IntelligentPad における部分構造検索

赤石美奈† 田中 譲†

IntelligentPad システムは，マルチ・メディア文書に加えて，アプリケーション・プログラムや，システムが提供する各種サービス・システムをも，すべて紙の表現をもつダイナミック・メディア・オブジェクトとして統一的に扱う．ユーザは，パッドを「貼る」というメタファにより，さまざまな部品を自由に組み合わせ，複雑な機能を容易に実現できる．これに伴い，さまざまなパッドが急速に増加し，それらを統合的に管理・検索できる機能が必要となった．IntelligentPad では，インスタンス（パッド）の合成により，新しいインスタンス（パッド）を定義する．これに対して，既存のオブジェクト指向データベース（OODB）は，クラスの定義に基づいてデータを管理する．また，ユーザが必要とする部品や合成パッドを検索するためには，合成パッドを構成する部品の種類やその貼り合わせ構造の一部を条件として指定することにより検索を行うことが有効である．しかし，この検索は，OODB のインデックス機能を用いても十分なパフォーマンスを得ることができない．このため，パッドの統合管理と検索を行うための新たなデータベース・システムが必要である．その開発のための基礎研究として，本論文は，パッドの部分構造を検索条件とした検索法とその高速化手法を提案し，詳細な性能評価を行っている．

An Efficient Search Method for the Context Queries in the IntelligentPad System

MINA AKAISHI† and YUZURU TANAKA†

The IntelligentPad System provides its users with a toolkit for the construction of various intaractive media objects including multimedia documents, desktop tools, and application systems. The wide coverage of this toolkit and the ease of object construction allow its user or the society of its users sharing its resouces to rapidly increase the accumulation of composed media objects, which requires their database managment. We first thought that these media objects can be efficiently managed by an object-oriented database (OODB) system, and found that this is not the case for two reasons. The current OODBs use class definitions as database schemes, while in our system new media objects are mainly defined by combining existing object instances, and hence have no corresponding class definitions. Besides, most queries in our system partially specify composition structures of the target object instances. Such queries called context queries show seriously poor performance when processed by any current OODB system. This paper gives an efficient search method for the processing of context queries in OODBs. This method will allow us to develop a new type of OODBs that can deal with a large amount of arbitrarily composed object instances.

1. はじめに

現在のコンピュータは，従来の文字や数値データに加え，テキスト，図表，静止画，動画，音声等のマルチメディアを扱うことができる．これらの多様なデータを管理するためのデータベースとして，オブジェクト指向データベース（OODB）[1),2)]が注目されている．

オブジェクト指向のプログラミングにおいては，クラスの定義と，インスタンスの合成という，二つのプログラミング手法が提供される．現在の OODB は，前者に対しては，クラスごとにデータを管理するための機能を提供している．しかし，後者に対しては，何も考慮されていない．そのため，部品の合成によるプログラミングには対応できないと思われる．部品の合成・分解・再利用などによって，さまざまな機能を実現するシステムにおいては，完成品，半製品，基本部品など，さまざまなものが混在する．著者らは，これらのジャンク・オブジェクト[3)]の検索には，“使用されている部品の種類と，その結合関係” が検索において有効な検索条件となり得ると考えた[4)]．本論文では，部品の部分構造による検索法の有効性について説明し，その検索を高速化するためにシグニチャ・ファイルを利用することを提案する．シグニチャ・ファイル

† 北海道大学工学部電気工学科
Electrical Engineering Department, Faculty of Engineering, Hokkaido University

は，テキスト検索の高速化に利用されており，word signature (WS) と superimposed cording (SC) というコード化の違いによる比較[5]や，WS, SC, Bit-block Compression (BC), Run Length Encoding (RL) というコード化の違いによる比較[6]が解析的になされている．本論で対象とするシステムの“使用されている部品の種類と，その結合関係”は，木構造で表現される．構造体（木構造）の検索にシグニチャ・ファイルを利用するものとしては，Structured Superimposed Code word (SSCW)[7),8)]が提案されている．SSCWは，各ノードの根からの深さや，同一ノードを親ノードとしてもつノード間の順序をシグニチャに反映させている．これに対して，本論で対象とするシステムでは，各部品は，木構造のあらゆる位置に現れ，合成された部品は，直接結び付いている部品と連携動作する．このため，本論で提案するシグニチャは，木構造における位置にかかわらず，各部品の種類を反映し，また，直結している部品間の関係を反映するものとなっている．

2章では，部品の合成によりさまざまな機能を実現する IntelligentPad[9),10)]システムについて簡単に述べ，3章では，その部分構造検索について述べる．4章では，この検索の高速化について述べる．

2. IntelligentPad システムの概要

IntelligentPad システムでは，文章，図表，静止画，動画，音声等のコンピュータ上のさまざまなメディアに加えて，アプリケーション・プログラムや，システムが提供する各種サービス・システムをも，すべて紙（パッド）として表現し，これらをダイナミック・メディア・オブジェクトとして統一的に扱う．それぞれに対応して各種のパッドが定義される．システムにより提供される基本的な部品をプリミティブ・パッドと呼ぶ．パッドは，「貼る」というメタファにより機能の合成ができる．合成されたパッドは，これを構成するプリミティブ・パッドの機能連携により，複雑な機能をもつ一枚のパッドとして定義される．さまざまなパッドを自由に組み合わせることができ，すべてのパッドは，新たなパッドを構築するための部品として再利用できる．新たなアプリケーションの開発は，すべてを新たに開発するのではなく，それまでのシステムに欠けていたプリミティブな機能のみをパッドとして実現し，これと既存のパッドを合成することにより遂行できる．

