Diagramas de Decisão Binária (BDDs)

Aula 2

Luiz Carlos Vieira

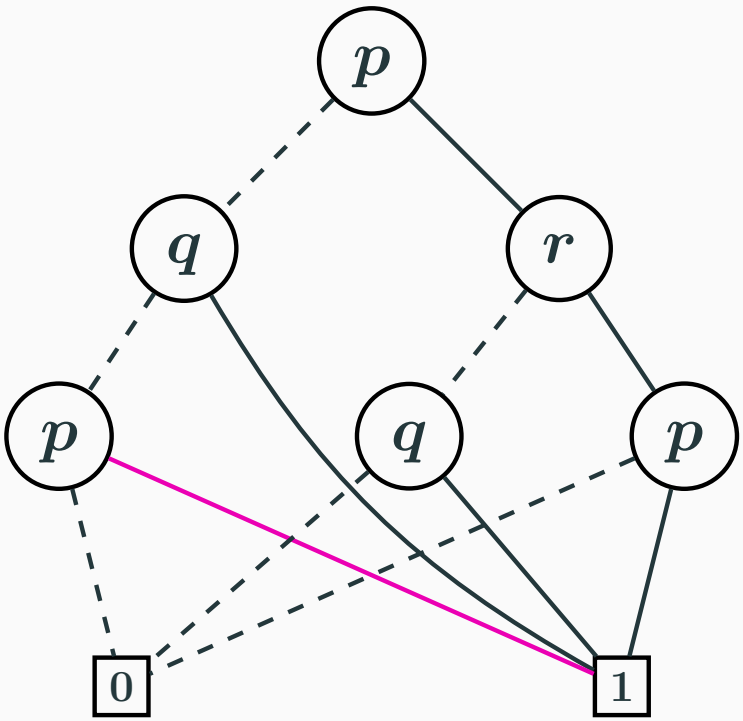
7 de outubro de 2015

MAC0239 - Introdução à Lógica e Verificação de Programas

Conteúdo

- BDDs ordenados e reduzidos (ROBDDs)
- Algoritmos para ROBDDs
  - algoritmo `reduzir`
  - algoritmo `aplicar`
  - algoritmo `restringir`
  - algoritmo `existe`

Relembrando: múltiplas ocorrências

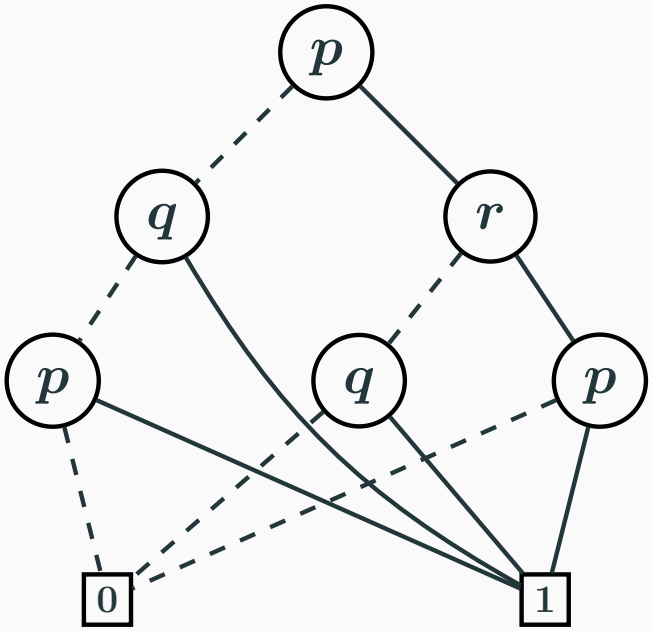


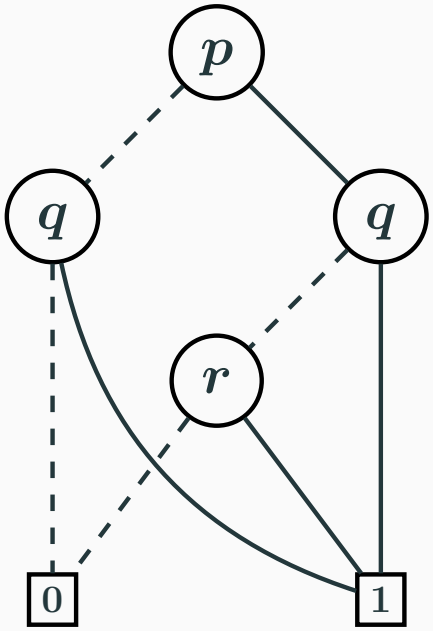
- A definição de BDDs não impede uma variável de ocorrer mais de uma vez em um caminho
- Mas tal representação pode incorrer em desperdícios
  - linha sólida do p à esquerda (colorida) jamais será percorrida

Esse é um resultado comum após as operações discutidas na aula anterior

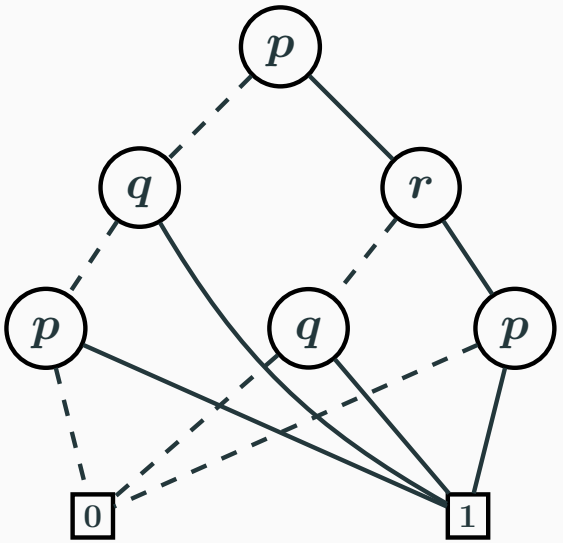
Relembrando: comparação de BDDs

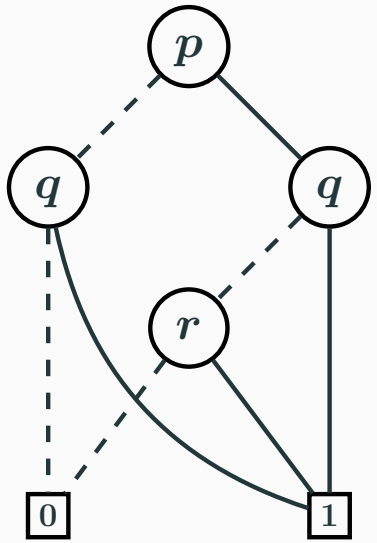
Além de tornar um BDD menos eficiente, ocorrências múltiplas de uma variável também dificultam a comparação de BDDs





