Universidade de Brasília
Faculdade Gama
Engenharia de Software

Disciplina: **FGA0238 – Testes de Software**

Critérios de Testes Caixa Preta ou Teste Baseado em Especificação

Elaine Venson
elainevenson@unb.br

Agenda

- Abordagens ou técnicas de teste
- Caixa preta ou teste baseado em especificação
  - Particionamento de Equivalência
  - Análise de Valor Limite
  - Grafo de Causa-Efeito
  - Error Guessing

Casos de Teste

- É impossível na prática executar todas as possibilidades de casos de teste
  - Teste exaustivo
- Quais situações testar?
- Restrições de tempo e custo

Qual é o subconjunto de todos os possíveis casos de teste com a maior probabilidade de detectar a maior parte dos defeitos?

Técnicas e Critérios de Teste

UnB

- É impraticável testar o domínio de entrada de forma completa
- No projeto dos casos de teste procura-se utilizar um subconjunto reduzido que tenha alta probabilidade de revelar defeitos
- Um **subdomínio de teste** é um subconjunto do domínio de entrada que contém dados de teste “semelhantes”

Técnicas e Critérios de Teste

UnB

- Se todos os casos de teste de um subdomínio se comportam do mesmo modo, basta executar o programa com apenas um deles
- Ao identificar todos os subdomínios e selecionar um caso de teste para cada subdomínio, consegue-se um conjunto de casos de testes bastante reduzido, mas que representa cada um dos seus elementos

Técnicas e Critérios de Teste

UnB

- Abordagens:
  - **Teste aleatório**: um grande número de casos de teste é selecionado aleatoriamente para que probabilisticamente se tenha uma boa chance de que todos os subdomínios estejam representados
  - **Teste de partição / teste de subdomínio**: estabelecer os subdomínios e selecionar casos de teste dos subdomínios
- Como estabelecer os subdomínios?
  - Critérios de teste: regras que identificam quando dados de teste devem estar no mesmo subdomínio ou não

Técnicas e Critérios de Teste

UnB

- Principais abordagens:
  - Teste de **caixa preta** ou **teste functional** ou teste **baseado em especificação**
  - Teste de **caixa branca** ou **teste estrutural**
  - Teste baseado em erros
- As abordagens se diferenciam pela pela **fonte** utilizada para definir os requisitos de teste: requisitos funcionais, código ou defeitos
- Cada abordagem explora determinados tipos de defeitos
- Recomenda-se combinar as abordagens

Caixa Preta

- Trata o software ou sistema como uma “caixa-preta”
- Orientado a dados, orientado a entradas/saídas
- Não é necessário entender a estrutura interna do código
- Concentra-se em identificar as situações em que a aplicação não se comporta de acordo com as especificações
- Testa a especificação e não garante que todas as partes do código foram testadas
  - Indica a parte da especificação que não foi atendida adequadamente

Caixa Preta

- Características
  - Testadores não precisam ser técnicos
  - Adequado para encontrar *bugs* que os usuários finais encontrarão
  - Os casos de testes podem ser projetados assim que as especificações funcionais estejam completas
  - Difícil de identificar todos as possíveis combinações de entradas em um tempo limitado do projeto
  - A qualidade do teste dependerá da qualidade das especificações de requisitos

Caixa Preta

- Para identificar todos os defeitos de um programa a partir desta abordagem seria necessário um **teste exaustivo de entradas** -> não é possível na prática
  - Por exemplo, para testar um compilador C seria necessário criar casos de teste para todos os programas válidos e inválidos (infinito...)
- Implicações:
  - O teste caixa-preta não garante que um programa não tenha defeitos
  - Questões econômicas: maximizar o número de defeitos encontrados com um número finito de casos de teste

Caixa Preta

- Como identificar o melhor conjunto de Casos de Testes?
- Técnicas de derivação de casos de teste (mais conhecidos):
  - Particionamento de equivalência
  - Análise de valor limite
  - Grafo causa-efeito
  - *Error guessing*
- A qualidade dos casos de testes derivados destas técnicas depende de uma boa especificação de requisitos

Particionamento de Equivalência

UnB

- Divide o **domínio de entrada** em classes de dados que são tratados da mesma maneira de acordo com a especificação de requisitos
- Para ajudar a identificação das classes pode-se observar na especificação **os termos "intervalo" e "conjunto"** ou palavras similares que indiquem que os dados são processados da mesma forma
- Considera-se que qualquer elemento de uma classe pode ser seu representante, uma vez que todos eles devem se comportar de forma similar
- Redução do domínio de entrada a um tamanho finito de casos
- Além do particionamento pelo domínio de entrada, alguns autores consideram também o domínio de saída

Particionamento de Equivalência

UnB

Procedimento:

1. Identificar classes de equivalência
   - Condições de entrada
   - Classes válidas e inválidas
2. Identificar casos de teste
   - Atribuir um número para cada classe de equivalência
   - Escrever casos de teste cobrindo o máximo de classes válidas até que todas as classes válidas tenham sido cobertas
   - Escrever um caso de teste exclusivo para cada classe inválida

Particionamento de Equivalência

Procedimento:

1. **Identificar classes de equivalência**
   - **Condições de entrada**
   - **Classes válidas e inválidas**
2. Identificar casos de teste
   - Atribuir um número para cada classe de equivalência
   - Escrever casos de teste cobrindo o máximo de classes válidas até que todas as classes válidas tenham sido cobertas
   - Escrever um caso de teste exclusivo para cada classe inválida

Particionamento de Equivalência

- Identificação das classes:
  - Identificar cada condição de entrada e particioná-la em uma classe válida e uma classe inválida

Condição de Entrada	Classes de Equivalência Válidas	Classes de Equivalência Inválidas