3. パッドベース

パッドの開発が進むにつれ，パッドの種類や数は増加していく．それに伴い，さまざまなパッドを統一的に管理するためのシステムが必要となる．パッドベースは，あらゆる種類のパッドを統合的に蓄積管理し，ユーザが必要とするパッドを自在に検索・更新する機能を提供するそとを目指しているデータベース・システムである．パッドベースにおいては，ユーザが必要とする機能（パッド）をどのように検索するかということが大きな問題となる．合成パッドの機能は，使われている部品の種類と，それらがどのように機能連携するかによって決定される．そこで，筆者らは，パッドを検索する際には，「合成パッドを構成しているパッドの種類と結合関係」を検索条件とした検索が有効であると考えた．パッドの貼り合わせ構造を条件とする検索を用いることで，使用されている部品の一部を指定し，必要なアプリケーションを得ることができる．以下では，パッドの貼り合わせ関係による検索について述べる．

3.1 貼り合わせ構造検索

図1に示すように，電卓パッドは，計算能力をもつパッドに，ボタンパッドとディジタル・ディスプレイ・パッドを貼ることで実現される合成パッドである．このようなパッドを多数のパッドの中から検索する方法は，いくつか考えられる．ひとつは，ブラウジングと呼ばれる方法で，データベース中のパッドをユーザがひとつずつ見て捜す方法である．これは，データベース中のデータが大量である場合には，現実的な方法ではない．また，名前などの属性値を条件として検索を行う方法もあるが，これを用いるには，どのような名前で各パッドが登録されているかをあらかじめ知っている必要がある．このような場合，パッドの貼り合わせの部分構造を指定し，それを条件として検索することが有効であると考えられる．電卓パッドを探したいとしよう．電卓パッドは，少なくとも10個の数字入力用のボタンパッドが何らかの台紙パッドの上に貼られているという構造を含んでいるはずである．このような貼り合わせ構造の部分構造を検索の条件として用いることができる．

図2にパッドベースの画面ハードコピーを示す．台紙となっているパッドは，パッドベースパッドというプリミティブパッドであり，Smalltalk[11)]と GemStone[12),13)]のインタフェースの役割を果たす

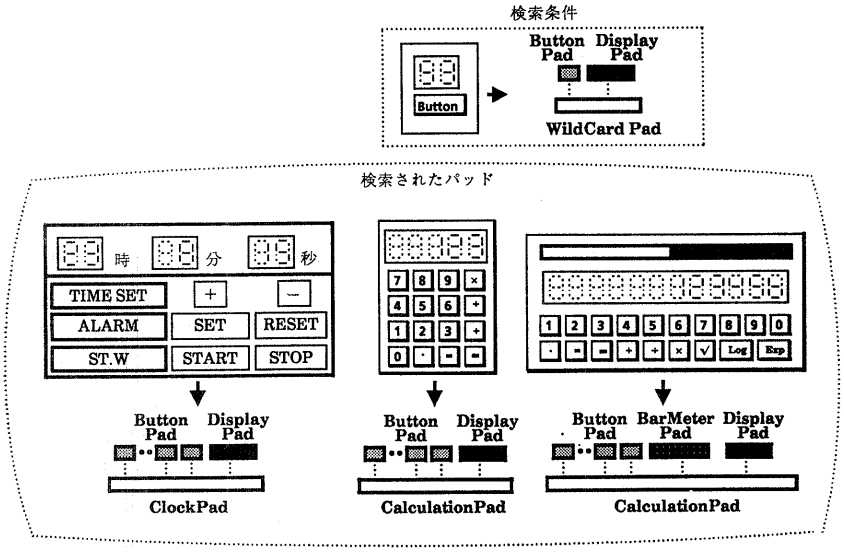

図 1 パッドの貼り合わせ構造による検索
Fig. 1 Search for the pads with the specified composition structure as their substructures.

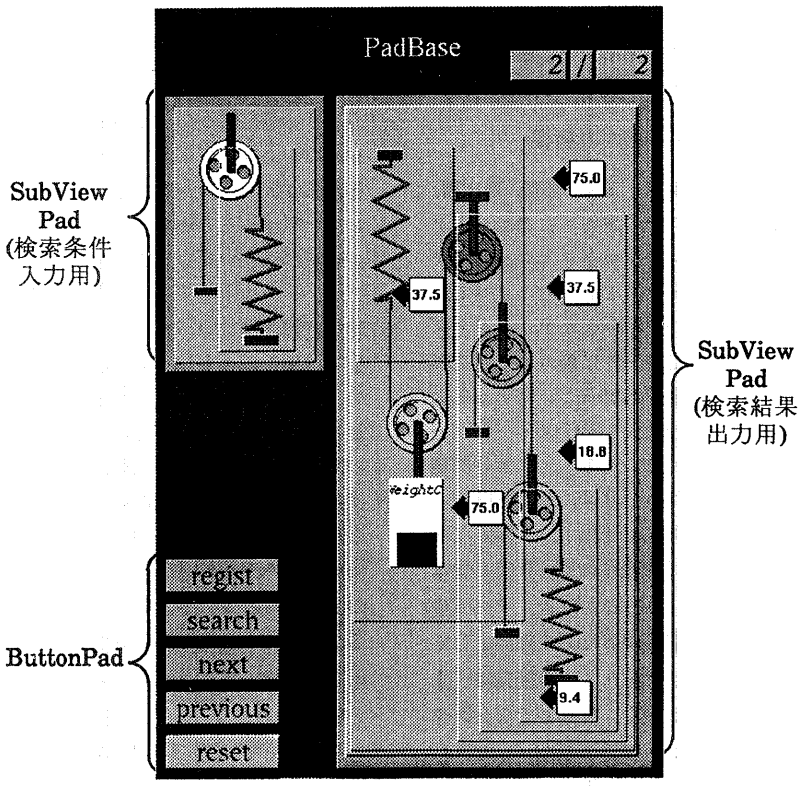

図 2 パッドベースの画面ハードコピー
Fig. 2 Display hardcopy of the PadBase.

