Lecture 4
Branch-and-Bound Optimization Method

Outline

- Branch-and-Bound Method

Remembering what we talked about last time

- Will proceed to completed solution if the implicant table reduces to “empty” after eliminating all essential prime implicant columns (and associated covered rows), dominant rows, and dominated columns
- If a cyclic core remains we need to apply some exhaustive search method to find which subset of implicants from the cyclic core will yield a covering with minimum cardinality
  - The Branch-and-Bound technique is used for this purpose

BIG PICTURE

- Branch-and-Bound is a general optimization technique
- It helps us create an entire decision tree for all possible values that decision variables can take
- The resulting cost can be computed and the best among all possible scenarios can be picked

Branch and Bound

- Branch
  - Pivoting by making a decision for one column at a time
  - A decision tree is constructed one branch at a time
- Bound
  - Will help us accelerate the search
  - Use lower bound and upper bound on the cost of the optimization problem to decide
    - A certain branch of the decision tree is not worth exploring any further
    - We have identified the optimal solution

Branch-and-Bound

- Given
  - Set of decision variables (p_1, ..p_k)
    - In our case this correspond to the prime implicants
  - Upper Bound U on the cost of solution
    - A sense of what is the worst we can do
  - Lower Bound L on the cost of solution
    - A sense of what is the least cost that a feasible solution should incur

Branch-and-Bound

- Build a binary decision tree, where at each branch we make a decision about one variable
- At each branch compute the following
  - CPS = cost of Current Partial Solution
  - PL = lower bound of the cost of the remaining partial problem
  - If CPS+PL > U
    - No need to further explore that branch, hence, move up backwards in the decision tree
  - Else
    - Keep exploring the decision path

Need a way to obtain PL.
If CPS + PL > U, then this branch is

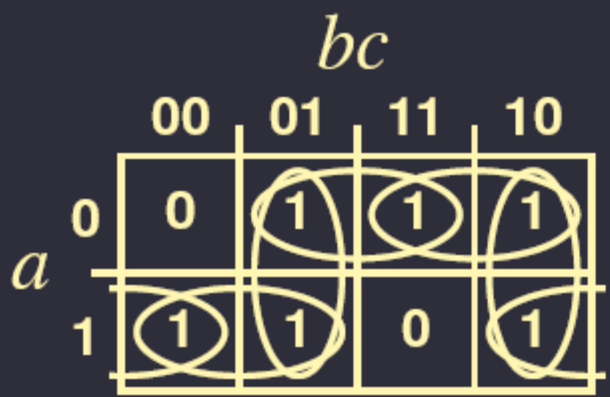
Branch-and-Bound

- Each time a complete path of the decision tree that has been traced, i.e., all decision variables have been assigned a value
  - Update U with the value of the current best solution
  - If U = L
    - Declare that optimal solution is found and stop searching other paths
  - Otherwise, keep searching downwards on all paths (that are not killed according to the first rule discussed earlier) and once the entire decision tree has been constructed declare the path with lowest cost as the optimal solution (even if U!=L at that point)

Branch-and-Bound

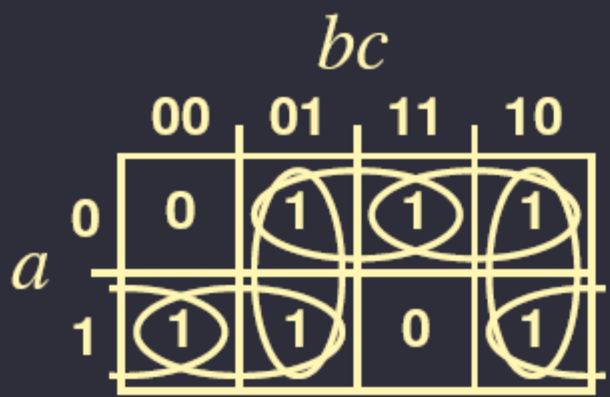
- First recap the process of exploring the implicant table by considering a simple example

Starting with the cyclic core



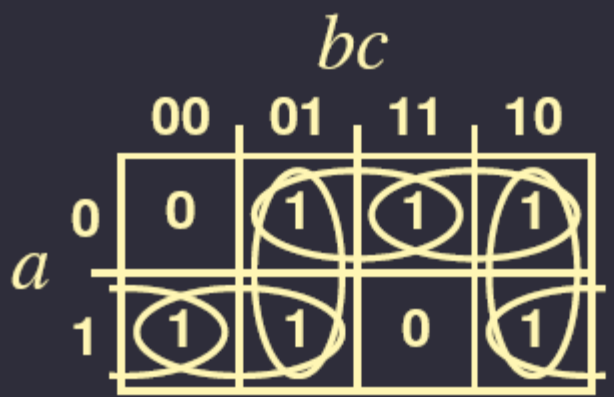
	p1	p2	p3	p4	p5	p6
001	1		1			
011	1	1				
010		1		1		
100					1	1
101			1		1	
110				1		1