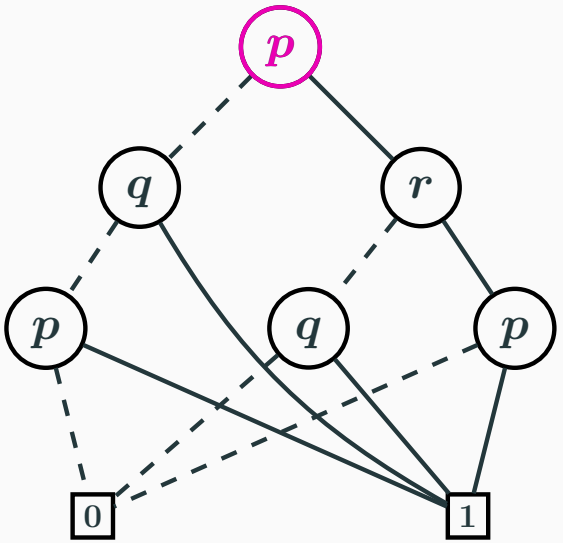
Conceito de ordem de um caminho

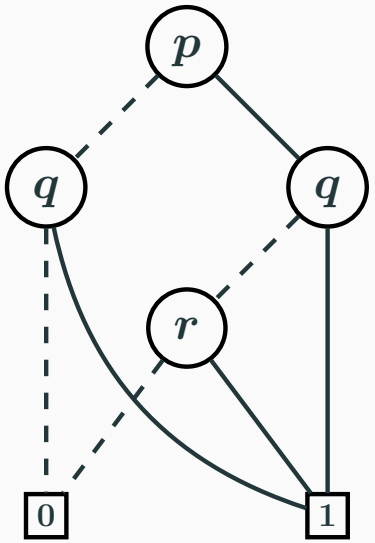




Conceito de ordem de um caminho

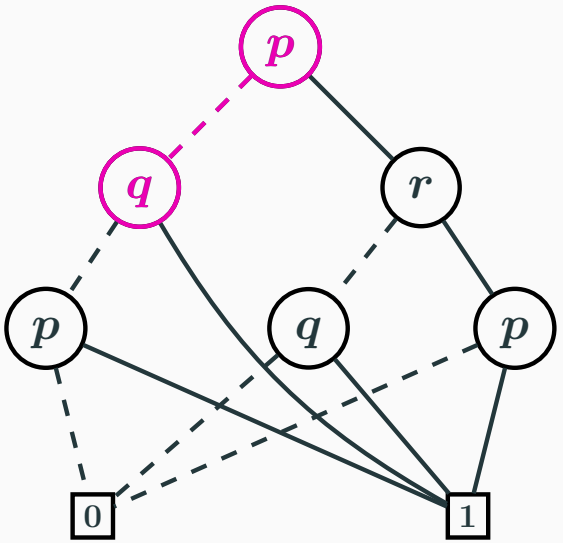
- $[p \qquad]$

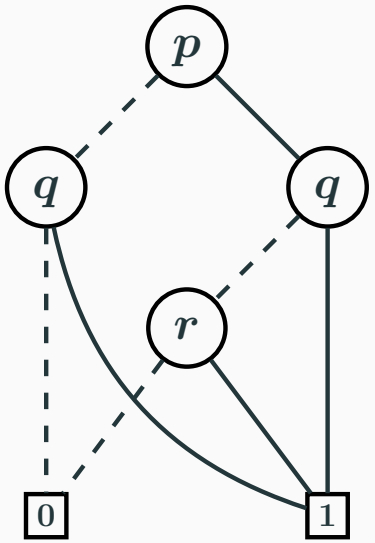




Conceito de ordem de um caminho

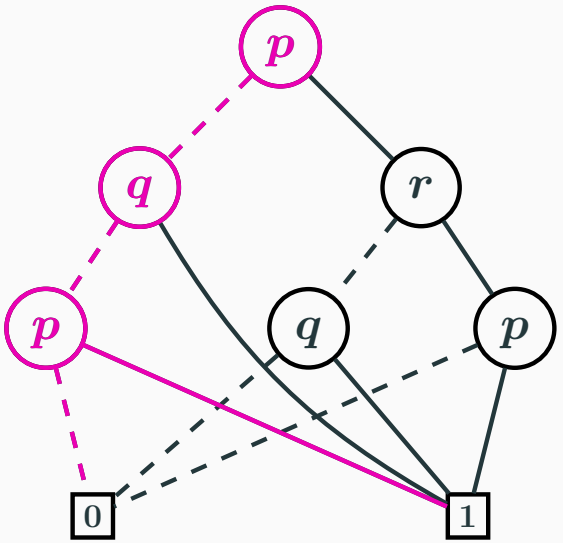
- $[p, q \quad]$

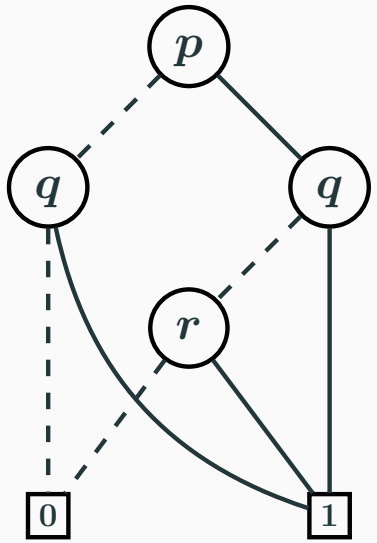




Conceito de ordem de um caminho

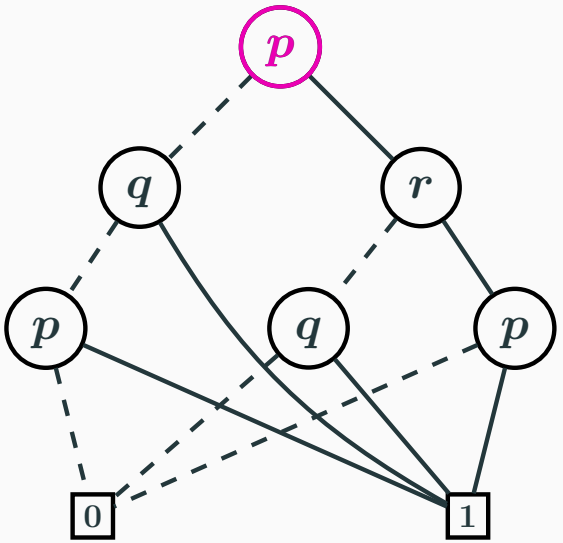
- $[p, q, p]$

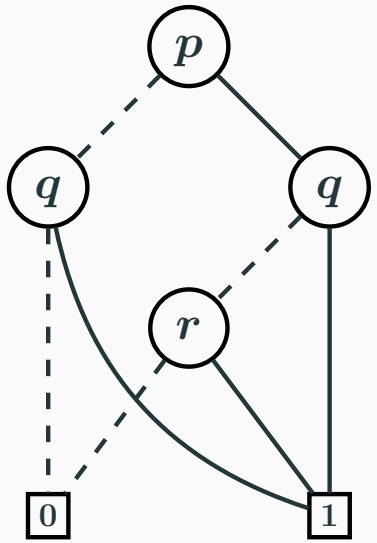




Conceito de ordem de um caminho

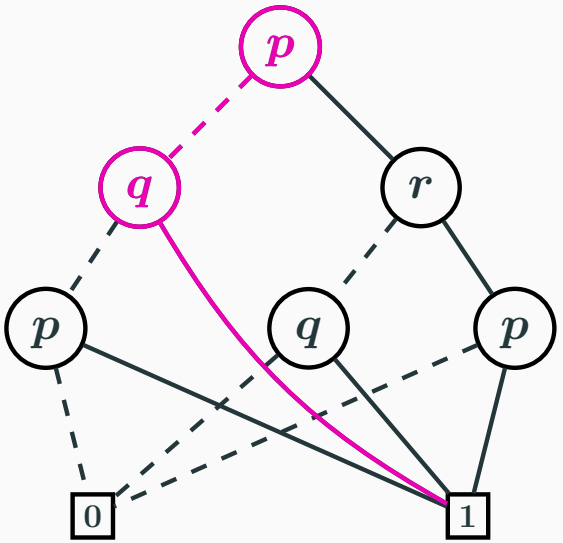
- $[p, q, p]$
- $[p \quad]$

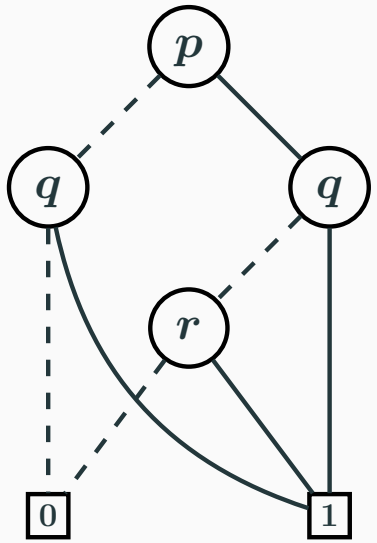




Conceito de ordem de um caminho

