命題論理式を充足する変数割当の網羅的探索手法について

戸田貴久
takahisa.toda@is.uec.ac.jp
電気通信大学

宋剛秀先生（神戸大）との共同研究

第103回人工知能学会 人工知能基本問題研究会
平成29年3月13日、大分県由布市

出典：*ACM Journal of Experimental Algorithmics, Vol. 21, No. 1, Article 1.12, 2016*.

目次

- AllSAT問題
- AllSATの応用
- 代表的ソルバの解説
- 性能評価
- まとめ

SATと派生問題

[番原,宋,田村,井上, '13]　[鍋島 '14]

- SATはNP完全性が証明された最初の問題
- NP完全問題の例
  - クリーク問題,点被覆問題,部分和問題,ハミルトン閉路
- 多くの応用
  - 論理合成、検証、プランニング、自動推論・定理証明
- SATを解くソフトウェア SATソルバ の開発が盛ん
- SAT派生問題

  - SMT, QBF, ASP, MaxSat, PBSat, #Sat, AllSAT, ...

All Solutions SAT Problem (AllSAT)

- AllSAT = 命題論理式の充足割当を“生成する”問題
  - もっとも広い意味では、CNFからDNFを求める問題
  - 出力に関して様々なバリエーション有り
    - 部分割当で良いか、完全割当が必要か？　部分割当は互いに素であるべきか？・・・
    - どれが真のAllSATか共通認識ないようである

リテラル(literal)

CNF (Conjunction Normal Form)

$(x2 \vee x3) \wedge (x1 \vee x2) \wedge (\neg x1 \vee x3)$

解の満たすべき制約の一覧

節(clause)

一種の表現形式の変換

項(term)

DNF (Disjunction Normal Form)

$(x1 \wedge \neg x2 \wedge x3) \vee (\neg x1 \wedge x2 \wedge \neg x3) \vee (x1 \wedge x2 \wedge x3) \vee (\neg x1 \wedge x2 \wedge x3)$

制約を満たす解の一覧　（＊）個々の項を部分割り当てとみるとき、全ての制約（節）を満たす解になる

論理における列挙問題 [Crama, Y., Hammer, P., '11]

- 論理関数の双対化
  - 論理関数 f のDNF → 双対関数 f^d の完全なDNF
  - f^dのCNFから完全なDNFを求める計算が本質
  - AllSATはDNFに完全性を求めない
- 極小ヒッティング集合の列挙 [Eiter, T., Makino, K., Gottlob, G., '08]
  - 単調論理関数の双対化と等価な問題
  - ハイパーグラフ横断の計算とも言われる

（＊）否定記号がないCNFは、節を集合とみなせば、集合族（ハイパーグラフ）とみなせる

補足：論理関数f:$\{0,1\}^n \to \{0,1\}$の双対 $f^d := \neg f(\neg x_1, \ldots, \neg x_n)$

完全なDNF＝全ての主項（prime implicants）からなるDNF

否定記号を用いないで記述可能な論理式を単調という

双対化の例

論理関数 f のDNF

$$(\neg x1 \wedge x3) \vee (x2 \wedge x3) \vee (x1 \wedge x2)$$

論理和と論理積を交換するだけ
（ここは楽）

双対関数 f^d のCNF

$$(\neg x1 \vee x3) \wedge (x2 \vee x3) \wedge (x1 \vee x2)$$

分配則を適用して展開すると
指数的に多くの項が生成される
（ここが大変！）

双対関数 f^d の完全なDNF

$$(x1 \vee \neg x2 \vee x3) \vee (x2 \wedge x3) \vee (\neg x1 \wedge x2 \wedge \neg x3)$$

単調双対化の従来研究

■ 単調と非単調
単調の場合は1つの解の発見ならば容易だが、非単調の場合は困難。
SATでいう所の矛盾のため、強力な枝刈り必要。
どちらの場合でも、これまで発見していない次の解を発見するのは容易ではない。

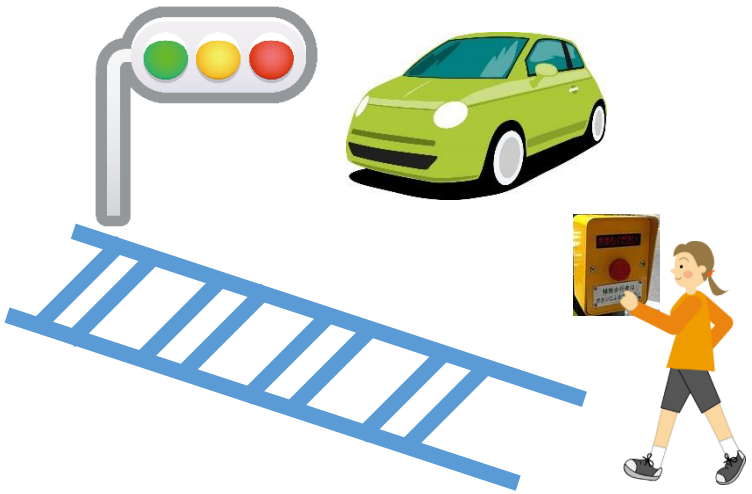
- 計算量の観点からよく研究されている
  - FredmanとKhachiyanの準多項式時間アルゴリズム
  - 多項式時間で解けるかは未解決問題
- 理論だけでなく応用なども含めてよくサーベイされている
  [Eiter, T., Gottlob, G., '02], [Eiter, T., Makino, K., Gottlob, G., '08]
- 実装がなかったため、本格的な実証研究は最近までなかった
  - [Murakami, K., Uno, T., 2014]
    逆探索に基づくアルゴリズム、効率的な極小性判定、省メモリで軽快に動作
    実証研究の基盤確立：Hypergraph Dualization Repository で主要技法の実装・ベンチマークを配布
  - [Toda, T., '13]
    BDD/ZDDに基づくアルゴリズムを提案し、実装を配布
    解をBDD/ZDDの形で保持するため、一般に大メモリ必要だが、効率的な圧縮・操作により、他では現実的な時間に計算できない問題でも解けることがある
  - [Gainer-Dewar, A., Paola V., '17]
    バイオインフォなど各種の応用、従来手法のサーベイ、包括的な評価実験
    村上・宇野の手法、戸田の手法が他の従来手法に比べ格段に優れていることを報告
    あらゆる従来手法の実装をGitHubで配布
- （SAT分野では）過制約システムの矛盾解消や矛盾の説明
  MCSとMUSの双対的な対応として単調双対が現れる

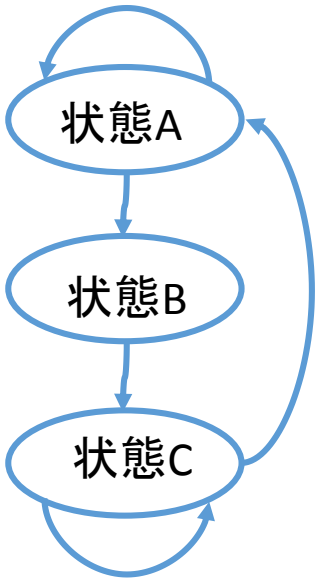
目次

- AllSAT問題
- AllSATの応用
- 代表的ソルバの解説
- 性能評価
- まとめ

モデル検査

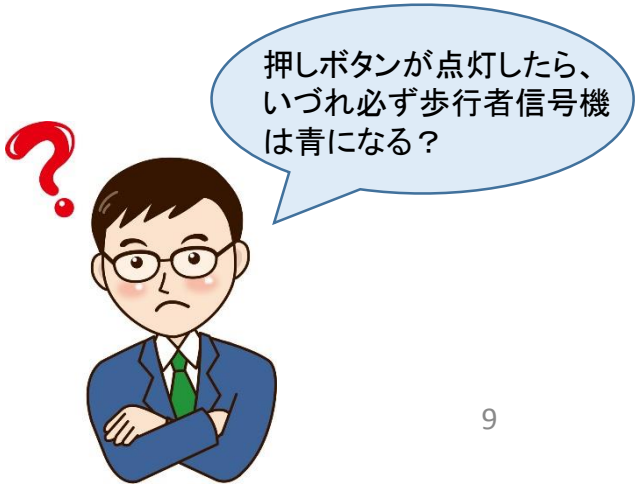
- システムが望ましい性質を持っているか？
- システムは状態遷移系、性質は時相論理で記述
- HW/SW産業への幅広い応用により、Clarke博士らは2007年Turing賞



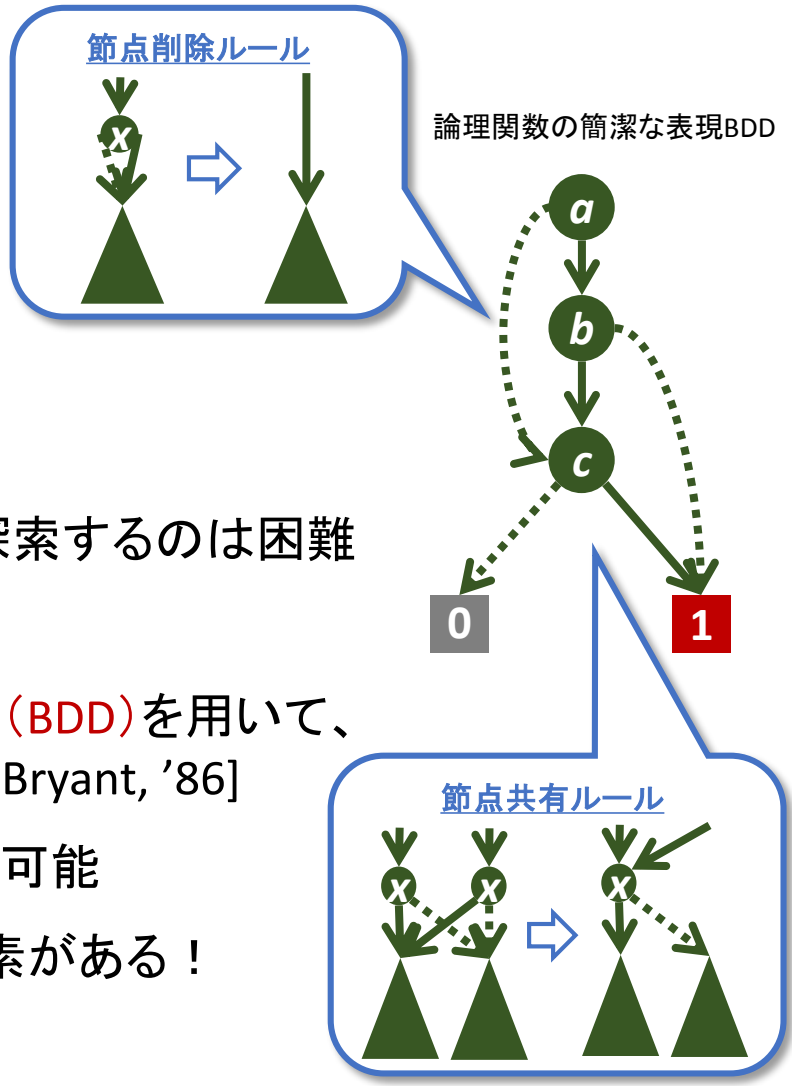


時間に関する様相を扱える

Xφ　次の時刻φが真
Gφ　今後ずっとφが真
Fφ　いつかφが真
φFψ ψが真になるまでずっとφが真

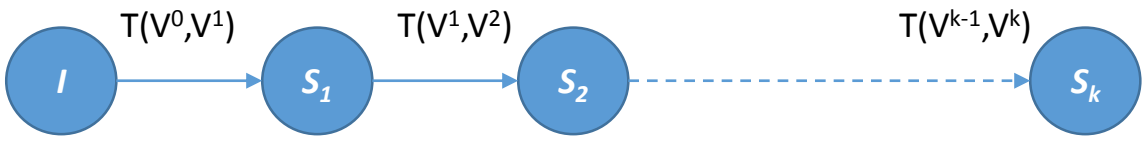


手法の移り変わり



- 素朴なグラフ探索
  - グラフが巨大すぎて、明示的に表現し探索するのは困難
- 記号モデル検査
  - 論理関数のデータ表現 二分決定グラフ（BDD）を用いて、状態集合を表現・操作する手法が導入[Bryant, ’86]
  - 状態要素が数百程度のHW設計に対応可能
  - しかし、現実の問題は何千もの状態要素がある！
- SATに基づく有界モデル検査
  - 問題をCNFに符号化し、SATソルバを用いて解く[Biere et al. ‘99]
  - スケーラビリティからBDDに置き換わり、現在でも主流の方法となる

有界モデル検査の考え方



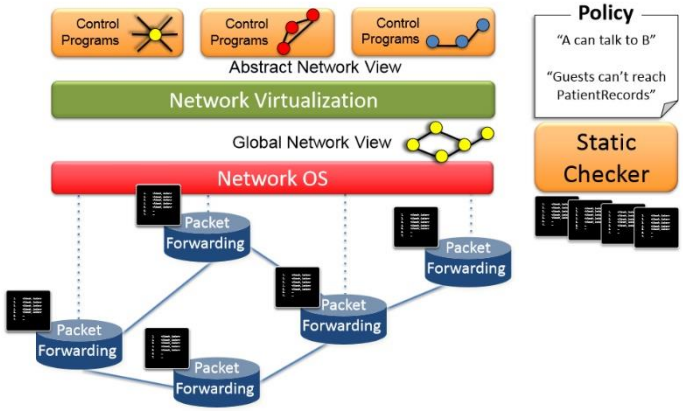
- 符号化の方法
  - 初期状態の集合Iからkステップで到達可能な状態集合S_k
    - $I(V^0) \wedge T(V^0,V^1) \wedge T(V^1,V^2) \wedge \ldots \wedge T(V^{k-1},V^k)$
  - 性質Pに違反する状態集合
    - $\neg P(V^k)$
- 性質に反する（反例がある）⇔ CNFが充足可能
  - Kステップまでしか検査しないので完全な検証ではない
  - しかし、バグ発見には有用

[山崎, 八鍬, 登内, '16]

ネットワークにおける信頼性

Software Defined Network (SDN)

- Software Defined Network（SDN）
  - ソフトウェアによってネットワークを定義
  - 柔軟な制御を実現
- 従来NWとの違い
  - データプレーンとコントロールプレーンの分離
  - NW上の少数のサーバにコントロールプレーンが置かれ、ルータにデータプレーンだけが実装；両者は標準化プロトコルで通信
- 信頼性
  - 制御プログラムのバグや設定ミスなどのリスク、NW全体に波及
  - ベンダーによる品質保証だけでは不十分
  - NWの大規模化・複雑化により、人手を要する実機テストは現実的でない



出典：N. McKeown, Mind the Gap. SIGCOMM Keynote, 2014.

プロトコルが標準化されたことから形式手法の適用容易に

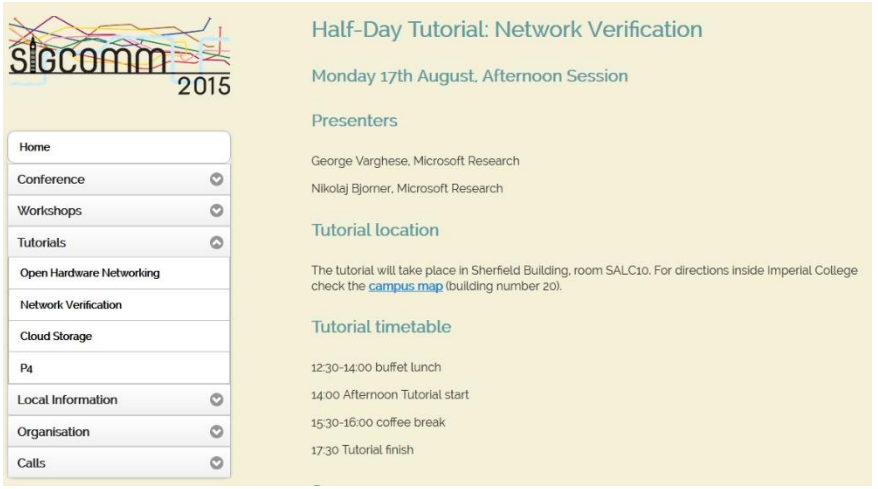
ネットワークの検証

[Qadir, J., Hasan, O., ’15]

[Zhang, S., Malik, S., McGeer, R., ’12]

- 代表的な検証項目
  - Reachability　ホストAからBにパケットは到達できるか？
  - Forwarding Loop　同一パケットが同じ場所に戻ることがあるか？
  - Packet Destination Control　ドロップされるか？～を通過するか？
  - Slice Isolation　２つのプライベートNWが分離されているか？
- 様々な手法
  - HSA　ternary simulationに似た手法
  - Anteater　CNF符号化でSATソルバーを用いる手法

AllSATの重要性



ネットワーク分野のトップ会議SIGCOMM（2015年）

- Microsoft Researchのグループが SDNにおけるAllSATの重要性を主張 [Lopes, N., et al., '13] [Lopes, N., et al., '15]

① Incremental Computation

- 動的設定で設定変更の差分だけの検証有効
- 到達可能なパケット集合を求めることで活用できる

② Intelligibility

Reasoning

- 1つの反例（違反したパケット）から不具合の原因特定するのは困難
- サブネットやポート範囲に対応するパケット集合があれば解析容易

これまでのAllSAT研究

- 今後、重要性がますます広がりそう
  - データマイニングへの応用
  - ネットワーク検証への応用
- しかし、ソルバ自体の研究はあまりない

課題

- 応用の場で最新技法が使われていない
  - いくつかの手法が提案されているが未整理
  - 昔からある方法がデファクトスタンダード
- 包括的評価なし
  - 少数のインスタンスに関する評価では本来の性能は見えてこない
  - どのアプローチがどういうインスタンスに効果的か不明
  - 種類の異なるソルバ同士大域的に比較されていない
- Publicなソルバがほぼ皆無
  - 新たな手法を考えても容易に他と比較できない
  - 実装の善し悪しで性能左右

これまでの私の研究

① AllSAT研究のサーベイ
（新規手法の提案も含む）

② 主要技法の実装

③ 網羅的な評価実験

④ コードや評価結果の一般公開

All Solutions SAT Repository
http://www.sd.is.uec.ac.jp/toda/code/allsat.html

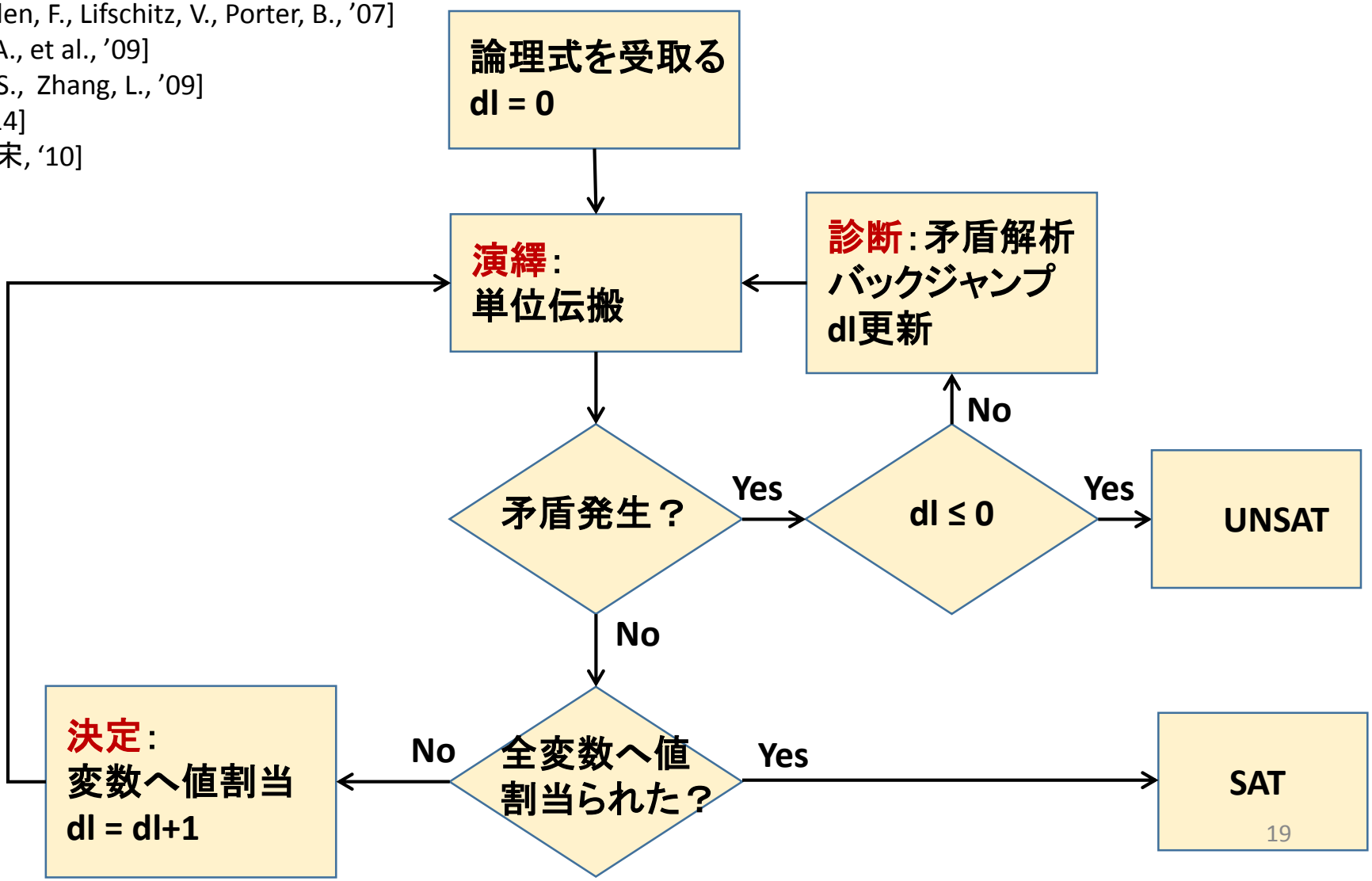
目次

- AllSAT問題
- AllSATの応用
- 代表的ソルバの解説
- 性能評価
- まとめ

基本原理

CDCL（SATを解くための現代的な枠組み）を基礎にしてAllSATソルバは作られる

[Harmelen, F., Lifschitz, V., Porter, B., ’07]
[Biere, A., et al., ’09]
[Malik, S., Zhang, L., ’09]
[鍋島 ‘14]
[鍋島，宋, ‘10]

