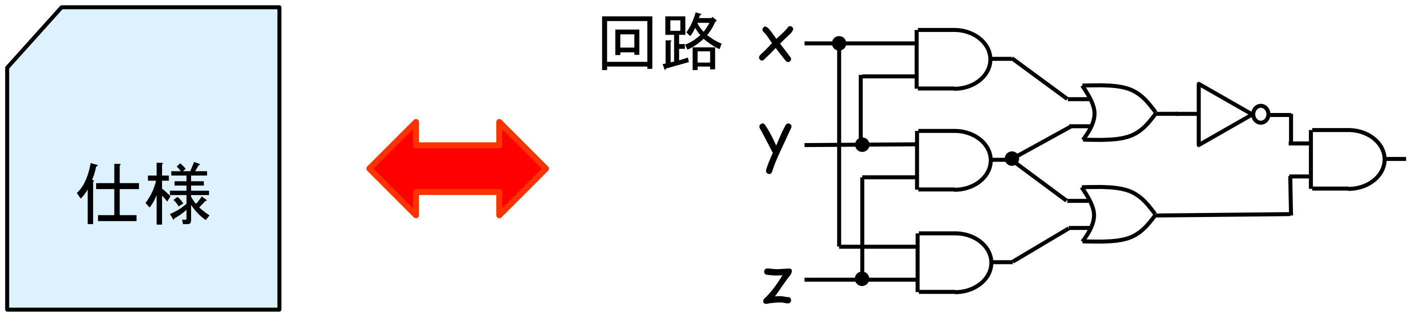
大規模知識処理特論

二分決定グラフの利用 (2)

北海道大学 情報科学研究院
堀山 貴史

BDD の応用: 論理回路の設計検証

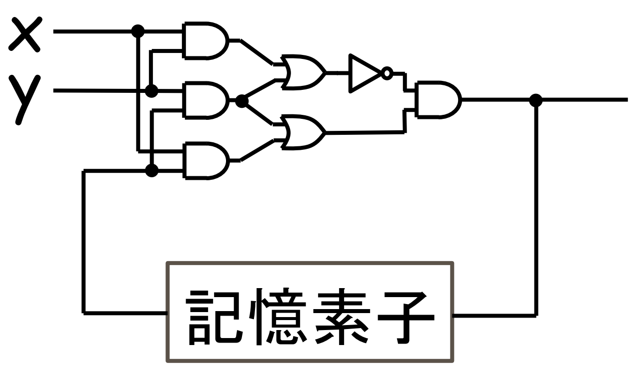
- **組合せ回路**の等価性の検証
  - **仕様の記述** (どんな入力から何を計算をするか) と **設計した回路** (多くの場合に最適化されている)
  - リファレンス実装の回路 と 設計した回路



- すべての入力に対して出力が同じになるか、網羅的にチェックする (真理値表だと、2^n 通りをすべて試す…)
- 仕様の論理関数と回路の論理関数を **BDD** で表す
  - **同じ関数**は、**同じ根節点**から始まる (一意性)

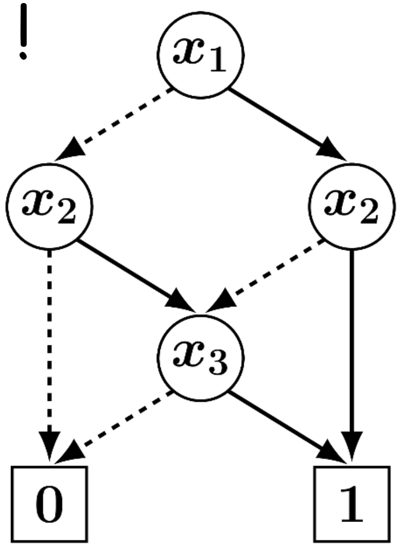
BDD の応用: 論理回路の設計検証

- 順序回路 (状態を持ち、入力により状態が変化する回路)
- すべての入力の系列に対し、同じ出力の系列になるか (組合せ回路と比べ物にならないぐらい、チェックが大変)
- 状態は、レジスタ (記憶素子) に記憶されている
- どの状態からどの入力で、どの状態に行くか、網羅的にチェックする



