

SSC0721 – Teste e inspeção de software

Teste Funcional

Prof. Marcio E. Delamaro

`delamaro@icmc.usp.br`

Relembrando

- O que diferencia as técnicas de teste é a forma de particionar o domínio
- Requisitos de teste definem os critérios de divisão do domínio
- No caso de teste funcional, apenas a especificação é utilizada
- Quando o testador seleciona casos de teste dos subconjuntos (partições ou subdomínios), o conjunto de testes resultante é uma boa representação de todo o domínio

Características

- Conjunto de teste é pequeno
- Número de subdomínios é pequeno
- Uma vez definidas as classes de equivalência, pode-se assumir, com alguma segurança, que qualquer elemento da classe pode ser considerado (???)
- Se um elemento detectar um defeito, qualquer outro também detecta; se não detectar, os outros também não detectam
- Particionar o domínio de saída também é válido

Histórico

- Década de 70
- Métodos para especificar sistemas como Análise e Projeto Estruturados
- Eram mencionados, embora não diretamente, aspectos de validação do sistema com relação à satisfação dos seus requisitos funcionais

Critérios

- Diversos critérios são definidos dentro desta técnica
- Particionamento de Equivalência, Análise do Valor Limite, Grafo Causa-Efeito e Error-Guessing. Além desses, também existem outros, como, por exemplo, Teste Funcional Sistemático, Syntax Testing, State Transition Testing e Graph Matrix

Vantagens

- Pode ser utilizado em todas as fases de teste.
- Independente do paradigma de programação utilizado.
- Eficaz em detectar determinados tipos de erros.
- Funcionalidade ausente, por exemplo.

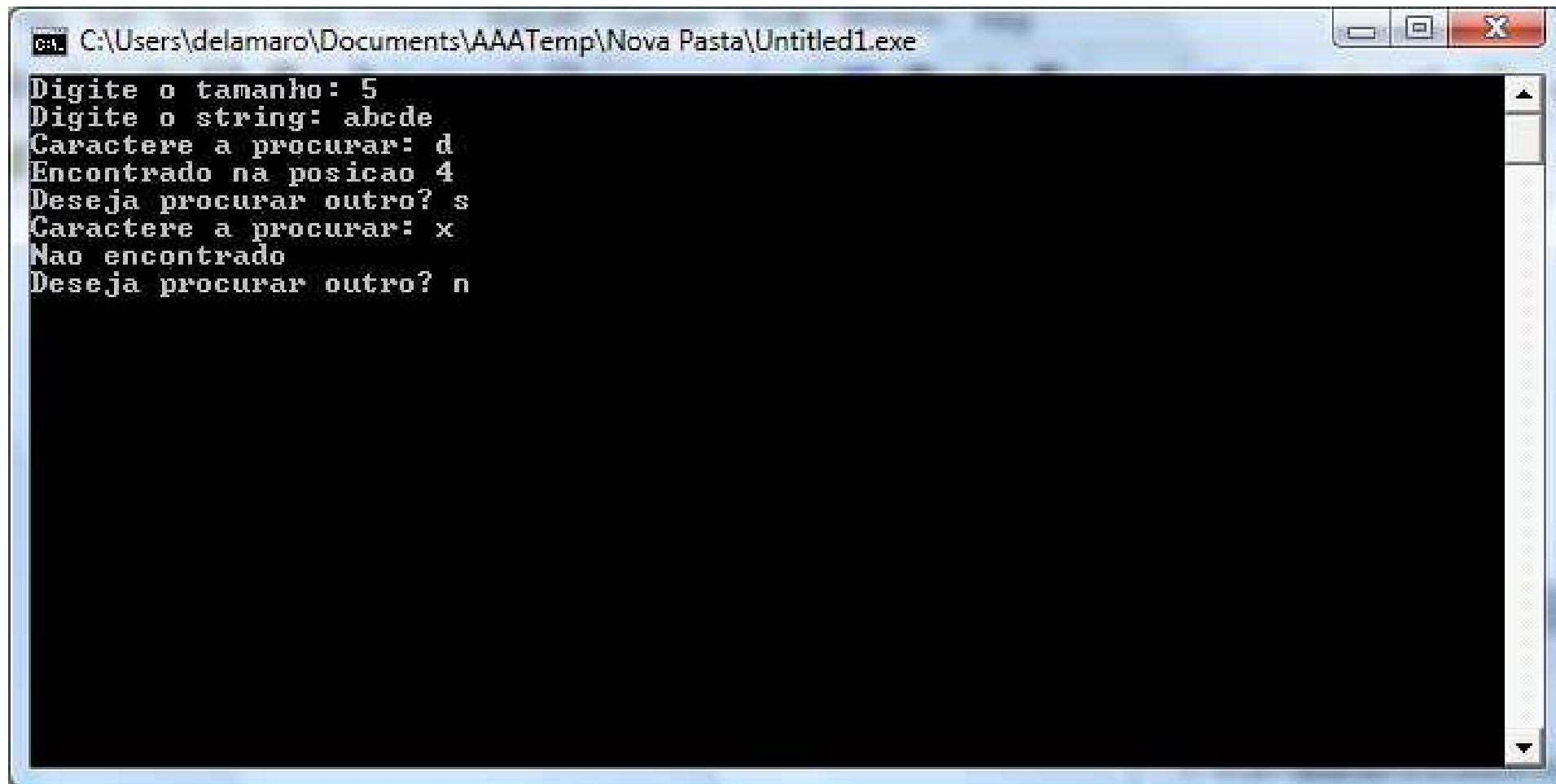
Desvantagens

- Dependente de uma boa especificação o que, em geral, não é bem feito.
- Não é possível garantir que partes essenciais ou críticas do software sejam executadas
- Ruim quando se tem entradas simples mas processamento complexo

Cadeia de caracteres

O programa solicita do usuário um inteiro positivo no intervalo entre 1 e 20 e então lê uma cadeia de caracteres desse comprimento. Após isso, o programa solicita um caractere e retorna a posição na cadeia em que o caractere é encontrado pela primeira vez ou uma mensagem indicando que o caractere não está presente na cadeia. O usuário tem a opção de procurar vários caracteres

Cadeia de caracteres



A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\delamaro\Documents\AAATemp\Nova Pasta\Untitled1.exe. The window has standard Windows window controls (minimize, maximize, close) on the right. The command prompt area is black with white text. The text shows a program that asks for a string size, a string, a character to search for, and whether to continue searching. The user has entered '5', 'abcde', 'd', 's', 'x', and 'n' respectively.

```
C:\Users\delamaro\Documents\AAATemp\Nova Pasta\Untitled1.exe
Digite o tamanho: 5
Digite o string: abcde
Caractere a procurar: d
Encontrado na posicao 4
Deseja procurar outro? s
Caractere a procurar: x
Nao encontrado
Deseja procurar outro? n
```

Particionamento em classes de equivalência

- Duas etapas
- Identificar classes de equivalência
- Definir os casos de teste para cobrir essas classes

Identificar classes de equivalência

- Identificar condições de entrada relevantes
- Particionar cada uma em dois ou mais grupos
- Para ajudar na identificação das partições, pode-se observar a especificação procurando termos como “intervalo” e “conjunto” ou palavras similares que indiquem que os dados são processados da mesma forma

Classes de equivalência

| Condição de entrada | Classes de equivalência válidas | Classes de equivalência inválidas |
|---------------------|---------------------------------|-----------------------------------|
| | | |

Identificando as entradas

- Tamanho da cadeia (**T**)

