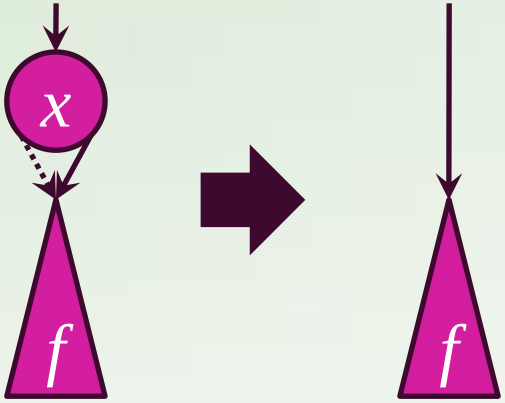
BDD演算の計算量について

吉仲 亮[1], 川原 純[1], ○伝住 周平[2], 有村 博紀[2], 湊 真一[1,2]

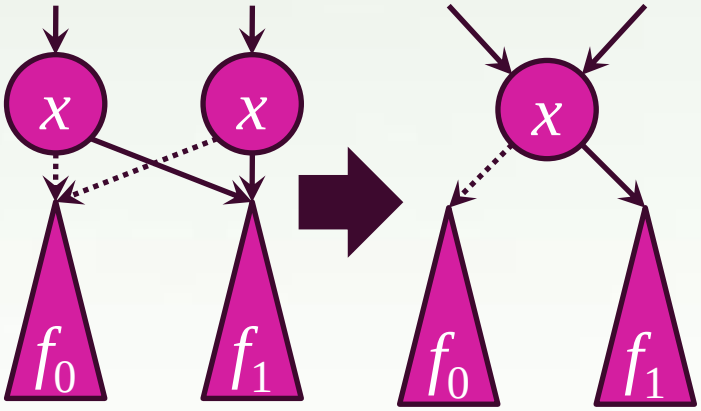
[1)] JST ERATO 湊離散構造処理系
[2)] 北海道大学大学院情報科学研究科

