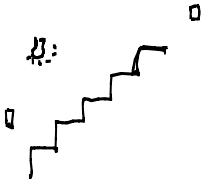
Section 7.2: Logic Networks

April 14, 2008

Abstract

We examine the relationship between the abstract structure of a Boolean algebra and the practical problem of creating logic networks for solving problems. There is a fundamental equivalence between Truth Functions, Boolean Expressions, and Logic Networks which allows us to pass from one to the other.

1 An Example Application, and Fundamental Parallels



Example: Two light switches, one light!

The problem is as follows: A light at the bottom of some stairs is controlled by two light switches, one at each end of the stairs. The two switches should be able to control the light independently. How do we wire the light?

- A Truth Function

S_1	S_2	L
On	On	On
On	Off	Off
Off	On	Off
Off	Off	On

$L(S_1, S_2)$

S_1	S_2	L
1	1	1
1	0	0
0	1	0
0	0	1

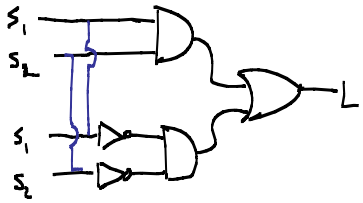
$$L = (S_1 \wedge S_2) \vee (S_1' \wedge S_2')$$

- A Boolean Expression

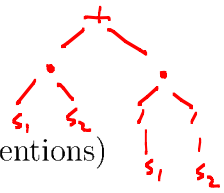
$$L = S_1 \cdot S_2 + S_1' \cdot S_2'$$

$$= S_1 S_2 + (S_1 + S_2)'$$

(builds a slightly tidier Logic network)



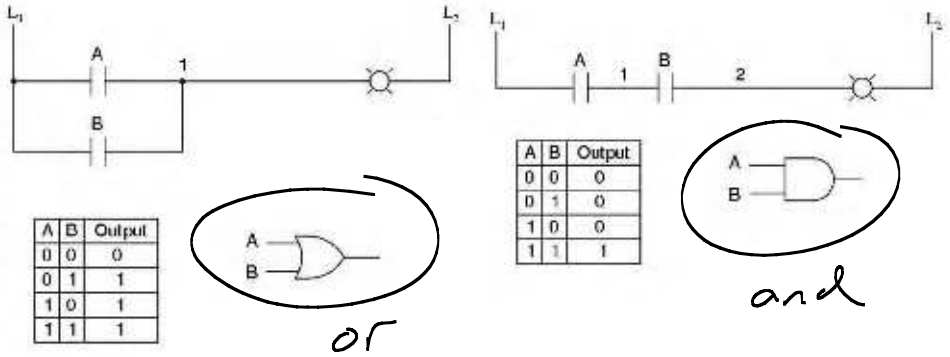
(Minimization)



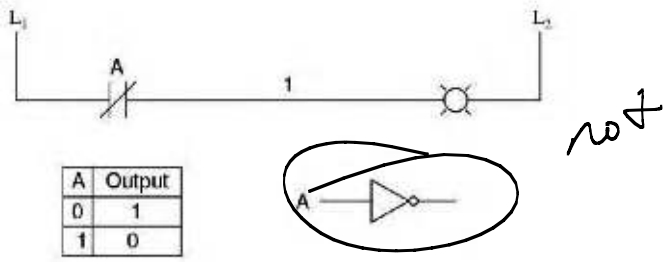
- A Logic Network (Basic Mechanics and Conventions)

  – Input or output lines are not tied together except by passing through gates:

    * OR gate
    * AND gate



    * NOT gate



  – Lines can be split to serve as input to more than one device.
  – There are not loops with output of a gate serving as input to the same gate (feedback).
  – There are no delay elements.