Identificando as entradas

- Tamanho da cadeia (**T**)
- A cadeia de caracteres. Pode ser qualquer, portanto somente seu tamanho é importante (**CC**)

Identificando as entradas

- Tamanho da cadeia (**T**)
- A cadeia de caracteres. Pode ser qualquer, portanto somente seu tamanho é importante (**CC**)
- O caractere a ser procurado (**C**)

Identificando as entradas

- Tamanho da cadeia (**T**)
- A cadeia de caracteres. Pode ser qualquer, portanto somente seu tamanho é importante (**CC**)
- O caractere a ser procurado (**C**)
- A opção por procurar mais caracteres (**O**)

Identificar as classes

- Algumas guidelines podem ser estabelecidas
- Uma condição de entrada estabelece um intervalo de valores
- *“O número de ítems pode variar de 1 a 999”*
- Uma classe válida
- $1 \leq \text{número de ítems} \leq 999$
- Duas classes não válidas
- $1 > \text{número de ítems}$
- $999 < \text{número de ítems}$

Guidelines

- A condição de entrada estabelece uma quantidade de valores
- *“Para um automóvel, de 1 a seis proprietários podem ser relacionados”*
- Uma classe válida
- De um a seis proprietários
- Duas classes inválidas
- Nenhum proprietário
- Mais do que seis proprietários

Guidelines

- A condição de entrada especifica um conjunto de valores que (acredita-se) devem ser tratados de maneiras diversas.
- *“Tipo de veículo deve ser: ônibus, caminhão, taxi, passageiro ou motocicleta”*
- Uma classe válida para cada valor
- Uma classe não válida, por exemplo: trailer

Guidelines

- Condição de entrada determina uma condição do tipo “tem que ser”
- “Primeira letra do identificador tem que ser uma letra”
- Uma classe válida
- Primeiro caractere é uma letra
- Classe não válida
- Caractere é um dígito

Guidelines

- Se existe alguma razão para acreditar que o programa não trata elementos de uma classe de maneira uniforme, a classe deve ser dividida em classes menores

Cadeia: T

- Tamanho da cadeia de caracteres

Cadeia: T

- Tamanho da cadeia de caracteres
- Se enquadra na primeira guideline: valores num intervalo

Cadeia: T

- Tamanho da cadeia de caracteres
- Se enquadra na primeira guideline: valores num intervalo
- Uma classe válida: de 1 a 20

Cadeia: T

- Tamanho da cadeia de caracteres
- Se enquadra na primeira guideline: valores num intervalo
- Uma classe válida: de 1 a 20
- Duas não válidas: menor que 1, maior que 20

Cadeia: CC

- Valor da cadeia de caracteres

Cadeia: CC

- Valor da cadeia de caracteres
- O valor em si não determina comportamentos diferentes do programa, apenas o seu tamanho

Cadeia: CC

- Valor da cadeia de caracteres
- O valor em si não determina comportamentos diferentes do programa, apenas o seu tamanho
- abcde

Cadeia: CC

- Valor da cadeia de caracteres
- O valor em si não determina comportamentos diferentes do programa, apenas o seu tamanho
- abcde
- 8y4e*

Cadeia: CC

- Valor da cadeia de caracteres
- O valor em si não determina comportamentos diferentes do programa, apenas o seu tamanho
- abcde
- 8y4e*
- Assim, não é necessário usar essa variável como condição de entrada

Cadeia: O

- Opção de continuar ou não

Cadeia: O

- Opção de continuar ou não
- Se enquadra na terceira guideline: conjunto de valores possíveis

Cadeia: O

- Opção de continuar ou não
- Se enquadra na terceira guideline: conjunto de valores possíveis
- Duas classes válidas: **s** ou **n**

Cadeia: O

- Opção de continuar ou não
- Se enquadra na terceira guideline: conjunto de valores possíveis
- Duas classes válidas: **s** ou **n**
- Uma classe não válida: **w**, por exemplo

Cadeia: C

- Caractere a ser procurado

Cadeia: C

- Caractere a ser procurado
- Se enquadra também na terceira guideline

Cadeia: C

- Caractere a ser procurado
- Se enquadra também na terceira guideline
- Duas classes válidas: C pertence à string e C não pertence à string

Cadeia: C

- Caractere a ser procurado
- Se enquadra também na terceira guideline
- Duas classes válidas: C pertence à string e C não pertence à string
- Nenhuma classe não válida

Cadeia: classes de equivalência

| Variável de entrada | Classes de equivalência válidas | Classes de equivalência inválidas |
|---------------------|----------------------------------|-----------------------------------|
| T | $1 \leq T \leq 20$ (1) | $T < 1$ (2) e $T > 20$ (3) |
| O | S (4) N (5) | Outro (6) |
| C | Pertence (7) Não pertence (8) | |

Computar casos de teste

- Próximo passo é usar as classes para computar os casos de teste necessários
- Regras a seguir:
- Atribuir um número para cada classe
- Definir casos de teste para cobrir o maior número de classes válidas possíveis, até que todas as classes válidas sejam cobertas
- Para cada classe não válida, desenhar um caso de teste específico

Comentário

- Entradas inválidas podem mascarar ou extrapolar outras entradas inválidas
- Por exemplo: **T** = 34 e **O** = w provavelmente não irá exercitar o classe inválida para a condição **O**

Cadeia: casos de teste

• $T = 3 \quad CC = abc \quad C = c \quad O = s \quad C = k \quad O = n$

Cadeia: casos de teste

- $T = 3$ $CC = abc$ $C = c$ $O = s$ $C = k$ $O = n$
- Saída: Encontrado na posição 3; Não encontrado

Cadeia: casos de teste

- $T = 3$ $CC = abc$ $C = c$ $O = s$ $C = k$ $O = n$
- Saída: Encontrado na posição 3; Não encontrado
- Classes cobertas: 1, 4, 5, 7, 8 (todas as válidas)

Cadeia: casos de teste

- $T = 3$ $CC = abc$ $C = c$ $O = s$ $C = k$ $O = n$
- Saída: Encontrado na posição 3; Não encontrado
- Classes cobertas: 1, 4, 5, 7, 8 (todas as válidas)
- $T = -3$

Cadeia: casos de teste

- $T = 3$ $CC = abc$ $C = c$ $O = s$ $C = k$ $O = n$
- Saída: Encontrado na posição 3; Não encontrado
- Classes cobertas: 1, 4, 5, 7, 8 (todas as válidas)
- $T = -3$
- Saída: Tamanho inválido

Cadeia: casos de teste

- $T = 3$ $CC = abc$ $C = c$ $O = s$ $C = k$ $O = n$
- Saída: Encontrado na posição 3; Não encontrado
- Classes cobertas: 1, 4, 5, 7, 8 (todas as válidas)
- $T = -3$
- Saída: Tamanho inválido
- Classe coberta: 2

Cadeia: casos de teste

- $T = 34$

Cadeia: casos de teste

- $T = 34$
- Saída: Tamanho inválido

Cadeia: casos de teste

- $T = 34$
- Saída: Tamanho inválido
- Classe coberta: 3

Cadeia: casos de teste

- $T = 34$
- Saída: Tamanho inválido
- Classe coberta: 3
- $T = 3$ $CC = abc$ $C = c$ $O = w$

Cadeia: casos de teste

- $T = 34$
- Saída: Tamanho inválido
- Classe coberta: 3
- $T = 3$ $CC = abc$ $C = c$ $O = w$
- Saída: Responda com s ou n

Cadeia: casos de teste

- $T = 34$
- Saída: Tamanho inválido
- Classe coberta: 3
- $T = 3$ $CC = abc$ $C = c$ $O = w$
- Saída: Responda com s ou n
- Classe coberta: 6

Exercício

- Poderíamos pensar numa condição de entrada como: “Número de vezes que um caractere é procurado no string”
- Nesse caso, como ficaria a tabela de classes? Modifique-a.
- Defina os casos de teste para cobrir essas novas classes.

