Lecture 5 Slides

Canonical Sums and Products (Minterms and Maxterms)

2-3 Variable Theorems

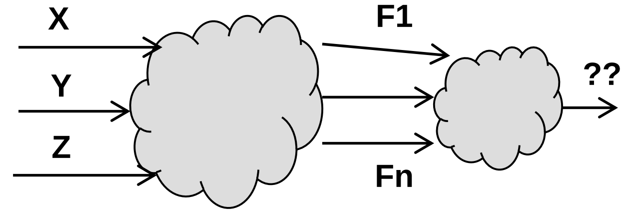
DeMorgan's Theorem

Using products of maxterms to implement a function

MAXTERMS

Question

- Is there a set of functions (F1, F2, etc.) that would allow you to build ANY 3-variable function
  - Think simple, think many



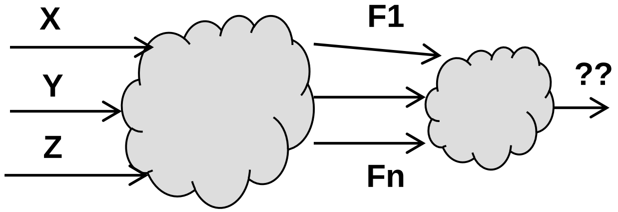
X	Y	Z	F1	F2	Fn	?
0	0	0				?
0	0	1				?
0	1	0				?
0	1	1				?
1	0	0				?
1	0	1				?
1	1	0				?
1	1	1				?

X	Y	Z	m0	m1	m2	m3	m4	m5	m6	m7	?
0	0	0	1	0	0	0	0	0	0	0	?
0	0	1	0	1	0	0	0	0	0	0	?
0	1	0	0	0	1	0	0	0	0	0	?
0	1	1	0	0	0	1	0	0	0	0	?
1	0	0	0	0	0	0	1	0	0	0	?
1	0	1	0	0	0	0	0	1	0	0	?
1	1	0	0	0	0	0	0	0	1	0	?
1	1	1	0	0	0	0	0	0	0	1	?

OR together any combination of m_i's

Question

- OR…this set of functions would also work.



X	Y	Z	M0	M1	M2	M3	M4	M5	M6	M7	?
0	0	0	**0**	1	1	1	1	1	1	1	?
0	0	1	1	**0**	1	1	1	1	1	1	?
0	1	0	1	1	**0**	1	1	1	1	1	?
0	1	1	1	1	1	**0**	1	1	1	1	?
1	0	0	1	1	1	1	**0**	1	1	1	?
1	0	1	1	1	1	1	1	**0**	1	1	?
1	1	0	1	1	1	1	1	1	**0**	1	?
1	1	1	1	1	1	1	1	1	1	**0**	?

G
1
0
1
0
1
1
0
1

AND together any combination of M_i's

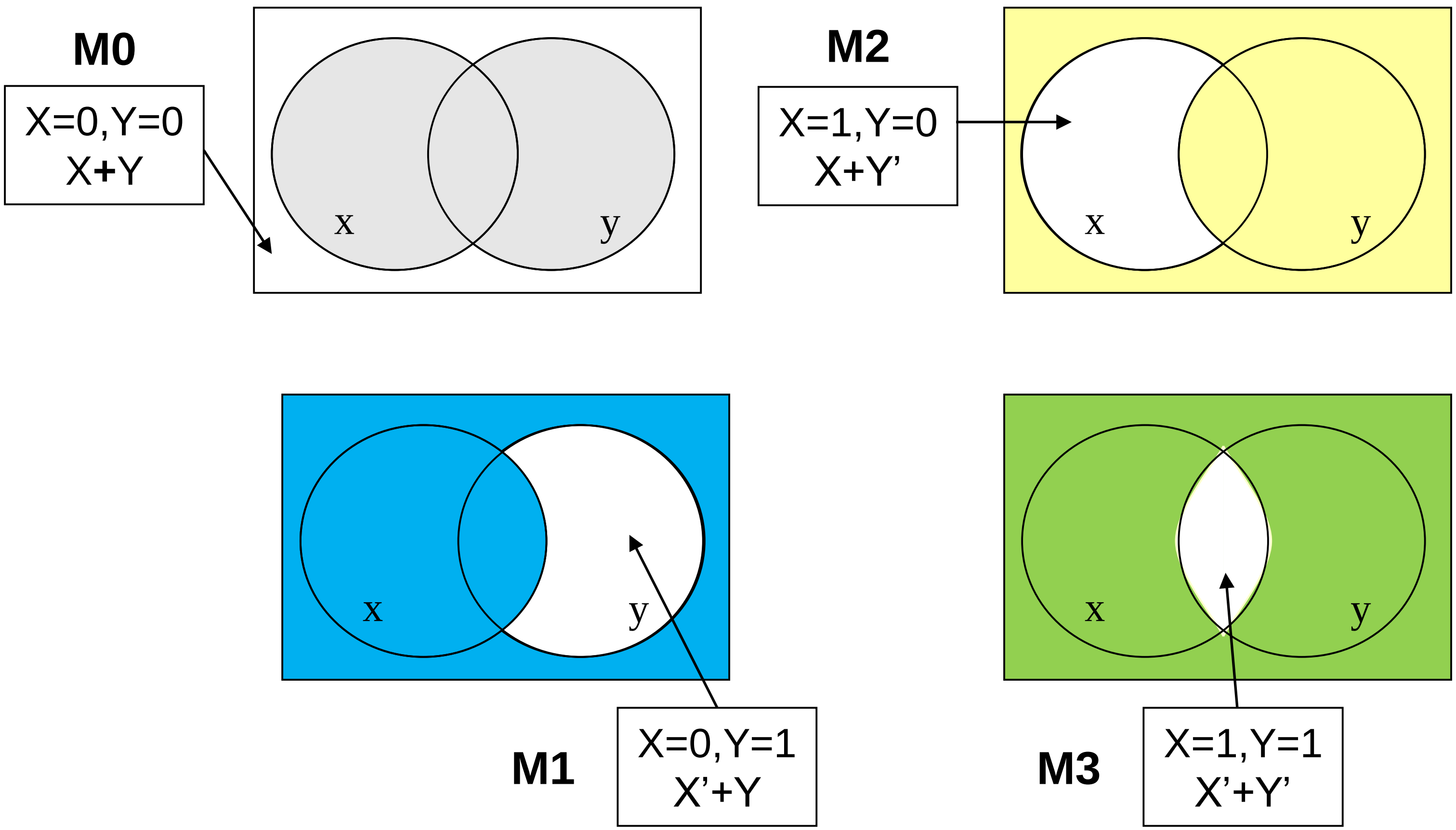
G = M1 • M3 • M6

Maxterm Definition

- **Maxterm**: A sum term where each input variable of a function appears exactly once in that term (either in its true or complemented form)
  - $f(x,y,z)$ =>
    - $x'+y'+z$
    - $x+y+z$
    - $y+z'$
    - $x'y'z'$

