BDDを利用した命題論理式の前提の列挙

2014/3/8

京都大学工学部　山口 慧
京都大学情報学研究科　吉仲 亮
京都大学情報学研究科　山本 章博

研究の背景

仮説発見問題

背景知識Bと観測した事象Eに対して，

$B \wedge H \models E$を満たす仮説Hを求める

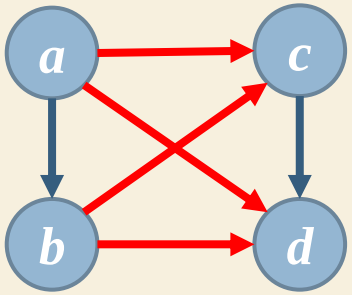
Eに対して$B \wedge H$を前提と呼ぶ

問題例(graph completion problem)

背景知識：node(X) ∧ node(Y) ∧ arc(X, Y) → path(X, Y),
node(X) ∧ node(Y) ∧ node(Z) ∧ arc(X, Y)
∧ path(Y, Z) → path(X, Z),
node(a), node(b), node(c), node(d), arc(a, b), arc(c, d)

観測事象： path(a, d)

仮説の例：arc(a, d), arc(a, c), arc(b, d), arc(b, c)



研究の背景

仮説発見問題

背景知識Bと観測した事象Eに対して，
$B \wedge H \models E$を満たす仮説Hを求める

Eに対して$B \wedge H$を前提と呼ぶ

問題例(graph completion problem)

- 本研究は命題論理を対象とする
- 解の個数はBとEのサイズに対して指数オーダー
  - 従来は解に制約を与えるなどの手法が提案された

⇒ 全ての解を効率的に求める手法を構成

研究の概要

問題

与えられた論理式Fに対して$H \models F$を満たすHを
重複なく全て列挙する

$B \wedge H \models E \Leftrightarrow H \models \neg B \vee E$ なので
$F = \neg B \vee E$に対して$H \models F$を満たすHを求めればよい

提案手法

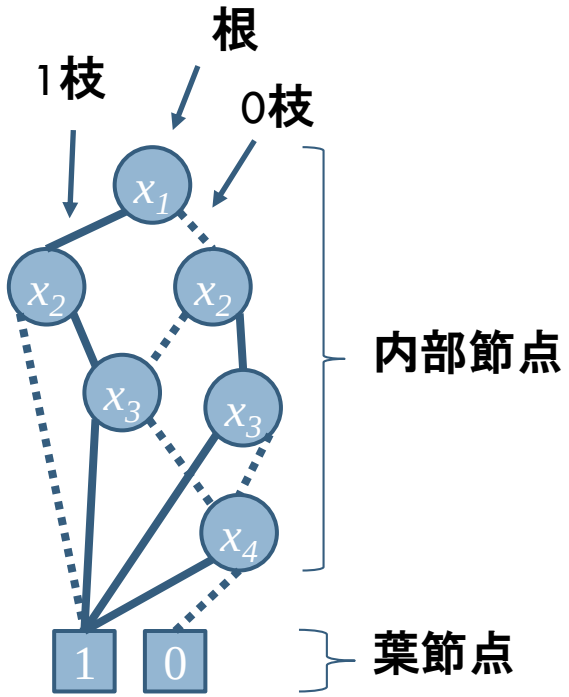
この問題に対して二分決定グラフ(BDD)を入出力とし,
中間グラフと呼ぶ新しいグラフを作り利用する

列挙に必要な情報をコンパクトに保存,
グラフ間の効率的な論理演算が可能

二分決定グラフ

- 二分決定グラフ(binary decision diagram)は図のような有向非循環グラフ
  - 論理関数を表すデータ構造として用いられる
  - 図のBDDは4変数の論理関数を表す

- 各内部節点は変数のラベルを持つ
- 関数の入力が得られたとき，根から節点のラベルに対応する変数の値に従い枝を辿る
  - 到達した葉節点が出力を表す

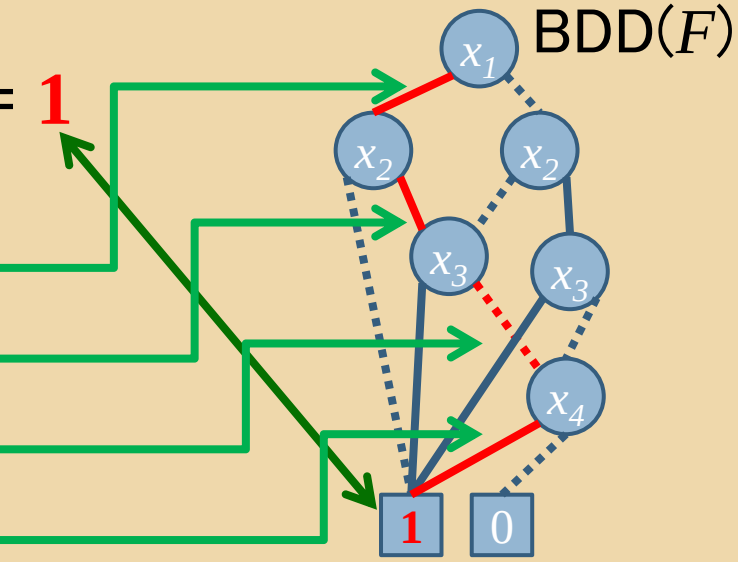


論理式と二分決定グラフ(BDD)

論理式の解釈とBDDのパスが対応している

$F=(x_1 \wedge \neg x_2) \vee x_3 \vee (x_2 \wedge x_4) = 1$

解釈
- $x_1=1$
- $x_2=1$
- $x_3=0$
- $x_4=1$



- BDDは本来論理関数を表すデータ構造

⇒ 各解釈に対する論理式の値を表すデータ構造ともみなせる

各解釈に対する論理式Fの値を表すBDDをBDD(F)と書く

前提の定義と構成法

前提の定義（$H \models F$）

命題論理式F, Hに関して，

Hを充足する解釈が全てFを充足するとき

HをFの前提と呼ぶ

パスと解釈が対応

BDDとして考えると

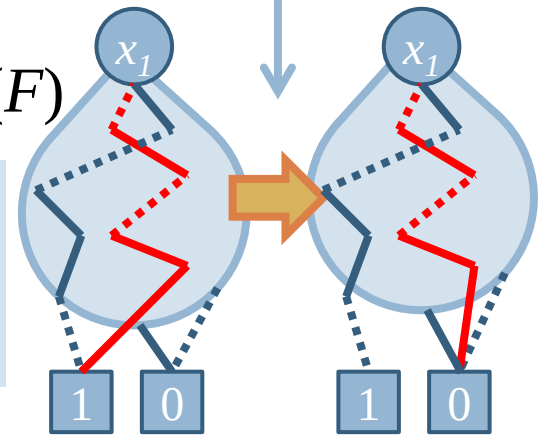
BDD(H)で1に至るパスに対応する全てのBDD(F)のパスが1に至る

BDD(F)で1に至るパスを0に繋ぎ直すことで

BDD(H)を構成できる

BDD(F)

BDD(H)



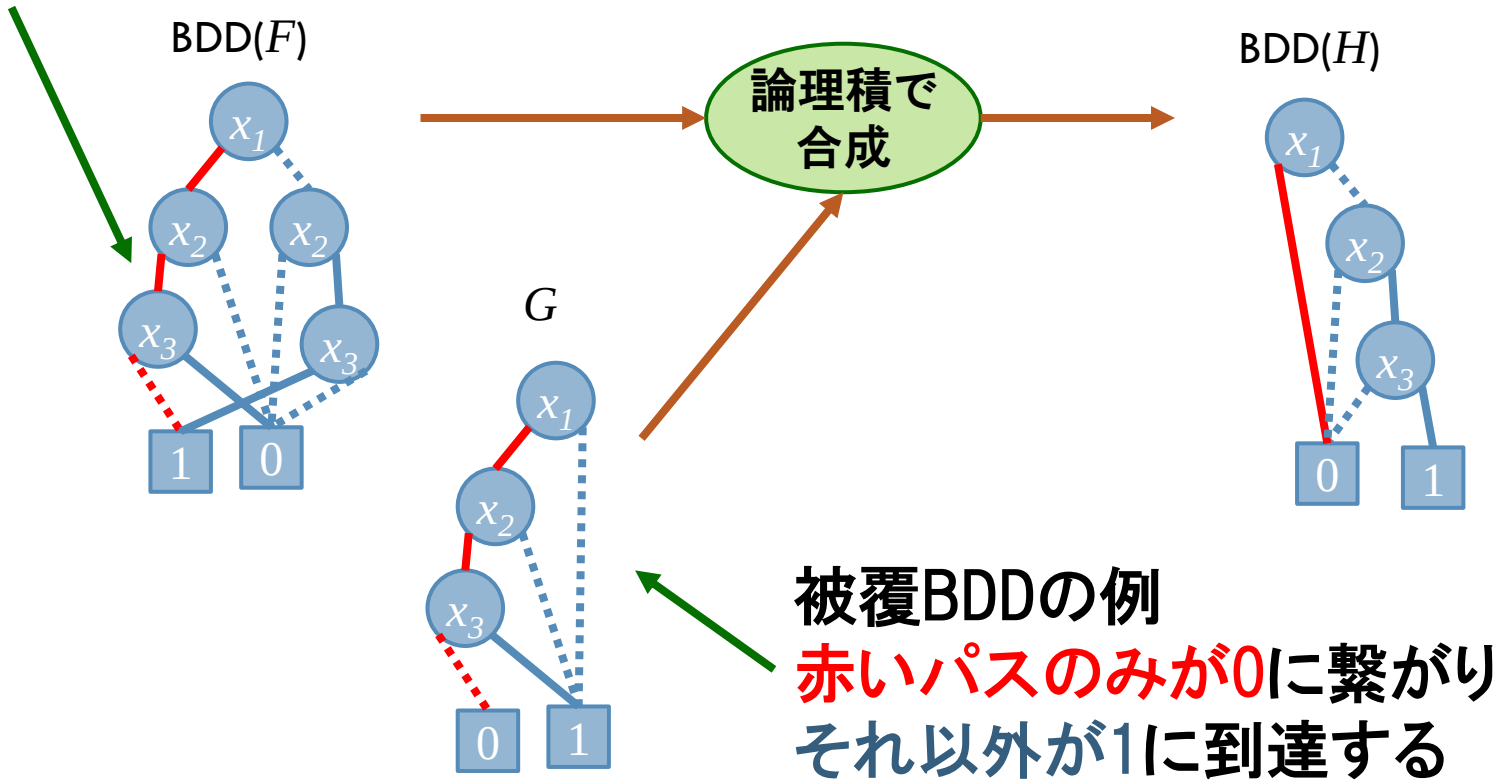
BDDのパスを直接操作するのは一般に困難

BDD間の論理演算（合成）を用いる

合成によるパスの操作

- BDD(F)で1に至るパスを0につなぎ換えるという操作は
  BDD(F)をGのようなBDDと論理積で合成することで実現できる
  - BDD(F)と合成するBDDを被覆BDDと呼ぶ

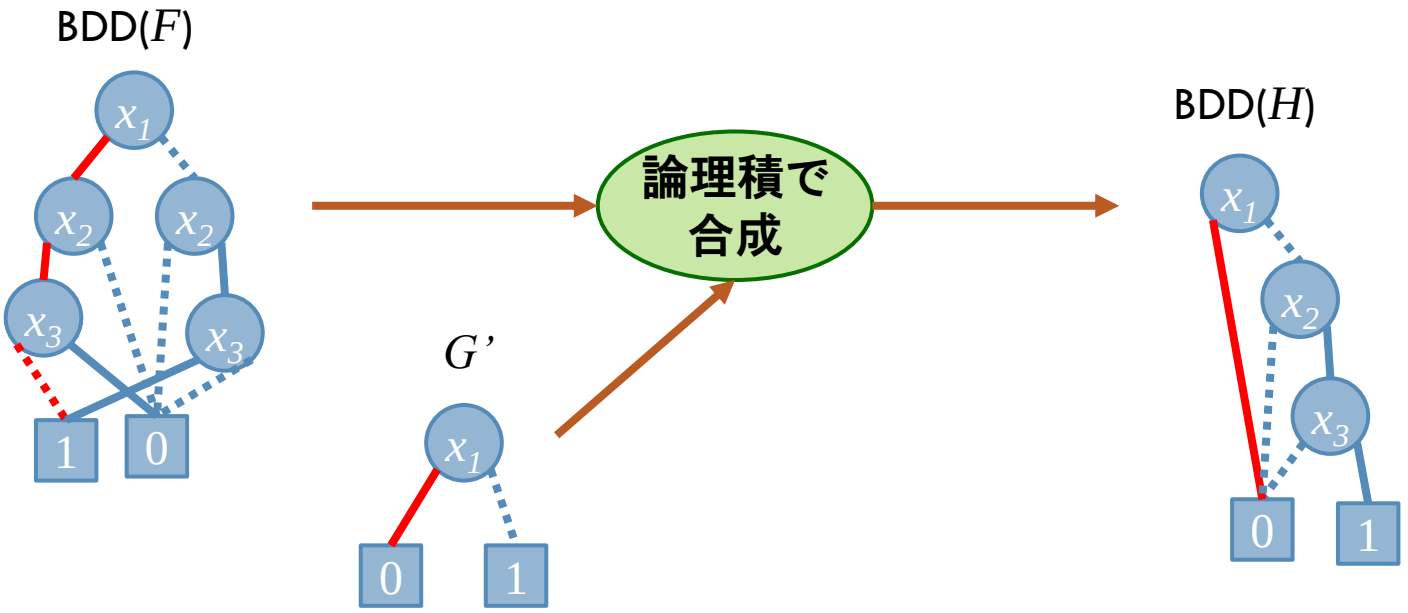
このパスを0につなぎ換える



被覆BDD

- 被覆BDDの取り方には自由度がある
  - 前スライドのGの代わりにG'を使っても同じようにパスを操作できる
- BDDの合成は節点数に依存する

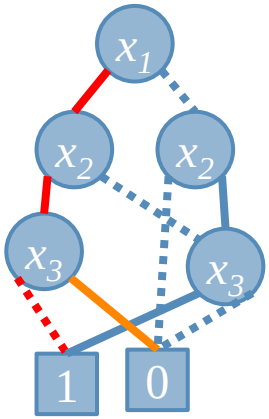
⇒ できるだけ少ない節点数の被覆BDDを作る



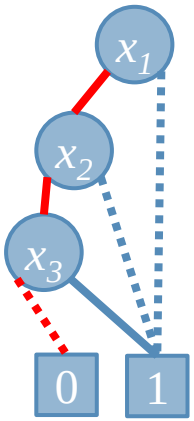
最小の被覆BDD

- BDD(F)の赤いパスを0につなぎ換えるとする
- 単純に被覆BDDを作るとGのようになる

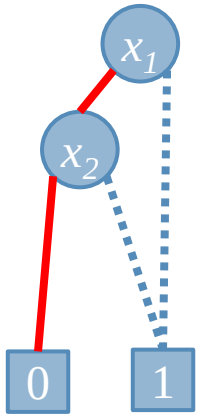
BDD(F)



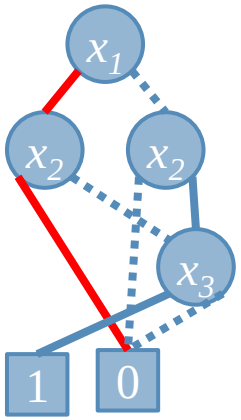
被覆BDD G



最小の被覆BDD



BDD(H)

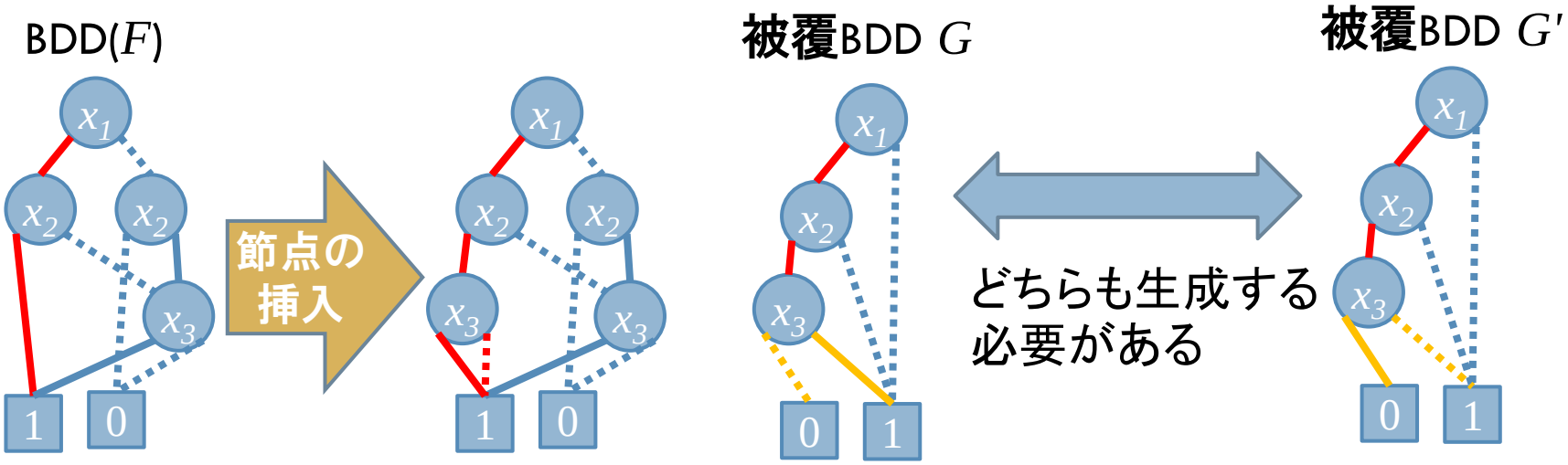


- パス上のx_3の節点は，**一方の枝**が0につながっている

BDD(F)において一方の枝が0に繋がっているような節点は飛ばしてもBDD(H)の他のパスに影響はない

被覆BDDを作る時の注意点

- BDD(F)のように節点を飛ばした枝がある場合には
  そのパスをたどる入力が複数存在する
  - BDD(F)において$(x_1, x_2, x_3) = (1, 1, 0), (1, 1, 1)$の両方の入力が
    赤いパスを通る