p1 in the solution



	p1	p2	p3	p4	p5	p6
001	1		1			
011	1	1				
010		1		1		
100					1	1
101			1		1	
110				1		1

p1 in the solution, table reduced



	p2	p3	p4	p5	p6
010	1		1		
100				1	1
101		1		1	
110			1		1

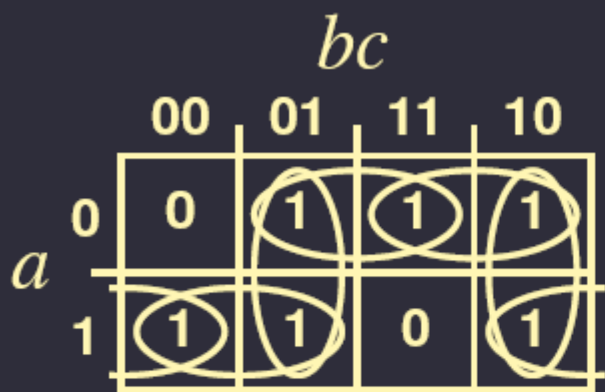
p1 in the solution

a \ bc	00	01	11	10
0	0	1	1	1
1	1	1	0	1

p4 dominates p2

	p2	p3	p4	p5	p6
010	1		1		
100				1	1
101		1		1	
110			1		1

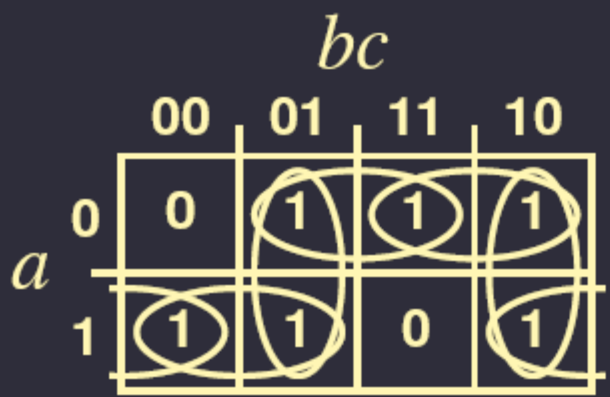
p1 in the solution



p5 dominates p3

	p2	p3	p4	p5	p6
010	1		1		
100				1	1
101		1		1	
110			1		1

p1 in the solution



After removing p2 and p3
p4 and p5 are essential now;
Solution found;

	p4	p5	p6
010	1		
100		1	1
101		1	
110	1		1

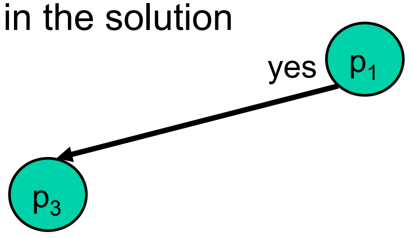
Just one solution, not necessarily

Branch-and-Bound

- Now, let's start with another hypothetical example
- 9 prime implicants $\{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9\}$
- Cost = number of prime implicants included in the solution
- Upper bound U = 10, L = 4
  - U = total number of implicants + 1 Why +1?
  - L found through a special technique, will be discussed later
    - For now, assume that you somehow know the value of L=4

Branch-and-Bound

try including p_1 in the solution



Branch-and-Bound

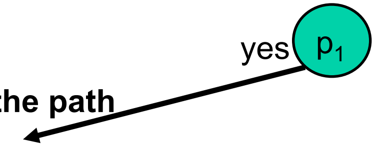
p_2 becomes essential after removing p1 from table

Sol={p_1, p_2} $\Rightarrow$ CPS = 2

PL = 3

PL+CPS = 5 < U

keep exploring the path



Branch-and-Bound

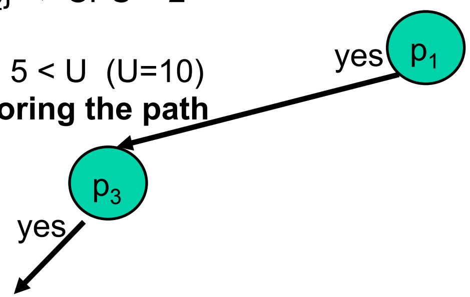
P_2 essen. after remov. p1 from table

Sol={p_1, p_2} $\Rightarrow$ CPS = 2

PL = 3

PL+CPS = 5 < U (U=10)

