Logic Notes

Hashan Punchihewa

https://hashanp.xyz

Contents

1 Logic **1**
- 1.1 Propositional Logic 1
- 1.2 Natural Deduction in Propositional Logic 4
- 1.3 Predicate Logic 7
- 1.4 Sorts 10
- 1.5 Specifying Haskell Programs 10
- 1.6 Natural Deduction in Predicate Logic 10

Chapter 1

Logic

1.1 Propositional Logic

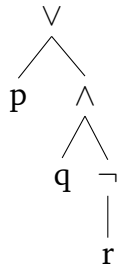
Definition 1.1.1

A propositional formula is made from finite applications of the following rules:

- *A propositional atom e.g.* p*,* q*,* r*,* p_0*,* p' *is a propositional formula.*
- *The logical constants* $\top$ *and* $\bot$ *are propositional formulae.*
- *If* A *is a formula, then* $\neg A$ *is a formula*
- *If* A *and* B *are formulae, then so are* $(A \vee B)$*,* $(A \wedge B)$*,* $(A \to B)$ *and* $(A \leftrightarrow B)$*.*

In order to remove brackets, the following binding convention is used $\neg$, $\wedge$, $\vee$, $\to$, $\leftrightarrow$ from strongest to weakest. But brackets should be used if it aids readability. Since $\vee$ and $\wedge$ are associative, $p \vee q \vee r$ and $p \wedge q \wedge r$ can be used.

Propositional formulae can be expressed as formation trees. E.g. $p \vee q \wedge \neg r$ can be represented as:



Note $\bigvee_{i=1}^{n} p_i = p_1 \vee \ldots \vee p_n$, and $\bigwedge_{i=1}^{n} p_i = p_1 \wedge \ldots \wedge p_n$.

Definition 1.1.2

- *A subformula is a formula that can be constructed from the formation tree.*
- *The principal connective is the connective at the root of a formation tree.*
- *A formula of the form $\top$, $\bot$ or p is an atomic.*
- *A formula of the form $\neg A$ is a negated formula.*
- *A formula of the form $\neg\top$, $\neg\bot$ or $\neg p$ is a negated-atomic.*
- *A formula of the form $A \vee B$ is a disjunction where A and B are the disjuncts.*
- *A formula of the form $A \wedge B$ is a conjunction where A and B are the conjuncts.*
- *A formula of the form $A \to B$ is an implication where A is the antecedent and B is the consequent.*
- *A formula that is either an atomic or a negated-atomic is a literal.*
- *A clause is a disjunction of literals.*

Definition 1.1.3

A situation is an assignment of true or false to each propositional atom.

- $\top$ is always true, and $\bot$ is always false.
- $A \vee B$ is true as long as at least one of A and B are true, and false otherwise.
- $A \wedge B$ is true when both A and B are both true, and false otherwise.
- $A \to B$ is true when A is false or B is true (or both), and false otherwise.
- $A \leftrightarrow B$ is true when A and B share the same truth value and false otherwise.

Note in Computer Science, 'or' is taken to be inclusive, and 'unless' is taken to mean or.

Definition 1.1.4

- *Given formula $A_1, \ldots, A_n, B$, $A_1, \ldots, A_n$ semantically entails B, i.e. $A_1, \ldots, A_n$ therefore B is a valid argument if whenever $A_1, \ldots, A_n$ are true B is also true.*
- *This is denoted as $A_1, \ldots, A_n \vDash B$.*
- *An argument of the form $A, A \to B$, therefore B is called modus ponens.*
- *An argument of the form $\neg B, A \to B$, therefore $\neg A$ is called modus tollens.*
- *A formula A is valid if it is true in every situation and written as $\vDash A$.*
- *A formula is satisfiable if it is true in at least one situation.*
- *If A is satisfiable, then $\neg A$ is not valid.*
- *Two propositional formula are logically equivalent if they are true in exactly the same situations.*