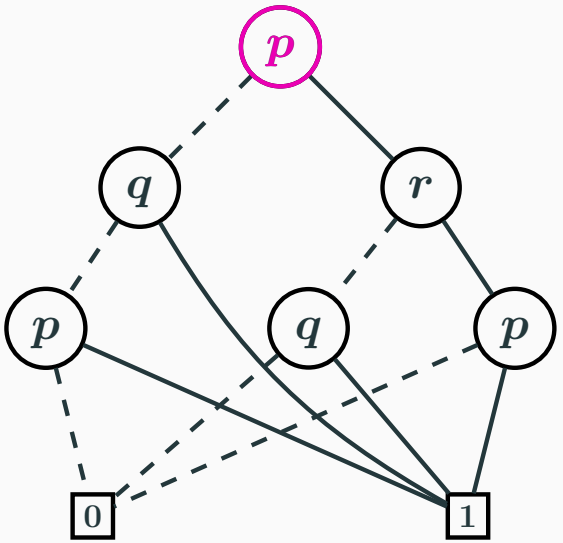
- $[p, q, p]$
- $[p, q]$

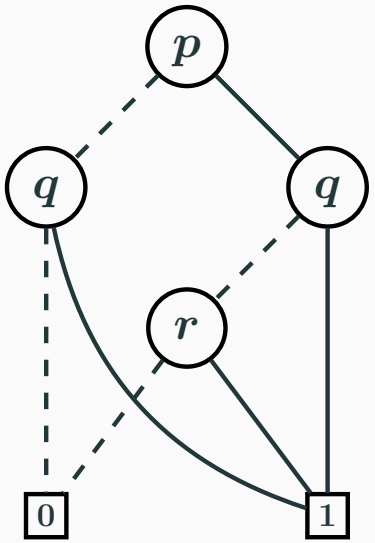




Conceito de ordem de um caminho

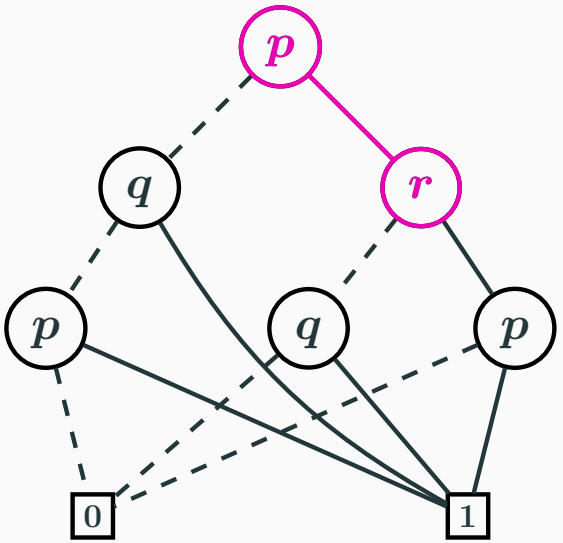
- $[p, q, p]$
- $[p, q]$
- $[p \qquad]$

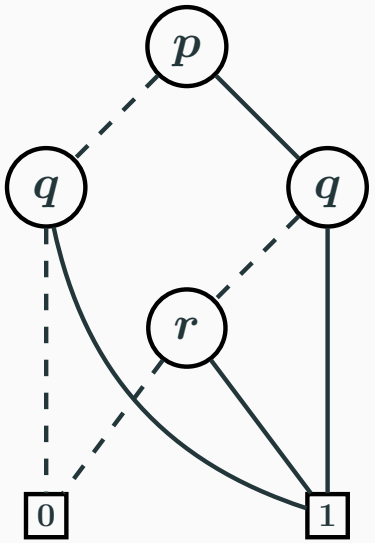




Conceito de ordem de um caminho

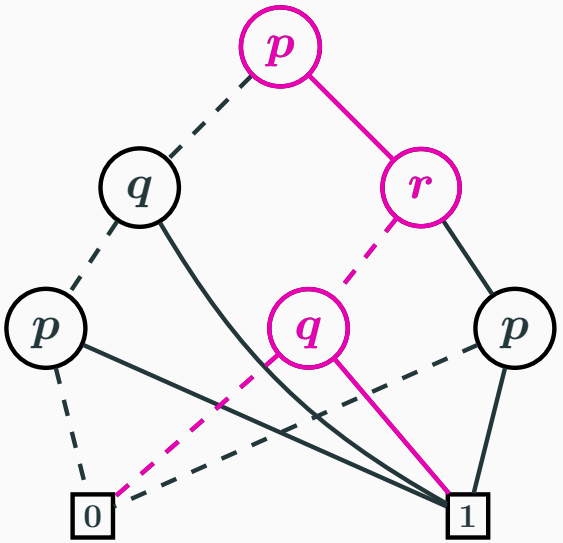
- $[p, q, p]$
- $[p, q]$
- $[p, r \quad]$

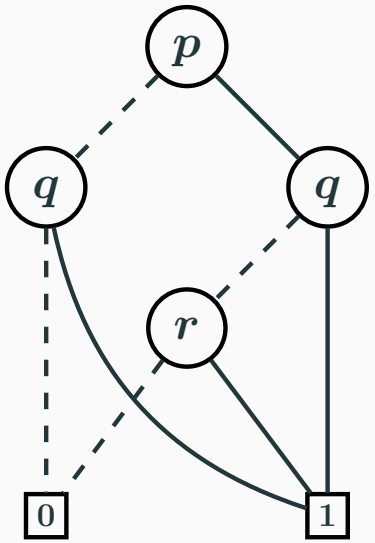




Conceito de ordem de um caminho

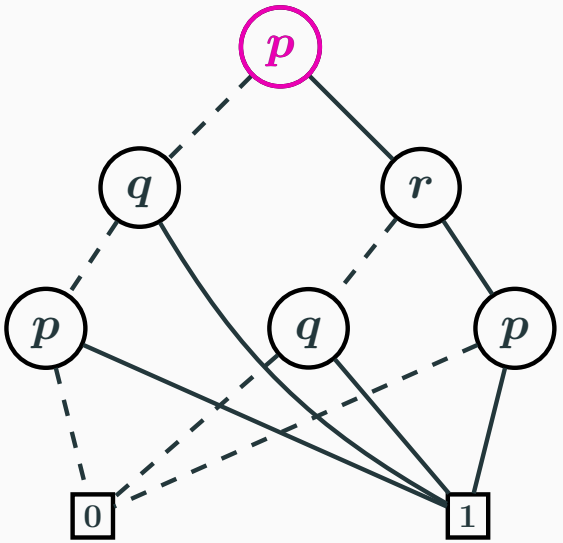
- $[p, q, p]$
- $[p, q]$
- $[p, r, q]$

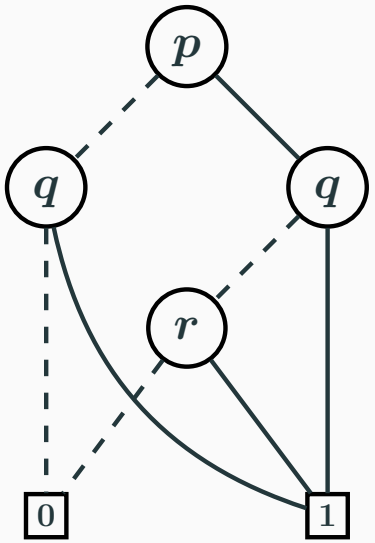




Conceito de ordem de um caminho

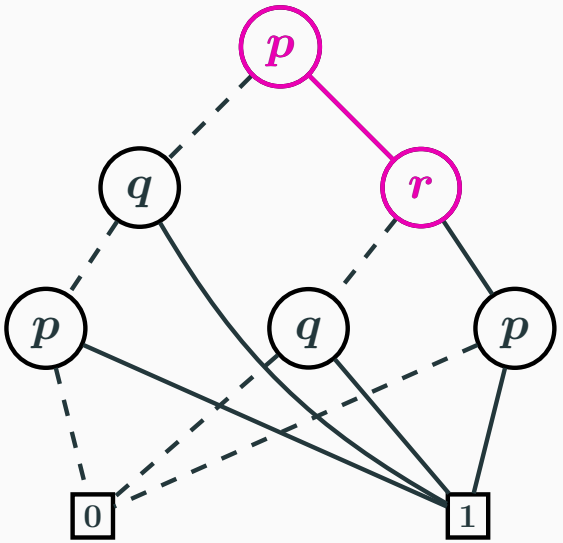
- $[p, q, p]$
- $[p, q]$
- $[p, r, q]$
- $[p \quad\quad]$

