学位請求論文

多様な論理構造を扱う
文書データベースに関する研究

A Study on a Document Database for Various Logical Structures

指導教官

安達 淳 教授

1996年 12月 20日 提出

東京大学大学院工学系研究科

電気工学専攻 47078

酒井 乃里子

目次

図目次

表目次

第１章

はじめに

1.1 本研究の背景

1.1.1 電子化文書の利点

近年、特にこの数年で計算機を取り巻く環境は急速な変化を遂げている。計算機の高速化、二次記憶装置の高速化・大容量化もさることながら、インターネットに代表されるネットワークの発達が目覚ましい。また、個人レベルへの計算機の普及が進んでいる。これらの状況があいまって、文書を扱う目的でも、電子化を進める勢いが強い。

文書を電子的に扱うことによって、

- 執筆・編集が容易になる
  - 挿入や削除などの推敲や変更
  - 複数の、時には物理的にはなれた著者の共著
  - 時間をおいて後の、再利用
- 文書処理の各段階間でのデータの受渡しが容易になる
  - 執筆から編集、整形、そして印刷へ、
- 文書の応用的な利用が可能になる
  - 蓄積する際の、保管スペースや費用の削減
  - 経時的劣化の防止
  - 検索や比較、また読者による抽出した抄訳作成
  - ユーザとデータが物理的に離れた場所での閲覧

といった利便が得られる。

こういった電子化文書の特長を、より大きくするためには、処理のために重要な情報を文書内に明示する、一定の規則を採用することが有用である。SGML (Standard Generalized Markup Language; ISO8879) はこの目的のために制定された国際規格である。

1.1.2 構造化文書の利点と課題

文書は論理構造を持つ。ここでいう論理構造とは、文書が二次情報の他に「章」という単位から構成され、さらに「章」はいくつかの「節」から構成されるといった、再帰的な内部の作りを指す。SGML では、この論理構造を文書中に明示的に書き込むことにより、執筆以降の一連の文書処理において、処理を効率化もしくは高度化する共通の手がかりとしようとするものである。

ところが SGML では、論理構造の定め方を定義する規格であり、実際の論理構造は著者などが、需要や好みに応じて設計することになっている。このため、学会や出版社などごとに、文書は多様な論理構造をもつようになっている。

このような文書を高度な利用のために、データベースとして蓄積することを考える。データベースの役割、すなわち収録データ数をより大きくするためには、なるべく多くの発行元による文書を対象とする必要がある。そこで、異なる論理構造を持つ複数の学会などの発行する文献を扱うことが望まれる。

文書データベースに求められる機能としては、なんらかの条件を満たす文献を検索することや、その結果あるいは誌名・巻号が周知の文献を閲覧することであろう。この機能の実現にとって、多様な文書を収容することは、検索条件の記述や実際の検索処理などで、評価すべき要素が各論理構造においてどれにあ

たるのか、判断が困難になることを意味する。また、ディスプレイに誌面を表示するための整形に当たっても、各論理構造が各々に独自の処理系を備えていた場合に、論理構造の種類に応じてアプリケーションを選んで起動する必要が生じる。このためシステムの処理の効率化のためには、論理構造の多様性を吸収する機構が求められる。

1.1.3 本研究の焦点

以上のような問題点を踏まえて、本研究では、

- 電子的に蓄積された
- SGML により論理構造を明記された

文書を対象にして、

- これらの文書を統合して扱う文書データベースのための
- 多様な論理構造を持つ文書を、必要に応じて統一的な論理構造として処理またユーザに提供するための

論理構造の変換手法を提案する。

またこの手法に関連して、

- 文書データベースの研究状況

をまとめるほか、

- 構造化文書の処理
- データベースとして求める機構

などについても検討を行った。

1.2 本論文の構成

この第1章以下本論文は、以下のような内容で構成される。

- 第2章
  本研究で前提としているSGMLの規格、類似規格との比較、SGMLの需要や普及への展望
- 第3章
  本研究の論理構造変換手法が必要とされる論理構造の多様性による課題や、手法を組み込むと想定しているデータベースの全体像
- 第4章
  本研究に関連する、構造化文書の処理にまつわる研究の動向
- 第5章
  想定しているデータベースに近いシステムとして「電子図書館」と称されるものの試行実験が盛んに行われている。その概観
- 第6章
  提案する論理構造変換手法の全体像。特に、変換仕様の記述法
- 第7章
  前章の仕様を用いた、実際のデータの処理方法
- 第8章
  提案した論理構造変換手法のプロトタイプと実際の文書データを用いた実験と、手法の評価及び展望

第 2 章

構造化文書の記述

本研究では構造化文書を記述する言語としてSGMLというものを採用している。そこで、まずSGMLとはどういうもので、どのくらい現在普及しあるいは将来的に重要視されていくものなのか、またSGMLを扱う場合どのようなことが課題とされているかについてまとめる。

本章の構成は以下の通り。

- 2.1: SGMLとは
  SGMLにおける文書の利用全体の考え方と、構造化文書の記述の仕方、その特徴について述べる。

- 2.2: 同類の規格ODA:
  構造化文書を扱う国際規格には、ODAというものもある。こちらも多く利用されているが、必ずしもSGMLと対立するものではない。ODAの規定する文書の記述法や活用法を述べ、SGMLとの関連について述べる。

- 2.3: HTML
  現在最も身近にSGMLと接する機会であり、SGMLという考えや言葉を広く一般にも定着させた、WWW用SGMLであるHTMLについて、その簡単な紹介と、展望などを述べる。

- 2.4: CALS
  専門的に文書を扱う業界では、業界の電子化文書の標準としてCALSという考えを採用し始めており、将来的にはこの機運にのって深くSGMLが浸透していくと考えられる。このようなCALSの経緯などをまとめる。

- 2.5: SGMLの処理の課題
  SGMLは強力な規格であるが、その能力ゆえに処理が難しいという側面も否めない。ここではSGMLの処理がなぜ難しいのか、ポイントをまとめる。

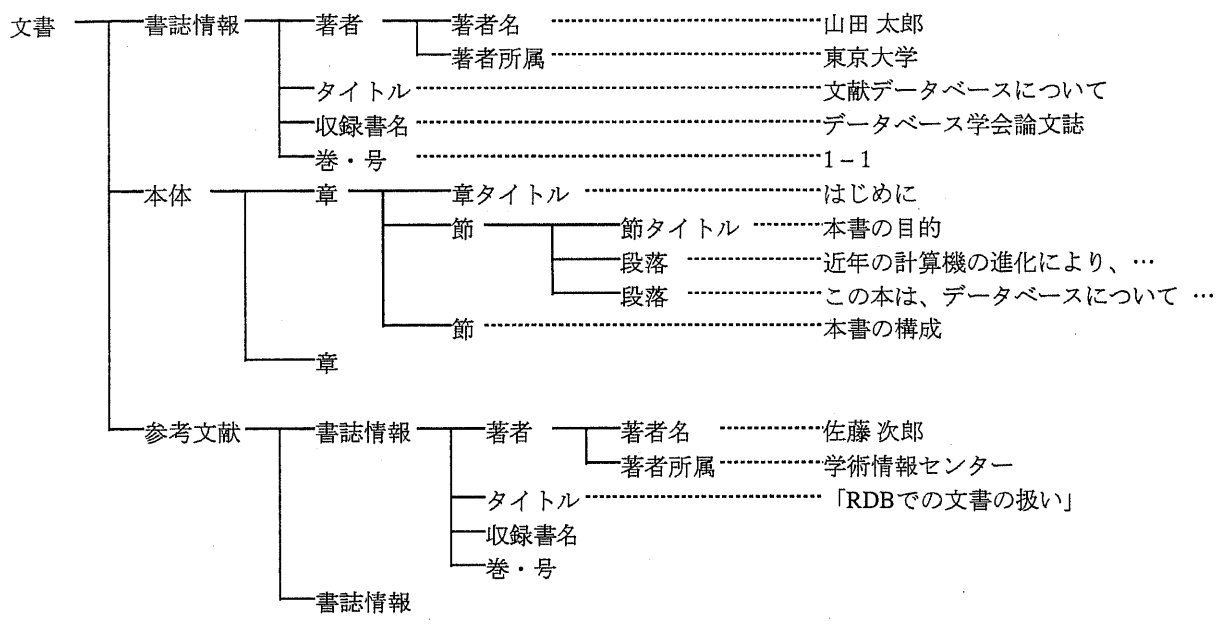


図 2.1: 文書の論理構造

2.1 SGML とは

2.1.1 文書の論理構造

文書は論理構造を持つ。ここでいう論理構造とは、文書が、より細かい単位へ再帰的に分解できる、階層的な作りのことをいう。図 2.1に例を示す。

この図では、文書全体が書誌情報と本文の部分（本体)、そして参考文献に関する情報から成っており、本体は、いくつかの章で、章はそのタイトルと、いくつかの「節」から構成される、というように示されている。

論理構造は、文書を構成する「書誌情報」「章」などの各単位を論理要素といい、その中にどの要素を含むかを再帰的に定義することを骨格としているが、図 2.1に示すように、実際の文書では同名・同位置の論理要素が複数回出現することがある他、出現する場所がそれほど限定されないような論理要素もある。たとえば図や脚注などは、本体の部分であれば、その範囲内でどこにでも出現する可能性を持つ。

2.1.2 SGML における文書処理

SGML (Standard Generalized Markup Language; ISO8879) [1] は、電子化文書の記述に際して、論理構造を明示する方式を規定した国際規格であり、1986 に制定された。SGML では、文書のデータであるテキストを、それの持つ論理構造を明示しながら記述する方式を採用しており、レイアウトやフォントなどに関する整形情報や、印刷などそれ以外の文書処理の各段階との関連とは切り離している点が特徴であるといえる。

この方式を用いた文書処理全体のイメージは、SGML が関与する範囲とその周辺を総合して、ISO の標準モデルとして図 2.2のように想定されている [2]。

この中で SGML は、文書の論理構造を定義し（文書型定義として参照される。)、論理構造を明示してデータそのものである文書を記述する段階のみを規定している。つまり、任意のエディタなどを用いて文

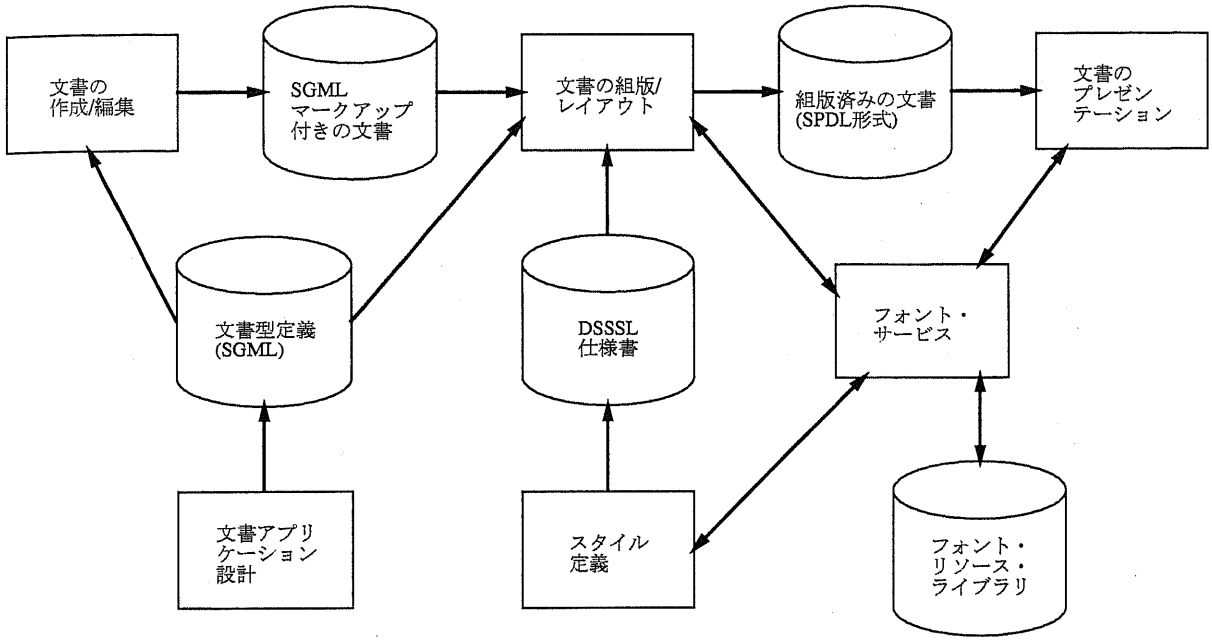


図 2.2: SGML と関連する国際規格の文書処理イメージ

書の論理構造 DTD (Document Type Definition) を定義し、またこの定義に基づく文書を作成する。この後、別に定めておいた出力形式を定義する DSSSL (Document Style Semantic Specification Language; ISO 10179) へと変換し、組版などの段階へデータが渡される。

2.1.3 SGML の構成

SGML は三つの部分から構成される。

- SGML 宣言 (SGML Declaration)
  SGML 宣言には、データを記述する文字コードや、論理要素の名前に使う文字列の条件、あるいは各種の制御情報などを定義する。

- 文書型定義 (Document Type Definition; DTD)
  論理構造を定義する。図 2.3に例を示す。図の一行めは、"Recordlist" という要素を定義しており、まず要素であるところのものの名前を宣言している。その次には、要素名を文中に書き込むタグの省略の可否を記している。SGML では、文書作成の負荷をなるべく軽減するために、明らかに要素の開始・終了が識別できる場合はタグを省略することが可能であり、省略可を "O" で、省略不可を "-" で示す。要素 "Recordlist" に関しては、要素の前後のタグはともに省略できない。

  要素名・タグの省略の可否に続いて、この要素 "Recordlist" の下位に来るべき論理要素について定義している。ここでは "Record" が複数個 ("Record" のあとの "+" の印) 出現してもよい、という定義になっている。これらのもう少し詳しい説明と問題点は第 2.5節で述べる。

- インスタンス
  文書そのもの。DTD で定義された論理構造に従って、定義された要素名を各論理要素に付与して論理要素の開始と終了を区切りながら文書を記述する。図 2.3では、たとえば "Doc1" という文字列は、文

```
<!ELEMENT  Recordlist  -  -  (Record)+>
<!ELEMENT  Record  -  -  Title, (Author)+,  Key>

<Recordlist><Record><Title>Doc1</Title>
<Author><Surname No = 1>Smith</Surname>
<Firstname No = 1>Bob</Firstname>
<Middlename No = 1>Thomas</Middlename>
<Affiliation>NACSIS</Affiliation></Author>
<Author><Surname No = 2>Jones</Surname>
<Surname>Nelson</Surname>
<Firstname>Mary</Firstname>
<Affiliation>NACSIS</Affiliation></Author>
<Key>Structure</Key>
<Key>Database</Key>
<Abst><Para>Study of <Ref>Book1</Ref> is referred.</Para></Abst>
</Record>
<Record><Title>Doc2</Title>
<Author><Firstname>Michael</Firstname>
<Surname>Brown</Surname>
<Affiliation>Univ. of Tokyo</Affiliation></Author>
</Author></Record></Recordlist>
```

図 2.3: DTD とインスタンスの例

書のタイトルを表しているが、タイトルつまり要素 "Title" にあたるということを、文字列 "Title" の前後に、それぞれ "<Title>," "</Title>" という記号を置いて、"<Title>Doc1</Title>" という形式で示している。この時、文字列の前にあるものを「開始タグ」、後ろにあるものを「終了タグ」と呼ぶ。

さらに、この要素 "Title" は上位要素 "Record" の下にあるので、"<Title>Doc1</Title>" を "Record" の名前を用いたタグで挟んで "<Record><Title>Doc1</Title>...</Record>"、という要領を再帰的に繰り返して、文書全体の論理構造を明示的に文書中に示す。このように文書に論理構造を示す印をつけることを、マークアップと呼ぶ。

このような三部分を合わせて処理することで、論理構造を把握したり、内部のデータの記述に的確に対応して、次の段階の組版や表示・印刷、検索などに必要なデータの形式を作り出す。このような、文書の処理を、文書の解析（パース）という。

```
'ページ割り付け' 選択 (
    'ページ割り付け A' (横書き, 左綴じ, 下方向割り付け)
        繰り返し  選択  ( '頭書'
                          '本体',
                          '脚書き'
                        )
    'ページ割り付け B' (横書き, 多段組, 左綴じ, 下方向割り付け
        繰り返し  選択  ( '頭書'
                          '本体',
                          '脚書き'
                        )
    'ページ割り付け C' (縦書き, 右綴じ, 下方向割り付け)
        繰り返し  選択  ( '頭書'
                          '本体',
                          '脚書き'
                        )
    'ページ割り付け D' (縦書き, 右綴じ, 左方向割り付け)
        繰り返し  選択  ( '頭書'
                          '本体',
                          '脚書き'
                        )
                )
```

図 2.4: 共通割り付け構造の例

2.2 ODA (Open Document Architecture)

文書の記述の仕方をある程度統一して、論理構造などの活用、文書の高度利用を促進する規格には、SGML だけでなく、ODA と呼ばれるものも普及している。

2.2.1 ODA の概要

ODA [3] [4] も国際規格 (ISO8613) であり、文書の交換を促進する目的を持つ。組版情報や論理構造をある程度予め定めて共有しようとし、また文書利用全体を対象とする規格である点が SGML とは異っている。

ODA では、文書を記述する構造として、論理構造の他に組版情報を示す割り付け構造も用いる。割り付け構造は、文書をページ、枠、区画といった単位で集めて、論理構造と同様に全体を木構造として把握する。

図 2.4に示した、共通割り付け構造では、ページ割り付けの種類が 4 種類あり、理科の教科書から法令の文書などにまで使える形態が選択できるようになっている。

2.2.2 ODA による文書の構成

ODA の大きな目的の一つは文書の交換であり、ODA に適合したソフトウェアであれば、ODA の定める交換様式を受け入れ、あるいは出力する。ODA では、データのフォーマットと、文書データの構成法、つまり論理構造と割り付け情報の取り入れ方で、それぞれいくつかの種類に文書の作り方を分類することができる。

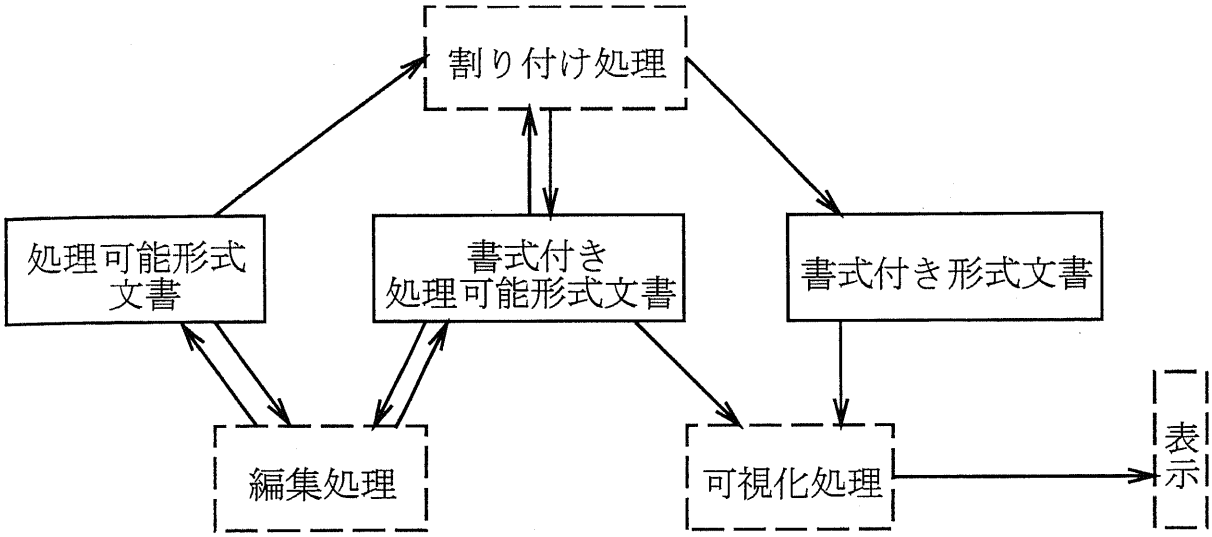


図 2.5: ODA の三種類の文書形式と処理の関係

- データのフォーマット
  - 開放型文書交換様式 (ODIF; Open Document Interchange Format)
    ODIF はバイナリデータであり、データの単位は「情報の種類を表すタグ、内容部の長さ、内容」という構成になっており、抽象構文記法 (ASN.1; Abstract Syntax Notation One) という異種システム間の情報の相互交換のための ISO のアプリケーションレベルのフォーマットを用いている。
  - 事務文書言語 (ODL; Office Document Language)
    ODL は SGML を採用している。つまりデータはタグでマークアップされたテキストデータなので、そのまま人が見たり読んだりなどできる

- 文書データの構成（論理構造と割り付け情報の取り入れ方）
  - 処理可能形式文書 (PDA)
    割り付け構造の情報を持たず、論理構造の情報のみを持ち、編集などが可能。共著作業などで用いられる。
  - 書式付き処理可能形式文書 (FPDA)
    論理構造と割り付け構造の両者の情報を持ち、可視化も一部の変更も可能である。
  - 書式つき形式文書 (FDA)
    論理構造の情報を持たず、割り付け構造の情報のみを持ち、ビューワなどで見ることのみが可能。契約書など改変しない文書に用いられる。

これらは文書を用いる局面に応じて使い分けるが、これらの形式は、共通/ 独自の論理・割り付け構造を用いて変換することができ、最終的には表示など、ユーザに提供できる（図 2.5）。

2.2.3 ODA と SGML

前述のように、ODA は文書の処理全体を対象にして規格化を進めている。しかし、文書の処理全体はかなり大きな手続きであり、それらを包括的に定義し、規格化することは大変な負荷であり、また実際かなりな困難を伴う。

一方、文書の処理を行う担当者は、通常段階ごとに分かれていて構わないし、むしろ、一つのデータを蓄積する段階と処理する段階で、別なものと考えて、蓄積したものを予期せぬ整形などの処理にも適用する方が文書の活用範囲が広まるとも言える。

このような観点から、また現場での SGML の普及も影響して、ODA は、文書を記述する部分に、SGML を取り入れる形で改訂が進められている。従って、文書の作成という局面において、SGML がより一層普及していくという方向性は確実だと予測できる。

2.3 HTML

昨今の、特に個人レベルでのコンピュータネットワークの普及は、WWW (World Wide Web) への関心の高まりによるところも大きい。この WWW のために文書を記述する言語が、HTML (HyperText Markup Language) と呼ばれるもので、もともとは SGML の準拠するものとして開発が始まった。

HTML [5] は、W3C (World Wide Web Consortium) が標準化を進めており、1996 年 11 月 6 日のバージョン 3.2（通称：Wilbur）が最新の推奨標準として公開されている。

HTML はしかし、版を重ねるごとに、本来 SGML が目指していた論理構造を中心に記述するという方向を離れて、おそらくユーザの要望が強かったレイアウト（フォントや、画面上の配置など）を記述する要素を多くとり入れ、あるいはユーザのそのような使い方を歓迎する方向で変更が重ねられている。また実際の使用に際しては、WWW を閲覧するソフト（ブラウザ）が独自の拡張、たとえば表やある種の動作を記述するための要素を HTML に付け加えて実装し、アプリケーションの起動など、様々な要素を付け加えるようになってきた。

このようにして加えられたものが事実上使用に際しての標準とみなされることも多くなった結果、HTML に準拠しているはずのブラウザの中でもものによって対応できるものとできないものに分かれ始めて、ブラウザごとに使用するソースファイルを使い分けるような準備も必要になってきた。

このような状況から、現在では SGML を離れた、独自の、WWW 用記述言語であるという見方をされることが多くなってきた。

2.4 CALS

SGML の普及のもう一つの柱は、産業界での関心の高まりである。この流れに乗って、CALS [6] [7] [8] という考え方が普及してきた。当初 CALS は Continuous Aquisition and Logistics Support の意味に解釈され、産業界での活動に欠かすことのできない書類を、会社規模で、あるいは業界規模、 さらには世界中で規格化することで、処理の効率化やデータ受渡しの容易化を図ろうとする潮流を支えてきた。この流れの中で、文書やデータ交換において、SGML による規格化を図ろうとしている。

この動きに先鞭をつけたのはアメリカ国防総省である。1984 年の冷戦後の物品調達システム（ロジスティクス）の改革に伴って、即時性・安全性・信頼性の向上を目指して、使用する物品のライフサイクル全般にわたる情報とプロセスのインテグレーションを実現した。

この一環として、陸海空三軍全体のための JCALS (Joint CALS; Computer-Aided Aquisition and Logistic Support) が設定され、国防総省と取り引きをするためには、この CALS を使用して仕様書などを納めなければならないようになった。また、私企業でも航空機メーカのボーイング社が、新型器設計に当たって、予めユーザである各国航空会社との意見を交換し、その要望を設計に反映させていこうという方式を採用したプロジェクトが弾みをつけた。このプロジェクトでは、CALS による書類の形式化により、情報やデータの受渡しが効率的になったのみならず、航空機の引渡し後の利用・整備マニュアル、メンテナンスのための交渉など、広い範囲に渡って文書電子化の恩恵を確認することができた。

日本でも、CALS 技術研究組合 [9] が日本における CALS の標準となる文書の構造を制定し NCALS (Nippon CALS) として公開している。これは HTML2.0 に基づいて、組合が想定する文書の機能の上限と位置付けられている。

一方で、ネットワークの一般化に伴って CALS は、Commerce At Light Speed とも称されるようになり、一般ユーザが使用する場面への適用が検討され始めた。インターネットを通じた商取り引きなどでも、採用が検討されている。しかしこのような利用法では、別途ネットワークのセキュリティや、金銭の授受に関してどのような仕組を採用するかといった点も問題として考えられることから、現在のところ WWW などのように急速に普及するまでには至っていない。

2.5 SGML の処理の課題

SGML での記述法は、自由度が高く柔軟に文書を記述する能力を備えているが、反面、その記述能力を網羅するためには、文書の解析コストが高くなる。その原因について検討する [10]。

2.5.1 接続子・接続指示子

DTD では多くの接続子や出現回数指定子を採用している。これにより、複雑な文書を柔軟に記述することができる。

- 接続子
  要素間の関係を示す。
  - Seq 接続子: “,”
    その両側の要素またはモデルグループは、実際の文書中で、記述された通りの順番で現れなければならない。
    (例)

    “<!ELEMENT MEMO - - ((To \& From), Body, Close?)>”

    において、“Body” はモデルグループ “To & From” で表された要素のあとに現れなければならない。
  - And 接続子: “&”
    その両側の要素またはモデルグループは、どちらが先に現れてもよいが、文書中に必ず両者が現れなければならないことを示す。
    (例) 上記の例で、要素 “To” と “From” は、どのような順序であれ必ず現れなければならない。
  - OR 接続子: “|”
    その両側の要素またはモデル群のどちらか一方だけが現れなければならないことを示す。
- 出現指示子
  要素が文書中に何回現れるかを示す。
  - 任意選択的出現子: “?”
    この指示子の前に置かれた要素または群は、1 回現れるか、または全く現れないことを示す。
    (例) 上記の例で、要素 “Close” は、1 回現れるか、または全く現れない。
  - 必須かつ反復可能出現指示子: “+”
    この指示子の前に置かれた要素またはモデルグループは 1 回以上複数会出現可能であることを示す。
  - 任意選択かつ反復可能出現指示子: “*”
    この指示子の前に置かれた要素またはモデルグループは、0 または 1 回以上出現可能であることを示す。

実際の文書はこれらを活用して記述する必要があるほど複雑である。たとえば、これらを文脈自由文法の BNF で記述しようとすると、図 2.6のように、その記述行数は著しく多くなる。

しかし一方で、これだけの種類の接続子・出現指示子を用意したことで、SGML 文書を解析するコストが増大したという側面も否めない。たとえば、

```
A & B
```

章	::=	章題 段落	章	::=	章題 段落群	章	::=	章題 段落群
章題	::=	文字列	章題	::=	文字列	章題	::=	文字列
							|	図 文字列
							|	文字列 図
			段落群	::=	段落	段落群	::=	段落
				|	段落群 段落		|	段落群 段落
段落	::=	文字列	段落	::=	データ	段落	::=	データ
			データ	::=	文字列	データ	::=	文字列
				|	データ 文字列		|	図
							|	データ 文字列
							|	データ 図

浮動要素・複数回出現のない場合 **複数回出現のある場合** **浮動要素・複数回出現のある場合**

```
<!ELEMENT 章 - - (章題, 段落+) +(図)>
<!ELEMENT 章題 - - #PCDATA>
<!ELEMENT 段落 - - #PCDATA>
```

註: #PCDATAは文字列に相当する

DTDによる記述

図 2.6: 浮動要素のBNFとDTDによる記述

という記述、つまり“A”と“B”は順序はどちらでもよいが、両方とも一度ずつ出現するという条件は、“&”を使わずとも、

```
(A, B) | (B, A)
```

という形で表現できる。少なくとも関わる要素の数がそれほど多くなければ、このような書換えで解析する手間が著しく軽減されるので、SGMLの規格として改良されるか、専用のパーザとしてはこのように接続子・出現指示子を取捨するという選択も考慮の余地がある。

2.5.2 タグの省略

ELEMENT宣言において、要素名に続いては開始・終了タグの省略の可否が記述される。確かに省略されても構造は明白な場合もある。たとえば、箇条書きのための要素を仮定して、

```
<!ELEMENT itemize - - item+>
<!ELEMENT item - O #PCDATA>
```

というものがあった場合、

```
<itemize>
<item>条件1
<item>条件2
</itemize>
```

のような内容を記述するためには、要素“item”の終了タグ“</item>”は省略しても構造に曖昧性はなく、そういう状況下ではタグは少ない方が見ためも簡単であり、入力も楽である。

しかし、

```
<!ELEMENT  itemize  -  -  (item+, ref?)>
<!ELEMENT  item  -  O  (#PCDATA, ref?)>
<!ELEMENT  ref  -  -  #PCDATA>
```

とすると、

```
<itemize>
<item>条件1
<ref>参考文献</ref>
<item>条件2
</itemize>
```

のような場合が可能になり、“ref”要素が“itemize”直下なのか、“item”の下にあるのか、判別がつかない。

当然このような曖昧性がないように熟慮した上でDTDは設計されるが、このようなタグの省略で得られる手間あるいはデータの大きさの利得は、省略をなくしてしまった場合の処理の明解さより勝るとも言い切れないと見込まれる。

SGMLインスタンスを作成する場合、直にタグをつけながら文章を入力することは稀でなんらかのエディタを用いることが多い。つまり、文書の著述ではタグの入力はあまり考慮せず、論理構造上の位置を指定しつつ文章を書き込むようなインタフェースを介して作成する。こういう操作環境下では、もともとタグを付与する手間についてはあまり気に留めていないと思われる。

2.5.3 添加要素・排除要素

文書の構成要素のうち、出現箇所が通常のものに比べて可能性が多いものがある。たとえば、脚注は文中のあらゆる箇所に出現して、文章の注釈から著者の異動まで様々なケースに用いられるし、図はヘッダなどを除く文書の本体部分であればどこにでも出現することが予想される。

これをBNFなどで記述すると図2.6にもあるように、論理構造を定義する行数が格段に増えてしまう。そこで、ある要素のすぐ下位にくると固定的に定められるのではなく、この下であればどこにでも出現するという定義ができれば、論理構造の記述の簡便化には効果的である。そこでSGMLは、これらの浮動的な要素を、添加要素として記述法を定めた。

```
<!ELEMENT  Memo  -  -  ((To  &  From),  Body,  Close?)  +(Fn)>
```

ここで、下位要素を記した後半の“+(　)”の部分が添加要素の記述であり、ここに記したものは、その上位である要素“Memo”の配下であれば、他の下位要素に混ざって任意に出現することができる。

添加要素は他の要素と同様に、またその下に構造を持つことが許されるが、添加要素に指定されたモデルグループは、接続子が一つしか仕様が許されない。

同様に、ある要素の配下では出現を禁じられる排除要素というものも規定されている。

```
<!ELEMENT  P  -  O  (Text|Fn)+>
<!ELEMENT  Fn  -  -  (P+)  -(Fn)>
```

上記の定義で、“Fn” の下位要素要素を定めたモデルグループのうち、後半の “-(　)” にの部分が排除要素の記述である。この例では、“P” と “Fn” 再帰的に出現することを防ぐように定めることができている。

厳密に SGML のパーザとして動作するためには、これらの添加・排除要素についても文法的に違反がないかチェックをし、整形などの処理を行う必要がある。しかし、前節の、タグの省略や多種にわたる接続子・出現指定子も含めて、SGML で記述される論理構造は、文書の本体部の根幹としてイメージされる文脈自由文法のクラスと比較して複雑であり、厳密な解析のコストは高いことが容易にわかる。

そこで、本手法では、負荷を軽減するために、SGML 的な意味での厳密な論理構造のチェックや整形などの処理は組み込まずに、論理構造の変換はそれに専念して、インスタンスからインスタンスを出力し、その後のチェックや整形はその論理構造に付随するもの、または汎用の SGML パーザの利用を求めるものとする。

従って、入力となる文書は、その段階で論理構造に誤りがなく、またタグの省略もないものとする。

その上で、文書が取り得る論理構造を包含していればよいので、排除要素は考慮しないものとする。

第 3 章

本研究の課題

研究の意義や役割を明確にするために、この研究の課題が何で、どのような役に立つのか、あるいは、どのような前提や枠組で取り組まれているのかについて述べる。

この章の構成は以下の通り。

- 3.1: 論理構造の多様性
  ここまでに述べてきたような構造化文書を扱う際の困難な点を整理し、解決すべき問題を明確化する。
- 3.2: 想定するデータベース
  このような課題を解決し、そのようなデータをどのように使いたいのか、そのモデルとして、想定しているデータベースのモデルを提示する。

3.1 論理構造の多様性

本研究の課題を明らかにするために、ここで論理構造の差異について説明する。図 3.1の上には、論理構造が違うことの影響を示すサンプルとしてある論文の一ページのイメージを挙げてある。論理構造は収容する情報に最適なように設計されているはずで、論理構造の差異と誌面レイアウトの差異は完全には対応しないが、実際には互いに影響を及ぼしていることも多いと考えられる。

たとえば、左では著者の所属は著者ごとに記しているので、著者個別に所属を保持する論理構造になっているはずであるし、右では複数著者の所属を一ヶ所にまとめて表示しているので、複数著者の所属を一括して納める論理要素が用意されている可能性が高い。

このような誌面の差異を生んでいる可能性もある差異を持つ論理構造を図 3.1の下に示した。左がサンプルとして用いるために示した図 2.3の文書を木で表現したものであり、右が左の論理構造を変換した結果得たい文書を同様に木で表現したものである。本研究の目的は、このような変換を実現する手法を提案することである。

以下本論文では変換前のもとの文書やそれに関する要素などを「旧」、変換後の文書やそれに関するものを「新」と呼ぶ。図中で木構造のノードを構成するのは SGML でいう論理要素であり、そこに記してあるのは要素名であるが、要素名、たとえば "Recordlist" に続いて記してある数字 "1" は、同名の要素が複数回出現する場合に、各種類の中で各々を識別するために本研究での処理用に割り振った番号であり、要素名には含まれない。 たとえば旧文書で、木構造で表して論文のタイトルにあたるノードは "Title" であり、二つ出現するが、これらをそれぞれ "1,2" と表して、必要に応じて "Title の 1 つめ" "Title の 2 つめ" と区別する。

またこの例では、旧文書の要素名を大文字から、新文書の要素名を小文字から始まる文字列として、区別を明確化している。

図 3.1に示された論理構造の差異は以下のように分類できる。

- 要素の取捨: 旧文書で記述されている内容が、新文書では必要なく、該当する要素がない。
  たとえば、旧文書ではミドルネームのための要素 "Middlename"(*3*) を設けて、それにあたる情報 "Thomas" を保持しているが、新文書ではミドルネームについての情報は扱われない。このような情報は変換に当たって捨てられる。

  反対に、新要素で扱う情報が旧要素で扱わず、該当する要素がない場合も考えられる。その場合の新要素は、旧文書で要素が設定可能でも情報がないために（ミドルネームを持たない著者など）設定されない旧要素と同様に、旧要素から変換されるべき内容がないケースとして処理される。

- 要素名の違い:
  内容が対応する新旧要素の要素名が異っている。たとえば、旧要素 "Title"(*0*) と新要素 "subject" (*c*) は、同様に文書の「題名」のための要素だが、要素の名前が違う。SGML では要素名も、著者など論理要素の設計者が定めるため、内容に応じた名前をつけたとしても英語（大文字、小文字）か日本語か、またそれぞれに同義語の可能性もあり、また内容に応じない機械的な文字列である可能性もあり、自然言語など高度な技術を仮に導入したとしても、要素名からその要素が持つべき内容や意味を判断し、新旧文書の要素間で対応を定められることは期待できない。

- 論理構造の深さの違い:
  保持している内容が該当する新旧要素でも、論理構造上でより深いまたは浅い位置にある。

  たとえば、新旧文書中で同じく著者の所属を表す要素でも、旧要素 "Affiliation"(*4*) は論理構造のルートから、"Recordlist"-"Record"-"Author"-"Affiliation" という経路をたどって 3 つめ位置

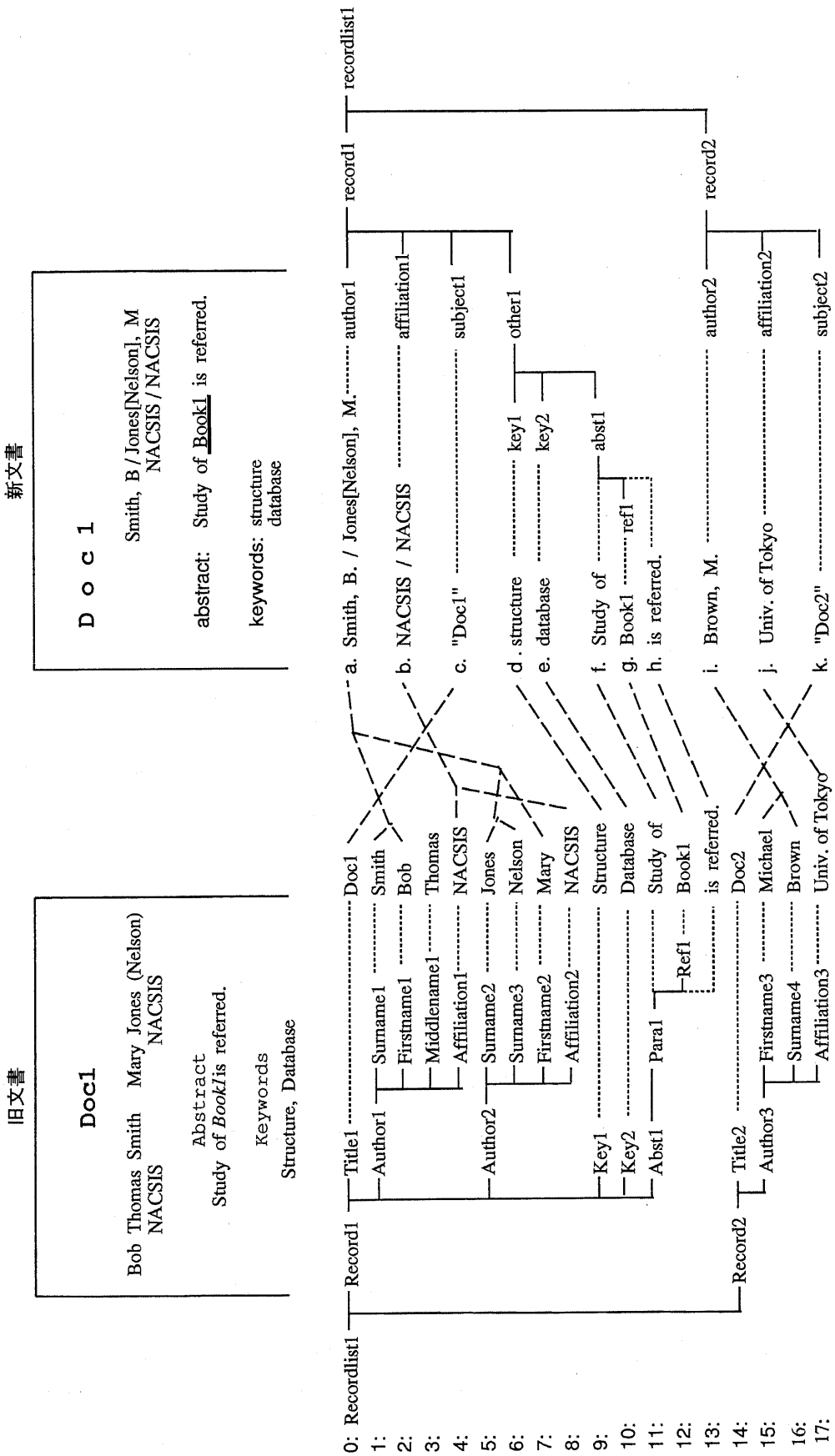


図 3.1: 論理構造の木表現と、論理構造間の差異

にあり、一方新要素 "affiliation" (*b*) は "recordlist"-"record"-"affiliation" とルートからの 2 つにあるので、"Affiliation" の方が "affiliation" よりも深い位置である。