keep exploring the path



What if p_3 is in the solution

Sol={p_1, p_2, p_3} $\Rightarrow$ CPS = 3

PL = 2

PL+CPS = 5 $\Rightarrow$ < **U**

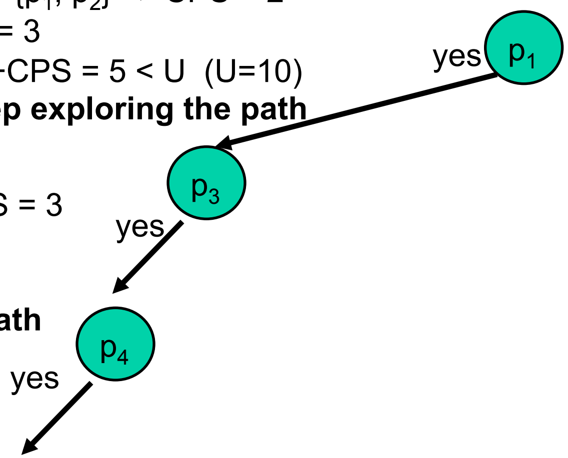
keep exploring the path

Branch-and-Bound

P_2 essen. after remov. p1 from table

Sol={p_1, p_2} $\Rightarrow$ CPS = 2
PL = 3
PL+CPS = 5 < U (U=10)
keep exploring the path

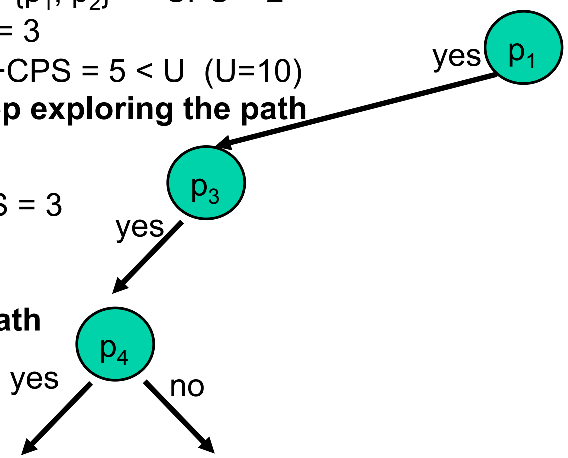
Sol={p_1, p_2, p_3} $\Rightarrow$ CPS = 3
PL = 2
PL+CPS = 5 $\Rightarrow$ < **U**
keep exploring the path



p_7 essential after removing p_4 from the table
p_7 dominates p_5, p_6, p_8, p_9
Sol={p1, p2, p3, p4, p7} $\Rightarrow$ CPS = 5
One path of the decision tree completed, i.e., all decision variables have been assigned a Yes or No value
U is updated **U = 5**
Table empty, final solution

Branch-and-Bound

P_2 essen. after remov. p1 from table
Sol={p_1, p_2} ⇒ CPS = 2
PL = 3
PL+CPS = 5 < U (U=10)
keep exploring the path



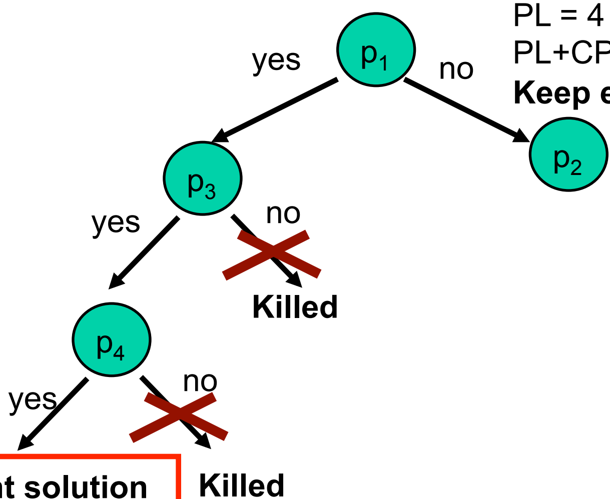
Sol={p_1, p_2, p_3} ⇒ CPS = 3
PL = 2
PL+CPS = 5 ⇒ < **U**
keep exploring the path

p_7 essential after removing p_4 from the table
p_7 dominates p_5, p_8, p_9
Sol={p1, p2, p3, p4, p7} ⇒ CPS = 5
One path of the decision tree completed, i.e., all decision variables have been assigned a Yes or No value
U is updated **U = 5**
Table empty, final solution

Backtrack upwards in the decision tree
Explore the alternative
Sol={p1, p2, p3} ⇒ CPS = 3
PL = 3
PL+CPS = 6 ⇒ **PL+CPS > U**
⇒ **Kill branch (current solution worse than best known solution so far)**

