Logic and Sets 4

axioms of Boolean algebra

correspondence to set theory

semantic equivalence classes

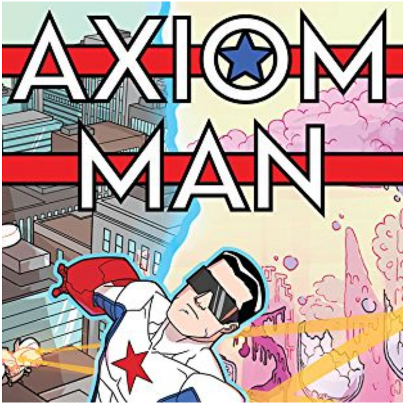
logic circuits

Today

1. Boolean algebra
   - axioms, i.e., *equations* between formulas, for $\equiv$
   - correspondence with set theory
2. Equivalence classes of $\equiv$
3. Logic circuits
   - implementing Boolean functions
   - adding bit strings (of 0's and 1's) in binary arithmetic

Axioms of semantic equivalence

(and its relation with the algebra of sets)



Boolean algebra: Axioms of $\equiv$ for $\wedge$, $\vee$, $\neg$

Commutativity:

$$\phi \vee \psi \equiv \psi \vee \phi$$
$$\phi \wedge \psi \equiv \psi \wedge \phi$$

Idempotence:

$$\phi \vee \phi \equiv \phi$$
$$\phi \wedge \phi \equiv \phi$$

Associativity:

$$\phi \vee (\psi \vee \chi) \equiv (\phi \vee \psi) \vee \chi$$
$$\phi \wedge (\psi \wedge \chi) \equiv (\phi \wedge \psi) \wedge \chi$$

Complement:

$$\phi \vee \neg\phi \equiv \top$$
$$\phi \wedge \neg\phi \equiv \bot$$

Distributivity:

$$\phi \vee (\psi \wedge \chi) \equiv (\phi \vee \psi) \wedge (\phi \vee \chi)$$
$$\phi \wedge (\psi \vee \chi) \equiv (\phi \wedge \psi) \vee (\phi \wedge \chi)$$

De Morgan:

$$\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$$
$$\neg(\phi \wedge \psi) \equiv \neg\phi \vee \neg\psi$$

Identities:

$$\phi \vee \bot \equiv \phi$$
$$\phi \wedge \top \equiv \phi$$

Domination:

$$\phi \vee \top \equiv \top$$
$$\phi \wedge \bot \equiv \bot$$

Involution:

$$\neg\neg\phi \equiv \phi$$

Boolean algebra is sound, complete and irredundant

Propositional logic is a Boolean algebra.

The axioms of Boolean algebra are:

- *Sound* for propositional logic.

  Its axioms equate only semantically equivalent formulas.

- *Complete* for propositional logic.

  All sound equations between formulas can be derived.

- *Irredundant*.

  No axiom can be derived from the other axioms.

Derivations in Boolean algebra

Question

Derive with the axioms of semantic equivalence:

$$(\phi \wedge \psi) \vee \chi \equiv (\phi \vee \chi) \wedge (\psi \vee \chi)$$

$$\begin{aligned}
(\phi \wedge \psi) \vee \chi &\equiv \chi \vee (\phi \wedge \psi) \\
&\equiv (\chi \vee \phi) \wedge (\chi \vee \psi) \\
&\equiv (\phi \vee \chi) \wedge (\psi \vee \chi)
\end{aligned}$$

Example

We derive with the axioms of semantic equivalence:

$$(\phi \to \psi) \wedge (\psi \to \phi) \equiv (\phi \wedge \psi) \vee (\neg\phi \wedge \neg\psi)$$

We also use $\phi \to \psi \equiv \neg\phi \vee \psi$.