反対にキーワードを持つための要素を見てみると、旧要素 "Key"(*9*) は "Recordlist"-"Record"-"Key" という経路で 2 つめにあり、新要素 "key"(*d*) は、"recordlist"-"record"-"other"-"key" という経路の 3 つめよりも浅い位置にある。

対応する新旧要素の論理構造上の位置が、深さもその要素までの経路上の中間ノードもすべて意味するものが同じであれば、旧要素のある中間ノードが切り替わったという状態を、新要素でもその対応するノードに反映させればよい。しかし、ノードの長さが違う可能性はあり、また数が同じであろうと異なっていようと、同じ順番にある新旧それぞれの中間ノードが持つ意味が異なっている可能性があるので、単純に同じ順番にあるノードを互いに対応させるだけでは不十分である。

- 一つの旧文書中の、同一位置にある同じ要素が複数回の出現:
  たとえば、旧要素 "Affiliation"(*4* と *8*) は、論理構造上で "Recordlist"-"Record"-"Author"-"Affiliation" のように同じ位置にあり、複数の著者の所属を個別に表すために複数回出現している。

  このような場合に、どのように新要素を構成するかについては、次の二通りを考慮する必要がある。

  - 各々別の新要素に該当させる:
    一つの旧要素から、一つの新要素を構成する。
    たとえば、旧要素 "Key"(*9* と *10*) はそれぞれ別々の新要素 "key"(*d* と *e*) にする。
  - 一つの新要素にまとめる:
    同じ位置にある旧要素は複数回出現しても、新要素の方は出現回数が一回のみと定義されている場合、旧要素 "Affiliation"(*4* と *8*) は一つの新要素 "affiliation"(*b*) にする。このようなケースを**繰り返し**と呼ぶ。
    さらにこのような場合、まとめてよい範囲と、別にしておく範囲とがあり、これを区別する必要もある。たとえば、同じ文書の著者の所属はまとめるが、違う文書の著者の所属は別の要素に切り離す、といった指定である。この場合は、要素が出現する文脈的な情報を使わなくては判断できない。

- 新旧要素の 1 対多の対応:
  二つ以上の旧要素を組み合わせて、一つの新要素を構成する。

  たとえば、旧文書では著者の名前が苗字と名前に分けて記録されていて、それぞれ "Surname"(*1*) と "Firstname"(*2*) という要素としてあったが、新文書では一つで姓名を表す新要素 "author"(*a*) として表している。このようなケースを**組み合わせ**と呼ぶ。

  このような場合も、同じ著者の苗字と名前は組み合わせるが、違う著者のものと混ぜて組み合わせることは望まれない。そのような判断でも、出現場所に関する文脈的情報を考慮する必要がある。

さらに、これらの論理構造の変換に伴って、

- データ形式の変換:
  組み合わせた場合や、繰り返しをまとめた場合は、どのような形式で複数の旧要素に由来するデータを統合するのか。そのような場合でなくても、データの形式は、カッコなど修飾的文字を添えたり、大文字・小文字を揃えるなどの処理が必要である。たとえば、例では旧文書の著者名が姓と名の別々な要素であったものを、新文書では一つにまとめて、"姓，名前の頭文字" という形式にしている (*15,16* から *i*)。

さらにデータの処理では自然言語的な処理も考えられ、たとえば中にある数値データの単位を変えたり、問い合わせがあったらそれに答えたり、と高度な機能も組み込めるだろう。しかしそれらを取り入れることは、徒らに処理を複雑にする虞があるので、処理の枠組を明確するために、ここではたかだか簡単な関数を起動する程度、基本的には文字列を置換するような、文字列的な処理を対象とすることにする。

に関する処理も必要となる。

提案する手法では、これらの差異を吸収して論理構造を変換することが目的となる。この手法を採用することにより、たとえば、データベースは様々な論理構造を持つ多くの学会などの文書を、一つのインタフェースや処理アプリケーションによってまとめて扱えることになる。

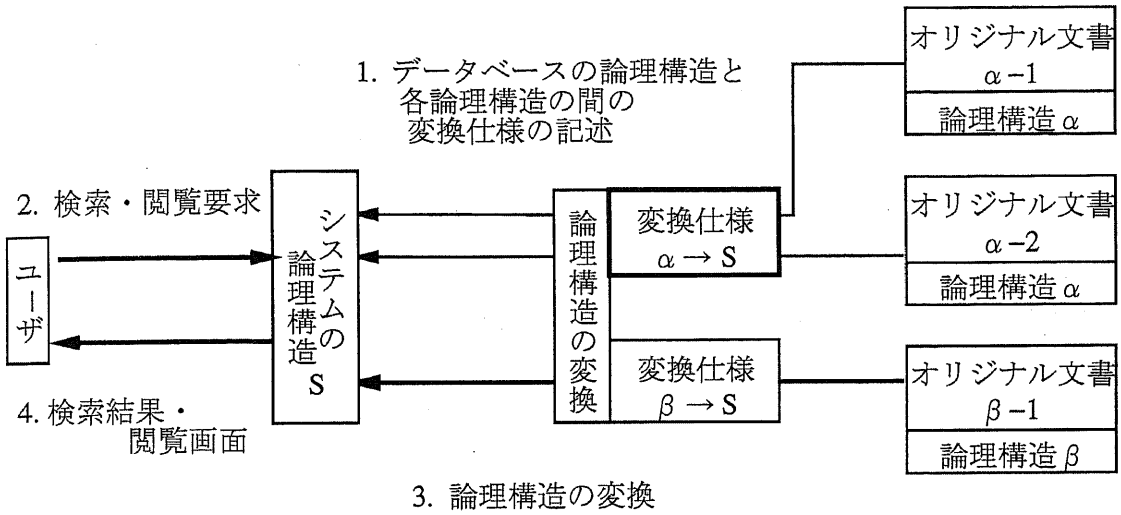


図 3.2: 提案するデータベース

3.2 想定するデータベース

本研究で検討しているような論理構造変換手法は、論理構造の「変換」だけでなくSGMLで記述された構造化文書に対する、一般的な論理構造の処理に適用できる。たとえば、章や節のタイトルだけを抜き出して目次に当たるビューを作成したり、二次情報を抽出したり、また論理構造を利用した検索などである。

本研究で想定しているデータベースは、これら論理構造を活用した機能を統合したようなものであり、論理構造を用いた様々な検索や閲覧など文書利用活動を支援する。このようなデータベースでは、ユーザが、求める情報をより簡単かつ正確に入手し、またより速くかつ深く文書を理解できるかという観点で、様々な検索や閲覧の機能、またビューや画面のインタフェースなどに多種多様な処理が必要とされるが、これらを実現するためには、論理構造を変換する処理はその核心をなす一つであり、これがなければ高度な機能の実現は難しい。

ここで言う、SGMLによる多種多様な論理構造を持つ科学技術論文を扱うためのデータベースとは、論理構造の変換の使われ方に関して、図3.2のようなものである。このデータベースの動作は次の通りである。

1. 各論理構造とデータベースの用意した論理構造との間で、論理構造の変換仕様を予め定める。
   データベースの準備作業として、予め、文書の著者・論理構造の設計者か、データベースの管理者が、各論理構造とシステムの論理構造の間で一度、変換変換仕様を定める。

2. データベースは、自身の論理構造に基づいた検索インタフェースなどをユーザに提示し、ユーザはそれをビューとして用いて検索や文書の閲覧を要求する。
   システムの論理構造や、それが持つ要素の名前で検索画面を提示し、条件が入力される。

3. データベースは各文書の論理構造をデータベースシステムのものに変換し、検索などを処理する。

4. 検索結果や、指定された閲覧誌面は、データベースの論理構造に基づいた形式に整形してユーザに提示する。
   結果が二次情報・目次ビュー・全誌面などいろいろな形式で見られる他、参照の活用など高度な閲覧機能へも移行できる。

このシステムの特徴は二点挙げられる。

- データベースも独自の論理構造を使用できる。
  データベースもその目的とする内容を表現するのに適した論理構造を持ち、あるいは自らが便利だと考える主張を表現できる個性を持つ余地があることが望ましい。

  システムに論理構造を持たせる目的では現在、HTML や、LaTeX などが多く用いられ、SGML 文書からそれらへ変換するツールなども開発されている。しかし、前者は一個の固定的な論理構造であるために柔軟性に欠け、個々のシステムの要望を実現できず、後者は緩い論理構造とむしろレイアウトを記述する指向を持つ言語であって、上記の目的には十分であるとは言えない。

- 文書は個々のオリジナルな形式のまま蓄積する。
  データベースは、検索や閲覧の機能の一環として、論理構造を解析したり（パーサ）、表示用に整形したりする、独自の処理系を用意するだろう。しかしこの他に、文書のオリジナルな各論理構造にも同様の処理系があったと推定されるので、元の文書の誌面を見慣れたユーザからの要望により、その付随する処理系で処理したオリジナルな出力を必要とする場合も考えられる。

  さらに、あるいは将来また違った処理系を作るかもしれないし、またユーザが個々の好みの処理系を用意する、複数のインタフェースを備えたデータベース、あるいは複数のデータベースによるデータの共有など、複数の処理系やインタフェースに柔軟に対応できると有利な局面も多く考えられる。この要望を満たすためには、オリジナルな論理構造から予め静的にシステムのものに変換して蓄積しておくと、他の論理構造に変換する場合にはシステムのものにしておいた処理が無駄であるし、オリジナルな処理系に対応するためには、これらの処理自体が無用である。

  その一方で、多量の文書データを扱うためには、なるべく記憶容量の節減も図らなくてはならない。上記のような事情に対処するために、それぞれの構造に変換して蓄積しておく方法は、求められる種類がカバーしきれず、またデータ量の点でも得策とは言えない。

  そこで、各文書はもとの論理構造のまま保存しておいて、ユーザからの指定に基づく検索など処理の際に、必要になった文書を変換するような、動的な処理が有効である。

本研究では、このようなシステムの核心となる、SGML で記述された科学技術論文の論理構造を変換する手法を提案する。

第 4 章

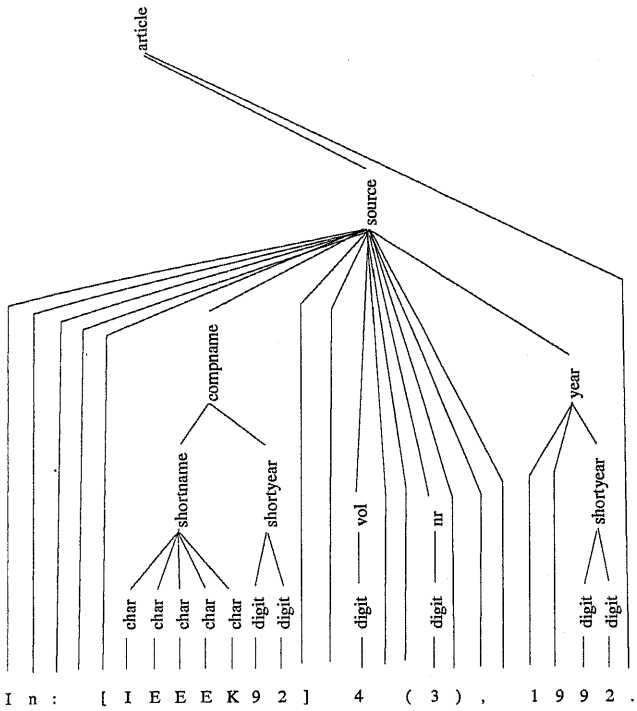
関連研究 1: 構造化文書の処理

論理構造やSGMLを扱う手法については、エディタや整形なども含めて多くの研究がなされている。特に文書の論理構造を造を扱う手法に限っても、従来は、文書の論理構造が文脈自由文法などに近いため、体系が整備された文法的な手法を活用する手法や、SGMLの整った体系である記法に準拠した方法が行われてきた。

ここではそれらの処理について、手法を紹介するとともに、得失を簡単にまとめる。そこから、その結果として挙げられるようになった課題を明らかにし、本研究で提案する手法と対比させ、妥当性を示す。

この章の構成は以下の通り。

- 4.1: 論理構造を文法的に扱う手法
  文書が構造化されていることを活かして、論理構造の木を木として扱う手法を紹介する。
- 4.2: SGML固有の手法
  SGMLの論理構造の変換手法を大きく分類するとともに、SGMLの記法に則って、そこへ論理構造の変換を組み込もうとする手法や、専用の言語を設計して変換を行おうとする手法を紹介する。



source := 'IN: ' compname ' ' vol '(' nr '), ' yea
compname:= '[' shortname shortyear ']'
nr := digit+
vol := digit+
shortname := char+
shortyear := digit digit
year := '19' shortyear

図 4.1: p-string（部分）と、対応する部分の構造の記述

4.1 論理構造を文法的に扱う手法

論理構造を扱う手法については、編集や整形なども含めて多くの研究が行われている。ここでは特に論理構造の変換についての研究をいくつか紹介する。

4.1.1 高度に構造化された文書の変換

文書の論理構造を、どのようなデータモデルで表現し、また操作するかという課題は、問い合わせや表示の処理の基礎となるだけでなく、多様な構造を持つ文書間の変換に対処するためにも必要となる。

OED (Oxford English Dictionary) を新版に変換する処理に関して、辞書の記述をツリー構造で表現し、機械的に処理できるようにしたのが、p-string である [11][12][13][14]。この手法では、対象が辞書の記述という高度に整形された構造を持っている利点も活用して、論理構造を表現し、検索や、表示・形式変換のための処理のオペレータを考案している。

OED の内容は、見出し語の下に発音記号・言葉の由来などと並んで意味が記述されている。ここで意味は、細かさに応じて幾層ものレベルに分けられており、また意味の歴史的変遷や、文学作品の中の使用例なども添えられている。これらを文字レベルまでパーズしてツリーで表したのが p-string (parsed string) であり、その部分例は図 4.1の通りである。

p-string では、論理構造は文法的に記述されている。この記述では、図 4.1のように BNF に類似する記法で、要素名を左辺に記述し、その要素の下部に当たる要素名や文字が、右辺に、出現する順序通りに、回数指定子なども用いて定義されている。

図の論理構造が文書型 'atrticle' の一部で、この部分に対応する文字列 "In: [IEEEK92] 4 (3), 1992." を

<article><source>In: [<compname><shortname>
IEEEK</shortname><shortyear>92</shortyear>]
<vl>4</vl> (<nr>3</nr>, <year>19<shoryear>92
</shortyear></source>.</article>

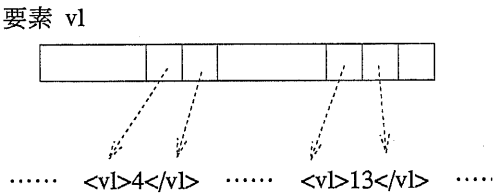


図 4.2: p-string の内部表現と要素索引

パーズする場合には、文字列をパーズしてツリー表現を得るオペレータ 'parsed-by' を用いて、

'In: [IEEEK92] 4 (3), 1992.' *parsed-by* article

を実行することで、図 4.1のような p-string が結果として返される。このほかにも、オペレータとして、

- reparsed-by
  新たに記述された構造に従って、文字列を再パーズする
- children
  指定された要素の下部ツリーを返す
- size
  指定された要素の下部ツリーの大きさを返す
- every ... in
  指定された位置の下部にある、指定された論理要素をすべて返す
- suppressing
  指定された要素を取り除いた形で、下部ツリーを返す
- where
  条件に合わない部分のツリーを削除した形で、下部ツリーを返す

など、ツリーを直接操作するものが考案されている。

これらを用いて、辞書の記述形式を変換する。例えば、元の OED にある記述で、新版には必要なければ、'surpressing' を用い、記述の順序や論理構造を変えるのであれば、"reparsed-by（新しい構造)" といった形で処理を行う。

内部的には、p-string は、文字列を要素名で前後を挟んだ表現をされており、また要素の種類ごとに、出現する位置の一覧をリストとして予め用意しておき、ツリーへのアクセスを助けている（図 4.2)。

この手法は、ツリーを直接操作することで、論理構造を活用しており、その意味では新しく、また意義深い。しかし、さらに実際のデータベースなどへの応用を考えた場合は、ユーザがこの操作にどのように関わってくるのか、入力方法や、ユーザの興味の反映させ方などの方面での応用方の研究が必要になってくる。

またこの手法は、一般の文書のような複雑な構造を持つものに対して、柔軟な処理を行うためには、十分ではない。

4.2 SGML 固有の手法

SGML の記述法に則って文書を操作する手法を分類すると、

- マッピング
  元の文書のある要素と新しい文書のある要素を対応付ける手法。
- 汎用言語の適用
  - イベントドリブン
    インスタンスに着目し、インスタンス中にタグや要素が出現するイベントに対して、アクションを定義する手法。
  - 文法ドリブン
    DTD に着目し、論理構造の定義に関連付けてアクションを定義する手法。
- 固有の言語

になる。それぞれについて例と得失を簡潔にまとめる。

4.2.1 マッピング: 1 対 1 変換

Warmer ら [15] は、各々のタグについて動作を定義するというマッピングの手法により、形式の変換を行った。たとえば、SGML 文書によるメモ形式を整形するという目的に応用できる。

この方法では、まず SGML のタグと整形時に出力する文字列の対応を定める。

```
<memo>          ``Memorandum: \n\n''
<sender>        ``From: ''
<receiver>      ``To: ''
</person>       ``    ''
</forename>     ``  ''
<body>          ``\n''
```

(ここで "\n" は改行の制御コード) 対応がこのような形式であれば、文字列データをそのまま出力する他に、

- "<sender>" が出現した時には区切り子 "From: " を出力
- "</person>" が出現した時には空白を二つ出力する

という動作が定義されたことになる。

ここへ入力する文書が、

```
<memo><sender><person>
<forename>Bob</forename>
<surname>Smith</surname></person></sender>
<receiver><person><forename>Mary</forename>
<surname>Jones</surname></person>
<person><forename>michael</forename>
<surname>Brown</surname></person></receiver>
<body>Hello</body></memo>
```

というものであれば、以下のような出力を得られる。

```
Memorandum:
From:  Bob  Smith
To: Mary  Jones    Michael  Brown
Hello
```

同様な手法、つまり要素やタグの出現に応じて動作を定義するというものは Prices ら [16] 他いくつか見られるが、この方法ではシンプルで実装しやすいこと、使う人も訓練がいらないことなど利点もあるが、欠点として、一言でいえば変換後の論理構造も、元と同じに留まるということが挙げられる。具体的には、

- 要素の出現順を変えたい
- 要素が出現する位置の上下関係が利用できない
  (例) 箇条書の項目は、番号つき箇条書と番号なし箇条書の両方の下に現れるとして、それぞれの位置によって整形処理が異なる。(番号つきであれば数字を振る、番号なしであれば黒丸を置く。)
- 同じ種類の要素が複数回出現する時に、その一つめか最後のものか、わからない。
- 文字列の置換に留まらない、なんらかのアクションを組み込みたい
  (例) データの変形、スペル/形式チェック
- 要素の取捨選択を行いたい
- 複数の要素を組み合わせたい

といった処理に対する要求は実現が難しく、課題が残る。

4.2.2 汎用言語の適用 1 (イベントドリブン): インスタンス対して関数を定義する変換

前項で、様々な処理が行えなかった不満を解消するために、タグに対して、既に用意した関数を定義し、これを起動することでさまざまなアクションを実現した。この関数は C 言語などの使い慣れたもので自由に作ればよく、採用する際の敷居は比較的低い。

この方式でのアクションの定義は下のようになる。この表では、開始タグ・終了タグ・中身のデータに対してそれぞれ適用する関数名を定めている。この関数では、それぞれのタイミングで付与したい接頭語や、データの変換をライブラリとして記述しておく。

この関数間でデータをやりとりすることにより、順番の並べ替えや、組み合わせの処理などが実現できる。つまり、先ほどの例の入力に対して、

```
From:  Smith, B.
```

というような変換結果を出力することも可能である。

4.2.3 汎用言語の適用 2 (グラマードリブン): DTD を利用する変換

Warmer ら [17] はまた、SGML 文書の論理構造の変換仕様を、DTD の論理構造中に処理を書き込むという手法により行った。この手法でもアクションは C 言語で記述された関数で、起動したい関数名を DTD での要素の定義に、その要素の出現前・出現時・出現後にわけて規定する。たとえば、もとの DTD と、そこにアクションを書き加えた仕様は、図 4.3のようになる。

表 4.1: タグに対して関数を定義する例

要素名	開始タグに対する関数	終了タグに対する関数	データに対する関数
"sender"	startsender	endsender	contentsender
"receiver"	startreceiver	endreceiver	contentreceiver
"body"	startbody	defaultend	defaultcontent
"person"	defaultstart	defaultend	defaultcontent
"memo"	defaultstart	defaultend	defaultcontent
"forename"	startforename	endforename	contentforename
"surname"	startsurname	endsurname	contentsurname

```
<!ELEMENT  chapter  0  0  (heading,  pp*,  appendix)>

<!ELEMENT  chapter  0  0  (heading,
                            {
                              /*  pp の出現前にする処理  */
                            }
                          pp(  /*  pp の処理に必要な変数  */)
                            {
                              /*  pp の出現時にする処理  */
                            }
                            {
                              /*  pp の出現後にする処理  */
                            }
                            {
                              /*  appendix の出現前にする処理  */
                            }
                          appendix)>
```

図 4.3: DTD に書き込む、グラマードリブンな変換手法

汎用言語を用いる手法をまとめると、イベントドリブンとグラマードリブンの裏表である得失は、以下のように言える。

- アクションの関数の記述が、慣れた言語で行え、容易である。
- イベントドリブン方式は、仕様の記述が容易である。
- イベントドリブン方式はDTDの変更に伴う仕様の更新が容易である。
- グラマードリブン方式は、要素の上下関係・同列関係など文脈などの論理構造を活かして仕様を記述できる。
- グラマードリブン方式には、DTDにそって文書を解析するのは負荷が大きい（第2.5章）。

簡潔にまとめれば、イベントドリブン方式は利用が容易だが能力が限られ、グラマードリブン方式はさらに高い能力を持つが解析が困難である。両者の良い点を取り入れることが求められる。

4.2.4 専用の変換仕様記述言語

高橋ら [18] は、論理構造の変換仕様を記述する専用の言語を提案している。この手法では、論理構造を解析するパーサの後処理系として、省略タグや実体参照などを補完したデータに対して、深さ優先でトラバースする動作を基本とし、その途上で、必要なノードに対して必要な処理を規定したものを変換仕様として、この仕様を記述するための言語を検討した。特徴は以下のようである。

- 処理を行うノードを選択するために、ノードの種別・属性と位置、つまり文脈情報を条件とできる。
- 処理スクリプトは
  - ノードを対象とするスクリプト
    条件を満たすノードを検索、下位ノードを順序の入れ換え・追加・削除など木構造そのものを変更する機能
  - データを対象とするスクリプト

  の二通りがあり、それぞれについて、
  - 対象ノードの先頭（開始タグ）での処理: プロローグ
  - 下位ノードへ移行するかどうか
  - 対象ノードの末尾（終了タグ）での処理: エピローグ

  の三つの指定を記述する。
- 複雑な処理では、プロセス間でデータの受渡しを行うパイプライン処理を取り入れる。

具体的には、図4.4のようになる。この例では、SGMLで脚注として参照ポインタと本体にわかれていたものを、LaTeX式に変換するために、木構造とデータの形式を変換している。

この手法によれば、プログラミング言語と同様に習熟次第では高度な変換などが実現できる。しかし反面、やはり同様に、使用に当たってはそれなりの訓練が必要である。

```
process  構造変換        #木構造に対する処理
{
        script  ELEMENT  FNREF        #脚注参照の置き換え
        {
                prolog{
                        var  fn_body;
                        #参照先の脚注本体を検索
                        fn_body ::= id_sarch(get_attr(self,  @REFID));
                        #脚注本体を親から切り離す
                        del_children(parent(fn_body),  fn_body);
                        #脚注参照を脚注本体で置き換える
                        repla_child(parent(self),  self,  fn_body);
                }
          }
}

process  LaTeX データ出力  #データに対する処理
{
        ...
        #（文章中に埋め込まれた）脚注本体の処理
        script  ELEMENT  FNBODY
        {
                #``\footnote{...}'' コマンドの出力
                prolog{puts(``\\footnote{'');}
                epilog{puts(``}'');}
         }
         ...

         #文字データ内容の変換と出力
         script  CDATA
         {
                #特殊文字の置き換えを行いつつ文字データ内容を出力する
                puts(conv_string(self,  latex_conv_table));
         }
}
```

図 4.4: 論理構造変換仕様記述言語 “AEsop” の例

4.3 まとめ

文書は、その論理構造の根幹部分は文脈自由文法に非常に近いため、文書を文法の手法に則って扱おうとする試みは多く見られる。しかし、このアプローチには二つの点で限界が見られる。

一つは、文書それ自体が文脈自由文法ではうまく記述のできない構造を持つ点である。SGML では、出現回数や出現順、また添加要素については出現場所も大幅に自由を持たせた記述としている。これを BNF で記述しようとすると、無理ではないが、記述に要する行数が激しく大きくなってしまい、実用的ではない。

もう一点は、論理構造の変換に関する課題である。論理要素が 1 対 1 で対応している場合はともかくとして、組み合わせがあるような場合の対処が難しい。可能性としては、文書中の任意の場所に出現した要素同志を組み合わせることも考慮しなくてはいけない。つまり、組み合わせたい要素が互い違いになっていたり、一組が遠く離れているために、その間に別の組み合わせが生じたりする。

これを木構造を意識して記述しようとすると、究極的には、新旧要素間の対応を、文書全体としてみた場合の影響さえも考慮して、矛盾のないように記述する必要につながる。この作業は高度に複雑で、容易ではない。

ここまでに見てきたものをこれらの課題の観点から考えると、木構造のまま処理しようとするアプローチや yacc (yet another compiler compiler) [19] のような文脈自由文法の形式で記述するアプローチでは変換仕様の記述が困難であるか、対象とできる処理が限られてしまう。一方、専用の言語を採用するようなアプローチでは、行える処理は高度であるがそれを行うための記述が困難である。

そこで本研究では、これらのアプローチの優れた点を踏まえながら、できるだけシンプルな記述法でなるべく高度な処理を実現できるような手法を目指すことにする。

第 5 章

関連研究 2: 電子図書館

電子化した文書を集積してデータベースとして扱うシステムは、近年のコンピュータネットワークの普及と密接に関連して、ますます注目を集めてきた。このような、電子化した文書を、検索や閲覧できて、さらにネットワークを経由してアクセスできるものは、特に電子図書館 (Electronic / Digital Library) と称されることが多い。

電子図書館では、画像などマルチメディアを対象としたものも盛んだが、その場合は、マルチメディアデータのインデクス付けや検索方法、転送に関してのコストなどに着目することが多い。

一方で、文書と言った場合には依然テキストデータが基本となる。電子図書館で主に取り上げられる課題は、現在のところデータベースそのものの課題だけでなく、電子図書館固有のものであることもあるが、構造化文書をどのように扱うかという観点で、様々行われている実験や検討は参考とすべき点が多い。そこでここでは、この電子図書館に関して、特に文書に関連するものを取り上げる。

この章の構成は以下の通り。

- 5.1: 大規模プロジェクトの動向
  欧米では、政府や複数の大学・出版社が共同して電子図書館の実験を進めている。そのような中から紹介する。

- 5.2: テキストを主に扱うデータベースの課題
  テキストをデータベースとして扱う場合に問題となる点をまとめて、それぞれに取り組んでいる解決例を紹介する。

- 5.3: Ariadne
  検索や閲覧など多様な高度機能を備えた電子図書館の総合的実験の一つであるシステムを紹介し、本研究が前提としているような将来型の全文データベースの例のイメージを持つことを助ける。

表 5.1: 主な電子図書館プロジェクト（科学技術・文書関連）

団体名	システム名	収容データ			
		イメージ	OCR	ASCII	SGML
Carnegie Mellon University (CMU)	LIS-II [1]	○		書誌	
University of California, San Francisco (UCSF)	RedSage	○			
De Montfort University	ELINOR [2]	○	○		
Cornell University	NCSTRL [3]	○		○	HTML
Elsevier Scinence	TULIP [4]				
University of Michigan	UMDL	○	○	書誌	
University of Illinois, Urbana-Champaign (UIUC)	InterSpace				○
Cornell University	CORE [5]	○		要旨	○
AT&T Bell Lab.	RightPages	○	○	書誌	
学術情報センター	NACSIS-ELS	○		書誌	
電子図書館研究会	Ariadne [6]	表紙など			HTML
奈良先端科学技術大学院大学	Mandala	○	○		

1: Library Information System - II
2: Electronic Library and Information Online Retrieval
3: Networked Computer Science Technical Reports Library
4: The University Licensing Program
5: Chemical Online Retrieval Experiment
6: Advanced Retriever for Information And Documents in the Network Environment

5.1 大規模プロジェクトの動向

電子図書館の試行実験は、文書を中心とし、科学技術論文を扱うものに限っても、主なもので表5.1のものなどが挙げられる [20] [21] [22] [23]。

現在行われている実験システムでは、

- 基本的に誌面をスキャンしたイメージデータを提供する
- 全体または部分的にOCR（Optical Character Recognition、画像データから、もしくは光学的に画像を読みとるスキャナを通した文字認識処理）によりテキスト化し、その範囲で検索を可能にしている

という機能を持つものが多い。OCRの正解率は99%程度に達しているもののまだ誤りはあり、検索結果にも影響がある。

これに対しては、

- 画像処理技術の改善
  ノイズや取り込んだ誌面の傾きなどを修正する
- 認識技術の改善
  判断すべき文字として予め保持する文字の種類を増やす

などによるOCRの正解率を上げる他に、

- 検索手法の改善
  認識を誤りやすい文字の対照表を作るなど、認識誤りを考慮して曖昧性を取り入れる

により、より良い結果を返す（検索結果に不必要なデータが少ない、検索結果に漏れるデータが少ない）ような工夫もされている。

また、書誌情報など限られた部分の情報をテキストデータとして作成することで 検索などをより正確に処理する方法も多く取り入れられている。

これらの実験システムに一般的な機能あるいは操作手順は、

- キーワードの AND / OR による論理演算
  書誌情報あるいは一次情報の一部または全部など、そのシステムがテキストとして持つ範囲で検索する

- 必要な結果を画面に表示
  検索処理にヒットしたもの、あるいは指定された雑誌の、適当なページをイメージとして画面に表示する

というシナリオである。また、電子図書館の特徴として、ネットワークを介した分散化・リモートアクセスも共通の興味課題となっており、現在では広く普及したWWW (World Wide Web) ブラウザをインタフェースとして用いるケースが一般的である。これにより、特別な設定をしない特定の学内ユーザ、さらには一般の世界中のユーザが、この目的以外のネットサーフィンと同様の環境と動作で、気軽に実験に参加し、意見などをフィードバックすることを促している。

研究者あるいは技術者からの視点に偏らない、ユーザの意見を取り入れながら開発を進めるために、WWWを活用する方式はインターネットの利点を正に活用したものと言え、どのような意見が寄せられるのかは興味深いところである。プロジェクトの中では、実験結果のみならず、進捗状況や途中の報告も随時WWW上のページに掲載するところもあり、その情報公開の進め方としても意義深く、大きくアピールしている。

その反面、インタフェース・通信方法として既存のWWWを用いることに伴い、WWW上で実現されている処理や表示などに機能が限定されてしまう点も否めない。従って、専用のプロトコルやブラウザを利用する形式についても検討が進められることも欠かせない。

複数の大学や企業が共同して、あるいは政府などが援助して、大規模に行われているプロジェクトが欧米や日本を中心にいくつも進められているが、このうち二つについて、紹介する。

5.1.1 TULIP とミシガン大学

科学技術論文の出版で世界的大手のElsevier Science社は、9つの大学と共同でTULIP (The University Licensing Program) を行った [24]。このプロジェクトでは、Elsevier社などの科学技術論文43種類について、誌面のスキャンイメージ（解像度 *300dpi*）と、書誌情報および本文のイメージをOCR処理したプレインテキストデータを作成し、協力する大学に提供した。ユーザの電子図書館の利用について、技術面・ユーザの行動面・経済面の三点について考察を進めることを目的として、1991年に始まり1995年に終了した。

TULIPプロジェクトの総括として、技術面で得られた見解は、

- この方式での電子図書館では、ネットワークの帯域がネックである

- しかし、紙に印刷するに耐える品質にするのは難しい

などとしている。将来はプレインテキストデータを編集して、SGML を採用したいと述べている。

プロジェクトに参加した 9 大学の一つ、ミシガン大学ではこのデータを用いて UMDL (University of Michigan Digital Library) を開発しており、検索インタフェースのデモを WWW 上で公開している [25] [26]。このデモでは、雑誌名を選ぶとその表紙が、さらに巻号を選ぶとその目次が表示され、目次からは要旨あるいは本文にそれぞれリンクが張ってあって呼び出せる他、選択の各段階で検索を実行できる。この検索では、全文・要旨・論文タイトル・著者名・雑誌名の 5 つの項目に対して検索語を自由に入力する。

UMDL ではネットワーク分散化を計画しており、システムにはエージェント制を取り入れ、そのための知的なエージェントも開発している。エージェントは、

- ユーザの検索意図を導き出すユーザインタフェースエージェント
- ネットワーク上で検索を行う仲介エージェント
- 自分の管理する資源で検索を行い結果を返すデータコレクションインタフェースエージェント

の三つの段階でシステムの設計を行っている。

5.1.2 DLI とイリノイ大学

DLI (Digital Library Initiative) [27] は、NII (National Information Infrastructure) 構想に伴って、情報を国民に届ける研究の一環として、7 つの大学が文書・マルチメディアなど各種データベースの構築や検索、またネットワーク上の分散データベースの統合などについて研究するプロジェクトである。NASA (National Aeronautics and Space Administration) など政府機関が支援している。7 大学の一つ、イリノイ大学のプロジェクト InterSpace は、IEEE (The Institution of Electrical and Electronics Engineers) などから提供された論文の SGML データを対象とした分散大規模データベースについて検討している [28] [29]。

このシステムの検索インターフェースは、WWW ブラウザ上に、

- 収容論文の題のシソーラス（タイトルシソーラス）
  タイトルの表現する意味間の近さを可視化して図示する
- 用語の共起リスト
  用語が同一文書中に同時（一定の語間以内）に出現する状況の索引
- 全文検索

の機能が備えられ、これら用いて検索する。タイトルシソーラスは、文書の内容とデータベースについて熟知した司書が人手で作成し、共起リストは機械的に作成する。

将来的には、共起リストを活用して、「語彙切り替え」の機能を追加することを計画している。語彙切り替えとは、同じ意味を表す用語を、異なる分野間で関連づける機能であり、不慣れな分野での検索を支援する。たとえば、橋の建設に関して「長い構造物に関する流体力学」を調べているとき、海底ケーブルの「長い構造物に関する流体力学」を参考にしたいと思えば、橋の建設に関する論文と、海底ケーブルに関する論文でそれぞれ「流体力学」と共に頻繁に出現する用語を示すことで、求める意味を海底ケーブル分野の用語で表現することができる。

語彙切り替えのプロトタイプも含めて公開中である。

5.2 テキストを主に扱うデータベースの課題

誌面のスキャンイメージとテキストデータを用いるデータベースの優劣は、この種のシステムを評価する場合常に話題となる。イメージを使う場合は、

- 既に親しんでいる誌面を表示できる
  長年講読を続けている場合、内容把握の容易さという点でユーザに優しい。また、紙に印刷するために培ってきた、細やかなレイアウトや個性的なフォントを損なうことがない。図表の表示についても、従来のポリシーやノウハウで統一した誌面がそのまま再現できる。

- これまでに紙により出版された巻を同様の処理で扱える
  従来の図書館に格納されている文献も、電子化してデータベースのデータに加えることができる。

という利点がある。また、

- 多言語化などに際しても、特別なフォントや設定が必要ない
  言語の違いを意識せず、あるいは特殊文字を多用する分野の文書も容易に格納・利用できる

という点も、これからますます国際化が進み、また対象とする範囲を越えてデータベースがデータ量を増やせるため、重要である。

一方で、テキストを主とすれば、

- 検索ができる

- 表示画面に合わせやすい
  画面の大きさや形、字体、解像度など、ユーザがその時使っている環境や要求に応じて動的に整形できる。イメージデータの場合、これらが固定なため、画面からはみ出したものをスクロールしたり、全体を縮小したりといった不便が伴う。

- 蓄積・送信するデータ量が少ない

などの利点がある。

表示に関しては、究極的には、どちらが「読みやすいか」といった嗜好の問題でもあり、決定的な結論を導くのは難しいと考えられる。しかし、少なくとも今後出版されるものに関しては、これら両方へ適用することのできることも含めて、テキストが全体として有利であり、電子的なテキストデータは、作成される文書中の割合としても、従って蓄積されるデータの量としても、飛躍的に増えていくことが予想される。

従って、イメージを主に扱うシステムの中でも、テキストを何らかの形で扱う、あるいは将来的にはテキストを主に扱うと計画するものも多い。しかし一方で、テキストを主に扱うことによる課題には新たに取り組まなくてはならない。

ここではこれらの、テキストを扱う場合の問題と、それの解決例として取り組まれているシステムについて述べる。

5.2.1 図表の扱い：化学論文での実験

テキストを対象とする場合、図表の扱いが課題となる。具体的には

- レイアウト的に誌面あるいは画面のどこに置くと判断するか
  文書中に混在させるか、文末にまとめたり、別画面を作ったりするか。混在させる場合、文章の理解を妨げない、さらに助けとなる位置とはどこか。

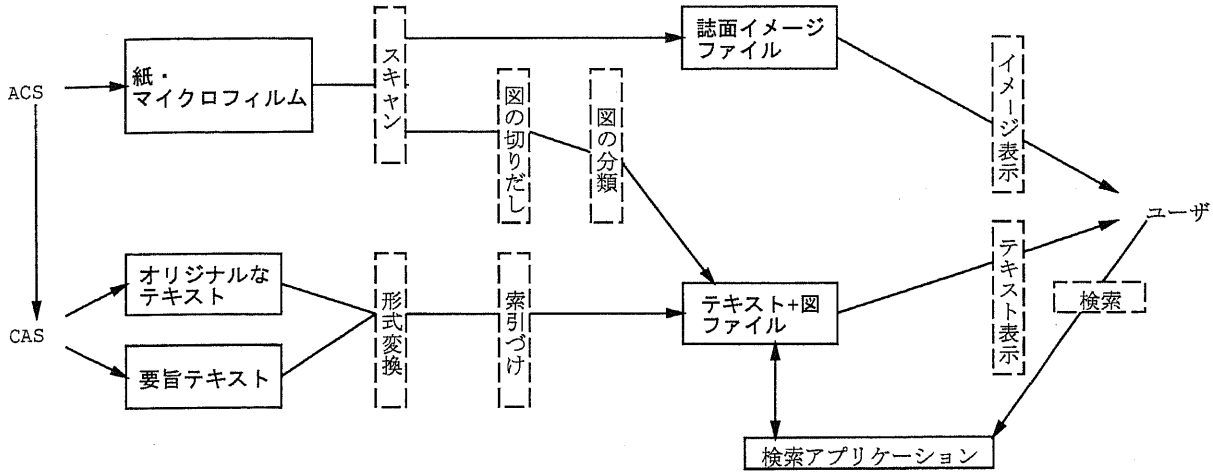


図 5.1: CORE におけるデータの流れ

- どのようなタイミングで表示するか
  はじめから常に表示するか、参照等のみを表示して、要求があった時に表示するか。

- どのデータ形式で持つか
  データの大きさ、処理の効率などをどう判断するか。

などの点である。

特に化学関係の学会は、図表の扱いの影響が大きい。化学式を多用するという文書の特徴によって論文中に現れる図表の割合や意味合いが、他の分野に比べて高いからである。

CORE (Chemical Online Retrieval Experiment) [30] [31] [32] は、Cornell 大学が Bellcore 社などと協力し、アメリカ化学会 (ACS; American Chemical Society) の論文を対象として、学生などの利用実験を含めて研究しているシステムである。システム全体は、UNIX の X-Windows 上に実現している。

データ

収容データは、ACS の発行する紙メディアによる論文誌と、CAS （Chemical Abstracts Service; ACS の一部門で、テキストによる論文に関する業務を行っている。）の提供する論文全体と CAS が作成した要旨のテキストをデータとして用いている（図 5.1)。

