解　説　論　文

BDD/ZDDを基盤とする離散構造と演算処理系の最近の展開

Recent Topics on Discrete Structures and Algebraic Operations Based on BDDs/ZDDs

湊　真一 Shin-ichi MINATO

アブストラクト　二分決定グラフ（BDD: Binary Decision Diagram）は，論理関数を効率良く表現するデータ構造の一種である．BDD に関する処理技法は主に VLSI 設計技術の分野で 1990 年代に発展したものであるが，近年ではデータマイニングや知識発見の分野でも効果的に活用されるようになってきている．中でも，ゼロサプレス型 BDD（ZDD: Zero-suppressed BDD）と呼ばれる BDD の変化形は，データベース解析の多くの問題で見られるような「疎な組合せの集合」を扱う場合に特に効果的である．本稿では，離散構造を処理するための基本的な技術として，BDD，ZDD をはじめとする種々の決定グラフの演算処理系を解説し，それらの効果的な応用について述べる．最後に，我々が現在進めている「離散構造処理系プロジェクト」の概要を紹介する．

キーワード　二分決定グラフ, BDD, ゼロサプレス型 BDD, ZDD, 離散構造, アルゴリズム

1. はじめに

二分決定グラフ（BDD: Binary Decision Diagram）[1]は，大規模論理関数を計算機の主記憶上に効率良く表現するデータ構造であり，1990 年ごろから，主に VLSI 設計自動化の分野で盛んに研究開発されてきた[2],[3]．近年，この BDD が，データマイニング・知識発見の分野においても活用できることが分かってきた．特に，ゼロサプレス型 BDD（ZDD: Zero-suppressed BDD）と呼ばれるタイプの BDD[4]が，現実的な多くのデータベースの処理に適することが明らかになり，実用的な技術として注目され始めている．このような論理関数や集合データを扱う問題を総合的に俯瞰して見ると，大規模な離散構造を高速に演算処理する技術が，様々な実用的問題を効率良く解くための鍵を握っている．

離散構造（discrete structure）とは，離散数学及び計算機科学の基礎をなすものであり，集合理論，記号論理，帰納的証明，グラフ理論，組合せ論，確率論などを含む数学的な構造の体系である．およそ計算機が扱うあらゆる問題は，単純な基本演算機能を要素とする離散構造として表現することができ，そのような離散構造の処理に帰着される．離散構造の処理は，最終的には膨大な個数の場合分け処理を必要とすることが多い．種々の離散構造データを計算機上にコンパクトに表現し，等価性・正当性の検証，モデルの解析，最適化などの処理を効率良く行う技法は，計算機科学の様々な応用分野に共通する基盤技術として非常に重要である．典型的な応用分野としては，ハードウェア/ソフトウェアの設計，大規模システムの故障解析，制約充足問題，データマイニングと知識発見，機械学習と自動分類，バイオインフォマティクス，Web 情報解析等，現代社会に対する大きな波及効果を持つ．BDD や ZDD は，そのような様々な離散構造の基礎となるデータ構造であると位置付けられる．

本稿では，離散構造を処理するための基盤技術として，BDD, ZDD をはじめとする種々の決定グラフの演算処理系を解説し，それらの効果的な応用について述べる．最後に，我々が最近開始した「離散構造処理系プロジェクト」(JST ERATO) の概要を紹介する．

2. BDD/ZDD による離散構造の表現と処理

2.1 BDD（二分決定グラフ）

BDD は，グラフ構造による論理関数の表現である．$F(a,b,c) = a\bar{b}c \vee \bar{a}b\bar{c}$ を表現した例を図 1 に示す．これは，論理関数の値をすべての変数について場合分けした結果を二分決定木で表し，これを縮約することにより得られる．このとき，場合分けする変数の順序を固定し，① 冗長な節点を削除する，② 等価な節点を共有する，という処理を可能な限り行うことにより「既約」な形が得られ(注1)，論理関数をコンパクトかつ一意に表せることが知られている[5]．更に，複数の論理関数を表す BDD の間においても，変数順序を固定すれば，互いにサブグラフを共

湊　真一　正員　北海道大学大学院情報科学研究科
E-mail　minato@ist.hokudai.ac.jp
Shin-ichi MINATO, Member (Graduate School of Information Science and Technology, Hokkaido University, Sapporo-shi, 060-0814 Japan).