Venn Diagram of Maxterms

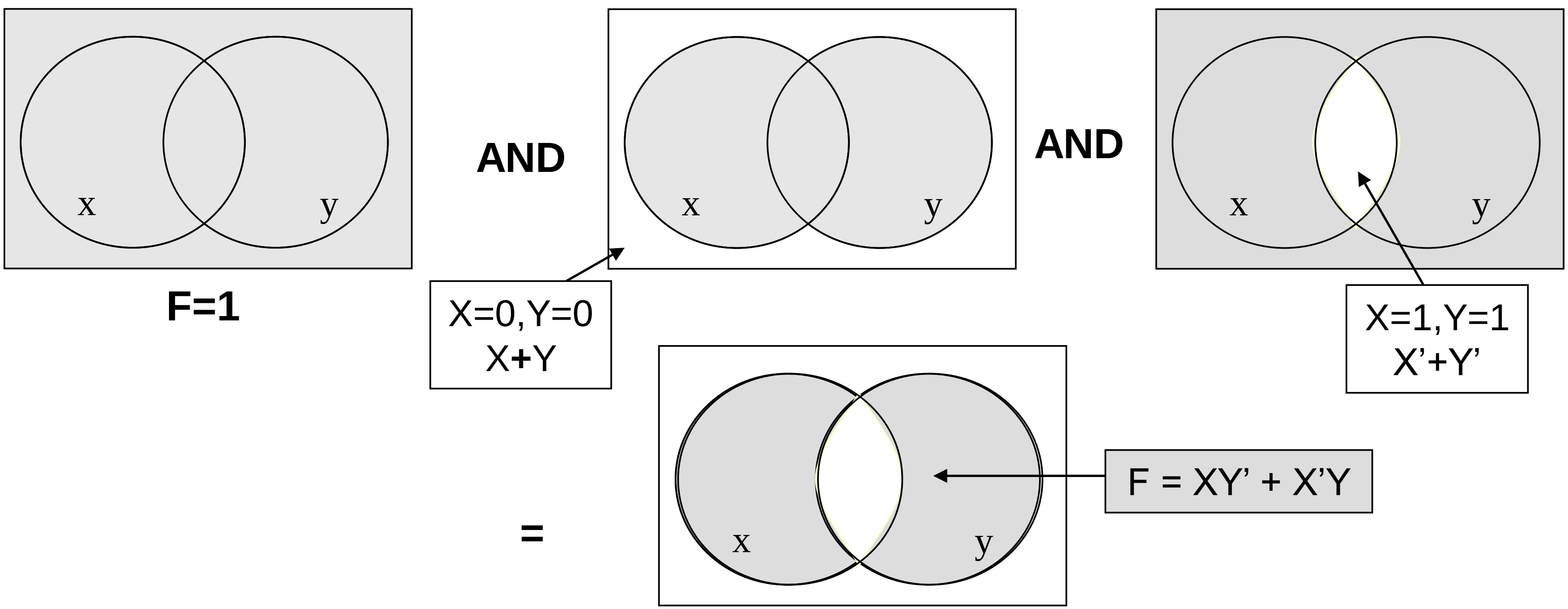
- Only one region OFF and all others ON



Product of Maxterms

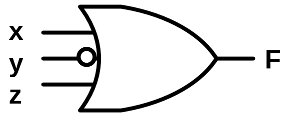
- To compose a function we can AND the maxterms from the function’s OFF-Set

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0



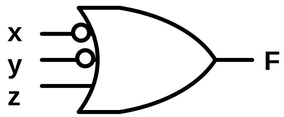
Checkers / Decoders

- An OR gate only outputs '0' for 1 combination
  - That combination can be changed by adding inverters to the inputs
  - We can think of the OR gate as "checking" or "decoding" a specific combination and outputting a '0' when it matches.