学会から提供されるテキストデータには化学式・構造式をはじめとする図などのデータは含まれないので、スキャンしたページイメージから切り出す。これは、領域切りわけの手法を用いて、横方向の濃度を判断して（図 5.2）図と文の領域を区分する。数式は PostScript 形式で、表はテキストと空白による構成に展開する他、参考文献なども含めて、これはテキストの参照される部分とリンク付ける前処理をしておく。

機能

CORE で行える検索は、テキストとして持つデータに対するキーワードを用いた検索であり、この検索処理の結果の一覧から個々の文書へリンクが張られているのでここから選択することにより、または論文誌の巻号を直接入力することにより、指定された文書を呼び出し画面に表示する。

CORE での表示機能は二通りが用意されている。

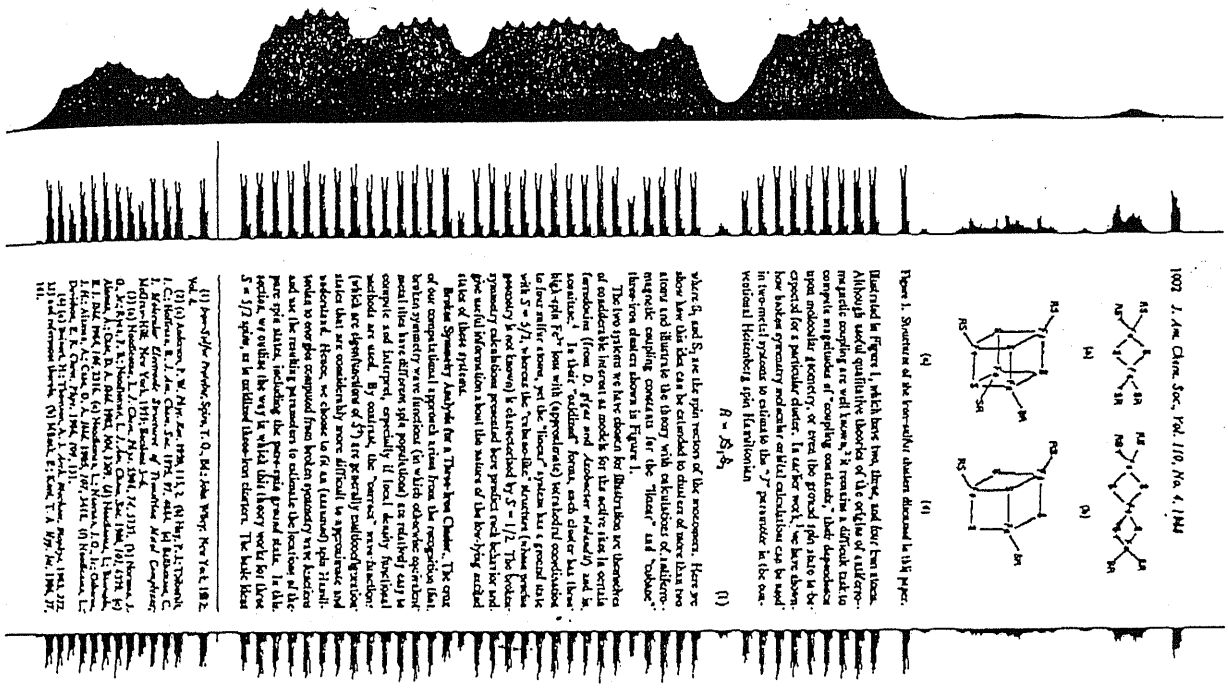


図 5.2: 図領域の切りわけ

- イメージデータの表示
  解像度 *100dpi* または *300dpi* が選べる。これにより、細部まで閲覧したいかどうか、通信ネットワークの帯域・混雑状況はどうか、のトレードオフからユーザが好みのものを選ぶことができる

- テキストデータを整形する表示
  文書が指定されると、テキストデータが表示画面に合わせて整形され、表示される。この時、参考文献や他文献の参照・脚注・数式・図は出現箇所にアイコンとして示され、これをクリックすることで他の箇所や他の文献に移動することができ、また別ウィンドウが作られて図などのデータを表示する。

この他に、

- 上の両者を併せたような、著者名・タイトルに加えて、論文の１ページ目のイメージを表示
- 論文中の図のみを列挙する表示

も用意されている。このうち最後のものは、その他の分野であれば目次を概要代わりに示すところを、文書中に出現する化学式を見ることで論文のポイントが把握できるという、化学会の論文ならではの要求をうまく捉えているといえる。

5.2.2 画面表示

LECTOR [14] はテキストを主とする文書データの画面表示のあり方を模索して、X-Windows 上に実装されたシステムである。著者らは SGML で記述された OED (Oxford English Dictionary) の処理法を巡る研究の一環として、表示について検討している。このシステムでは、紙メディアによる辞書と同様に基本的な機能で表示できることに加えて、

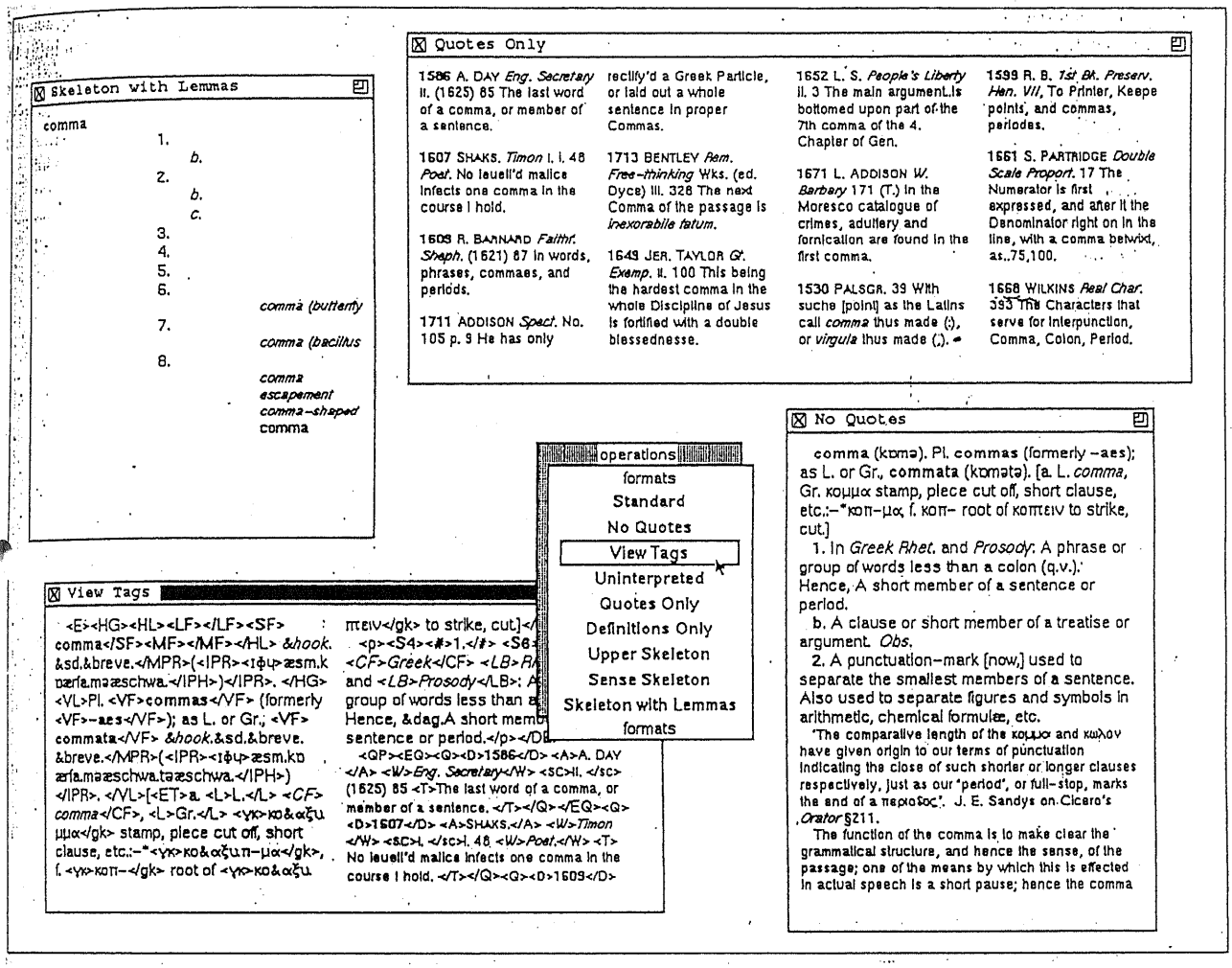


図 5.3: LECTOR でのテキストの様々な表示例

- 項目番号だけを示す
- 用例文だけを抜きだして示す
- 用例文抜きで示す
- タグ付きテキストのソースを示す

など、様々に提案している。必要な論理構造上の要素のみを、あるいは必要な細かさで抜粋して表示できるのは、まさにテキストを主とすることの強みであり、ユーザが迅速に内容を理解し、必要なところのみを閲覧するといった動きを助けることができる。

5.2.3 電子メディア独自の機能

電子メディアによるテキストを扱うことにより、従来の紙メディアでは難しい機能も実現される。そのような機能とは、たとえば、

- 音による読み上げ

- 自動翻訳
- 講読履歴の自動記録
- 辞書類・シソーラス・WWW など参考図書や「メタインデックス」との連動
- ファイルでの保存やコピー

など多数考えられ、将来的な文書の活用として大きな期待を寄せられる側面でもある。

RightPages [33] [34] [35] は、誌面のスキャンイメージを主としながらも、電子メディアによる図書館ならではの機能を取り入れている。AT&T Bell 研究所が、所員を対象として実験をしたシステムであり、前述の CORE との共同作業も行われている。

このシステムでは、ユーザの経験を尊重して、図書館での使い勝手のコンピュータ上での再現を目指している。たとえば、ユーザがシステムを起動すると、図書室の入り口にある棚を連想させる雑誌の表紙の一覧が表示される。そこから見慣れた表紙を持つ探している一冊を視覚的に探し、最新号を何となく開いて目次から興味のある記事を探す、という動作ができる。これは、目次を画像的に解析して、タイトルとページ数が示された一定領域を、その部分が表す論文本体に割り当ててリンク付けしておき、その領域をクリックすることでページのデータが画面に呼び出される、という仕組みである。

RightPages が取り入れた電子メディアならでは最大の特徴的な機能は、ユーザがキーワードを登録しておけば、新着雑誌からキーワードが示す興味に合う論文をチェックし、電子メールで通知する機能である。これによりユーザが図書館を無駄に訪れ、新刊には欲しい記事がないことを確認するような手間をを省くことができる。さらにその後の文献複写依頼など、一連の作業も電子的に行えるように機能や制度が整えられている。

1992 年当時の段階では、10 社 60 種類の雑誌を、100 人程度の研究所員を対象に実験が行われた。実験が行われた AT&T ベル研究所では、このシステムを肯定的に評価する所員が多く、さらに収録雑誌数や実験対象となるメンバーを増やして本格的な運用に近付いている。

5.3 Ariadne

前節でも触れた電子メディアによる文書の特性を多くとり入れて、全体としてシステムの実験を進めている例として Ariadne (Advanced Retriever for Information And Documents in the Network Environment) [36] [37] [38] を紹介する。

Ariadne は、電子図書館研究会が電子図書館の具体像を見せることを目的に、関西文化学術研究都市の高速データ伝送の実験の一つとして 1994 年に公開デモンストレーションを行ったシステムである。

5.3.1 収容データ

Ariadne の三種類のデータを収納する。

- 一次データ
  - 図書・論文のテキスト（HTML 形式）
  - 美術品などの画像・動画像
- 二次情報
  - 書誌情報
  - 表紙イメージ
  - 目次
  - 帯情報
    書籍して刊行される際の帯。図書の概要や、利用目的・範囲がわかる。
  - 項目メニュー（画像と動画像の一覧）
- 参考情報
  シソーラス（一般用語と専門用語）、複合語辞書、英和・和英・国語・専門用語辞書

これらのデータは互いにリンク付けされているので、たとえば書誌情報で調べた内容を参考情報でキーワードを広げたり狭めたりしながら、探している情報をより具体化し的確な結果を得る、といった行動が可能である

5.3.2 講読支援機能

Ariadne における講読支援機能は、通常の講読の他、実装されている（もしくは代替機能がある）機能は以下の通りである。

- 注・引用文献・他の箇所参照などの場所へ移動し、戻ってくる。
- 付箋をつける、メモをとる
- 検索語を入力すると、語の場所へ移動する、語の近傍を文脈がわかる程度に表示する。
- テキストを音読する。
- 翻訳して言語の違いを意識させない。
- 読了の印をつけて、次回は続きを読む。

これらを機能の点から分類すると、三種類になる。

- 参考情報の利用
  文書の講読では、辞書・辞典・検索など他の文書や情報を使うと助けになることも多い。このために、各種辞書類が、ハイパーテキストの形で格納されていて参照できる他、翻訳を行ったり、検索機能と連動して、検索条件を改善することも可能である。

  またインターネットに接続しているので、インターネット上の各種インデックスも、システムのメタインデックスであるかのように取り入れられる。

- メモと付箋、利用者環境の保存
  端末のウィンドウ機能により、別ウィンドウのエディタに、画面間コピーができる。このとき、原資料の文書名や文書中の位置も自動的に記録されるので、原資料を簡単に呼び出すことができる。また、電子メールとして送信することもできる。

  付箋も同様に、エディタ入力や画面間コピーにより、一行分のコメントを、位置情報とともに記録することができる。

  利用者環境は、一次情報・参考情報の利用履歴、画面表示の設定などが自動的に記録され、またホットリストも作成できる。

- 朗読と翻訳
  日本語テキストの音声による読み上げと、日英・英日翻訳が組み込まれている。将来的には、多言語対応や、点字などへの変換も計画されている。

5.3.3 検索機能

Ariadne における検索機能は、キーワード検索とハイパーテキストを活用した検索の融合、論理構造を活用した検索の実現、表紙画像の有効利用の三点を目指している。検索にはいずれの場合にも、各種辞書類やシソーラスを用いて翻訳を含めてより望ましい結果を得るための支援が期待できる。

まず、基本的な検索機能としては、通常の書誌情報に対するキーワード検索、フルテキスト検索などがあるが、検索の際には、文字列判断だけでなく、属性値（「重量がいくら」など）を用いた条件判断を行う機能があり、たとえば Kg と g など単位の統一もできるため、その不一致による検索漏れも防げる。

いずれも、検索結果から一次情報へリンクが張られており、フルテキスト検索の結果では、表示したテキスト中の検索語の位置も示される。検索結果により文書を表示し、検索語の位置を示す。将来的には、データベース中の各種情報間のリンク（静的リンク）や、ユーザの操作結果から作られるリンク（動的リンク）の意味を検討し、検索対象に取り入れることを計画している。

高度な検索機能としては次のようなものがある。

- 階層構造検索
  文書の論理構造、特に章題などは、構造上の上下に応じて広くまた狭い概念を表していると仮定することができる。この仮定の元に、階層構造を用いた検索を実現している。

  たとえば、タイトルに「人工知能」という語を持つ文書を検索する場合、“BT（人工知能)”と指示し、「人工知能を哲学の観点から論じている」文書を検索する場合は“BT（人工知能）& T（哲学 > 人工知能)”と指示する。

- 質問文による検索
  Ariadne には、簡単な自然文を入力して、その検索意図を解釈する機能がある。この機能は、質問文を用言とそれを結ぶ動詞の連結として、「構造化」し（図 5.4)、データベース中の同型に整理された

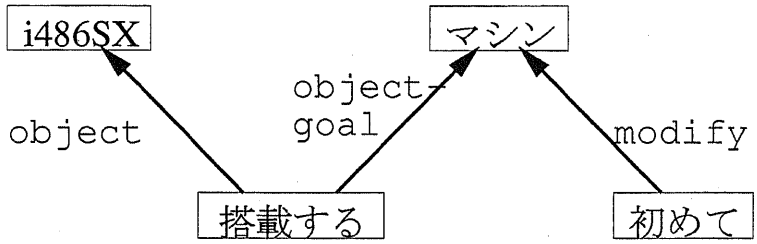


図 5.4: 質問文の構造化

データと照合して検索を行う [37]。この方法による利点は、従来のキーワードの AND/OR のみに頼る検索では避けられなかった、用語が異なる意味で用いられる場合、つまり「*i486SX* の開発」と「*i486SX* による開発」などが識別できる点である [39]。これは、自然言語処理からのアプローチであるが、全文を対象とした新しい検索という視点で、どういう方向で発展するか、楽しみである。

5.4 まとめ

計算機の高速化、ネットワークの普及という点で文書データベースが電子図書館として非常な身近さを持ってきたことは、正に時代の要請に応えていくものである。そしてさらに、数的にも、また規模、つまり政府などが加担する形態の増加という意味でも、その研究の勢いは加速している。

しかしこの段階では、まだ確立した何かがあるわけではなく、むしろ、きちんと文書を高度に活用しようとした場合に、ユーザがどのような検索などを求めているのか、文書の論理構造を活かすにはどのような検索機能を備えればよいのか、検索以外にはどうか、表示などはどうか、という根本的な問いに立ち戻ってしまうのが現状である。

WWW に加わる形で素早く普及する形態や、マルチメディア化などの華やかな部分だけではなく、一方でこういった地道なところでの検討は続けていかなくてはならない。この検討のためにも、一般のユーザにも手の届きやすい形で公開されている実験というのは、要望のフィードバックを得るためにはこの上ない形態であろう。両面からの進展を期待したい。

第6章

手法の全体像と変換仕様

ここまでに述べてきた課題や動向また関連研究を踏まえて、本研究の論理構造変換手法について取るべきアプローチをまず述べる。続いてこの手法がどのような段階から構成され処理されるのか、その全体像を説明する。また、この手法では論理構造の変換仕様として、新文書の各要素が旧文書のどの要素によってどのように作られるかという内容を指定する。この仕様を作成するために必要な概念や用語について説明し、実際にどのような変換仕様を記述するのか、その形式と意味について述べる。

この章の構成は以下の通り。

- 6.1: 本研究のアプローチ
  紹介してきた課題や動向を踏まえて、本研究の取った方向とその妥当性を述べる。
- 6.2: 提案する変換手法の全体像
  提案する手法がどういう段階から構成され、また処理がどのような流れで行われるのか、全体的に概観する。
- 6.3: コンポーネントの優位性
  処理を行うに当たっては、論理要素の名前や識別番号以上の評価方法が必要になる。その評価基準を定義し、評価法を述べる。
- 6.4: 要素などの属性
  処理を形式的に記述するために、論理要素などにここまでに述べたいくつかの特徴や区分を用いる。ここでこれらを改めて属性として定義する。
- 6.5: 変換仕様の記述
  論理構造の変換の内容を指定する変換仕様の記述法のうち、主に要素間の対応付けについて述べる。
- 6.6: データ形式の記述法
  変換仕様の記述法のうち、データ形式の指定の仕方について述べる。

6.1 本研究のアプローチ

6.1.1 SGML 間での論理構造変換

本研究では、文書はオリジナルも SGML で記述され、それを異なる論理構造を持つ SGML へ出力するという方式を採用している。

現在は、電子化文書の構造化といえば前述のように HTML や LaTeX が普及しており、様々な論理構造の文書を集めて、あるいは一般的な形式に統一して処理する場合にもそれらが採用されることが多い。現在の技術で当面の要求を満たすためには、ツール類も充実しており、多くの人が環境を持っているという観点からそれは妥当な選択であるとしても、より充実した電子化文書の活用を目指すために、将来的には自由度として物足りない。

前述のように、データベースも収容する内容に応じた論理構造を持つことが望ましい場合がある。例えば、シンプルな論理構造を持ち、ユーザが一見して内容を把握するためのインタフェースを備え、従来の二次情報型データベースに 近い環境を提供しようとする場合も考えられれば、一方で画面上での閲覧に際してきめ細やかなレイアウトや情報の分類を提示し、論理構造を活用した高度な検索条件の使用を可能にする場合も考えられる。

また本研究ではデータベースの実装については細かい検討を保留しているが、変換した出力を、検索や表示・印刷にとどまらず、更になんらかの形式に変換して、他のシステムなどへデータを転用する要求にも拡張できる。

つまり、一度の変換で文書の論理構造による記述力は落とすことなく、同じ程度の情報を持つとする方式には利点が多く、従って SGML 入力を SGML で出力するという手法が有利であることが言える。むしろこれが可能になるように技術を研究することが望ましい。

6.1.2 インスタンス間での論理構造の変換

論理構造を木構造のまま、つまり文脈自由文法の形で扱えば、体系として整っていると言え、学問的な見地とも合致していて従来の研究などが適用できる点で望ましい。しかし、この方向は次の二点で難しい。

まず、実際の文書は文脈自由文法の範囲には納まり切らず、SGML でも文脈自由文法の水準を越える、浮動要素という種類を取り入れたりしている。これを BNF で記述すると、生成規則の数が膨大になる。

また第 4.1.1節に述べたように、文法的な手法が取り扱える変換処理も、現実的な要求を満たすには物足りない。たとえば、隣あった要素の組み合わせやデータの変形程度であれば、YACC を用いてアクションとして記述すればよい。しかし、組み合わせる要素が近いものに限らなかったり、互い違いに出現する要素を組み合わせたりすることを考慮していくと、文書全体の影響をうまく表現するような仕様の作成が必要となり、複雑すぎて負荷として現実的ではない。

6.1.3 要素間の対応付け

SGML 文書を扱う場合、論理構造の変換仕様の記述の一つの方向として専用の言語を用いる手法があり、第 4.2.4節で紹介したが、実際の使用者にとって新しい言語を習得する負荷は大きく、容易ではない。

簡単に使えてかつ SGML らしさを完璧に活用するためには、DTD に基づいて文書の解析などを行うことが望ましいのは一理あるが、第 2.5章でも述べたように、SGML が対象としている論理構造は可能性として複雑過ぎることから、DTD に基づいた処理は処理量が大きく、効果との比率から考えて得策ではない。

ただし論理構造の変換以外でも、整形や検索には厳密なパーサやビューワがやはり必要であり、SGML の汎用処理系なども提供されているのでこれらを利用する方法がある。問題はこのような解析ルーチンの作成が大きな負荷である点で、さらに解析しながら変換するような複数の段階をまとめて行う処理系を構

築するのは得策ではない。別々に作成すれば、それぞれの作成の負荷は軽減される上、市販や汎用アプリケーションを活用すればよい段階もある。

もう一点論理構造の厳密な処理を必要とするのは、変換仕様作成の段階である。言語的に厳密な論理構造のチェックは省けるとしても、その全体像が把握しなくては論理構造の対応が決められない。そこで実用の際には、変換仕様作成インタフェースの一環として、論理構造の可能性、つまり複数回出現（しかし回数は不定）や浮動要素（具体的な出現箇所は未定）といった形で、ビジュアルに論理構造を示す仕組みが必要である。このような動作をする汎用処理系などを援用する必要がある。

つまり DTD の厳密な適用は必要最低限にして、要素の対応付けで処理を行う方向を検討すべきである。

6.1.4 論理要素の位置情報の活用

要素間の対応付けにより論理構造の変換を行う場合、第 4.2節で取り上げたような研究例があるが、これだけでは同名の要素には同一の処理しかできないといった点で不満が残る。そこで、これをどのような文脈、つまり上位要素が何であるかという条件を加味して該当する論理要素を判断できるような方式が必要である。そこで本研究では、対応付けにあたって、論理要素が構造上ルートからどのような位置にあるか、経路をパスの形で表現して、これを用いて指定することにする。

パス形式は、UNIX など計算機で広く使われておりなじみがあり、理解しやすい。

6.1.5 その他の前提

本研究では、厳密な SGML 文書の解析を目的としていない。従って、

- タグの省略
- 実体参照の展開
- DTD との適合性

などのチェックは組み込まず、既に達成されているものとする。また、SGML 文書を処理するのが第一義的な目的であるから、関連する可能性を網羅するよりも、対象とする範囲に以下のような制限を設ける。

- 排除要素は考慮しない
- 変換での記述力は DTD での記述力を目安とする

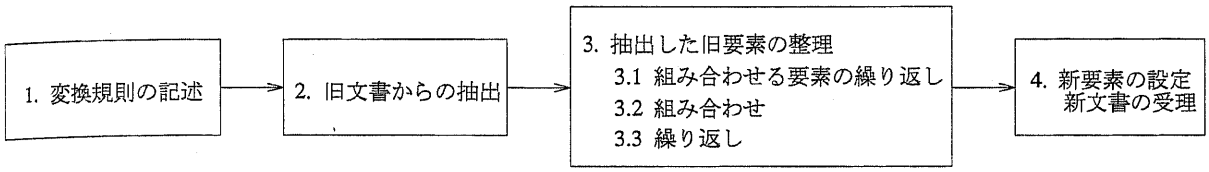


図 6.1: 提案する手法の処理の流れ

6.2 提案する変換手法の全体像

6.2.1 本手法の全体像

本研究で提案する論理構造の変換手法は次のような段階で構成される。(図 6.1)。

1. 論理構造の変換規則を記述する。
   データベースの準備作業として、各出版社や雑誌の個々の論理構造について、それとシステムの持つ論理構造の間でどのような変換が求められているのか、要素間の対応関係を予め一度記述する。

2. 旧文書を解析して、新文書を構成するために必要な旧要素を抽出する。
   旧文書の論理要素から、新文書で必要なものを取捨選択する。

3. 抽出した旧要素を繰り返しや組み合わせを考慮して整理する。
   旧要素の繰り返しや組み合わせといったケースを、それらの間の影響を整理して段階を追って処理する。

4. 用意された旧要素に相当する新要素を設定して、新文書として受理する。
   前段階までで旧要素がどのような新要素を表現するものか整理されているので、それに従って旧要素群に見合う新要素を設定する。そして、要素群からパスを手がかりに木で表現される論理構造を再構成し、タグつきのマークアップ文書として出力する。

以降では仕様の記述を説明する。そこで前もってそれに必要な概念などを準備する。

6.2.2 要素の表現

本研究では、実際のデータと直接関わっている論理構造の最下位要素に着目し、文書とはこの論理要素のリストであると解釈する。そしてその各要素が 文字列などのデータを値として持っていると捉える。

このとき論理要素は、同名でありながら論理構造上で異る位置に出現する場合が あり得る。例えば「名前」に相当する論理要素があったとしても、論理構造上で「著者」という要素の下位にあればその文書の著者である人物の名前を表すと考えられるし、「発行」という要素の下位にあれば発行人である人物(出版社や学会の担当者)の名前を表す要素であると推測される。

これらの場合を区別し、意図するデータを的確に指し示すことができるように、論理要素は論理構造上の位置をパスとして持つ。このパスは、論理構造のルートからその論理要素までの木構造上の道筋を表し、原則としてルートからその要素まで、省略や曖昧性を含まない形で示される。これを**絶対的な位置表現・パス表現**(絶対パス)と呼ぶ。

パスは、ルートからその論理要素までの道筋を構成する途中の論理要素、つまり論理構造上の中間ノードを列挙することで表現される。このとき、パスを表現する中間ノードを**コンポーネント**と呼ぶ。パスは、コンポーネントを区切り子 "/" を挟んでリスト状に繋げた形式で表現される。

コンポーネント、つまり論理要素は、一つの文書を構成する論理構造中で複数回出現する可能性がある。たとえば先ほどの例では、「名前」に相当する論理要素が二ヶ所に使われることを意味しており、また著者が複数名挙げられる場合は、「著者」の下に著者名を意図した「名前」が著者の人数分だけ列挙されることになる。このような場合に個々の「名前」を識別して指し示すために、コンポーネントは同一の種類、たとえば「名前」の中では一意な番号により識別する。この番号を **ID** と呼ぶ。ID は同一種の中で重複せず一意であることが条件であり、異なる種類の要素の ID 群と重複しない、文書中の出現順に応じて登り順である、抜けのない自然数である、などの付帯的な設定は必要としない。ID は、必要な場合にコンポーネントの名前に続いて、数字の添字として表される。

たとえば、

- 要素 E' は、
- 値が "Smith" で、
- パスが "/Recordlist1/Record1/Author1/Surname1"、
- パス中のコンポーネントに "Recordlist" という名前のものがあり、
- それは ID が 1 である。

などと言う。以降ではこのようなセットで扱われるものを要素と呼ぶ。

6.2.3 変換規則の内容

論理構造の変換には仕様が必要である。これから仕様である、変換規則の記述について説明する。

変換規則は以下の三つの内容を含む。

1. 各新要素を構成する、旧要素または旧要素群
   どの新要素が、どの一つもしくは複数の旧要素により作られるかの割り当て
2. 旧要素の繰り返しの判断や、新要素の設定の際のパスの評価法
   繰り返しでまとめるものとそれ以外を判別し、また新要素を設定する時に適切なパス、特にコンポーネントの ID を割り振るための手がかり
3. データの形式の指定
   旧要素を組み合わせる場合、繰り返しとしてまとめる場合、あるいは旧要素とは違う表現、たとえば大文字を小文字で、括弧をつけて、などでデータを持ちたい場合、などのためのデータの整形の仕方

以降では、これらの内容について順次説明するが、このうち、特別な方法を必要とする 2. のパスの評価法に関連して、「優位なコンポーネント」を定義して、評価の方法を述べる。

6.3 コンポーネントの優位性

パスの処理において、名前だけでは判断のできないケースがある。二つのパスの名前やIDが同じあるいは異なるからというのが、同様の処理をすべきかどうかの基準にならない場合である。これらを説明しながら、優位なコンポーネントを定義する。

6.3.1 繰り返しの評価

同じ旧文書中に現れる、同じパスを持つ複数の要素が、一つの新要素にまとめるものか、別々にしておくものかを判断する。たとえば、同じレコードに関するものは著者の所属をまとめて一項目にしたいが、違うレコードの著者の所属はまとめたくない、という要求を考える。これはたとえば、割り当て規則に

> 著者の所属に関する旧要素 "/Recordlist/Record/Author/Affiliation" が新要素 "/recordlist/record/affiliation" を構成する。ただし、同じレコードの著者についてはまとめるが、異なるレコードの著者についてはまとめない。

と定義された場合、図3.1で、

- *4:* /Recordlist1/Record1/Author1/Affiliation1 ... NACSIS
- *8:* /Recordlist1/Record1/Author2/Affiliation2 ... NACSIS

は同じ新要素になる。このような旧要素を**繰り返し**と呼ぶ。

一方、

- *14:* /Recordlist1/Record2/Author3/Affiliation3 ... NACSIS

は別の新要素になる。

これらの二組の旧要素では、まとめられる方もそうでない方もパスの名前は同じだが、まとめられる旧要素群の中でもパスのIDは必ずしも同じではない。つまり、これらのケースの区別では名前とIDだけでは基準が足りない。そこで、これらの両組を分けているものは何かと考えると、名前はいずれも同じであるので考察から除けば、IDについて "Record" のIDが区別を判断するの鍵となっている。つまり、コンポーネント "Record" のIDが同じであればまとめるものである、という基準になっている。その一方で "Author" のIDは鍵とならない。

従ってコンポーネントの中には、そのIDが繰り返しを判断する鍵となるものと、そうでないものの二通りがあることがわかる。そこで、鍵となっていた "Record" を優位と定義する。実際には優位なコンポーネントの方が数が多いことから、変換規則では優位でないコンポーネントの方の前に "!" という印を付与して、"!/Author" という形式で優位でないことを明示する。

繰り返しの評価においては、最も下位の優位なコンポーネントがどれであるかという点が効力を持つ。というのは、そのコンポーネントのIDが違っていればその時点でまとめない繰り返しであり、逆に同じである場合はその要素間は上位の要素もIDが同じであるからである。

6.3.2 新要素の設定の評価

パスの名前やIDだけで処理の方法が決められないもう一つの段階が、新要素のパスを設定する処理である。対応する新旧要素のパスの長さ、つまりパス中のコンポーネントの数が異なる可能性があり、長さが異なる場合には旧要素のある コンポーネントの同異を新要素のどのコンポーネントの同異に反映させるか、という新旧要素のパス中のコンポーネントの対応関係が定まらない。

図 6.2: 優位なコンポーネントの役割

たとえば前節と同様の例で、"Author" 以下の ID だけが違っている要素は新要素ではまとめ、"Record" の ID の違うものは別の新要素にするとした。このときの "Record" の違いを新要素にどう反映させるかを決定するのが優位性である。

たとえば、

"Record" の ID が違う旧要素群は、新要素でコンポーネント "record" の ID の違いで表す

と指定する。つまり、前節では旧要素 *4* と *8* は繰り返しとしてまとめて、"Record" 以下のコンポーネントの ID が異る旧要素 *14* は別の要素としたが、このとき作られる新要素は互いに "record" の ID が異なるものとなる。

- *4:* /Recordlist1/Record1/Author1/Affiliation1 ... NACSIS
- *8:* /Recordlist1/Record1/Author2/Affiliation2 ... NACSIS
  ↓
  b: /recordlist1/record1/affiliation1 ... NACSIS/NACSIS
- *14:* /Recordlist1/Record2/Author3/Affiliation3 ... Univ. of Tokyo
  ↓
  g: /recordlist1/record2/affiliation2 ... Univ. of Tokyo

これを示すために、"Record" と "record" は共に優位であるように定義する。さらに、優位なコンポーネントの中で "Record" と "record" とが関連していることを表すために、それらが優位なコンポーネントのうち、新旧それぞれのパスの中でルートから数えて同じ順番にあるようにする。例では、さらに上位 "Recordlist" と "recordlist" も優位であるように定め、従って "Record" と "record" とはともに優位であり、かつ、優位なコンポーネントのルートから二番めの位置にある、というように定義した (図 6.2)。

割り当て規則に組み合わせがある場合、組み合わせる旧要素群の共通でない部分のパスは、個々の要素で名前も同じではない可能性がある。そこでそれらの個々のパスについて、対応する一つの新要素のパスと照らし合わせて ID などを判断するのは意味がなくなるため、新要素を設定する以前に組み合わせの処理は終えて、新要素の設定の際には組み合わせる旧要素群の共通ではない部分は必要でなくなる動作とする。また優位性は組み合わせる旧要素群の共通でない部分には適用せず、共通な部分のみと組み合わせがない場合のパス全体にに適用される。

以上のことから、対応する新旧要素のパス中で優位なコンポーネントは同数である必要がある。また上位コンポーネントの ID の変更は下位コンポーネントに影響する。

図 6.3ではまた、優位なコンポーネントを様々に指定した場合の効果を示した。*(0)* はもとの例の割り当て規則の優位性の指定を表している。つまり、"Recordlist" が "recordlist" に、"Record" が "record"

図 6.3: 優位性の様々な定義の影響

に反映されている。*(1)* では *(0)* に加えて、"Author" の ID の変更が "author" に影響するように指定する。すると、"Author" の違う上二つと下二つの要素は今度は別々に繰り返しとしてまとめられ、旧要素での "Author" の ID の違いが新要素ではコンポーネント "affiliation" の ID の違いとして反映される。

一方 *(2)* は *(0)* に加えて、旧要素での "Affiliation" の ID の違いが新要素では "affiliation" の違いとして表される。すると、四つの旧要素はすべて "Affiliation" が異るので、四つの別々の新要素となり、それらの新要素はすべて "affiliation" の ID が互いに異なっていることで、旧要素の違いを反映している。

この時、*(1)* で前後のコンポーネントが優位でそれぞれ対応している状況で、その間にある "Record" の違いが "record" の違いに反映する、という指定を優位でないようにしても実際的な意味は持たない。むしろ、*(2)* のように、パス中のコンポーネントの数が異なる場合に、どれが対応関係からはみ出させられるか、という指定を記述しているという捉え方もできる。

いずれにしても優位性を用いることにより、繰り返しでどの場合をまとめるか否かや、新要素の構文木の構造について、設計者の要求に応じて柔軟に記述することができる。

6.4 要素などの属性

ここでは本手法で用いる属性などを定義し、一般的な形式で変換処理を述べる準備を進める。

以下に述べる記号類では、旧要素とそのパスやコンポーネントに関するものは "′" を付して、"E'" のような形式で表現する。新要素に関するものは、印のつかないそのままで表現する。

6.4.1 要素の属性

本手法において要素は二通りの状況で用いられる。それらは、

- 論理構造の骨格として変換規則を構成：(狭義の) 要素
  実際のデータを伴わない
  要素の繰り返し出現回数、浮動要素の出現位置などが定まらない
  (それらの可能性のみをルールとして含む)

- 文書を構成：要素インスタンス
  実際の文書として、データを持つ
  繰り返し出現回数、浮動要素の位置などが確定

であり、以降では区別して、それぞれ "A [1] " と "D [2] " と表す。

変換規則に用いられる要素 A は次のような属性を持つ。

- パス：$A.path$
  A は、同名でも出現位置が異なるものを区別するために、パスで論理構造上の位置を表す。これは原則として、論理構造のルートからその論理要素までの道筋にあたるコンポーネントを、略さずにすべて列挙したもの (**絶対パス**) とする。
  (例)：$A.path$ は、"/Recordlist/Record/Author/Surname"

- A が組み合わせを含む割り当て規則かどうか：$A.comb$ [3]
  $A.comb$ はフラグであり、組み合わせを含むか否かに応じて $COMB$ または NOT の値を持つ。

具体的な文書を構成する要素インスタンス D は以下のような属性を持つ。

- パス：$D.path$
  文書は要素 D のリストとして捉えられており、やはり同名で出現位置の異なる論理要素と区別するため論理構造上の位置をパスとして持つ。このパスは必ず絶対パスである。なぜならば、D は具体的な文書中の要素インスタンスであるため、その構造に曖昧性はないからである。
  (例) $D.path$ は、"/Recordlist1/Record1/Author1/Surname1"

- 値：$D.val$ [4]
  要素の値とは文書の実体である実際のデータを言い、その内容は文字列や画像データなど、SGML の実装に応じたデータ形式である。
  (例)：$D.val$ は "Smith"

- D が関係する変換規則の番号：$D.ass$ [5]
  $D.ass$ は処理の過程で、その要素インスタンスが旧要素インスタンスであればどの変換規則によって

[1] A = Elements in Assignments
[2] D = Elements in Documents
[3] comb = combination
[4] val = value
[5] ass = assignment

新文書で必要と判断されたか、また新要素インスタンスであればどの規則によって作られたか、という情報を持つ。この情報を変換規則の番号として $D.ass$ に持つ。値は実装に依存するが、本研究では整数を用いている。

6.4.2 コンポーネントの属性

次に、論理構造上の中間ノードであり、パスを構成するコンポーネントについて考える。コンポーネントは C と表し、特にルートに当たるものを C_0 と表す。
C は次のような属性を持つ。

- 名前：$C.name$
  C の名前、つまり種類を表す。
  (例)：$C.name$ はたとえば "Recordlist"

- 識別番号：$C.id$
  $C.id$ は個々の種類（"Recordlist" など）の中では一意な数字で、各々のコンポーネントを識別する。これにより、繰り返して出現する論理要素、たとえば複数個ある章などを区別して扱うことができる。ここでは自然数を用いており、また異なる名前のコンポーネントにわたっても一意である必要はないが、これらはどちらでもかまわない。

- C が優位であるかどうか：$C.sup$ [6]
  優位性は、組み合わせのない規則の場合新旧要素のパス全体に、組み合わせのある場合には新要素のパス全体と、組み合わせる旧要素群に共通な部分パスに適応される。その役割や組み合わせがある場合の旧要素群の残りのパスに当てはまらないことは前節で述べた。
  従って、$C.sup$ は三種類の値をとる。
  - 優位である：$C.sup = SUP$
  - 優位でない：$C.sup = NOT$
  - 組み合わされる旧要素群の、共通でない部分のパス：$C.sup = COMB$
    この場合、優位であるかどうかという判断は適用されない。

6.4.3 パスの属性

要素のパス、あるいはパスの一部（**部分パス**）はコンポーネントを区切り子を挟んで列挙したリストとして構成される。パスは P と表し、以下の三通りに分けられる。

$$
\begin{array}{lcl}
P & : & C.sup \text{ が不定、もしくは考慮しない場合} \\
P_{SUNO}{}^{7} & : & C.sup = SUP \text{ もしくは } NOT \text{ の場合} \\
P_{COMB} & : & C.sup = COMB \text{の場合}
\end{array}
$$

以降で二つのパスが同じ、あるいはパスのコンポーネントの名前や ID が同じであると言う時には、パス全体を通してそれらのパス中のコンポーネントがルートから順に名前や ID が同じであることを意味するものとする。ただし、あるパスの途中のコンポーネントの ID が違えば、論理構造の木構造から明らかなように、そこより下位のコンポーネントの ID はすべて違うものになる。

