BLACK BOX TESTING (III)

Testing with decision tables

Behavior-Driven Development (BDD) Agile Software Development

Scenario 1: account is in credit

C1:given the account is in credit
C2:and the card is valid
C3:and the dispenser contains cash
C4:when the customer requests cash
A1:then ensure the account is debited
A2:and ensure cash is dispensed
A3:and ensure the card is returned

Short ID

IF (<pre-condition(s)>)
AND (data-condition(s)>)
AND (<input-event-sequence>)
THEN (<action sequence>)
AND (<output event sequence>)
AND(<post-condition(s)>)

Translation to decision tables

Decision Tables: General notes

Decision tables form a compact way to model complicated logic. Decision tables, like if-then-else and switch-case statements, associate **conditions** with **actions** to perform.

Decision tables associate many *independent* conditions with several actions.

Decision Tables – Use (1)

Decision tables make it easier to observe that all possible conditions are accounted for (handling *multiple* inputs).

Decision tables used for:

- Specifying complex program logic
- Generating test cases

Test cases considered in:

- ***structural testing*** when applied to structure (i.e. control flow graph of an implementation).
- ***functional testing*** when applied to a specification.

Decision Tables – Use (2)

Test cases generated with decision tables considered in:

- ***structural testing*** when applied to structure (i.e. control flow graph of an implementation).
- ***functional testing*** when applied to a specification.

Decision tables and control flowcharts

Decision Tables - Structure

Condition entry

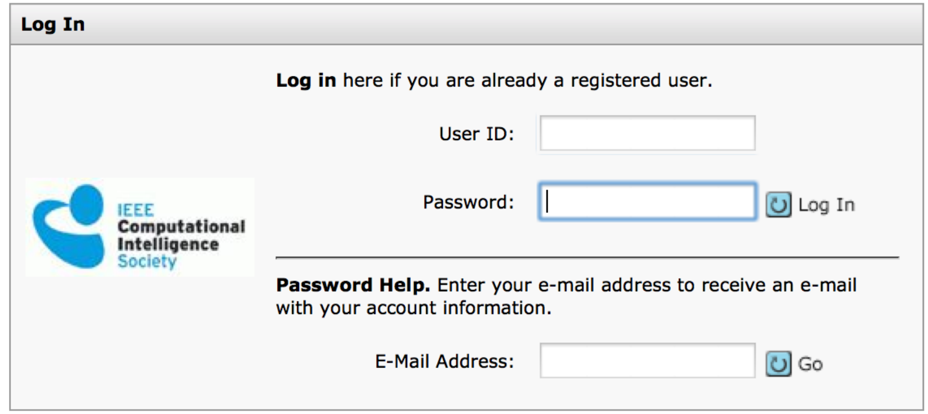
	Rule1	Rule2	Rule3	Rule4
Condition1	Yes	Yes	No	No
Condition2	Yes	X	No	X
Condition3	No	Yes	No	X
Condition4	No	Yes	No	Yes
Action1	Yes	Yes	No	No
Action2	No	No	Yes	No
Action3	No	No	No	Yes

Condition stub

Action stub

Action Entry

Example (1)



Condition stub

Action stub

	Rule1	Rule2	Rule3	Rule4
User ID	F	F	T	T
password	F	T	F	T
Action1	error	error	error	login

Example (2)

	Rule1	Rule2	Rule3…	Rule8
User registered	F	F	…	T
No outstanding fees	F	F	…	T
Under borrow limit	F	T	…	T
Borrow book	No	No	..	Yes

Decision Tables - Structure

Stub	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6
c1	T	T	T	F	-	T
c2	F	T	T	T	-	T
c3	T	T	-	T	T	T
c4	T	F	F	T	T	T
a1	X	X		X	X	X
a2		X				
a3	X			X		
a4			X			X

Decision Tables - Structure

Conditions - *(Condition stub)*	Condition Alternatives – *(Condition Entry)*
Actions – *(Action Stub)*	Action Entries

condition corresponds to a variable, relation or predicate

possible values for conditions are listed among the condition alternatives

- Boolean values (True / False) – Limited Entry Decision Tables
- Several values – Extended Entry Decision Tables
- Don't care value

action is a procedure or operation to perform

entries specify whether the action is to be performed

Default rules

default rules specified to indicate the action to be taken when none of the other rules apply.

When using decision tables in testing, default rules and their associated predicates must be explicitly provided.

	Rule5	Rule6	Rule7	Rule8
Condition1	X	No	Yes	Yes
Condition2	X	Yes	X	No
Condition3	Yes	X	No	No
Condition4	No	No	Yes	X
Default action	Yes	Yes	Yes	Yes

Example-3

Conditions	Printer does not print	Y	Y	Y	Y	N	N	N	N
	A red light is flashing	Y	Y	N	N	Y	Y	N	N
	Printer is unrecognized	Y	N	Y	N	Y	N	Y	N
Actions	Heck the power cable			X					
	Check the printer-computer cable	X		X					
	Ensure printer software is installed	X		X		X		X	
	Check/replace ink	X	X			X	X		
	Check for paper jam		X		X				

Printer Troubleshooting

Example-4

	Conditions/ Courses of Action	Rules					
		1	2	3	4	5	6
Condition Stubs	Employee type	S	H	S	H	S	H
	Hours worked	<40	<40	40	40	>40	>40
Action Stubs	Pay base salary	X		X		X	
	Calculate hourly wage		X		X		X
	Calculate overtime						X
	Produce Absence Report		X				

Example-5

The word-processor may present portions of text in three different formats:

plain text (p),
boldface (b),
italics (i).

The following commands may be applied to each portion of text:

make text plain (P),
make boldface (B),
make italics (I),
emphasize (E),
super emphasize (SE).

Commands are available to dynamically set E to mean either B or I (denote such commands as E=B and E=I, respectively.)

Similarly, SE can be dynamically set to mean either B (command SE=B) or I (command SE=I), or B and I (command SE=B+I.)

P	*								
B		*							*
I			*						*
E				*	*				
SE						*	*	*	
E = B				*					
E = I					*				
SE = B						*			
SE = I							*		
SE = B + I								*	
action	**p**	**b**	**i**	**b**	**i**	**b**	**i**	**b,i**	**b,i**

Example - 7

Age of driver below 16, 16-18, 18 or more
The driver successfully passed the theoretical exam (or not)
Whether an authorized driver is present or absent
Whether the individual successfully passed the practical driving exam

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Age A:<16 16<B<18 C: 18	A	A	A	A	A	A	A	A	B	B	B	B	B	B	B	B	C	C	C	C	C	C	C	C
Theory OK?	O	O	O	O	N	N	N	N	O	O	O	O	N	N	N	N	O	O	O	O	N	N	N	N
Accompanying driver present?	O	O	N	N	O	O	N	N	O	O	N	N	O	O	N	N	O	O	N	N	O	O	N	N
License obtained?	O	N	O	N	O	N	O	N	O	N	O	N	O	N	O	N	O	N	O	N	O	N	O	N
Driving authorized?	N (a)	N (a)	N (a)	N (a)	N (a)	N (a)	N (a)	N (a)	N (b)	N	N (b)	N	N (b)	N (d)	N (b)	N (d)	O	O	O	N	N (c)	N (d)	N (c)	N (d)

O- yes N- no

Design of Decision Tables

Determine conditions and values

Determine maximum number of rules

Determine actions

List possible rules

List appropriate actions for each rule

Simplify the rules (reduce, if possible, the number of columns)

Decision Tables – Design Issues

completeness of rules

- *every* combination of predicate truth values <u>plus</u> default cases are explicit in the decision table

consistency of rules

- every combination of predicate truth values results in only one action or set of actions