X	Y	Z	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

OR gate decoding (checking for) combination 010

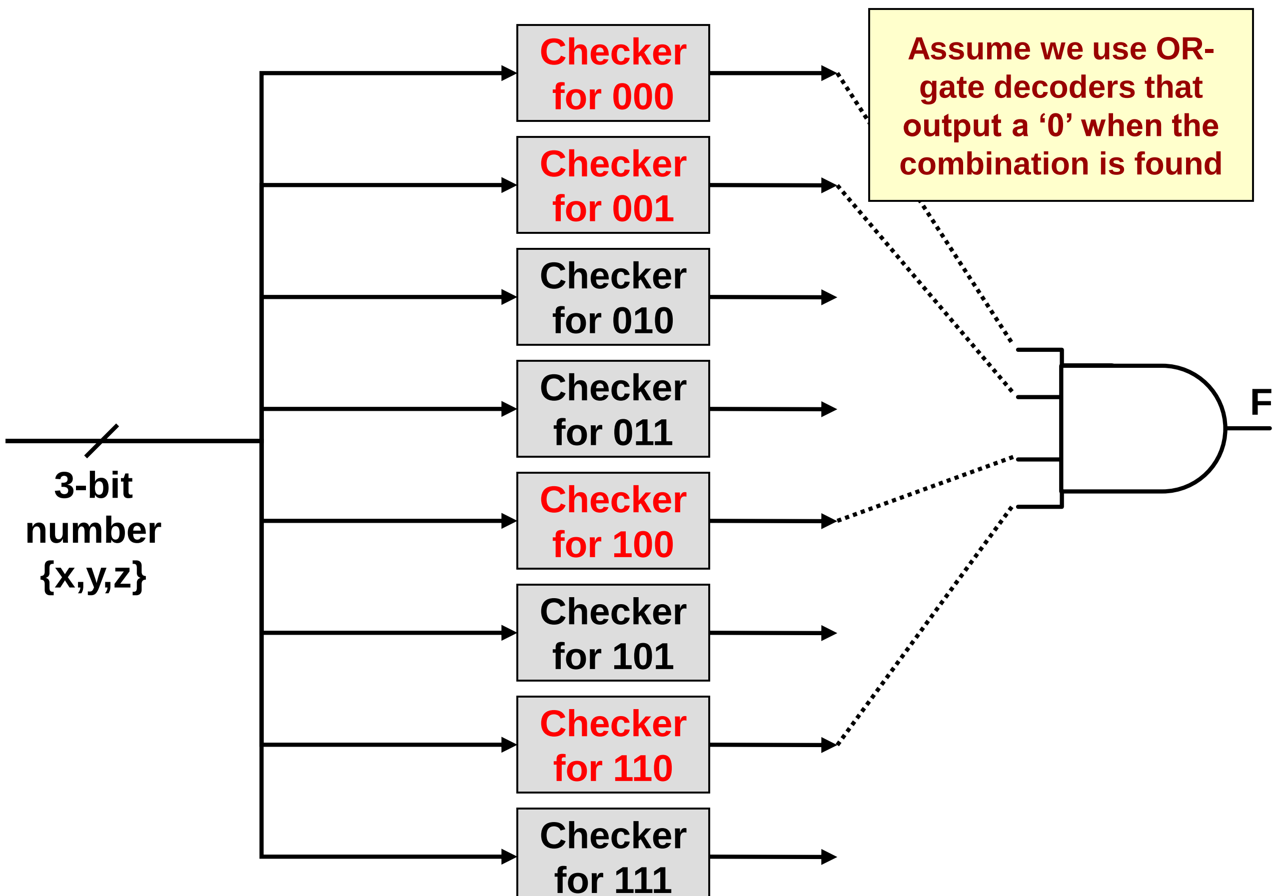


X	Y	Z	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

OR gate decoding (checking for) combination 110

Finding Equations/Circuits

- Given a function and checkers (called decoders) for each combination, we just need to AND together the checkers where F = 0



X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

LOGIC FUNCTION NOTATION

Canonical Sums

- We OR together all the minterms where F = 1
  - (Σ = SUM or OR of all the minterms)

$F = m_2+m_3+m_5+m_7$

Canonical Sum:

$F = \Sigma_{xyz}(2,3,5,7)$

List the minterms where F is 1.

	X	Y	Z	F
m_0	0	0	0	0
m_1	0	0	1	0
m_2	0	1	0	1
m_3	0	1	1	1
m_4	1	0	0	0
m_5	1	0	1	1
m_6	1	1	0	0
m_7	1	1	1	1

Canonical Products

- We AND together all the maxterms where F = 0

$$F = M_0 \bullet M_1 \bullet M_4 \bullet M_6$$

Canonical Product:

$$F = \Pi_{xyz}(0,1,4,6)$$

List the maxterms where F is 0.

	X	Y	Z	F
M_0	0	0	0	0
M_1	0	0	1	0
M_2	0	1	0	1
M_3	0	1	1	1
M_4	1	0	0	0
M_5	1	0	1	1
M_6	1	1	0	0
M_7	1	1	1	1

Canonical Form Practice

- $G = \sum_{XYZ}(\qquad\qquad) = \prod_{XYZ}(\qquad\qquad)$
- $B = \sum_{X,Y,Z}(5,6,7)$
- $F =$

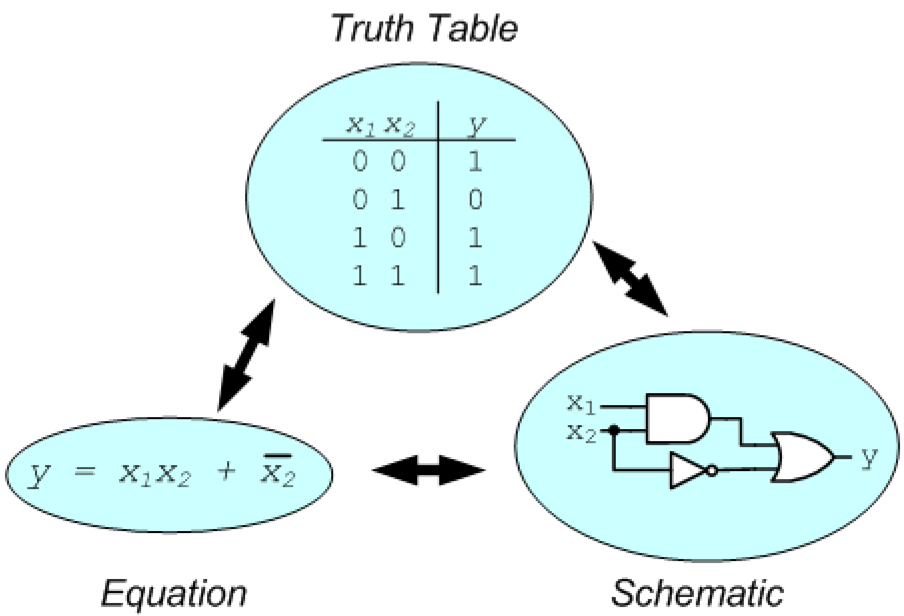
X	Y	Z	G
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

X	Y	Z	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

P. 60 and 61 in the Lecture Notes

Logic Functions

- A logic function maps input combinations to an output value (‘1’ or ‘0’)
- 3 possible representations of a function
  - Equation
  - Schematic
  - Truth Table
- Can convert between representations
- Truth table is only unique representation*



* Canonical Sums/Products (minterm/maxterm) representation provides a standard equation/schematic form that is unique per function

Unique Representations

- Canonical => Same functions will have same representations
- Truth Tables along with Canonical Sums and Products specify a function *uniquely*
- Equations/circuit schematics are NOT inherently canonical

Truth Table

x	y	z	P
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Canonical Sum

$$P = \sum_{x,y,z} \underbrace{(2,3,5,7)}$$

ON-Set of P (minterms)

Yields SOP equation, AND-OR circuit

Canonical Product

$$P = \prod_{x,y,z} \underbrace{(0,1,4,6)}$$

OFF-Set of P (maxterms)

Yields POS equation, OR-AND circuit

Boolean Algebra Terminology

- **SOP (Sum of Products) Form**: An SOP expression is a logical sum (OR) of product terms.
  - Correct Examples: [x'•y'•z + w + a'•b•c], [w + x'•z•y + y'z]
  - Incorrect Examples: [x'•y•z + w•(a+b)], [x•y + (y'•z)']
- **POS (Product of Sums) Form**: A POS expression is a logical product (AND) of sum terms.
  - Correct Examples: [(x+y'+z) • (w'+z) • (a)], [z'•(x+y)•(w'+y)]
  - Incorrect Examples: [x' + y•(x+w)], [(x+y)•(x+z)']

Check Yourself

Expression	SOP / POS / Both / Neither
w•x•(y•z)’ + xy’z + w	
xy+xz+(w’yz)	
(w+y’+z)(w+x)	
(w+y)x(w’+z)	
w’y + w’y + xy’	
w+x+y	

Check Yourself

Expression	SOP / POS / Both / Neither
w•x•(y•z)' + xy'z + w	Neither (Can't have complements of sub-expressions…only literal)
xy+xz+(w'yz)	SOP (parentheses are unnecessary)
(w+y'+z)(w+x)	POS
(w+y)x(w'+z)	POS (a single literal is a sum term)
wy + wy + xy'	SOP (redundancy doesn't matter)
w+x+y	Both (individual literals are both a product and sum term)

