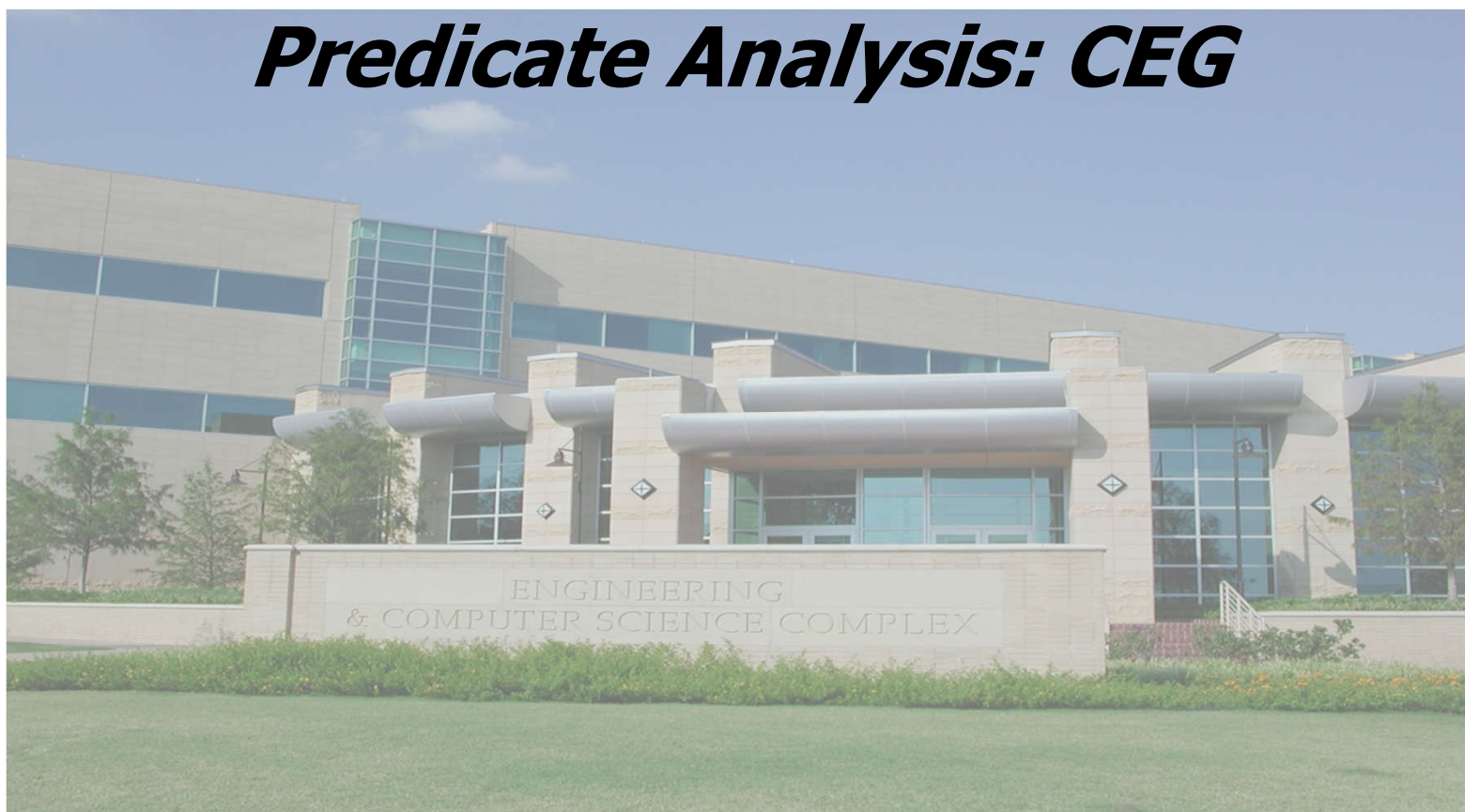


Predicate Analysis: CEG



Dr. Mark C. Paulk


SE 4367 – Software Testing, Verification, Validation, and Quality Assurance

Topics: Software Testing

Part II: Test Generation

3. Domain Partitioning

4. Predicate Analysis

- 
- **Domain Testing**
 - **Cause-Effect Graphing**
 - **Tests Using Predicate Syntax**
 - **Tests Using Basis Paths**
 - **Scenarios and Tests**

5. Test Generation from Finite State Models

6. Test Generation from Combinatorial Designs

Domain Testing

A program contains a domain error if on an input it follows an incorrect path due to incorrect condition or incorrect computation.