Derivations in Boolean algebra

For brevity, most applications of the commutativity axioms are omitted.

$$
\begin{array}{lll}
 & (\phi \to \psi) \ \wedge \ (\psi \to \phi) & \\
\equiv & (\neg\phi \vee \psi) \ \wedge \ (\neg\psi \vee \phi) & (\text{2x } \_ \to \_ \ \equiv \ \neg\_ \vee \_) \\
\equiv & ((\neg\phi \vee \psi) \wedge \neg\psi) \ \vee \ ((\neg\phi \vee \psi) \wedge \phi) & (\text{dist.}) \\
\equiv & ((\neg\phi \wedge \neg\psi) \vee (\psi \wedge \neg\psi)) \ \vee \ ((\neg\phi \wedge \phi) \vee (\psi \wedge \phi)) & (\text{2x dist.}) \\
\equiv & ((\neg\phi \wedge \neg\psi) \vee \bot) \ \vee \ (\bot \vee (\psi \wedge \phi)) & (\text{2x comp.}) \\
\equiv & (\neg\phi \wedge \neg\psi) \ \vee \ (\psi \wedge \phi) & (\text{2x id.}) \\
\equiv & (\phi \wedge \psi) \ \vee \ (\neg\phi \wedge \neg\psi) & (\text{2x comm.})
\end{array}
$$

Correspondence with the algebra of sets

Corresponding symbols:

propositional logic	$\bot$	$\top$	$\vee$	$\wedge$	$\neg$
algebra of sets	$\emptyset$	$\mathbf{U}$	$\cup$	$\cap$	$'$

- $\emptyset$ is the *empty set*
- $\mathbf{U}$ is the *universe* of all elements
- $\cup$ returns the *union* of two sets
- $\cap$ returns the *intersection* of two sets
- $'$ returns the *complement* of a set

Propositional logic coincides with set theory with a universe of a single element.

Set theory is also a Boolean algebra

Commutativity:

$$A \cup B = B \cup A$$
$$A \cap B = B \cap A$$

Associativity:

$$A \cup (B \cup C) = (A \cup B) \cup C$$
$$A \cap (B \cap C) = (A \cap B) \cap C$$

Distributivity:

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$
$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

Identities:

$$A \cup \emptyset = A$$
$$A \cap \mathbf{U} = A$$

Domination:

$$A \cup \mathbf{U} = \mathbf{U}$$
$$A \cap \emptyset = \emptyset$$

Idempotence:

$$A \cup A = A$$
$$A \cap A = A$$

Complement:

$$A \cup A' = \mathbf{U}$$
$$A \cap A' = \emptyset$$

De Morgan:

$$(A \cup B)' = A' \cap B'$$
$$(A \cap B)' = A' \cup B'$$

Involution:

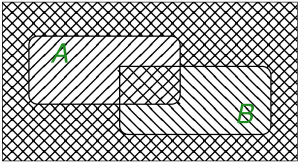
$$(A')' = A$$

Axioms for the algebra of sets

A set equation

From the axioms for sets we can derive:

$$(A' \cup B) \;\cap\; (A \cup B') \;=\; (A \cap B) \;\cup\; (A' \cap B')$$



Difficult? Not at all! It corresponds to

$$(\phi \rightarrow \psi) \;\wedge\; (\psi \rightarrow \phi) \quad\equiv\quad (\phi \wedge \psi) \;\vee\; (\neg\phi \wedge \neg\psi)$$

i.e. $$(\neg\phi \vee \psi) \;\wedge\; (\phi \vee \neg\psi) \quad\equiv\quad (\phi \wedge \psi) \;\vee\; (\neg\phi \wedge \neg\psi)$$

Absorption and set difference

Questions

To which equations for $\equiv$ in propositional logic do the following *absorption* equations for sets correspond?

- $A \cup (A \cap B) \;=\; A$
- $A \cap (A \cup B) \;=\; A$

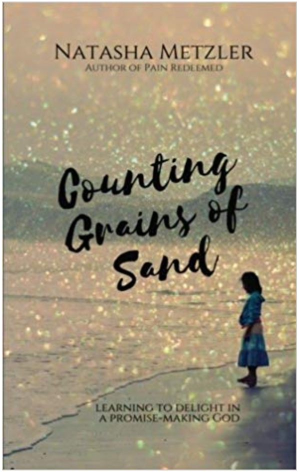
Questions

Express *set difference* $A \setminus B$ in terms of intersection and complement.