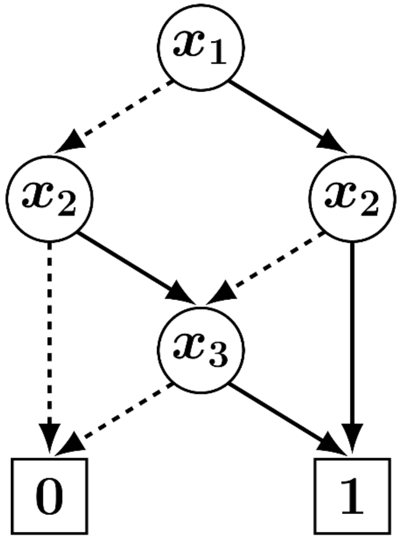
BDD の応用: 組合せ問題と最適化

- 論理変数を組合せ問題と対応させる
- 組合せ問題の**制約条件** (満たすべき条件) を、論理関数で表す
- その論理関数を **BDD** の形で得る
  - **すべての解**が手に入った！
- BDD では、線形和のコストの評価は容易
  - BDD から**最適解**が簡単に得られる！



BDD の利用 (1)

- 与えた組合せが解かどうかの判定
  - 根節点からグラフをたどる
  - 例) { x_1, x_3 } は？　{ x_2 } は？
- BDD から解をすべて列挙する
  - **深さ優先探索**
  - 1-定数節点に来たら、解が得られる

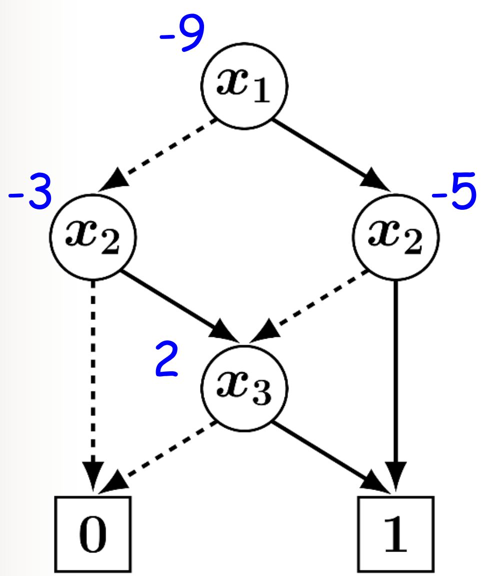


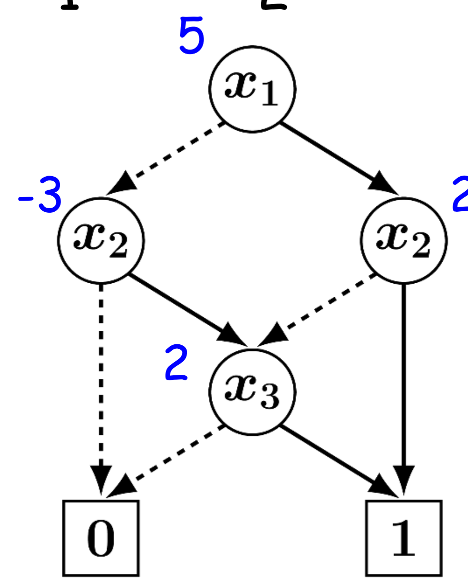
BDD の利用 (2)

- 線形和のコストの評価
  - 各節点で、0-枝側のコストと 1-枝側のコストを見比べて、良い方をその節点でのコストとして採用
  - 根節点のコストが、最適解

例) minimize $-4x_1 - 5x_2 + 2x_3$

maximize $3x_1 - 5x_2 + 2x_3$





休 憩

- ここで、少し休憩しましょう。

- 深呼吸したり、肩の力を抜いてから、
次のビデオに進んでください。

組合せ最適化：ナップサック問題

価格	60円	30円	40円	50円	40円	30円
うれしさ	36	27	12	50	28	24

- おやつは **予算100円 (100円以内)**
  - 各品物は、それぞれ1個まで選べる
  - 品物を分割してはいけない
- **「うれしさ」の総和**が**最大**になるようにしたい