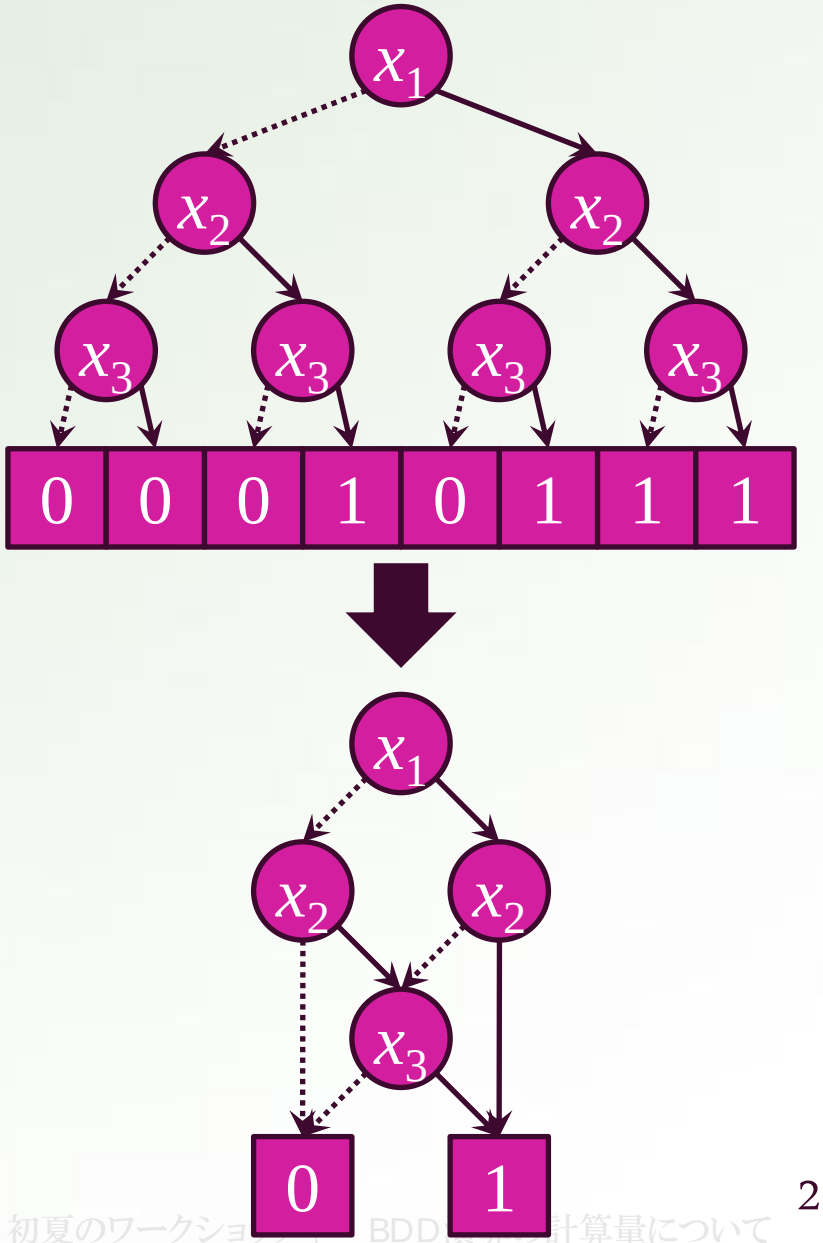
Binary Decision Diagram

- 論理関数を表現するデータ構造
- 二分木に簡約化規則を適用
  - 冗長節点の削除



  - 等価な節点の共有





Getnode演算

Input: x: variable, p_0, p_1: BDD nodes

Result: BDD node with $[x, p_0, p_1]$

1: if $p_0 = p_1$ then

2: return p_0; //冗長節点の削除

3: else if there is a node p with the key $<x, p_0, p_1>$ in UNIQUETABLE then

4: return p; //等価な節点の共有

5: else

6: create a node p with $p \rightarrow [x, p_0, p_1]$;

7: register p to UNIQUETABLE with the key $<x, p_0, p_1>$;

8: return p; //節点の新規生成

9: end if

BDD 演算

- 基本的な論理演算
  - $\diamond \in \{\wedge, \vee, \Rightarrow, \ldots\}$: 自明なものも含めて $2^4 = 16$ 種類
  - Apply($\diamond$, f, g)で実行
  - 再帰的な演算で効率よく実現
- 最悪時間計算量 (証明済み)
  - $B(f)$: BDD の節点数を表す
  - $O(|B(f)|\,|B(g)|)$: 演算キャッシュを利用して示す
- Bryantの予想 (1986)
  - $O(|B(f)| + |B(g)| + |B(f \diamond g)|)$: 入出力線形
  - 経験的には正しく感じる
  - 現在まで未解決

Apply演算

Input: $\diamond$: binary Boolean operation, f, g: BDD nodes;

Result: BDD node for $f \diamond g$

1: if the value of $\mathrm{Apply}(\diamond, f, g)$ is trivial then let h be such that $h = f \diamond g$;

2: return h;

3: else if there is a node h with the key $<\diamond, f, g>$ in cache then return h;

4: else

5: let x, f_0, f_1, y, g_0, g_1 be such that $f \rightarrow [x, f_0, f_1]$ and $g \rightarrow [y, g_0, g_1]$;

6: if $x = y$ then

7: let h be $\mathrm{Getnode}(x, \mathrm{Apply}(\diamond, f_0, g_0), \mathrm{Apply}(\diamond, f_1, g_1))$;

8: else if $x < y$ then

9: let h be $\mathrm{Getnode}(x, \mathrm{Apply}(\diamond, f_0, g), \mathrm{Apply}(\diamond, f_1, g))$;

10: else

11: let h be $\mathrm{Getnode}(y, \mathrm{Apply}(\diamond, f, g_0), \mathrm{Apply}(\diamond, f, g_1))$;