(図2で示している IntelligentPad は，Smalltalk によって構築され，これを GemStone で管理している)．この上に貼られているサブビューパッドというプリミティブパッドは，自分の上に貼られたパッドをデータとして保持する機能をもつ．左上のサブビューパッドは，パッド間のリンクにより，その上に貼られたパッドをデータとして，台紙のパッドに送る．右側のサブビューパッドは，検索の結果を台紙から受け取り，それを自分の上にパッドとして貼る．検索結果のパッドは，そのままパッドとして利用できる．台紙には登録や，検索を行うためのボタンパッドが貼られている．これらのボタンが押されると，パッドベースパッドは，GemStone を起動し，登録や検索を行いその結果を返す．図2の例では，滑車パッドと，バネパッドを組み合わせたものを検索条件とし，これを部品として利用しているアプリケーションを検索している．

3.2 貼り合わせ構造検索とインデックス

パッドの貼り合わせ構造は，パッドというインスタンスをダイナミックに合成することにより，定義される．しかし，従来の OODB においては，このようなダイナミックに定義される構造に対するインデックスについての考慮はなされていない．これまでに提案されたインデックスは，クラス定義によるスタティックな構造のみを扱っている．文献14)においては，入れ子インデックス，パスインデックス，マルチインデックスの3種類のインデックスについて論じているが，いずれもクラス定義に基づいたスタティックな構造を前提としたものである．文献15)は，入れ子インデックスを採用している GemStone を用いて，パッドの貼り合わせ構造検索について，4種類の検索処理法(全数探索法，台紙索引法，動的索引法，親子索引法)を提案し，検索速度の比較を行った．いずれの方法でも，十数分から数十分の時間を要する結果となっている．つまり，部品によるプログラミングを主とするシステムにおいて，従来の OODB におけるインデックスでは，部品の検索に十分なパフォーマンスを提供することはできない．CAD などの応用分野を考えたときには，さまざまな部品を自在に組み合わせて定義されるさまざまな合成部品を管理する必要性がある．これに対応できるためには，ダイナミックに定義される構造に関しても，高速に検索を行えるような手法が必要である．

4. シグニチャ・ファイルによる部分構造検索の高速化

ここでは，シグニチャ・ファイルを利用した貼り合わせ構造検索の高速化について述べる．

図3に重ね合わせ符号法の検索の流れを示す(図中，パッドの貼り合わせ構造を木構造で表現する)．まず，各合成パッドを構成するプリミティブ・パッドの種類と貼り合わせ構造をコード化したビットマップを作成する．このビットマップをパッドの種類と貼り合わせ関係に対するシグニチャと呼ぶ．すべてのパッドは，登録される際に，シグニチャが生成され，これらを集めてシグニチャ・ファイルを作成しておく(パッドがデータベースから消去される際，そのパッドのシグニチャもシグニチャ・ファイルから消去される)．検索するときは，シグニチャ・ファイルの中から，Query の構造を示すビットマップ (Query シグニチャ) と同じ位置にビットの立っているものを選び出す．この中には，検索条件の貼り合わせ構造を満たさないもの (false drop と呼ばれる) も含まれる可能性があるため，その検査を行い，false drop を取り除いて，検索を完了する．

4.1 パッドのコード化

パッドのコード化の手順を以下に示す．各プリミテ

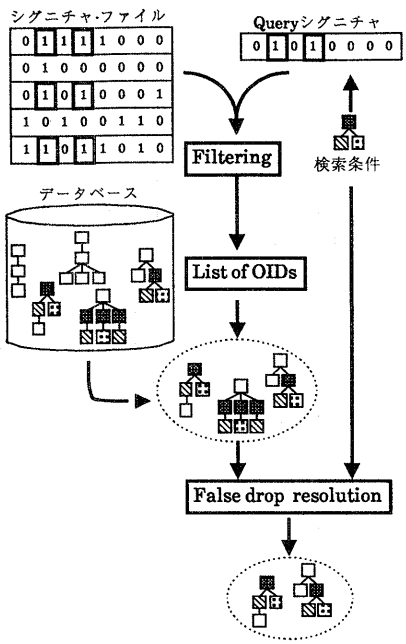

図 3 シグニチャ・ファイルを用いた検索の流れ
Fig. 3 Our access method using a signature file.

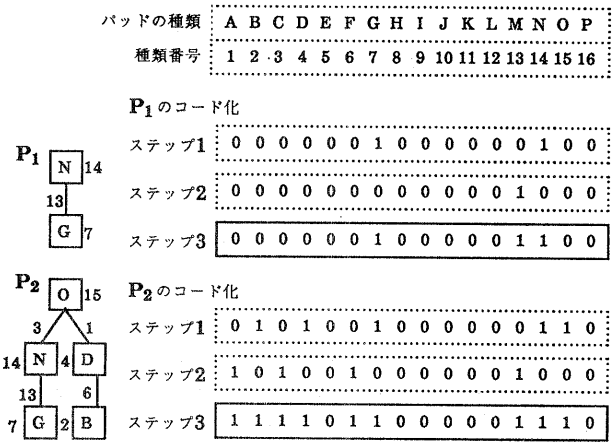

図 4 パッドのコード化
Fig. 4 Superimposed coding of pads.

ィブ・パッドの種類は，種類番号 k で示される．シグニチャの大きさを f とする．

【ステップ1】：パッドの種類番号 k より，ハッシュ関数 $h(k)$ によって求められる位置にビットを立てる．

$$h(k)=\alpha k \bmod f \quad : \bmod \text{は剰余演算子} \qquad ①$$

（α は任意定数，α と f は互いに素）

【ステップ2】：パッドの貼り合わせ構造を示す位置にビットを立てる．位置はハッシュ関数 $h(k_p, k_c)$ によって求める（親パッドの種類番号を k_p，子パッドの種類番号を k_c とする）．

$$h(k_p, k_c)=(\beta k_p+\gamma k_c) \bmod f \qquad ②$$

（β，γ は任意定数，β，γ，f は互いに素）

【ステップ3】：ステップ1，2より作成されたビットマップを重ね合わせる（論理和をとる）．

図4に，パッドのコード化の例を示す．A～Pは，パッドの種類を示し，それぞれの種類番号は図中に示されているものとする．ビットマップの長さ f は 16 とし，$\alpha=1$，$\beta=3$，$\gamma=5$ としてコード化する（図4において，各ノードの横の数字は①式により計算された位置，各アークの横の数字は②式により計算された位置を示す）．パッド P_1 のコード化は，まず，ステップ1により，14番目（N）と7番目（G）に1が立つ．次に，ステップ2により，13番目（$h(14, 7)=(3\times14+5\times7) \bmod 16=13$）に1が立つ．ステップ3で，これらの論理和をとり，P_1 には 0000001000001100 というビットマップが割り当てられる．同様にして，P_2 には，1111011000001110 というビットマップが割り当てられる．P_1 は，P_2 の部分構造を示しており，P_1 のシグニチャにおいて，1の立っている部分には，P_2 のシグニチャにも1が立っている．このため，P_1 を条件として，P_2 を引き出すことができる．