[6] sup = superiority
[7] suno = superior or not

6.5 変換仕様の記述

論理構造の変換仕様はデータベースの準備作業として、検索などの処理に提供するに先だって予め作成する。これは通常、文書の著者やデータベースの管理者など、もとの文書とデータベースの両方の論理構造を把握する、ある程度論理構造を扱いなれた者が行う。

変換仕様は、一行ごとに一つに新要素について、大きく分けて、

- 要素の割り当て
  その新要素をどの旧要素または旧要素群が構成するか
- データの形式
  新要素を構成する際にデータの形式をどう整形するか

という指定の二つの部分を併記することで構成されるが、この章ではこのうちまず、割り当てについて記述の形式とその意味を述べる。形式の指定については続けて第6.6節で説明する。

割り当ての記述形式は、その変換規則が組み合わせを含むかどうかで分けられる。

まず最も簡単な、組み合わせを持たない場合、つまり一つの旧要素が一つの新要素を構成する場合、割り当て規則はパスで表したその新要素と旧要素を、区切り子 “;” で挟んで記述したリストになる。たとえば、図6.4の *1.*

/recordlist/record/affiliation ; /Recordlist/Record/Author/Affiliation
（新要素）；（旧要素）

は、新要素 “/recordlist/record/affiliation” は旧要素 “/Recordlist/Record/Author/Affiliation” で構成されることを示す。これによりたとえば、

旧要素 “/Recordlist/Record/Author/Affiliation... NACSIS”

が

新要素 “/recordlist/record/affiliation ... NACSIS”

になることを表している。

一方一つの新要素が複数の旧要素から構成される場合、つまり組み合わせのある場合の割り当て規則は、

- 新要素をパスで表したもの
- 旧要素群が共通とするパス
- 各旧要素の残りのパス
  組み合わせる旧要素の数だけ

を並べて記述したものとなる。そして、新要素に関する部分と旧要素に関する部分の区切りには “;” を、旧要素に関する部分内の各項目間の区切りには “:” を用いる。

たとえば、図6.4の *5.*

/recordlist /record /author ; /Recordlist/Record/Author : /Surname : /Firstname
（新要素）；（旧要素の共通部分パス）：（旧要素の残りのパス）：･･･：（旧要素の残りのパス）

は、新要素 “/recordlist/record/author” を構成する旧要素群は、パスの共通部分が “/Recordlist/Record/Author” で残りの部分が “Surname” と “Firstname” であるもの、つまり “/Recordlist/Record/Author/Surname” と “/Recordlist/Record/Author/Firstname” であることを示している。たとえば

旧要素 "/Recordlist/Record/Author/Surname ... Smith"
"/Recordlist/Record/Author/Firstname ... Bob"

が

新要素 "/recordlist/record/author ... Smith, B"

を作ることを表している。データの形式については次節で述べる。

一般に i 番目の割り当て規則は次のように表される。

割り当て規則は個々の新要素に対して定められ、各々の新要素に関する一行には、新要素自身のパス $A_i.path$ とそれを構成する旧要素についてのパス $A'_i.path$ が、";" で区切って記述される。

$$A_i.path; A'_i.path$$

この中で、新要素に関する部分 A_iは、優位なコンポーネント・そうでないコンポーネントを取り混ぜて構成されるパスを持つことから、次のように表される。

$$A_i.path : P_{SUNO}$$

旧要素に関する部分 A'_iは、組み合わせの持つか否かによって、二通りがある。組み合わせがない場合は、パス全体が優位なコンポーネントあるいはそうでないコンポーネントを取り混ぜて構成される。一方組み合わせを持つ場合は、共通部分は優位なものとそうでないものの混成であるが、残りの部分は優位か否かが適用されないため、専用の組み合わせる要素の残りの部分パスという扱いになる。

$$A'_i.path : \begin{cases} P'_{SUNO} & \text{(組み合わせのない場合)} \\ P'_{SUNO} : P'_{COMB} : \cdots : P'_{COMB} & \text{(組み合わせのある場合)} \\ \text{(共通部分のパス)} : \text{(残りの部分のパス)} : \cdots : \text{(残りの部分のパス)} & \end{cases}$$

本稿の例では、割り当て規則全体は図 6.4のようになる。図で左端の数字は説明のための番号であり、"#" で始まる行は、コメント行であるものとする。

ここまでに述べた新旧要素の割り当てに関する内容に続く部分では、データの形式を指定している。

```
   # 変換仕様
   #1. 組み合わせのない場合
   #    新要素 ; 旧要素 ; データ形式 ; 繰り返しのある時のデータ形式
0. /recordlist /record !/subject; /Recordlist /Record !/Title; ``$1''; $$ ($1)
1. /recordlist /record !/affiliation; /Recordlist /Record !/Author !/Affiliation;
                                                                  $1; $$ / $1
2. /recordlist /record !/other /key; /Recordlist /Record /Key;
                                                  &lowercase($1); $$, $1
3. /recordlist /record !/other /abst; /Recordlist /Record /Abst !/Para; $1; $$ $1
4. /recordlist /record !/other /abst /ref ; /Recordlist /Record /Abst !/Para /Ref;
                                                                     $1; $$ $1

   #2. 組み合わせのある場合
   #    新要素 ; 旧要素群のパスの共通部分 : {各旧要素のパスの残りの部分}+ ;
   #        {各要素のデータ形式}+ : {繰り返しのある時のデータ形式}+ :
   #        組み合わせのデータ形式 : 組み合わせたものの繰り返しのデータ形式

5. /recordlist /record !/author; /Recordlist /Record !/Author/: Surname: Firstname;
                        $1: &initial($2).; $$ [$1]: $$ [$2]; $1, $2; $$ / $1
```

図 6.4: 変換仕様の記述例（左端の数字は、説明のための番号）

6.6 データ形式の指定法

変換仕様の記述で新旧要素の対応をパスで記述した後ろは、論理構造の変換におけるデータの整形について指定する部分である。本手法ではデータである文字列を解釈して処理するような踏み込んだ処理は行わず、文字列置換などの形式的な処理の範囲に集中する。

処理の中では、旧要素のデータを整形していって求める新要素のデータ形式を得る。このデータ形式の記述法を説明する。

6.6.1 データ整形の種類

論理構造の変換で必要なデータ整形は、いくつかの段階に分けることができる。

- 個々の旧要素の整形
  (例)：「データを二重引用符で挟む（図 3.1の *0*“Doc1” から *c*‘‘‘Doc1’’’ へ)」
  「すべて小文字で表現する（*9*“Structure” から *d*“structure” へ）
- 組み合わせ、つまり１つの新要素を複数の旧要素が構成する場合の、旧要素群の組み合わせ方
  (例)：著者名について、「もとの文書では姓と名とが別の要素だったが、これを１つの要素内で “姓, 名前の頭文字” という形式にしたい（*15, 16*“Michael”“Brown” から *i*“Brown, M.” へ)」
- まとめる繰り返しの時の、まとめ方
  割り当ては１対１であっても、その旧要素が複数回現れ、かつ新文書では該当する要素の複数回出現が定義上許されておらず、複数個作れない場合、旧要素はまとめて１つの新要素にしなければならない。(繰り返し)
  (例)：「もとの文書で複数回現れたら “/” を挟んでつないで１つの要素にする（*4, 8*“NACSIS” から b“NACSIS / NACSIS” へ)」

繰り返しは、１対１の場合以外に、組み合わせたもの全体にも起こり得る。

さらに、繰り返しは、組み合わせる要素群のうちの各々にも起こる。たとえば、

新要素 “/recordlist/record/author”

は２つの旧要素群

“/Recordlist/Record/Author/Sirname”
“/Recordlist/Record/Author/Firstname”

で構成されるが、このうちの前者は（*5*“Jones” と *6* “Nelson”）で複数回現れている。これらは、組み合わせる前にまとめてしまう。

可能性としては、さらにこの上、組み合わせと繰り返しは相互に無制限に起こり得る。しかし本研究では、実用面での実際の文書であり得る範囲と処理及び処理システム作成の負荷の観点から、上記に述べた再帰性のみを取り上げることとする。

以上を、組み合わせの有無や処理の順序を考えて整理すると、必要な形式の指定は次の通りになる。

- 組み合わせのない場合
  1. 個々の旧要素の整形
  2. 繰り返される場合の整形
- 組み合わせのある場合

1. 個々の旧要素の整形
2. 個々の旧要素が繰り返される場合の整形
3. 組み合わせる整形
4. 組み合わせたものが繰り返される場合の整形

6.6.2 データ形式の記述と動作

仕様書における形式の指定は、記述の容易さ、また処理の簡便さのために、求めるデータの形式をできるだけそのまま記述することとし、“‘‘’’” で囲むなら “‘‘’’” で囲んだ形式を、“/” で区切ってつなげるならば区切って並べた形式を記述する。

その際データを表す変数には yacc (yet another compiler compiler) と同様の記法を採用する。つまり、それまでに作られたデータを “$$”、新しいデータを、パスを記述する割り当て部分に記された順に “$1, $2, ···” と表す。

また簡単な文字列処理のために、もとの文字列データを引数として渡すと、整形したデータを返り値として返すような関数を予め用意しておき、これを呼び出して必要な時に必要な個々のデータの整形処理を行う方式も取り入れた。これを呼び出す場合、“&” に続けて関数名を記し、引数として渡すデータを yacc 式記法の変数を介して与える。

たとえば “&initial($1)” は、予め用意した initial() 関数にデータ$1 を引数として渡し、その結果として返ってくるデータと置き換えることを意味する。例では、頭文字のみを残す initial() 関数と、すべて小文字に揃える lowercase() 関数が使用されている。

これらの表記を用いて具体的なデータ形式の記述の仕方を述べる。

まず組み合わせのない場合には、前節の 2 つの項目、つまり個々の旧要素の整形と繰り返される場合の整形の形式を、“;” で区切って順に記述する。たとえば図 6.4の *0*

```
‘‘$1’’ ; $$ ($1)
```

では、割り当て部分に記された旧要素は一つであるから、“$1” は “/Recordlist/Record/Title” のデータを意味し、“$$” はその時までに処理し終わったデータを示す。この部分全体では、

1. 旧要素 “/Recordlist/Record/Title” の一つ一つのデータは二重引用符 ‘’’’ で挟み、
2. 繰り返しでまとめる場合には、それまでにできあがっていたデータ “$$” に、今新たに出現して個別の整形を加えたものを括弧に入れて後ろに並べる

という指定を表す。従って、一つめの “/Recordlist/Record/Title” のデータ、たとえば “Doc1” は‘‘‘Doc1’’’ となり、もしまとめる二つめの “/Recordlist/Record/Title” がある場合には、そのデータたとえば “subtitle1” を ‘‘‘subtitle1’’’ にし、さらに、それまでにできていた$$、つまり ‘‘‘Doc1’’’ のうしろにつなげて ‘‘‘Doc1’’ (‘‘subtitle’’)’ となる。

一方組み合わせがある場合には、上記の 4 つの項目を “;” で区切って順に記述し、さらに各項目中の個々の旧要素に関する記述を “:” で区切って指定を記述する。このときの記述順や記述中のデータを指す変数名 (“$1” など) は、割り当て規則での出現順に準拠する。たとえば *5* の、

```
$1 :  &initial($2) ; $$ [$1] :  $$ [$2] ; $1, $2 ; $$ / $1
```

では、割り当て規則で一つめにある “/Recordlist/Record/Author/Surname” のデータが$1 であり、二つめの “/Recordlist/Record/Author/Firstname” のデータが$2 となる。このデータ形式の指定部分全体では、

1. 個々の旧要素の形式は、(割り当て規則で一つめの) “Surname” のデータはそのまま、(二つめの) “Firstname” は関数 “initial()” にデータを与えた結果に置き換え、
2. 個々の旧要素が繰り返される場合には、両方とも新たに出現したデータの個々の整形結果を “[]’” で挟んで後ろにつなげ、
3. 組み合わせ方は “Surname のデータ, Firstname のデータ” という形式、
4. 組み合わせたものが繰り返される場合には、“/” で区切って後ろに並べる

という指定を表している。

以上の指定に従う整形処理は、記述に基づき、

- “$” から始まる部分は該当するデータに置き換え、
- “&” から始まる部分は該当関数で処理した結果に置き換え、
- それ以外の ‘‘‘’’’ などは文字列置換を行う

という処理で、所望の形式のデータを作成することができる。

第 7 章

処理過程

ここまでに述べてきた仕様の記述を用いて、実際の処理を行うためのアルゴリズムや処理例を述べる。この処理は、前に述べたように四段階で行われる。さらに、相対的なパス表現を用いた仕様の拡張について述べる。

この章の構成は以下の通り。

- 7.1: 旧文書からの要素の抽出
  変換前のもとの文書から、新文書を構成するのに必要な要素を抜き出す。
- 7.2: 抽出した旧要素の整理
  旧要素は、組み合わせる要素の繰り返し、組み合わせ、繰り返しの三段階で処理し、新要素と一対一となり、データの形式も整えられる。
- 7.3: 新要素の設定と新文書の受理
  それぞれの旧要素から作られる新文書のパスなどが決定される。
- 7.4: タグつきテキストとしての出力
  新要素がパス表現から木構造を再構成されて、タグ付きテキストとして出力される。
- 7.5: 相対的パスへの拡張
  浮動要素の指定を含む柔軟な変換仕様の記述のためには、パスの記述を絶対的パスに加えて相対的パスを採用する必要がある。この場合の処理プロセスについて述べる。

7.1 旧文書からの要素の抽出

割り当て規則では、各新要素を構成する旧要素がそれぞれパス形式で記されている。そこで、ここに記されている旧要素は新文書を構成するのに必要な要素であるので、これを旧文書から抽出して以降の処理段階に入力する。この判断は、旧文書中でそのとき見ている論理要素が仕様の旧要素の部分に記述されているものとパスの名前が一致するかどうかを基準にする。

この処理は一般の構文解析と同様に、各要素インスタンスが論理構造上のどこに位置するのか、ID を割り振りパスを形成しながら割り当て規則に基づいてパスを評価する。

旧文書を解析している時、現在解析している論理構造上の点を表すパスを「カレントパス」と呼ぶ。

アルゴリズム 1.1: カレントパスを設定する。

1. 初期状態ではカレントパスは空とし、旧文書の冒頭から解析を始める。
2. 開始タグが出現した時：
   (a) その時出現した、つまり開始タグに名前が記述されているコンポーネント C'に対して、
   - その種類の中では今までのものと重ならないような、適当な ID、つまり $C'.id$ を割り振る。
   - $C'.name$ をセットする。

   これをカレントパスの最下位に追加する。それ以前のカレントパスを P'_{prev}とすると、この段階でのカレントパスは $P'_{prev}C'$となる。

   (b) これらで表されるその時の旧要素インスタンスは、次の開始タグあるいは終了タグまでの間にあるデータを値とする。

   この旧要素インスタンスをアルゴリズム 1.2 によって評価する。
3. 終了タグが出現した時：
   (a) その時に閉じた（終了タグに名前の記述してある）コンポーネント C' はカレントパスの最下位であるので、それをカレントパスから削除する。つまり、それまでのカレントパスが $P'C'$であったとすると、カレントパスは P'になる。
4. カレントパスが再び空になった時、文書の末尾まで到達したと判断して処理を終了する。

アルゴリズム 1.1 終

アルゴリズム 1.2: 旧要素インスタンスが必要かどうか評価する。

1. カレントパス P'を評価する。各割り当て規則の旧要素部 A'_iに対して、
   - A'_iが組み合わせを持たない場合（つまり $A'_i.comb = NOT$）
     $A'_i.path = P'_{0SUNO}$として、P'_{0SUNO}と P'が名前が同じであること。
   - A'_iが組み合わせを持つ場合（つまり $A'_i.comb = COMB$）
     $A'_i.path = P'_{0SUNO} : P'_{1COMB} : \cdots : P'_{nCOMB}$として、カレントパス P'と、A'_i中のどれか一つの $P'_{kCOMB}(1 \leq k \leq n)$ を用いた $P'_{0SUNO}P'_{kCOMB}$とが名前が同じであること。
2. D'が上の 2 つのうち 1 つの条件を満たしたら、この旧要素インスタンスは新文書でも必要と判断され、カレントパスで定まっていなかったコンポーネントの属性について、

- カレントパスのコンポーネントには、割り当て規則 A'_iのコンポーネントから優位性 $C'.sup$ を写す。

という処理を行い、このカレントパスに基づき次のような旧要素インスタンス D'がセットされる。

- 値を整形して $D'.val$を整形する。
- 必要と判断した割り当て規則の番号は $D'.ass = i$。
- 論理構造上の位置を表すパスは $D'.path = P'$

この要素インスタンスは次のステップの入力となる。

アルゴリズム 1.2 終

本稿の例では、抽出結果を図 7.1に示す。
この図では、抽出された各旧要素を一行でパス表現にして、各行は、

- 条件を満たした割り当て規則の番号 $D'.ass$
- パス $D'.path$
  各コンポーネントについて、
  - 名前 $C'.name$
  - ID $C'.id$
  - 優位性 $C'.sup$

  の組をコンポーネントの数だけ。
- 値 $D'.val$

と並べて記述している。
たとえばこの中の *1*

"/Recordlist1/Record1/Title1 ... Doc1"

は、パスの名前が "/Recordlist/Record/Title" であり、割り当て規則の *0* 番の旧要素部

"/Recordlist /Recordlist !/Title"

と名前が一致していることから抜き出され、その値は "Doc1" である。パス中のそれぞれのコンポーネントは（一つめの旧要素インスタンスであったことから）すべて 1 という ID を割り振られ、優位性は割り当て規則から写して、ルートからそれぞれ優位である・優位である・優位ではないと決定された。

```
D'.ass : {C'.name:C'.id:C'.sup }+ :D'.val
(C'.sup:  1=SUP 0=NOT -1=COMB)

 1.  0: /Recordlist1:1 /Record1:1 /Title1:0 : Doc1
 2.  5: /Recordlist1:1 /Record1:1 /Author1:0 /Surname1:-1 : Smith
 3.  5: /Recordlist1:1 /Record1:1 /Author1:0 /First-name1:-1 : B.
 4.  1: /Recordlist1:1 /Record1:1 /Author1:0 /Affiliation1:0: NACSIS
 5.  5: /Recordlist1:1 /Record1:1 /Author2:0 /Surname2:-1 : Jones
 6.  5: /Recordlist1:1 /Record1:1 /Author2:0 /Surname3:-1 : Nelson
 7.  5: /Recordlist1:1 /Record1:1 /Author2:0 /Firstname2:-1 : M.
 8.  1: /Recordlist1:1 /Record1:1 /Author2:0 /Affiliation2:0 : NACSIS
 9.  2: /Recordlist1:1 /Record1:1 /Key1:1 : structure
10.  2: /Recordlist1:1 /Record1:1 /Key2:1 : database
11.  3: /Recordlist1:1 /Record1:1 /Abst1:1 /Para1:0 : Study of
12.  4: /Recordlist1:1 /Record1:1 /Abst1:1 /Para1:0 /Ref1:1 : Book1
13.  3: /Recordlist1:1 /Record1:1 /Abst1:1 /Para1:0 : is referred.
14.  0: /Recordlist1:1 /Record2:1 /Title2:0 : Doc2
15.  5: /Recordlist1:1 /Record2:1 /Author3:0 /Firstname3:-1 : M.
16.  5: /Recordlist1:1 /Record2:1 /Author3:0 /Surname4:-1 : Brown
17.  1: /Recordlist1:1 /Record2:1 /Author3:0 /Affiliation3:0 : Univ. of Tokyo
```

図 7.1: 旧文書からの要素の抽出結果（左端の数字は説明のための番号）

7.2 抽出した旧要素の整理

続いて、抽出した旧要素インスタンスを、繰り返しや組み合わせを整理して、新要素インスタンスと一対一にする。この処理は、第6.6節でデータの整形に関連して述べたと同様の理由で、三段階に分けて行う。つまり、組み合わせがない場合には繰り返しのみを考慮すればよいが、組み合わせがある場合には、繰り返しと組み合わせは再帰的に起こる可能性があることから、以下の三段階を考える。

1. 組み合わせる要素インスタンスの繰り返し
2. 組み合わせ
3. 繰り返し

可能性としてはさらに繰り返しと組み合わせが再帰的に出現する可能性もある。たとえば、「組み合わせの組み合わせ（以下反復）」などである。しかしここでは、処理のコストと現実的な文書の形態、また処理を指定する人間の負荷から考えて、この三段階にとどめる。

組み合わせのない場合は3. でまとめて処理する。

7.2.1 組み合わせる旧要素の繰り返しの処理

複数の旧要素から一つの新要素を構成するケースを組み合わせと呼んだ。たとえば、旧文書では著者名が姓と名をそれぞれわけて保持する要素を設けていたのに対して、新文書では著者名はそれ全体で一つの要素とする場合である。

複数の旧要素を組み合わせる時、その各々について繰り返しが起こる可能性がある。たとえば、著者名は通常一つの姓と一つの名前から構成され、これらを組み合わせて一つの著者名を作ればよいが、改姓などをした著者のために二つ以上の姓を用いたいというケースも考えられる。このようなケースでは組み合わせ処理を容易にするために、この段階で繰り返しをまとめてしまう。ここで繰り返しとしてまとめる際、組み合わせる要素の範囲を超えてはいけないので、共通部分パスに関する条件は組み合わせる場合と同じにする。ここでは、共通部分パスは名前もIDも一致する旧要素インスタンスであることを条件とする。

この判断を行う基準は、組み合わせのある割り当て規則の中で記述されている旧要素と同じ名前のパスを持つ要素同志が、つまり全体に渡ってそれら間でも名前は同じであるが、共通部分パスでは加えてID($C'.id$)が同じであることである。

たとえば、割り当て規則の *5*

`“/recordlist /record !/author; /Recordlist /Record !/Author; Surname; firstname”`

について、二つの旧要素インスタンス

`“/Recordlist1/Record1/Author2/Surname2”`
`“/Recordlist1/Record1/Author2/Surname3”`

はまとめるものと判断される。なぜならば、共通部分 `“/Recordlist/Record/Author”` ではコンポーネントの名前とIDが同じで、残りの部分 `“Surname”` の名前が同じだからである。一方旧要素インスタンス

`“/Recordlist1/Record2/Author3/Surname4”`

はまとめられない。なぜならば、パスの共通部分 `“Author”` が異るIDを持つからである。

この過程を形式的に記述すると次のようになる：

アルゴリズム 2.1: 組み合わされる要素の繰り返しを判断する。

1. パスを評価する。

   (a) i を組み合わせを持つ割り当て規則の番号、つまり $A'_i.comb = COMB$として、$D'_j.ass = D'_k.ass = i$ である旧要素 D'_j と D'_k [1]に関して、

   (b) A'_iに記述されている旧要素を $A'_i.path = P'_{0SUNO} : P'_{1COMB} : \cdots : P'_{2COMB}$とする。つまり共通部分パスが P'_{0SUNO}で、残りのパスが P'_{1COMB}と P'_{2COMB}である。その中の一つの $P'_{hCOMB}(h = 1 or 2)$ について、$D'_j.path$ と $D'_k.path$ がパス全体に渡って名前が同じ $P'_{0SUNO}P'_{hCOMB}$であり、

   (c) さらに、$D'_j.path$ と $D'_k.path$ の共通部分の ID が同じである。

   これらを満たす時、D'_j と D'_kは組み合わせる要素の繰り返しであると判断される。

2. 旧要素インスタンスの値 $D'_j.val$と $D'_k.val$は指定に従って整形され、$D'_j.val$に格納する。D'_kは削除する。

アルゴリズム 2.1 終

本稿の例では、組み合わせる要素の繰り返しについての処理結果は図 7.2に示される。図 7.1の 5 と 6 の要素

"/Recordlist1/Record1/Author2/Surname2 ... Jones"
"/Recordlist1/Record1/Author2/Surname3 ... Nelson"

とがまとめられて、図 7.2の 5 の要素インスタンス

"/Recordlist1/Record1/Author2/Surname2 ... Jones[Nelson]"

になっている。これらは共通部分 "/Recordlist/Record/Author" の名前と ID が同じで、残りの部分 "Surname" の名前が同じである。

7.2.2 組み合わせの処理

組み合わせでは、複数の旧要素、たとえば著者の姓と名であったものをまとめて一つの新要素を構成する段階に近づけるという処理を行うが、この際、たとえば同じ著者の姓と名前は組み合わせるが、違う著者の姓と名前はまとめない、というような指定がしたい。そこで、旧要素の組み合わせを判断するためには、パスの共通部分の名前と ID が同じであることが基準となる。割り当て規則の *5*

"/recordlist /record !/author; /Recordlist /Record !/Author/; Surname; Firstname"

によれば、二つの旧要素

"/Recordlist1/Record1/Author1/Surname1"
"/Recordlist1/Record1/Author1/Firstname1"

は組み合わせられる。それは共通部分 "/Recordlist/Record/Author" は名前と ID が同じで、残りが割り当て規則にある "Surname" と "Firstname" だからである。一方、旧要素

"/Recordlist1/Record1/Author2/Firstname2"

[1] ここでは二つの要素に関して述べているが、さらに多くの要素が関与する場合でも手法は同様に適用される。

```
D'.ass : {C'.name:C'.id:C'.sup }+ :D'.val
(C'.sup:  1=SUP 0=NOT -1=COMB)

 1.  0: /Recordlist:1:1 /Record:1:1 /Title:1:0 : Doc1
 2.  5: /Recordlist:1:1 /Record:1:1 /Author:1:0 /Surname:1:-1 : Smith
 3.  5: /Recordlist:1:1 /Record:1:1 /Author:1:0 /Firstname:1:-1 : B
 4.  1: /Recordlist:1:1 /Record:1:1 /Author:1:0 /Affiliation:1:0 : NACSIS.
 5.  5: /Recordlist:1:1 /Record:1:1 /Author:2:0 /Surname:2:-1 : Jones[Nelson].
 6.  5: /Recordlist:1:1 /Record:1:1 /Author:2:0 /Firstname:2:-1 : M
 7.  1: /Recordlist:1:1 /Record:1:1 /Author:2:0 /Affiliation:2:0 : NACSIS
 8.  2: /Recordlist:1:1 /Record:1:1 /Key1:1 : structure
 9.  2: /Recordlist:1:1 /Record:1:1 /Key2:1 : database
10.  3: /Recordlist:1:1 /Record:1:1 /Abst1:1 /Para1:0 : Study of
11.  4: /Recordlist:1:1 /Record:1:1 /Abst1:1 /Para1:0 /Ref1:1 : Book1
12.  3: /Recordlist:1:1 /Record:1:1 /Abst1:1 /Para1:0 : is referred.
13.  0: /Recordlist:1:1 /Record:2:1 /Title:2:0 : Doc2
14.  5: /Recordlist:1:1 /Record:2:1 /Author:3:0 /Firstname:3:-1 : M
15.  5: /Recordlist:1:1 /Record:2:1 /Author:3:0 /Surname:4:-1 : Brown
16.  1: /Recordlist:1:1 /Record:2:1 /Author:3:0 /Affiliation:3:0 : Univ. of Tokyo
```

図 7.2: 組み合わせる要素の繰り返しの処理結果

は組み合わせられない。なぜならば、共通部分で "Author" の ID が異なるからである。

ここではアルゴリズム 2.1 の処理により、組み合わせに関与する旧要素の各々には、繰り返しは最早起こらない。

形式的にはこの処理は次のように記される：

アルゴリズム 2.2: 組み合わせを判断する。

1. パスを評価する。

   (a) i を組み合わせのある割り当て規則の番号、つまり $A'_i.comb = COMB$として、$D'_j.ass = D'_k.ass = i$ である D'_jと D'_kに関して、

   (b) 割り当て規則の旧要素部に記述してある $A'_i.path = P'_{0SUNO} : P'_{1COMB} : P'_{2COMB}$とする。つまり共通部分が P'_{0SUNO}で、残りの部分が P'_{1COMB}と P'_{2COMB}である。このとき、$D'_j.path = P'_{0SUNO}P'_{1COMB}$でありかつ $D'_k.path = P'_{0SUNO}P'_{2COMB}$、またはその逆である、つまり、$D'_j$と D'_kはそれぞれ A'_iにある旧要素群の各々であり、

   (c) さらに D'_jと D'_kの共通部分の ID が同じである。

   これらを満たす時、D'_jと D'_kは組み合わせる要素群であると判断される。

2. 旧要素群の値 $D'_j.val$と $D'_k.val$は指定に従って整形し、$D'_j.val$に代入する。また、共通部分のパスのみを新たに $D'_j.path$ として、この D'_jは次の段階に送られる。つまり、新しいパスは P_{0SUNO}のみとなる。D'_kは削除する。

```
D'.ass : {C'.name:C'.id:C'.sup }+ :D'.val
(C'.sup:  1=SUP 0=NOT -1=COMB)

 1.  0: /Recordlist:1:1 /Record:1:1 /Title:1:0 : Doc1
 2.  5: /Recordlist:1:1 /Record:1:1 /Author:1:0 : Smith, B
 3.  1: /Recordlist:1:1 /Record:1:1 /Author:1:0 /Affiliation:1:0 : NACSIS
 4.  5: /Recordlist:1:1 /Record:1:1 /Author:2:0 : Jones[Nelson], M
 5.  1: /Recordlist:1:1 /Record:1:1 /Author:2:0 /Affiliation:2:0 : NACSIS
 6.  2: /Recordlist:1:1 /Record:1:1 /Key1:1 : structure
 7.  2: /Recordlist:1:1 /Record:1:1 /Key2:1 : database
 8.  3: /Recordlist:1:1 /Record:1:1 /Abst1:1 /Para1:0 : Study of
 9.  4: /Recordlist:1:1 /Record:1:1 /Abst1:1 /Para1:0 /Ref1:1 : Book1
10.  3: /Recordlist:1:1 /Record:1:1 /Abst1:1 /Para1:0 : is referred.
11.  0: /Recordlist:1:1 /Record:2:1 /Title:2:0 : Doc2
12.  5: /Recordlist:1:1 /Record:2:1 /Author:3:0 : Brown, M
13.  1: /Recordlist:1:1 /Record:2:1 /Author:3:0 /Affiliation:3:0 : Univ. of Tokyo
```

図 7.3: 組み合わせの処理結果

アルゴリズム 2.2 終

本稿の例では、組み合わせ処理の結果は図 7.3に示される。図 7.2の 2 と 3

"/Recordlist1/Record1/Author1/Surname1 ... Smith"
"/Recordlist1/Record1/Author1/Firstname1 ... B."

が割り当て規則 *5* にある要素で、共通部分の ID も同じなのでまとめられて、図 7.3の 2

"/Recordlist1/Record1/Author1 ... Smith, B"

になる。パスは共通部分のみになり、データも整形されている。

7.2.3 繰り返しの処理

旧文書で複数回現れる内容が新文書では一度しか現れてはいけないように定義されている可能性がある。本稿の例では、著者の所属は旧文書では一つのレコードに二回現れているが、新文書では一度にまとめる。この処理の必要の有無は、割り当て規則でコンポーネントに記述した優位性によって表現される。

繰り返しである旧要素インスタンスを判断するには、パス全体で名前が同じで、さらに優位なコンポーネントの ID も同じ旧要素インスタンス群を見つけ出す。たとえば割り当て規則の *1*

"/recordlist /record !/affiliation; /Recordlist /Record !Author !/Affiliation,"

によると、二つの旧要素インスタンス

"/Recordlist1/Record1/Author1/Affiliation1"
"/Recordlist1/Record1/Author2/Affiliation2"

はまとめる繰り返しである。なぜならば、優位なするコンポーネント "Recordlist" と "Record" の名前とIDが同じで、優位でないコンポーネント "Affiliation" は名前が同じだからである。一方で、旧要素インスタンス

"/Recordlist1/Record2/Author3/Affiliation3"

は、優位なコンポーネント "Record" のIDが異るのでまとめない。

ただし、パスのルートからリーフまでコンポーネントの名前とIDが一致するもの同志は、同一の要素インスタンスで途中に他の要素インスタンスを挟んだものである。

```
/Recordlist1/Record1/Abst1/Para1 ...  Study of
/Recordlist1/Record1/Abst1/Para1/Ref1 ...  Book1
/Recordlist1/Record1/Abst1/Para1 ...  is referred.
```

このようなものは繰り返しではなく、除外する。

形式的にはこの処理は次のように記述される：

アルゴリズム 2.3: 繰り返しを判断する。

1. パスを評価する。
   (a) 同じ割り当て規則に関連している旧要素インスタンス、つまり $D'_j.ass = D'_k.ass = i$ である D'_j と D'_k について、
   (b) D'_j と D'_k がパス全体で名前が同じで、
   (c) さらに優位である $(C'.sup = SUP)$ コンポーネントは、ID$(C'.id)$ も同じである。
   (d) ただし、パス全体でIDも一致するものは除く。

   このとき、D'_j と D'_k はまとめられる繰り返しであると判断される。
2. 要素の値 $D'_j.val$ と $D'_k.val$ とは指定に従って整形され、$D'_j.val$ とする。D'_k は削除する。

アルゴリズム 2.3 終

本稿の例では、繰り返しの処理結果は図7.4のようになる。図7.3の3と4、

```
"/Recordlist1/Record1/Author1/Affiliation1 ...  NACSIS"
"/Recordlist1/Record1/Author2/Affiliation2 ...  NACSIS"
```

が、全体に渡って名前が同じで優位なコンポーネント "Recordlist" "Record" のIDが同じことからまとめられて、図7.4の3、

```
"/Recordlist1/Record1/Author1/Affiliation1 ...  NACSIS / NACSIS"
```

になる。データも整形されている。

```
D'.ass : {C'.name:C'.id:C'.sup }+ :D'.val
(C'.sup:  1=SUP 0=NOT -1=COMB)

 1.  0: /Recordlist1:1 /Record1:1 /Title1:0 : Doc1
 2.  5: /Recordlist1:1 /Record1:1 /Author1:0 : Smith, B / Jones[Nelson], M
 3.  1: /Recordlist1:1 /Record1:1 /Author1:0 /Affiliation1:0 : NACSIS / NACSIS
 4.  2: /Recordlist1:1 /Record1:1 /Key1:1 : structure
 5.  2: /Recordlist1:1 /Record1:1 /Key2:1 : database
 6.  3: /Recordlist1:1 /Record1:1 /Abst1:1 /Para1:0 : Study of
 7.  4: /Recordlist1:1 /Record1:1 /Abst1:1 /Para1:0 /Ref1:1 : Book1
 8.  3: /Recordlist1:1 /Record1:1 /Abst1:1 /Para1:0 : is referred.
 9.  0: /Recordlist1:1 /Record2:1 /Title2:0 : Doc2
10.  5: /Recordlist1:1 /Record2:1 /Author3:0 : Brown, M
11.  2: /Recordlist1:1 /Record2:1 /Author3:0 /Affiliation3:0 : Univ. of Tokyo
```

図 7.4: 繰り返しの処理結果

旧要素インスタンス/A1/B1/C2　から　新要素インスタンス/a1/b1/c2のIDを決定した
旧要素インスタンス/A1/B1/F1　から　新要素インスタンス/a/b/fのIDを決定する

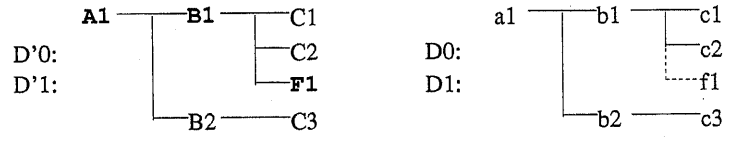


図 7.5: 新要素の ID の決定（優位なコンポーネントのみを記す）

7.3 新要素の設定と新文書の受理

ここまでの処理により、一つの旧要素インスタンスは一つの新要素インスタンスに相当するようになっている。従って新文書を構成する要素インスタンスは、旧要素インスタンスの各々から作られる。

まずパスのコンポーネントの名前は、各旧要素インスタンスに関連する割り当て規則に記述された新要素のパスから、そのパスと同じに決定する。続いて ID はコンポーネントの優位性を利用して決める。

ここでコンポーネントの ID が持つ意味を、木構造と対照させて考える。同じ名前と ID を持つコンポーネントは論理構造木で一つのノードを共有していたことを意味し、文書の構成や論理構造のまとまりを表す。旧要素でノードを共有していた要素インスタンスに対応する新要素インスタンスは、特に指定がない限り、新文書でもノードを共有してなくてはならない。この、「旧要素インスタンスでもともと共有していたノードであれば、対応する新要素インスタンスのコンポーネントも共有しているように、一方もともと共有していなければ共有しないコンポーネントとなるように考慮する」という指定がコンポーネントの優位性である。反対に優位でないコンポーネントは、前後のコンポーネントに合わせる、という意味であると言える。

そこで新要素のパスの ID は、そのもととなる旧要素インスタンスについて、それ以前に処理の終わった旧要素インスタンスの中から旧文書中で最も多くのノードを共有していた、つまり最も多く優位なコンポーネントが一致するものを見つけ出し、その新要素インスタンスから作られた新要素に準じて ID を定める。この時、既に処理された旧要素インスタンス群から選ばれたものとそれに対応して作られた新要素インスタンスを、**手がかり（新旧）要素インスタンス**と呼ぶ。

たとえば図 7.5の状況を考える。この図では優位なコンポーネントのみを記しているものとする。左の論理構造木中での旧要素インスタンス "/A1/B1/F1"（D'_1とする）の位置は "/A1/B1/C2"（D'_0とする）に近い、つまり多くのノードを共有している。そこで D'_1に対応して作られる新要素インスタンス "/a/b/c"（D_1とする。）の位置も、D'_0に対応して作られる "/a1/b1/c2"（D_0とする。）の近くであることが求められる。

このためには、既に処理の終わった旧要素インスタンスでなるべく論理構造上近いものを見つけて、その旧要素インスタンスにより決定した新要素インスタンスに、今作る新要素インスタンスも近くなるよう ID を決定する。つまり、D'_0の優位なコンポーネント "A, B, F" のうち、D'_1と名前 ID ともに共通な "A, B" が、新要素インスタンスの "a, b" の ID を決定したので、D_1でも "a, b" の ID を同じにする。

また最も下位の共有する優位なコンポーネントからその次の優位なコンポーネントまでの間にある優位でないコンポーネントについては、今考慮している旧要素インスタンスと手がかり旧要素インスタンスの関係とはコンポーネントの名前や数が同じである保証はないので参考にすることは基本的にできないが、「特に違いを反映するよう指定されていないので、共有していても構わない」という判断により、手がかり新要素インスタンスとコンポーネントの名前が同じであれば ID を受け継ぐ。

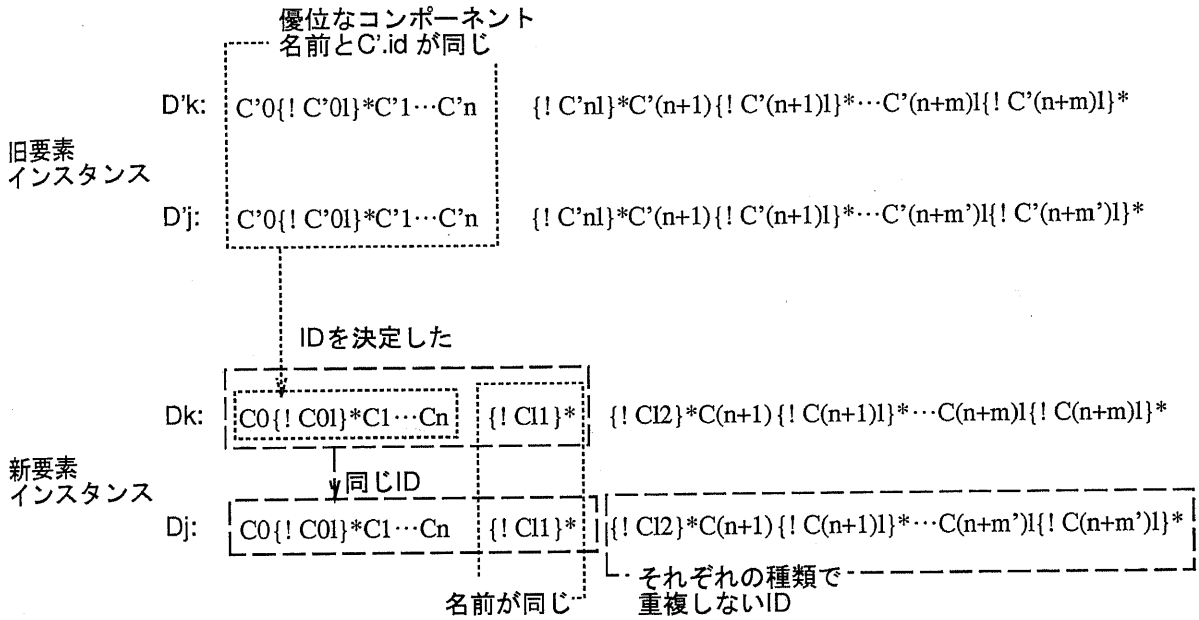


図 7.6: 新要素インスタンスのパスの ID の決定

この処理に用いる手がかり旧要素インスタンスとしては、名前も ID もともに一致する優位なコンポーネントがルートから一番多くて、新要素インスタンスでそれに相当する新コンポーネントより先の優位でないコンポーネントがなるべく多くこれから作る新要素インスタンスと名前が同じものを選べばよい。