Análise de valor limite

- É mais proveitoso explorar condições limites
- Aquelas que estão sobre, acima e abaixo dos limites das classes de equivalência
- Assim, em vez de selecionar um caso de teste qualquer, deve-se tomar um ou mais casos de teste de modo que cada limitante da classe seja testado
- Além disso, deve-se considerar também o domínio de saída para derivar casos de teste (classes de equivalência de saída)

Guidelines

- Dizem que é difícil definir um “livro de receitas”

Guidelines

- Dizem que é difícil definir um “livro de receitas”
- Ainda assim, algumas dicas podem ser observadas

Guidelines

- Dizem que é difícil definir um “livro de receitas”
- Ainda assim, algumas dicas podem ser observadas
- Se uma condição de entrada define um intervalo de valores: casos de teste nos limites do intervalo e logo além dos limites (casos não válidos)

Guidelines

- Dizem que é difícil definir um “livro de receitas”
- Ainda assim, algumas dicas podem ser observadas
- Se uma condição de entrada define um intervalo de valores: casos de teste nos limites do intervalo e logo além dos limites (casos não válidos)
- Por exemplo se o intervalo for -1.00 a 1.00, devemos tomar -1.00, 1.00, -1.01 e 1.01

Guidelines

- Dizem que é difícil definir um “livro de receitas”
- Ainda assim, algumas dicas podem ser observadas
- Se uma condição de entrada define um intervalo de valores: casos de teste nos limites do intervalo e logo além dos limites (casos não válidos)
- Por exemplo se o intervalo for -1.00 a 1.00, devemos tomar -1.00, 1.00, -1.01 e 1.01
- Se especifica uma quantidade de valores, escolher casos de teste nos limites, uma unidade acima e uma abaixo

Guidelines

- Dizem que é difícil definir um “livro de receitas”
- Ainda assim, algumas dicas podem ser observadas
- Se uma condição de entrada define um intervalo de valores: casos de teste nos limites do intervalo e logo além dos limites (casos não válidos)
- Por exemplo se o intervalo for -1.00 a 1.00, devemos tomar -1.00, 1.00, -1.01 e 1.01
- Se especifica uma quantidade de valores, escolher casos de teste nos limites, uma unidade acima e uma abaixo
- Por exemplo se um arquivo de entrada deve conter de 1 a 255 registros, casos de teste devem contemplar 0, 1, 255 e 256 registros

Guidelines

- Aplicar a 1a. regra para o domínio de saída

Guidelines

- Aplicar a 1a. regra para o domínio de saída
- Por exemplo, um programa deve calcular a dedução que uma empresa tem sobre um determinado imposto

Guidelines

- Aplicar a 1a. regra para o domínio de saída
- Por exemplo, um programa deve calcular a dedução que uma empresa tem sobre um determinado imposto
- Diz a especificação que essa dedução é de, no mínimo, \$0,00 e no máximo \$1.165,25

Guidelines

- Aplicar a 1a. regra para o domínio de saída
- Por exemplo, um programa deve calcular a dedução que uma empresa tem sobre um determinado imposto
- Diz a especificação que essa dedução é de, no mínimo, \$0,00 e no máximo \$1.165,25
- Aplicar entrada que cause \$0,00 de dedução e outra que cause \$1.165,25

Guidelines

- Aplicar a 1a. regra para o domínio de saída
- Por exemplo, um programa deve calcular a dedução que uma empresa tem sobre um determinado imposto
- Diz a especificação que essa dedução é de, no mínimo, \$0,00 e no máximo \$1.165,25
- Aplicar entrada que cause \$0,00 de dedução e outra que cause \$1.165,25
- Tente achar casos de teste que cause dedução negativa ou maior que \$1.165,25

Guidelines

- Aplicar a 1a. regra para o domínio de saída
- Por exemplo, um programa deve calcular a dedução que uma empresa tem sobre um determinado imposto
- Diz a especificação que essa dedução é de, no mínimo, \$0,00 e no máximo \$1.165,25
- Aplicar entrada que cause \$0,00 de dedução e outra que cause \$1.165,25
- Tente achar casos de teste que cause dedução negativa ou maior que \$1.165,25
- Nem sempre é possível achar casos de teste além dos limites

Guidelines

- Aplicar 2a. regra para saídas

Guidelines

- Aplicar 2a. regra para saídas
- Por exemplo, um sistema de recuperação de informação mostra no máximo 4 abstracts para cada consulta

Guidelines

- Aplicar 2a. regra para saídas
- Por exemplo, um sistema de recuperação de informação mostra no máximo 4 abstracts para cada consulta
- Achar casos de teste que façam com que 0, 1 e 4 abstracts sejam mostrados

Guidelines

- Aplicar 2a. regra para saídas
- Por exemplo, um sistema de recuperação de informação mostra no máximo 4 abstracts para cada consulta
- Achar casos de teste que façam com que 0, 1 e 4 abstracts sejam mostrados
- Tentar caso de teste que erroneamente mostre 5 registros

Guidelines

- Se a entrada ou a saída for um conjunto ordenado (um arquivo sequencial ou uma lista ligada), dar atenção ao primeiro e último elemento

Guidelines

- Se a entrada ou a saída for um conjunto ordenado (um arquivo sequencial ou uma lista ligada), dar atenção ao primeiro e último elemento
- Use a sua criatividade para definir outras condições limites

Cadeia: classes

| Variável de entrada | Classes de equivalência válidas | Classes de equivalência inválidas |
|---------------------|----------------------------------|-----------------------------------|
| T | $1 \leq T \leq 20$ (1) | $T < 1$ (2) e $T > 20$ (3) |
| O | S (4) N (5) | Outro (6) |
| C | Pertence (7) Não pertence (8) | |

Cadeia: classes

| Variável de E/S | Classes de equivalência válidas | Classes de equivalência inválidas |
|-----------------|----------------------------------|-----------------------------------|
| T | $1 \leq T \leq 20$ (1) | $T < 1$ (2) e $T > 20$ (3) |
| O | S (4) N (5) | Outro (6) |
| C | Pertence (7) Não pertence (8) | |
| Posição | $1 \leq pos \leq 20$ (9) | $pos < 1$ (10) $pos > 20$ (11) |

Cadeia: casos de teste

• **T = 1 CC = a C = c O = s C = k O = n**

Cadeia: casos de teste

- $T = 1 \quad CC = a \quad C = c \quad O = s \quad C = k \quad O = n$
- Saída: Encontrado na posição 1; Não encontrado

Cadeia: casos de teste

- **T = 1 CC = a C = c O = s C = k O = n**
- Saída: Encontrado na posição 1; Não encontrado
- **T = 20 CC = abcdefghijklmnopqrst C = t O = n**

Cadeia: casos de teste

- **T = 1 CC = a C = c O = s C = k O = n**
- Saída: Encontrado na posição 1; Não encontrado
- **T = 20 CC = abcdefghijklmnopqrst C = t O = n**
- Saída: Encontrado na posição 20

Cadeia: casos de teste

- **T = 1 CC = a C = c O = s C = k O = n**
- Saída: Encontrado na posição 1; Não encontrado
- **T = 20 CC = abcdefghijklmnopqrst C = t O = n**
- Saída: Encontrado na posição 20
- **T = 0**

