INSTITUTO DE CIÊNCIAS EXATAS

DEPARTAMENTO DE COMPUTAÇÃO

Teste Funcional

Eduardo Kinder Almentero
ekalmentero@gmail.com

Agenda

1. Introdução ao teste funcional
2. Alguns critérios
    1. Particionamento de Equivalência
    2. Análise do Valor Limite
    3. Grafo de Causa-Efeito
    4. *Error-Guessing*
3. Conclusão

Introdução ao Teste Funcional

- O **teste exaustivo**, ou seja, **avaliar o software** utilizando **todos os elementos do domínio de entrada** é, na grande maioria das vezes, **impossível**.
- Devemos **dispor de recursos** que nos **permitam selecionar um subconjunto de entradas** do software com **grande probabilidade** de encontrar **erros**.
- Chamamos de **cobertura de teste** o conjunto de **casos de testes** elaborados com o **subconjunto de entradas** selecionado e **executados para avaliar o software**.
- Através da **cobertura de teste**, deve ser possível alcançar o **grau de confiabilidade** pretendido para o software.

Introdução ao Teste Funcional

- No teste funcional, os **casos de teste** são **elaborados** considerando o **software** a ser testado uma **caixa-preta.**
  - Os detalhes de **como o software** foi construído **não são considerados.**
- O software é **avaliado do ponto de vista do usuário.**
- **A avaliação baseia-se na especificação dos requisitos do software.**
- Através do **teste funcional** é possível **detectar todos os defeitos** do software
  - Para isto teríamos que realizar o **teste exaustivo**, isto é, realizar **todos os casos de teste possíveis**, o que não é viável.

Introdução ao Teste Funcional

- A **especificação do software** pode ser **diferente** (de propósito ou não) da **necessidade do usuário**.
- As falhas provenientes de uma **especificação incorreta** não serão **encontrados** através de **teste funcional**.
- Por outro lado, utilizar a **especificação** do software **facilita** a **elaboração** e **execução** dos casos de teste.

Critérios

- Chamaremos de **critérios** as **abordagens** que estudaremos para auxiliar na **seleção de um subconjunto de dados de entrada** para **realização de testes**.
- Os critérios que estudaremos são:
  - Particionamento de Equivalência;
  - Análise do Valor Limite;
  - Grafo Causa-Efeito;
  - Error-Guessing.

Particionamento de Equivalência

- O domínio de entrada é **dividido em classes de equivalência**
  - De forma que possamos assumir, com certa segurança, que **qualquer elemento** de uma **classe é um representante dela;**
  - **Todos elementos** da mesma classe devem apresentar **comportamento similar**, ou seja, se **um elemento detectar um defeito**, **todos os demais também deveriam**;
    - **Se um** elemento **não detectar**, **nenhum outro** da mesma classe **detectaria**.
  - **Reduz o domínio de entrada**, possibilitando o teste de todos seus elementos.
  - Alguns trabalhos sugerem a **análise** do **domínio de saída** para **detecção das classes de equivalência**.

Particionamento de Equivalência

- Para auxiliar na identificação, devemos **procurar na especificação dos requisitos** termos que expressam "**intervalos**" ou "**conjuntos**" que sejam **processados da mesma forma**.
- Uma **classe de equivalência** representa uma **quantidade de estados válidos** ou **inválidos** para as **condições de entrada**.

Particionamento de Equivalência

- Ex.: um programa **que calcule o novo salário dos funcionários** de uma empresa de acordo com a seguinte **regra de aumento**:
  - Se o salário for menor ou igual que R$ 1.500, aumento de 8%;
  - Se o salário for maior que R$ 1.500 e menor ou igual que R$ 5.000, aumento de 4%;
  - Se o salário for maior que R$ 5.000 e menor ou igual que R$ 10.000, aumento de 2%;
  - Nenhum funcionário da empresa deve ganhar mais que R$ 10.000.
- Quais possíveis classes de equivalência?

Particionamento de Equivalência

- Possíveis classes de equivalência:
  - salário ≤ 1.500 → aplicar 8%
  - 1.500 < salário ≤ 5.000 → aplicar 4%
  - 5.000 < salário ≤ 10.000 → aplicar 2%
  - salário > 10.000 → salário inválido
  - salário ≤ SALÁRIO-MÍNIMO → salário inválido
- Neste exemplo, selecionaríamos **um ou dois** dados de entrada quaisquer, **pertencentes a cada uma destes domínios**, para avaliar o funcionamento do software.

