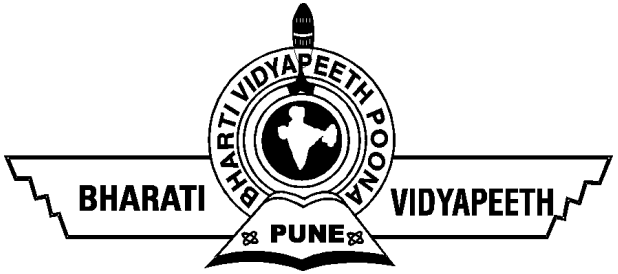
DISCRETE STRUCTURES
UNIT II



Poset and Lattices

Poset

Suppose R is a relation on set S which satisfy the following properties

(Reflexive) for any a $\in$ S we have aRa

(Antisymmetric) if aRb and bRa then a=b

(Transitive) if aRb and bRc then aRc.

Then this relation R is called partial order relation and the set S is called partially ordered set or POSET. A partial order relation is usually denoted by the symbol $\leq$

Means (S , $\leq$) is a partially ordered set.

Comparable elements

Two elements x, y in a partially ordered set (A, ≤) are said to be comparable if either $x \leq y$ or $y \leq x$

Uncomparable elements

Two elements x, y in a partially ordered set (A, ≤) are said to be uncomparable if

$x \leq y$ or $y \leq x$ both do not hold

Ex in the poset (z+ , /) are the integers 3 and 9 comparable? Are 5 and 7 comparable

Yes 3 and 9 are comparable but 5 and 7 and 7 and 5 are not

Totally Ordered Set

If $(S , \leq)$ is a poset and every two element of a set are comparable then S Is called totally ordered set or linearly ordered set.Totally ordered set is also known as chain.

Ex the poset $(Z , \leq)$ is a totally ordered set

Hasse Diagram Of A Poset

If (S, ≤) is a poset then its hasse diagram is drawn as follows

The element of S are represented by dots

Since a partial order is reflexive , hence each vertex of A must be related to itself , so the edges from a vertex to itself are deleted in hasse diagram.

Since a partial order is transitive , hence whenever aRb and bRc we have aRc. Eliminate all edges that are implied by the transitive property .

If a vertex'a' is connected to vertex 'b' by an edge i.e. aRb then vertex b appears above vertex 'a' . Therefore arrows may be ommitted from the edges in hasse diagram.

Q1 consider the set A= {k,l,m,n,p} and the corresponding relation R = { (k , k) (l,l) (m, m) (n,n) (p,p) (k,m) (k,l) (k,n) (k,p) (m,n) (m,p) (n,p) (l,p)}

construct the directeg graph and corresponding hasse diagram.

Q2 let D= {1,2,4,5,10,20,25,50,100} and let the relation is for divisibility

Determine the GLB of B where B= {10,20}

Determine the LUB of B where B= {10,20}

Determine the GLB of B where B= {5,10,20,25}

Determine the LUB of B where B= {5,10,20,25}

Maximal And Minimul Element

An element of a Poset is called maximal if it is not less than any element of the poset .

That is a is maximal in the poset $(S, \leq)$ if there is no $b \in S$ such that $a<b$

A poset may have more than one maximal element.

An element of a Poset is called minimal if it is not greater than any element of the poset .

That is a is minimal in the poset $(S, \leq)$ if there is no $b \in S$ such that $b< a$

A poset may have more than one minimal element.

Maximum and minimum elements are easy to spot in hasse diagram they are top and bottom element .

Greatest And Least Element

An element in a poset that is greater then every other element is called the greatest element .

Greatest element is unique when it exists

An element is called least element if it is less then all other element in POSET

Least element is unique when it exists.

Upper And Lower Bound

An element that is greater then all the elements in a subset A of a poset (S , ≤) .If u is an element of S such that a ≤ u for all elements a ∈ A then u is called an upper bound of A .

An element that is less then all the elements in A if l is an element of S such that l ≤ a for all elements a ∈ A then l is called lower bound of A.

LEAST UPPER BOUND AND GREATEST LOWER BOUND

The element x is called the least upper bound (SUPREMUM) of the subset A if x is an upper bound that is less the every other upper bound of A it is also calledit will be unique element

The element y is called the greatest lower bound (INFIMUM)of the subset A if y is a lower bound that is greater then every other lower bound of A it will be unique element.