Proposition 1.1.1

- *(Commutativity)* $A \vee B = B \vee A$
- *(Commutativity)* $A \wedge B = B \wedge A$
- *(Associativity)* $(A \vee B) \vee C = A \vee (B \vee C)$
- *(Associativity)* $(A \wedge B) \wedge C = A \wedge (B \wedge C)$
- *(Idempotence)* $A \vee A = A$
- *(Idempotence)* $A \wedge A = A$
- $A \vee \top = \top$
- $A \vee \bot = A$
- $A \wedge \top = A$
- $A \wedge \bot = \bot$
- $A \vee \neg A = \top$
- $A \wedge \neg A = \bot$
- $\neg \top = \bot$
- $\neg \bot = \top$
- $\neg \neg A = A$
- $A \to B = \neg A \vee B$
- $A \leftrightarrow B = (A \to B) \wedge (B \to A)$
- $A \leftrightarrow B = \neg A \leftrightarrow \neg B$
- $\neg(A \leftrightarrow B) = \neg A \leftrightarrow B$
- *(De Morgan)* $\neg(A \wedge B) = \neg A \vee \neg B$
- *(De Morgan)* $\neg(A \vee B) = \neg A \wedge \neg B$
- *(Distributivity)* $A \wedge (B \vee C) = (A \wedge B) \vee (A \wedge C)$
- *(Distributivity)* $A \vee (B \wedge C) = (A \vee B) \wedge (A \vee C)$
- *(Absorption)* $A \wedge (A \vee B) = A$
- *(Absorption)* $A \vee (A \wedge B) = A$

Definition 1.1.5

1. *A formula is in disjunctive normal form if it is a disjunction of conjunctions of literals.*
2. *A formula is in conjunctive normal form if it is a conjunction of disjunctions of literals i.e. a conjunction of clauses.*

1.2 Natural Deduction in Propositional Logic

Definition 1.2.1

1. *$A_1, \ldots, A_n$ syntactically entail B, denoted as, $A_1, \ldots, A_n \vdash B$, if there exists a natural deduction proof B taking $A_1, \ldots, A_n$ as assumptions.*
2. *If $\vdash B$, this means that no givens are needed at all, and B is called a theorem.*
3. *A proof system is sound if every theorem is valid.*
4. *A proof system is complete if every valid formula is a theorem.*
5. *A formula A is consistent if $\nvdash \neg A$.*
6. *A formula is consistent if and only if it is satisfiable.*
7. *A set of formulae $A_1, \ldots$ is consistent if $\bigwedge_{i=1}^{n} A_i$ is consistent. for every integer $n \geq 1$.*

Natural deduction is sound and complete.

Here are the rules for natural deduction:

- $\wedge$ Introduction (or $\wedge I$) is where you have already proven A and B, and can therefore state $A \wedge B$.

1.	A	ass
2.	B	ass
3.	$A \wedge B$	$\wedge I(1,2)$

- $\wedge$ Elimination (or $\wedge E$) is where you have already proven $A \wedge B$, and can therefore state either A or B.

1.	$A \wedge B$	ass
2.	A	$\wedge E(1)$
3.	B	$\wedge E(1)$

- $\rightarrow$ Introduction (or $\rightarrow I$) is where you assume A and prove B having assumed A, so can show $A \rightarrow B$ generally.

1.	A	ass
2.	B	somehow
3.	$A \rightarrow B$	$\rightarrow I(1,2)$

- $\rightarrow$ Elimination (or $\rightarrow E$) is where you have already proven $A \rightarrow B$ and A, so can state B.

1.	$A \to B$	ass
2.	A	ass
3.	B	$\to E(1,2)$

- $\vee$ Introduction (or $\vee I$) is where you have already proven either A or B, and can therefore state $A \vee B$.

1.	A	ass
2.	$A \wedge B$	$\vee I(1)$

- $\vee$ Elimination (or $\vee E$) is where you have already proven shown $A \vee B$, and therefore show from both A and B separately some C can be proven, so C can be stated.

1.	$A \vee B$	ass
2.	A	ass
3.	C	somehow
4.	B	ass
5.	C	somehow
6.	C	$\vee E(1,2,3,4,5)$

- $\neg$ Introduction (or $\neg I$) is where you assume A and derive a contradiction (i.e. prove $\perp$ from A). This means you can state $\neg A$.

1.	A	ass
2.	$\perp$	somehow
3.	$\neg A$	$\neg I(1,2)$

- $\neg$ Elimination (or $\neg E$) is where we have shown A and $\neg A$ and can therefore state $\perp$.

