Diagramas de Decisão Binária (BDDs)

Aula 1

Luiz Carlos Vieira

2 de outubro de 2015

MAC0239 - Introdução à Lógica e Verificação de Programas

Conteúdo da aula

- Representação de Funções Booleanas
- Representação com fórmulas proposicionais e tabelas-verdade
- Diagramas de Decisão Binária (BDDs)
- Operações sobre BDDs

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, ..., e x, y e z, ...

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 $f(x_1, x_2, \ldots, x_n)$ 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**:
  - $p \wedge q$ denota $p \cdot q$
  - $p \vee q$ denota $p + q$
  - $\neg p$ denota $\overline{p}$
  - e $\top$ e $\bot$ denotam, respectivamente, $\mathbf{1}$ e $\mathbf{0}$
- **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	$f(x, y)$
0	0	1
0	1	0
1	0	0
1	1	0

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

p	q	ϕ
F	F	V
F	V	F
V	F	F
V	V	F

Sobre o sistema utilizado...

- No contexto desta aula, tabelas-verdade, fórmulas proposicionais e BDDs (em estudo) são diferentes formas de representação computacional de funções booleanas

- Uma vez que tais representações são facilmente traduzíveis entre si, os símbolos da lógica proposicional serão utilizados com o objetivo de facilitar o entendimento
  - a única distinção será a utilização de $\mathbf{0}$ e $\mathbf{1}$ no lugar de $\boldsymbol{F}$ e $\boldsymbol{V}$ nas representações de tabelas-verdade e diagramas

Vantagens e desvantagens

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

	Tabelas-Verdade	**Fórmulas Proposicionais**
Vantagens	verificações[1] simples	representação compacta
Desvantagens	ineficientes em espaço	verificações[1] não tão simples

Ambas são computacionalmente caras para muitas variáveis

[1] satisfação, validade e equivalência

Também nas operações booleanas

As operações booleanas ($\wedge$, $\vee$ e $\neg$) entre duas funções ϕ e ψ também são simples:

- Com tabelas-verdade
  - operação diretamente aplicada a cada linha
  - acrescentando variáveis inexistentes, se necessário
  - mas computacionamente caro (2^n linhas)
- Com fórmulas proposicionais
  - manipulação sintática da Lógica Proposicional
  - de realização imediata

Por exemplo: disjunção

$\phi \equiv \neg p \wedge q$

p	q	ϕ
0	0	0
0	1	1
1	0	0
1	1	0

$\psi \equiv r$

r	ψ
0	0
1	1

Por exemplo: disjunção

$\phi \equiv \neg p \wedge q$

p	q	ϕ
0	0	0
0	1	1
1	0	0
1	1	0

$\psi \equiv r$

r	ψ
0	0
1	1

$\omega \equiv \phi \vee \psi$

Por exemplo: disjunção

$\phi \equiv \neg p \wedge q$ $\quad$ $\psi \equiv r$ $\quad$ $\omega \equiv \phi \vee \psi$

p	q	r	ϕ
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

p	q	r	ψ
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Por exemplo: disjunção

$\phi \equiv \neg p \wedge q$

p	q	r	ϕ
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

$\psi \equiv r$

p	q	r	ψ
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$\omega \equiv \phi \vee \psi$

p	q	r	ω
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

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 igualmente fácil
- Facilita a operação de conjunção ($\wedge$)
  - se ϕ e ψ 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 funções booleanas	compacta?	teste de satisfação	teste de validade	operações booleanas $\cdot$	operações booleanas $+$	operações booleanas $^-$
fórmulas proposicionais	muitas vezes	difícil	difícil	fácil	fácil	fácil
fórmulas CNF	algumas vezes	difícil	fácil	fácil	difícil	difícil
fórmulas NDF	algumas vezes	fácil	difícil	difícil	fácil	difícil
tabelas-verdade ordenadas	nunca	difícil	difícil	difícil	difícil	difícil

Resumo da eficiência das representações

Representação de funções booleanas	compacta?	teste de satisfação	teste de validade	operações booleanas ·	operações booleanas +	operações booleanas −
fórmulas proposicionais	muitas vezes	difícil	difícil	fácil	fácil	fácil
fórmulas CNF	algumas vezes	difícil	fácil	fácil	difícil	difícil
fórmulas NDF	algumas vezes	fácil	difícil	difícil	fácil	difícil
tabelas-verdade ordenadas	nunca	difícil	difícil	difícil	difícil	difícil
OBDDs[2] reduzidos	muitas vezes	fácil	fácil	mais ou menos	mais ou menos	fácil

[2] Diagramas de Decisão Binária Ordenados – que serão explorados a seguir

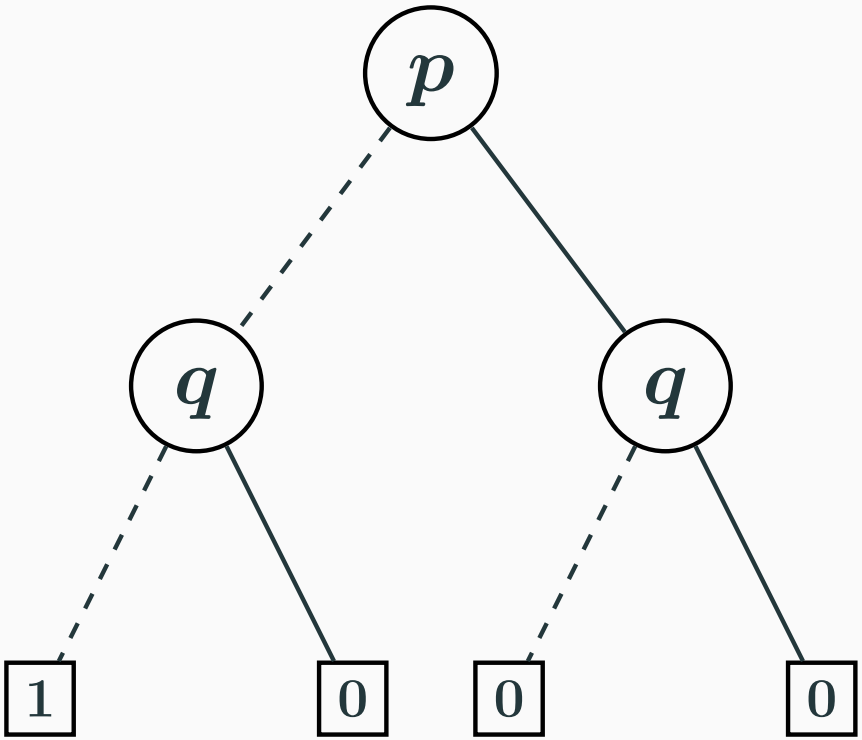
Definição: Árvore de Decisão Binária

Definição 6.3

Seja T uma árvore binária cujos nós não-terminais (nós de teste) contêm variáveis booleanas e cujos nós terminais contêm os valores 0 ou 1. Então T é uma árvore de decisão binária finita e determina *uma única* função booleana f da seguinte forma:

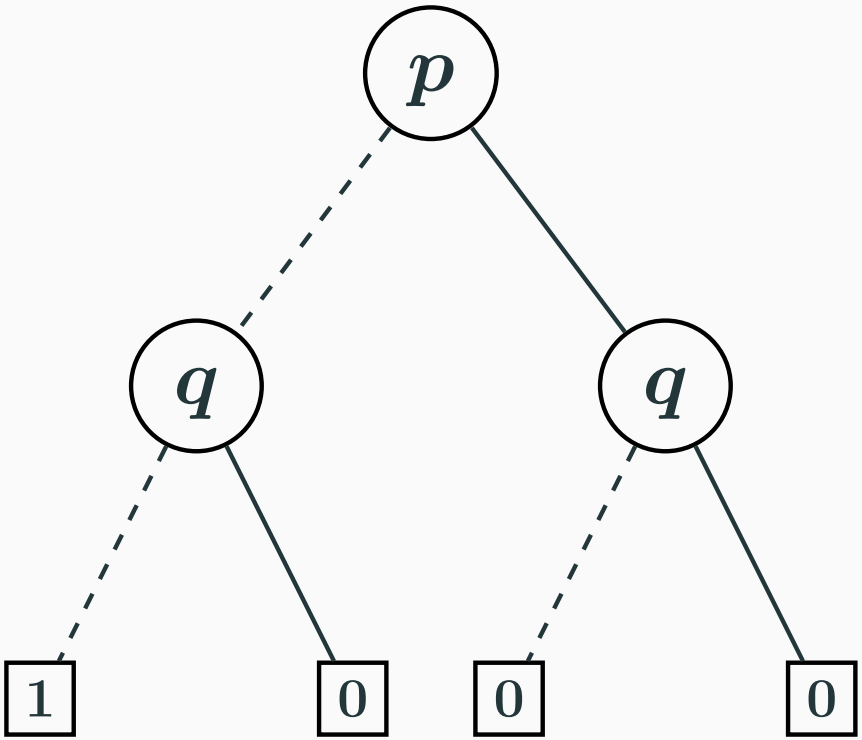
> *Dada uma atribuição de 0's e 1's às variáveis booleanas que ocorrem em f, começamos pela raiz de T e pegamos a linha tracejada sempre que o valor da variável no nó atual é 0; caso contrário, percorremos a linha sólida. O valor da função é o valor do nó terminal atingido.*

Por exemplo



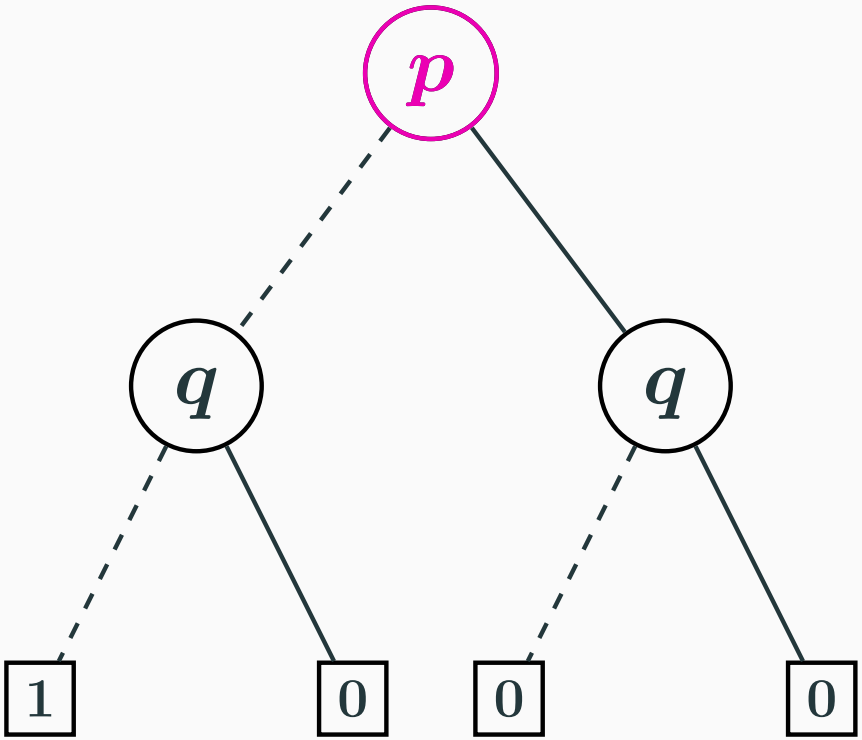
- Árvore da função:
  $\phi \equiv \neg(p \vee q)$

Por exemplo



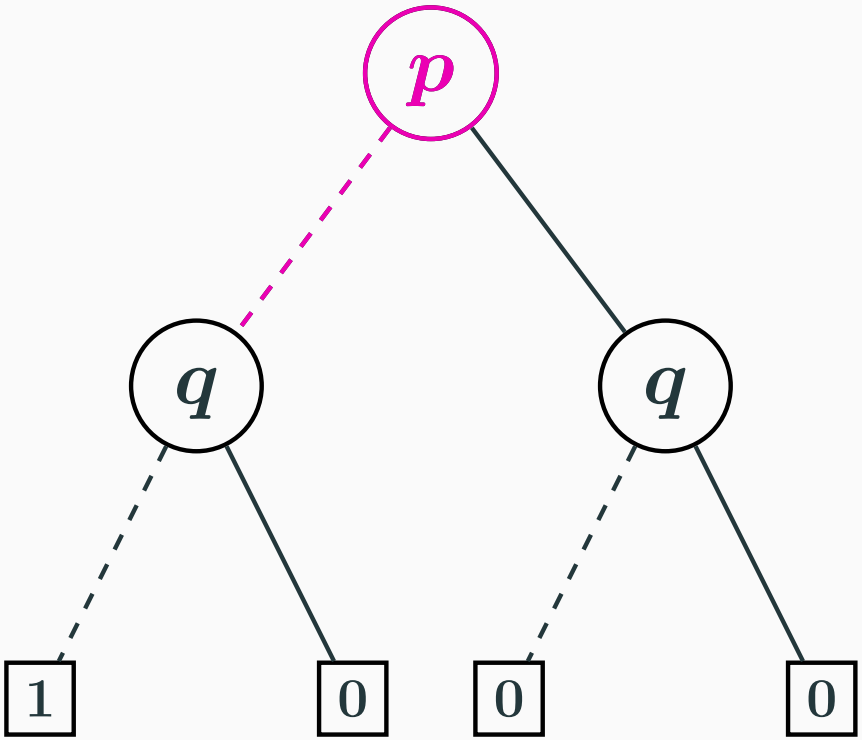
- Árvore da função:
  $\phi \equiv \neg(p \vee q)$
- Para encontrar $[\![\phi]\!]_{v_{(0,1)}}$:

Por exemplo



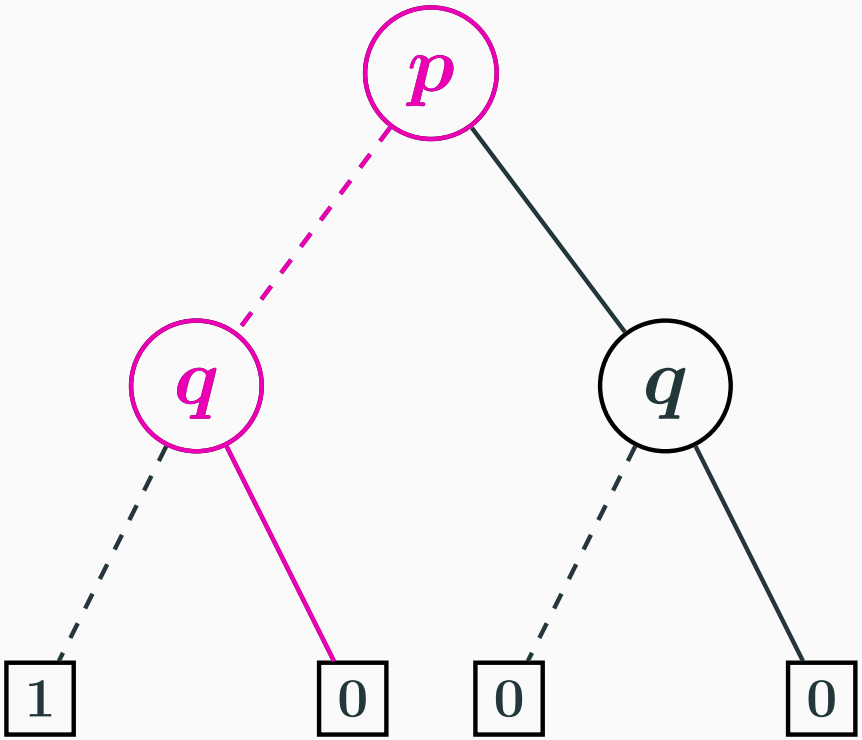
- Árvore da função:
  $\phi \equiv \neg(p \vee q)$
- Para encontrar $[\![\phi]\!]_{v_{(0,1)}}$:
  1. inicia-se pela raiz

Por exemplo



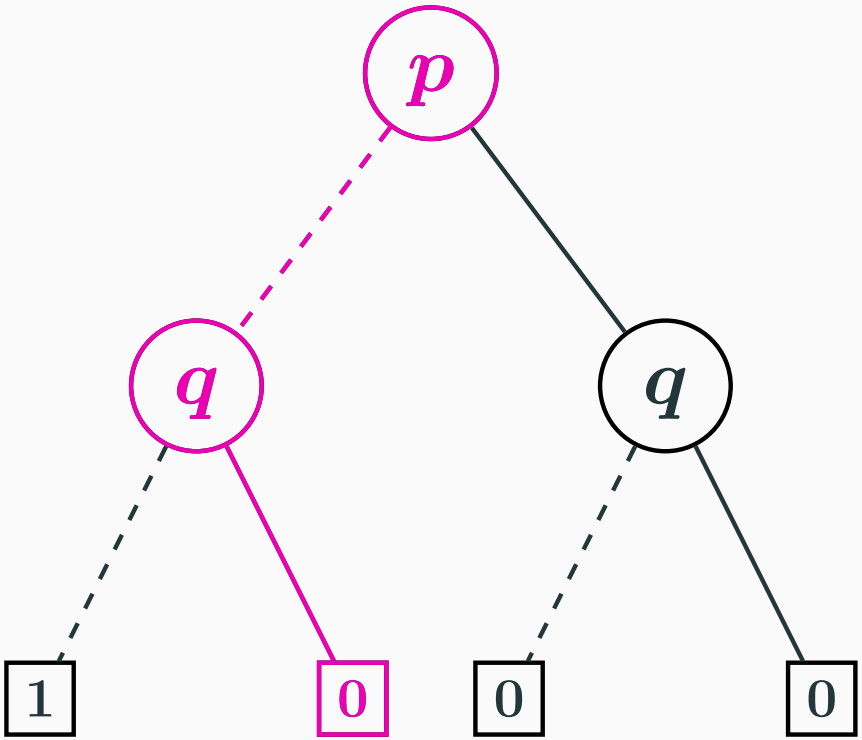
- Árvore da função:
  $\phi \equiv \neg(p \vee q)$
- Para encontrar $[\![\phi]\!]_{v_{(0,1)}}$:
  1. inicia-se pela raiz
  2. como p é 0, segue-se pela linha pontilhada

Por exemplo



- Árvore da função:
  $\phi \equiv \neg(p \vee q)$
- Para encontrar $[\![\phi]\!]_{v_{(0,1)}}$:
  1. inicia-se pela raiz
  2. como p é 0, segue-se pela linha pontilhada
  3. como q é 1, segue-se pela linha sólida

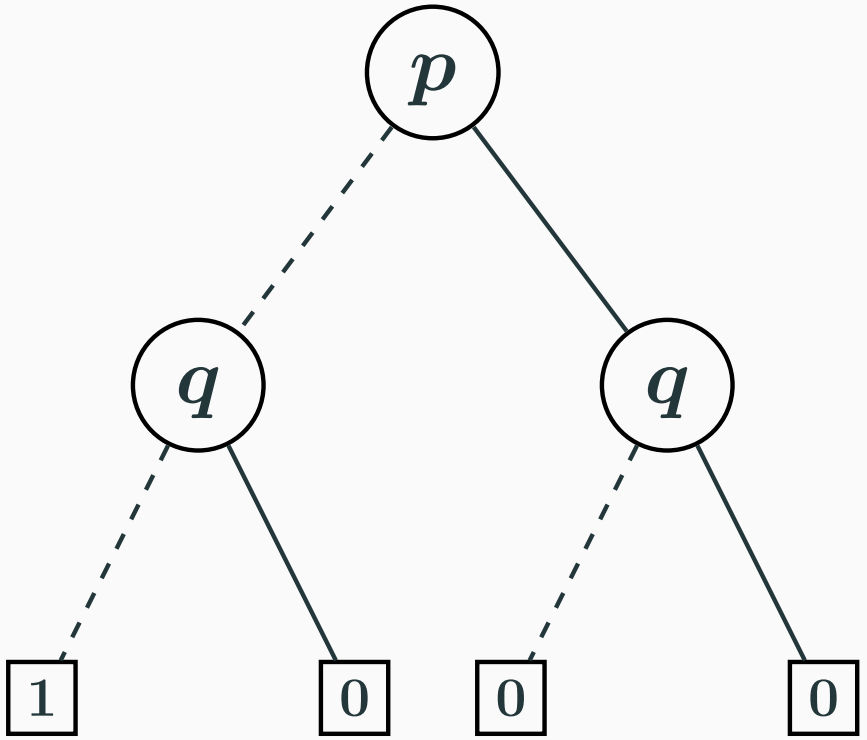
Por exemplo



- Árvore da função:
  $\phi \equiv \neg(p \vee q)$
- Para encontrar $[\![\phi]\!]_{v_{(0,1)}}$:
  1. inicia-se pela raiz
  2. como p é 0, segue-se pela linha pontilhada
  3. como q é 1, segue-se pela linha sólida
  4. chega-se à folha 0; logo $[\![\phi]\!]_{v_{(0,1)}} = 0$

Comparando com a tabela-verdade

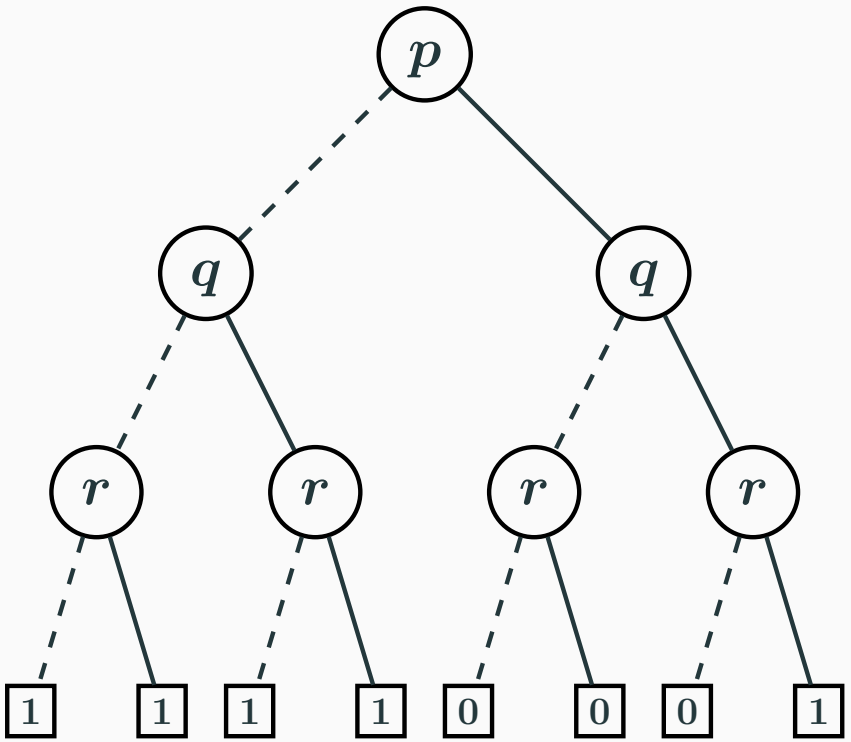
Função booleana: $\phi \equiv \neg(p \vee q)$:



p	q	ϕ
0	0	1
0	1	0
1	0	0
1	1	0

Outro exemplo comparativo

Função booleana: $\psi \equiv p \rightarrow (q \wedge r)$:

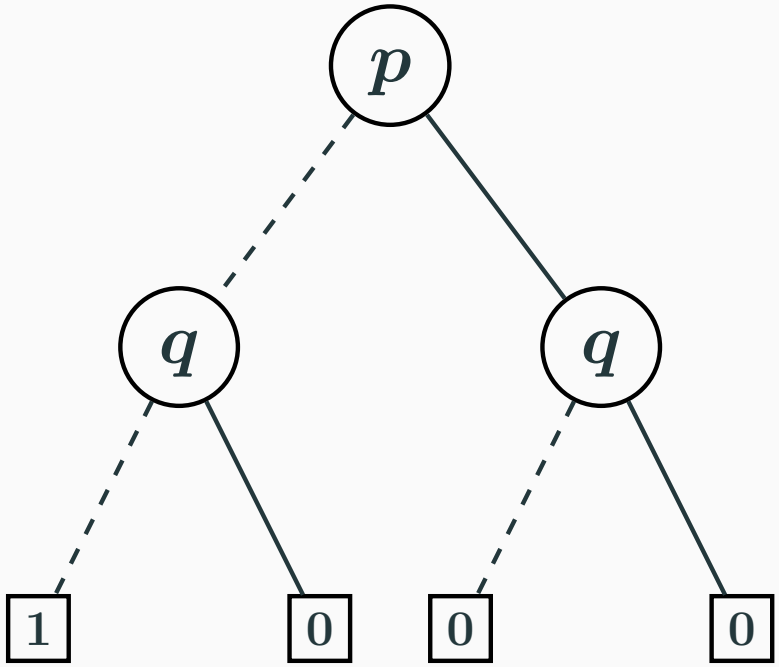


p	q	r	ψ
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

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

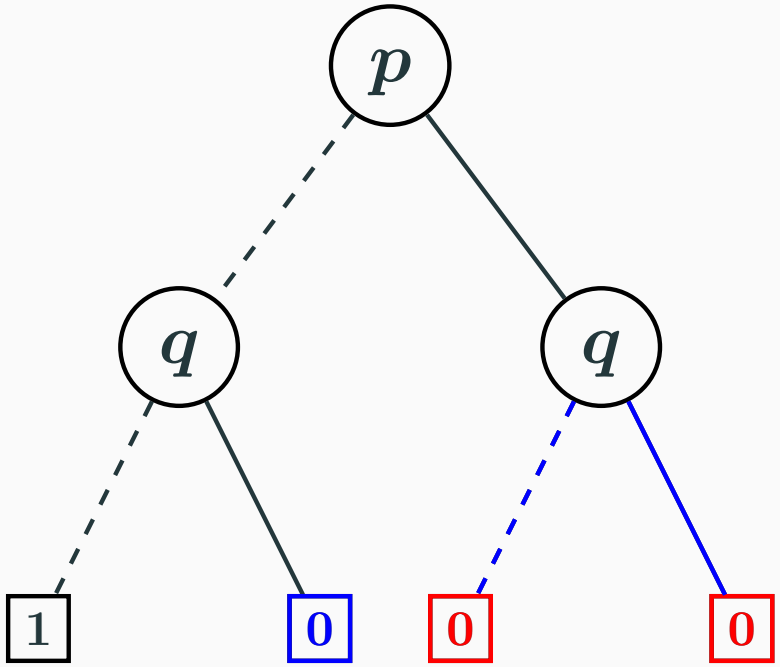
Primeira simplificação



C1. Remoção de terminais duplicados

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

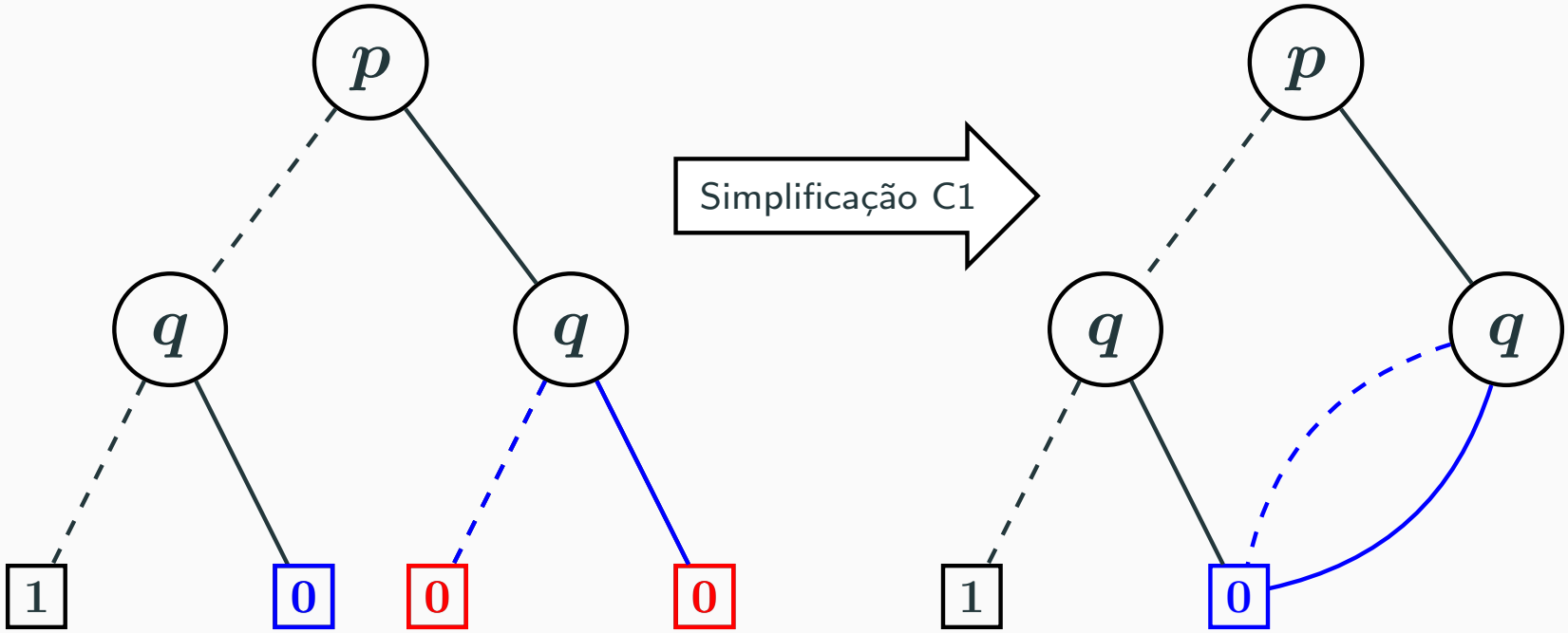
Primeira simplificação



C1. Remoção de terminais duplicados

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

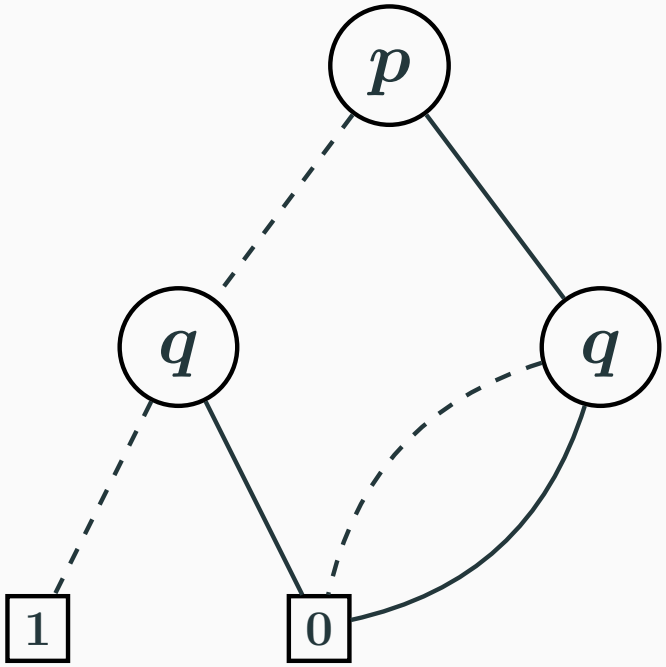
Primeira simplificação



C1. Remoção de terminais duplicados

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

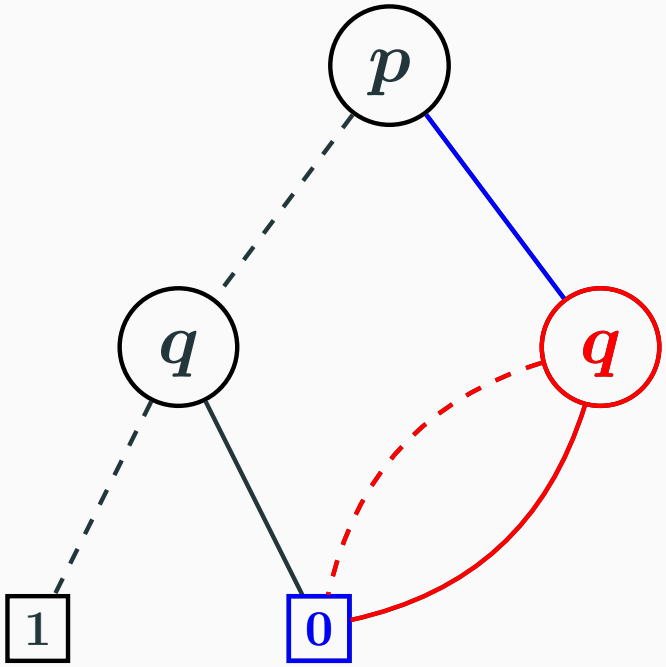
Segunda simplificação



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.

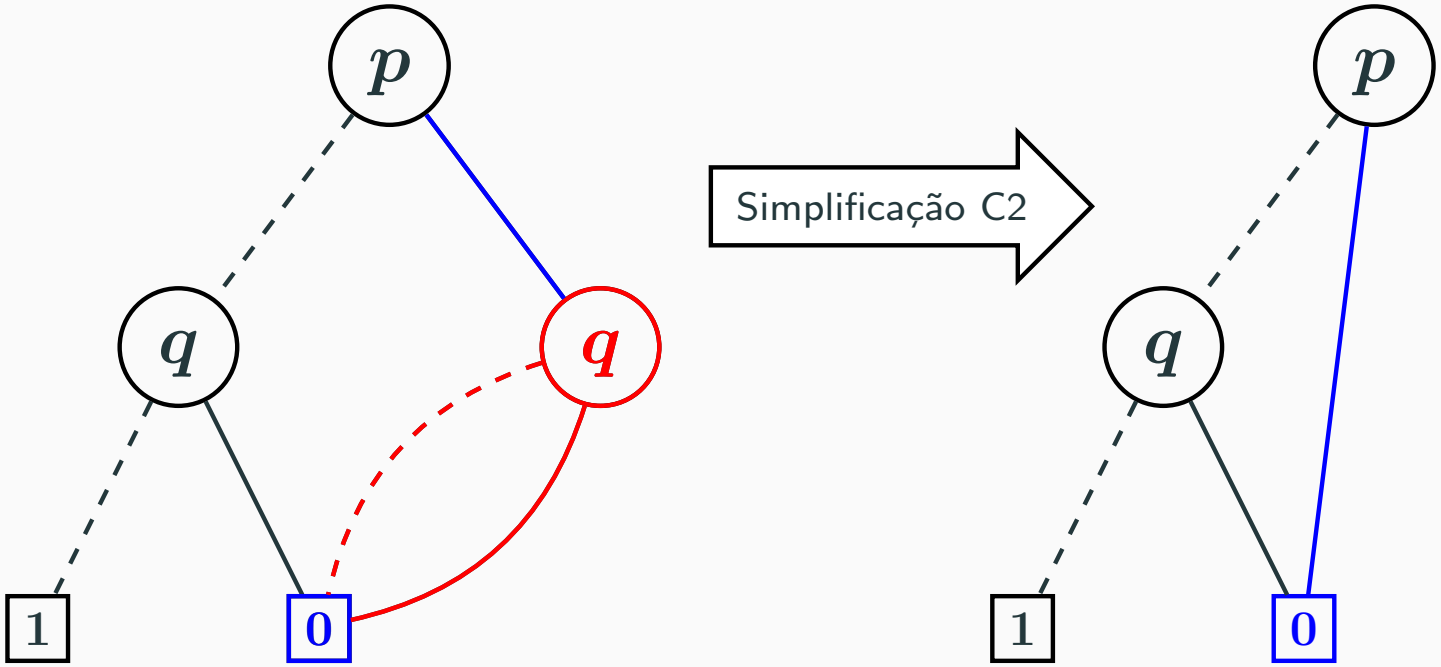
Segunda simplificação



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.

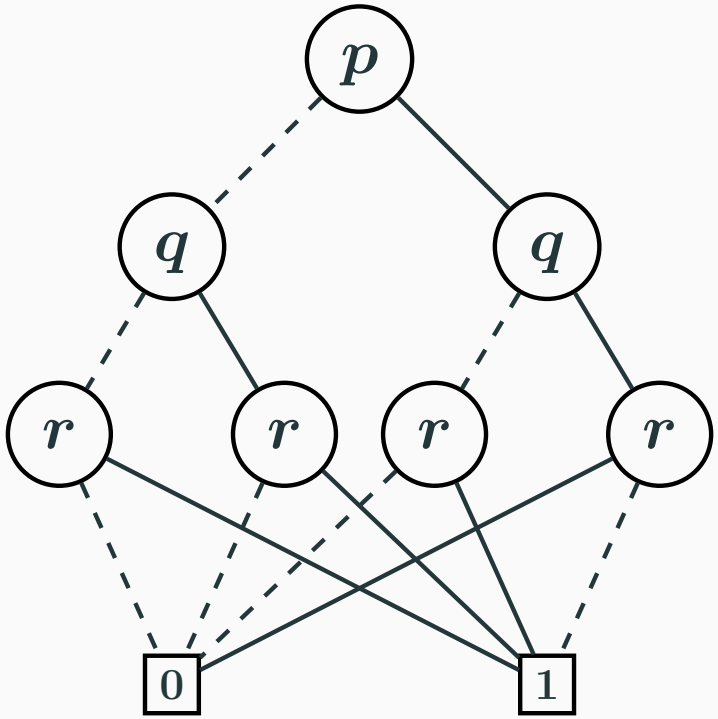
Segunda simplificação



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.