Particionamento de Equivalência

UnB

- Uma classe de equivalência representa um **conjunto de valores válidos e inválidos** como condições de entrada
- Identificação das classes [Myers]:
  - **Intervalo de valores**: definir uma classe válida (valores pertencentes ao intervalo) e duas inválidas (valores menores que o limite inferior e valores maiores que o limite superior
    - Ex.: comprimento entre 1 m e 50 m -> uma válida (1 <= comprimento <= 50), uma inválida (comprimento < 1) e outra inválida (comprimento > 50)
  - **Quantidade de valores**: definir uma classe válida e duas inválidas
    - Ex.: grupo pode ter de 1 a 4 membros -> uma válida entre 1 e 4, uma inválida para nenhum membro e outra inválida para mais de 4 membros
  - **Lista de valores de entrada que podem ser tratados de forma diferente**: uma classe de equivalência para cada valor (válidas) e uma classe para valor fora da lista (inválida)
    - Ex.: tipo de disciplina (obrigatória, optativa, módulo livre, obrigatória seletiva) -> uma classe válida para cada tipo de disciplina e uma classe inválida (eletiva)
  - **Situação do tipo "deve ser de tal forma"**: uma classe válida e uma inválida
    - Ex.: "o primeiro caractere deve ser uma letra" -> uma classe válida (uma letra) e uma inválida (caractere não letra)

Particionamento de Equivalência

Procedimento:

1. Identificar classes de equivalência
   - Condições de entrada
   - Classes válidas e inválidas
2. **Identificar casos de teste**
   - **Atribuir um número para cada classe de equivalência**
   - **Escrever casos de teste cobrindo o máximo de classes válidas até que todas as classes válidas tenham sido cobertas**
   - **Escrever um caso de teste exclusivo para cada classe inválida**, uma vez que um elemento de uma classe inválida pode mascarar a validação do elemento de outra classe inválida

Particionamento de Equivalência

UnB

Exemplo – Programa Cadeia de Caracteres:

1. *O programa solicita do usuário um inteiro positivo no intervalo de 1 a 20*
2. *Em seguida, solicita uma cadeia de caracteres no tamanho informado*
3. *Após isso, o programa solicita um caractere*
4. *Então o sistema retorna:*
   - *a posição na cadeia em que o caractere é encontrado pela primeira vez*
   - *Ou uma mensagem de erro indicando que o caractere não está presente*
5. *O usuário tem a opção de procurar outros caracteres. Continuar (sim) ou sair do programa (não)*

Particionamento de Equivalência

- Exemplo: (cont)
  - Identificar cada condição de entrada

Condição de Entrada	Classes Válidas	Classes Inválidas
Tamanho da CC (T)		
Cadeia de caracteres (CC)		
Caracter a ser procurado (C)		
Opção de procurar mais caracteres (O)		

Particionamento de Equivalência

- Exemplo: (cont)
  - Particionar cada condição em classes válidas e inválidas

Condição de Entrada	Classes Válidas	Classes Inválidas
Tamanho da CC (T)	1 <= T <= 20	T < 1 e T > 20
Cadeia de caracteres (CC)	CC = T	CC ≠ T
Caracter a ser procurado (C)	Pertence Não pertence	
Opção de procurar mais caracteres (O)	S N	Outro

Particionamento de Equivalência

- Exemplo: (cont)
  - Atribuir um número para cada classe de equivalência

Condição de Entrada	Classes Válidas	Classes Inválidas
Tamanho da CC (T)	1 <= T <= 20 (1)	T < 1 (2) e T > 20 (3)
Cadeia de caracteres (CC)	CC = T (4)	CC ≠ T (5)
Caracter a ser procurado (C)	Pertence (6) Não pertence (7)	
Opção de procurar mais caracteres (O)	S (8) N (9)	Outro (10)

Particionamento de Equivalência

- Exemplo: (cont)
  - Identificadas as classes, escolhem-se, arbitrariamente os elementos de cada classe para definir os casos de teste:

Variável de Entrada				Saída Esperada	Classes
T	CC	C	O		
3	abc	c	s	O Caractere c aparece na posição 3 da cadeia	1, 4, 6, 8
		k	n	O Caractere k não pertence à cadeia	7, 9
0				Entre com um intervalo entre 1 e 20	2
34				Entre com um intervalo entre 1 e 20	3
5	Xxx			Cadeia com tamanho inválido	5
2	Av	X	X	Opção inválida	10

Particionamento de Equivalência

UnB

- Possibilita **redução do domínio de entrada**
- Criação de dados de teste baseados **unicamente na especificação**
- Adequado para aplicações em que as **variáveis de entrada** podem ser identificadas com facilidade e assumem valores específicos
- Não é facilmente aplicável quando o domínio de entrada é simples, mas o **processamento é complexo**
- **Problemas** ocorrem quando a especificação sugere que dados são processados de forma semelhante, mas na prática não ocorre
- Não fornece **diretrizes para a determinação dos dados de teste** e combinações entre eles que possam cobrir as classes de equivalência de forma mais eficiente

Elabore casos de testes para o seguinte requisito

Especificação: a senha do usuário deve conter entre 6 e 8 caracteres. Deve conter letras, caracteres especiais e pelo menos dois dígitos numéricos, obrigatoriamente.

- Exemplos
  - abc*12 (válido)
  - wxyz*1 (inválido)
  - 3b@n@n@s (inválido)
  - 1a@#$%^& (inválido)

Identifique as condições de entrada

Especificação: a senha do usuário deve conter entre 6 e 8 caracteres. Deve conter letras, caracteres especiais e pelo menos dois dígitos numéricos, obrigatoriamente.

Condição de Entrada	Classes Válidas	Classes Inválidas

Identifique as condições de entrada

Especificação: a senha do usuário deve conter entre 6 e 8 caracteres. Deve conter letras, caracteres especiais e pelo menos dois dígitos numéricos, obrigatoriamente.

Condição de Entrada	Classes Válidas	Classes Inválidas
Tamanho T da senha		
Contém letras		
Contém caracteres especiais		
Quantidade D de dígitos		

Identifique as classes válidas e inválidas

Especificação: a senha do usuário deve conter entre 6 e 8 caracteres. Deve conter letras, caracteres especiais e pelo menos dois dígitos numéricos, obrigatoriamente.

Condição de Entrada	Classes Válidas	Classes Inválidas
Tamanho T da senha		
Contém letras		
Contém caracteres especiais		
Quantidade D de dígitos		

Identifique as classes válidas e inválidas

Especificação: a senha do usuário deve conter entre 6 e 8 caracteres. Deve conter letras, caracteres especiais e pelo menos dois dígitos numéricos, obrigatoriamente.