2- and 3-variable Theorems

BOOLEAN ALGEBRA AGAIN

2 & 3 Variable Theorems

T8	XY+XZ = X(Y+Z)	**T8'**	(X+Y)(X+Z) = X+YZ
T9	X + XY = X	**T9'**	X(X+Y) = X
T10	XY + XY' = X	**T10'**	(X+Y)(X+Y') = X
T11	XY + X'Z + YZ = XY + X'Z	**T11'**	(X+Y)(X'+Z)(Y+Z) = (X+Y)(X'+Z)

T8’ Proof

- The police are looking for drunk drivers. They can arrest a person if his breath test is positive OR if they find an open container in the car and the contents are alcoholic.
  - Arrest = Breath + OpenContainer • ContentsAlcoholic
- By T8’
  - Arrest = (Breath + OpenContainer)(Breath + ContentsAlcoholic)

B	O	C	O•C	B+O•C	B+O	B+C	(B+O)(B+C)
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					

Boolean Algebra Terminology

- **Literal**: A literal is an instance of a single variable or its complement.
  - Correct Examples: x, x', y, ALARM, (LON)'
  - Incorrect Examples: x+y, x'•y (these are expressions)
- **Product Term**: A single literal or a logical product (AND'ing) of two or more literals.
  - Correct Examples: x'•y'•z, w'•x'•y, w•x'•a'b, c'
  - Incorrect Examples: (x+y)•z•w, (x•y)'
  - Only evaluates to '1' for a single input combination
- **Sum Term**: A single literal or a logical sum (OR'ing) of two or more literals.
  - Correct Examples: z'+w+y, x + y', x'
  - Incorrect Examples: ab+c, (x + z)'
  - Only evaluates to '0' for a single input combination
- **SOP (Sum of Products) Form**: An SOP expression is a logical sum (OR) of product terms. (Convert to SOP by distributing fully using T8)
  - Correct Examples: [x'•y'•z + w + a'•b•c], [w + x'•z•y + y'z]
  - Incorrect Examples: [x'•y•z+w• (a+b)], [x•y + (y'•z)']
- **POS (Product of Sums) Form**: A POS expression is a logical product (AND) of sum terms. (Convert to POS by distributing fully using T8')
  - Correct Examples: [(x+y'+z) • (w'+z) • (a)], [z'•(x+y)•(w'+y)]
  - Incorrect Examples: [(x'+y) +y(x+w)], [(x+y) •(x+z)']

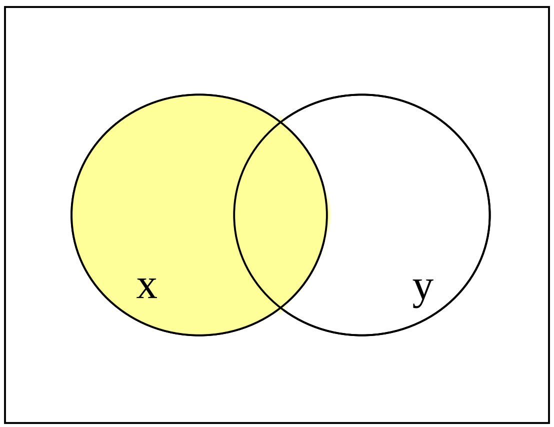
Convert to SOP/POS

Expression	SOP or POS
x' + y•(x+w) to SOP	
x+y'+zx' to POS	
(XZ + X'Z')(WY + W'Y') to SOP	
X'Y + (X+Y)Z to POS	