Redundancy and consistency

Conditions	1 to 4	5	6	7	8	9
c1	T	F	F	F	F	T
c2	-	T	T	F	F	F
c3	-	T	F	T	F	F
a1	X	X	X	-	-	X
a2	-	X	X	X	-	-
a3	X	-	X	X	X	X

redundancy

Conditions	1 to 4	5	6	7	8	9
c1	T	F	F	F	F	T
c2	-	T	T	F	F	F
c3	-	T	F	T	F	F
a1	X	X	X	-	-	-
a2	-	X	X	X	-	X
a3	X	-	X	X	X	-

inconsistency

Example: driver's license (1)

Age of driver below 16, 16-8, 18 or more
The driver successfully passed the theoretical exam (or not)
Whether an authorized driver is present or absent
Whether the individual successfully passed the practical driving exam

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Age A:<16 16<B<18 C: 18	A	A	A	A	A	A	A	A	B	B	B	B	B	B	B	B	C	C	C	C	C	C	C	C
Theory OK?	O	O	O	O	N	N	N	N	O	O	O	O	N	N	N	N	O	O	O	O	N	N	N	N
Accompanying driver present?	O	O	N	N	O	O	N	N	O	O	N	N	O	O	N	N	O	O	N	N	O	O	N	N
License obtained?	O	N	O	N	O	N	O	N	O	N	O	N	O	N	O	N	O	N	O	N	O	N	O	N
Driving authorized?	N (a)	N (a)	N (a)	N (a)	N (a)	N (a)	N (a)	N (a)	N (b)	N	N (b)	N	N (b)	N (d)	N (b)	N (d)	O	O	O	N	N (c)	N (d)	N (c)	N (d)

O- yes N- no

Example: driver's license (2)- reduction of decision table

Cases (Constraints):

Driving is not allowed due to non – fulfillment of of the prerequisite for the theoretical exam and driving with an authorized driver or not

Cases when driving accompanied or not is possible

Theoretically impossible cases to applicable legal prerequisites

Example: driver's license

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Age A:<16 16<B<18 C: 18	A	A	A	A	A	A	A	A	B	B	B	B	B	B	B	B	C	C	C	C	C	C	C	C
Theory OK?	O	O	O	O	N	N	N	N	O	O	O	O	N	N	N	N	O	O	O	O	N	N	N	N
Accompanying driver present?	O	O	N	N	O	O	N	N	O	O	N	N	O	O	N	N	O	O	N	N	O	O	N	N
License obtained?	O	N	O	N	O	N	O	N	O	N	O	N	O	N	O	N	O	N	O	N	O	N	O	N
Driving authorized?	N (a)	N (a)	N (a)	N (a)	N (a)	N (a)	N (a)	N (a)	N (b)	N	N (b)	N	N (b)	N (d)	N (b)	N (d)	O	O	O	N	N (c)	N (d)	N (c)	N (d)

(a)Age below 16

(b)Driving license obtained and age below 18

(c) Theoretical exam has not been passed and practical exam passed

(d) theoretical and practical exam have not been passed

O- yes N- no

Example: driver's license (3)- reduction of decision table

(a)Age below 16

(b)Driving license obtained and age below 18

(c) Theoretical exam has not been passed and practical exam passed

(d) theoretical and practical exam have not been passed

	01	09	10	12	14	17	18	19	20	21
Age A: <16 $16<B<18$ C: 18	A	B	B	B	–	C	C	C	C	C
Theory OK?	–	–	Y	Y	N	Y	Y	Y	Y	N
Accompanying driver present?	–	–	Y	N	–	Y	Y	N	N	–
License obtained?	–	Y	N	N	N	Y	N	Y	N	Y
Driving authorized?	N (a)	N (b)	N	N	N (d)	Y	Y	Y	N	N (c)
Weight	8	4	1	1	4	1	1	1	1	2

Test Case Design

- Once the specification has been verified, the objective is to demonstrate that the implementation provides the correct action for all combinations of predicate values:
  - if there are *k* rules over *n* binary predicates, then there are at least *k* cases and at most 2^n cases to consider.
- Test design based on unexpanded rules or on the expanded rules with 2^n tests
  - find the input vector to realize each case.

Test Case Design

- To identify test cases with decision tables, we interpret conditions as inputs, and actions as outputs.
- Sometimes conditions end up referring to equivalence classes of inputs, and actions refer to major functional processing portions of the item being tested.
- The rules are then interpreted as test cases.

Decision Table for the Triangle Problem

Conditions											
C1: a < b+c?	F	T	T	T	T	T	T	T	T	T	T
C2: b < a+c?	-	F	T	T	T	T	T	T	T	T	T
C3: c < a+b?	-	-	F	T	T	T	T	T	T	T	T
C4: a=b?	-	-	-	T	T	T	T	F	F	F	F
C5: a=c?	-	-	-	T	T	F	F	T	T	F	F
C6: b=c?	-	-	-	T	F	T	F	T	F	T	F
Actions											
A1: Not a Triangle	X	X	X								
A2: Scalene											X
A3: Isosceles							X		X	X	
A4: Equilateral				X							
A5: Impossible					X	X		X			

Test Cases for the Triangle Problem

Case ID	a	b	c	Expected Output
DT1	4	1	2	Not a Triangle
DT2	1	4	2	Not a Triangle
DT3	1	2	4	Not a Triangle
DT4	5	5	5	Equilateral
DT5	?	?	?	Impossible
DT6	?	?	?	Impossible
DT7	2	2	3	Isosceles
DT8	?	?	?	Impossible
DT9	2	3	2	Isosceles
DT10	3	2	2	Isosceles
DT11	3	4	5	Scalene

Usage of decision tables

- The use of the decision-table model is applicable when :
  - the specification is given or can be converted to a decision table .
  - the order in which the predicates are evaluated does not affect the interpretation of the rules or resulting action.
  - the order of rule evaluation has no effect on resulting action .
  - once a rule is satisfied and the action selected, no other rule need be examined.
  - the order of executing actions in a satisfied rule is of no consequence.
- The restrictions do not, in reality, eliminate many potential applications.
  - In most applications, the order in which the predicates are evaluated is immaterial.
  - Some specific ordering may be more efficient than some other but in general the ordering is not inherent to the program's logic.

Development Guidelines

- Decision table testing is most appropriate for programs with
  - a lot of decision making
  - important logical relationships among input variables
  - calculations involving subsets of input variables
  - cause and effect relationships between input and output
  - complex computation logic (*high* cyclomatic complexity)
- Decision tables *do not scale up* very well
- Decision tables can be *iteratively* refined

Testing with the use of cause-effect graphs

Cause-Effect Graphs

•Formal representation using which we convert specifications into logical relationships between inputs and outputs (inputs and transformations)

•undirected Boolean graph showing dependencies (relationships) between causes and effects:
inputs – causes, outputs – effects

