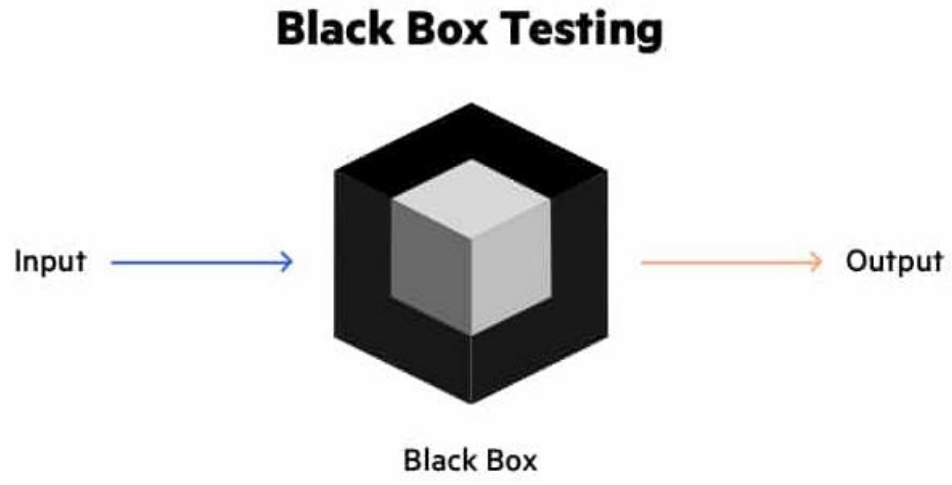
Tehnike crne kutije

FUNKCIONALNO TESTIRANJE

Funkcionalno testiranje

- Program se posmatra kao funkcija koja mapira vrednosti iz ulaznog domena u izlazni. Implemenacija nije poznata, program predstavlja "crnu kutiju" (kao što se tako kaže za neuronske mreže).
- Jedina informacija koja se koristi za određivanje test primera je specifikacija programa (dokumentacija).



Testiranje tehnikama crne kutije

- Podela na klase ekvivalencije (klasična definicija) – biće dodatno obrađeni u novom terminu
- Analiza graničnih vrednosti – biće dodatno obrađeni u novom terminu
- Testiranje zasnovano na tabeli odlučivanja
  - Uzročno-posledični graf
- Testiranje zasnovano na modelu stanja
- Testiranje sintakse
- Kombinatorno testiranje
  - Zasnovano na ortogonalnim nizovima

Uzročno-posledični grafovi[1]

- Alternativni naziv tehnike je modelovanje zavisnosti
- Fokusira se na modelovanje veza između ulaznih uslova programa, tzv. *uzroka* i izlaznih uslova, tzv. *posledica.*
- Zavisnost se predstavlja vizuelno u vidu uzročno-posledičnog grafa. U suštini je reč o logičkoj relaciji koja može da se predstavi bulovim izrazom.
- Graf omogućava izbor različitih kombinacija ulaznih vrednosti za testiranje. Kombinatorna eksplozija u broju testova izbegava se upotrebom heuristika tokom generisanja testova.

[1]The Art of Software Testing 3rd Edition, Glenford J. Myers, Corey Sandler, Tom Badgett, 2012.

Osnovni koraci u primeni ove metode

1. Specifikacija programa se podeli u radne delove koji se nezavisno testiraju. To je neophodno zbog prevelike kompleksnosti grafova kojima bismo pokušali da predstavimo velike specifikacije.

   Pojam radnog dela nije precizno definisan, ali to je, na primer, pri testiranju skript interpretera deo specifikacije koji se odnosi na jednu njegovu naredbu; ili, pri testiranju transakcionog sistema deo specifikacije koji se odnosi na pojedinačnu transakciju itd.

2. Određuju se **uzroci** i **posledice** u specifikaciji. Uzroci su u suštini klase ekvivalencije ulaznih uslova. Posledice su efekti na ponašanje programa koje ti ulazni uslovi imaju, tj. akcija programa koju oni izazivaju.

   Na primeru transakcionog sistema to bi bilo izmereno stanje u bazi podataka nakon transakcije ažuriranja ili odgovarajuća poruka o grešci, ukoliko je korisnik pokušao da unese neodgovarajuće podatke. Svim uzrocima i posledicama se dodeljuju jedinstveni identifikacioni brojevi.