T9 Proof

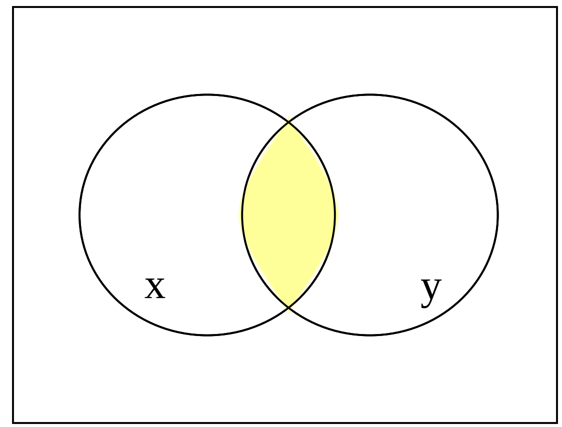
$$X + XY = X$$

X



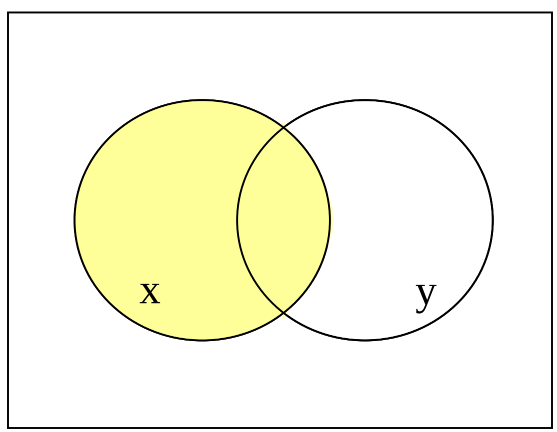
+

XY



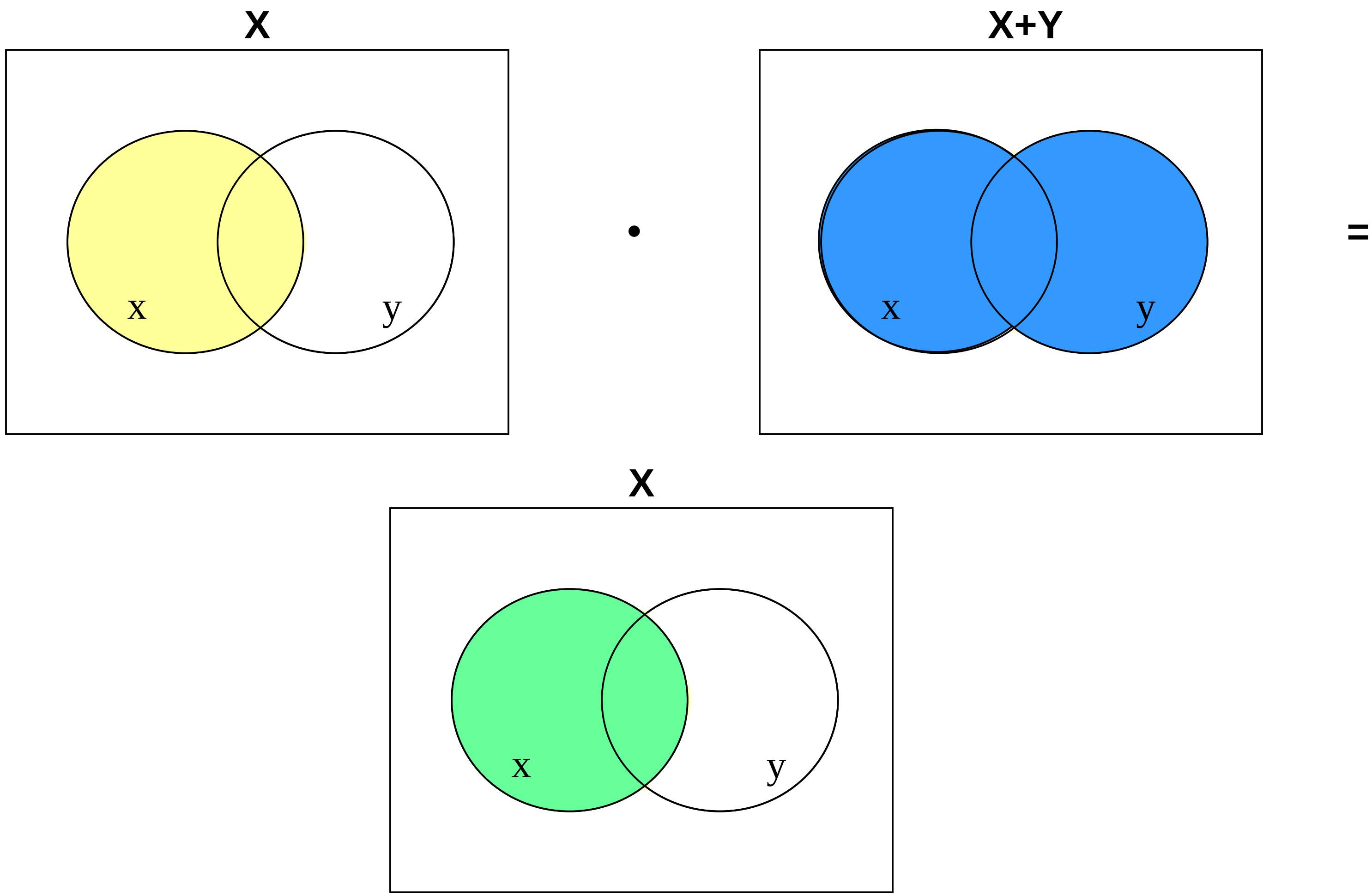
=

X



T9’ Proof

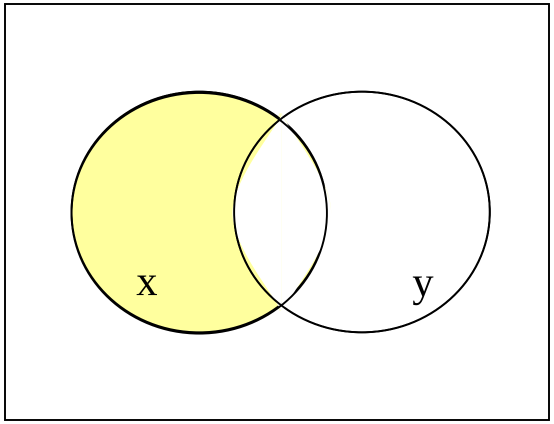
$$X(X+Y) = X$$



T10 Proof

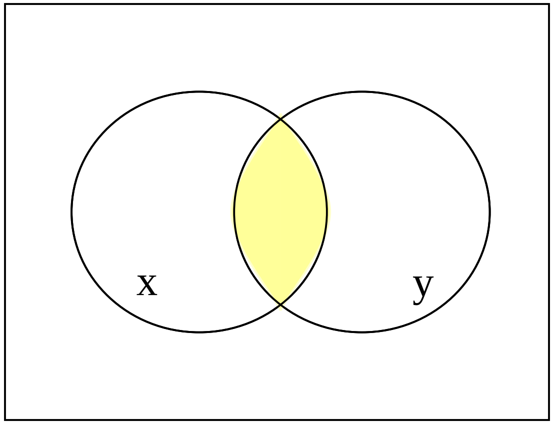
$$XY' + XY = X$$

XY'



