# 1 Lectures

### Lecture 1: Prop intro

A proposition is a declarative sentence that is true or false. There are three types of symbols in propositional logic: variables, connectives, and punctuation (eg brackets).

Well formed propositional formula is defined inductively: In the base case, a single propositional variable is well formed. If $\alpha$ is well formed, then $(\neg\alpha)$ is well formed. If $\alpha, \beta$ are well formed, then $(\alpha \star \beta)$ is well formed, where $\star$ is one of the four binary connectives.

Translations reference:

1. $(p \wedge q)$: p but q, not only p but q, p while q, p despite q, p yet q, p although q
2. $(p \implies q)$: q if p, p only if q, q when p, p is sufficient for q, q is necessary for p
3. $(p \iff q)$: p is equivalent to q, p exactly if q, p is necessary and sufficient for q

### Lecture 2: Prop syntax

Theorem: There is a unique way to construct each well formed formula.

Other properties of WFF: every WFF has at least one propositional variable, has an equal number of opening and closing brackets, every proper prefix has more opening than closing brackets. A proper prefix is a non empty segment of a formula starting from the first symbol and ending before the last symbol.

### Lecture 4: Prop semantics

A truth valuation is a function $t$ mapping from the set of all propositional variables $P$ to the set $\{F, T\}$. In other words, a formula or variable under $t$ is either true or false. Every line in a truth table corresponds to a truth valuation.

Tautology: For every truth valuation $t$, $\alpha^t = T$.

Satisfiable: There exists a truth valuation $t$, $\alpha^t = T$.

Contradiction: For every truth valuation $t$, $\alpha^t = F$.

Note that every tautology is also satisfiable. Thus, we frequently distinguish a third category: satisfiable but not a tautology where under some truth valuations, the formula is true and under others, it is false.

To determine if a formula is a tautology, contradiction, or satisfiable, we can use a truth table, reasoning with logical equivalence, or a valuation tree. The point of a valuation tree is to split variables off into true/false and simplify the formula in that way. It is a more compact truth table.

## Lecture 5: Prop equivalence

Two formulas $\alpha, \beta$ are logically equivalent iff their truth table values are the same, and $(\alpha \iff \beta)$ is a tautology.

Strategies for proving logical equivalence: try getting rid of implications and iff, move negations inwards using De Morgan's laws, work from the more complex side first. Each line of the proof should have only one step.

Dead code: the conditions leading up to this branch of code is always false. Thus, this code will never be executed.

Adequate set of connectives: With an adequate set, you can express every propositional logic formula with its elements. This means any connective not in the set can be expressed with elements in the set. An example would be $\{\wedge, \vee \implies, \iff, \neg\}$ although it can be simplified further. Further examples of adequate sets:

1. $\{\wedge, \vee, \neg\}$. Note that $(a \implies b) \equiv ((\neg a) \vee b)$ and $(a \iff b) \equiv ((a \wedge b) \vee ((\neg a) \wedge (\neg b)))$
2. $\{\wedge, \neg\}, \{\vee, \neg\}, \{\implies, \neg\}$

Showing that a set is not adequate: usually, it cannot express not. For example, $\{\vee, \wedge\}$ is not adequate since it cannot express not. Prove this by structural induction on a formula where all the propositional variables are true.

## Lecture 6: Semantic enailment

Semantic entailment formalizes logic: is a conclusion true if we assume all the premises to be true¿

Let $\Sigma$ be a set of premises. Let $\varphi$ be the conclusion. A truth valuation $t$ satisfies $\Sigma$, denoted $\Sigma^t = T$, if all formulas in $\Sigma$ are true. We say that this set of formulas, $\Sigma$ is satisfied by $t$. Then, $\Sigma$ entails $\varphi$, or $\Sigma \vDash \varphi$ if for all truth valuations $t$, if $\Sigma^t = T$, then $\varphi^t = T$.

Entailment in different conditions: If $\Sigma \vDash \varphi$ and:

1. $\Sigma$ is the empty set or a tautology. Then, $\varphi$ is a tautology
2. $\Sigma$ is a contradiction. Then $\varphi$ can be anything.
3. $\varphi$ is a tautology. Then $\Sigma$ can be anything.

## Lecture 7: Prop logic, Natural deduction

Natural deduction is a proof system. A proof starts with a set of premises and transforms these premises based on a set of inference rules into a conclusion. If this proof exists from the set of formulas (premises) $\Sigma$ to a conclusion $\varphi$, then we write $\Sigma \vdash \varphi$.

**Lecture 9: Soundness and completeness**

Semantic entailment: $\Sigma \vDash \varphi$ iff every valuation where $\Sigma^t = T$ implies that $\varphi^t = T$.

Natural deduction: $\Sigma \vdash \varphi$ iff there is an ND proof beginning with the premises of $\Sigma$ and ending with $\varphi$.

Natural deduction is both sound and complete. Soundness means that if $\Sigma \vdash \varphi$, then $\Sigma \vDash \varphi$ and completeness is that if $\Sigma \vDash \varphi$, then $\Sigma \vdash \varphi$. Soundness shows that a conclusion of a proof is a logical consequence of the premise and completeness shows that all logical consequences are provable in Natural Deduction.

