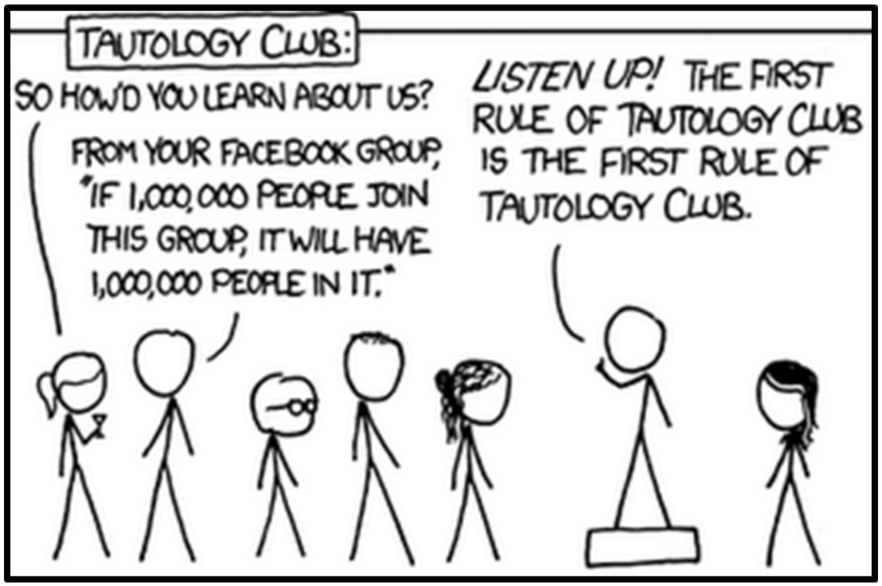
CSE 311: Foundations of Computing

Lecture 4: Boolean Algebra, Circuits, Canonical Forms



Last Time: Proofs of Equivalence

To show A is equivalent to B

- Apply a series of logical equivalences to sub-expressions to convert A to B

To show A is a tautology

- Apply a series of logical equivalences to sub-expressions to convert A to T

Prove this is a Tautology: Option 1

$$(p \wedge r) \rightarrow (r \vee p)$$

Use a series of equivalences like so:

$$(p \wedge r) \rightarrow (r \vee p) \equiv$$
$$\equiv$$
$$\equiv$$
$$\equiv$$
$$\equiv$$
$$\equiv$$
$$\equiv$$
$$\equiv$$
$$\equiv \mathbf{T}$$

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

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

Absorption
- $p \vee (p \wedge q) \equiv p$
- $p \wedge (p \vee q) \equiv p$

Negation
- $p \vee \neg p \equiv \mathrm{T}$
- $p \wedge \neg p \equiv \mathrm{F}$

Identity
- $p \wedge \mathrm{T} \equiv p$
- $p \vee \mathrm{F} \equiv p$

Domination
- $p \vee \mathrm{T} \equiv \mathrm{T}$
- $p \wedge \mathrm{F} \equiv \mathrm{F}$

Idempotent
- $p \vee p \equiv p$
- $p \wedge p \equiv p$

Commutative
- $p \vee q \equiv q \vee p$
- $p \wedge q \equiv q \wedge p$

Prove this is a Tautology: Option 1

$$(p \wedge r) \rightarrow (r \vee p)$$

Use a series of equivalences like so:

$(p \wedge r) \rightarrow (r \vee p)$	$\equiv \neg(p \wedge r) \vee (r \vee p)$	Law of Implication
	$\equiv (\neg p \vee \neg r) \vee (r \vee p)$	De Morgan
	$\equiv \neg p \vee (\neg r \vee (r \vee p))$	Associative
	$\equiv \neg p \vee ((\neg r \vee r) \vee p)$	Associative
	$\equiv \neg p \vee (p \vee (\neg r \vee r))$	Commutative
	$\equiv (\neg p \vee p) \vee (\neg r \vee r)$	Associative
	$\equiv (p \vee \neg p) \vee (r \vee \neg r)$	Commutative (twice)
	$\equiv \mathbf{T} \vee \mathbf{T}$	Negation (twice)
	$\equiv \mathbf{T}$	Domination/Identity