このコード化は，種類と貼り合わせ構造をコード化しているため，同種のプリミティブパッドを部品として用いているアプリケーションでも，その構造によって区別できるという利点をもつ．また，指定された部分構造が木構造のどの位置に現れていても，その構造を含むものはすべて選び出すことができる．

4.2 シグニチャの解析

重ね合わせ符号法を用いた場合の検索パフォーマンスは，シグニチャ・ファイルを利用することで，どの程度まで検索対象を絞り込めるかによる．つまり，検索時間の短縮は，false drop をいかに少なくできるかに大きく依存する．さらに，false drop の多寡は，シグニチャの重み（シグニチャにおいて，1の立っているビットの数を，そのシグニチャの重みという）とQueryシグニチャの重みに関係する．本節では，構成部品数に応じて異なるコード化規則を用いることにより，false drop

表 1 記 号 表
Table 1 List of symbols used in the text.

記 号	定 義
f	シグニチャの大きさ
W	シグニチャの重み
W_q	Query シグニチャの重み
k	パッドの種類に関して立てるビットの数
s	パッドの貼り合わせ構造に関して立てるビットの数
n	合成パッドを構成するパッドの枚数
K	パッドの種類の数
$W(k, s, n)$	n枚で構成されたパッドの種類に関して k ビット，貼り合わせ構造に関して s ビット立てたときのシグニチャの重み
$P_s(W, W_q)$	Wq 個の指定された位置に1の立っているシグニチャの数がデータベース内のシグニチャ数に占める割合
P_q	Query パッドが指定した貼付条件を満たすパッドの存在確率
$P_q(n)\min$	n枚で構成される合成パッドが，2枚で指定された Query パッドの示す条件を満たす確率
Fd	falsedrop の確率（条件を満たさないパッドのうち，そのシグニチャが Query シグニチャを満たすものの割合）

を小さく押さえられることを解析的に示す．ここでは，任意のパッドを任意に貼り合わせた合成パッドによるデータベースを仮定する．使われているパッドの種類や，貼り合わせ構造に偏りはないものとする．また，**表1**は，ここで使われる記号をまとめたものである．

4.2.1 パッドシグニチャの重みと Query シグニチャの重みの関係

シグニチャの重みWと，Query シグニチャの重み W_q により，抽出されるシグニチャ数の全シグニチャ数に占める割合 P_s は以下の式で求められる（付録A参照）．

$$P_s(W, W_q)=\prod_{i=0}^{W_q}(\mathrm{W}-i)/(\mathrm{f}-i) \qquad (1)$$

f=128 としたときの P_s の計算結果を**図5**に示す．

文献5),6)によると，重ね合わせ符号法においては，シグニチャの重みはシグニチャの大きさの半分が適当である．シグニチャの重みが，ビットマップの長さの半分のときを基準にすると，シグニチャの重みが半分以上になると，検出されるシグニチャの数は，基準の場合に比べて多くなる．また，半分以下の場合には，少なくなる．このため，パッドのシグニチャにおける重みは，$f/2$ 以下であることが望ましい．また，最も条件の甘い Query パッドの貼り合わせは，パッドを二枚貼り合わせたものである（Query パッドが，プリミティブパッド（一枚）である場合は，パッドの種類に関するインデックスを利用する）．このパッドをコード化したときのシグニチャの重みを $W_q$min とする．$W_q$min が大きくなると P_s の値は小さくなるので，$W_q$min は大きい方がよい．前述のコード化の方法では，各パッドの種類と貼り合わせ構造に対して1ビットずつ立ててシグニチャを作成した．パッドの種類に関して立てるビットの数 k と，パッドの貼り合わせ構造に関して立てるビットの数 s を増やすことによって，$W_q$min を大きくすることができる．次節において，k, s をどのように増やせば良いかを検討する．

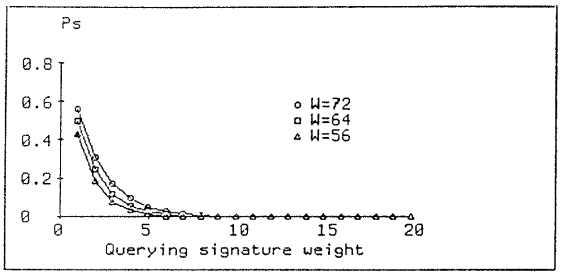

図5 Query の重みと抽出されるシグニチャ数の関係
Fig. 5 Qualifying signatures probability as a function of the querying signature weight.

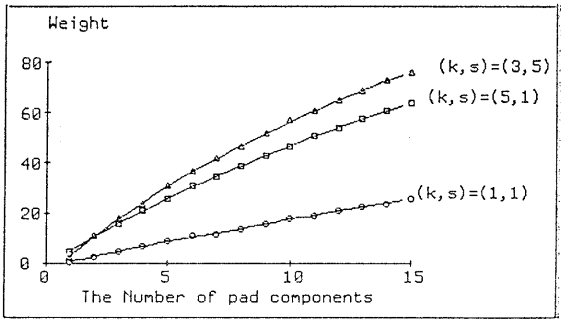

図6 パッドの構成枚数と重みの関係
Fig. 6 Signature weight as a function of the number of pad components.

