Part IB Revision Notes
Logic And Proof

Michael Smith

April 11, 2004

1 Definitions

- An **interpretation** is a specific assignment of metavariables to values.
- A **valid** statement is satisfied by all possible interpretations.
- A **consistent** or **satisfiable** set of statements are all satisfied by some interpretation.
- If a set of statements S **entails** another statement A ($S \models A$), then any satisfying intrepretation of S also satisfies A.

2 Propositional Logic

- An interpretation is a function from the propositional letters (atomics) in a statement to $\{t, f\}$. If an interpretation I satisfies a formula A (i.e. it evaluates to t), then $\models_I A$.
- Implication: $A \models B$ iff $\models A \to B$ ($\models \neg A \vee B$).
- Equivalence: $A \simeq B$ iff $\models A \leftrightarrow B$ ($A \models B$ and $B \models A$).
- Converting to **negation normal form** requires you to firstly get rid of all implications (using $A \to B \simeq \neg A \vee B$ etc.), and then to push in all the negations according to de Morgan's laws.
- Converting to **conjunctive normal form** requires conversion to NNF, then the pushing in of all disjunctions using the distributive laws ($A \vee (B \wedge C) \simeq (A \vee B) \wedge (A \vee C)$). You can then simplify using:
  - Delete any disjunction containing both P and $\neg P$.
  - Delete any disjunction that contains another.
  - Use $(P \vee A) \wedge (\neg P \vee A) \simeq A$.
- A tautology will reduce to t under CNF, otherwise a falsifying truth assignment will be visible by inspection. Similarly, a contradiction will reduce to f under DNF (disjunctive normal form).
- A Hilbert style proof system works on the basis of a number of axioms, from which you may deduce other tautologies using inference rules. If we can deduce A from the assumptions S then $S \vdash A$. A proof system may have the properties:

  - Soundness (everything you can prove is valid) – $S \vdash A \rightarrow S \models A$.
  - Completeness (everything valid can be proven) – $S \models A \rightarrow S \vdash A$.
  - Deduction – $S \cup \{A\} \vdash B \rightarrow S \vdash A \rightarrow B$.

- The **sequent calculus** follows a Gentzen style natural deduction system by defining independant rules for each logical connective. A sequent is of the form $A_1, \ldots, A_m \Rightarrow B_1, \ldots, B_n$, whereby if all the assumptions A are true, then one of the goals B must be true.

  - Basic sequent: $\dfrac{}{A, \Gamma \Rightarrow A, \Delta}$ (BASIC)
  - Using a lemma: $\dfrac{\Gamma \Rightarrow \Delta, A \qquad \Gamma, A \Rightarrow \Delta}{\Gamma \Rightarrow \Delta}$ (CUT)
  - Not rules: $\dfrac{\Gamma \Rightarrow \Delta, A}{\neg A, \Gamma \Rightarrow \Delta} (\neg l) \qquad \dfrac{A, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, \neg A} (\neg r)$
  - And rules: $\dfrac{A, B, \Gamma \Rightarrow \Delta}{A \wedge B, \Gamma \Rightarrow \Delta} (\wedge l) \qquad \dfrac{\Gamma \Rightarrow \Delta, A \qquad \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \wedge B} (\wedge r)$
  - Or rules: $\dfrac{A, \Gamma \Rightarrow \Delta \qquad B, \Gamma \Rightarrow \Delta}{A \vee B, \Gamma \Rightarrow \Delta} (\vee l) \qquad \dfrac{\Gamma \Rightarrow \Delta, A, B}{\Gamma \Rightarrow \Delta, A \vee B} (\vee r)$
  - Implication rules: $\dfrac{\Gamma \Rightarrow \Delta, A \qquad B, \Gamma \Rightarrow \Delta}{A \rightarrow B, \Gamma \Rightarrow \Delta} (\rightarrow l) \qquad \dfrac{A, \Gamma \Rightarrow \Delta, B}{\Gamma \Rightarrow \Delta, A \rightarrow B} (\rightarrow r)$

3 OBDDs

- A **decision diagram** is a binary tree, with each node representing a variable, and the two branches from each node signifying a true (solid line) or false (dotted line) assignment. Each depth level in the tree corresponds to one boolean variable, and the leaves are the values of the whole expression (0/1).

