

Boolean Functions and Gates

CS31

Pascal Van Hentenryck



BROWN

Overview

We have seen so far that

- computers deal with words of binary digits
- these words can be used to store a variety of objects (e.g. numbers, characters, ...)

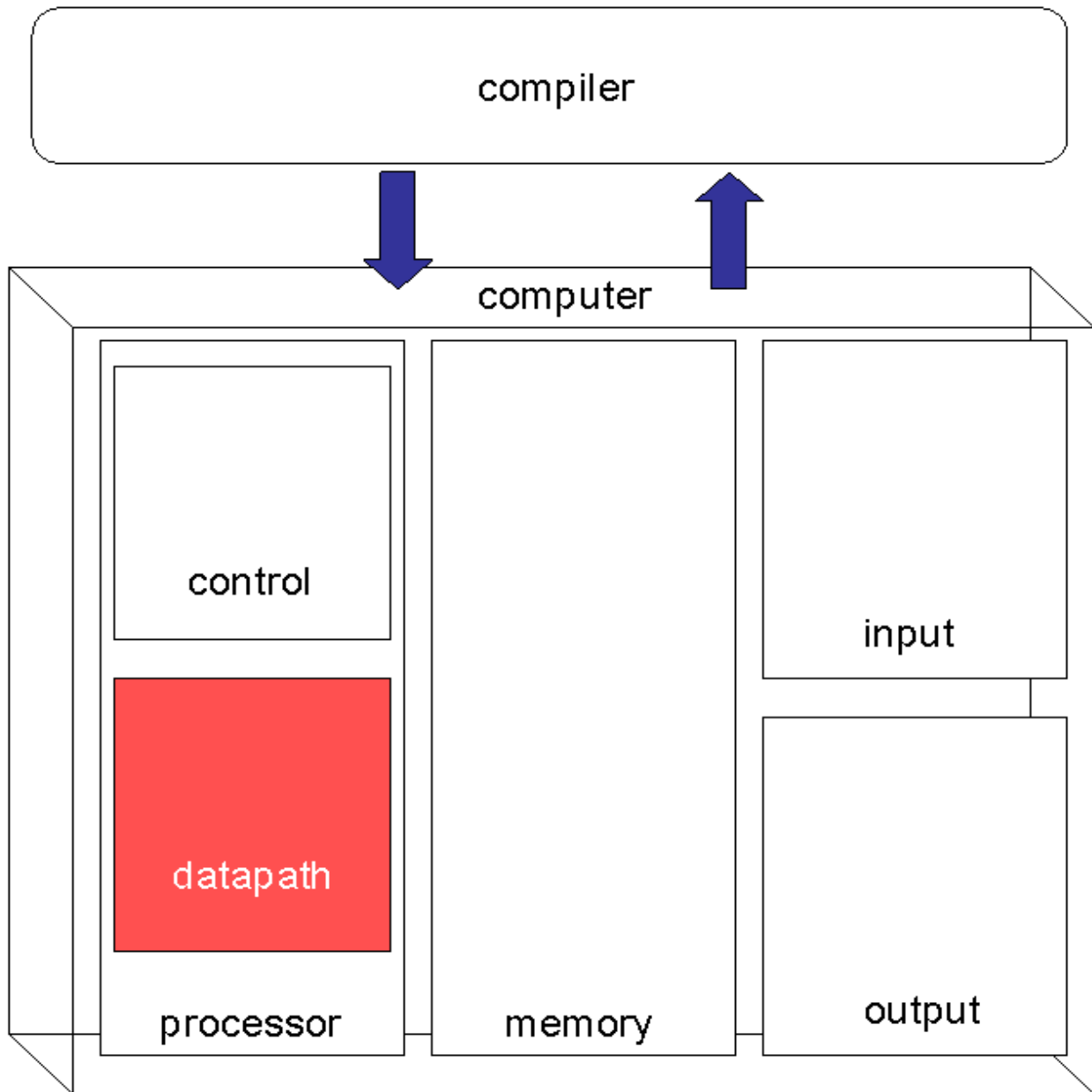
We now study

- how to build circuits to perform operations on these representations

Fundamental Idea

- a new abstraction called combinational device

The Big Picture



Abstraction Hierarchy

Programming Language

Assembly Language

Machine Language

Sequential Circuit

Combinational Circuit

Binary Value

Voltage

Combinational Devices

A combinational device is a circuit having

- some binary input
- some binary output
- a functional specification, stating the values of the output in terms of the inputs
- a timing specification, giving the maximum time necessary to compute the outputs given the inputs