Cadeia: casos de teste

- **T = 1 CC = a C = c O = s C = k O = n**
- Saída: Encontrado na posição 1; Não encontrado
- **T = 20 CC = abcdefghijklmnopqrst C = t O = n**
- Saída: Encontrado na posição 20
- **T = 0**
- Saída: Tamanho inválido

Cadeia: casos de teste

- $T = 21$

Cadeia: casos de teste

- $T = 21$
- Saída: Tamanho inválido

Cadeia: casos de teste

- $T = 21$
- Saída: Tamanho inválido
- $T = 3$ $CC = abc$ $C = c$ $O = w$

Cadeia: casos de teste

- $T = 21$
- Saída: Tamanho inválido
- $T = 3$ $CC = abc$ $C = c$ $O = w$
- Saída: Responda com s ou n

Grafo causa-efeito

- Uma das limitações dos critérios anteriores é que eles não exploram combinações dos dados de entrada
- O critério causa-efeito procura suprir essa deficiência
- O grafo é uma linguagem formal na qual a especificação é traduzida

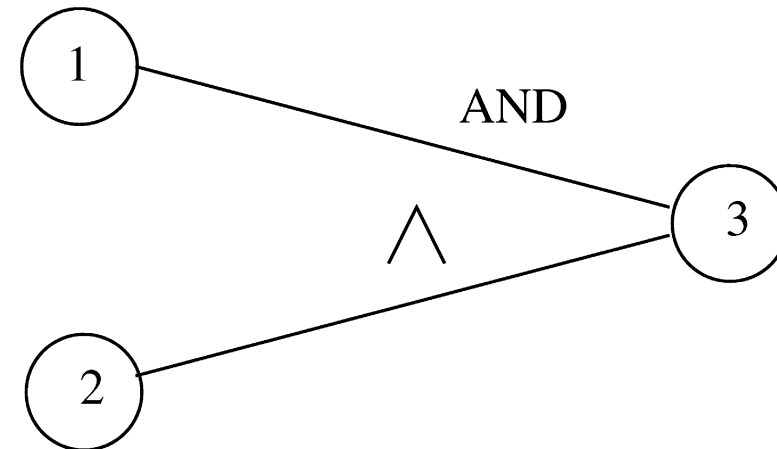
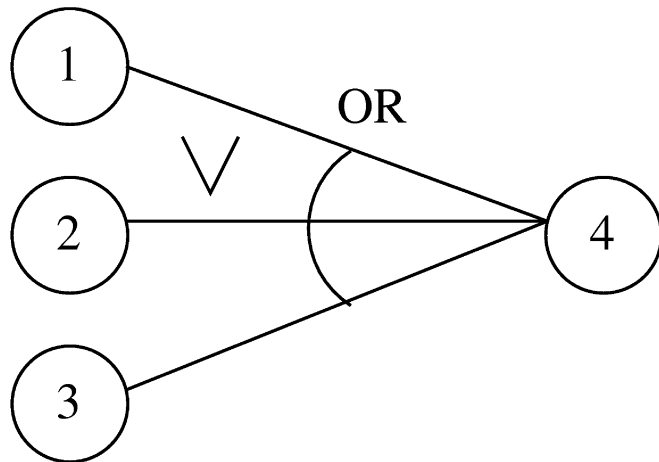
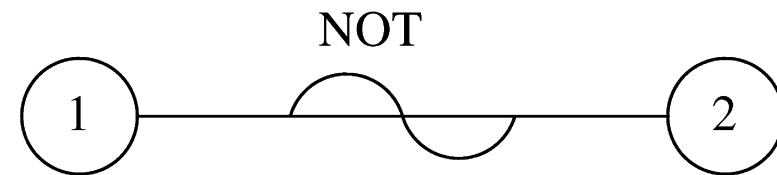
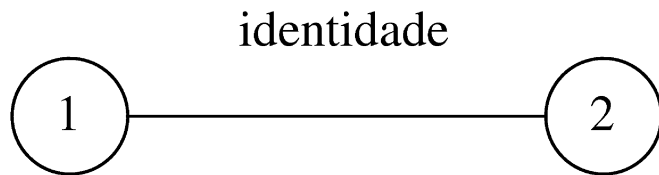
- Dividir a especificação do software em partes, pois a construção do grafo para grandes especificações torna-se bastante complexa.
- Identificar as causas e efeitos na especificação. As causas correspondem às condições de entrada, ou uma particular classe de equivalência, estímulos, ou qualquer coisa que provoque uma resposta do sistema em teste e os efeitos correspondem às saídas, mudanças no estado do sistema ou qualquer resposta observável. Uma vez identificados, a cada um deve ser atribuído um único número.

- Analisar a semântica da especificação e transformar em um grafo booleano – o Grafo Causa-Efeito – que liga as causas e os efeitos.
- Adicionar anotações ao grafo, as quais descrevem combinações das causas e efeitos que são impossíveis devido a restrições sintáticas ou do ambiente.
- Converter o grafo em uma tabela de decisão, na qual cada coluna representa um caso de teste.
- Converter as colunas da tabela de decisão em casos de teste.

Notação

- Cada nó tem o valor 0 ou 1
- Valor 1 indica que aquele determinado estado existe (é verdadeiro)
- Valor 0 indica que estado não é verdadeiro
- Função identidade: se nó “1” é 1, então nó “2” 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 1, então nó “4” é 1; senão nó “4” é 0.
- Função and: se ambos nós “1” e “2” são 1, então nó “3” é 1; senão nó “3” é 0.

Notação



Um exemplo

- “Imprime Mensagens”: o programa lê dois caracteres e, de acordo com eles, mensagens serão impressas
- O primeiro caractere deve ser um “A” ou um “B”
- O segundo caractere deve ser um dígito.
- Nessa situação, o arquivo deve ser atualizado
- Se o primeiro caractere é incorreto, enviar a mensagem X.
- Se o segundo caractere é incorreto, enviar a mensagem Y

1 - caractere na coluna 1 é “A”

Causas

- 1 - caractere na coluna 1 é “A”
- 2 - caractere na coluna 1 é “B”

Causas

- 1 - caractere na coluna 1 é “A”
- 2 - caractere na coluna 1 é “B”
- 3 - caractere na coluna 2 é um dígito

70 - a atualização é realizada

70 - a atualização é realizada

71 - a mensagem X é enviada

- 70 - a atualização é realizada
- 71 - a mensagem X é enviada
- 72 - a mensagem Y é enviada