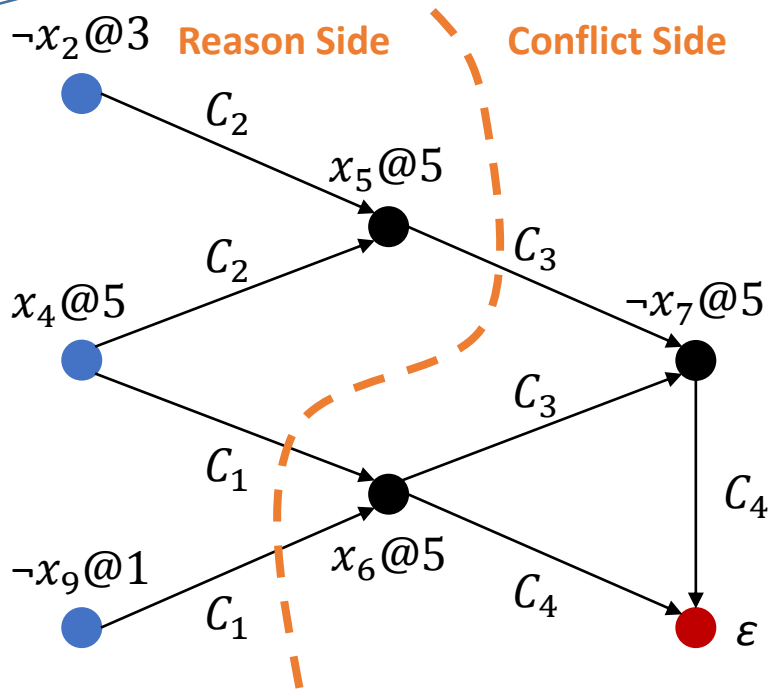


矛盾からの節学習

$C_1: \neg x_4 \vee x_9 \vee x_6$
$C_2: \neg x_4 \vee x_2 \vee x_5$
$C_3: \neg x_5 \vee \neg x_6 \vee \neg x_7$
$C_4: \neg x_6 \vee x_7$

- 含意グラフによる矛盾解析
  - 矛盾の“原因”を解析
    - $x_5 \wedge x_4 \wedge \neg x_9$ が矛盾導出
  - バックラックレベルの計算
    - $\neg x_9$の決定レベル1に戻る
- 学習節の記憶
  - 同様の矛盾に陥るのを回避
    - 原因の否定をCNFに追加
    - $\neg(x_5 \wedge x_4 \wedge \neg x_9) = \neg x_5 \vee \neg x_4 \vee x_9$
    - 原因の部分割当てを避けながら探索
    - 将来起こりうる矛盾を回避可能

非時間順バックトラックあるいはバックジャンプと呼ばれる



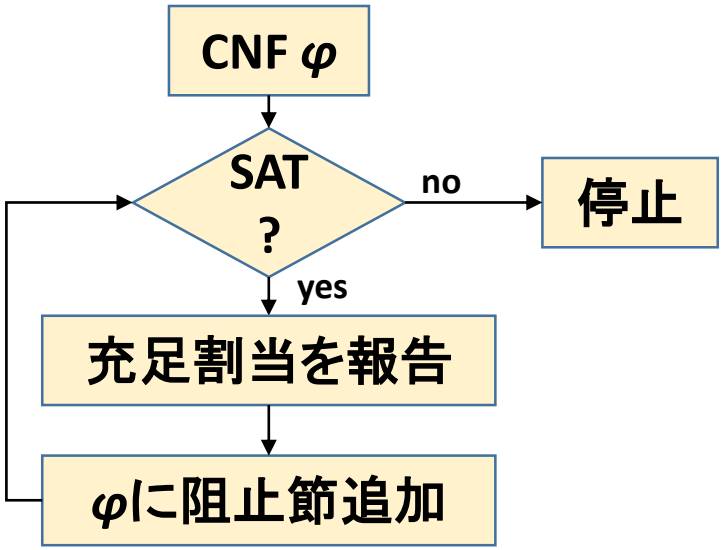
含意グラフ：変数割当と含意関係を表現

図の例は Vizel et al., Boolean Satisfiability Solvers and Their Applications in Model checking, Proceedings of The IEEE, 2015 からとった。

代表的なAllSATソルバの型

1. ブロッキング型
2. 非ブロッキング型
3. BDD型
   - BDD＋ブロッキング型
   - BDD＋非ブロッキング型

ブロッキング型ソルバ [McMillan, K., '02]



- 阻止節（blocking clause）を記憶して、解の再発見を回避
  - 充足割当 x1 ∧ ¬x2 ∧ x3 を発見
  - 阻止節 ¬ (x1 ∧ ¬x2 ∧ x3) = ¬x1 ∨ x2 ∨ ¬x3 を計算
  - これをCNFに追加すると、今後、x1 ∧ ¬x2 ∧ x3 は解にならない！
- AllSAT実装の代名詞
- SATソルバをほぼそのまま利用可能
  ①実装が楽　②最新ソルバのパワーをそのまま継承

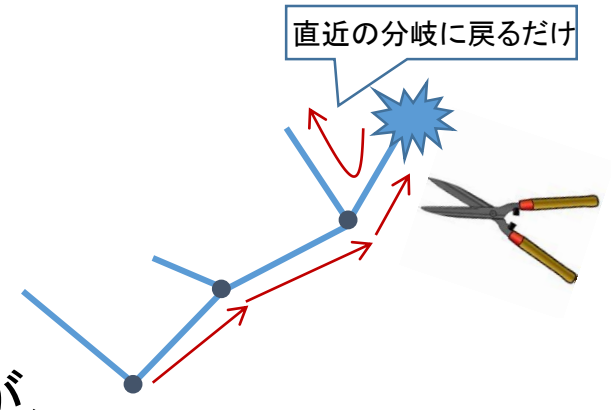
CNFサイズ増大により、単位伝搬の性能が著しく低下
（＊）CDCLの大部分の処理は単位伝播が占める！

非ブロッキング型ソルバ

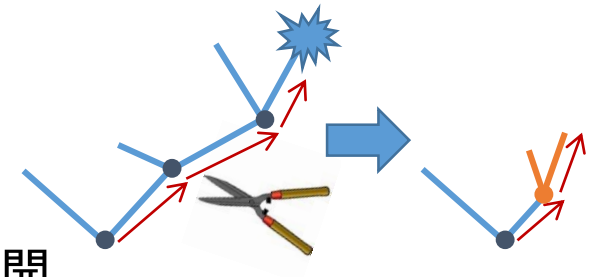
[Grumberg, O., Schuster, A., Yadgar, A., ’04]

- 阻止節を使わずに、
  探索アルゴリズムによって解の再発見を回避
- 単位伝播の性能劣化なし！

バックトラックにおける時間順と非時間順



- 時間順
  - 探索の決定木は、矛盾に至る枝は刈るが、他のこれまでの分岐は取り消さない
  - 浅いレベルに矛盾の原因があるとき、そこまで戻るのが困難
- 非時間順
  - 決定木は動的に変わる
    - 矛盾に至るとき、複数の分岐を取り消して再開



  - 学習節により、刈られた部分は複製されない（同じ失敗しない）
  - SATの場合これで十分。しかし、AllSATの場合、安易な非時間順バックトラックは同じ解の再発見、無限ループに陥る！

非ブロッキング型の基本戦略

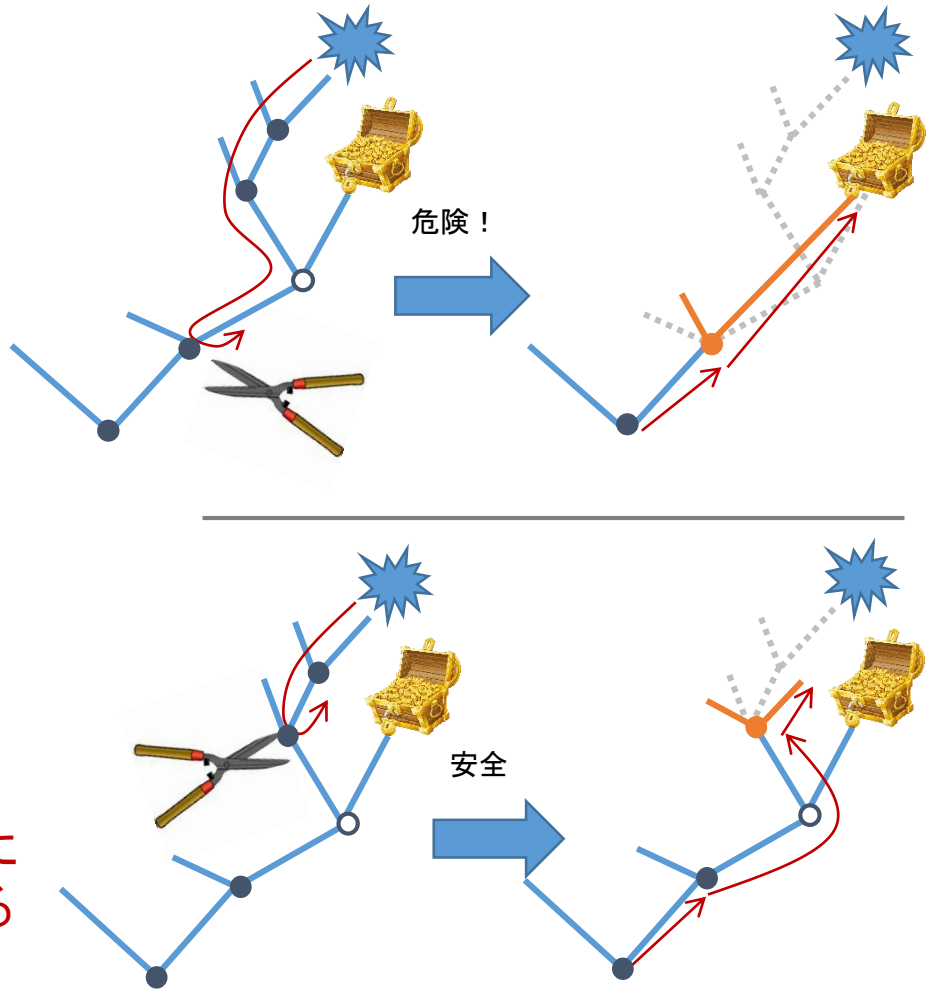
- 解が発見されたとき
  - 時間順バックトラック
- 矛盾が生じたとき
  - 解の再発見がされないバックトラック
    - **BT** 時間順バックトラック
    - **BJ** レベル制限付きバックジャンプ
    - **CBJ** Conflict-directed Backjump
    - **BJ+CBJ** BJとCBJの組合せ

非ブロッキング型の特徴
① 単位伝搬の性能が低下しない
② 解の再発見を探索アルゴリズムで回避しつつも、可能な限りSATのパワーを取り込む

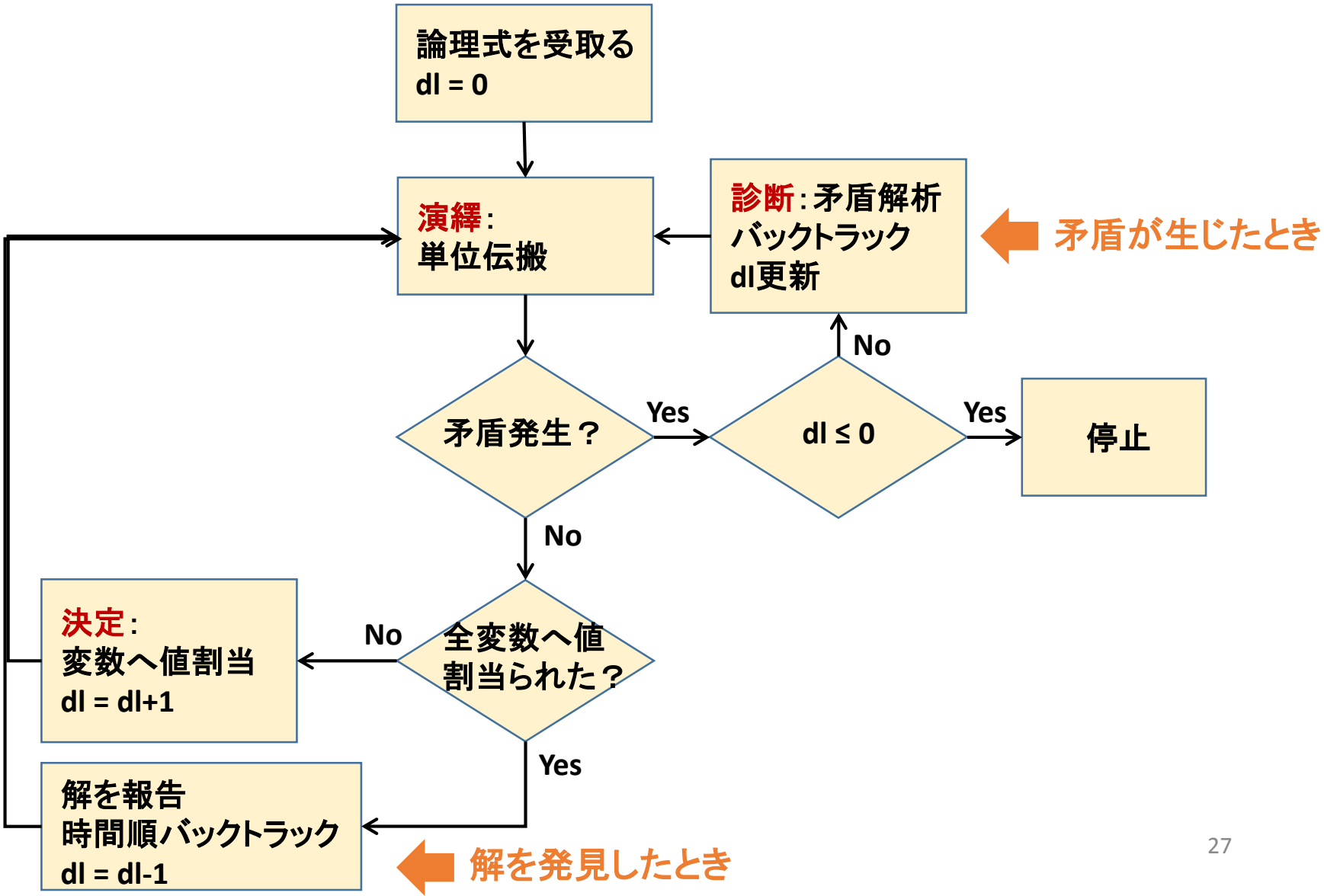
レベル制限付きバックジャンプ

[Gebser, M., et al., ’07]

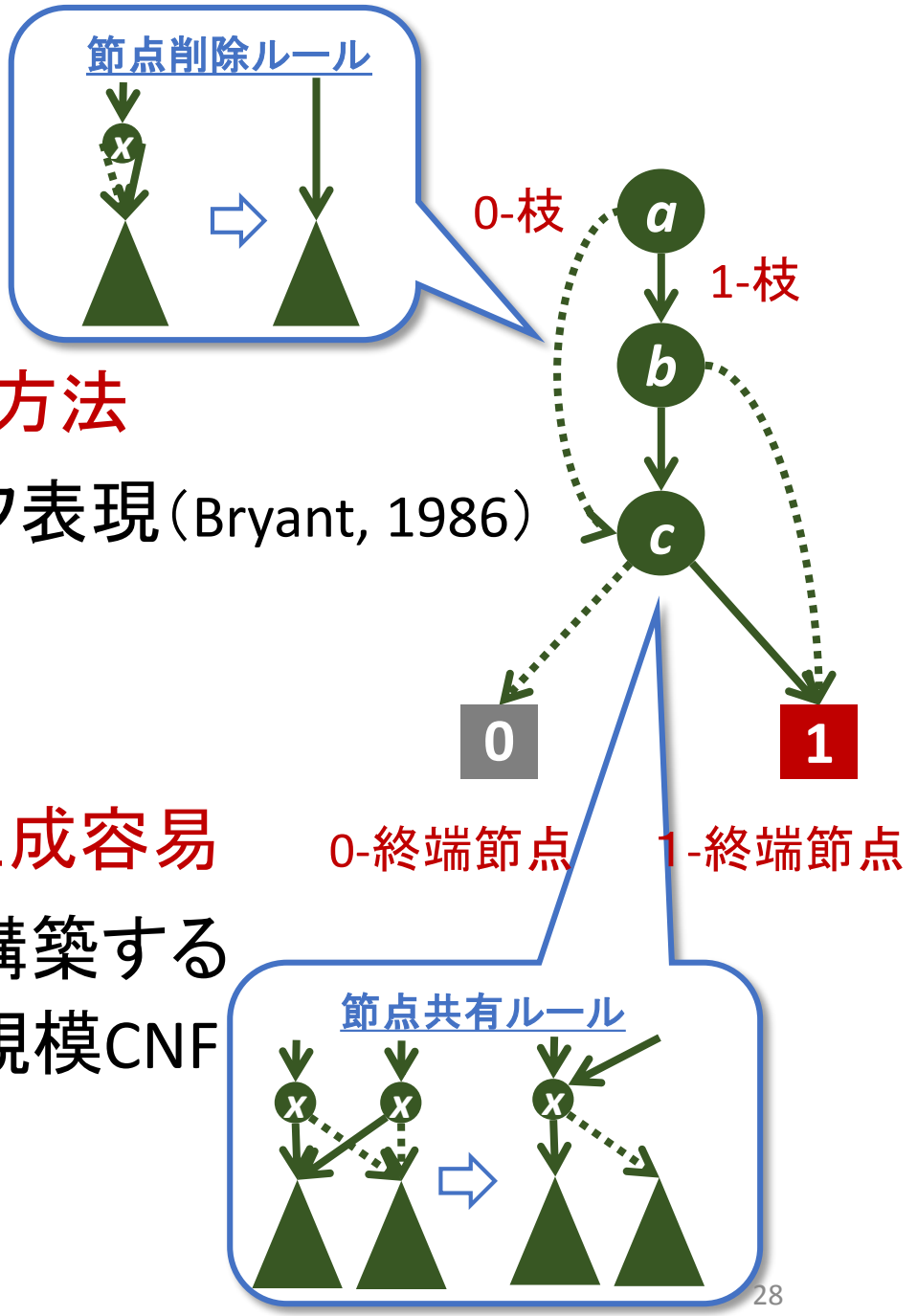
- なぜ解を再発見してしまうか？
  - バックトラックレベルが浅すぎるから
- 安全なバックトラックレベルlim
  - これまでに発見した解のうち、
    現在の割当と同一の決定割当を
    与えたレベルの最大値
- 基本戦略
  - 矛盾解析で計算したレベル ≥ lim のとき
    - 通常のバックジャンプ
  - そうでないとき
    - 時間順バックトラック
- 解の見つかっていない空間ではしきりに
  バックジャンプして解を見つけようとする