- **an input domain partitioning technique**

The requirements for domain testing are extracted from the code.

- **white-box testing**

Domain Errors

A path condition consists of a set of predicates that occur along the path in conditional or a loop statement.

Gives rise to a path domain that has a boundary.

The boundary consists of one or more border segments.

There is one border corresponding to each predicate in the path condition.

An error in any of the predicates in a path condition leads to a border shift.

Border Shifts

A domain error occurs because of an unintended shift in the border.

Sample points inside and outside the displaced area to try to detect the error.

Closed border for operators \leq , \geq , and $=$

- points that lie on the closed border are part of the path domain

Open border for operators $>$, $<$, and \neq

- points that lie on the open border are not part of the path domain

ON-OFF Points

ON point must satisfy the path condition associated with the border

- need not be on the border

OFF point must be as close as possible to the ON point for that border

Only one ON and one OFF point are needed for inequalities

One ON and two OFF points are needed for = and \neq constraints

Generation of Border Shift Test Cases

Generate ON and OFF points for each predicate in the path condition

- **avoid reuse of points to increase chance of finding errors**

Determine the expected (correct) value of the program for each point

Run the program, comparing the expected output to the actual output for the test case

Mathur, Example 4.2

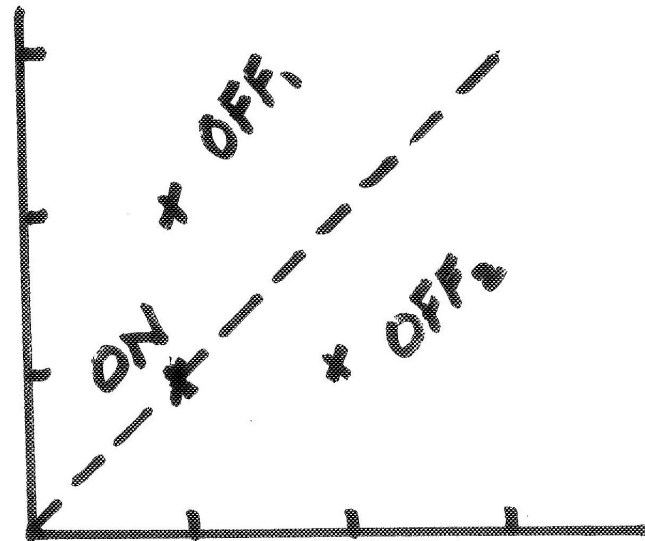
Predicate $x==y$ defines a border

Need one ON and two OFF points

ON: $(x=1, y=1)$

OFF1: $(x=1, y=2)$

OFF2: $(x=2, y=1)$



OFF points are on opposite sides of the border

Suppose the correct border is defined by $x==y+1$

Then test cases ON and OFF2 lead to different outcomes

	<u>ON</u>	<u>OFF1</u>	<u>OFF2</u>
$x==y$	t 1==1	f 1==2	f 2==1
$x==y+1$	f 1==1+1	f 1==2+1	t 2==1+1

Topics: Software Testing

Part II: Test Generation

3. Domain Partitioning

4. Predicate Analysis

- Domain Testing
- • Cause-Effect Graphing
- Tests Using Predicate Syntax
- Tests Using Basis Paths
- Scenarios and Tests

5. Test Generation from Finite State Models

6. Test Generation from Combinatorial Designs

Cause-Effect Graphing

aka dependency modeling

Models dependency relationships among program input conditions (causes) and output conditions (effects)

Expressed as Boolean expression

Use heuristics to avoid combinatorial explosion

Generic Cause-Effect Procedure

Identify causes and effects by reading the requirements.

- **Assign each cause and effect a unique identifier.**
- **Note that an effect can also be a cause for another effect.**

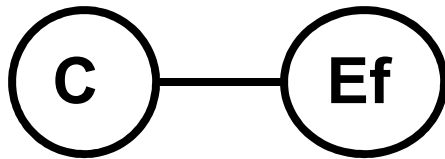
Express the relationship between causes and effects using a cause-effect graph.