- An **ordered binary decision diagram** should contain no duplicate trees (e.g. there should only be two leaves), and no redundant tests (i.e. if both branches from a node lead to the same node).

- To produce an OBDD in canonical form efficiently, build up recursively, combining two OBDDs on the basis of an operation. If Z and Z' are both canonical, and are being combined as $Z \bullet Z'$ (where $\bullet$ is a generic operand):

  - Either Z or Z' is t or f - trivial case; depends on operand.
  - $Z = \textbf{if}(P, X, Y)$ and $Z' = \textbf{if}(P', X', Y')$:
    - $P = P'$ – return $\textbf{if}(P, X \bullet X', Y \bullet Y')$.
    - $P < P'$ – return $\textbf{if}(P, X \bullet Z', Y \bullet Z')$.
    - $P > P'$ – return $\textbf{if}(P', Z \bullet X', Z \bullet Y')$.

- Hash tables can be used to optimise, so that comparisons are done by pointer equality (we never build the same OBDD twice). The ordering of variables can be crucial to generating a small OBDD.

4 First Order Logic

- **Function symbols** represent n-place functions, with a **constant symbol** being a 0-place function. **Variables** range over a set of individuals. A **term** is a variable, a constant, or a function whose arguments are terms.

- **Relation symbols** represent n-place relations (e.g. equality is a 2-place relation). An **atomic formula** is a relation whose arguments are terms. A **formula** is built from atomic formulæusing propositional operators and quantifiers.

- The **universal quantifier** can be used as $\forall x.A$ (for all values of x, A holds). The **existential quantifier** can be used as $\exists x.A$ (there exists a value of x such that A holds).

- An **interpretation** $\mathcal{I}$ is a tuple (D, I), such that D is a non-empty set (the domain) and I is a mapping of function and relation symbols to real functions and relations (the domain is essentially the set of values that constants can be, functions can take as arguments and return, and relations can talk about):
  - Constant symbol c – $I[c] \in D$.
  - Function symbol f (n-place) – $I[f] \in D^n \to D$.
  - Relation symbol R (n-place) – $I[R] \subseteq D^n$.

- A **valuation** $\mathcal{V}$ is a function from the variables to the domain D (it supplies the values of free variables). $\mathcal{I}_{\mathcal{V}}[t]$ is the evaluation of the term t under the interpretation $\mathcal{I}$ and the valuation $\mathcal{V}$. The interpretation converts function, relation and constant symbols into real functions, relations and constants, and the valuation converts the variables into actual values.

- A formula A is satisfied under an interpretation $\mathcal{I}$ and valuation $\mathcal{V}$ ($\models_{\mathcal{I},\mathcal{V}} A$) if the evaluation of its components satisfies the top-level connective (defined recursively). For example, if A is a predicate $P(t)$, then the evaluation of the term t ($\mathcal{I}_{\mathcal{V}}[t]$) must be in the set given by the predicate P ($I[p]$). For quantifiers we must extend the valuation to cover the bound variable (if $A = \exists x B$, we have $\exists m \in D.\ \models_{\mathcal{I},\mathcal{V}\{m/x\}} B$).

- A formula A is **satisfiable** if $\models_{\mathcal{I}} A$ for some $\mathcal{I}$ (i.e. $\exists \mathcal{I}.\forall \mathcal{V}.\ \models_{\mathcal{I},\mathcal{V}} A$).

- A variable specified by a quantifier is **bound** within its scope. Bound variables may be renamed (α-conversion). **Free variables** may be substituted for a term, using $A[t/x]$, so long as x is not bound, and t contains no bound variables (otherwise, rename the bound variable(s)).

- The basic quantifier equivalence is that $\neg(\forall x.A) \simeq \exists x.\neg A$. A quantifier can expand its scope so long as no previously free variables would become bound, and can decrease its scope so long as no previously bound variables would become free.