•Logic operators: *and, or, not*

•Constraints among inputs
•Dependencies between outputs

Cause-Effect Graphs: the development process

Specification divided into workable pieces

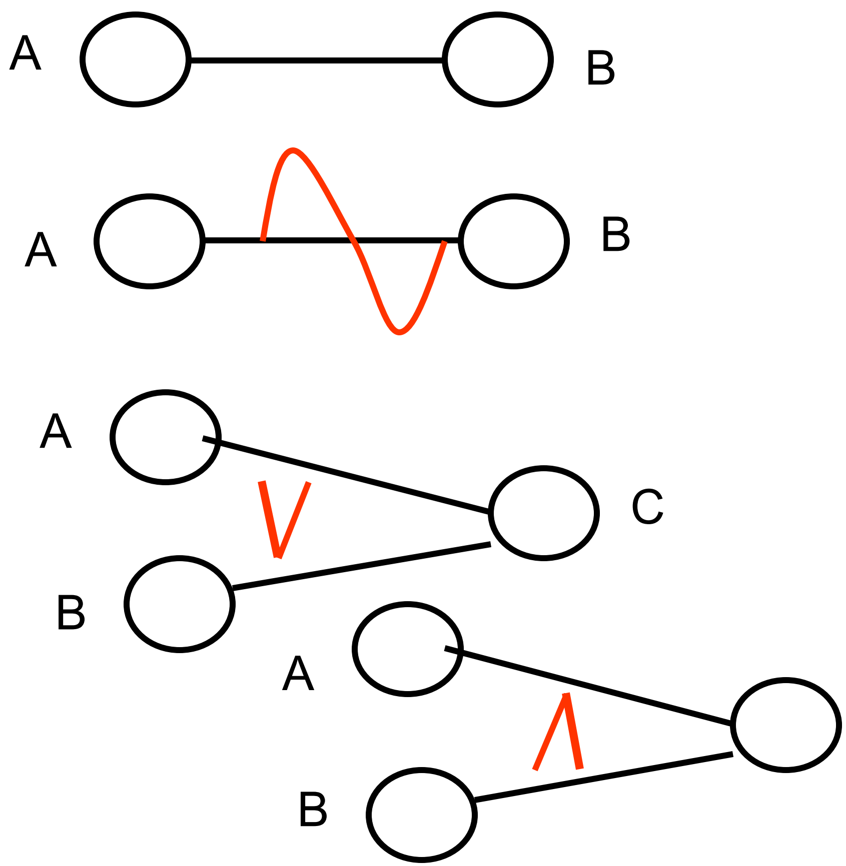
Causes and effects in the specification document are identified

Semantic content of the specification is analyzed and transformed into a Boolean graph linking causes and effects (cause-effect graph)

The graph is annotated by constraints describing combinations of causes and/or effects that are impossible because of existing constraints

Through tracing, the graph is converted into a limited – entry decision table; each column of this table represents a test case

Cause-Effect Graphs: Basic graphic symbols



Identity
If A then B

negation
If (not A) then B

OR
If (A or B) then C

AND
If (A and B) then C

Cause-Effect Graphs: Example

Specifications:

The character in column 1 must be an “A” or a “B”. The character in column 2 must be a digit. In this case the file is updated
If the first character is incorrect, message ERROR_1 is issued
If the second character is not a digit, message ERROR_2 is generated

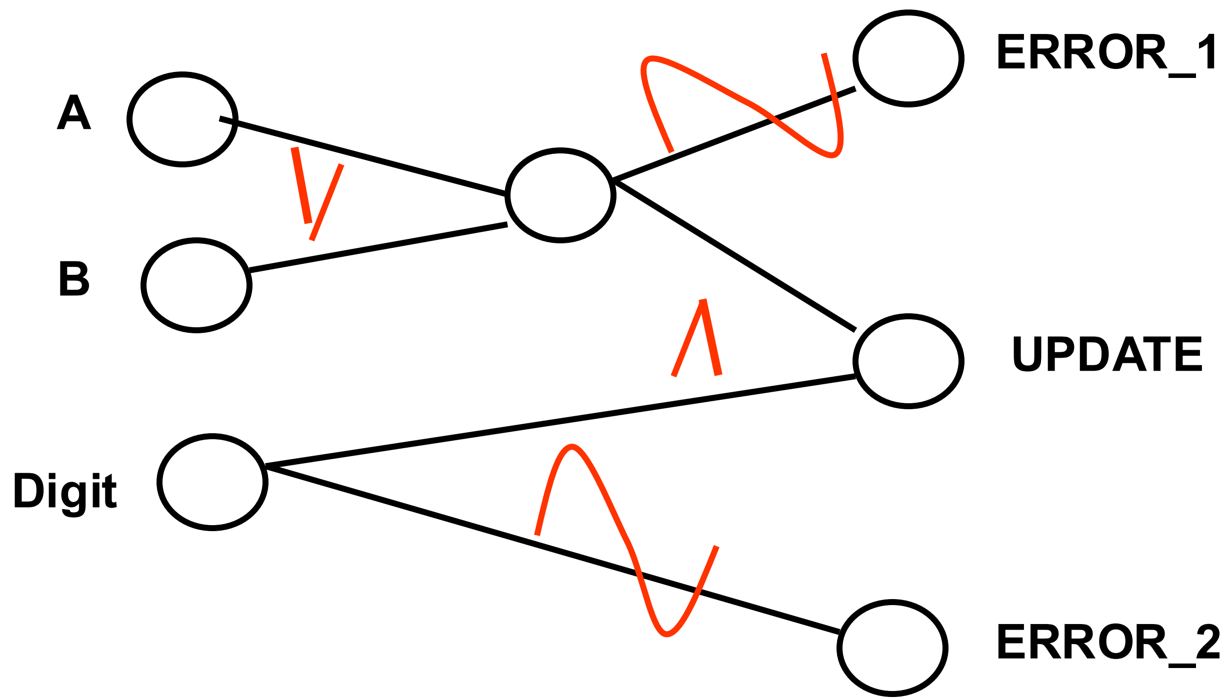
Causes:

1 – character in column 1 is A
2 – character in column 1 is B
3 – character in column 2 is a digit

Effects:

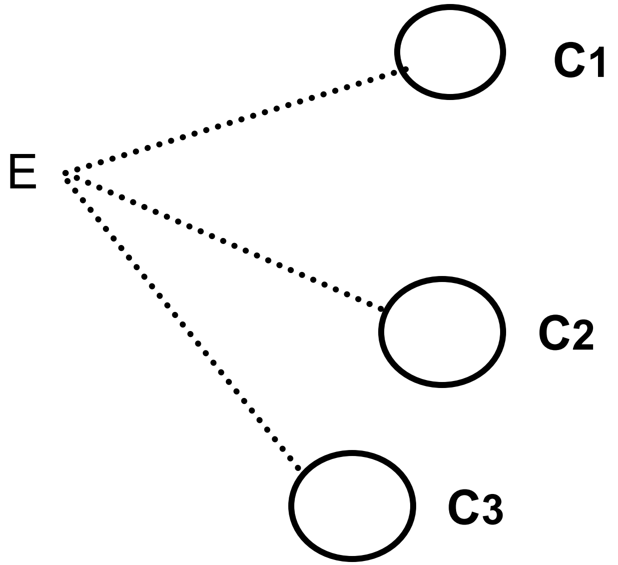
10 – UPDATE
11 – ERROR_1 issued
12 – ERROR_2 issued

Cause-Effect Graphs: An example



Cause-Effect Graphs: Constraints among causes

Exclusive



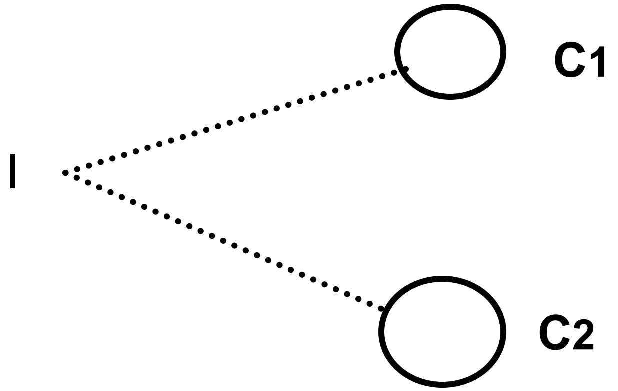
<u>Either C1 or C2 or C3</u>

arity>1

C1	C2	C3
0	0	0
1	0	0
0	1	0
0	0	1

Cause-Effect Graphs: Constraints among causes

Inclusive



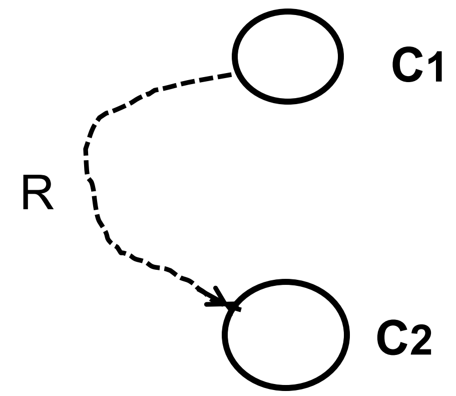
<u>At least C1 or C2 must be present</u>

