

# Diagramas de Decisão Binários (BDDs)

---

Luiz Carlos Vieira

28 de Setembro de 2015

Instituto de Matemática e Estatística da Universidade de São Paulo

- Representação de Funções Booleanas
  - fórmulas proposicionais e tabelas-verdade
  - diagramas de decisão binários (BDDs)
  - diagramas de decisão binários ordenados (OBDDs)
- Algoritmos para OBDDs Reduzidos
  - algoritmo reduzir
  - algoritmo aplicar
  - algoritmo restringir
  - algoritmo existe

# funções booleanas

- Parte fundamental do formalismo descritivo de sistemas de *hardware* e *software*
- Que precisa ser computacionalmente representado de forma eficiente

# definição: variáveis booleanas

## Definição 6.1(a)

Uma variável booleana  $x$  é uma variável que só pode assumir os valores 0 e 1. Denotamos variáveis booleanas por  $x_1, x_2, \dots$ , e  $x, y$  e  $z, \dots$ .

# definição: funções booleanas

## Definição 6.1(b)

As seguintes funções são definidas no conjunto  $\{0, 1\}$ :

- $\overline{0} \stackrel{\text{def}}{=} 1$  e  $\overline{1} \stackrel{\text{def}}{=} 0$ ;
- $x \cdot y \stackrel{\text{def}}{=} 1$  se  $x$  e  $y$  têm valor 1; caso contrário,  $x \cdot y \stackrel{\text{def}}{=} 0$ ;
- $x + y \stackrel{\text{def}}{=} 0$  se  $x$  e  $y$  têm valor 0; caso contrário,  $x + y \stackrel{\text{def}}{=} 1$ ;
- $x \oplus y \stackrel{\text{def}}{=} 1$  se exatamente um entre  $x$  e  $y$  é igual a 1; caso contrário,  $x \oplus y \stackrel{\text{def}}{=} 0$ .

# funções e variáveis booleanas

- Uma função booleana  $f$  com  $n$  variáveis é uma função de  $\{0, 1\}^n$  para  $\{0, 1\}$ .
- Escreve-se  $f(x_1, x_2, \dots, x_n)$  ou  $f(\mathcal{V})$  para indicar que uma representação sintática de  $f$  só depende das variáveis booleanas em  $\mathcal{V}$ .

# alguns exemplos de funções booleanas

$$1. f(x, y) \stackrel{\text{def}}{=} x \cdot (y + \overline{x})$$

$$2. g(x, y) \stackrel{\text{def}}{=} x \cdot y + (1 \oplus \overline{x})$$

$$3. h(x, y, z) \stackrel{\text{def}}{=} x + y \cdot (x \oplus \overline{y})$$

$$4. k() \stackrel{\text{def}}{=} 1 \oplus (0 \cdot \overline{1})$$

# wffs e tabelas-verdade

As fórmulas proposicionais bem-formadas (*wffs*) e as tabelas-verdade são duas representações de funções booleanas

- **fórmulas proposicionais:**

- $\wedge$  denota  $\cdot$
- $\vee$  denota  $+$
- $\neg$  denota  $-$
- e  $\top$  e  $\perp$  denotam, respectivamente, 1 e 0

- **tabelas-verdade:** representam funções booleanas de maneira óbvia



# tabelas-verdade de funções booleanas

Tabela-verdade da função  
booleana  $f(x, y) \stackrel{\text{def}}{=} \overline{x + y}$

| $x$ | $y$ | $f(x, y)$ |
|-----|-----|-----------|
| 1   | 1   | 0         |
| 0   | 1   | 0         |
| 1   | 0   | 0         |
| 0   | 0   | 1         |

Tabela-verdade da fórmula  
proposicional  $\phi \equiv \neg(p \vee q)$

| $p$ | $q$ | $\phi$ |
|-----|-----|--------|
| $V$ | $V$ | $F$    |
| $F$ | $V$ | $F$    |
| $V$ | $F$ | $F$    |
| $F$ | $F$ | $V$    |

# vantagens e desvantagens

Há vantagens e desvantagens no uso de tabelas-verdade e fórmulas proposicionais para representar funções booleanas

|                     | <b>Tabelas-Verdade</b>            | <b>Fórmulas<br/>Proposicionais</b>        |
|---------------------|-----------------------------------|---|
| <b>Vantagens</b>    | verificações <sup>1</sup> simples | representação compacta                    |
| <b>Desvantagens</b> | ineficientes em espaço            | verificações <sup>1</sup> não tão simples |

Ambas são computacionalmente caras para muitas variáveis

---

<sup>1</sup>satisfação, validade e equivalência

# também nas operações booleanas

As operações booleanas ( $\cdot$ ,  $+$ ,  $\oplus$  e  $-$ ) entre duas funções  $f$  e  $g$  também são simples:

- Com tabelas-verdade
  - operação diretamente aplicada a cada linha
  - acrescentando variáveis inexistentes, se necessário
- Com fórmulas proposicionais
  - manipulação sintática da Lógica Proposicional

Computacionalmente caro com tabelas-verdade ( $2^n$  linhas) e imediata com fórmulas proposicionais (por exemplo,  $f \cdot g$  e  $f \oplus g$  são respectivamente  $\phi \wedge \psi$  e  $(\phi \wedge \neg\psi) \vee (\neg\phi \wedge \psi)$ )

# utilizando formas normais

- A representação de fórmulas proposicionais em formas normais é facilitada em alguns aspectos
  - mas é dificultada em outros
- De forma geral, elas podem ser muito longas no pior caso

# forma normal conjuntiva (CNF)

- Facilita o teste de validade
  - cláusula disjuntiva sem preposições complementares
  - teste de satisfação não é semelhante
- Facilita a operação de conjunção ( $\wedge$ )
  - se  $\phi$  e  $\psi$  são CNFs, o resultado de  $\phi \wedge \psi$  é CNF
- Dificulta as demais operações ( $\vee$  e  $\neg$ )
  - aplicação de distributividade para manter CNF