12: end if

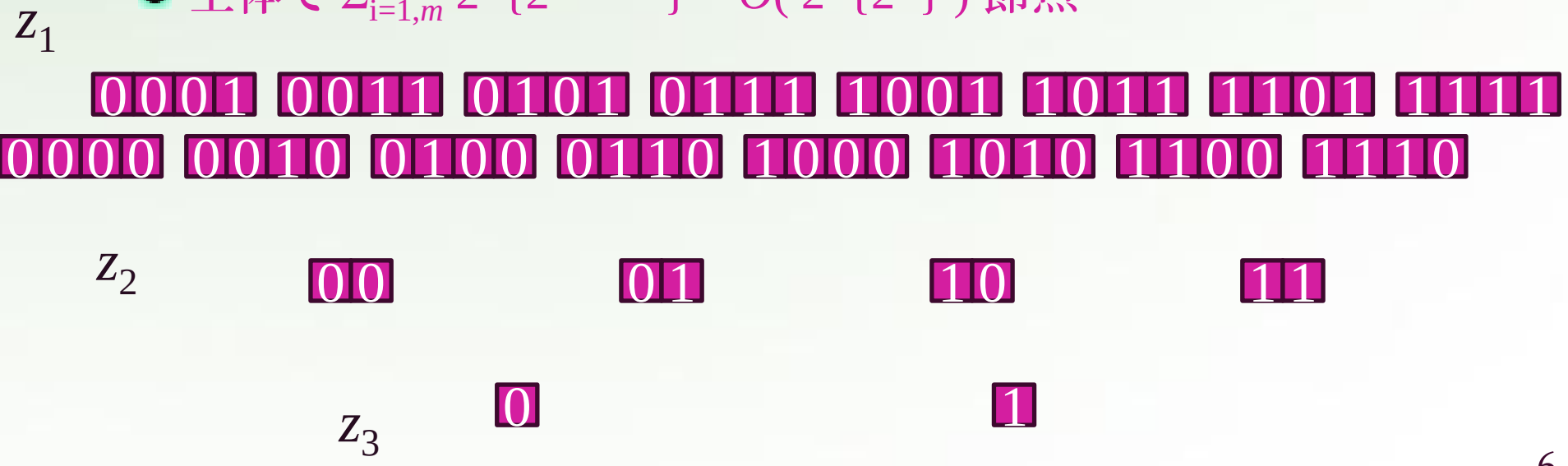
13: register h to cache with the key $<\diamond, f, g>$;

14: return h;