Isomorphic Ordered Set

If X and Y are partially ordered sets. A one to one function f:X---→Y is called an isomorphic mapping from X in to Y if f preserves the ordered relation, means if the following two conditions are hold for any pair a and a' in X:

If $a \leq a'$ then $f(a) \leq f(a)'$

If a || a' (noncomparable) then f(a) || f(b)

And if A and B are linearly ordered, then only (a) is needed for f to be an isomorphic mapping

WELL ORDERED SET

A set $(S , \leq)$ is a well ordered set if it is a poset such that $\leq$ is a total ordering and such that every non empty subset of S has a least element.

LATTICES

A lattice is a partially ordered set $(L, \leq)$ in which every pair of element $a, b \in L$ has a greatest lower bound and a least upper bound.

The greatest lower bound of a subset $\{a,b\} \subseteq L$ will be denoted by $a * b$ and the least upper bound by $a \oplus b$. GLB $\{a,b\} = a * b$ the **meet** or product of a and b, and the LUB $\{a, b\} = a \oplus b$ the **join** or sum of a and b.

Properties Of Lattices

If two binary operations of meet and join on a lattice $(L, \leq)$ are denoted by $*$ and $\oplus$ then for any $a, b, c \in L$, we have

$a * a = a$ (idempotent law)	$a \oplus a = a$
$a * b = b * a$ (commutative law)	$a \oplus b = b \oplus a$
$(a * b) * c = a * (b * c)$ (associative law)	$(a \oplus b) \oplus c = a \oplus (b \oplus c)$
$a * (a \oplus b) = a$ (absorption law)	$a \oplus (a * b) = a$

Bounded Lattice

In a bounded lattice $(L, *, \oplus, 0, 1)$, an element $b \in L$ is called a complement of an element $a \in L$ if

$a * b = 0$ and $a \oplus b = 1$

Complemented Lattice

A lattice $(L, *, \oplus, 0, 1)$ is said to be a complemented lattice if evewry element of L has atleast one complement.

Boolean Algebra

Boolean Algebra named after George Boole who used it to study human logical reasoning – calculus of proposition.

Events : *true* or *false*

Connectives : a *OR* b; a *AND* b, *NOT* a

Example: Either "it has rained" *OR* "someone splashed water", "must be tall" *AND* "good vision

a	b	a AND b
F	F	F
F	T	F
T	F	F
T	T	T

a	b	a OR b
F	F	F
F	T	T
T	F	T
T	T	T

a	NOT a
F	T
T	F

Two-valued Boolean Algebra

Set of elements: {0,1}

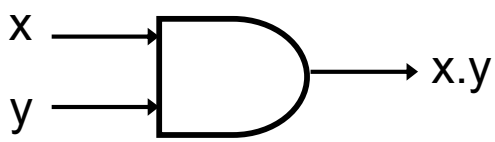
Set of operations: { ., + , ¬ }

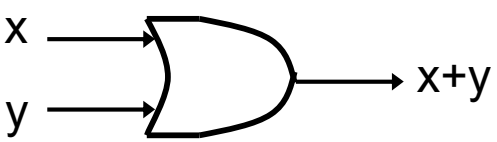
Sometimes denoted by ’, for example **a’**

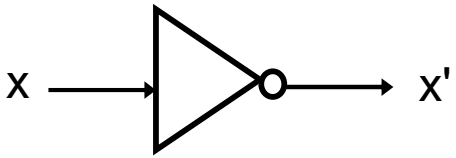
x	y	x . y
0	0	0
0	1	0
1	0	0
1	1	1

x	y	x + y
0	0	0
0	1	1
1	0	1
1	1	1

x	¬x
0	1
1	0







Signals: High = 5V = 1; Low = 0V = 0

Boolean Algebra Postulates

A **Boolean algebra** consists of a set of elements *B*, with two binary operations {+} and {.} and a unary operation {'}, such that the following axioms hold:

The set *B* contains at least two distinct elements x and y.