Condição de Entrada	Classes Válidas	Classes Inválidas
Tamanho T da senha	6 <= T <= 8	T < 6 e T > 8
Contém letras	Sim	Não
Contém caracteres especiais	Sim	Não
Quantidade D de dígitos	D >= 2	D < 2

Enumere as classes

Especificação: a senha do usuário deve conter entre 6 e 8 caracteres. Deve conter letras, caracteres especiais e pelo menos dois dígitos numéricos, obrigatoriamente.

Condição de Entrada	Classes Válidas	Classes Inválidas
Tamanho T da senha	6 <= T <= 8	T < 6 e T > 8
Contém letras	Sim	Não
Contém caracteres especiais	Sim	Não
Quantidade D de dígitos	D >= 2	D < 2

Enumere as classes

Especificação: a senha do usuário deve conter entre 6 e 8 caracteres. Deve conter letras, caracteres especiais e pelo menos dois dígitos numéricos, obrigatoriamente.

Condição de Entrada	Classes Válidas	Classes Inválidas
Tamanho T da senha	6 <= T <= 8 (1)	T < 6 (2) e T > 8 (3)
Contém letras	Sim (4)	Não (5)
Contém caracteres especiais	Sim (6)	Não (7)
Quantidade D de dígitos	D >= 2 (8)	D < 2 (9)

Elabore os casos de teste

Casos de teste:
- Casos de teste cobrindo o máximo das classes válidas
- Um caso de teste para cada classe inválida

Condição de Entrada	Classes Válidas	Classes Inválidas
Tamanho T da senha	6 <= T <= 8 (1)	T < 6 (2) e T > 8 (3)
Contém letras	Sim (4)	Não (5)
Contém caracteres especiais	Sim (6)	Não (7)
Quantidade D de dígitos	D >= 2 (8)	D < 2 (9)

Elabore os casos de teste

Condição de Entrada	Classes Válidas	Classes Inválidas
Tamanho T da senha	6 <= T <= 8 (1)	T < 6 (2) e T > 8 (3)
Contém letras	Sim (4)	Não (5)
Contém caracteres especiais	Sim (6)	Não (7)
Quantidade D de dígitos	D >= 2 (8)	D < 2 (9)

- Casos de teste:

Entrada	Saída Esperada	Classes
aaab*88	Senha válida	1, 4, 6, 8
Ab12$	Senha inválida	2
Rsrs12345^	Senha inválida	3
$$$$$12	Senha inválida	5
12abcd	Senha inválida	7
One&two2	Senha inválida	9

Análise de Valor Limite

- Técnica baseada em seleção de dados de testes onde **valores extremos** são escolhidos
  - Valores que estão exatamente sobre ou imediatamente acima ou abaixo dos limitantes das classes de equivalência
  - Máximo, mínimo, perto dos limites (min, min-1, max, max+1), valores típicos, valores de erro
- Se baseia no fato de que, se o sistema funciona corretamente com esses **valores especiais**, então funcionará com todos os valores de dentro dos limites
- Técnica **usada em conjunto com o particionamento de equivalência**, explorando os limitantes de cada classe de equivalência
- Exige criatividade e um certo domínio da área relacionada ao sistema

Análise de Valor Limite

- Recomendações [Myers]:

1. Se a condição de entrada especifica um **intervalo de valores**, devem ser definidos dados de teste para os limites do intervalo e para as classes inválidas vizinhas, imediatamente subsequentes

    - Ex.: para um intervalo de temperaturas entre -20,0 e + 100,0 definir os dados de teste:

        - - 20,0 (valor válido)
        - +100,0 (valor válido)
        - -20,001 (valor inválido)
        - +100,001 (valor inválido)

Análise de Valor Limite

- Recomendações [Myers]:

2. Se a condição de entrada especifica uma **quantidade de valores**, devem ser definidos dados de teste para as quantidades de valores válidas e para as classes inválidas vizinhas, imediatamente subsequentes
    - Ex.: se um arquivo de entrada pode ter de 1 a 255 registros definir dados de teste para:
        - 1 registro (classe válida)
        - 255 registros (classe válida)
        - Nenhum registro (classe inválida)
        - 256 registros (classe inválida)

Análise de Valor Limite

- Recomendações [Myers]:

3. Usar a diretriz 1 para as condições de saída
4. Usar a diretriz 2 para as condições de saída
5. Se a entrada ou saída for um conjunto ordenado, deve ser dada maior atenção aos primeiro e último elementos desse conjunto
6. Usar a intuição para definir outras condições limites

Análise de Valor Limite

- **Exemplo - Programa Cadeia de Caracteres**
  - Considerando as classes de equivalência identificadas anteriormente:

Condição de Entrada	Classes Válidas	Classes Inválidas
Tamanho da CC (T)	1 <= T <= 20 (1)	T < 1 (2) e T > 20 (3)
Cadeia de caracteres (CC)	CC = T (4)	CC ≠ T (5)
Caracter a ser procurado (C)	Pertence (6) Não pertence (7)	
Opção de procurar mais caracteres (O)	S (8) N (9)	Outro (10)

Análise de Valor Limite

- Exemplo: (cont)
  - Identificadas as classes, escolhem-se, arbitrariamente os elementos de cada classe para definir os casos de teste:

Variável de Entrada				Saída Esperada	Classes
T	CC	C	O		
1	a	a	s	O Caractere a aparece na posição 1	1, 4, 6, 8
1	a	k	n	O Caractere k não pertence à cadeia	1, 4, 7, 9
0				Entre com um intervalo entre 1 e 20	2
21				Entre com um intervalo entre 1 e 20	3
1	xx			Cadeia com tamanho inválido	5
20	abcdefghijklmnopqrstu			Cadeia com tamanho inválido	5
20	abcdefghijklmnopqrst	a	n	O caracter a aparece na posição 1	1, 4, 6, 8
20	abcdefghijklmnopqrst	t	n	O caracter t aparece na posição 20	1, 4, 6, 8

Análise de Valor Limite

- As vantagens e desvantagens do critério de Análise do Valor Limite são similares às do critério Particionamento de Equivalência
- Adicionalmente existem diretrizes para que os dados de teste sejam estabelecidos (exploração dos limites)

Elabore casos de testes para os seguintes requisitos

- As alíquotas de IR dependem do valor do salário (base de cálculo)
  - Até 1.499,15: isentos
  - De 1.499.16 até 2.246,75: 7,5%
  - De 2.246,76 até 2995,70: 15%
  - De 2995,71 até 3.743,19: 22,5%
  - Acima de 3.743,19: 27,5%