- The sequent calculus rules for quantifiers ($\forall l$ and $\exists r$ can create many instances, and $\forall r$ and $\exists l$ hold provided x is not free in the conclusion):
  - $\forall$ rules: $$\frac{A[t/x], \Gamma \Rightarrow \Delta}{\forall x.A, \Gamma \Rightarrow \Delta}\ (\forall l) \qquad \frac{\Gamma \Rightarrow \Delta, A}{\Gamma \Rightarrow \Delta, \forall x.A}\ (\forall r)$$
  - $\exists$ rules: $$\frac{A, \Gamma \Rightarrow \Delta}{\exists x.A, \Gamma \Rightarrow \Delta}\ (\exists l) \qquad \frac{\Gamma \Rightarrow \Delta, A[t/x]}{\Gamma \Rightarrow \Delta, \exists x.A}\ (\exists r)$$

5 Refutations in Propositional Logic

- A **clause** is a disjunction of literals $\neg K_1 \vee \ldots \vee \neg K_m \vee L_1 \vee \ldots \vee L_n$. This can be written as a set of literals, or as an implication ($K_1, \ldots, K_m \rightarrow L_1, \ldots, L_n$ or $L_1, \ldots, L_n \leftarrow K_1, \ldots, K_m$). A **Horn clause** contains only one positive literal. An empty clause ($\Box$) is a contradiction.

- To prove a propositional statement A, we get a contradiction from assuming $\neg A$. There are two steps in this. Firstly, we negate the goal (A) and convert into CNF, which is a conjunction of clauses. We then transform the clauses (preserving consistency) using one of the methods below. If we arrive at the empty clause (contradiction), we have proven A (the proof fails if we arrive at the empty clause set, as $\neg A$ is satisfiable).

- The **Davis-Putnam-Logeman-Loveland** method:
  - Delete all tautological clauses (containing P and $\neg P$).
  - For each unit clause $\{L\}$, assign L to true (delete clauses containing L, and delete occurrences of $\neg L$ in other clauses).
  - Look for a pure literal L (one for which no clause contains $\neg L$) and delete all clauses containing P.
  - If none of the above can happen, perform a case split on a literal - set the literal to true (as in the case of a unit clause), and continue, but also seperately continue with the literal set to false. If the empty clause is derived in both cases, then the original clauses are inconsistent.

- The method of **resolution** (works on the basis that $(B \vee A) \wedge (\neg B \vee C) \rightarrow A \vee C$):
  - The resolution rule: $\dfrac{\{B, A_1, \ldots, A_m\} \qquad \{\neg B, C_1, \ldots, C_n\}}{\{A_1, \ldots A_m, C_1, \ldots, C_n\}}$
  - Special cases: $\dfrac{\{B\} \qquad \{\neg B, C_1, \ldots, C_n\}}{\{C_1, \ldots, C_n\}} \qquad \dfrac{\{B\} \qquad \{\neg B\}}{\Box}$
  - The **saturation algorithm** works by initially labelling all clauses as passive. One is selected (current), and becomes active. The current clause is resolved with all active clauses to get some new clauses. These are then used to simplify the active and passive clauses (e.g. deletion by subsumption of clauses that contain the new clauses). The new clauses then become passive and the process repeats.
  - **Linear resolution** yields a proof with a linear structure - each resolvent is a parent clause in the next resolution step, and the other parent clause is one of the original clauses.

6 Refutations in First Order Logic

- Conversion to **prenex normal form** is carried out by converting to NNF, then moving the quantifiers to the front. A prenex formula is of the form $Q_1 x_1 \ldots Q_n x_n.(A)$ where the matrix (A) is quantifier free, and the prefix contains quantifiers Q_i.

- **Skolemization** – taking a formula in prenex normal form, go through each existential quantifier in turn, and assign the bound variable to a k-place function symbol, such that there are k universal quantifiers preceding it. So $\forall x_1 \ldots \forall x_k.\exists y.Q_1 z_1 \ldots Q_n z_n.A$ becomes $\forall x_1 \ldots \forall x_k.Q_1 z_1 \ldots Q_n z_n.A[f(x_1, \ldots, x_k)/y]$.