O grafo

1

71

2

70

3

72

O grafo

1

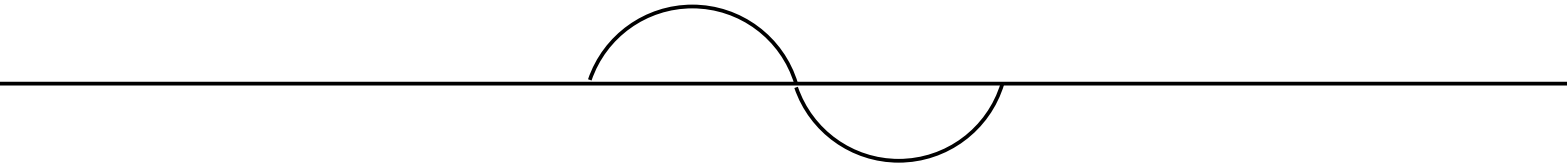
71

2

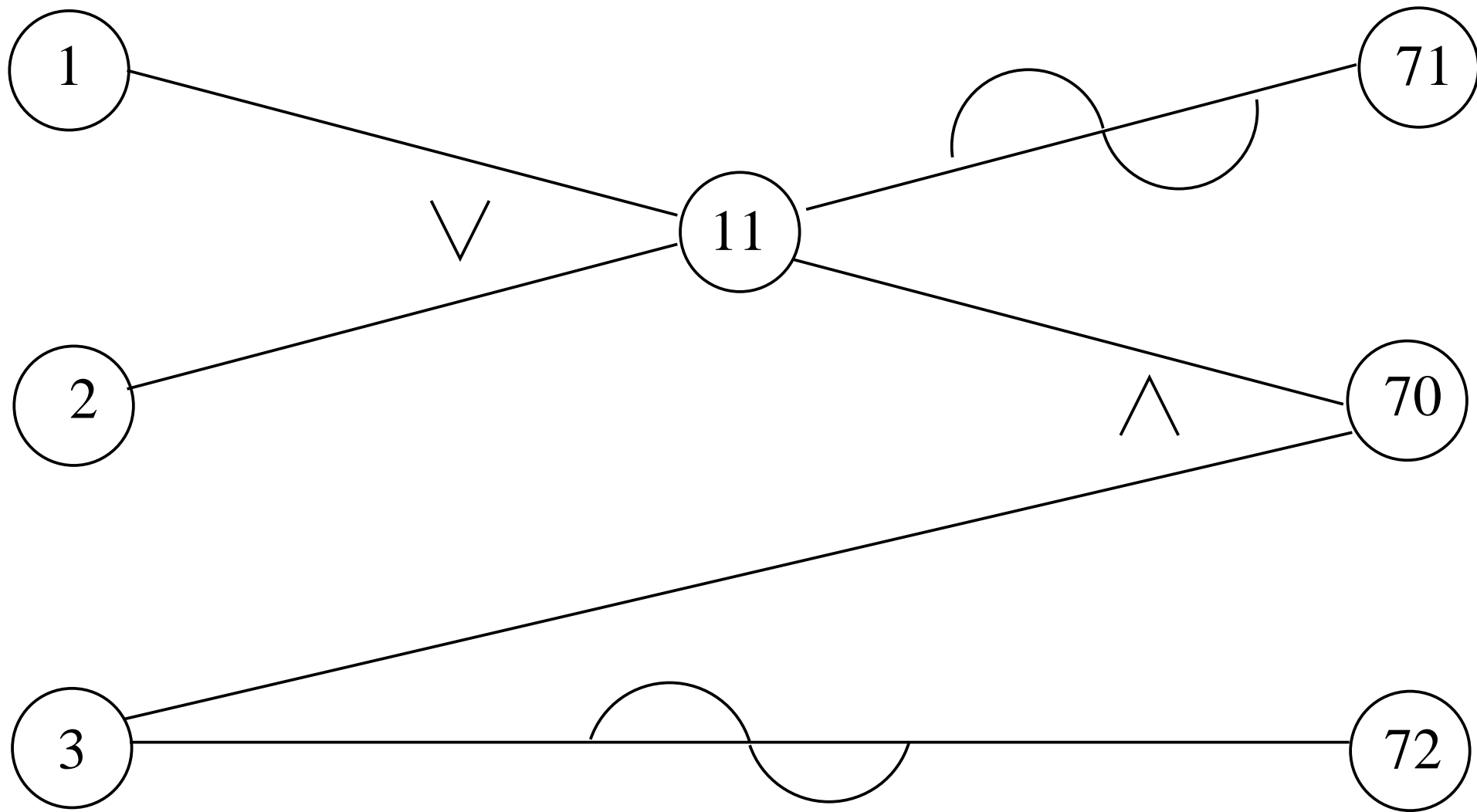
70

3

72



O grafo

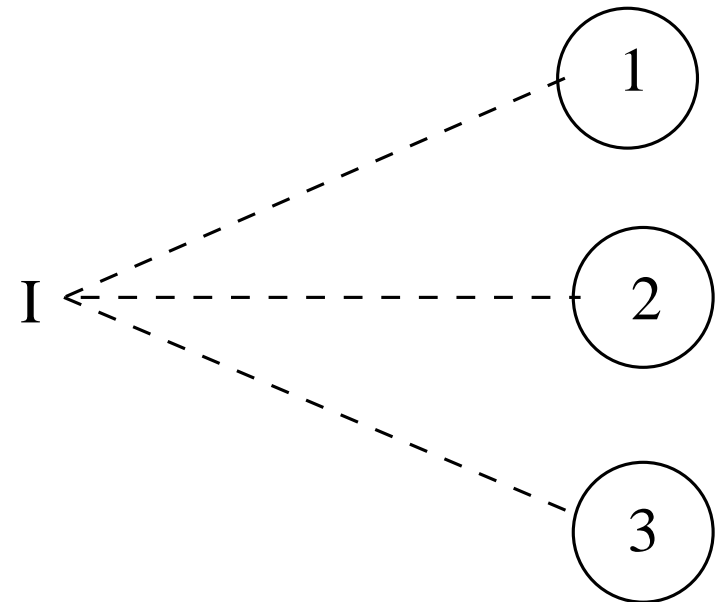
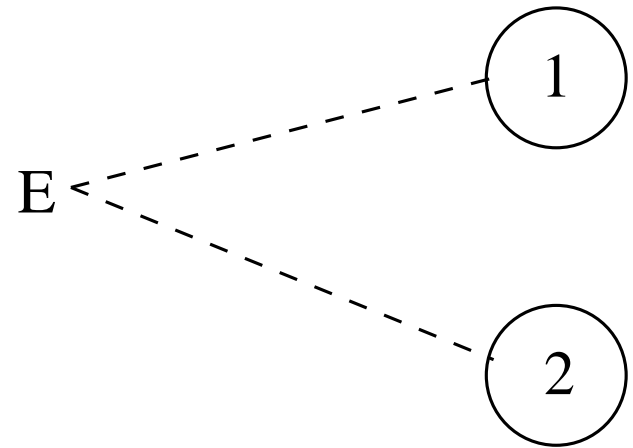


Restrições

- Para verificar se o grafo está correto, deve-se atribuir valores 0 e 1 para as causas e verificar se os efeitos assumem o valor correto
- Existem combinações que não são possíveis
- Essas restrições devem ser anotadas no grafo
- Por exemplo, causas 1 e 2 não podem ser verdadeiras ao mesmo tempo