Static Discipline

- This is the guarantee that, given valid inputs, the circuit will deliver valid outputs (after some propagation delay)

This is the simplest tool to build computers

Combinational devices can be combined to produce other combinational devices when:

- the circuit contains no directed cycle

The main goal

Show how to build combinational devices

- for the operations we have seen
- given a set of basic boolean functions

Organization

- Basic Boolean functions
- Truth table
- Properties
- Normal forms

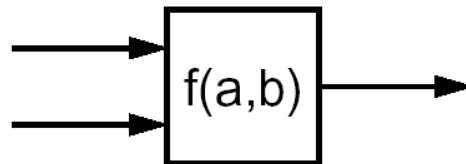
Truth Tables

a	b	c	f(a,b,c)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Truth Tables

Specification of combinational devices

a	b	f(a,b)
0	0	f(0,0)
0	1	f(0,1)
1	0	f(1,0)
1	1	f(1,1)



- lots of ways to choose f (2^4 , to be exact)

a	b	f(a,b)
0	0	1
0	1	0
1	0	0
1	1	1

And

and(a,b) = 1 *if and only if* a = 1 and b = 1

a	b	and (a,b)
0	0	0
0	1	0
1	0	0
1	1	1

Alternative notations for **and**(a,b) include:

- ab (we'll be using this one)
- a·b
- a \wedge b

This is also called the *conjunction* operator.

Or

or(a,b) = 1 if and only if a = 1 or b = 1

a	b	or (a,b)
0	0	0
0	1	1
1	0	1
1	1	1

Alternative notations for **or**(a,b) include:

- $a + b$ (we'll be using this one)
- $a \vee b$

This is also called the *disjunction* operator.

Not

not(a) = 1 *if and only if* a = 0

a	not (a)
0	1
1	0

Alternative notations for **not**(a) include:

- a'
- $\neg a$

This is also called the *negation* operator.

De Morgan's Laws

If you have **and** and **not**, you can create the **or** function; if you have **or** and **not**, you can create the **and** function.

$$a + b = (a'b')'$$

a	b	a'	b'	a'b'	(a'b')'	a + b
0	0	1	1	1	0	0
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	0	1	1

Similarly (try this one out yourself)

$$ab = (a' + b')'$$

Exclusive Or (xor)

switch a



switch b

$$ab' + a'b$$

a	b	ab'	$a'b$	$ab' + a'b$
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

Nand

nand(a,b) = 1 *if and only if* it is not the case that both a = 1 and b = 1

a	b	nand(a,b)
0	0	1
0	1	1
1	0	1
1	1	0

Nand is universal:

$$a' =$$

$$ab =$$

$$=$$

$$a+b=$$

Nand

$\text{nand}(a,b) = 1$ if and only if it is not the case that both $a = 1$ and $b = 1$

a	b	$\text{nand}(a,b)$
0	0	1
0	1	1
1	0	1
1	1	0

Nand is universal:

$$a' = \text{nand}(a,a)$$

$$ab =$$
$$=$$

$$a+b =$$
$$=$$
$$=$$
$$=$$

Nand

$\text{nand}(a,b) = 1$ if and only if it is not the case that both $a = 1$ and $b = 1$

a	b	$\text{nand}(a,b)$
0	0	1
0	1	1
1	0	1
1	1	0

Nand is universal:

$$a' = \text{nand}(a,a)$$

$$\begin{aligned} ab &= \text{nand}(\text{nand}(a,b), \text{nand}(a,b)) \\ &= (\text{nand}(a,b))' \end{aligned}$$

$$a+b=$$

=

=

=

Nand

$\text{nand}(a,b) = 1$ if and only if it is not the case that both $a = 1$ and $b = 1$

a	b	nand(a,b)
0	0	1
0	1	1
1	0	1
1	1	0

Nand is universal:

$$a' = \text{nand}(a,a)$$

$$\begin{aligned} ab &= \text{nand}(\text{nand}(a,b), \text{nand}(a,b)) \\ &= (\text{nand}(a,b))' \end{aligned}$$

$$\begin{aligned} a+b &= \text{nand}(\text{nand}(a,a), \text{nand}(b,b)) \\ &= \text{nand}(a', b') \end{aligned}$$

Associativity

Basic rules

- $A (B C) = (A B) C = A B C$
- $A + (B + C) = (A + B) + C = A + B + C$

Name Conventions

- ABC is called a product
- $A+B+C$ is called a sum

Other Properties

Commutativity

- $A B = B A$
- $A + B = B + A$

Distributivity

- $A (B + C) = A B + A C$
- $A + (B C) = (A + B) (A + C)$

Which rule is not true for integers?