Particionamento de Equivalência

- Vantagem
  - **Intensidade de redução no tamanho do domínio** de entrada;
  - **Testes** podem ser feitos com **base na especificação**.
  - **Dependendo do tipo de entrada**, pode ser **fácil** identificar as **classes de equivalência**.
- Desvantagem
  - Quanto o processamento é complexo, pode ser **difícil identificar** classes de equivalência;
  - Embora a **especificação possa sugerir** que um conjunto de dados é processado da mesma forma, **na prática isto pode não ocorrer**;
  - As classes de equivalência possuem **certa subjetividade**, portanto, é aconselhável que a equipe de teste **defina as classes em conjunto**.

Análise do Valor Limite

- A experiência demonstra que **casos de teste que utilizam condições limite** apresentam **maior chance de detectar defeitos**.
- Uma **condição limite** é aquela que corresponde a **valores logo abaixo**, **exatamente iguais**, ou **logo acima daqueles descritos na especificação do software**.
  - Aqueles que são **utilizados** para **definição das classes de equivalência**.
- Este critério, assim como o particionamento de equivalência, pode ser utilizado **considerando o domínio de saída**.
  - É mais comum a observação do domínio de saída para a análise do valor limite.

Análise do valor limite

- É utilizado em **conjunto com o critério de classe de equivalência**, onde em **vez de adotar uma escolha aleatória**, se opta pelos **valores na fronteira das classes**.
  - Em vez de **selecionar** os **dados de forma** aleatória dentro de uma partição, são selecionados aqueles que **exploram os limites**.
- Ex.
  - salário ≤ 1.500 → aplicar 8%
    - Valor de fronteira -1, 0, 1, 1.498, 1.499, 1.500
  - 1.500 < salário ≤ 5.000 → aplicar 4%
    - Valor de fronteira 1.501, 1.502, 4.998, 4.999, 5.000
  - 5.000 < salário ≤ 10.000 → aplicar 2%
    - Valor de fronteira 5.001, 5.002, 9.999, 10.000
  - salário > 10.000 → salário inválido
    - 10.001, 10.002
  - Quando uma condição de entrada especifica texto, com até 255 caracteres, utilizar:
    - Em branco,
    - 1, 2 caracteres,
    - 254, 255, 256 caracteres.

Análise do Valor Limite

- Exemplo:
  - Segundo a **descrição do Gmail**, o **tamanho máximo** de um **arquivo** encaminhado em **anexo** a um e-mail deve ser de **25MB**
    - 1 Megabyte (MB) = 1.000.000 bytes (B)
      - 25 MB = 25 000 000 (B)
      - 24,999999 MB = 24 999 999 B
      - 25 000 001 B = 25,000001 B
    - Ex. de valores limites: 24,999998 24,999999 25 25,000001 25,000002

Análise do Valor Limite

- Vantagem
  - Similar a do particionamento de equivalência: **reduz drasticamente a quantidade de casos de teste**.
- Desvantagem
  - Não considera as **possíveis combinações** dos **valores de entrada.**

Teste Funcional Sistemático

- **Combinação dos critérios de Particionamento de Equivalência e do Valor Limite**
  - Valores numéricos
    - Valores discretos: testar todos os valores;
    - Intervalos: testar os extremos e um valor do interior do intervalo;
  - Tipo de valores diferentes e casos especiais
    - Espaço em branco (pode ser interpretado como zero em um campo numérico);
    - Zero deve ser sempre selecionado individualmente, mesmo que dentro do limite de um intervalo de valores;
    - Devem ser testados valores nos limites da representações binária dos dados. Ex.: em campos inteiros de 16bits, os valores -32.768 e +32.767 devem ser selecionados.