電子情報通信学会　基礎・境界ソサイエティ
Fundamentals Review Vol.4 No.3 pp.224–230 2011 年 1 月

(注1)：一般に BDD という場合には，変数の順序が固定されないものや，既約でないグラフも対象に含まれるが，本稿では上記の簡約化処理を施したものだけを扱う．

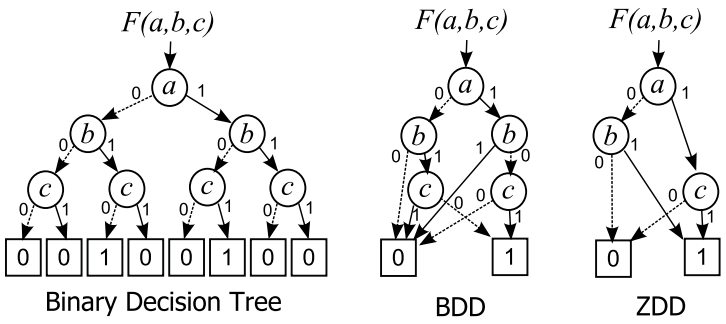

図 1　Binary Decision Tree（**二分決定木**）, BDD, ZDD

a	b	c	F	
0	0	0	0	$\rightarrow S$
0	0	1	0	
0	1	0	1	$\rightarrow b$
0	1	1	0	
1	0	0	0	
1	0	1	1	$\rightarrow ac$
1	1	0	0	
1	1	1	0	

論理関数とみた場合:
$F = a\bar{b}c \vee \bar{a}b\bar{c}$

組合せ集合とみた場合:
$S = \{ac,\ b\}$

図 2　**論理関数と組合せ集合の対応**.

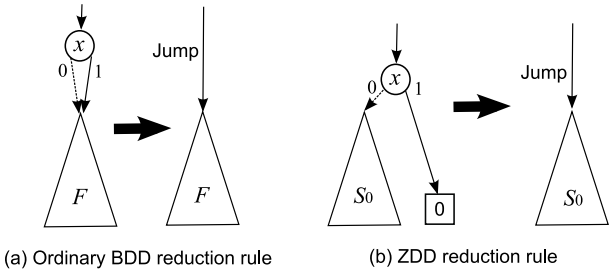

図 3　ZDD **の簡約化規則**.

有することが可能であり, 一つのメモリ空間の中で多数の論理関数データを扱うことができる.

BDD は, 多くの実用的な論理関数を比較的少ない記憶量で一意に表現することができる. また, 二つの BDD を入力とし, それらの 2 項論理演算 (AND,OR 等) の結果を表す BDD を直接生成するアルゴリズム[(1)]が考案されている. このアルゴリズムはハッシュテーブルを巧みに用いることで, データが計算機の主記憶に収まる限りは, データ量にほぼ比例する時間内で論理演算を効率良く実行できるという特長を持っている （詳細は文献 (2) を参照）.

BDD は, 基本的には計算機の主記憶上にすべてのデータを置いて処理する技法であり, 主記憶のランダムアクセス性を利用していることが本質的な点である. 近年, 安価な PC でもギガバイト級の主記憶を装備するようになり, 従来は解くことが不可能だった大規模な問題を扱えるようになってきた. このような背景から, 特に 2000 年代以降, BDD の応用範囲が広がりつつある.

2.2　ZDD（**ゼロサプレス型** BDD）

BDD は元々は論理関数を表現するために考案されたものだが, これを用いて「組合せ集合」を表現することもできる. 組合せ集合とは,「n 個のアイテムから任意個を選ぶ組合せ」を要素とする集合である. n 個のアイテムから任意個を選ぶ組合せは 2^n 通り存在するので, 組合せ集合としては 2^{2^n} 通りの取り方がある. 例えば, a, b, c, d, e という五つのアイテムに関して, $\{ab, e\}$, $\{abc, cde, bd, acde, e\}$, $\{\lambda, cd\}$, $\emptyset$ は, いずれも組合せ集合の一例である. (本稿では “λ” は空の組合せ要素を表し, “$\emptyset$” は空の集合を表すこととする.) 組合せ集合は, 組合せ問題の解集合を表現する基本的・はん用的なデータ構造であり, 実問題においては, 購入履歴データベース, Web のリンクの表現, システム故障要因の表現等, 様々な局面で現れる.

ある一つの組合せ集合は, 一つの論理関数（特性関数と呼ばれる）に対応付けることができる. 例えば, 図 2 は $a\bar{b}c \vee \bar{a}b\bar{c}$ という論理関数を表す真理値表であるが, 見方を変えると $\{ac,\ b\}$ という組合せ集合も表現している. 特性関数を BDD で表すことにより, 組合せ集合を非明示的に表現することができる. このとき, 論理関数の AND/OR 演算は, 集合の交わり (intersection)/結び (union) の演算にそのまま対応するので, 多数の要素を含む集合同士の演算を, BDD 処理系でまとめて実行することが可能となる. たとえ要素数が非常に多くても, 類似する組合せが多ければ, 部分的に共通する組合せがグラフ上で共有されて, 記憶量や計算時間が大幅に（時には指数関数的に）削減できる場合がある.