Closure: For every x, y in *B*,

- x + y is in *B*
- x . y is in *B*

Commutative laws: For every x, y in *B*,

- x + y = y + x
- x . y = y . x

Associative laws: For every x, y, z in *B*,

- $(x + y) + z = x + (y + z) = x + y + z$
- $(x \cdot y) \cdot z = x \cdot (y \cdot z) = x \cdot y \cdot z$

Identities (0 and 1):

- $0 + x = x + 0 = x$ for every x in *B*
- $1 \cdot x = x \cdot 1 = x$ for every x in *B*

Distributive laws: For every x, y, z in *B*,

- $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$
- $x + (y \cdot z) = (x + y) \cdot (x + z)$

Complement: For every x in *B*, there exists an element x' in *B* such that

- $x + x' = 1$
- $x \cdot x' = 0$

Duality

Duality Principle – every valid Boolean expression (equality) remains valid if the operators and identity elements are interchanged, as follows:

$+ \leftrightarrow .$

$1 \leftrightarrow 0$

Example: Given the expression

$$a + (b.c) = (a+b).(a+c)$$

then its dual expression is

$$a . (b+c) = (a.b) + (a.c)$$

Standard Forms

Certain types of Boolean expressions lead to gating networks which are desirable from implementation viewpoint.

Two Standard Forms:

Sum-of-Products and ***Product-of-Sums***

Literals: a variable on its own or in its complemented form.
Examples: x, x' , y, y'

Product Term: a single literal or a logical product (AND) of several literals.

Examples: x, x.y.z', A'.B, A.B, e.g'.w.v

Standard Forms

Sum Term: a single literal or a logical sum (OR) of several literals.

Examples: x, x+y+z', A'+B, A+B, c+d+h'+j

Sum-of-Products (SOP) Expression: a product term or a logical sum (OR) of several product terms.

Examples: x, x+y.z', x.y'+x'.y.z, A.B+A'.B',
A + B'.C + A.C' + C.D

Product-of-Sums (POS) Expression: a sum term or a logical product (AND) of several sum terms.

Examples: x, x.(y+z'), (x+y').(x'+y+z),
(A+B).(A'+B'), (A+B+C).D'.(B'+D+E')

Standard Forms

Every Boolean expression can either be expressed as sum-of-products or product-of-sums expression.

Examples:

SOP: $x'.y + x.y' + x.y.z$

POS: $(x + y').(x' + y).(x' + z')$

both: $x' + y + z$ or $x.y.z'$

neither: $x.(w' + y.z)$ or $z' + w.x'.y + v.(x.z + w')$

Minterm & Maxterm (1/3)

Consider two binary variables x, y.

Each variable may appear as itself or in complemented form as literals (i.e. x, x' & y, y')

For two variables, there are four possible combinations with the AND operator, namely:

x'.y', x'.y, x.y', x.y

These product terms are called the *minterms*.

A **minterm** of *n* variables is the product of *n* literals from the different variables.

Minterm & Maxterm

In general, n variables can give 2^n minterms.

In a similar fashion, a **maxterm** of n variables is the sum of n literals from the different variables.

Examples: x'+y', x'+y, x+y',x+y

In general, n variables can give 2^n maxterms.

Minterm & Maxterm

The minterms and maxterms of 2 variables are denoted by m0 to m3 and M0 to M3 respectively:

Minterms				Maxterms	
x	y	term	notation	term	notation
0	0	x'.y'	m0	x+y	M0
0	1	x'.y	m1	x+y'	M1
1	0	x.y'	m2	x'+y	M2
1	1	x.y	m3	x'+y'	M3

Each minterm is the complement of the corresponding maxterm:

Example: m2 = x.y'

m2' = (x.y')' = x' + (y')' = x'+y = M2

Canonical Form: Sum of Minterms

What is a canonical/normal form?

- A unique form for representing something.

Minterms are product terms.

- Can express Boolean functions using Sum-of-Minterms form.

Canonical Form: Sum of Minterms

a) Obtain the truth table.

Example:

x	y	z	F 1	F 2	F 3
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	0	0
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	1	1	0
1	1	1	0	1	0

Canonical Form: Sum of Minterms

b) Obtain Sum-of-Minterms by gathering/summing the minterms of the function (where result is a 1)

$F1 = x.y.z' = \Sigma m(6)$

$F2 = x'.y'.z + x.y'.z' + x.y'.z + x.y.z' + x.y.z$

$= \Sigma m(1,4,5,6,7)$

$F3 = x'.y'.z + x'.y.z$

$+ x.y'.z' + x.y'.z$

$= \Sigma m(1,3,4,5)$

x	y	z	F1	F2	F3
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	0	0
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	1	1	0
1	1	1	0	1	0

Canonical Form: Product of Maxterms

Maxterms are sum terms.

For Boolean functions, the maxterms of a function are the terms for which the result is 0.

Boolean functions can be expressed as Products-of-Maxterms.

