Propositional Basics

- "Correctness" - follows expected behaviour
- "Completeness" - does all that is expected
- "Proposition" - a declaration that can be true / false but not both

• Symbols

- Replace English-language statements w/ symbols
- An infinite set of optionally sub-scripted upper-case letters

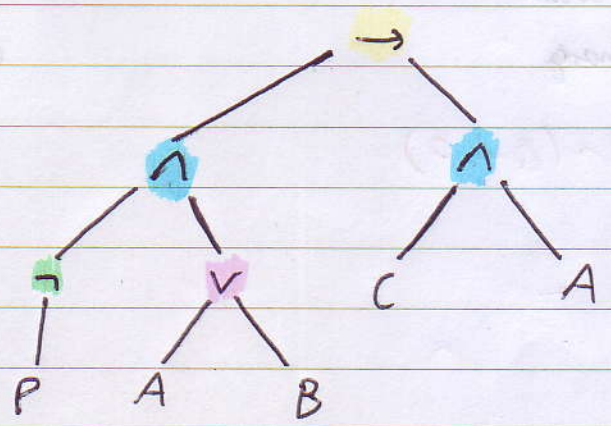
• Connectives

¬	Negation
∧	And
∨	Or
→	Implication
↔	Equivalence

This is the order of precedence, from the top going down.

• Syntactic Decomposition Tree

- Used to "unravel" complex formulae.
- Eg. $(\neg P \wedge [A \vee B]) \rightarrow (C \wedge A)$



• Interpretations

- A well-formed formula (wff) must be interpreted to give a true/false value.

- If I( ) is an interpretation formula and P is a wff, then $I(P) \in \{0,1\}$.

• Truth Tables

- Eg. $A = (P \vee \neg Q) \wedge R$

I	P	Q	R	¬Q	P∨¬Q	A
0	0	0	0	1	1	0
1	0	0	1	1	1	1
2	0	1	0	0	0	0
3	0	1	1	0	0	0
4	1	0	0	1	1	0
5	1	0	1	1	1	1
6	1	1	0	0	1	0
7	1	1	1	0	1	1

Interpretation
- Qty will be 2 to pow. of number of prop. symbols

Prop. symbols
- Tip: count in binary

Elements of formula

Truth value of interpretation

• Special Terms.

- A tautology is true in all interpretations
- A contradiction is false in all interpretations
- Two or more formulae are equivalent if they give the same truth value in all interpretations, denoted by $\equiv$.
  - Eg. $A \rightarrow B \equiv \neg A \vee B$
- A contingency is neither a tautology or a contradiction

Equivalencies

$P \vee P \equiv P \wedge P \equiv P$ — Idempotent

$P \vee Q \equiv Q \vee P$
$P \wedge Q \equiv Q \wedge P$
} Commutative

$P \wedge (Q \wedge R) \equiv (P \wedge Q) \wedge R$
$P \vee (Q \vee R) \equiv (P \vee Q) \vee R$
} Associative

$P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$
$P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$
} Distributive

$\neg(P \wedge Q) \equiv (\neg P \vee \neg Q)$
$\neg(P \vee Q) \equiv (\neg P \wedge \neg Q)$
} De Morgan's Laws (note: operator flip)

$P \rightarrow Q \equiv \neg P \vee Q$

$P \rightarrow Q \equiv \neg Q \rightarrow \neg P$

$\neg(P \rightarrow Q) \equiv P \wedge \neg Q$

- Subformulae can be replaced with equivalent formula without changing the result.

$P \leftrightarrow Q \equiv (P \rightarrow Q) \wedge (Q \rightarrow P)$

CNF/DNF

- A literal is a prop. symbol or a negation thereof.

• Conjunctive Normal Form

- A formula is in CNF if it is a conjunction of one or more formulae, each of which is a disjunction of one or more literals

• Disjunctive Normal Form

- A formula is in DNF if it is a disjunction of one or more formulae, each of which is a conjunction of one or more literals.