この処理は、形式的には以下のように記述される：

アルゴリズム 3: 新要素インスタンスを設定する。

1. 今から処理する旧要素インスタンス D'_jが関連する割り当て規則の番号 $D'.ass = i$ を持つとして、今から作る新要素インスタンス D_jのパスは A_iと名前を同じにする。

2. 新要素インスタンス D_jのパスの ID を決定するために、既に処理済みである旧要素インスタンスから手がかりとなる新旧要素インスタンス D_k、D'_kを探す（図 7.6）。

   ここからの記述では優位なコンポーネントに着目するため、優位なコンポーネントの間に 0 個以上の優位でないコンポーネントが存在する、という表記をとる。変換規則の記述に倣って、“!” の付与されているコンポーネントが優位でないものとする。つまり、これか処理する新旧要素インスタンスと手がかり新旧要素インスタンスは次のように表せる。

$$
\begin{aligned}
D'_k.path &= C'_0\{!C'_{0l}\}^*C'_1\cdots C'_n \quad \{!C'_{nl}\}^*C'_{(n+1)}\{!C'_{(n+1)l}\}^*\cdots C'_{(n+m)}\{!C'_{(n+m)l}\}^* \\
D'_j.path &= C'_0\{!C'_{0l}\}^*C'_1\cdots C'_n \quad \{!C'_{nl}\}^*C'_{(n+1)}\{!C'_{(n+1)l}\}^*\cdots C'_{(n+m')}\{!C'_{(n+m')l}\}^* \\
&\downarrow \\
D_k.path &= C_0\{!C_{0l}\}^*C_1\cdots C_n\{!C_{nl_1}\}^* \quad \{!C_{nl_2}\}^*C_{(n+1)}\cdots C_{(n+m)}\{!C_{(n+m)l}\}^* \\
D_j.path &= C_0\{!C_{0l}\}^*C_1\cdots C_n\{!C_{nl_1}\}^* \quad \{!C_{nl_2}\}^*C_{(n+1)}\cdots C_{(n+m')}\{!C'_{(n+m')l}\}^*
\end{aligned}
$$

   これらの条件は次の通り。

```
C.ass:  {C.name:C.id:C.sup}+  :D.val
(C.sup:  1=SUP 0=NOT)

 1.  0: /recordlist0:1 /record0:1 /subject0:0 : Doc1
 2.  5: /recordlist0:1 /record0:1 /author0:0 : Smith, B. / Jones[Nelson], M.
 3.  1: /recordlist0:1 /record0:1 /affiliation0:0 : NACSIS/ NACSIS
 4.  2: /recordlist0:1 /record0:1 /other0:0 /key0:1 : structure
 5.  2: /recordlist0:1 /record0:1 /other0:0 /key1:1 : database
 6.  3: /recordlist0:1 /record0:1 /other0:0 /abst0:1 : Study of
 7.  4: /recordlist0:1 /record0:1 /other0:0 /abst0:1 /ref0:1 : Book1
 8.  3: /recordlist0:1 /record0:1 /other0:0 /abst0:1 : is referred.
 9.  0: /recordlist0:1 /record1:1 /subject1:0 : Doc2
10.  5: /recordlist0:1 /record1:1 /author1:0 : Brown, M.
11.  2: /recordlist0:1 /record1:1 /affiliation1:0 : Univ. of Tokyo
```

図 7.7: 新要素の設定結果

- D'_kから D_kが、D'_jから D_jが、それぞれ作られた。
- D'_kと D'_jに関して、$C'_0 \cdots C'_n$の $n+1$ 個の優位なコンポーネントの名前と ID が共通、一方 $C'_{(n+1)} \cdots C'_{(n+m)}$と $C'_{(n+1)} \cdots C'_{(n+m')}$はそうでない。
- D_kの、$C_0 \cdots C_n$は D'_kの $C_0 \cdots C'_n$により、また $C_{(n+1)} \cdots C_{(n+m)}$は $C'_{(n+1)} \cdots C'_{(n+m)}$により、決定された部分パスである。
- D_kの $\{!C_{nl_1}\}^*$と D_jの $\{!C_{nl_1}\}^*$は名前の一致する範囲が最も広い。(この範囲は、優位なコンポーネントを含まない。)

このような D'_kと D_kを手がかり新旧要素インスタンスとする。

3. 次のように D_jの ID を決定する。
   - 次の二つ
     - D'_kと D'_jの優位なコンポーネント同志が名前と ID を共有している範囲の部分パス $C'_0 \cdots C'_n$に対応している $C_0\{!C_{0l}\}^*C_1 \cdots C_n$
     - それに続いて D_kと D_jの優位でないコンポーネントが名前が同じである $\{!C_{nl_1}\}^*$

     は、その範囲である D_kの部分パスの $C_0\{!C_{0l}\}^*C_1 \cdots C_n\{!C_{nl_1}\}^*$と同じ ID を写す。
   - 残りの部分パス $\{!C_{nl_2}\}^*C_{(n+1)} \cdots C_{(n+m')}\{!C'_{(n+m')l}\}^*$は、それぞれのコンポーネントの種類で、それまでと重複しない新しい ID を割り振る。
4. D_jの値は、$D'_j.val$として既に所望の形式に整形されているので、これを写してくる。

アルゴリズム 3. 終

本稿の例について、処理結果は図 7.7のようになる。
具体的に、この処理を考える。たとえば図 7.4の *1* から *6* の処理が終わっているとして、

"*7:* /Recordlist1/Record1/Abst1/Para1/Ref1 ... Book1"

について検討する場面を考える。これがアルゴリズム 3 の D'_jにあたる。すると、これが関連してきた割り当て規則は

"*4:* /recordlist /record !/other /abst /ref ; /Recordlist /Record /Abst !/Para /Ref"

なので、今から作る新要素インスタンス D_jの名前は、

"/recordlist/record/other/abst/ref"

と決定する。

それまでに処理の終わっている旧要素インスタンスの中で、D'_jとなるべく多くの優位なコンポーネントの名前と ID が同じものを選ぶと、

"*6:* /Recordlist1/Record1/Abst1/Para1 ... Study of"

が、"Recordlist1" "Record2" "Abst1" の三つの優位な コンポーネントが D'_jと共通なことから候補になり、他にはないので D'_kに決定する。

ここで、一つの旧要素インスタンスから一つの新要素インスタンスが作られているので、既に設定の終わっている新要素インスタンスは、同様に図 7.7の *1* から *6* が作られており、今決定した D'_kから作られた新要素インスタンスを見てみると、

"*6:* /recordlist0/record1/other0/abst0 ... Study of"

となっており、これが D_kである。これの、D'_kと D'_jの共通な優位なコンポーネントから定められた部分は "/recordlist0/record1" であり、D'_kの次の優位なコンポーネント（ここでは存在しない）以前の優位ではないコンポーネントは、"other0" "abst0" である。一方 D_jの同様のコンポーネントは "other" "abst" なので、名前が共通するコンポーネントとはこれらの二つに決定する。

D_jのパスの ID を決定する。ここまでの考察により、アルゴリズムに従うと D'_jと D'_kが共有していたコンポーネント "Recordlist" "Record" から決定された D_kのコンポーネント "recordlist" "record" と、さらにそれ以降で次の優位なコンポーネント以前の D_kと D_jで名前が同じであるコンポーネント "other" "abst" は、D_jは D_kと同じ ID にして、"/recordlist0/record0/other0/abst0" と決定する。残りの "ref" は倣うべきものがないので、それまでにこのコンポーネント種 "ref" が使っていない ID、ここまではまだ一つも出てきていないので 0 を付与し、"ref0" と決定する。つまり

"/recordlist0/record0/other0/abst0/ref0 ... Book1"

となる。

7.4 タグ付きテキストとしての出力

設定された新要素は、パス表現から木構造を再構成し、タグを付与されたマークアップ文書として出力する。

新文書を構成するためのデータは中間ノードの名前とIDをパスの形で持っているので、同じ名前で同じIDのノードを束ねていくことにより木で表される論理構造を再構成することができる。

ここでも再構成している論理構造の現時点をカレントパスと呼ぶ。あるカレントパスを基準に論理構造を考慮するのは、カレントパスと比較して一つ短いパスが名前とIDが同じであるケースである。これには三通りある。ここでいうコンポーネントの同異は、厳密に名前もIDも同じかどうかという意味を指す。

- ここまでのパスは終端まで同じで、長さがさらに長い
  たとえば、カレントパス

  "/recordlist0/record0/other0/abst0 ... Study of"

  に対して

  "/recordlist0/record0/other0/abst0/ref0 ... Book1"

  テキスト中に埋め込まれた別の要素インスタンスである。カレントパスからその階層までの間にある要素インスタンスについて同様の三種の処理をして、その結果であるデータはそのまま連結する。

  上の例では "abst" より深い階層を処理した結果の "<ref>Book1</ref>" を今調べている要素インスタンスのデータに連結する。

- パスの長さが同じで、終端のコンポーネントが同じ
  たとえば、カレントパス

  "/recordlist0/record0/other0/abst0 ... Study of"

  に対して

  "/recordlist0/record0/other0/abst0 ... is referred."

  間にここまでのパスが全く同じでさらに長いパスを持つ、前項のような要素インスタンスがあったはずである。データはそのまま連結する。

  上の例では、現在作っているデータに "is referred" を連結する。

- パスの長さが同じで、終端のコンポーネントが異なる
  たとえば、カレントパス

  "/recordlist0/record0/subject0 ... Doc1"

  に対して

  "/recordlist0/record0/author0 ... Smith, B. / Jones[Nelson],M"

  同じ深さにある別の要素。上記の処理をして整えて、タグをつけて連結する。

  上の例では、"author0" 以下を処理した結果が "<author>Smith, B. / Jones[Nelson],M</author>" であれば、これを連結する。

従って、まずカレントパスの終端まで同じで、同じ長さまたはさらに長い要素インスタンスを見つけて処理し、それが終わったらその終端コンポーネントの名前によるタグで括って、続いて一つ短いパスまで名前と ID が同じ要素インスタンスを見つけ出して処理する、という動作を再帰的に行う。

その処理は形式的には以下のようになる。

アルゴリズム 4: 新文書を構成して、タグ付きテキストとして出力する

1. 設定された要素群の中から適当なものを始点として選ぶ。
2. 以下に当てはまるものを順次探し、なくなるまで処理する。
   (a) 今選んでいる要素インスタンスのパスをカレントパスにする。この段階の値を形成する変数 V_{LOC} を設け、選んだ要素インスタンスの値を写す。
   (b) 処理される新要素群のうち、カレントパスと末端までパスの名前も ID もともに一致するものを選ぶ。
   - 選んだ要素の方がパスが長い時
     このときは、文字列中に埋め込まれたさらに下位にまでパスを持つ要素である。*(a)* へ入る。その戻り値を V_{LOC} に連結する。
   - 選んだ要素とパスの長さが同じ時
     この時は、途中に図などを挟んだ文書の続きなど、カレントパスにした要素の続きである。値をこのレベルの V_{LOC} にそのまま連結し、処理を待つ要素群から除く。

   このような要素インスタンスがなくなったら、この項でできた値にカレントパスの終端コンポーネントの名前を用いたタグを前後に付与して、次へ進む。

   (c) カレントパスよりも一つ短いパスと名前と ID が一致する要素インスタンスを選ぶ
   (a) に入る。戻り値を連結する。

   これらの条件を満たす要素がなくなったらカレントパスを一つ短くして、この階層でできた値を返り値として一段戻る。
3. 処理を待つ新要素がなくなったらカレントパスも長さが 0 になっているはずであり、ルートに当たるコンポーネントの名前によるタグを全体に付して、最終的に出力する。

アルゴリズム 4 終

例では、出力結果は図 7.8のようになる。

最初のカレントパスは要素インスタンスリストの先頭にあった

"*1:* /recordlist0/record0/subject0"

とし、終端まで同じ要素インスタンスは存在しないので、"<subject>Doc1</subject>" が形成され、カレントパスよりも一つ短いパスが同じである、

"*2:* /recordlist0/record0/author0 ... Smith, B. / Jones[Nelson],M"
"*3:* /recordlist0/record0/affiliation0 ... NACSIS / NACSIS"

が、それぞれ「そこまで同じでそれより深い」要素インスタンスがないのでそのままタグをつけて、"<author>Smith, B. / Jones[Nelson],M</author><affiliation>NACSIS / NACSIS</affiliation>" となり、それまでのデータに連結される。

ところが処理が次の

```
<recordlist><record><author>Smith, B. / James[Nelson], M</author>
<affiliation>NACSIS / NACSIS</affiliation>
<subject>Doc1</subject>
<other><key>structure</key>
<key>database</key>
<abst>Study of<ref>Book1</ref>is referred.</abst></other></record>
<record><author>Brown, M</author>
<affiliation>Univ. of Tokyo</affiliation>
<subject>Doc2</subject>
</record></recordlist>
```

図 7.8: タグ付き文書としての出力結果

"*4:* /recordlist0/record0/other0/key0 ... structure"

に来ると、これはカレントパスより長いので、あらたにこれをカレントパスとする階層のルーチンとして他の要素を処理する。すると再びここまでの階層に戻ってくる時までに結果は、"<other><key>structure</key><key>database</key><abst>Study of<ref>Book1</ref>is referred.</abst></other>" と得られるので、これをこれまでの結果に連結する。

以下同様にして最終的には全体が一つのタグ付き文書として出力される。

この結果はタグの省略のない一般の SGML 文書なので、一般の SGML パーザなどに入力、あるいは新文書の論理構造で既に構築していた処理系に入力することにより、整形やデータベースでの検索など、様々な処理のデータとして活用することができる。

7.5 相対的パスへの拡張

7.5.1 相対的パス表現の必要

第2.5節でも述べたように、SGML では「浮動要素」が採用されていて、これにより柔軟な文書が記述でき、また一方で DTD での記述を容易にしているが、SGML 文書が文脈自由文法の手法ではなかなか扱いきれない一つの大きな原因になっている。

しかし図や脚注、参照など浮動要素で表すのが適当なものも多く、文書の表現の上では欠かせないので、本手法でも添加要素に対処するために処理を相対的なパスの記述を用いる方向の拡張を検討した。変換規則の記述と処理方法ついて述べる。

相対的なパスの記述による変換仕様は浮動要素の処理にとどまらず、本編や参考文献に関わらず著者名を抜き出すなど、文書の処理一般に応用することができる。

7.5.2 記述法

相対的なパス記述、つまり論理構造のルートからすべてのコンポーネントをパスとして記述せず、一部を省略する形式を用いた変換規則の記述は、第一義的に SGML の添加要素への対処として導入する。そのため、添加要素に見合う記述能力を持つことで当面の目的は達せられること、またさらに高度化した場合の処理コストを考慮して、

- パス中で一箇所のみを省略できる
- ルートは各文書の論理構造に一つであり、いつも明らかである。また最末端はデータを持つ部分であり、各要素の種別を決定的に表すものであるから、最上位や終端には省略を指定できない

の二点を条件として、0 個以上のコンポーネントの代わりにアスタリスク "*" を用いてパスを記述する。この記述は変換仕様の割り当て規則を記述する部分に適用される。たとえば、

"/論文 /本文 * /図 /図題"

という表現になり、論理構造のルートであるコンポーネント「論文」の下の本文の下のどこかにある、コンポーネント「図」の下の「図題」を表す要素を指す。

新要素・旧要素ともに、前述の理由で各パスに 1 つのアスタリスクを含むことができる。そこで、新旧それぞれのアスタリスクを含むパス表現は、次のようになる。

$$\text{新要素} \quad : \quad C_0\{!C_{0l}\}^* \cdots C_n\{!Cn_l\}^* * \{!C_{*l}\}^* C_{(n+1)}\{!C_{(n+1)l}\}^* \cdots C_{(n+m)}\{!C_{(n+m)l}\}^*$$

$$\text{旧要素} \quad : \quad C'_0\{!C'_{0i}\}^* \cdots C'_n\{!C'_{nl}\}^* * \{!C'_{*l}\}^* C'_{(n+1)}\{!C'_{(n+1)l}\}^* \cdots C'_{(n+m')}\{!C'_{(n+m')l}\}^*$$

ここで、省略された部分のパスがどのようなものであるかはまったく想定できず、また実際、いくつのどのようなコンポーネントが補完されるのかは未定であるから、省略された部分に優位なコンポーネントを仮定することはできない。従って、この形式の時点で明示されている部分のパスで、優位なコンポーネントの数は新旧同じである必要がある。

以下では新旧それぞれの要素に省略がある場合を分けて、順に考える。

7.5.3 旧要素のパスの省略

まず、旧要素に省略のある場合を考える。要素インスタンスではなく、割り当て規則の旧要素のパスを扱う局面は、

- 旧文書から要素を抽出する

だけである。このときの省略されたパスの影響を考える。

旧文書から要素を抽出する場合、必要かどうか評価されている側である、その時々の要素インスタンスのカレントパスは（ここでは、わかりやすさのために、コンポーネントを B で表す。）、

$$B'_0 B'_1 \cdots B'_n$$

と表すことができ、このパスには省略などの曖昧性はない。従って、割り当て規則の旧要素部に省略がある場合に、要素の抽出を行うアルゴリズム 1.2 を次のように変更することで、省略の影響を排除して要素の抽出を処理することができる。これは、割り当て規則に省略がある場合にアルゴリズム 1.2 から呼び出される。

アルゴリズム 1.2': 相対的なパスの記述の場合の旧文書からの要素の抽出

1. 割り当て規則の中の旧要素を、ここではコンポーネントの優位性を考慮しないので、次のように表す。

$$A'.path : C'_0 C'_1 \cdots C'_k * C'_{(k+1)} \cdots C'_n$$

またアルゴリズム 1 で作られたその時のカレントパスを、次のように表す。

$$D'.path : B'_0 B'_1 \cdots B'_m$$

$C'_0.name$ と $B'_0.name$、$C'_1.name$ と $B'_1.name$ と順次 C'_k まで比較して、それらが異なったら当てはまらないと判断する。

2. C'_k まで名前が同じで、$A'.path$ の次のコンポーネントがアスタリスク "*" になったとき、

   (a) $A'.path$ も $D'.path$ も次に調べるコンポーネントは、それぞれのパスの末尾のコンポーネント C'_n と B'_m とする。

   (b) 末端からルートの方向へ、C'_n と B'_m、$C'_{(n-1)}$ と $B'_{(m-1)}$、と名前を順次比較し、異なるものがあれば、当てはまらないと判断する。

3. $P'.path$ が、ルートから調べたコンポーネントと重複することなく、$A'.path$ の次のコンポーネントがアスタリスクであるところまで到達したら、つまり、C'_n から C'_{k+1} と B'_m から $B'_{\{m-(n-k-1)\}}$ の計 $n-k$ 個のコンポーネントが終端コンポーネントからルートへ向けて名前が同じで、さらに、$k \leq m-(n-k-1)$ であるとき、

$$D'.path : B'_0 B'_1 \cdots B'_k B'_{(k+1)} \cdots B'_{\{m-(n-k)\}} B'_{\{m-(n-k-1)\}} \cdots B'_m$$

この要素インスタンス D' は必要とされる条件を満たしたと判断する。

アルゴリズム 1.2' 終

$B'_{(k+1)} \cdots B'_{\{m-(n-k)\}}$ の $m-n$ 個がアスタリスクが補完された部分パスになる。

この 2. で、続きを末尾から判断するのは、カレントパス・割り当て規則のどちらにしても、パス中に同じ名前のコンポーネントが複数含まれる場合、そのうちのどのコンポーネントが該当するか、その判断が複雑になるのを避けるためである。

たとえば、前述の割り当て規則例 "/論文 /本文 * /図 /図題" に対して要素インスタンス "/論文/本文/章/節/段落/図/図題" が必要と判断されるのは、以下のような処理に基づく。

- まずルートから“論文”、“本文”と評価して、割り当て規則にアスタリスクが現れたので、
- 続いて終端から“図題”、“図”と評価する。ここで再び割り当て規則にアスタリスクが現れた。
- 要素インスタンスの方で重複してパスを評価していないので、これは必要であると結論できる。残りのパス“/章/節/段落”が補完されたことになる。

7.5.4 新要素のパスの省略

新要素の割り当て規則を扱うのは次の局面である。

- 新要素として設定される

従ってこの時、パスを省略した影響が表れる。

この場合、今設定したい新要素インスタンスの割り当て規則でのパスの指定が、ここでは優位性を考慮するので、次のように表される。以降ではコンポーネントの同異を強調するために、コンポーネントを表す記号をB、C、F、Kなどとして区別する。

$$A.path : C_0\{!C_{0l}\}^* \cdots C_h\{!C_{hl}\}^* * \{!C_{*l}\}^* C_{(h+1)}\{!C_{(h+1)l}\}^* \cdots C_n\{!C_{nl}\}^*$$

今から処理する旧要素を次のように表す。

$$D'_j.path : K'_0\{!K'_{0l}\}^* \cdots K'_g\{!K'_{gl}\}^* B'_{(g+1)}\{!B'_{(g+1)l}\}^* \cdots B'n\{!B'n_l\}^*$$

これから定められた手がかり要素インスタンスが、旧新それぞれ次のようであるとする。ここでコンポーネントを表すために、Fも用いる。

$$\begin{aligned} D'_k.path &: K'_0\{!K'_{0l}\}^* \cdots K'_g\{!K'_{gl}\}^* F'_{(g+1)}\{!F'_{(g+1)l}\}^* \cdots F'_m\{!F'_{ml}\}^* \\ D_k.path &: K_0\{!K_{0l}\}^* \cdots K_g\{!K_{gl}\}^* F_{(g+1)}\{!F_{(g+1)l}\}^* \cdots F_m\{!F_{ml}\}^* \end{aligned}$$

とする。つまり、手がかり旧要素と今着目している旧要素は、$K'_0 \cdots K'_g$の優位なコンポーネントが名前もIDも一致していることを表す。

ここから新要素を設定するには、まず名前は割り当て規則のパス記述から、省略された部分以外決定する。

$$D'_j : C_0\{!C_{0l}\}^* \cdots C_h\{!C_{hl}\}^* * \{!C_{*l}\}^* C_{(h+1)}\{!C_{(h+1)l}\}^* \cdots C_n\{!C_{nl}\}^*$$

残りのコンポーネントの名前とIDの決定には、名前とIDが一致した優位なコンポーネントの範囲とアスタリスクの位置の関係により、二通りに方法が分類できる。

- 一致した範囲がアスタリスクより下位までの場合。つまり$h < g$（図7.9）
  D_jがD_kからIDを受ける範囲、つまりD'_kとD'_jが同じ優位なコンポーネントを持っている範囲に相当する“$C_0\{!C_{0l}\}^* \cdots C_g$に$D'_j$のアスタリスクは含まれるので、$D_k$の“$K_0 \cdots K_g$からコンポーネント名も受け継げよい。従って、アスタリスクのない場合と同様な処理でよい。

- 一致した範囲がアスタリスクより上位の場合。つまり$h \geq g$
  この場合が起こり得る状況について考える。可能性としては、この省略はどのような場所にでも記述され得るように考えられる。しかし文書を構成するという意味を考えると、手がかりとなる新要素インスタンスD_kとの、共有部分“C_1C_2”から離れたところに出現するのは、図7.10のような状態を意味している。

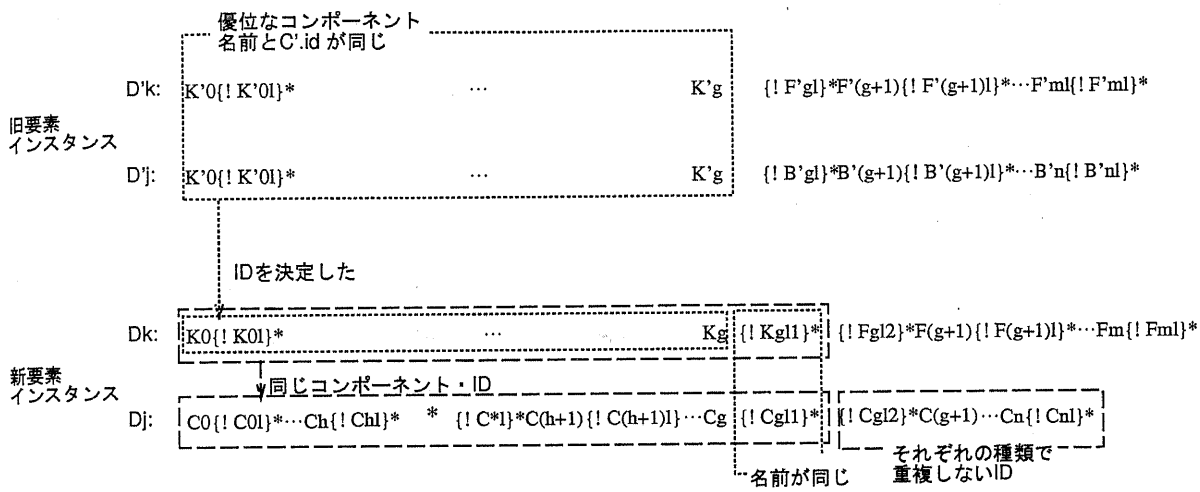


図 7.9: 一致した範囲がアスタリスクより下位までの場合（$h < g$）

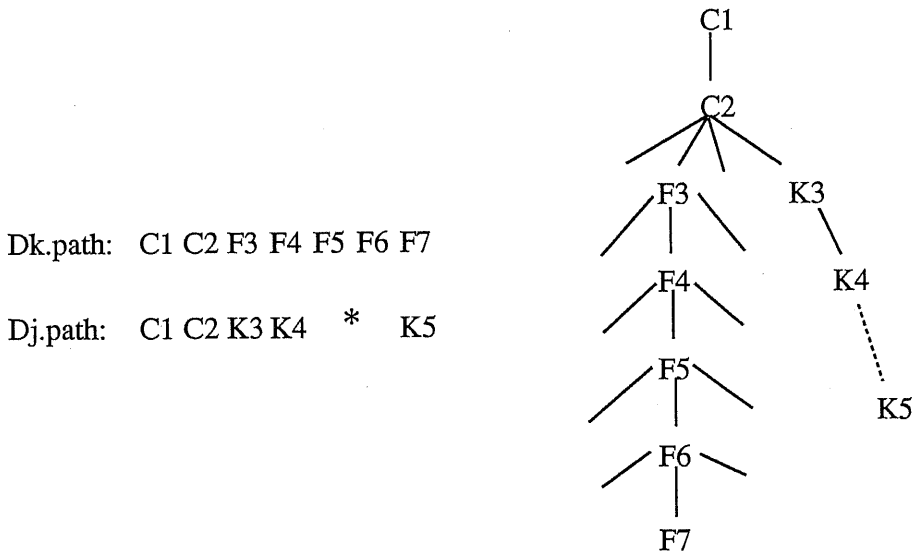


図 7.10: パスの離れたところに出現するアスタリスク

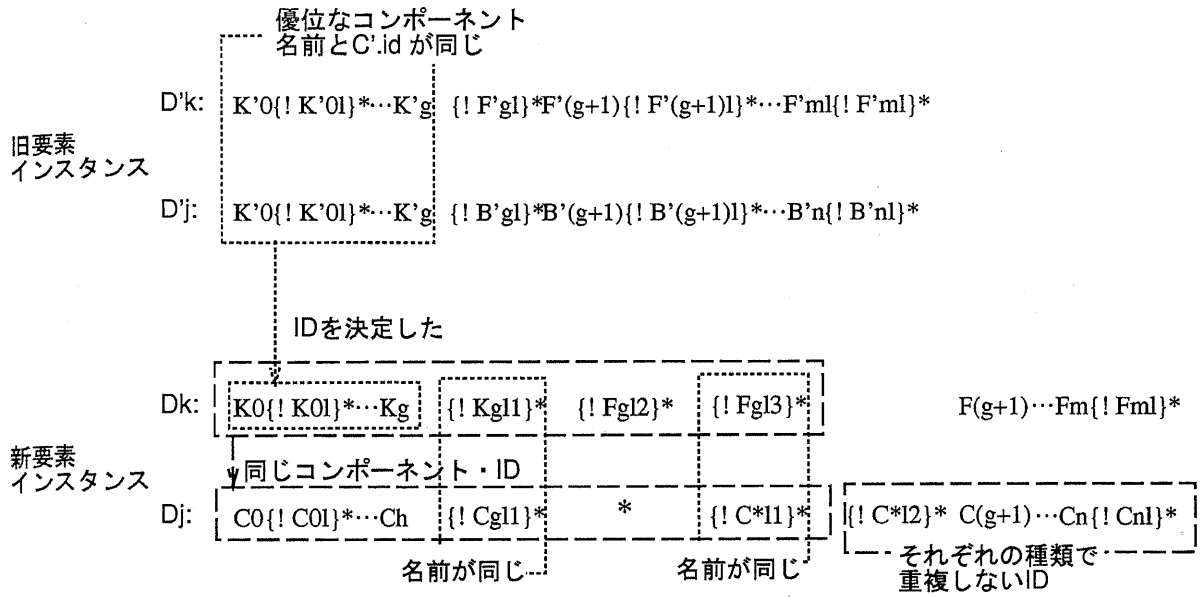


図 7.11: 一致した範囲がアスタリスクより上位の場合 ($h \geq g$)

論理構造上一つの要素インスタンスだけ他の要素インスタンスとノードを共有しない部分が続き、突出して深い位置にある要素が出現するというのは不自然である。つまり、実際にはどの要素も省略されるすぐ上までは一致する手がかりがあると想定できる。このように考えると、$h = g$しか考慮しなくてよいと判断できる。

そこで新要素のパスとしては、優位でないコンポーネントの範囲で名前と ID を引き継ぐ（図 7.11）。なぜならば、省略されたコンポーネントの中には優位なコンポーネントはないと規定したからである。

以上より相対的なパスを用いた変換仕様の記述では、手がかり旧要素インスタンスを見つける段階までは通常の処理と変わらず、そのあと新要素インスタンスのパスを決定する際に、アスタリスクが手がかり新要素インスタンスのどこに出現しているかを判別し、上記の方法によりパスを補完すればよいことがわかる。

第 8 章

評価

本研究では、文書の構造化やその多様性の必要性を、文書が表現力を増し個性的になること、検索が高度化すること、閲覧がより機能的になることなどの利便から認める一方、これを扱うデータベースを実現するためには困難が伴う点に着目し、論理構造を変換することにより、上記の利便を、統一的なユーザインタフェースなどの状況下で提供するシステムの構築に寄与する方向を提案し、その論理構造の変換手法について検討した。

この章では、実際の文書に本手法を適用した実験の成果をまとめるとともに、本手法の利点などについて考察し、さらに将来的な課題として、データベースとして全体が機能するまでの道筋について述べる。

本章の構成は以下の通り。

- 8.1: 実際の文書での実験
  本手法を実装したプロトタイプについて述べるとともに、サンプルとして作成したデータではなく、実際にこの目的のためではなく作成されたSGML文書を用いた処理について述べ、本手法の有効性を示す。

- 8.2: 評価
  第4章などで取り上げた経緯や課題を踏まえて、本手法の有利な点、また限界についてまとめる。

- 8.3: データベースへの課題
  本手法が機能しても、データベースとして全体のサービスを行うには検討すべきことは多くある。それらについてまとめる。

8.1 実際の文書での実験

8.1.1 プロトタイプについて

本手法はプロトタイプとして実装され、実際に文書処理を行っている。内部のデータ構造は、各構成要素がそれぞれの属性と前方及び後方へのポインタを持った構造体であり、それらのリストとして全体が表現されている。要素のヘッダが前方及び後方へのポインタを持っているのは、要素を処理する過程で、処理の終わったものや他の要素インスタンスに統合されたものをリストから外しても、リストの継続性を修正できるできるためである。

また現在のものは、処理速度やメモリ量などには特段の配慮はしていない。

動作は、新旧のDTD名（ファイル名）、旧文書名（ファイル名）を入力することにより、このようなデータ構造にデータを展開し、第7章に述べた処理のアルゴリズムに従って各段階ごとに変形していく。その様子や結果は、同じく第7章に図などで示した。

8.1.2 実験データ

第7章までは、本研究の課題をすべて象徴的な形で含むようなデータを作成し、それを処理することにより、各課題がどのように解決されるか明確に示せることを主眼としてきた。

本研究では続いて、この手法及びプロトタイプが、説明用ではなく実際の文書に対して機能することを、実験を行って有効性を示した。ここからはこの実験の前提や過程、そして結果などについてのべる。

実験に当たって、実験データに対して、

- DTD が入手できること
  その特定の文書に限らず、全体の論理構造が把握できる。また出力データのDTDは、どのような形式で出力したいのか決定するために、変換仕様作成の過程で必須である。

- インスタンスが入手でき、できればタグの省略などがないこと
  構文解析のような動作をするためにはタグの省略やアンバランスは大きな障害である。多くのSGMLパーサでもタグの復元を（自らするにせよ、前処理としてするにせよ）明確に意識している。

- パーザや表示などの処理系が利用できること
  論理構造の確認や解析、あるいは結果の提示のために、出力された結果を受け入れられることを示す確実で、おそらく数少ない方法である。

という条件を課した。最も望ましい、三点ともに満たしているデータについては見つけることができなかったため、次善の策として、

- 入力データ: 学術情報センター紀要第7号論文データ及びDTD
  「紀要」作成のために得た電子化文書を、センター内での実験のためSGML化したもの。独自の処理系はないが、DTDと、ほぼタグの省略のない文書インスタンスが入手できた。タグが省略されている箇所については、手作業で補完した。

- 出力形式: HTML
  論理構造がそれほど強固ではないという弱点はあるものの、DTDが公開されており、表示系であるブラウザが充実している。

という準備を整えた。学術情報センター紀要及びHTMLのDTDは、実体参照などを展開して、変換仕様作成の際に両論理構造を把握する資料とした。実験用に手を加える前のオリジナルのものを付録に添える。

また、論理構造に沿っていることの確認や、出力結果の活用例であるビューワとして

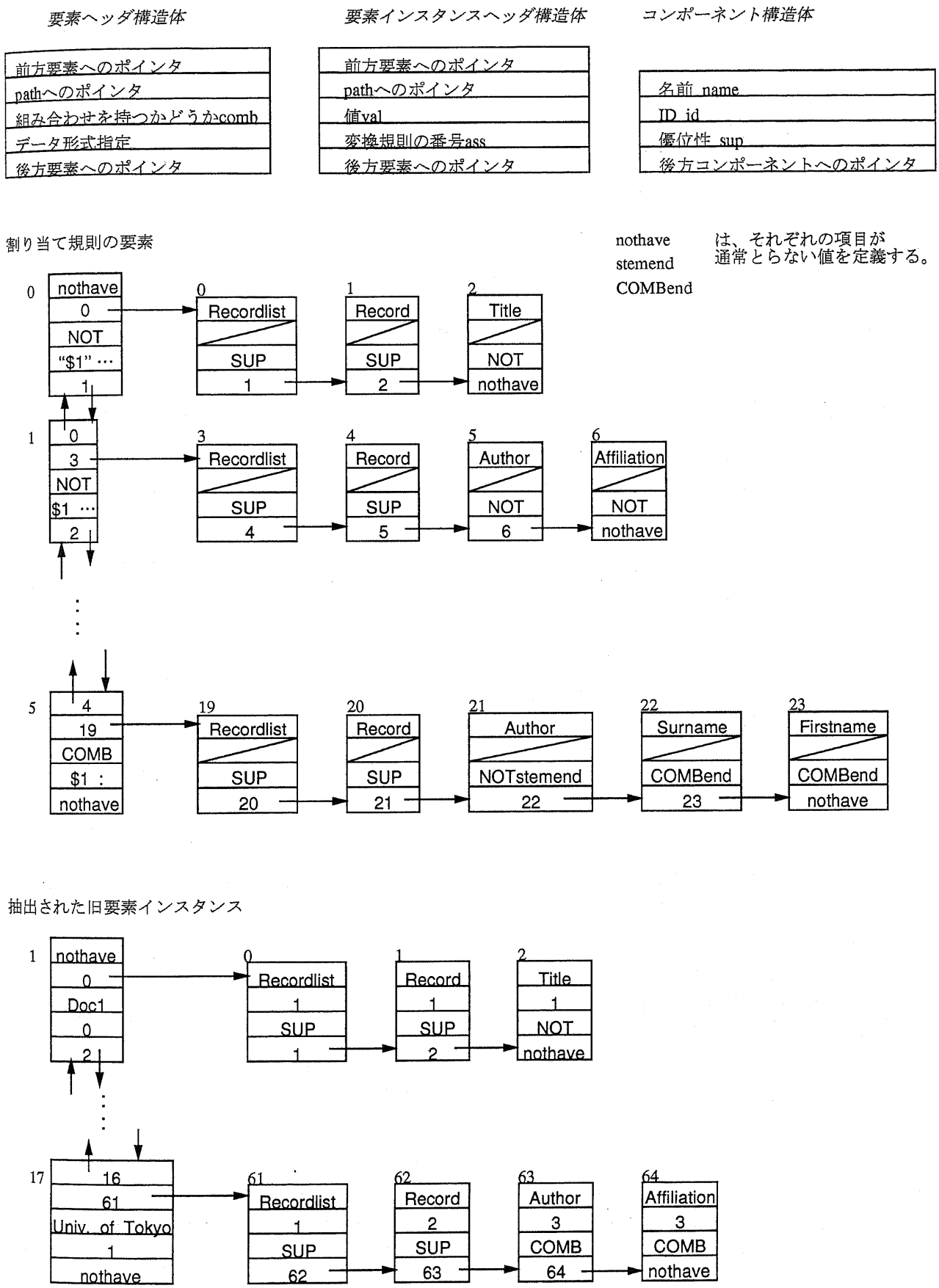


図 8.1: プロトタイプの内部データ構造

- 汎用 SGML エディタ "SGML-PLUS Ver 2.0" [40] :
  DTD を予め内部形式に変換しておき、それとインスタンスを併せて読み込むことにより、論理構造を織り込んだ文書の著述環境を提供する。また、DTD を作成したり、DTD に則ってインスタンスを作成したりすることもできる。

  読み込ませた DTD にインスタンスが合致していること、そしてその論理構造の要素が見られる。

- HTML ブラウザ: Netscape Navigator Version 3.0 [41]:
  WWW ブラウザであり、またローカルファイルであっても HTML 文書などを閲覧することができる。

  インスタンスが HTML の DTD に合致していること、そしてその整形結果が見られる。

を用意した。

8.1.3 変換仕様の作成

この実験では、処理が必要以上に重くなるのを避けるため、入手した二編の論文が変換できることを目指した。従って、変換仕様の作成手順は以下のようにした。

1. 二編の論文の論理構造を調べて、双方の論理構造の和を取る。つまり、それらで現れる論理要素を列挙する。そこで得られたデータは、図 8.2、8.3のようになった。ここで、" #PCDATA" や "EMPTY" は実際のデータの形態を表す SGML の予約語である。

2. 得られた要素を、組み合わせを考慮して整理する。

3. 整理された要素に対して、適当な HTML の要素を、絶対パスを用いて記述する。

4. コンポーネントの対応を決定する。コンポーネントの優位性は、

   - パスの長さが異なる時は短い方に数が合うことを原則として、
   - 各コンポーネントの示す意味がルートから順になるべく合致するように
   - また、互いの周囲の要素と矛盾が生じないように

   という基準で決定した。ただし、長さが等しい場合や、長さが異なる場合でも短い方に合わせるのではなく、意味的に、あるいは前後の要素との整合により、それらより短い要素にしたところもある。

この方法は、変換先の論理構造、つまり HTML が簡単な論理構造にとどまるために有効な方法であり、一般的には、DTD で表現し得る論理構造の形態を、ツリー状に表現して、それを見比べながら作成する。

作成した変換仕様は図 8.4、8.5に示す。著者名や文中のリファレンスを複数要素の組み合わせで、キーワードを繰り返しで処理している。今回の実験では、変換先の論理構造が HTML であり、そのブラウザがウェブブラウザということで、最終的にフォーマット画面を得る時にも、整形の方法に選択の余地がない。従って、手法の本来の趣旨からははずれるが、「改行」("<BR>") や横罫線 ("<HR>") など誌面の整形に関する指定も、応用的に必要に応じて取り込んでいる。