- それぞれの入力に対応するパスを0につなぎ換える
  被覆BDDを作る必要がある

  ⇒ 節点を挿入すると二つのパスを区別できる

中間グラフ

- 被覆BDDを効率的に生成するためBDD(F)から中間グラフと呼ぶグラフを生成する
- 被覆BDDに関する2つの性質を踏まえて，以下の操作を施したグラフを中間グラフとする

1. 一方の枝が0に繋がる節点は飛ばすことができる

   ⇒ そのような節点は削除する

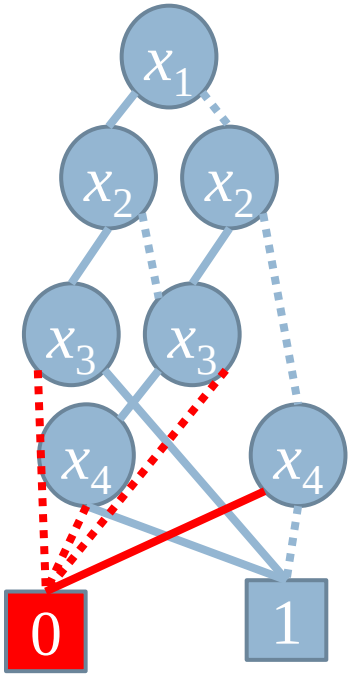
2. 節点を飛ばした枝を通る複数の入力に対してそれぞれパスをつなぎ換える必要がある

   ⇒ 節点を飛ばした枝には節点を挿入する

中間グラフの構成手続き

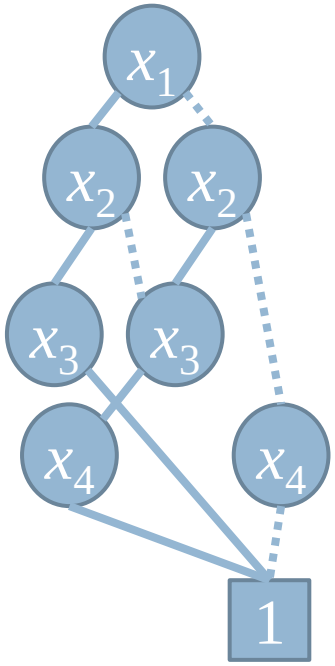
1. 0に繋がる枝を全て削除する
2. 変数を飛ばした枝にノードを挿入する
3. 子が一つだけになったノードnを削除し，nに繋がっていた枝をnの残る子に繋ぐ

入力されたBDD



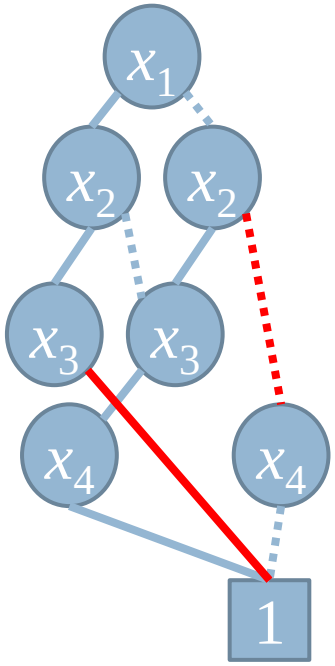
中間グラフの構成手続き

1. 0に繋がる枝を全て削除する
2. 変数を飛ばした枝にノードを挿入する
3. 子が一つだけになったノードnを削除し，nに繋がっていた枝をnの残る子に繋ぐ



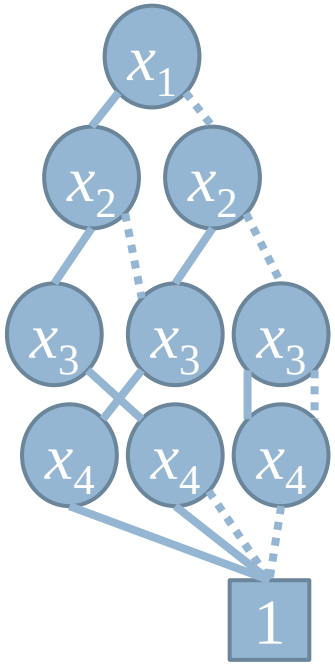
中間グラフの構成手続き

1. 0に繋がる枝を全て削除する
2. 変数を飛ばした枝にノードを挿入する
3. 子が一つだけになったノードnを削除し，nに繋がっていた枝をnの残る子に繋ぐ



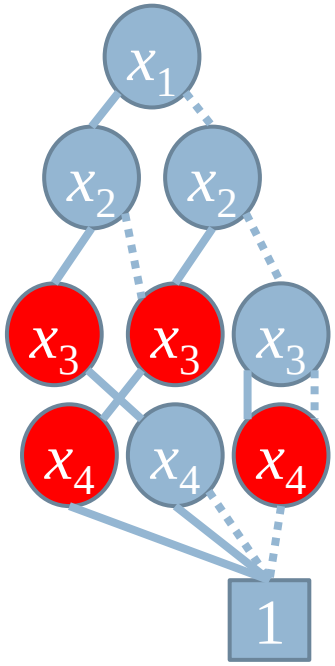
中間グラフの構成手続き

1. 0に繋がる枝を全て削除する
2. 変数を飛ばした枝にノードを挿入する
3. 子が一つだけになったノードnを削除し，nに繋がっていた枝をnの残る子に繋ぐ



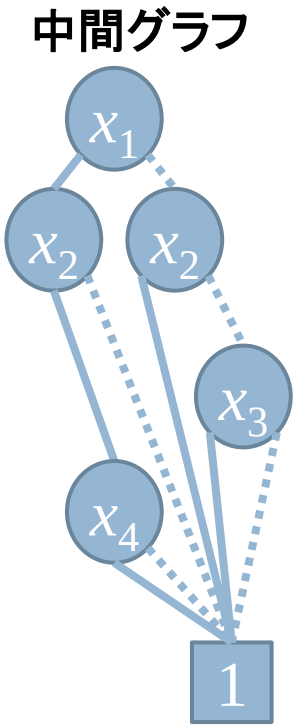
中間グラフの構成手続き

1. 0に繋がる枝を全て削除する
2. 変数を飛ばした枝にノードを挿入する
3. 子が一つだけになったノードnを削除し，
   nに繋がっていた枝をnの残る子に繋ぐ



中間グラフの構成手続き

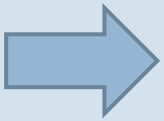
1. 0に繋がる枝を全て削除する
2. 変数を飛ばした枝にノードを挿入する
3. 子が一つだけになったノードnを削除し，
   nに繋がっていた枝をnの残る子に繋ぐ



被覆BDDの生成

- 中間グラフの枝を0に繋ぎ替えることで被覆BDDを生成することができる
  - 中間グラフの各パスはBDD(F)で1に至るパスと1対1対応している

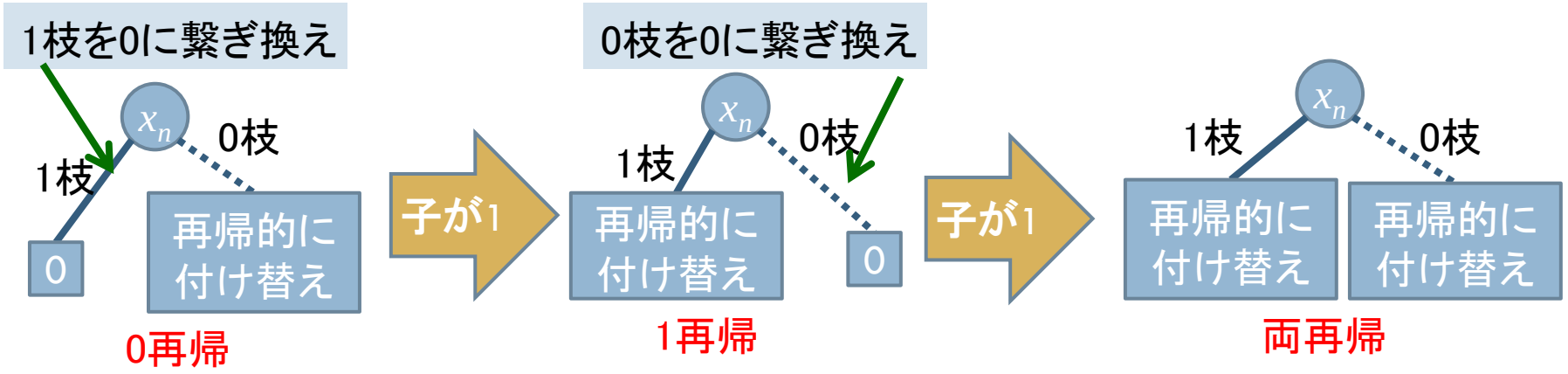
- 中間グラフの枝の付け替え方のすべての組み合わせに対応する被覆グラフを生成する必要がある