Grafo Causa-Efeito

- As técnicas de particionamento de equivalência e análise de valor limite consideram cada valor de entrada isoladamente
- A técnica Grafo de Causa-Efeito ajuda a definir um conjunto de casos de teste que exploram combinações de entradas
- O grafo é uma **linguagem formal** para a qual a especificação é traduzida
  - Na realidade é um circuito digital com uma notação simples
- Utiliza tabelas de decisão e árvores de decisão
- Pode apontar ambiguidades e incompletezas na especificação

Grafo Causa-Efeito

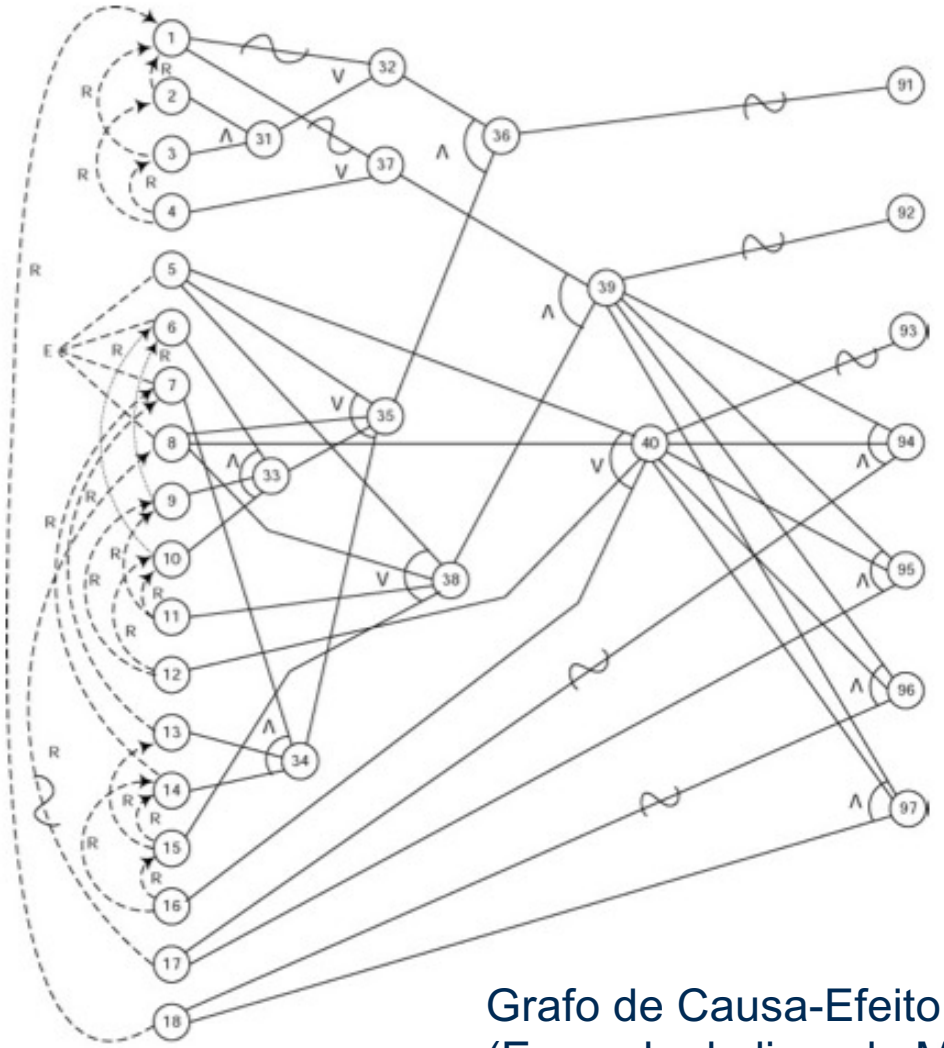
- Causas -> condições de entrada ou classe de equivalência
- Efeitos -> resultados gerados em resposta às diferentes condições de entrada
- Exemplo:
  - Causa:
    - 7,0 <= nota < 9,0 **E** frequência >= 75%
  - Efeito:
    - conceito = MS

Grafo Causa-Efeito

Procedimento:

1. Desenhar o grafo:
   - Identificar as causas (entradas) e listar verticalmente à esquerda do diagrama
   - Identificar efeitos (saídas) e listar verticalmente à direita do diagrama
   - Relacionar as causas como os efeitos através da notação específica do grafo causa-efeito
2. Transformar o grafo em tabela de decisão
   - Para cada um dos efeitos gerar combinações entre causas que façam com que os efeitos sejam ativados
3. As colunas da tabela de decisão são convertidas em casos de teste

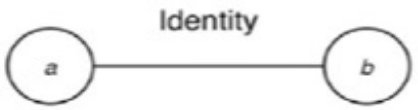
Grafo Causa-Efeito



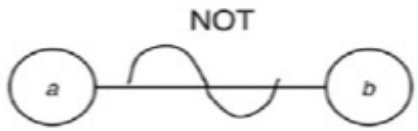
Grafo de Causa-Efeito do comando DISPLAY
(Exemplo do livro do Myers)

Grafo Causa-Efeito

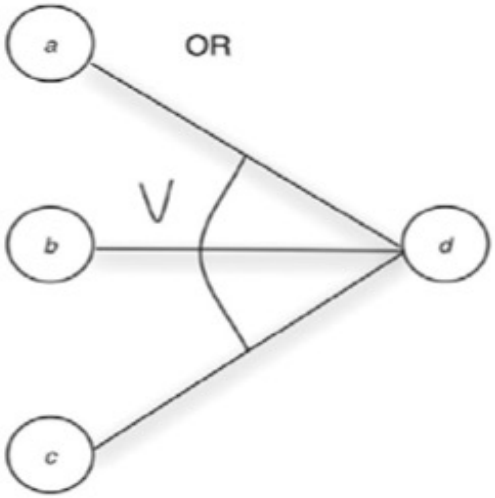
- Notação do grafo e interpretação
  - Cada nó possui valor 0 (estado ausente) ou 1 (estado presente)



