



HOKKAIDO UNIVERSITY

Title	木と要素が対応する新しいZDD
Author(s)	宇野, 毅明
Description	ERATO湊離散構造処理系プロジェクト : 2010年度初冬のワークショップ (ERATO合宿) . 2010年11月29日 (月) ~12月1日 (水) . 札幌北広島クラッセホテル.
Relation	2010年度科学技術振興機構ERATO湊離散構造処理系プロジェクト講究録. p.454-455.
Issue Date	2011-06
Doc URL	https://hdl.handle.net/2115/48339
Type	conference presentation
File Information	17.uno.pdf



木と要素が対応する新しいZDD

宇野 毅明 (国立情報学研究所,
総合研究大学院大学)



2010年11月 ERATO合宿

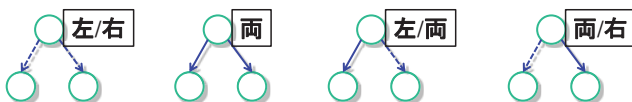


パスから木へ

- ZDDは、非閉路的グラフのパスと記憶したいオブジェクトが1体1対応するような構造(パスで覚えている)
- パスの代わりに木(あるいは部分グラフ)を使って物事を覚える
 - ➔ 根を持つ非閉路的グラフの、根を根とし、全ての頂点から0ノードか1ノードに到達可能なような部分グラフを考える(展開すると木になる)
- 簡単に言えば、ZDDは各内点(0,1ノードでないノード)で、右に行くか左に行くかを選んでいる。ここで、「両方選ぶ」という操作をアリにして、分岐を含めるようにする。

選択ラベル(choice label)

- 各頂点での子供の選び方は、今までのZDD・BDDでは「左か右か」を選ぶことのみ。ここにバリエーションを与える
- 具体的には「左か右」「左か両方」「右か両方」「左か右か両方」「必ず両方」の5通り。これを、各ノードがラベルとして覚える(各頂点での選択を記述しているラベルがついた)
- 実際のデータ構造は、ZDD/BDDの各頂点に、この選択ラベルがついたものになる。

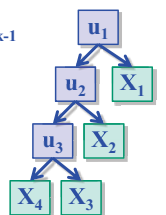


モデル化

- 具体的にある種の構造を記憶するときには、どのように選択ラベルを置くか、を決めるモデル化が必要
- + (k)パーティション
 - + (重複を許す)部分集合族
 - + グラフの部分(順序)木
 - + 木
 - + 部分非閉路グラフ

(k) パーティション

- 部分集合族を記憶する通常のZDDに、その組合せを表す屋根がついたモデル
- 入力は $E=\{v_1, \dots, v_n\}$ の分割の集合。大きさ k に限られててもいい
- ZDDの変数は、 $v_1, \dots, v_n$ に加え、ダミー変数 $u_1, \dots, u_n$
 - 変数の順序は、上の方から $u_1, \dots, u_n, v_1, \dots, v_n$
- 各パーティション $\{X_1, \dots, X_k\}$ に対して、各 $X_i \sim X_{k-1}$ を変数 $v_1, \dots, v_n$ を使って記憶。それらの先頭をダミー頂点 $u_1, \dots, u_{k-2}$ のはしごでつなぐ。
($X_1, \dots, X_k$ は最小要素でソートして添え字付け)
- v_i のラベルは**左右**、 u_i は**両**



■ つづく (次回をお楽しみに)

次回、衝撃の事実が明らかに!?