Canonical Form: Product of Maxterms

E.g.: $F2 = \Pi M(0,2,3) = (x+y+z).(x+y'+z).(x+y'+z')$

$F3 = \Pi M(0,2,6,7)$

$= (x+y+z).(x+y'+z).(x'+y'+z).(x'+y'+z')$

x	y	z	F 1	F 2	F 3
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	0	0
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	1	1	0
1	1	1	0	1	0

Canonical Form: Product of Maxterms

Why is this so? Take F2 as an example.

$$F2 = \Sigma m(1,4,5,6,7)$$

The complement function of F2 is:

$$F2' = \Sigma m(0,2,3)$$

$$= m0 + m2 + m3$$

(Complement functions' minterms are the opposite of their original functions, i.e. when original function = 0)

x	y	z	F 2	F 2 '
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

Canonical Form: Product of Maxterms

From previous slide, F2' = m0 + m2 + m3

Therefore:

$$F2 = (m0 + m2 + m3)'$$

$$= m0' \cdot m2' \cdot m3' \qquad \text{DeMorgan}$$

$$= M0 \cdot M2 \cdot M3 \qquad m_x' = M_x$$

$$= \Pi M(0,2,3)$$

Every Boolean function can be expressed as either Sum-of-Minterms or Product-of-Maxterms.

Conversion of Canonical Forms

Sum-of-Minterms $\Rightarrow$ Product-of-Maxterms

- Rewrite minterm shorthand using maxterm shorthand.
- Replace minterm indices with indices not already used.

  Eg: $F1(A,B,C) = \sum m(3,4,5,6,7) = \prod M(0,1,2)$

Product-of-Maxterms $\Rightarrow$ Sum-of-Minterms

- Rewrite maxterm shorthand using minterm shorthand.
- Replace maxterm indices with indices not already used.

  Eg: $F2(A,B,C) = \prod M(0,3,5,6) = \sum m(1,2,4,7)$

Conversion of Canonical Forms

Sum-of-Minterms of F $\Rightarrow$ Sum-of-Minterms of F'

- In minterm shorthand form, list the indices not already used in F.

Eg: $F1(A,B,C) = \sum m(3,4,5,6,7)$

$F1'(A,B,C) = \sum m(0,1,2)$

Product-of-Maxterms of F $\Rightarrow$ Prod-of-Maxterms of F'

- In maxterm shorthand form, list the indices not already used in F.

Eg: $F1(A,B,C) = \prod M(0,1,2)$

$F1'(A,B,C) = \prod M(3,4,5,6,7)$

Conversion of Canonical Forms

Sum-of-Minterms of F $\Rightarrow$ Product-of-Maxterms of F'

- Rewrite in maxterm shorthand form, using the same indices as in F.

Eg: $F1(A,B,C) = \sum m(3,4,5,6,7)$

$F1'(A,B,C) = \prod M(3,4,5,6,7)$

Product-of-Maxterms of F $\Rightarrow$ Sum-of-Minterms of F'

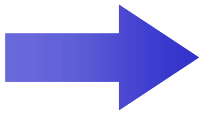
- Rewrite in minterm shorthand form, using the same indices as in F.

Eg: $F1(A,B,C) = \prod M(0,1,2)$

$F1'(A,B,C) = \sum m(0,1,2)$

A two-variable function has four possible minterms. We can re-arrange these minterms into a Karnaugh map.

x	y	minterm
0	0	x'y'
0	1	x'y
1	0	xy'
1	1	xy



x \ y	0	1
0	x'y'	x'y
1	xy'	xy

x \ y	0	1
0	x'y'	x'y
1	xy'	xy

x \ y	y'	y
x'	x'y'	x'y
x	xy'	xy

Now we can easily see which minterms contain common literals.

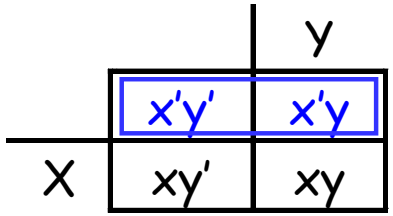
- Minterms on the left and right sides contain y' and y respectively.
- Minterms in the top and bottom rows contain x' and x respectively.

Karnaugh map simplifications

Imagine a two-variable sum of minterms:

$$x'y' + x'y$$

Both of these minterms appear in the top row of a Karnaugh map, which means that they both contain the literal x'.