A forma normal disjuntiva (DNF) – disjunção de conjunções – é dual com a CNF em relação a essas propriedades

# resumo da eficiência das representações

| Representação de<br>funções booleanas | compacta?        | teste de   |          | operações booleanas |                  |         |
|---------------------------------------|------------------|------------|----------|---------------------|------------------|---------|
|                                       |                  | satisfação | validade | ·                   | +                | -       |
| <b>fórmulas<br/>proposicionais</b>    | muitas vezes     | difícil    | difícil  | fácil               | fácil            | fácil   |
| <b>fórmulas CNF</b>                   | algumas<br>vezes | difícil    | fácil    | fácil               | difícil          | difícil |
| <b>fórmulas NDF</b>                   | algumas<br>vezes | fácil      | difícil  | difícil             | fácil            | difícil |
| <b>tabelas-verdade<br/>ordenadas</b>  | nunca            | difícil    | difícil  | difícil             | difícil          | difícil |
| <b>OBDDs<sup>2</sup> reduzidos</b>    | muitas vezes     | fácil      | fácil    | mais ou<br>menos    | mais ou<br>menos | fácil   |

<sup>2</sup>Diagramas de Decisão Binários Ordenados – que serão explorados a seguir

# definição: árvore de decisão binária finita

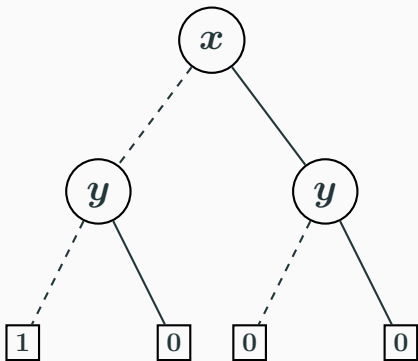
## Definição 6.3

Seja  $T$  uma árvore de decisão binária finita. Então  $T$  determina *uma única* função booleana das variáveis nos nós não-terminais da seguinte maneira:

*Dada uma atribuição de 0's e 1's às variáveis booleanas que ocorrem em  $T$ , começamos pela raiz de  $T$  e pegamos a linha tracejada sempre que o valor da variável no nó atual é 0; caso contrário, percorremos a linha sólida. O valor da função é o valor do nó terminal atingido.*

# por exemplo

- Árvore da função:  
 $f(x, y) \stackrel{\text{def}}{=} \overline{x + y}$



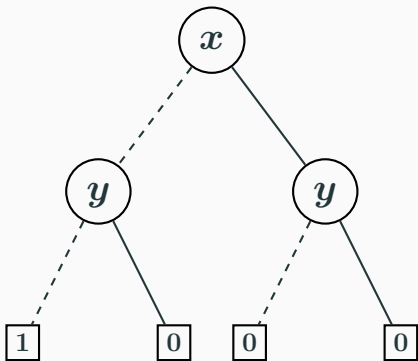


# por exemplo

- Árvore da função:

$$f(x, y) \stackrel{\text{def}}{=} \overline{x + y}$$

- Para encontrar  $f(0, 1)$ :



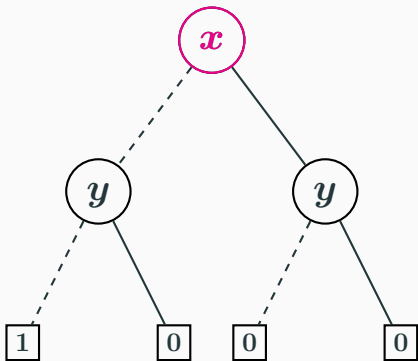
# por exemplo

- Árvore da função:

$$f(x, y) \stackrel{\text{def}}{=} \overline{x + y}$$

- Para encontrar  $f(0, 1)$ :

1. inicia-se pela raiz



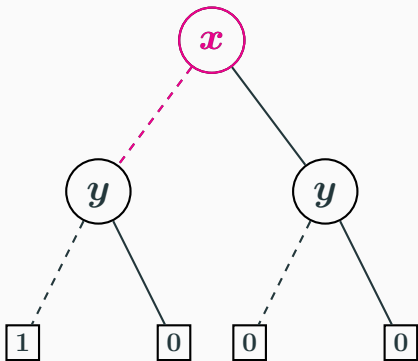
# por exemplo

- Árvore da função:

$$f(x, y) \stackrel{\text{def}}{=} \overline{x + y}$$

- Para encontrar  $f(0, 1)$ :

1. inicia-se pela raiz
2. como  $x$  é 0, segue-se pela linha pontilhada



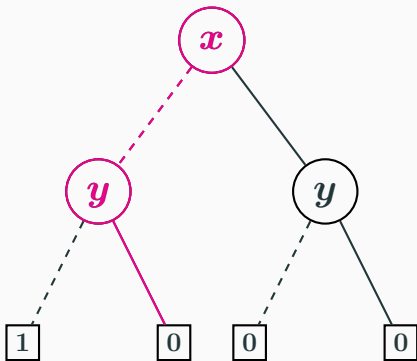
# por exemplo

- Árvore da função:

$$f(x, y) \stackrel{\text{def}}{=} \overline{x + y}$$

- Para encontrar  $f(0, 1)$ :

1. inicia-se pela raiz
2. como  $x$  é 0, segue-se pela linha pontilhada
3. como  $y$  é 1, segue-se pela linha sólida