1.	A	ass
2.	$\neg A$	ass
3.	$\perp$	$\neg E(1,2)$

- $\perp$ Introduction (or $\perp I$) is another name for $\neg E$,
- $\perp$ Elimination (or $\perp E$) is where you have shown $\perp$ and can therefore prove anything.

1.	$\bot$	ass
2.	A	$\bot E(1)$

- $\neg\neg$ Elimination (or $\neg\neg E$) is where we have shown $\neg\neg A$, and can therefore show A:

1.	$\neg\neg A$	ass
2.	A	$\neg\neg E(1)$

- $\top$ Introduction (or $\top I$) is where you can introduce $\top$ from anywhere in the proof:

1.	$\top$	$\top I$

- $\leftrightarrow$ Introduction (or $\leftrightarrow$ I) is where you have shown $A \rightarrow B$ and $B \rightarrow A$, and can therefore state $A \leftrightarrow B$:

1.	$A \rightarrow B$	ass
2.	$B \rightarrow A$	ass
3.	$A \leftrightarrow B$	$\leftrightarrow I(1,2)$

- $\leftrightarrow$ Elimination (or $\leftrightarrow$ E) is where you have shown $A \leftrightarrow B$, some from either A or B, you can state the other.

1.	$A \leftrightarrow B$	ass
2.	A	ass
3.	B	$\leftrightarrow E(1,2)$

- The only lemma you are allowed to simply quote is $A \vee \neg A$, which can be justified as 'Lemma'.
- One derived rule you can use is proof by contradiction or (PC), where assuming $\neg A$ you show $\bot$, and can therefore show A:

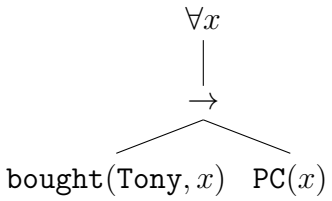
1.	$\neg A$	ass
2.	$\bot$	somehow
3.	A	$PC(1,2)$

1.3 Predicate Logic

Definition 1.3.1

- *A signature consists of a set of constants (*$\mathtt{c}$*,* $\mathtt{d}$*, ...), a set of relation symbols with specified arities and a set of function symbols with specified arities.*
- *A term in a given signature L is a constant in L, a variable (x, y, ...) or for an n-ary function symbol R, $R(t_1, \ldots, t_n)$, where $t_1, \ldots, t_n$ are L-terms.*
- *A ground term (or a closed term) is a term that is not a variable.*
- *A formula for a given signature L is made from finite applications of the following rules:*
  - *If R is an n-ary relation symbol in L, then $R(t_1, \ldots, t_n)$ is an atomic L-formula where $t_1, \ldots, t_n$ are L-terms.*
  - *If t and t' are L-terms, then $t = t'$ is an atomic L-formula.*
  - *$\top$ and $\bot$ are atomic L-formulae.*
  - *If A and B are L-formulae, then so are $(\neg A)$, $(A \vee B)$, $(A \wedge B)$, $(A \to B)$ and $(A \leftrightarrow B)$.*
  - *If x is a variable and A is an L-formula, then $(\forall x A)$ and $(\exists x A)$.*

$\forall x$ and $\exists x$ has the same binding strength as $\neg$. Consider also the formation tree for $\forall x\ (\mathtt{bought}(\mathtt{Tony}, x) \to \mathtt{PC}(x))$:



Definition 1.3.2

- *A structure for a given signature L, M consists of a non-empty set of objects, called the domain (or universe) of M, denoted as $\mathrm{dom}(M)$, and provides an interpretation of constants, relation symbols and function symbols in L. More specifically, a constant in L, is given some object by M, that is in $\mathrm{dom}(M)$, a relation symbol is given some relation on $\mathrm{dom}(M)$ and an n-ary function symbol is given function $f : \mathrm{dom}(M)^n \to \mathrm{dom}(M)$.*

Definition 1.3.3

For some formula A:

- *The occurrence of some variable x in A, is said to be bound if it is under $\forall x$ or $\exists x$ in the formation tree.*
- *Otherwise, the occurrence is considered free.*