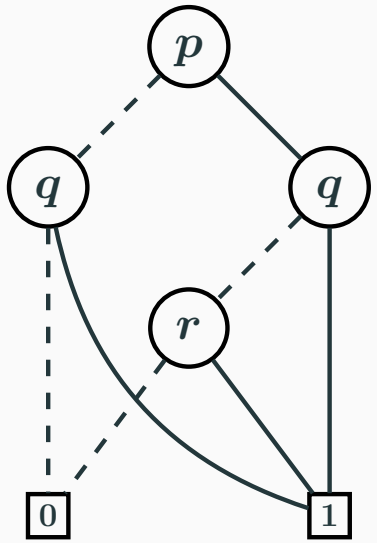




Conceito de ordem de um caminho

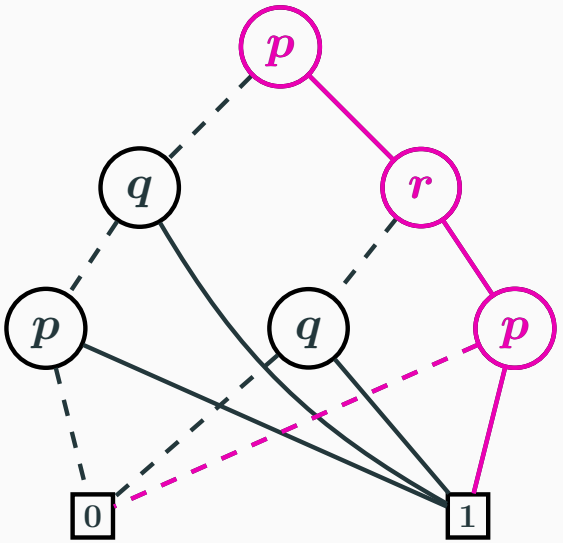
- $[p, q, p]$
- $[p, q]$
- $[p, r, q]$
- $[p, r \quad]$

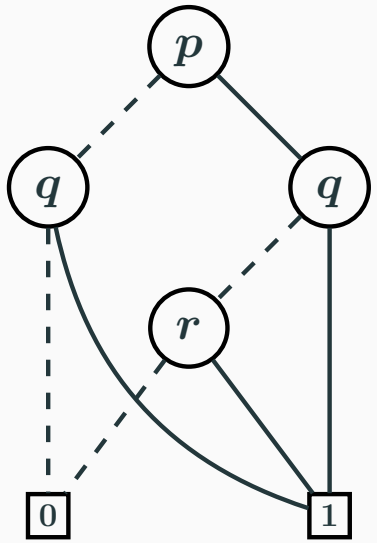




Conceito de ordem de um caminho

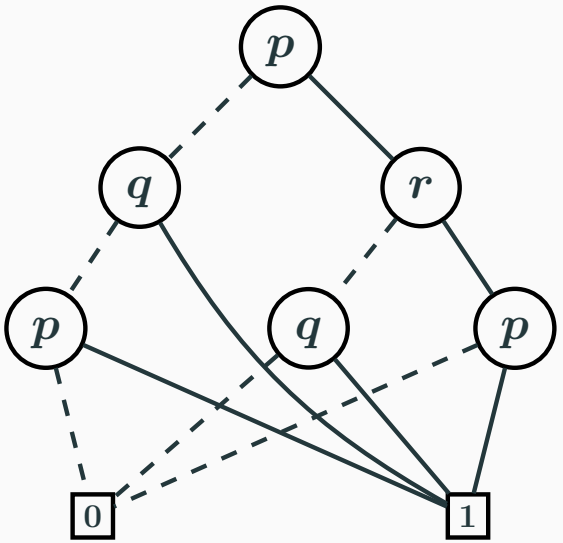
- $[p, q, p]$
- $[p, q]$
- $[p, r, q]$
- $[p, r, p]$