Branch-and-Bound

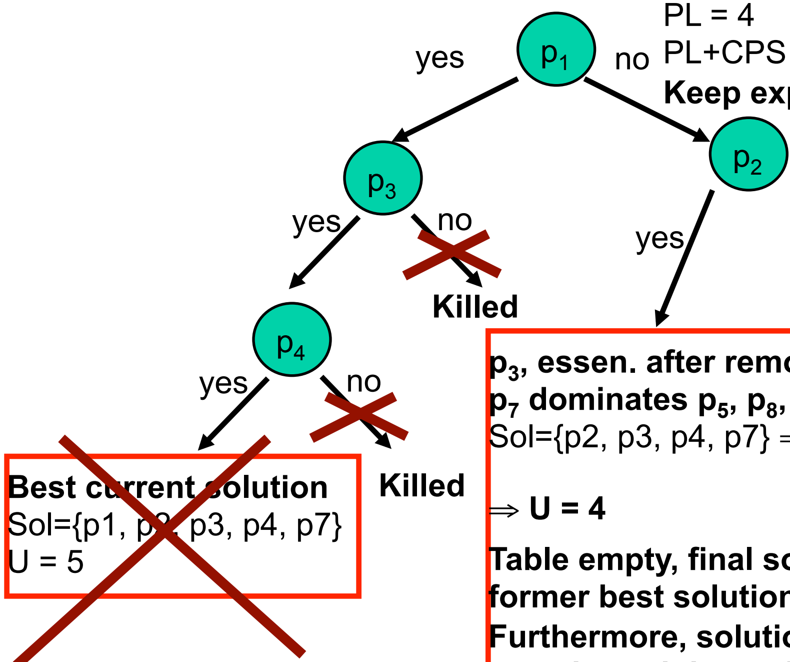
Sol={ } $\Rightarrow$ CPS = 0
PL = 4
PL+CPS = 4 $\Rightarrow$ PL+CPS < U
Keep exploring the path



Best current solution
Sol={p1, p2, p3, p4, p7}
U = 5

Branch-and-Bound

Sol={ } ⇒ CPS = 0
PL = 4
PL+CPS = 4 ⇒ PL+CPS < U
Keep exploring the path



p_3, essen. after remov. p_2 from table
p_7 dominates p_5, p_8, p_9
Sol={p2, p3, p4, p7} ⇒ CPS = 4

⇒ **U = 4**

Table empty, final solution replaces the former best solution
Furthermore, solution cost is equal to the lowerbound determined for this problem as we started:
Hence, no need to further explore/backtrack.
Optimal solution is found!

OPTIMAL SOLUTION

Branch and Bound

- Summarizing the impact of bounding on the efficiency of the exploration
  - At any given node along the decision tree
    - We are not interested in covers that have bigger cost than the current best solution
      - We can start with an arbitrarily large upper bound first
      - Set the upper bound to the best solution encountered as we go along
    - We evaluate the cost of the current solution plus the lower bound on the cost of completing it
      - If this total cost turns out larger than or equal to the current upper bound, that path is not worth exploring further
    - If the cost of any finished path equals the **global lower bound** of the covering problem
      - That path is optimal, no need to start any new recursions

Branch and Bound Strategy

- How do we know when to stop
- Two possibilities
  - We have explored the entire decision tree, all possible paths and have compared the costs of all finished recursions and declared the path with least cost as the optimal solution
  - We have hit one path with cost equal to the global lower bound of the problem

Complexity of Branch and Bound

- In the worst case we might end up exploring all possible combinations corresponding to all possible paths, which grows exponentially in number with respect to the number of decision variables

Branch-and-Bound

- Identifying the global lower bound for QM-based 2-level logic minimization
- Compute independent set I of the implicant table

$$|\text{Solution of Unate Covering}| \geq |I|$$

Finding a Lower Bound

a \ bc	00	01	11	10
0	0	1	1	1
1	1	1	0	1

	0X1	01X	X01	X10	10X	1X0
001	1		1			
011	1	1				
010		1		1		
100					1	1
101			1		1	
110				1		1

Finding a Lower Bound

a \ bc	00	01	11	10
0	0	1	1	1
1	1	1	0	1

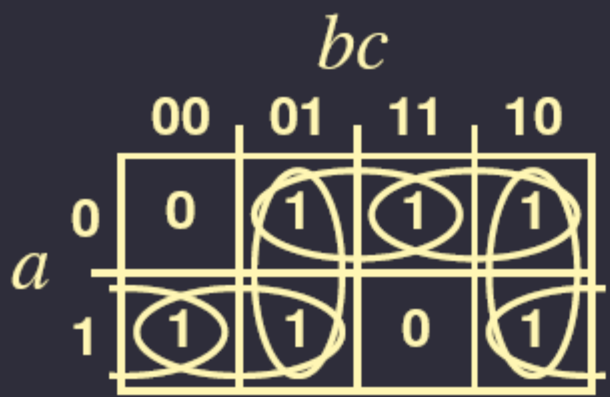
	0X1	01X	X01	X10	10X	1X0
001	1		1			
011	1	1				
010		1		1		
100					1	1
101			1		1	
110				1		1