ZDD(ゼロサプレス型 BDD)[(4)]は, このような組合せ集合データの処理に特化された BDD である. ZDD では, 等価な節点を共有する規則は通常の BDD と同様であるが, 冗長な節点を削除する際の規則が異なる. すなわち, 図 3 に示すように, 1-枝が 0-終端節点を直接指している場合に, この節点を取り除く. その代わり, 通常の BDD で削除されるような節点は削除しない. このような ZDD の簡約化規則によっても表現の一意性は保たれる.

一般に, 組合せ集合に類似した要素（部分的組合せ）が多数含まれる場合, BDD では等価なサブグラフが多く出現し, それらが互いに共有されて, コンパクトに圧縮された表現となる. 更に, ゼロサプレス型の簡約化規則を用いると, 組合せ集合に無関係な

表 1　ZDD の基本演算

$\emptyset$	空集合．(0-定数節点を返す.)
$\{\lambda\}$	0 個のアイテムからなる組合せの集合 (1-定数節点を返す.)
$P.top$	組合せ集合 P の最上位の節点のアイテム番号を返す.
$P.offset(v)$	P の中でアイテム v を含まない組合せを集めた部分集合を取り出す.
$P.onset(v)$	P の中でアイテム v を含む組合せを集めた部分集合を取り出し，各組合せから v を取り除いた組合せ集合を返す.
$P.change(v)$	P の各組合せについて，アイテム v の有無を反転させる.
$P \cup Q$	P と Q の結び (union).
$P \cap Q$	P と Q の交わり (intersection).
$P \setminus Q$	差集合．(P にあって Q にないもの.)
$P.count$	P の要素数を数える.
(筆者[6]により導入された拡張演算)	
$P * Q$	P と Q の直積集合 (Cartesian product).
P/Q	P を Q で割った商 (quotient).
$P\%Q$	P を Q で割った剰余 (remainder).

（一度も出現しない）アイテムに関する節点が自動的に削除されることになり，通常の BDD よりも効率良く組合せ集合を表現・操作することができる．この簡約化規則は，特に疎な組合せの集合に対して顕著な効果がある．例えば，組合せ集合の各要素に含まれるアイテムの平均出現頻度が 1%であれば，ZDD は BDD よりも 100 倍コンパクトになる可能性がある．データマイニングで扱われる現実のデータベースでは，アイテムの平均出現頻度が非常に小さい場合が多い（例えば，店舗での陳列商品の総数に比べて，顧客一度に購入するアイテム数は極めて少ない）．したがって，ZDD による圧縮効果は極めて大きい．

更に ZDD 表現では，データ圧縮が全く効かない場合，すなわち等価な節点が存在せず共有が全く行われない場合でも，すべてのタプルを線形リスト表現で明示的に列挙表現したときの記憶量（のオーダ）を超えないという好ましい性質がある．

また，ZDD の利点はデータの圧縮だけでなく，ZDD 同士の様々な集合演算を圧縮されたデータ量にほぼ比例する計算時間で実行できるということが挙げられる．つまり，圧縮データを元に戻すことなく，圧縮したままで高速に演算処理できるという優れた特長がある．

ZDD 処理系の基本演算を表 1 に示す．このうち，$\emptyset$, $\{\lambda\}$, 及び $P.top$ は，ZDD のサイズにかかわらず定数時間で実行できる．$P.offset(v)$, $P.onset(v)$, 及び $P.change(v)$ の演算は，アイテム v の節点が ZDD の最上位にあれば定数時間で実行でき，そうでなければ，v より上位にある節点数に比例する計算時間となる．更に，二つの組合せ集合の結び，交わり，差集合の 2 項演算は，ZDD のサイズにほぼ比例する計算時間となる．ZDD を構築する場合，まず単一のアイテム変数からなる ZDD を作って置き，それらの 2 項集合演算を繰り返し実行することにより，より複雑な組合せ集合を表す ZDD を構築していく．

上記の表のうち，最後の三つの演算は組合せ集合に関する直積と除算であり，これは興味深い代数系[6]を構成する．Knuth は，このアイデアから触発されて，更に多くの演算（$P \sqcap Q$, $P \sqcup Q$, $P \boxplus Q$, $P \nearrow Q$, $P \searrow Q$, $P^{\uparrow}$, $P^{\downarrow}$, 等）を考案し，これらを総称して “Family Algebra” と呼んで，彼の有名な教科書 “The Art of Computer Programming” の最新の分冊[7]で詳しく論じている．

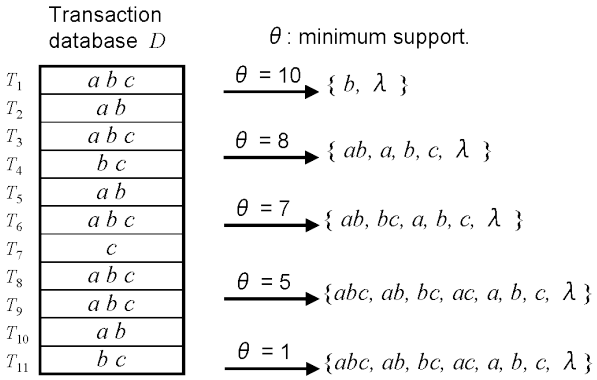

図 4　頻出アイテム集合マイニング.