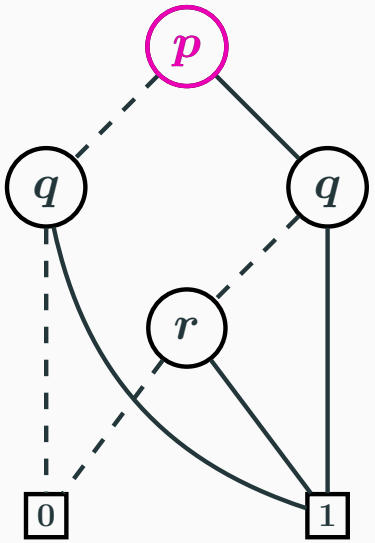


Conceito de ordem de um caminho

- $[p, q, p]$
- $[p, q]$
- $[p, r, q]$
- $[p, r, p]$

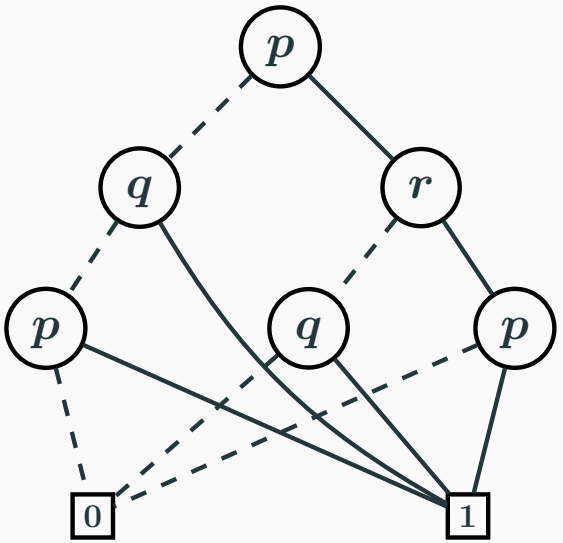


- $[p \quad]$

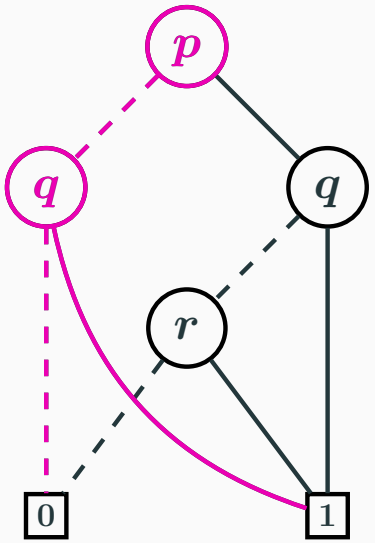


Conceito de ordem de um caminho

- $[p, q, p]$
- $[p, q]$
- $[p, r, q]$
- $[p, r, p]$

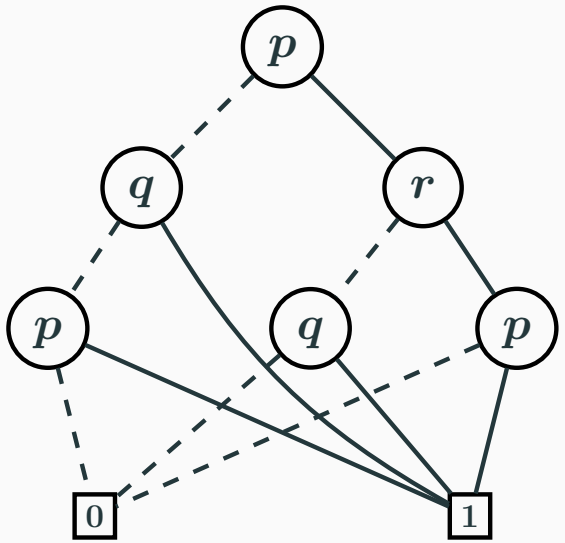


- $[p, q]$

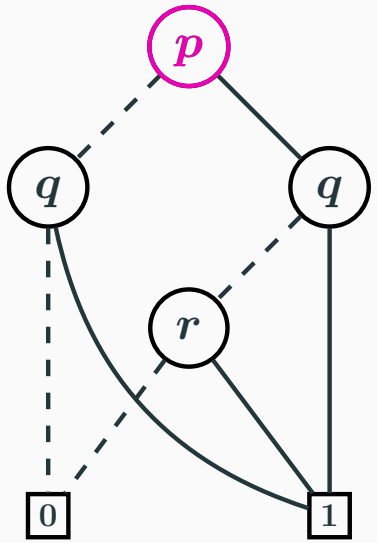


Conceito de ordem de um caminho

- $[p, q, p]$
- $[p, q]$
- $[p, r, q]$
- $[p, r, p]$

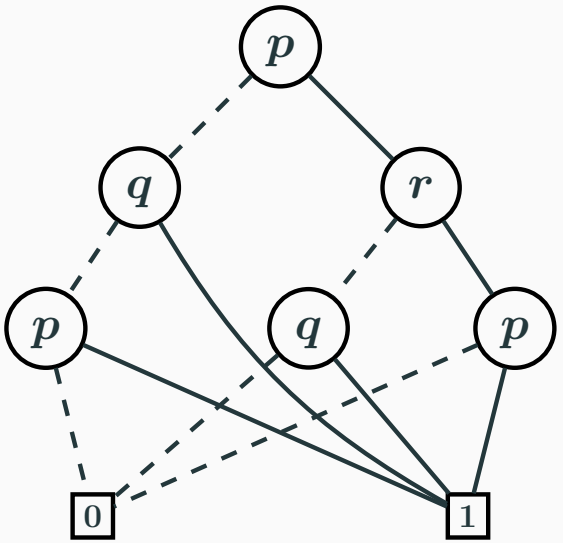


- $[p, q]$
- $[p \quad]$

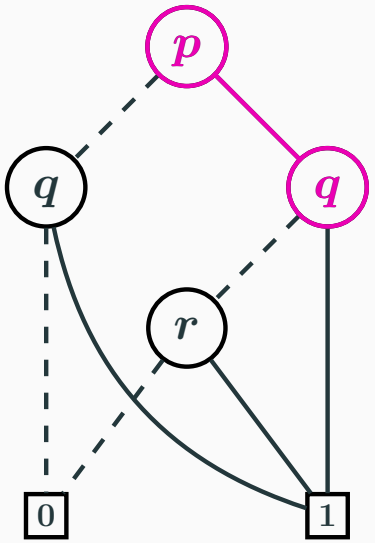


Conceito de ordem de um caminho

- $[p, q, p]$
- $[p, q]$
- $[p, r, q]$
- $[p, r, p]$

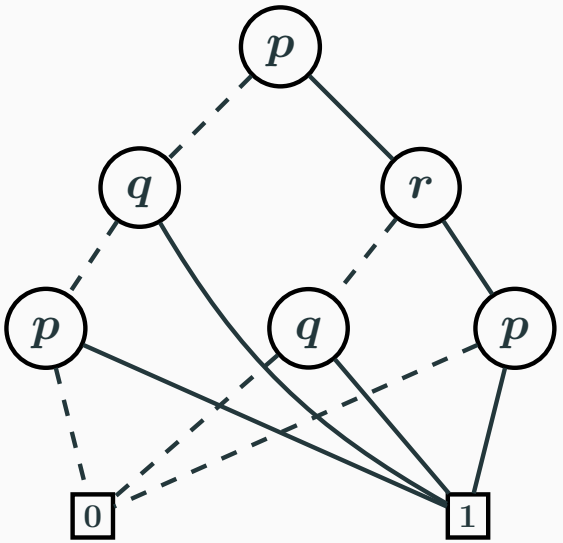


- $[p, q]$
- $[p, q \quad]$

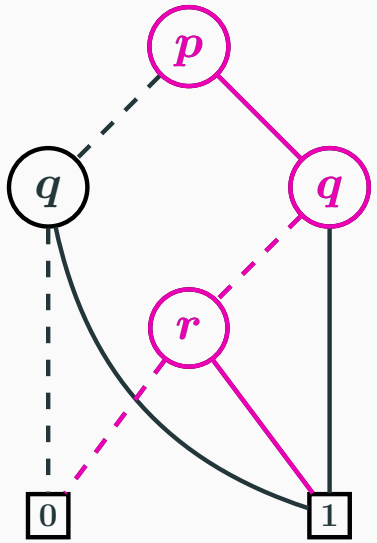


Conceito de ordem de um caminho

- $[p, q, p]$
- $[p, q]$
- $[p, r, q]$
- $[p, r, p]$

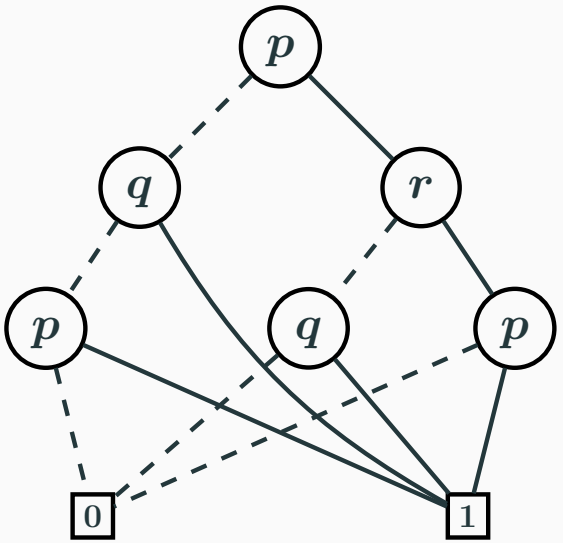


- $[p, q]$
- $[p, q, r]$

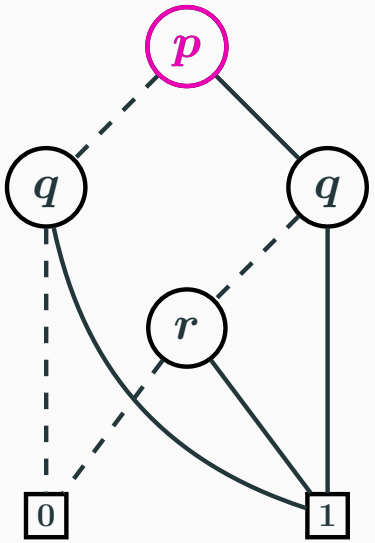


Conceito de ordem de um caminho

- $[p, q, p]$
- $[p, q]$
- $[p, r, q]$
- $[p, r, p]$

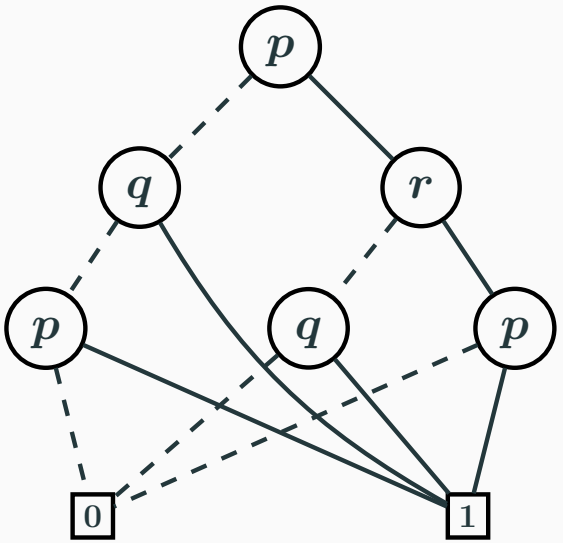


- $[p, q]$
- $[p, q, r]$
- $[p \quad]$

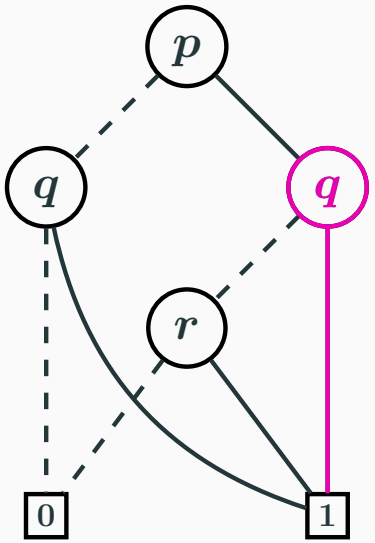


Conceito de ordem de um caminho

- $[p, q, p]$
- $[p, q]$
- $[p, r, q]$
- $[p, r, p]$

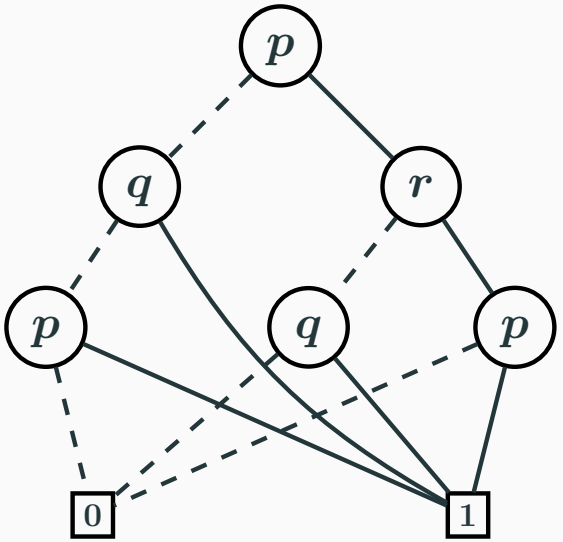


- $[p, q]$
- $[p, q, r]$
- $[p, q]$



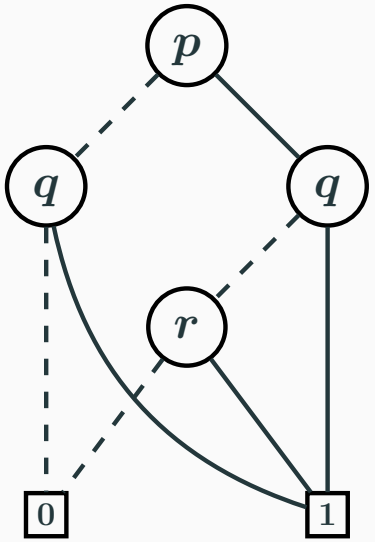
Conceito de ordem de um caminho

- $[p, q, p]$
- $[p, q]$
- $[p, r, q]$
- $[p, r, p]$



Ocorrem repetições e não há um padrão na ordenação