Finding a Lower Bound

a \ bc	00	01	11	10
0	0	1	1	1
1	1	1	0	1

	0X1	01X	X01	X10	10X	1X0
001	1		1			
011	1	1				
010		1		1		
100					1	1
101			1		1	
110				1		1

Finding a Lower Bound



	0X1	01X	X01	X10	10X	1X0
001	1		1			
011	1	1				
010		1		1		
100					1	1
101			1		1	
110				1		1

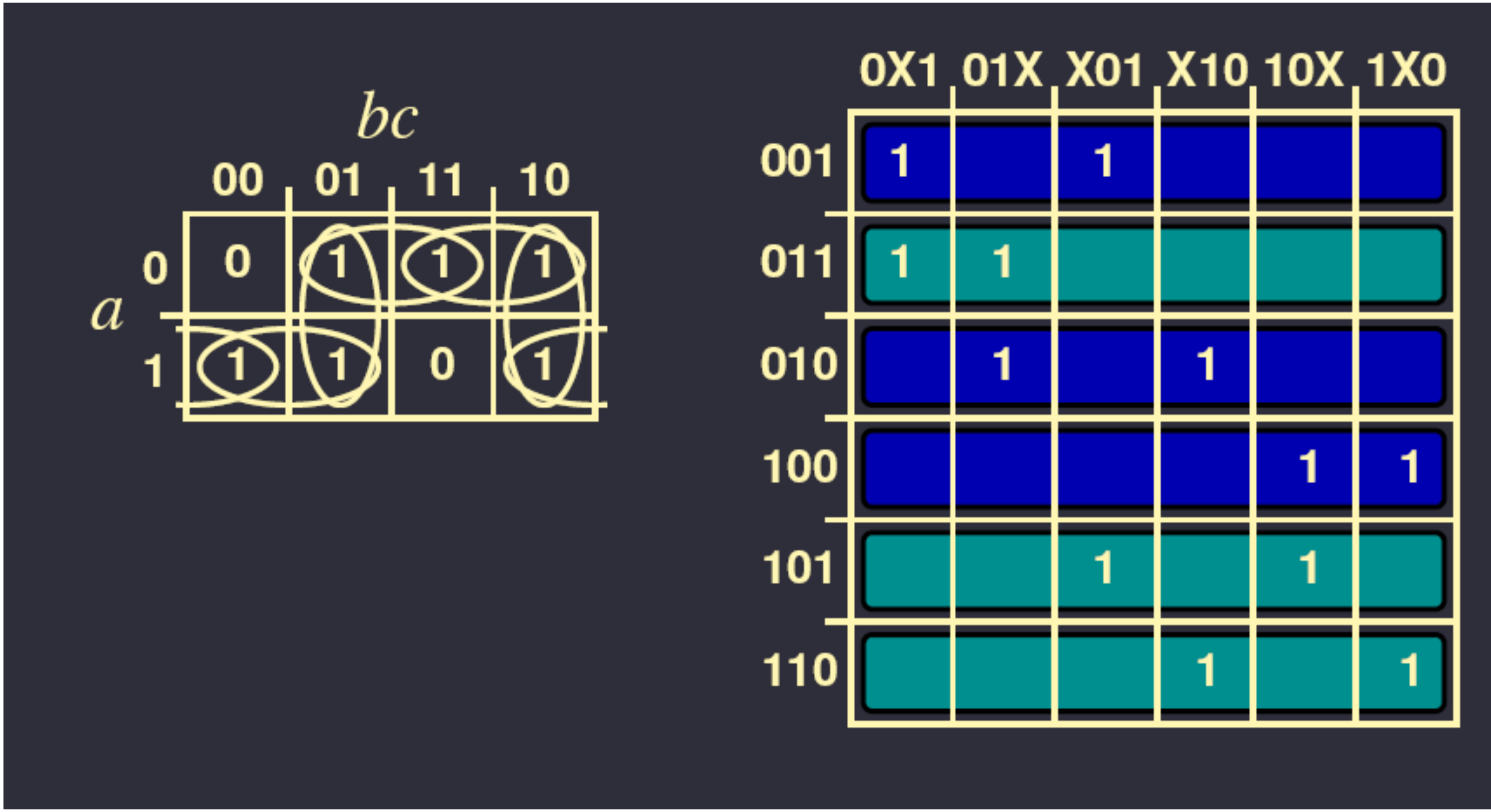
Any two rows which do not contain a "1" entry in the same location are mutually disjoint