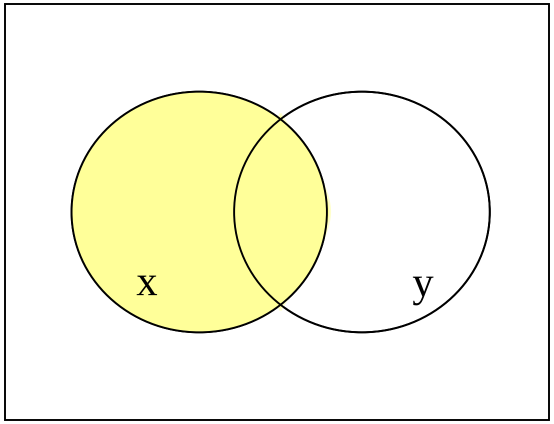
+

XY



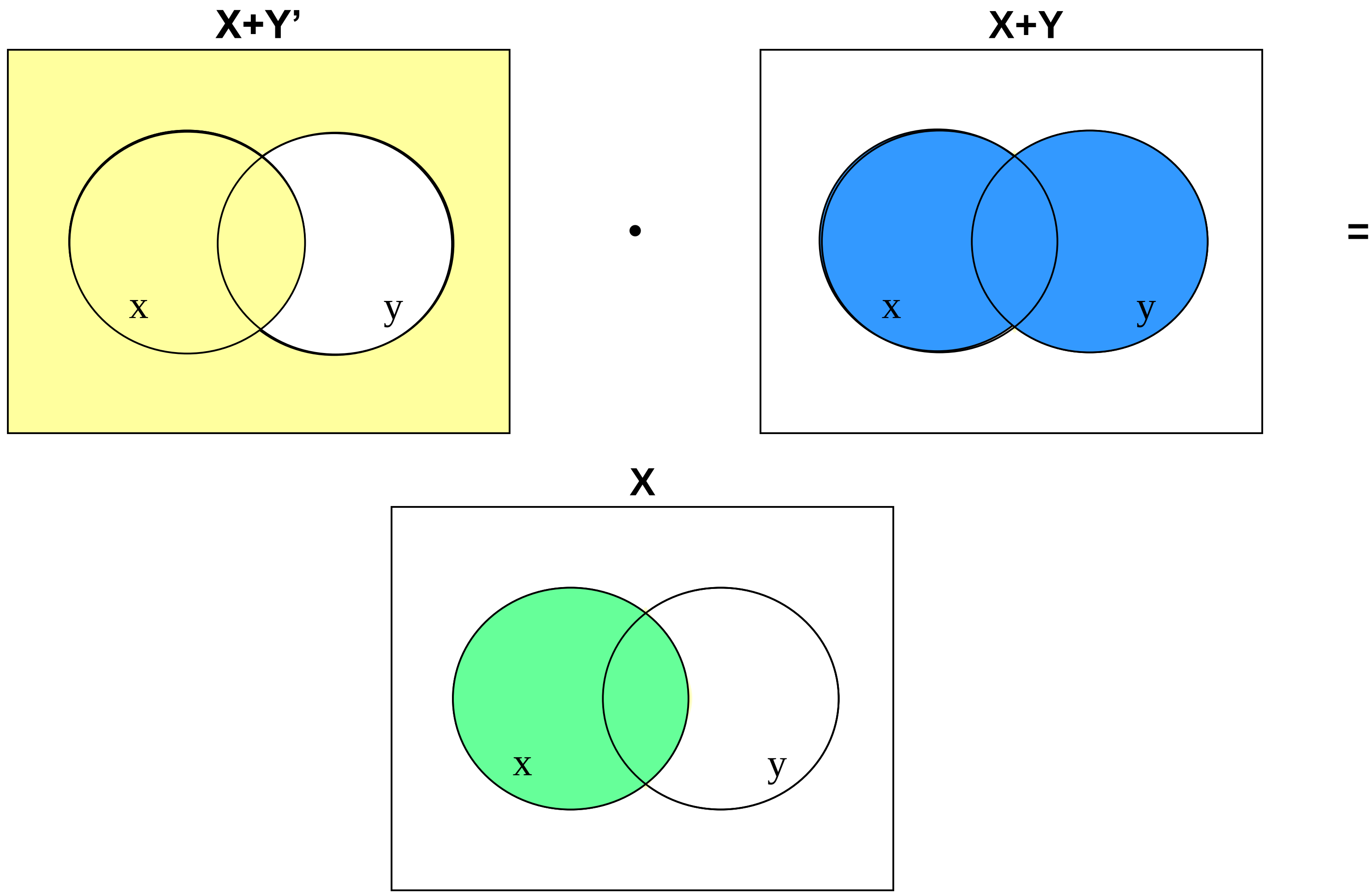
=

X



T10' Proof

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

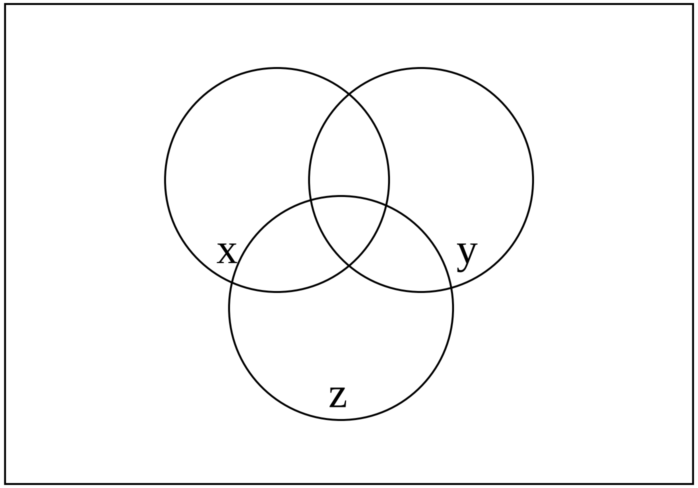


Proof by Other Theorems

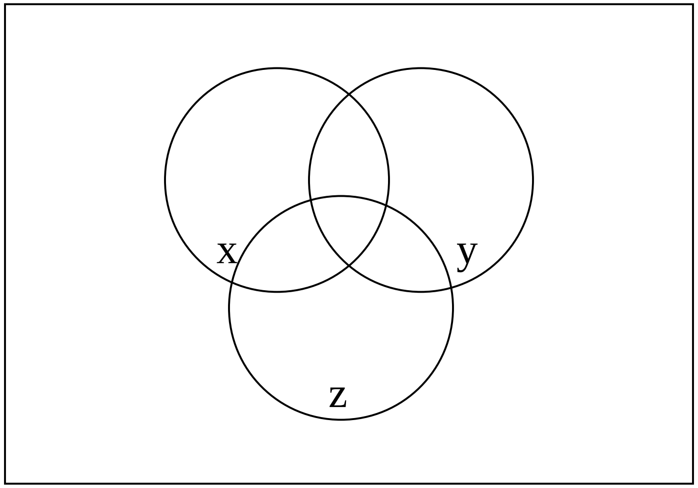
T9: X+XY = X	T9': X(X+Y) = X
T10: XY'+XY = X	T10': (X+Y')(X+Y) = X

T11 Proof

- Proof T11: XY + X'Z + YZ = XY + X'Z



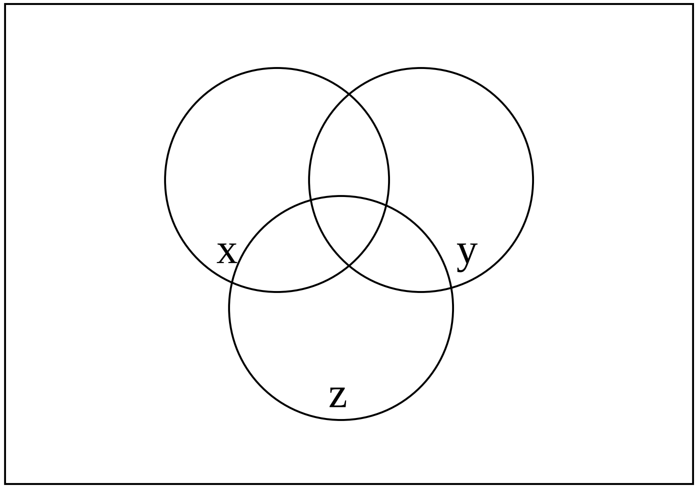
XY+X'Z+YZ



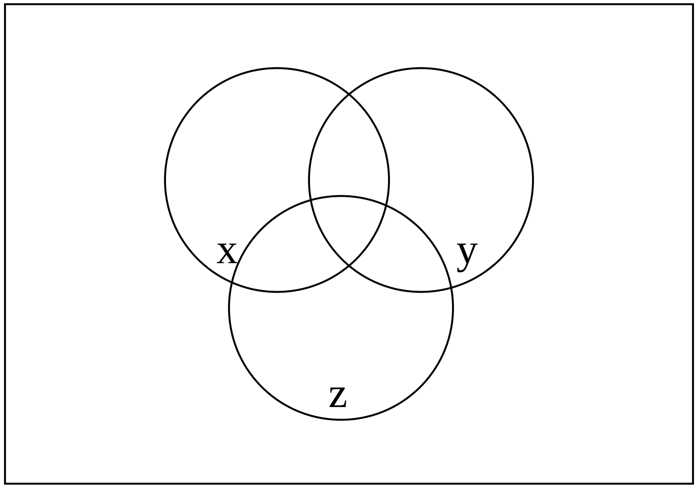
XY+X'Z

T11' Proof

- Proof T11: (X+Y)(X'+Z)(Y+Z) = (X+Y)(X'+Z)



(X+Y)(X'+Z)(Y+Z)



(X+Y)(X'+Z)

DeMorgan’s Theorem

- Consider the statement
  “You will get a Good grade if you do your Homework and go to Class”
- When will you get a bad grade?
- Consider the statement:
  “USC will Win if we Play better or UCLA Turns the ball over”
- When will USC lose?