- *The free variables of A, are those with free occurrences in A.*
- *A is a sentence if it has no free variables.*

Definition 1.3.4

- *If M is a structure, then an assignment into M allocates each variable an object in $\mathrm{dom}(M)$.*
- *The value of a term is then the value given by M if the term is a constant, if it is a variable it is the value given by the assignment, and otherwise if it is a function symbol with terms $t_1, \ldots, t_n$, whose values are $a_1, \ldots, a_n$, its value is $f(a_1, \ldots, a_n)$, where f is given by M.*

To evaluate an L-formula in some L-structure M, with under an assignment h:

- If R is an n-ary relation symbol with terms $t_1, \ldots, t_n$, where the value of the terms in M under h is $a_1, \ldots, a_n$, then $M, h \vDash R(t_1, \ldots, t_n)$ if $(a_1, \ldots, a_n)$ is in the corresponding relation in M.
- If t and t' are terms, then $M, \vDash (t = t')$ if t and t' have the same value in M under h.
- $M, h \vDash \top$ and $M, h \nvDash \bot$.
- $A \wedge B$, $A \vee B$, $\neg A$, $A \to B$, $A \leftrightarrow B$, works similarly to propositional logic.
- $M, h \vDash \forall x A$, if for every assignment g, where g has the same value for every variable in h except for x, $M, g \vdash A$.
- $M, h \vDash \exists x A$, if for some assignment g, where g has the same value for every variable in h except for x, $M, g \vdash A$.

Note for an L-formula A, whether or not for some structure M and assignment h, $M, h \vDash A$ depends only on h, for those variables that are free in A.

Note the following for when counting in predicate logic:

- $\exists x\ P(x)$, means there is at least one object, that satisfies $P(x)$.
- $\exists x \exists y\ P(x) \wedge P(y) \wedge (x \neq y)$, means there is at least two objects, that satisfy $P(x)$.
- But this can also be written as $\forall x \exists y P(y) \wedge (x \neq y)$.
- $\exists x \exists y \exists z\ (P(x) \wedge P(y) \wedge P(z) \wedge (x \neq y) \wedge (x \neq z) \wedge (y \neq z))$ means there is at least three objects, that satisfy $P(x)$.
- This can, again, be written more concisely: $\forall x \forall y \exists z\ (P(z) \wedge (x \neq z) \wedge (y \neq z))$.

Definition 1.3.5

- *If $A_1, \ldots, A_n, B$ are L-formulae, then $A_1, \ldots, A_n$ therefore B is a valid argument, if for any L-structure M, and assignment h, if $M, h \vDash A_i$ for $1 \leq i \leq n$, then $M, h \vDash B$. In this case we write, $A_1, \ldots, A_n \vDash B$.*

- *A L-formula A is valid if for any L-structure M and assignment h, $M, h \vDash A$, and this is denoted as $\vDash A$.*
- *An L-formula A is satisfiable if for some L-structure M and assignment h, $M, h \vDash A$.*
- *Two L-formulae A and B are logically equivalent if for every L-structure M and assignment h, $M, h \vDash A$ if and only if $M, h \vDash B$.*

Proposition 1.3.1

There exists no algorithmic procedure to identify valid arguments in predicate logic.

Proposition 1.3.2

- $\forall x \forall y A = \forall y \forall x A$
- $\exists x \exists y A = \exists y \exists x A$
- $\neg \forall x A = \exists x \neg A$
- $\neg \exists x A = \forall x \neg A$
- $\forall x (A \wedge B) = (\forall x A) \wedge (\forall x B)$
- $\exists x (A \vee B) = (\exists x A) \vee (\exists x B)$

Proposition 1.3.3

If x doesn't occur in A as a free variable.