4.2.2 各パッドに割り当てられるビット数の条件

n 枚で構成されているパッドに関して，使われるパッドの種類に関して k ビット，貼り合わせ構造に関して s ビット立てるとする（初期値は，$k=s=1$ である）．コード化規則 (k, s) で，パッドをコード化するには，4.1 節で述べたコード化において，ステップ1を k 回，ステップ2を s 回繰り返し，ステップ3を行う（ただし，各回において，α, β, γ の値は異なる）．このとき，シグニチャの重み $W(k, s, n)$ は以下のように求めることができる．

文献6)より，1ワード当たり m 個パンチングするとして，大きさ RC のフィールドに w ワードパンチングするとき，パンチングされる穴の期待値 Ge は以下の近似式により求められる．

$$\mathrm{Ge}\doteqdot \mathrm{RC}(1-e^{-wm/\mathrm{RC}}) \qquad (2)$$

ここで，

$$\mathrm{RC}=\mathrm{f}$$

$$wm=nk+(n-1)s$$

を代入すると W(k, s, n) は以下のように求められる．

$$W(k, s, n)\doteqdot f(1-e^{-((nk+(n-1)s)/f)}) \qquad (3)$$

二枚構成の Query パッドの重みを増すために k, または s を増やす必要がある．**図6**に示すように，k, s の増やし方は，複数ある．しかし，同時にパッドシグニチャの重みが $f/2$ 以下であることを満たさなければならない．IntelligentPad では，使うパッドの種類，貼り合わせ方，枚数などに制限はない．図6に示すように，いずれのコード化規則を用いるにしろ，構

枚数の増加により重みも増え，あらゆるパッドのシグニチャに対して，構成パッドの種類や枚数にかかわらず，常に重みが $f/2$ 以下であることを満たすことはできない．しかし，$f/2$ 以下の重みで表現できるパッドの枚数が多い方が良いのは，4.2.1 節より明らかである．つまり，k と s を増かすときには，(i)二枚構成のパッドの重みの増加率は大きく，(ii)$f/2$ 以下の重みで表現できるパッドの数が多いことが条件となる．

【条件(i)】

③式より，

$$\partial W/\partial k-\partial W/\partial s=e^{-((nk+(n-1)s)/f}>0 \quad (4)$$

である．k を増やしたときの W の増加量は，s を増やしたときの W の増加量より大きく，構成枚数 n の少ないほど，その差は大きいので，条件(i)を満たすためには k を優先的に増やせばよい．k と s の初期値は 1 なので，k を優先的に増やすことにより $k \geqq s$ が導かれる．

【条件(ii)】

ルール (k, s) でビットを立てた場合に，重みが半分となるときのパッドの枚数を N，ルール (k', s') の場合を N' とする．このとき，$N>N'$ となるためには，$k+s<k'+s'$ であればよい（図 6 および，付録B参照）．このため，k，または s を増加させるとき，その k，s の増やし方が複数ある場合には，$k+s$ が最小のものを選べばよい．

条件(i)と，$k \geqq 1$，$s \geqq 1$ より，(k, s) の存在範囲は図 7 の斜線部となる．このとき，$k+s$ が最小となるのは，$(k, s)=(1, 1)$ のときである．次にシグニチャの重みを増やすためには，$(k, s)=(2, 1)$ とするとよい．さらに増やすためには，$(k, s)=(3, 1)$ か，$(k, s)=(2, 2)$ が考えられる．ここで，パッドの全種類数を K とすると，パッドの種類に関しては K パターン以上，貼り合わせ構造に関しては $K \times K$ パターン以上をシグニチャにおいて表現できることが望まれる．パッドの貼り合わせに関して立てるビット数 s が多い方が，より多くのパターンを表現できるため，$(k, s)=(2, 2)$ が望ましい．つまり，(k, s) は，$k=s$，あるいは，$k=s+1$ を満たしながら，増やしていけばよい．

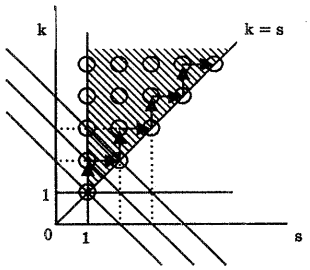

図 7 k, s の増やし方
Fig. 7 The method of increasing k and s.

4.2.3 パッドの構成枚数による場合わけ

k, s を変えることにより，false drop がどのように変化するかを解析する．false drop の割合 Fd は，貼り合わせ条件を満たさないパッドのうち，そのシグニチャが Query シグニチャを満たすものの割合を示す[5]．ゆえに，F_d は P_s と P_q を用いて以下の式で定義される．

$$F_d=(P_s-P_q)/(1-P_q) \quad (5)$$

Query パッドは，二枚のプリミティブ・パッドにより構成されているものとする．このときの P_q は，

$$P_q(n)\min=1-(K^2-1)/K^2)^{n-1} \quad (6)$$

で計算できる（付録C参照）．

図 8 に，f=128 として計算した F_d のグラフを示す．横軸は，データベース内の各合成パッドを構成するプリミティブ・パッドの枚数を示す．図 8 より，パッドの構成枚数が 33 枚以上においては，$(k, s)=(1, 1)$ のグラフの F_d の値が最も小さくなる．パッドの構成枚数が 20 枚から 32 枚ぐらいのときは，$(k, s)=(2, 2)$ の F_d が最も小さくなっており，12 枚から 20 枚くらいのときには，$(k, s)=(3, 3)$ の F_d が最も小さくなっている．つまり，パッドの構成枚数が少ないときには，k や s を増加させることにより F_d を小さくすることができるが，パッドの構成枚数が多くなると，

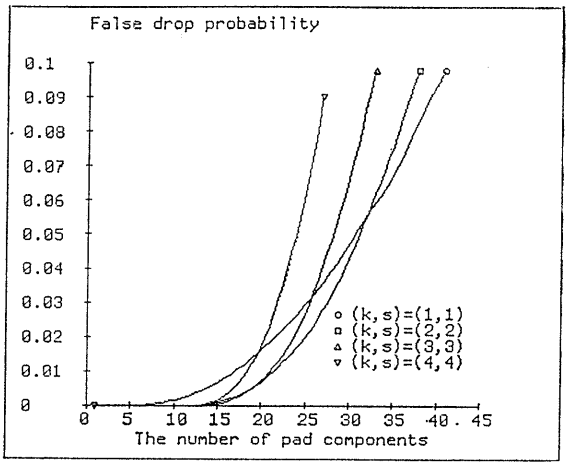

図 8 パッドの構成枚数とフォルスドロップの割合の関係
Fig. 8 False drop probability as a function of the number of pad components.