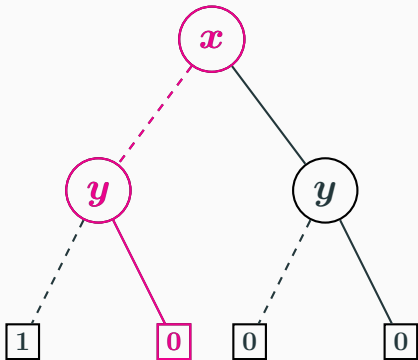
# por exemplo

- Árvore da função:

$$f(x, y) \stackrel{\text{def}}{=} \overline{x + y}$$

- Para encontrar  $f(0, 1)$ :

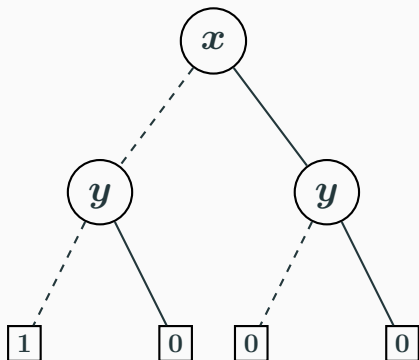
1. inicia-se pela raiz
2. como  $x$  é 0, segue-se pela linha pontilhada
3. como  $y$  é 1, segue-se pela linha sólida
4. chega-se à folha 0; logo  $f(0, 1) = 0$



# comparando com a tabela-verdade

Para a função booleana  $f(x, y) \stackrel{\text{def}}{=} \overline{x + y}$ :

equivalente à fórmula proposicional  $\phi \equiv \neg(p \vee q)$

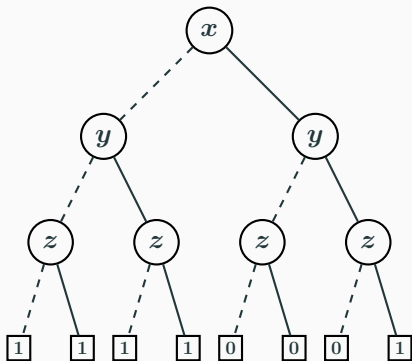


| $x$ | $y$ | $f(x, y)$ |
|-----|-----|-----------|
| 0   | 0   | 1         |
| 0   | 1   | 0         |
| 1   | 0   | 0         |
| 1   | 1   | 0         |

# outro exemplo comparativo

Para a função booleana  $f(x, y, z) \stackrel{\text{def}}{=} \bar{x} + (y \cdot z)$ :

equivalente à fórmula proposicional  $\phi \equiv p \rightarrow (q \wedge r)$



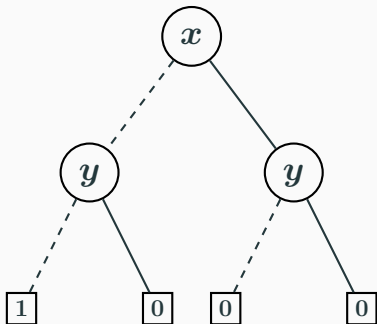
| $x$ | $y$ | $z$ | $f(x, y, z)$ |
|-----|-----|-----|--------------|
| 0   | 0   | 0   | 1            |
| 0   | 0   | 1   | 1            |
| 0   | 1   | 0   | 1            |
| 0   | 1   | 1   | 1            |
| 1   | 0   | 0   | 0            |
| 1   | 0   | 1   | 0            |
| 1   | 1   | 0   | 0            |
| 1   | 1   | 1   | 1            |

# semelhanças com tabelas-verdade

- Árvores de Decisão Binárias são semelhantes às tabelas-verdade em relação ao tamanho
  - se  $f$  depender de  $n$  variáveis booleanas, a árvore correspondente terá pelo menos  $2^{n+1} - 1$  nós (contra as  $2^n$  linhas da tabela verdade)
- Mas muitas vezes elas contêm redundâncias que podem ser exploradas



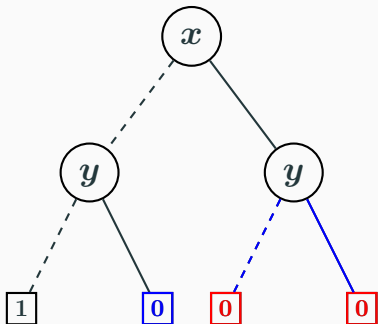
# primeira simplificação



## C1. Remoção de terminais duplicados

*Se há mais de um nó terminal 0, todas as arestas que apontam para tais nós são redirecionadas para apenas um deles. O processo é então repetido para os nós terminais 1*

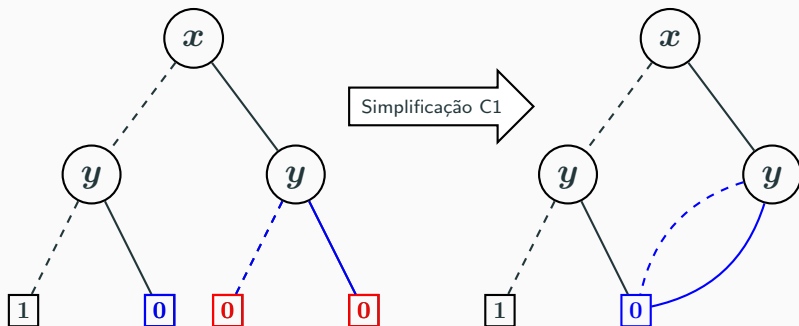
# primeira simplificação



## C1. Remoção de terminais duplicados

*Se há mais de um nó terminal 0, todas as arestas que apontam para tais nós são redirecionadas para apenas um deles. O processo é então repetido para os nós terminais 1*

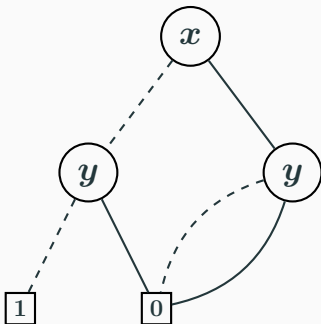
# primeira simplificação



## C1. Remoção de terminais duplicados

*Se há mais de um nó terminal 0, todas as arestas que apontam para tais nós são redirecionadas para apenas um deles. O processo é então repetido para os nós terminais 1*