Osnovni koraci u primeni ove metode

3. Na osnovu semantičke analize specifikacije kreira se uzročno-posledični graf. Uzroci i posledice predstavljaju početne i završne čvorove grafa, a veze se ostvaruju preko logičkih operatora koji čine grane grafa. Po potrebi se ubacuju i međučvorovi koji predstavljaju određene logičke kombinacije ulaznih uslova.
4. Ovako kreiran graf se dopunjuje ograničenjima. Ograničenja opisuju one kombinacije među uzrocima i među posledicama, koje je nemoguće ostvariti.
5. Graf se jednim metodološkim postupkom prevodi u tabelu odlučivanja. **Etaloni test primera su kolone ove tabele.**
6. Test primeri se generišu iz kolona tabele odlučivanja.

Formiranje grafa

Svaki čvor grafa predstavlja jedan ulazni uslov, kombinaciju ulaznih uslova ili posledicu i može biti u stanju 0 ili 1. Pri tome 1 označava da je uslov koji čvor predstavlja ispunjen, a 0 da nije ispunjen.

Grane grafa označavaju logičke funkcije koji važe među čvorovima grafa, a te funkcije mogu biti:

identity - ako čvor **a** ima vrednost 1 i čvor b ima vrednost 1, a u suprotnom ima vrednost 0

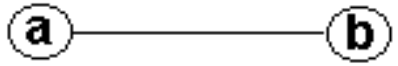
not - ako čvor a ima vrednost 1, čvor b ima vrednost 0, a u suprotnom ima vrednost 1

or - ako bar neki od čvorova a, b ili c ima vrednost 1 i čvor d ima vrednost 1, a u suprotnom ima vrednost 0. Broj ulaza nije ograničen.

and - ako i čvor a i čvor b imaju vrednost 1, onda i čvor c ima vrednost 1, a u suprotnom ima vrednost 0. Broj ulaza nije ograničen.