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

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

Absorption
- $p \vee (p \wedge q) \equiv p$
- $p \wedge (p \vee q) \equiv p$

Negation
- $p \vee \neg p \equiv \mathrm{T}$
- $p \wedge \neg p \equiv \mathrm{F}$

Identity
- $p \wedge \mathrm{T} \equiv p$
- $p \vee \mathrm{F} \equiv p$

Domination
- $p \vee \mathrm{T} \equiv \mathrm{T}$
- $p \wedge \mathrm{F} \equiv \mathrm{F}$

Idempotent
- $p \vee p \equiv p$
- $p \wedge p \equiv p$

Commutative
- $p \vee q \equiv q \vee p$
- $p \wedge q \equiv q \wedge p$

Prove this is a Tautology: Option 2

$$(p \wedge r) \rightarrow (r \vee p)$$

Make a Truth Table and show:

$$(p \wedge r) \rightarrow (r \vee p) \equiv \mathbf{T}$$

$\boldsymbol{p}$	$\boldsymbol{r}$	$\boldsymbol{p} \wedge \boldsymbol{r}$	$\boldsymbol{r} \vee \boldsymbol{p}$	$(\boldsymbol{p} \wedge \boldsymbol{r}) \rightarrow (\boldsymbol{r} \vee \boldsymbol{p})$
T	**T**	**T**	**T**	**T**
T	**F**	**F**	**T**	**T**
F	**T**	**F**	**T**	**T**
F	**F**	**F**	**F**	**T**

Boolean Logic

Combinational Logic

– output = F(input)

Sequential Logic

– $output_t = F(output_{t-1}, input_t)$

- output dependent on history
- concept of a time step (clock, t)

Boolean Logic

Combinational Logic

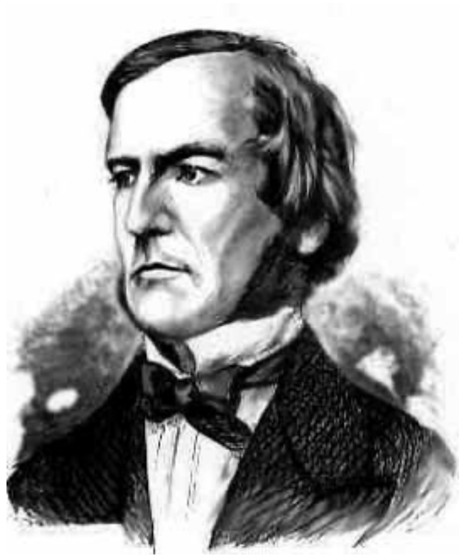
– output = F(input)

Boolean Algebra: another notation for logic consisting of...

– a set of elements B = {0, 1}
– binary operations { + , • } (OR, AND)
– and a unary operation { ’ } (NOT)

Boolean Algebra

- Usual notation used in circuit design
- Boolean algebra
  - a set of elements B containing {0, 1}
  - binary operations { + , • }
  - and a unary operation { ' }
  - such that the following axioms hold:



For any a, b, c in B:

1. closure:	a + b is in B	a • b is in B
2. commutativity:	a + b = b + a	a • b = b • a
3. associativity:	a + (b + c) = (a + b) + c	a • (b • c) = (a • b) • c
4. distributivity:	a + (b • c) = (a + b) • (a + c)	a • (b + c) = (a • b) + (a • c)
5. identity:	a + 0 = a	a • 1 = a
6. complementarity:	a + a' = 1	a • a' = 0
7. null:	a + 1 = 1	a • 0 = 0
8. idempotency:	a + a = a	a • a = a
9. involution:	(a')' = a	

A Combinational Logic Example

Sessions of Class:

We would like to compute the number of lectures or quiz sections remaining *at the start* of a given day of the week.