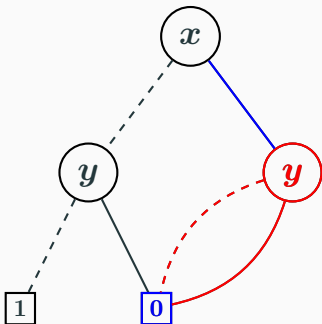
# segunda simplificação



## C2. Remoção de testes redundantes

*Se ambas as arestas de um nó  $n$  apontam para o mesmo nó  $m$ , o nó  $n$  é eliminado e todas as arestas que nele chegavam são redirecionadas diretamente para o nó  $m$ .*

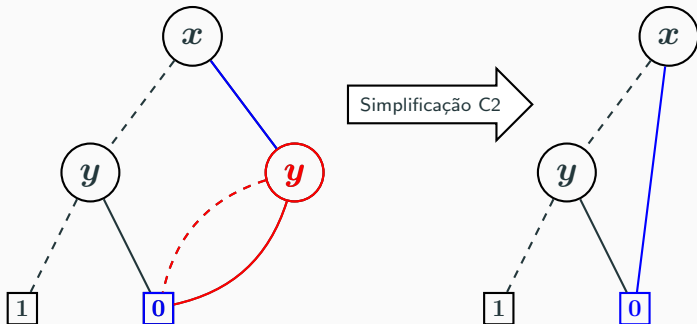
# segunda simplificação



## C2. Remoção de testes redundantes

*Se ambas as arestas de um nó  $n$  apontam para o mesmo nó  $m$ , o nó  $n$  é eliminado e todas as arestas que nele chegavam são redirecionadas diretamente para o nó  $m$ .*

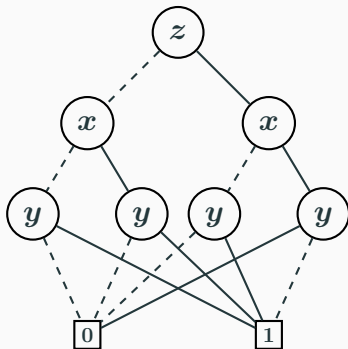
# segunda simplificação



## C2. Remoção de testes redundantes

*Se ambas as arestas de um nó  $n$  apontam para o mesmo nó  $m$ , o nó  $n$  é eliminado e todas as arestas que nele chegavam são redirecionadas diretamente para o nó  $m$ .*

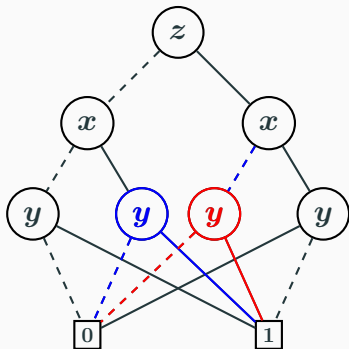
# terceira simplificação



## C3. Remoção de nós não-terminais duplicados

*Se dois nós distintos  $n$  e  $m$  são raízes de subárvores idênticas, um dos nós é eliminado e todas as arestas que nele chegavam são redirecionadas para o outro*

# terceira simplificação

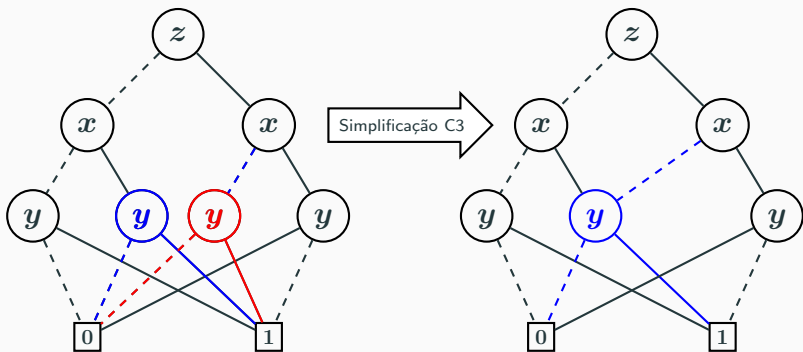


## C3. Remoção de nós não-terminais duplicados

*Se dois nós distintos  $n$  e  $m$  são raízes de subárvores idênticas, um dos nós é eliminado e todas as arestas que nele chegavam são redirecionadas para o outro*



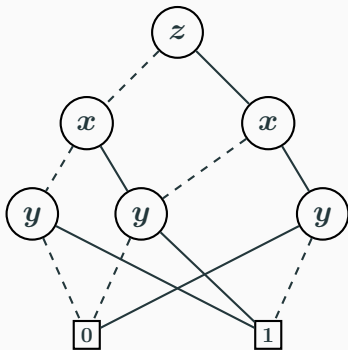
# terceira simplificação



## C3. Remoção de nós não-terminais duplicados

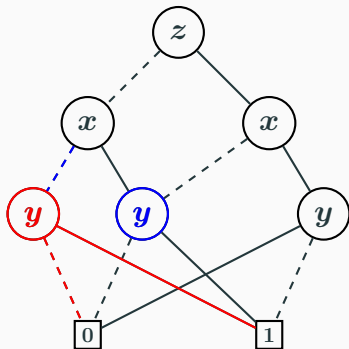
*Se dois nós distintos  $n$  e  $m$  são raízes de subárvores idênticas, um dos nós é eliminado e todas as arestas que nele chegavam são redirecionadas para o outro*