お菓子の画像は、(株) 明治、森永製菓 (株) のウェブサイトより
https://www.meiji.co.jp https://www.morinaga.co.jp

ナップサック問題： 変数 x_i

品物	品物$_1$	品物$_2$	品物$_3$	品物$_4$	品物$_5$	品物$_6$
価格 (円)	60円	30円	40円	50円	40円	30円
うれしさ	36	27	12	50	28	24

- $x_i = \begin{cases} 1 & \cdots \text{選択する} \\ 0 & \cdots \text{選択しない} \end{cases}$
- おやつは **予算100円 (100円以内)**
  - $60\,x_1 + 30\,x_2 + 40\,x_3 + 50\,x_4 + 40\,x_5 + 30\,x_6 \leqq 100$

ナップサック問題： 定式化

品物	品物$_1$	品物$_2$	品物$_3$	品物$_4$	品物$_5$	品物$_6$
価格 (円)	p_1	p_2	p_3	p_4	p_5	p_6
うれしさ	u_1	u_2	u_3	u_4	u_5	u_6

- **目的関数**
  - 最大化： $\sum_{i=1}^{6} u_i x_i$
- **制約条件**
  - $\sum_{i=1}^{6} p_i x_i \leqq 100$
  - $x_i \in \{0, 1\} \quad (i = 1, 2, \ldots, 6)$

0-1 整数計画問題
として記述した

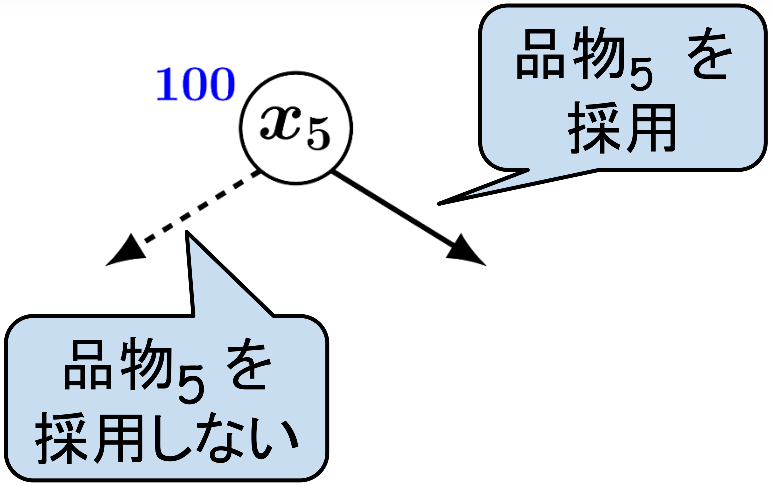
組合せ最適化：ナップサック問題

- 制約条件 $\Sigma_i\, p_i\, x_i \leqq c$ を BDD で表す

  (どうやって？ → この後で)

- 目的関数 $\Sigma_i\, u_i\, x_i$ を最大化する

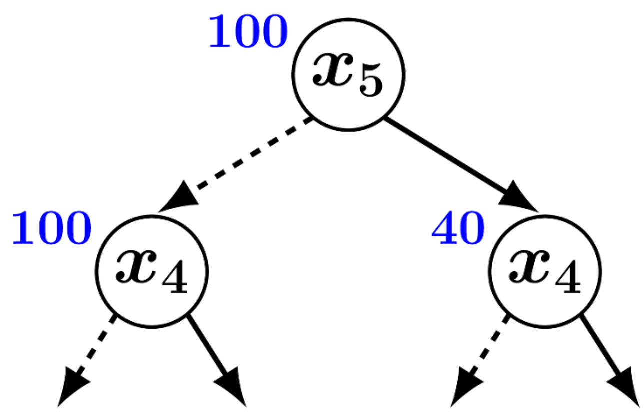
ナップサック問題の制約を BDD で



予算
100 円

品物	品物$_5$	品物$_4$	品物$_3$	品物$_2$	品物$_1$
価格 (円)	60円	30円	30円	60円	30円
うれしさ	36	27	12	50	28

