Introductory Logic

Lecture 2: Propositional Logic

Alan Mycroft

Computer Laboratory, University of Cambridge, UK
http://www.cl.cam.ac.uk/˜am21

MPhil in ACS – 2011/12

Lecture Outline

- Forms of Logic
- Propositional (or Sentential) Logic.
    - Wffs, and the Computer Science view.
    - Valuation of a formula, Truth tables
    - Satisfiability, Tautology
    - Effectiveness, Feasibility.
    - Compactness Theorem.

Forms of Logic

There are many formulation of logic; all model what we think of as logic, but some are more sophisticated than others:

- Propositional (or Sentential) Logic. No variables.
- Predicate Logic. Adds variables, $\forall$, $\exists$. Role of equality.
- Modal Logic. Adds a notion of modality – e.g. temporal logics in which we can talk about time and express statements like "once A has become true then it remains true".
- Intuitionistic Logic. Disallows reasoning based on axioms such as "$A \vee \neg A$" (justification: Gödel and others).

We start with the simplest form.

Syntax 1

We assume a (countable) set $\mathcal{A} = \{A_1, A_2, \ldots\}$ of *propositional variables*. The *logical connectives* are $\{\wedge, \vee, \neg, \rightarrow, \leftrightarrow\}$.
The set $\overline{\mathcal{A}}$ of *well-formed formulae (wffs)*, ranged over by σ, is the smallest set such that:

- $\mathcal{A} \subseteq \overline{\mathcal{A}}$
- whenever $\sigma \in \overline{\mathcal{A}}$ then $(\neg\sigma) \in \overline{\mathcal{A}}$
- whenever $\sigma, \sigma' \in \overline{\mathcal{A}}$ then $(\sigma \wedge \sigma') \in \overline{\mathcal{A}}$
- ditto for $\vee, \rightarrow, \leftrightarrow$.

Note that this is an inductive definition of set $\overline{\mathcal{A}}$.

Formally all wffs are fully bracketed, but we elide them for humans: e.g. $\neg A \vee B \wedge C$.
['$\neg$' binds tightest, then '$\wedge$', then '$\vee$'.]

Syntax 2

Computer Scientists have an additional formalism to specify inductively-defined sets like that of wffs – we write BNF:

$$\sigma ::= \boldsymbol{A} \mid \sigma \wedge \sigma' \mid \sigma \vee \sigma' \mid \neg\sigma \mid \sigma \rightarrow \sigma' \mid \sigma \leftrightarrow \sigma'$$

Seen as a grammar on *strings* this is ambiguous, but seen as a grammar on *trees* then this is fine.

Semantics

How do we determine when a wff is true, or false?
For propositional variables we need a *truth assignment* (or valuation) $v : \mathcal{A} \longrightarrow \mathbb{B}$ to tell us.
Don't confuse '$\longrightarrow$' (function space) and '$\rightarrow$' (implication in the logic).

We extend v to all wffs (not just propositional variables) with a function $\overline{v} : \overline{\mathcal{A}} \longrightarrow \mathbb{B}$ using truth tables:

$$
\begin{aligned}
\overline{v}(A) &= v(A) \\
\overline{v}(\sigma \wedge \sigma') &= \textit{true} \text{ if } \overline{v}(\sigma) = \textit{true} \text{ and } \overline{v}(\sigma') = \textit{true} \\
&= \textit{false} \text{ otherwise} \\
\overline{v}(\sigma \vee \sigma') &= \text{see over}
\end{aligned}
$$

Semantics 2

$$
\begin{array}{rcl}
\overline{v}(A) & = & v(A) \\
\overline{v}(\sigma \wedge \sigma') & = & \textit{true} \text{ if } \overline{v}(\sigma) = \textit{true} \text{ and } \overline{v}(\sigma') = \textit{true} \\
 & = & \textit{false} \text{ otherwise} \\
\overline{v}(\sigma \vee \sigma') & = & \textit{true} \text{ if } \overline{v}(\sigma) = \textit{true} \text{ or } \overline{v}(\sigma') = \textit{true} \\
 & = & \textit{false} \text{ otherwise} \\
\overline{v}(\neg\sigma) & = & \textit{true} \text{ if } \overline{v}(\sigma) = \textit{false} \\
 & = & \textit{true} \text{ otherwise} \\
\overline{v}(\sigma \rightarrow \sigma') & = & \textit{true} \text{ if } \overline{v}(\sigma) = \textit{false} \text{ or } \overline{v}(\sigma') = \textit{true} \\
 & = & \textit{false} \text{ otherwise} \\
\overline{v}(\sigma \leftrightarrow \sigma') & = & \textit{true} \text{ if } \overline{v}(\sigma) = \overline{v}(\sigma') \\
 & = & \textit{false} \text{ otherwise}
\end{array}
$$

Semantics 3

What we're really doing is modelling 'real' 'and', 'or' etc. in the logic. Indeed, if we write $AND : \mathbb{B} \times \mathbb{B} \longrightarrow \mathbb{B}$ (and similarly for the other connectives) then the equations simply mirror our informal understanding of logic within the formal logic:

$$
\begin{array}{rcl}
\overline{v}(A) & = & v(A) \\
\overline{v}(\sigma \wedge \sigma') & = & AND(\overline{v}(\sigma), \overline{v}(\sigma')) \\
\overline{v}(\sigma \vee \sigma') & = & OR(\overline{v}(\sigma), \overline{v}(\sigma')) \\
\overline{v}(\neg\sigma) & = & NOT(\overline{v}(\sigma)) \\
\overline{v}(\sigma \rightarrow \sigma') & = & IMP(\overline{v}(\sigma), \overline{v}(\sigma')) \\
\overline{v}(\sigma \leftrightarrow \sigma') & = & EQV(\overline{v}(\sigma), \overline{v}(\sigma'))
\end{array}
$$