# processo de redução



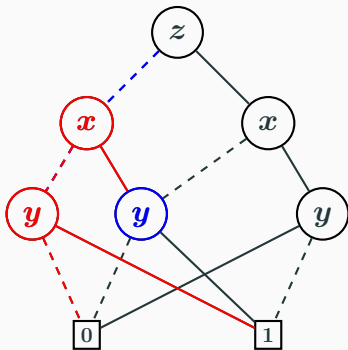
*As simplificações são encadeadas até não mais ser possível. O exemplo anterior é completamente reduzido após a eliminação de um dos nós  $y$  duplicados (C3) seguida da eliminação de um ponto de decisão  $x$  redundante (C2)*

# processo de redução



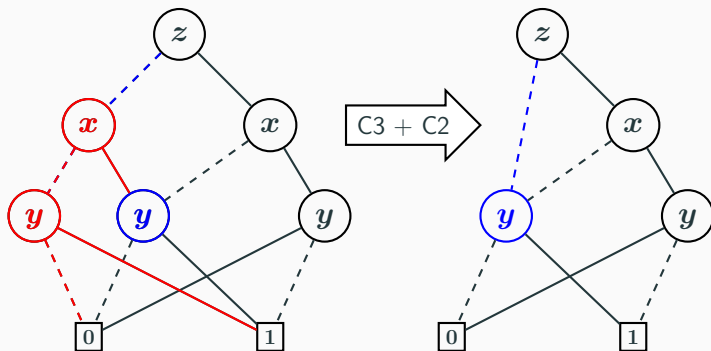
*As simplificações são encadeadas até não mais ser possível.  
O exemplo anterior é completamente reduzido após a eliminação de um dos nós  $y$  duplicados (C3) seguida da eliminação de um ponto de decisão  $x$  redundante (C2)*

# processo de redução



*As simplificações são encadeadas até não mais ser possível. O exemplo anterior é completamente reduzido após a eliminação de um dos nós  $y$  duplicados (C3) seguida da eliminação de um ponto de decisão  $x$  redundante (C2)*

# processo de redução

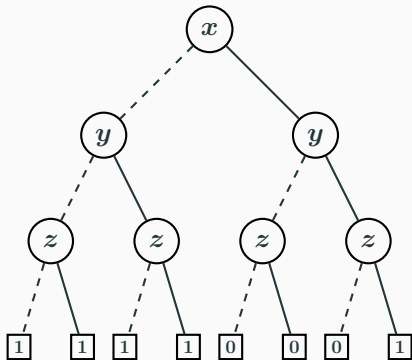


*As simplificações são encadeadas até não mais ser possível. O exemplo anterior é completamente reduzido após a eliminação de um dos nós  $y$  duplicados (C3) seguida da eliminação de um ponto de decisão  $x$  redundante (C2)*

# exercício 1

Reduza a árvore de decisão binária da função

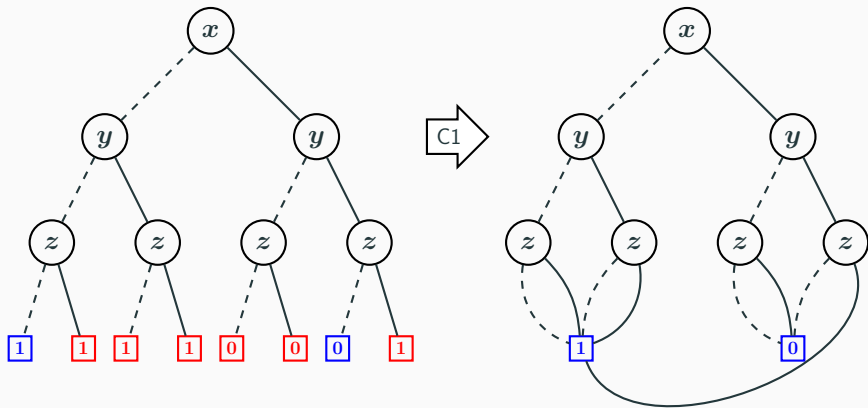
$f(x, y, z) \stackrel{\text{def}}{=} \overline{x} + (y \cdot z)$  apresentada anteriormente:



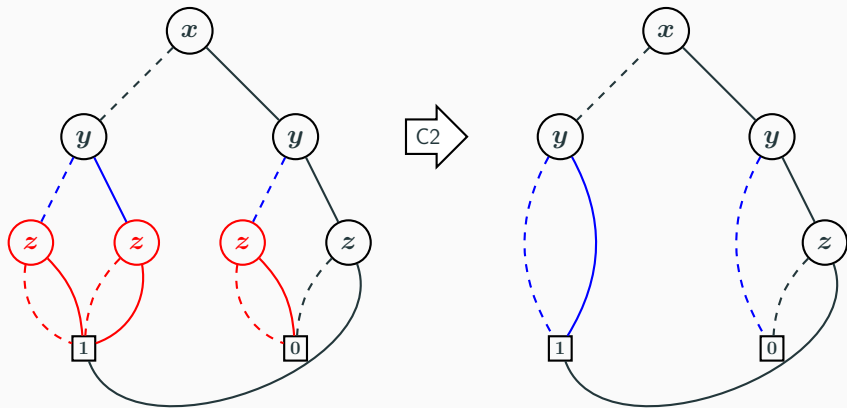
Resumo das simplificações:

- C1. Remoção de nós terminais duplicados
- C2. Remoção de testes redundantes
- C3. Remoção de nós não-terminais duplicados