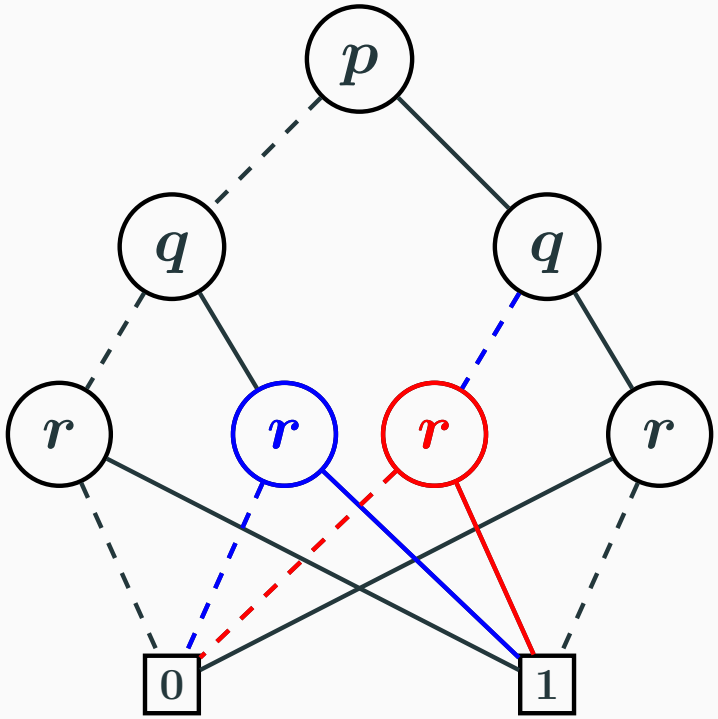
Terceira simplificação



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

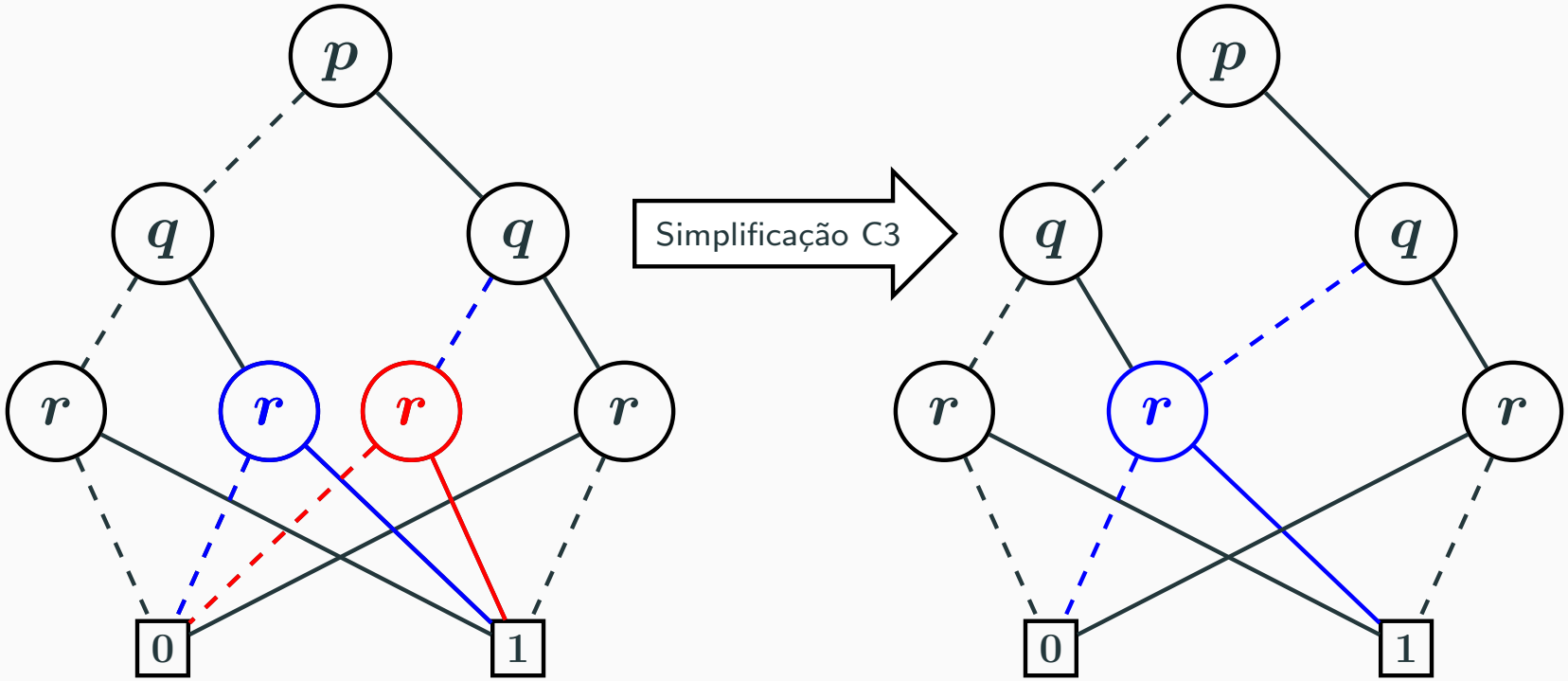
Terceira simplificação



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

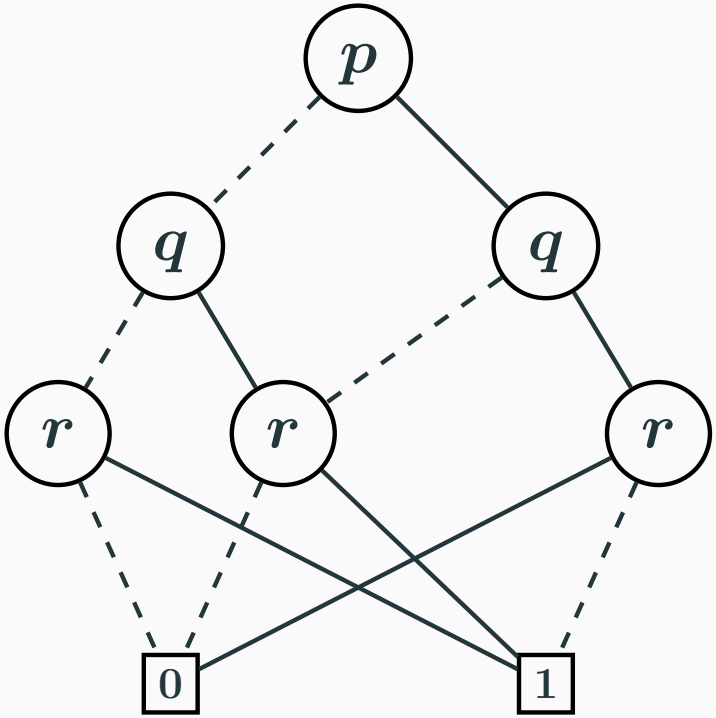
Terceira simplificação



C3. Remoção de nós não-terminais duplicados

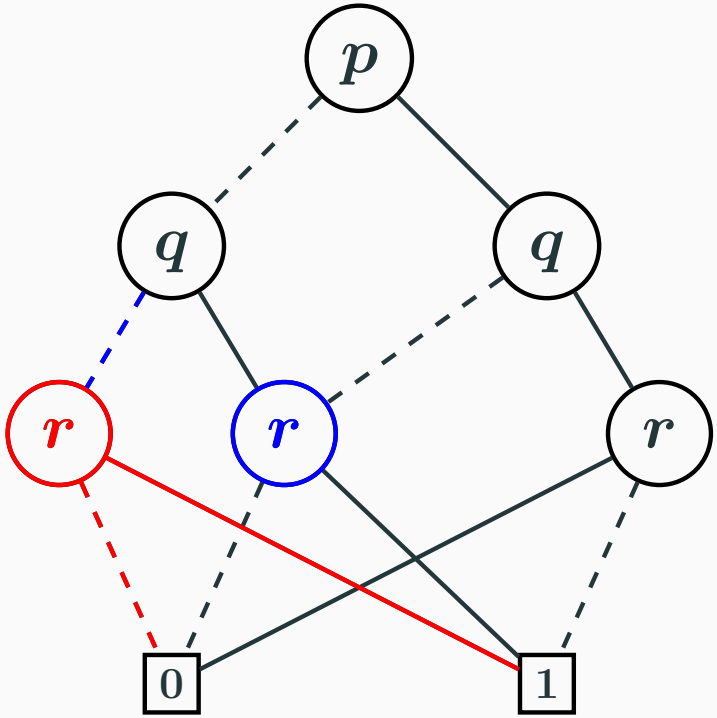
Se dois nós distintos n e m são raízes de subárvores idênticas, um dos nós é eliminado e todas as arestas que nele chegavam são redirecionadas para o outro

Processo de redução



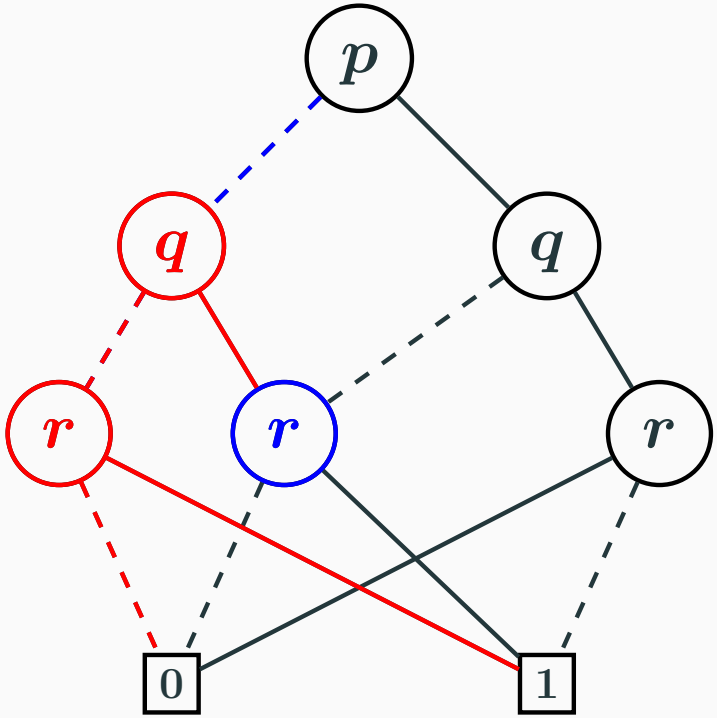
As simplificações são encadeadas até não mais ser possível. O exemplo anterior é completamente reduzido após a eliminação de um dos nós r duplicados (C3) seguida da eliminação de um ponto de decisão q redundante (C2)

Processo de redução



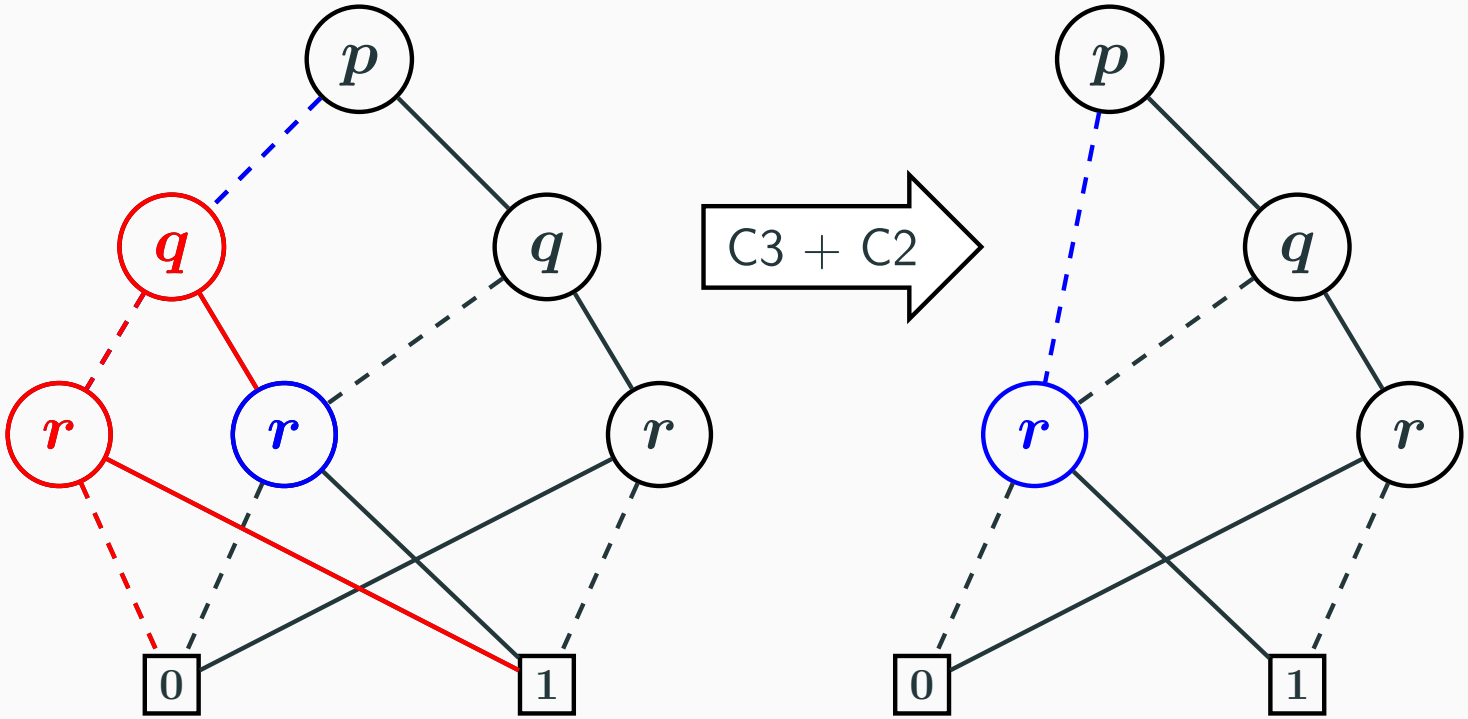
As simplificações são encadeadas até não mais ser possível. O exemplo anterior é completamente reduzido após a eliminação de um dos nós r duplicados (C3) seguida da eliminação de um ponto de decisão q redundante (C2)

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 r duplicados (C3) seguida da eliminação de um ponto de decisão q redundante (C2)

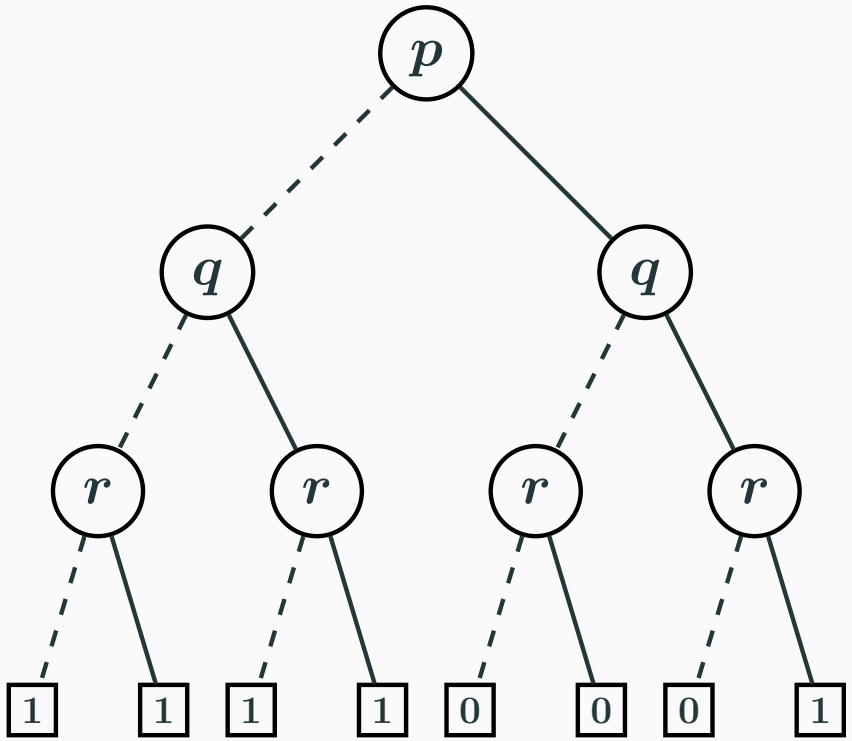
Processo de redução



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

Exercício 1

Reduza a árvore de decisão binária da função $\psi \equiv p \rightarrow (q \wedge r)$ apresentada anteriormente:



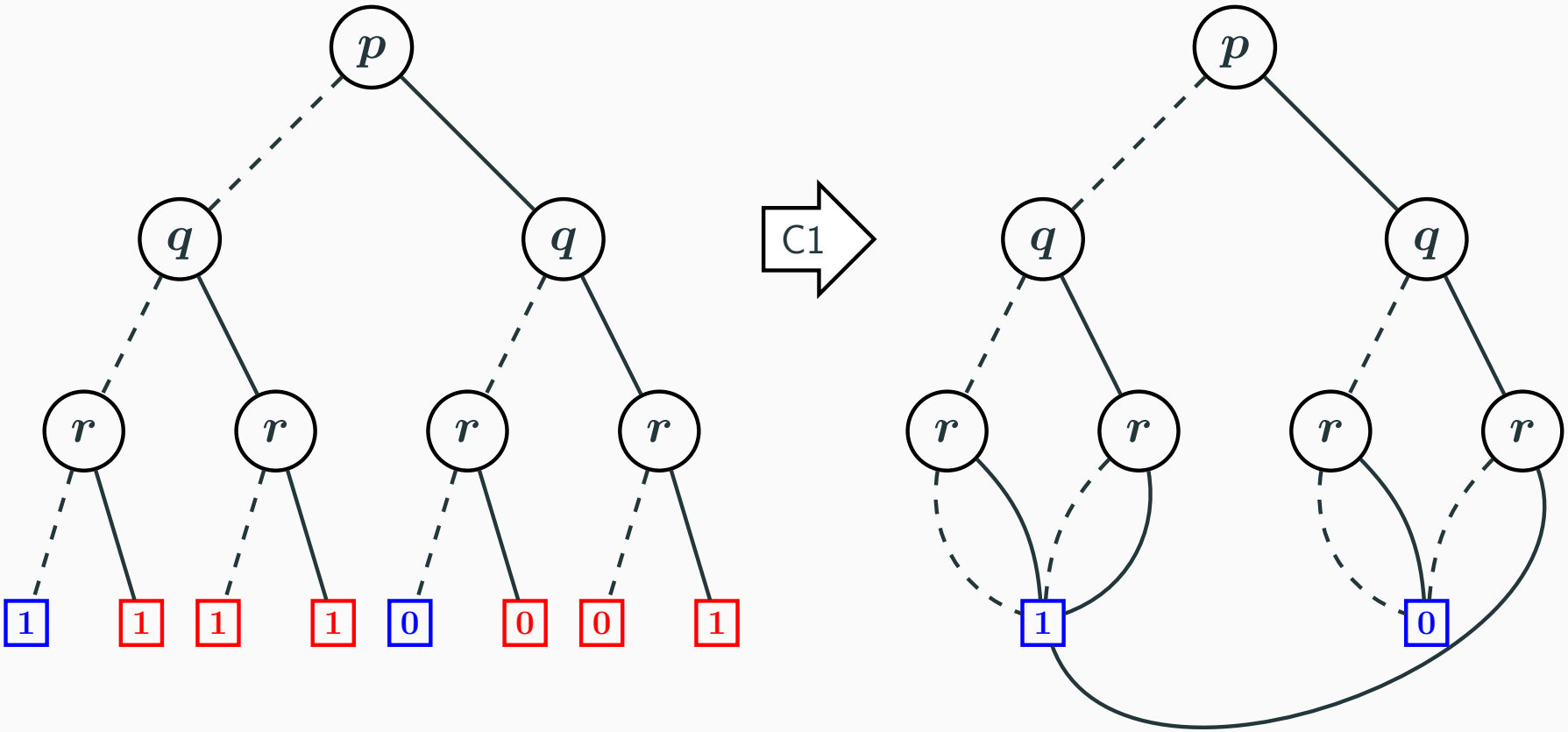
Resumo das simplificações:

C1. Remoção de nós terminais duplicados

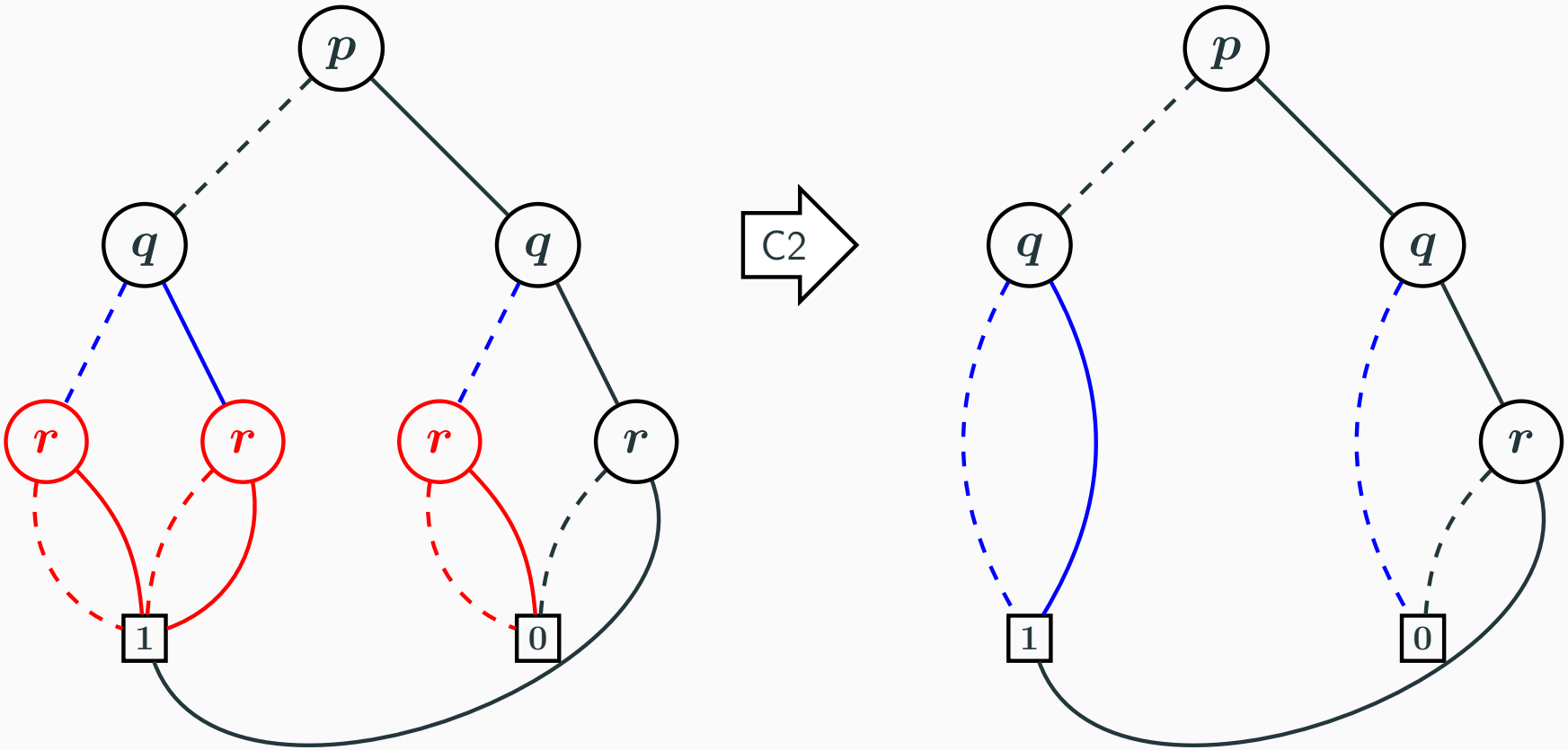
C2. Remoção de testes redundantes

C3. Remoção de nós não-terminais duplicados

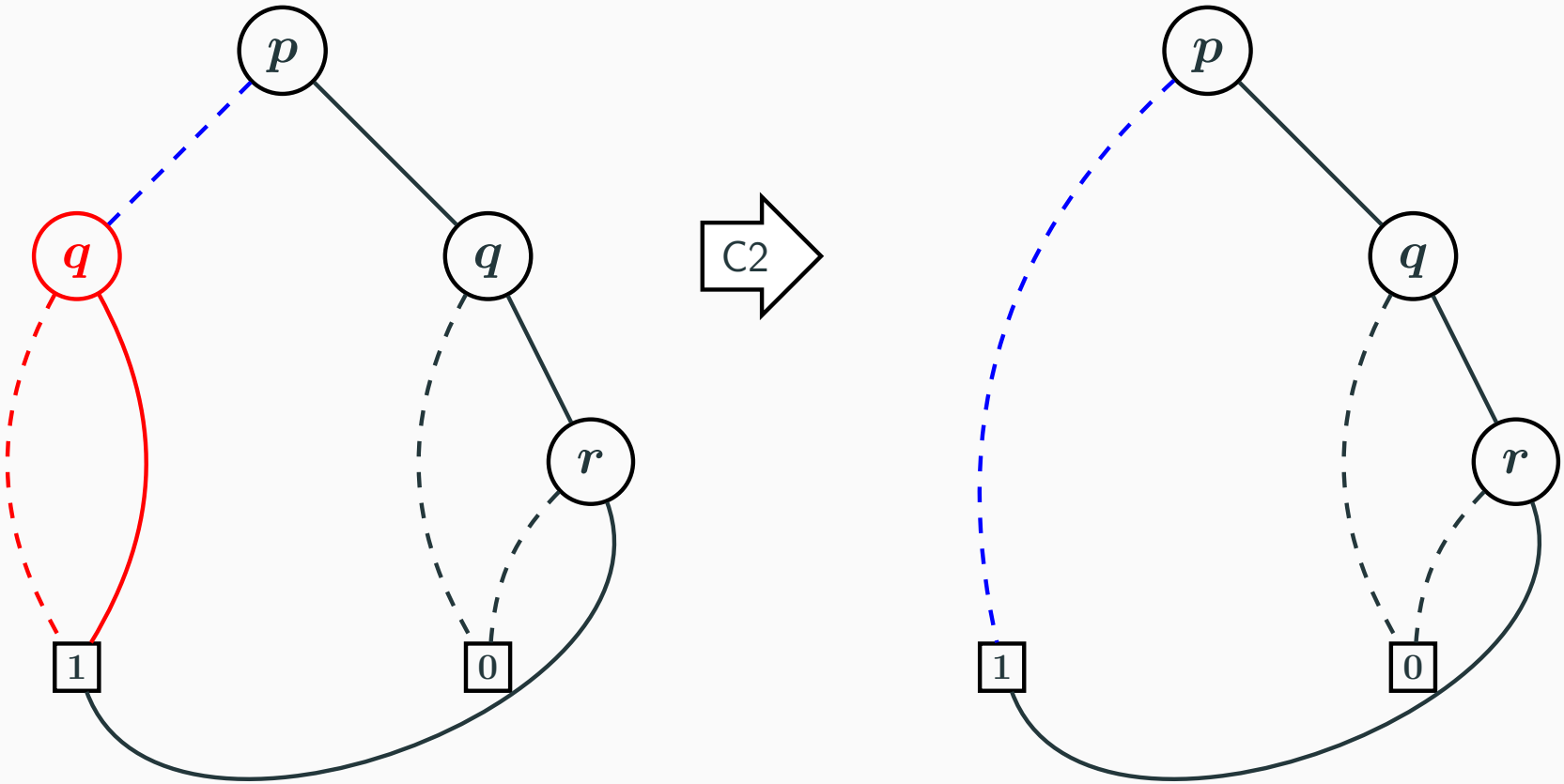
Solução – 1° passo



Solução – 2º passo

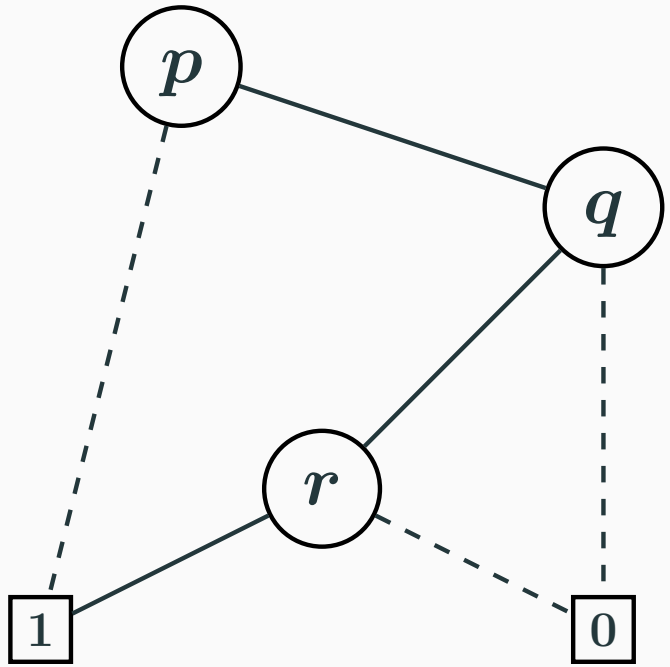


Solução – 3º passo



Comparando com a tabela-verdade

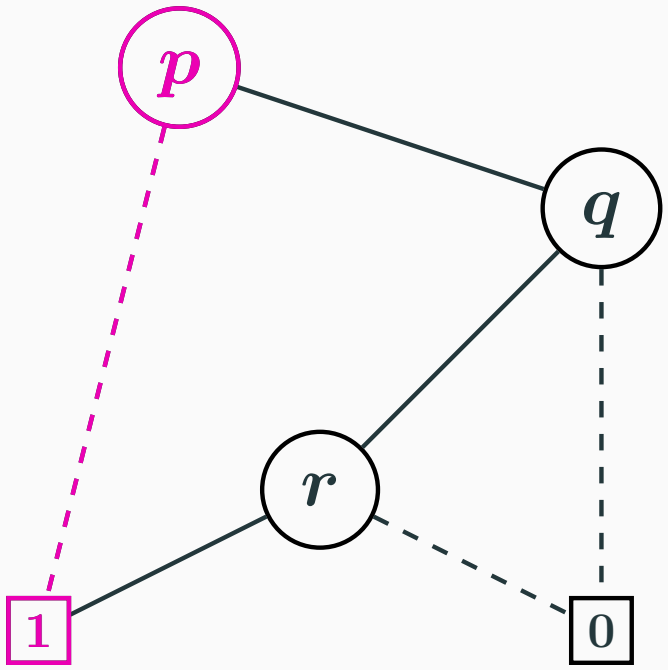
Função booleana: $\psi \equiv p \rightarrow (q \wedge r)$:



p	q	r	ψ
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

Comparando com a tabela-verdade

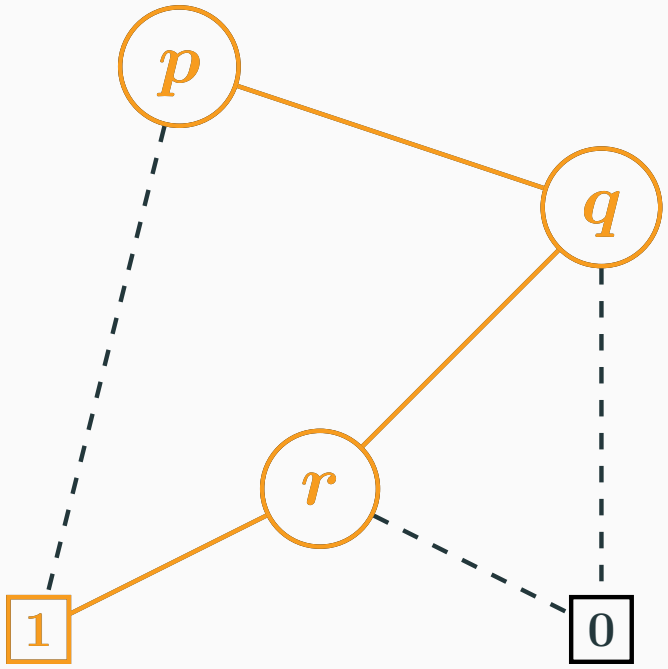
Função booleana: $\psi \equiv p \rightarrow (q \wedge r)$:



p	q	r	ψ
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

Comparando com a tabela-verdade

Função booleana: $\psi \equiv p \rightarrow (q \wedge r)$:



p	q	r	ψ
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 Binária (BDDs)**.

Definição: DAG

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 $v_1 \rightarrow v_2 \rightarrow ... \rightarrow v_n \rightarrow v_1$. Um grafo direcionado acíclico (DAG) é um grafo direcionado que não contém nenhum ciclo. Um nó em um DAG 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 DAG finito com um único nó inicial, onde todos os nós terminais são marcados com $\mathbf{0}$ ou $\mathbf{1}$ 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 $\mathbf{0}$ e outra com $\mathbf{1}$ (representadas como uma linha pontilhada e uma linha sólida, respectivamente).

BDD como DAG

- 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 (DAG) e têm um único nó inicial
- As simplificações C1–C3 preservam essas propriedades
  - BDDs totalmente reduzidos têm 1 ou 2 nós terminais

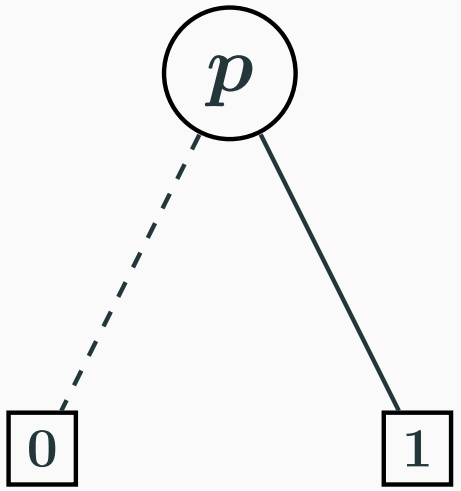
BDDs elementares

- O BDD B_0 representa a função booleana constante 0
- O BDD B_1 representa a função booleana constante 1
- O BDD B_p representa a variável booleana p

B_0
$(\phi \equiv \bot)$

B_1
$(\phi \equiv \top)$

B_p
$(\phi \equiv p)$



Verificações sobre BDDs

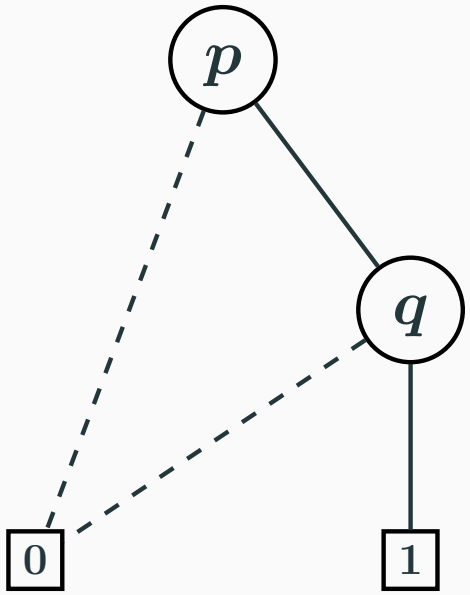
- **Satisfação**. Um BDD representa uma função que pode ser satisfeita se um nó terminal $\mathbf{1}$ pode ser acessado da raiz por meio de um caminho consistente

- **Validade**. Um BDD representa uma função válida se nenhum ponto terminal $\mathbf{0}$ é acessível por um caminho consistente

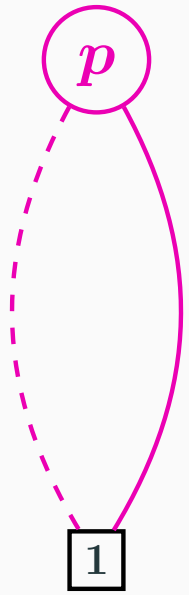
Um caminho consistente é aquele que, iniciado no nó raiz, segue apenas por uma valoração possível para cada variável booleana e atinge um único nó-terminal com valor $\mathbf{0}$ ou $\mathbf{1}$

Exemplos óbvios

$\phi \equiv p \wedge q$



$\psi \equiv p \vee \neg p$



$\omega \equiv q \wedge \neg q$

Comparação das representações

Considere a função de paridade par $f_{par}(p_1, p_2, ..., p_n)$ que é definida como 1 se existe um número par de variáveis p_i com valor 1, e como 0 caso contrário.

Bit de paridade (par ou ímpar) é uma das formas mais simples de detecção de erros na comunicação de dados

- Ela tem representação exponencial em outros sistemas (wffs ou tabelas-verdade, por exemplo)
- Enquanto que um BDD precisa de apenas $2n + 1$ nós para representá-la

Ilustração: tabela-verdade para $n = 4$

p_1	p_2	p_3	p_4	ϕ
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

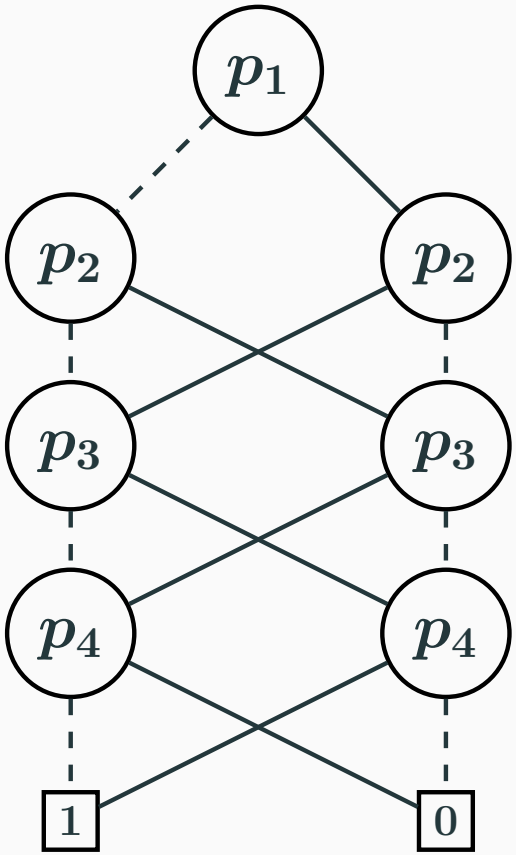
- Total de linhas: $2^n = 2^4 = 16$

Ilustração: wff para $n = 4$

$$\phi \equiv \neg(((p_1 \oplus p_2) \oplus p_3) \oplus p_4)$$

- Lembrando do ou-exclusivo:
  - $(p \oplus q) \equiv ((p \vee q) \wedge \neg(p \wedge q))$
- Número de símbolos: $14(n-1)+1 = 43$

Ilustração: BDD para $n = 4$



- Total de nós: $2n + 1 = 9$

Operações sobre BDDs

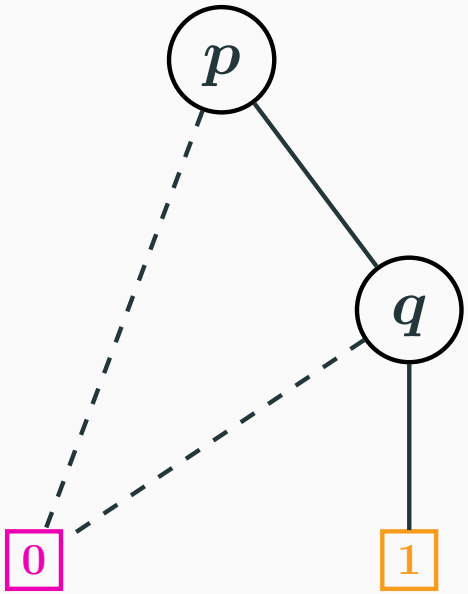
- **Operação de negação ($\neg$).** Obtem-se um BDD que representa $\neg\phi$ substituindo todos os terminais $\mathbf{0}$ em $\boldsymbol{B_\phi}$ por terminais $\mathbf{1}$ e vice-versa

- **Operação de conjunção ($\wedge$).** Obtem-se um BDD que representa $\phi \wedge \psi$ substituindo todos os nós terminais $\mathbf{1}$ em $\boldsymbol{B_\phi}$ diretamente por $\boldsymbol{B_\psi}$

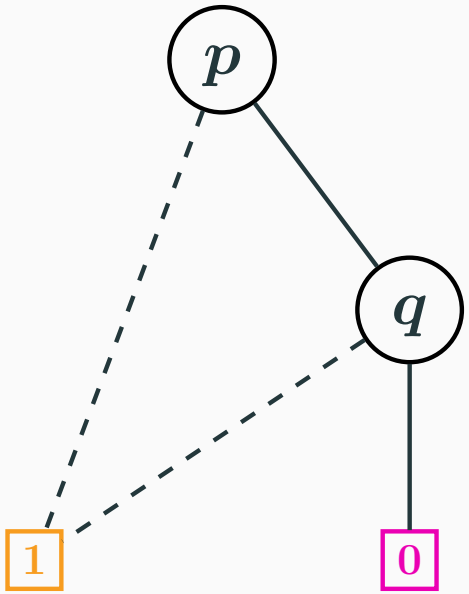
- **Operação de disjunção ($\vee$).** Obtem-se um BDD que representa $\phi \vee \psi$ substituindo todos os nós terminais $\mathbf{0}$ em $\boldsymbol{B_\phi}$ diretamente por $\boldsymbol{B_\psi}$