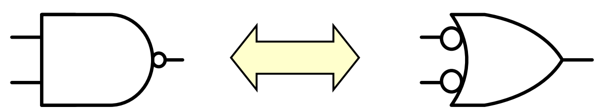
DeMorgan's Theorem

- Inverting output of an AND gate = inverting the inputs of an OR gate
- Inverting output of an OR gate = inverting the inputs of an AND gate

A function's inverse is equivalent to inverting all the inputs and changing AND to OR and vice versa

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

$\overline{A \bullet B}$

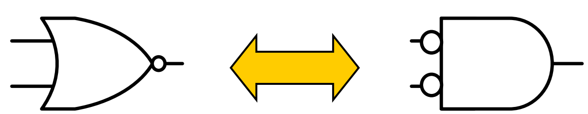


$\overline{A}+\overline{B}$

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

$\overline{A+B}$



$\overline{A}\bullet\overline{B}$

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

DeMorgan’s Theorem

$$F = (\overline{X}+Y) + \overline{Z} \cdot (Y+W)$$

To find F’, invert both sides of the equation and then use DeMorgan’s theorem to simplify…

$$\overline{F} = \overline{(\overline{X}+Y) + \overline{Z} \cdot (Y+W)}$$

$$\overline{F} = \overline{(\overline{X}+Y)} \cdot \overline{(\overline{Z} \cdot (Y+W))}$$

$$\overline{F} = (\overline{\overline{X}} \cdot \overline{Y}) \cdot (\overline{\overline{Z}} + \overline{(Y+W)})$$

$$\overline{F} = (X \cdot \overline{Y}) \cdot (Z + (\overline{Y} \cdot \overline{W}))$$

Generalized DeMorgan's Theorem

$$F'(X_1,\ldots,X_n,+,\bullet) = F(X_1',\ldots,X_n',\bullet,+)$$

To find F', swap AND's and OR's and complement each literal. However, you must maintain the original order of operations.

Note: This parentheses doesn't matter (we are just OR'ing X', Y, and the following subexpression)

$$F = (\overline{X}+Y) + \overline{Z} \bullet (Y+W)$$

$$F = \overline{X}+Y + (\overline{Z} \bullet (Y+W))$$

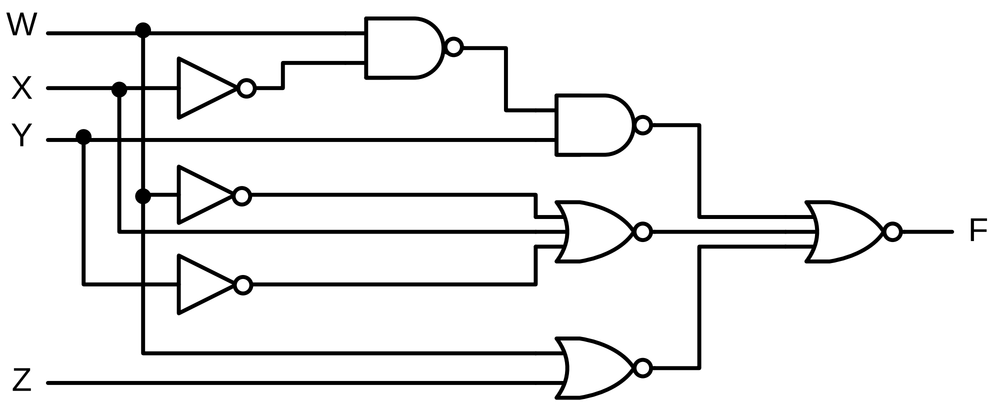
Fully parenthesized to show original order of ops.

$$\overline{F} = X \bullet \overline{Y} \bullet (Z + (\overline{Y} \bullet \overline{W}))$$

AND's & OR's swapped
Each literal is inverted

DeMorgan's Theorem Example

- Cancel as many bubbles as you can using DeMorgan's theorem then convert to either SOP or POS.



OLD LECTURE 5

Unique Representations

- Canonical => Same functions will have same representations
- Truth Tables along with Canonical Sums and Products specify a function *uniquely*
- Equations/circuit schematics are NOT inherently canonical

Truth Table

x	y	z	P
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Canonical Sum

$$P = \sum_{x,y,z} \underbrace{(2,3,5,7)}_{\text{rows where P=1}}$$

rows where P=1

Yields SOP equation, AND-OR circuit

Canonical Product

$$P = \prod_{x,y,z} \underbrace{(0,1,4,6)}_{\text{rows where P=0}}$$

rows where P=0

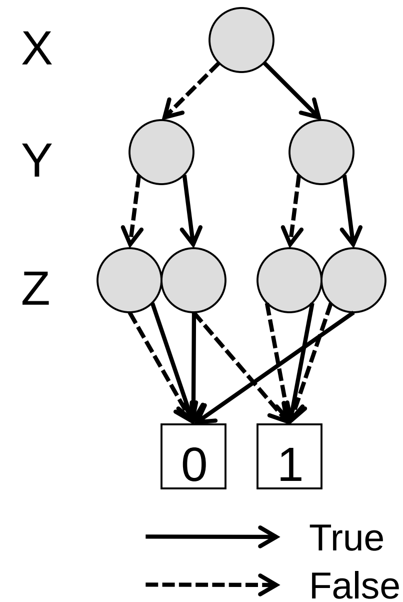
Yields POS equation, OR-AND circuit

Binary Decision Diagram

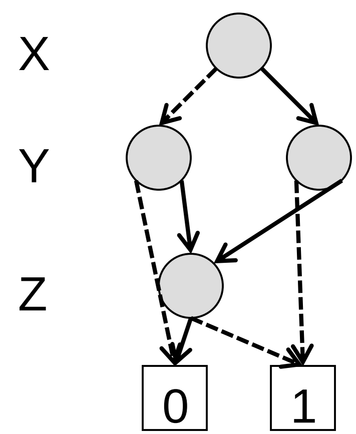
- Graph (binary tree) representation of logic function
- Vertex = Variable/Decision
- Edge = Variable value (T / F)