# solução – 1º passo

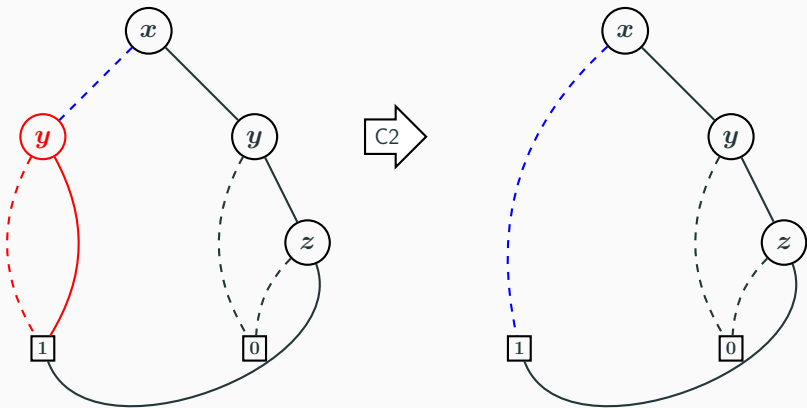


## solução – 2º passo



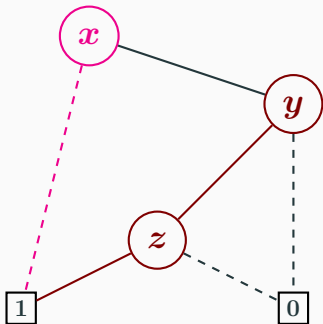


## solução – 3º passo



# comparando com a tabela-verdade

Função booleana:  $f(x, y, z) \stackrel{\text{def}}{=} \overline{x} + (y \cdot z)$ :



| $x$ | $y$ | $z$ | $f(x, y, z)$ |
|-----|-----|-----|--------------|
| 0   | 0   | 0   | 1            |
| 0   | 0   | 1   | 1            |
| 0   | 1   | 0   | 1            |
| 0   | 1   | 1   | 1            |
| 1   | 0   | 0   | 0            |
| 1   | 0   | 1   | 0            |
| 1   | 1   | 0   | 0            |
| 1   | 1   | 1   | 1            |

# BDDs

A redução faz com que as árvores se tornem grafos. Por isso, passam a ser chamados de **Diagramas de Decisão Binários (BDDs)**.

# definição: gda

## Definição 6.4

Um grafo direcionado é um conjunto  $G$  e uma relação binária  $\rightarrow$  em  $G : \rightarrow \subseteq G \times G$ . Um ciclo em um grafo direcionado é um caminho finito no grafo que começa e termina no mesmo nó, isto é, um caminho da forma  $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_n \rightarrow v_1$ . Um grafo direcionado acíclico (gda) é um grafo direcionado que não contém nenhum ciclo. Um nó em um gda é dito inicial se não há arestas apontando para ele. Um nó é dito terminal se não há arestas saindo dele.

# definição: BDDs

## Definição 6.5

Um diagrama de decisão binário (BDD) é um grafo finito com um único nó inicial, onde todos os nós terminais são marcados com **0** ou **1** e todos os nós não-terminais são marcados com uma variável booleana. Cada nó não-terminal tem exatamente duas arestas saindo dele, uma marcada com **0** e outra com **1** (representadas como uma linha pontilhada e uma linha sólida, respectivamente).

# BDD como gda

- Por convenção, as linhas sólidas ou pontilhadas de um BDD são sempre consideradas como indo para baixo
  - por isso eles são grafos direcionados
- Os BDDs são acíclicos (gda) e têm um único nó inicial
- As simplificações C1–C3 preservam essas propriedades
  - BDDs totalmente reduzidos têm 1 ou 2 nós terminais

# BDDs elementares

O BDD  $B_0$  representa a função booleana constante 0; analogamente, o BDD  $B_1$  representa a função booleana constante 1; e, finalmente, o BDD  $B_x$  representa a variável booleana  $x$

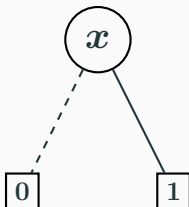
$B_0$



$B_1$



$B_x$



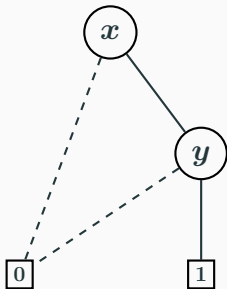
# verificações sobre BDDs

- **Satisfação.** Um BDD representa uma função que pode ser satisfeita se um nó terminal **1** pode ser acessado da raiz por meio de um caminho consistente
- **Validade.** Um BDD representa uma função válida se nenhum ponto terminal **0** é acessível por um caminho consistente



# exemplos óbvios

$$f(x, y) \stackrel{\text{def}}{=} x \cdot y$$



$$g(x) \stackrel{\text{def}}{=} x + \bar{x}$$



$$h(y) \stackrel{\text{def}}{=} y \cdot \bar{y}$$

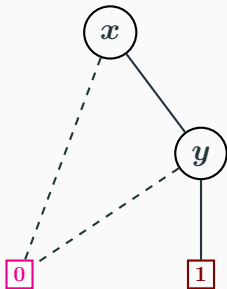


# operações sobre BDDs

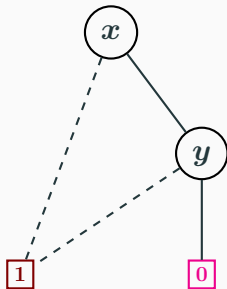
- **Operação de negação ( $\neg$ ).** Obtem-se um BDD que representa  $\overline{f}$  substituindo todos os terminais 0 em  $B_f$  por terminais 1 e vice-versa
- **Operação de conjunção ( $\cdot$ ).** Obtem-se um BDD que representa  $f \cdot g$  substituindo todos os nós terminais 1 em  $B_f$  diretamente por uma cópia de  $B_g$
- **Operação de disjunção ( $+$ ).** Obtem-se um BDD que representa  $f + g$  substituindo todos os nós terminais 0 em  $B_f$  diretamente por uma cópia de  $B_g$