3. ZDD を用いたデータベース解析

3.1 頻出アイテム集合マイニング

頻出アイテム集合マイニング (Frequent Itemset Mining)（または頻出パターンマイニングとも呼ばれる）とは，最も基本的なデータマイニングの問題であり，最小頻度 θ を与えたときに，データベース中に θ 回以上頻出するアイテムの部分集合 (パターン）を列挙せよ，という問題である．図 4 に簡単な例を示す．ここでデータベース D は全部で 11 個のレコードからなる．パターン b はそのうち 10 回出現しており，パターン a, c, 及び ac は 8 回，bc は 7 回，ac と abc は 5 回出現している．したがって，もし $\theta = 7$ を与えたとすると，頻出パターンの集合は $\{ab, bc, a, b, c, \lambda\}$ となる．最小頻度のしきい値 θ が小さいほど，多くのパターンが頻出となる．$\theta = 1$ を与えた場合，すべての部分集合が頻出となるため，頻出パターンの総数は，元のデータベースの 1 レコードに含まれる最大のアイテム数に対して指数関数的な個数となる．

頻出アイテム集合マイニングに関しては，Agrawal ほか[8]による Apriori アルゴリズムの研究に始まり，現在までに様々なアルゴリズムが提案されている[9],[10]．近年，Minato ほかは，

Itemset	Freq.
a b c	5
a b	3
b c	2
c	1

Itemset-histogram for DB records

Itemset	Freq.
a b c	5
a b	8
a c	5
b c	7
a	8
b	10
c	8
λ	11

Itemset-histogram for sub-patterns

図 5　組合せ頻度表の例.

itemset	frequency	F_2	F_1	F_0
abc	5 (101)	1	0	1
ab	3 (011)	0	1	1
bc	2 (010)	0	1	0
c	1 (001)	0	0	1

F_0 = {abc, ab, c}
F_1 = {ab, bc}, F_2 = {abc}

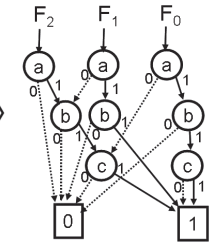

図 6　組合せ頻度表の ZDD ベクトル表現.

"LCM over ZDDs"[11]と呼ぶ高速なアルゴリズムを開発した．これは，2003 年に宇野が開発した世界最高速の頻出アイテム集合マイニング法「LCM」[12]に，ZDD 処理技術を組み合わせて，膨大な頻出パターン集合を表す ZDD をメモリ上に高速に生成し，ZDD 同士の演算により，頻出パターンの様々な解析処理を行えるようにしたものである．

3.2　組合せ頻度表と ZDD ベクトル表現

頻出アイテム集合マイニングの例でも見られるとおり，データマイニングの応用ではデータベース中のアイテム組合せの出現頻度を数えることがしばしば重要となる．「組合せ頻度表」(itemset-histogram) はそのような用途に使われるデータ表現である．図 5 に例を示す．左側は，図 4 のデータベースの各レコードのアイテム組合せの頻度表であり，右側は各レコードに出現する部分的パターンをすべて抽出して頻度表にしたものである．完全な頻度表を作ろうとするとデータ量が非常に大きくなる場合があり，効率の良いデータ構造を考えることは実用上重要である．

ZDD は組合せ集合の表現であるため，各組合せ要素が集合に含まれるかどうかだけを識別し，個数を数えることができない．ZDD を用いて出現頻度を保持するために，我々は整数値を m ビットの 2 進数に符号化して $(2^m - 1)$ までの頻度を表すこととし，図 6 のように 2 進数の各けたを表す ZDD: $\{F_0, F_1, \ldots, F_{m-1}\}$ を並べた「ZDD ベクトル表現」[13),(14)]を開発した．F_0 は 2 進数の最下位ビットに相当し，出現頻度が奇数である組合せを集めた集合となる．F_1 は下から 2 ビット目が 1 となる組合せを集めた集合で，同様に F_{m-1} までの ZDD を並べることで整数値を持つ組合せ集合を表現できる．図 6 の例では，組合せ頻度表は，$F_0 = \{abc, ab, c\}$, $F_1 = \{ab, bc\}$, $F_2 = \{abc\}$, と分解でき，それぞれのけたは単純な ZDD で表現できる．更にそれらの ZDD は互いに節点を共有して表現できる．

