Logic

Paolo Bettelini

Contents

1 Boolean Algebra

$$
\begin{aligned}
x \vee 0 &= x \\
x \wedge 0 &= 0 \\
x \vee 1 &= 1 \\
x \wedge 1 &= x \\
x \vee x &= x \\
x \wedge x &= x \\
x \wedge (x \vee y) &= x \\
x \vee (x \wedge y) &= x \\
x \wedge \neg x &= 0 \\
x \vee \neg x &= 1 \\
\neg x \wedge \neg y &= \neg(x \vee y) \\
\neg x \vee \neg y &= \neg(x \wedge y)
\end{aligned}
$$

2 Logical inference

A logical inference is a logical deduction to infer the truth of a statement given a premise.

There are 4 forms of hypothetical syllogisms.

2.1 Modus Ponens

Modus Ponens or *affirming the antecedent* is a valid form hypothetical syllogism.

$$\frac{P \implies Q \qquad P}{Q}$$

If P implies Q and P is true, then Q is also true.

2.2 Modus Tollens

Modus Tollens or *denying the consequent* is a valid form hypothetical syllogism.

$$\frac{P \implies Q \qquad \neg Q}{\neg P}$$

If P implies Q and Q is false, then P is also false.

2.3 Fallacy of affirming the consequent

Affirming the consequent is an invalid form hypothetical syllogism.

$$\frac{P \implies Q \qquad Q}{P}$$

If P implies Q and Q is true, then P is also true.

2.4 Fallacy of denying the antecedent

Denying the antecedent is an invalid form hypothetical syllogism.

$$\frac{P \implies Q \qquad \neg P}{\neg Q}$$

If P implies Q and P is false, then Q is also false.

3 Necessity and sufficiency

3.1 Sufficiency

Given two statements P and Q where $P \implies Q$, P suffices for Q to be true.

3.2 Necessity

Given two statements P and Q where $P \implies Q$, Q is a necessity for P to be true ($Q \impliedby P$), but Q does not necessarily imply Q.

3.3 Biconditional logical connective

A biconditional logical connective (written as *iff* or *xnor*) is the relation of equivalence between two statements P and Q. The relation $P \iff Q$ is both a sufficient condition and a necessary condition.

$$P \iff Q \equiv (P \implies Q) \wedge (P \impliedby Q)$$

4 Induction

Induction can be used to prove a statement in the form $P(n)$ for all $n \in \mathbb{N}$. For induction to work one must need to prove the base base, which is usually $P(0)$, but other starting points can be used to prove the statement from that point on.

Weak and strong induction are equivalent.

4.1 Weak induction

Proving that the base case $P(0)$ is true, along with proving the induction step $P(n) \implies P(n+1)$, implies $P(n)$ for all $n \geq 0$. In second-order logic

$$\forall P\,(P(0) \land \forall n\,(P(n) \implies P(n+1)) \implies \forall n\,(P(n)))$$

4.2 Strong induction

Proving that the base case $P(k)$ for $k < m$ is true, along with proving the induction step $P(m)$, implies $P(n)$ for all n.

The base case here is when we need to prove $P(m)$ for $m = 0$.

5 Propositional Logic

5.1 Propositional variable

A *propositional variable* is an input boolean variable. A propositional variable represents the value of a proposition (E:g. *it is snowy today*).

5.2 Connectives

Propositional variables can be connected using *connectives*. They usually are $\land$, $\lor$, $\neg$ or $\implies$. These connectives are not independent and could be defined in terms of the others. Terms like $\land$, $\lor$ and $\neg$ could also be defined as a composition of a single connective.

5.3 k-ary Boolean function

A k-ary Boolean function is a mapping from $\{T, F\}^k \to \{T, F\}$

5.4 0-ary Boolean function

The 0-ary Boolean function are the *verum* ($\top$) and *falsum* ($\bot$) connectives. The represent respectively the True value and the False value.

5.5 Propositional formula

A *propositional formula* is a formula which has a unique truth value given all variables.

The set of all formulas is countable.

5.6 Truth assignment

A *truth assignment* is a function which maps a set of propositional variables $V = \{p_1, p_2, \ldots, p_n\}$ to a boolean value

$$\tau : V \to \{T, F\}$$

A formula A involving the variables $V = \{p_1, p_2, \ldots, p_n\}$ defines a k-ary boolean function $f_A(x_1, x_2, \ldots, x_n)$ where $x_n = \tau(p_n)$.

5.7 Tautology

A propositional formula A is a *tautology* $\vDash A$ if its k-ary boolean function f_A is always T.
Otherwise, we say $\nvDash A$.

5.8 Satisfiability

A propositional formula A is *satisfiable* if f_A is T for some input.

If Γ is a <u>set</u> of propositional formulas, Γ is satisfiable if there are some assignments to satisfy all its members.

$\Gamma \vDash A$ (tautologically implies A) if every truth assignment satisfying Γ also satisfies A.

5.9 Substitution

A *substitution* σ is a mapping from a set of propositional variables to the set of propositional formulas. If A is a propositional formula, $A\sigma$ is equal to the formula obtained by simultaneously replacing each variable appearing in A by its image under σ.

5.10 Propositional Proof System

A *Propositional Proof System* $\mathcal{F}$ has every substitution into the axioms scheme as his axioms and a set of inference rules.
If A has an $\mathcal{F}$-proof, then $\vdash A$. If the proof needs extra hypothesis Γ (which may not be tautologies), then $\Gamma \vdash A$.

5.11 Soundness $\mathcal{F}$

$\mathcal{F}$ is sound iff every $\mathcal{F}$-formula is logically valid with respect to the semantics of the system.

5.12 Completeness $\mathcal{F}$

$\mathcal{F}$ is complete iff it can prove any valid formula, meaning that the semantic noion of validity and the syntactic notion of provability coincide, and a formula is valid iff it is has an $\mathcal{F}$-proof.