# exemplo da negação

$$f(x, y) \stackrel{\text{def}}{=} x \cdot y$$

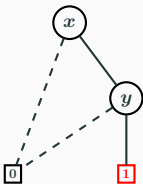


$$g(x, y) \stackrel{\text{def}}{=} \overline{x \cdot y}$$

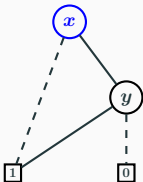


# exemplo da conjunção

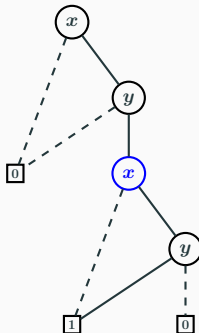
$$f(x, y) \stackrel{\text{def}}{=} x \cdot y$$



$$g(x, y) \stackrel{\text{def}}{=} \bar{x} + y$$

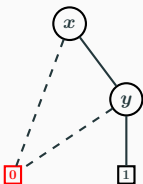


$$h(x, y) \stackrel{\text{def}}{=} (x \cdot y) \cdot (\bar{x} + y)$$

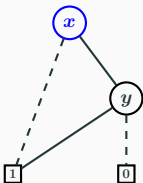


# exemplo da disjunção

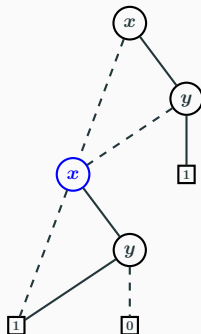
$$f(x, y) \stackrel{\text{def}}{=} x \cdot y$$



$$g(x, y) \stackrel{\text{def}}{=} \bar{x} + y$$



$$h(x, y) \stackrel{\text{def}}{=} (x \cdot y) + (\bar{x} + y)$$

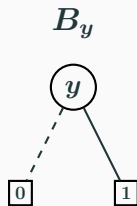
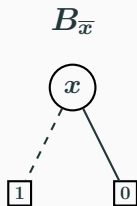
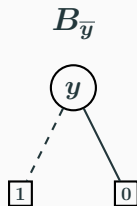
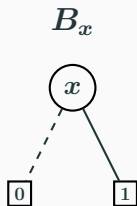


# forma “inocente” de construir BDDs

1. Para cada variável booleana em uma função, um BDD de variável ( $B_{x_i}$ ) é criado
2. Tais BDDs são então unidos conforme as operações booleanas constantes na função
3. Por fim, o BDD resultante é reduzido com as simplificações C1-C3

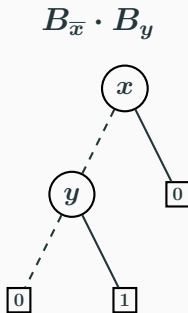
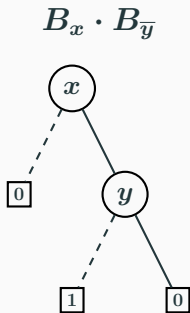
exemplo:  $(x \cdot \overline{y}) + (\overline{x} \cdot y)$

Passo 1: criação de  $B_{x_i}$



exemplo:  $(x \cdot \bar{y}) + (\bar{x} \cdot y)$

Passo 2a: união dos BDDs conforme as operações

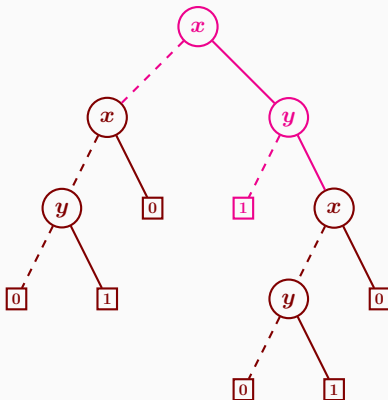




exemplo:  $(x \cdot \bar{y}) + (\bar{x} \cdot y)$

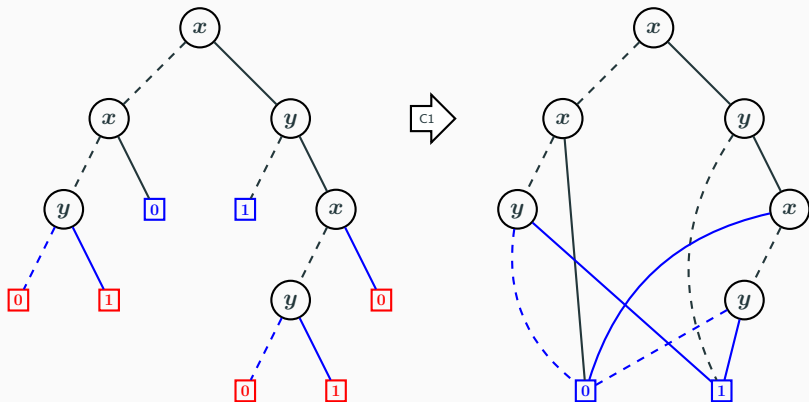
Passo 2b: união dos BDDs conforme as operações

$$(B_x \cdot B_{\bar{y}}) + (B_{\bar{x}} \cdot B_y)$$



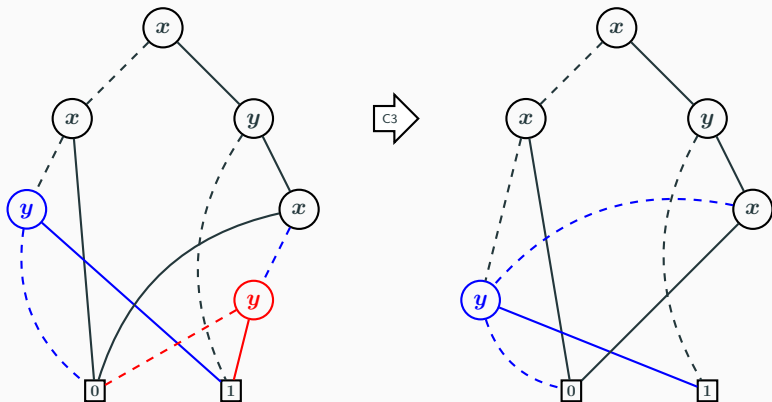
exemplo:  $(x \cdot \bar{y}) + (\bar{x} \cdot y)$

