MAE113 Discrete Techniques for Computing

Prof. Wilfrid Hodges
Mathematics room 156
w.hodges@qmul.ac.uk

Course web site
http://www.maths.qmul.ac.uk/∼wilfrid/dtc/

1.1

Logic circuits

Syllabus

Logic: Truth tables, T/F and 1/0 notation,
gates, logic formulas and
their simplifications

1.2

A *logic circuit* is a circuit with

- one or more inputs, each of which is
  either live (1) or dead (0) at any given time;
- an output, which is either live (1) or dead (0)
  depending on which of the inputs are live.

We use variables p, q, r etc. for the inputs and output.
A variable whose values are 0 or 1 is called a *boolean*.
We often read 1 as T (for True) and 0 as F (for False).

1.3

George Boole
introduced boolean variables
around 1850

1.4

Example

The following table describes one logic circuit with inputs p and q and output r:

p	q	r
1	1	1
1	0	0
0	1	0
0	0	0

This is the *truth table* of the logic circuit.

1.5

A logic circuit with this truth table is called an AND gate and is written

The value of r is a function of the values of p and q; for AND gates we write $r = p \wedge q$ (or sometimes pq), and we call $p \wedge q$ the *boolean formula* of the circuit. Think of $\wedge$ as short for And.

1.6

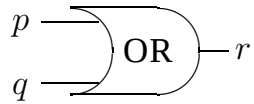
Example

For the following truth table we write $r = p \vee q$, and we read $\vee$ as 'or'.

p	q	r
1	1	1
1	0	1
0	1	1
0	0	0

1.7

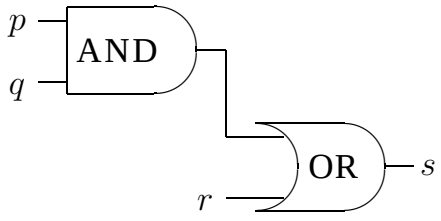
We write



for a logic circuit with boolean formula $p \vee q$, and we call the circuit an OR gate.

1.8

We can join two or more logic circuits to make a new one:



The formula is $s = (p \wedge q) \vee r$.

1.9 The truth table for $(p \wedge q) \vee r$:

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

1.10

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

1.11

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

1.12

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

1.13

Writing truth tables:

- Under the letters, the lefthand column changes slowest, the righthand fastest.
- T comes above F.
  (Some people do the opposite;
  this is OK as long as you are consistent.)
- If there are n letters then you need 2^n rows of Ts and Fs.

1.14

- Put enough brackets in the formula to make clear in what order the columns are calculated.
  The columns inside the most brackets are calculated first.
- The last column calculated (shown with $\Uparrow$)
  is the output column of the truth table.
  It is the output of any logic circuit with this boolean formula.

1.15

Another truth table for you to calculate

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

1.16

The truth tables for

$$(p \wedge q) \vee r,$$
$$(p \vee r) \wedge (q \vee r)$$

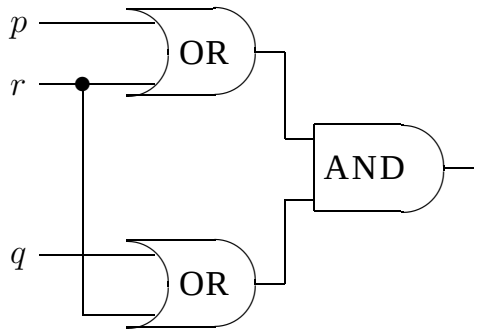
have the same output column.

So we say that these two boolean formulas are *equivalent*,
and that they *have the same truth table*.
We express the same by writing

$$(p \wedge q) \vee r \ \equiv \ (p \vee r) \wedge (q \vee r)$$

1.17

A logic circuit with the formula $(p \vee r) \wedge (q \vee r)$:



1.18

We say that two logic circuits are *equivalent* if they have equivalent boolean formulas.

We can check equivalence of two logic circuits
by finding their boolean formulas
and showing that these formulas are equivalent.

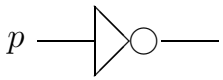
This is often more efficient than checking the 2^n possible input values, where there are n inputs.

1.19

A NOT gate has the truth table

p	
T	F
F	T

and is written



Its formula is written $\neg p$, or sometimes p', and called the *negation* of p. (Don't write $p\neg$ instead of p'.)

1.20

More truth tables to calculate:

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

1.21

This proves that

$$p \wedge q \;\equiv\; \neg(\neg p \vee \neg q).$$

Now putting $\neg$ in front of a formula reverses the Ts and the Fs.
So $\neg\neg$ cancels out, and the equivalence above tells us

$$\neg(p \wedge q) \;\equiv\; \neg p \vee \neg q.$$

These equivalences are known as *De Morgan's laws*.

1.22

There are two more De Morgan's laws:

$$p \vee q \;\equiv\; \neg(\neg p \wedge \neg q),$$
$$\neg(p \vee q) \;\equiv\; \neg p \wedge \neg q.$$

1.23

Two more boolean laws:

$$p \wedge (q \wedge r) \;\equiv\; (p \wedge q) \wedge r,$$
$$p \vee (q \vee r) \;\equiv\; (p \vee q) \vee r.$$

One can check these by truth tables.