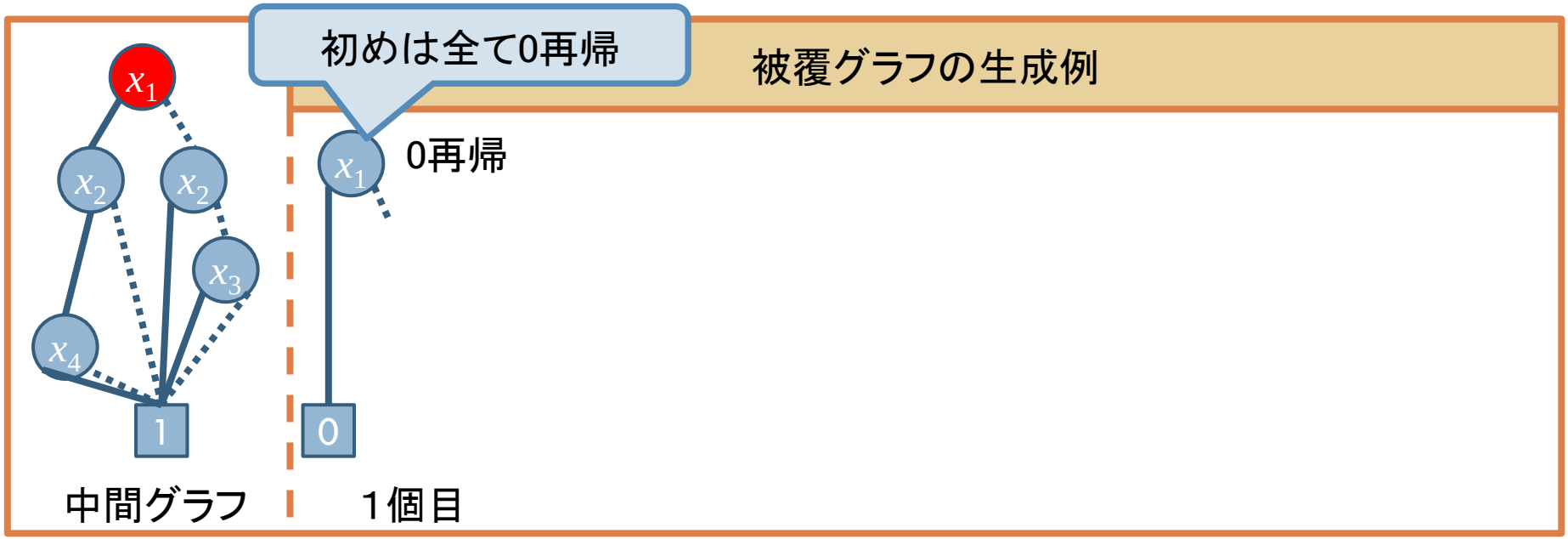


列挙において一つ前に生成した被覆BDDを参照することで効率的に被覆BDDの生成を行う

被覆BDDの生成手続き

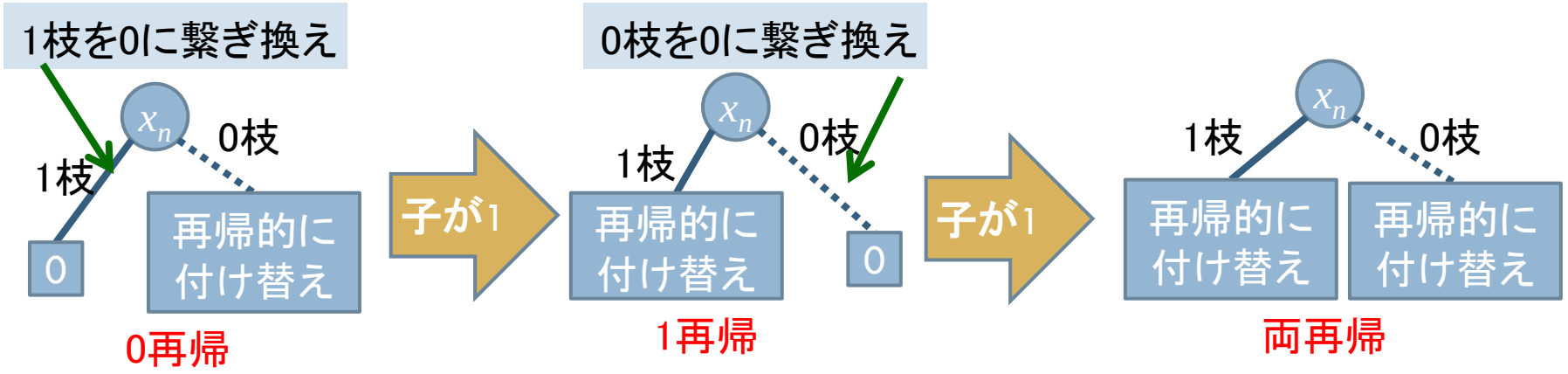
中間グラフのパスを0に繋ぎ換える

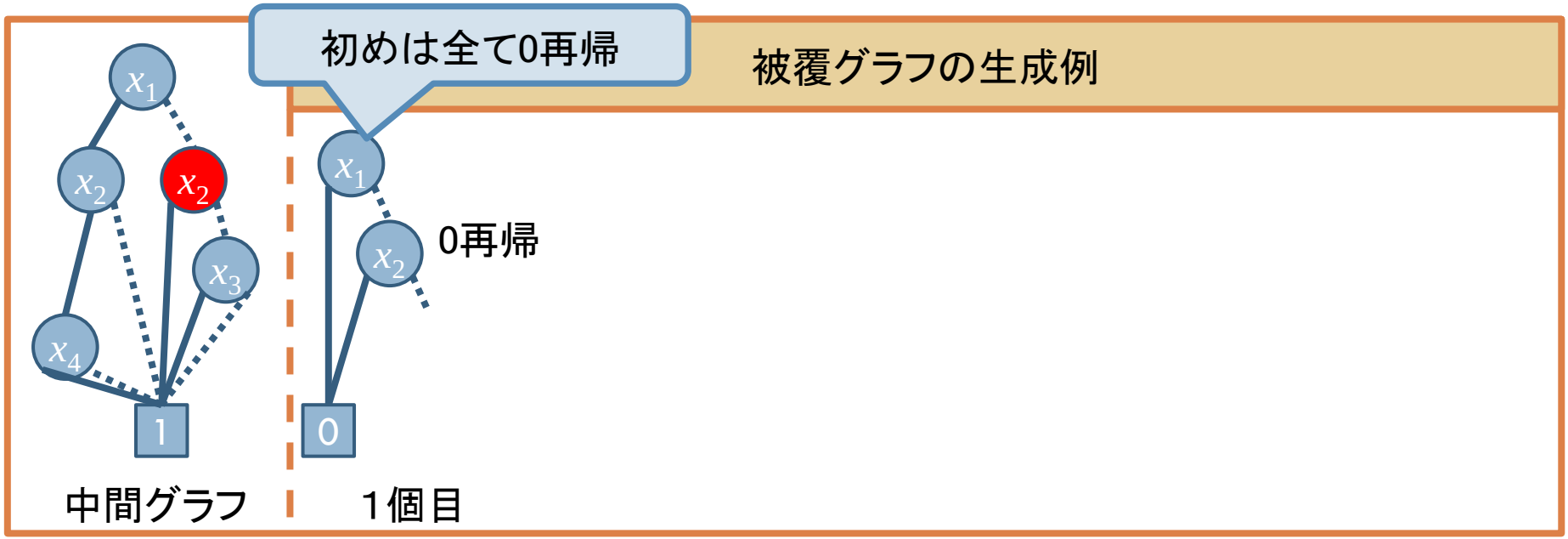




被覆BDDの生成手続き

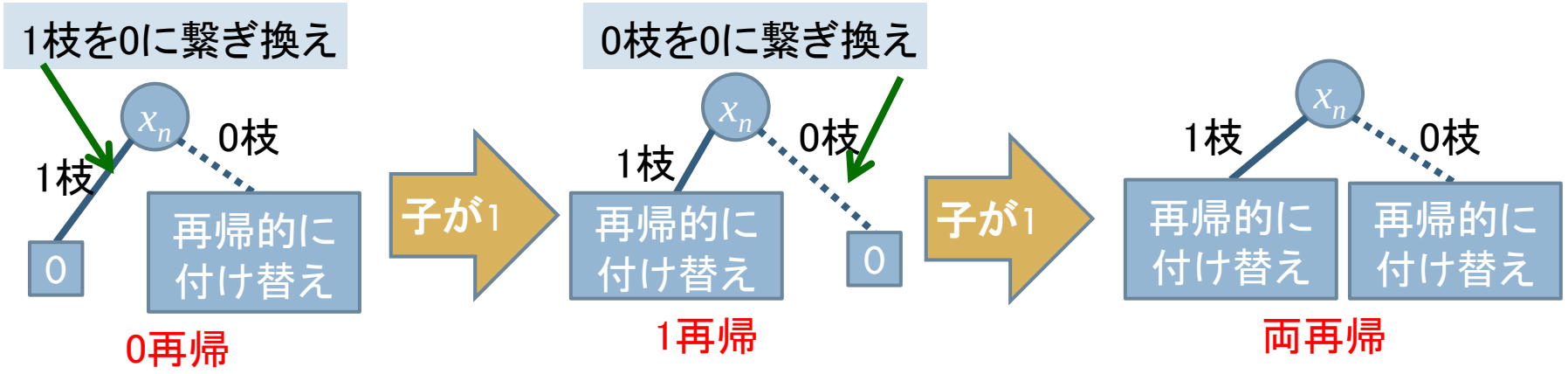
中間グラフのパスを0に繋ぎ換える

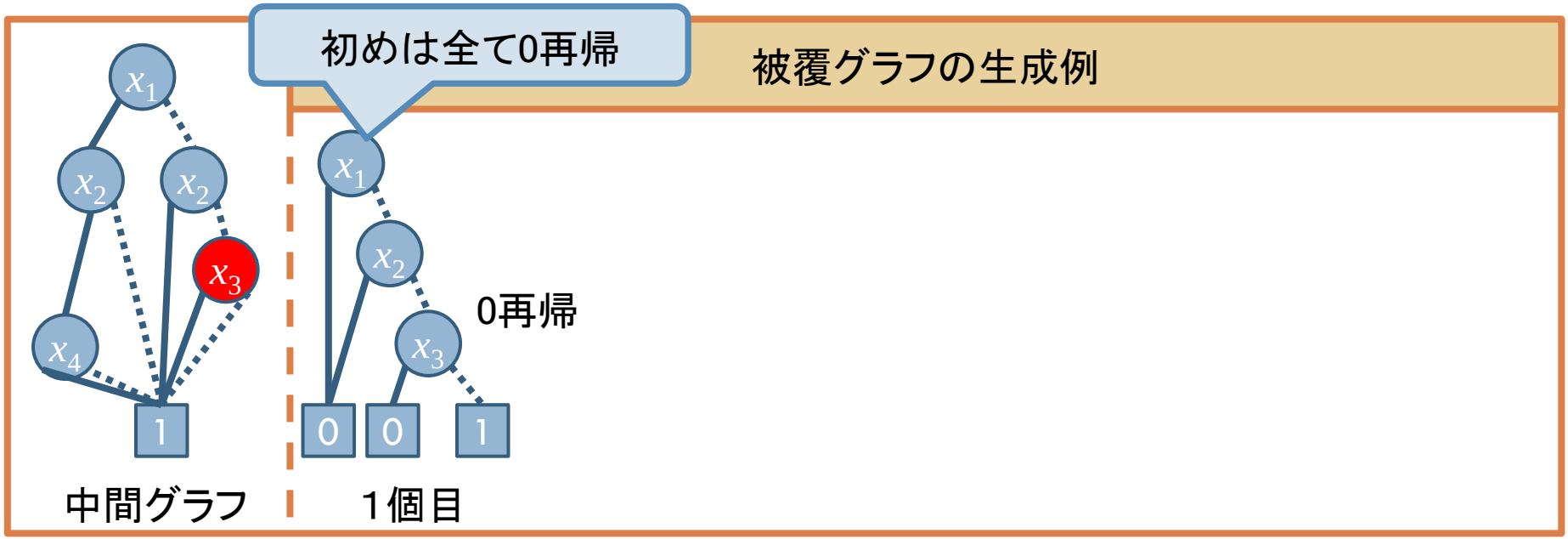




被覆BDDの生成手続き

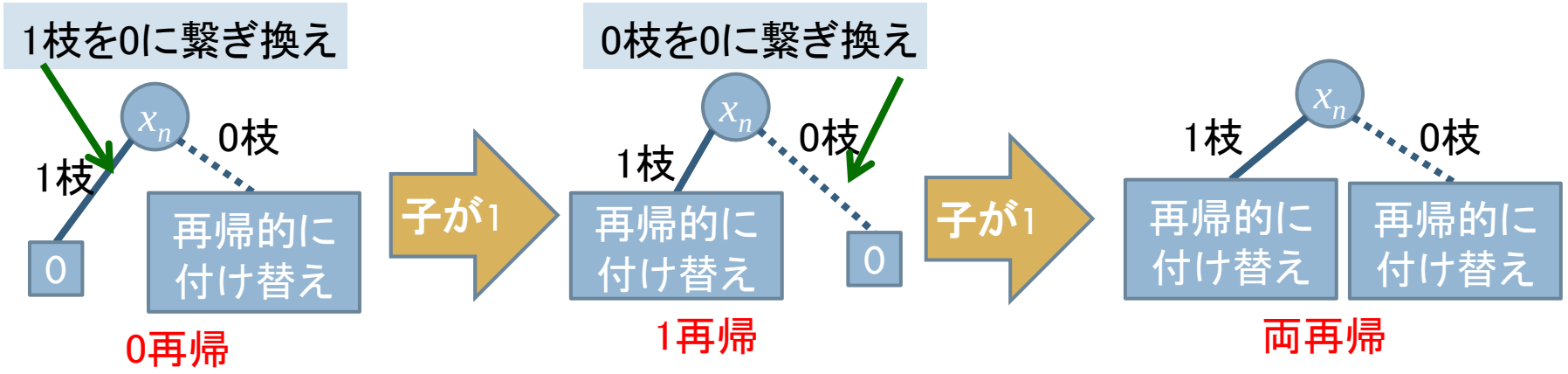
中間グラフのパスを0に繋ぎ換える

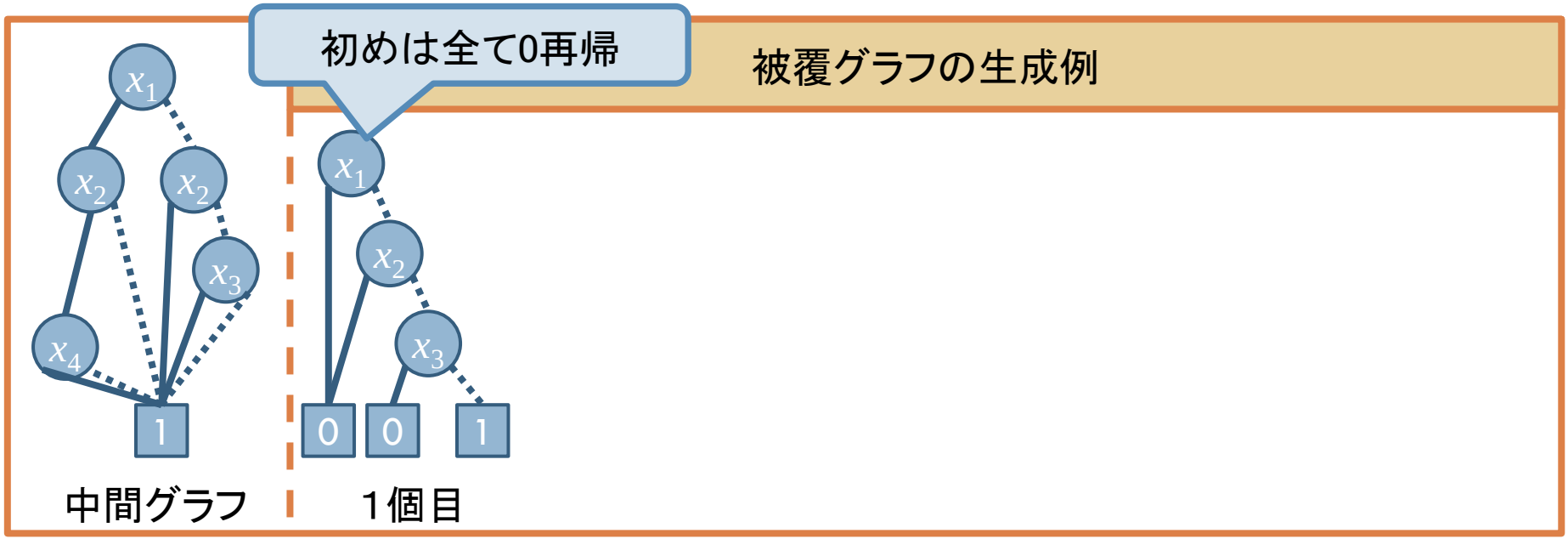




被覆BDDの生成手続き

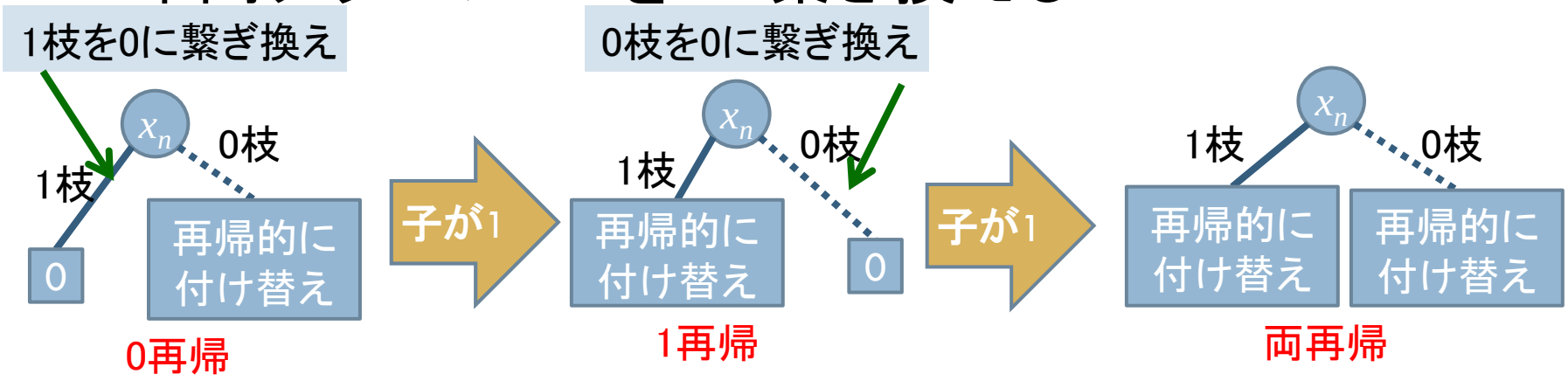
中間グラフのパスを0に繋ぎ換える