• Conversion Methods

- Keep applying equivalency rules until CNF/DNF is reached.
- Using truth tables:
  - DNF is the disjunction of all true lines, with symbols joined by ∧.
  - CNF is the conjunction of the negation of all false lines, with symbols joined by ∧

• Example:

$P \vee (Q \rightarrow R)$

I	P	Q	R	$Q \rightarrow R$	$P \vee (Q \rightarrow R)$
0	0	0	0	1	1
1	0	0	1	1	1
2	0	1	0	0	0
3	0	1	1	1	1
4	1	0	0	1	1
5	1	0	1	1	1
6	1	1	0	0	1
7	1	1	1	1	1

DNF = $(\neg P \wedge \neg Q \wedge \neg R) \vee (\neg P \wedge \neg Q \wedge R) \vee (\neg P \wedge Q \wedge R) \vee \ldots\ldots$

CNF = $\neg(\neg P \wedge Q \wedge \neg R)$

= $P \vee \neg Q \vee R$

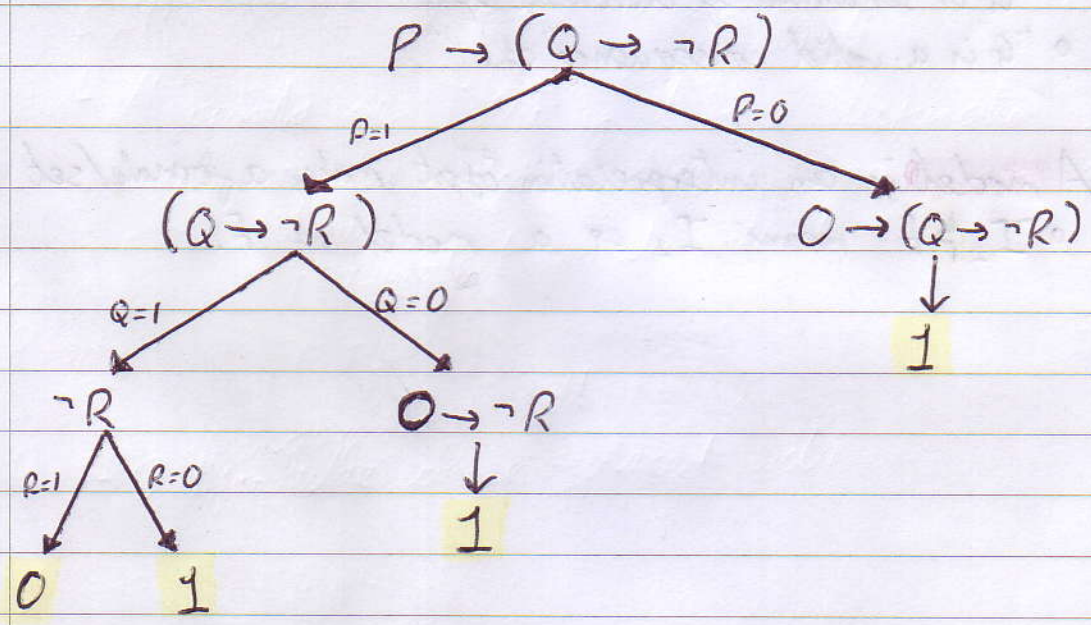
Substitution

- $W(P/Q)$ means the formula W with all values of P simultaneously replaced with Q.

- Eg. $W = P \to R$

$$W(P/Q) = Q \to R$$

Quine's Method / Tree



- All ends are 1 → Tautology
- All ends are 0 → Contradiction
- Ends are mixed → Contingency.

• Satisfiability

- Formula F is satisfiable if there is an interpretation I making F true.

- A set of formulae $S = \{F_1, F_2 \dots F_n\}$ is satisfiable/consistent if there is an interpretation I making every formula in S true.
  - The set $S = \{F_1, F_2 \dots F_n\}$ is satisfiable iff the conjunction $F_1 \wedge F_2 \dots F_n$ is satisfiable.

- $F_1, F_2, \dots, F_n \models G$ means G follows from $F_1 \dots F_n$.
  - "G is semantically entailed by..."
  - "G is a valid consequence of..."

- A model is an interpretation that makes a formula/set true.
  - $I_2 \models F$ means I_2 is a model of F.

• Inference Systems / Natural Deduction

- ND is a type of inference system
  - It derives new sentences from existing ones.
- $F_1, F_2, \cdots, F_n \vdash G$ derives that G is derived from $F_1 \ldots F_n$
- An inference system must be correct (sound):

  if $A \vdash B$, then $A \models B$
- An ideal inference system is complete:

  if $A \models B$, then $A \vdash B$
- $\vdash G$ has no condition, meaning it is always true
- Alternate definitions:
  - "Sound" - all the rules work
  - "Complete" - there are enough rules to prove all true interpretations.