Teste Funcional Sistemático

- Números reais
  - São **mais problemáticos que valores inteiros**: são fornecidos na **entrada como números decimais**, para processamento, são **armazenados de forma binária** e, na **saída, são convertidos para decimais novamente**;
  - O **limite** para números reais **pode não ser exato**, mas ainda assim **deve ser incluído** nos casos de teste.
  - Incluir uma **margem de erro** que se ultrapassada o valor deve ser considerado diferente.
- Intervalos variáveis
  - O intervalo de uma variável depende do valor de outra variável.
    - Por exemplo, o valor da variável x pode variar de zero ao valor da variável y
    - Nesse caso, os seguintes dados de entrada devem ser definidos:
      - $x = y = 0$;
      - $x = 0 < y$;
      - $0 < x = y$;
      - $0 < x < y$.
    - Além disso, devem-se também selecionar os seguintes valores ilegais:
      - $y = 0 < x$;
      - $0 < y < x$;
      - $x < 0$; e
      - $y < 0$.

Teste Funcional Sistemático

- Arranjos
  - Devemos considerar o fato de o tamanho do arranjo ser variável, bem como de os dados serem variáveis.
  - Os elementos do arranjo devem ser testados como se fossem variáveis comuns, como mencionado nos itens anteriores.
  - O tamanho do arranjo deve ser testado com valores intermediários, com os valores mínimo e máximo.
  - O arranjo deve ser testado primeiro como uma única estrutura, depois como uma coleção de subestruturas, e cada subestrutura deve ser testada independentemente.
- Dados tipo texto
  - A entrada deve explorar comprimentos variáveis;
  - Os dados de entrada podem ser apenas alfabéticos, alfanuméricos e com caracteres especiais.
    - Todas estas situações devem ser exploradas.

Teste Funcional Sistemático

- Vantagens
  - Relaciona **diretrizes** para **facilitar a geração de casos de teste** e **enfatiza a seleção de mais de um caso de teste por partição e/ou limite**, aumentando a cobertura (probabilidade de revelar defeitos)
- Desvantagens
  - Apresenta os **mesmos problemas que os critérios em que é baseado**: Particionamento de Equivalência e Análise do Valor Limite
    - Não considera as **possíveis combinações** dos **valores de entrada.**

Grafo Causa-Efeito

- Uma das limitações das técnicas anteriores é que estas não realizam a **combinação dos dados de entrada**.
- O critério Grafo Causa-Efeito auxilia na **elaboração de um conjunto de casos de teste** que permite identificar **ambiguidades e incompletudes** nas especificações.
- Para criar os casos de teste utilizando este critério é preciso:
  1. Decompor a especificação do software em partes, para simplificar a construção do grafo;
  2. Identificar as causas e efeitos na especificação e atribuir um identificador único a cada um (número).
     - Analisar a especificação e ligar as causas aCausas são as entradas/ estímulos que provoquem resposta do software;
     - Efeitos são as saídas, mudança no estado ou qualquer alteração observável.
  3. os efeitos;
  4. Adicionar restrições ao grafo devido a combinações de causas e efeitos como consequência de restrições sintáticas ou de ambiente;
  5. Converter o grafo em uma tabela de decisão;
  6. Converter as colunas da tabela de decisão em casos de teste.

Grafo Causa-Efeito

- Ao executar o passo 2, realizar as seguintes observações:
  1. Quando o nó for do **tipo OR** e a **saída ter que ser 1**, **nunca atribuir mais de uma entrada com valor 1 simultaneamente**. O objetivo disso é evitar que alguns erros não sejam detectados pelo fato de uma causa mascarar outra.
  2. Quando o nó for do **tipo AND** e a **saída ter que ser 0**, **todas as combinações de entrada que levem à saída 0 devem ser enumeradas**. No entanto, se a situação é tal que uma entrada é 0 e uma ou mais das outras entradas é 1, não é necessário enumerar todas as condições em que as outras entradas sejam iguais a 1.
  3. Quando o nó for do **tipo AND e a saída ter que ser 0**, **somente uma condição em que todas as entradas sejam 0 precisa ser enumerada**. (Se esse AND estiver no meio do grafo, de forma que suas entradas estejam vindo de outros nós intermediários, pode ocorrer um número excessivamente grande de situações nas quais todas as entradas sejam 0.)