非ブロッキング型の処理の流れ



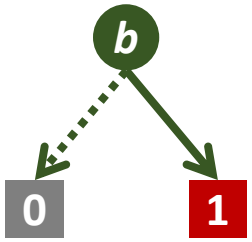
BDD型ソルバ



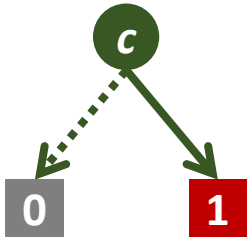
- CNFからBDDを構築する方法
- BDDは論理関数のグラフ表現（Bryant, 1986）
  - 簡潔さ
  - 各種の論理演算、クエリ
- BDDが作れたら、解の生成容易
- 論理演算を用いてBDD構築する典型的アプローチは大規模CNFには不向き

論理演算によるBDD構築の例

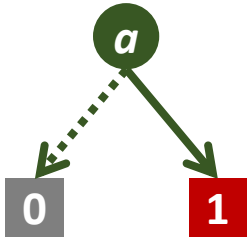
変数b



変数c



変数a

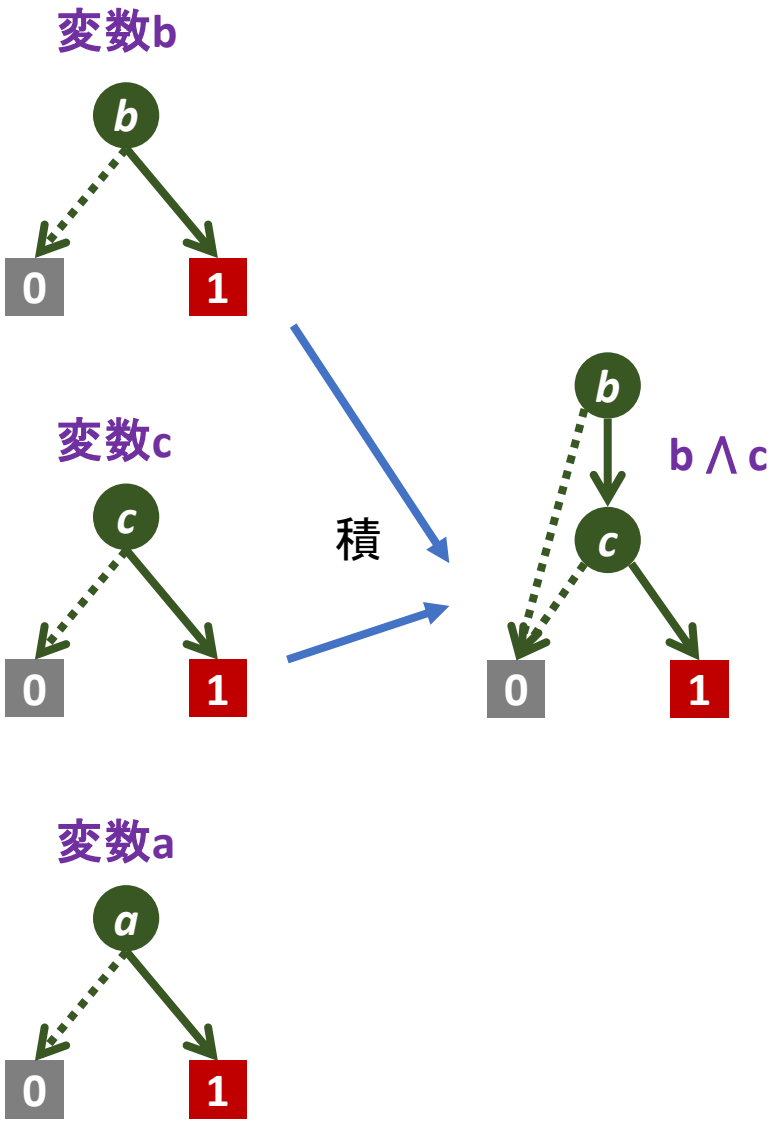


目標

$\neg a \wedge c \vee a \wedge \neg b \vee a \wedge b \wedge c$

?

論理演算によるBDD構築の例

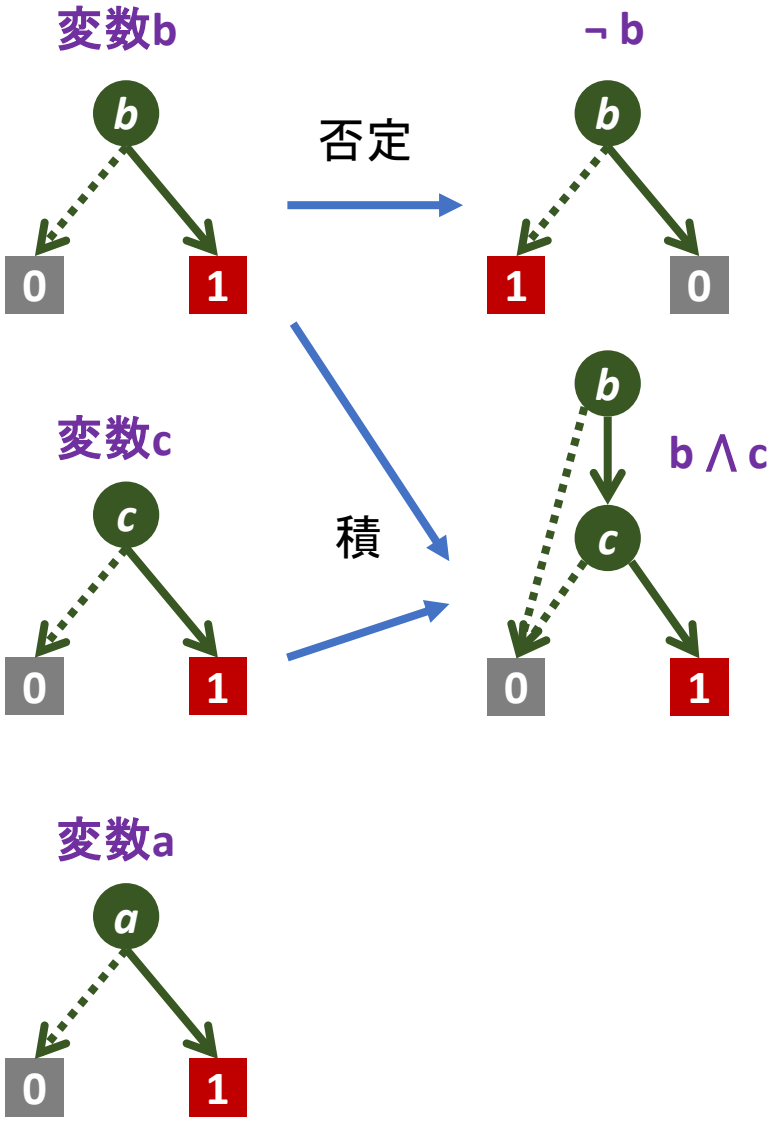


目標

¬ a ∧ c ∨ a ∧ ¬ b ∨ a ∧ b ∧ c

?

論理演算によるBDD構築の例

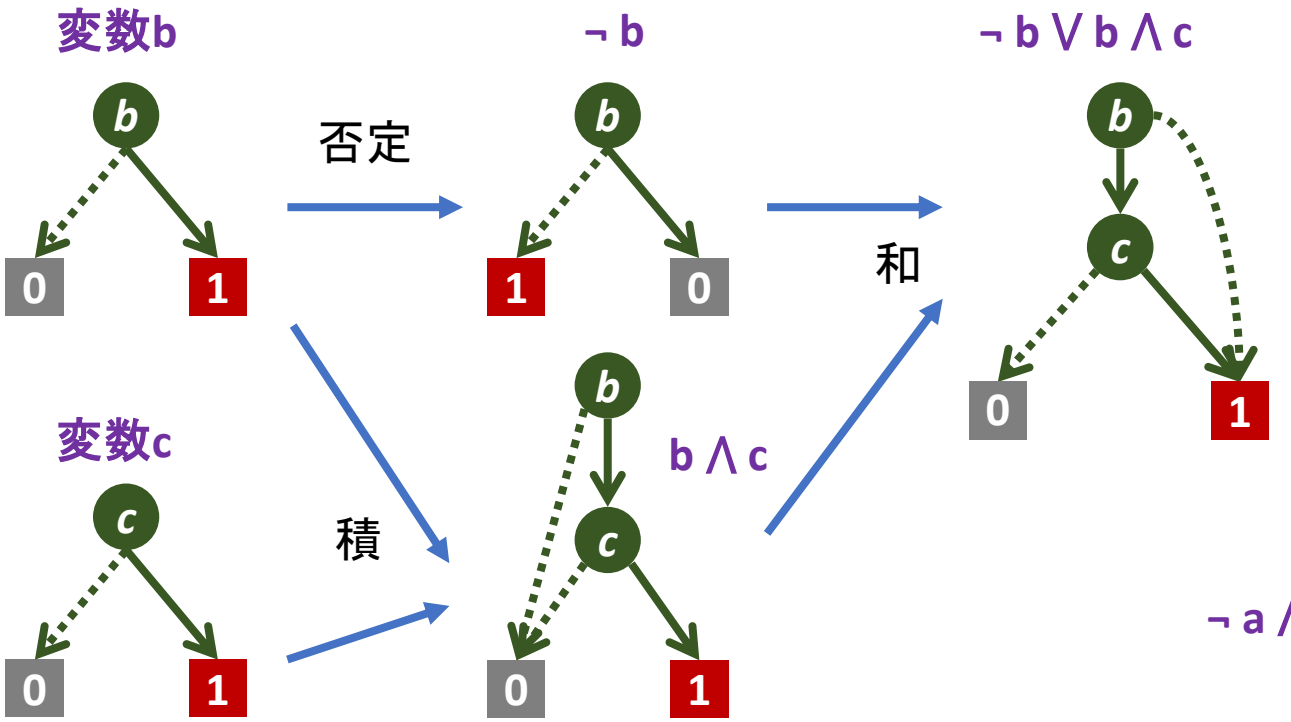


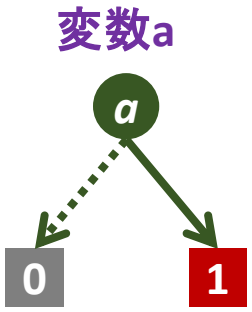
目標

¬ a ∧ c ∨ a ∧ ¬ b ∨ a ∧ b ∧ c

?

論理演算によるBDD構築の例



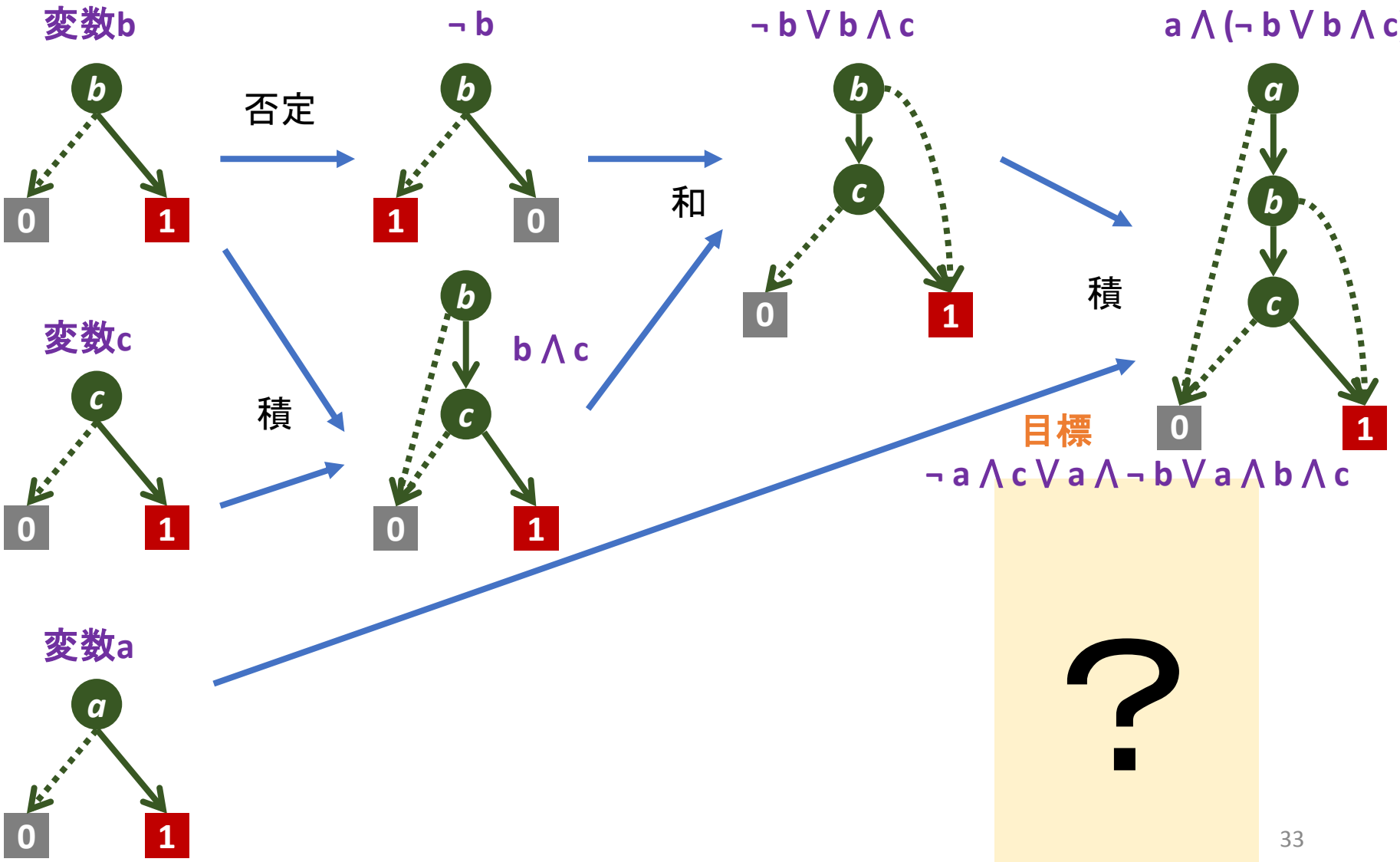


目標

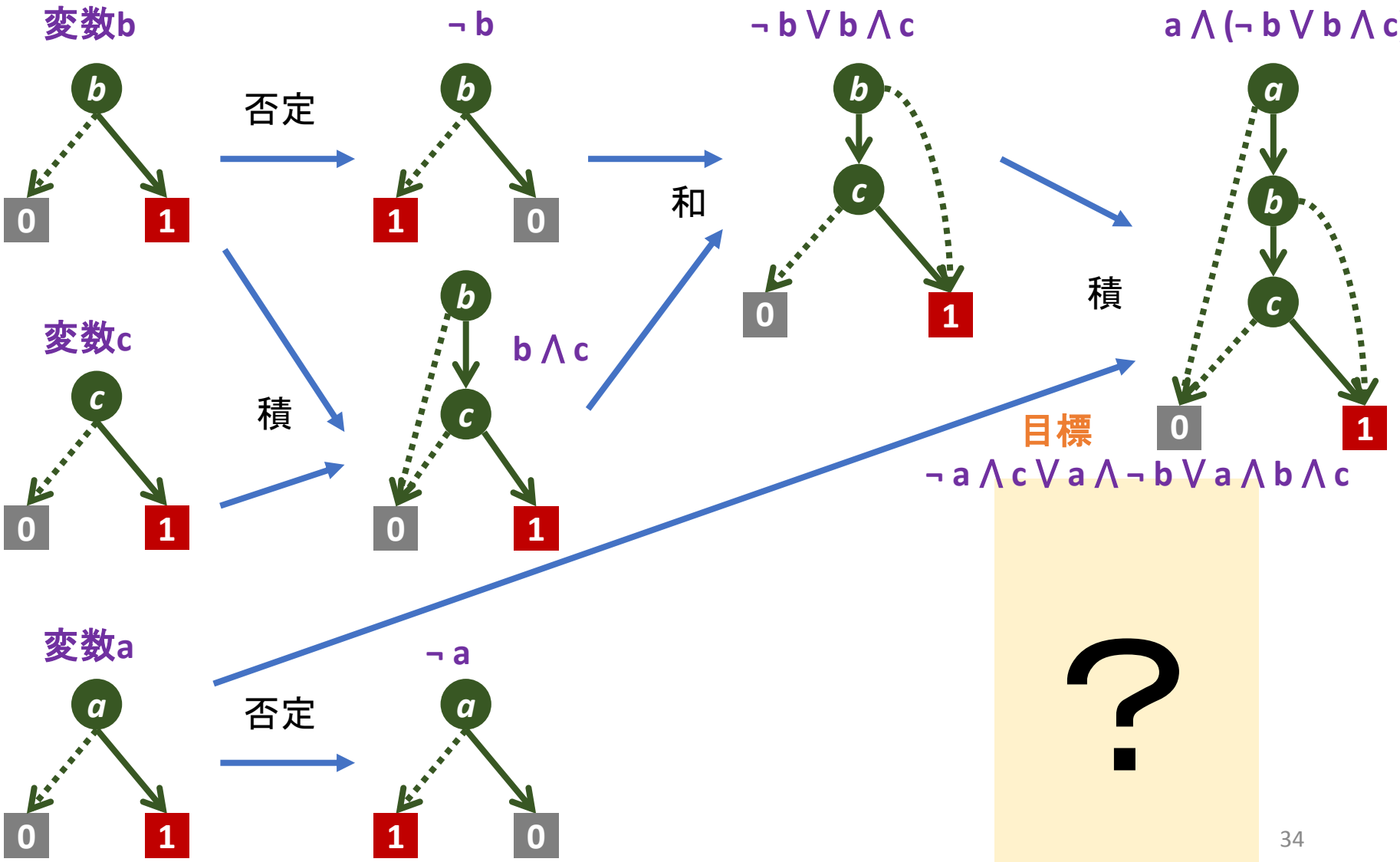
¬ a ∧ c ∨ a ∧ ¬ b ∨ a ∧ b ∧ c

?