• Basic Rules

$$\frac{G, H}{G \wedge H} \quad (\wedge\text{-I}) \qquad \frac{G \wedge H}{G \text{ (or } H)} \quad (\wedge\text{-E})$$

$$\frac{G}{G \vee H} \quad (\vee\text{-I}) \qquad \frac{G \vee H,\ G \to I,\ H \to I}{I} \quad (\vee\text{-E})$$

↑ Can append anything with ∨

$$\frac{G,\ G \to H}{H} \quad (\to E) \text{ or } (\text{Modus Ponens})$$

$$\frac{G \to H,\ G \to \neg H}{\neg G} \quad (\neg I) \qquad \frac{\neg G \to H,\ \neg G \to \neg H}{G} \quad (\neg E)$$

• Derived Rules

$$\frac{G \vee H,\ \neg G}{H} \quad (\vee E2) \text{ or } (\text{Disjuntive Syllogism})$$

$$\frac{G \to I,\ H \to J,\ G \vee H}{I \vee J} \quad (CD) \text{ or } (\text{Constructive Dilemma})$$

$$\frac{G \to I,\ H \to J,\ \neg I \vee \neg J}{\neg G \vee \neg H} \quad (DD) \text{ or } (\text{Destructive Dilemma})$$

$$\frac{\neg G}{G \to H} \quad (\to I1) \qquad \frac{H}{G \to H} \quad (\to I2)$$

False can imply anything

Anything can imply a truth

$$\frac{G \to H}{\neg G \vee H} \quad (\to E1)$$

$$\frac{\neg\neg G}{G} \quad (\neg E1) \qquad \frac{\neg G \to H,\ G \to H}{H} \quad (\neg E2)$$

$$\frac{P \to R,\ \neg P \to Q}{R \vee Q} \quad (\text{Resolution}) \qquad \frac{G \to H,\ H \to I}{G \to I} \quad (\text{Hypothetical Syllogism})$$

Predicate Logic

- Universal Quantifier : $\forall x$
  - "for all x..."
- Existential Quantifier : $\exists x$
  - "there exists an x..."

- Functional symbol - returns a member of the set
- Predicate symbol - returns true/false

• Scope

$\exists x(P(x)) \wedge Q(x)$

$\exists x(P(x) \wedge Q(x))$

= Bound by $\exists$
= Free from $\exists$

$\exists x(P(x) \wedge Q(x)) \wedge R(x)$

• Interpretation

- An interpretation allows a truth value to be assigned to a wff with no free variables
  - Also called a sentence or a closed wff.

• Substitution

- $F(x/d)$ denotes the wff obtained by replacing all free instances of x with d.

• Quantifier Relationships

$\neg \forall x F \equiv \exists x \neg F$

$\forall x F \equiv \neg \exists x \neg F$ $\quad \forall x \equiv \neg \exists x \neg$

$\neg \exists x F \equiv \forall x \neg F$ $\quad \exists x \equiv \neg \forall x \neg$

$\exists x F \equiv \neg \forall x \neg F$

• Order of Quantifier Appearance

$\forall x \forall y F \equiv \forall y \forall x F$
$\exists x \exists y F \equiv \exists y \exists x F$
} Same quantifiers, order doesn't matter

$\forall x \exists y F \not\equiv \exists y \forall x F$ - Different quantifier, order matters

$\exists y \forall x F \vDash \forall x \exists y F$
but
$\forall x \exists y F \not\vDash \exists y \forall x F$
} Because $\forall x$ could be "empty"

• Distribution of Quantifiers

$\forall x (P(x) \wedge Q(x)) \equiv \forall x P(x) \wedge \forall x Q(x)$

$\exists x (P(x) \vee Q(x)) \equiv \exists x P(x) \vee \exists x Q(x)$

$\forall$ distributes over $\wedge$ $\quad$ $\exists$ distributes over $\vee$

BUT...

$\forall x (P(x) \vee Q(x)) \vDash \forall x P(x) \vee \forall x Q(x)$ but not conversely!
$\exists x (P(x) \wedge Q(x)) \vDash \exists x P(x) \wedge \exists x Q(x)$ but not conversely!

• Rename Rule

$\exists x F \equiv \exists y F(x/y)$ if y is a new variable name

• Equivalencies w/ Restrictions

- If x does not appear in F, or is bound by another quantifier...

Simplification: $\forall x F \equiv \exists x F \equiv F$

Disjunction: $\forall x(F \vee G(x)) \equiv F \vee \forall x G(x)$
$\exists x(F \vee G(x)) \equiv F \vee \exists x G(x)$

Conjunction: $\forall x(F \wedge G(x)) \equiv F \wedge \forall x G(x)$
$\exists x(F \wedge G(x)) \equiv F \wedge \exists x G(x)$

Implication: $\forall x(F \to G(x)) \equiv F \to \forall x G(x)$
$\exists x(F \to G(x)) \equiv F \to \exists x G(x)$

$\forall x(F(x) \to G) \equiv \exists x F(x) \to G$
$\exists x(F(x) \to G) \equiv \forall x F(x) \to G$

Note the quantifier swap when the symbol with a variable comes first.

