Session Objectives

- Define formulae in propositional calculus
- Write down truth tables for the main logical operators
- Determine whether formulae are logically equivalent

Propositional Logic

Propositional logic allows us to define, check and prove logical arguments. It is concerned with collections of statements and the deduction of new information from existing knowledge. It has applications in

Hardware design Propositional logic allows us to model and reason with the behaviour of gates and circuits.

Argument checking We can use propositional logic to check the soundness of logical argument. The rules of inference determine what can be proved if certain statements are taken to be true.

Automated reasoning We can use propositional logic to automate the process of reasoning (i.e., prove new theorems, or discover new information, in a logically well-founded way).

Propositions: Definition

Informally speaking, a proposition is a complete statement of some alleged fact that must be either true or false. We exclude questions and imperative statements (orders).

> Is anyone awake?
> Go to sleep!

We also exclude statements that refer to themselves or to other statements:

> The statement below is false.
> The statement above is true.

Try working out a consistent set of truth values for this pair of statements.

Examples

It is raining.
Two plus two equals five.
George W. Bush is one of the great intellects of our age.
Wishes are horses.
Beggars will ride.

- In propositional calculus, we represent each of these statements by a single variable (e.g. p_1, p_2, p_3).
- The word variable just denotes a way of referring to propositions. These variables are not part of the logic itself (unlike in predicate calculus).

Compound Propositions

To express logical arguments, we need to be able to combine propositions to form compound propositions. This is done using logical operators or connectives such as ‘and’, ‘or’, ‘implies’ and ‘not’. For example:

$$\neg p_3 = \text{George W. Bush is not one of the great intellects of our age}$$

$$p_4 \rightarrow p_5 = \text{Wishes are horses implies that beggars will ride.}$$

We can define the operators as follows:

$\neg\phi$ is true if ϕ is false, and is false if ϕ is true. (Negation).

$\phi \wedge \psi$ is true if both ϕ and ψ are true and false otherwise. (Logical ‘and’, or conjunction).

$\phi \vee \psi$ is true if either ϕ or ψ is true (including the case where both are true) and is false otherwise. (Logical ‘or’, or disjunction).

$\phi \rightarrow \psi$ is true unless ϕ is true and ψ is false. (Logical implication).

Truth Tables

In a truth table we write down all possible (logical) values that propositions may take and, in the final column, the corresponding value of the logical expression when they are joined by an operator. To make sure you get all the values, think of them as binary numbers with 0 for T and 1 for F.

p	q	$p \wedge q$
T	T	T
T	F	F
F	T	F
F	F	F

Truth table for conjunction (logical and).

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

Truth table for disjunction (logical or).

p	$\neg p$
T	F
F	T

Truth table for negation.

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Truth table for implication.

Semantics of Operators

- These interpretations are broadly in line with our usual expectations, except, perhaps, for implication.
- The problem with $\rightarrow$ comes about because the English term ‘implies’ (or ‘if . . . then’) has a causal connotation which is not present in the logical definition. We should not expect that $p \rightarrow q$ means ‘p implies q’ in the ordinary sense.
- For example, if p is the statement $5+4=9$ and q is the statement ‘Paris is the capital of France’, then $p \rightarrow q$, because p is false and q is true; however, we would hardly conclude that q was true because of p.

Formulae

A formula (or 'well-formed formula' wff) is defined recursively:

- Every propositional variable is a formula. The symbol $\perp$ (which means a contradiction) is a formula.
- If ϕ and ψ are formulae, then so are $(\neg\phi)$, $(\phi \wedge \psi)$, $(\phi \vee \psi)$ and $(\phi \rightarrow \psi)$.

A string of symbols is a wff if and only if it can be built up by repeated application of the above two rules. To improve readability (by reducing the number of brackets), we normally make use of precedence rules:

1. not $\neg$;
2. and $\wedge$, or $\vee$;
3. implies $\rightarrow$.

Exercise

Which of the following expressions are wffs?

1. $\neg p \rightarrow (q \vee r)$
2. $(\neg p \rightarrow q) \rightarrow ((\neg p \rightarrow \neg q) \rightarrow p)$.
3. $p \vee q \neg$
4. $(p \wedge (q \rightarrow r)) \vee ((\neg p) \wedge (r \rightarrow q))$

Solution

Which of the following expressions are wffs?

1. $\neg p \longrightarrow (q \vee r)$. Yes
2. $(\neg p \longrightarrow q) \longrightarrow ((\neg p \longrightarrow \neg q) \longrightarrow p)$. Yes
3. $p \vee q \neg$. No
4. $(p \wedge (q \longrightarrow r)) \vee ((\neg p) \wedge (r \longrightarrow q))$. Yes

Logical Operators in Ada

The Boolean type is a predefined enumeration type:

```
type Boolean is (False, True);
```

The Boolean operators are defined as follows:

`NOT` This unary operator changes `True` to `False` and vice versa.

`AND` This binary operator is defined by the same truth table as $\wedge$.

`OR` This binary operator is defined by the same truth table as $\vee$.

`XOR` This is a binary operator. The result is `True` if exactly one operand is `True` and is `false` if either both operands are `True` or are `False`.

These operators may also be applied to Boolean arrays (of propositional variables). For binary operators, the two operands must have the same number of components.

Truth of Formulae

- A formula ϕ which is true under all possible assignments of truth values to its propositional variables is called a tautology, and we write $\models \phi$. An example of a tautology is $p \vee \neg p$. Such a formula is true because of its structure rather than because of the truth or otherwise of its component variables.
- A formula which is false under all assignments is called a contradiction; for example, $p \wedge \neg p$.
- A formula is satisfiable if there is at least one assignment that can make it true; so a tautology is satisfiable, and a contradiction is not.

Logical Equivalences

- Two formulae f and g are logically equivalent if they have the same truth tables. This happens if they have the same truth values for all possible truth values of their constituent propositions. We can write this as $f \leftrightarrow g$.
- If S is a set of formulae and f is a single formula, then f is a logical consequence of S, written $S \models f$ if, for any assignment making all members of S true, f is also true.

Example

The formulae $p \rightarrow q$ and $(\neg p) \vee q$ are logically equivalent. We can prove this by writing down the corresponding truth tables.

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

Truth table for implication.

p	q	$\neg p$	$\neg p \vee q$
T	T	F	T
T	F	F	F
F	T	T	T
F	F	T	T

Truth table for $\neg p \vee q$.

Show that the formulae $p \wedge (q \vee r)$ and $(p \wedge q) \vee (p \wedge r)$ are logically equivalent. (This is called the distributive law). How many variables are there? How many rows does the truth table need?

Solution

p	q	r	$q \vee r$	$p \wedge (q \vee r)$	$p \wedge q$	$p \wedge r$	$(p \wedge q) \vee (p \wedge r)$
T	T	T	T	T	T	T	T
T	T	F	T	T	T	F	T
T	F	T	T	T	F	T	T
T	F	F	F	F	F	F	F
F	T	T	T	F	F	F	F
F	T	F	T	F	F	F	F
F	F	T	T	F	F	F	F
F	F	F	F	F	F	F	F

Session Objectives

- Define formulae in propositional calculus
- Write down truth tables for the main logical operators
- Determine whether formulae are logically equivalent