Passo 3a: redução do BDD gerado



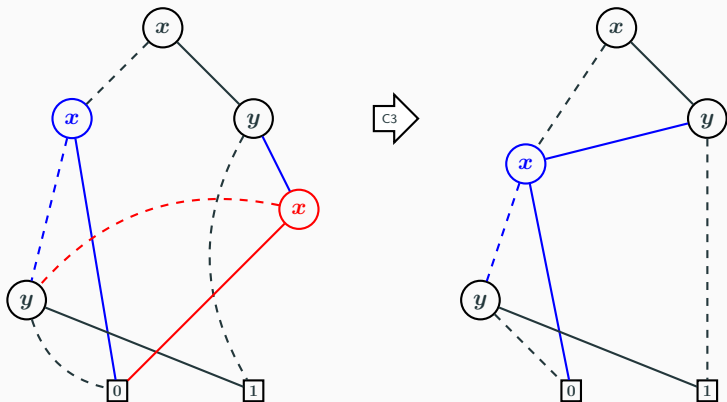
exemplo:  $(x \cdot \bar{y}) + (\bar{x} \cdot y)$

Passo 3b: redução do BDD gerado



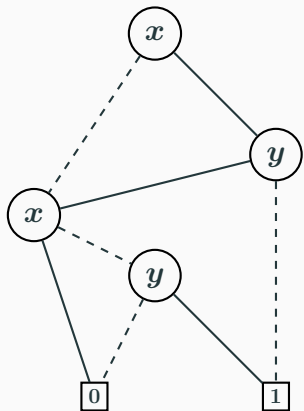
exemplo:  $(x \cdot \bar{y}) + (\bar{x} \cdot y)$

Passo 3c: redução do BDD gerado



# comparação com a tabela-verdade

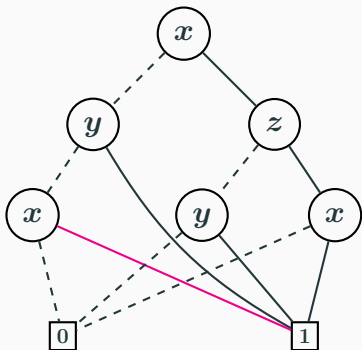
$$f(x, y) \stackrel{\text{def}}{=} (x \cdot \bar{y}) + (\bar{x} \cdot y)$$



| $x$ | $y$ | $f(x, y)$ |
|-----|-----|-----------|
| 0   | 0   | 0         |
| 0   | 1   | 1         |
| 1   | 0   | 1         |
| 1   | 1   | 0         |

# múltiplas ocorrências de mesma variável

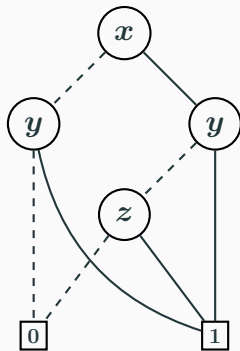
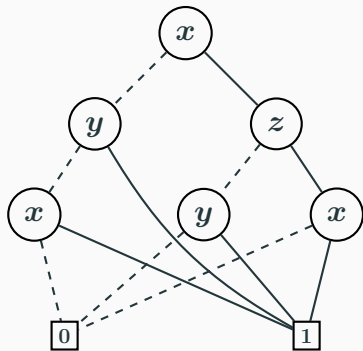
- A definição não impede uma variável de ocorrer mais de uma vez em um caminho
- Mas tal representação pode incorrer em desperdícios
  - linha sólida do  $x$  à esquerda (colorida) jamais será percorrida



Comum após as operações de conjunção e disjunção discutidas anteriormente (algoritmos melhores serão discutidos à frente)

# comparação de BDDs

Além de tornar um BDD ineficiente, ocorrências múltiplas de uma variável também dificultam a comparação de BDDs



# BDDs ordenados

- Quando a ordem das variáveis de teste nos caminhos que levam da raiz até uma folha é sempre a mesma, o BDD é dito *ordenado*
  - e passa a ser chamado Diagrama de Busca Binária Ordenado (OBDD)
- Esse compromisso com a ordem dá uma representação única de funções booleanas com OBDDs



# teorema: obdds reduzidos são únicos

## Teorema 6.7

A representação em OBDD reduzido de uma função dada  $f$  é única. Isto é, sejam  $B$  e  $B'$  dois OBDDs reduzidos com ordens compatíveis. Se  $B$  e  $B'$  representam a mesma função booleana, então eles têm estruturas idênticas.

# características de OBDDs

- As simplificações C1-C3 em um OBDD produzem sempre o mesmo OBDD reduzido
  - chamado então de *forma canônica*
- ODDBs permitem representações compactas de certas classes de funções booleanas
  - que seriam exponenciais em outros formatos/representações
- Por outro lado, as operações  $\cdot$  e  $+$  apresentadas anteriormente não funcionam
  - pois podem introduzir ocorrências múltiplas de uma mesma variável

# importância da representação canônica

- **Ausência de variáveis redundantes.** Se o valor de uma função booleana não depende de uma variável, então nenhum ODDB reduzido que a represente contém tal variável;
- **Teste de equivalência semântica.** Se duas funções são representadas por ODDBs com ordem compatível, é possível decidir eficientemente se são equivalentes reduzindo seus ODDBs e comparando sua estrutura;
- **Teste de validade.** Se uma função booleana é válida, seu ODDB reduzido é igual a  $B_1$ ;
- **Teste de implicação.** Pode-se testar se uma função  $f$  implica em outra  $g$  calculando o ODDB para  $f \cdot g$  e verificando que ele é igual a  $B_0$ ;
- **Teste de satisfação.** Se uma função booleana é satisfeita, então seu ODDB reduzido não é igual a  $B_0$ .

# algoritmo reduzir

# algoritmo aplicar

# algoritmo restringir

# algoritmo esiste