組合せ頻度表を ZDD ベクトルで表す場合，2 進数のけた数（ZDD の個数）は出現頻度の最大値の log に比例する．また，各けたの ZDD の節点数は，すべてのアイテム組合せの出現頻度の総和を超えないことが容易に示せる．もし，部分的に類似する組合せが多く含まれる場合は，ZDD のサブグラフが共有されてコンパクトな表現が得られる．

3.3　組合せ頻度表の代数系

ZDD ベクトル表現で組合せ頻度表を表すと，様々な演算を効率良く行うことができる．表 2 に組合せ頻度表の基本的な演算を示す．これらの演算は，幾つかの ZDD の演算を組み合せた手順で実現できる．計算時間はおおよそ ZDD の総節点数に比例する．

表 2　組合せ頻度表の基本演算

c $(c \in N)$	λ だけが c 回出現する頻度表を返す．($c = 0$ のときは空集合を表す.)
$P.top$	頻度表 P の最上位の節点のアイテム番号を返す.
$P.offset(v)$	P の中でアイテム v を含まない組合せを集めた部分頻度表を取り出す.
$P.onset(v)$	P の中でアイテム v を含む組合せを集めた部分頻度表を取り出し，その各組合せから v を取り除いた頻度表を返す.
$P.change(v)$	P の各組合せについて，アイテム v の有無を反転させる.
$P + Q$	P と Q の対応する組合せ同士の頻度を算術加算した頻度表を返す.
$P - Q$	P と Q の対応する組合せ同士の頻度を引き算した頻度表を返す.（ただし減算結果が負になる場合は 0 とする.）
$P.count$	P の組合せの種類数を数える.
$P.upperbound$	P の組合せのうち最大頻度の数値を返す.
$P.unitset$	P に含まれる組合せの頻度をすべて 1 に正規化した頻度表を返す.

$max(P, Q)$	$P + (Q - P)$ を返す.
$min(P, Q)$	$P - (P - Q)$ を返す.
$P * Q$	P と Q の直積集合 (Cartesian product).
P/Q	P を Q で割った商 (quotient).
$P\%Q$	P を Q で割った剰余 (remainder).

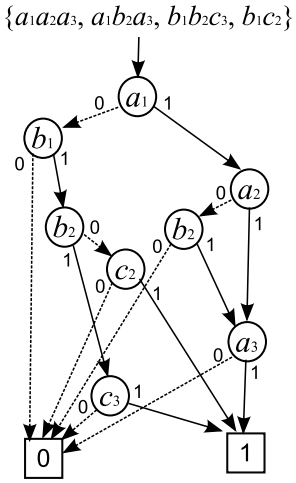

図 7　アイテム符号化により ZDD で系列集合を表した例

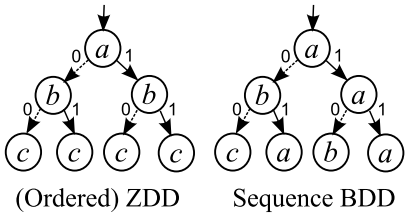

図 8　Sequence BDD の変数順序の規則

これらの演算を組み合わせることにより，頻出パターンマイニングの後処理過程として，様々なデータ解析を行うことが可能となる[15]．例えば，

- 頻出パターンの中から特定の部分パターンを見つけ出す．
- アイテム数の多いパターン/少ないパターンだけを取り出す．
- 二つのデータベースの頻出パターン同士の比較（共通部分，差分等）を行う．
- 統計的な数値を計算する（support, confidence 等）

等が挙げられる．

ところで，表 1 と表 2 を比べると，両者が非常に類似していることが分かる．組合せ頻度表は，組合せの多重集合を表現しているともいえるので，組合せ集合の自然な拡張となっていると考えてよい．組合せの多重集合だとすると，出現頻度の数値を非負整数に限定しているのは自然なことであるが，これを負数も扱えるように更に拡張することもできる．我々はこのようなモデルを「重み付き積和集合」(Valued Sum-Of-Products; VSOP)[16],[17]と名付け，C-Shell 風のスクリプトで代数演算を表現する文法を定義し，これを ZDD で効率良く計算するインタプリタを開発している(注2)．

(注2)：このソフトウェアは公開している．

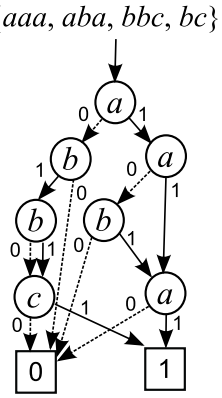

図 9　Sequence BDD の例

4. Sequence BDD を用いた系列集合の処理

4.1 系列集合の表現

系列集合（set of sequences または set of strings）は，テキスト文書，遺伝子情報，時系列イベント等，様々なデータを表現可能な基盤的な離散構造の一つである．計算理論の分野では言語（language）とも称される．使用する文字の集合（アルファベット）$\Sigma = \{a, b, c\}$ のとき，$\{aaa, aba, bbc, bc\}$ は系列集合の一例である．なお，ここでは有限長の系列集合のみを扱うものとする．