Grane grafa - grafička predstava

IDENTITY

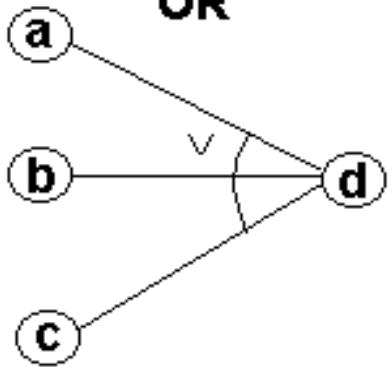


if a then b

NOT

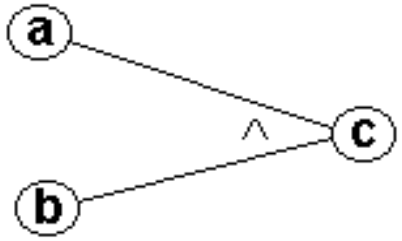
if (not a) then b

OR



if (a or b or c) then d

AND



if (a and b) then c

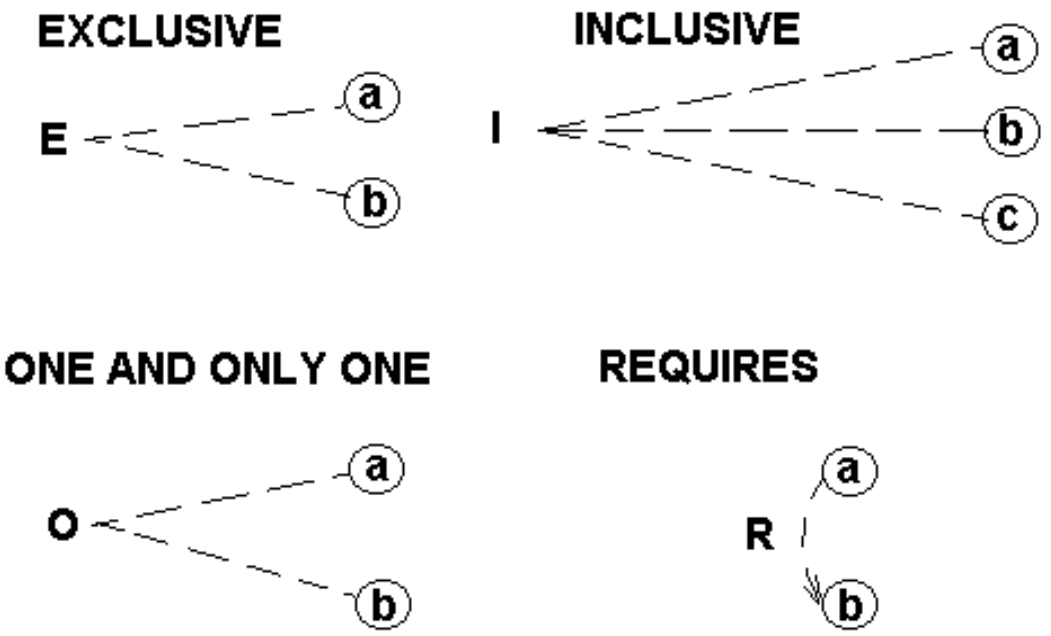
Ograničenja među uzrocima

E - najviše jedan od čvorova a i b može imati vrednost 1

I - bar jedan od čvorova a, b ili c mora imati vrednost 1

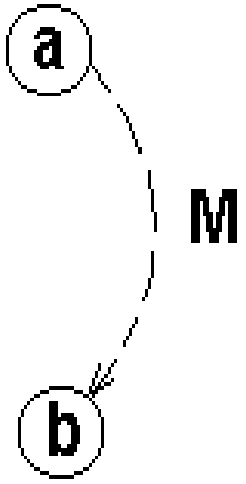
O - jedan, i samo jedan, od čvorova a i b mora imati vrednost 1

R - da bi čvor a imao vrednost 1, i čvor b mora imati vrednost 1



Ograničenje među posledicama

M - ako je završni čvor a na vrednosti 1, završni čvor b mora imati vrednost 0



Primer

Ef_1 : Generate "Shipping invoice".
Ef_2 : Generate an "Order not shipped" regret letter.

Effect Ef_1 occurs when an order can be met from the inventory. Effect Ef_2 occurs when the order placed cannot be met from the inventory or when the ordered item has been discontinued after the order was placed. However, Ef_2 is masked by Ef_1 for the same order, that is both effects cannot occur for the same order.

Primer

Iz britanskog standarda o testiranju softverskih komponenata
http://www.testingstandards.co.uk/

Deo specifikacije zahteva

- Razmotrimo funkciju koja obradjuje podizanje novca. Ulazi funkcije su količina novca koja se podiže, tip računa i trenutno stanje računa, dok su izlazi novo stanje računa i kod akcije.
- Tip računa može biti poštanski (‘p’) ili klasični (‘c’).
- Kod akcije može biti:
  - **D&L** - obradi podizanje novca i pošalji notifikaciju,
  - **D** - samo obradi podizanje novca,
  - **S&L** - blokiraj račun i pošalji notifikaciju,
  - **L** - samo pošalji notifikaciju

Nastavak...

Specifikacija funkcije data je u nastavku:

- Ukoliko ima dovoljno novčanih sredstava na računu ili bi novo stanje bilo u granicama dozvoljenog minusa, podizanje novca se obradjuje.
- Ukoliko bi podizanje novca dovelo do prekoračenja dozvoljenog minusa, podizanje novca nije moguće, dodatno ukoliko je u pitanju poštanski račun vrši se privremeno blokiranje istog.
- Notifikacija se šalje za sve obavljene transakcije u slučaju poštanskog računa, kao i za ne-poštanski račun ukoliko nema dovoljno novčanih sredstava (tj. račun više nije u plusu).

Uzroci i posledice:

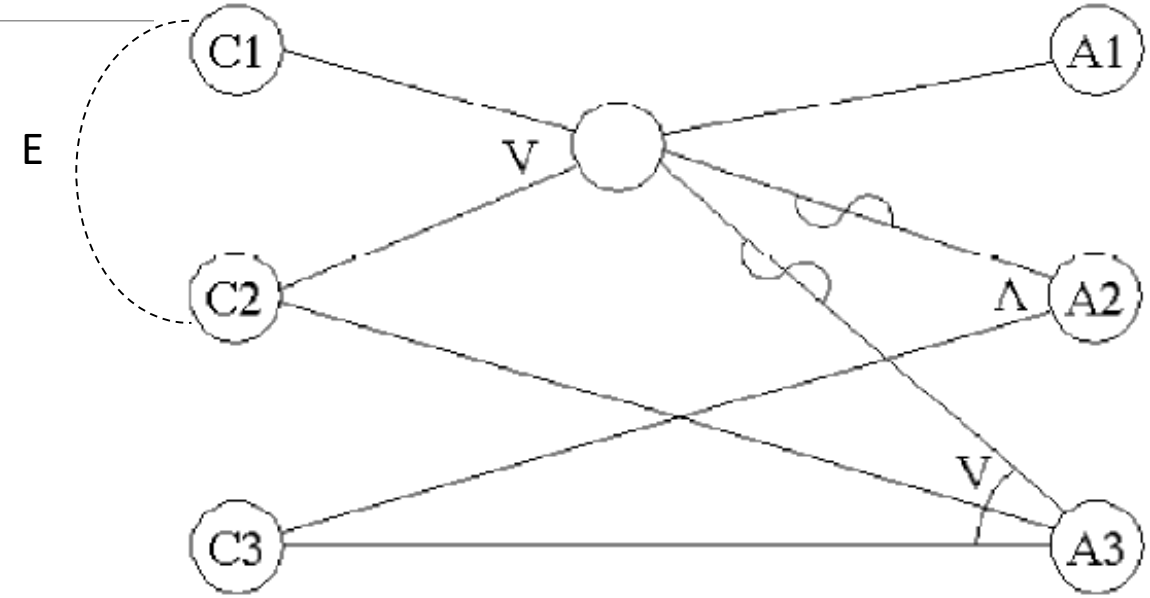
Uzroci:

- C1 - Novo stanje je u plusu
- C2 - Novo stanje je u dozvoljenom minusu
- C3 - Račun je poštanski

Posledice:

- A1- Obrada podizanja novca
- A2 - Privremeno blokiranje računa
- A3 - Slanje notifikacije

Uzročno posledični Graf



C1 Novo stanje je u plusu

C2 Novo stanje je u dozvoljenom minusu

C3 Račun je poštanski

A1 Podizanje novca

A2 Blokiranje računa

A3 Slanje notifikacije

Primer tabele odlučivanja

Dvodimenzionalno mapiranje uzroka na posledice može se izvesti iz uzročno posledičnog grafa

Rules:	1	2	3	4	5	6	7	8
C1: New balance in credit	F	F	F	F	T	T	T	T
C2: New balance overdraft, but within authorised limit	F	F	T	T	F	F	T	T
C3: Account is postal	F	T	F	T	F	T	F	T
A1: Process debit	F	F	T	T	T	T	*	*
A2: Suspend account	F	T	F	F	F	F	*	*
A3: Send out letter	T	T	T	T	F	T	*	*

- ima po jedan red za svaki uzrok i posledicu
- Kolone odgovaraju test primerima.
  (T-mora biti ispunjeno, F-ne sme biti ispunjeno,
  *-kombinacija uzroka nemoguća, te akcije nisu definisane)

Primer tabele odlučivanja

Za prethodni primer tabele odlučivanja, teorijski postoji $2^3=8$ test primera (kombinacija vrednosti uzroka).

- Upotrebom **Majersove tehnike redukcije kombinacija uzroka** ovaj broj se smanjuje.

(*Glenford Myers, The Art of Software Testing, Second Edition, John Wiley&Sons, 2004)*

Transformacija grafa u tabelu odlučivanja

1. Jedan po jedan završni čvor se postavlja na vrednost 1.
2. Ide se od tog završnog čvora unazad kroz graf i pronalaze *sve* kombinacije ulaznih čvorova koje ga setuju na 1, vodeći računa o ograničenjima.
3. Od svake kombinacije ulaznih čvorova, koja setuje odabrani završni čvor na 1, formira se jedna kolona u tabeli odlučivanja. U slučaju da vrednost nekog ulaznog čvora nije bitna ili da se podrazumeva na osnovu vrednosti ostalih ulaznih čvorova i važećih ograničenja, taj ulaz u koloni tabele odlučivanja se ostavlja praznim.
4. Za svaku od gornjih kombinacija se određuje i efekat koji ima na stanje ostalih završnih čvorova. Ovo je korisno za fazu testiranja programa pomoću generisanih test problema, jer se unapred identifikuju efekti koji se očekuju pri izvršavanju programa pomoću test problema.
5. Ukoliko se pojave kolone u tabeli odlučivanja koje se preklapaju, vrši se čihovo sažimanje. Preklapanje se javlja kada su im vrednosti po svim čvorovima jednake ili kad je na određenim pozicijama vrednost u jednoj koloni nebitna, a u drugoj fiksno određena (tada se uzima ona fiksna vrednost).