arity>1

C1	C2
1	0
0	1
1	1

Cause-Effect Graphs: Constraints among causes

requires



<u>R(C_1, C_2) C_1 requires C_2</u>

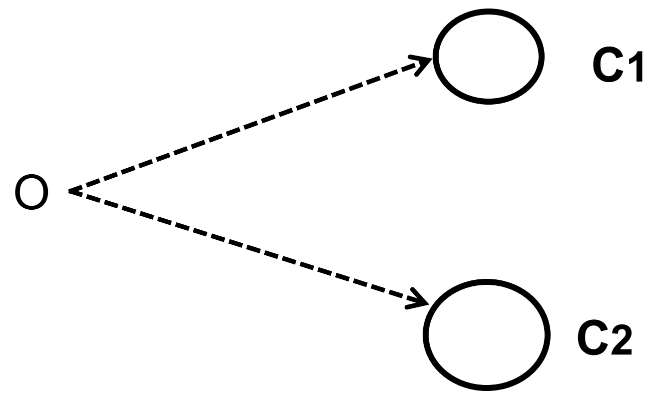
applies to two causes

C_1	C_2
1	1
0	0
0	1

No combination where C_1=1, C_2 =0

Cause-Effect Graphs: Constraints among causes

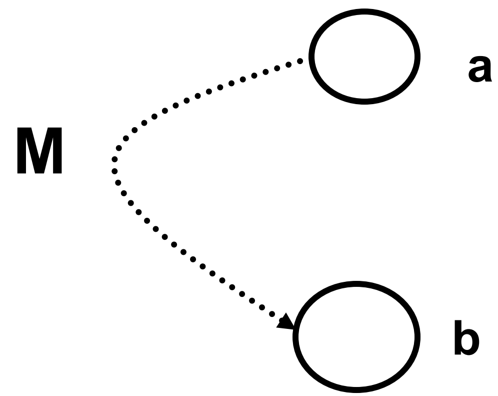
one and only one



O(C_1, C_2) one and only one of C_1 and C_2

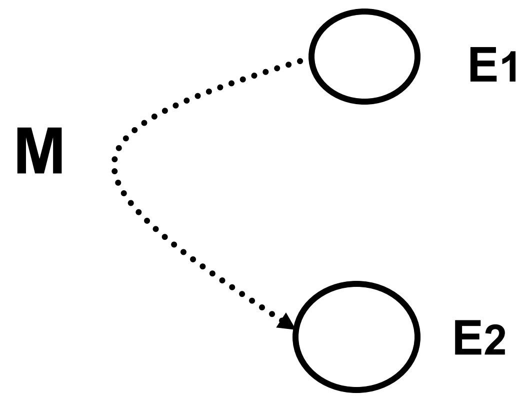
C1	C2
1	0
0	1

Cause-Effect Graphs: Constraints among effects



If effect "a" is 1, effect "b" is forced to 0
Effect "a" <u>masks</u> effect of "b"

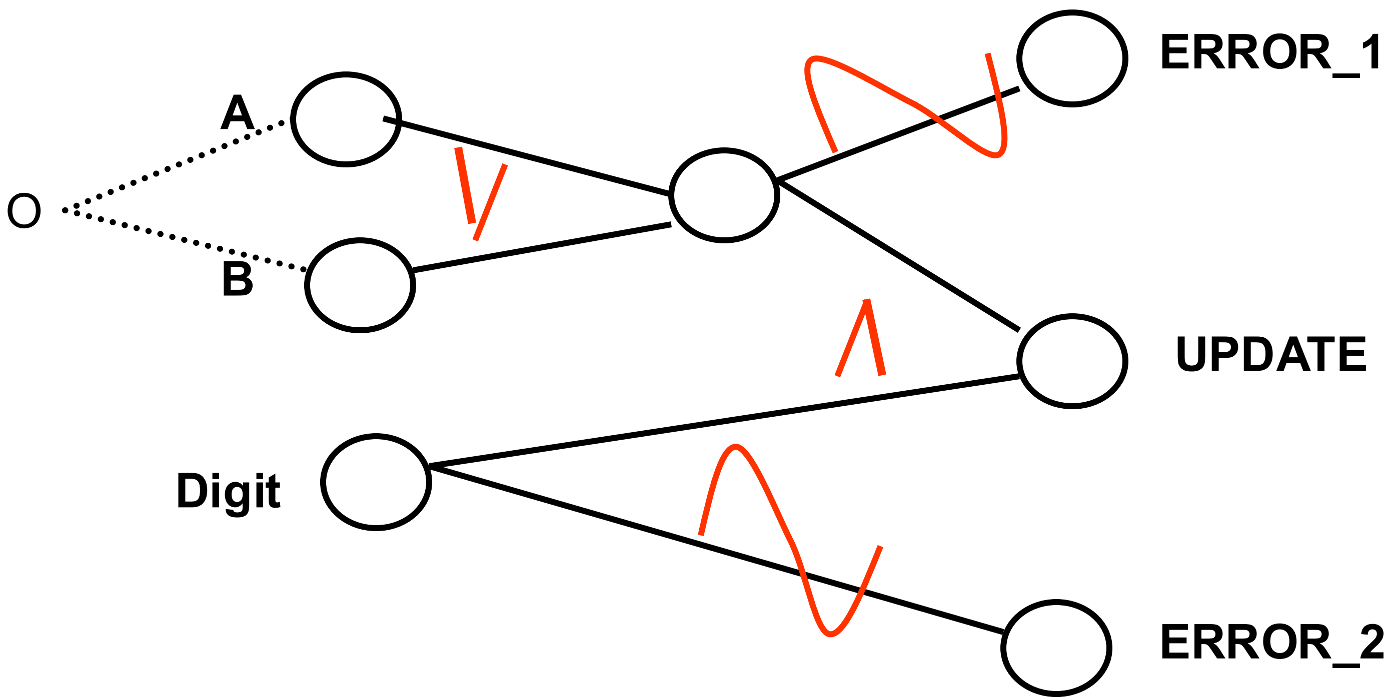
Cause-Effect Graphs: Constraints among effects



E1: generate “shipping invoice”
E2: generate “order not shipped – regret letter”

E2 is masked by E1

Cause-Effect Graphs: An example



Cause-Effect Graphs: An example

A non-resident will pay 1% of the gross pay in city tax

Residents pay on the following scale:
-if gross pay is no more than $30,00, the tax is 1%
- If gross pay is more than $30,000, but no more than $50,000, the tax is 5%
- If gross pay is more than $50,000, the tax is 15%

Causes:
(1) non-resident
(2) Resident
(3) 0<= Gross Pay<= 30k
(4) $30k< Gross Pay <=50k
(5) Gross Pay >50k

Effects:
(11) 1% tax (12) 5% tax (13) 15% tax

Example

Water level monitoring system reporting to an agency involved with flood control

INPUT: the syntax of the function is LEVEL (A, B)
A: the height of (in meters) of the water behind the dam
B: the number of centimeters of rain in the last 24 hours

PROCESSING: The function calculates whether the water level is within a safe range or too high

OUTPUT: The screen shows one of the messages
Level = safe (result is safe)
Level = high (result is high)
Invalid Syntax

Example

Causes

1. The first five characters of the command LEVEL
2. The command contains exactly two parameters separated by a comma and enclosed in parentheses
3. The parameters A and B are real numbers such that the water level is calculated to be SAFE
4. The parameters A and B are real numbers such that the water level is calculated to be HIGH