Transform the cause-effect graph into a limited entry decision table.

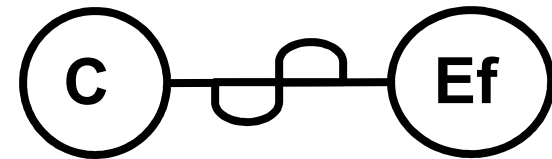
Generate tests from the decision table.

Cause-Effect Notation

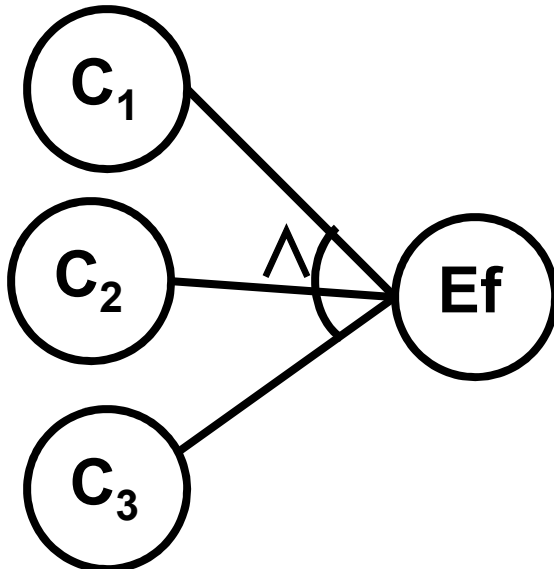
C implies Ef



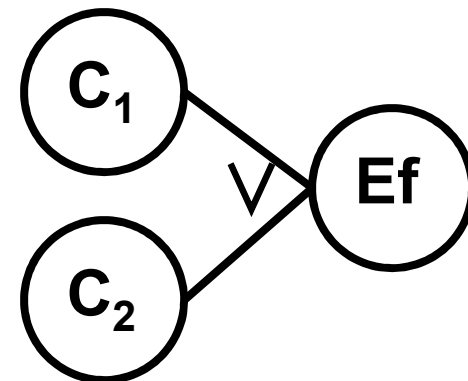
not C implies Ef



Ef when C_1 and C_2 and C_3

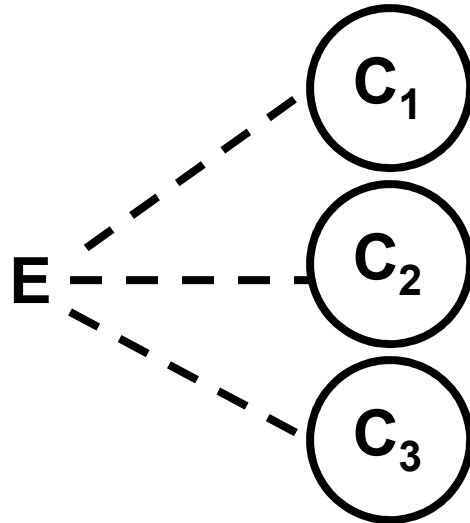


Ef when C_1 or C_2

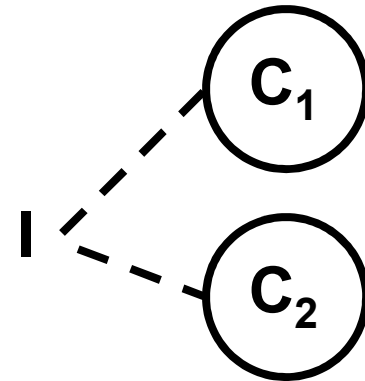


Constraints Among Causes

Exclusive: either C_1 or C_2 or C_3

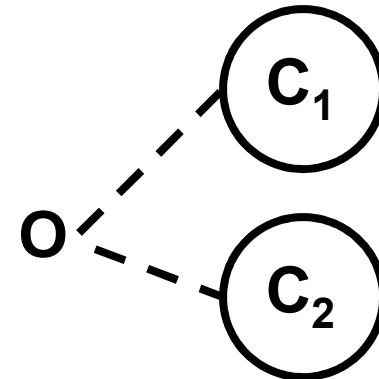
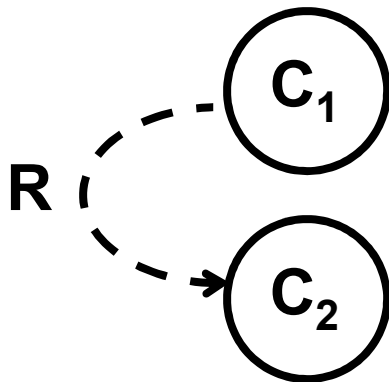