Transformacija grafa u tabelu odlučivanja

Majersove heuristike

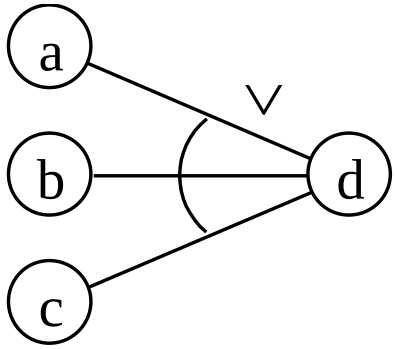
Da bi se smanjio broj kombinacija koje se međusobno preklapaju, pri izvođenju koraka 2 se primenjuju sledeće "olakšice":

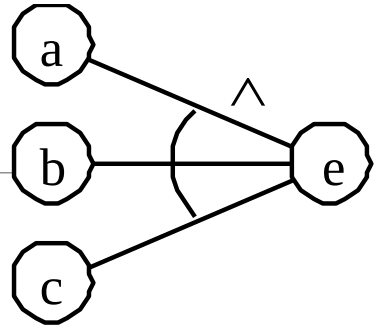
1. Kada se ide unazad kroz čvor sa operatorom **OR**, čiji izlaz treba da bude **1**, nikada se ne ispituje situacija u kojoj je više od jednog ulaza postavljeno na 1. Cilj ovoga je da se izbegne mogućnost da ne dođe do otkrivanja greške u slučaju da jedan ulazni uslov *maskira* drugi.
2. Kada se ide unazad kroz čvor sa operatorom **AND**, čiji izlaz treba da bude **0**, treba ispitati sve kombinacije ulaza koje dovode izlaz na 0. Međutim, za one kombinacije u kojima su neki od ulaza na vrednosti 1, nije potrebno ispitati sve kombinacije koje taj ulaz dovode na vrednost 1, već je dovoljna samo po jedna kombinacija.
3. Kada se ide unazad kroz čvor sa operatorom **AND**, čiji izlaz treba da bude **0**, za slučaj kada svi ulazi imaju vrednost 0, nije potrebno ispitati sve kombinacije koje dovode svaki od ulaza na vrednost 0, već je dovoljna samo po jedna kombinacija.

Majersove heuristike

∀ - označava da treba uzeti u obzir sve moguće kombinacije

∃ - označava da treba izabrati samo jednu jedinu kombinaciju





d / ∀0

a	b	c
∀0	∀0	∀0

d / ∀1

a	b	c
∀0	∀0	∀1
∀0	∀1	∀0
∀1	∀0	∀0

e / ∀1

a	b	c
∀1	∀1	∀1

e / ∀0

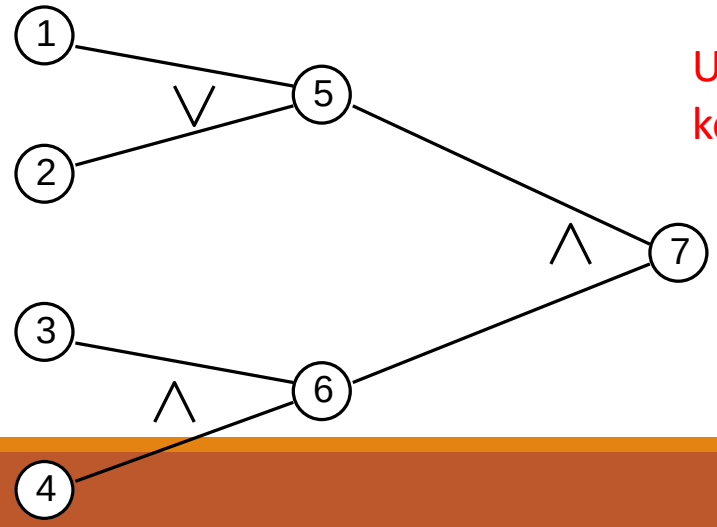
a	b	c
∃0	∃0	∃0
∀0	∀0	∃1
∀0	∃1	∀0
∀0	∃1	∃1
∃1	∀0	∀0
∃1	∀0	∃1
∃1	∃1	∀0

Primer transformacije

Pretpostavimo da želimo da odredimo sve kombinacije ulaznih uslova, koje dovode izlaz na stanje **nula**. Olakšica 3 kaže da bi trebalo da uzmemo samo jednu kombinaciju za slučaj kada su čvorovi 5 i 6 na nuli. Olakšica 2 kaže da bi u slučaju kada je čvor 5 na jedinici i čvor 6 na nuli, trebalo uzeti samo jednu kombinaciju koja dovodi čvor 5 na jedinicu, umesto da razmatramo sve moguće kombinacije koje dovode čvor 5 na jedinicu.