Grafo Causa-Efeito

- Notação
  - Considerando que **cada nó** do grafo pode assumir os **valores 0 ou 1**, que representam, respectivamente, **ausência ou presença de estado**, a notação é:

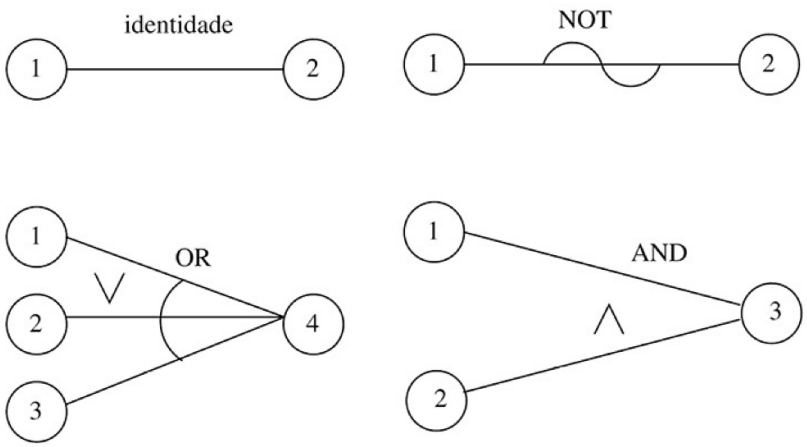


Figura retirada de
Delamaro, Marcio; Jino, Mario; Maldonado, Jose. Introdução ao Teste de Software (Kindle Location 1691). GEN LTC. Kindle Edition.

- **Função identidade:** se nó "1" é 1, então nó dois é 1; senão nó 2 é 0;
- **Função NOT:** se nó "1" é 1, então nó "2" é 0, senão nó "2" é 1;
- **Função OR:** se nó "1" ou "2" ou "3" é ', então nó "4" é ', senão nó "4" é 0;
- **Função AND:** se ambos os nós "1" e "2" forem ', então nó "3" é ', senão nó "3" é 0.

Grafo Causa-Efeito

- Exemplo: programa imprime mensagens[1]
  - Lê dois caracteres e imprime mensagens da seguinte forma:
    - O primeiro caractere deve ser um A ou B;
    - O segundo caractere deve ser um dígito;
    - Caso as condições acima sejam verdadeiras, o arquivo é atualizado.
    - Se o primeiro caractere é incorreto, envia mensagem X
    - Se o segundo caractere é incorreto, envia a mensagem Y

[1]J. Myers et al. The Art of Software Testing. 2a ed. New York, NY, USA: John Wiley & Sons, 2004.