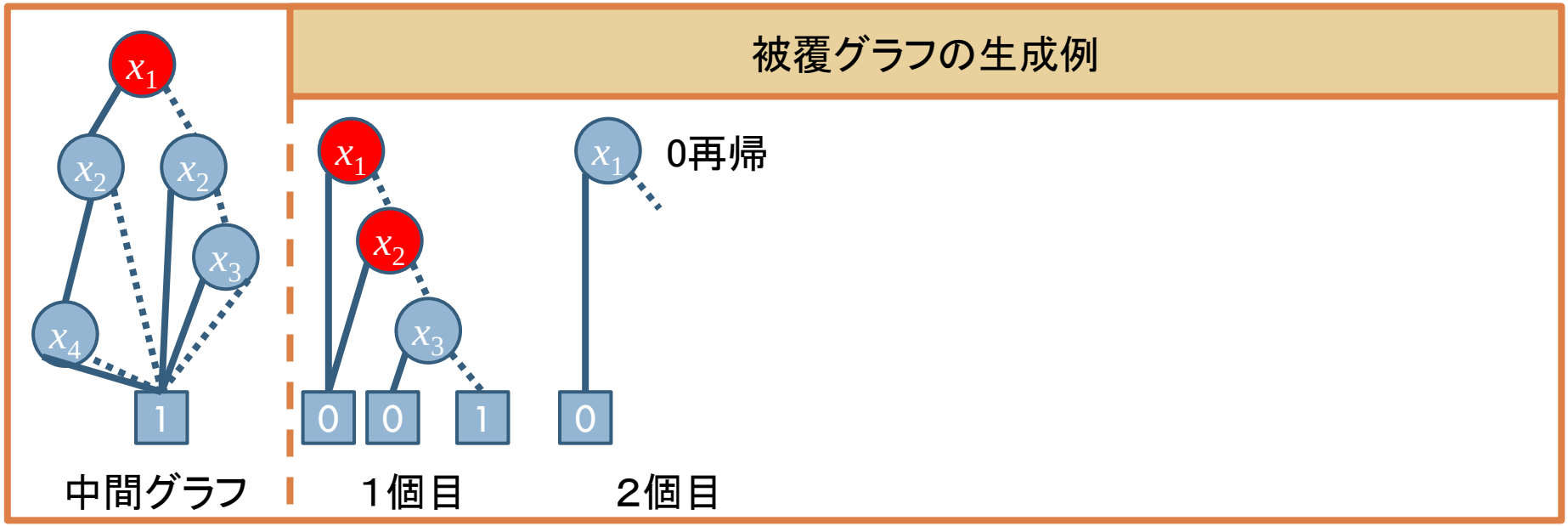




被覆BDDの生成手続き

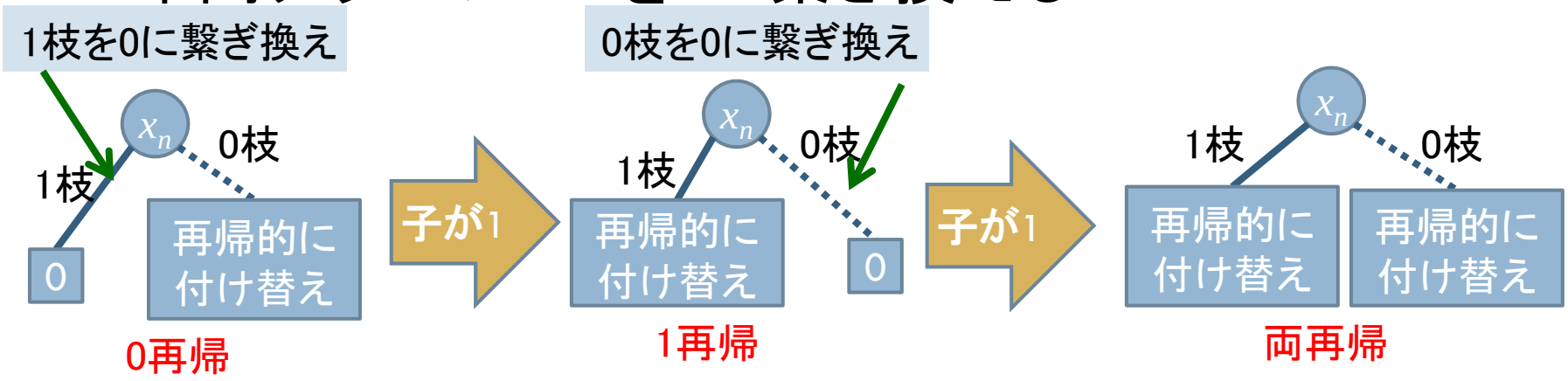
中間グラフのパスを0に繋ぎ換える

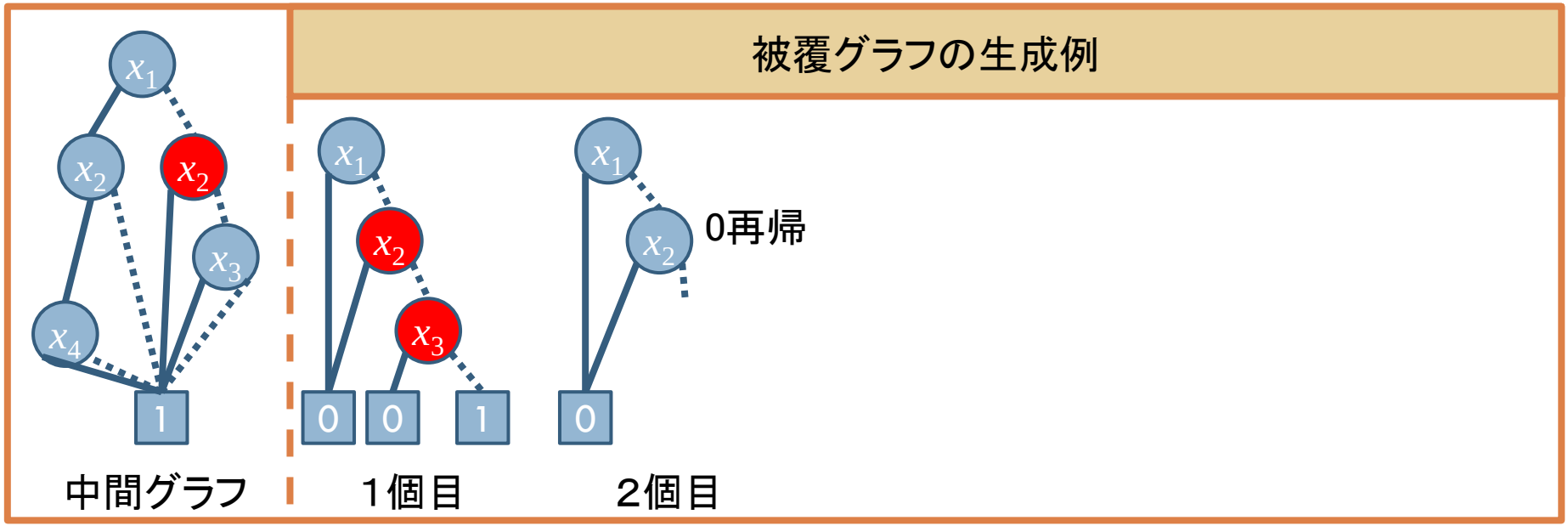




被覆BDDの生成手続き

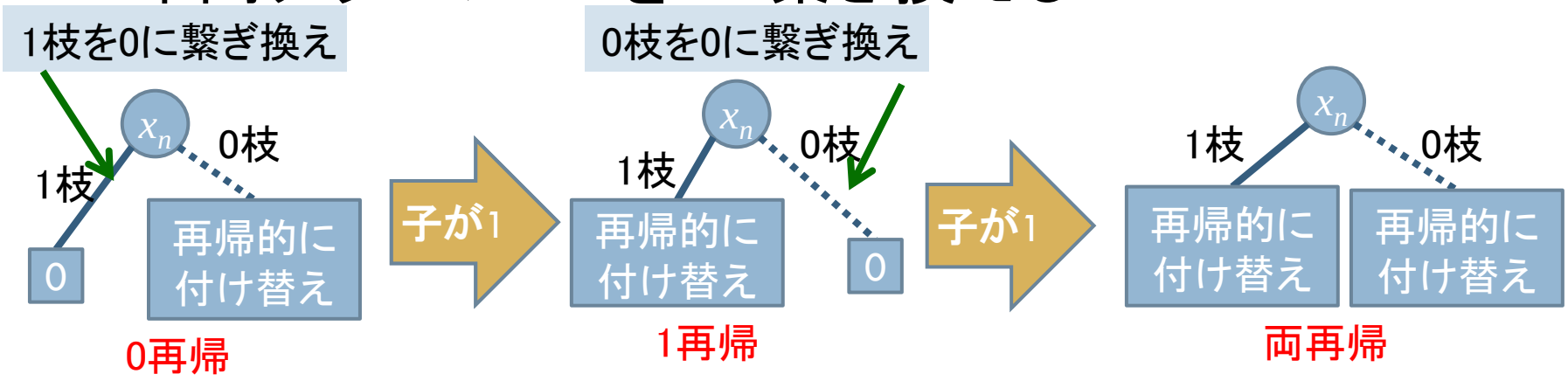
中間グラフのパスを0に繋ぎ換える



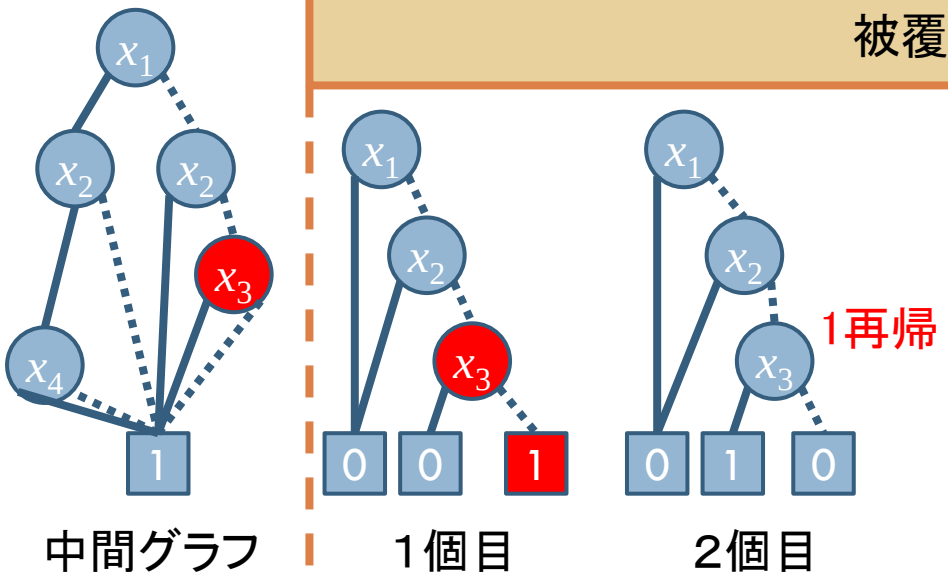


被覆BDDの生成手続き

中間グラフのパスを0に繋ぎ換える

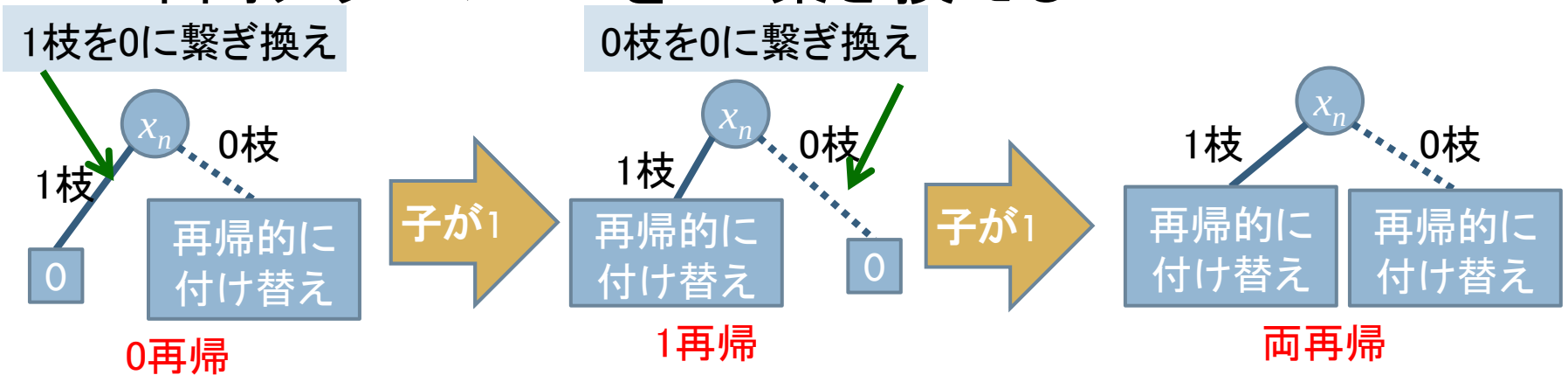


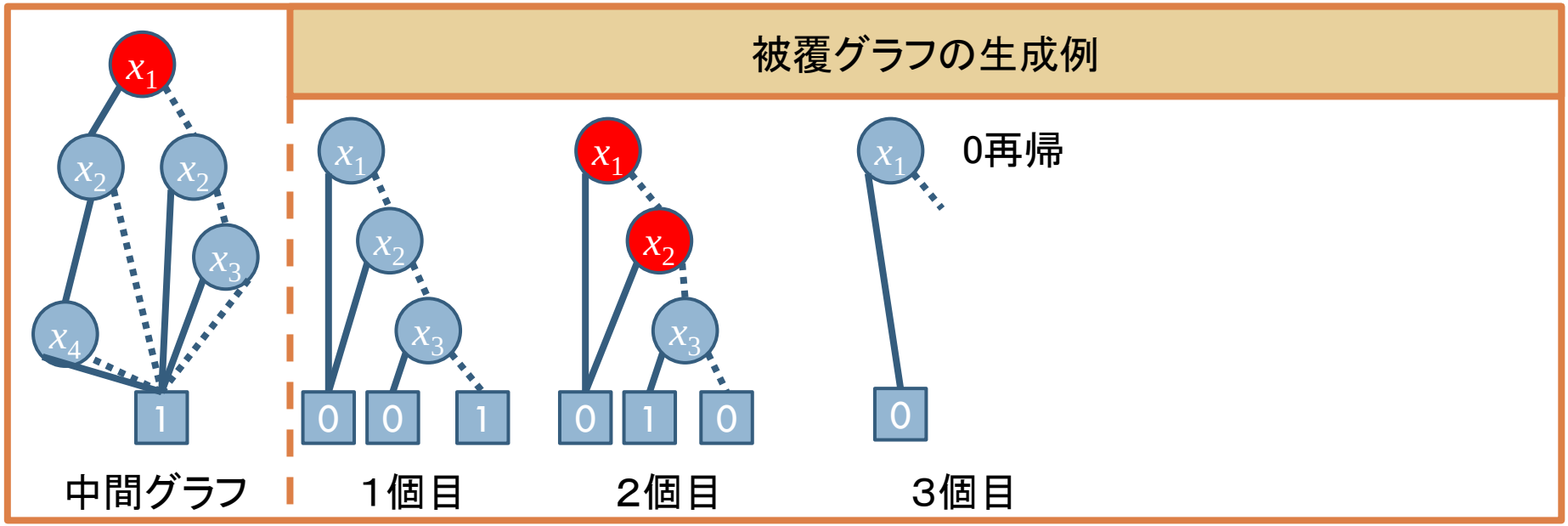
被覆グラフの生成例



被覆BDDの生成手続き

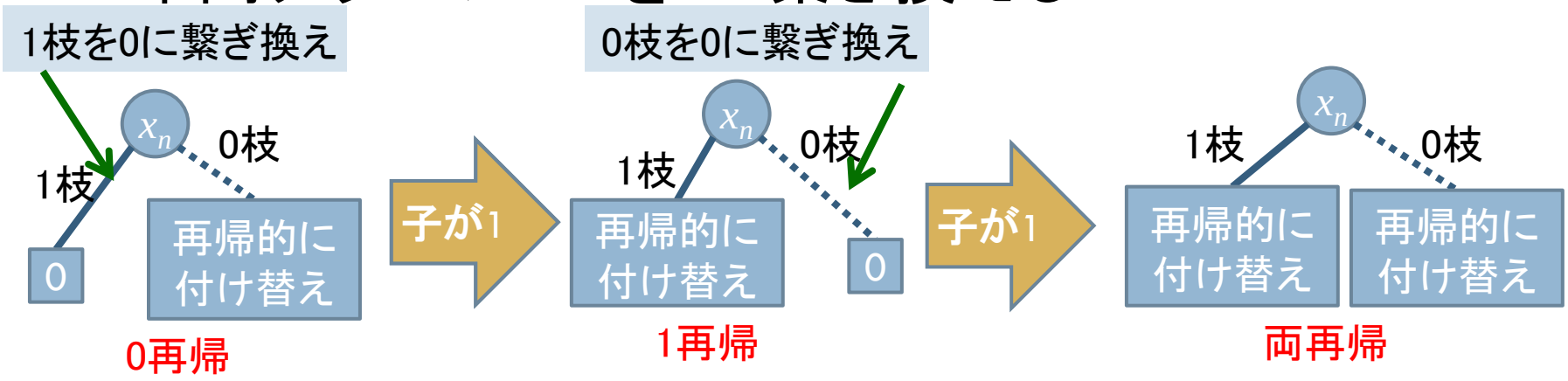
中間グラフのパスを0に繋ぎ換える

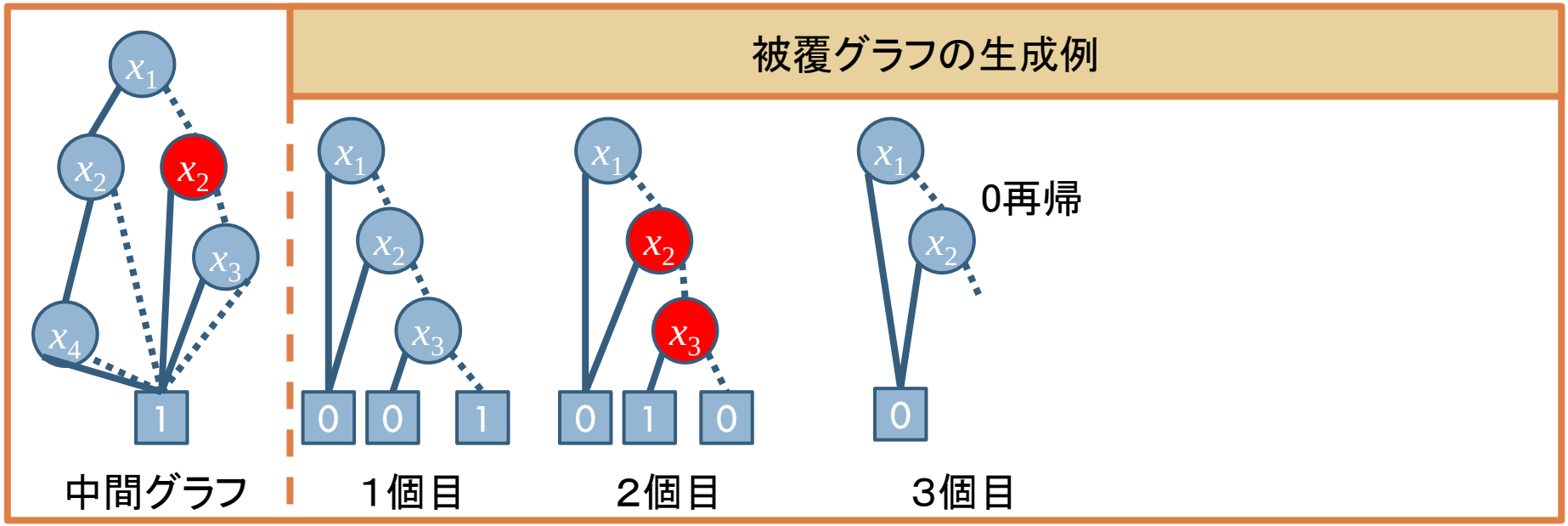