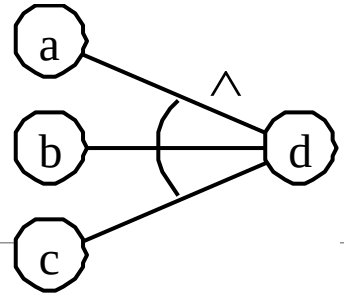
Za slučaj kada je čvor 5 na nuli, a čvor 6 na jedinici, treba ispitati samo jednu kombinaciju koja dovodi čvor 6 na jedinicu. Olakšica 1 kaže da kada čvor 5 treba da ima vrednost jedan, ne treba istovremeno postavljati čvorove 1 i 2 na jedinicu. Tako dolazimo do **pet kombinacija** vrednosti čvorova 1 do 4, kao na primer:

- 0 0 0 0 (5=0, 6=0)
- 1 0 0 0 (5=1, 6=0)
- 1 0 0 1 (5=1, 6=0)
- 1 0 1 0 (5=1, 6=0)
- 0 0 1 1 (5=0, 6=1)

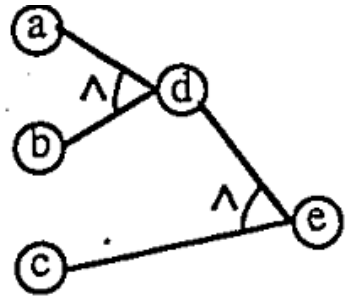


Umesto 13 mogućih kombinacija koje dovode čvor 7 na nulu, imamo 5!

Kritike majersovog pristupa



1. Nekonzistencija u definiciji pravila



(a) An Example CEG

∀

		1	2	3	4	5
Cause						
	a	0	1	0	0	1
	b	0	1	0	1	0
	c	0	0	1	1	1
Effect						
	e	0	0	0	0	0

∃

		1	2	3
Cause				
	a	0	1	0
	b	0	1	0
	c	0	0	1
Effect				
	e	0	0	0

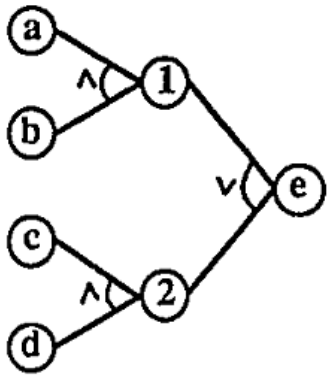
$\frac{d}{\forall 0}$

a	b	c
∃0	∃0	∃0
?0	?0	∃1
?0	∃1	?0
?0	∃1	∃1
∃1	?0	?0
∃1	?0	∃1
∃1	∃1	?0

? Na jednom mestu u knjizi ∀, na drugom ∃

Kritike majersovog pristupa

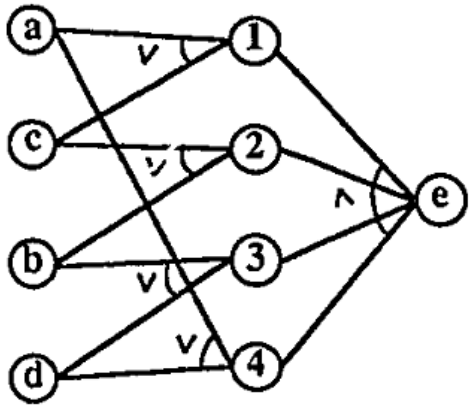
2. Različiti grafovi sa istim značenjem daju različite tabele odlučivanja



(a) CEG of e = (a and b) or (c and d)

	1	2	3	4	5	6
Causes						
a	1	1	1	0	1	0
b	1	1	1	0	0	1
c	0	1	0	1	1	1
d	0	0	1	1	1	1
Effects						
e	1	1	1	1	1	1

(c) Decision Table from CEG in (a)