Which propositional formula corresponds with $A \setminus B$?

To which equation for $\equiv$ in propositional logic does the equation $A \setminus A = \emptyset$ for sets correspond?

Formula equivalence classes of $\equiv$



Equivalence relation

A binary relation R is an *equivalence relation* if, for all elements x, y, z in its domain, R satisfies the following three properties:

- Reflexive: $x \, R \, x$
- Symmetric: $x \, R \, y$ implies $y \, R \, x$
- Transitive: $x \, R \, y$ and $y \, R \, z$ implies $x \, R \, z$

Equivalence relation R gives rise to *equivalence classes*.

Within an equivalence class, all elements are related by R.

Elements in *different* equivalence classes are *not* related by R.

Semantic equivalence $\equiv$

Exercise 1.5.5 in Huth & Ryan

The relation $\equiv$ is reflexive, symmetric and transitive.

So $\equiv$ is an equivalence relation.

The set of formulas of propositional logic can be partitioned into equivalence classes with respect to $\equiv$.

Equivalence classes of $\equiv$

Within an equivalence class, all formulas are semantically equivalent.

Formulas from different classes are *not* semantically equivalent.

One of the equivalence classes contains all tautologies.

Another class contains all contradictions.

Each equivalence class contains *infinitely* many formulas.

For example: $\phi \equiv \phi \vee \phi \equiv \phi \vee (\phi \wedge \phi) \equiv \cdots$

Equivalence classes of $\equiv$ for one variable p

Question

Describe the equivalence classes of propositional formulas that contain (at most) one variable p.

Answer: There are 4 equivalence classes of formulas with only the variable p, described by representatives:

- $\top$
- $\bot$
- p
- $\neg p$

Number of equivalence classes of $\equiv$

For 1 variable p there are $2^1 = 2$ valuations (mapping p to $\mathtt{T}$ or $\mathtt{F}$).

Each valuation has 2 possible outcomes ($\mathtt{T}$ or $\mathtt{F}$).

Hence there are $2^2 = 4$ equivalence classes.

p	ϕ_1	ϕ_2	ϕ_3	ϕ_4
T	T	T	F	F
F	T	F	T	F

Equivalence classes of $\equiv$ for two variables p and q

$p \to (p \vee q)$	p	$p \to q$	$(p \to q) \to q$		$p \wedge \neg p$
$(p \wedge q) \to p$	$\neg\neg p$	$p \to (p \to q)$	$\neg q \to p$		$q \wedge \neg(p \to q)$
$p \to p$	$(p \to p) \to p$	$\neg(\neg q \wedge p)$	$p \vee q$		$\neg(p \to p)$
$p \vee \neg p$	$\neg p \to p$	$\neg q \to \neg p$	$q \vee p$	11 more classes	$\bot$
$p \to (p \to p)$	$p \wedge p$	$\dots$	$(p \vee q) \wedge (p \to p)$		$\dots$
$\top$	$\dots$	$\dots$	$\dots$		$\dots$
$\dots$	$\dots$	$\dots$	$\dots$		$\dots$
tautologies					*contradictions*

All tautologies are semantically equivalent.

The same holds for all contradictions.

Each variable has 2 possible values ($\mathtt{T}$ or $\mathtt{F}$).

So for 2 variables p, q there are $2^2 = 4$ valuations.

Each valuation has 2 possible outcomes ($\mathtt{T}$ or $\mathtt{F}$).

So there are $2^{2^2} = 2^4 = 16$ equivalence classes.

Number of equivalence classes of $\equiv$

For 2 variables p, q there are 4 valuations.

Each valuation has 2 possible outcomes (T or F).