Because of them we often write for example

$$p \wedge q \wedge r \wedge s \quad (\text{or } pqrs)$$

without brackets. A formula consisting of smaller formulas joined by $\wedge$ is called a *conjunction*.

1.24

For a similar reason we write we often write for example

$$p \vee q \vee r \vee s$$

without brackets.

This formula is called the *disjunction* of p, q, r and s.

1.25

Also one can check the equivalences

$$\begin{aligned} p \wedge q &\equiv q \wedge p, \\ p \vee q &\equiv q \vee p. \end{aligned}$$

So in a conjunction or a disjunction of formulas,
the order of the formulas doesn't make any difference to the truth table.

1.26

A *literal* is a single letter or the negation of a single letter.

Example: p, q, $\neg r$, r' are literals.

$p \wedge q$ and q'' are not literals.

A *minterm* is a conjunction of literals.
(NB The Schaum book gets this definition wrong.)

1.27

Suppose we have a truth table for the inputs $p_1, \ldots, p_n$,
and the output is 1 in just one row, say row R.

For each i from 1 to n, put

$$p_i^\star = \begin{cases} p_i & \text{if } p_i \text{ is True in row } R, \\ p_i' & \text{if } p_i \text{ is False in row } R. \end{cases}$$

Then the truth table is the truth table of the minterm

$$p_1^\star \ldots p_n^\star.$$

1.28

Next, suppose the truth table has the same inputs but has 1s in the rows $R_1, \ldots, R_k$.

For each row R_j, let M_j be the minterm we calculated in 1.27 for row R_j.

Then the truth table is the truth table of

$$M_1 \vee \ldots \vee M_k.$$

1.29

Finally suppose the truth table has 0 in every row.

Then it's the truth table of the minterm

$$p_1 p_1'.$$

We've proved:

Every truth table is the truth table of a disjunction of one or more minterms.

1.30

Consequence 1: Every boolean formula is equivalent to a disjunction of one or more minterms.
(A formula that is a disjunction of one or more minterms is sometimes said to be in *disjunctive normal form*.)

Consequence 2: Every logic circuit is equivalent to a logic circuit where all the NOT gates are at the left, the AND gates in the middle and the OR gates at the right.

1.31

Example. Suppose given the truth table

	p	q	r	
(1)	T	T	T	F
(2)	T	T	F	T
(3)	T	F	T	F
(4)	T	F	F	F
(5)	F	T	T	F
(6)	F	T	F	T
(7)	F	F	T	F
(8)	F	F	F	T

1.32

The truth table has T in rows (2), (6), (8).

The minterm for row (2) is pqr'.

The minterm for row (6) is $p'qr'$.

The minterm for row (8) is $p'q'r'$.

So the truth table is the truth table of

$$pqr' \vee p'qr' \vee p'q'r.$$

1.33

One for you to do: find a disjunction of minterms that has the truth table

p	q	
1	1	1
1	0	0
0	1	1
0	0	1

1.34

Another for you: find a disjunction of minterms that has the truth table

p	q	
1	1	1
1	0	0
0	1	0
0	0	1

1.35

We introduce two symbols to represent the truth tables in 1.33 and 1.34,
$\rightarrow$ (called 'arrow' or 'if . . . then') and
$\leftrightarrow$ (called 'double arrow' or 'if and only if').

p	q	$p \rightarrow q$	$p \leftrightarrow q$
1	1	1	1
1	0	0	0
0	1	1	0
0	0	1	1

1.36

The symbols $\rightarrow$ and $\leftrightarrow$ are useful for writing down conditions,
when we use the letters p, q etc. for true or false statements.

E.g. if p says 'Light A is on'
and q says 'The oil needs filling up', then

$$p \rightarrow q$$

expresses 'If light A is on then the oil needs filling up'.

(When boolean formulas are thought of as making statements, we call them *propositions*.)

1.37

Simplification of formulas

It's often important to bring logic circuits or boolean formulas to as simple a form as possible.

There are various techniques.
We consider one that uses the disjunctive normal form.

For more information, including the technique of Karnaugh maps, see Chapter 8 'Simplification of logic circuits' in the Schaum book.

1.38

Consider the formula

$$p'qr \vee p'q'r.$$

Because order doesn't matter in conjunctions, this is equivalent to

$$(p'r)q \vee (p'r)q'.$$

Using truth tables, this is equivalent to

$$(p'r) \wedge (q \vee q').$$

1.39

Now $q \vee q'$ always has the value T, so
we can leave it out and the formula is equivalent to

$$p'r.$$

If two of the minterms are the same except that ONE LETTER, say p, appears as p in one and p' in the other, then we can leave out the letter and so get just one minterm in place of two.
(DON'T USE THIS TO DROP TWO LETTERS AT ONCE.)

1.40

Example (Exam 2003): Simplify the expression

$$pq'r \vee pq'r' \vee pqr \vee p'qr \vee p'q'r'.$$

Solution

$$\begin{aligned} & (pq'r \vee pq'r') \vee (pqr \vee p'qr) \vee p'q'r' \\ \equiv \quad & pq' \vee qr \vee p'q'r'. \end{aligned}$$