• ND With Pred. Logic

- All rules apply as before, plus...

∀-inst: $\dfrac{\forall x\, F(x)}{F(a)}$ where a is arbitrary

∀-gen: $\dfrac{F(a)}{\forall x\, F(x)}$ where a is arbitrary

∃-inst: $\dfrac{\exists x\, F(x)}{F(a)}$ where a is a new variable

∃-gen: $\dfrac{F(a)}{\exists x\, F(x)}$ where a did not come from universal instantiation

Programming With Logic

• Terms

- A clause is a finite disjunction of one or more literals
- A horn clause is a clause with at most one +ve literal
- A definite clause is a clause with exactly one +ve literal

• Making Rules from Definite Clauses

$$F \vee \neg F_1 \vee \neg F_2 \vee \cdots \vee \neg F_n \equiv (F_1 \wedge F_2 \wedge \cdots \wedge F_n) \rightarrow F$$

- F: one +ve
- $\neg F_1 \vee \neg F_2 \vee \cdots \vee \neg F_n$: zero or more -ve
- $(F_1 \wedge F_2 \wedge \cdots \wedge F_n)$: Body
- F: Head

• Definite Clause Programming

- Body of a rule may also look like: $F_1, F_2, \cdots, F_n \rightarrow F$
- Look for rules with a matching head, replace with body.
- Example:
  - Rules: $\rightarrow P$, $P \rightarrow Q$, $P \rightarrow R$, $Q \rightarrow S$, $S, R \rightarrow T$
  - Question: $?T$, $?S, R$, $?Q, R$, $?P, R$, $?R$, $?P$, $\square$

Leads to empty body on $\rightarrow P$, so success ($\square$) or use ■ for fail.

• Derivation Tree.

P_1 Rules: $\rightarrow P$

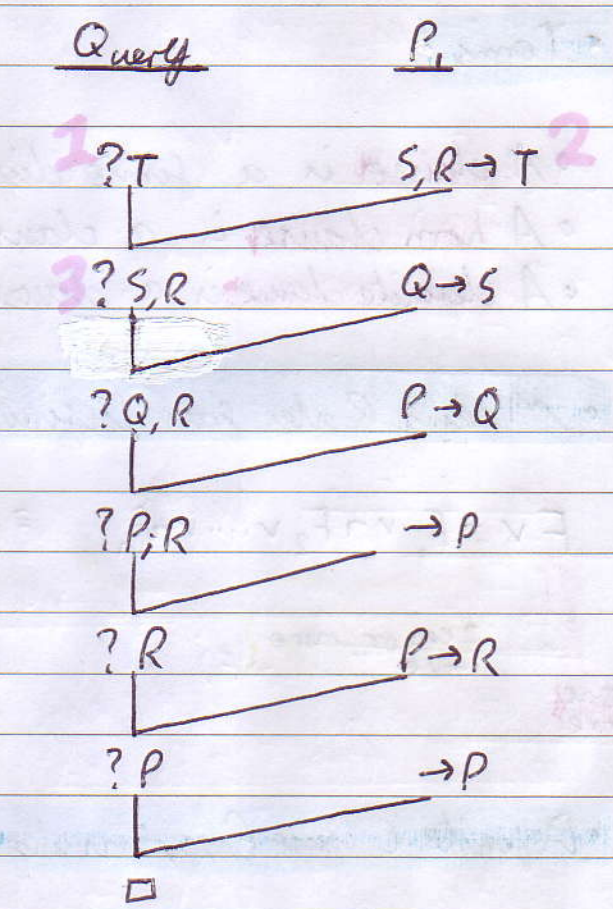
$P \rightarrow Q$

$P \rightarrow R$

$Q \rightarrow S$

$S, R \rightarrow T$

From 1, apply 2 to get 3



Prenex Normal Form (PNF)

- A formula is in PNF if it starts with 0 or more quantifiers and is followed by a quantifier-free formula.