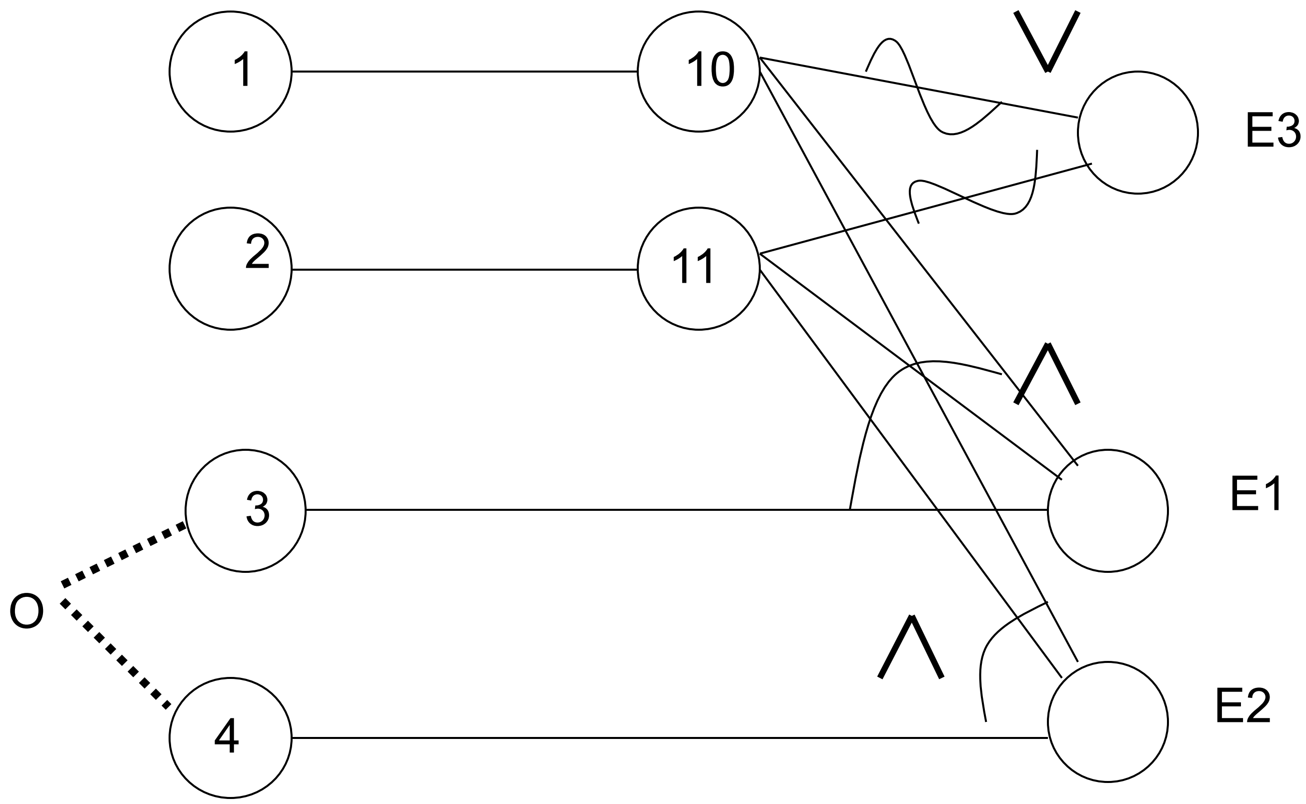
Effects

E1 The message “LEVEL = SAFE” is displayed
E2 The message “LEVEL = HIGH” is displayed
E3 The message “INVALID SYNTAX” is displayed

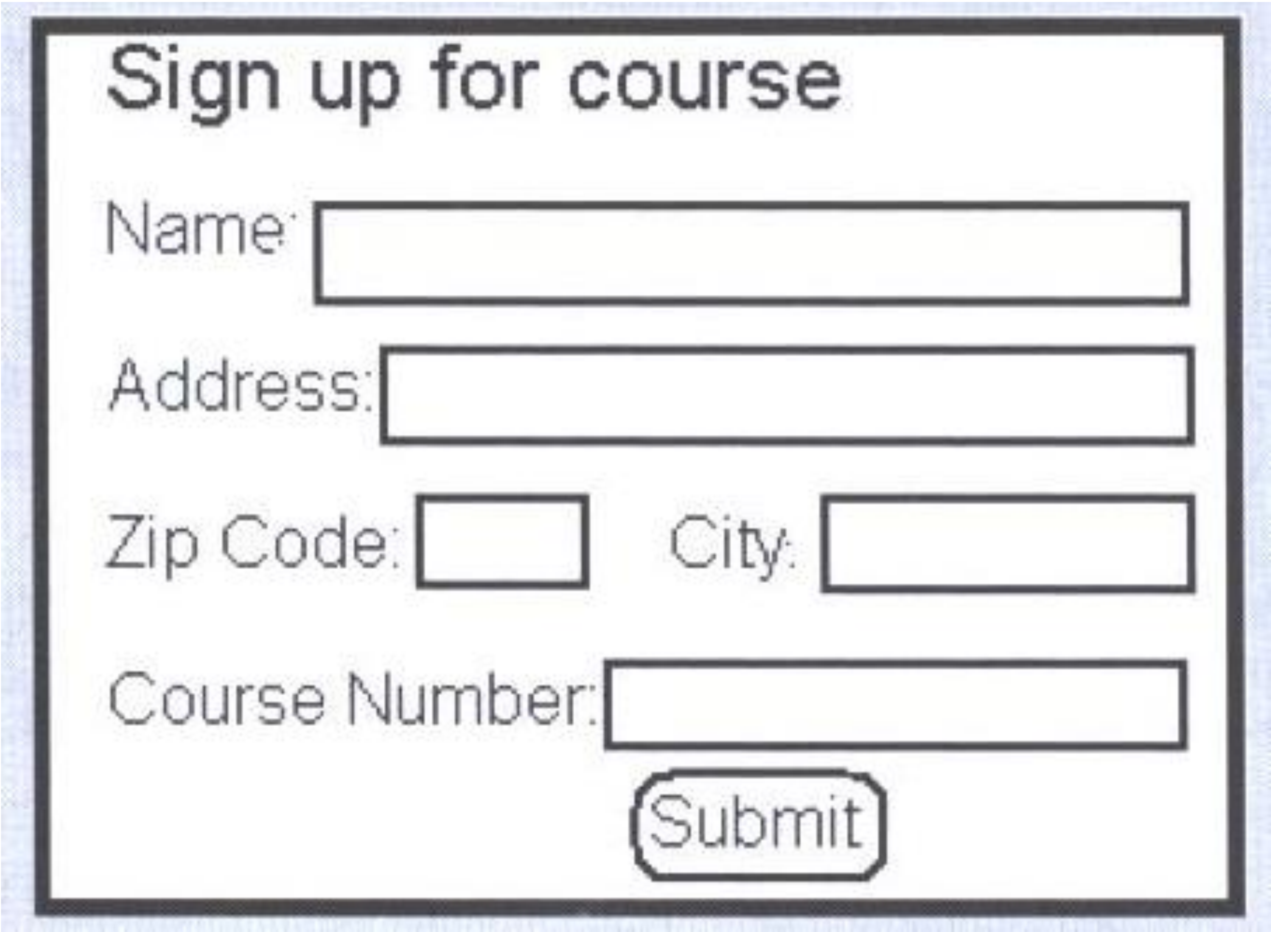
Intermediate nodes

10 The command is syntactically valid
11 The operands are syntactically valid

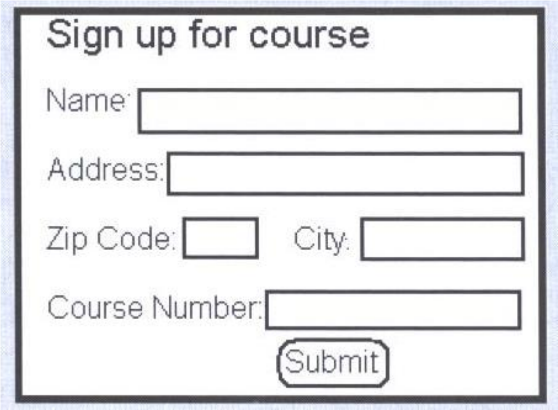
Example



Example – Web registration(1)