15: end if

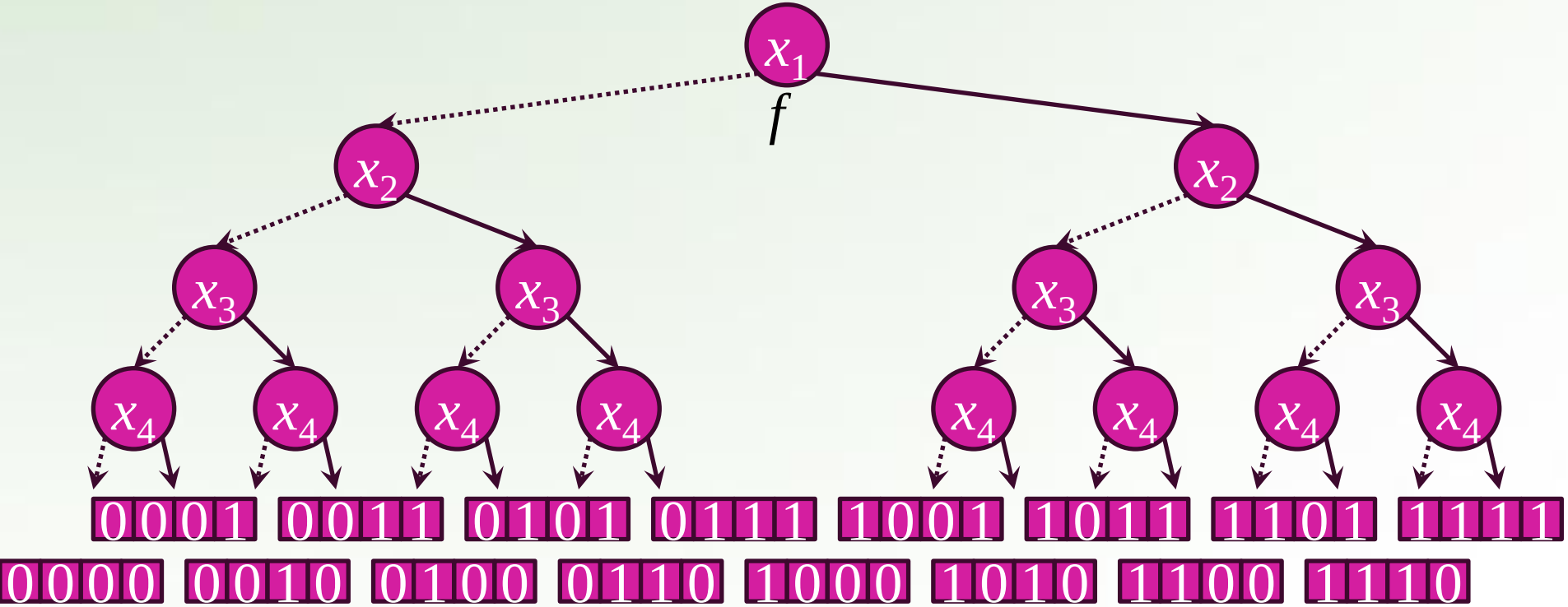
構成

- BDDは真理値表に対応する
  - m 変数論理関数では真理値表の長さは 2^m
  - 可能なパターンは $2^{\{2^m\}}$ 種類
- m変数に対して可能 な全ての異なるBDDを考える
  - 変数 $z_1, z_2, \ldots, z_m$：変数順序はこの順
  - 変数 z_i をラベルにもつ節点は高々 $2^{\{2^{m-i+1}\}}$ 個
  - 全体で $\Sigma_{i=1,m}\ 2^{\{2^{m-i+1}\}} = O(\ 2^{\{2^m\}}\ )$ 節点