被覆BDDの生成手続き

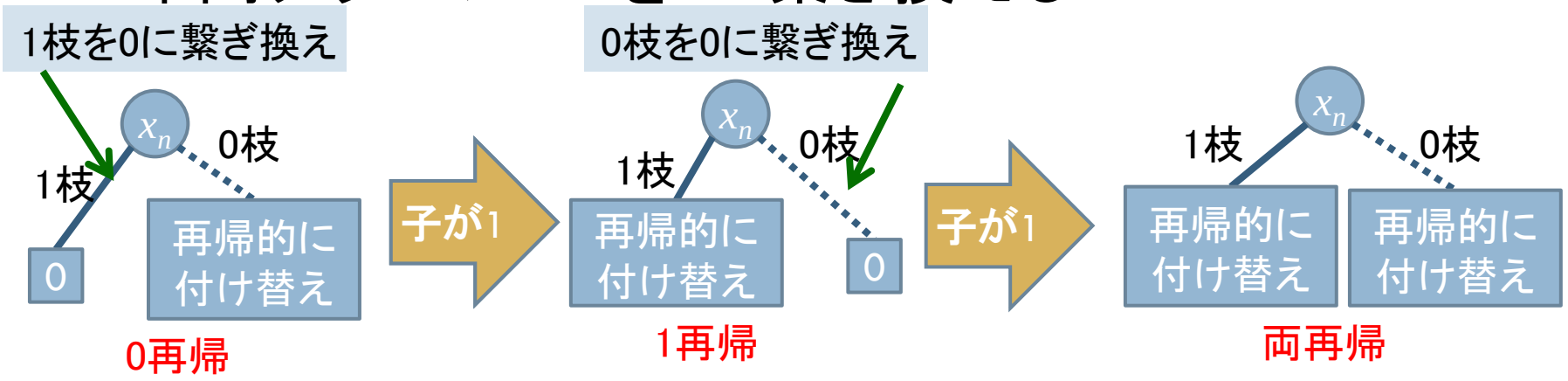
中間グラフのパスを0に繋ぎ換える

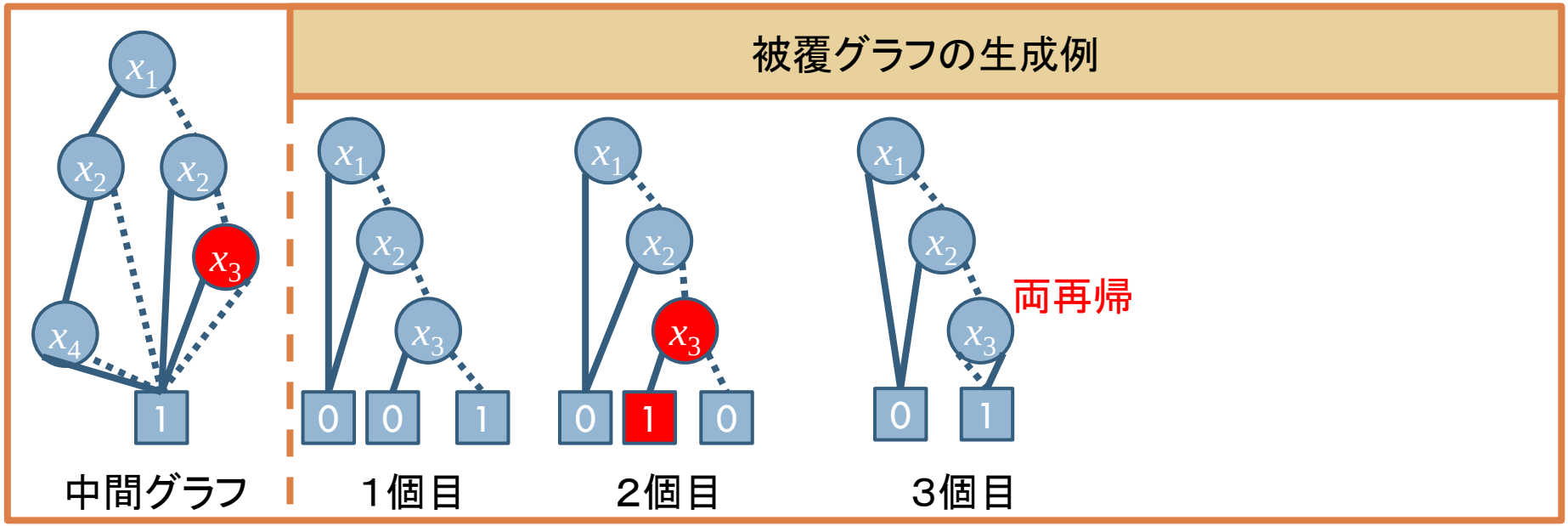




被覆BDDの生成手続き

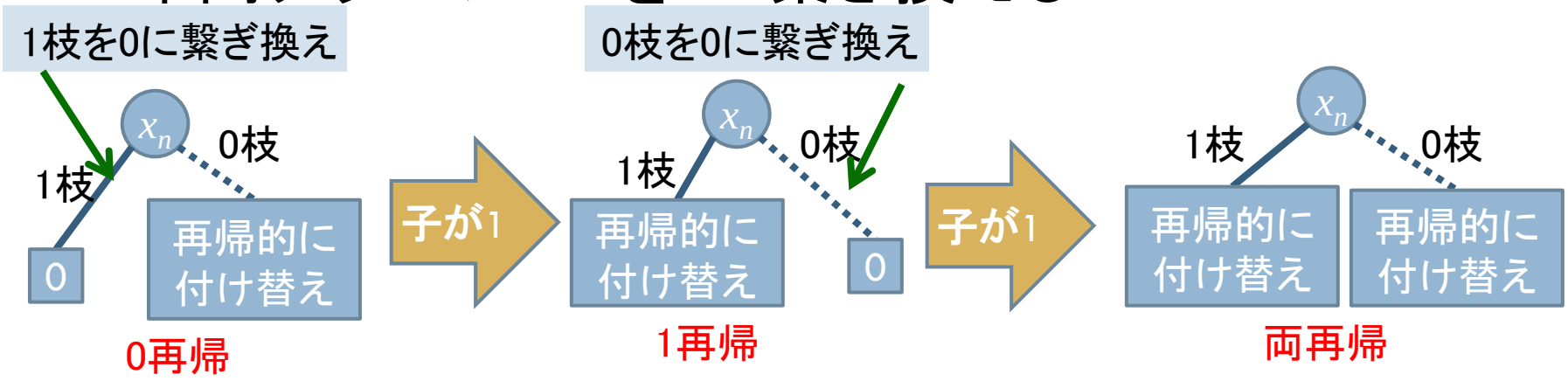
中間グラフのパスを0に繋ぎ換える



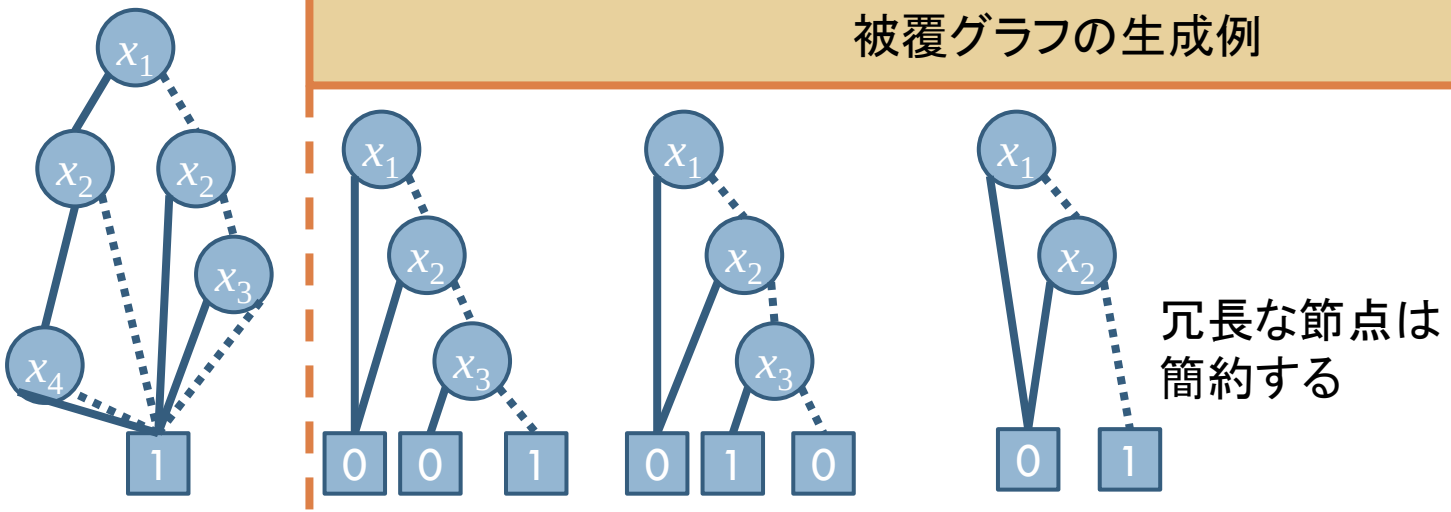


被覆BDDの生成手続き

中間グラフのパスを0に繋ぎ換える

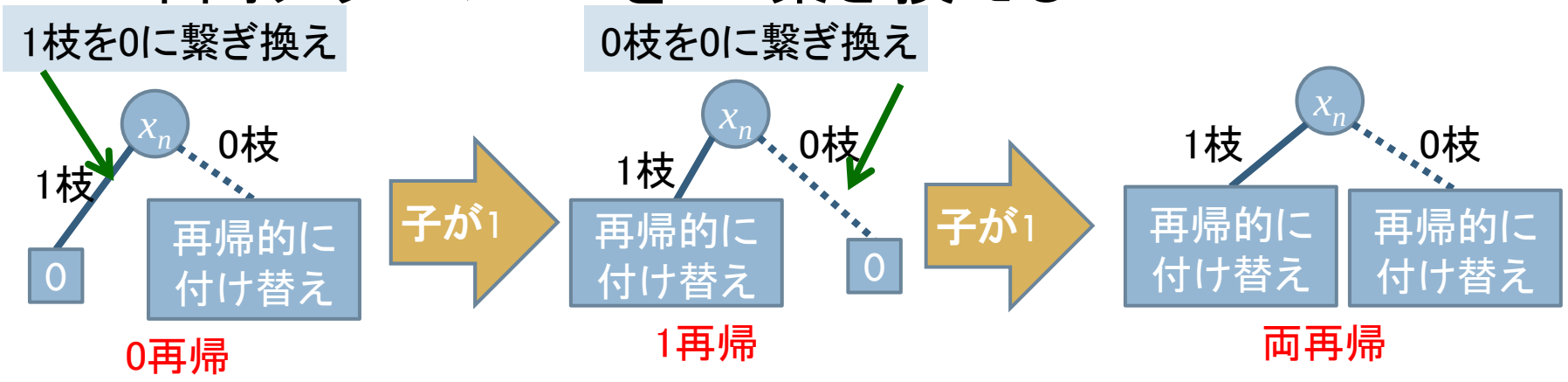


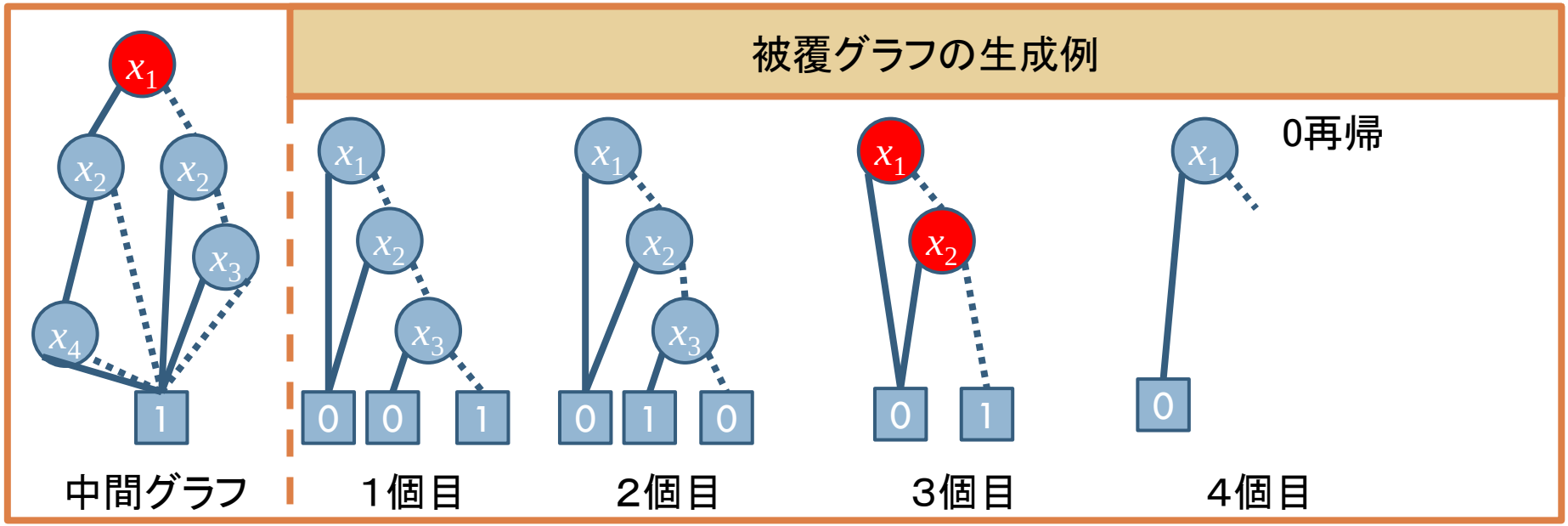
被覆グラフの生成例



被覆BDDの生成手続き

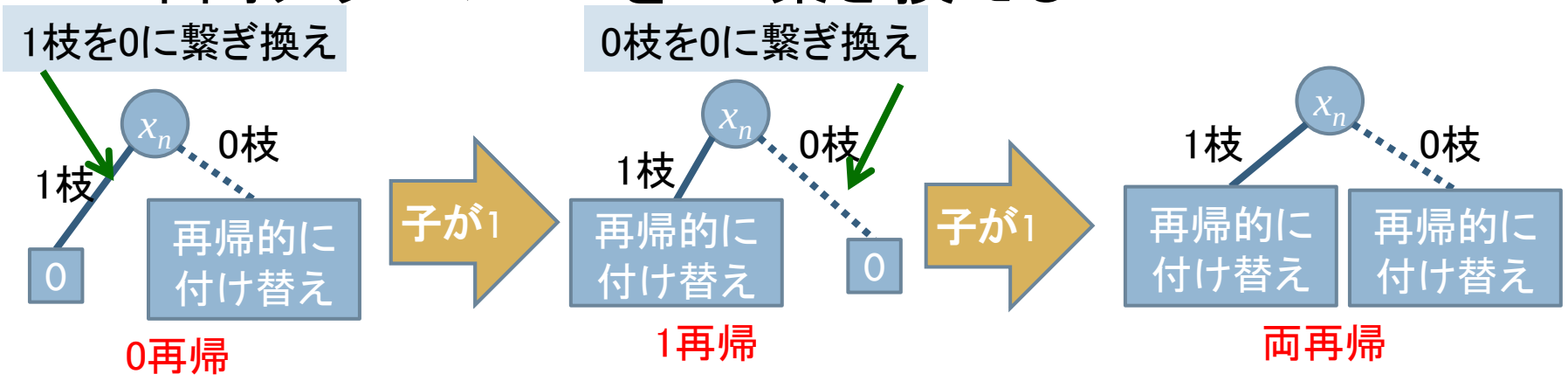
中間グラフのパスを0に繋ぎ換える

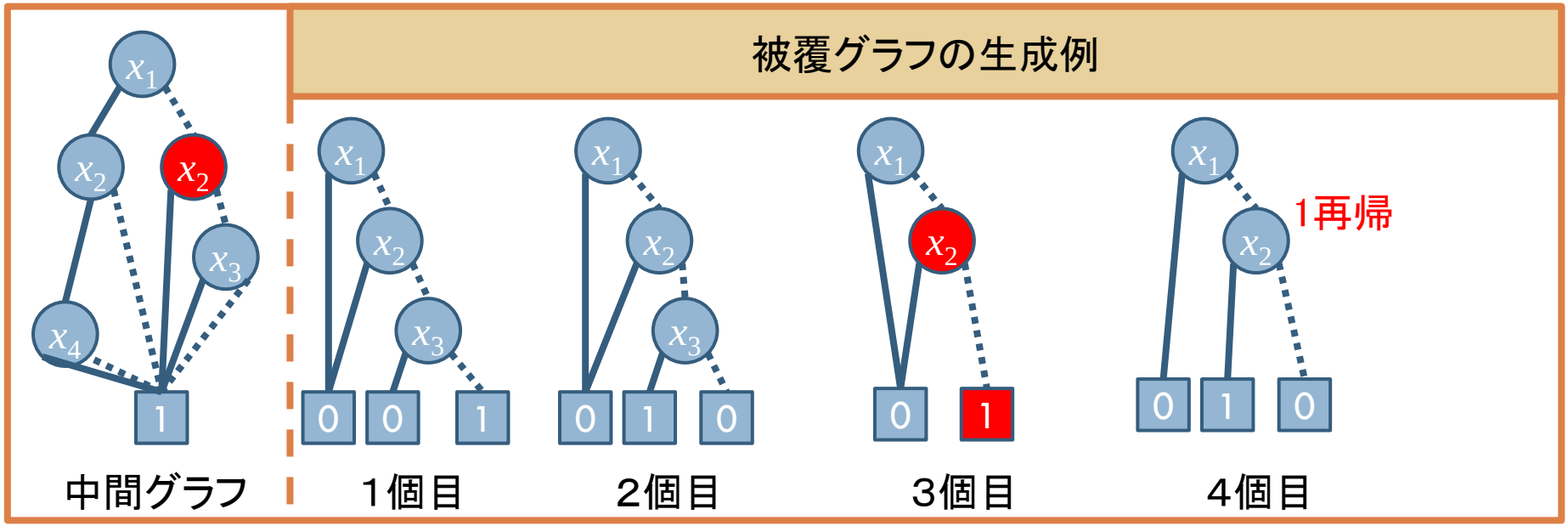




被覆BDDの生成手続き

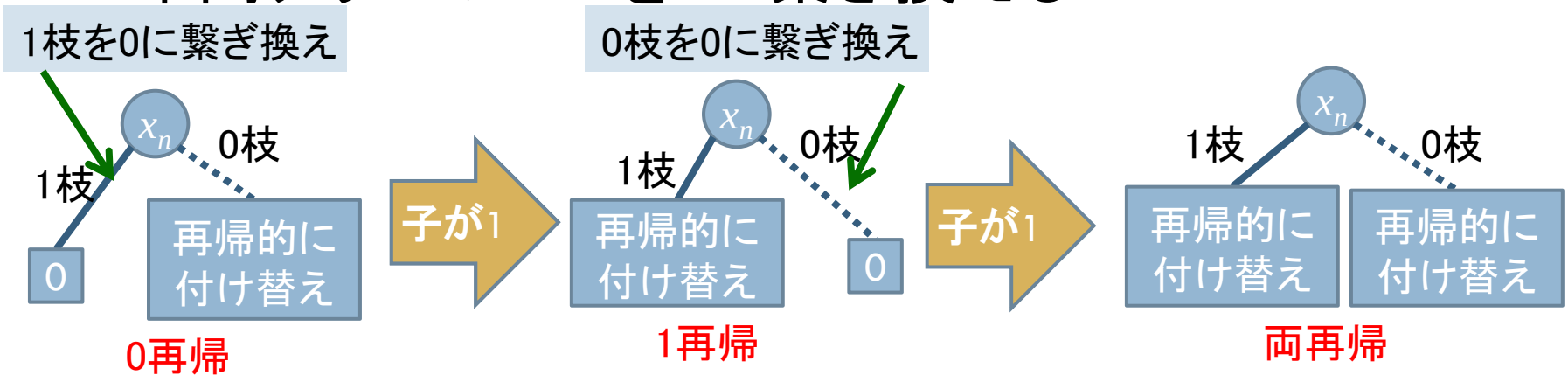