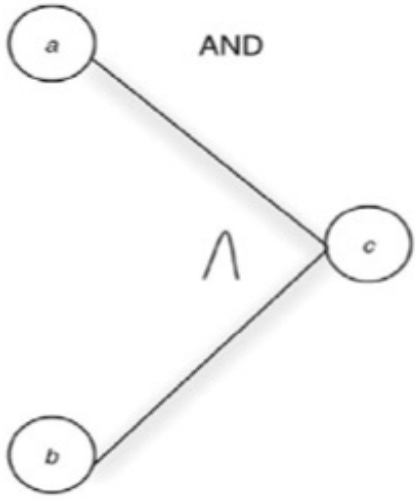
Se a é 1, b é 1; senão b é 0



Se a é 1, b é 0; senão b é 1



Se a ou b ou c são 1, d é 1; senão d é 0

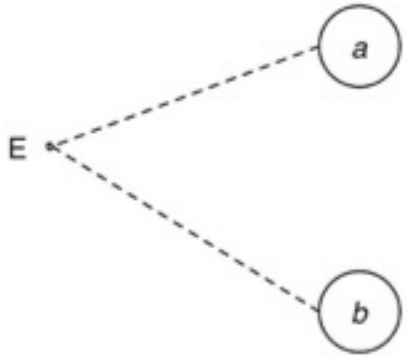


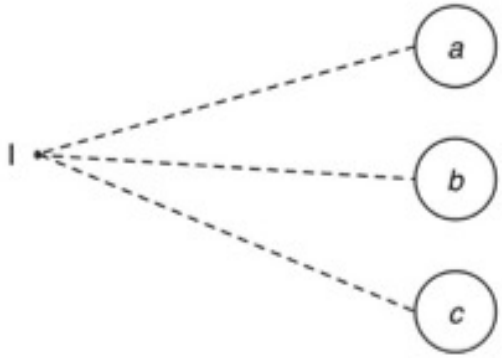
Se a e b são 1, c é 1; senão c é 0

Grafo Causa-Efeito

- Notação de restrições para entradas

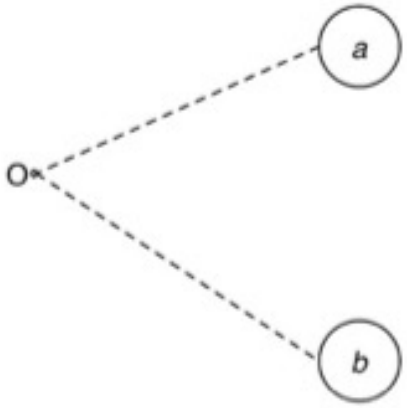
No máximo um entre A e B pode ser igual a 1

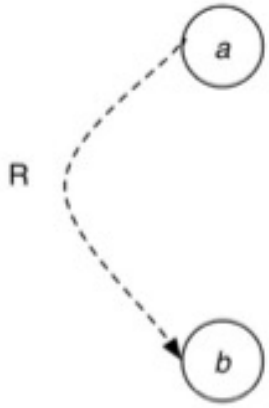




No mínimo um entre A, B e C deve ser igual a 1

Um e somente um entre A e B deve ser igual a 1



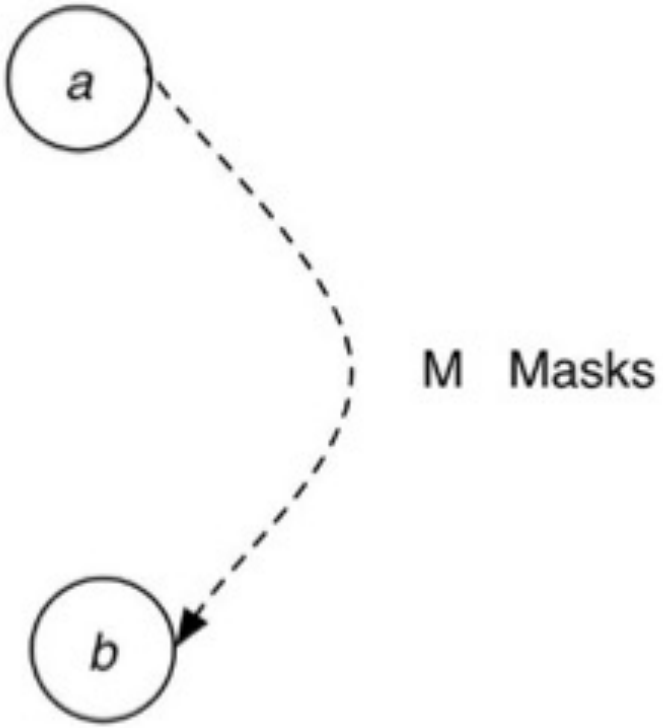


Para que A seja igual a 1, B deve ser igual a 1

Grafo Causa-Efeito

- Notação de restrição para saídas

Se o efeito A é 1, o efeito B é forçado a ser 0



Grafo Causa-Efeito

- **Exemplo - Programa Imprime Mensagens:**
  - O programa lê dois caracteres e, de acordo com eles, mensagens serão impressas da seguinte forma:
    - O primeiro caractere deve ser A ou B
    - O segundo caractere deve ser um dígito
    - Exemplos: “A1”, “B8”
    - Nessa situação, efetua uma atualização no arquivo
  - Se o primeiro caractere for incorreto (ex. “C1”), enviar mensagem X
  - Se o segundo caractere for incorreto (ex. “AA”), enviar mensagem Y