- $[p, q]$
- $[p, q, r]$
- $[p, q]$



Não ocorrem repetições e há um padrão na ordenação

BDDs ordenados

Quando a ordem das variáveis em qualquer caminho é sempre a mesma, o BDD passa a ser chamado Diagrama de Busca Binária Ordenado (OBDD)

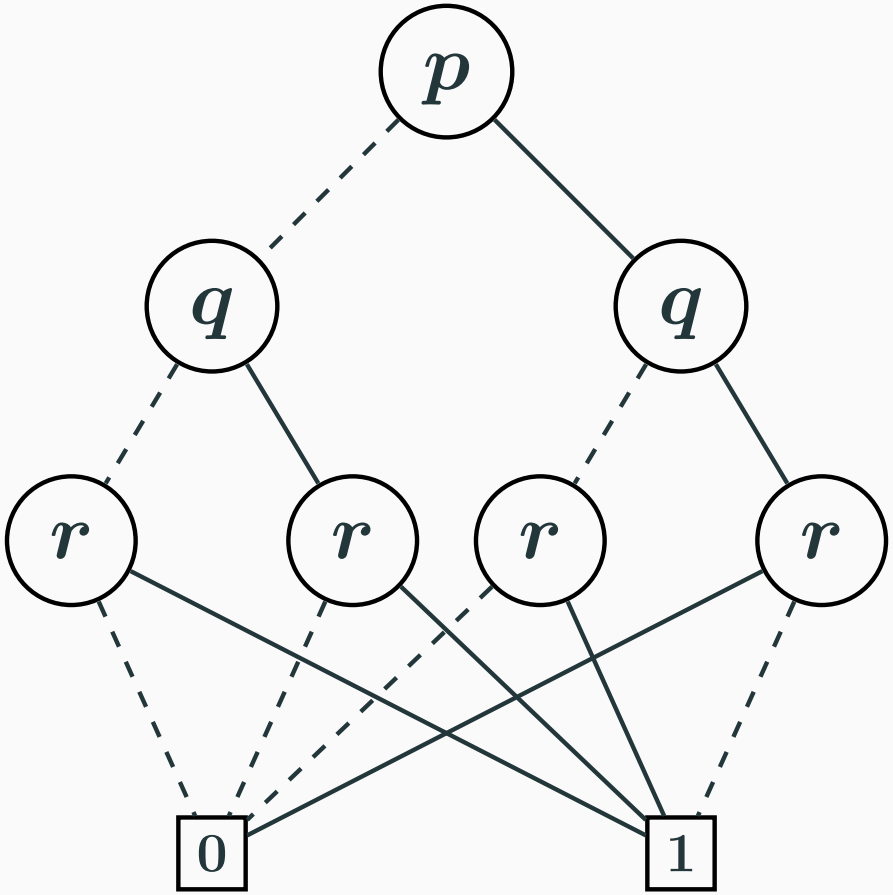
Definição: OBDDs

Definição 6.6

Seja $[p_1, p_2, ..., p_n]$ uma lista ordenada de variáveis sem duplicação e seja B um BDD tal que todas as suas variáveis aparecem em algum lugar da lista. Dizemos que B tem a ordem $[p_1, p_2, ..., p_n]$ se todos os nós de variáveis de B ocorrem na lista, e, para toda ocorrência de p_i seguido de p_j ao longo de qualquer caminho em B temos $i < j$.

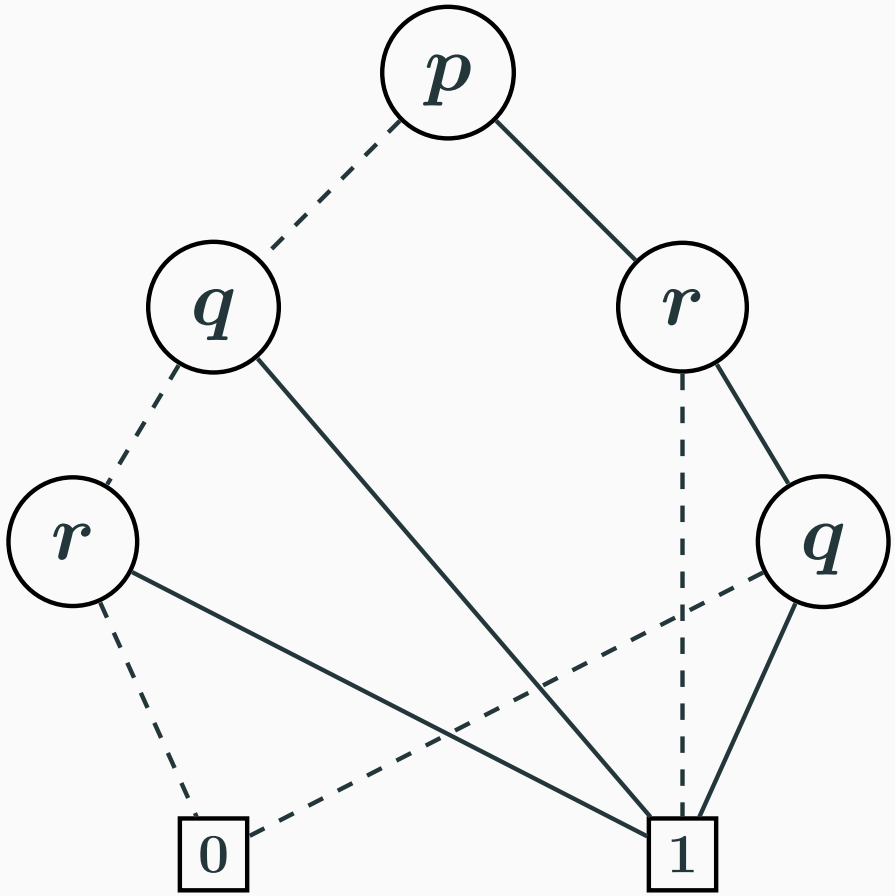
Exemplo de BDD ordenado

Ordem: $[p, q, r]$



Exemplo de BDD não ordenado

Não ordenado ($[p, q, r]$ à esquerda e $[p, r, q]$ à direita)



Vantagens da ordenação de BDDs

- A comparação de dois OBDDs de ordens compatíveis é imediata
  - basta verificar se suas estruturas são idênticas
- Aplicações das reduções C1-C3 em um OBDD garantidamente mantêm a ordem original
- O compromisso com a ordem produz uma representação única de funções booleanas
  - chamada de *forma canônica*

OBDDs reduzidos

OBDDs reduzidos são, por sua vez, chamados de Diagramas de Busca Binária Ordenados Reduzidos (ROBDD)

Teorema: ROBDDs são únicos

Teorema 6.7

A representação em ROBDD de uma função dada ϕ é unica. Isto é, sejam B e B' dois ROBDDs com ordens compatíveis. Se B e B' representam a mesma função booleana, então eles têm estruturas idênticas.