Example – Web registration(2)



Causes:
C1- name field is filled in
C2- name contains only letters and spaces
C3- Address field is filled in
C4- zip code is filled in
C5- city is filled in
C6 – course number is filled in
C7- course number exists in the system

Example – Web registration(3)

Intermediate variable :

I30- and (C1, C3, C4, C5, C6)

Effects

E51 – registration of student in system
E52 – message shown: All fields should be filled in
E53 – message shown: only letters and spaces in name
E54 – message shown: unknown course number
E55 – message shown – you have been registered

Example – Web registration(4)

Causes:
C1- name field is filled in
C2- name contains only letters and spaces
C3- Address field is filled in
C4- zip code is filled in
C5- city is filled in
C6 – course number is filled in
C7- course number exists in the system

Intermediate variable :

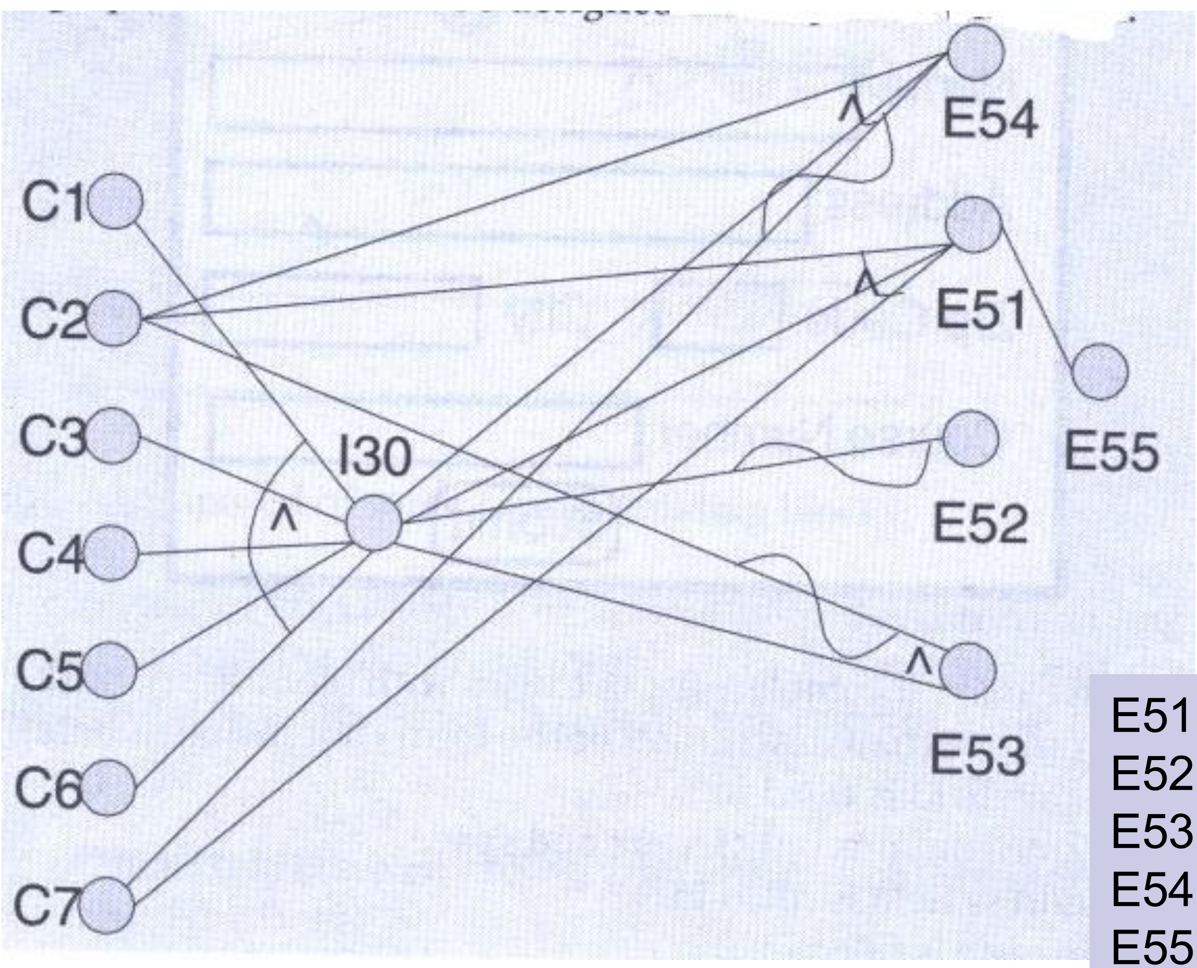
I30- and (C1, C3, C4, C5, C6)

Effects

E51 – registration of student in system
E52 – message shown: All fields should be filled in
E53 – message shown: only letters and spaces in name
E54 – message shown: unknown course number
E55 – message shown – you have been registered

E51= and (I30, C2, C7)
E52 = not(I30)
E53= and(I30,not C2)
E54=and(I30, C2, not C7)
E55 = E51

Example – Web registration(5)



E51= and (I30, C2, C7)
E52 = not(I30)
E53= and(I30,not C2)
E54=and(I30, C2, not C7)
E55 = E51

Example – GUI-based computer purchase system (1)

CPU	Printer	Monitor	RAM
CPU 1	PR 1	M 20	RAM 256
CPU 2	PR 2	M 23	RAM 512
CPU 3		M 30	RAM 1G

GUI – 5 windows– selection from CPU, Printer, Monitor, RAM
free giveaway

Assume RAM available only as an upgrade, one unit of each item purchased

Example – GUI-based computer purchase system (2)

CPU 1 CPU 2 CPU 3	PR 1 PR 2	M 20 M 23 M 30	RAM 256 RAM 512 RAM 1G

Monitors M 20 and M 23 can be purchased with any CPU or as a standalone item.
M 30 can be only purchased with CPU 3.
PR 1 is available free with a purchase of CPU 2 or CPU 3.
Monitors and printers, except for M 30 can also be purchased separately without purchasing any CPU.
Purchase of CPU 1 gets RAM 256 upgrade and purchase of CPU 2 or CPU 3 gets a RAM 512 upgrade.
The RAM 1G upgrade and a free PR 2 is available when CPU 3 is purchased with monitor M 30.

Example – GUI-based computer purchase system (3)

CPU 1 CPU 2 CPU 3	PR 1 PR 2	M 20 M 23 M 30	RAM 256 RAM 512 RAM 1G

Causes:
C_1 purchase CPU 1
C_2 purchase CPU 2
C_3 purchase CPU 3
C_4 purchase PR 1
C5 purchase PR 2
C6 purchase M 20
C7 purchase M 23
C8 purchase M 30

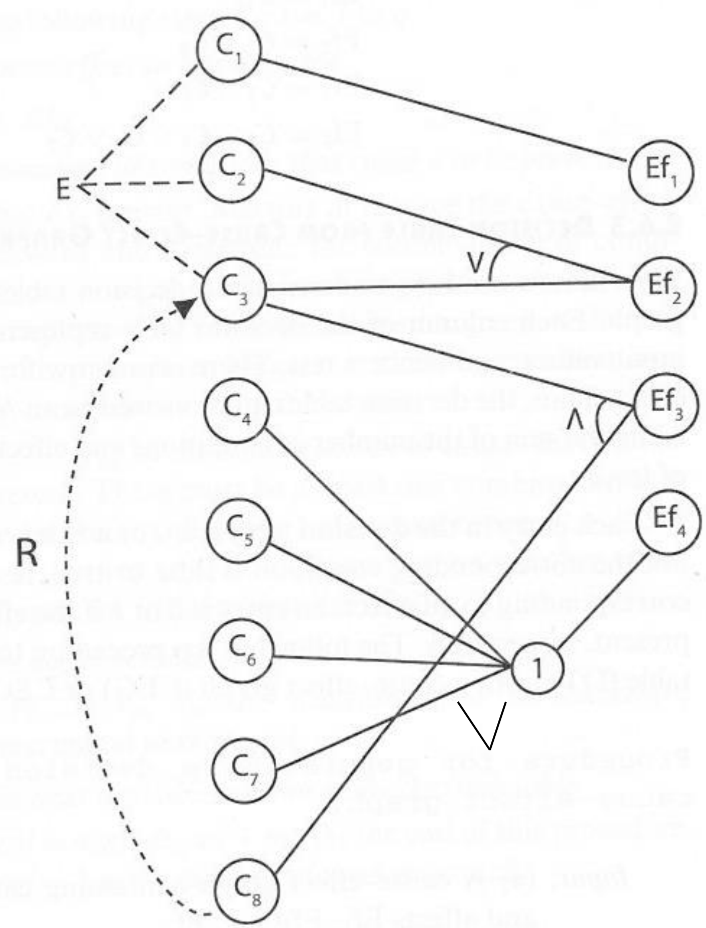
"free" display window:

effects:
Ef_1 RAM 256
Ef_2 RAM 512 and PR 1
Ef_3 RAM 1G and PR 2
Ef_4 no giveaway with this item