Truth tables

The functions *AND*, *OR* etc. can be written as *truth tables*:

AND	*true*	*false*
true	*true*	*false*
false	*false*	*false*

OR	*true*	*false*
true	*true*	*true*
false	*true*	*false*

NOT	*true*
true	*false*
false	*true*

IMP	*true*	*false*
true	*true*	*false*
false	*true*	*true*

EQV	*true*	*false*
true	*true*	*false*
false	*false*	*true*

Truth tables 2

People who have done Computer Hardware/Digital Electronics have seen this all before.

One particular aspect is that *any* function $\mathbb{B} \times \cdots \times \mathbb{B} \longrightarrow \mathbb{B}$ can be written as a composition of *AND*, *OR* and *NOT*.

We say $\{AND, OR, NOT\}$ is *universal* for boolean functions.

Note that $\{NAND\}$ is also universal where

$$NAND(x, y) = NOT(AND(x, y))$$

as is $\{NOR\}$.

Satisfaction, Tautology

We say, given a wff σ that:

- valuation v *satisfies* σ if $\overline{v}(\sigma) = \mathit{true}$
- σ is *satisfiable* if there is a valuation which satisfies σ
- σ is a *tautology* if every valuation satisfies σ
- σ is *unsatisfiable* if no valuation satisfies σ

Tautologous Implication

A particularly interesting question is when does wff σ 'imply' wff σ'.
We *could* write $\sigma \rightarrow \sigma'$ but we're interested in situations where the LH σ behaves like a theory and the RH σ' behaves like a question (or some hypotheses and a conclusion).

So we write $\sigma \models \tau$ where $\models$ is part of our mathematics, not part of the logic.
In fact we generalise to the form $\Sigma \models \tau$ where Σ is a set of wffs and define $\Sigma \models \tau$ to hold

if whenever a valuation satisfies all $\sigma \in \Sigma$ then it also satisfies τ

[Note the use of 'hold' when we're talking about maths (the meta-level) rather than 'is *true*' which is a value within the logic.]

Tautology revisited

Given wff σ consider $\emptyset \models \sigma$, often written $\models \sigma$.
By vacuous reasoning this holds whenever σ is a tautology.

Some tautologies:

- $A \vee \neg A$ (excluded middle)
- $\neg(A \vee B) \leftrightarrow \neg A \wedge \neg B$ (de Morgan)
- $\neg(A \wedge B) \leftrightarrow \neg A \vee \neg B$ (de Morgan)
- $A \wedge (B \vee C) \leftrightarrow (A \wedge B) \vee (A \wedge C)$ (distributivity)
- $A \vee (B \wedge C) \leftrightarrow (A \vee B) \wedge (A \vee C)$ (distributivity)

Meta-theorem (duality): swapping $\wedge$ and $\vee$ in a tautology only involving $\wedge$, $\vee$, $\neg$, $\leftrightarrow$ gives another tautology.

Computability, Feasibility

- Determining whether a statement (in propositional logic) is satisfiable or a tautology is (effectively) computable, as indeed almost anything else about propositional logic.
- But satisfiablity is NP-complete, often called *infeasible* as the best-known algorithm is exponential in the number of variables in the worst case.
- However there's a growth industry in SAT solvers which get fast results on many cases which arise in practice.

Digression: Truth versus Proof

Intuitively there are two ways to show two expressions are equivalent:

- show they have the same output value for every input value
- do algebraic manipulations on both until they are syntactically equal.

Note that (interpreting 'expression' as 'wff') we have only exploited the former version here – which is easy and computable because there are only a finite number of possible input values.

So we haven't bothered with the latter (which corresponds to proof). But it will rise in prominence when we turn to *predicate calculus* (a.k.a. *first-order logic*) when the range of input values may be infinite.

Compactness

One more tricky question concerns the behaviour of $\Sigma \models \tau$ when Σ is infinite (perhaps not even countable).
Note the the more members we put in Σ the less satisfiable it becomes (and vice versa):

- $\{A\}$ is satisfiable
- $\{\neg A\}$ is satisfiable
- $\{A, \neg A\}$ is not satisfiable

We have seen that sometimes odd things happen when we move to infinite sets, so the question we ask is (*compactness*):

> *Is the behaviour of $\Sigma \models \tau$ explained by the behaviour of $\Sigma' \models \tau$ where Σ' ranges over all finite subsets of Σ.*

We answer the question in the affirmative.

Compactness Theorem

Notation: a set Σ of wffs is satisfiable if there is a truth assignment which satisfies every $\sigma \in \Sigma$.

Theorem (compactness): a set Σ of wffs is satisfiable iff every finite subset $\Sigma_0 \subseteq \Sigma$ is satisfiable.

Equivalent form of compactness: if $\Sigma \models \tau$ then there is a finite $\Sigma_0 \subseteq \Sigma$ such that $\Sigma_0 \models \tau$.

Why is this important? Reading Σ as a set of hypotheses (allowed to be infinite) which imply τ, we want to be able to construct a textual proof (which must be finite and so can't use an infinite number of hypotheses in Σ).