通常の ZDD は組合せ集合を表すので，文字の出現順序の違い（例えば $\{ab, ba\}$）や，文字の重複（例えば $\{aa, aaa, ab, aabb\}$）は区別できない．簡単に思いつく方法としては，文字の種類と出現位置の二つを組にして符号化したものを ZDD のアイテム変数に割り当てれば，系列集合を ZDD で表現できる[18]．例えば，系列集合 $\{aaa, aba, bbc, bc\}$ は，$\{a_1a_2a_3, a_1a_2a_3, b_1b_2c_3, b_1c_2\}$ のように符号化できる．図 7 に，このような符号化を用いて構築した ZDD を示す．ここでは a_1, a_2, a_3 は同じ文字でも違う出現位置にあるため，異なるアイテム変数とみなして，組合せ集合により系列集合を表現している．

この方法を使えば，系列集合を ZDD の演算処理系により操作することが可能となる．しかし，系列の最大長とアルファベットサイズの積に比例してアイテム変数の数が非常に多くなることや，出現位置が少しでも異なると同じ部分系列でも共有できないという欠点があり，残念ながら余り強力な手法とはいえない．

4.2 Sequence BDD

最近，Sequence BDD (SeqBDD) と呼ばれる新しい系列集合の表現方法が，Loekito ら[19]により提案された．これは ZDD とほとんど同じデータ構造であるが，ただ 1 点だけ，変数順序付けの規則が異なる．通常の ZDD では，すべての親節点と子節点の間で，あらかじめ固定された変数順序が守られているのに対し，SeqBDD では，0-枝側のみ変数順序に従うが，1-枝側は変数順序の制約を外しており，記号の重複や逆順が許される．

表 3　Sequence BDD の基本演算

$\emptyset$	空集合．(0-定数節点を返す.)
$\{\lambda\}$	空列のみからなる集合．(1-定数節点を返す.)
$P.top$	P の最上位の節点の文字 ID を返す.
$P.onset(x)$	文字 x で始まる系列の部分集合を抽出し，各系列の先頭の x を取り除く.
$P.offset(x)$	文字 x 以外で始まる系列の部分集合を抽出する.
$P.push(x)$	P の各系列の先頭に文字 x を付加する.
$P \cup Q$	P と Q の結び (union).
$P \cap Q$	P と Q の交わり (intersection).
$P \setminus Q$	差集合．(P にあって Q にないもの.)
$P.count$	P の要素数を数える.
$P * Q$	P と Q の直積集合 (Cartesian product). P と Q から任意の一つの系列を取り出し連結した系列の集合.

図 8 に，SeqBDD の変数順序の例を示す．このように規則を半分だけ緩めたことにより，SeqBDD 上の根節点から 1-終端節点への経路上に，同じ文字が複数回出現することができる．図 7 と同じ系列集合を表す SeqBDD の例を図 9 に示す．SeqBDD では，最上位の節点は系列の先頭文字に対応しており，その変数番号が v であるとすると，先頭が v で始まる系列と v 以外で始まる系列に場合分けをしていることになる．1-枝側に進むと次の位置の文字で同様に場合分けを行う．最終的に 1-終端節点に到達した場合は，そのような系列が集合に含まれるという意味になる．

表 3 は SeqBDD の基本演算をまとめたものである．ZDD の代数系と非常に類似していることが分かる．*onset, offset* 及び *push* の各演算は通常の ZDD と少しだけ異なっているが，そのほかの 2 項演算（union, intersection, difference 等）は，ほとんど同じアルゴリズムで実装できる．もう一つ興味深いのは，SeqBDD では，左右非対称なゼロサプレス型の簡約化規則が不可欠であるということである．SeqBDD は変数順序が左右非対称なので，通常の BDD の左右対称の簡約化規則では不都合が起こってしまう．(そういう意味では SeqZDD と称した方が適切と思われる.) a SeqBDD は出現位置による符号化を必要とせず，系列集合を直接表現できる．有限系列の集合しか表現できないという制限はあるが，あらかじめ系列長の上限を決める必要はなく，SeqBDD が主記憶容量に納まるかどうかだけが制約事項となる．SeqBDD は，非常に長い系列と短い系列が混在するような系列集合を表現する場合に特に効果的である．伝住ら[20]は，SeqBDD 表現を用いて接尾辞木 (Suffix-tree) を構築する「Suffix-DD」と呼ぶ技法を考案し，現在開発を進めている．Suffix-DD により，テキスト文書のすべての部分文字列の索引集合を効率良く表現し，更に複数の索引集合同士の様々な集合演算ができるようになると期待されている．