Características de ROBDDs

- RODDBs permitem representações compactas de certas classes de funções booleanas
  - que seriam exponenciais em outros formatos/representações
- Por outro lado, as operações $\wedge$ e $\vee$ apresentadas anteriormente não funcionam
  - pois introduzem ocorrências múltiplas de uma mesma variável

Impacto da escolha da ordenação

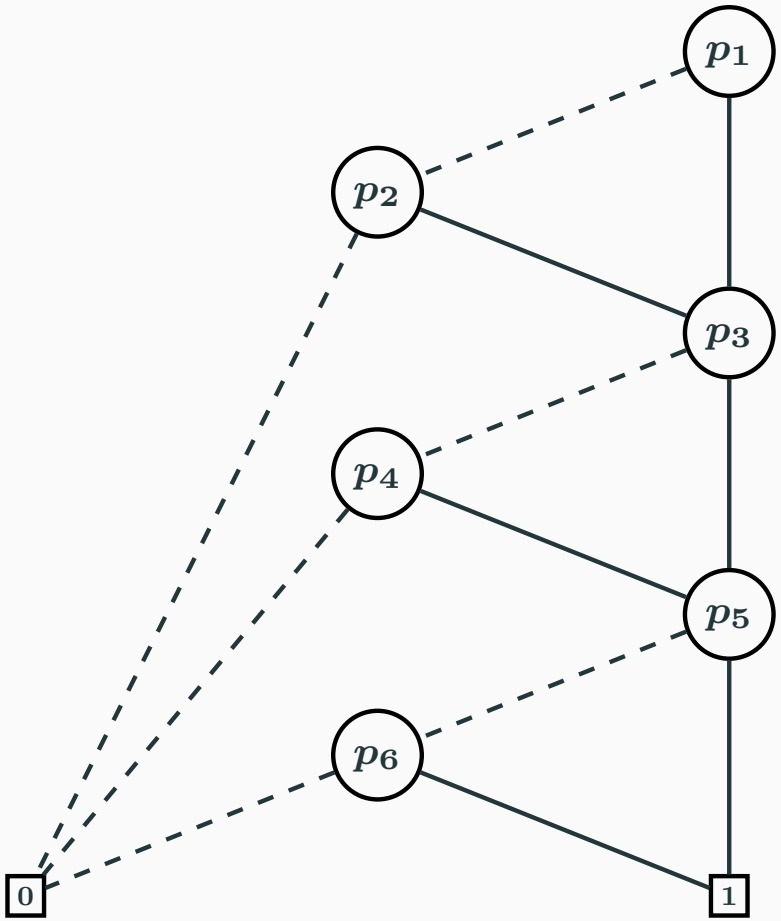
Considere a escolha da ordem de variáveis para a seguinte função booleana em CNF:
$\phi \equiv (p_1 \vee p_2) \wedge (p_3 \vee p_4) \wedge \ldots \wedge (p_{2n-1} \vee p_{2n})$

- Se a escolha for a "*ordem natural de ocorrência na fórmula*" ($[p_1, p_2, p_3, \ldots, , p_{2n-1}, p_{2n}]$), o ROBDD terá $2n + 2$ nós

- Se a escolha for "*índices impares antes de índices pares*" ($[p_1, p_3, p_5, \ldots, p_{2n-1}, p_2, p_4, p_6, \ldots, p2n]$), o ROBDD terá 2^{n+1} nós

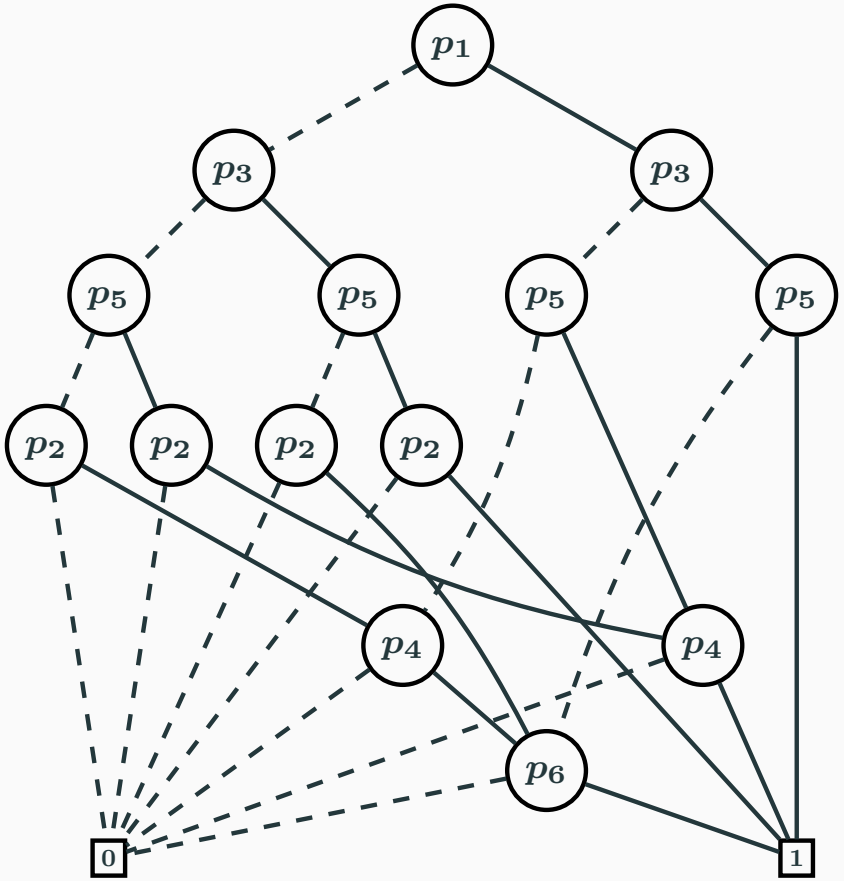
Ordem "natural" para $n = 3$

ROBDD para $\phi \equiv (p_1 \vee p_2) \wedge (p_3 \vee p_4) \wedge (p_5 \wedge p_6)$ com a ordem de variáveis $[p_1, p_2, p_3, p_4, p_5, p_6]$



Ordem "ímpar/par" para $n = 3$

ROBDD para $\phi \equiv (p_1 \vee p_2) \wedge (p_3 \vee p_4) \wedge (p_5 \wedge p_6)$ com a ordem de variáveis $[p_1, p_3, p_5, p_2, p_4, p_6]$



Escolha da ordenação

- A sensibilidade do tamanho de um ROBDD à ordem escolhida é um preço que se paga pelas vantagens obtidas
- Encontrar a ordem ótima também é um problema computacional caro
  - mas há heurísticas que produzem ordens razoavelmente boas

Importância da representação canônica

- **Ausência de variáveis redundantes**. Se o valor de uma função booleana não depende de uma variável, então nenhum ROBDD que a represente contém tal variável;

