

Notes for ECS 20, Lecture 3 – April 7, 2009 –

Today: o Sentential Logic (continued)

Review:

- Definition of a well formed formula (for sentential logic, over same set of proposition symbols P) is:

Equivalent to a *Propositional Formula* (in the text): truth depends on the values of its *logical variables*.

- “Order of Precedence”

\neg binds most tightly

\wedge

\vee

\rightarrow

\leftrightarrow binds least tightly

Parenthesis can change the default order. Within a given precedence, the usual convention is that things group right-to-left.

For example, $C \vee \neg A \wedge B \rightarrow B$ Is equivalent to: $(C \vee ((\neg A) \wedge B)) \rightarrow B$

1. We can associate any WFF to a **tree** where the leaves are the proposition symbols and constants 0 and 1 and the **internal nodes** are marked with **logical operators**. Given a **truth assignment**, which was described last time, you can propagate up truth value to every node. We showed how to do this for the example above.

Some practice designing formulas and circuits, too

Exercise 1 Who won the fight?

Two fighters, A and B. Three judges, each votes “0” if he thinks A won and “1” if he thinks B won. We want to create a Boolean formula that computes who won, according to *majority vote*.

Majority(P,Q, R) = 1 iff at least two of P,Q,R are 1, and 0 otherwise.

First write out a truth table for what you want:

P	Q	R	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Disjunctive normal form (DNF) – The formula is the OR of terms, and each term is the AND of variables or their complements. We can take ANY truth table and “read it out” as DNF. Above, we get

$$(\neg P)QR \vee P(\neg Q)R \vee PQ(\neg R) \vee (PQR)$$

We can simplify this a bit using:

$$QR \vee PR \vee PQ$$

Exercise 2: Find WFF that is 1 if *exactly one* of A, B, C, and D, are true. (NOT DONE IN CLASS)

For exactly one of these variables to be true, we need that *at least one* of the variables is true and *at most one* of the variables is true. The first is easy to translate into sentential logic:

$$A \vee B \vee C \vee D$$

The second is a little trickier: we want to say if A is true, for example than B must be false, that is, $A \rightarrow \neg B$; and $A \rightarrow \neg C$, and so forth:

$$(A \rightarrow \neg B) (A \rightarrow \neg C) (A \rightarrow \neg D) (B \rightarrow \neg A) (B \rightarrow \neg C) (B \rightarrow \neg D) (C \rightarrow \neg A) (C \rightarrow \neg B) (C \rightarrow \neg D) (D \rightarrow \neg A) (D \rightarrow \neg B) (D \rightarrow \neg C).$$

So and this formula and the prior one and you are done. More generally, to say that exactly one of X_1, \dots, X_n are true, we could use a formula of the form

$$(\bigvee_i X_i) \wedge \bigwedge_{i \neq j} (X_i \rightarrow \neg X_j)$$

Logical completeness. By virtue of the “DNF algorithm” that turns any **truth table** to a **WFF** that corresponds to the functionality, we know that any Boolean formula—or any truth table—can be represented using only the logical connectives $\{\wedge, \vee, \neg\}$. We say that this set of connectives are **logically complete**.

In fact, we can use a smaller set of connectives, eliminating **or** by using the identity:

$$P \vee Q \equiv \neg(\neg P \wedge \neg Q)$$

which is known as DeMorgan's law. We could, alternatively, eliminate **and** by using the DeMorgan law:

$$P \wedge Q \equiv \neg(\neg P \vee \neg Q).$$

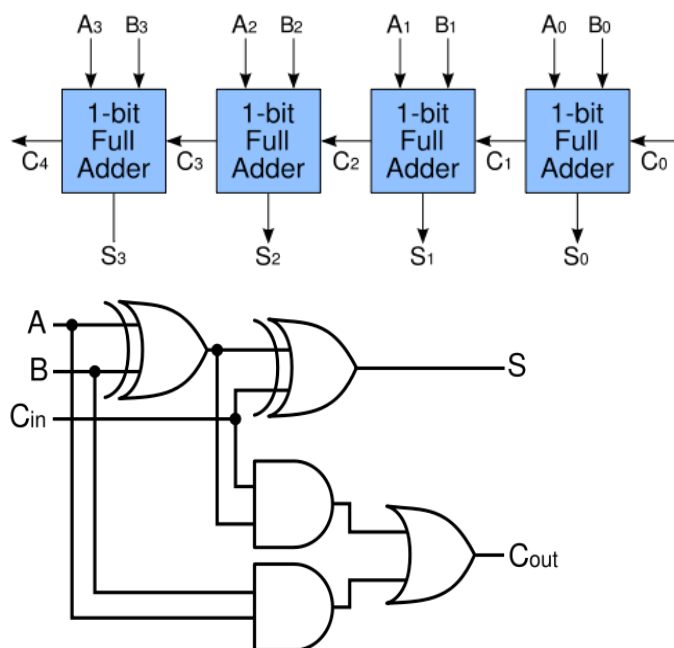
So we have show that

- \wedge & \neg are logically complete,
- \vee & \neg are logically complete. In addition,
- NAND, all by itself, is logically complete, because we can rewrite \wedge and \neg using NAND (tie the inputs of the NAND together to make an inverter) and
- NOR, all by itself, is logically complete, because we can rewrite \vee and \neg using NOR (tie the inputs of NOR together to make an inverter).

Exercise 3: Find a circuit to add up two 4-bit binary numbers.

Example: add 1011 (11) to 0110 (6) to get 10001 (17) showing sums and carries.

Here is a diagram for how to use a **full adder**, and then a circuit for the adder. (In class we first wrote out the truth table for the full adder.) Note that A_i and B_i are the bits of the two inputs, S_i is the current Sum bit, and C_i is the carry to go to the next place.





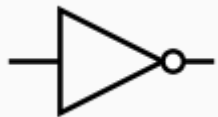

Tautologies, satisfiability, and logical equivalence

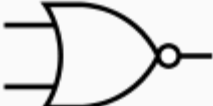

Definitions:

Given WFF's α and β ,

- α is a **tautology** if α is True for all truth assignments t of its variables
- α is **satisfiable** if there exists a truth assignment t s.t. α is true.
- α is a **contradiction** if α is False for all truth assignments t .
- α and β are (logically) **equivalent** if $t(\alpha)=t(\beta)$ for all truth assignments t .

Gates Diagrams:

Type	Distinctive shape	Truth table																		
AND		<table><tr><th colspan="2">INPUT</th><th>OUTPUT</th></tr><tr><th>A</th><th>B</th><th>A AND B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	INPUT		OUTPUT	A	B	A AND B	0	0	0	0	1	0	1	0	0	1	1	1
INPUT		OUTPUT																		
A	B	A AND B																		
0	0	0																		
0	1	0																		
1	0	0																		
1	1	1																		
OR		<table><tr><th colspan="2">INPUT</th><th>OUTPUT</th></tr><tr><th>A</th><th>B</th><th>A OR B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	INPUT		OUTPUT	A	B	A OR B	0	0	0	0	1	1	1	0	1	1	1	1
INPUT		OUTPUT																		
A	B	A OR B																		
0	0	0																		
0	1	1																		
1	0	1																		
1	1	1																		
NOT		<table><tr><th>INPUT</th><th>OUTPUT</th></tr><tr><th>A</th><th>NOT A</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	INPUT	OUTPUT	A	NOT A	0	1	1	0										
INPUT	OUTPUT																			
A	NOT A																			
0	1																			
1	0																			
NAND		<table><tr><th colspan="2">INPUT</th><th>OUTPUT</th></tr><tr><th>A</th><th>B</th><th>A NAND B</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	INPUT		OUTPUT	A	B	A NAND B	0	0	1	0	1	1	1	0	1	1	1	0
INPUT		OUTPUT																		
A	B	A NAND B																		
0	0	1																		
0	1	1																		
1	0	1																		
1	1	0																		

		<table><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	1	0	1	1	1	0												
1	0	1																		
1	1	0																		
<u>NOR</u>		<table><tr><th colspan="2">INPUT</th><th>OUTPUT</th></tr><tr><th>A</th><th>B</th><th>A NOR B</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	INPUT		OUTPUT	A	B	A NOR B	0	0	1	0	1	0	1	0	0	1	1	0
INPUT		OUTPUT																		
A	B	A NOR B																		
0	0	1																		
0	1	0																		
1	0	0																		
1	1	0																		
<u>XOR</u>		<table><tr><th colspan="2">INPUT</th><th>OUTPUT</th></tr><tr><th>A</th><th>B</th><th>A XOR B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	INPUT		OUTPUT	A	B	A XOR B	0	0	0	0	1	1	1	0	1	1	1	0
INPUT		OUTPUT																		
A	B	A XOR B																		
0	0	0																		
0	1	1																		
1	0	1																		
1	1	0																		