Grafo Causa-Efeito

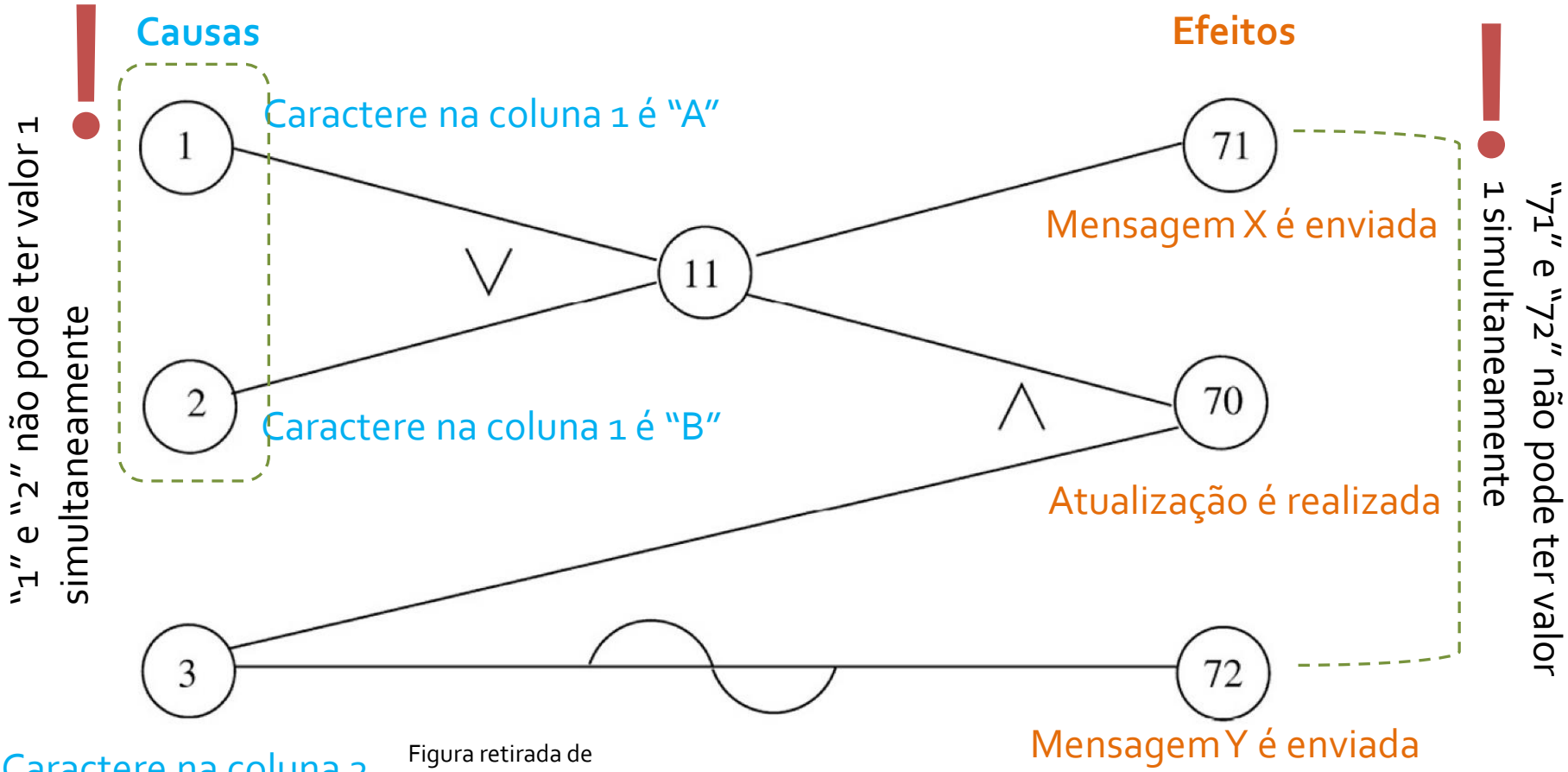
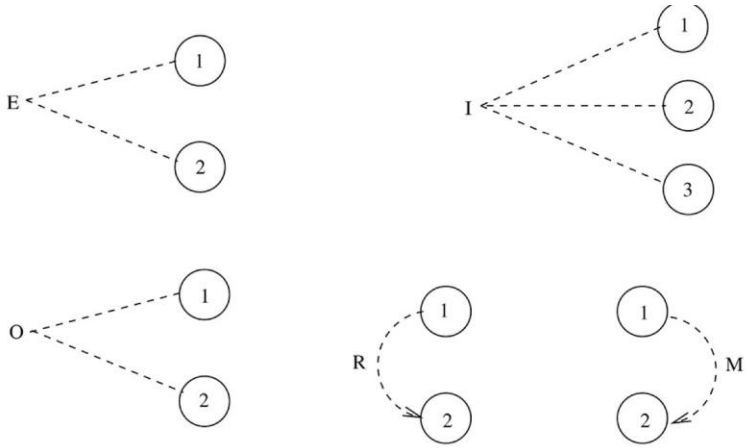


Figura retirada de
Delamaro, Marcio; Jino, Mario; Maldonado, Jose. Introdução ao Teste de Software (Kindle Location 1691). GEN LTC. Kindle Edition.

Grafo Causa-Efeito

- Notações de restrições



- **Restrição E:** no máximo um entre "1" e "2" pode ser igual a 1;
- **Restrição I:** no mínimo um entre "1" e "2" deve ser igual a 1;
- **Restrição O:** um e somente um entre "1" e "2" deve ser igual a 1.
- **Restrição R:** para que "1" seja igual a 1, "2" deve ser igual a 1;
- **Restrição M:** se o efeito "1" é 1 o efeito "2" é forçado a ser 0.

Figura retirada de
Delamaro, Marcio; Jino, Mario; Maldonado, Jose. Introdução ao Teste de Software (Kindle Location 1691). GEN LTC. Kindle Edition.