- **Inputs:** Day of the Week, Lecture/Section flag
- **Output:** Number of sessions left

Examples: Input: (Wednesday, Lecture) Output: 2

Input: (Monday, Section) Output: 1

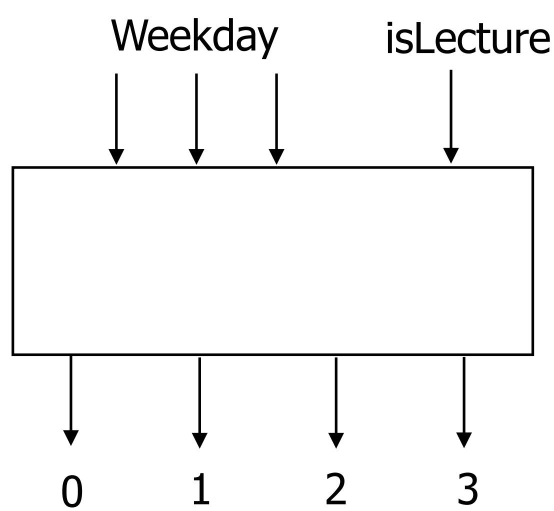
Implementation in Software

```
public int classesLeftInMorning(int weekday, boolean isLecture) {
    switch (weekday) {
        case SUNDAY:
        case MONDAY:
            return isLecture ? 3 : 1;
        case TUESDAY:
        case WEDNESDAY:
            return isLecture ? 2 : 1;
        case THURSDAY:
            return isLecture ? 1 : 1;
        case FRIDAY:
            return isLecture ? 1 : 0;
        case SATURDAY:
            return isLecture ? 0 : 0;
   }
}
```

Implementation with Combinational Logic

Encoding:

- How many bits for each input/output?
- Binary number for weekday
- One bit for each possible output



Defining Our Inputs!

Weekday Input:

- Binary number for weekday
- Sunday = 0, Monday = 1, …
- We care about these in binary:

Weekday	Number	Binary
Sunday	0	$(000)_2$
Monday	1	$(001)_2$
Tuesday	2	$(010)_2$
Wednesday	3	$(011)_2$
Thursday	4	$(100)_2$
Friday	5	$(101)_2$
Saturday	6	$(110)_2$

Converting to a Truth Table!

```
case SUNDAY or MONDAY:
    return isLecture ? 3 : 1;
case TUESDAY or WEDNESDAY:
    return isLecture ? 2 : 1;
case THURSDAY:
    return isLecture ? 1 : 1;
case FRIDAY:
    return isLecture ? 1 : 0;
case SATURDAY:
    return isLecture ? 0 : 0;
```

Weekday		isLecture	c_0	c_1	c_2	c_3
SUN	000	0				
SUN	000	1				
MON	001	0				
MON	001	1				
TUE	010	0				
TUE	010	1				
WED	011	0				
WED	011	1				
THU	100	-				
FRI	101	0				
FRI	101	1				
SAT	110	-				
-	111	-				

Converting to a Truth Table!

```
case SUNDAY or MONDAY:
    return isLecture ? 3 : 1;
case TUESDAY or WEDNESDAY:
    return isLecture ? 2 : 1;
case THURSDAY:
    return isLecture ? 1 : 1;
case FRIDAY:
    return isLecture ? 1 : 0;
case SATURDAY:
    return isLecture ? 0 : 0;
```

Weekday		isLecture	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

Truth Table to Logic (Part 1)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

Let's begin by finding an expression for c_3. To do this, we look at the rows where c_3 = 1 (true).

Truth Table to Logic (Part 1)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

DAY == SUN && L == 1

DAY == MON && L == 1

Truth Table to Logic (Part 1)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

$d_2d_1d_0$ == 000 && L == 1

$d_2d_1d_0$ == 001 && L == 1

Substituting DAY for the binary representation.

Truth Table to Logic (Part 1)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

d_2 == 0 && d_1 == 0 && d_0 == 0 && L == 1

d_2 == 0 && d_1 == 0 && d_0 == 1 && L == 1

Splitting up the bits of the day; so, we can write a formula.

Truth Table to Logic (Part 1)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

$d_2' \cdot d_1' \cdot d_0' \cdot L$

$d_2' \cdot d_1' \cdot d_0 \cdot L$

Replacing with Boolean Algebra...

Truth Table to Logic (Part 1)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

SUN, 000, L = 1 → $d_2' \cdot d_1' \cdot d_0' \cdot L$

MON, 001, L = 1 → $d_2' \cdot d_1' \cdot d_0 \cdot L$

How do we combine them?

Truth Table to Logic (Part 1)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

$d_2' \cdot d_1' \cdot d_0' \cdot L$

$d_2' \cdot d_1' \cdot d_0 \cdot L$

Either situation causes c_3 to be true. So, we "or" them.

$$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$$

Truth Table to Logic (Part 2)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$

Now, we do c_2.