中間グラフのパスを0に繋ぎ換える

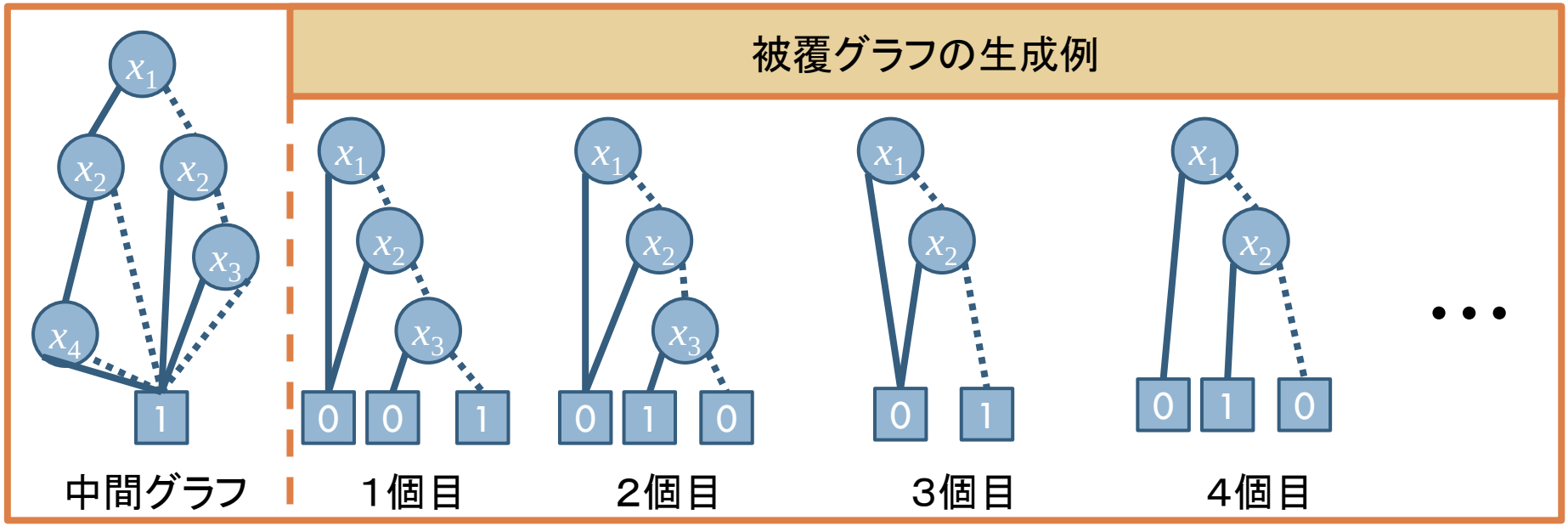




被覆BDDの生成手続き

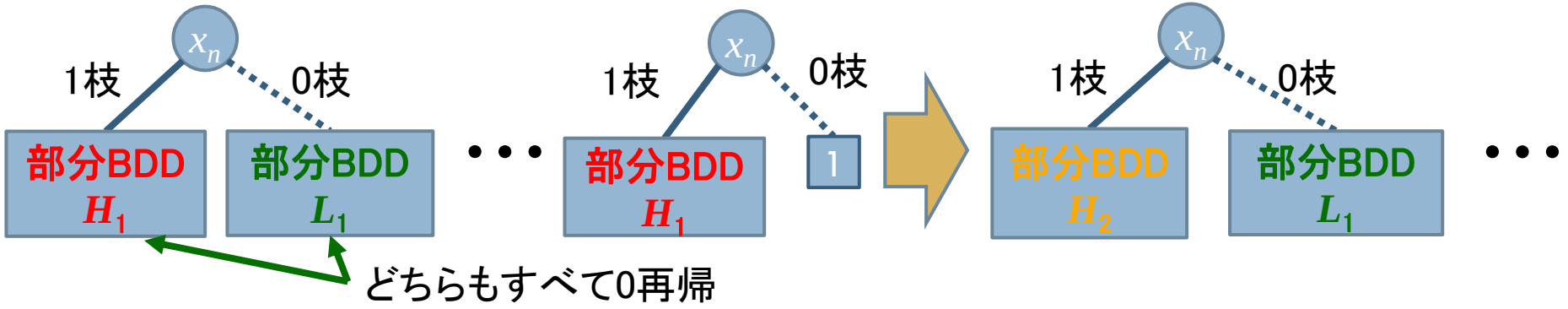
中間グラフのパスを0に繋ぎ換える



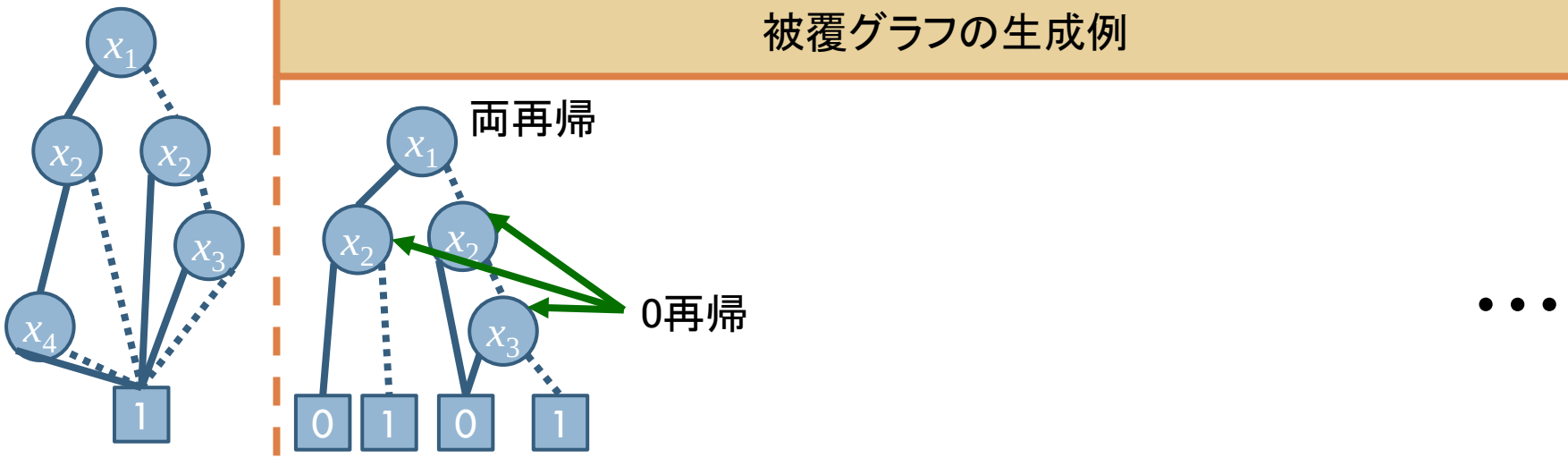


被覆BDDの生成手続き(両再帰)

中間グラフのパスを0に繋ぎ換える



被覆グラフの生成例

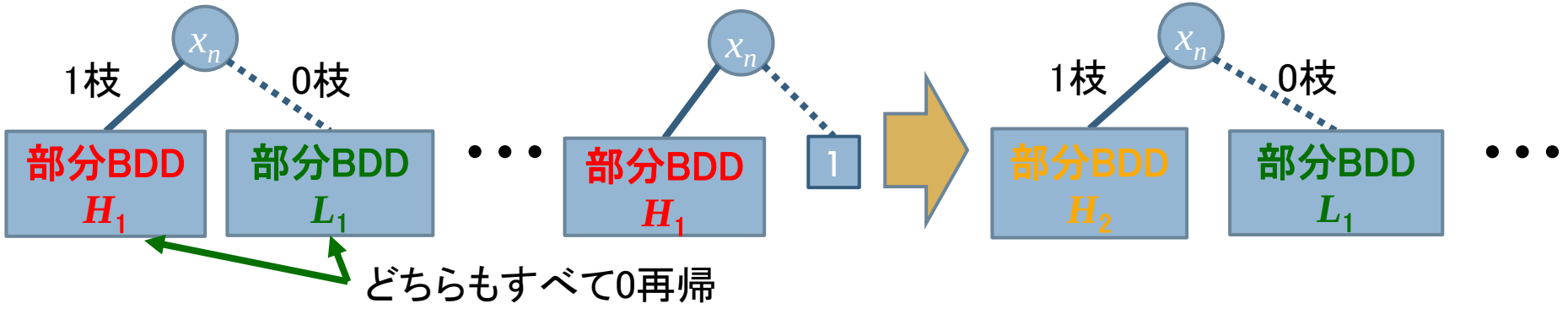


中間グラフ

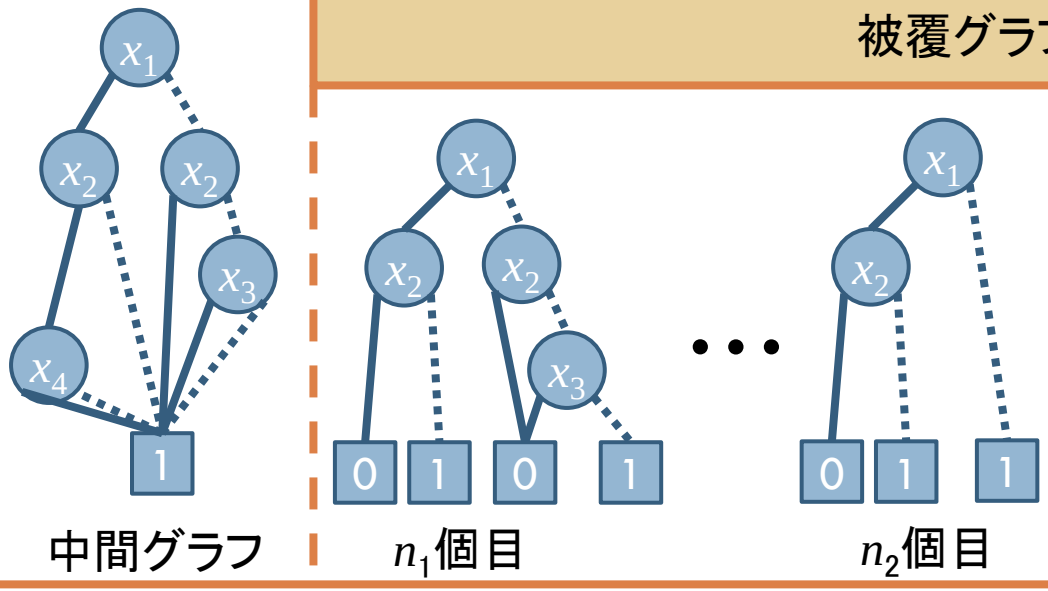
n_1個目（初めてx_1の節点が両再帰となった被覆BDD）

被覆BDDの生成手続き(両再帰)

中間グラフのパスを0に繋ぎ換える

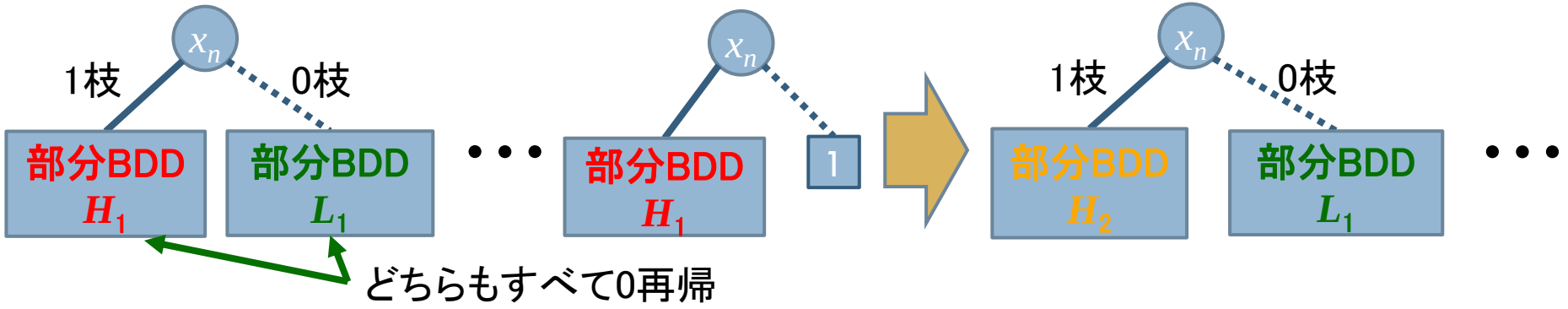


被覆グラフの生成例

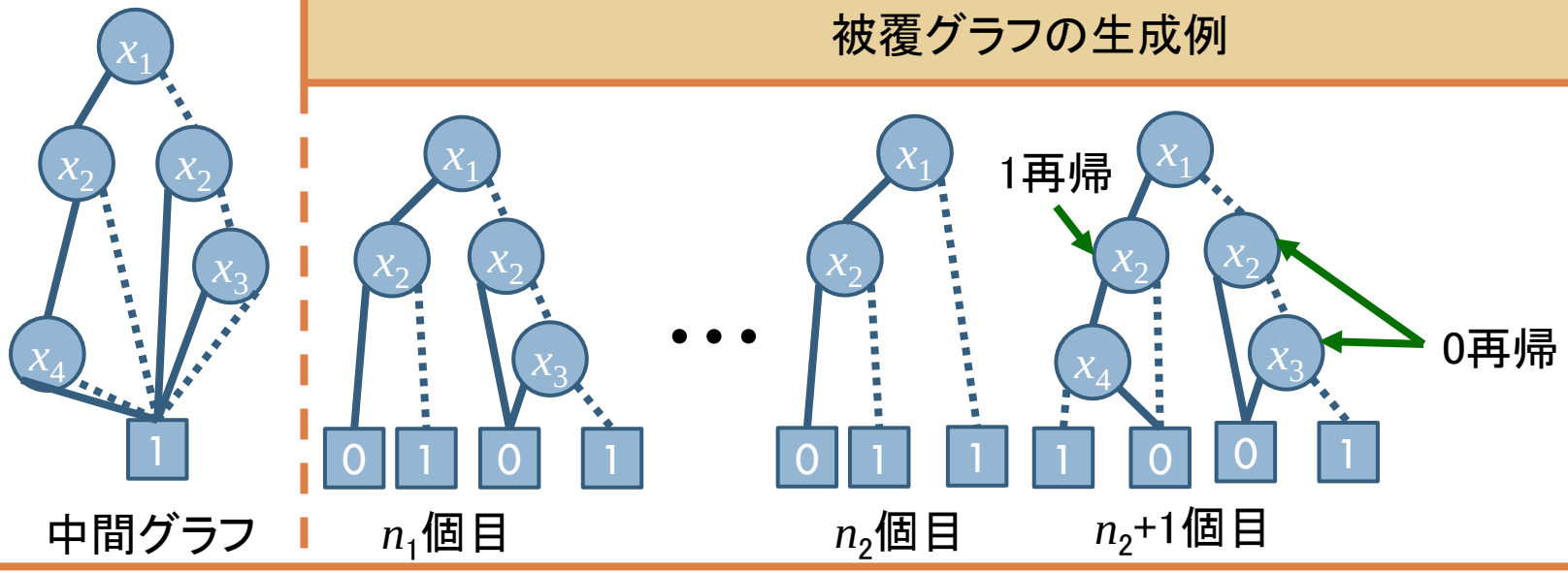


被覆BDDの生成手続き(両再帰)