So there are $2^4 = 16$ equivalence classes.

p	q	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5	ϕ_6	ϕ_7	ϕ_8	ϕ_9	ϕ_{10}	ϕ_{11}	ϕ_{12}	ϕ_{13}	ϕ_{14}	ϕ_{15}	ϕ_{16}
T	T	T	T	T	T	F	T	T	F	T	F	F	T	F	F	F	F
T	F	T	T	T	F	T	T	F	T	F	T	F	F	T	F	F	F
F	T	T	T	F	T	T	F	T	T	F	F	T	F	F	T	F	F
F	F	T	F	T	T	T	F	F	F	T	T	T	F	F	F	T	F

Number of equivalence classes of $\equiv$

Questions

How many *valuations* are there for 3 variables p, q, r ?

How many *equivalence classes* are there for formulas with 3 variables p, q, r ?

How many *valuations* are there for n variables ?

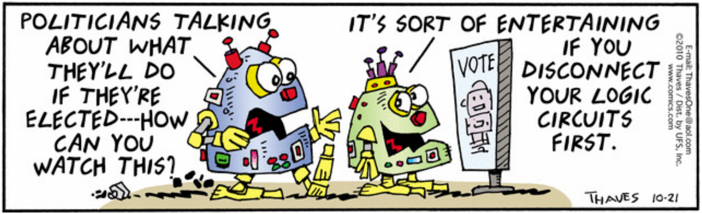
How many *equivalence classes* are there for formulas with n variables ?

Growth of equivalence classes is double exponential

prop. var's	# var's	# lines of truth table	# equivalence classes
p	1	$2^1 = 2$	$2^2 = 4$
p, q	2	$2^2 = 4$	$2^4 = 16$
p, q, r	3	$2^3 = 8$	$2^8 = 256$
p, q, r, s	4	$2^4 = 16$	$2^{16} = 65\,536$
p, q, r, s, t	5	$2^5 = 32$	$2^{32} = 4\,294\,967\,296$
$\dots$	$\dots$	$\dots$	$\dots$
$p_1, \dots, p_n$	n	2^n	$2^{(2^n)}$

With only 9 variables, the number of equivalence classes already far exceeds the number of atoms in the universe.

Logic circuits



Boolean functions

Definition

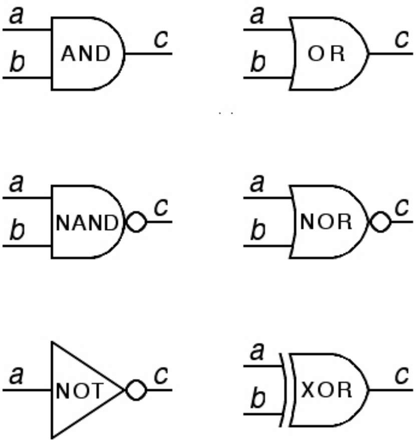
A *Boolean function* maps

- tuples (e.g. pairs, triples) of the truth values 1 and 0
- to the truth values 1 and 0.

Boolean functions can be represented by:

- truth tables (*we write* T *and* F *for* 1 *and* 0)
- formulas of propositional logic (*idem*)
- logic circuits (*CS course Computer Organization*)
- ordered binary decision diagrams (*in lecture 5*)

Electrical engineers' notation for Boolean functions



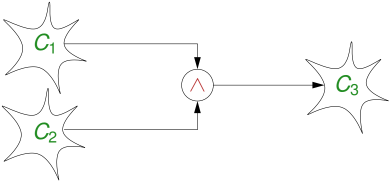
We focus on AND, OR and NOT gates.

(They are functionally complete.)

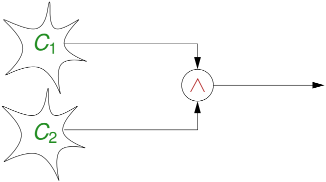
Logic circuits

Logic circuits C_1 and C_2 represent propositional formulas ϕ_1 and ϕ_2.