Inclusive: at least C_1 or C_2



One and only one of C_1 and C_2

C_1 requires C_2



E, I, R, O Possible Values

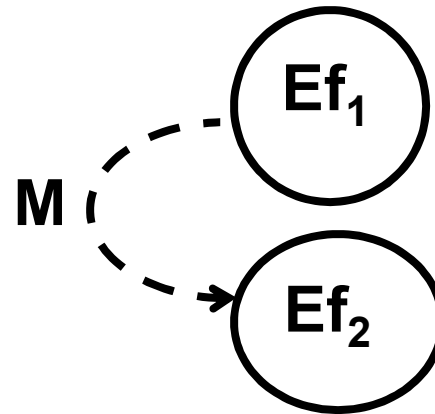
<u>Constraint</u>	Possible values		
	<u>C1</u>	<u>C2</u>	<u>C3</u>
E (C1,C2,C3)	0	0	0
	1	0	0
	0	1	0
	0	0	1
I (C1,C2)	1	0	--
	0	1	--
	1	1	--
R (C1,C2)	1	1	--
	0	0	--
	0	1	--
O (C1,C2,C3)	1	0	0
	0	1	0
	0	0	1

Masking

One constraint on effects

Both effects cannot occur for the same state of the program

Ef_1 masks Ef_2



Creating Cause-Effect Graphs

Identify causes and effects from the requirements.

Identify constraints.

Assign a unique identifier to each cause and effect.

Incrementally construct the cause-effect graph.

Mathur, Example 4.5

Buying a computer system

Possible configurations

- **processor: CPU1, CPU2, CPU3**
- **printer: PR1, PR2**
- **monitor: M20, M23, M30**
- **memory: RAM256, RAM512, RAM1G**

Example 4.5

Business Rules

M20, M23 buy with any CPU or standalone

M30 buy only with CPU3

- **cannot buy M20, M23 with CPU3**

PR1 available free with CPU2, CPU3

PR1, PR2 buy standalone

CPU1 includes RAM256

CPU2, CPU3 includes RAM512

CPU3+M30 includes PR2, RAM1G

Total price, free window updated with selections

Example 4.5
Business Rules Example

<u>Items selected</u>	<u>“Free” window</u>	<u>Price</u>
CPU1	RAM256	\$499
CPU1, PR1	RAM256	\$628
CPU2, PR2, M23	PR1, RAM512	\$2257
CPU3, M30	PR2, RAM1G	\$3548

Example 4.5
Business Rules Better Example

<u>Items selected</u>	<u>“Free” window</u>	<u>Price</u>
CPU1	RAM256	\$499
CPU2	PR1, RAM512	\$xxxx
CPU3	PR1, RAM512	\$xxxx
CPU3, M30	PR2, RAM1G	\$3548
Other	No giveaway	

Example 4.5

Causes and Effects

Causes

C_1 : purchase CPU1
 C_2 : purchase CPU2
 C_3 : purchase CPU3
 C_4 : purchase PR1
 C_5 : purchase PR2
 C_6 : purchase M20
 C_7 : purchase M23
 C_8 : purchase M30