Grafo Causa-Efeito

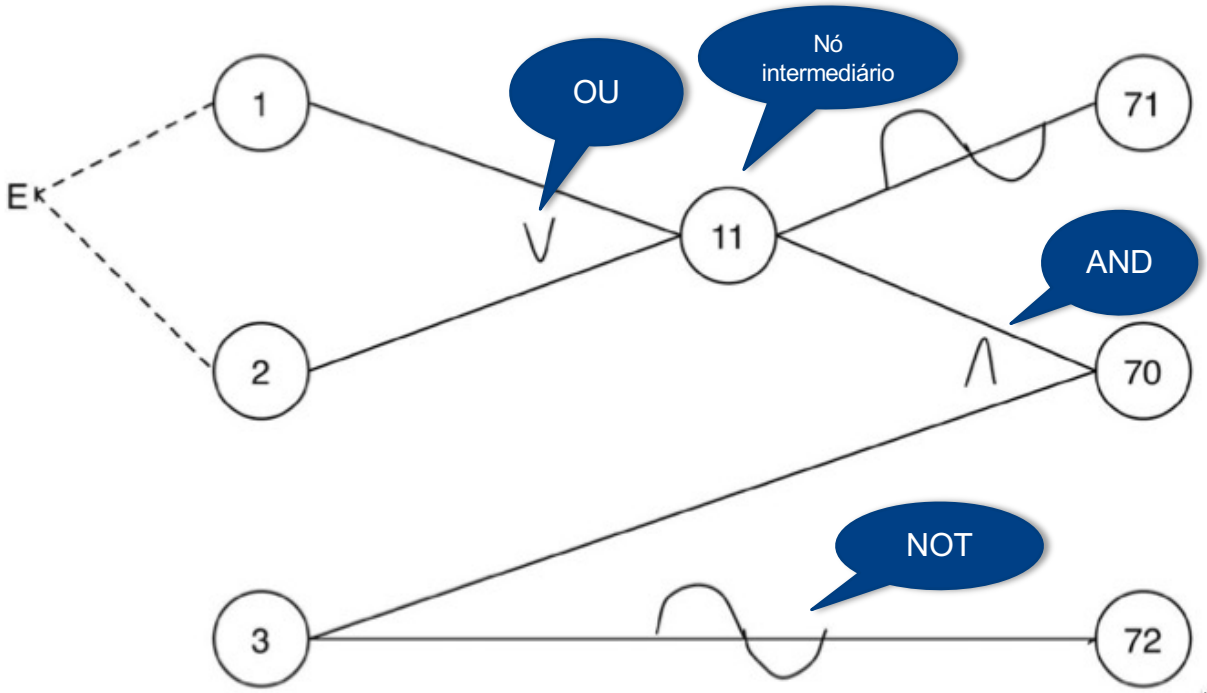
Causas, Efeitos e Grafo Correspondente

Causas

1) primeiro caractere é A
2) primeiro caractere é B
3) segundo caractere é um dígito

Efeitos

70) A atualização é realizada
71) A mensagem X é enviada
72) A mensagem Y é enviada



Grafo Causa-Efeito

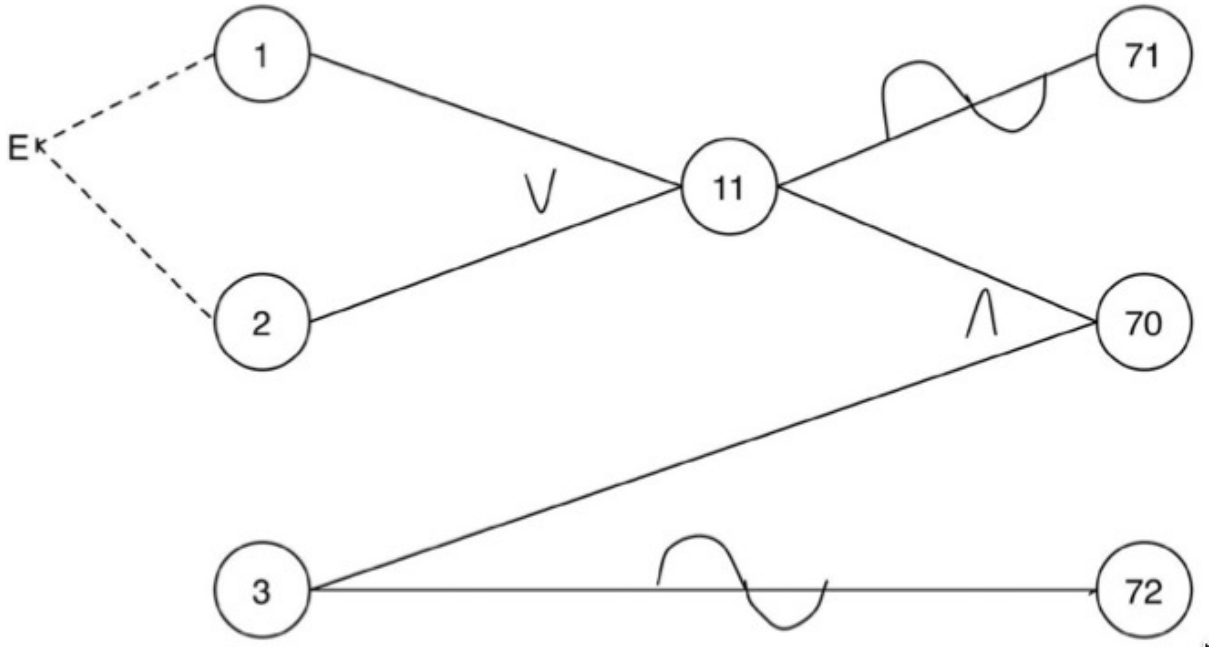
- Procedimento para elaborar a **tabela de decisão**
  1. Selecionar um efeito para estar no estado presente, isto é, com valor 1
  2. Rastrear o grafo para trás, encontrado todas as possíveis causas (sujeitas a restrições) que fazem com que esse efeito seja 1
     - Quando o nó for do tipo OR e a saída deve ser 1, nunca atribuir mais de uma entrada com valor 1 simultaneamente
     - Quando o nó for do tipo AND e a saída deve ser 0:
       - todas as combinações de entrada que levem à saída 0 devem ser enumeradas. Mas quando uma entrada é 0 e uma ou mais das outras for 1, não é necessário enumerar todas as condições em que as outras entradas sejam 1
       - somente uma condição em que todas as entradas sejam 0 precisa ser enumerada (caso específico de nós intermediários)
  3. Criar uma coluna na tabela de decisão para cada combinação de causas
  4. Determinar, para cada combinação, os estados de todos os outros efeitos, anotando na tabela