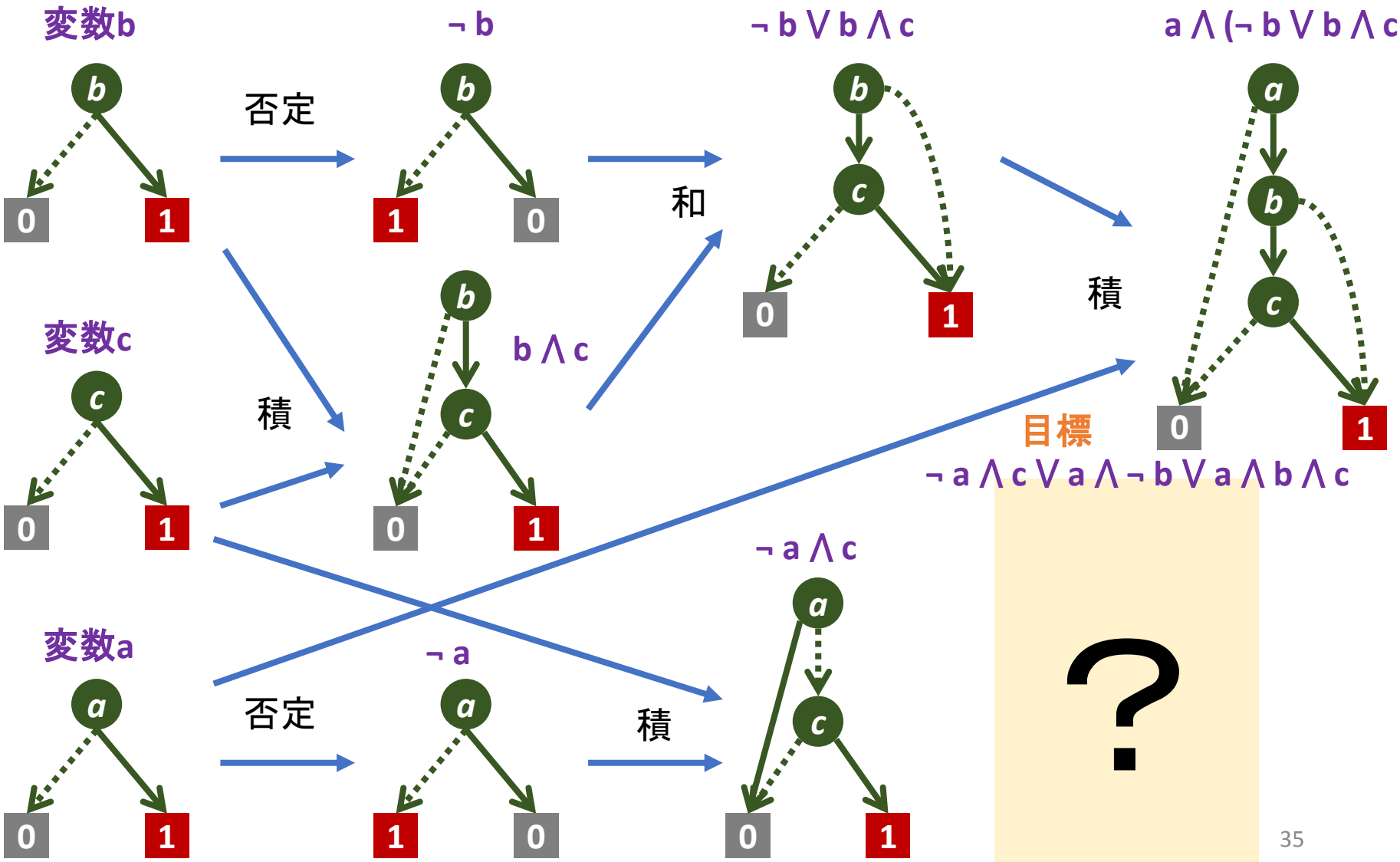
論理演算によるBDD構築の例



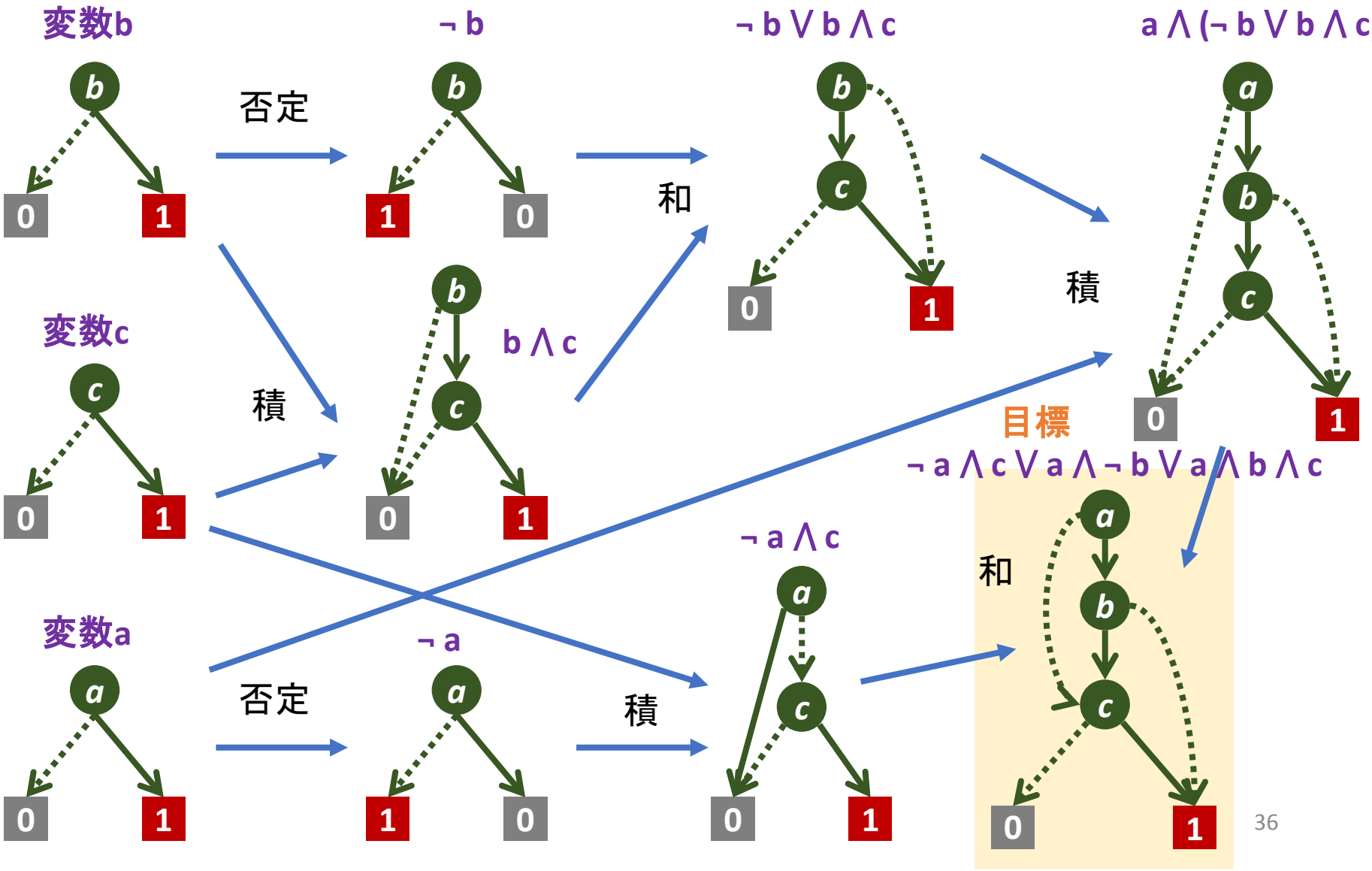
論理演算によるBDD構築の例



論理演算によるBDD構築の例

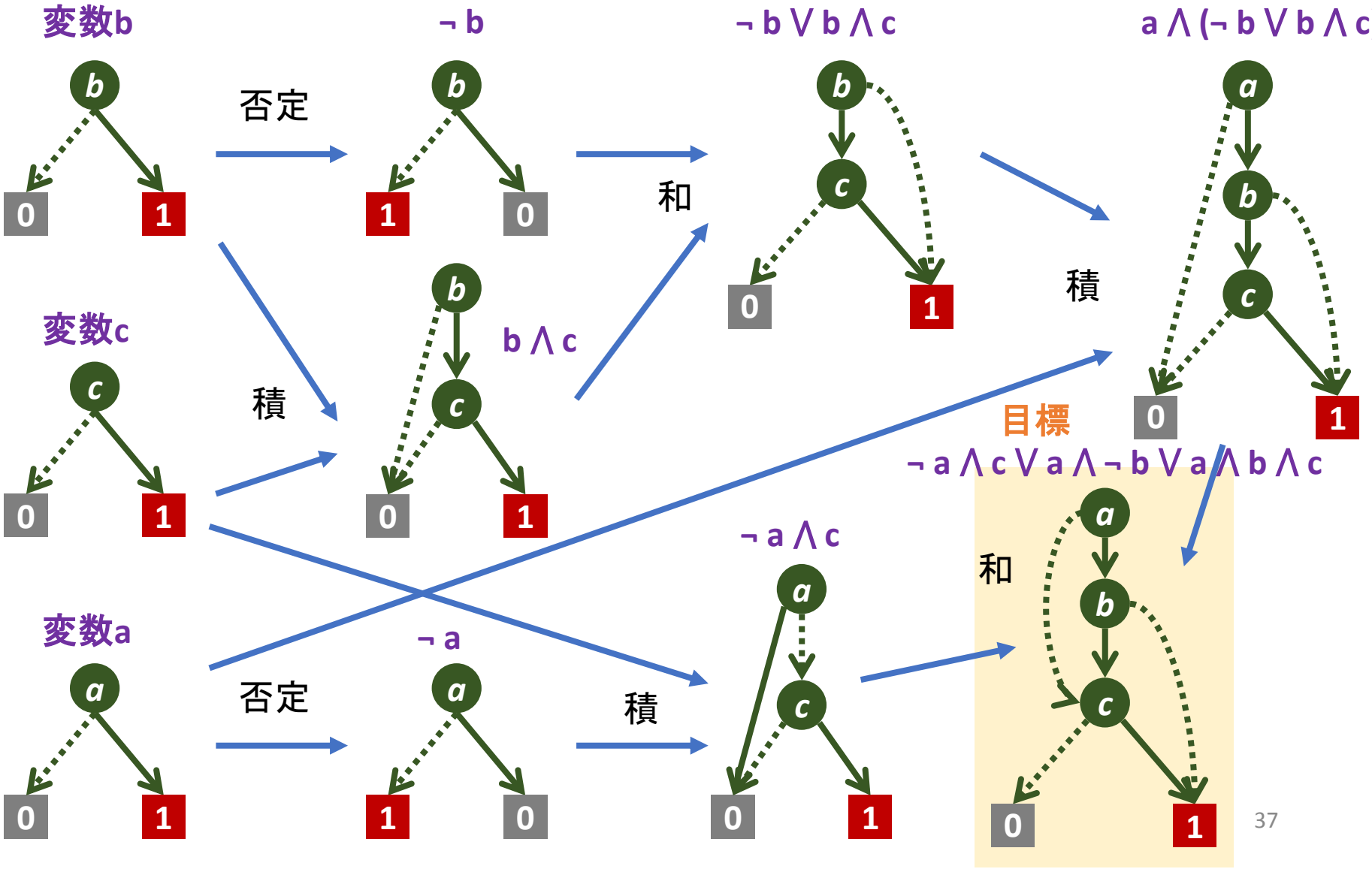


論理演算によるBDD構築の例

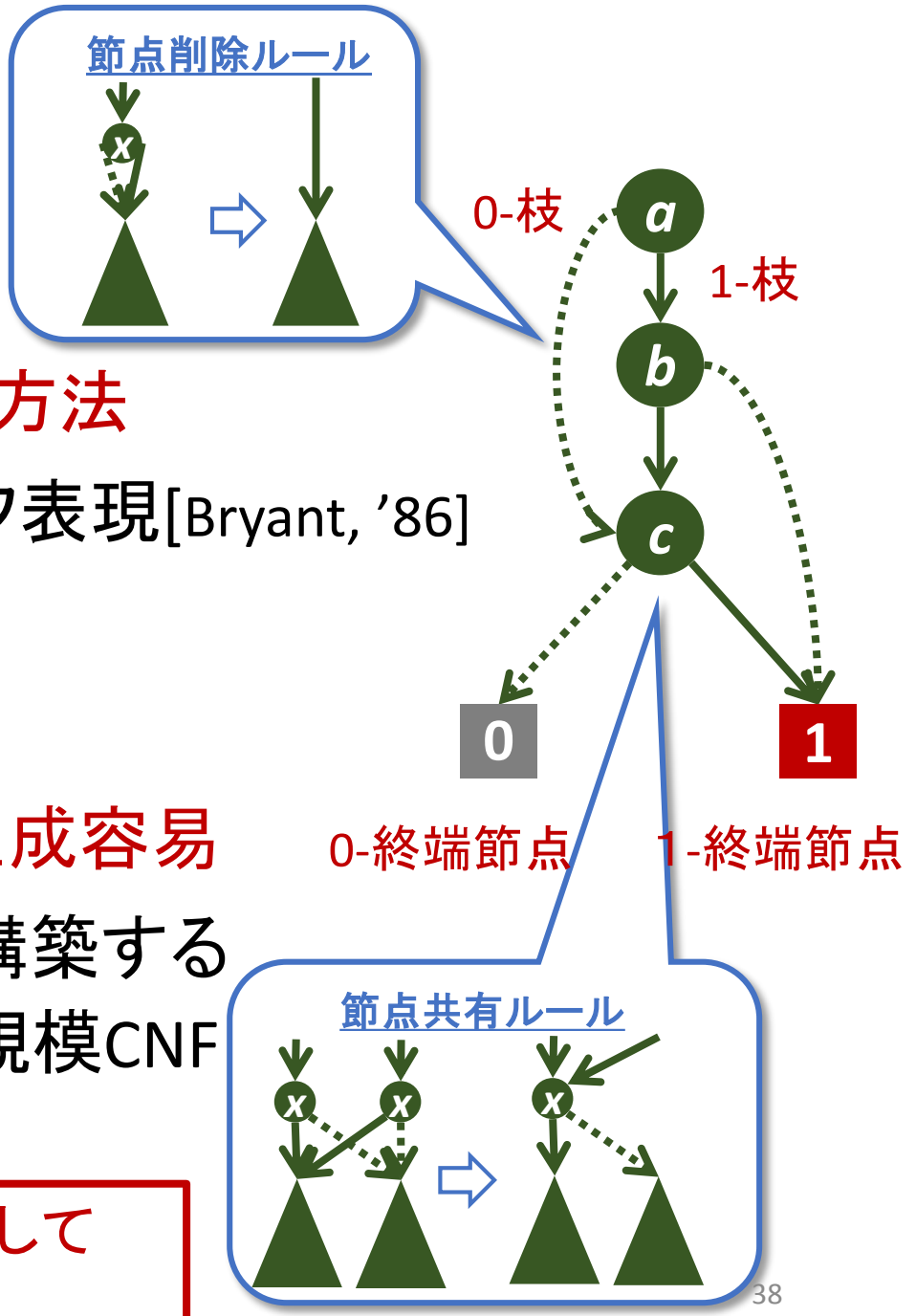


論理演算によるBDD構築の例

これが典型的アプローチ。個々の演算は効率的だが、中間BDDサイズが増大しがち



BDD型ソルバ



- CNFからBDDを構築する方法
- BDDは論理関数のグラフ表現[Bryant, ’86]
  - 簡潔さ
  - 各種の論理演算、クエリ
- BDDが作れたら、解の生成容易
- 論理演算を用いてBDD構築する典型的アプローチは大規模CNFには不向き

BDD型ソルバはCDCLを活用して
効率的にBDDを構築

知識コンパイル

- 知識コンパイル [Darwiche, A., Marquis, P., ‘02]
  命題論理の表現を与える様々な言語を分類
  ①簡潔さ、②サポートするクエリ・変換の計算量
  - 一方の言語で記述された知識ベースを、
    上記2点の観点から、実用に適した別の言語に変換
  - CNFからOBDDの計算は一種の知識コンパイル
- 探索の言語 [Huang, J., Darwiche, A., ’07]
  自動推論において用いられる各種の網羅的探索アルゴリズムは、命題論理の言語に対応する

変数順を固定した網羅的なDPLL ⇔ OBDD

（＊）DPLL＋矛盾からの節学習＝CDCL

OBDDコンパイル

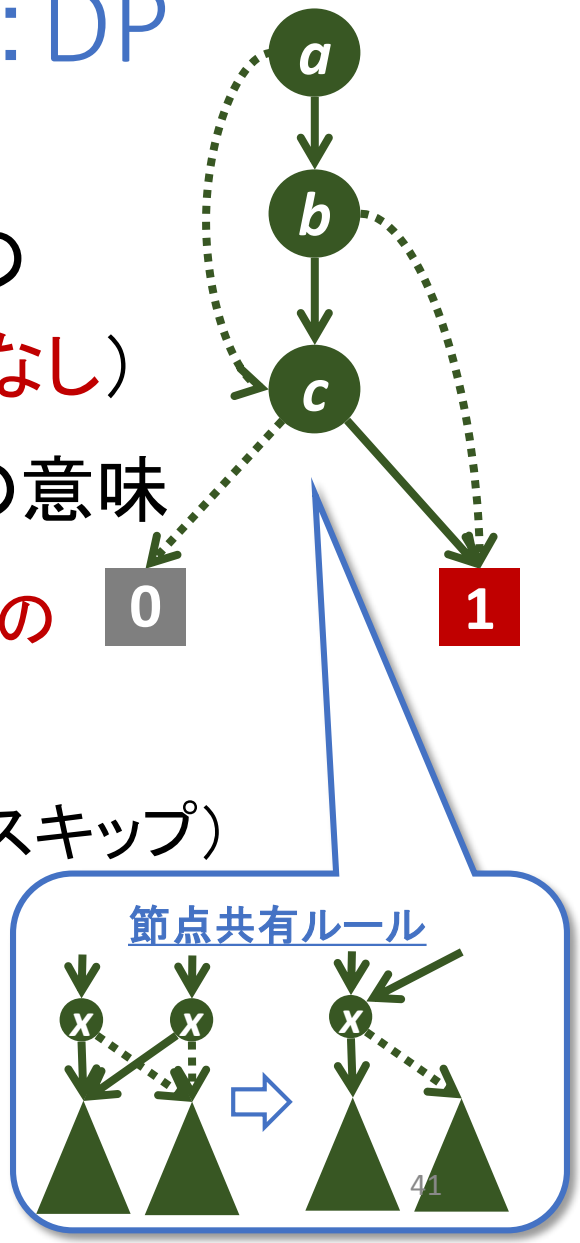
- （変数順を固定した）DPLLアルゴリズムの探索プロセスをグラフとして表現したものはOBDDに対応
- SATソルバを動かしながら同時にOBDD構築可能！
- Huang and Darwicheの貢献：
  ① SAT技術をBDDコンパイルに活用
  ②知識コンパイルと自動推論技法（SATなど）の橋渡し

AllSATが主目的ではなく、他のAllSATアルゴリズムとの比較評価はなかった。

[Huang, J., Darwiche, A., ’05]

BDD型ソルバのポイント：DP

- ブロッキング型や非ブロッキング型の
  トレースは二分木に対応（節点共有なし）
- 探索と同時にOBDDを構築することの意味
  - OBDD＝既に解いた部分問題と解空間の
    対応を与える“DPテーブル“
  - 等価な問題を一度だけしか解かない（スキップ）
  - スキップできればできるほど、
    探索の手間を省ける（指数的な効果）

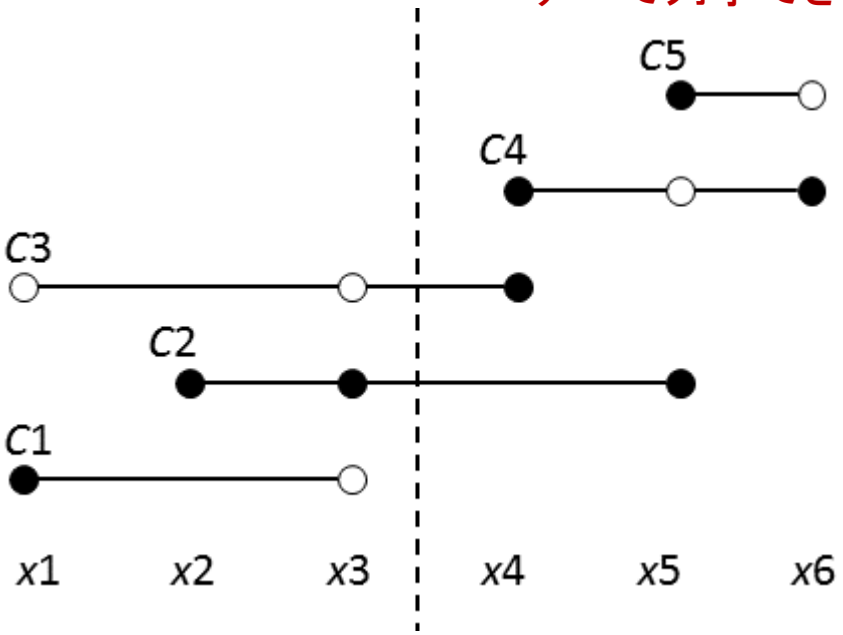


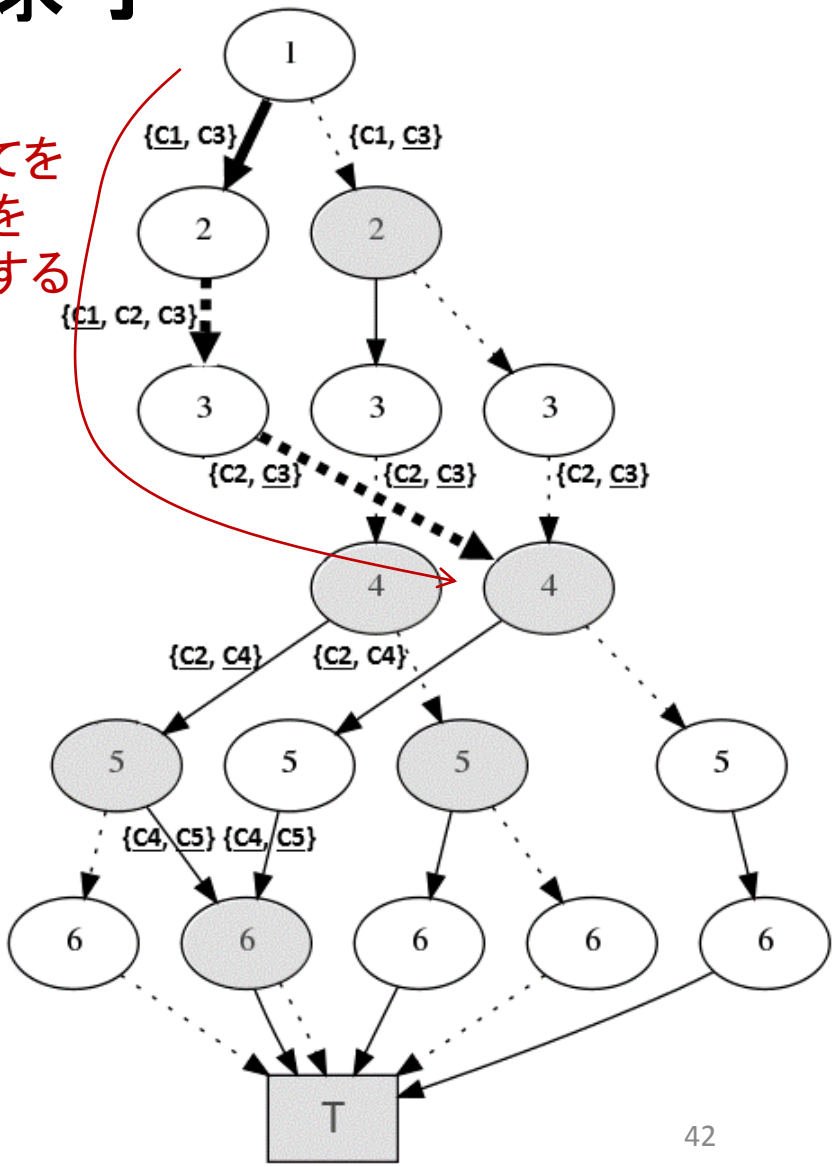
下線は節が充足されたことを意味する

例）BDDの拡大の様子

[Toda, T., Tsuda, K., ’15]