Truth Table to Logic (Part 3)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

Now, we do c_1:

$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$

$c_2 = d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L$

Truth Table to Logic (Part 3)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3	
SUN	000	0	0	1	0	0	→ $d_2' \cdot d_1' \cdot d_0' \cdot L'$
SUN	000	1	0	0	0	1	
MON	001	0	0	1	0	0	→ $d_2' \cdot d_1' \cdot d_0 \cdot L'$
MON	001	1	0	0	0	1	
TUE	010	0	0	1	0	0	→ $d_2' \cdot d_1 \cdot d_0' \cdot L'$
TUE	010	1	0	0	1	0	
WED	011	0	0	1	0	0	→ $d_2' \cdot d_1 \cdot d_0 \cdot L'$
WED	011	1	0	0	1	0	
THU	100	-	0	1	0	0	→ ???
FRI	101	0	1	0	0	0	
FRI	101	1	0	1	0	0	→ $d_2 \cdot d_1' \cdot d_0 \cdot L$
SAT	110	-	1	0	0	0	
-	111	-	1	0	0	0	

Now, we do c_1:

$$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$$

$$c_2 = d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L$$

Truth Table to Logic (Part 3)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

Now, we do **c_1**:

$d_2' \cdot d_1' \cdot d_0' \cdot L'$

$d_2' \cdot d_1' \cdot d_0 \cdot L'$

$d_2' \cdot d_1 \cdot d_0' \cdot L'$

$d_2' \cdot d_1 \cdot d_0 \cdot L'$

$d_2 \cdot d_1' \cdot d_0'$

$d_2 \cdot d_1' \cdot d_0 \cdot L$

No matter what L is, we always say it's 1. So, we don't need L in the expression.

$$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$$
$$c_2 = d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L$$

Truth Table to Logic (Part 3)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

Now, we do c_1:

$d_2' \cdot d_1' \cdot d_0' \cdot L'$

$d_2' \cdot d_1' \cdot d_0 \cdot L'$

$d_2' \cdot d_1 \cdot d_0' \cdot L'$

$d_2' \cdot d_1 \cdot d_0 \cdot L'$

$d_2 \cdot d_1' \cdot d_0'$

$d_2 \cdot d_1' \cdot d_0 \cdot L$

No matter what L is, we always say it's 1. So, we don't need L in the expression.

$$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$$

$$c_2 = d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L$$

$$c_1 = d_2' \cdot d_1' \cdot d_0' \cdot L' + d_2' \cdot d_1' \cdot d_0 \cdot L' + d_2' \cdot d_1 \cdot d_0' \cdot L' + d_2' \cdot d_1 \cdot d_0 \cdot L' + d_2 \cdot d_1' \cdot d_0' + d_2 \cdot d_1' \cdot d_0 \cdot L$$

Truth Table to Logic (Part 4)

	$d_2d_1d_0$	L	c_0	c_1	c_2	c_3
SUN	000	0	0	1	0	0
SUN	000	1	0	0	0	1
MON	001	0	0	1	0	0
MON	001	1	0	0	0	1
TUE	010	0	0	1	0	0
TUE	010	1	0	0	1	0
WED	011	0	0	1	0	0
WED	011	1	0	0	1	0
THU	100	-	0	1	0	0
FRI	101	0	1	0	0	0
FRI	101	1	0	1	0	0
SAT	110	-	1	0	0	0
-	111	-	1	0	0	0

$c_1 = d_2' \cdot d_1' \cdot d_0' \cdot L' + d_2' \cdot d_1' \cdot d_0 \cdot L' +$
$d_2' \cdot d_1 \cdot d_0' \cdot L' + d_2' \cdot d_1 \cdot d_0 \cdot L' +$
$d_2 \cdot d_1' \cdot d_0' + d_2 \cdot d_1' \cdot d_0 \cdot L$

$c_2 = d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L$

$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$

Finally, we do c_0:

FRI 101 0 → $d_2 \cdot d_1' \cdot d_0 \cdot L'$

SAT 110 - → $d_2 \cdot d_1 \cdot d_0'$

- 111 - → $d_2 \cdot d_1 \cdot d_0$

Truth Table to Logic (Part 4)