Example – GUI-based computer purchase system (4)

Causes:
C_1 purchase CPU 1
C_2 purchase CPU 2
C_3 purchase CPU 3
C_4 purchase PR 1
C5 purchase PR 2
C6 purchase M 20
C7 purchase M 23
C8 purchase M 30

effects:
Ef_1 RAM 256
Ef_2 RAM 512 and PR 1
Ef_3 RAM 1G and PR 2
Ef_4 no giveaway with this item



Example –ATM

A bank database which allows two commands

- Credit acc# amt
- Debit acc# amt

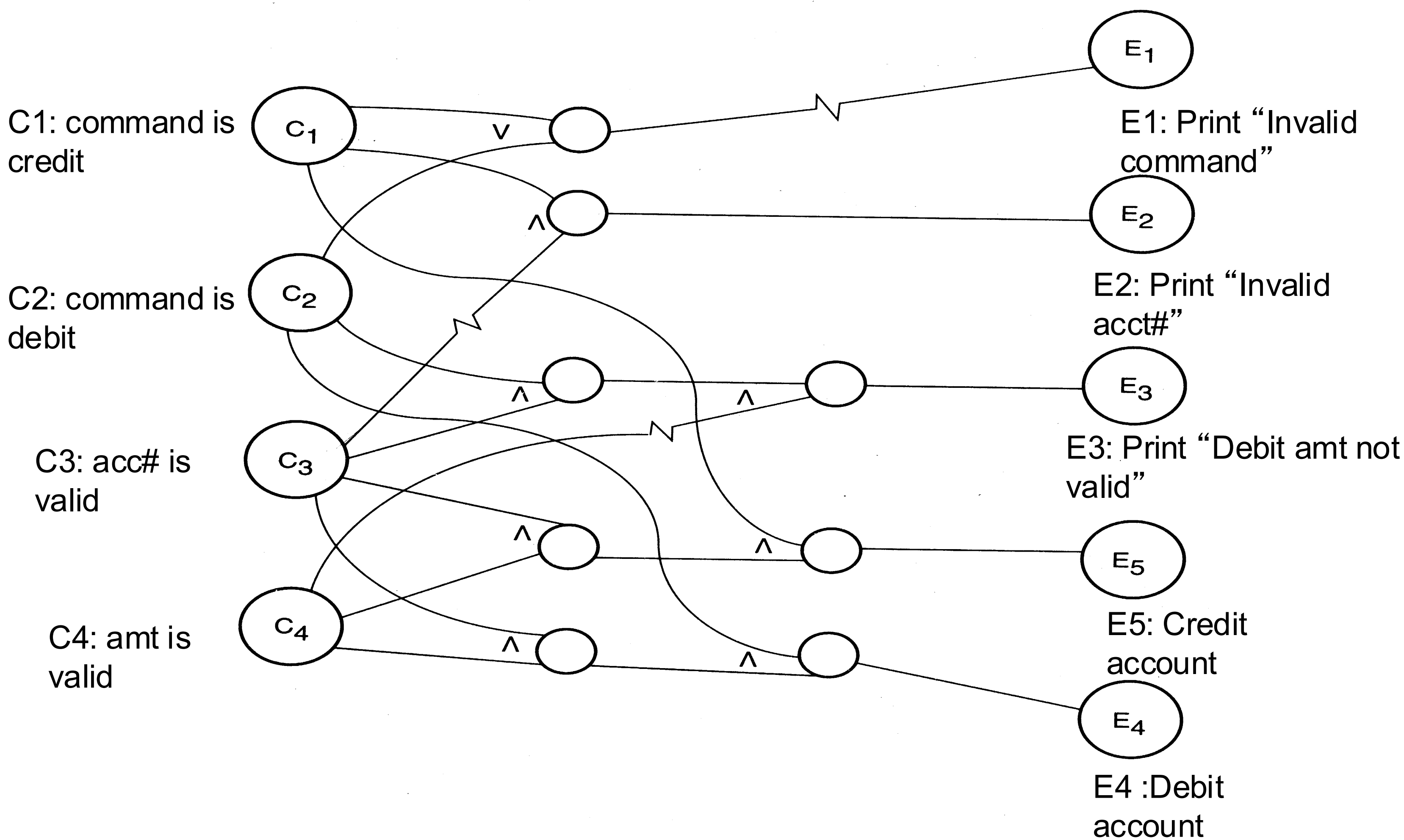
Requirements

- If credit and acc# valid, then credit
- If debit and acc# valid and amt less than balance, then debit
- Invalid command - message

Example-ATM

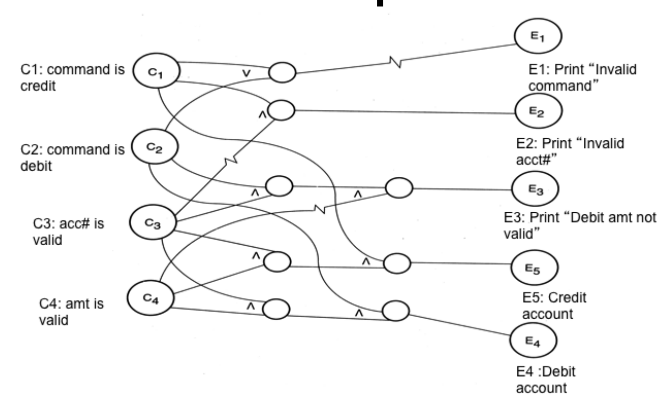
- Causes
  - C1: command is credit
  - C2: command is debit
  - C3: acc# is valid
  - C4: amt is valid
- Effects
  - Print “Invalid command”
  - Print “Invalid acct#”
  - Print “Debit amt not valid”
  - Debit account
  - Credit account