今、この部分割り当てを
延長して得られる解を
すべて列挙できたとする





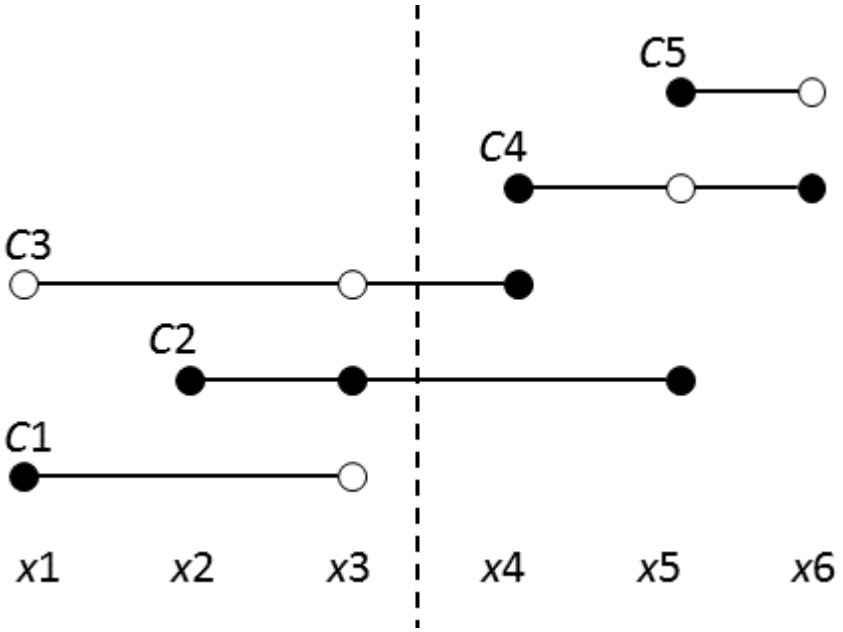
区間によって表現されたCNF：
黒丸が正リテラル、白丸が負リテラル、
1つの区間が1つの節を表す。

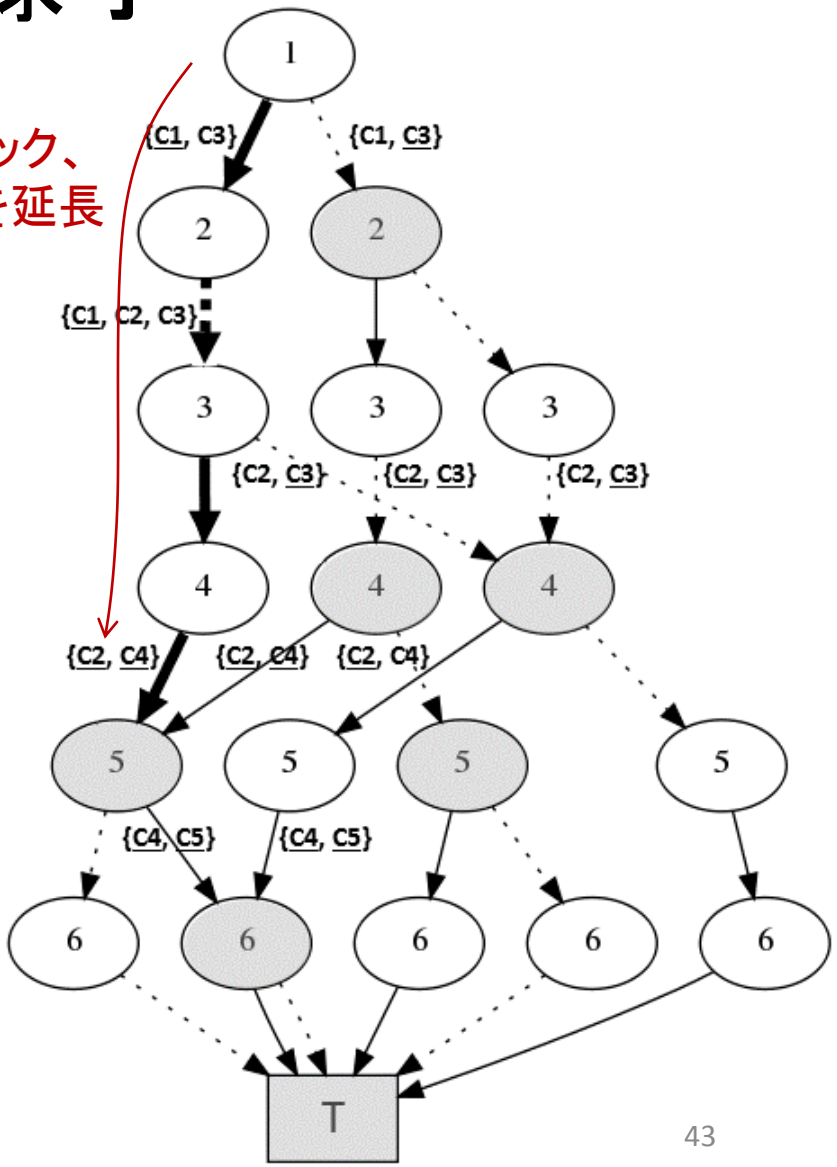
下線は節が充足されたことを意味する

例）BDDの拡大の様子

[Toda, T., Tsuda, K., ’15]

①時間順バックトラック、
次の部分割り当てを延長





区間によって表現されたCNF：
黒丸が正リテラル、白丸が負リテラル、
1つの区間が1つの節を表す。

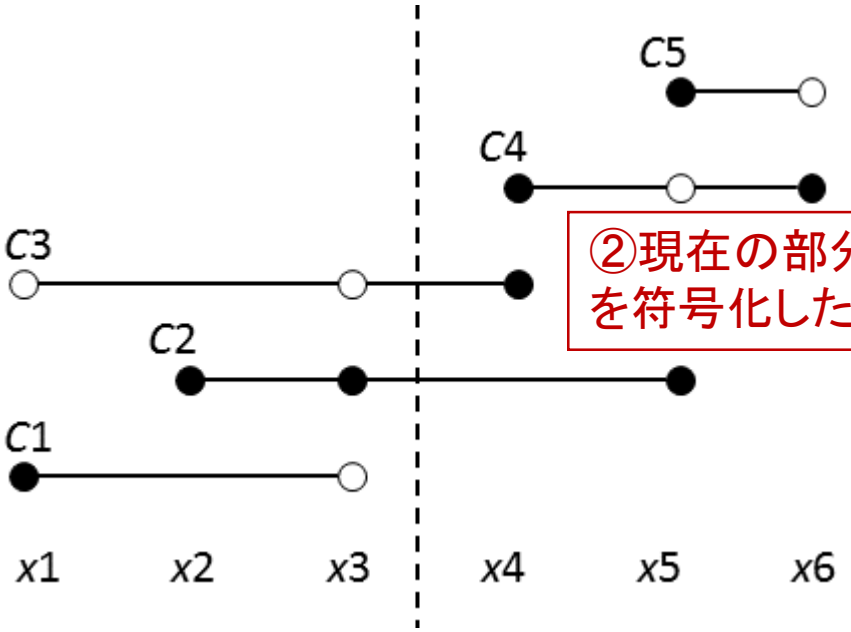
下線は節が充足されたことを意味する

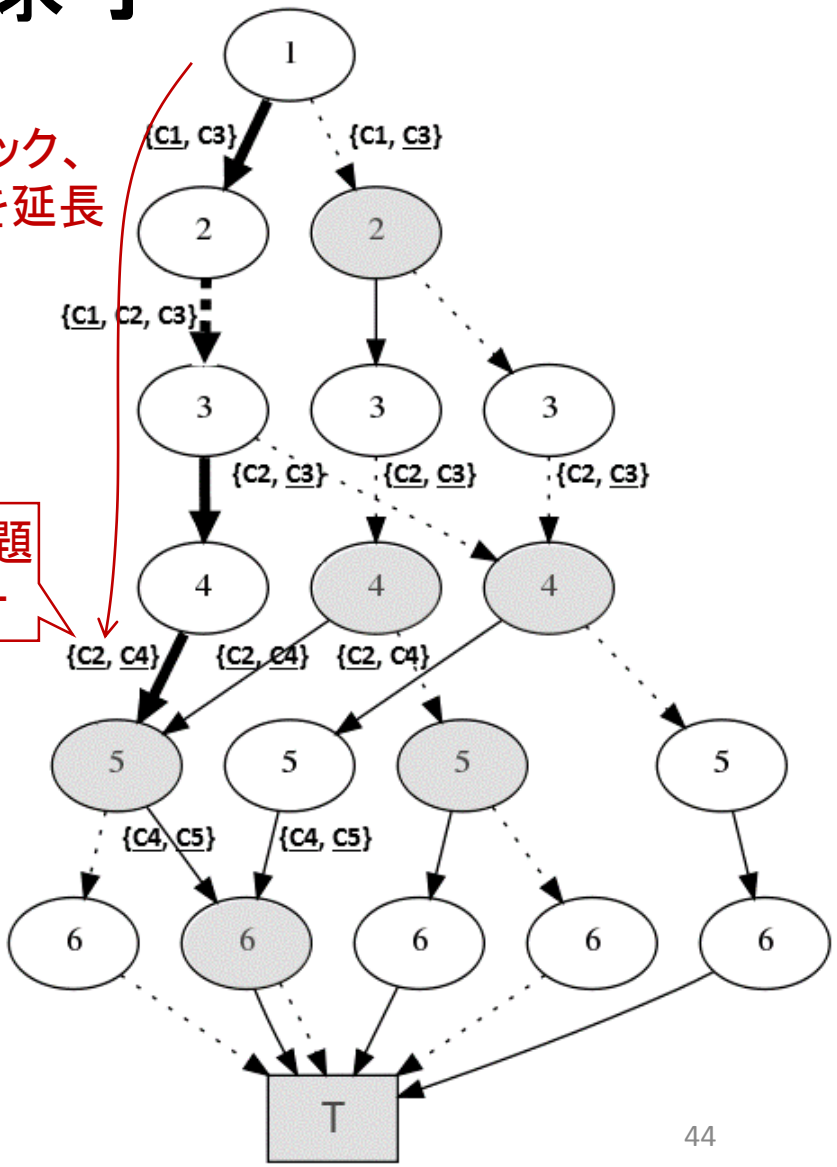
例）BDDの拡大の様子

[Toda, T., Tsuda, K., ’15]

①時間順バックトラック、
次の部分割り当てを延長

②現在の部分問題
を符号化したキー





区間によって表現されたCNF：
黒丸が正リテラル、白丸が負リテラル、
1つの区間が1つの節を表す。

下線は節が充足されたことを意味する

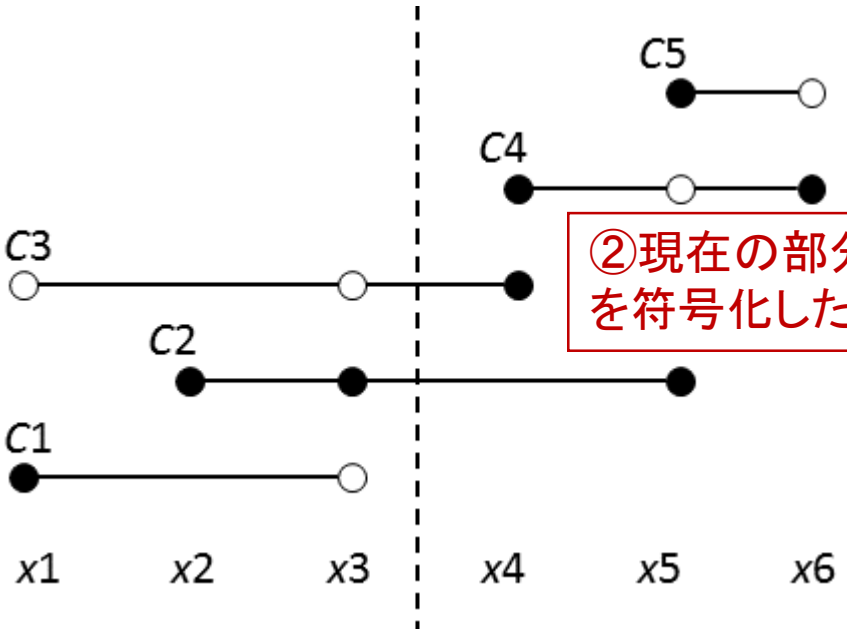
例）BDDの拡大の様子

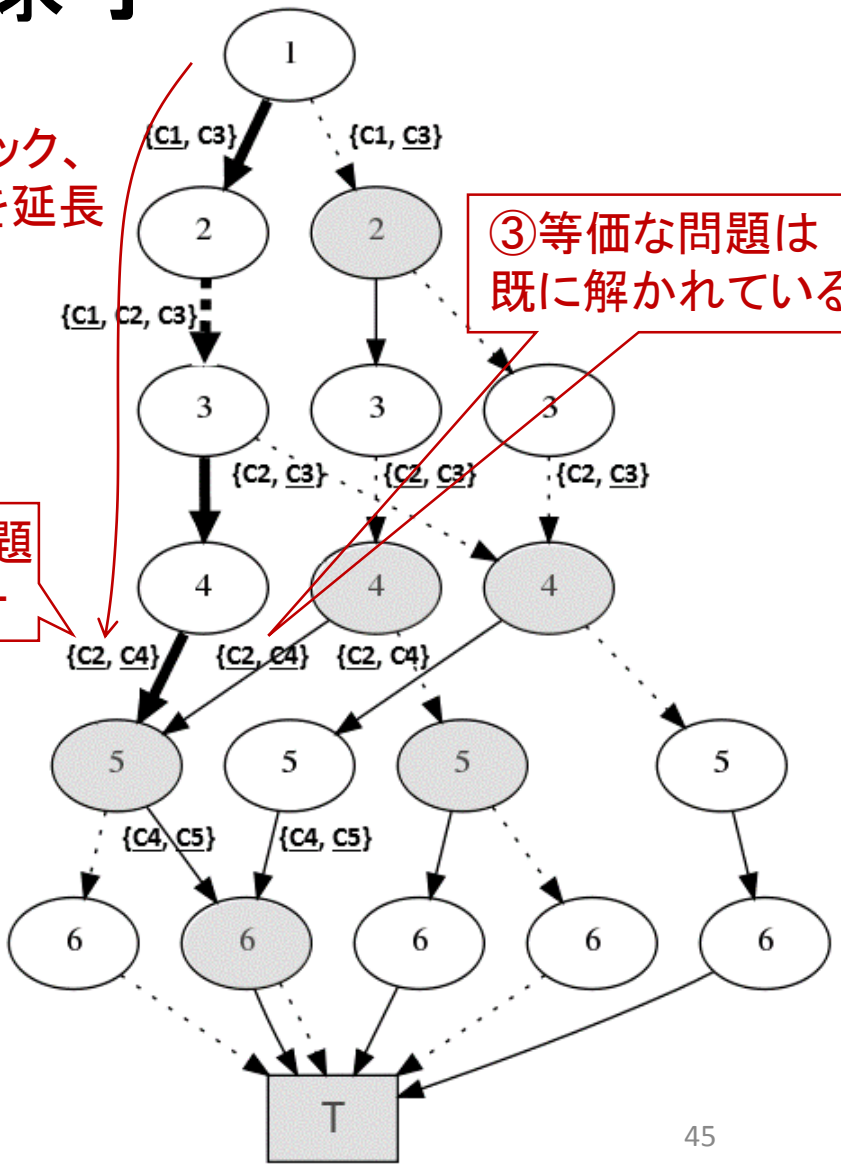
[Toda, T., Tsuda, K., ’15]

①時間順バックトラック、
次の部分割り当てを延長

②現在の部分問題
を符号化したキー

③等価な問題は
既に解かれている



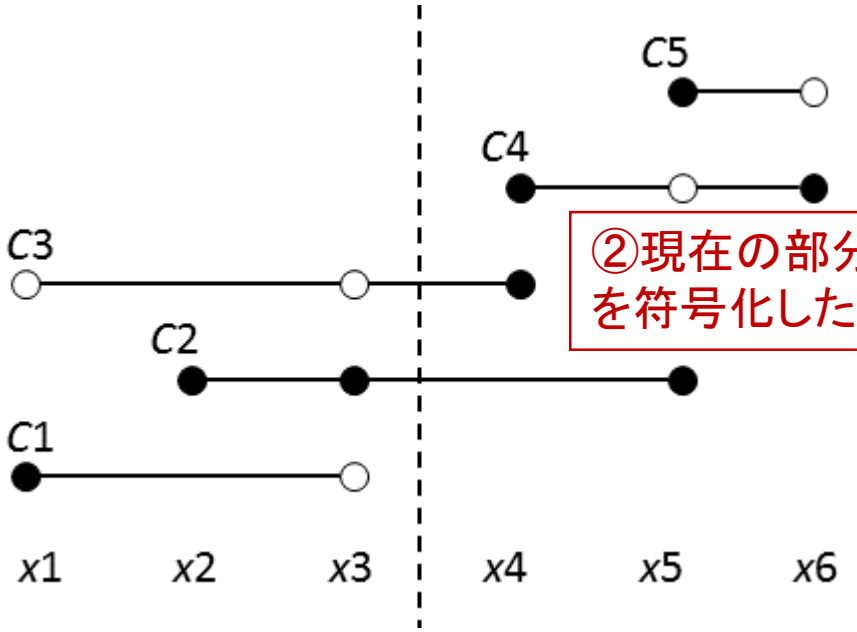


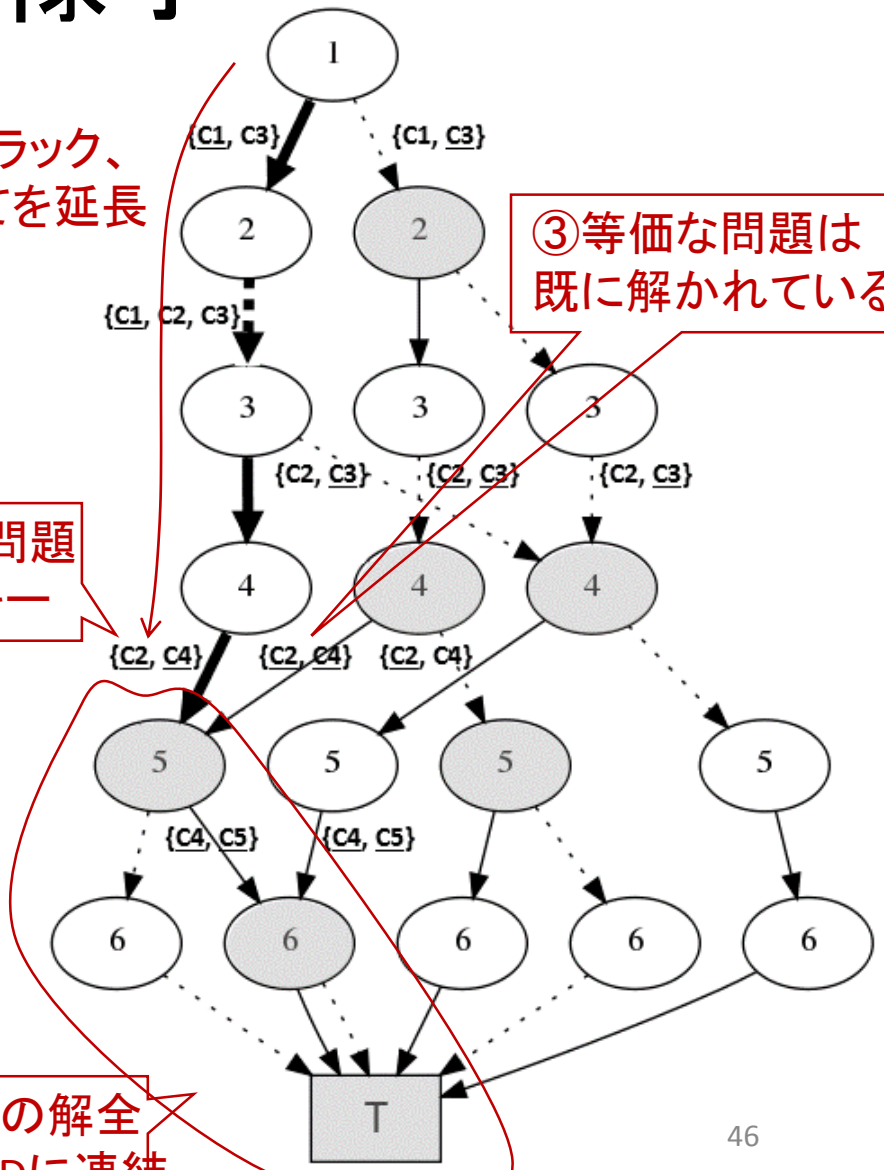
区間によって表現されたCNF：
黒丸が正リテラル、白丸が負リテラル、
1つの区間が1つの節を表す。