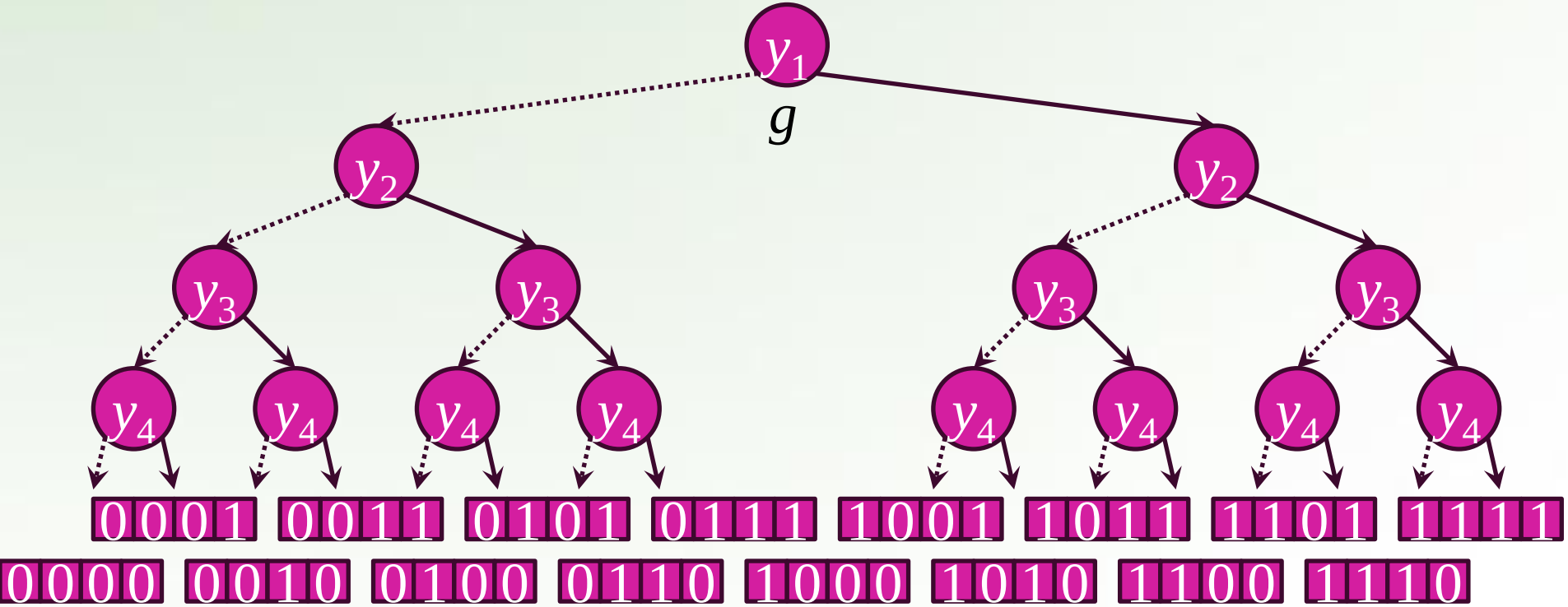
構成

- 二分木状のBDDで 2^{2^m} 個のBDDを選択する
  - 必要な変数は $n = 2^m$ 個
  - 変数 $x_1, x_2, \ldots, x_n$ による f と 変数 $y_1, y_2, \ldots, y_n$ による g を用意
  - 変数順序は $x_1, y_1, x_2, y_2, \ldots, x_n, y_n$ と互い違いにする



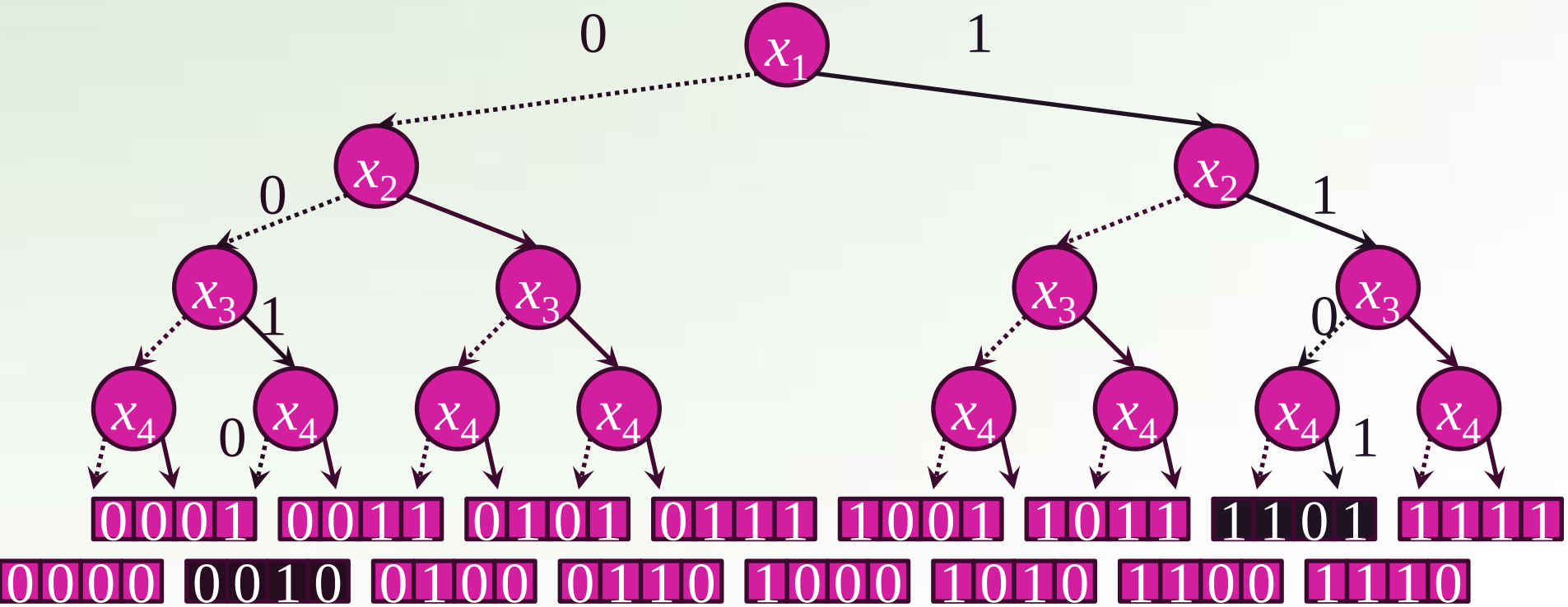
構成

- 二分木状のBDDで 2^{2^m} 個のBDDを選択する
  - 必要な変数は $n = 2^m$ 個
  - 変数 $x_1, x_2, \ldots, x_n$ による f と 変数 $y_1, y_2, \ldots, y_n$ による g を用意
  - 変数順序は $x_1, y_1, x_2, y_2, \ldots, x_n, y_n$ と互い違いにする