Finding a Lower Bound

a \ bc	00	01	11	10
0	0	1	1	1
1	1	1	0	1

	0X1	01X	X01	X10	10X	1X0
001	1		1			
011	1	1				
010		1		1		
100					1	1
101			1		1	
110				1		1

Finding a Lower Bound



Finding a Lower Bound

a \ bc	00	01	11	10
0	0	1	1	1
1	1	1	0	1

	0X1	01X	X01	X10	10X	1X0
001	1		1			
011	1	1				
010		1		1		
100					1	1
101			1		1	
110				1		1

We need *at least* as many implicants as the number of disjoint rows (a.k.a. independent rows)

3 disjoint rows -> 3 columns required

need at least 3 columns to cover

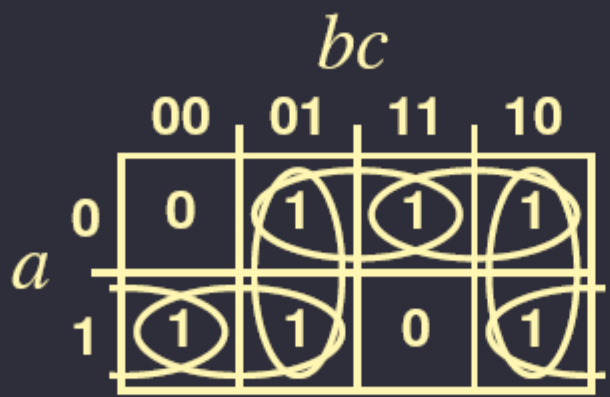
Use Bound to Constrain Search Space

- Eliminate rows covered by essential columns
- Eliminate dominant rows
- Eliminate columns dominated by other columns
- Branch-and-bound on cyclic problems
  - Use independent sets to bound
- Speed improved, still problem is NP-complete

QM Method-Recap Overview

- Compute prime implicants with a well-defined algorithm
- Start from minterms
- Merge adjacent implicants until further merging impossible
- Select minimal cover from prime implicants
  - Unate covering problem

Starting with the cyclic core

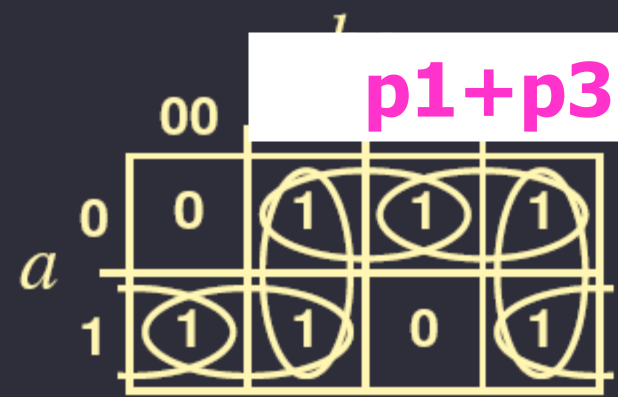


	p1	p2	p3	p4	p5	p6
001	1		1			
011	1	1				
010		1		1		
100					1	1
101			1		1	
110				1		1

Why do we call this problem Unate Covering?

- Let's look at each row of the matrix
- First row can be described with the expression
  - p1+p3
  - We interpret "p1 = 1" as "column p1 is selected"
  - The expression (p1+p3) is "1" when the first row is covered

Relationship between implicants and the Function



p1+p3

	p1	p2	p3	p4	p5	p6
001	1		1			
011	1	1				
010		1		1		
100					1	1
101			1		1	
110				1		1

Relationship between implicants and the Function

- In order to be able to represent the function completely
  - All of the following expressions must be true
    - p1 + p3
    - p1+p2
    - p2 + p4
    - p5+p6
    - p3+p5
    - p4+p6

Relationship between implicants and the Function

- This means
  - Expressing this in logic terms
  - $(p1+p3)(p1+p2)(p2+p4)(p5+p6)(p3+p5)(p4+p6) = 1$

Relationship between implicants and the Function

- This equation is also called the constraint equation ***C*** of this covering problem
- We need to find an assignment of ones and zeros to the variables pi, such that the equation is satisfied
  - while setting a minimum number of pi variables to true (picking the fewest number of implicants to cover the function, that is our cost metric)
  - (p1+p3)(p1+p2)(p2+p4)(p5+p6)(p3+p5)(p4+p6)= 1

Relationship between implicants and the Function

- Note that all variables in this equation appear **uncomplemented**
  - (p1+p3)(p1+p2)(p2+p4)(p5+p6)(p3+p5)(p4+p6)= 1
- A formula where **no variable appears in both** phases is called **unate** All positive/negative
- Because of this form of the constraint equation ***C***, the covering problem for two level logic minimization is also called unate covering