An AND-gate representing $\phi_1 \wedge \phi_2$ serves as *input* to logic circuit C_3.



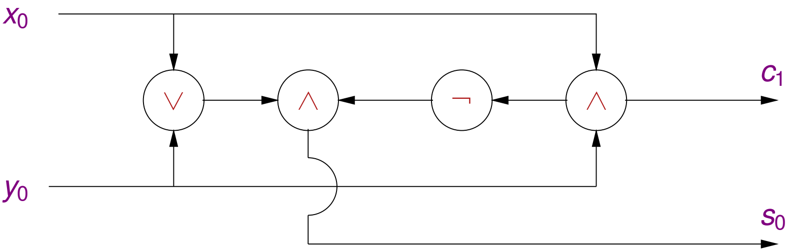
The following logic circuit *outputs* $\phi_1 \wedge \phi_2$:



Question: What are the *inputs* of this entire circuit ?

Implementing Boolean functions by logic circuits

The *logic circuit* below adds two *bits* x_0 and y_0.



It takes as *input* x_0 and y_0.

It *outputs* bits s_0 and c_1 with $x_0 + y_0 = c_1\, s_0$ in binary arithmetic.

Decimal representation of natural numbers

We use 10 as the base for our arithmetic.

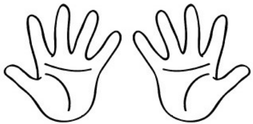
For example, 1,376 means $1{\cdot}10^3 + 3{\cdot}10^2 + 7{\cdot}10^1 + 6{\cdot}10^0$.

In general, each number is written in the form

$$d_{k-1}{\cdot}10^{k-1} + d_{k-2}{\cdot}10^{k-2} + \ldots + d_1{\cdot}10^1 + d_0{\cdot}10^0$$

where $d_0, \ldots, d_{k-1} \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

That we use 10 is rather arbitrary.



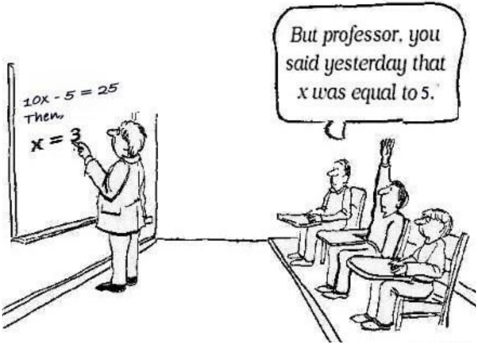
It works for any number greater than 1; in particular, for 2.

Adding two decimal numbers

A trip down memory lane: primary school arithmetic.

Example:

$$\begin{array}{rrl} & 7\,3\,8 & \\ + & 5\,4\,5 & \\ & 1\,0\,1\,0 & \text{(carry)} \\ \hline = & 1\,2\,8\,3 & \text{(result)} \end{array}$$



Binary representation of natural numbers

Natural numbers up to 2^k can be written in binary form as:

$$b_{k-1}\cdot 2^{k-1} + b_{k-2}\cdot 2^{k-2} + \ldots + b_1\cdot 2^1 + b_0\cdot 2^0$$

where $b_0, \ldots, b_{k-1} \in \{0,1\}$.

Example: Let $k = 4$.

decimal	binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

decimal	binary
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

"My date last night was a 10. Of course, I'm using the binary system."

Addition of bits in binary arithmetic

$$0 + 0 = 0 \qquad 1 + 0 = 1 \qquad 1 + 1 = 10 \qquad 1 + 1 + 1 = 11$$

Example 1:

$$
\begin{array}{rrl}
 & 1\,1\,1 & (7) \\
+ & 0\,0\,1 & (1) \\
 & 1\,1\,1\,0 & \text{(carry)} \\
\hline
= & 1\,0\,0\,0 & (\text{result} = \ 8)
\end{array}
$$

Example 2:

$$
\begin{array}{rrl}
 & 1\,0\,1\,1 & (11) \\
+ & 1\,1\,1\,0 & (14) \\
 & 1\,1\,1\,0\,0 & \text{(carry)} \\
\hline
= & 1\,1\,0\,0\,1 & (\text{result} = \ 25)
\end{array}
$$

Use of $\wedge$ in binary addition

We connect propositional logic to binary arithmetic.