Grafo Causa-Efeito

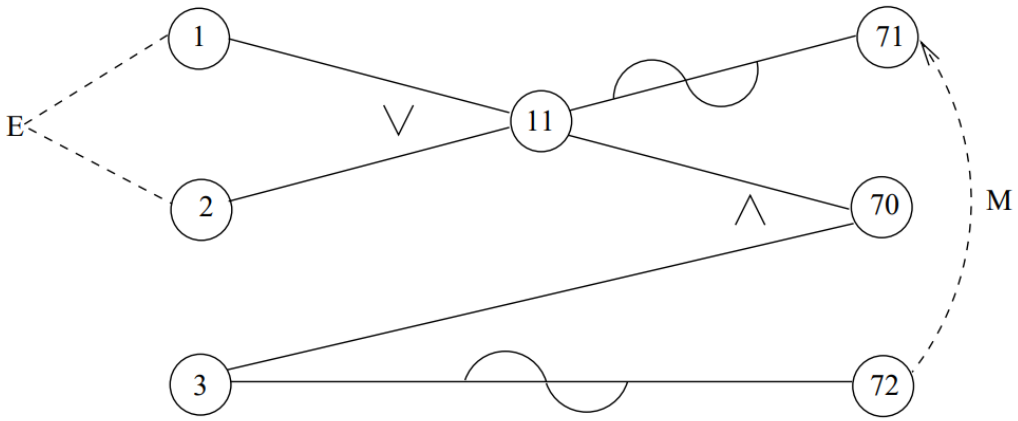
Colunas darão origem aos casos de teste

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 com restrições



Grafo Causa-Efeito

- Casos de teste

1. {(Col1=D, Col2=H), (msg=X)}
2. {(Col1=D, Col2=5), (msg=X)}
3. {(Col1=A, Col2=5), (msg=Atual. Real.)}
4. {(Col1=B, Col2=7), (msg=Atual. Real.)}
5. {(Col1=A, Col2=F), (msg=Y)}

Casos de teste: 1 2 3 4 5

1	0	0	1	0	1
2	0	0	0	1	0
3	0	1	1	1	0

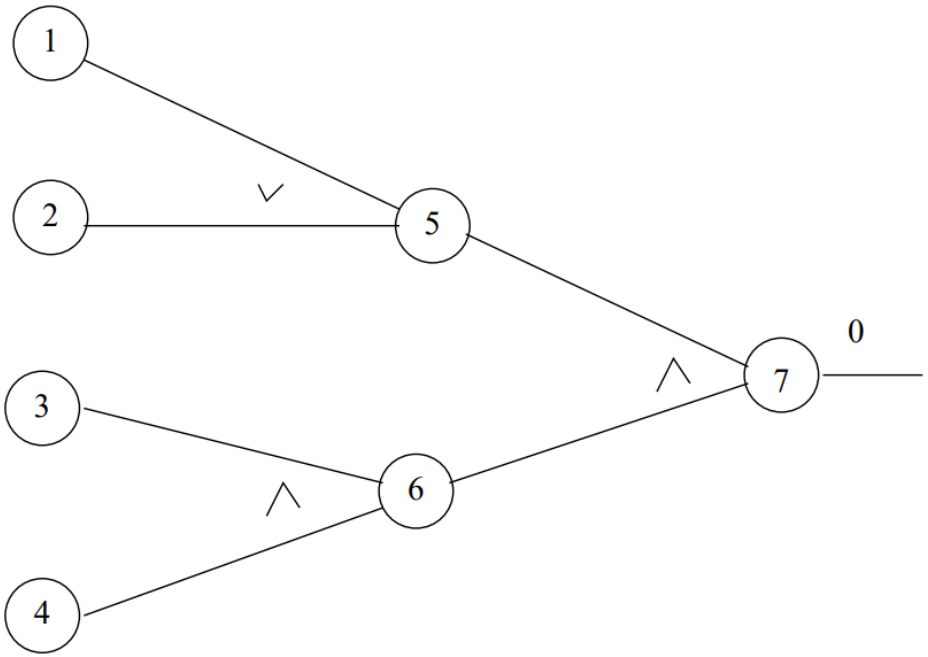
70	0	0	1	1	0
71	1	1	0	0	0
72	-	0	0	0	1

1. Caractere na coluna 1 é A
2. Caractere na coluna 1 é B
3. Caractere na coluna 2 é um dígito

70. Atualização realizada
71. Mensagem X enviada
72. Mensagem Y enviada

Grafo Causa-Efeito

- Exercício:



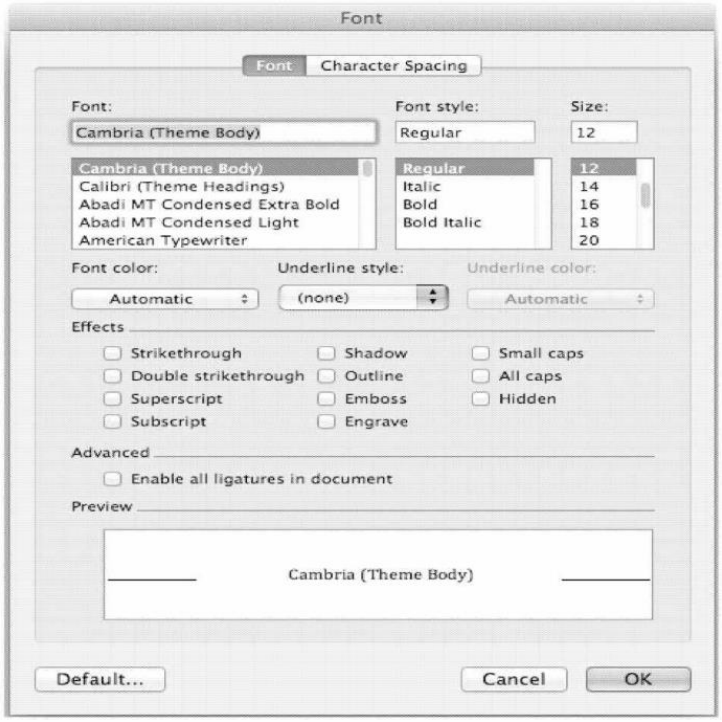
Grafo Causa-Efeito

- Vantagens
  - **Exercita a combinação de dados de teste**, que muitas vezes não seriam realizadas.
  - Os **resultados esperados são parte do processo de criação do caso de teste**, isto é, estão presentes na tabela de decisão.
- Desvantagem
  - **Complexidade em se desenvolver o grafo booleano**.
    - Grafos grandes são mais complexos;
    - Comportamentos complexos do software dão origem a muitos grafos;
  - Converter o **grafo em uma tabela de decisão** também é uma **atividade trabalhosa**.

Teste combinatorial

- Como o próprio nome informa, considera as **combinações de dados de entrada**, com o **intuito de testar diversas possibilidades** de **combinação;**
- Pode ser utilizado, por exemplo, quando os **dados de entrada de teste** devem ser **fornecidos através de uma interface** com o **usuário**.

Teste combinatorial



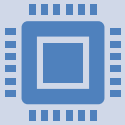
- Em efeitos (*effects*) são 11 caixas que possuem 2 estados: selecionada ou não
  - Número de possibilidades $2^{11} = 2048$
- Outras possibilidades
  - *Font*: 327
  - *Font style*: 4
  - *Size*: 16
  - *Font* color: 50
  - *Underline style*: 17
  - *Underline color*: 50
  - *Advanced*: 2
- O número total de possibilidades e dado pelo **produto da quantidade de itens**, que totaliza mais de 3 trilhões de possibilidades

Exemplo retirado de Delamaro, Marcio; Jino, Mario; Maldonado, Jose. Introdução ao Teste de Software (Kindle Locations 1866-1867). GEN LTC. Kindle Edition.

Error guessing (adivinhação de erro)

- Baseada no uso de **intuição** e **experiência** para elaboração de casos de teste;
- A abordagem por trás deste critério é:
  - Listar **possíveis erros**, ou **situações que possam gerar falhas**;
  - Definir **casos de teste para explorar estas situações** e encontrar uma falha.
- Ex.: explorar **características de arredondamento** em software que fazem **cálculos científicos**.

Conclusão



Uma **vantagem do teste funcional** é que ele pode ser **aplicado a qualquer software,** pois não **depende da estrutura interna** e das **tecnologias utilizadas no desenvolvimento, apenas da especificação;**

Entretanto, como os critérios se **baseiam apenas na especificação,** não é possível **garantir que partes críticas** e **essenciais** do software tenham **sido cobertas**.

É fundamental que as **técnicas de testes sejam vistas como complementares** e **utilizadas em conjunto,** para que o software seja **avaliado** a partir de **pontos de vista distintos**.

INSTITUTO DE CIÊNCIAS EXATAS

DEPARTAMENTO DE COMPUTAÇÃO

Perguntas?