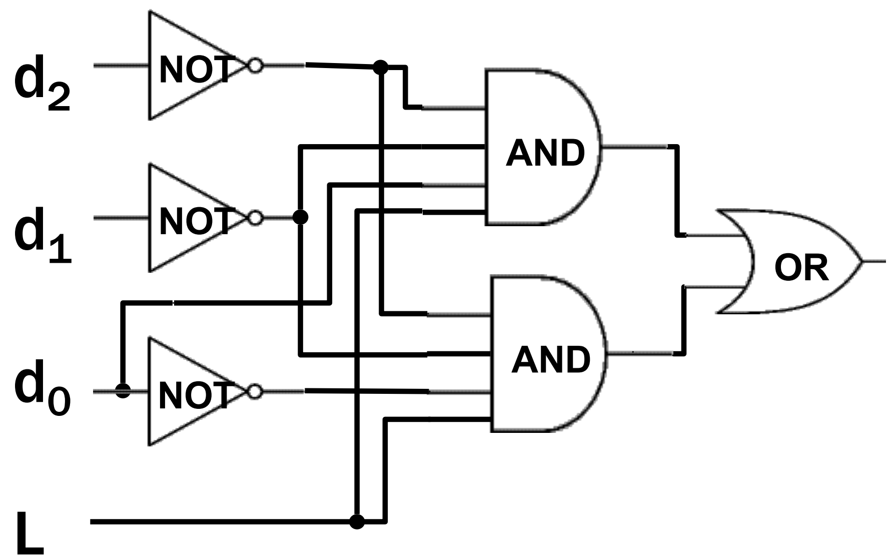
$c_0 = d_2 \cdot d_1' \cdot d_0 \cdot L' + d_2 \cdot d_1 \cdot d_0' + d_2 \cdot d_1 \cdot d_0$

$c_1 = d_2' \cdot d_1' \cdot d_0' \cdot L' + d_2' \cdot d_1' \cdot d_0 \cdot L' + d_2' \cdot d_1 \cdot d_0' \cdot L' + d_2' \cdot d_1 \cdot d_0 \cdot L' + d_2 \cdot d_1' \cdot d_0' + d_2 \cdot d_1' \cdot d_0 \cdot L$

$c_2 = d_2' \cdot d_1 \cdot d_0' \cdot L + d_2' \cdot d_1 \cdot d_0 \cdot L$

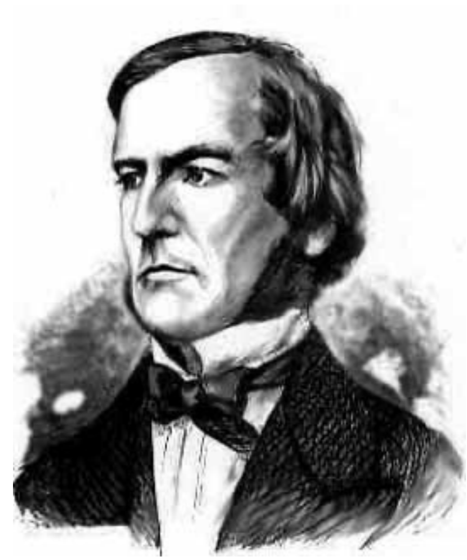
$c_3 = d_2' \cdot d_1' \cdot d_0' \cdot L + d_2' \cdot d_1' \cdot d_0 \cdot L$

Here's c_3 as a circuit:



Boolean Algebra

- **Usual notation used in circuit design**
- **Boolean algebra**
  - a set of elements B containing {0, 1}
  - binary operations { + , • }
  - and a unary operation { ’ }
  - such that the following axioms hold:



For any a, b, c in B:

1. closure:	a + b is in B	a • b is in B
2. commutativity:	a + b = b + a	a • b = b • a
3. associativity:	a + (b + c) = (a + b) + c	a • (b • c) = (a • b) • c
4. distributivity:	a + (b • c) = (a + b) • (a + c)	a • (b + c) = (a • b) + (a • c)
5. identity:	a + 0 = a	a • 1 = a
6. complementarity:	a + a’ = 1	a • a’ = 0
7. null:	a + 1 = 1	a • 0 = 0
8. idempotency:	a + a = a	a • a = a
9. involution:	(a’)’ = a	