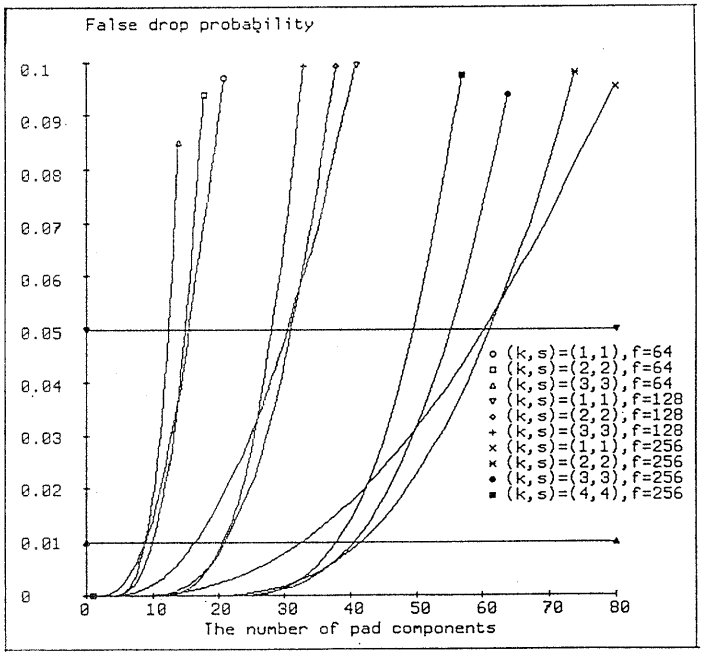

図 9 パッドの構成枚数とフォルスドロップの割合の関係
Fig. 9 False drop probability as a function of the number of pad components.

k や s の値が小さい方が F_d を小さくできる．これらの特性から，パッドの枚数により，ルール (k, s) を変えることで，false drop を低く押さえられることがわかる．

しかし，$(k, s)=(1, 1)$ としても，パッドの構成枚数の増加により F_d も増加する．パッドの貼り合わせ枚数には制限がないので，あらかじめ十分な長さのシグニチャ長を決定することはできない．そこで，F_d をある値以下に保つために，シグニチャの長さは，パッドの枚数に応じて変えることとした．**図 9** は，$f=64$, 128, 256 のときの F_d を計算したものである．パッドの構成枚数により，ルール (k, s) や，ビットマップの長さ f を変えることで，F_d を常に一定の値以下に押さえることができる．つまり，F_d の最大値 $\max F_d$ と，f の変化のさせ方をあらかじめ設定し，(5)式を計算することにより，パッドの構成枚数に適した k, s を求めシグニチャを作成する．**表 2** において，$\max F_d$ を 0.01，あるいは 0.05 に設定し，シグニチャの長さ f_i を，$f_{i+1}=2f_i (i=0, 1, 2, \cdots)$ $(f_0=64)$ としたときの例を示す（表 2 においては，$i=0, 1, 2$ の場合を示した）．**図 10** および**図 11** は，表 2 のようにコード化したシグニチャ・ファイルを用いて検索を行った場合の F_d を示す．いずれも，$\max F_d$ で設定した値以下の F_d を示している．また，**図 12** は n 枚以下で構成されるパッドが同数ずつ存在するデータベースにおいて，表 2 のコード化を用いて F_d を計算したものである．図の横軸は，パッドの最大構成数を示す．これも，F_d を一定値以下に保つことができている．

以上より，パッドの構成枚数に応じて，コード化規則や，シグニチャの長さを変えることで F_d の値を一定値以下に押さえられることを示した．

4.3 コード化規則と検索時間

(i)パッドの構成枚数や種類に関係なく，一定の規則で固定長のシグニチャを作成する場合と，(ii)パッドの構成枚数や種類に応じてシグニチャを作成する場合の検索時間について検討する．それぞれの検索の流れを，**図 13** に示す．検索時間は，Query シグニチャを作成する時間，シグニチャ・ファイルを検査するのに要する時間，シグニチャの検査によって抽出されたパッドに関する false drop の検査

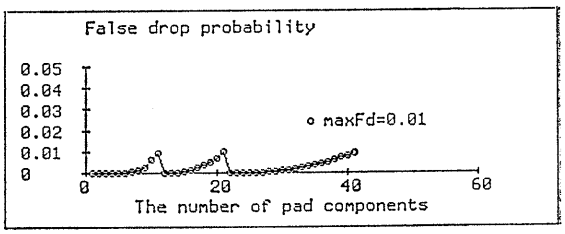

図 10 パッドの構成枚数とフォルスドロップの割合の関係
Fig. 10 False drop probability as a function of the number of pad components.

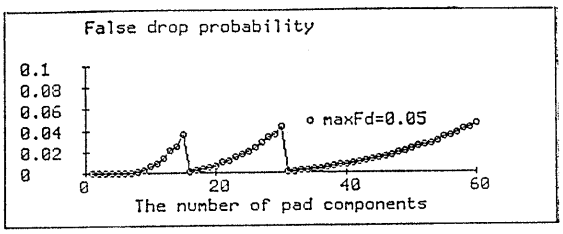

図 11 パッドの構成枚数とフォルスドロップの割合の関係
Fig. 11 False drop probability as a function of the number of pad components.

表 2 パッドの構成枚数，シグニチャの長さ，コード化規則の対応表
Table 2 Relationships among the number of primitive pads, the signature size and the different coding rule.