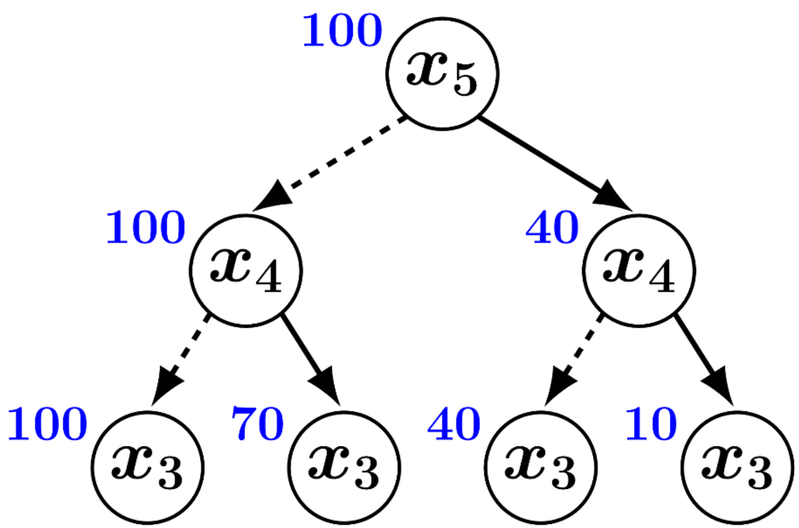
ナップサック問題の制約を BDD で



予算
100 円

品物	品物$_5$	品物$_4$	品物$_3$	品物$_2$	品物$_1$
価格 (円)	60円	30円	30円	60円	30円
うれしさ	36	27	12	50	28