- A Skolemized formula is converted into clauses by 'ignoring' the universal quantifiers, and taking each term in the conjunction as a clause. Note that Skolemization preserves consistency but not validity.

- The **Herbrand universe** H of a set of clauses S is the set of all terms that can be written using just the constants and functions in S. It is defined by:
  - $H_0 = \mathcal{C}$ ($\mathcal{C}$ is the set of constants in S).
  - $H_{i+1} = H_i \cup \{f(t_1, \ldots, t_n) | t_1, \ldots, t_n \in H_i \wedge f \in \mathcal{F}_n\}$ ($\mathcal{F}_n$ is the set of n-place functions in S).
  - $H = \bigcup_{i \geq 0} H_i$.

- The **Herbrand base** HB is the set of all possible applications of the predicates in S to H, defined as $HB = \{P(t_1, \ldots, t_n) | t_1, \ldots, t_n \in H \wedge P \in \mathcal{P}_n\}$ ($\mathcal{P}_n$ is the set of n-place predicates in S).

- A **Herbrand interpretation** is a subset of HB. A set S of clauses is unsatisfiable iff no Herbrand interpretation satisfies S.

- The **Skolem-Gödel-Herbrand theorem** states that a set S of clauses is unsatisfiable iff there is a finite unsatisfiable set S' of ground instances (substituting all variables) of clauses of S.

- A **substitution** is a finite set of replacements, $\theta = [t_1/x_1, \ldots, t_k/x_k]$ where each variable x is distinct, and is replaced by a term t not equal to x. Substitutions compose, such that $t(\phi \circ \theta) = (t\phi)\theta$ (associative but not commutative).

- A substitution θ is a **unifier** of terms t_1 and t_2 if $t_1\theta = t_2\theta$. θ is **more general** than ϕ if $\phi = \theta \circ \sigma$, and is **most general** if it is more general than every other unifier.

- To perform unification, a term is represented as a binary tree of variables, constants and pairs of terms:
  - Constants do not unify with different constants or with pairs.
  - The unifier of a variable x and term t is $[t/x]$, unless x occurs in t.
  - The unifier of two pairs, (t, t') and (u, u'), is $\theta \circ \theta'$ if θ unifies t and u, and θ' unifies t' and u'.

- Resolution can take place on FOL clauses, using unification:
  - **Binary resolution**: $\dfrac{\{B, A_1, \ldots, A_m\} \qquad \{\neg D, C_1, \ldots, C_n\}}{\{A_1, \ldots A_m, C_1, \ldots, C_n\}\sigma}$ provided $B\sigma = D\sigma$.
  - **Factorisation**: $\dfrac{\{B_1, \ldots, B_k, A_1, \ldots, A_m\}}{\{B_1, A_1, \ldots A_m\}\sigma}$ provided $B_1\sigma = \ldots = B_k\sigma$.

- Prolog uses linear resolution on Horn clauses (left-to-right through clauses and clause literals). It uses a depth first search (may not terminate even if a solution), and performs unification without an occurs check (unsound).

- To use a Prolog-like method for first order logic in general, use contrapositives - treat a clause $\{A_1, \ldots, A_m\}$ as m Horn clauses, on that basis that if all A except A_k are false, then A_k must be true for the clause to be true ($A_k \leftarrow \neg A_1, \ldots, \neg A_{k-1}, \neg A_{k+1}, \ldots, \neg A_m$). Also, when proving P you can assume $\neg P$ (extension rule).