Effects

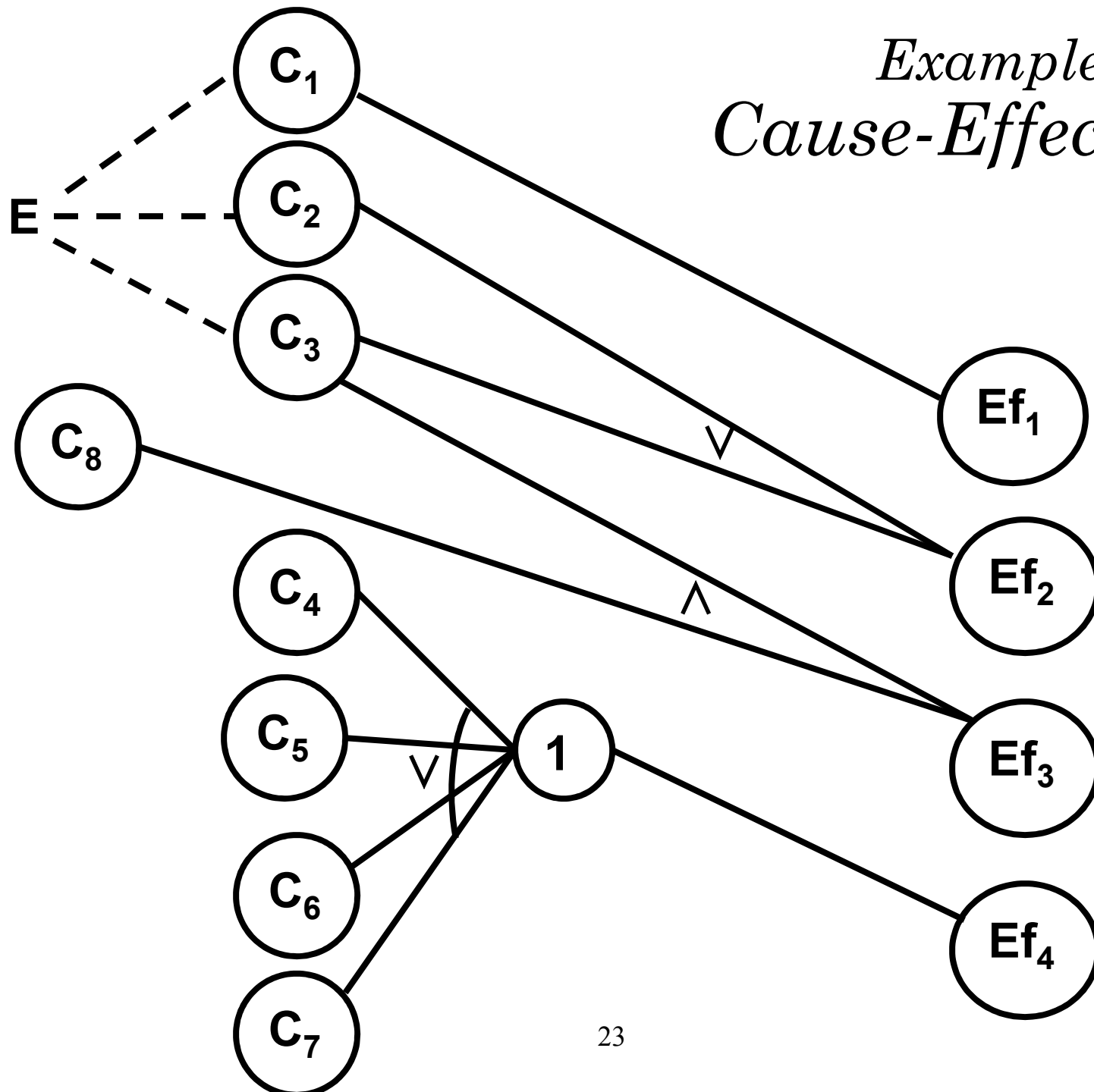
Ef_1 : RAM256
 Ef_2 : RAM512 and PR1
 Ef_3 : RAM1G and PR2
 Ef_4 : no giveaway

Boolean Expressions

$Ef_1 = C_1$
 $Ef_2 = C_2 \vee C_3$
 $Ef_3 = C_3 \wedge C_8$
 $Ef_4 = C_4 \vee C_5 \vee C_6 \vee C_7$

- Ef_4 is different from Mathur
- which is correct?
- is more needed?

Example 4.5
Cause-Effect Graph



Cause-Effect Graph Decision Table (CEGDT)

p causes, q effects

V_k is vector of size $p+q$ with 0/1 entries

- V_j with $1 \leq j \leq p$ indicates state of C_j**
- V_l with $p < l \leq p+q$ indicates state of Ef_l**

1) Initialize DT to empty, DT is $N=p+q$ by M

2) For $i=1, q$ for effects

2.1) Select the next effect to process, $e=Ef_i$

2.2) Find combinations of conditions that cause e
• use heuristics to prevent combinatorial explosion