5. 離散構造処理系プロジェクト

以上のように，二分決定グラフと離散構造の演算処理に関して様々な興味深い性質が見られることから，離散構造を統合的に扱う基本処理系として BDD/ZDD を位置付け，分野横断的な応用を持つ技術体系として研究開発を進めたいと考えている．幸いなことに，この構想が独立行政法人科学技術推進機構 (JST)

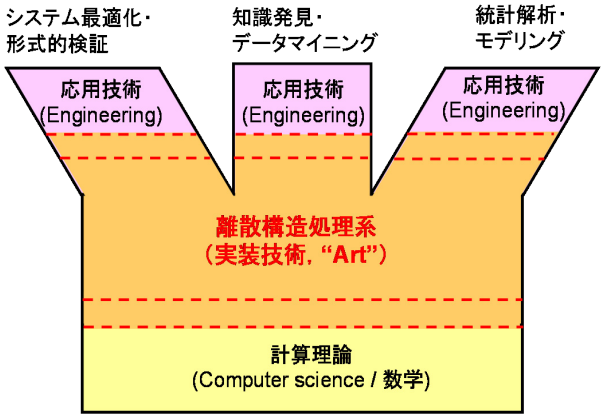

図 10　ERATO プロジェクトが対象とする技術領域

により「ERATO 湊離散構造処理系プロジェクト」として 2009 年 10 月に採択され，2010 年 4 月から 5 年間の計画で本格的な研究活動が開始されている．

本研究構想が対象とする技術領域を図 10 に示す．計算機科学の基礎部分には分野横断的な計算理論の領域がある．一方，実問題を解くために様々な工学的応用に特化した技術領域が多数並列に存在している．その中間層として，本研究が扱う離散構造処理系の技術領域が存在する．この層は，上下の層との境界は明確ではないが，研究の指向性が異なる．概念的・理論的成果だけではなく，実用に耐えるアルゴリズムを実装することを重視する．しかし，各応用分野の固有の問題にアドホックに対応するのではなく，技術基盤としての簡潔さやはん用性を重視する．この指向性こそが Science と Engineering をつなぐ，いわゆる「Art」であると我々は考えている．

本研究構想では，BDD/ZDD 技術を，離散構造を統合的に扱うための基本処理系として位置付け，以下の 2 点を目指している．

(1)　BDD/ZDD 研究を新しい視点でとらえ直し，はん用的な離散構造処理系として再構築を行う．社会的重要性が高い複数の応用に適用し，従来技法に比べて，10〜100 倍以上の性能向上を達成する．

(2)　トップデータを論文に書いて終わりにするのではなく，それをきちんと実装し，長期にわたって活用される基盤ソフトウェアを提供する．更に，それを使いこなせる人材を育成し，産業界に還元する．

本プロジェクトの技術面のポイントは，BDD 風の考え方（膨大なデータを圧縮したままで効率良く計算する）を発展させた新しいデータ構造を追及するところにある．しかもそのデータ構造は，処理操作のための体系的な演算 (algebra; 代数系) を伴っていることが重要である．BDD/ZDD は，論理関数や組合せ集合の表現であるが，本プロジェクトでは，より高次の離散構造，例えば，ビット列集合や，文字列 (sequence) 集合，順列 (permutation) 集合，分割 (partition) 集合，DAG, Network 等々，様々な離散構造の代数系と，その演算処理ソフトウェアの開発を目指している．

6. おわりに

本稿では，決定グラフに基づくデータ構造とその演算の代数系の例として，BDD による論理関数処理系, ZDD による組合せ集合の処理系，ZDD ベクトル表現による組合せ頻度表の演算処理系，及び Sequence BDD による系列集合の演算処理系について述べた．これらは，共通する代数構造を有しており，互いに強く関係し合っている．それぞれの代数系は興味深い性質を持ち，多くの実用的な応用に対して有用であると期待される．

計算機科学の多くの応用分野で，離散構造の処理技法が，様々な実用的問題を効率良く解くための鍵となる．そこで我々は，離散構造を統合的に扱う基本処理系として BDD/ZDD を位置付け，分野横断的な応用を持つ技術体系として再構築することを目指して ERATO プロジェクトを開始している．大規模離散構造の演算処理アルゴリズム（いわゆる「Art 層」）に関心を持つ優秀な研究者を集めて組織化し，研究開発を加速したいと考えている．関係各位の御理解・御協力をお願いする次第である．