8.1.4 変換処理

もとの文書データは、付録 B に示す SGML インスタンスである。このインスタンスが、付録 A に示した DTD に則っていることを示すために、前述の SGML エディタ上に表示したものが図 8.6〜8.10 である。このエディタ画面上では、論理構造の階層の深さがインデントで、またその要素名とデータが、要素名のマークがデータの前後におかれるタグ形式で階層構造に示されている。

```
article front atl nihongo.title #PCDATA
article front atl english.title #PCDATA
article front runt #PCDATA
article front author g.name kanji #PCDATA
article front author g.name furigana #PCDATA
article front author g.name english #PCDATA
article front author f.name kanji #PCDATA
article front author f.name furigana #PCDATA
article front author f.name english #PCDATA
article front author position kanji #PCDATA
article front author position furigana #PCDATA
article front author position english #PCDATA
article front abstract abstract.nihongo p #PCDATA
article front abstract abstract.english p #PCDATA
article front keywords keyword kanji #PCDATA
article front keywords keyword furigana #PCDATA
article front keywords keyword english #PCDATA
article body sec t #PCDATA
article body sec p #PCDATA
article body sec p ref book ptitl #PCDATA
article body sec p ref book obi #PCDATA
article body sec p ref book pub #PCDATA
article body sec p ref book year #PCDATA
article body sec p ref book auth #PCDATA
article body sec p ref art auth #PCDATA
article body sec p ref art t #PCDATA
article body sec p ref art ptitl #PCDATA
article body sec p ref art vol #PCDATA
article body sec p ref art page #PCDATA
article body sec p ref art year #PCDATA
article body sec p ref p #PCDATA
article body sec p xref EMPTY
article body sec p e2 #PCDATA
article body sec ln li p #PCDATA
article body sec ln li p ref book ptitl #PCDATA
article body sec ln li p ref book obi #PCDATA
article body sec ln li p ref p #PCDATA
article body sec lm li p #PCDATA
article body sec ld term #PCDATA
article body sec ld def p #PCDATA
```

図 8.2: データとした論文の論理構造の和 (1)

```
article body sec tb t #PCDATA
article body sec tb fgart #PCDATA
article body sec fg t #PCDATA
article body sec fg fgart #PCDATA
article body sec ss1 t #PCDATA
article body sec ss1 p #PCDATA
article body sec ss1 p xref EMPTY
article body sec ss1 p ref book auth #PCDATA
article body sec ss1 p ref book ptitl #PCDATA
article body sec ss1 p ref book pub #PCDATA
article body sec ss1 p ref book year #PCDATA
article body sec ss1 p ref book page #PCDATA
article body sec ss1 p citref EMPTY
article body sec ss1 p e2 #PCDATA
article body sec ss1 fg t #PCDATA
article body sec ss1 fg fgart #PCDATA
article body sec ss1 ln li p #PCDATA
article body sec ss1 lu li p #PCDATA
article body sec ss1 tb t #PCDATA
article body sec ss1 tb head row c p #PCDATA
article body sec ss1 df      #PCDATA
article body sec ss1 df subt #PCDATA
article body sec ss1 tb row c p #PCDATA
article body sec ss1 ss2 t #PCDATA
article body sec ss1 ss2 ld term #PCDATA
article body sec ss1 ss2 ld def p #PCDATA
article body sec ss1 ss2 p #PCDATA
article end references EMPTY
article end acknowledge p #PCDATA
```

図 8.3: データとした論文の論理構造の和 (2)

```
#
# nacsis7 to HTML 論理構造変換仕様
#
#日本語/英語タイトル
/HTML /BODY /H1 ; /article /front !/atl /nihongo.title ; 「$1」 ; $$ ($1)
/HTML /BODY /H1 ; /article /front !/atl /english.title ; "$1" ; $$ ($1)
#タイトルバー
/HTML /HEAD /TITLE ; /article /front /runt ; $1; $$ ($1)
#著者名（日本語のみ、組み合わせ）
/HTML /BODY /P ; /article /front /author : /f.name /kanji : /g.name /kanji :
            /position /kanji ; $1 : $2 : ($3) ; $$ ($1) : $$ ($2) : $$ / $3 ;
             $1 $2 $3 ; $$ / $1
#日本語要旨
/HTML /BODY /P ; /article /front /abstract !/abstract.nihongo !/p ;
             $1 <HR>; $$ <BR>$1
#キーワード（日本語/英語）
/HTML /BODY /P ; /article /front /keywords !/keyword !/kanji ; $1 ; $$, $1
/HTML /BODY /P ; /article /front /keywords !/keyword !/english ; $1 ; $$, $1
#章題
/HTML /BODY /H1; /article /body !/sec /t ; $1 ; $$ ($1)
#本体
/HTML /BODY /P ; /article /body !/sec /p ; $1 ; $$ $1
#参照
/HTML /BODY /P /I ; /article /body !/sec /p !/ref /book : /ptitl : /obi : /pub :
            /year : /auth ; $1 : "$2" : $3 : $4 : $5 ; $$ $1 : $$ $1 : $$ $1 :
            $$ $1 : $$ $1 ; $1 $2 ($5, $3, $4) ; $$ / $1
/HTML /BODY /P /I ; /article /body !/sec /p !/ref /art : /auth : /t : /ptitl :
            /vol : /page : /year ; $1 : "$2" : $3 : $4 : $5 : $6 ; $$ $1 : $$ $1 :
            $$ $1 : $$ $1 : $$ $1 : $$ $1 ; $2 ($1, $3, $4, $5. $6) ; $$ / $1
/HTML /BODY /P /I; /article /body !/sec /p /ref !/p ; $1 ; $$ <BR> $1
/HTML /BODY /P /TT ; /article /body !/sec /p /e2 ; $1 ; $$ $1
/HTML /BODY /P /OL /LI ; /article /body !/sec /ln /li /p ; $1 ; $$ $1
/HTML /BODY /P /OL /LI /P /I ; /article /body !/sec /ln /li /p /ref /book : /ptitl :
            /obi ; "$1" : "$2" ; $$ ($1) : $$ ($2) ; $1 ($2) ; $$ / $1
/HTML /BODY /P /OL /LI /I ; /article /body !/sec /ln /li !/p /ref /p ; $1 ; $$ / $1
/HTML /BODY /P /UL ; /article /body !/sec /lm /li !/p ; $1 ; $$ / $1
/HTML /BODY /P /DL /DT ; /article /body /sec /ld /term ; $1 ; $$ / $1
/HTML /BODY /P /DL /DD ; /article /body /sec /ld /def !/p ; $1 ; $$ <BR> $1
/HTML /BODY /P /TT; /article /body /sec !/fg /t ; (図$1) ; $$ / $1
/HTML /BODY /H2 ; /article /body !/sec /ss1 !/t ; $1 ; $$ ($1)
/HTML /BODY /P ; /article /body !/sec !/ss1 /p ; $1 ; $$ $1
/HTML /BODY /P /I ; /article /body !/sec !/ss1 /p !/ref /book : /ptitl : /pub :
            /year : /page : /auth;  $1 : $2 : $3 : $4 : $5 ; $$ $1 : $$ $1 :
            $$ $1 : $$ $1 : $$ $1 ; $1, $2 ($5, $3, $4) ; $$ / $1
```

図 8.4: 実験用の変換仕様 (1)

```
/HTML /BODY /P /EM; /article /body /sec !/ss1 /p !/e2 ; $1 ; $$ <BR> $1
#図（タイトルのみ残す）
/HTML /BODY /P ; /article /body !/sec !/ss1 !/fg /t ; （図$1） ; $$ / $1
/HTML /BODY /P /OL /LI /P ; /article /body /sec !/ss1 /ln /li /p ; $1 ; $$ $1
/HTML /BODY /P /UL /LI /P ; /article /body /sec !/ss1 /lu /li /p ; $1 ; $$ $1
/HTML /BODY /P /TT ; /article /body /sec !/ss1 /ss2 !/t ; $1 ; $$ <BR> $1
/HTML /BODY /P /DL /DT ; /article /body /sec !/ss1 !/ss2 /ld /term ; $1 ; $$ <BR> $1
/HTML /BODY /P /DL /DD ; /article /body /sec !/ss1 !/ss2 /ld /def !/p ; $1 ;
            $$ <BR> $1
/HTML /BODY /P ; /article /body !/sec !/ss1 /ss2 /p ; $1 ; $$ <BR> $1
#謝辞
/HTML /BODY /P ; /article /end !/acknowledge /p ; $1 ; $$ / $1
```

図 8.5: 実験用の変換仕様 (2)

ここで入力したデータは、実際に学術情報センター紀要 7 号に掲載された論文であり、それを実験用に SGML 化したものである。データの大きさは、（タグなど込みで）約 30Kbytes である。図など非テキストデータはこの中に含んでおらず、データへのポインタのみを要素の属性として持っているが、本手法では現在のところ属性に関する操作は行っていないので、ポインタを形だけ示している。

処理の所要時間は、Sun ワークステーション Ultra2 で数秒間程度である。

結果は、付録 D に示すような HTML ファイルが出力される。それを Netscape Navigater に表示させると、図 8.11のような画面が得られ、確かに所望の形式が出力されたことがわかる。たとえば、

- 著者名と所属が一つにまとめて記載されている。
- タイトルが和文は “「」” で、英文は “‘‘’’” で囲まれている。
- 旧文書で各ページの上につけるためのタイトルがウィンドウのタイトルになっている。
- キーワード（横罫線下）は並べて、“,” を挟んで列挙されている。
- 文中の参照文献（第一章、三段落め、第一行後半から）は、字体を変えて（イタリック体)、“タイトル、‘‘URL’’” などの形になって埋め込まれている。

などの変換・整形の様子が見られる。また整形からは見えないが、深さの違いや、繰り返しなども処理されている。

【日本語SGMLエディタ：nacsis.PDT】

ファイル(F) 編集(E) 検索(S) アウトライン(O) マークアップ(M) 略語(R) オプション(P) ビュー(V)

```
article°
  front
    atl
      [
        nihongo.title  インターネットに適応した全文データベース検索システムの構成
                       nihongo.title
        english.title  Configuration of a Full Text Database Retrieval System for I
                       nternet Environment
                       ▲english.title
      ]
    runt  インターネットに適応した全文データベース検索システムの構成
          runt
    author
      g.name
        [
          [
            kanji     敬三
                      kanji
            furigana  ケイゾウ
                      furigana
          ]
          english  Keizo
                   english
        ]
      f.name
        [
          [
            kanji     大山
                      kanji
            furigana  オオヤマ
                      furigana
          ]
          english  OYAMA
                   english
        ]
      position
        [
          [
```

<english.title>

図 8.6: 入力データの論理構造表示 (1)

【日本語SGMLエディタ: nacsis.PDT】

ファイル(F) 編集(E) 検索(S) アウトライン(O) マークアップ(M) 略語(R) オプション(P) ビュー(V)

kanji 学術情報センター
/kanji
furigana ガクジュツ ジョウホウ センター
/furigana
]
english National Center for Science Information Systems
/english
]
abstract
[
abstract.nihongo
p 本論文では現在のインターネット上での情報サービス環境を最大限に利用しつつ、高度で使用しやすい全文データベース検索サービスを提供できるシステムの構成方法を述べている。WWWのクライアントであるMosaicとサーバであるHTTPDを用い、既存の高度な検索エンジンとの間を独自のゲートウェイにより結合する。階層構造を持つ文書データを対象に柔軟で効率的な検索を可能とするインタフェースを実現している。 /p
abstract.english
p This paper describes a configuration method of a system which can provide full text database retrieval services. The system is easy to use and has advanced features, while utilizing the information services environment currently available on the Internet at most. It adopts a client software Mosaic and a server software HTTPD for the WWW service as a basis, and combines an existing advanced text retrieval engine with them through an original gateway. It realizes a user interface through which users can access to document data with hierarchical structures flexibly and efficiently. /p
]
keywords
keyword
[
[
kanji 全文データベース
/kanji
furigana ゼンブン データベース
/furigana

図 8.7: 入力データの論理構造表示 (2)

【日本語SGMLエディタ: nacsis.PDT】

ファイル(F) 編集(E) 検索(S) アウトライン(O) マークアップ(M) 略語(R) オプション(P) ビュー(V)

```
    ]
    english  full text database
             /english
  ]
keyword
  [
    [
      kanji     情報検索
                /kanji
      furigana  ジョウホウ ケンサク
                /furigana
    ]
    english  information retrieval
             /english
  ]
keyword
  [
    [
      kanji     文書構造
                /kanji
      furigana  ブンショ コウゾウ
                /furigana
    ]
    english  document structure
             /english
  ]
keyword
  [
    [
      kanji     インターネット
                /kanji
      furigana  インターネット
                /furigana
    ]
    english  Internet
             /english
  ]
keyword
```

<keyword><>

図 8.8: 入力データの論理構造表示 (3)

【日本語SGMLエディタ：nacsis.PDT】

ファイル(F) 編集(E) 検索(S) アウトライン(O) マークアップ(M) 略語(R) オプション(P) ビュー(V)

kanji ゲートウェイ
kanji
furigana ゲートウェイ
furigana
english gateway
english
keyword
kanji グラフィカルユーザインタフェース
kanji
furigana グラフィカル ユーザ インターフェース
furigana
english graphical user interface
english
コメント
body
sec
t 1. はじめに
t
p インターネットの整備とワークステーションの普及に伴い、インターネットを通じて様々な情報サービスが提供されており、その数は急激に増えつつある。これらの多くは情報収集・提供のための組織的な支援体制を伴わないため、内容やサービスの質に問題を抱えている例も多く、情報を探索し利用する立場からは必ずしも望ましくない状況も発生している。このため、インターネットを通じた情報サービスに批判的な意見も散見される。 p
p しかし、一方で、従来から独自のネットワークや方式で提供してきた情報サービスを、インターネット経由でも利用可能にし、さらにはインターネットをサービスの中心に据える例も少なくない。これらの多くは従来からの膨大な蓄積を、より効率的に、より利用しやすく、できるだけ多くのユーザに提供しようとするものである。こ

keyword><><

図 8.9: 入力データの論理構造表示 (4)

【日本語SGMLエディタ：nacsis.PDT】

ファイル(F) 編集(E) 検索(S) アウトライン(O) マークアップ(M) 略語(R) オプション(P) ビュー(V)

のような努力は、従来からのユーザにとってのみでなく、潜在的なユーザにとっても歓迎すべきものと言えるであろう。 p

p 筆者が所属する学術情報センターでも、従来からいわゆる情報検索サービスを提供している

ref°

book

ptitl 「情報検索サービス NACSIS-IR」 ptitl

obi http://www.nacsis.ac.jp/ir/ir-j.html obi

が、これまであまりインターネットの機能を有効に利用できないでいた。その原因の一つとしては、サービス提供のためのホストシステムがいわゆるメインフレームコンピュータであり、インターネットで標準的に利用されている環境が適用できなかったことがあげられる。また、センターのユーザの多くが自由にインターネットを使える状態になかったためにセンターが積極的にインターネット対応を推進する動機を欠いたことも一つの理由としてあげることができよう。 p

p しかし、ここ数年間で大学や研究機関における状況は大きく変わり、希望するものは誰でもインターネットを利用できるようになった。このような状況の変化に対し、情報サービスの提供機関は学術情報センターに限らず、ユーザの利便性を第一に考え、できるだけ迅速に適応していくことがきわめて重要であると筆者は考えている。本論文では、このような目的を満たすために筆者が中心になって研究開発を進めているインターネット上の情報サービスシステムの構成方法について述べる。 p

sec°

t 2. 情報サービスのツールの現状と利用方針 t

p 筆者はインターネットを介したサービスを、既存のサービスの代替ではなく、今後のサービスの中心的役割を担うものと考えている。そのため、既存のサービスが備えている機能の見直しを含め、新しい環境に適応した高度な機能を実現することが重要である。一方で、インターネットはさまざまなボランティアが提供してきた多くのパブリックドメインのソフトウェアと、それを元にして開発された商用の製品群によって大きく発展してきた。このような発展形態は今後も継続するものと考えられる。 p

p 本研究においては、インターネット上で一般的に利用されているツ

図 8.10: 入力データの論理構造表示 (5)

Netscape - [インターネットに適応した全文データベース検索システムの構成]

File Edit View Go Bookmarks Options Directory Window Help

Back Forward Home Edit Reload Images Open Print Find Stop

Location: http://www.adl.rd.nacsis.ac.jp:80/~sakai/oy.html

What's New? What's Cool? Destinations Net Search People Software

「インターネットに適応した全文データベース検索システムの構成」

"Configuration of a Full Text Database Retrieval System for Internet Environment"

大山 敬三（学術情報センター）

本論文では現在のインターネット上での情報サービス環境を最大限に利用しつつ、高度で使用しやすい全文データベース検索サービスを提供できるシステムの構成方法を述べている。WWWのクライアントであるMosaicとサーバであるHTTPDを用い、既存の高度な検索エンジンとの間を独自のゲートウェイにより結合する。階層構造を持つ文書データを対象に柔軟で効率的な検索を可能とするインタフェースを実現している。

全文データベース , 情報検索 , 文書構造 , インターネット , ゲートウェイ , グラフィカルユーザインタフェース

full text database , information retrieval , document structure , Internet , gateway , graphical user interface

1. はじめに

インターネットの整備とワークステーションの普及に伴い、インターネットを通じて様々な情報サービスが提供されており、その数は急激に増えつつある。これらの多くは情報収集・提供のための組織的な支援体制を伴わないため、内容やサービスの質に問題を抱えている例も多く、情報を探索し利用する立場からは必ずしも望ましくない状況も発生している。このため、インターネットを通じた情報サービスに批判的な意見も散見される。

しかし、一方で、従来から独自のネットワークや方式で提供してきた情報サービスを、インターネット経由でも利用可能にし、さらにはインターネットをサービスの中心に据える例も少なくない。これらの多くは従来からの膨大な蓄積を、より効率的に、より利用しやすく、できるだけ多くのユーザに提供しようとするものである。このような努力は、従来からのユーザにとってのみでなく、潜在的なユーザにとっても歓迎すべきものと言えるであろう。

筆者が所属する学術情報センターでも、従来からいわゆる情報検索サービスを提供している（*情報検索サービス NACSIS-IR*） *"http://www.nacsis.ac.jp/ir/ir-j.html"* (, ,) が、これまであまりインターネットの機能を有効に利用できないでいた。その原因の一つとしては、サービス提供のためのホストシステムがいわゆるメインフレームコンピュータであり、インターネットで標準的に利用されている環境が適用できなかったことがあげられる。また、センターのユーザの多くが自由にインターネットを使える状態になかったためにセンターが積極的にインターネット対応を推進する動機を欠いたことも一つの理由としてあげることができよう。

2. 情報サービスのツールの現状と利用方針

筆者はインターネットを介したサービスを、既存のサービスの代替ではなく、今後のサービスの中心的役割を担うものと考えている。そのため、既存のサービスが備えている機能の見直しを含め、新しい環境に適応した高度な機能を実現することが重要である。一方で、インターネットはさまざまなボランティアが提供してきた多くのパブリックドメインのソフトウェアと、それを元にして開発された商用の製品群によって大きく発展してきた。このような発展形態は今後も継続するものと考えられる。

本研究においては、インターネット上で一般的に利用されているツールを有効に組み合わせながら必要な機能を実現していくことを基本的な考え方とする。特にユーザ環境としては、広く利用可能な一般的なツールを用いてサービスを利用できることを最重要の課題と考えている。そこで、まず本章では、インターネット上の情報サービスのためのツールの現状を整理し、それらの利用方針を検討する。

Document: Done

図 8.11: 出力データのブラウザ表示

8.2 評価

本研究では、SGML に従って記述された、多様な論理構造を持つ文書を対象としたデータベースを前提として、文書の表現力と個性のために多様性は容認する一方で、ユーザには統一的なインタフェースを提供して、さらにデータベースも表現力などを持つために任意の論理構造を持てることを目的として、このようなシステムを実現する核心である、任意の論理構造間の論理構造変換手法を提案した。

この手法によれば、そのようなデータベースが実現できる他、一般に SGML 文書の論理構造の変換や、情報の抽出ルーチンとして使うこともできる。

第 2.5節でも述べたことなどをまとめると、このような手法の評価基準は、以下のような点であると言える。

- 対象とする変換の範囲が広いか
  扱える変換が、新旧要素間の一対一対応に限られないか、文脈情報を利用した動作の指定ができるか、など。

- 仕様の記述が容易であるか
  新たなプログラム言語を習得するような敷居の高さはないか、記述すべき内容や形式が把握しやすいか、など。

これらの点について、提案した手法は、次のような利点を持つ。

- 組み合わせや繰り返しなどを処理できる
  任意の数の要素を組み合わせて、新しい文書で一つの要素として出力できる。

- 論理構造の文脈情報を活用している
  仕様書の記述に、要素をパス表現で指定することから、必要に応じて、各要素の上位の要素が何かという文脈に応じて処理を切替えられる。

- データの整形が行える
  文字列処理の範囲で整形が行え、予め用意した関数を呼び出して任意の処理をすることができる。

- 仕様の記述が容易である
  組み合わせや繰り返しにかかわらず各変換仕様は独立であるため、組み合わせる二つの旧要素群が互い違いに出現するような、規則間の相互の影響を避けることができ、文書全体の論理構造の具合を考えることなく仕様を記述できる。

- 仕様の記述が柔軟である
  割り当て規則に相対的なパスを用いることにより、SGML の浮動要素をそのままの形で、あるいは同じ名前を持つ要素をすべてまとめてしまうような、柔軟な記述も可能である。

一方、この手法がより高機能になるためには、いくつか取り入れるべき点もある。

- 属性の授受
  図のような非テキストなどは、文書の中に直接表現されているのではなく、外部ファイルとして作成され、文書中にポインタ（アンカ）のみが要素として存在して、ファイル名をその要素の属性として、パーサやフォーマタの段階でその外部ファイルを処理するアプリケーションを起動するなどの処置を取ることが多い。また、HTML ではレイアウトに関する情報を、「行中央に配置」などの属性を要素に持たせることで、整形を制御している。

対応する新旧要素間でパスの長さが一致するとも限らないし、属性の使い道や形態が二つの論理構造で一致するとも限らないので、その意味ではそのまま属性値を受け渡すのか、ここにも変形などを指定する機会を与えて、何らかの変換を施してから値を渡すのか、検討が必要なところであろう。

そのまま渡すのであれば、現在のシステムからの拡張は、優位性により対応付けられた新旧コンポーネント間で受け渡すことにすれば、当面の要求は満たせるだろう。

- 要素の出現順

今回実験に使ったDTDはどちらも、要素の出現順には概ね寛容な設計をされていた。実験のために他にいくつか調べたDTDでも順序に関する自由度の余地は大きかった。

しかし、それぞれのDTDからパーサを通してフォーマットなどするためにはDTDに指定されているかもしれない要素の出現順も合致するように並べ変えなくてはいけない。この処理では、DTDと対照する論理構造チェッカを組み込まざるを得ないため、第2.5で述べたような問題点がまた浮上してくる。

この点についてはさらに検討が必要だろう。

- 変換仕様作成インタフェース

変換仕様は規則の構成や記述法自体は簡単であるものの、要素名と形式を列挙する記号の並びなので、これを手作業で記述するのは間違いなどを生じやすい。そこで、要素名を打ち込むよりも選ぶようなインタフェースがあればこの問題は解決されるだろう。

そのインタフェースの動作は、たとえば次のようなものである。

1. DTDをパーサで解析して、論理構造を図示する。
2. その図上で対応する新旧要素をマウスなどポインティングデバイスで指示し、要素を決定する。
3. パスを絶対パスか相対パスか指示する。相対の場合は省略する部分を指し示す。
4. 形式については、必要な項目をシステムの側が判断し、その題目とともに入力者に尋ねる。簡単なものやシステムが用意したものは選択するようにする。

また、コンポーネントの優位性や対応づけについてはいくつか手がかりを得たものの、体系化するまでには検討が至らなかった。たとえば、

- 論理構造上で前後にあたる要素から継承するとよいこともある。
- ルートから順にコンポーネントの意味合いを合致させていき、合わないところ（余るもの、あるいは順序が入れ替わるもの）を優位でないと定義するとよい。

などといういくつかの法則は発見した。このようにして全体的に通じる規則がまとめられれば、作成インタフェースの整備や、仕様記述自体のさらなる簡略化が進められるかもしれない。

8.3 データベースへの課題

ここまでに述べてきた論理構造変換の機構を実際のデータベースに組み込むためには、データベースの動作などについて検討が必要である。この点での課題について、二点を取り上げる。

8.3.1 検索と変換のタイミング

動的な変換の負荷とデータ量の兼ね合い、あるいは処理量・処理速度（検索などが発せられてから、結果が返るまでの所要時間）のトレードオフについては検討が必要だろう。

たとえば学術情報センターのデータベースについて調べてみると、電気系のデータベースの収容件数は、数千から数十万件といったところである。これを、現在のプロトタイプシステムで論理構造の変換処理を行えば、一件一秒としても、数十分から数十時間かかることになる。プログラムとしての高速化や高機能計算機の導入を図るとしても、この上にさらに検索や整形などの処理が加わることを考えれば、このままではユーザが検索などを入力した時に、まったくその場で変換処理を全文書に対して行うという実現方法は難しいと予想される。

しかし一方で、第3章で述べたように、オリジナルな形式のままでデータを蓄積する利点は欠かせない。すると、まったく該当しないような文献を粗く選別するための、

- 二次情報
- システム側の論理構造に基づく、論理構造と出現するキーワードの表
- 単語のシグネチャ

など、何らかのインデックスを作成しておき、それを途中に挟んだ形で、検索入力・インデックスによる粗選別・論理構造の変換・厳密な検索などの処理・整形など結果の提示、という流れを組み立てるような工夫が必要であろう。

8.3.2 検索条件・検索インタフェース

文書に論理構造が明示してあると、可能になる機能も多い。たとえば、章節などのタイトルだけを取り出して目次ビューを提供し、全体をユーザに把握させた後、必要な段落だけを詳細に表示する、という閲覧などである。これらをデータベースに組み込むことで、ブラウザとして、より豊かな、つまりユーザがより早く、楽に、深く文書を理解し、活用するのを助けるようなシステムに一歩近付けることになる。

一方で期待されているのは論理構造を用いた検索である。章・節などのタイトルを論理構造のように提示する目次情報が、シソーラスのように概念の上位/下位あるいは広い視点/狭い視点を表すと仮定すると、「データベースについて述べている章の中の関係モデルについて解説してある箇所」といった検索条件が考えられる。

しかし、実際には章のタイトルや章のまとめなどの上位に当たる記述は細かな内容を表す文章に対してあまりにも内容が少なく、またタイトルには「はじめに」など内容を表さないケースも多い。そこで、検索においてどのように論理構造を活用したいのかという点については模索が続いているというのが現状であろう。

さらに、そのような高度な検索が何らかの形で採用されたとして、その条件をどのように入力するかも検討の余地がある。現在のキーワードによる検索に見られる検索条件入力は、検索としての能力には様々な物足りなさも訴えられているが、条件の入力は間違いなく簡便である。高度な検索を実現するためには、条件入力もそれなりに高度になると予想され、それは先に述べた例に倣えば、「（論理構造の用語）中の（キーワード）で、（キーワード）に関して解説している（論理構造の用語）」といった複雑さを持つ。

これを文章の形式になるのか、あるいは新たな記号を用いて、論理的あるいは図形的に表現することになるのか、試みは行われているものの、まだ未知数であると言えるだろう。システムとしてどのくらいのことができるのか、ユーザが何を求めているのか、ユーザがどのくらいなら使いこなせるのか、という多方面からの検討が望まれる。

本研究に当たって関連研究を探した中では、論理構造を活用した検索とは何かについての研究があまり見当たらなかった。しかし反面、第5章で触れたように、多くの電子図書館の試行が進んでおり、しかも、インターネットなどを通じて専門家のみならず、一般ユーザを対象として実験を公開し、意見を交換し合っているケースも見受けられる。このような形態で行われていれば、様々な人々の要望がフィードバックされることだろう。その方面からの成果を期待したい。

第 9 章

本論文のまとめ

9.1 本論文の成果

本論文では以下のような内容について論じ、提案を行ってきた。

まず第 2章では、本研究で前提としている構造化文書について、特に論理構造を記述する規格であるSGML に視点を置いて、SGML による構造化文書の記述法と、同様の目的を持つ規格 ODA の場合とを比較し、構造化文書の記述についての解説を行った。また、SGML による記述が既に普及しており、また将来的にも一層の浸透が見込まれる証左として、HTML および CALS として SGML が活用されている事例を挙げるとともに、SGML について処理などで課題とされる三点を取り上げ、一般的に困難であるとされる例や理由、またこれらの特徴の本研究の前提への影響などについて述べた。

続いて第 3章では、本研究で対象とする論理構造あるいは構造化された文書とはどういうものか、またその論理構造が多様であるとはどういう状態をいうのか、具体的に例を挙げて説明し、本研究の対象とする範囲を明確にした。また本研究で目指している文書活用システムとはどのようなシステムなのか、その機能や特徴を示し、それを実現するために論理構造の変換が必要であることを述べた。

続く第 4章では、これまでに構造化文書を扱う場合、どのような手法が検討されてきたか、文法理論の体系に添う物や SGML の特性を活かしたものを概論し、それらの試行を踏まえて本研究で採るべき方向について、論理要素間の対応付けで、論理構造の出現位置を考慮する方式が望ましいことが裏付けられた。

また第 5章で、同様に関連する研究の中から、文書データベースをコンピュータネットワークを活用して構築する「電子図書館」について多く実験が行われている状況に着目して、それらの動向をまとめ、かつ文書の記述には SGML を採用するケースが現在も多く、将来的には一層増加することを裏付け、本研究で SGML を前提とする妥当性を示した。同時に SGML を用いる場合やそれ以外でも、文書を扱う際に課題とされる点を整理して文書データベースが実用化されるために検討が待たれる様々な局面を紹介し、一方で現在の技術で統合的な文書データベースとしてはどのようなシステムが実装され、試行実験が行われているか、システムの全体的イメージやその達成度について理解を得る情報を提供した。

ここまでに述べた課題や研究状況を踏まえて、本研究では有用な文書活用システムを実現するためには論理構造の変換手法が大きな役割を果たすものと考えて、本論文では引続きこの手法を提案する。第 6章でその手法について、従来の研究から本研究が採るべき前提や立場、全体の処理の構成、処理の記述に必要な概念、そして変換手法の核心となる変換仕様の記述法について説明した。

続いて第 7章では、前章に述べた変換仕様を用いた文書の変換を、サンプルデータを用いた処理の様子とともに、一般的な記述を用いた汎用のアルゴリズムとして説明した。また、SGML では文書の構成の特徴として、比較的出現範囲が柔軟な浮動要素の記述を採用しているが、これに本手法も対応し、またそれ以外の用途でも記述力を高めるために変換仕様の記述法を拡張した。

さらに第 9章では本手法をプロトタイプとして実装し、実際の論文で SGML 化されたデータと現在広く使われている HTML を用いて実験を行い、より実際に近い状況で本手法が有効に機能することを証明した。その上で、従来の同様の手法を第 4章などで概観した課題についての観点で比較して、本手法がより有利であることを述べ、一方次の段階として本手法による論理構造の変換がより高機能になるために、取り組むべき課題を挙げ、またデータベースあるいはブラウザとして総合的なシステムを構築するために解決すべき課題を述べた。

以上の記述により、現在の電子化・構造化文書にまつわる状況が把握されるとともに将来的な見通しが立ち、それを踏まえた上で、本研究が描く次世代的なデータベースが、文書全体に対してデータベースはもちろん、閲覧などを含めた統合的な処理ができる文書活用システムであり、その機能などが現在から将来への状況やユーザのニーズといった各方面から鑑みて妥当でありかつ実現が待たれるものであることを示した。さらにその実現のために欠くことのできない、論理構造の多様性を吸収する機構について、その道筋を一般的な記述でまた具体的なケースを対象として示した。

9.2 おわりに

本研究では、SGML による多様な論理構造を持つ文書を、統一的に扱いユーザに提示するようなデータベースを提案し、その核となる論理構造の変換手法について検討した。このような手法を採用することにより、文書データベースは、学会や出版社が独自に使用している論理構造を統合して扱うことができ、さらにユーザには必要に応じて統一した論理構造に基づく検索や、同じレイアウトに整形した画面表示などを提供することができる。

文書はますますマルチメディア、つまり画像などを取り入れる方向が盛んになっているが、それらも含めて土台となるのはやはりテキストであり、文書の骨格である論理構造であることは当分変わらないであろう。しかし中でも触れたように、文書の論理構造をどのように操作したいのか、という基本的な問いにはいまのところ明解な解答や進歩は見い出せていない。

コンピュータやネットワークの普及もさらに進むだろうし、そうするとより多くの人が意見や要望を持つだろう。電子図書館という形態でオープンな実験も進んでいることから、フィードバックの方も増えていくだろう。

一方で文書の使い道も確かに変わってきている。SGML は論理構造のみを扱うことを目指していたが、HTML がほぼレイアウト指向のマークアップに移行したように、その他の文書でもレイアウトは切り離せないという意見も大きくなってきた。確かに、周囲と字体の違う部分を「強調」と呼べば論理構造であるが、「イタリック」と呼べばレイアウトに関連する。もっと率直に、要素の配置を意識する需要も確実にある。

電子化文書を取り巻く環境は、過渡期にあると言えるだろう。それはまた、コンピュータやネットワーク自体の過渡期の一端でもあるかもしれない。そういう時期にこの研究に取り組んだというのは、展開の大きさと速さから興味深さとつらさのどちらも大きかったが、このような山を越えたころに見えてくるはずである一歩前進した電子化文書の活用に本研究が一役買えていれば、まさに望外の喜びである。

謝辞

本研究を進めるに当たりまして、修士・博士の両課程、合わせて5年という長い間御指導くださいました、安達淳教授に、心から感謝いたします。安達教授は常に熱意を持って、時に弛緩しがちな私の研究を叱咤激励してくださいました。また、単にこの研究を進めるに関することのみならず、広くコンピュータに関して学問的な命題も、趣味的な喜びも、豊富な知識と経験、また盛んな興味や好奇心、さらに深い御考察と思慮を働かせて、幅広い見聞を与え、またそのお姿を通じて、自らもそのようになりたいと思う刺激を加えていただきました。

また研究室の形としても、十分な計算機のパワーやその他資源を整えてくださり、またそれらを主体的に使うことで理解を深める機会を持てるよう、お心を配っていただき、一方で学会発表などへの参加も私たちが望むだけ実現させていただけるような、恵まれた環境も常に用意しておいていただきました。先生も研究室と近くに関わることを楽しんでいらっしゃるようで、定期的にミーティングを持つことをはじめ、しばしば研究室を訪れてはあれこれ質問やさりげない会話など気さくになげかけてくださいました。先生に気にかけていただいているという感覚は研究を充実させる上で、とても励みになったように思います。

安達教授と同じく、学術情報センターで助教授の職を勤めていらっしゃる高須淳宏先生にも感謝いたします。先生は、理論的な側面を特に得意とされていて、ともすると、アドホックあるいは実用的になりすぎたり、勘といったものだけで暴走しそうになる私の研究を、折々関連する事項を示し、また理論的裏付けの観点を指摘して引き戻してくださいました。

また、大学院生活で唯一外国に行っての発表に際しては、海外旅行自体がはじめてで右も左もわからない私を、暖かくかつ過保護にならず様々を体験できる距離で付き添っていただきました。発表自体はお世辞にも満足のいくものではありませんでしたが、私にとっては大変に大きな経験になりました。

研究室でともに過ごした方々にもお礼を申し上げます。

安達研の第一期生でもある倉島顕尚さんとは、四学年上級でいらっしゃったということで、私が安達研に来た最初の一年をご一緒しただけですが大変に強い印象を受け、多くを学ばせていただきました。研究室にいらしたころは、自分が未熟過ぎてそれほど理解が及びませんでしたが、研究面でもりっぱな成果を挙げられて、以降社会に出ても充実したご活躍ぶりは折々にお会いするたびに痛感していました。研究や仕事以外でも、学情三研究室の求心力となってイベントや日々のコミュニケーション、計算機のメンテナンスなど研究室生活自体を親密にされるな存在でいらしゃったのみならず、ネットワークニュースなどを通じて、その御活動が広い範囲に及んでいらっしゃったことには、自分がそういう学年になった今、つくづく敬服しました。

同様に、三年上級でいらっしゃった片山紀生さんとは、研究室で三年ご一緒したのち、片山さんが学術情報センターに就職されたことにより教官として二年間、勉強をさせていただきました。片山さんも学生当時より、御自身の研究はもとより、様々なことを学んで、他のメンバーの研究のヒントや議論をしばしば提示してくださる上、フリーのソフトウェア作りに参加されるなど積極的に活動されていらして、広範でかつ深い知識を持っていらっしゃり、さらに増やす姿勢を強く持っていらしたこと、また研究におかれても強い意志と集中力を持って取り組んでいらしていたこと、そして身体能力も含めて全般的に向上を目

指されるお姿には、折々反省もし勉強もさせていただくきっかけとなりました。残念ながら教官となられてからはなかなかお話や指導をいただく機会には恵まれませんでしたが、部屋は近かったこともあり、変わらない姿勢でお仕事に取り組まれていらっしゃる御様子やお噂などを聞くたびに、襟を正す心地を新たにしていました。

さらに一年上級でいらっしゃる西澤格さんとは、最も長い四年間をご一緒させていただきました。西澤さんも確実に研究を仕上げていかれる御様子を感銘深く拝見していましたが、それに加えて、学問的なこともちろん、インターネット上やその他のトレンドなど、多岐に渡って旺盛な興味を持って吸収されていくことが印象に残りました。

その他挙げればきりはありませんが、今年最高学年となり不慣れな立場で口うるさくまた至らないところもあったであろう様々な局面で、助けてくれた二年下の太田学さんはじめ安達研のみなさまには、私が諸先輩方から受けたような感化を示せるほどになれなかったことを申し訳なく思うとともに、楽しい研究室生活をもたらしてくださったことを感謝します。

研究室は学情という、学科の中では三研究室だけでひとつの単位を作るような状況の中で濱田喬先生・浅野正一郎先生他諸先生方には、時には研究に必要な議論や道具立てをくださり、また計算機の管理を通じて計算機とはどういうものかを考え、ユーザとしてではなく計算機を使うようになれる機会をくださったこと、また例年の忘年会などの折りには含蓄のあるあるいは愉快な話を聞かせてくださったことなど、感謝いたします。

同じく、なにかと相談に乗っていただき、真剣なあるいはゲームめかした議論を 楽しませてくださった濱田研で学年は一つ違う中澤聡さんを含めて、濱田研究室・浅野研究室の諸先輩方、同級の方々、そして後輩の皆様には、研究そのものを直接議論する機会はあまりなかったものの、共同作業を通じて、折々のイベントを通じて、あるいは日常的な関わり合いを通じて様々な知識や思い出を与えていただきました。ありがとうございました。

さらに、就職しても研究室が離れていても親交を保っていてくださった友人諸氏にも心から感謝申し上げます。やはりこの方々との語らいなどの交流により、研究に行き詰まった折りにも鼓舞し、立ち直ることができたのだと、振り返って痛感します。主に学部時代からの学科の同期の皆様や、卒論時代から、学部を卒業しても寄らせていただいた、浅田邦博教授はじめ浅田研究室の関係のみなさまは、心の拠り所とさせていただいていたように思います。

その他、個別に申し上げることができなくて心苦しいのですが、この間に御指導あるいは御助力を、または研究をすすめるために組織を運営するための様々な活動をされていた方々、みなさまにお礼を申し上げます。

今後の活動でもみなさまの御恩を忘れることなく、いくらかでも報いるべく、仕事やその他あらゆる局面で肝に銘じて歩んでいきたいと思います。ほんとうにありがとうございました。

参考文献

[1] Eric van Herwijnen （SGML 懇談会実用化 WG 訳）: “実践 SGML,” 日本規格協会 (1992).

[2] Brian E. Travis, Dale C. Waldt （学研/スリーエーシステムズ SGML 事業室訳）: “SGML 実践ガイド – 導入と活用の手引,” シュプリンガー・フェアラーク東京 (1996).

[3] Dawna Travis Dewire （本間浩一他訳）: “テキスト・マネジメント,” RID Telecom (1995).

[4] 若鳥 陸夫, 坂入 隆, 真野 芳久: “ODA: 多様な文書のための標準様式,” 情報処理, vol. 37 – no. 3, pp. 199 – 206 (March 1996).

[5] “HyperText Markup Language (HTML),” http://www.w3.org/pub/WWW/MarkUp/MarkUp.html

[6] CALS 推進協議会編: “日本版 CALS – 実践のためのガイドブック –,” オーム社 (1995).

[7] 手塚 潤治: “ハロー！CALS,” オーム社 (1995).

[8] 石塚 英弘, 根岸 正光: “情報システム基盤技術としての SGML – 文書データベースから WWW そして CALS まで –,” 情報処理, vol. 37 – no. 3, pp. 207 – 212 (March 1996).