Proof of soundness: done on structural induction on a proof. A proof either does not need any inference rules to reach the conclusion or uses an inference rule on one or more subproofs in the proof.

**Lecture 10: Predicate logic, intro and translations**

Relationships, generalizations of patterns, and infinite domains are hard to express in propositional logic.

Generalized De Morgan's Laws:

$$(\neg(\forall x\ P(x))) \equiv (\exists x\ (\neg P(x))), (\neg(\exists x\ P(x))) \equiv (\forall x\ (\neg P(x)))$$

The $\forall$ quantifier pairs well with $\implies$ and $\exists$ pairs well with $\wedge$.

**Lecture 12: Predicate logic, syntax**

Domain: non-empty set of objects

Constants: concrete objects in domain

Variable: placeholders for concrete objects in domain

Functions: takes objects in domain as input, returns object in domain, lowercase letter

Predicate: takes objects in domain as input, returns T/F, uppercase letter

Quantifiers: $\forall, \exists$

Arity of a function/predicate: $f^{(1)}, P^{(3)}$

Term: refers to object in domain. Constants and variables are atomic terms. $f(t_1, \ldots, t_n)$ is a term where $t_1, \ldots, t_n$ are terms and $f$ is an n-ary function.

Predicate formula: $P(t_1, \ldots, t_n)$ is an atomic formula if $P$ is a predicate and $t_i$ is a term. For terms $\alpha, \beta$, then $(\neg\alpha), (\alpha \star \beta), (\forall x\ \alpha), (\exists x\ \alpha)$ are all formulas, where $\star$ is one of the binary connectives.

In the formula $(\forall x\ P(y))$, $y$ is still a free variable since the quantifier is for $x$.

Closed formula: formula with no free variables.

$\alpha[t/x]$ is the formula $\alpha$ with all FREE instances of $x$ replaced by $t$. Also, $t$ must be a term.

Variable capture: When you do a substitution but the term you're substituting has variables which are the same name as bound variables. Then, rename all the bound instances of this variable then do the substitution.

In a formula $(\forall x\ \alpha)$, the scope of the quantifier is $\alpha$.

**Lecture 13: Predicate logic, semantics**

To determine whether a formula is true, we need an interpretation and possibly environment if there are free variables.

Interpretation: outlines the domain, meaning for each constant symbol ($a^I = 3$), meaning for each function symbol (eg: $f^I(x) = x + 1$ or $f^I(1) = 2$), meaning for each predicate symbol (eg: $P^I(x, y)$ is true iff $x > y$, or $P^I = \{\langle 1, 2\rangle\}$ to indicate the tuples such that $P^I$ is true with these tuples).

An alternate way of saying $P^I(5) = T$ is $I \vDash P(5)$

A function must be total: For the function $f^{(k)}$, $f^I(d_1, \ldots, d_k)$ must be in the domain for any $k$-tuple $(d_1, \ldots, d_k)$.

Environment: maps every variable symbol to domain element. For example, $E(x) = 3$. Rules:

1. An environment has to map every variable symbol to a domain element, even if the variable does not appear in a formula and even if the variable is bound.
2. Bound variables get meaning from the quantifier. Free variables get meaning from the environment. If a formula has no free variable, an interpretation is enough to give the formula meaning. No need to define environment but when proving the truth valuation of a formula, say that you are using an arbitrary environment.

Environment override: $E[x \mapsto d]$ keeps all mappings in $E$ intact except reassignments $x$ to $d \in dom(I)$. Thus, $E[x \mapsto 2](x) = 2, E[x \mapsto 2][y \mapsto 100](x) = 2$ and if $x$ is a free variable such that $E(x) = 5$, then $E[y \mapsto 1](x) = 5$.

A formula $\alpha$ is valid if $I \vDash_E \alpha$ for every $I, E$. It is satisfiable if $I \vDash_E \alpha$ for some $I, E$. It is unsatisfiable if $I \nvDash_E \alpha$ for all $I, E$.

**Lecture 14: Predicate logic, semantic entailment**

Let $\Sigma$ be a set of predicate logic formulas. Let $\varphi$ be a predicate formula. Then, $\Sigma \vDash \varphi$ iff for every $I, E$, if $I \vDash_E \Sigma$, then $I \vDash_E \varphi$.

**Lecture 17: Predicate logic, soundness and completeness**

To prove that a set of formulas is satisfiable, find one pair $(I, E)$ such that all formulas under $(I, E)$ are true. To prove a set is unsatisfiable, consider an arbitrary $(I, E)$ and show that at least one formula is false.