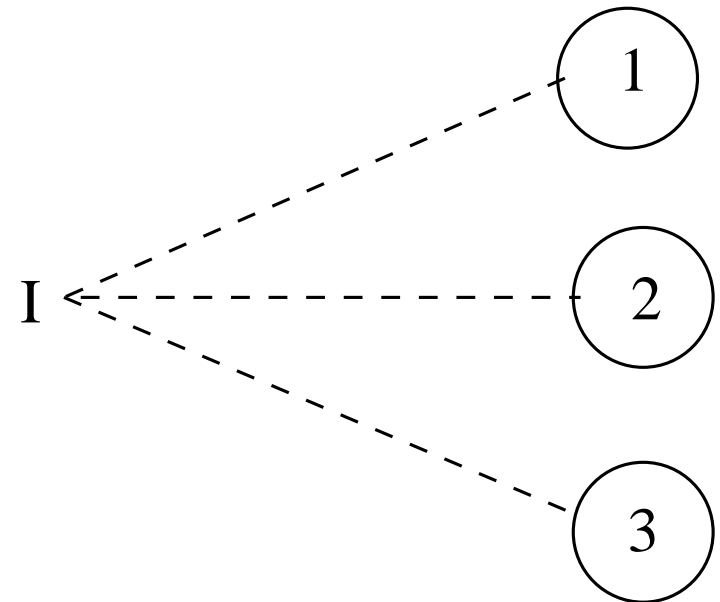
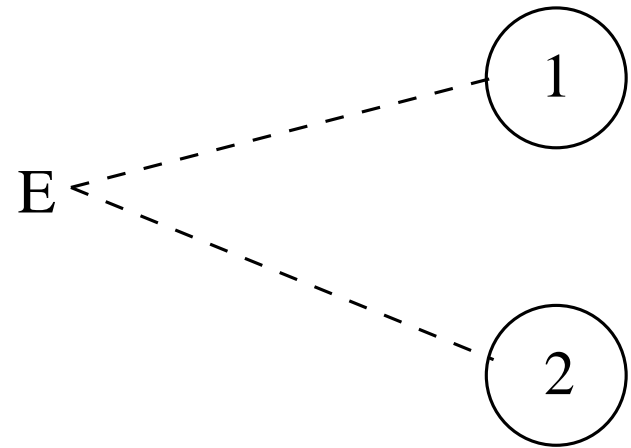
Restrições no grafo

- Restrição E: no máximo um entre “1” e “2” pode ser igual a 1 (ou seja, “1” e “2” não podem ser 1 simultaneamente).



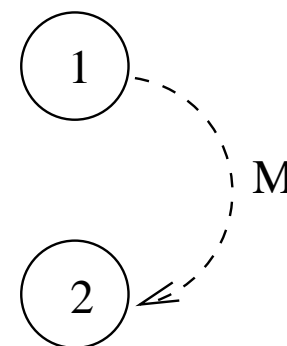
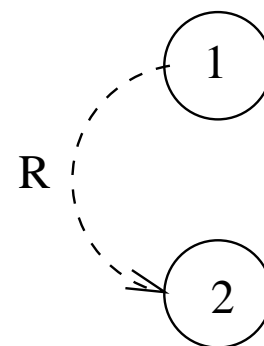
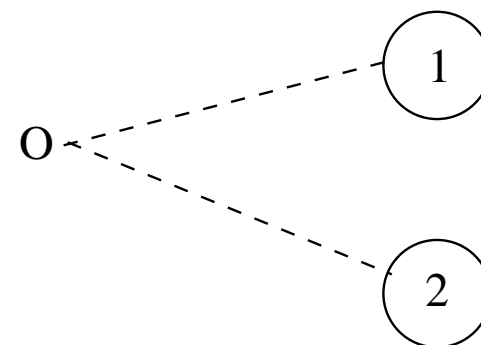
Restrições no grafo

- Restrição E: no máximo um entre “1” e “2” pode ser igual a 1 (ou seja, “1” e “2” não podem ser 1 simultaneamente).
- Restrição I: no mínimo um entre “1”, “2” e “3” deve ser igual a 1 (ou seja, “1”, “2” e “3” não podem ser 0 simultaneamente).



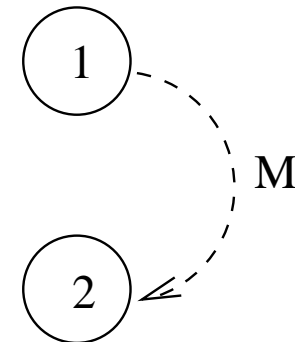
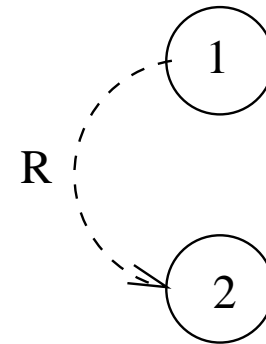
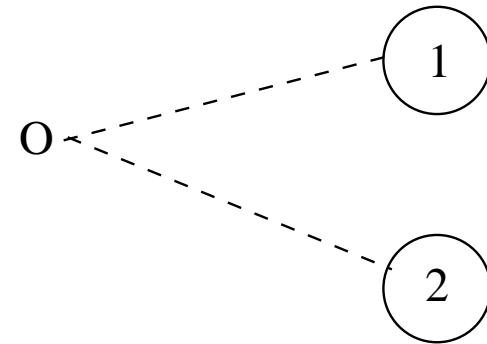
Restrições no grafo

- Restrição O: um e somente um entre “1” e “2” deve ser igual a 1.



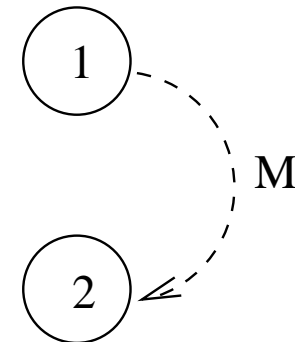
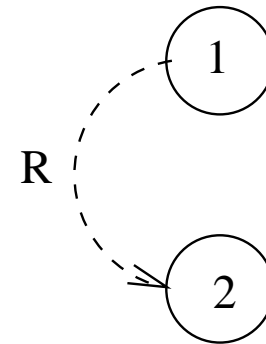
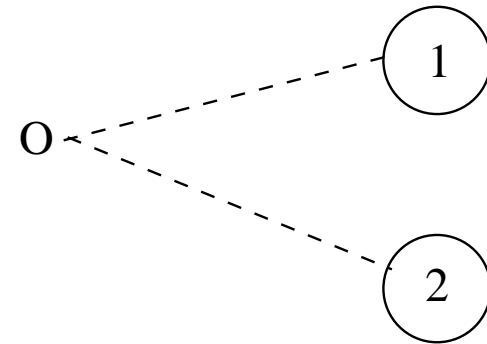
Restrições no grafo

- 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 (ou seja, é impossível que “1” seja 1 se “2” for 0).



Restrições no grafo

- 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 (ou seja, é impossível que “1” seja 1 se “2” for 0).
- Restrição M: se o efeito “1” é 1 o efeito “2” é forçado a ser 0.