- When reasoning about equality (reflexive, symmetric, transitive), you can use:
  - **Paramodulation rule**: $$\frac{\{B[t], A_1, \ldots, A_m\} \qquad \{t = u, C_1, \ldots, C_n\}}{\{B[u], A_1, \ldots A_m, C_1, \ldots, C_n\}}$$

7 Modal Logics

- If W is the set of **possible worlds** and R is an **accessibility relation** between the worlds then (W, R) is a **modal frame** . The modal operators are:
  - $\Box A$ – A is necessarily true.
  - $\Diamond A$ – A is possibly true.
  - $\neg\Diamond A \simeq \Box\neg A$.
- An interpretation I of a frame (W, R) maps propositional letters to a subset of W (i.e. the worlds in which they are true):
  - $w \Vdash A$ – A is true in world w.
  - $\models_{W,R,I} A$ – $w \Vdash A$ for all worlds in W.
  - $\models_{W,R} A$ – $w \Vdash A$ for all worlds in W and for all interpretations.
  - $\models A$ – A is universally valid ($\models_{W,R} A$ for all frames).
- A Hilbert-style proof system needs to be extended with:
  - Distribution: $\Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$
  - Necessitation: $\dfrac{A}{\Box A}$
  - Treat $\Diamond$ as a definition ($\neg\Box\neg A$).
- Pure modal logic (K) can be constrained with the axioms:
  - T – $\Box A \rightarrow A$ (reflexive).
  - S4 – $\Box A \rightarrow \Box\Box A$ (transitive).
  - S5 – $A \rightarrow \Box\Diamond A$ (symmetric).
- Sequent calculus rules for S4 (where $\Gamma^* = \{\Box B \mid \Box B \in \Gamma\}$ and $\Delta^* = \{\Diamond B \mid \Diamond B \in \Delta\}$):
  - $\Box$ rules: $\dfrac{A, \Gamma \Rightarrow \Delta}{\Box A, \Gamma \Rightarrow \Delta}\ (\Box l)$ $\qquad$ $\dfrac{\Gamma^* \Rightarrow \Delta^*, A}{\Gamma \Rightarrow \Delta, \Box A}\ (\Box r)$
  - $\Diamond$ rules: $\dfrac{A, \Gamma^* \Rightarrow \Delta^*}{\Diamond A, \Gamma \Rightarrow \Delta}\ (\Diamond l)$ $\qquad$ $\dfrac{\Gamma \Rightarrow \Delta, A}{\Gamma \Rightarrow \Delta, \Diamond A}\ (\Diamond r)$

8 Tableaux-Based Methods

- Working in NNF, we need only 6 connectives ($\wedge, \vee, \forall, \exists, \Box$ and $\Diamond$) and one side to each sequent:
  - Basic sequent: $$\frac{}{\neg A, A, \Gamma \Rightarrow}\ (\textsc{basic})$$
  - Using a lemma: $$\frac{\neg A, \Gamma \Rightarrow \qquad A, \Gamma \Rightarrow}{\Gamma \Rightarrow}\ (\textsc{cut})$$
  - $\wedge$: $$\frac{A, B, \Gamma \Rightarrow}{A \wedge B, \Gamma \Rightarrow}\ (\wedge l)$$
  - $\vee$: $$\frac{A, \Gamma \Rightarrow \qquad B, \Gamma \Rightarrow}{A \vee B, \Gamma \Rightarrow}\ (\vee l)$$
  - $\forall$: $$\frac{A[t/x], \Gamma \Rightarrow}{\forall x.A, \Gamma \Rightarrow}\ (\forall l)$$
  - $\exists$ (x not free in conclusion): $$\frac{A, \Gamma \Rightarrow}{\exists x.A, \Gamma \Rightarrow}\ (\exists l)$$
  - $\Box$ ($\Gamma^* = \{\Box B \mid \Box B \in \Gamma\}$): $$\frac{A, \Gamma \Rightarrow}{\Box A, \Gamma \Rightarrow}\ (\Box l)$$
  - $\Diamond$: $$\frac{A, \Gamma^* \Rightarrow}{\Diamond A, \Gamma \Rightarrow}\ (\Diamond l)$$
- The proof system works by contradiction - assume the negation of the goal, and arrive at the basic sequent ($A \wedge \neg A$).
- To use for first order logic we need to include:
  - Unification – insert a new free variable with ($\forall l$), then use unification at the end to try to get to the basic sequent. Instantiating a free variable to a term updates the entire proof tree.
  - Skolemization – remove existential quantifiers so we don't need to worry about instantiating to a variable that could be unified to a term. Before Skolemizing, push the quantifiers in (miniscoping), as this leads to shorter proofs.