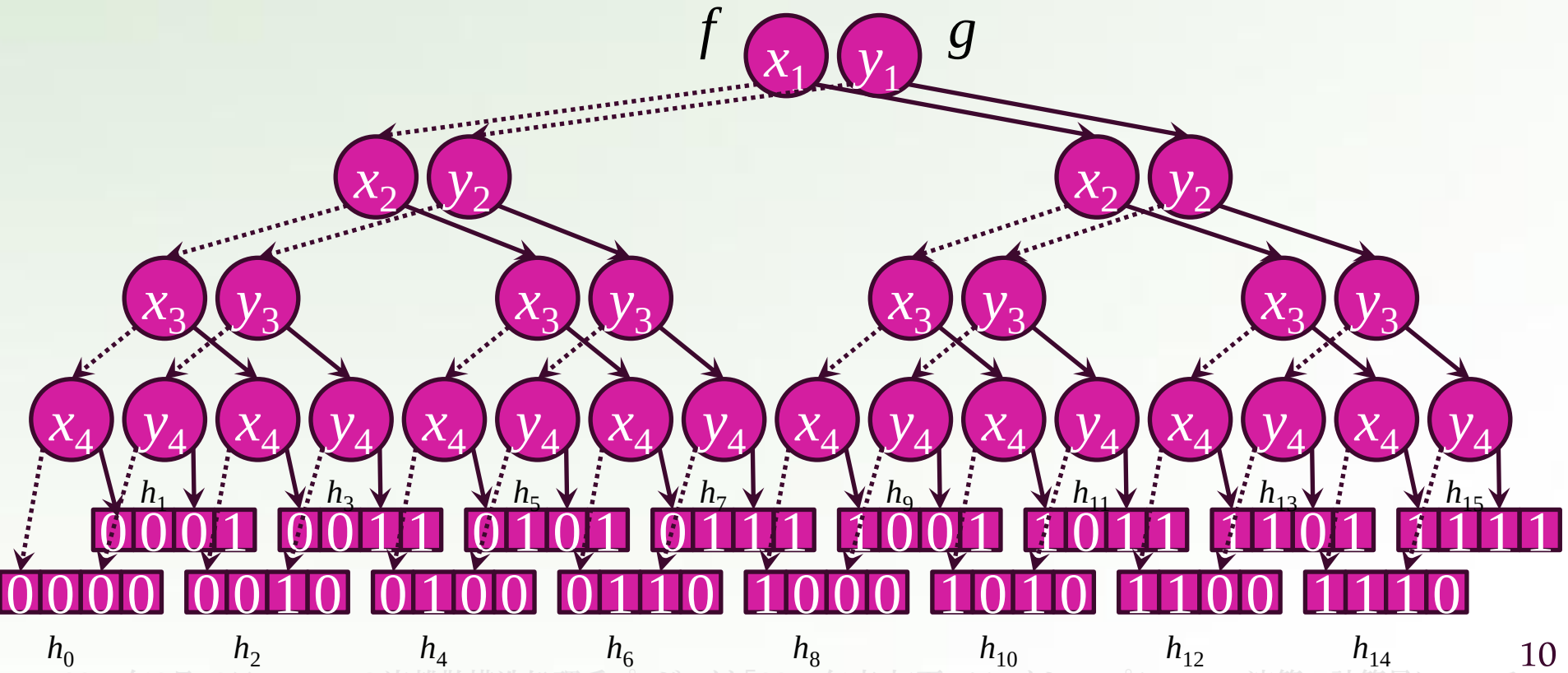
意味

- マルチプレクサを効率悪い変数順序によって表現している
  - 変数列 x, y に代入した 0, 1 の列が真理値表になっている
  - x_i (y_i) が真理値表の左から i 番目を決定している
  - 節点数は f, g ともに $4 \cdot 2^n$ で抑えられる