O grafo

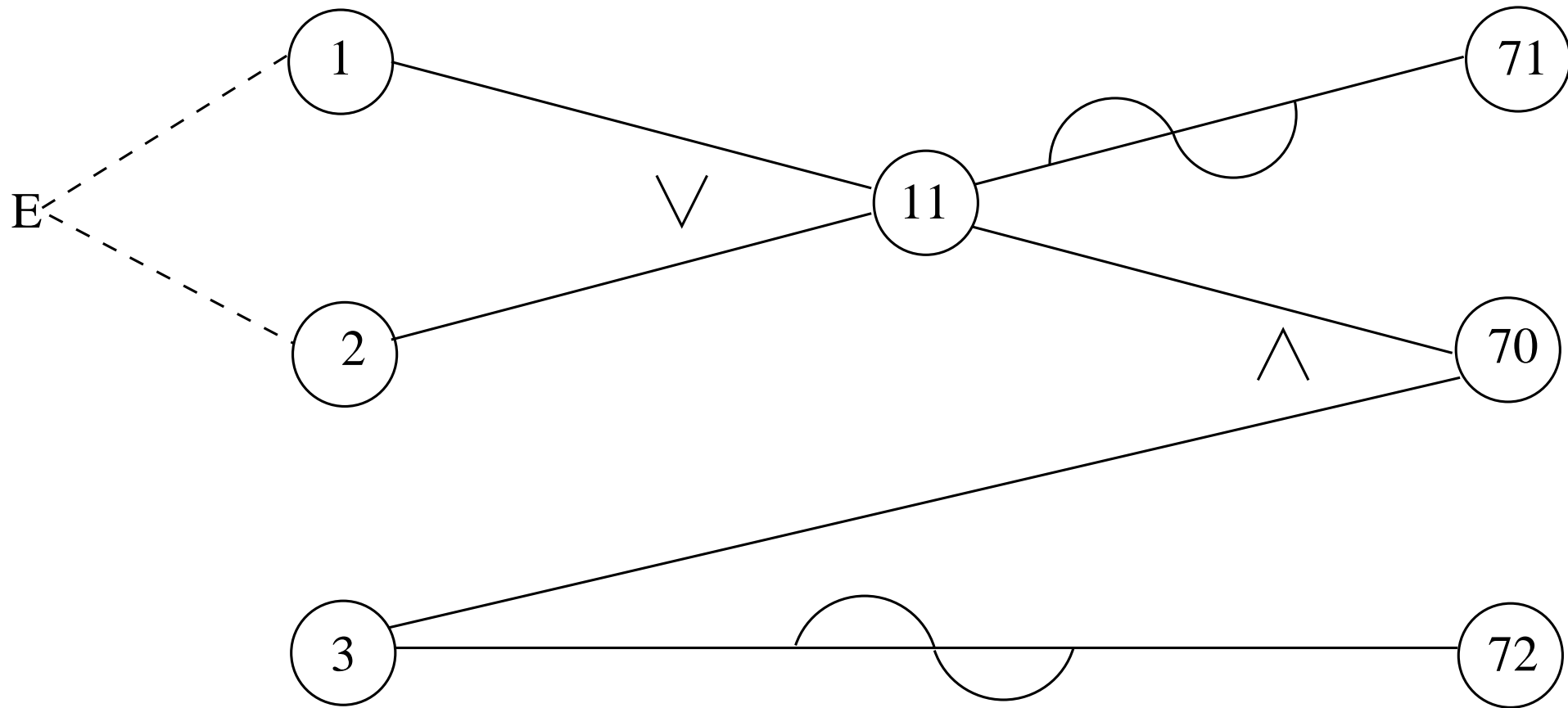


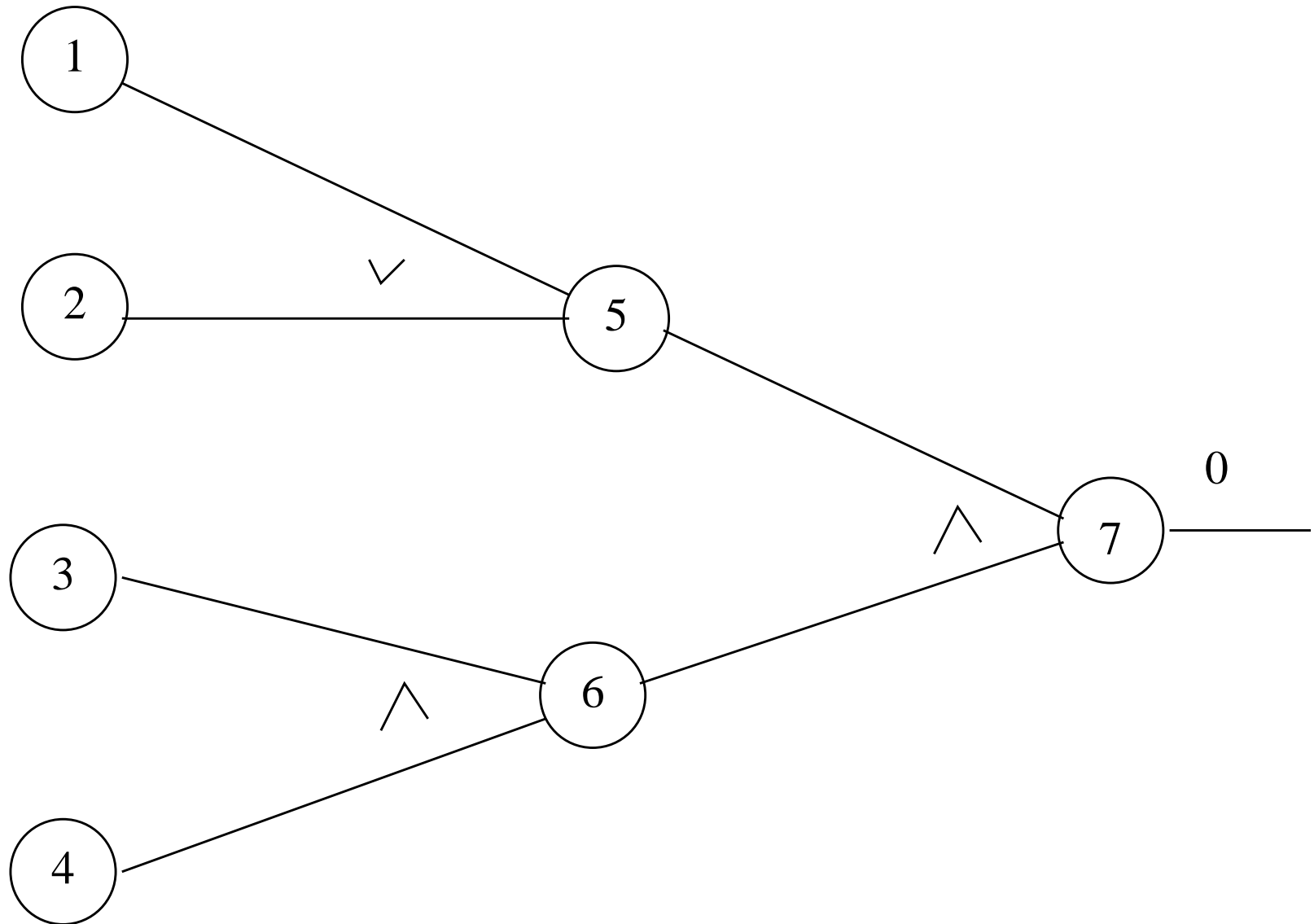
Tabela de decisão

- Forma de representar condições e ações
- O próximo passo é estudar sistematicamente o grafo e construir uma tabela de decisão.
 - 1 Selecionar um efeito para estar com valor 1.
 - 2 Rastrear o grafo para trás, encontrando todas as combinações de causas (sujeitas a restrições) que fazem com que esse efeito seja 1.
 - 3 Criar uma coluna na tabela de decisão para cada combinação de causa.
 - 4 Determinar, para cada combinação, os estados de todos os outros efeitos, anotando na tabela.

Considerações

- Ao executar o passo 2, fazer as seguintes considerações:
 - 1 Quando o nó for do tipo OR e a saída deva 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 deva 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.