Let's Go back to solving the Unate Covering Problem

- The search for the optimal cover with minimum number of prime implicants (minimum number of columns) ends when
  - The implicant matrix has no rows left

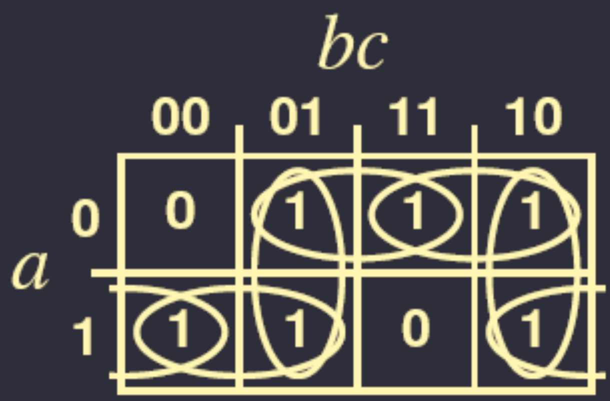
Use Bound to Constrain Search Space

- Pick one implicant (column) and assume it will be included in the final cover
  - Let's pick p1 (p1 = 1) and evaluate the constraint equation with (p1 =1)
  - $C_{p1} = (1+p3)(1+p2)(p2+p4)(p5+p6)(p3+p5)(p4+p6) = 1$
  - $C_{p1} = (p2+p4)(p5+p6)(p3+p5)(p4+p6) = 1$
  - This operation is also called ***cofactoring C*** with respect to p1
    - Kind of taking a derivative in Boolean logic

Use Bound to Constrain Search Space

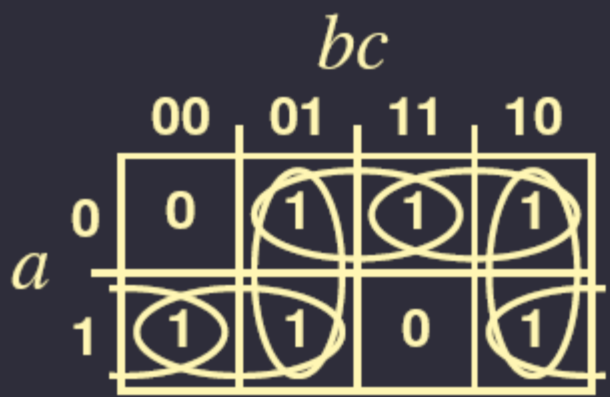
- Let's see the matrix after deciding to include p1 in the cover solution

p1 in the solution



	p1	p2	p3	p4	p5	p6
001	1		1			
011	1	1				
010		1		1		
100					1	1
101			1		1	
110				1		1