Satisfaction rules:

1. $(\forall x\ \alpha)^{(I,E)} = T$ iff $\alpha^{(I,E[x\mapsto d])} = T$ for every $d \in dom(I)$
2. $(\exists x\ \alpha)^{(I,E)} = T$ iff $\alpha^{(I,E[x\mapsto d])} = T$ for some $d \in dom(I)$
3. $(\alpha \implies \beta)^{(I,E)} = T$ iff $\alpha^{(I,E)} = F$ or $\alpha^{(I,E)} = \beta^{(I,E)} = T$
4. $(\alpha \vee \beta)^{(I,E)} = T$ iff $\alpha^{(I,E)} = T$ or $\beta^{(I,E)} = T$
5. $(\alpha \iff \beta)^{(I,E)} = T$ iff $\alpha^{(I,E)} = \beta^{(I,E)}$

**Lecture 18: Program verification**

Hoare triple $(\!|P|\!)C(\!|Q|\!)$: $P$ is precondition, $C$ is code/program, $Q$ is postcondition. The triple means: If the state of program $C$ before execution satisfies $P$, then the ending state of $C$ after execution satisfies $Q$.

Partial correctness: $(\!|P|\!)C(\!|Q|\!)$ is satisfied under partial correctness iff for every state $s_1$ that satisfies $P$, if $C$ from $s_1$ terminates in $s_2$, then $s_2$ satisfies $Q$.

Total correctness: $(\!|P|\!)C(\!|Q|\!)$ is satisfied under total correctness iff for every state $s_1$ that satisfies $P$, $C$ from $s_1$ terminates in state $s_2$ and $s_2$ satisfies $Q$. If a program is totally correct, then it is partially correct as well.

**Lecture 23: Undecidablility**

Decision problem: question with a yes or no answer

Algorithm solves a problem iff it produces the correct output for the problem for every input.

A decision problem is decidable iff there is an algorithm to solve the problem.

Halting problem: Given a program $P$ and input $I$, does $P$ halt on $I$?

## 2 Other

When dealing with formulas rather than single variables, you have to concretely define these formulas. For example, to prove $\{(\alpha \implies \beta)\} \not\models (\beta \implies \alpha)$, you can't just say: consider a truth valuation where $\alpha^t = F, \beta^t = T$ since if $\beta = (p \wedge (\neg p))$, it can never be true. Instead, say: choose $\alpha = p, \beta = q$ to be single propositional variables. Then, consider a valuation...

### 2.1 Structural induction template

We want to prove for every ... (well-formed formula ...) that $P(\varphi)$ holds, where $P$ is the property...

Base case:

$\varphi$ is a propositional variable $q$. Then... Thus, $P(\varphi)$ holds.

Induction step:

Suppose $P(\alpha), P(\beta)$ hold for some well-formed formulas $\alpha, \beta$.

Case 1: $\varphi$ is $(\neg\alpha)$, where $\alpha$ is a well formed formula. Then, by the induction hypothesis, $P(\alpha)$ holds so... Thus, $P(\varphi)$ holds.

Case 2: $\varphi$ is $(\alpha \star \beta)$ where $\star$ is one of the four binary connectives: $\implies, \iff, \vee, \wedge$. By the induction hypothesis, $P(\alpha)$ and $P(\beta)$ hold... Thus, $P(\varphi)$ holds.

By the principle of structural induction, $P(\varphi)$ holds for every well formed formula $\varphi$.

### 2.2 Logical equivalences

Note that $p$ unless $q$, if $(\neg q)$ then $p$, $(\neg p)$ only if $q$ are all equivalent. In the table below, the left side is logically equivalent to the right side.

| | | |
|---|---|---|
| $p \implies q$ | $((\neg p) \vee q)$ | |
| $(\neg(p \implies q))$ | $(p \wedge (\neg q))$ | |
| $p$ only if $q$ | $(p \implies q)$ | |
| | $((\neg q) \implies (\neg p))$ | contrapositive of above |
| $p$ even if $q$ | $p$ | $q$ is irrelevant |
| $p$ unless $q$ | $(p \vee q)$ | |
| | $((\neg q) \implies p)$ | Eat fruit unless it is an apple means I eat all fruits except apples |
| $p$ if $q$ | $(q \implies p)$ | |
| $p$ or (exclusive) $q$ | $((p \vee q) \wedge (\neg(p \wedge q)))$ | |
| | $((p \vee q) \wedge ((\neg p) \vee (\neg q)))$ | |
| | $((p \wedge (\neg q)) \vee ((\neg p) \wedge q))$ | |
| | $(\neg(p \iff q))$ | |
| | $((\neg p) \iff q) \equiv (p \iff (\neg q))$ | |
| $((p \wedge q) \implies r)$ | $(p \implies (q \implies r))$ | |
| | $(q \implies (p \implies r))$ | |

Soundness and completeness: to show that a ND proof does not exist, show that that there is no entailment. This is equivalent to proving the contrapositive of soundness.

To prove semantic entailment: we can now also use soundness (find a ND proof). In addition to using truth tables, valuation trees, and a series of logical equivalences.

The empty set of premises only entails tautologies since the empty set is true under every truth valuation so the conclusion must also be true under every truth valuation. A contradiction entails anything since there are no truth values under which a contradiction is true so the conclusion does not need to be true under any truth valuations as well.