Considerações – exemplo



Todas possibilidades

● 0 – 0 – 0 – 0

● 0 – 0 – 0 – 1

● 0 – 0 – 1 – 0

● 0 – 0 – 1 – 1

● 0 – 1 – 0 – 0

● 0 – 1 – 0 – 1

● 0 – 1 – 1 – 0

● 1 – 0 – 0 – 0

● 1 – 0 – 0 – 1

● 1 – 0 – 1 – 0

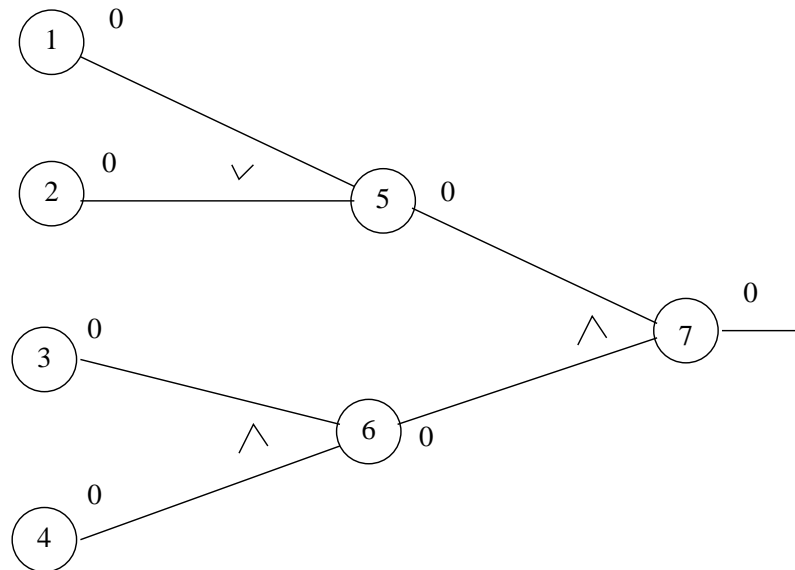
● 1 – 1 – 0 – 0

● 1 – 1 – 0 – 1

● 1 – 1 – 1 – 0

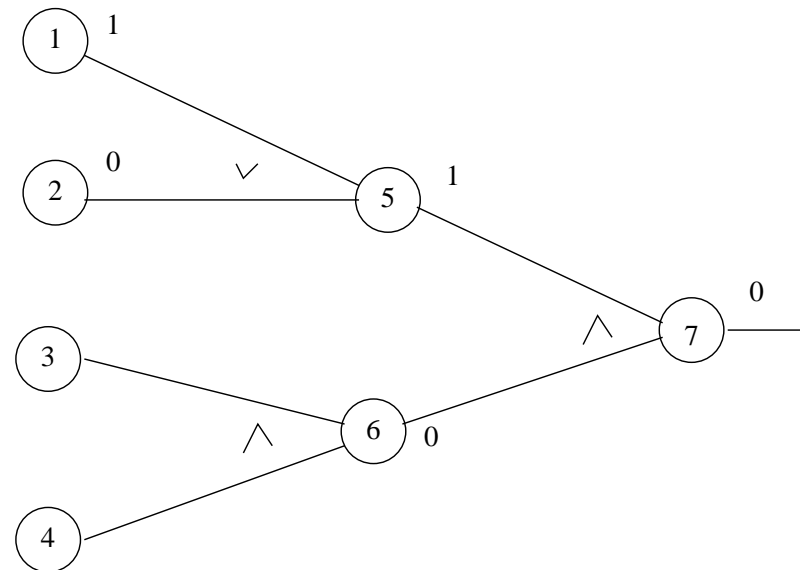
Consideração 3

- Somente uma situação em que 5 e 6 são ambos 0
- Por exemplo, selecionamos 0 – 0 – 0 – 0
- Eliminamos 0 – 0 – 0 – 1 e 0 – 0 – 1 – 0



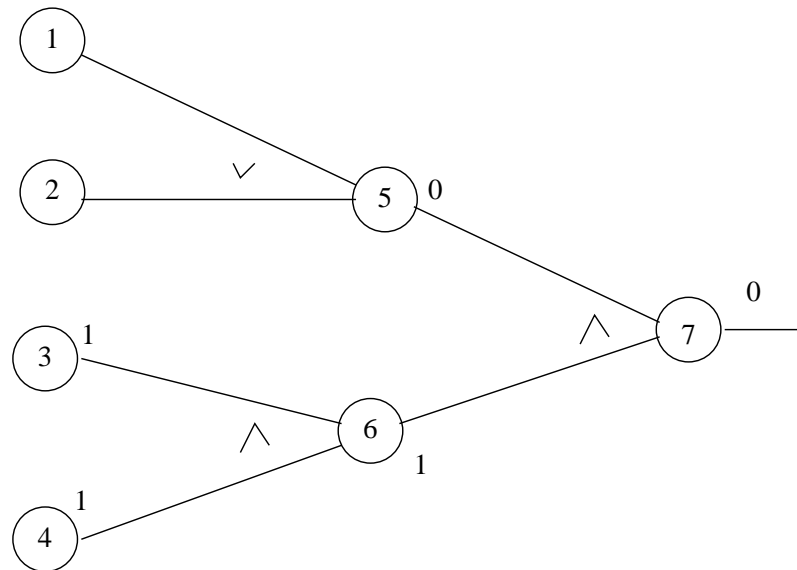
Consideração 2

- Quando 5 é 1 e 6 é 0, somente uma condição para que 5 seja 1 é requerida
- Consideração 1 diz que não se deve selecionar 1 – 1
- Por exemplo, selecionamos 1 – 0 – * – *
- Eliminamos 0 – 1 – * – * e 1 – 1 – * – *



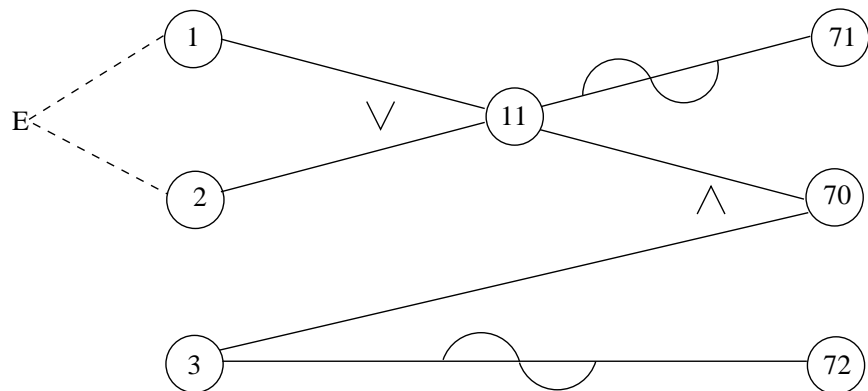
Consideração 2

- Da mesma forma, para 5 valendo 0 e 6 valendo 1, apenas uma condição para que 6 seja 1 é requerida
- Na verdade somente uma existe, que seria 0 – 0 – 1 – 1



Voltando ao exemplo

| | Nós | Casos de teste | | | | |
|---------|-----|----------------|---|---|---|---|
| Causas | 1 | 0 | 0 | 1 | 0 | 1 |
| | 2 | 0 | 0 | 0 | 1 | 0 |
| | 3 | 0 | 1 | 1 | 1 | 0 |
| Efeitos | 70 | 0 | 0 | 1 | 1 | 0 |
| | 71 | 1 | 1 | 0 | 0 | 0 |
| | 72 | 1 | 0 | 0 | 0 | 1 |



Exercício

- Faça o grafo causa-efeito para o exemplo da busca na cadeia de caracteres

Exercício: causas e efeitos

- Causas

- 1 inteiro positivo no intervalo 1-20;
- 2 caractere a ser procurado na cadeia;
- 3 procurar outro caractere.

- Efeitos:

- 20 inteiro fora do intervalo;
- 21 posição do caractere na cadeia;
- 22 caractere não encontrado;
- 23 término do programa.

Exercício: solução

| Nós | Casos de teste | | | |
|-----|----------------|---|---|---|
| 1 | 0 | 1 | 1 | - |
| 2 | - | 1 | 0 | - |
| 3 | - | 1 | 1 | 0 |
| 20 | 1 | 0 | 0 | 0 |
| 21 | 0 | 1 | 0 | 0 |
| 22 | 0 | 0 | 1 | 0 |
| 23 | - | 0 | 0 | 1 |