$x'y' + x'y = x'(y' + y)$ [Distributive]

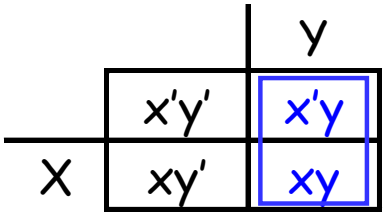
$= x' \bullet 1$ [$y + y' = 1$]

$= x'$ [$x \bullet 1 = x$]

More two-variable examples

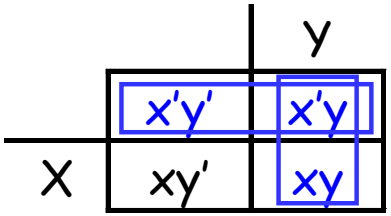
Another example expression is x'y + xy.

- Both minterms appear in the right side, where y is uncomplemented.
- Thus, we can reduce x'y + xy to just y.



How about x'y' + x'y + xy?

- We have x'y' + x'y in the top row, corresponding to x'.
- There's also x'y + xy in the right side, corresponding to y.
- This whole expression can be reduced to x' + y.



A three-variable Karnaugh map

For a three-variable expression with inputs x, y, z, the arrangement of minterms is more tricky:

X \ YZ	00	01	11	10
0	x'y'z'	x'y'z	x'yz	x'yz'
1	xy'z'	xy'z	xyz	xyz'

X \ YZ	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

X \ Y				
	x'y'z'	x'y'z	x'yz	x'yz'
X	xy'z'	xy'z	xyz	xyz'
Z				

X \ Y				
	m_0	m_1	m_3	m_2
X	m_4	m_5	m_7	m_6
Z				

Another way to label the K-map (use whichever you like):

Making the example K-map

Next up is drawing and filling in the K-map.

- Put 1s in the map for each minterm, and 0s in the other squares.
- You can use either the minterm products or the shorthand to show you where the 1s and 0s belong.

In our example, we can write f(x,y,z) in two equivalent ways.

$f(x,y,z) = x'y'z + xy'z + xyz' + xyz$

		y		
	x'y'z'	x'y'z	x'yz	x'yz'
X	xy'z'	xy'z	xyz	xyz'
		Z		

$f(x,y,z) = m_1 + m_5 + m_6 + m_7$

		y		
	m_0	m_1	m_3	m_2
X	m_4	m_5	m_7	m_6
		Z		

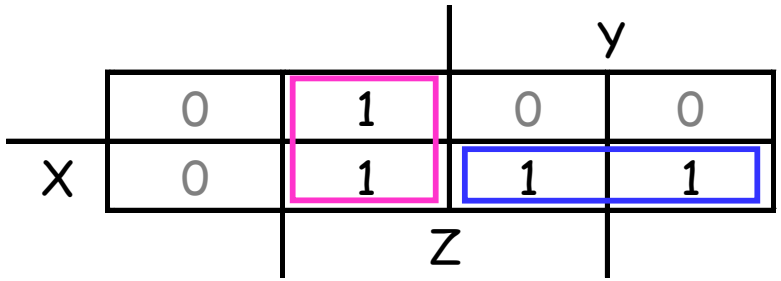
In either case, the resulting K-map is shown below.

		y		
	0	1	0	0
X	0	1	1	1
		Z		

Grouping the minterms together

The most difficult step is grouping together all the 1s in the K-map.

- Make rectangles around groups of one, two, four or eight 1s.
- All of the 1s in the map should be included in at least one rectangle.
- Do *not* include any of the 0s.