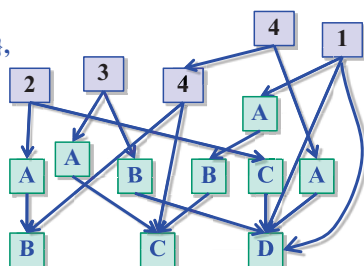
単なる文字列と何が違うか

- 文字列でパーティションを表現すると、「末尾が同じとき」しか共有されない
- 木でやると、「各分割の要素の末尾が同じとき」に共有される

例) $V = AUBUCUD$,

パーティション集合 =

$\{AUBUC, D\}, \{AUB, CUD\},$
 $\{AUC, BUD\}, \{AUD, B, C\}$

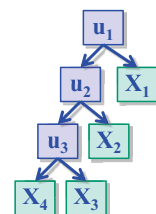


(重複可の)部分集合族の族

- パーティションとほとんど同じ
- 入力 $E = \{v_1, \dots, v_n\}$ の部分集合族の集合
- ZDDの変数は、 $v_1, \dots, v_n$ に加え、ダミー変数 $u_1, \dots, u_n$
 変数の順序は、上の方から $u_1, \dots, u_n, v_1, \dots, v_n$

- 各部分集合族 $\{X_1, \dots, X_k\}$ に対して、各 $X_1 \sim X_k$ を変数 $v_1, \dots, v_n$ を使って記憶。それらの先頭をダミー頂点 $u_1, \dots, u_{k-1}$ のはしごでつなぐ。
 ($X_1, \dots, X_k$ は辞書順でソートして添え字付け)

- v_i のラベルは**左右**、 u_i は**両**。右左の子供が同じである頂点がありうる



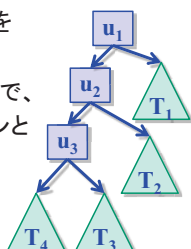
順序完全二分木

- 入力は順序完全二分木の集合。右の子・左の子が明示。グラフ $G=(V,E)$ 、 $V=\{v_1, \dots, v_n\}$ の部分順序完全二分木の集合でもよい。

- ZDDの変数はラベル集合 $(v_1, \dots, v_n)$ とダミー頂点。変数の順序はいつでもいい

- 基本的に、**両** 木ラベルを使って順序完全二分木を構築してZDD(BDD)にするだけ。
 各二分木の根をスーパールートにつなげるところで、**「右左」**ラベルのダミー頂点を使う。(パーティションと同じように)

- 木の順序は、「左優先探索のコード」でつける



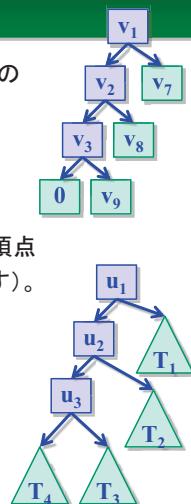
順序木

- 入力は順序木の集合。子供の数に制限がないので、子供が一人のこともある。

- ZDDの変数はラベル集合。変数の順序は適当

- 1つの頂点をはしごで表し、はしごの一番下の頂点の左の子供を0にする(これではしごの終点を表す)。葉は1ノード、あるいは右の子が1で左の子が0のノードで表す。

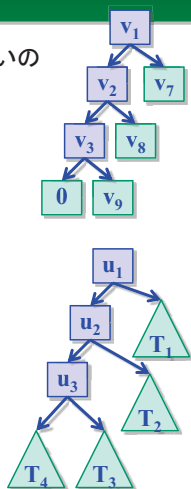
- 根のつなぎ方は先の方法と同じ
 木の順序は、「左重標準型のコード」でつける



無順序木

- 入力は無順序木の集合。子供の数に制限がないので、子供が一人のこともある。

- ZDDの変数はラベル集合。変数の順序は適当
- 無順序木を「左重標準型」に変換して記憶
 → 順序木のフォーマットでそのまま行ける



順序アサイクリックグラフ

- 入力は順序アサイクリックグラフの集合
 ← 各頂点には「子供の順序」が与えられている
- ZDDの変数はラベル集合 + ダミー頂点 + 「フォワード頂点」。変数の順序は適当

- まず、左の子供を優先して探索する深さ優先探索を行い、その木を記憶。残りの枝は、「フォワード頂点」の右の子にすることで表現

- 子供の順序を再帰的に付けることで、無順序なアサイクリックグラフも同様に格納