- $\exists x (A \wedge B) = A \wedge \exists x B$
- $\forall x (A \vee B) = A \vee \forall x B$
- $\forall x (A \to B) = A \to \forall x B$
- $\exists x (A \to B) = A \to \exists x B$
- $\forall x (B \to A) = \exists x B \to A$
- $\exists x (B \to A) = \forall x B \to A$

Proposition 1.3.4

- *If x is any variable, and y is a variable that does not occur in A, then a logically equivalent formula can be gained by replacing bound occurrences of x with y, and replacing $\forall x$ with $\forall y$ and $\exists x$ with $\exists y$.*
- *$t = t$ is valid for any term t.*
- *$t = u$ is equivalent to $u = t$.*
- *Leibniz's principle: If y doesn't occur in a formula A, and B is got by replacing all free occurrences of x with y, then $(x = y) \to (A \leftrightarrow B)$ is valid.*

1.4 Sorts

Many-sorted logic is an extension of predicate logic where there is a finite collection of sorts; each variable and constant is associated with a sort, e.g. $c : s$, and each function n-ary function symbol f is defined as $f : (s_1, \ldots, s_n) \to s$. In addition, each relation symbol is defined over $s_1, \ldots, s_n$. $t = t'$, is only meaningful when t and t' have the same sort. Sorts can be annotated to quantification e.g. $\forall x : S$ and $\exists y : S$. A structure assigns each object in its domain to a single sort.

1.5 Specifying Haskell Programs

- `Nat` is used to denote Natural, and `[Nat]` is used to denote lists over the naturals. Typically the length of a list is denoted by the function symbol #. Usual Haskell functions have the same.
- To represent partial functions, relations can be used or an arbitrary value can be chosen. Typically the latter approach is used.
- Equality is denoted using a single equals.
- A pre-condition imposes a restriction on the input of a function.
- A post-condition imposes a restriction on the output of a function.

1.6 Natural Deduction in Predicate Logic

Note: if A is a formula, and x a variable, then $A(t/x)$ refers to the formula got from A by replacing all free occurrences of x by t.

- $\exists$ Introduction (or $\exists I$) is where from $A(t/x)$, you can write $\exists x A$.

1.	$A(t/x)$	ass
2.	$\exists x A$	$\exists I(1)$

- $\exists$ Elimination (or $\exists E$) is where if you have $\exists x A$, and can prove B (which does not contain c) having assumed $A(c/x)$, for some new constant c, then you can state B.

1.	$\exists x A$	ass
2.	$A(c/x)$	ass
3.	B	somehow
4.	B	$\exists E(1, 2, 3)$

- $\forall$ Introduction (or $\forall I$) is where if you can show for some new constant, you assume to exist, that $A(c/x)$ holds, then you can state $\forall x A$. Note this is the only time in the logic where a line in natural deduction is a constant, not a formula.

1.	c	$\forall I$ const
2.	$A(c/x)$	somehow
3.	$\forall x A$	$\forall I(1,2)$

- $\forall$ Elimination (or $\forall E$) is where given $\forall x A$, you can write $A(t/x)$ for any closed term t.

1.	$\forall x A$	ass
2.	$A(t/x)$	$\forall E(1)$

- $\forall \to E$ is a derived rule where given $\forall x(A \to B)$, $A(t/x)$, you can derive $B(t/x)$.

1.	$\forall x(A \to B)$	ass
2.	$A(t/x)$	ass
3.	$B(t/x)$	$\forall \to E(2,1)$

- The reflexivity rule (or refl) allows you to say $t = t$ for any closed term t.

1.	$t = t$	refl

- The substitution rule (or sub) allows you to say given $t = u$, and $A(t/x)$, $A(u/x)$.

1.	$A(t/x)$	ass
2.	$t = u$	ass
3.	$A(u/x)$	sub$(1,2)$

- The symmetry rule (or sym) allows you to say given $t = u$, $u = t$.

1.	$t = u$	ass
2.	$u = t$	sym(1)

Definition 1.6.1

1. *$A_1, \ldots, A_n$ are sentences in predicate logic that syntactically entail B, denoted as, $A_1, \ldots, A_n \vdash B$, if there exists a natural deduction proof B taking $A_1, \ldots, A_n$ as assumptions.*
2. *If $A_1, \ldots, A_n$ are sentences in predicate logic, if $A_1, \ldots, A_n \vdash B$ then $A_1, \ldots, A_n \vdash B$,*

the proof system is sound.

3. *If* $A_1, \ldots, A_n$ *are sentences in predicate logic, if* $A_1, \ldots, A_n \vDash B$ *then* $A_1, \ldots, A_n \vdash B$*, the proof system is complete.*

Natural deduction is sound and complete with respect to predicate logic.