Let V_1, V_2, \dots be the combinations of causes that lead to e

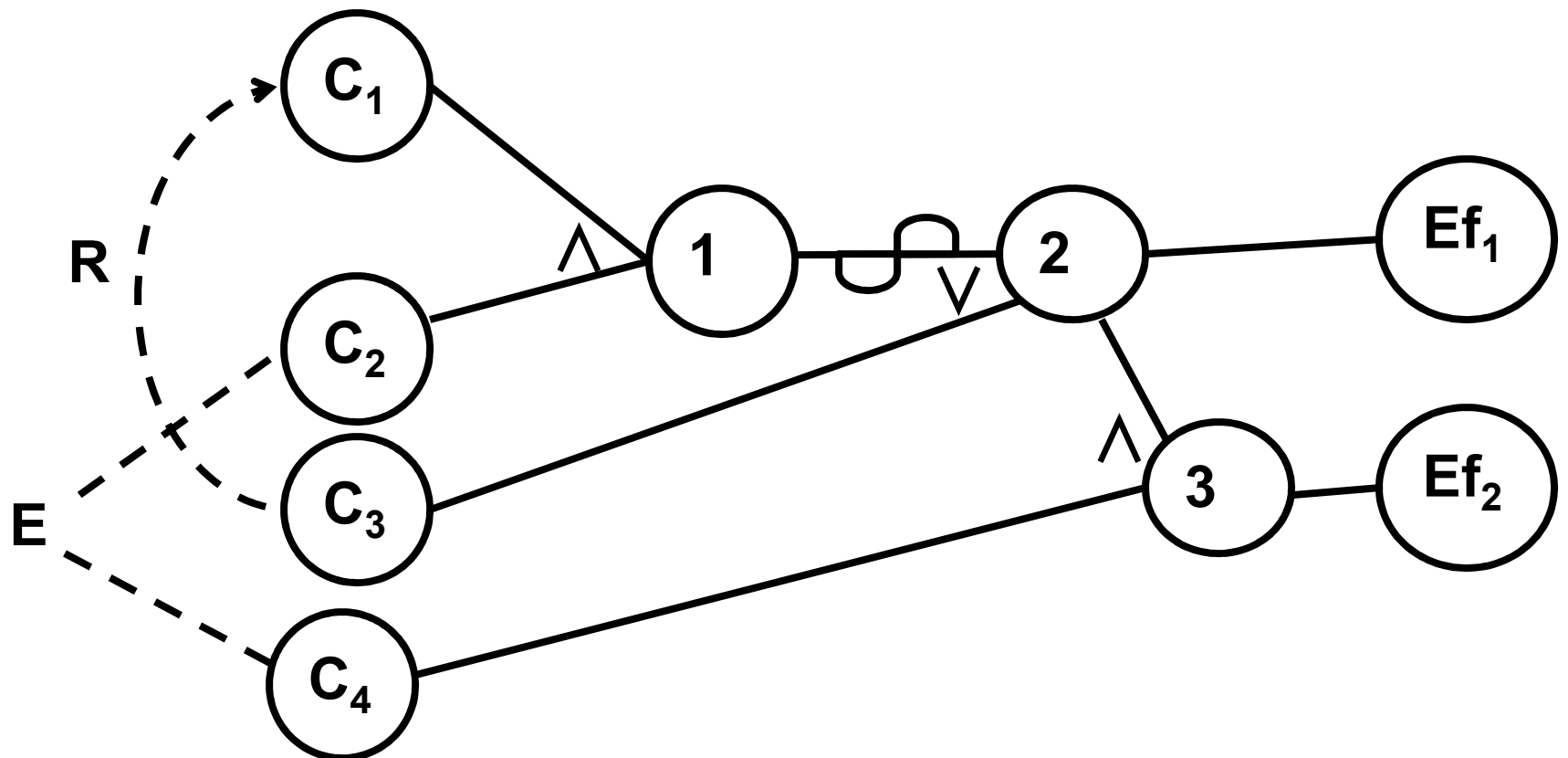
Set $V_k(l)$, $p < l \leq p+q$ to 0 or 1 depending on whether EF_{l-p} is present for all conditions in V_k

2.3) Update DT with V_1, V_2, \dots

2.4) Go to the next available column in DT...