Cause-effect graph



Decision table

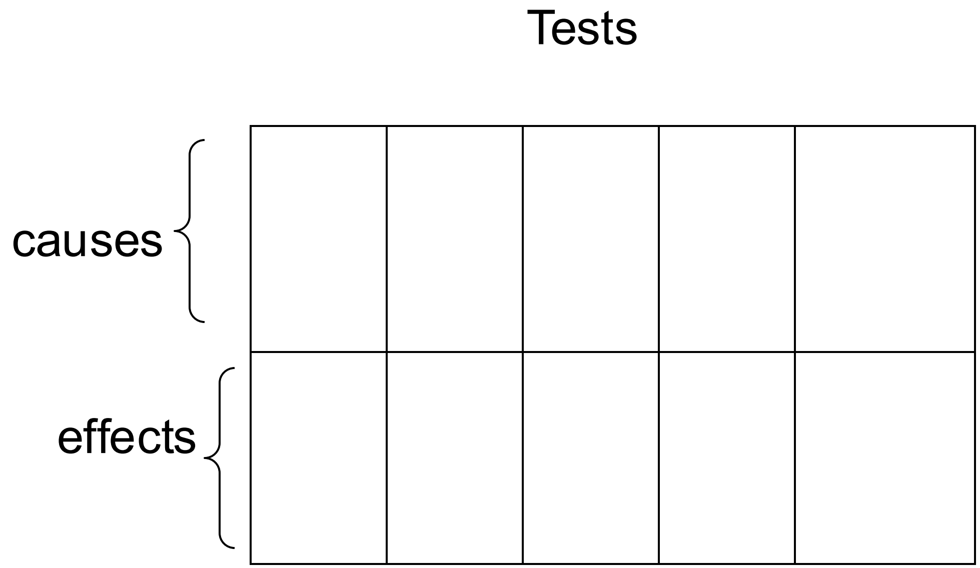
#	1	2	3	4	5
C1	0	1	x	x	1
C2	0	x	1	1	x
C3	x	0	1	1	1
C4	x	x	0	1	1
E1	1				
E2		1			
E3			1		
E4				1	
E5					1



Cause-Effect Graphs: Generation of decision table

Decision table:
causes – conditions
effects – actions

Each column == test

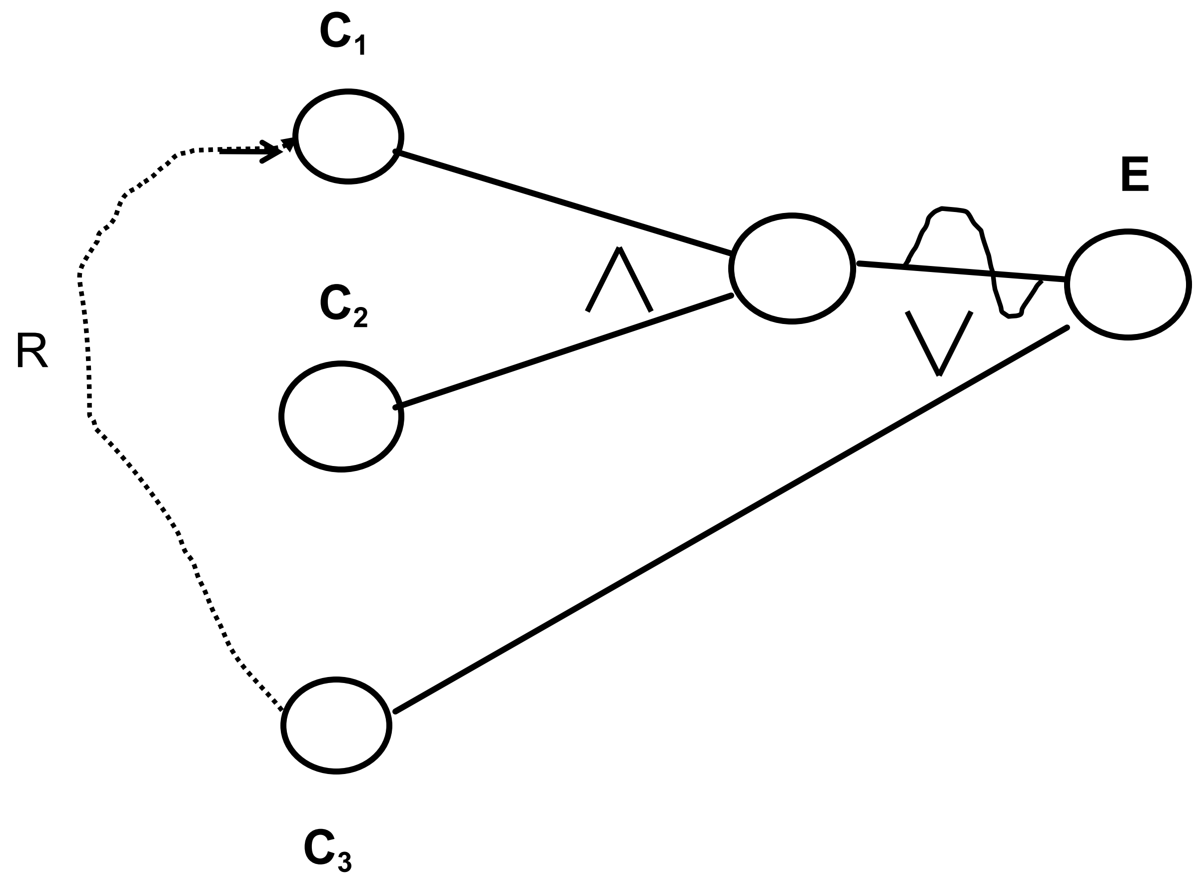


Select an effect to be present (1)

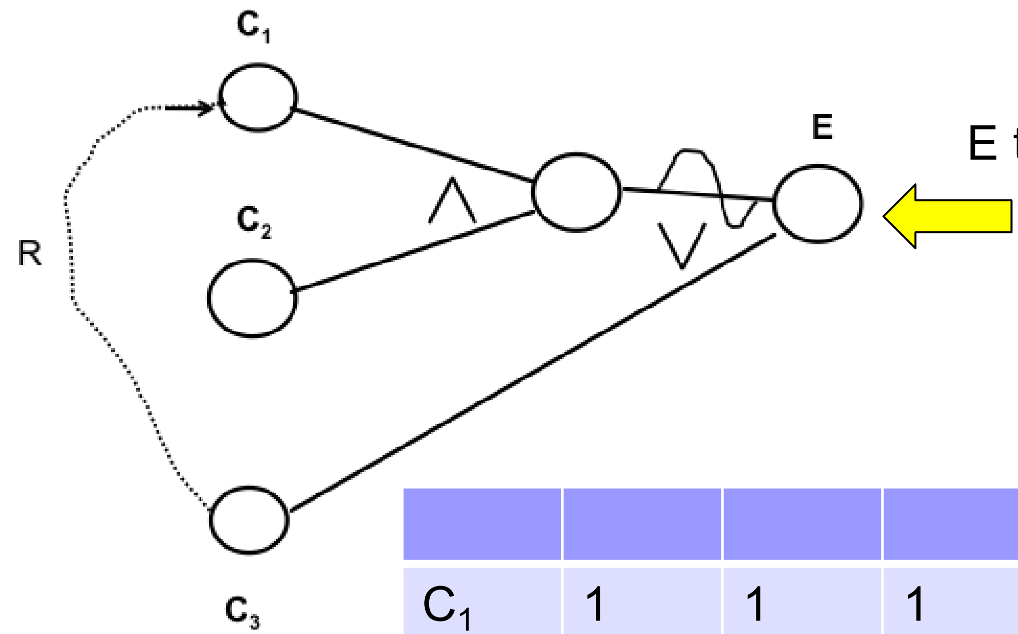
Trace back through the graph, find all combinations of causes (subject to constraints) that will set this effect to 1

Create a column in the decision table for each combination of causes

Generation of decision table: illustrative example



Generation of decision table: illustrative example



E to be equal to 1

C_1	1	1	1	0	0	0	0
C_2	0	1	0	1	0	1	0
C_3	1	1	0	0	0	1	1
E	1	1	1	1	1	1	1

C_3 requires C_1 C_1 must be 1 for C_3 to be 1

Tracing back the graph: heuristics

Reduce the number of test cases (possible combinatorial explosion)

Heuristics for tracing back through the graph :

and nodes

or nodes

Tracing back the graph path sensiting

Path sensitizing

failure to detect error because one error can mask another one.

Example

if (c1 or c2) then print message

erroneous implementation

If (c1 or (not (c2)) then print message

Cause –effect graph

Translation of specification → logic networks

Helps uncover ambiguities and incompleteness in specifications

General testing strategy

Specifications with combinations of input conditions –start with **cause-effect graph**

Use **boundary value analysis** (some tests could have been already generated by the cause-effect technique)

Identify valid and invalid **equivalence classes**

Error- guessing technique