下線は節が充足されたことを意味する

例）BDDの拡大の様子

[Toda, T., Tsuda, K., ’15]





①時間順バックトラック、
次の部分割り当てを延長

②現在の部分問題
を符号化したキー

③等価な問題は
既に解かれている

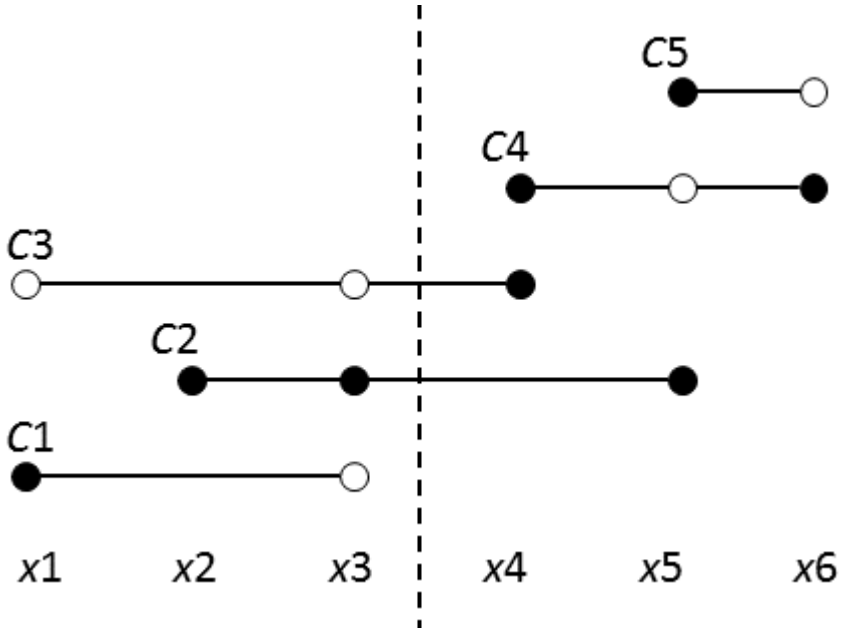
④部分問題の解全
体を表すBDDに連結

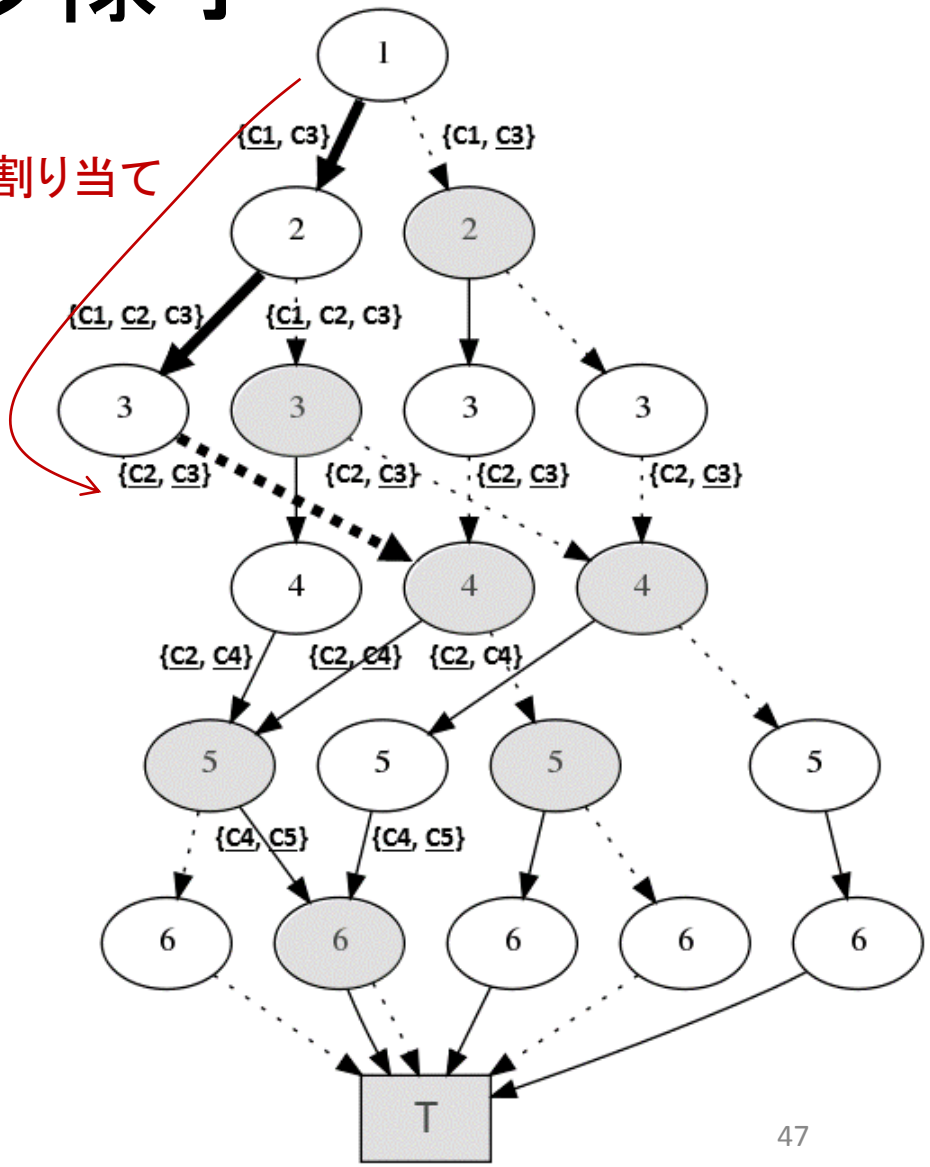
区間によって表現されたCNF：
黒丸が正リテラル、白丸が負リテラル、
1つの区間が1つの節を表す。

下線は節が充足されたことを意味する

例）BDDの拡大の様子

[Toda, T., Tsuda, K., ’15]



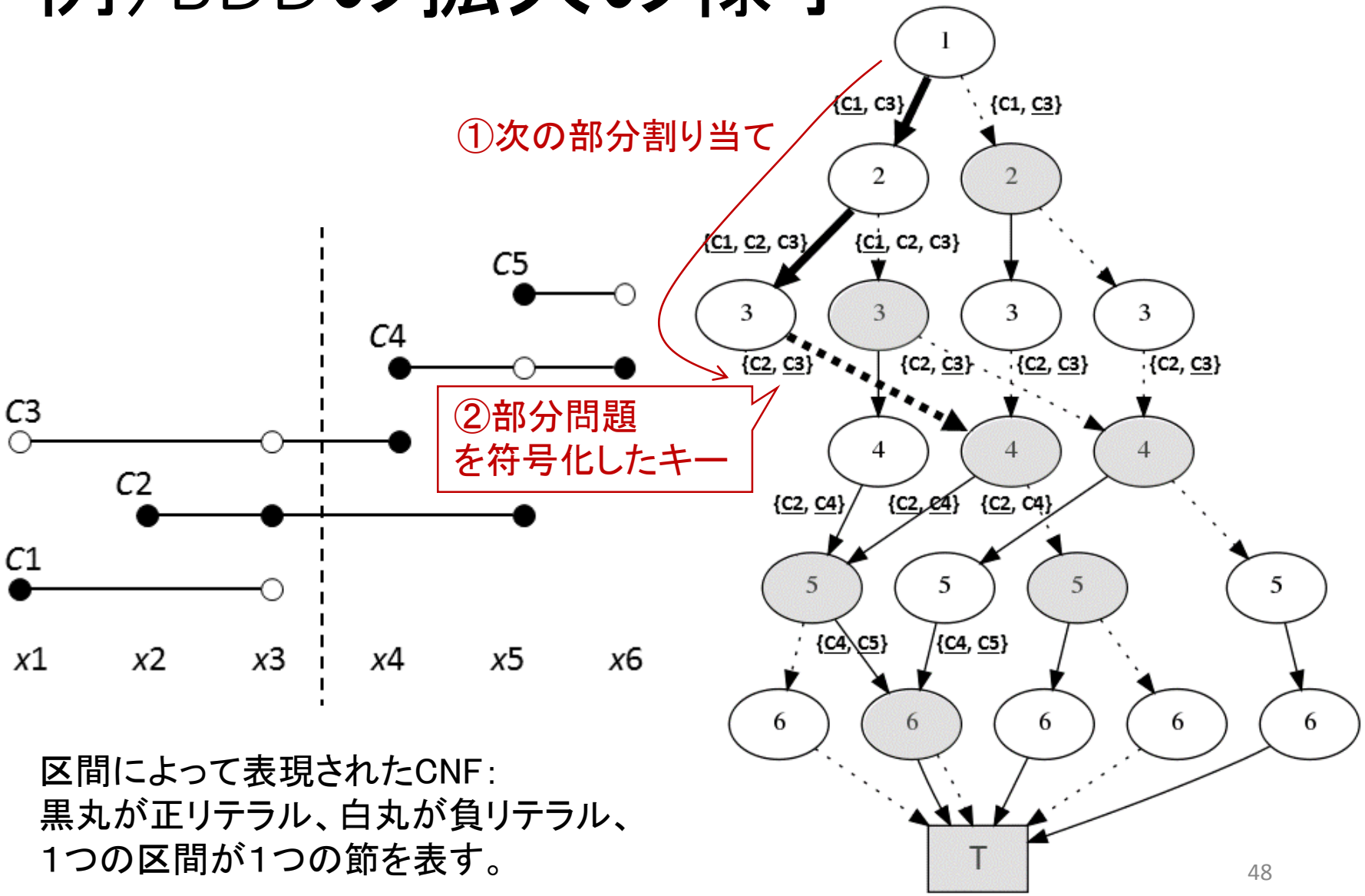


区間によって表現されたCNF：
黒丸が正リテラル、白丸が負リテラル、
1つの区間が1つの節を表す。

下線は節が充足されたことを意味する

例）BDDの拡大の様子

[Toda, T., Tsuda, K., ’15]

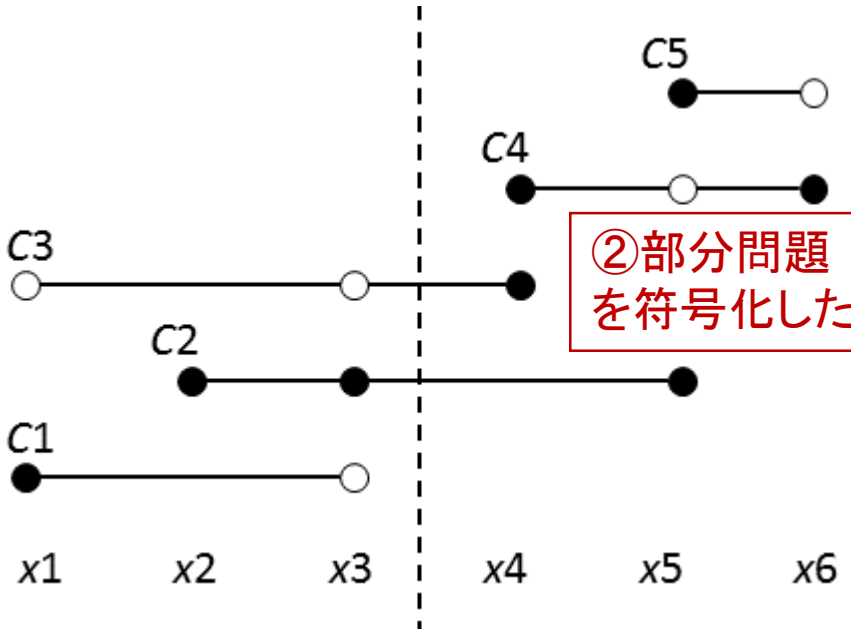


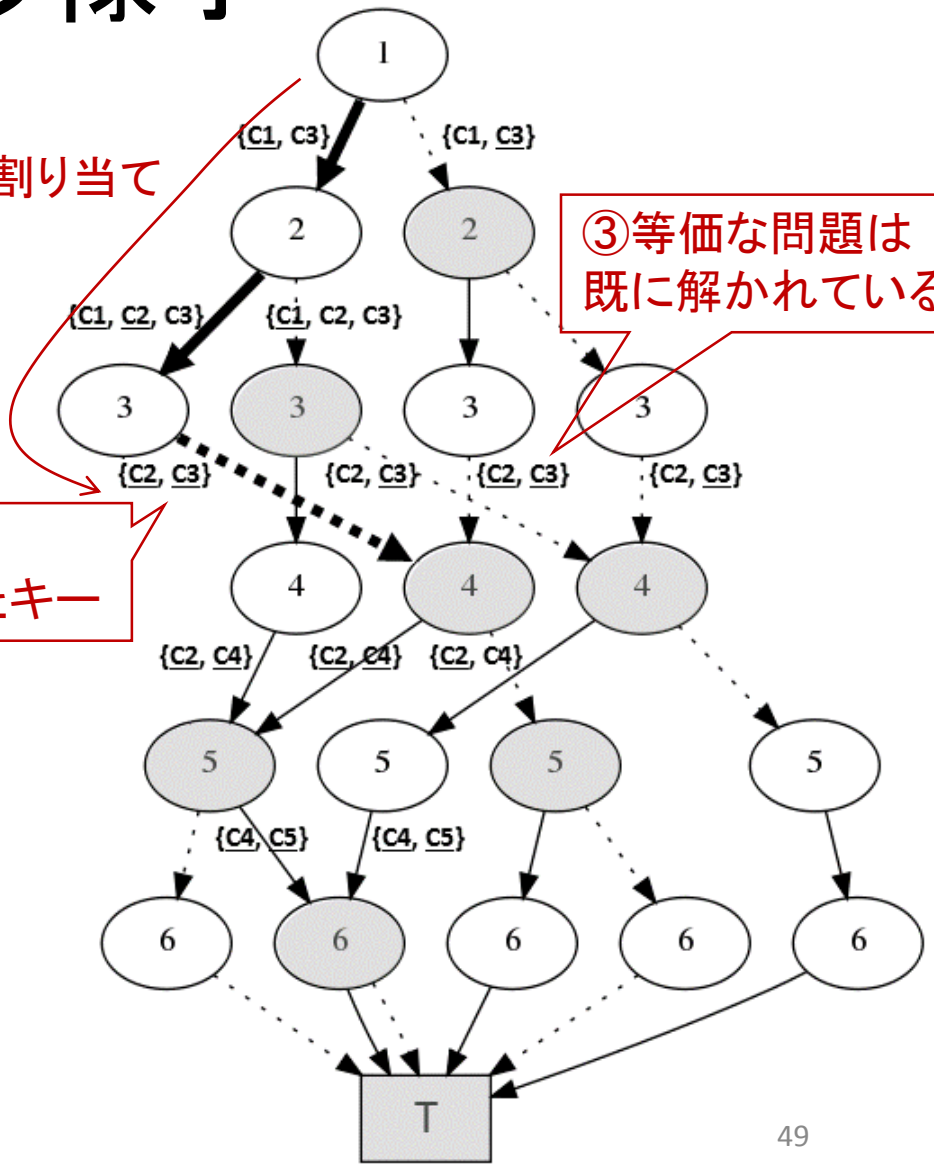
区間によって表現されたCNF：
黒丸が正リテラル、白丸が負リテラル、
1つの区間が1つの節を表す。

下線は節が充足されたことを意味する

例）BDDの拡大の様子

[Toda, T., Tsuda, K., ’15]





①次の部分割り当て

②部分問題
を符号化したキー

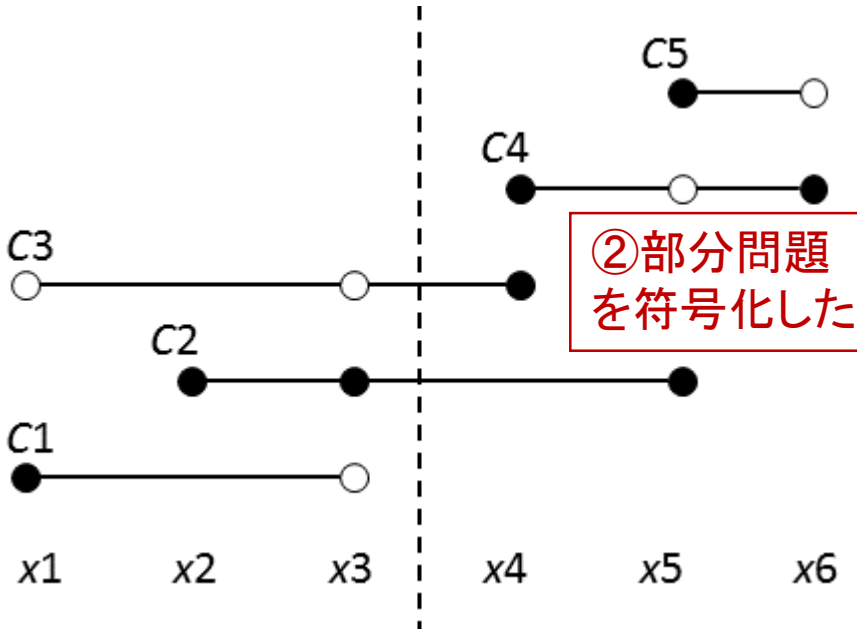
③等価な問題は
既に解かれている

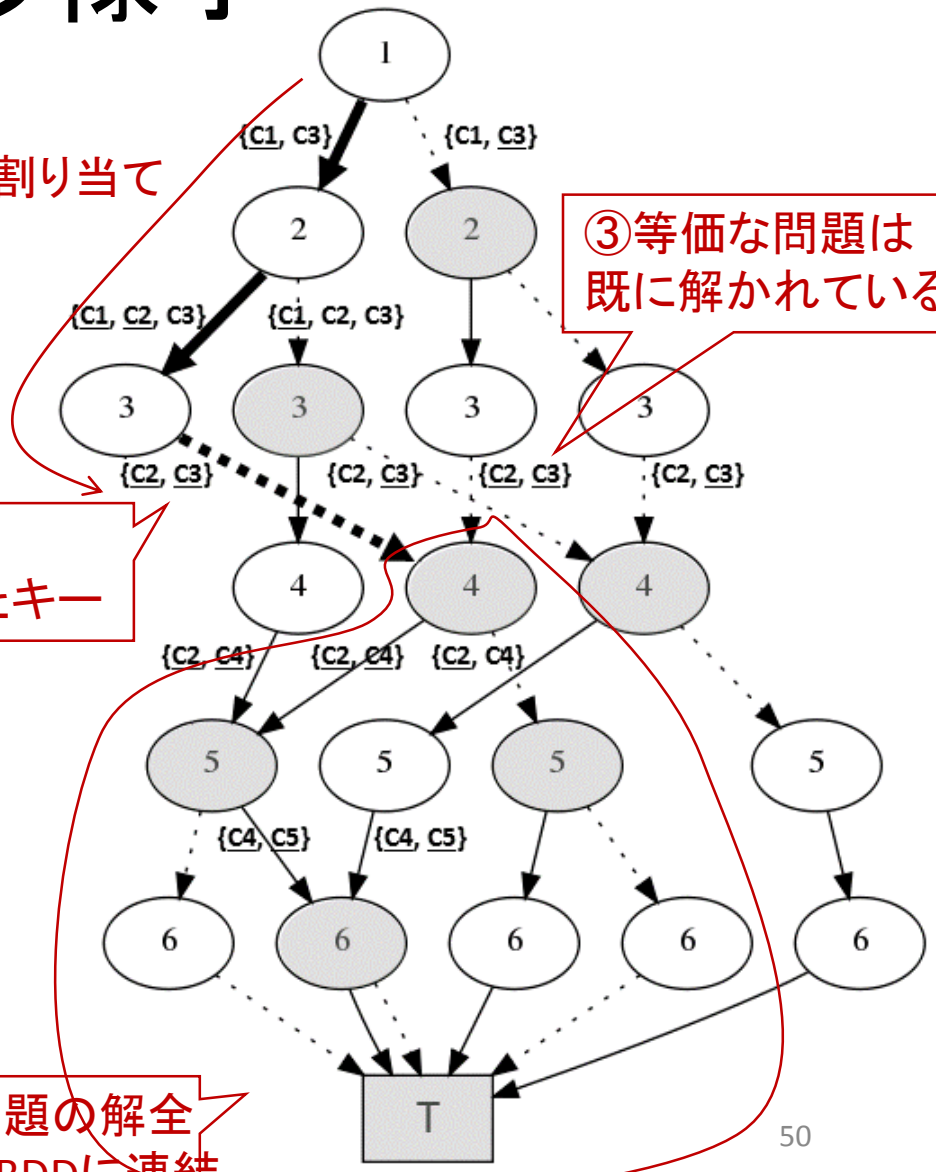
区間によって表現されたCNF：
黒丸が正リテラル、白丸が負リテラル、
1つの区間が1つの節を表す。