$\max F_d=0.01$			$\max F_d=0.05$		
n	f	(k,s)	n	f	(k,s)
0～1	64(f_0)	(3,3)	0～6	64(f_0)	(3,3)
～11	64(f_0)	(2,2)	～15	64(f_0)	(2,2)
～19	128(f_1)	(3,3)	～19	128(f_1)	(3,3)
～21	128(f_1)	(2,2)	～30	128(f)	(2,2)
～25	256(f_2)	(4,4)	～37	256(f_2)	(3,3)
～37	256(f_2)	(3,3)	～60	256(f_2)	(2,2)
～41	256(f_2)	(2,2)	|	|	|
|	|	|	|	|	|

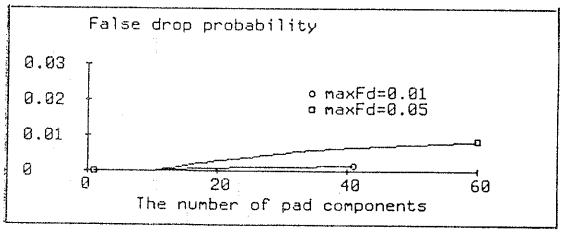

図 12 パッドの構成枚数とフォルスドロップの割合の関係
Fig. 12 False drop probability as a function of the number of pad components.

の時間の和によって求められる．このうち，Query シグニチャを作成する時間とシグニチャ・ファイルを検査するのに要する時間は，false drop の検査の時間に比べるときわめて小さいので，無視できる．つまり，検索時間の長短は false drop 数の大小によって表現できる．そこで，(i)，(ii)の場合の false drop 数をそれぞれ F i，F ii とする（図8で示した F_d は，F i に相当し，図9および図10で示した F_d は，F ii に相当する）．

データベース内のパッドの構成枚数や種類が，(i)の場合のシグニチャで十分に表現できる枚数以下である場合には，F i $<F$ ii となり，(i)の場合の方が早く検索できるだろう．しかし，(ii)は，false drop を一定値以下に保つことができるので，この場合でも十分なパフォーマンスを得ることができる．一方，データベース内のパッドの構成枚数や種類が，(i)の場合のシグニチャで十分に表現できる枚数以上である場合には，F i $>F$ ii となり，(ii)の場合の方が早く検索できる．しかも，(i)の false drop は，指数関数的に増加するため，十分なパフォーマンスを得ることができない．結局，(ii)では，データベース内のパッドの構成によらず，安定したパフォーマンスが得られることがわかる．

5. おわりに

パッドベースは，多様な部品とそれらの合成によるアプリケーションの統合管理を目指しているシステムである．本論文では，部品の組合せによるアプリケーションにおいて，その部分構造を条件とした検索について述べた．さまざまな部品や，半製品，製品などのなかから必要なものを検索するためには，その部分構

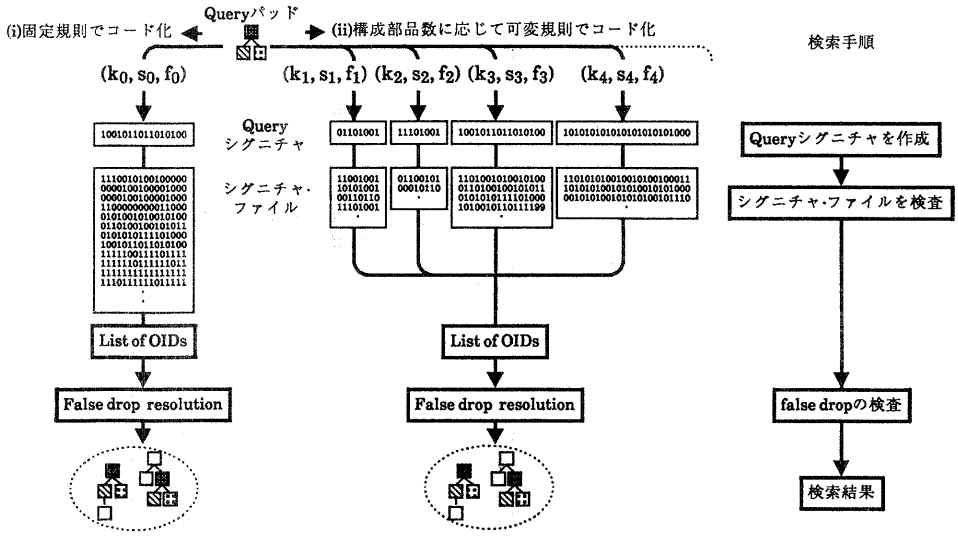

図 13 コード化の違いによる検索の流れの比較
Fig. 13 Two different search schemes for the two different coding schemes.

造を指定して検索を行うことが有効な場合がある．この検索の高速化のために，重ね合わせ符号法を用いた．アプリケーションを構成するパッドの数に応じて，コード化するときに立てるビットの数のシグニチャの長さを変えることで，false drop を少なく押さえられることを示した．

また，IntelligentPad では，パッド間のリンクをパッドとして実現している[16]．このため，リンクの種類や構造を検索する際にも，本論文で提案した方法を利用できると思われる．