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

X	Y	Z	F
0	0	0	0
		1	0
	1	0	1
		1	0
1	0	0	1
		1	1
	1	0	0
		1	1

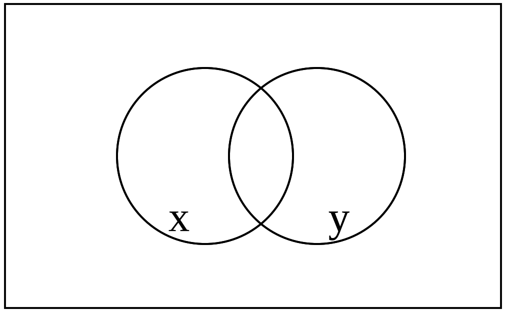


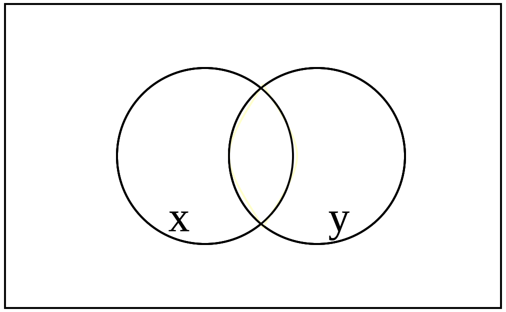
BDD for F



Venn Diagrams Practice

$$F = (X \cdot Y)'$$

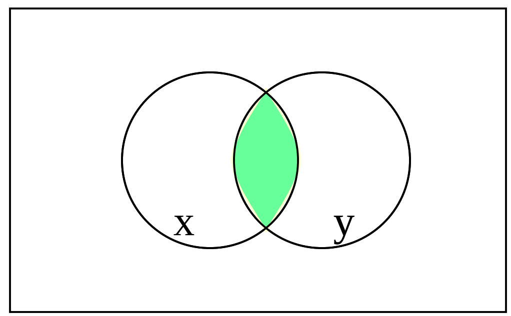




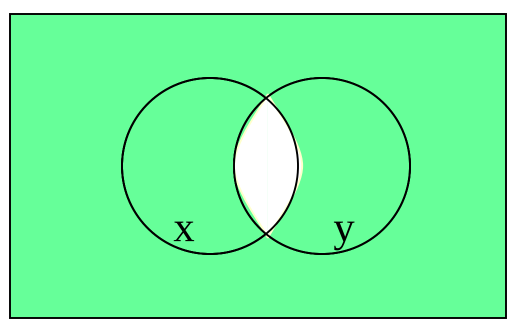
Venn Diagrams Answer

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

X•Y

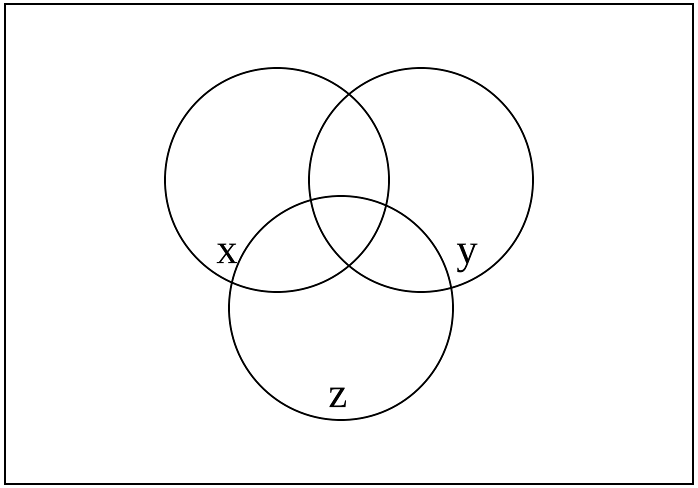


F = (X•Y)'



3-Variable Venn Diagrams

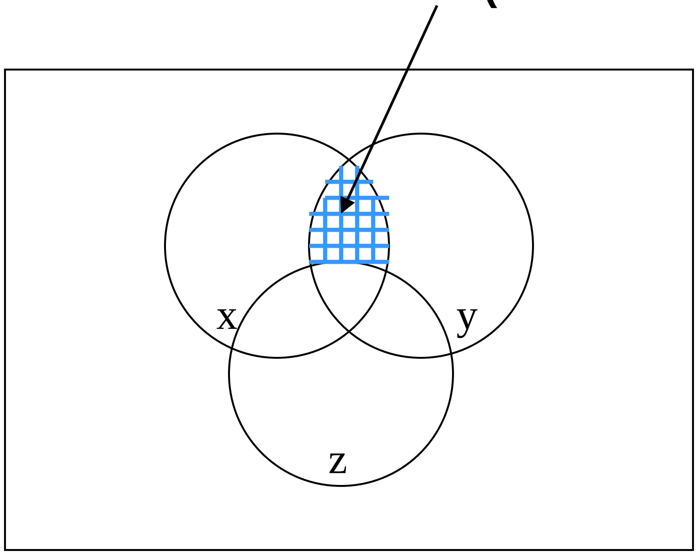
- All the same rules apply
- Since there are 3-variables, we must have 8 disjoint regions (1 for each combination / minterm / maxterm)



3-Variable Venn Diagrams

- Each area represents a different combination of X, Y, and Z

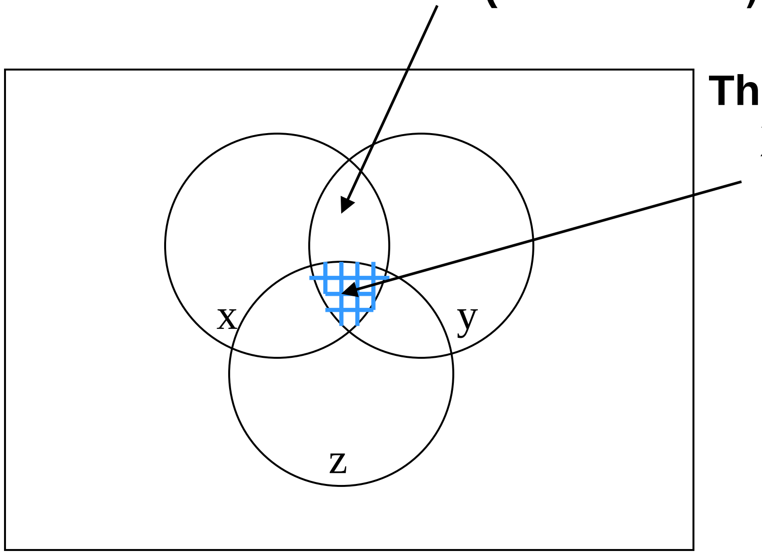
This area represent where X=1 and Y=1 and Z=0 (i.e. F=X•Y•Z')



3-Variable Venn Diagrams

- Each area represents a different combination of X, Y, and Z

This area represent where
X=1 and Y=1 and Z=0
(i.e. F=X•Y•Z')



This area represent where
X=1 and Y=1 and Z=1
(i.e. F=X•Y •Z)

3-Variable Venn Diagrams

- Each area represents a different combination of X, Y, and Z