中間グラフのパスを0に繋ぎ換える

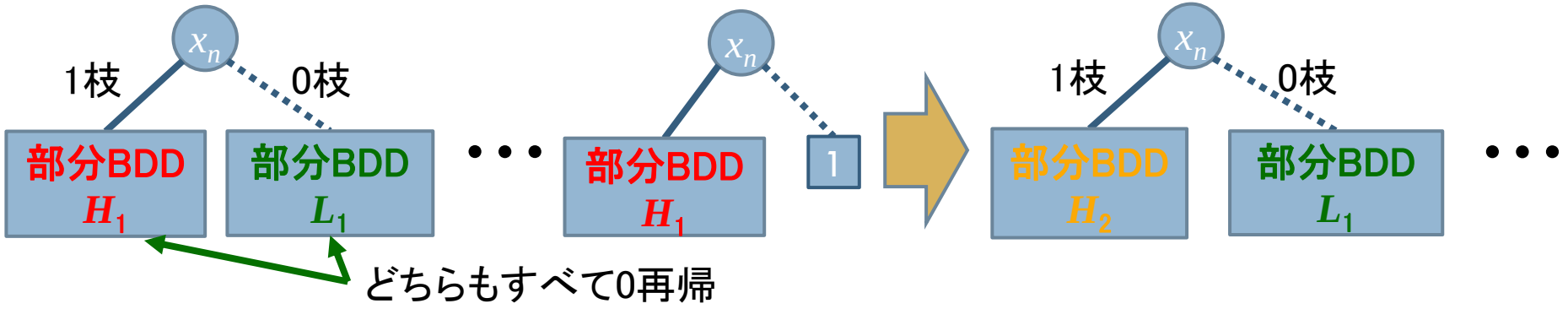


被覆グラフの生成例

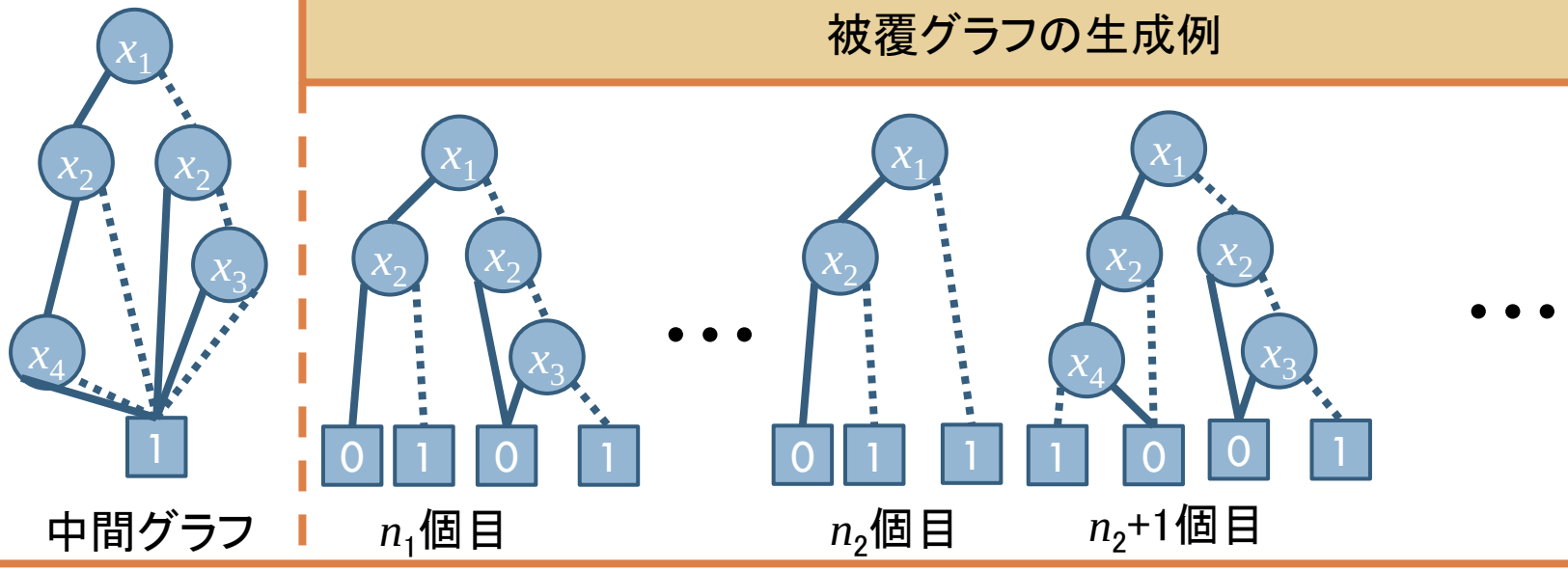


被覆BDDの生成手続き(両再帰)