Importância da representação canônica

- **Ausência de variáveis redundantes**. Se o valor de uma função booleana não depende de uma variável, então nenhum ROBDD que a represente contém tal variável;

- **Teste de equivalência semântica**. Se duas funções são representadas por OBDDs com ordens compatíveis, é possível decidir eficientemente se são equivalentes reduzindo seus OBDDs e comparando sua estrutura;

Importância da representação canônica

- **Ausência de variáveis redundantes**. Se o valor de uma função booleana não depende de uma variável, então nenhum ROBDD que a represente contém tal variável;

- **Teste de equivalência semântica**. Se duas funções são representadas por OBDDs com ordens compatíveis, é possível decidir eficientemente se são equivalentes reduzindo seus OBDDs e comparando sua estrutura;

- **Teste de validade**. Se uma função booleana é válida, seu ROBDD é igual a B_1;

Importância da representação canônica

- **Ausência de variáveis redundantes**. Se o valor de uma função booleana não depende de uma variável, então nenhum ROBDD que a represente contém tal variável;

- **Teste de equivalência semântica**. Se duas funções são representadas por OBDDs com ordens compatíveis, é possível decidir eficientemente se são equivalentes reduzindo seus OBDDs e comparando sua estrutura;

- **Teste de validade**. Se uma função booleana é válida, seu ROBDD é igual a B_1;

- **Teste de satisfação**. Se uma função booleana é satisfeita, então seu ROBDD não é igual a B_0.

Importância da representação canônica

- **Ausência de variáveis redundantes**. Se o valor de uma função booleana não depende de uma variável, então nenhum ROBDD que a represente contém tal variável;
- **Teste de equivalência semântica**. Se duas funções são representadas por OBDDs com ordens compatíveis, é possível decidir eficientemente se são equivalentes reduzindo seus OBDDs e comparando sua estrutura;
- **Teste de validade**. Se uma função booleana é válida, seu ROBDD é igual a B_1;
- **Teste de satisfação**. Se uma função booleana é satisfeita, então seu ROBDD não é igual a B_0.
- **Teste de implicação**. Pode-se testar se uma função ϕ implica em outra ψ calculando o ROBDD para $\phi \wedge \psi$ – a implicação é verdadeira se e somente este ROBDD é igual a B_0;

Estrutura de dados

A estrutura de dados para representar um ROBDD é composta de:

- Uma tabela $T : n \mapsto \langle v, t, f \rangle$
  - que associa a cada identificador n um nó com variável de teste v, filho esquerdo t e filho direito f
- Uma tabela inversa $T^{-1} : \langle v, t, f \rangle \mapsto n$
  - que associa nós em identificadores
  - devido ao compartilhamento de sub-grafos
  - usada para garantir que os diagramas sejam reduzidos

Ilustração dessa estrutura de dados

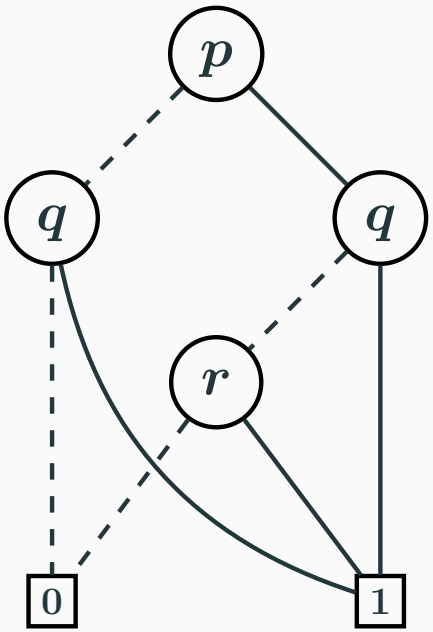


Tabela $T : n \mapsto \langle v, t, f \rangle$

n	$T(n)$
0	$\langle p_6, \text{NULL}, \text{NULL} \rangle$
1	$\langle p_6, \text{NULL}, \text{NULL} \rangle$
2	$\langle p, 3, 4 \rangle$
3	$\langle q, 0, 1 \rangle$
4	$\langle q, 5, 1 \rangle$
5	$\langle r, 0, 1 \rangle$

Tabela $T^{-1} : \langle v, t, f \rangle \mapsto n$

$\langle v, t, f \rangle$	$T^{-1}(\langle v, t, f \rangle)$
$\langle p_6, \text{NULL}, \text{NULL} \rangle$	0
$\langle p_6, \text{NULL}, \text{NULL} \rangle$	1
$\langle p, 3, 4 \rangle$	2
$\langle q, 0, 1 \rangle$	3
$\langle q, 5, 1 \rangle$	4
$\langle r, 0, 1 \rangle$	5

p_6: variável auxiliar usada nos nós terminais para manter a uniformidade da tabela

Observações

Nos algoritmos estudados a seguir, assume-se que:

- $T(n) = T^{-1}(\langle v, t, f \rangle) = \text{NULL}$ sempre que $(n, \langle v, t, f \rangle) \notin T$
- A tabela T é uma variável global e $|T|$ é o número de entradas existentes nessa tabela

Algoritmo de inicialização

Cria a tabela T de um ROBDD. Funciona assim:

- Recebe uma entrada m indicando o número máximo de variáveis existentes na expressão booleana
- Inicia a tabela T com duas tuplas especiais
  - representando os nós terminais 0 e 1
  - para garantir uniformidade, associa os nós terminais à uma variável auxiliar p_{m+1}

Pseudocódigo de INIT

1: **procedure** $\mathrm{INIT}(\boldsymbol{T}, \boldsymbol{m})$

2: $\quad \boldsymbol{T} \leftarrow \{(\mathbf{0}, \langle \boldsymbol{m}+\mathbf{1}, \textsc{null}, \textsc{null} \rangle\}),$
$\quad \{(\mathbf{1}, \langle \boldsymbol{m}+\mathbf{1}, \textsc{null}, \textsc{null} \rangle\})$

Algoritmo de inserção de nós

Insere um nó em um ROBDD, mantendo-o reduzido e ordenado. Funciona assim:

- Recebe como entrada uma variável v e os identificadores de seus filhos t e f
- Se o nó v for redundante ($t = f$), devolve imediatamente o identificador do nó filho (t)
- Caso o nó v já tenha sido criado, devolve seu identificador
- Caso o nó v seja novo, cria-o e devolve o identificador

Pseudocódigo de INS

1: **function** $\mathrm{INS}(T, v, t, f)$
2: **if** $t = f$ **then**
3: **return** t
4: $n \leftarrow T^{-1}(\langle v, t, f \rangle)$
5: **if** $n = \mathrm{NULL}$ **then**
6: $n \leftarrow |T|$
7: $T \leftarrow T \cup \{(n, \langle v, t, f \rangle)\}$
8: **return** n

Expansão de Shannon

Algoritmo de construção de ROBDDs

Constrói um ROBDD a partir de uma expressão booleana em CNF. Funciona assim:

- Recebe como entrada uma variável v e os identificadores de seus filhos t e f
- Se o nó v for redundante ($t = f$), devolve imediatamente o identificador do nó filho (t)
- Caso o nó v já tenha sido criado, devolve seu identificador
- Caso o nó v seja novo, cria-o e devolve o identificador

Pseudocódigo de BUILD

1: **function** $\mathrm{BUILD}(v, \varepsilon_t, \varepsilon_f)$
2: **if** $\varepsilon_t, \varepsilon_f \in \{0, 1\}$ **then**
3: **return** $\mathrm{INS}(v, \varepsilon_t, \varepsilon_f)$
4: **if** $\varepsilon_f \in \{0, 1\}$ **then**
5: **return** $\mathrm{INS}(v, \mathrm{BUILD}(\varepsilon_t), \varepsilon_f)$
6: **if** $\varepsilon_t \in \{0, 1\}$ **then**
7: **return** $\mathrm{INS}(v, \varepsilon_t, \mathrm{BUILD}(\varepsilon_f))$
8: **return** $\mathrm{INS}(v, \mathrm{BUILD}(\varepsilon_t), \mathrm{BUILD}(\varepsilon_f))$