Mathur Example 4.6

Without heuristics



Mathur Example 4.6
Boolean Expression

$$Ef_1 = (2) = \text{NOT } (1) \text{ OR } C_3$$

$$= \text{NOT } (C_1 \text{ AND } C_2) \text{ OR } C_3$$

$$Ef_2 = (3) = (2) \text{ AND } C_4$$

$$= (\text{NOT } (C_1 \text{ AND } C_2) \text{ OR } C_3) \text{ AND } C_4$$

Excluding “R” and “E”

Mathur Example 4.6

Ef_1

$Ef_1 \leftarrow (2)$

$(2) \leftarrow (1) \text{ or } C_3$

$(1) \leftarrow \sim(C_1 \text{ and } C_2)$

C_3 requires C_1 therefore
 $(0,1,1)$ and $(0,0,1)$ are not
feasible.

- Note that $R(C_3, C_1)$ does not
come in automatically with
“or C_3 ”

C_1	C_2	C_3	Ef_1
0	0	0	1
0	0	1	1→0
0	1	0	1
0	1	1	1→0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Set C_4 to 0 and Ef_1 to 1 and
 Ef_2 to 0 gives

	C_1	C_2	C_3	C_4	Ef_1	Ef_2
$V_1 =$	1	0	1	0	1	0
$V_2 =$	1	1	1	0	1	0
$V_3 =$	1	0	0	0	1	0
$V_4 =$	0	1	0	0	1	0
$V_5 =$	0	0	0	0	1	0

Mathur Example 4.6
Add Ef_1 to the Decision Table

Transpose the vectors and add to DT

	V_1	V_2	V_3	V_4	V_5
C_1	1	1	1	0	0
C_2	0	1	0	1	0
C_3	1	1	0	0	0
C_4	0	0	0	0	0
Ef_1	1	1	1	1	1
Ef_2	0	0	0	0	0

Mathur Example 4.6

Ef_2

$Ef_2 \leftarrow (3)$

$(3) \leftarrow (2)$ and C_4

We already calculated (2) for Ef_1

C_1	C_2	C_3	C_4	Ef_2
0	0	0	1	1
0	1	0	1	1 \rightarrow 0
1	0	0	1	1
1	0	1	1	1
1	1	1	1	1 \rightarrow 0

From the CE graph, C_2 and C_4 are exclusive, therefore discard $(1,1,1,1)$ and $(0,1,0,1)$

giving the vectors with Ef_1 added

	C_1	C_2	C_3	C_4	Ef_1	Ef_2
V_6	1	0	1	1	1	1
V_7	1	0	0	1	1	1
V_8	0	0	0	1	1	1

Mathur Example 4.6
Add Ef_2 to the Decision Table

Transpose the vectors and add to DT

	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8
C_1	1	1	1	0	0	1	1	0
C_2	0	1	0	1	0	0	0	0
C_3	1	1	0	0	0	1	0	0
C_4	0	0	0	0	0	1	1	1
Ef_1	1	1	1	1	1	1	1	1
Ef_2	0	0	0	0	0	1	1	1

OR Heuristics to Avoid Combinatorial Explosion

OR: $e = n_1 \vee n_2$

Desired state of e is 0

- **H_1 : enumerate all combinations of inputs to n_1 and n_2 such that $n_1 = n_2 = 0$**

Desired state of e is 1

- **H_2 : enumerate all combinations of inputs to n_1 and n_2 other than those for which $n_1 = n_2 = 0$**
 - **a test case for (1,1) will not detect (1,0) if short circuit evaluation for Boolean expressions is used**