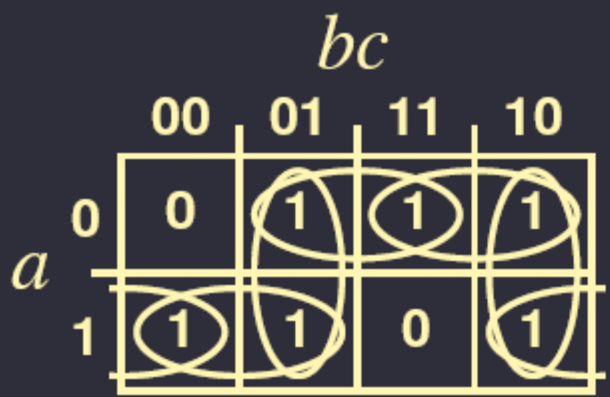
p1 in the solution, matrix reduced



p1 = 1

	p2	p3	p4	p5	p6
010	1		1		
100				1	1
101		1		1	
110			1		1

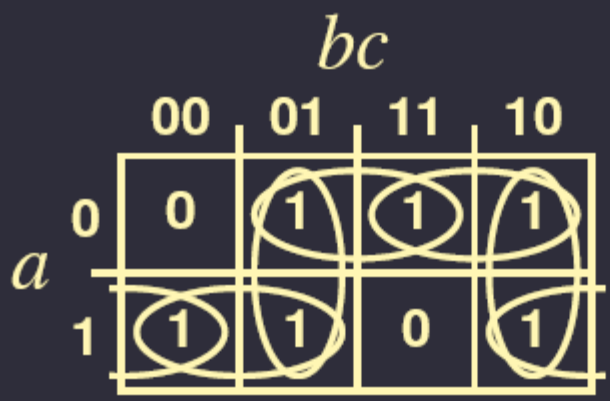
p1 in the solution, matrix reduced



p4 dominates p2

	p2	p3	p4	p5	p6
010	1		1		
100				1	1
101		1		1	
110			1		1

p1 in the solution, matrix reduced

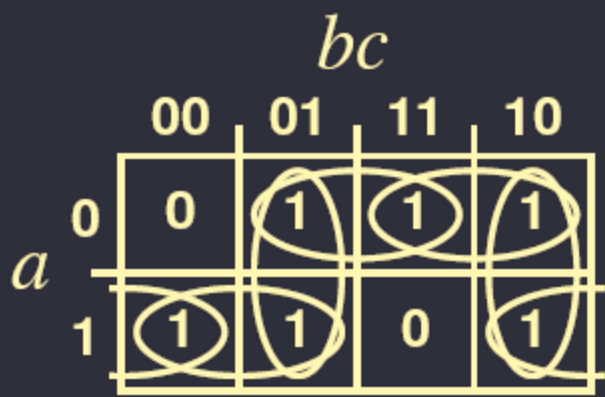


p5 dominates p3

	p2	p3	p4	p5	p6
010	1		1		
100				1	1
101		1		1	
110			1		1

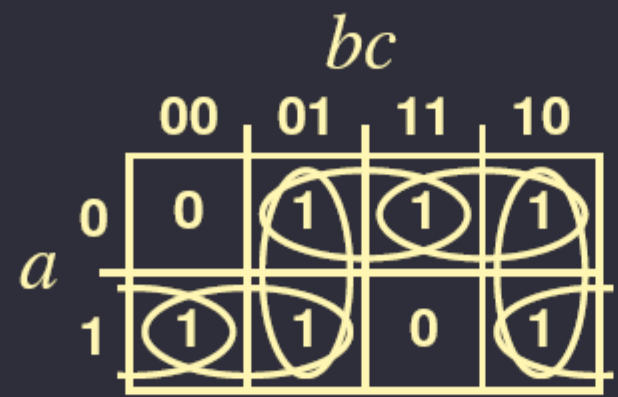
p1 in the solution, matrix reduced

After removing p2 and p3



	p4	p5	p6
010	1		
100		1	1
101		1	
110	1		1

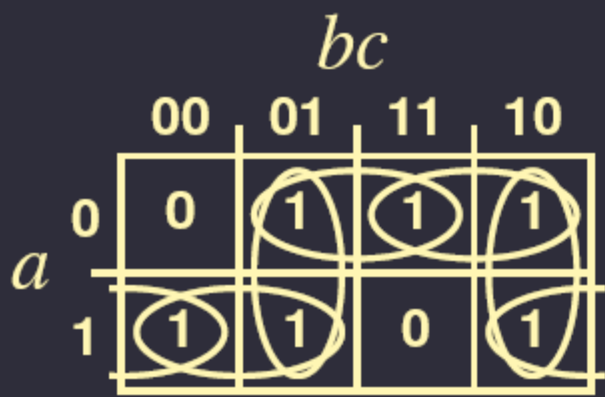
p1 in the solution, matrix reduced



p4 and p5 are essential

	p4	p5	p6
010	1		
100		1	1
101		1	
110	1		1

p1, p4, p5 in the solution, matrix reduced to nothing



Removing p4 and p5
Reduces the table completely

Reaching the bottom of recursion

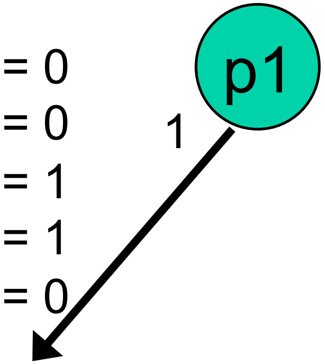
- We retain the dominant columns p4 and p5
  - p1 = 1, p4 = 1, p5 =1
  - What has happened to the constraint equation?
  - $\boldsymbol{C_{p1}}$ = (**1**+p3)(**1**+p2)(p2+p4)(p5+p6)(p3+p5)(p4+p6) = 1
  - $\boldsymbol{(C_{p1})_{p4}}$ = (p2+**1**)(p5+p6)(p3 +p5)(**1** +p6) = 1
  - $\boldsymbol{((C_{p1})_{p4})_{p5}}$ = (**1** +p6) = **1** = 1

How many columns have we picked?

- We picked p1, p4, and p5
  - **3 columns**
- We had first identified the lower bound for the solution as 3 columns
  - 3 independent rows
- We know we cannot do any better
- So, we can stop the search and declare the optimal solution

The decision tree

p3 = 0
p2 = 0
p4 = 1
p5 = 1
p6 = 0



Cost = 3
U = 3
U=L=3
Declare optimal solution found
Finish the B&B process

Summary

- We have learned
  - 1. exact (optimal) technique for two-level minimization
  - 2. an optimization problem called unate covering
    - It has many other applications
  - 3. a widely used optimization technique called Branch and Bound
  - 4. Some concepts (cofactoring) which will reappear in our future discussions