Grafo Causa-Efeito

Exempo: Tabela de Decisão

1. Selecionar um efeito para estar no estado presente, isto é, com valor 1

1	
2	
3	
70	
71	1
72	



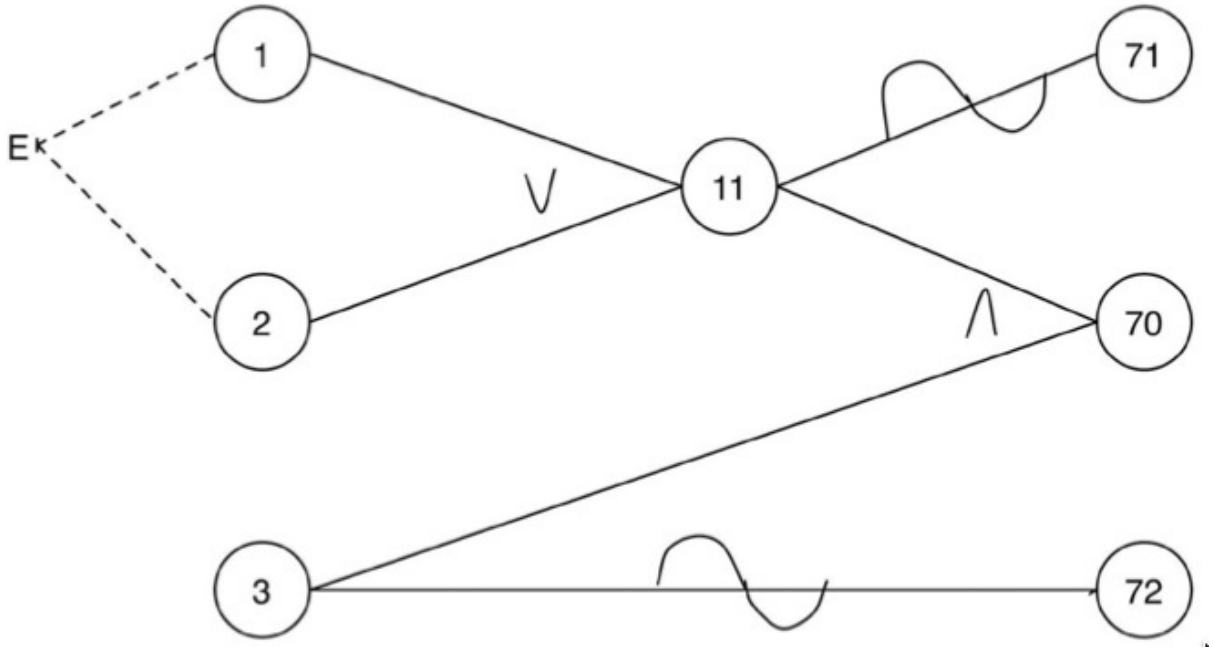
Grafo Causa-Efeito

Exempo: Tabela de Decisão

2. Rastrear o grafo para trás, encontrado todas as possíveis causas (sujeitas a restrições) que fazem com que esse efeito seja 1

- Quando o nó for do tipo OR e a saída deve ser 1, nunca atribuir mais de uma entrada com valor 1 simultaneamente