AND Heuristics to Avoid Combinatorial Explosion

AND: $e = n_1 \wedge n_2$

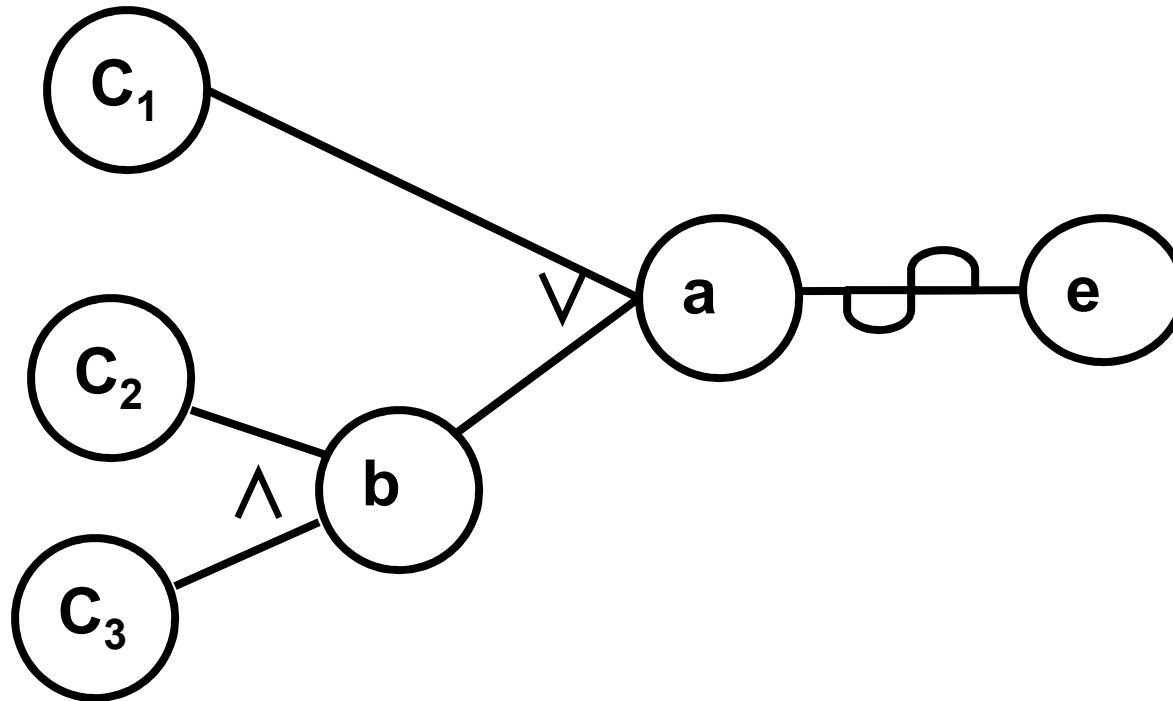
Desired state of e is 0

- **H_3 : enumerate all combinations of inputs to n_1 and n_2 such that each of the possible combinations appears exactly once and $n_1 = n_2 = 1$ does not appear**
 - **for two nodes, there are three: (0,0), (0,1), (1,0)**
 - typo for (0,0) of (1,0) in book
 - **for k nodes ANDed, there are $2^k - 1$ combinations**

Desired state of e is 1

- **H_4 : enumerate all combinations of inputs to n_1 and n_2 such that $n_1 = n_2 = 1$**

Mathur Example 4.7
Cause-Effect Graph



Boolean expression: $\sim (C_1 \vee (C_2 \wedge C_3))$

Mathur Example 4.7

All Combinations

	C_1	C_2	C_3	a	b	e
$(e) \leftarrow \text{not } (a)$	0	0	0	0	0	1
$(a) \leftarrow C_1 \text{ or } (b)$	0	0	1	0	0	1
	0	1	0	0	0	1
$(a) \leftarrow C_1$	0	1	1	1	1	0
$(a) \leftarrow (b)$	1	0	0	1	0	0
$(b) \leftarrow C_2 \text{ and } C_3$	1	0	1	1	0	0
	1	1	0	1	0	0
	1	1	1	1	1	0

Mathur Example 4.7
Deriving All Combinations First

C₁	C₂	C₃	Inputs to node a
0	0	0	C1 = 0, b = 0
0	0	1	
0	1	0	
0	1	1	C1 = 0, b = 1
1	0	0	C1 = 1, b = 0
1	0	1	
1	1	0	
1	1	1	C1 = 1, b = 1

Mathur Example 4.7

Applying OR Heuristics

(a) is an OR node, 0 desired

$(a) \leftarrow C_1 \vee (b)$

H_1 applies: enumerate all combinations of inputs such that $C_1 = b = 0$

C_1	C_2	C_3	Inputs to node a
0	0	0	$C1 = 0, b = 0$
0	0	1	
0	1	0	

Example seems to have a flawed explanation.

Summary – Things to Remember

Cause-effect graphs

- notation (implies, not, and, or)
- constraints (E, I, R, O)
- masking

CEG decision tables

Questions and Answers

