σ-terms and f is a k-ary function symbol then $f(t_1, \dots, t_k)$ is a σ-term.

Next, if P is a k-ary predicate symbol and $t_1, \dots, t_k$ are σ-terms, then $P(t_1, \dots, t_k)$ is a formula. For each formula F, $\neg F$, and for another formula G, $(F \vee G)$ and $(F \wedge G)$ are both formulas. Further, if x is a variable and F is a formula then $\exists x\, F$ and $\forall x\, F$ are both formulas.

A σ-structure $\mathcal{A}$ consists of a non-empty universe $U_{\mathcal{A}}$, where for each constant c, variable x there are elements $c_{\mathcal{A}} \in U_{\mathcal{A}}$, $x_{\mathcal{A}} \in U_{\mathcal{A}}$. Additionally for every k-ary predicate symbol P in σ, there is a k-ary relation $P_{\mathcal{A}} \subseteq U_{\mathcal{A}}^k$, and for every k-ary function symbol f in σ, there is a k-ary function $f_{\mathcal{A}} : U_{\mathcal{A}}^k \to U_{\mathcal{A}}$.

We can then define $\mathcal{A}[t]$ inductively where $\mathcal{A}[c] = c_{\mathcal{A}}$, $\mathcal{A}[x] = x_{\mathcal{A}}$, $\mathcal{A}[f(t_1, \dots, t_k)] = f_{\mathcal{A}}(\mathcal{A}[t_1], \dots, \mathcal{A}[t_k])$.

A formula F is satisfiable if there is some structure $\mathcal{A}$ such that $\mathcal{A} \vDash F$. It is valid if for every $\mathcal{A}$, $\mathcal{A} \vDash F$.

Theorem 4 *Satisfiability and validity are undecidable, but validity is semi-decidable (RE).*

Lemma 5 *Suppose that $\mathcal{A}$ and $\mathcal{A}'$ are σ-assignments with the same universe and identical interpretations of the predicate, function, and constant symbols in σ. If $\mathcal{A}$ and $\mathcal{A}'$ give the same interpretation to each variable occurring free in some σ-formula F then $\mathcal{A} \vDash F$ if and only if $\mathcal{A}' \vDash F$.*

This follows by two inductions.

First-Order normal forms

A formula is rectified if no variable occurs both bound and free. We have that every formula is equivalent to a rectified formula in prenex form, and we say that a formula in RPF is in Skolem form if it doesn't contain any occurrences of $\exists$.

We say that a σ-structure $\mathcal{H}$ is called a Herbrand structure if the universe $U_{\mathcal{H}}$ is the set of ground terms (terms without a variable) over σ, for every constant symbol c we have $c_{\mathcal{H}} = c$, and for every k-ary function symbol f in σ and for all ground terms $t_1, t_2, \dots, t_n \in U_{\mathcal{H}}$ we have $f_{\mathcal{H}}(t_1, \dots, t_k) = f(t_1, \dots, t_k)$.

The intuition for what this changes is that whereas normally we have a σ-structure $\mathcal{A}$ which maps variables and constants to elements of $U_{\mathcal{A}}$, function symbols to functions, and predicate symbols to predicates, a Herbrand structure is a specific structure wherein the universe is every possible sequence of applications of function symbols to constants.

Theorem 6 (Herbrand's theorem) *Let $F = \forall x_1 \dots \forall x_n F^*$ be a closed formula in skolem form. Then F is satisfiable iff it has a Herbrand model.*

If F has a Herbrand model, immediately F is satisfiable by definition, as $\mathcal{H}$ models F. In reverse, if F is satisfied by some $\mathcal{A}$, we construct $\mathcal{H}$ to mimic $\mathcal{A}$. Given a k-ary P we defined $(t_1, \dots, t_k) \in P_{\mathcal{H}}$ iff $\mathcal{A} \vDash P(t_1, \dots, t_k)$. By inducting on the number of quantifiers we can then show that for any closed formula $G = \forall y_1 \dots \forall y_n G^*$, if $\mathcal{A} \vDash G$ then $\mathcal{H} \vDash G$.

In general, a satisfiable formula may not have a finite model. We would like to prove that if F is unsatisfiable, then there is always a ground resolution proof of $\square$ from F. Fix a signature σ, and let $F = \forall x_1 \dots \forall x_n F^*$ be a closed formula in Skolem form with matrix F^*. The Herbrand expansion is defined as

$$E(F) := \{F^*[t_1/x_1] \dots [t_n/x_n] \,:\, t_1, \dots, t_n \text{ ground } \sigma\text{-terms } \}$$

which is the set of ways to substitute ground terms into the matrix F^*. $E(F)$ has a Herbrand model iff it is satisfiable as a propositional formula, allowing us to say that F is satisfiable iff $E(F)$ is satisfiable propositionally.

Thus we can apply resolution and compactness to get that a closed formula F is unsatisfiable iff there is a resolution refutation of $E(F)$.

Ground resolution is where we replace each variable by a ground symbol in order to apply normal resolution to infer statements. This requires that one has an intuition for the ground substitution that you use (e.g. we prove $1 + 1 \in \mathbb{N}$ by knowing beforehand that it is 2), and so the predicate version of resolution is slightly different.

A substitution *unifies* or solves a system of term equations $\{s_1 \stackrel{?}{=} t_1, \dots, s_n \stackrel{?}{=} t_n\}$ if $s_1\theta = t_1\theta, \dots, s_n\theta = t_n\theta$. We say that θ is a most general unifier if for any other unifier θ_1 there exists θ_2 such that $\theta_1 = \theta\theta_2$. A system is solved if it has the form $\{x_1 \stackrel{?}{=} t_1, \dots, x_n \stackrel{?}{=} t_n\}$ such that the x_i are distinct variables that do not appear in any term t_j. In this case the substitution $[t_1/x_1] \dots [t_n/x_n]$ is an mgu.

The unification rewriting rule is as follows: we can simplify $\{t \stackrel{?}{=} t\}$, decompose $\{f(s_1, \dots, s_n) \stackrel{?}{=} g(t_1, \dots, t_n)\}$ to $\{s_1 \stackrel{?}{=} t_1, \dots, s_n \stackrel{?}{=} t_n\}$ if $f = g$, to $\perp$ otherwise, $\{x \stackrel{?}{=} t\} \cup S$ to $\{x \stackrel{?}{=} t\} \cup S[t/x]$ if x occurs in S but not t, and $\{x \stackrel{?}{=} t\} \cup S$ to $\perp$ if x is a proper subterm of t.

As an example of the application of unification, take the following statements:
1. $\forall x\,.\, A(x, 0, x)$
2. $\forall x\,.\, \forall y\,.\, \forall z\,.\, (A(x, y, z) \to A(x, s(y), s(z)))$
3. $\forall x\,.\, \neg A(s(0), s(0), x)$

intending to use the first two to prove the contradiction of the third. We are then motivated to have, for $A(x_1, 0, x_1)$, $\neg A(x_2, y_2, z_2) \vee A(x_2, s(y_2), s(z_2))$...

Expressiveness

Definition 7 (Eherenfeucht-Fraïssé game) *Let $k, m \in \mathbb{N}$, σ be a finite relational signature, $\mathcal{A}, \mathcal{B}$ σ-structures, $a \in A^m$, $b \in B^m$, with the game $G_k((\mathcal{A}, a), (\mathcal{B}, b))$ a k-round game. In each round the spoiler chooses either $a \in A$ or $b \in B$, and the duplicator chooses an element of the other structure.*