[9] CIF ホームページ: “http://www.cif.or.jp/”

[10] Jim Hearth, and Larry Welsh: “Difficulties in Parsing SGML,” ACM Conference on Document Processing Systens (Decenber 1988).

[11] G. E. Blake, T. Bray, and F. WM. Tompa: “Shortening the OED: Experience with a Grammar-Defined Database,” ACM Transaction Information Systems, Vol.10/No.3 (July 1992), pp. 213–232.

[12] Gaston H. Gonnet, and Frank WM. Tompa: “Mind Your Grammar: A New Approach to Modelling Text,” Proceedings of the 13th International Conference on VLDB (Very Large Data Bases) (1987), pp. 339–346.

[13] Darrell R. Raymond, and Frank WM. Tompa: “Hypertext and the Oxford English Dictionary,” Communication of the ACM, Vol.31 – No.3 (July 1988), pp.871–879.

[14] Darrell R. Raymond: “Flexible Text Display with Lector,” IEEE Computer vol. 25 – no. 8 (August, 1992).

[15] Jos Warmer, and Sylvia van Egmond: “The Implementation of the Amsterdam SGML Parser,” Electronic Publishing Vol.2/No.2 (July 1989), pp. 65–90.

[16] Lynne A. Price, and Joe Schneider: “Evolution of an SGML Application Generator,” ACM Conference on Document Processing Systems (1988), pp. 51–60.

[17] Warmer, Jos and van Vliet, Hans "Processing SGML Documents," Electronic Publishing, Vol.4/No.1 (March 1991), pp. 3–26.

[18] 高橋 亨, 東野 純一: "SGML 文書の変換・再利用のための言語 'AESOP'," 情報知識学会第 3 回研究報告会講演論文集 (May 1995).

[19] S. C. Johnson: "YACC – yet another compiler compiler," Computing Science Technical Report 32, AT&T Bell Laboratories (1975).

[20] 永田 治樹: "大学におけるディジタル図書館 – 英国並びにオランダの大学図書館での試み," 第 5 回ディジタル図書館ワークショップ, http://www.dl.ulis.ac.jp/DLjournal/No_5/sugimoto/nagata.html (1995).

[21] 杉本 重雄: "Digital Libraries へのアプローチ – 米国を中心とする取組み事例に基づく考察," 第 5 回ディジタル図書館ワークショップ, http://www.dl.ulis.ac.jp/DLjournal/No_5/sugimoto/sugimoto.html (1995).

[22] "Digital Library Research, " http://www.dlib.org/projects.html

[23] 杉本 雅則, 片山 紀生, 越塚 美加, 神門 典子, 高須 淳宏, 安達 淳: "世界の電子図書館の研究動向について," 学術情報センター紀要 no. 8 (March 1996).

[24] "TULIP: The University Licensing Program," http://www.elsevier.nl/info/projects/tulip.htm

[25] Laurie Crun: "University of Michigan Digital Library Project," Communications of the ACM, vol. 38 – no. 4, pp. 63 – 64 (April 1995).

[26] "TULIP Journals Browser," http://tulipsrvr.engin.umich.edu:80/cgi-bin/demo/listjournal?demo

[27] "Illinois Digital Library Initiative Project," http://www.grainger.uiuc.edu/dli/

[28] Bruce Schatz, William H. Mischo, Timothy W. Cole, Joseph B. Hardin, Ann P. Bishop, and Hsinchum Chen: "Federating Diverse Collections of Scientific Literature," IEEE Computer vol. 29 – no. 5 (May 1996).

[29] Bruce Schatz: "Building the Interspace: The Illinois Digital Library Project," Communications of the ACM, vol. 38 – no. 4, pp. 63 – 64 (April 1995).

[30] Michael Lesk: "The CORE Electronic Chemistry Library," Bellcore 社資料 (1992).

[31] Richard Entlich, Lorrin Garson, Michael Lesk, Lorraine Normore, Jan Olsen, and Stuart Weibel: "Making a Digital Library: The Chemistry Online Retrieval Experiment," Communications of th ACM vol. 38 – no. 4 (April 1995).

[32] Richard Entlich, Lorrin Garson, Michael Lesk, Lorraine Normore, Jan Olsen, and Stuart Weibel: "Making a Digital Library: The Contents of the CORE Project," http://community.bellcore.com/lesk/chem94/chtx.html .

[33] Guy A. Story, Lawrence O'Gorman, David Fox, Louis Levy Schaper, and H. V. Jagadish: " The RightPages Image-Based Electronic Library for Alerting and Browsing," IEEE Computer, vol. 25 – no. 9 (September, 1992).

[34] Melia M. Hoffman, Lawrence O'Gorman, Guy A. Story, James Q. Arnold, and Nina H. Macdonald: "The RightPages^{TM} Service: An Image-Based Electronic Library," Journal of the American Society for Information Science, vol. 44 – no. 8, pp. 446 – 452 (August 1993).

[35] Lawrence O'Gorman: "Image and Document Processing Techniques for the RightPages Electronic Library System," Proceedings of the 11th IAPR (International Association for Pattern Recognition) International Conference, pp. 260 – 263 (1992).

[36] 長尾 真, 原田 勝, 石川 徹也, 谷口 敏夫, 澤田 芳郎, 吉田 哲三, 柿元 俊博: "電子図書館 Ariadne の開発 (1)–(5)," 情報管理, vol. 38 –no. 3-7 (June - October, 1995).

[37] 原田 勝: "電子図書館 Ariadne について – 検索機能を中心として –," 情報処理学会情報学研究会報告 (情報学基礎) 95-FI-37, pp. 31 – 36 (May 1995).

[38] "Electronic Library," http://ariadne0.kuee.kyoto-u.ac.jp/

[39] 岸本 行生, 須之内 美幸, 塚田康博, 千葉 滋, 石川 徹也: "テキストの構造化に基づく検索システム," 情報処理学会論文誌, vol. 35 – no. 5, pp. 908 – 916 (May 1994).

[40] 日商岩井インフォコムシステムズ: "日本語 SGML エディタ SGML-PLUS Ver. 2.0," ユーザーズマニュアル (1996).

[41] Netscape Communications ホームページ: http://www.netscape.com.

発表文献

酒井 乃里子, 高須 淳宏, 安達 淳:
"SGML 文書構造の文法を用いた変換処理,"
情報処理学会情報学基礎研究会 (機械振興会館, 東京),
研究報告 94-FI-33 vol.94-no. 37, 1994 年 5 月 18 日.

Noriko Sakai, Atsuhiro Takasu, Jun Adachi:
"Full-text Database of Scientific Documents with Various Structures,"
FID (International Federation for Information and Documentation) 47th Conference and Congress (Sonic Center, Omiya), October 8th, 1994.

酒井 乃里子, 高須 淳宏, 安達 淳:
"SGML 文書本体部の論理構造の変換"
1995 年電子情報通信学会総合大会 (福岡工業大学, 福岡市),
予稿集 D-94, 1995 年 3 月 30 日.

酒井 乃里子, 高須 淳宏, 安達 淳:
"SGML 文書による全文データベースのための文法的処理を用いた論理構造の変換手法,"
学術情報センター紀要 第 7 号, 1995 年 3 月 31 日.

酒井 乃里子, 高須 淳宏, 安達 淳:
"文書の論理構造の変換におけるデータ形式の指定法,"
情報処理学会第 53 回全国大会 (大阪工業大学, 枚方市),
予稿集 3S-4, 1996 年 9 月 5 日.

Noriko Sakai, Atsuhiro Takasu, Jun Adachi:
"Logical Structure Transformation between SGML Documents,"
IPIC'96 (the International Working Conference on Integration of Enterprise Information and Processes, "Rethinking Documents," MIT, Boston), November 14th, 1996.

酒井 乃里子, 高須 淳宏, 安達 淳:
"SGML 文書の論理構造変換手法,"
情報処理学会論文誌 (投稿中).

酒井 乃里子, 高須 淳宏, 安達 淳:
"SGML 文書データベースの論理構造変換における浮動要素の処理手法,"
電子情報通信学会データ工学研究会第 8 回ワークショップ DEWS'97 (投稿中)

付録 A

学情紀要第 7 号 DTD

学情紀要第 7 号 DTD

```
<!DOCTYPE article [
<!-- Academic DTD -->
<!-- JOURNAL NAME -->
<!-- @(#)template.dtd        1.2 6/11/93 -->
<!-- Author: DTD AUTHOR -->
<!-- Copyright 1993, NACSIS and Uniscope, Inc. -->
<!--                    All Rights Reserved                                    -->
<!-- THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF NACSIS AND UNISCOPE, INC. -->
<!--   The copyright notice above does not evidence any actual     -->
<!--   or intended publication of such source code.                -->

    <!-- ENTITY % dtd        "INCLUDE"         include dtd declarations-->
    <!-- ENTITY % doc        "IGNORE"         ignore embedded documentation -->

    <!-- ENTITY % PublicEnt PUBLIC
        "-//Gakujutsu//ENTITIES Declarations For Public Element Sets//EN" -->
    <!-- %PublicEnt; -->

    <!-- Public text ENTITY set ENTITY -->
    <!-- ENTITY % TextEnt PUBLIC
        "-//Gakujutsu//ENTITIES Declarations For Text Entity Sets//EN" -->
    <!-- %TextEnt; -->

<!-- [ %dtd; [ -->
<!-- doctype article [  -->

        <!ENTITY amp                SDATA        "@amp@"        --ISOnum-->
        <!ENTITY lt                 SDATA        "@lt@"         --ISOnum-->
        <!ENTITY gt                 SDATA        "@gt@"         --ISOnum-->
        <!ENTITY eacute             SDATA        "@eacute@"        --ISOlat1-->
        <!ENTITY agrave             SDATA        "@agrave@"        --ISOlat1-->
        <!ENTITY ccedil             SDATA        "@ccedil@"        --ISOlat1-->
        <!ENTITY egrave             SDATA        "@egrave@"        --ISOlat1-->
        <!ENTITY oacute             SDATA        "@oacute@"        --ISOlat1-->
        <!ENTITY ugrave             SDATA        "@ugrave@"        --ISOlat1-->
        <!ENTITY ecirc              SDATA        "@ecirc@"        --ISOlat1-->
        <!ENTITY ocirc              SDATA        "@ocirc@"        --ISOlat1-->
        <!ENTITY acirc              SDATA        "@acirc@"        --ISOlat1-->
        <!ENTITY icirc              SDATA        "@icirc@"        --ISOlat1-->
        <!ENTITY ucirc              SDATA        "@ucirc@"        --ISOlat1-->
        <!ENTITY oelig              SDATA        "@oelig@"        --ISOlat2-->
        <!ENTITY ouml               SDATA        "@ouml@"        --ISOlat2-->
        <!ENTITY uuml               SDATA        "@uuml@"        --ISOlat2-->
        <!ENTITY iuml               SDATA        "@iuml@"        --ISOlat2-->
        <!ENTITY auml               SDATA        "@auml@"        --ISOlat2-->
        <!ENTITY ccaron             SDATA        "@ccaron@"        --ISOlat2-->
        <!ENTITY scaron             SDATA        "@scaron@"        --ISOlat2-->
        <!ENTITY zcaron             SDATA        "@zcaron@"        --ISOlat2-->
        <!ENTITY oslash             SDATA        "@oslash@"        --ISOlat2-->
        <!ENTITY dyogh              SDATA        "@dyogh@"        --ISOlat2-->
        <!ENTITY Gamma              SDATA        "@Gamma@"        --ISOlat2-->
        <!ENTITY hstrok             SDATA        "@hstrok@"        --ISOlat2-->
        <!ENTITY istrok             SDATA        "@istrok@"        --ISOlat2-->
        <!ENTITY jcaron             SDATA        "@jcaron@"        --ISOlat2-->
        <!ENTITY nlw                SDATA        "@nlw@"         --ISOlat2-->
        <!ENTITY eng                SDATA        "@eng@"         --ISOlat2-->
        <!ENTITY clsomg             SDATA        "@clsomg@"        --ISOlat2-->
        <!ENTITY rfh                SDATA        "@rfh@"        --ISOlat2-->
        <!ENTITY jtilde             SDATA        "@jtilde@"        --ISOlat2-->
```

```
<!-- Parameter entities -->
<!-- ENTITY % computertext        "#PCDATA|arg|key"
                                    text into or out of computer -->
<!ENTITY % lists        "la|lm|ln|lu|ld|ldp"        -- list ELEMENTs -->
<!ENTITY % abstract.copy        "p|%lists;"
                                 -- set of paragraph-level ELEMENTs
                                 allowed in abstract -- >
<!ENTITY % copy        "p|fg|xmp|%lists;|df|dfg|tb"
                                 -- paragraph-level ELEMENTs -- >
<!ENTITY % emph        "e1|e2"                -- emphasis - text that is either
                                 italic or bold in the english -- >
<!ENTITY % text        "#PCDATA|%emph;|sbs|sps|ob|ub|sc"
                                 -- simple text -- >
<!ENTITY % txtlmts        "%text;|xref|fn|f|arg|key|ref|citref|prompt|input"
                                    -- text ELEMENTs -- >
<!ENTITY % f.oper  "mark | markref | break | sup | sub | sum | integral|
                  product | plex| frac | diff|  sqrt | root| square | power
        | pile | matrix | fence | middle | tensor | mfn |  box | vec "
                     -- all the operators + chemical formula -->
<!ENTITY % f.text       " #PCDATA | roman | italic | ov" -- "text" -->
<!ENTITY % f.align      "centre | left | right" -- alignment of elements -->

<!ENTITY % f.pos        "post | pre | mid"      -- position of elements -->

<!ENTITY % f.style      "single | double | triple | dash | dots | bold"
                                 -- style of elements -->
<!ENTITY % f.type       "paren | bracket | angbrack | brace | bar | none"
                                 -- fence type   -->
<!ENTITY % f.ov    "dot | dotdot | dot3 | dot4| tie | tiebrace | hat |
           haczeck | acute | grave | cedil | ring | macron | ognonek | dblac | breve |
           tilde | vec | rvec | dyad | bar"
                                 -- over chars -->
<!ENTITY % f.diff       " normal | partial"
                                 -- types of differentiation -->
<!ENTITY % f.func1      "and | antilog | arc | arccos | arcsin | arctan |
         arg | colog | cos | cosh | cot | csc | ctn | deg | det | dim | exp | for |
         gcd | glb | hom | if | Im | ker | lg | lim | ln | log | lub | max | min"
                                 -- 32 first functions -->
<!ENTITY % f.func2      "mod | Re | sec | sin | sinh | tan | tanh"
                                 -- remaining functions -->
<!-- ]] -->

<!ENTITY % def.article.type        "paper"
                                 -- an article's default type is
                                 "paper" -->
    <!ENTITY % undef.element "input|prompt|key|arg"
                          --        undefine elements   added by kato@Fujitu -->
    <!ELEMENT (%undef.element;)        - -        (#PCDATA)
                          --        undefine elements   added by kato@Fujitu -->

<!-- Element set ENTITY references -->
<!-- %ArticleKeywordsEle; -->
<!ELEMENT article        - 0        (front, body, end)
                                        -- an article in a journal -->
<!ATTLIST article        journal        NMTOKENS        #IMPLIED
                                        -- the name of the journal --
                    vol        NUMBER        #IMPLIED
                                        -- the volume number of the
                                        article --
```

```
                    num         NUMBER        #IMPLIED
                                             -- the issue number within
                                             the volume --
                    page        NUMBER         #IMPLIED
                                             -- the starting page number
                                             of the published article --
                    page2        NUMBER         #IMPLIED
                                             -- the ending page number of
                                             the published article --
                    date        NUMBER         #IMPLIED
                                             -- the publication date of the
                                             article --
                    recvd        NUMBER         #IMPLIED
                                             -- the date the article was
                                             received by the publisher --
                        type        (note|letter|paper) %def.article.type;
                                             -- what type of article it
                                             is --
                    acptd        CDATA         #IMPLIED
                                             -- 原稿審査完了日（複数ある場合
                                             は、最終日付） --
                    revdd        CDATA         #IMPLIED
                                             -- 原稿改訂日（複数ある場合は、
                                             最終日付） --
                    txtl        CDATA        #IMPLIED
                                             -- 本文言語コード（目録言語
                                             コード使用） --
                    field        CDATA         #IMPLIED
                                             -- 論文の分野（雑誌に示されてい
                                             る分野） -->

<!ELEMENT front        0 0        (atl, runt?, author+, abstract?, keywords?, ptoc?)
                                             -- stuff at the beginning
                                             of the article, much of which
                                             is defined in other .ele files
                                             -->
<!ELEMENT runt         - 0        ((%txtlmts;)*)        -- running title for continuation
                                             pages -->
<!ELEMENT atl         - 0        ((nihongo.title, english.title?) | (english.title,
                    nihongo.title?))
                                                    -- 論文タイトル -->
<!ELEMENT nihongo.title
                        - 0        ((%txtlmts;)*, subt?)
                                                 -- 日本語によるタイトル -->
<!ELEMENT english.title
                        - 0        ((%txtlmts;)*, subt?)
                                                 -- 欧文によるタイトル -->
<!ELEMENT ptoc         - 0        EMPTY                -- 内容目次 -->

<!-- %AbstractEle; -->
<!ELEMENT abstract        - -        ((abstract.nihongo, abstract.english?) |
                            (abstract.english, abstract.nihongo?))
                                    -- nihongo and/or english language
                                    abstract of the article -->
<!ELEMENT abstract.nihongo        - 0        ((%abstract.copy;)+|atopic+)
                                    -- nihongo version of the
                                    abstract -->
<!ELEMENT abstract.english        - 0        ((%abstract.copy;)+|atopic+)
                                    -- english language version of the
                                    abstract -->
<!ELEMENT atopic        0 0        (t, (%abstract.copy;)+) -(subt)
                                    -- topic within the abstract -->
```

```
<!-- %AuthorEle; -->
<!ELEMENT author        - O        (g.name, f.name, title?,
                              position?, address?, present.position?,
                              present.address? ,photo?, profile?)
                                     -- the full details about each
                                     author collected from the various
                                     parts of the article -->
<!ELEMENT g.name        - O        (((kanji, furigana), english?)| (english, (kanji,
                                                furigana)?))
                                     -- the 'given' name of the author -
                                     the first name for westerners, and
                                     the second name for japanese.
                                     Non-japanese names will tend not to
                                     use the kanji and furigana fields,
                                     although these can be used for
                                     katakana renderings of western
                                     names -->
<!ELEMENT f.name        - O        (((kanji, furigana), english?)| (english, (kanji,
                                                furigana)?))
                                     -- the author's family name - the
                                     second name for westerners, and the
                                     first name for japanese.  See also
                                     <g.name> -->
<!ELEMENT title         - O        ((kanji, english?) | (english, kanji?))
                                     -- honorifics before a person's name
                                     -->
<!ELEMENT position        - O        (((kanji, furigana), english?)| (english, (kanji,
                                                furigana)?))
                                     -- the position the author held when
                                     the article was submitted -->
<!ELEMENT present.position
                        - O        (((kanji, furigana), english?)| (english, (kanji,
                                                furigana)?))
                                     -- the author's present position if
                                     it has changed since the article was
                                     submitted -->
<!ELEMENT profile        - O        (((kanji, furigana), english?)| (english, (kanji,
                                                furigana)?))
                                     -- the short profile of the author
                                     that may appear at the end of the
                                     article -->
<!ELEMENT photo         O O        (fgart)
                                     -- the photo that may accompany the
                                     author's profile text -->
<!ELEMENT address        - O        ((kanji, english?) | (english, kanji?))
                                         -- 住所 -->
<!ELEMENT present.address
                        - O        ((kanji, english?) | (english, kanji?))
                                     -- 現住所 -->

<!-- %ChemicalEle; -->
    <!-- %ChemEle; -->

<!-- %DateEle; -->
<!ELEMENT date          O O        ((month, (day? & year?)) |
                              (day, (month? & year?)) |
                              (year, (month? & day?)))
                                     -- Date model: allows month, day
                                     and year in any order, but requires
                                     one of these ELEMENTs to be present
                                     -->
```

```
<!ELEMENT year        - O        (#PCDATA)
                                   -- the year -->
<!ELEMENT month        - O        (#PCDATA)
                                   -- the month -->
<!ELEMENT day        - O        (#PCDATA)
                                   -- the day of the month
                                   -->

<!-- %EndEle; -> EndAppendix -->
<!ELEMENT end        O O        (acknowledge?, appendix*, references?)
                                   -- end of article, comprising
                                   possible acknowledgements and
                                   possible references -->
<!ELEMENT acknowledge        - O        (p+)
                                   -- acknowledgements comprising
                                   one or more paragraphs -->
<!ELEMENT appendix                - O        (t?, (%copy;)*, sec*)
                                   -- appendix comprises
                                   optional title, optional
                                   paragraph-level ELEMENTs
                                   and optional sections -->
<!ATTLIST appendix                id        ID        #IMPLIED
                                   -- ID for cross-referencesencing -->

<!ELEMENT references        - O        EMPTY
                                   -- empty ELEMENT in case
                                   formatting system needs place
                                   marker to put references.  Actual
                                   references comprising one or
                                   more references to other papers
                                   or notes to clarify the text
                                   appear in the body of the text -->
<!ELEMENT notes                - O        EMPTY
                                       -- 注記 -->

<!-- %EquationEle; -->
    <!-- %MathEle; -->
<!ELEMENT  f            - -  (%f.text; |%f.oper;)+          >
<!ELEMENT df            - -  ((%f.text; |%f.oper;)+| fgart) >
<!ATTLIST df            id      ID      #IMPLIED
                    -- align        (%f.align)      left --
                    -- hmove        CDATA           0 --
                    num     (num | nonum)   nonum                    >
<!ELEMENT dfg           - -  (df)+                                      >
<!ATTLIST dfg           id      ID              #IMPLIED
                    align   (%f.align)      left
                    num     (num | nonum)   nonum                    >
<!ELEMENT mark          - O    EMPTY                                    >
<!ATTLIST mark          id      ID      #REQUIRED                       >
<!ELEMENT markref       - O   EMPTY                          >
<!ATTLIST markref       refid   IDREF           #REQUIRED               >
<!ELEMENT break         - O   EMPTY                                     >
<!ATTLIST break         type    (required | optional)   required      >
<!ELEMENT box           - -     (%f.text; |%f.oper;)+                   >
<!ATTLIST box           style   (%f.style;)      single                 >
<!ELEMENT ov            - -     (%f.text; |%f.oper;)+                   >
<!ATTLIST ov            type    (%f.ov;)        bar
                    pos     (above | below | mid)   above
                    style   (%f.style; )    single                  >
```

```
<!ELEMENT (sup|sub)      - -     (%f.text; |%f.oper;)+                  >
<!ATTLIST (sup|sub)      pos     (%f.pos;)       post                   >
<!ELEMENT  tensor        - -     (#PCDATA)                              >
<!ATTLIST tensor         posf    (sup|sub)       sup
                     suffix  CDATA           #REQUIRED              >
<!ELEMENT (roman|italic)         - -     (#PCDATA)                      >
<!ELEMENT vec            - -     (#PCDATA)                              >
<!ELEMENT sum            - -     ((from | to)*, of?)                    >
<!ELEMENT integral       - -     ((from | to)*, of?)                    >
<!ELEMENT product        - -     ((from | to)*, of?)                    >
<!ELEMENT plex           - -     (operator, (from | to)*, of?)          >
<!ELEMENT (from|to|of)   - 0     (%f.text; | %f.oper;)+                 >
<!ELEMENT operator       0 0     (%f.text;)                             >
<!ELEMENT frac           - -     (numer, over)                          >

<!ATTLIST frac           align   (%f.align;)     centre                 >
<!ELEMENT numer          0 0     (%f.text;|%f.oper;)+                   >
<!ELEMENT over           - 0     (%f.text;|%f.oper;)+                   >
<!ELEMENT diff           - -    (diffof, by)                           >
<!ATTLIST diff           type    (%f.diff;)      normal                 >
<!ELEMENT diffof         0 0    (%f.text; |%f.oper;)+                  >
<!ELEMENT by             0 0     (%f.text; |%f.oper;)+                  >
<!ELEMENT pile           - -     (above1, (above)+)                     >
<!ATTLIST pile           align   (%f.align;)    centre                  >
<!ELEMENT above1         0 0     (%f.text; | %f.oper;)+                 >
<!ELEMENT above          - 0     (%f.text; |%f.oper;)+                  >
<!ELEMENT matrix         - -     ((col)+)                               >
<!ELEMENT col            - -     (above1, (above)+)                     >
<!ATTLIST col            align   (%f.align;)     centre                 >
<!ELEMENT sqrt           - -     (%f.text; |%f.oper;)+                  >
<!ELEMENT root           - -     (degree, of)                           >
<!ELEMENT degree         0 0     (%f.text; |%f.oper;)+                  >
<!ELEMENT square         - -     (%f.text; |%f.oper;)+                  >
<!ELEMENT power          - -     (degree, of)                           >
<!ELEMENT mfn            - -     ((fname, of) | (%f.text; | %f.oper;)+) >
<!ATTLIST mfn            type1   (%f.func1;)     #IMPLIED
                     type2   (%f.func2;)     #IMPLIED               >
<!ELEMENT fname          - 0     (#PCDATA)                              >
<!ELEMENT fence           - -    (%f.text; | %f.oper;)+                 >
<!ATTLIST fence          type    (%f.type;)      paren
                     open    CDATA           #IMPLIED
                     close   CDATA           #IMPLIED
                     style   (%f.style;)     single                 >
<!ELEMENT middle         - -     (%f.text;)                             >
<!ATTLIST middle         style   (%f.style;)     single                 >


<!-- %FigureEle; -->
<!ELEMENT fg         - - (t?, fgart)
                                     -- figure -->
<!ATTLIST fg         id        ID        #IMPLIED
                                     -- id for cross-referencesencing --
                     numbered (YES|NO)        YES
                                     -- numbering attribute -->
<!ELEMENT fgart        0 0 ((fg|%txtlmts;)*)
                                     -- figure art -->
<!ATTLIST fgart        file        CDATA        #IMPLIED -- NMTOKEN #CONREF
                                      filename -->
```

```
<!-- %KeywordsEle; -->
<!ELEMENT keywords        - -        (keyword+)
                                    -- list of keywords -->
<!ELEMENT keyword        - O        (((kanji, furigana), english?)| (english, (kanji,
                                                furigana)?))
                                    -- nihongo and/or english language
                                    version of the keyword -->

<!-- %LineEle; -->
<!ELEMENT l                - O ((%txtlmts;)*)
                                    -- line of text -->
<!ATTLIST l        part        NAME        #FIXED        all
                                    -- this text represents a
                                    whole line of text -->

<!-- %ListsEle; -->
<!ELEMENT ld        - - ((term,def)+)        -- definition list -- >
<!ELEMENT ldp        - - ((term,def)+)        -- definition list in paragraphs -- >

<!ELEMENT term        - O ((%txtlmts;)*)        -- term  in a definition list -- >
<!ELEMENT def        - O ((%copy;)*)        -- definition of a term -- >

<!ELEMENT la        - - (li+)        -- alphabetic list -- >
<!ELEMENT lm        - - (li+)        -- marked list -- >
<!ELEMENT ln        - - (li+)        -- numbered list -- >
<!ELEMENT lu        - - (li+)        -- unmarked list -- >
<!ELEMENT li        - O ((%copy;)*)                -- list item -- >

<!-- %NameEngEle; -> Name -->
<!ELEMENT kanji        - O        ((%text;)+)
                                    -- the kanji (or kana) version of a
                                    name or other text-->
<!ELEMENT furigana        - O        ((%text;)+)
                                    -- the furigana reading of a name -->
<!ELEMENT english        - O        ((%text;)+)
                                    -- the english or romanji version of a
                                    name -->

<!-- %ParagraphEle; -->
<!ELEMENT p                O O (((%txtlmts;)|l)*)
                                    -- paragraph -->
<!ELEMENT xmp        - - (((%txtlmts;)|l)*)
                                    -- example paragraph -->
<!ATTLIST xmp        width        (column|page)        column        >

<!-- %StructureSecSs1TEle; -->
<!ELEMENT body        O O        (t?, (%copy;)*, sec*)
                                    -- body of article comprises
                                    optional title, optional
                                    paragraph-level ELEMENTs
                                    and optional sections -->
<!ELEMENT sec        - O        (t, (%copy;)*, ss1*)
                                    -- section comprises
                                    title, optional paragraph-level
                                    ELEMENTs and optional
                                    subsections -->
<!ATTLIST sec        id        ID        #IMPLIED
                                    -- ID for cross-referencesencing -->
```

```
<!ELEMENT ss1        - O        (t, (%copy;)*, ss2*)
                                -- subsection comprises
                                title, optional paragraph-level
                                ELEMENTs and optional
                                sub-subsections -->
<!ATTLIST ss1        id        ID        #IMPLIED
                                -- ID for cross-referencesencing -->
<!ELEMENT ss2        - O        (t, (%copy;)*, ss3*)
                                -- sub-subsection comprises
                                title, optional paragraph-level
                                ELEMENTs and optional
                                sub-sub-subsections -->
<!ATTLIST ss2        id        ID        #IMPLIED
                                -- ID for cross-referencesencing -->
<!ELEMENT ss3        - O        (t, (%copy;)*)
                                -- sub-sub-subsection comprises
                                title and only paragraph-level
                                ELEMENTs -->
<!ATTLIST ss3        id        ID        #IMPLIED
                                -- ID for cross-referencesencing -->

<!-- %TextEle; -->
<!--ELEMENT xref        - O ((%text;)*)                cross referencesence -->
<!ELEMENT xref        - O EMPTY                -- cross referencesence -->
<!ATTLIST xref         refid        NMTOKEN        #REQUIRED --#CONREF --
                        type        (number|name|page|external)        page        >
<!ELEMENT e1        - - ((%text;)*)        -(%emph;)
                                -- italic japanese emphasis -->
<!ELEMENT e2        - - ((%text;)*)        -(%emph;)
                                -- bold japanese emphasis -->
<!ELEMENT sc        - - ((%text;)*) -(%emph;)
                                -- small capitals text -->
<!ELEMENT sbs        - - ((%text;)*)
                                -- subscript text -->
<!ELEMENT sps        - - ((%text;)*)
                                -- superscript text -->
<!ELEMENT ob        - - ((%text;)*)        -(ob)
                                -- overbar -->
<!ELEMENT ub        - - ((%text;)*)        -(ub)
                                -- continuous underbar -->

<!ELEMENT fn        - -        ((%copy;)*) -(note | fn)
                                -- foot note -->
<!ELEMENT note        - O        ((%copy;)*) -(note | fn)
                                        -- 個々の注記 -->
<!ATTLIST note        id        NMTOKEN        #IMPLIED --#CONREF        -->

<!-- %TitleEle; -->
<!ELEMENT t                - O ((%txtlmts;)*, subt?)
                                -- title used by (nearly) everything -->
<!ELEMENT subt        - O ((%txtlmts;)*)
                                -- title subtitle -->

    <!-- %RefEle; -->
<!ELEMENT ref        - -        ((ref|book|art|artbk|artconf|%copy;)*)
                                -- referencesence to a note in
                                the <referencesences> section -->
<!ATTLIST ref        id        NMTOKEN        #IMPLIED
                                -- id of the referencesence
                                being referencesred to -->
```

```
<!ELEMENT book        - -        (auth?, ptitl, ed?, (place? & pub?), year?, page?, obi?)
                                        -- referencesence to a book -->
    <!ELEMENT art        - -        (auth?, t?, ptitl, (place? & vol? & page? & year?), obi?)
                                        -- referencesence to an article -->
    <!ELEMENT auth        - -        ((%text;)*)
                                        -- authors' name in a
                                        referencesence -->
    <!ELEMENT vol        - -        (#PCDATA)
                                        -- volume number in a referencesence
                                        -->
    <!ELEMENT page        - -        (#PCDATA)
                                        -- starting page number in a
                                        referencesence -->
    <!ELEMENT ed        - -        ((%text;)*)
                                        -- edition number in a book
                                        referencesence -->
    <!ELEMENT place        - -        ((%text;)*)
                                        -- place a book was published
                                        -->
    <!ELEMENT pub        - -        ((%text;)*)
                                        -- publisher's name in a book
                                        referencesence -->
    <!ELEMENT citref        - 0        EMPTY
                                        -- referencesence to a referencesence -->
    <!ATTLIST citref        refid        NMTOKEN        #REQUIRED --#CONREF --
                                        -- id of referencesence to references to --
                        type        (number|page)        number
                                        -- show page or number of
                                        referencesence -->
    <!ELEMENT artbk        - -        (auth?, t?, ptitl, ed?, editor?, (place? & pub?),
                                                    year?, page?, obi?)
                                            -- 書籍中の論文の出版に関する詳細情報 -->
    <!ELEMENT ptitl        - -        ((%txtlmts;)*, subt?)
                                            -- 書籍あるいは雑誌名のタイトル -->
    <!ELEMENT editor        - -        ((%text;)*)
                                            -- 編集者名 -->
    <!ELEMENT obi        - -        ((%txtlmts;)*)
                                            -- その他の関連書誌情報 -->
    <!ELEMENT artconf        - -        (auth?, t?, ptitl, editor?, (hplace? & hdate? &
                                hbody?), (place? & pub?), year?, page?, obi?)
                                            -- 会議報告：図書中の１論文の詳細情報 -->
    <!ELEMENT hplace        - -        ((%text;)*)
                                            -- 会議開催地 -->
    <!ELEMENT hdate        - -        ((%text;)*)
                                            -- 会議開催年月 -->
    <!ELEMENT hbody        - -        ((%text;)*)
                                            -- 会議開催機関 -->

        <!-- %TableEle; -->
    <!ELEMENT tb        - - (t?, ((head?, row+) | fgart+), caption?)
                                        -- table -->
    <!ATTLIST tb        id        ID        #IMPLIED
                                        -- id for cross-referencesencing --
                        edge (LEFT & RIGHT & TOP & BOTTOM)        #IMPLIED
                                        -- edges that are on --
                        numbered (YES|NO)        YES
                                        -- numbering attribute -->
    <!ELEMENT head        - - (row+)        -- headrows -->
```

```
    <!ELEMENT row        - O (c+)        -- row -->
    <!ELEMENT c                - O ((%copy;)*)        -- cell -->
    <!ATTLIST c                colwidth        NUTOKEN        1
                                        -- straddle width --
                           rowdepth        NUTOKEN        1
                                        -- straddle depth --
                        edge                NMTOKENS        #IMPLIED
                                        -- edges that are on -->
    <!ELEMENT caption        - O ((%copy;)*)
                                        -- table caption -->
<!--]-->
<!-- end of template.dtd -->
]>
```

付録 B

学情紀要第 7 号から、実験入力データ

学情紀要第 7 号より、実験入力データ

```
<article journal="bunascis" num="7" page="13" page2="27" date="19950331" txtl="jpn">
<front>
<atl>
<nihongo.title>
インターネットに適応した全文データベース検索システムの構成
</nihongo.title>
<english.title>
Configuration of a Full Text Database Retrieval System for Internet Environment
</english.title>
</atl>
<runt>
インターネットに適応した全文データベース検索システムの構成
</runt>
<author>
<g.name>
<kanji>
敬三
</kanji>
<furigana>
ケイゾウ
</furigana>
<english>
Keizo
</english>
</g.name>
<f.name>
<kanji>
大山
</kanji>
<furigana>
オオヤマ
</furigana>
<english>
OYAMA
</english>
</f.name>
<position>
<kanji>
学術情報センター
</kanji>
<furigana>
ガクジュツ ジョウホウ センター
</furigana>
<english>
National Center for Science Information Systems
</english>
</position>
</author>
<abstract>
<abstract.nihongo>
<p>
本論文では現在のインターネット上での情報サービス環境を最大限に利用しつつ、高度で使用しやすい全文データベース検索サービスを提供できるシステムの構成方法を述べている。WWW のクライアントである Mosaic とサーバである HTTPD を用い、既存の高度な検索エンジンとの間を独自のゲートウェイにより結合する。階層構造を持つ文書データを対象に柔軟で効率的な検索を可能とするインタフェースを実現している。
</p>
</abstract.nihongo>
```

```
<abstract.english>
<p>
This paper describes a configuration method of a system which can provide full text
database retrieval services. The system is easy to use and has advanced features, while
utilizing the information services environment currently available on the Internet at
most. It adopts a client software Mosaic and a server software HTTPD for the WWW service
as a basis, and combines an existing advanced text retrieval engine with them through an
original gateway. It realizes a user interface through which users can access to document
data with hierarchical structures flexibly and efficiently.
</p>
</abstract.english>
</abstract>
<keywords>
<keyword>
<kanji>
全文データベース
</kanji>
<furigana>
ゼンブン データベース
</furigana>
<english>
full text database
</english>
</keyword>
<keyword>
<kanji>
情報検索
</kanji>
<furigana>
ジョウホウ ケンサク
</furigana>
<english>
information retrieval
</english>
</keyword>
<keyword>
<kanji>
文書構造
</kanji>
<furigana>
ブンショ コウゾウ
</furigana>
<english>
document structure
</english>
</keyword>
<keyword>
<kanji>
インターネット
</kanji>
<furigana>
インターネット
</furigana>
<english>
Internet
</english>
</keyword>
```

<keyword>
<kanji>
ゲートウェイ
</kanji>
<furigana>
ゲートウェイ
</furigana>
<english>
gateway
</english>
</keyword>
<keyword>
<kanji>
グラフィカルユーザインタフェース
</kanji>
<furigana>
グラフィカル ユーザ インターフェース
</furigana>
<english>
graphical user interface
</english>
</keyword>
</keywords>
</front>
<body>
<sec>
<t>
1. はじめに
</t>

インターネットの整備とワークステーションの普及に伴い、インターネットを通じて様々な情報サービスが提供されており、その数は急激に増えつつある。これらの多くは情報収集・提供のための組織的な支援体制を伴わないため、内容やサービスの質に問題を抱えている例も多く、情報を探索し利用する立場からは必ずしも望ましくない状況も発生している。このため、インターネットを通じた情報サービスに批判的な意見も散見される。

しかし、一方で、従来から独自のネットワークや方式で提供してきた情報サービスを、インターネット経由でも利用可能にし、さらにはインターネットをサービスの中心に据える例も少なくない。これらの多くは従来からの膨大な蓄積を、より効率的に、より利用しやすく、できるだけ多くのユーザに提供しようとするものである。このような努力は、従来からのユーザにとってのみでなく、潜在的なユーザにとっても歓迎すべきものと言えるであろう。

筆者が所属する学術情報センターでも、従来からいわゆる情報検索サービスを提供している
<ref id="r07r001">
<book>
<ptitl>
「情報検索サービス NACSIS-IR」
</ptitl>
<obi>
http://www.nacsis.ac.jp/ir/ir-j.html
</obi>
</book>
</ref>
が、これまであまりインターネットの機能を有効に利用できないでいた。その原因の一つとしては、サービス提供のためのホストシステムがいわゆるメインフレームコンピュータであり、インターネットで標準的に利用されている環境が適用できなかったことがあげられる。また、センターのユーザの多くが自由にインターネットを使える状態になかったためにセンターが積極的にインターネット対応を推進する動機を欠いたことも一つの理由としてあげることができよう。

しかし、ここ数年間で大学や研究機関における状況は大きく変わり、希望するものは誰でもインターネットを利用できるようになった。このような状況の変化に対し、情報サービスの提供機関は学術情報センターに限らず、ユーザの利便性を第一に考え、できるだけ迅速に適応していくことがきわめて重要であると筆者は考えている。本論文では、このような目的を満たすために筆者が中心になって研究開発を進めているインターネット上の情報サービスシステムの構成方法について述べる。

</sec>
<sec>
<t>
2. 情報サービスのツールの現状と利用方針
</t>

筆者はインターネットを介したサービスを、既存のサービスの代替ではなく、今後のサービスの中心的役割を担うものと考えている。そのため、既存のサービスが備えている機能の見直しを含め、新しい環境に適応した高度な機能を実現することが重要である。一方で、インターネットはさまざまなボランティアが提供してきた多くのパブリックドメインのソフトウェアと、それを元にして開発された商用の製品群によって大きく発展してきた。このような発展形態は今後も継続するものと考えられる。

本研究においては、インターネット上で一般的に利用されているツールを有効に組み合わせながら必要な機能を実現していくことを基本的な考え方とする。特にユーザ環境としては、広く利用可能な一般的なツールを用いてサービスを利用できることを最重要の課題と考えている。そこで、まず本章では、インターネット上の情報サービスのためのツールの現状を整理し、本システムでの利用方針を検討する。