1	0
2	0
3	
70	
71	1
72	

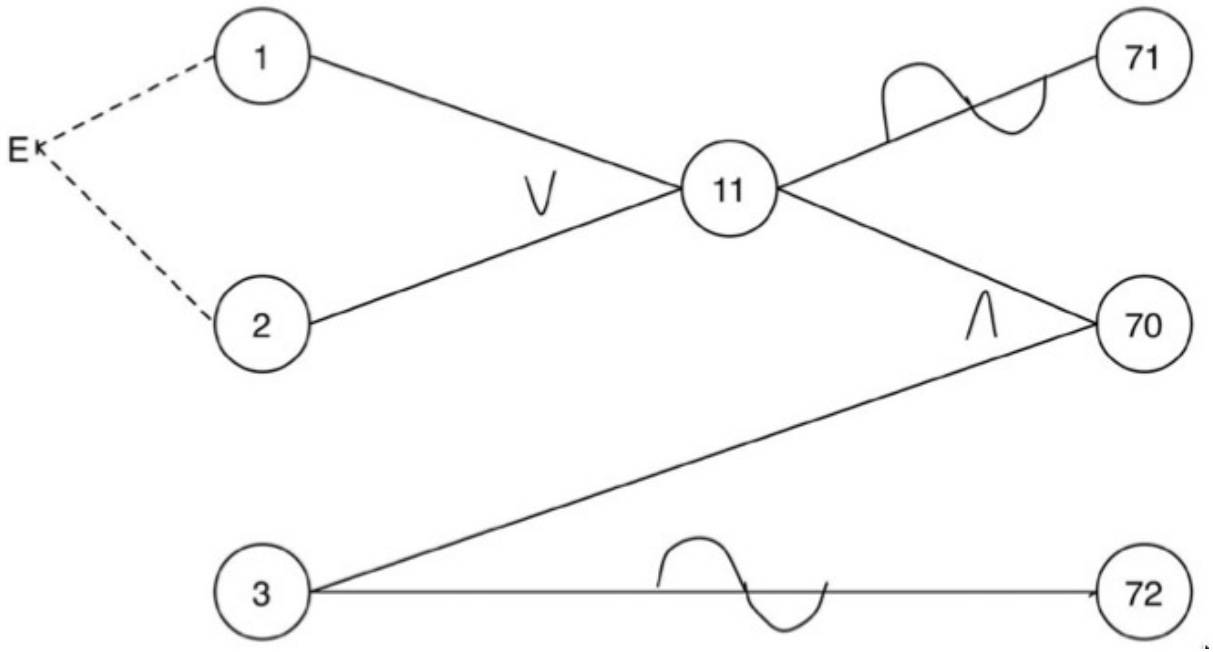


Grafo Causa-Efeito

Exempo: Tabela de Decisão

3. Criar uma coluna na tabela de decisão para cada combinação de causas

1	0	0
2	0	0
3	0	1
70		
71	1	1
72		

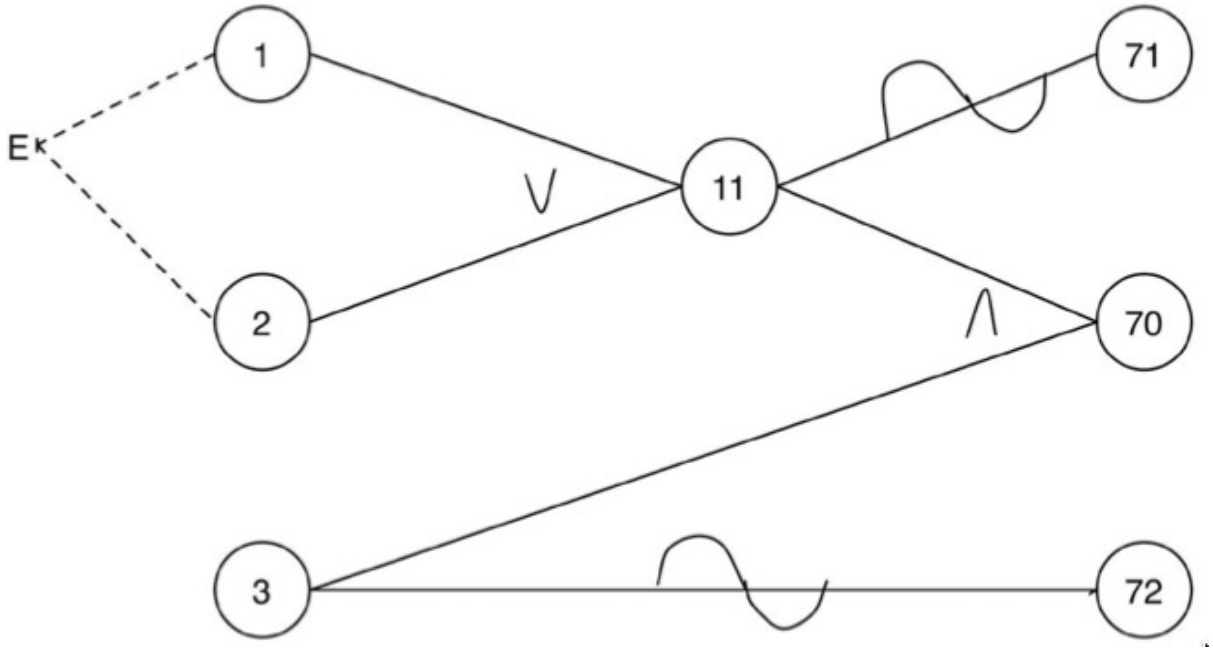


Grafo Causa-Efeito

Exempo: Tabela de Decisão

4. Determinar, para cada combinação, os estados de todos os outros efeitos, anotando na tabela

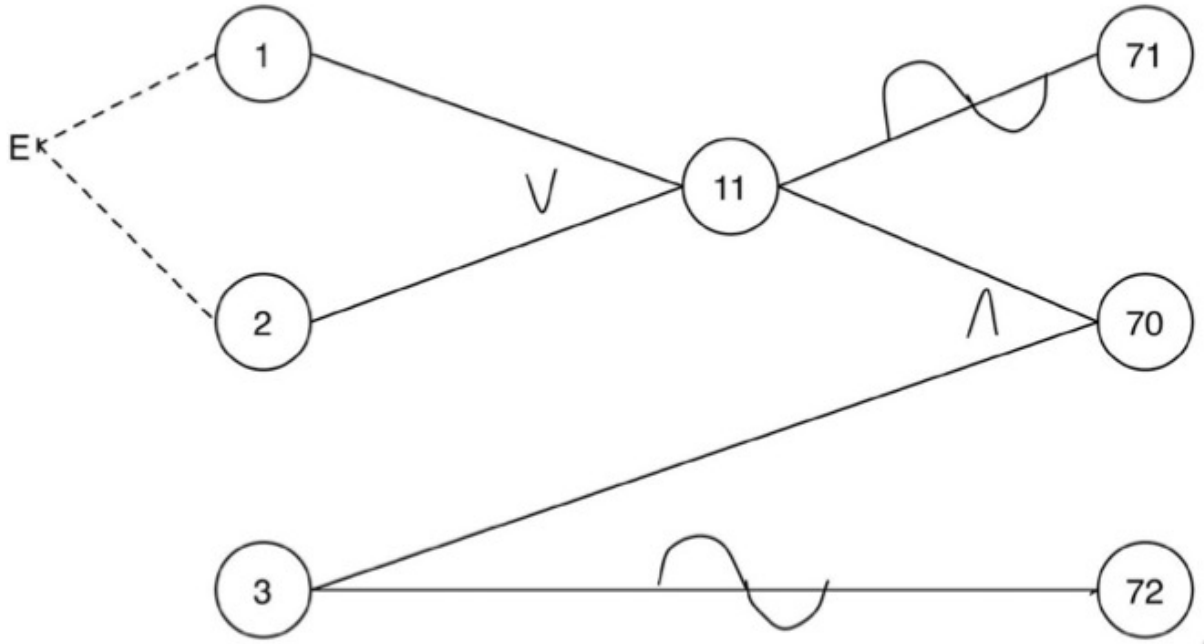
1	0	0
2	0	0
3	0	1
70	0	0
71	1	1
72	1	0



Grafo Causa-Efeito

Exempo: Tabela de Decisão

1	0	0	1	0	1
2	0	0	0	1	0
3	0	1	1	1	0
70	0	0	1	1	0
71	1	1	0	0	0
72	1	0	0	0	1



Grafo Causa-Efeito

- Geração dos Casos de Teste
  - Cada combinação de entradas deve gerar um caso de teste
  - Exemplo:

1	0	0	1	0	1
2	0	0	0	1	0
3	0	1	1	1	0
70	0	0	1	1	0
71	1	1	0	0	0
72	1	0	0	0	1

Causa 1	Causa 2	Causa 3	Efeito
Caracter coluna 1 (A)	Caracter coluna 1 (B)	Caracter coluna 2	Saída esperada
X	-	X	Mensagem X e Mensagem Y
X	-	5	Mensagem X
A	-	3	Atualiza
-	B	9	Atualiza
A	-	X	Mensagem Y

Error Guessing

- Abordagem ad-hoc, onde se supõe por intuição e experiência alguns tipos prováveis de erros, gerando a partir daí os casos de teste
- A ideia é enumerar possíveis erros e definir casos de teste para explorá-las
- Não há procedimento específico

Referências

- Delamaro, M. E., Maldonado, J. C., Jino, M., Introdução ao Teste de Software. Ed. Elsevier, 2007.
- Myers, G. J. et al, The Art of Software Testing. Ed. John Wiley & Sons, 2012 (capítulo 4).