下線は節が充足されたことを意味する

例）BDDの拡大の様子

[Toda, T., Tsuda, K., ’15]





①次の部分割り当て

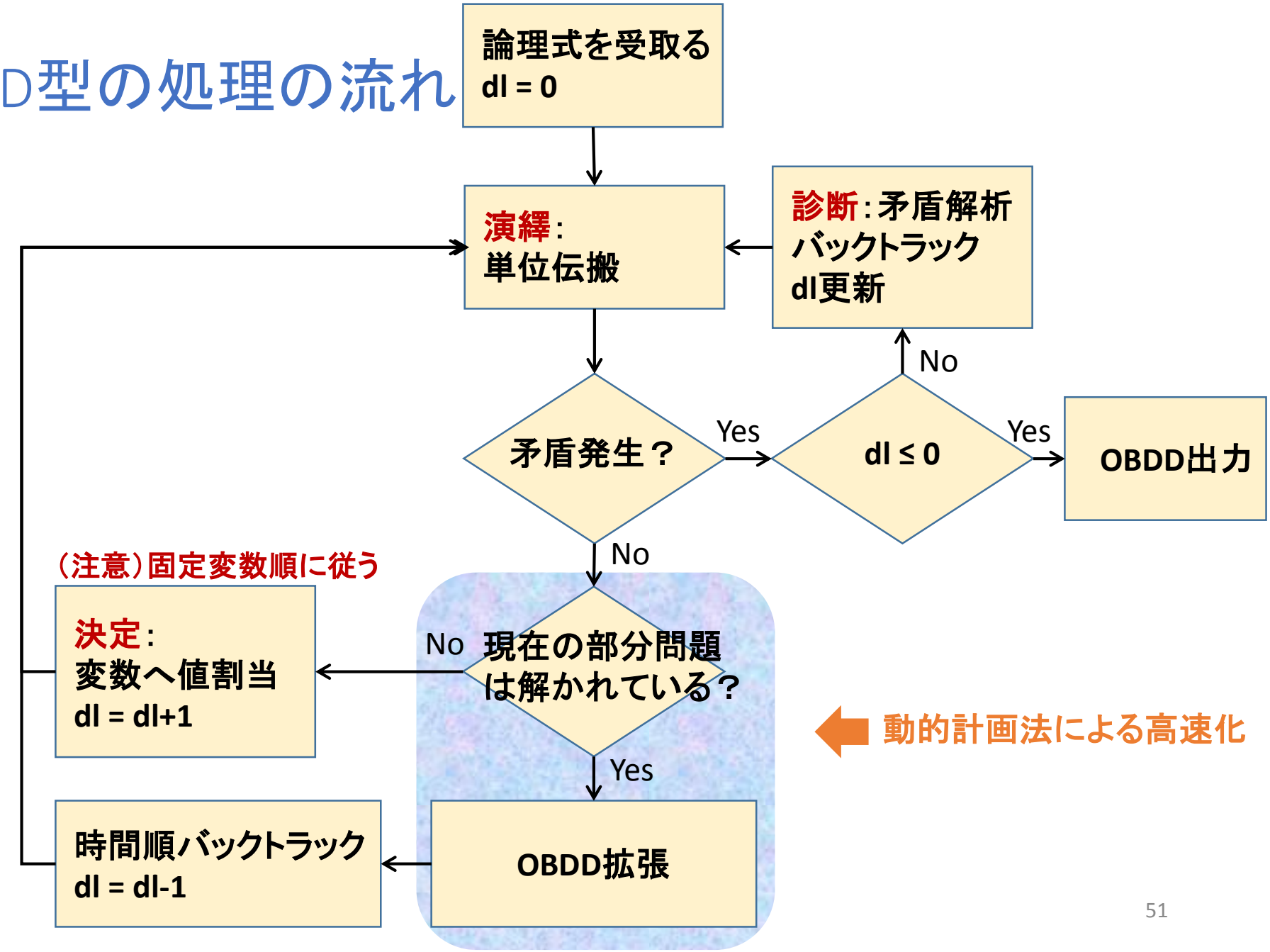
②部分問題
を符号化したキー

③等価な問題は
既に解かれている

④部分問題の解全
体を表すBDDに連結

区間によって表現されたCNF：
黒丸が正リテラル、白丸が負リテラル、
1つの区間が1つの節を表す。

BDD型の処理の流れ


論理式を受取る
dl = 0
演繹:
単位伝搬
診断:矛盾解析
バックトラック
dl更新
No
矛盾発生？
Yes
dl ≤ 0
Yes
OBDD出力
No
(注意)固定変数順に従う
決定:
変数へ値割当
dl = dl+1
No
現在の部分問題
は解かれている？
Yes
時間順バックトラック
dl = dl-1
OBDD拡張
動的計画法による高速化

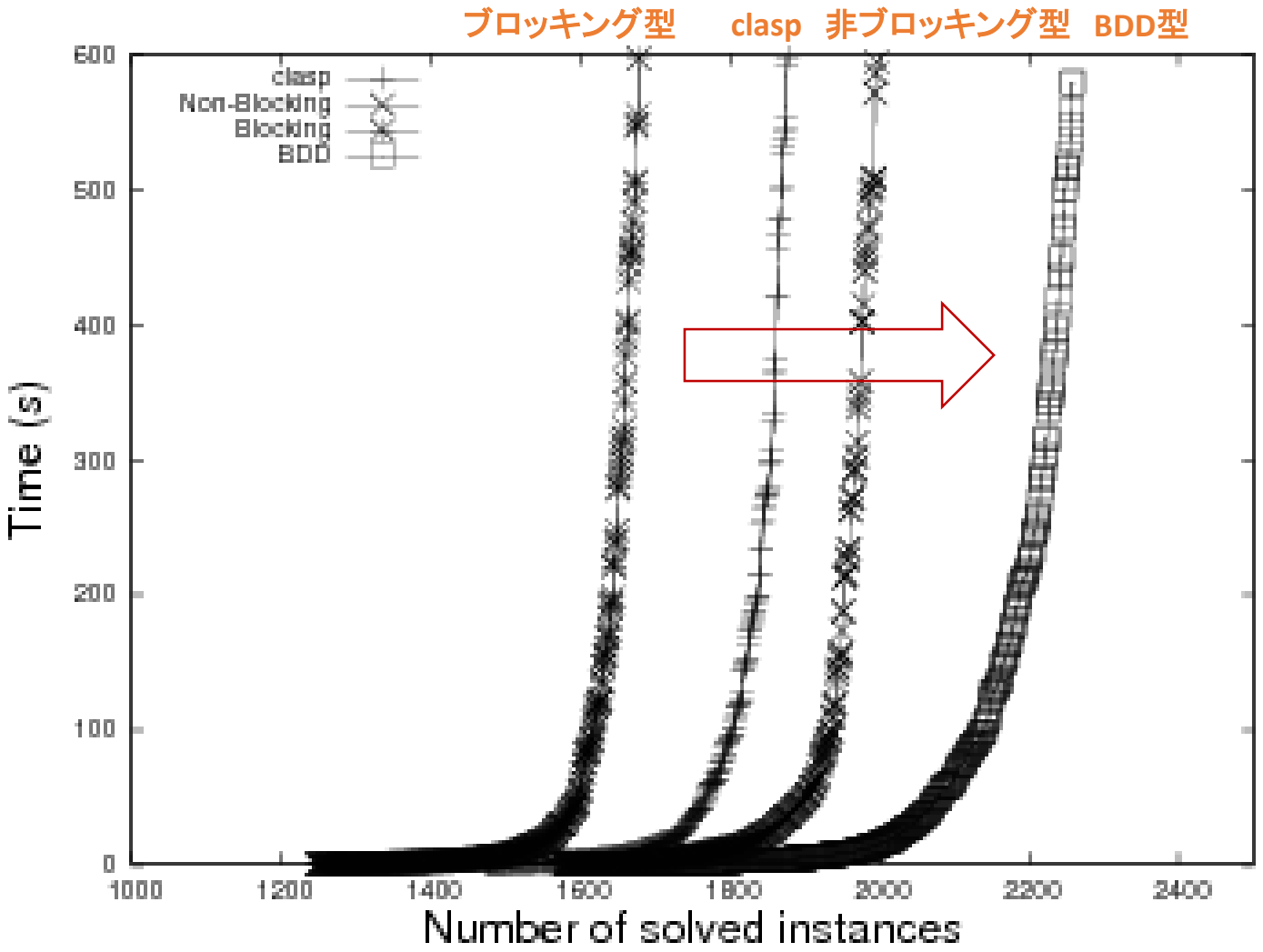
目次

- AllSAT問題
- AllSATの応用
- 代表的ソルバの解説
- 性能評価
- まとめ

実験の設定・環境

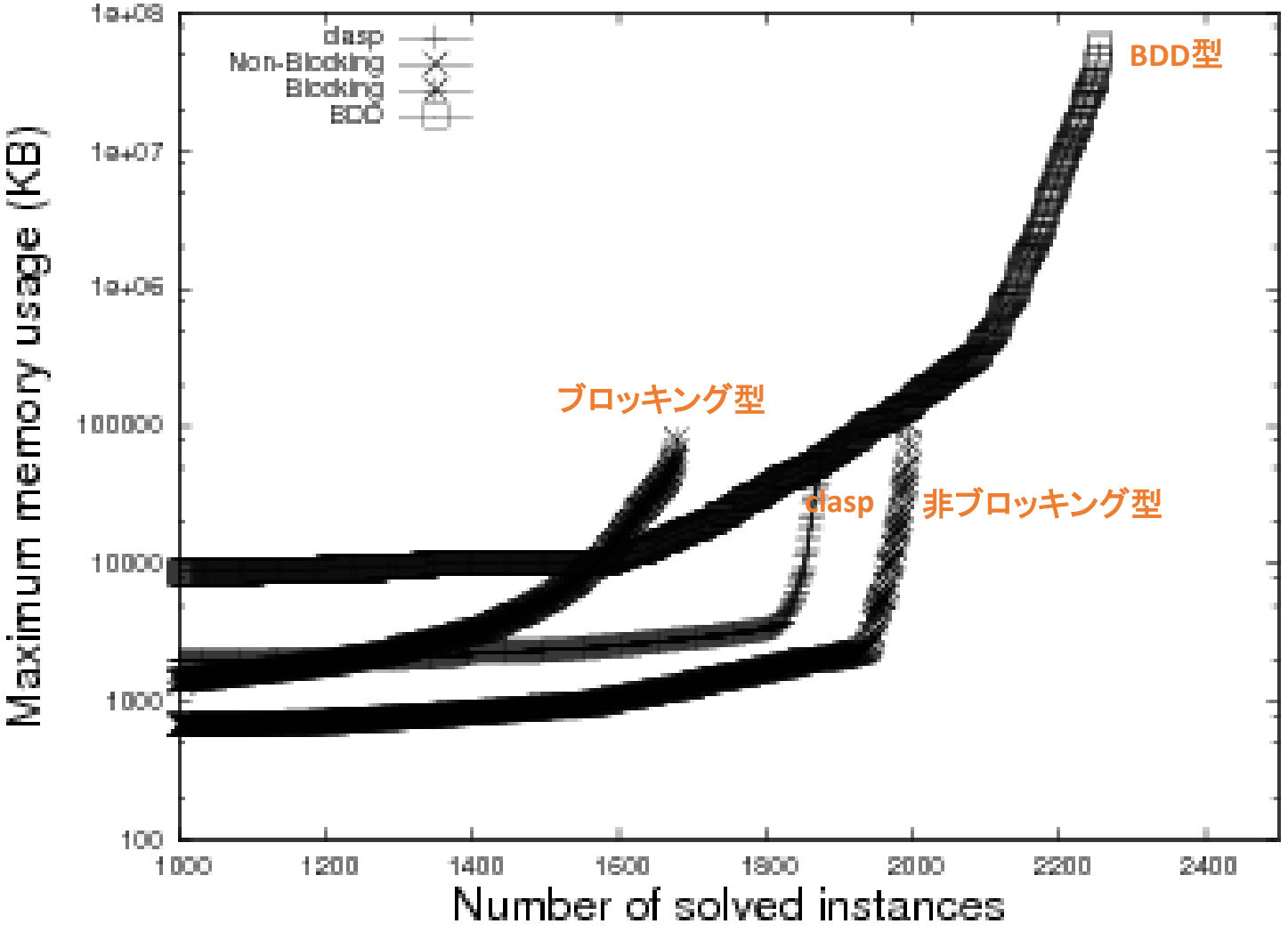
- **比較ソルバ**（設定の詳細は省く）
  - **ブロッキング型**
  - **非ブロッキング型**
  - **BDD型**
  - **Clasp**：ASPソルバ（ポスダム大Potasscoプロジェクト開発）
- **インスタンス**：合計2930個のCNF
  - SATLIB, SAT Completition 2014, ISCAS85/89で提供されているCNFのうち、clasp 3.1.2, glucose 4, minisat 2.2, minisat 1.3のいづれかで600秒以内にSATと判定できたもの（計2867個）
  - CP4IMで提供されているトランザクションデータから頻出パタンマイニング問題をCNFで符号化して得られたもの（計63個）
- **制限時間**：600秒

実行時間の比較



(＊)インスタンスが解ける:そのインスタンスの全ての充足割当(解)を制限時間内に計算できる
グラフの見方:各プロット点は解けたインスタンスを表す。Y軸はそのインスタンスを解くのに要した時間、
X軸は解けたインスタンスの中での順番を表す。緩やかな曲線を描くソルバほど効率的。

最大使用メモリの比較


dlasp
Non-Blocking
Blocking
BDD
BDD型
ブロッキング型
dlasp
非ブロッキング型
Maximum memory usage (KB)
1e+08
1e+07
1e+06
100000
10000
1000
100
1000
1200
1400
1600
1800
2000
2200
2400
Number of solved instances

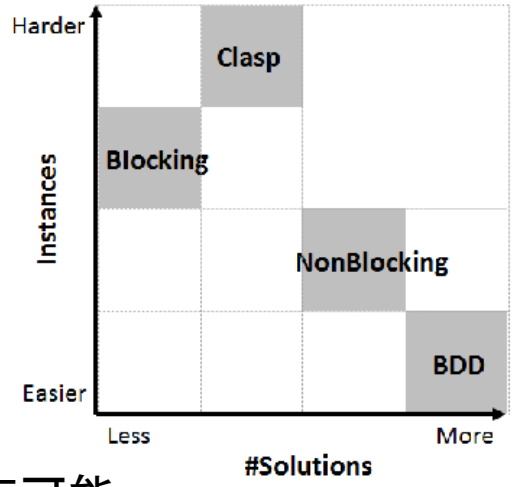
可解インスタンスの分布

各クラスに属する可解インスタンスの個数

インスタンスの持つ解の個数に関するクラス分類

	Blocking	clasp	Non-Blocking	BDD	
$(0, 10^1)$	47	45	48	45	
$[10^1, 10^2)$	20	20	20	18	
$[10^2, 10^3)$	426	426	426	424	
$[10^3, 10^4)$	689	690	690	688	
$[10^4, 10^5)$	413	415	414	412	
$[10^5, 10^6)$	83	138	136	136	100万
$[10^6, 10^7)$	0	87	83	86	
$[10^7, 10^8)$	0	56	88	91	1億
$[10^8, 10^9)$	0	0	68	83	
$[10^9, 10^{10})$	0	0	23	92	100億
$[10^{10}, 10^{11})$	0	0	0	83	
$[10^{11}, 10^{12})$	0	0	0	44	
$[10^{12}, 10^{13})$	0	0	0	29	
$[10^{13}, 10^{14})$	0	0	0	19	
$[10^{14}, \infty)$	0	0	0	8	≥ 1000兆
Total	1,678	1,877	1,996	2,258	

評価結果のまとめ



- **BDD型**
  - 可解インスタンスが最多
  - 1000兆以上の充足割当をもつインスタンスにも対応可能
  - 大メモリ必要だが、キャシュヒット率を犠牲にして使用メモリ制御可能
- **非ブロッキング型、および、Clasp**
  - それぞれ高々100億，1億程度の充足割当を持つインスタンス
  - 両者は似た特性を示す
  - Claspは難しいインスタンスに向く
- **ブロッキング型**
  - 高々100万程度の充足割当を持つインスタンス
  - 少数の充足割当をもつ，難しいインスタンスに向く

BDDリフレッシュ機能
詳細は[Toda, T., Soh, T., '16]
の4.3.4節をご参照ください。

全体のまとめ

- AllSAT問題 ：SATの列挙版
  - 論理における列挙問題：双対化
- 各種の応用
  - モデル検査
  - ネットワーク検証
  - データマイニング
- 従来研究のサーベイ、新規手法の提案、
  網羅的な評価、主要ソルバの実装公開

  All Solutions SAT Repository

  http://www.sd.is.uec.ac.jp/toda/code/allsat.html

参考文献

- Takahisa Toda, Takehide Soh, Implementing Efficient All Solutions SAT Solvers, ACM Journal of Experimental Algorithmics, Vol. 21, No. 1, Article 1.12, 2016.
- 番原 睦則,宋 剛秀,田村 直之,井上 克巳,私のブックマーク:SAT ソルバー, Vol.28 No.2 (2013/03), https://www.ai-gakkai.or.jp/my-bookmark_vol28-no2/
- Yves Crama and Peter L. Hammer. 2011. Boolean Functions—Theory, Algorithms, and Applications. Encyclopedia of mathematics and its applications, Vol. 142. Cambridge University Press, Cambridge. http://www.cambridge.org/gb/knowledge/isbn/item6222210/?site_locale=en_GB.
- Eiter, T., Gottlob, G.: Hypergraph transversal computation and related problems in logic and AI. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS (LNAI), vol. 2424, pp. 549–564. Springer, Heidelberg (2002)
- Eiter, T., Makino, K., Gottlob, G.: Computational aspects of monotone dualization: A brief survey. Discrete Applied Mathematics 156, 2035–2049 (2008)
- Keisuke Murakami and Takeaki Uno. 2014. Efficient algorithms for dualizing large-scale hypergraphs. Discrete Appl. Math. 170 (2014), 83–94. DOI:http://dx.doi.org/10.1016/j.dam.2014.01.012
- Takahisa Toda. 2013. In Proceedings of the 12th International Symposium on Experimental Algorithms (SEA2013). Springer, Berlin, 91–102. DOI:http://dx.doi.org/10.1007/978-3-642-38527-8_10
- Gainer-Dewar, Andrew, and Paola Vera-Licona. “The minimal hitting set generation problem: algorithms and computation.” *SIAM Journal on Discrete Mathematics* 31.1 (2017): 63-100.
- Randal E. Bryant. 1986. Graph-based algorithms for boolean function manipulation. IEEE Trans. Comput. C-35, 8 (Aug. 1986), 677–691. DOI:http://dx.doi.org/10.1109/TC.1986.1676819
- Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. 1999. Symbolic model checking without BDDs. In Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'99). Springer-Verlag, London, UK, 193–207. http://dl.acm.org/citation.cfm?id=646483.691738
- Junaid Qadir and Osman Hasan. 2015. Applying formal methods to networking: Theory, techniques, and applications. IEEE Commun. Surv. Tutorials. 17, 1 (First quarter 2015), 256–291. DOI:http://dx.doi.org/10.1109/COMST.2014.2345792
- 山崎智史, 八鍬豊, 登内敏夫. "形式手法による SDN/OpenFlow 高信頼化技術の研究動向." *コンピュータ ソフトウェア* 33.2 (2016): 2_26-2_42.
- Shuyuan Zhang, Sharad Malik, and Rick McGeer. 2012. In Proceedings of the 10th International Symposium on Automated Technology for Verification and Analysis. Springer, Berlin, 1–16. DOI:http://dx.doi.org/10.1007/978-3-642-33386-6_1
- Nuno Lopes, Nikolaj Bjørner, Patrice Godefroid, and George Varghese. 2013. Network Verification in the Light of Program Verification. Technical Report. Microsoft Research. http://research.microsoft.com/apps/pubs/default.aspx?id=201589.
- Nuno P. Lopes, Nikolaj Bjørner, Patrice Godefroid, Karthick Jayaraman, and George Varghese. 2015. Checking beliefs in dynamic networks. In Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation (NSDI'15). USENIX Association, Berkeley, CA, 499–512. http://dl.acm.org/citation.cfm?id=2789770.2789805
- Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter. 2007. Handbook of Knowledge Representation. Elsevier Science, San Diego.
- Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh. 2009. Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications. IOS Press, Amsterdam, The Netherlands, The Netherlands.
- Sharad Malik and Lintao Zhang. 2009. Boolean satisfiability from theoretical hardness to practical success. Commun. ACM 52, 8 (Aug. 2009), 76–82. DOI:http://dx.doi.org/10.1145/1536616.1536637
- 鍋島英知, 高速SATソルバーの原理, 湊離散構造処理系プロジェクトERATOセミナー, 2014, http://www-erato.ist.hokudai.ac.jp/docs/seminar/nabeshima.pdf