1 and 0 represent the truth values T and F, respectively.

For example, the truth table for $\wedge$ becomes:

x	y	$x \wedge y$
1	1	1
1	0	0
0	1	0
0	0	0

$x \wedge y$ equals the *left* bit that results when adding *bits* x and y:

$$1 + 1 = 10 \qquad 1 + 0 = 01 \qquad 0 + 1 = 01 \qquad 0 + 0 = 00$$

Question: Which logical operation corresponds to the *right* bit ?

Use of $\oplus$ in binary addition

Recall that the connective $\oplus$ represents *exclusive or*.

$x \oplus y$ is 1 if exactly one of x and y is 1.

x	y	$x \oplus y$
1	1	0
1	0	1
0	1	1
0	0	0

$x \oplus y$ equals the *right* bit that results when adding *bits* x and y:

$$1 + 1 = 10 \qquad 1 + 0 = 01 \qquad 0 + 1 = 01 \qquad 0 + 0 = 00$$

Addition of two bits captured in logic

Consider *addition* of two bits x_0 and y_0 in binary arithmetic:

$$\begin{array}{rccl} & & x_0 & \\ + & & y_0 & \\ & c_1 & & \text{(carry)} \\ \hline & s_1 & s_0 & \text{(result)} \end{array}$$

We observed that:

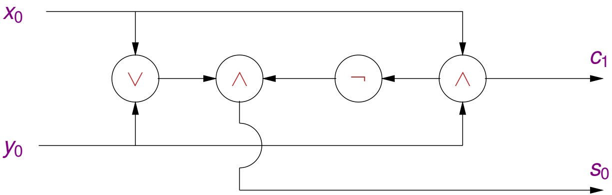
- $s_0 = x_0 \oplus y_0$
- $c_1 = s_1 = x_0 \wedge y_0$

Logic circuit for adding two bits

For adding two bits x_0 and y_0, we found the following logical expressions for the carry and sum bits c_1 and s_0:

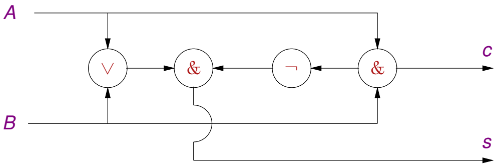
- $s_0 = x_0 \oplus y_0$
- $c_1 = x_0 \wedge y_0$

We express this as a logic circuit with AND, OR and NOT gates, using $x_0 \oplus y_0 \equiv (x_0 \vee y_0) \wedge \neg(x_0 \wedge y_0)$:



Half adder in Mendelson

Mendelson calls this circuit the half adder.



Note the different notations in Mendelson:

- & for conjunction $\wedge$
  (and $+$ for exclusive or $\oplus$)
- input bits denoted as A, B
- carrier and sum as c and s, respectively

Question: Give a circuit for the half adder using an AND and an XOR gate.

Application of two XOR's

Recall that $\oplus$ is *associative*.

x	y	z	$x \oplus y$	$x \oplus y \oplus z$
1	1	1	0	1
1	1	0	0	0
1	0	1	1	0
1	0	0	1	1
0	1	1	1	0
0	1	0	1	1
0	0	1	0	1
0	0	0	0	0

$x \oplus y \oplus z$ is 1 if and only if an *odd* number of its arguments is 1.