Exemplo da negação

$\phi \equiv p \wedge q$

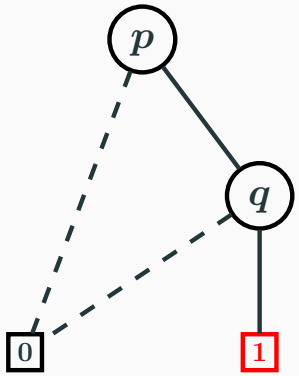


$\neg\phi \equiv \neg(p \wedge q)$

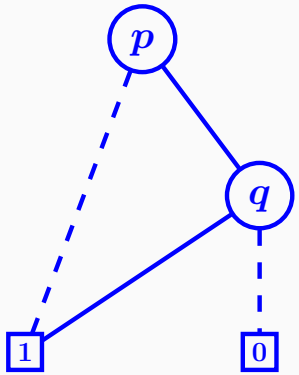


Exemplo da conjunção

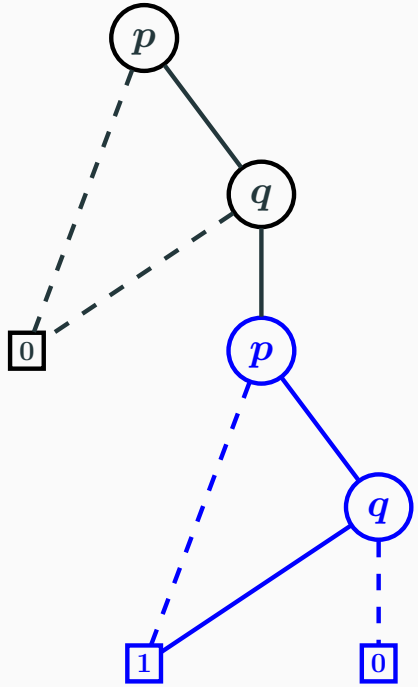
$$\phi \equiv p \wedge q$$



$$\psi \equiv \neg p \vee q$$

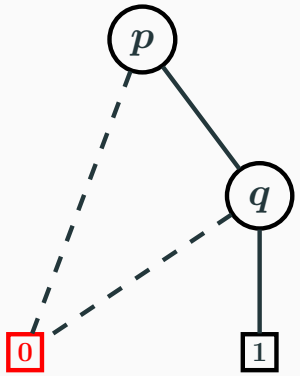


$$\phi \wedge \psi \equiv (p \wedge q) \wedge (\neg p \vee q)$$

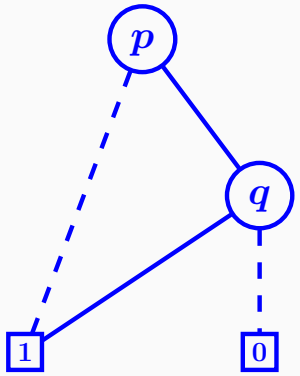


Exemplo da disjunção

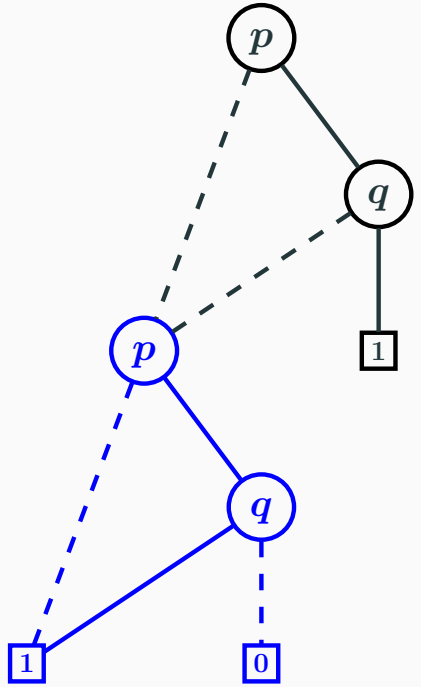
$$\phi \equiv p \wedge q$$



$$\psi \equiv \neg p \vee q$$



$$\phi \vee \psi \equiv (p \wedge q) \vee (\neg p \vee q)$$



Forma “inocente” de construir BDDs

1. Para cada variável booleana em uma função, um BDD de variável ($\boldsymbol{B}_{p_i}$) é criado

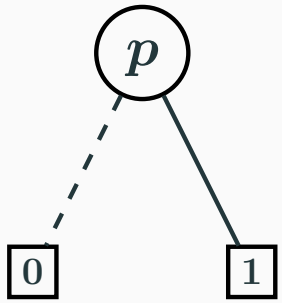
2. Tais BDDs são então unidos conforme as operações booleanas na função

3. Por fim, o BDD resultante é reduzido com as simplificações C1-C3

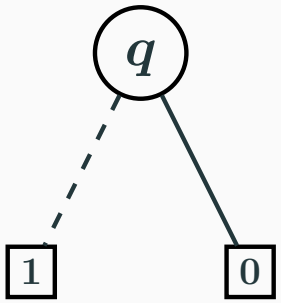
Exemplo: $(p \wedge \neg q) \vee (\neg p \wedge q)$

Passo 1: criação de B_{p_i}

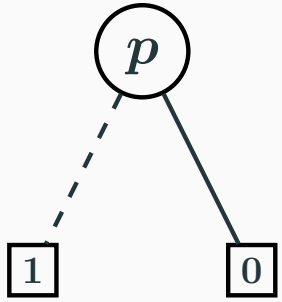
B_p



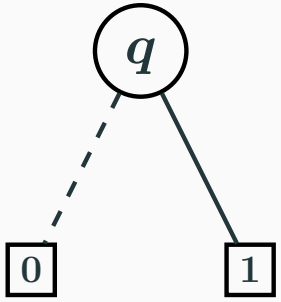
$B_{\neg q}$



$B_{\neg p}$



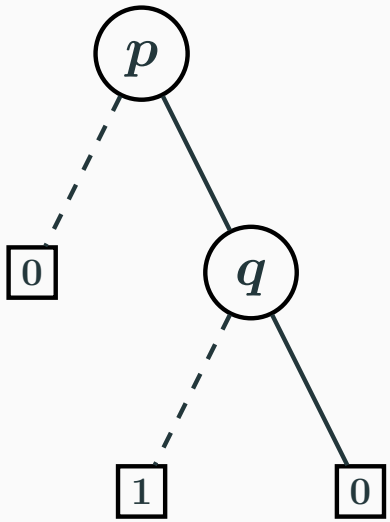
B_q



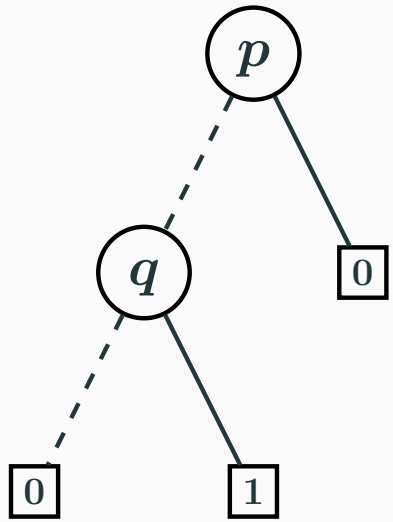
Exemplo: $(p \wedge \neg q) \vee (\neg p \wedge q)$

Passo 2a: união dos BDDs conforme as operações

$B_p \wedge B_{\neg q}$



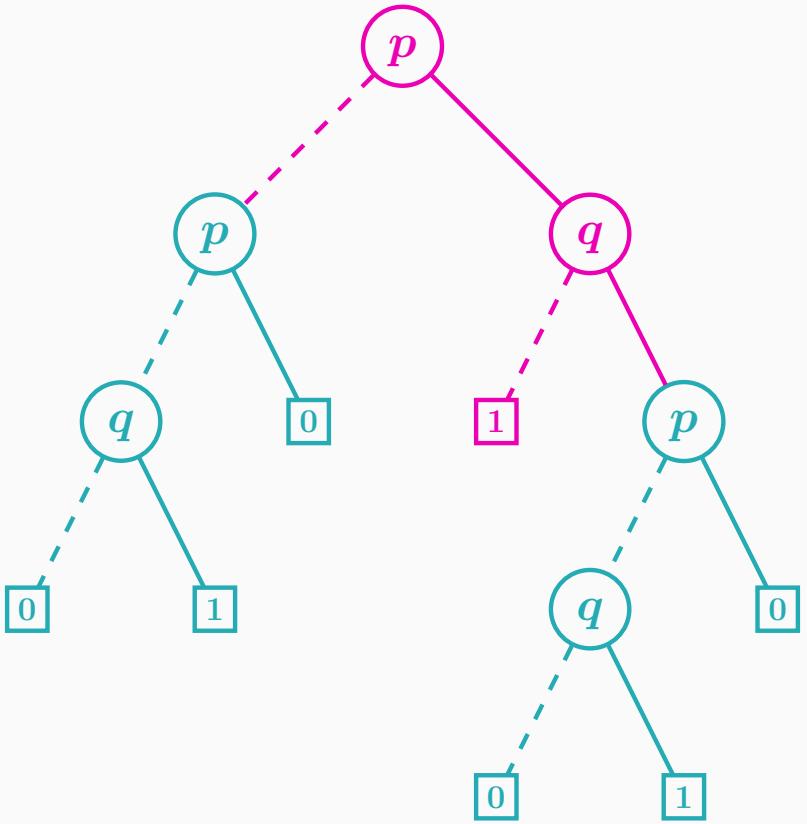
$B_{\neg p} \wedge B_q$



Exemplo: $(p \wedge \neg q) \vee (\neg p \wedge q)$

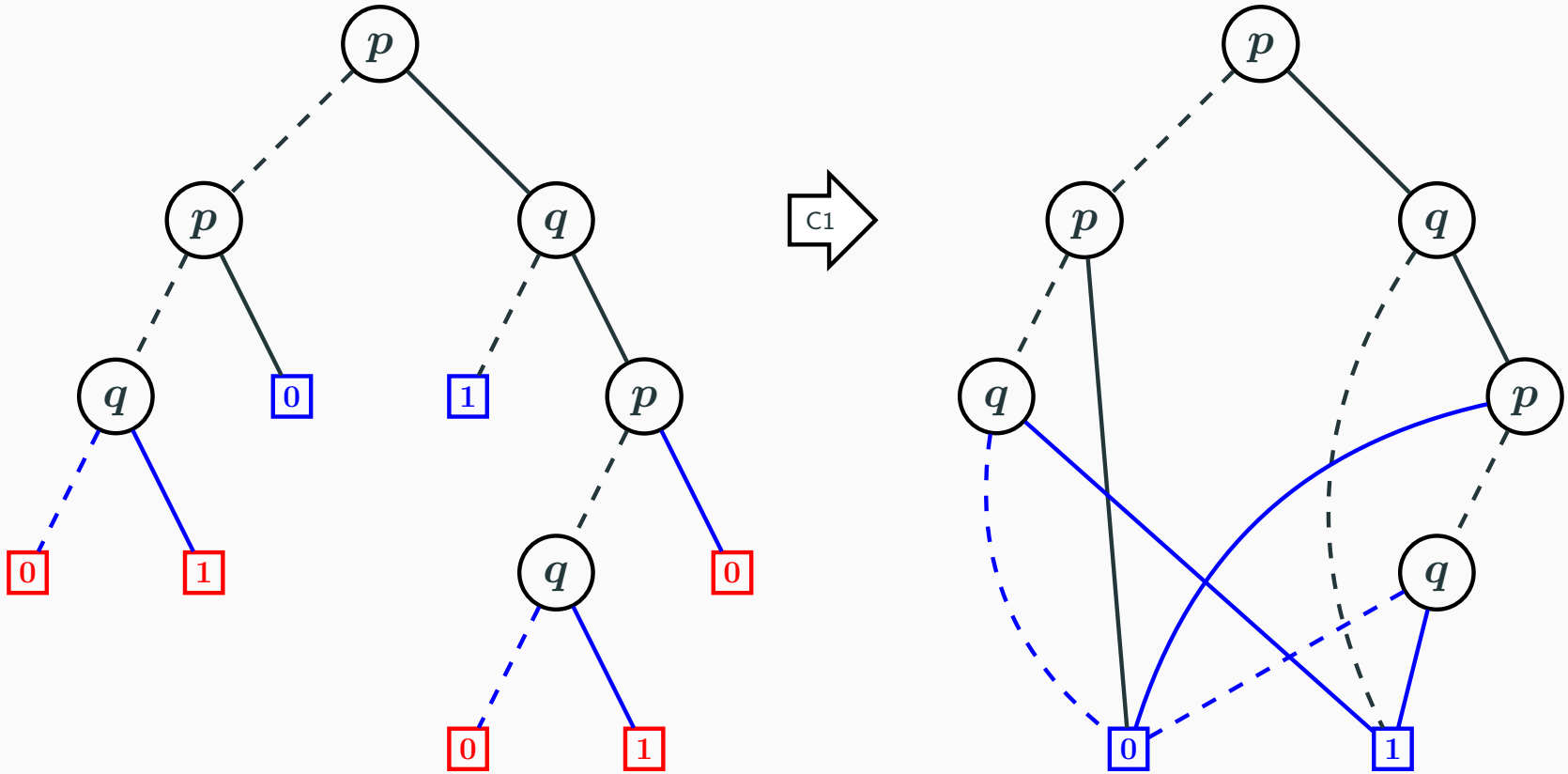
Passo 2b: união dos BDDs conforme as operações