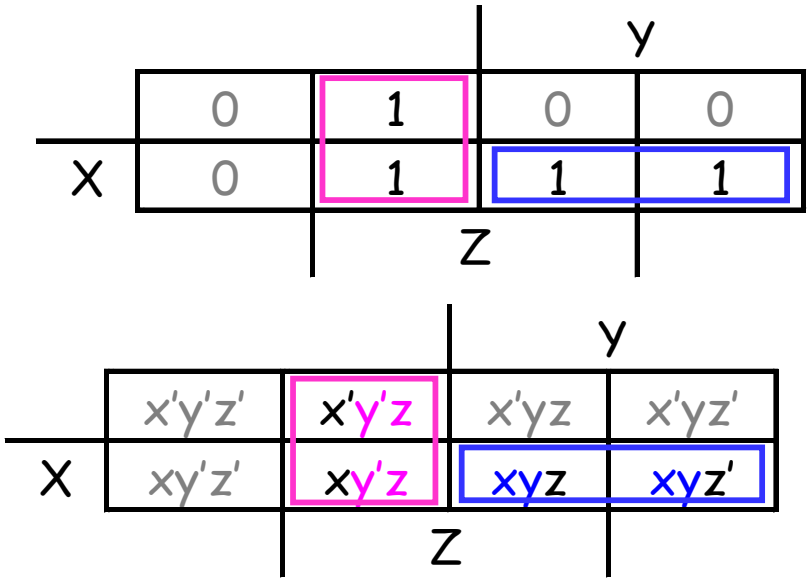


- Each group corresponds to one product term. For the simplest result:
- Make as few rectangles as possible, to minimize the number of products in the final expression.
- Make each rectangle as large as possible, to minimize the number of literals in each term.
- It's all right for rectangles to overlap, if that makes them larger.

Reading the MSP from the K-map

Finally, you can find the MSP.

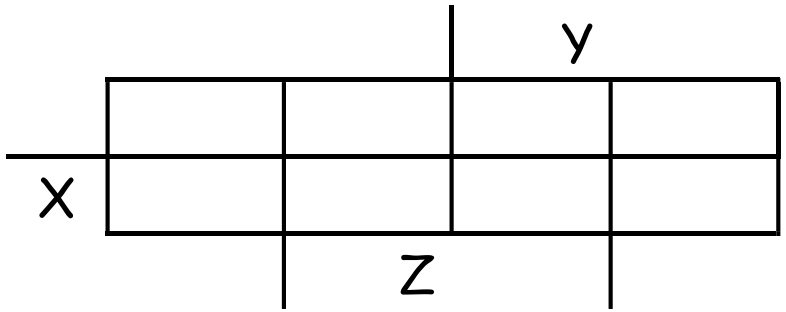
- Each rectangle corresponds to one product term.
- The product is determined by finding the common literals in that rectangle.

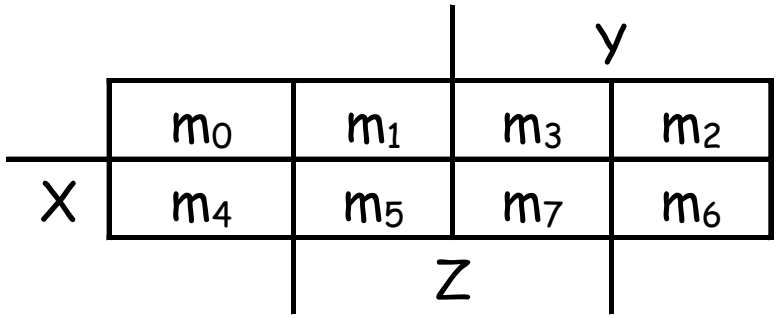


For our example, we find that xy + y'z + xz = y'z + xy. (This is one of the additional algebraic laws from last time.)

Practice K-map 1

Simplify the sum of minterms $m_1 + m_3 + m_5 + m_6$.

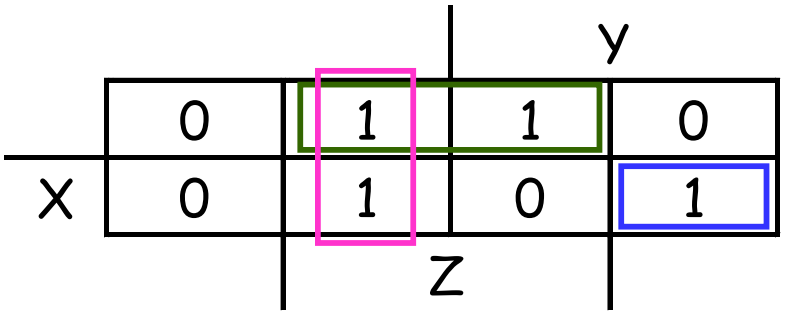




Solutions for practice K-map 1

Here is the filled in K-map, with all groups shown.

- The magenta and green groups overlap, which makes each of them as large as possible.
- Minterm m_6 is in a group all by its lonesome.



The final MSP here is x'z + y'z + xyz'.

Four-variable K-maps

We can do four-variable expressions too!