Addition of two pairs of bits captured in logic

Consider addition of two *pairs* of bits in binary arithmetic.

$$\begin{array}{rccccl} & & & x_1 & x_0 & \\ + & & & y_1 & y_0 & \\ & & c_2 & c_1 & & \text{(carry)} \\ \hline & & s_2 & s_1 & s_0 & \text{(result)} \end{array}$$

- $s_0 = x_0 \oplus y_0$

  $c_1 = x_0 \wedge y_0$

- $s_1 = x_1 \oplus y_1 \oplus c_1$

  $c_2 = (x_1 \wedge y_1) \vee (c_1 \wedge (x_1 \oplus y_1))$

- $s_2 = c_2$

Addition of two bit strings captured in logic

$$
\begin{array}{rll}
 & x_{b-1} \ldots x_{i+1}\, x_i \ldots x_0 & \\
+ & y_{b-1} \ldots y_{i+1}\, y_i \ldots y_0 & \\
 & c_b\, c_{b-1} \ldots c_{i+1}\, c_i \ldots c_0 & \text{(carry)} \\
\hline
 & s_b\, s_{b-1} \ldots s_{i+1}\, s_i \ldots s_0 & \text{(result)}
\end{array}
$$

- $c_0 \; = \; 0$
- for $i = 0, \ldots, b-1$:

$$s_i \quad = \; x_i \oplus y_i \oplus c_i$$

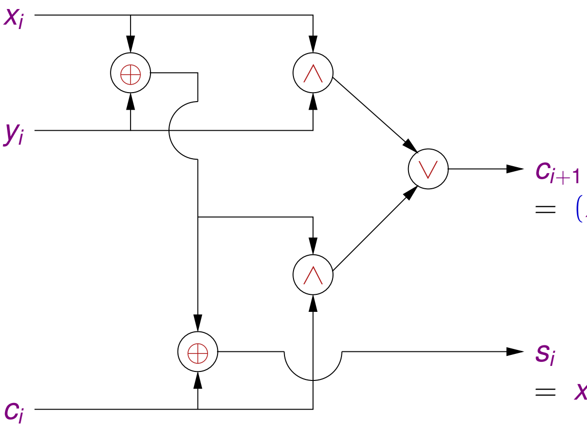
$$c_{i+1} \; = \; (x_i \wedge y_i) \; \vee \; (c_i \wedge (x_i \oplus y_i))$$

- Finally, $s_b \; = \; c_b$

Full adder

Question

Give a circuit for the full adder using AND, OR and XOR gates with inputs x_i, y_i and c_i and outputs s_i and c_{i+1}.



$$c_{i+1} = (x_i \wedge y_i) \vee (c_i \wedge (x_i \oplus y_i))$$

$$s_i = x_i \oplus y_i \oplus c_i$$

Full adder in Mendelson

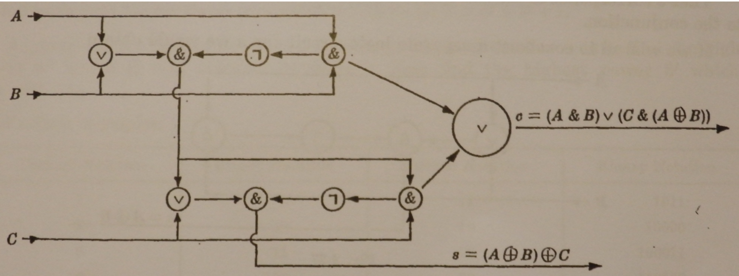


Fig. 4-30

The circuit of Fig. 4-30 is called a *full adder*.

Read x_i, y_i and c_i for A, B and C, respectively.

Read s_i and c_{i+1} for s and c, respectively.

Full adder

By repeatedly applying the full adder circuit, $s_0, \ldots, s_b$ and $c_1, \ldots, c_b$ can be computed, given $x_0, \ldots, x_b, y_0, \ldots, y_b$.

Question

Why is no circuit needed to compute c_0 ?

Take home

- axioms for $\equiv$
  - relation with the algebra of sets
- equivalence classes of $\equiv$
- logic circuits
  - AND, OR, NOT and XOR gates
  - binary arithmetic
  - half and full adder