2 Applications

2.1 Converting Truth Tables to Boolean Expressions (Canonical Sum-of-Products Form)

Example: Practice 11, p. 485/558

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
1	1	1	1
1	1	0	0
1	0	1	1
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	1
0	0	0	1

$f(x_1, x_2, x_3) =$

$x_1 x_3$ { $x_1 x_2 x_3 +$; $x_1 x_2' x_3 +$; $x_1 x_2' x_3' +$ } $x_1 x_2'$

$x_1' x_2'$ { $x_1' x_2' x_3 +$; $x_1' x_2' x_3'$

$x_1 x_2 x_3 + x_1 x_2' x_3 = x_1 x_3 (x_2 + x_2') = x_1 x_3 \cdot 1 = x_1 x_3$

$f(x_1, x_2, x_3) = x_1 x_3 + x_1 x_2' + x_1' x_2'$

$= x_1 x_3 + x_2'$

Example: Exercise 10/11, p. 495/568

$f(x_1, x_2, x_3) =$ $x_1 x_2' x_3 +$; $x_1 x_2' x_3' +$ } $x_1 x_2'$

$x_3' (x_1 x_2' + x_1' x_2)$ { $x_1' x_2 x_3'$

$f(x_1, x_2, x_3) = x_1 x_2' + x_1' x_2 x_3'$

(Can we do better?)

2.2 Converting Boolean Expressions to Logic Networks

Example: Exercise 1b, p. 493/566 (Simplify first.)

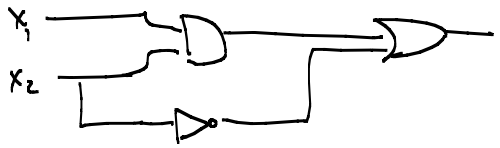
$(x_1 + x_2)' + x_1' x_3$

$= x_1' \cdot x_2' + x_1' x_3$

$= x_1' \cdot (x_2' + x_3)$

2.3 Converting Logic Networks to Truth Functions or Boolean Expressions

Example: Exercise 2, p. 493/567



x_1	x_2	$f(x_1, x_2)$
1	1	1
1	0	1
0	1	0
0	0	1

$$(x_1 \cdot x_2) + x_2' = f(x_1, x_2)$$

$$f(x_1, x_2) = (x_1' \cdot x_2)'$$

$$= x_1 + x_2'$$

$$= x_1 \cdot x_2 + x_2'$$

2.4 Simplifying Canonical Form

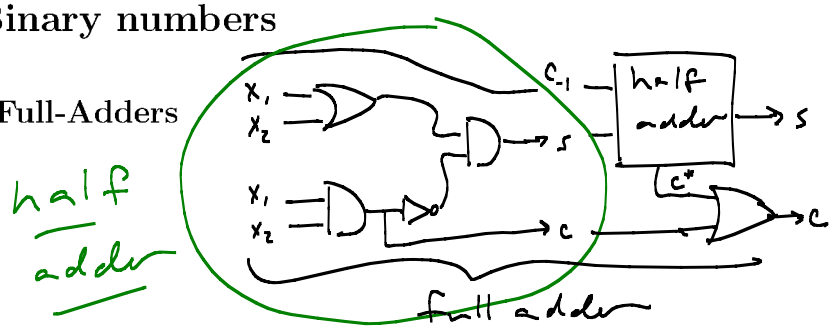
We can use properties of Boolean algebra to simplify the canonical form, creating a much simpler logic network as a result.

Example: Practice 11, p. 485/558

(Already did this above)

2.5 Adding Binary numbers

Half-Adders and Full-Adders



x_1	x_2	s	c
1	1	0	1
1	0	1	0
0	1	1	0
0	0	0	0

Half-Adder: Adds two binary digits.

written: $s = x_1' x_2 + x_1 x_2'$

carried: $c = x_1 x_2$

Note, however, that the half-adder doesn't implement s in this way: instead,

$$s = (x_1 + x_2) \cdot (x_1 x_2)' = (x_1 + x_2)(x_1' + x_2')$$
$$= x_1 x_1' + x_1 x_2' + x_2 x_1' + x_2 x_2' \quad \checkmark$$

Questions:

1 How?

2 Why? Re-use the carry digit calculation!

Full-Adder: Adds two digits plus the carry digit (made up of two half-adders, essentially!).

- Give c_{i-1}, x_i, y_i
- Simply use a half-adder to add the carry digit c_{i-1} to the sum digit s of a half-adder of x_i, y_i, to get s_i.
- To get the carry digit c_i, compare carry digits of both half-adders, to see if either gives a 1 (in which case $c_i = 1$).

Example: Practice 12, p. 490/563