- Eg. $\forall x \exists y \; P(x) \wedge (Q(y) \vee \neg R(x,y))$
  Prefix Matrix

• Conversion Method

1) Remove $\rightarrow$ and $\leftrightarrow$
2) Move $\neg$ inwards to put $\neg$ only by atoms
} Use equivalencies
3) Standardise variables
4) Move quantifiers to front (preserve order)

- Standardise Variables

° Rename all variables that have different scope to have distinct names.

- Move Quantifiers

° $F \wedge Q x\, G \equiv Q x (F \wedge G)$ $\quad Q \in \{\forall, \exists\}$
° $F \vee Q x\, G \equiv Q x (F \vee G)$

° $Q x\, F \wedge Q y\, G \equiv Q x\, Q y (F \wedge G)$
° $Q x\, F \vee Q y\, G \equiv Q x\, Q y (F \vee G)$

• Horn / Definite Clauses

- A horn clause exists in PNF if...
  - There is no more than one +ve literal
  - All quantifiers are universal
- A definite clause is a horn clause with exactly one +ve literal
- A definite clause can become a rule:

$$\forall x \forall y \forall z \neg F(x) \vee \neg F(y) \vee F(z)$$

$$\equiv \forall x \forall y \forall z F(x) \wedge F(y) \rightarrow F(z)$$

$$\equiv \forall x \forall y \forall z F(x), F(y) \rightarrow F(z)$$

Body — Head

• First Order Definite Clause Programming

- A query to a program must be in the form

$$\exists_1 \cdots \exists_n F$$

where F is a conjunction of +ve atoms and $\exists_1 \cdots \exists_n$ quantify variables in F.

- Programs execute as normal, but variables may need to be renamed to match.

• Substitutions and Unifiers

- A **substitution** S is a finite set of pairs $\{(x_1/t_1), \dots, (x_n/t_n)\}$
  - $x_1 \dots x_n$ are **distinct variables**
  - $t_1 \dots t_n$ are **terms** (variables, constants or functions applied to terms)

!! Constants can't be substituted !!

- $S(F)$ is the resulting formula of S applied to F.
- Formulae F and G are **unified** if $S(F) \equiv S(G)$
- The **Most General Unifier** is the substitution that makes no unnecessary substitutions
  - If S_{MGU} is the MGU then for all other unifiers S_x there is some S_n such that $S_x = S_n[S_{MGU}(\dots)]$

Programming With Logic - Part 2

• Definite Clause Programming - Formal Definition

- **Input:** a program P of definite rules and a query:
  $Q = \exists x_1 \cdots \exists x_m \; a_1 \wedge \cdots \wedge a_n$
  - $\wedge$ can be swapped for ,
  - We can drop $\exists x_1 \cdots \exists x_m$ and assume all variables in $a_1 \cdots a_n$ are existentially qualified.

- **Output:** if $Q = a_1 \cdots a_n$ is a logical consequence of the program P then output Yes and substitution of variables in Q, else output No.

- **Method:**
  - 1. Initialise progress = true
  - 2. While Query ≠ empty and progress = true...
    - 2.1 Choose atom a_i from $Q = a_1 \cdots a_n$
    - 2.2 Choose rule $\beta_1, \cdots, \beta_m \rightarrow g$ such that g and a_i unify
      - 2.2.1 If necessary, rename variables in $\beta_1 \cdots \beta_n \rightarrow g$.
      - 2.2.2 Find mgu, S, for a_i and g
      - 2.2.3 Replace a_i (in Q) by $\beta_1 \cdots \beta_m$
      - 2.2.4 Apply S to new query $a_1, \cdots, a_{i-1}, \beta_1 \cdots \beta_m, \cdots a_{i+1} \cdots a_n$
    - 2.3 Else progress = false
  - 3. If (Query = empty and progress = true)
    output Yes and substitution of variables in query
    Else
    output No

- Some steps looked at in detail on next page.

- 2.2.1 : Rename variables.
  - If x is a variable in the query atom and the rule, rename x in the rule using a new variable that does not appear in the rule or the query.

- Representing and / or

- P if A and B and C ....
  $\Rightarrow A, B, C, \dots \rightarrow P$

- P if A or B or C ....
  $\Rightarrow A \rightarrow P$
  $B \rightarrow P$
  $C \rightarrow P$
  ....

- Negation as failure

- Definite rules cannot contain negated atoms, so things like $\neg guilty(x) \rightarrow innocent(x)$ are impossible

- Closed World Assumption
  - The CWA is the assumption that what cannot currently be shown to be true is false
  - We can use the symbol "not", meaning "is not known to be true".
    - "not x" is proved if there is no successful derivation tree
    - "$\neg x$" is proved if x can be explicitly proven to be false.