$(B_p \wedge B_{\neg q}) \vee (B_{\neg p} \wedge B_q)$



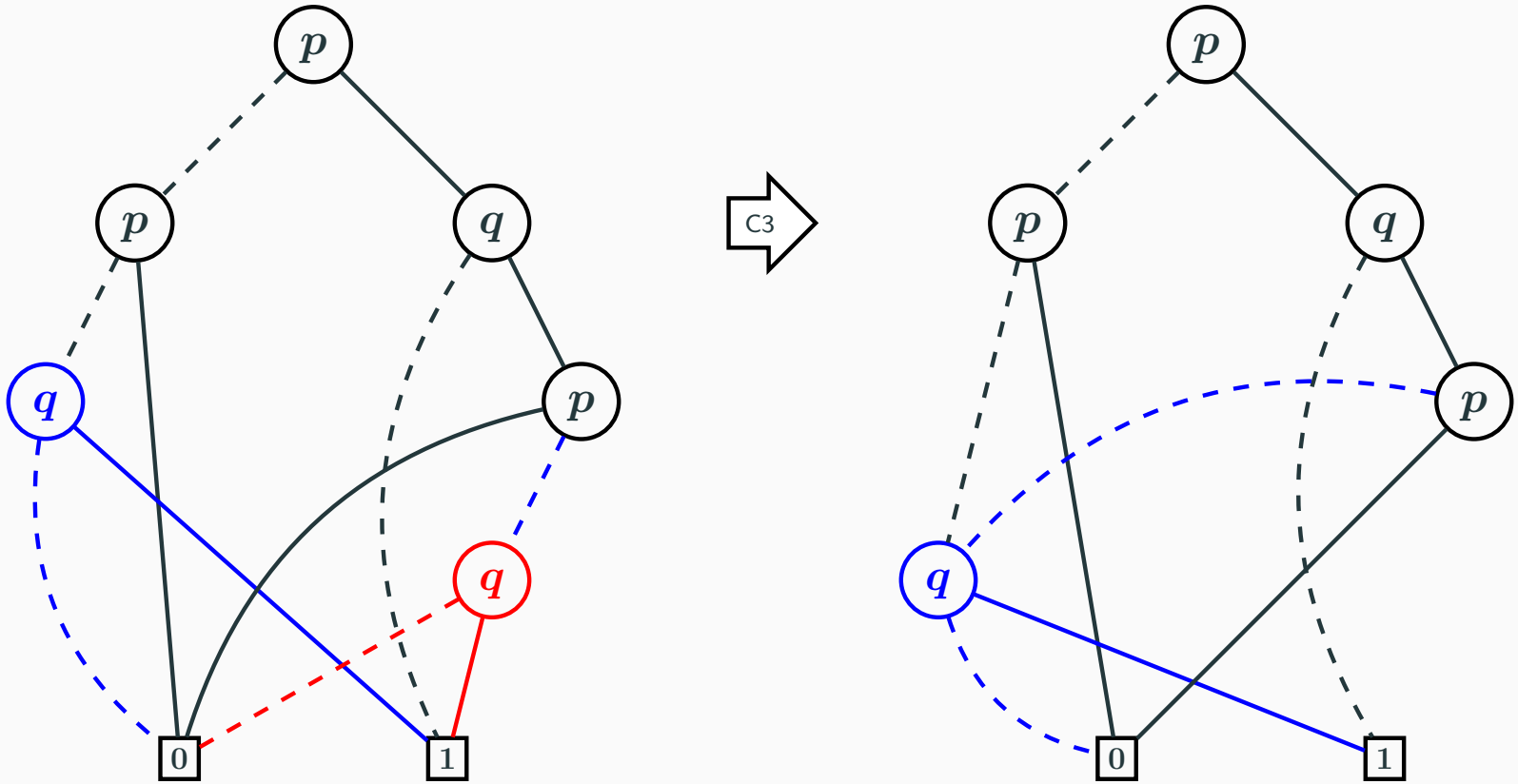
Exemplo: $(p \wedge \neg q) \vee (\neg p \wedge q)$

Passo 3a: redução do BDD gerado



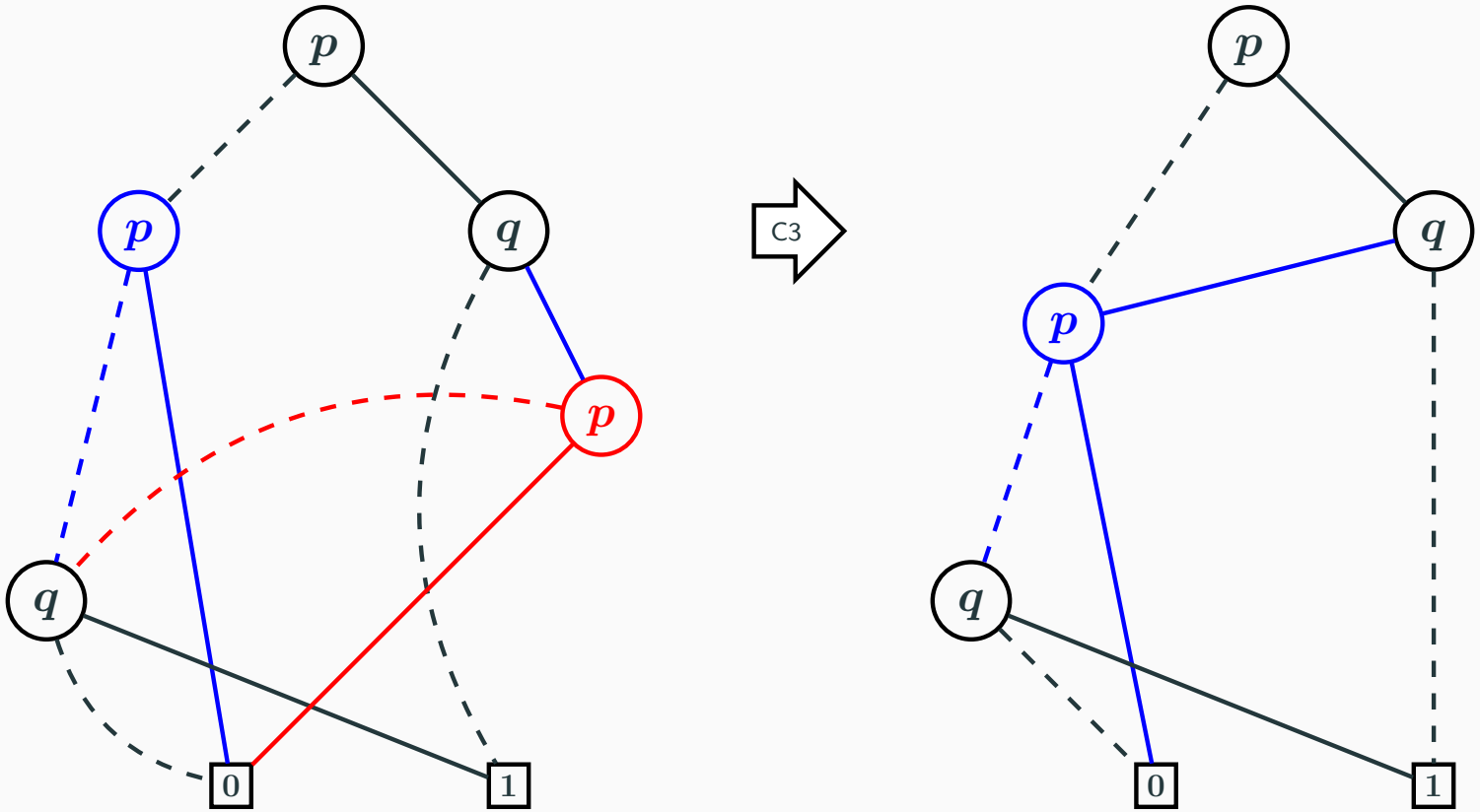
Exemplo: $(p \wedge \neg q) \vee (\neg p \wedge q)$

Passo 3b: redução do BDD gerado



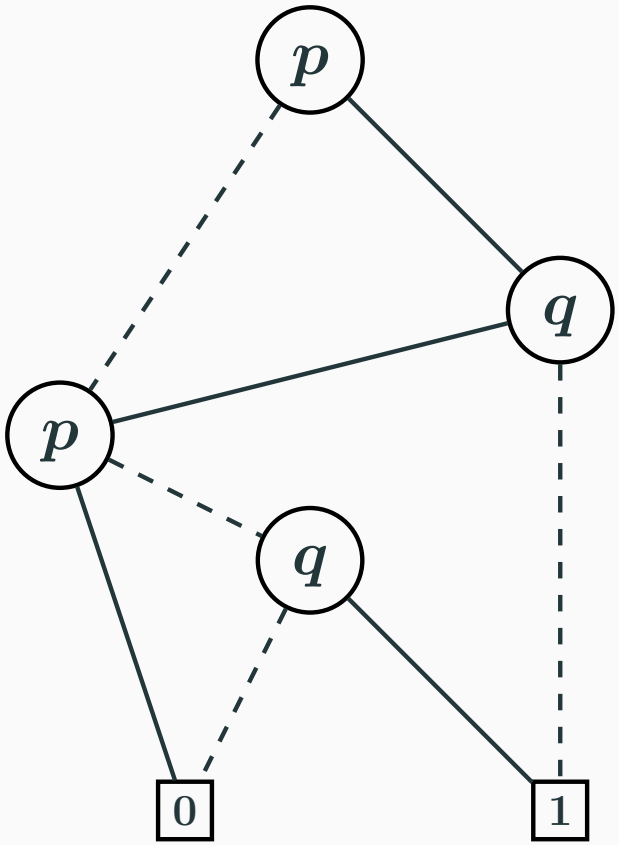
Exemplo: $(p \wedge \neg q) \vee (\neg p \wedge q)$

Passo 3c: redução do BDD gerado



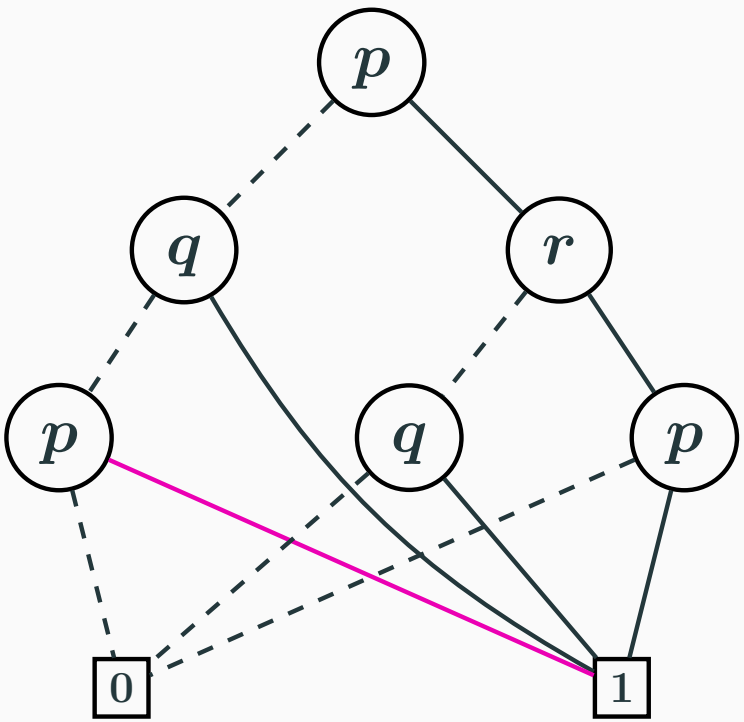
Comparação com a tabela-verdade

$$\phi \equiv (p \wedge \neg q) \vee (\neg p \wedge q)$$



p	q	ϕ
0	0	0
0	1	1
1	0	1
1	1	0

Múltiplas ocorrências de mesma variável



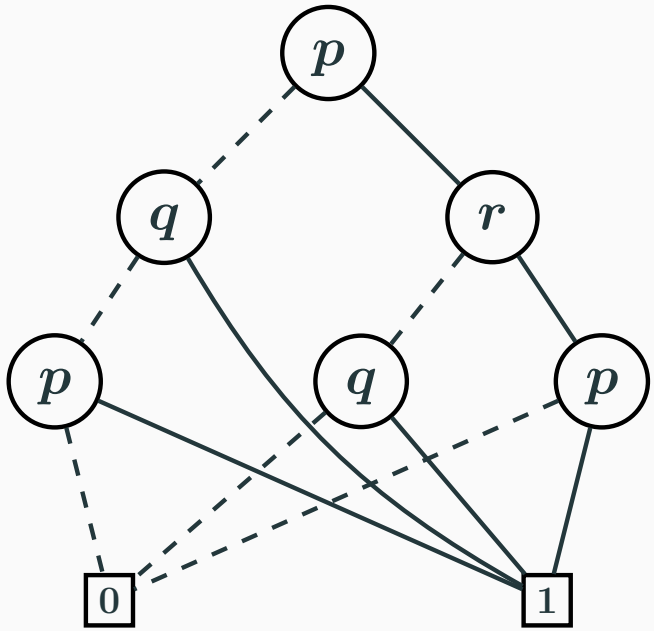
- A definição não impede uma variável de ocorrer mais de uma vez em um caminho
- Mas tal representação pode incorrer em desperdícios
  - linha sólida do p à esquerda (colorida) jamais será percorrida

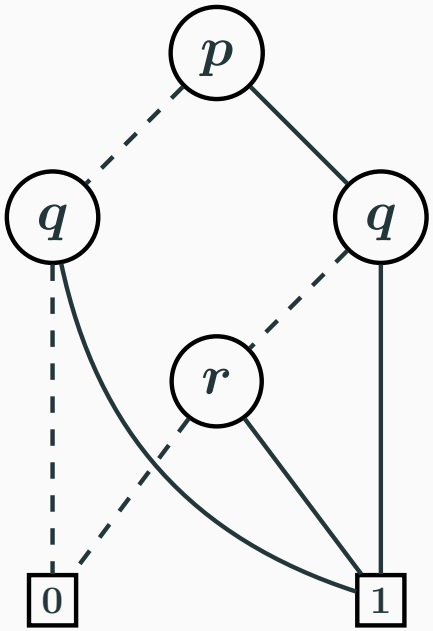
Esse é um resultado comum após as operações discutidas anteriormente – algoritmos melhores serão apresentados posteriormente

Comparação de BDDs

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

Exercício: Os BDDs abaixo são equivalentes?





Na próxima aula...

- BDDs ordenados
- Algoritmos para BDDs ordenados