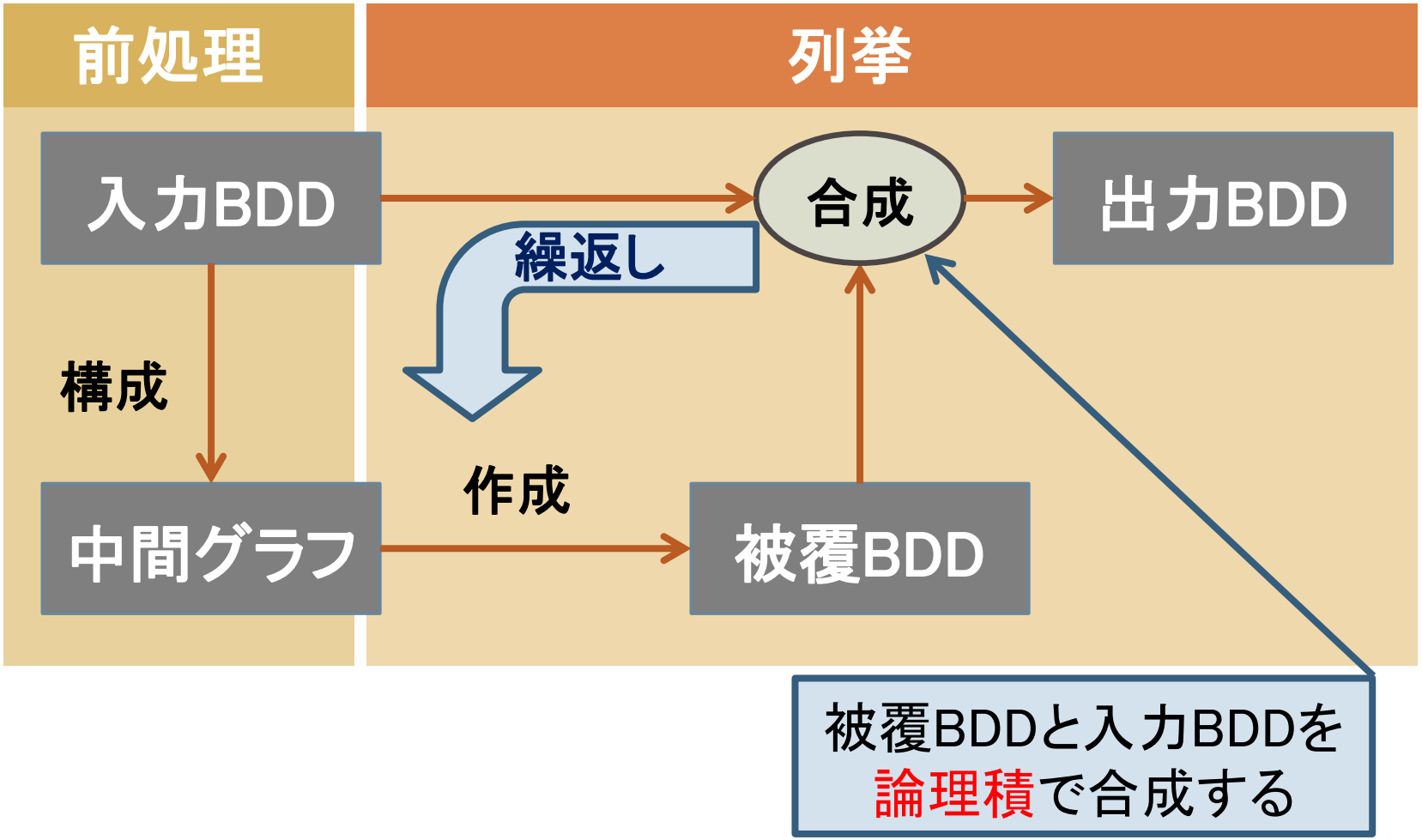
中間グラフのパスを0に繋ぎ換える



被覆グラフの生成例

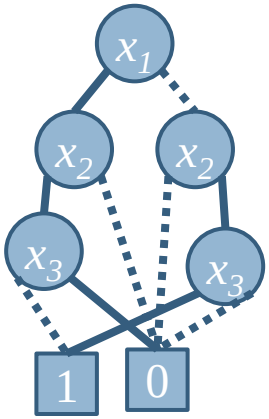


提案手法の流れ


前処理
列挙
入力BDD
合成
出力BDD
繰返し
構成
作成
中間グラフ
被覆BDD
被覆BDDと入力BDDを
論理積で合成する

全ての探索の構造

入力BDD



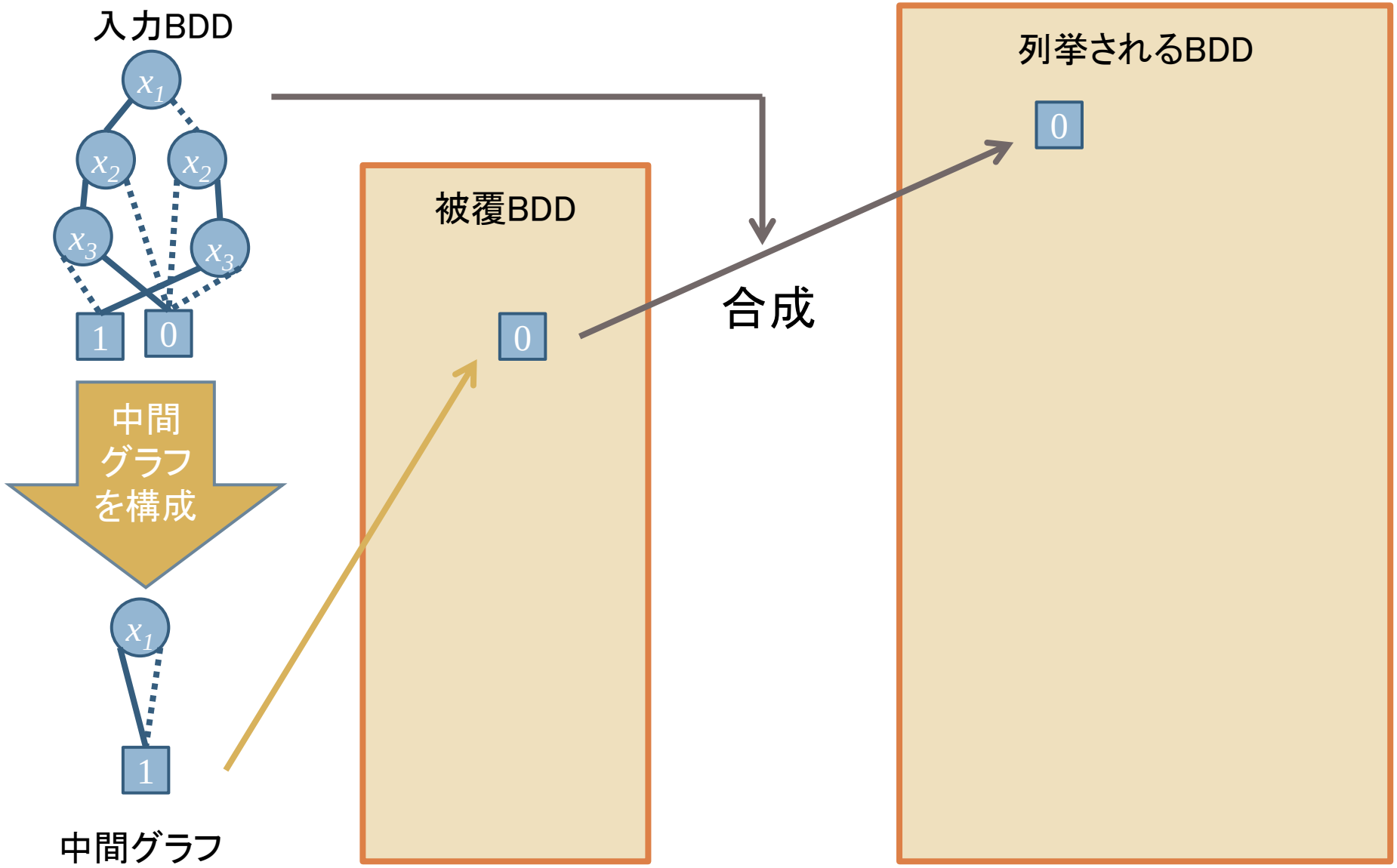
中間
グラフ
を構成

中間グラフ

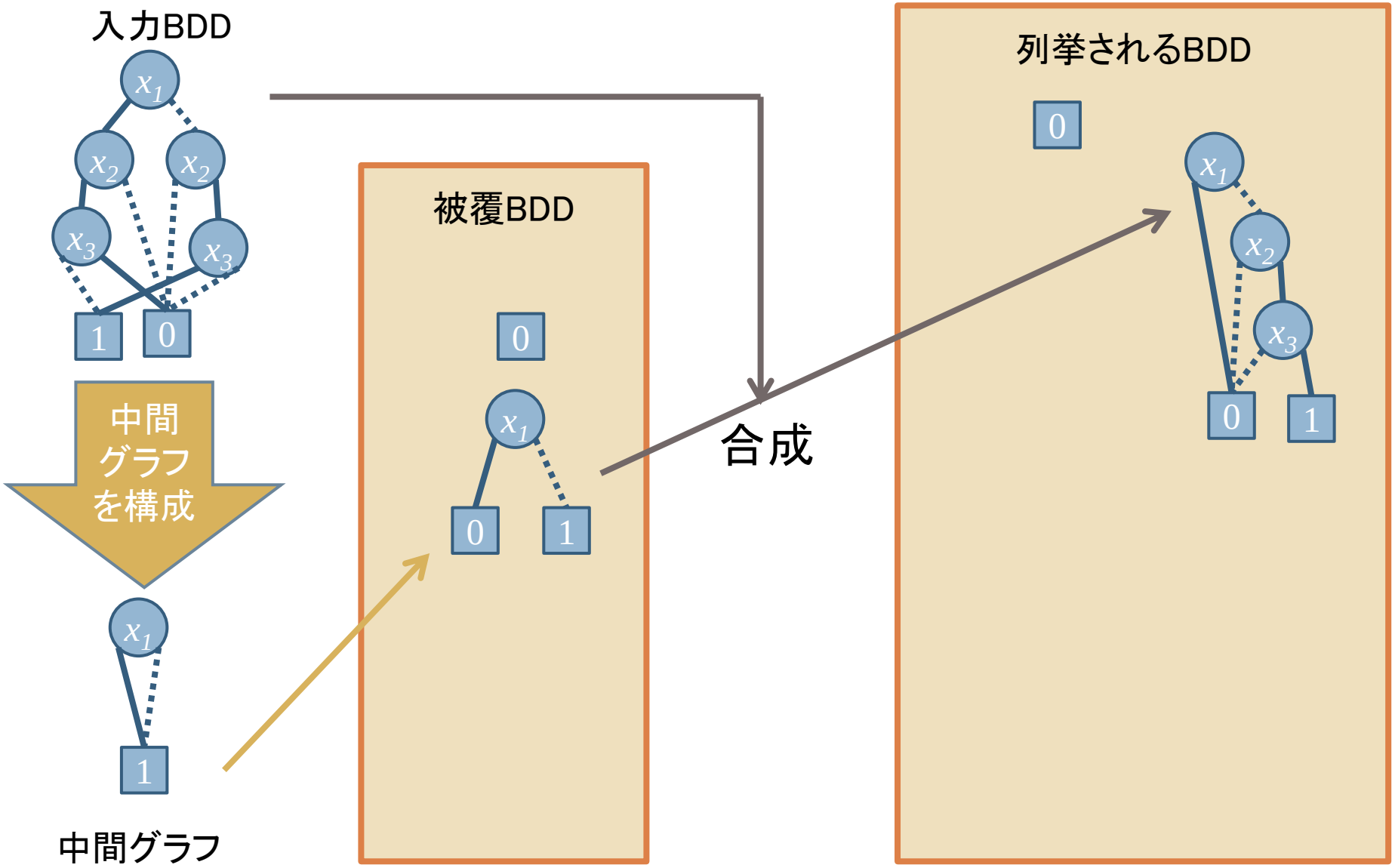
被覆BDD

列挙されるBDD

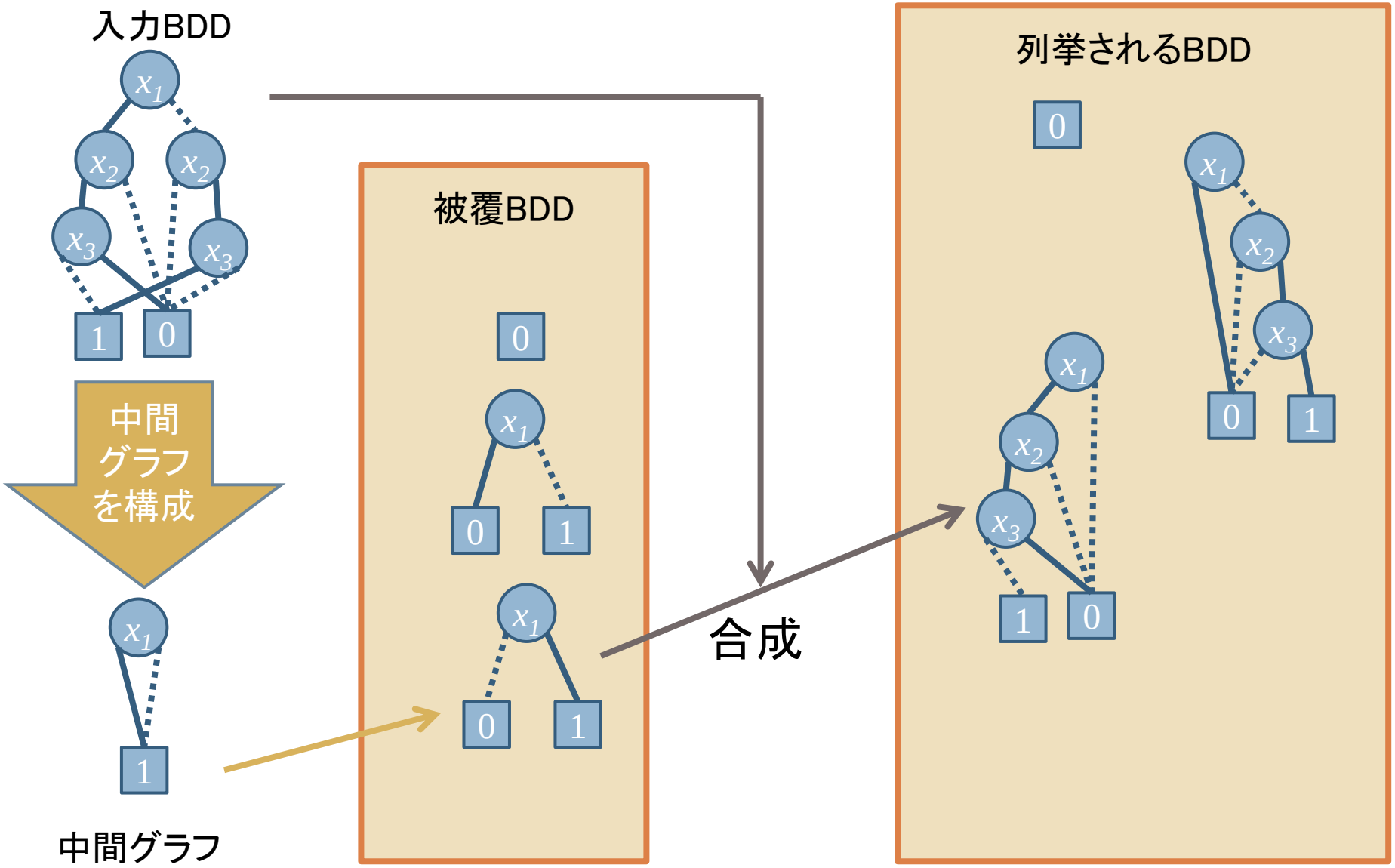
全ての探索の構造


入力BDD
x_1
x_2
x_2
x_3
x_3
1
0
中間
グラフ
を構成
x_1
1
中間グラフ
被覆BDD
0
合成
列挙されるBDD
0

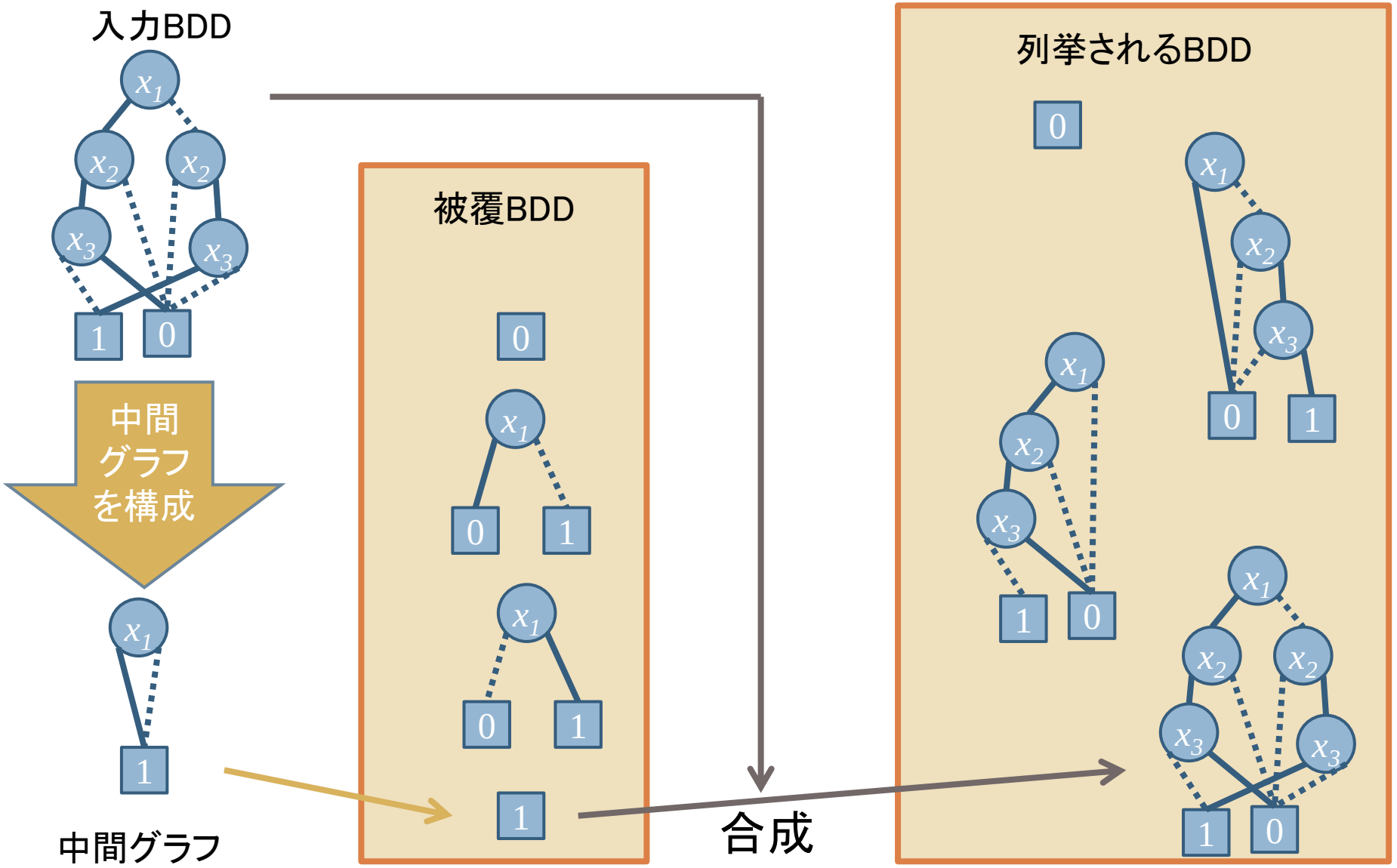
全ての探索の構造


入力BDD
x_1
x_2
x_2
x_3
x_3
1
0
中間
グラフ
を構成
x_1
1
中間グラフ
被覆BDD
0
x_1
0
1
合成
列挙されるBDD
0
x_1
x_2
x_3
0
1

全ての探索の構造


入力BDD
x_1
x_2
x_2
x_3
x_3
1
0
中間
グラフ
を構成
x_1
1
中間グラフ
被覆BDD
0
x_1
0
1
x_1
0
1
合成
列挙されるBDD
0
x_1
x_2
x_3
1
0
x_1
x_2
x_3
0
1

全ての探索の構造



遅延時間の評価

- 列挙は被覆BDDの生成とBDDの合成の二つの処理で行う

被覆BDDの生成

- 一般的な遅延時間を求めるのは困難
- 中間グラフが高さhの平衡なグラフの場合
  - 平均遅延時間は$O(h)$
  - 最大遅延時間は$O(h^2)$

BDDの合成[Bryant, 1986]

- 合成するBDDの節点数をそれぞれm_1, m_2とすると
  計算時間は$O(m_1 m_2)$

まとめと課題

まとめ

- 与えられた命題論理式の全ての前提の重複ない列挙を二分決定グラフ(BDD)を用いて行った
  - 前処理として入力のBDDから中間グラフを構成する
  - 中間グラフから被覆BDDを生成し入力のBDDとの合成を繰り返すことで列挙を行う
- 平衡な中間グラフに対する被覆BDD列挙の遅延時間を求めた
  - 平均は$O(h)$，最悪の場合$O(h^2)$

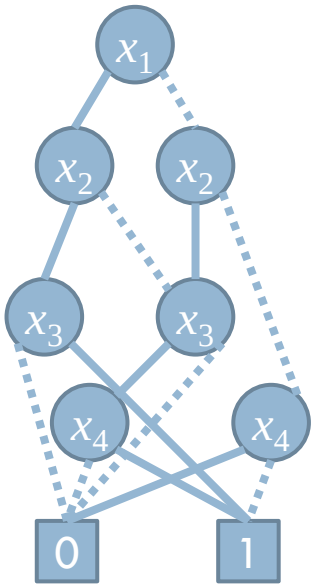
課題

- 遅延時間をより詳しく調べる
- 被覆BDDの列挙の最大遅延時間の削減
  - 中間グラフの枝の付け替え順の見直し

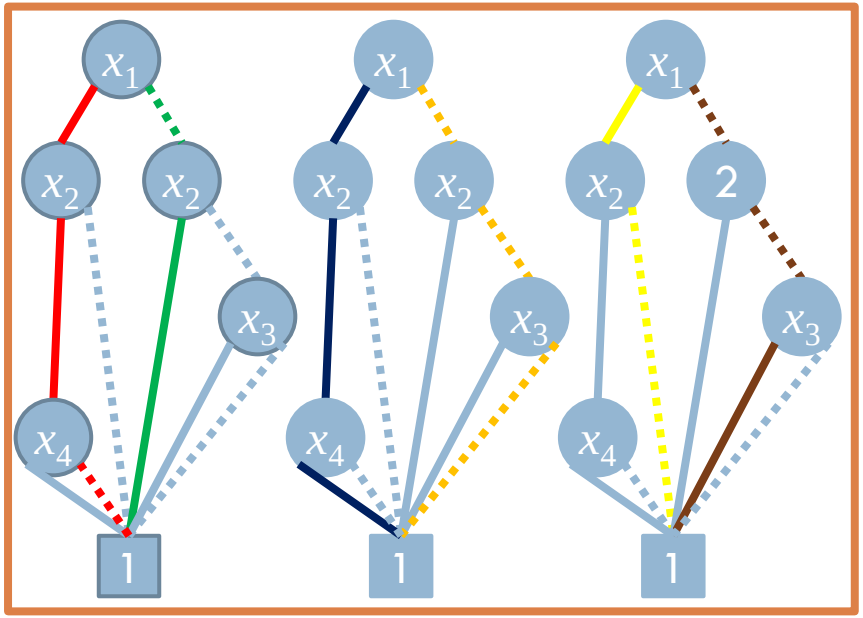
中間グラフの性質

- 中間グラフの各パスは入力論理式の真理値表において右下のような各部分に対応する
  - 表の灰色の部分はパスにおいて対応するノードがないことを表す
- 各部分にはそれぞれ1が一つだけ含まれるので，それぞれ異なるパスを繋ぎ換えた被覆グラフからできる前提は互いに異なる

BDD(F)



中間グラフ（全て同じもの）

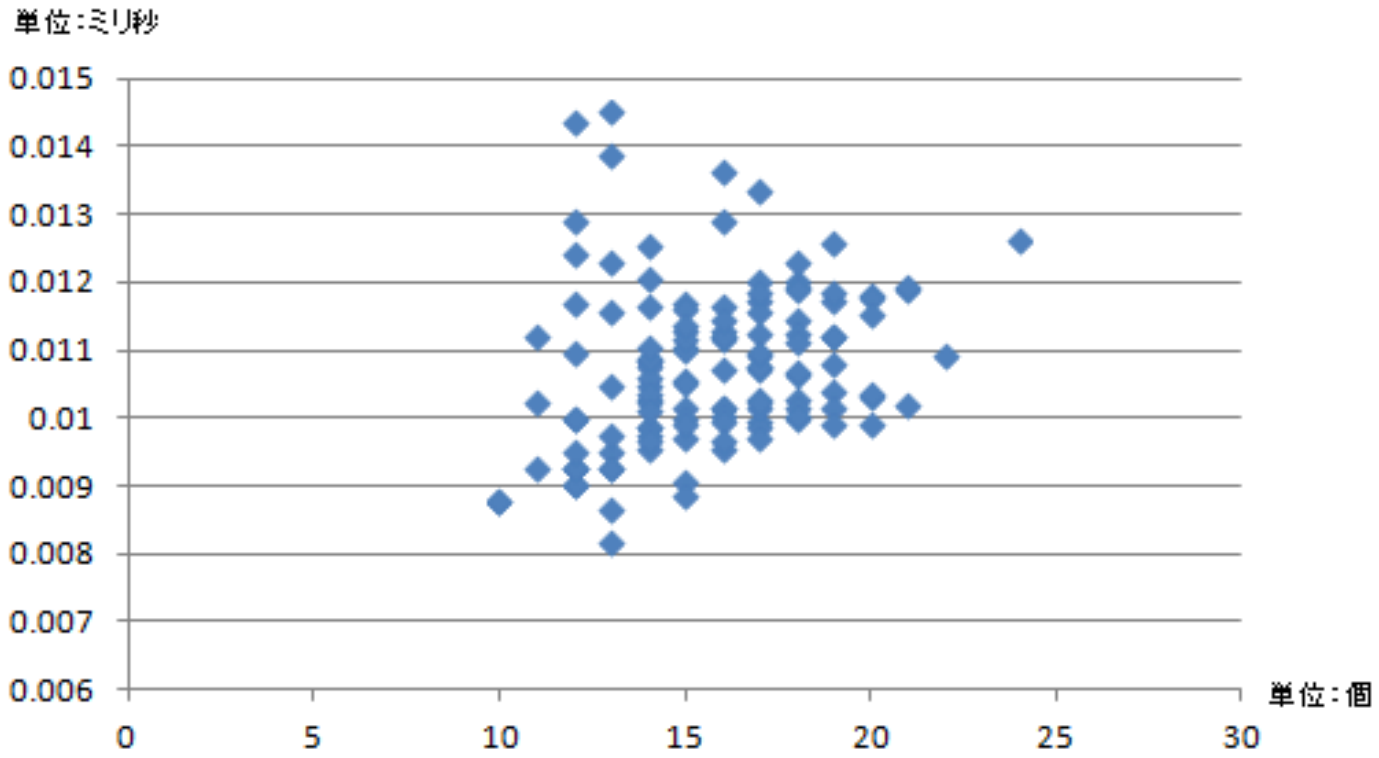


x1	x2	x3	x4	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

実験結果

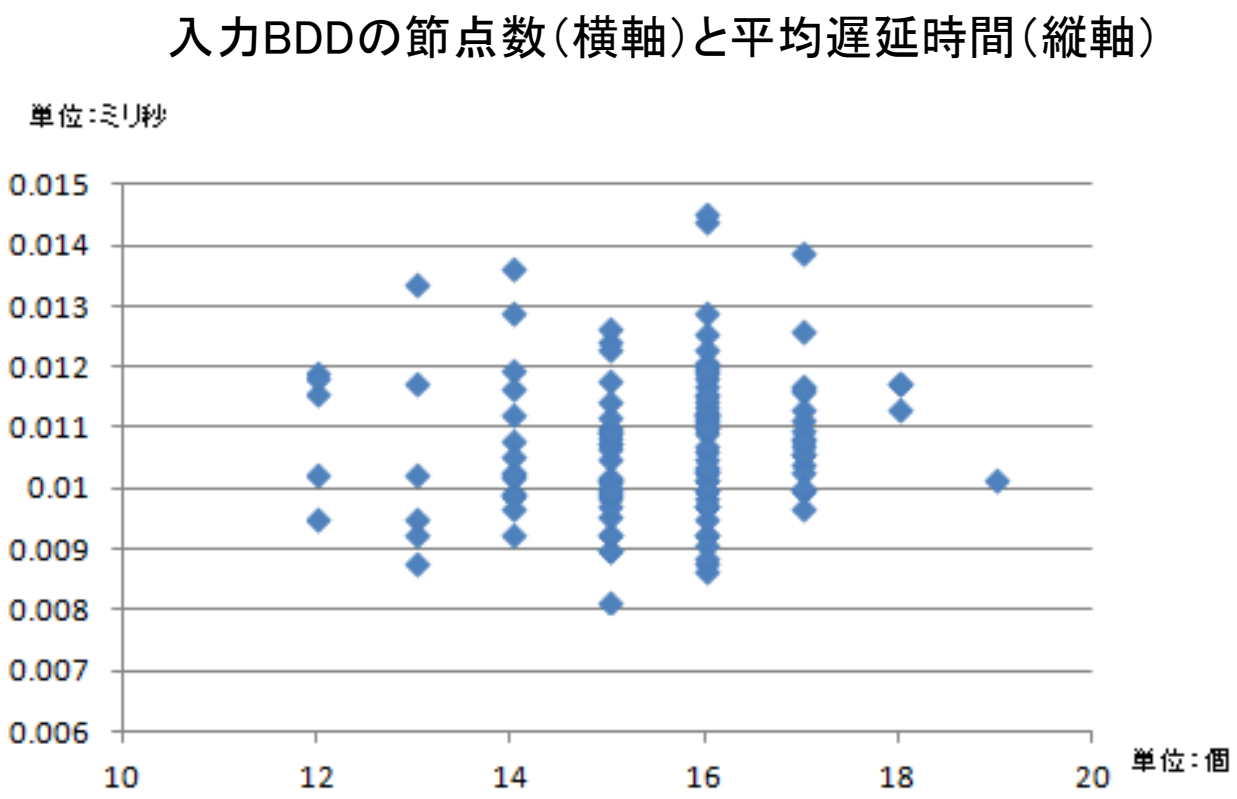
- 5 変数のランダムな二分決定グラフを132 個生成
- それらに対して全列挙を行った

入力論理式を充足する解釈の個数（横軸）と平均遅延時間（縦軸）の関係



実験結果

- 5 変数のランダムな二分決定グラフを132 個生成
- それらに対して全列挙を行った



実験結果

- 5 変数のランダムな二分決定グラフを132 個生成
- それらに対して全列挙を行った