- The minterms in the third and fourth columns, *and* in the third and fourth rows, are switched around.
- Again, this ensures that adjacent squares have common literals.

			y	
	w′x′y′z′	w′x′y′z	w′x′yz	w′x′yz′
	w′xy′z′	w′xy′z	w′xyz	w′xyz′
W	wxy′z′	wxy′z	wxyz	wxyz′
	wx′y′z′	wx′y′z	wx′yz	wx′yz′
		Z		

X

			y	
	m_0	m_1	m_3	m_2
	m_4	m_5	m_7	m_6
W	m_{12}	m_{13}	m_{15}	m_{14}
	m_8	m_9	m_{11}	m_{10}
		Z		

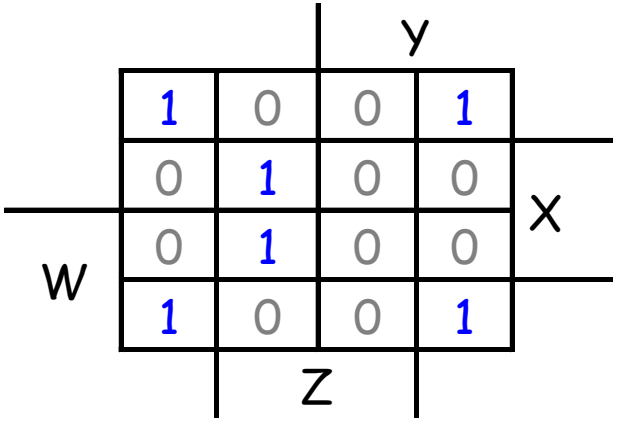
X

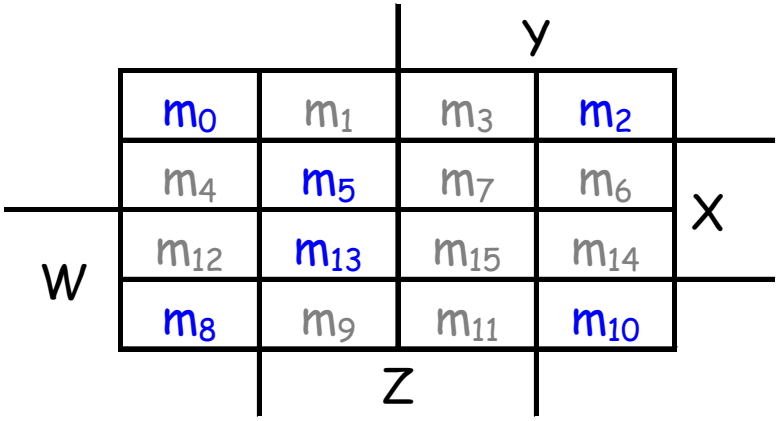
Grouping minterms is similar to the three-variable case, but:

- You can have rectangular groups of 1, 2, 4, 8 or 16 minterms.
- You can wrap around *all four* sides.

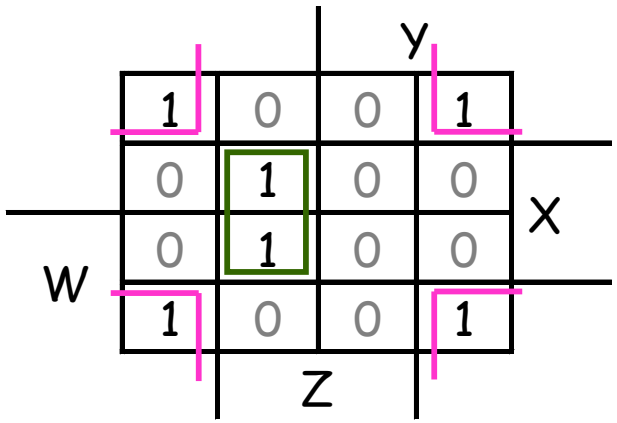
Example: Simplify $m_0+m_2+m_5+m_8+m_{10}+m_{13}$

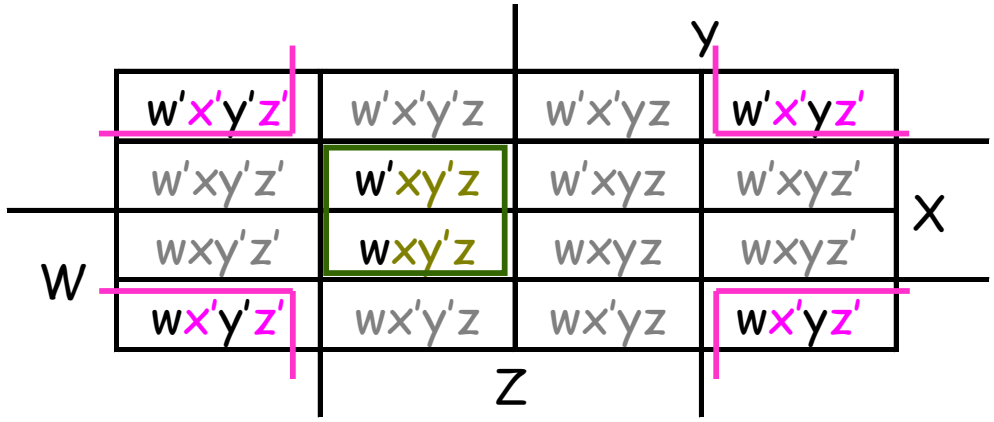
The expression is already a sum of minterms, so here's the K-map:





We can make the following groups, resulting in the MSP x'z' + xy'z.





Recurrence Relation

Recurrence Relation

A recurrence relation is a recursive formula that counts the number of ways to do a procedure involving **n** objects in terms of the number of ways to do it with fewer objects.

E.g., $a_n = c_1 a_{n-1} + c_2 a_{n-2}$, $a_1 = 0, a_2 = 1$

A recurrence relation's starting values are called initial conditions.

Reccurence Relation

Proving things about a recurrence relation usually is done by mathematical induction.

Typical forms of recurrence relations include:

- $a_n = c_1 a_{n-1} + c_2 a_{n-2} + \ldots + c_r a_{n-r}$
- $a_n = c_1 a_{n-1} + c_2$
- $a_n = c_1 a_{n-1} + f(n)$

Recursive definition of a sequence

Specify one or more initial terms

Specify rule for obtaining subsequent terms from preceding terms

We can use such definitions to solve counting problems that cannot easily be solved using techniques

Recurrence Relations

When rule for finding subsequent terms from previous is used for solving counting problems, we call the rule a recurrence relation

Stated more formally: A recurrence relation for the sequence $\{a_n\}$ is an equation that expresses a_n in terms of one or more of the previous terms of the sequence $a_0, a_1, \ldots a_{n-1}$ for all integers n with $n \geq n_0$, where n_0 is non-negative

Solutions

A sequence whose terms satisfy a recurrence relation is called a solution of the recurrence relation

Example 1: Let $\{a_n\}$ be a sequence that satisfies the recurrence relation $a_n = a_{n-1} - a_{n-2}$ for n = 2, 3, 4 …

- Suppose $a_0=3$ and $a_1=5$. What are a_2 and a_3?
- $a_2 = a_1 - a_0 = 2$, $a_3 = a_2 - a_1 = -3$

Example 2

Find the first 5 terms of a sequence defined as follows:

- recurrence relation: $an = na_{n-1} + n^2a_{n-2}$
- initial condition: $a_0 = 1$, $a_1 = 1$

Applying the rules:

$a_2 = 2(1) + (2)^2 1 = 6$

$a_3 = 3(6) + (3)^2 1 = 27$

$a_4 = 4(27) + (4)^2 6 = 204$

Example 2

Determine whether {an} is a solution of the recurrence relation $a_n = 2a_{n-1}-a_{n-2}$ for n=2, 3, 4 … where $a_n = 3n$

if $a_n = 3n$, then for $n \geq 2$:

$2a_{n-1} - a_{n-2} = 2[3(n-1)] - 3(n-2)$

$= 2(3n - 3) - 3n + 6$

$= 6n - 6 - 3n + 6 = 3n$

So $\{a_n\}$, where $a_n = 3n$, is a solution

Example 3

Determine whether $\{a_n\}$ is a solution of the recurrence relation $a_n = 2a_{n-1}-a_{n-2}$ for n=2, 3, 4 … where $a_n = 2^n$:

- By this rule, $a_0 = 2^0 = 1$; $a_1 = 2^1 = 2$; $a_2 = 2^2 = 4$
- Applying original recurrence relation:

  $a_n = 2a_{n-1} - a_{n-2}$

  $a_2 = 2a_1 - a_0$ substituting actual values:

  $4 = 2*2 - 1$

  $4 = 3$ not true, so $\{a_n\}$ where $a_n = 2^n$ is not a solution

Summary of Recurrence Relations

Initial conditions for a sequence specify terms that precede the first term where recurrence relation takes effect

Recurrence relation & initial conditions uniquely determine a sequence, providing a recursive definition of the sequence

Any term of a sequence can be found from initial conditions using recurrence relation a sufficient number of times (but there are better ways for computing certain classes of sequences)