Simplification using Boolean Algebra

uniting:

10. $a \bullet b + a \bullet b' = a$

10D. $(a + b) \bullet (a + b') = a$

absorption:

11. $a + a \bullet b = a$

11D. $a \bullet (a + b) = a$

12. $(a + b') \bullet b = a \bullet b$

12D. $(a \bullet b') + b = a + b$

factoring:

13. $(a + b) \bullet (a' + c) = a \bullet c + a' \bullet b$

13D. $a \bullet b + a' \bullet c = (a + c) \bullet (a' + b)$

consensus:

14. $(a \bullet b) + (b \bullet c) + (a' \bullet c) = a \bullet b + a' \bullet c$

14D. $(a + b) \bullet (b + c) \bullet (a' + c) = (a + b) \bullet (a' + c)$

de Morgan's:

15. $(a + b + \ldots)' = a' \bullet b' \bullet \ldots$

15D. $(a \bullet b \bullet \ldots)' = a' + b' + \ldots$

Proving Theorems

2. commutativity:	$a + b = b + a$	$a \bullet b = b \bullet a$
3. associativity:	$a + (b + c) = (a + b) + c$	$a \bullet (b \bullet c) = (a \bullet b) \bullet c$
4. distributivity:	$a + (b \bullet c) = (a + b) \bullet (a + c)$	$a \bullet (b + c) = (a \bullet b) + (a \bullet c)$
5. identity:	$a + 0 = a$	$a \bullet 1 = a$
6. complementarity:	$a + a' = 1$	$a \bullet a' = 0$
7. null:	$a + 1 = 1$	$a \bullet 0 = 0$
8. idempotency:	$a + a = a$	$a \bullet a = a$
9. involution:	$(a')' = a$	

Using the laws of Boolean Algebra:

prove the Uniting theorem: $X \bullet Y + X \bullet Y' = X$

$X \bullet Y + X \bullet Y' =$

prove the Absorption theorem: $X + X \bullet Y = X$

$X + X \bullet Y =$

Proving Theorems

2. commutativity:	$a + b = b + a$	$a \bullet b = b \bullet a$
3. associativity:	$a + (b + c) = (a + b) + c$	$a \bullet (b \bullet c) = (a \bullet b) \bullet c$
4. distributivity:	$a + (b \bullet c) = (a + b) \bullet (a + c)$	$a \bullet (b + c) = (a \bullet b) + (a \bullet c)$
5. identity:	$a + 0 = a$	$a \bullet 1 = a$
6. complementarity:	$a + a' = 1$	$a \bullet a' = 0$
7. null:	$a + 1 = 1$	$a \bullet 0 = 0$
8. idempotency:	$a + a = a$	$a \bullet a = a$
9. involution:	$(a')' = a$	

Using the laws of Boolean Algebra:

prove the Uniting theorem: $X \bullet Y + X \bullet Y' = X$

distributivity	$X \bullet Y + X \bullet Y' = X \bullet (Y + Y')$
complementarity	$= X \bullet 1$
identity	$= X$

prove the Absorption theorem: $X + X \bullet Y = X$

identity	$X + X \bullet Y = X \bullet 1 + X \bullet Y$
distributivity	$= X \bullet (1 + Y)$
commutativity	$= X \bullet (Y + 1)$
null	$= X \bullet 1$
identity	$= X$

Proving Theorems

Using truth table:

For example, de Morgan's Law:

$(X + Y)' = X' \bullet Y'$

NOR is equivalent to AND with inputs complemented

X	Y	X′	Y′	(X + Y)′	X′ • Y′
0	0	1	1	1	1
0	1	1	0	0	0
1	0	0	1	0	0
1	1	0	0	0	0

$(X \bullet Y)' = X' + Y'$

NAND is equivalent to OR with inputs complemented

X	Y	X′	Y′	(X • Y)′	X′ + Y′
0	0	1	1	1	1
0	1	1	0	1	1
1	0	0	1	1	1
1	1	0	0	0	0

Simplifying using Boolean Algebra

c3 = d2’•d1’•d0’•L + d2’•d1’•d0•L

= d2’•d1’•(d0’ + d0)•L

= d2’•d1’•1•L

= d2’•d1’•L