参考文献

1) Cattell, R. G. G.: *Object Data Management: Object-Oriented and Extended Relational Database Systems*, Addison-Wesley, Reading, Massachusetts (1991).
2) Bancilhon, F., Delobel, C. and Kanellakis, P.: *Building an Object-Oriented Database System*, Morgan Kaufmann, San Mateo, California (1992).
3) Tsichiritzis, D.: Object-Oriented Development for Open Systems, *Proc. IFIP Congress '89*, San Francisco, pp. 1033-1040 (Aug. 1989).
4) Tanaka, Y.: A Toolkit System for the Synthesis and the Management of Active Media Objects, *Proc. 1st International Conference on Deductive and Object-Oriented Databases*, Kyoto, pp. 269-277 (Dec. 1989).
5) Faloutsos, C. and Christodoulakis, S.: Signature Files: An Access Method for Documents and Its Analytical Performance Evaluation, *ACM Transactions on Office Information Systems*, Vol. 2, No. 4, pp. 267-288 (1984).
6) Stiassny, S.: Mathematical Analysis of Various Superimposed Coding Methods, *American Documentation*, Vol. 6, pp. 155-169 (1960).
7) 森田幸伯，和田光教，伊藤英則：スーパーインポーズドコードを用いた構造体の検索方式，第33回情報処理学会全国大会論文集，pp. 1277-1278 (1985).
8) 森田幸伯，物井秀俊，仲瀬明彦，柴山茂樹：MPPMを用いた知識ベースマシン(3)，第35回情報処理学会全国大会論文集第5分冊，pp. 123-124 (1987).
9) Tanaka, Y.: A Synthetic Dynamic-Media System, *Proc. International Conference on Multimedia Information Systems*, Singapore, pp. 299-310 (Jan. 1991).
10) Tanaka, Y., Nagasaki, A., Akaishi, M. and Noguchi, T.: A Synthetic Media Architecture for an Object-Oriented Open Platform, *Proc. IFIP 12th World Computer Congress Madrid*, Spain, pp. 194-110 (Sept. 1992).
11) Goldberg, A. and Robson, D.: *Smalltalk-80: The Language and Its Implementation*, Addison-Wesley, (1983).
12) Copeland, G. and Maier, D.: Making Smalltalk a Database System, *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 316-325 (1984).
13) Maier, D. et al.: Development of an Object-Oriented DBMS, *OOPSLA '86 Proceedings*, pp. 472-482 (1986).
14) Bertino, E. and Kim, W.: Indexing Techniques for Queries on Nested Object, *IEEE trans. Knowledge and Data Engineering*, Vol. 1, No. 2, pp. 196-214, June (1989).
15) 赤石美奈，田中 譲：IntelligentPad におけるパッド管理データベースシステムの開発，日本ソフトウェア科学会第8回大会論文集，pp. 349-352 (1991).
16) 葛西裕昭，田中 譲：ハイパーメディアにおけるリンクの機能拡張と機能合成，1993情報学シンポジウム講演論文集，pp. 75-83 (1993).

付録 A

シグニチャSの重み W と，Query シグニチャの重み W_q により，抽出されるシグニチャ数の全シグニチャ数に対する割合 P_s を求める．

Query シグニチャで，1の立てられている W_q 個のビットを $b_1 b_2 \cdots b_{W_q}$ とする．Sにおいて，b_1 の位置に1が立てられている確率は，W/f である．次に，b_2 の位置に1が立てられている確率は $(W-1)/(f-1)$ である．ゆえに，SがQuery シグニチャの条件をすべて満たす確率は，$(W/f)\cdot((W-1)/(f-1))\cdots((W-W_q)/(f-W_q))$ となる．

ゆえに，

$$P_s(W_p, W_q)=\prod_{i=0}^{W_q}(W-i)/(f-i) \qquad (A.1)$$

となる．

付録 B

$W(k, s, n_0)=W(k', s', n_0)$，$W(k, s, N)=W(k', s', N')=f/2$ のとき，$N>N'$ であるためには，$k+s<k'+s'$ であればよいことを示す．

$W(k, s, n)\fallingdotseq f(1-e^{-((nk+(n-1)s)/f)})$

(本文式⑤より)

$W(k, s, n_0)=W(k', s', n_0)$ より，

$$(k+s)n_0-s=(k'+s')n_0-s' \qquad (B.1)$$

$W(k, s, N)=W(k', s', N')$ より，

$$(k+s)N-s=(k'+sN')'-s' \qquad (B.2)$$

(B.1), (B.2) より,

$$(k+s)(N-n_0)=(k'+s')(N'-n_0) \qquad (B.3)$$

このとき, $(0<)k+s<k'+s'$ であれば, $N-n_0>N'-n_0(>0)$ なので,

$$N>N' \qquad (B.4)$$

が導かれる.

付 録 C

n 枚で構成された合成パッドPが, 2枚で指定された Query パッドの条件を満たす確率 $P_q(n)\min$ は以下の式で表される. ただし, Pは, K 種類のプリミティブ・パッドのうち任意のパッドを任意に貼り合わせたものとする.

Query パッドを構成する2枚のプリミティブ・パッドのうち, 台紙になっているプリミティブ・パッドを Q_1, その上に貼られているパッドを Q_2 とする. まず, Pを構成するパッドから, 直接貼り合わせ関係にある2枚のプリミティブ・パッド P_1 と P_2 を任意に選ぶ. P_1 の上に P_2 が貼られているものとする. このとき, P_1 が Q_1 と同種のパッドではない確率は $(K-1)/K$ である. また, P_1 と Q_1 が同種のパッドであり, かつ P_2 と Q_2 が同種のパッドではない確率は $(1/K)(K-1)/K$ となる. ゆえに, P_1 と P_2 が, Query パッドの条件を満たさない確率は, $(K^2-1)/K^2$ である. P_1 と P_2 の選び方は $n-1$ 通りであるので, Pが Query パッドの条件を満たさない確率は $((K^2-1)/K^2)^{n-1}$ となる. ゆえに, Pが Query パッドの貼り合わせ構造を内部構造としてもつ確率は, $1-((K^2-1)/K^2)^{n-1}$ となる. よって,

$$P_q(n)\min=1-((K^2-1)/K^2)^{n-1} \qquad (C.1)$$

となる.

(平成5年 3 月25日受付)
(平成5年10月14日採録)

赤石 美奈(正会員)

昭和42年生. 平成2年北海道大学工学部電気工学科卒業. 平成4年同大学院修士課程修了. 現在, 北海道大学大学院工学研究科博士後期課程電気工学専攻在学中. ヒューマンインタフェース, オブジェクト指向データベース等の研究に従事. ソフトウェア科学会, 人工知能学会各会員.

田中 譲(正会員)

昭和25年生. 昭和47年京都大学電気工学科卒業. 昭和49年京都大学電子工学専攻修士課程修了. 工学博士. 現在, 北海道大学電気工学科教授. データベースマシン, データベース理論, メディア・ベース, 論理型プログラミング等の研究に従事. 主たる著書「コンピュータ・アーキテクチャ」(オーム社, 共著). IEEE, ソフトウェア科学会, 人工知能学会各会員.