文　献

(1) R. E. Bryant, “Graph-based algorithms for Boolean function manipulation,” IEEE Trans. Comput., vol.C-35, no.8, pp.677–691, 1986.
(2) 藤田昌宏, 佐藤政生, “BDD（二分決定グラフ）,” 情報処理, vol.34, no.5, pp.584–630, 1993.
(3) S. Minato, Binary decision diagrams and applications for VLSI CAD, Kluwer Academic Publishers, 1996.
(4) S. Minato, “Zero-suppressed BDDs for set manipulation in combinatorial problems,” In Proc. 30th ACM/IEEE Design Automation Conference, pp. 272–277, 1993.
(5) S. B. Akers, “Binary decision diagrams,” IEEE Trans. Comput., vol.C-27, no.6, pp.509–516, 1978.
(6) S. Minato, “Zero-suppressed BDDs and their applications,” Int. J. Softw. Tools Technol. Transf. (STTT), vol.3, no.2, pp.156–170, Springer 2001.
(7) D. E. Knuth, The Art of Computer Programming: Bitwise Tricks & Techniques; Binary Decision Diagrams, vol. 4, fascicle 1, Addison-Wesley, 2009.
(8) R. Agrawal, T. Imielinski, and A. N. Swami, “Mining association rules between sets of items in large databases,” in Proc. 1993 ACM SIGMOD International Conference on Management of Data, P. Buneman and S. Jajodia, ed., vol.22, no.2, of SIGMOD Record, pp.207–216, 1993.
(9) B. Goethals, “Survey on frequent pattern mining,” 2003. http://www.cs.helsinki.fi/u/goethals/publications/survey.ps.
(10) M. J. Zaki, “Scalable algorithms for association mining,” IEEE Trans. Knowl. Data Eng., vol. 12, no. 2, pp. 372–390, 2000.
(11) S. Minato, T. Uno, and H. Arimura, “LCM over ZBDDs: Fast generation of very large-scale frequent itemsets using a compact graph-based representation,” In Proc. 12-th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2008), (LNAI 5012, Springer), pp. 234–246, 2008.
(12) T. Uno, Y. Uchida, T. Asai, and H. Arimura, “LCM: an efficient algorithm for enumerating frequent closed item sets,” in Proc. Workshop on Frequent Itemset Mining Implementations (FIMI’03), 2003. http://fimi.cs.helsinki.fi/src/.
(13) S. Minato and H. Arimura, “Efficient combinatorial item set analysis based on zero-suppressed BDDs,” In Proc. IEEE/IEICE/IPSJ International Workshop on Challenges in Web Information Retrieval and Integration (WIRI-2005), pp. 3–10, 2005.
(14) 湊　真一, 有村博紀, “ゼロサプレス型二分決定グラフを用いたトランザクションデータベースの効率的解析手法,” 信学論 (D), vol. J89-D, no. 2, pp. 172–182, Feb. 2006.
(15) S. Minato and H. Arimura, “Frequent pattern mining and knowledge indexing based on zero-suppressed BDDs,” In Proc. 5th International Workshop on Knowledge Discovery in Inductive Databases (KDID’06), pp.83–94, 2006.
(16) 湊　真一, “VSOP:ゼロサプレス型 BDB に基づく「重み付き積和集合」計算プログラム,” 信学技報, COMP2005-5, pp. 31–38, May 2005.
(17) S. Minato, “VSOP (Valued-Sum-Of-Products) calculator for knowledge processing based on zero-suppressed BDDs,” In Federation over the Web, K. P. Jantke, et al. ed., LNAI 3847, pp.40–58, 2006.
(18) R. Kurai, S. Minato, and T. Zeugmann, “N-gram analysis based on zero-suppressed BDDs,” In New Frontiers in Artificial Intelligence, Joint JSAI 2006 Workshop Post-Proceedings, T. Washio, et al. ed., LNAI 4384, pp. 289–300, 2006.
(19) E. Loekito, J. Bailey, and J. Pei, “A binary decision diagram based approach for mining frequent subsequences,” Knowl. Inf. Syst., (DOI:10.1007/s10115-009-0252-9), 2009.
(20) S. Denzumi, H. Arimura, and S. Minato, “Substring indices based on sequence BDDs,” Hokkaido University, Division of Computer Science, TCS Technical Reports, TCS-TR-A-10-42, 2010.

（VLD 研究会提案，平成 22 年 11 月 1 日受付）

湊　真一（正員）

1988 京大・工・情報卒，1990 同大学院修士，1995 博士（社会人）了．博士（工学）．1990 日本電信電話株式会社入社．NTT 研究所にて大規模論理データ処理アルゴリズムの研究に従事．2004 北大大学院情報科学研究科助教授．2010-10 から同教授．2009 から JST ERATO 湊離散構造処理系プロジェクト研究総括（兼務）．BDD(二分決定グラフ) を用いた離散構造の処理に興味を持つ．著書 “Binary Decision Diagrams and Applications for VLSI CAD” (Kluwer, 1995)．情報処理学会, 人工知能学会, IEEE 各会員．