- 鍋島英知，宋剛秀，高速SATソルバーの原理，人工知能学会誌，25巻1号，pp.68-76，2010年 ― http://ci.nii.ac.jp/naid/110007504954
- Ken L. McMillan. 2002. In Proceedings of the 14th International Conference on Computer Aided Verification (CAV 2002). Springer, Berlin, 250–264. DOI:http://dx.doi.org/10.1007/3-540-45657-0_19
- Orna Grumberg, Assaf Schuster, and Avi Yadgar. 2004. In Proceedings of the 5th International Conference on Formal Methods in Computer-Aided Design (FMCAD 2004). Springer, Berlin, 275–289. DOI:http://dx.doi.org/10.1007/978-3-540-30494-4_20
- Martin Gebser, Benjamin Kaufmann, Andr´e Neumann, and Torsten Schaub. 2007. In Proceedings of the 9th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2007). Springer, Berlin, 260–265. DOI:http://dx.doi.org/10.1007/978-3-540-72200-7_23
- Darwiche, Adnan, and Pierre Marquis. "A knowledge compilation map." Journal of Artificial Intelligence Research 17.1 (2002): 229-264.
- Jinbo Huang and Adnan Darwiche. 2005. In Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing (SAT 2004), Revised Selected Papers. Springer, Berlin, 157–172. DOI:http://dx.doi.org/10.1007/11527695_13
- Jinbo Huang and Adnan Darwiche. 2007. The language of search. J. Artif. Intell. Res. (JAIR) 29 (2007), 191–219. DOI:http://dx.doi.org/10.1613/jair.2097
- Takahisa Toda and Koji Tsuda. 2015. BDD construction for all solutions SAT and efficient caching mechanism. In Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC'15). ACM, New York, NY, 1880–1886. DOI:http://dx.doi.org/10.1145/2695664.2695941

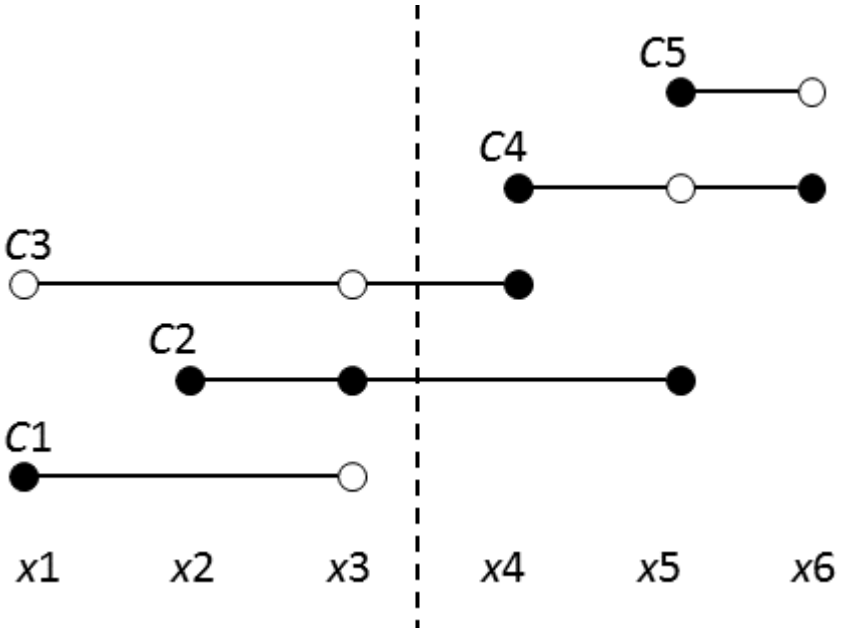
補足資料

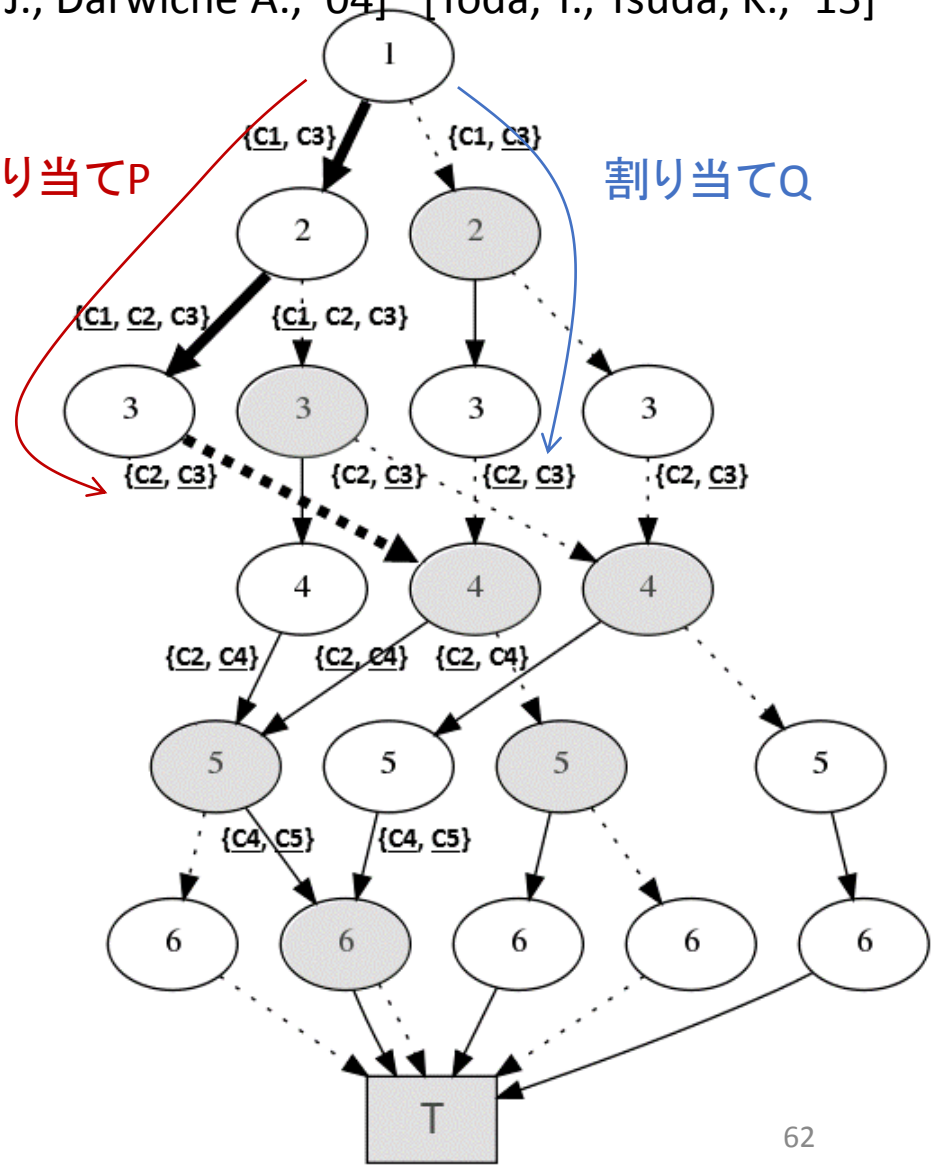
下線は節が充足されたことを意味する

部分問題の符号化の考え方

X3までの割当てPとQは異なるが、
割当てをCNFに適用して得られる
論理式は論理関数として等価

[Huang, J., Darwiche A., ’04] [Toda, T., Tsuda, K., ’15]





区間によって表現されたCNF：
黒丸が正リテラル、白丸が負リテラル、
1つの区間が1つの節を表す。

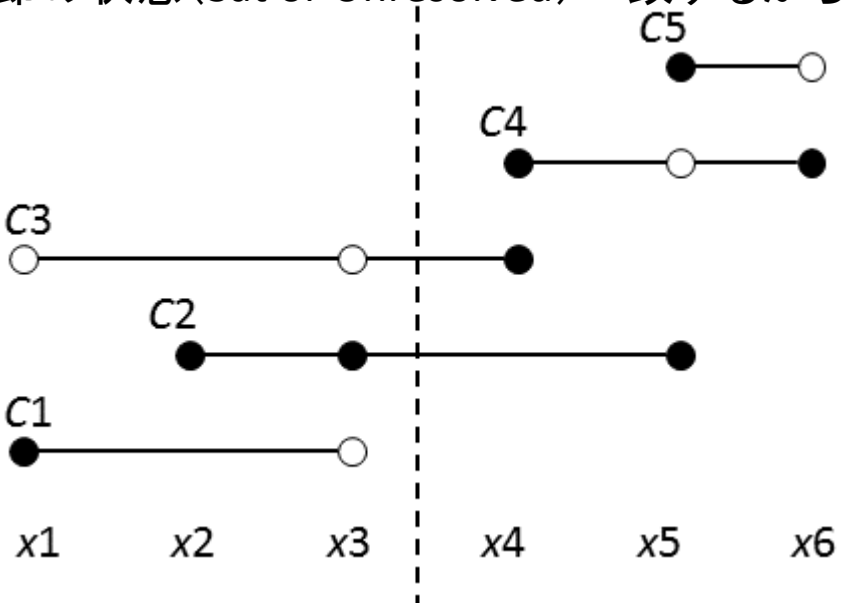
下線は節が充足されたことを意味する

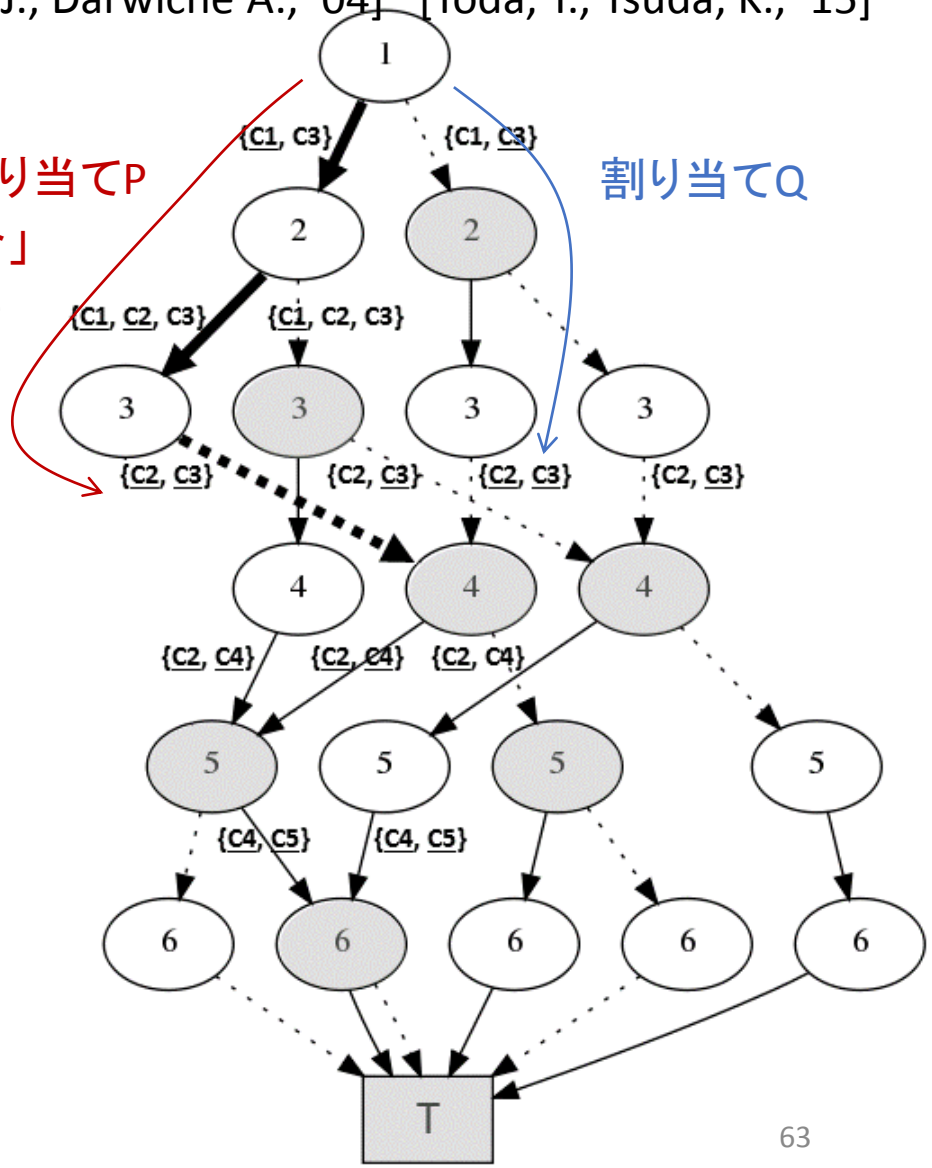
部分問題の符号化の考え方

X3までの割当てPとQは異なるが、
割当てをCNFに適用して得られる
論理式は論理関数として等価

[Huang, J., Darwiche A., ’04] [Toda, T., Tsuda, K., ’15]

「X3のカットセット＝下図の点線をまたぐ節集合」
で各節の状態（Sat or Unresolved）一致するから





区間によって表現されたCNF：
黒丸が正リテラル、白丸が負リテラル、
1つの区間が1つの節を表す。

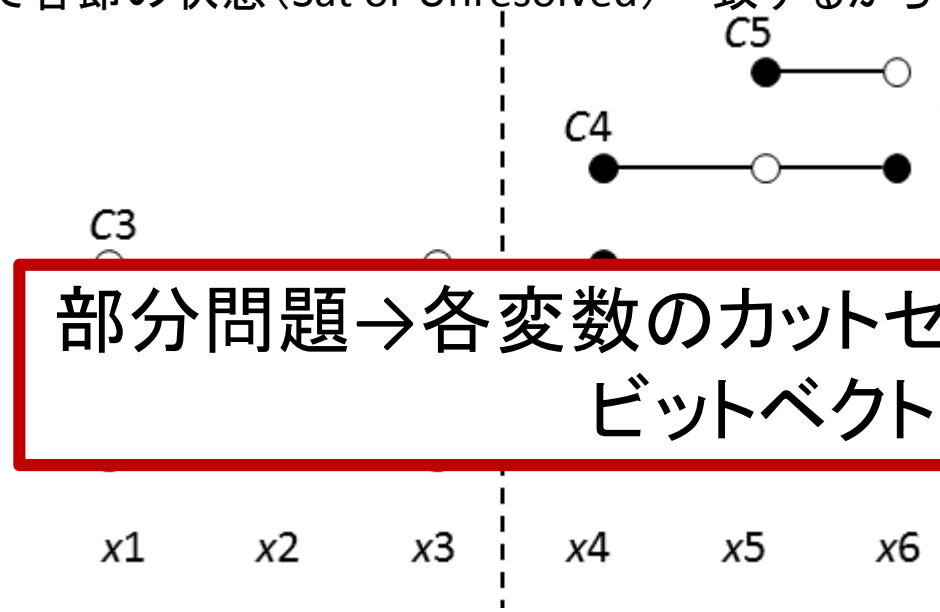
下線は節が充足されたことを意味する

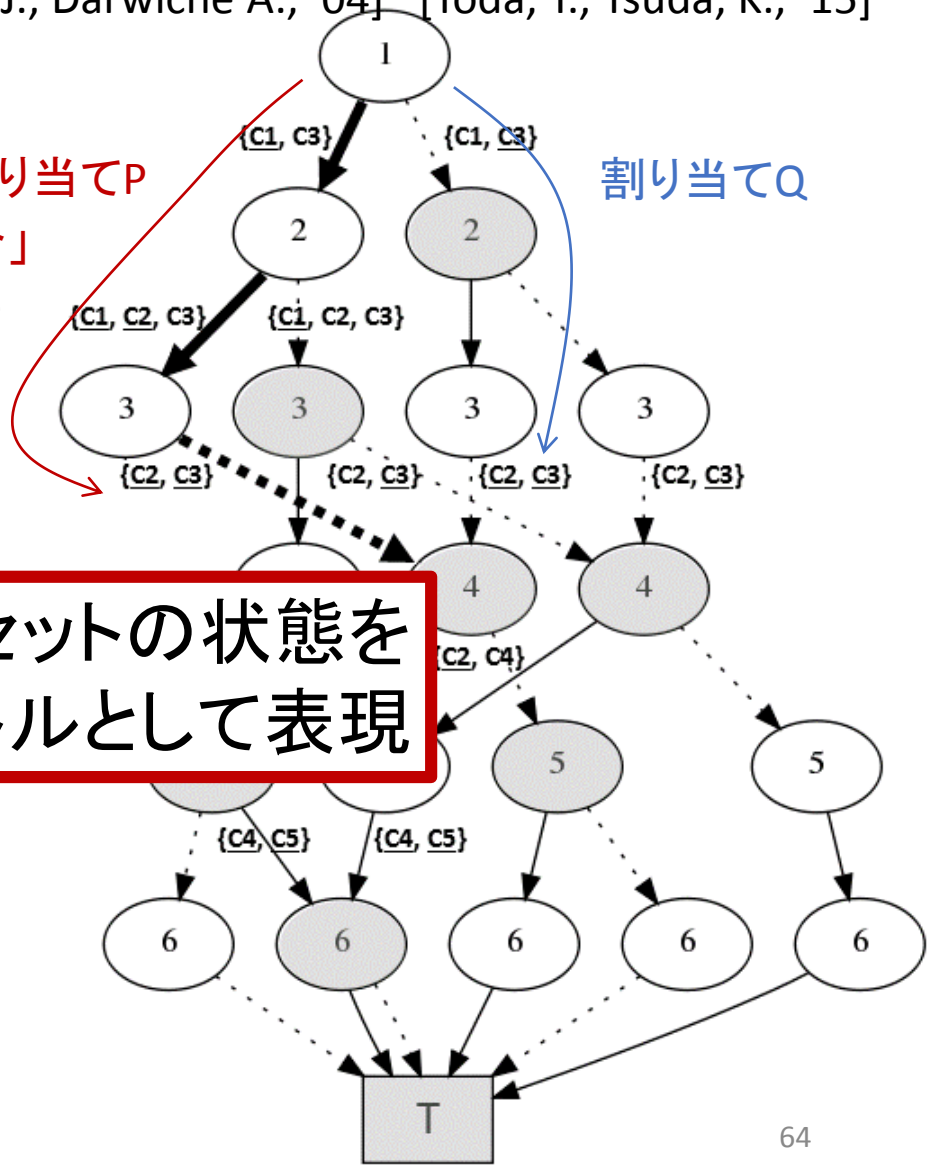
部分問題の符号化の考え方

X3までの割当てPとQは異なるが、
割当てをCNFに適用して得られる
論理式は論理関数として等価

[Huang, J., Darwiche A., '04] [Toda, T., Tsuda, K., '15]

「X3のカットセット＝下図の点線をまたぐ節集合」
で各節の状態（Sat or Unresolved）一致するから





**部分問題→各変数のカットセットの状態を
ビットベクトルとして表現**

区間によって表現されたCNF：
黒丸が正リテラル、白丸が負リテラル、
1つの区間が1つの節を表す。