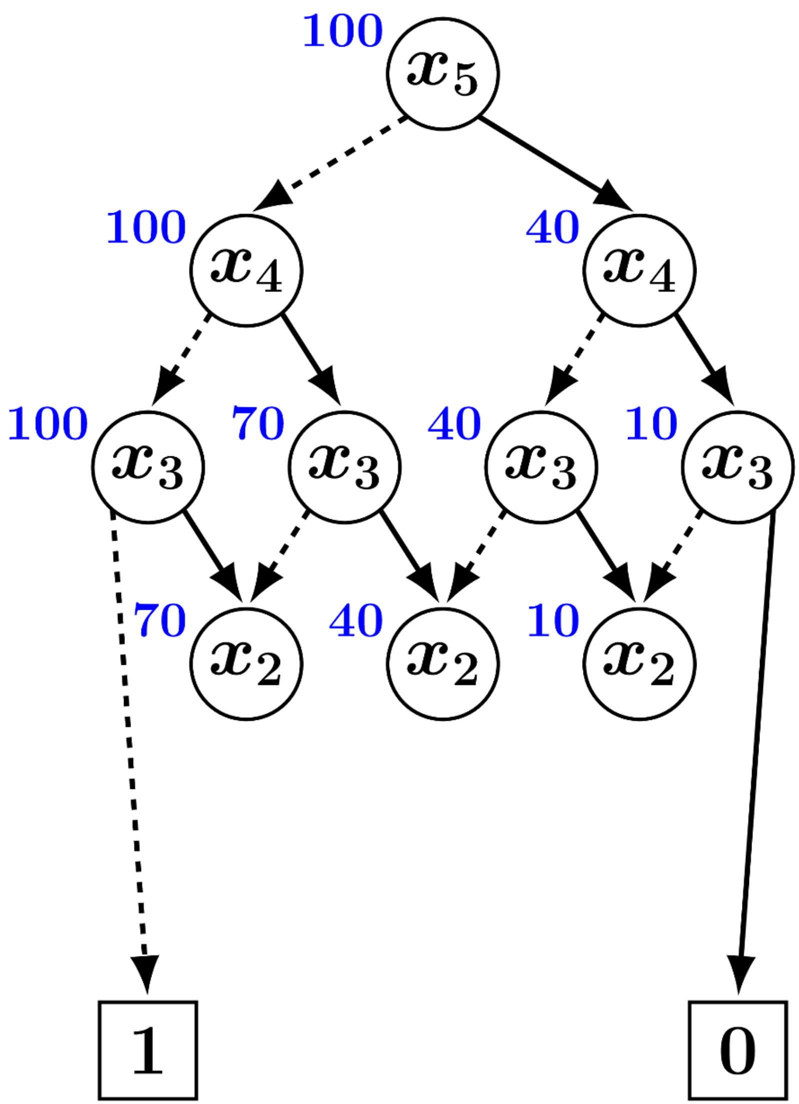
ナップサック問題の制約を BDD で



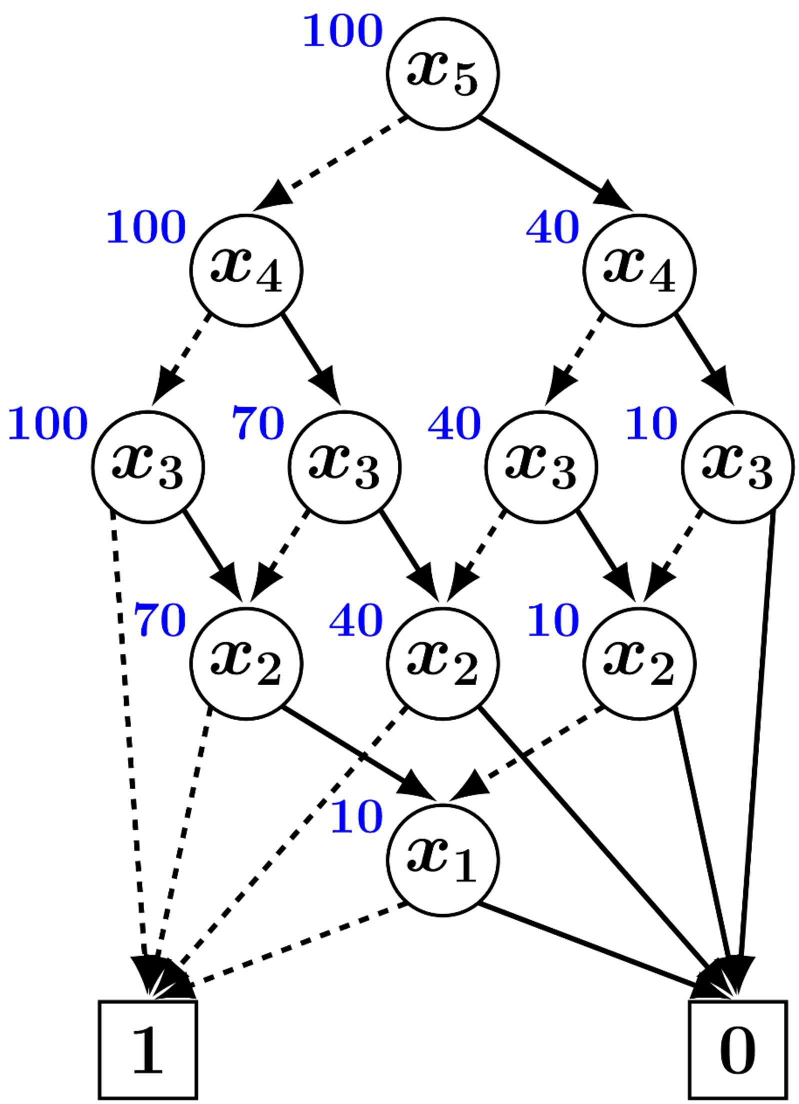
予算
100 円

品物	品物$_5$	品物$_4$	品物$_3$	品物$_2$	品物$_1$
価格 (円)	60円	30円	30円	60円	30円
うれしさ	36	27	12	50	28

ナップサック問題の制約を BDD で



ナップサック問題の制約を BDD で



ナップサック問題の制約を BDD で

- 制約条件を表す BDD が手に入れば
線形和の評価関数の minimize/maximize は簡単

- ただし、単に最適化するだけなら、
BDD を作る必要はない (計算時間・メモリ使用量が大)

- BDD を持つメリット
  - 評価関数を何度も変えながら最適化する
(何度も解空間を探索し直さなくてもよい)
  - 他の制約と組み合わせた制約条件を作れる
(Apply 演算が容易)

まとめ

- 論理回路の設計検証
- 組合せ問題 と 最適化
  - 与えた組合せが解かどうかの判定
  - 線形和のコストの評価
  - 例）ナップサック問題