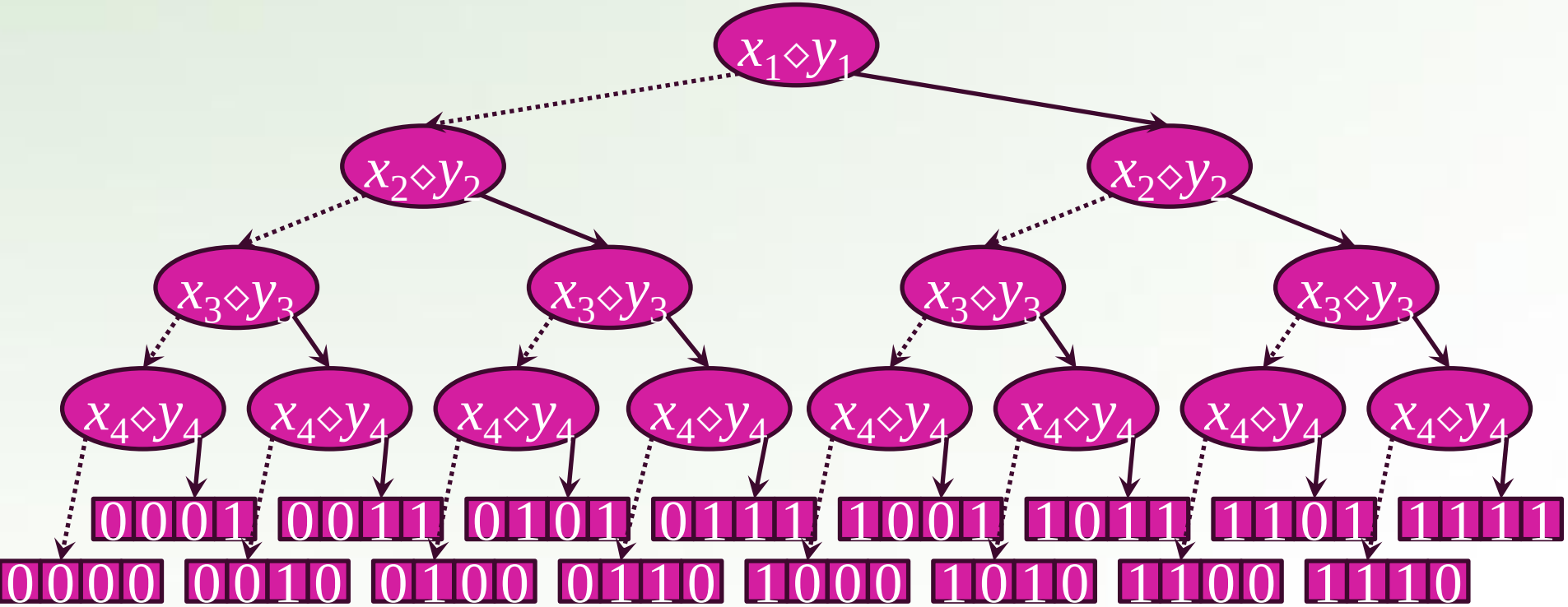
計算

- Apply(◇, f, g) の途中計算を数える. x, yは必ず両側へ再帰
  - 変数 z_1 をもつ節点 2^n 個を順番に $h_0, h_1, \ldots, h\_\{2^n - 1\}$ とする
  - 任意の h の組に対し Apply(◇, h_i, h_j) が起こる($0 \leqq i, j \leqq 2^n$)
  - この組の数は $(2^n)^2$ 存在する (入力サイズオーダーの二乗)



出力サイズ

- 出力 $f \diamond g$ の節点数
  - x_i と y_i をまとめて見ると二分木状 ($x_i \diamond y_i$ は高々3個の節点になる)
  - $x_i \diamond y_i$ の結果で真理値表の左から i 番目が決定される
  - 高々 $6 \cdot 2^n$ 節点 (入力に対し線形)



まとめ

- $f \diamond g$ の入力節点数
  - それぞれ高々 $4 \cdot 2^n = O(2^n)$ 個
- 出力節点数
  - 高々 $6 \cdot 2^n = O(2^n)$ 個
- 計算時間
  - 少なくとも $(2^n)^2 = O((2^n)^2)$ 回 Apply が呼び出される
  - 入出力サイズの二乗なので $O(|B(f)| + |B(g)| + |B(f \diamond g)|)$ でない
- 結果
  - BDDの二項演算の時間計算量は入出力線形ではない
  - Bryantの予想に対する反例が示された