Other Properties

Commutativity

- $A B = B A$
- $A + B = B + A$

Distributivity

- $A (B + C) = A B + A C$
- $A + (B C) = (A + B) (A + C)$

Which rule is not true for integers?

Distributivity:

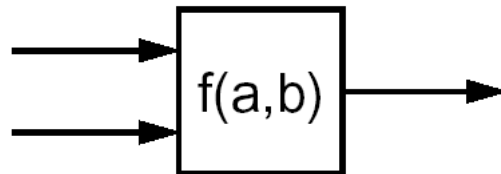
$$2 + (3*3) \quad \boxed{\text{W}} \quad (2+3)(2+3)$$

$$11 \quad \boxed{\text{W}} \quad 25$$

Truth Tables

Specification of combinational devices

a	b	f(a,b)
0	0	f(0,0)
0	1	f(0,1)
1	0	f(1,0)
1	1	f(1,1)



- lots of ways to choose f (2^4 , to be exact)

a	b	f(a,b)
0	0	1
0	1	0
1	0	0
1	1	1

Truth Tables

a	b	c	f(a,b,c)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Sum of Products

Representing a function by a logical expression

a	b	c	f(a,b,c)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

The expression is

$$a'b'c + a'bc + ab'c' + ab'c$$

Basic Idea

Associate a product term with each row with an output at 1; the term has the variable if its value is 1 and the negation of the variable otherwise

Product of Sums

a	b	c	f(a,b,c)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

The expression is

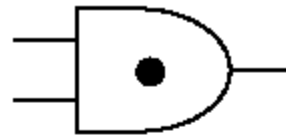
$$(a+b+c)(a+b'+c)(a'+b'+c)(a'+b'+c')$$

Basic Idea

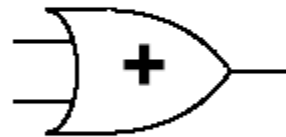
The result is a product term. Each sum in this product negates a row which has a zero output. Associate a sum term with each row with an output at 0; the term has the variable if its value is 0 and the negation of the variable otherwise

Logical Gates

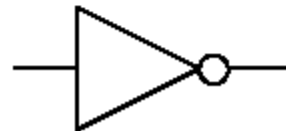
and



or

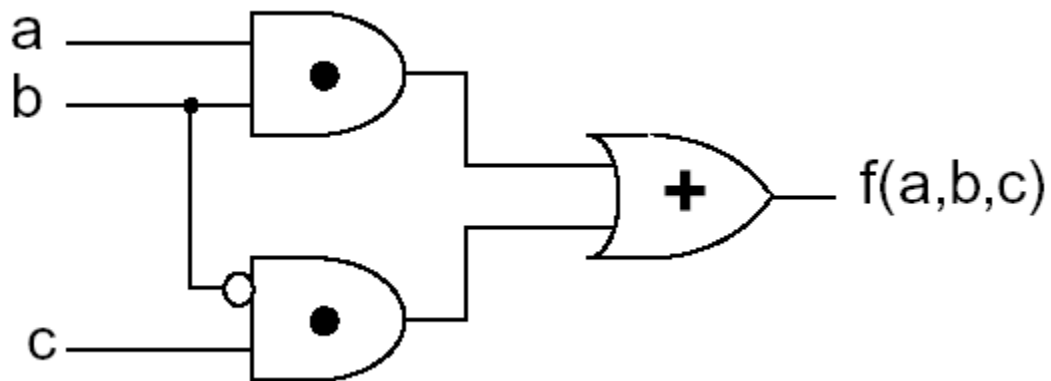


not (inverter)



Combinational Device

$$f(a,b,c) = ab + b'c$$



**We still have a way to go to
get to our Linux boxes...**

...but we'll get there!