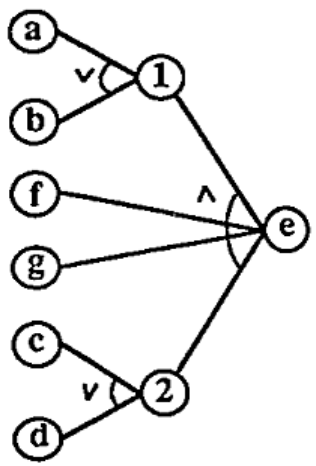
(b) CEG of e = (a or c) and (a or d) and (b or c) and (b or d)

	1	2
Causes		
a	1	0
b	1	0
c	0	1
d	0	1
Effects		
e	1	1

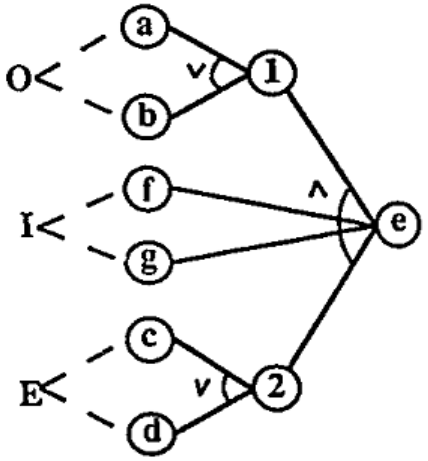
(d) Decision Table from CEG in (b)

Kritike majersovog pristupa

3. Nije lako uočiti grešku izostavljanja ograničenja uzroka



(a) Example of a CEG with missing constraints



(b) Added constraints to CEG in (a)

	1	2	3	4
Causes				
a	1	1	0	0
b	0	0	1	1
c	1	0	1	0
d	0	1	0	1
f	1	1	1	1
g	1	1	1	1
Effects				
e	1	1	1	1

(c) Same decision table for both CEGs in (a) and (b)

Kritike majersovog pristupa

4. Traže se samo kombinacije uzroka koje dovode posmatranu posledicu na 1.

Na ovaj način, nije lako uočiti da li je posledica uvek na 1 ili uvek na 0 (nezavisno od uzroka), što bi potencijalno ukazalo na grešku u specifikaciji.

Tehnika senzitizacije putanja

Alternativna tehnika za izvođenje tabele odlučivanja iz uzročno-posledičnog grafa

Def. 1: **Skup uzroka** posledice E je skup uzroka koji imaju uticaja na E.

Def. 2: **Senzitizacija** posledice E na određeni uzrok C znači da se svi drugi ulazi dovode na fiksne vrednosti tako da vrednost E direktno zavisi samo od C (direktno ili invertovano).

Def. 3: **Senzitizovana putanja** je putanja od uzroka C do posledice E na kojoj su svi međučvorovi senzitizovani na uzrok C.

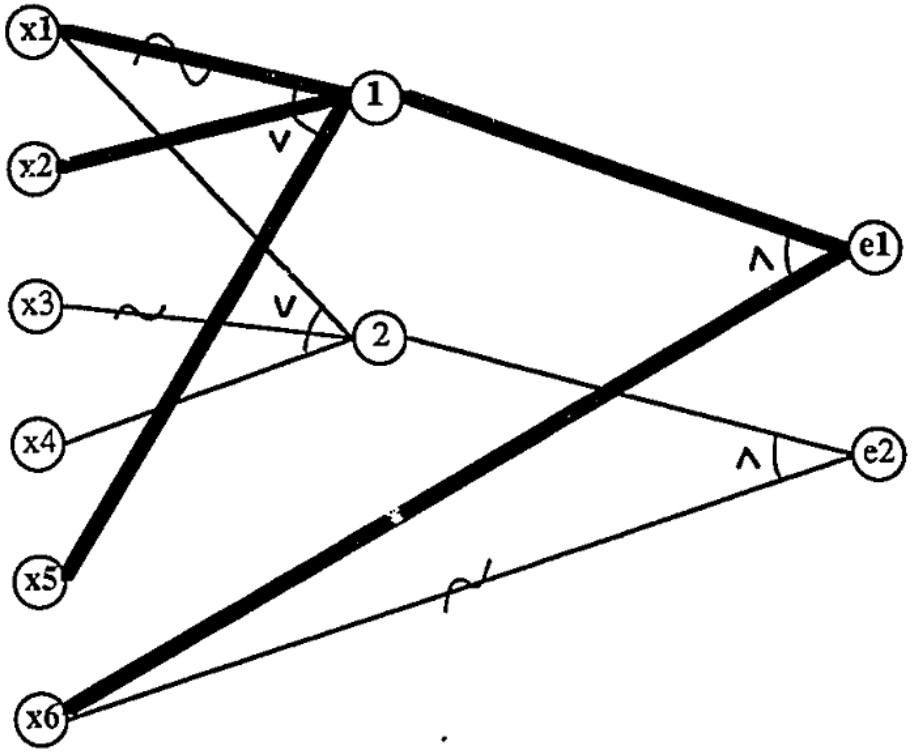
Tehnika senzitizacije putanja-primer

CauseSet(e_1)={x_1, x_2, x_5, x_6}

CauseSet(e_2)={x_1, x_3, x_4, x_6}

Želimo da senzitivizujemo e_1 na x_2:

- Moramo prvo da senzitizacija 1 na x_1, znači x_1=1 i x_5=0, zbog **ILI** čvora
- Sada smo senzitizaciju e_1 na x_2 sveli na senzitizaciju e_1 na 1, tako da još treba x_6=1, jer je e_1 čvor **I**
- Dakle postigli smo e_1=x_2



Tehnika senzitizacije putanja

For each effect, ej in a cause-effect graph CEG do

For each cause cj in the Cause Set C of effect ej do

-> Sensitize effect ej to cause cj

-> For each possible combinations create two cause combinations

one with cj set to 1 and another one with cj set to 0

-> Create a column in the decision table for each cause combination

-> Use the values set to the causes in C and the constraints present in the CEG to determine the values of the other causes in the cause effect graph

-> Use the values set to the causes in C in order to determine the values of the other effects in CEG for these two combinations

-> Remove any redundant column

End for

-> If possible, create two additional cause combinations where in one most of the causes in C are absent and in the other most of them are present. *

-> Create a column in the decision table for each cause combination

-> Use the values set to the causes in C and the constraints present in the CEG to determine the values of the other causes in the CEG

-> Use the values set to the causes in C in order to determine the values of the other effects in CEG for these two combinations

-> Remove any redundant column

End for

-> Derive test scenarios from the decision table obtained

Primer – sendfile komanda

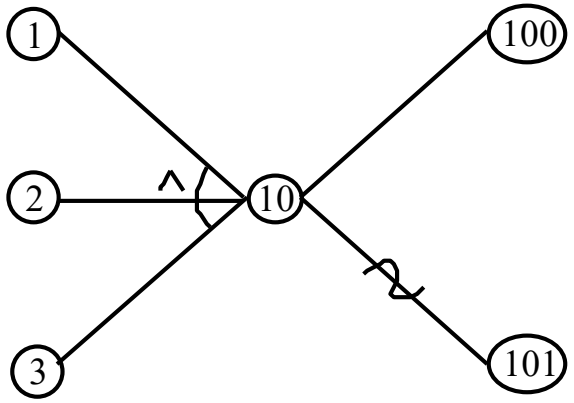
U datoj mreži, **Sendfile** komanda se koristi da pošalje datoteku korisniku na nekom drugom serveru datoteka. Sendfile komanda uzima tri argumenta:

- prvi argument treba da bude postojeća datoteka u home direktorijumu pošiljaoca,
- drugi argument treba da bude ime datoteke servera primaoca,
- treći argument treba da bude userid primaoca.

Ako su svi argumenti tačni, datoteka se uspešno šalje, u suprotnom pošiljalac dobija poruku o grešci.

Primer – sendfile komanda

Uzroci	Posledice
1. Prvi argument je ime postojećeg fajla u home direktorijumu pošiljaoca. 2. Drugi argument je ime servera primaoca. 3. Treći argument je primaočev userid.	100. Fajl je uspešno poslat. 101. Pošiljalac dobija poruku o grešci.



Senzitizacija putanja *

	1	2	3	4	5
Causes					
1	**1**	**0**	1	1	0
2	**1**	1	**0**	1	0
3	**1**	1	1	**0**	0
Effects					
100	**1**	**0**	**0**	**0**	0
101	**0**	**1**	**1**	**1**	1

Majers

	1	2	3	4	5	6	7	8
Causes								
1	1	0	1	0	0	1	1	0
2	1	0	0	1	0	1	0	1
3	1	0	0	0	1	0	1	1
Effects								
100	1	0	0	0	0	0	0	0
101	0	1	1	1	1	1	1	1

Uzročno-posledični grafovi

Nisu praktično primenljivi na sisteme koji:

- Poseduju vremenska ograničenja
- Poseduju konkurentne procese