インターネット上の情報サーバとしては archie、WWW (World Wide Web)、gopher、WAIS 等が一般的である
<ref id="r07r002">
<art>
<auth>
斎藤正史, 山口英
</auth>
<t>
「インターネットの情報サービス」
</t>
<ptitl>
情報処理
</ptitl>
<vol>
Vol.34, No.12
</vol>
<page>
pp.1415-1421
</page>
<year>
1993
</year>
</art>
</ref>
。これらはいずれもパブリックドメインのソフトウェアとし提供されており、さらに WWW と WAIS については機能や性能を強化した商用版も提供されている。これらはそれぞれ目的を異にし、実現可能なサービスにもそれぞれ特徴がある。全文データベース検索機能の実現という観点からこれらを整理すると以下のようになる。

<ln>

archie は FTP で提供されているファイル名に対する文字列検索のみを提供する

```
<ref id="r07r003">
<book>
<ptitl>
``Archie Reference Manual Pages"
</ptitl>
</book>
</ref>
。内容に対するサーチ機能はなく、情報へのアクセスも別途FTPを用いて行う必要がある。
</p>
</li>
<li>
<p>
WWWはネットワーク上に分散した情報資源をハイパーテキストの概念に基づいて利用可能にするもので、HTMLという形式で文書間に予め張られたリンクをたどることによってアクセスを行う
<ref id="r07r004">
<book>
<ptitl>
``A Beginner's Guide to HTML"
</ptitl>
<obi>
http://www.ncsa.uiuc.edu/demoweb/html-primer.html
</obi>
</book>
</ref>
。他の情報資源に対するリンクの設定も可能であるが、WWWサーバ自体には内容に対する検索機能はない。
</p>
</li>
<li>
<p>
gopherはネットワーク上に分散した情報資源をメニュー形式で利用可能にするもので、予め階層構造に分類整理されたメニューをたどることによってアクセスを行う
<ref id="r07r005">
<book>
<ptitl>
``Gopher Reference Manual Pages"
</ptitl>
</book>
</ref>
。情報資源の一つの形態として内容に対する簡単な文字列サーチ機能を持つが、情報検索機能としてはきわめて不十分である。
</p>
</li>
<li>
<p>
WAISはサーバ中に格納されている文書に対し、文書中に現れる自由語を用いて頻度や類似度に応じた情報検索機能を提供する
<ref id="r07r006">
<p>
``WAIS 2.0 Technical Documentation", http://www.wais.com/newhomepages/techtalk.html
</p>
</ref>
。情報検索の知識がないユーザでも比較的容易に使用できるが、履歴や集合演算の機能がないため、複雑な検索を試行錯誤的に行うには不十分である。
</p>
</li>
</ln>
<p>
また、これらの情報サービスを利用するためのユーザインタフェースとなるクライアントには、GUIを用いて統合的に対話が可能なMosaicがパブリックドメインソフトウェアとして普及しており
```

```
<ref id="r07r007">
<p>
``NCSA Mosaic Home Page",
</p>
<p>
http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html
</p>
</ref>
、商用版も各種のものが利用可能となってきている
<ref id="r07r008">
<art>
<t>
「社内情報共有に新時代ーインターネットの WWW が急浮上」
</t>
<ptitl>
日経コミュニケーション
</ptitl>
<page>
pp.76-97
</page>
<year>
1995.1.6
</year>
</art>
</ref>
(以下、これらを総称して Mosaic と呼ぶ)。Mosaic は元々WWW のビューワとして開発されたが、現在では WWW サーバ (HTTP プロトコルによる) ばかりでなく、gopher サーバや WAIS サーバ (Z39.50 プロトコルによる) へのアクセスが可能であり、また、HTTP プロトコルによりゲートウェイを介して archie サーバなど、他の情報サーバへもアクセス可能である。gopher や WAIS には専用のクライアントもあるが、普及の度合いや使いやすさを考慮すると、Mosaic をクライアントとして利用するのがユーザにとって最も有効であろう。
</p>
<p>
一方、全文データベースの検索には、単に自由語による内容のサーチを可能とするだけでなく、文書の階層構造に即した検索と取り出しが可能であることが必須である。既存の情報検索システムの中では、カナダ OpenText 社製の Pat
<ref id="r07r009">
<book>
<ptitl>
``Pat Reference Manual"
</ptitl>
<pub>
OpenText Corp
</pub>
</book>
</ref>
が機能的に必要条件をもっとも良く満たしている。また、今後、より高度なシステムが利用可能になることも期待される。本研究の目的は Mosaic を通じてそれらを適時に利用可能にするための方式を確立することにある。
</p>
<p>
Mosaic から利用可能な標準的なプロトコルとしては Z39.50 プロトコルのサブセットである WAIS プロトコルと、WWW で用いられている HTTP プロトコルがある。Z39.50 は情報検索用の汎用の標準通信プロトコルであり、セッションの保持機能があって比較的高度な検索が可能であるが、WAIS プロトコルはこのセッション保持機能を省略している
<ref id="r07r010">
<book>
<ptitl>
``WAIS(tm) Network Publishing using Z39.50"
</ptitl>
```

```
<obi>
http://www.wais.com/newhomepages/z3950.html
</obi>
</book>
</ref>
。また、Z39.50 では検索対象のデータの階層構造を利用した検索・表示要求に対応できない。
</p>
<p>
HTTP は HTML 文書などのハイパーテキストを転送するためのプロトコルであるが、HTML の FORM の機能を
用いるとユーザに提示する検索用の GUI を送ることができる
<ref id="r07r011">
<book>
<ptitl>
``The Common Gateway Interface"
</ptitl>
<obi>
http://hoohoo.ncsa.uiuc.edu/cgi/overview.html
</obi>
</book>
</ref>
。HTML ではセマンティクスは何も規定しないので、FORM を設計するときにユーザに容易に理解可能な
ように考慮することが重要である。FORM に入力されたデータはゲートウェイを通じて情報サーバに届
けられ、情報サーバからの出力を FORM にフィードバックすることも可能であるので、Mosaic を通じた
ユーザとの対話処理用に適している。
</p>
<p>
HTTP 自体はステートレスでありセッションの保持機能がないが、HTTP のゲートウェイの高度な構成方
式によりこの機能が実現できれば、Pat などの強力な全文検索エンジンと組み合わせることにより、
Mosaic を通じて試行錯誤を含む高度な対話処理を可能とする全文データベース検索システムを構成す
ることができる。
</p>
</sec>
<sec>
<t>
3. 全文データベース検索システムの要件
</t>
<p>
本システムでは、検索の対象を、学術文献 (数ページ) から技術マニュアル (数千ページ) 程度の、階層
構造を持つ文書の全文データとする。必ずしも必要な条件ではないが、SGML(Standard Generalized
Markup Language) で記述されているものと仮定する。データベースにはこのような相互に比較的独立
な全文データが多数格納されている。
</p>
<p>
本章では、このような環境において全文データベース検索システムが扱うべき文書データの構造を分
析し、多数の文書の中から所望の文書を探し出すためにユーザが必要とする機能を検討することにより
、システムに要求される要件を明らかにする。
</p>
<ss1>
<t>
3.1 文書データの構造と検索要求
</t>
<p>
本システムが対象とする文書データは一般に図
<xref refid="r07f001"></xref>
のような階層構造を持つ。この他にも参考文献とその参照のようなリンク構造もあるが、多数の文書
データに対して一括して検索する場合にこれが有効な手がかりとなることはあまり無いと考えられる
ため、検索対象は階層構造のみとする。
</p>
<fg id="r07f001">
<t>
文書の階層構造
</t>
```

```
<fgart file="r07f001.tif">
</fgart>
</fg>
<p>
SGML で階層構造定義に関連して定義可能なものには、出現順などを指定する結合子と、繰り返しなど
を指定する出現指示子がある
<ref id="r07r012">
<book>
<auth>
Goldfarb, Charles F.
</auth>
<ptitl>
''The SGML Handbook"
</ptitl>
<pub>
Clarendon Press
</pub>
<year>
1990
</year>
<page>
663pp
</page>
</book>
</ref>
。これらの中で、出現順などは多数の文書の中からキーワードを頼りに検索する環境ではあまり有効
な利用方法は考えられない。一方、繰り返しは適切に取り扱えないと精度の高い検索が期待できない。
</p>
<p>
ユーザが必要とするであろうと考えられる検索条件の例としては、以下のようなものが考えられる。
</p>
<ln>
<li>
<p>
節タイトルに「実験方法」が含まれ、節の本文中に「真空蒸着」が含まれるような節を持つ論文を表
示する。
</p>
</li>
<li>
<p>
同一の段落に「バブル経済」と「経済摩擦」が含まれるような論文の書誌情報を表示する。
</p>
</li>
<li>
<p>
同一の段落に「バブル経済」と「経済摩擦」が含まれるような論文について、書誌情報と章・節のタ
イトル、およびその段落を、出現順に表示する。
</p>
</li>
</ln>
<p>
これらのうち、(1)、(2) は検索対象と表示対象の関係が比較的単純であるため条件の記述も容易であ
るが、(3) はやや複雑となるため Mosaic を通じてユーザが条件の指定をできるような簡潔な HTML の
FORM を作成することは困難である。本システムでは、検索対象と表示対象の関係が単純な包含関係に
ある場合のみを想定する。
</p>
</ss1>
<ss1>
<t>
3.2 システムの要件
</t>
```

情報検索システムの一般的な機能には大きく分けてサーチ機能と表示機能がある。全文データベース検索では通常の書誌型、索引型、あるいは抄録型の情報検索と比較して、以下のような機能が特に重要である。

<ss2>
<t>
(1) サーチ機能
</t>
<ld>
<term>
(a) 文字列サーチ
</term>
<def>

ユーザが利用目的に応じてさまざまな角度から検索できるよう、テキスト中に出現する任意文字列でのサーチ、範囲検索、近接演算 (順序指定、順序非指定) などが可能でなければならない。特に日本語の学術情報では、辞書にない単語や複合語でのサーチができることも重要である。

</def>
<term>
(b) 論理演算
</term>
<def>

検索対象が階層構造を持っているため、階層に即した論理演算が必要であり、各階層における任意の文書要素および一致文字列に対して、論理積、論理和、論理差、否定などの論理演算が可能でなければならない。

</def>
<term>
(c) 集合演算
</term>
<def>

ユーザにとって、複雑な検索を最初から単一の論理式で記述することは困難であり、試行錯誤を繰り返す上でも過去の検索結果集合を組み合わせた集合演算が可能であることが重要である。論理演算同様、文書要素集合および一致文字列集合に対する共通集合、合併集合、排他集合、補集合などの集合演算が可能でなければならない。

</def>
<term>
(d) 階層演算
</term>
<def>

階層構造中の任意の文書要素を基点にしてサーチを行い、さらに異なる階層の文書要素に基点を移してサーチを行うためには、階層構造にある文書要素・文書要素間および文書要素・一致文字列間の包含と包摂の演算が、論理演算および集合演算の双方において可能でなければならない。

</def>
</ld>
</ss2>
<ss2>
<t>
(2) 表示機能
</t>
<ld>
<term>
(a) 出現位置の周辺表示
</term>

```
<def>
<p>
文中にある任意の文字列に対するキーワード検索を行う場合、その語の使われ方をユーザが予めすべて予想することは困難な場合が多いため、キーワードに一致した文字列がどのような文脈で現れたかを簡略に表示する、いわゆる KWIC 表示が有効である。
</p>
</def>
<term>
(b) 文書要素を指定した表示
</term>
<def>
<p>
キーワードによるサーチやその結果の組み合わせにより得られた文書が適切であるかどうか判断し、また、それらの中から有用と思われるものを選択するためには、そのために必要かつ十分な部分を一覧できるような表示機能が必要である。たとえば文献情報部分のみを一覧表示させたり、文書ごとの目次(文書タイトルと章節の見出しなど)を表示させたり、キーワードが見つかった段落を表示させたりできることが要求される。また、マニュアルなどでは該当する節を表示させるだけでユーザの目的を達せられることもある。
</p>
</def>
<term>
(c) 全文を簡易整形して表示する機能
</term>
<def>
<p>
選択した文書を通覧するためには、ある程度読みやすい形でオンライン表示できる必要がある。最終的には SGML で記述された原データを加工して厳密な整形出力をオフライン印刷できればよいであろうが、ユーザの利用性を高めるためには対話処理のなかで簡易整形表示できる機能が重要である。
</p>
</def>
</ld>
</ss2>
<ss2>
<t>
</t>
<p>
これらの機能のうち、サーチ機能はサーバで実現する必要がある。表示機能はサーバ、クライアントのいずれでも実現可能であるが、本システムでクライアントに想定している Mosaic にはこれらを実現する機能がないため、サーバで実現する必要がある。また、仮にクライアントがこれらを実現する機能をもっていたとしても、通信の負荷を考慮すると多数ユーザの環境では現実的ではない。
</p>
<p>
全文データベース検索では特に文書がもつ階層構造を適切に処理する必要があるため、上記のような複雑な機能を持つシステムが必要であるが、一般のユーザがこのような機能を理解し、対象とする文書の具体的な構造を覚えて検索に有効に利用するのは極めて難しい。このため、メニューなどを用いることにより特別の知識を持たないユーザでも上記のような高度な機能を容易に利用できるようなユーザインタフェースを実現することが重要である。
</p>
</ss2>
</ss1>
</sec>
<sec>
<t>
4. システム構成
</t>
<p>
本章では、上記のような諸々の条件を満たすために採用した全体構成とその動作の概要を述べ、本システムの開発の中心となるフロントエンドについて詳述する。
</p>
<ss1>
<t>
4.1 全体構成
</t>
```

```
<p>
本システムでは前述の通り、クライアントとしてMosaicを利用し、WWWサーバを基礎として全文デー
タベース検索サーバを構成する。全体構成を図
<xref refid="r07f002"></xref>
(a)に示す。各部分の機能と相互間の関係を簡単に述べる。
</p>
<fg id="r07f002">
<t>
システムの概要
</t>
<fgart file="r07f002.tif">
</fgart>
</fg>
<ss2>
<t>
(1)HTTPクライアント
</t>
<p>
HTTPクライアントはユーザがシステムと対話するための直接のインタフェースを提供する。ユーザが
インターネット上の情報サーバにアクセスするときに随時起動する。現在利用可能なものにはMosaic
以外にもいくつかの種類が存在し、フォームを扱う機能を持つものであれば本システムのクライアン
トとして利用可能である。しかし、ユーザインタフェースの質や利用可能なプラットホームの多様性
から、現在ではMosaicが最も利用しやすい。
</p>
</ss2>
<ss2>
<t>
(2)WWWサーバ
</t>
<p>
WWWサーバはクライアントからのアクセスに対して要求された情報を提供する。通常は予め用意され
たHTML文書や画像ファイルを送信するだけであるが、本システムではフロントエンドで動的に作成さ
れるHTML文書をクライアントに送信する。全文データベース検索サーバの起動時に1回だけ起動され、
システム停止まで存続する。クライアントとはHTTPプロトコルによりTCP/IPを介して通信する。ゲー
トウェイ側のインタフェースはCGI(Common Gateway Interface)として規格が定められている。WWWサー
バにはパブリックドメインや市販製品として利用可能な既製のものがいくつか存在するが、今回は
CERN版のHTTPDを用いることとした。
</p>
</ss2>
<ss2>
<t>
(3)ゲートウェイ
</t>
<p>
ゲートウェイはWWWサーバとフロントエンドの間のデータの仲介を行う。フォームが送られてくるご
とにWWWサーバから起動され、標準入出力を通してWWWサーバと情報の授受を行う。フロントエンドと
の通信は各フロントエンドと1対1に対応したFIFOファイルを通じて行う。結果の情報をWWWサーバ
に返すと消滅する。頻繁に生成と消滅を繰り返すのでシステムの性能を律する可能性があるため、可
能な限り機能を簡素化してある。
</p>
</ss2>
<ss2>
<t>
(4)フロントエンド
</t>
<p>
フロントエンドはユーザの要求を全文検索エンジンのコマンドに解釈して実行させるとともに、結果
やデータの表示に必要なHTMLのフォームや文書を作成する。セッションの維持や検索の履歴の管理に
必要な処理も行う。個々のセッションに対応して1つずつ動的に生成され、セッションが終了すると
消滅する。全文検索エンジンとはパイプを通じて通信する。既存の構成要素で提供されていない機能
の大部分はここで実現され、本システムのプログラム開発の大部分が含まれている。
</p>
```

```
</ss2>
<ss2>
<t>
(5) 全文検索エンジン
</t>
<p>
全文検索エンジンには、各種のパブリックドメインソフトウェアや市販製品を比較検討した結果、カ
ナダ OpenText 社製の Pat サーバを採用することとした。ただしこれは全文データベースに対する検索
機能を評価した結果の選択であって、システムの構成上からの必要性によるものではない。従って、
今後、より適当なものが利用可能になればそれを採用することも可能である。全文検索エンジンはフ
ロントエンドにより管理されており、セッションが開設されるたびに起動され、セッションが終了す
ると消滅する。
</p>
</ss2>
</ss1>
<ss1>
<t>
4.2 動作の概要
</t>
<p>
全体の動作の概要を図
<xref refid="r07f002"></xref>
(b) に示す。ユーザはまず HTTP クライアントである Mosaic を起動し、ログインフォームを WWW サーバで
ある HTTPD から取り出す。フォームにユーザ ID などの必要な情報を記入し、使用するデータベースを
指定して HTTPD に送る (ログイン要求)。HTTPD はゲートウェイプロセス (GWP) を起動してフォームに記
入されたデータを渡す。GWP はセッションの環境を作成してからフロントエンドプロセス (FEP) を生成
する。FEP は全文検索エンジン (Pat) を起動し初期化してから、検索フォームを GWP、HTTPD を経由して
転送して Mosaic に表示させる。
</p>
<p>
ユーザは、検索フォームに検索条件を記入して HTTPD に送り返す (検索要求)。HTTPD は新たな GWP を起
動してフォームに記入された検索条件を渡す。GWP は対応する FEP を見つけて検索条件を渡す。FEP は
これを Pat のコマンドに解釈して Pat に渡す。Pat から結果を受け取ると FEP は検索フォームに検索集合
の一覧と内容の簡略表示を埋め込んで結果フォームを作成し、GWP、HTTPD を経由して Mosaic に送り返
し、表示させる。
</p>
<p>
結果フォームに簡略表示されたもののなかで詳細な表示をさせたいものがある場合は、ユーザは文書
を選択して HTTPD に送り返す (表示要求)。HTTPD は GWP を起動して要求を FEP に渡し、FEP はそれをコマ
ンドに解釈して Pat に処理を行わせる。取り出した文書データは FEP が HTML 形式に簡易整形して GWP、
HTTPD を通じて Mosaic に転送する。
</p>
<p>
一連の検索を終えると、ユーザは結果フォーム中に終了の指示を記入して HTTPD に送り返す (ログアウ
ト要求)。GWP を通じてログアウト要求を受け取った FEP は、Pat を終了させ利用統計などを記した終了
通知の HTML 文書を作成し、これを GWP、HTTPD を経由して Mosaic に転送した後に終了する。
</p>
</ss1>
<ss1>
<t>
4.3 フロントエンドの構成
</t>
<p>
フロントエンドは本システムの開発の中心であり、機能的にも最も複雑である。機能には大きく分け
て、セッション管理機能、コマンド解釈機能、フォーム表示機能、文書表示機能がある。以下でこれ
らについて説明する。
</p>
<ss2>
<t>
(1) セッション管理機能
</t>
```

セッションの開始時には、ログインフォームに記入された情報からセッション ID を生成し、指定されたデータベースに対して Pat を起動するとともに、以後のセッションの中で GWP からのアクセスに用いる FIFO ポートを作成してからフォーム表示機能を呼び出す。セッション ID はユーザ名、クライアントホスト名、サーバホスト名、FEP のプロセス ID、セッション開始時刻などから容易に想像できないように生成する。課金などのためにユーザ管理が必要な場合は、上記の処理を開始する前に、ログインフォームに記入されたユーザ ID やパスワード、および HTTPD から渡されるクライアントのホスト ID などを用いてユーザ認証を行う。

セッションの終了処理はユーザからのセッション終了要求を受けて行う。Pat を終了させ、利用統計などを記入した終了通知を作成して返す。ユーザ管理が必要な場合は各種統計情報をファイルに書き出す。使用済みの FIFO ポートを削除してプロセスを終了する。

HTTP にはセッションの概念がないため、クライアントとサーバは 1 回のアクセスごとに接続と切断を繰り返す。従って、クライアントがログアウト要求を送らないまま終了してしまうと、サーバはこれを検知できない。このため、FEP はセッションの保留時間を監視し、ユーザから要求がない状態が一定時間継続した場合は中断処理を行う。現在は中断処理では強制的に終了処理を実行しているが、サービスの内容に応じは、後で回復可能なように中断時の FEP と Pat の状態を保存してセッションを解放するというような機能の導入も考えられる。

</ss2>
<ss2>
<t>
(2) コマンド解釈機能
</t>

検索フォームに埋め込まれた検索要求を取り出し、Pat のコマンドに解釈して処理を実行させる。検索要求にはサーチ要求と表示要求がある。サーチ要求に対しては検索条件を Pat のサーチコマンドに翻訳して実行させる。要求内容と Pat が作成した検索集合を履歴として保持し、実行結果をフォーム表示機能に渡す。表示要求に対しては、選択された文書の中から指定された文書要素を取り出すための Pat のコマンドに翻訳して実行させる。取り出した文書データは文書表示機能に渡す。

</ss2>
<ss2>
<t>
(3) フォーム表示機能
</t>

あらかじめ対象データベースごとに用意された検索フォームのテンプレートに、サーチの履歴と検索集合の内容の簡略表示を埋め込んで HTTPD に渡す。履歴には最近の指定回数分の検索集合の番号、その件数、およびサーチ要求の内容を表示する。簡略表示には最新の検索集合に含まれる文書データの先頭部分を指定件数分だけ表示する。検索フォームのテンプレートにはサーチ文字列の入力フィールド、サーチ範囲を限定するための文書要素のメニュー、集合演算のための演算子のメニュー、階層演算のための文書要素のメニュー、表示要求で文書要素を指定するための文書要素のメニューなどが予め準備されている。履歴にはチェックボックスを付加し、集合演算のオペランドとして指定できるようにする。簡略表示にもチェックボックスを付加し、表示要求の文書指定をできるようにする。

</ss2>
<ss2>
<t>
(4) 文書表示機能
</t>

表示モードとして整形出力とソース出力を持つ。整形出力モードでは与えられた SGML 文書を HTML 形式に変換してクライアントに送る。変換のためのフィルタはデータベースごとに独立したプログラムとして用意しておく。ソース出力モードでは元の SGML 文書をそのままプレインテキストとしてクライアントに送る。

</ss2>
</ss1>
</sec>

```
<sec>
<t>
5. ユーザインタフェース
</t>
<p>
階層構造を持つ文書を対象とする全文データベース検索システムのユーザインタフェースでは、文書
の階層構造の処理を含む検索要求を入力できる必要があり、指定すべきパラメータの種類が多い。ユー
ザがこれらをすべて記憶し、適切に誤りなく直接入力することはきわめて困難である。そこで、本シ
ステムではこれらのパラメータをユーザに提示し、ユーザはそれらの中から選択してゆくだけで検索
要求を入力できるようなユーザインタフェースを実現する。HTML のフォームで Mosaic に表示できるグ
ラフィカルユーザインタフェースの構成要素には限界があり、ユーザの操作に対応したきめ細かい動
作はできないが、基本的な検索の対話をするための要求は満たせると考えている。
</p>
<p>
図
<xref refid="r07f003"></xref>
、
<xref refid="r07f04a"></xref>
、
<xref refid="r07f04b"></xref>
および図
<xref refid="r07f005"></xref>
にユーザインタフェースの例を示す。図
<xref refid="r07f003"></xref>
はログインフォームの表示である。ユーザはユーザ ID、パスワードを入力し、データベースの一覧の
中から 1 つを選択する。図
<xref refid="r07f04a"></xref>
(a)、図
<xref refid="r07f04b"></xref>
(b) は検索フォームの表示例である。サーチ要求の記入部分、履歴表示部分、簡略表示部分から構成
されている。ユーザが直接入力するのはサーチ文字列だけである。図
<xref refid="r07f005"></xref>
は簡易整形した文書表示の例である。
</p>
<fg id="r07f003">
<t>
ログインフォーム
</t>
<fgart file="r07f003.tif">
</fgart>
</fg>
<fg id="r07f04a">
<t>
検索フォーム (その 1)
</t>
<fgart file="r07f04a.tif">
</fgart>
</fg>
<fg id="r07f04b">
<t>
検索フォーム (その 2)
</t>
<fgart file="r07f04b.tif">
</fgart>
</fg>
<fg id="r07f005">
<t>
詳細表示
</t>
<fgart file="r07f005.tif">
</fgart>
</fg>
```

```
<p>
図
<xref refid="r07f04a"></xref>
(a)、図
<xref refid="r07f04b"></xref>
(b) に示した検索フォームの操作について次に説明する。
</p>
<ld>
<term>
(1) キーワードによる文字列サーチ
</term>
<def>
<p>
検索に用いる任意の文字列をフォームの「検索文字列」フィールドに入力する。検索範囲を限定する
場合は「サーチ範囲」から対象とする文書要素を選択する。履歴表示部分に結果の集合番号、件数、
および検索内容が表示され、簡略表示部分に該当部分のリストが表示される。表示する履歴の数と簡
略表示の数はパラメータ設定部分の「履歴表示件数」と「簡略表示件数」で指定する。なお、この検
索フォームでは論理演算はサポートしておらず、単一文字列によるサーチで集合を作っておいてから
集合演算を行うことで代替する。
</p>
</def>
<term>
(2) 集合演算と階層演算
</term>
<def>
<p>
演算の種類を「演算」から選択し、オペランドを検索集合か文書要素の中から指定して「実行」する。
演算の種類には集合演算、階層演算、近接演算がある。集合演算は2つのオペランドの文書要素の種
類が一致しなければあまり意味がない。階層演算は2つのオペランドの文書要素の種類の間の上下関
係が適切でないと意味の無い結果となる。近接演算は2つのオペランドが文字列サーチの結果でない
と出現位置の間の距離を反映しなくなる。なお、距離(文字数)はパラメータ設定部分の「近接演算距
離」で指定する。
</p>
</def>
<term>
(3) 簡略表示
</term>
<def>
<p>
文字列サーチや集合・階層演算を行うと最新の検索集合の簡略表示が行われるが、以前の検索集合を
見たいときには履歴部分の表示項目をクリックする。また、表示している部分の前後を見たい場合は
上下の矢印のリンクをクリックする。
</p>
</def>
<term>
(4) 詳細表示
</term>
<def>
<p>
詳細表示には2つの方法がある。表示したい文書の簡略表示を直接クリックすると、その文書が詳細
表示される。また、簡略表示の頭にあるチェックボックスをチェックしてから「表示」を行うと複数
の文書を一括して詳細表示することができる。「全部表示」をチェックすると全ての文書を一括表示
できる。表示の形式はパラメータ設定部分の「詳細表示形式」で選択する。
</p>
</def>
<term>
(5) パラメータの設定
</term>
<def>
<p>
各種のパラメータの設定はパラメータ設定部分の項目を指定して「設定」を行う。
</p>
</def>
```

```
<term>
(6) その他
</term>
<def>
<p>
セッションの終了は「ログアウト」で行う。また、データベースの内容に関する情報と検索フォーム
の使用方法に関する情報はマニュアルのリンクをたどることで参照可能である。
</p>
</def>
</ld>
</sec>
<sec>
<t>
6. 終わりに
</t>
<p>
本論文ではインターネット上でもっとも広く用いられているユーザ環境から利用可能な全文データベー
ス検索システムの構成方法について述べた。このシステムは、データベースに収録されている文書の
構造についての特別な知識なしに GUI を通じて容易に検索できることを特長とし、本格的で先進的な
情報検索サービスの利用を可能とする。
</p>
<p>
本論文で述べた検索フォームは基本的なものであり、データベースの特性に応じて予めアクセス形態
のモデル化が可能なものについては、それに沿った検索をより効率的に行うための検索フォームの作
成を行うのがよいであろう。複数の検索フォームの中から、適切なものを選んで使用することも容易
に実現できる。
</p>
<p>
本構成方法は、より一般的な既存の情報検索サービスにも適用可能であり、利用方法が難しいために
ユーザ層が限定されていたり有効な機能が十分に活用されていなかったような分野のサービスにも、
より広く有効に利用される可能性を開くものである。
</p>
<p>
本システムは現在開発中であり、本論文で述べた内容は若干変更される可能性がある。完成後は利用
実験による評価、改良を行った後にサービスを一般に公開する予定である。また、同様のサービスを
行おうとする者のために、開発したソースコード、HTML フォームなども公開する予定である。
</p>
<p>
ただし、HTML の記述能力や Mosaic の表示能力の制約から、不満足な点が残っていることは否めない。
履歴や簡略表示がスクロールできないのもそのひとつである。これらは HTML の拡張や Mosaic の機能強
化により徐々に解決するであろうが、より専門的なユーザのためには専用のクライアントと通信プロ
トコルを用意すべきであろう。
</p>
<p>
学術情報センターでは現在、本システムの他に、文字型ユーザインタフェースの全文データベース検
索システムを開発中であり、また、電子図書館の一部として機能する全文データベース検索サーバも
開発中である。これらについては別の機会に紹介する。
</p>
</sec>
</body>
<end>
<references>
</end>
</article>
```

付録 C

HTML3.2 の DTD

HTML3.2 の DTD

```
<!--
        W3C Document Type Definition for the HyperText Markup Language
        This version is code named Wilbur, and also as "HTML 3.2".

        Draft: Tuesday August 21st 1996

        Author: Dave Raggett <dsr@w3.org>

        This is subject to change, pending final approval by the W3C
        member companies.

        HTML 3.2 aims to capture recommended practice as of early '96
        and as such to be used as a replacement for HTML 2.0 (RFC 1866).
        Widely deployed rendering attributes are included where they
        have been shown to be interoperable. SCRIPT and STYLE are
        included to smooth the introduction of client-side scripts
        and style sheets. Browsers must avoid showing the contents
        of these element Otherwise support for them is not required.
        ID, CLASS and STYLE attributes are not included in this version
        of HTML.

        The next version of HTML after Wilbur is code named Cougar and
        will add support for <OBJECT>, client-side scripting, style
        sheets, and extensions to fill-out forms.
-->

<!ENTITY % HTML.Version
        "-//W3C//DTD HTML 3.2 Draft 19960821//EN"

        -- Typical usage:

            <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Draft 19960821//EN">
            <html>
            ...
            </html>
        --
        >

<!--================== Deprecated Features Switch ==========================-->

<!ENTITY % HTML.Deprecated "INCLUDE">

<!--================== Imported Names ======================================-->

<!ENTITY % Content-Type "CDATA"
        -- meaning a MIME content type, as per RFC1521
        -->

<!ENTITY % HTTP-Method "GET | POST"
        -- as per HTTP specification
        -->

<!ENTITY % URL "CDATA"
        -- The term URL means a CDATA attribute
           whose value is a Uniform Resource Locator,
           See RFC1808 (June 95) and RFC1738 (Dec 94).
        -->
```

```
<!-- Parameter Entities -->

<!ENTITY % head.misc "SCRIPT|STYLE|META|LINK" -- repeatable head elements -->

<!ENTITY % heading "H1|H2|H3|H4|H5|H6">

<!ENTITY % list "UL | OL |  DIR | MENU">

<![ %HTML.Deprecated [
    <!ENTITY % preformatted "PRE | XMP | LISTING">
]]>

<!ENTITY % preformatted "PRE">

<!-----------------= Character mnemonic entities ==========================-->

<!ENTITY % ISOlat1 PUBLIC
       "ISO 8879-1986//ENTITIES Added Latin 1//EN//HTML">
%ISOlat1;

<!-----------------= Entities for special symbols ========================-->
<!-- &trade &shy and &cbsp are not widely deployed and so not included here -->

<!ENTITY copy   CDATA "&#169"   -- copyright sign     -->
<!ENTITY reg    CDATA "&#174"   -- registered sign    -->
<!ENTITY amp    CDATA "&#38;"   -- ampersand          -->
<!ENTITY gt     CDATA "&#62;"   -- greater than       -->
<!ENTITY lt     CDATA "&#60;"   -- less than          -->
<!ENTITY quot   CDATA "&#34;"   -- double quote       -->
<!ENTITY nbsp   CDATA " "  -- non breaking space -->

<!-----------------= Text Markup ======================================-->

<!ENTITY % font "TT | I | B  | U | STRIKE | BIG | SMALL | SUB | SUP">

<!ENTITY % phrase "EM | STRONG | DFN | CODE | SAMP | KBD | VAR | CITE">

<!ENTITY % special "A | IMG | APPLET | FONT | BASEFONT | BR | SCRIPT | MAP">

<!ENTITY % form "INPUT | SELECT | TEXTAREA">

<!ENTITY % text "#PCDATA | %font | %phrase | %special | %form">

<!ELEMENT (%font|%phrase) - - (%text)*>

<!-- there are also 16 widely known color names although
  the resulting colors are implementation dependent:

   aqua, black, blue, fuchsia, gray, green, lime, maroon,
   navy, olive, purple, red, silver, teal, white, and yellow

 These colors were originally picked as being the standard
 16 colors supported with the Windows VGA palette.
 -->

<!ELEMENT FONT - - (%text)*     -- local change to font -->
<!ATTLIST FONT
    size    CDATA   #IMPLIED    -- [+]nn e.g. size="+1", size=4 --
    color   CDATA   #IMPLIED    -- #RRGGBB in hex, e.g. red: color="#FF0000" --
    >
```

```
<!ELEMENT BASEFONT - O EMPTY    -- base font size (1 to 7)-->
<!ATTLIST BASEFONT
    size    CDATA   #IMPLIED    -- e.g. size=3 --
    >

<!ELEMENT BR    - O EMPTY    -- forced line break -->
<!ATTLIST BR
        clear (left|all|right|none) none -- control of text flow --
        >

<!--================== HTML content models ================================-->
<!--
    HTML has three basic content models:

        %text       character level elements and text strings
        %flow       block-like elements e.g. paragraphs and lists
        %bodytext   as (b) plus headers and ADDRESS
-->

<!ENTITY % block
     "P | %list | %preformatted | DL | DIV | CENTER |
      BLOCKQUOTE | FORM | ISINDEX | HR | TABLE">

<!-- %flow is used for DD and LI -->

<!ENTITY % flow "(%text | %block)*">

<!--=================== Document Body ====================================-->

<!ENTITY % body.content "(%heading | %text | %block | ADDRESS)*">

<!ENTITY % color "CDATA" -- a color specification: #HHHHHH @@ details? -->

<!ENTITY % body-color-attrs "
        bgcolor %color #IMPLIED
        text %color #IMPLIED
        link %color #IMPLIED
        vlink %color #IMPLIED
        alink %color #IMPLIED
        ">

<!ELEMENT BODY O O  %body.content>
<!ATTLIST BODY
        background %URL #IMPLIED  -- texture tile for document background --
        %body-color-attrs;  -- bgcolor, text, link, vlink, alink --
        >

<!ENTITY % address.content "((%text;) | P)*">

<!ELEMENT ADDRESS - - %address.content>

<!ELEMENT DIV - - %body.content>
<!ATTLIST DIV
        align   (left|center|right) #IMPLIED -- alignment of following text --
        >

<!-- CENTER is a shorthand for DIV with ALIGN=CENTER -->
<!ELEMENT center - - %body.content>
```

```
<!--=================== The Anchor Element ==================================-->

<!ELEMENT A - - (%text)* -(A)>
<!ATTLIST A
        name    CDATA   #IMPLIED    -- named link end --
        href    %URL    #IMPLIED    -- URL for linked resource --
        rel     CDATA   #IMPLIED    -- forward link types --
        rev     CDATA   #IMPLIED    -- reverse link types --
        title   CDATA   #IMPLIED    -- advisory title string --
        >

<!--=================== Client-side image maps ============================-->

<!-- These can be placed in the same document or grouped in a
     separate document although this isn't yet widely supported -->

<!ENTITY % SHAPE "(rect|circle|poly|default)">
<!ENTITY % COORDS "CDATA" -- comma separated list of numbers -->

<!ELEMENT MAP - - (AREA)*>
<!ATTLIST MAP
    name    CDATA   #IMPLIED
    >

<!ELEMENT AREA - O EMPTY>
<!ATTLIST AREA
    shape   %SHAPE  rect
    coords  %COORDS #IMPLIED  -- always needed except for shape=default --
    href    %URL    #IMPLIED  -- this region acts as hypertext link --
    nohref (nohref) #IMPLIED  -- this region has no action --
    alt     CDATA   #REQUIRED
    >

<!--=================== The LINK Element ==================================-->

<!ENTITY % Types "CDATA"
        -- See Internet Draft: draft-ietf-html-relrev-00.txt
           LINK has been part of HTML since the early days
           although few browsers as yet take advantage of it.

           Relationship values can be used in principle:

                a) for document specific toolbars/menus when used
                   with the LINK element in document head:
                b) to link to a separate style sheet (rel=stylesheet)
                c) to make a link to a script (rel=script)
                d) by stylesheets to control how collections of
                   html nodes are rendered into printed documents
                e) to make a link to a printable version of this document
                   e.g. a postscript or pdf version (rel=print)
-->

<!ELEMENT LINK - O EMPTY>
<!ATTLIST LINK
        id      ID      #IMPLIED    -- SGML ID attribute --
        href    %URL    #IMPLIED    -- URL for linked resource --
        rel     %Types  #IMPLIED    -- forward link types --
        rev     %Types  #IMPLIED    -- reverse link types --
        title   CDATA   #IMPLIED    -- advisory title string --
        >
```

```
<!--=================== Images ============================================-->

<!ENTITY % Length "CDATA" -- nn for pixels or nn% for percentage length -->
<!ENTITY % Pixels "CDATA" -- integer representing length in pixels -->

<!-- Suggested widths are used for negotiating image size
     with the module responsible for painting the image.
     align=left or right cause image to float to margin
     and for subsequent text to wrap around image -->

<!ENTITY % IAlign "(top|middle|bottom|left|right)">

<!ELEMENT IMG    - O EMPTY --  Embedded image -->
<!ATTLIST IMG
        src     %URL     #REQUIRED  -- URL of image to embed --
        alt     CDATA    #IMPLIED   -- for display in place of image --
        align   %IAlign  #IMPLIED   -- vertical or horizontal alignment --
        height  %Pixels  #IMPLIED   -- suggested height in pixels --
        width   %Pixels  #IMPLIED   -- suggested width in pixels --
        border  %Pixels  #IMPLIED   -- suggested link border width --
        hspace  %Pixels  #IMPLIED   -- suggested horizontal gutter --
        vspace  %Pixels  #IMPLIED   -- suggested vertical gutter --
        usemap  %URL     #IMPLIED   -- use client-side image map --
        ismap   (ismap)  #IMPLIED   -- use server image map --
        >

<!-- USEMAP points to a MAP element which may be in this document
  or an external document, although the latter is not widely supported -->

<!--=================== Java APPLET tag ==================================-->
<!--
  This tag is supported by all java enabled browsers. Applet resources
  (including their classes) are normally loaded relative to the document
  URL (or <BASE> element if it is defined). The CODEBASE attribute is used
  to change this default behavior. If the CODEBASE attribute is defined then
  it specifies a different location to find applet resources. The value
  can be an absolute URL or a relative URL. The absolute URL is used as is
  without modification and is not effected by the documents <BASE> element.
  When the codebase attribute is relative, then it is relative to the
  document URL (or <BASE> tag if defined).
-->
<!ELEMENT APPLET - - (%text)* +(PARAM)>
<!ATTLIST APPLET
        codebase %URL     #IMPLIED   -- code base --
        code     CDATA    #REQUIRED  -- class file --
        alt      CDATA    #IMPLIED   -- for display in place of applet --
        name     CDATA    #IMPLIED   -- applet name --
        width    %Pixels  #REQUIRED  -- suggested width in pixels --
        height   %Pixels  #REQUIRED  -- suggested height in pixels --
        align    %IAlign  #IMPLIED   -- vertical or horizontal alignment --
        hspace   %Pixels  #IMPLIED   -- suggested horizontal gutter --
        vspace   %Pixels  #IMPLIED   -- suggested vertical gutter --
        >

<!ELEMENT PARAM - O EMPTY>
<!ATTLIST PARAM
        name    NAME      #REQUIRED  -- The name of the parameter --
        value   CDATA     #IMPLIED   -- The value of the parameter --
        >
```

```
<!--
Here is an example:

    <applet codebase="applets/NervousText"
        code=NervousText.class
        width=300
        height=50>
    <param name=text value="Java is Cool!">
    <img src=sorry.gif alt="This looks better with Java support">
    </applet>
-->

<!--==================== Horizontal Rule ===================================-->

<!ELEMENT HR    - O EMPTY>
<!ATTLIST HR
        align (left|right|center) #IMPLIED
        noshade (noshade) #IMPLIED
        size  %Pixels #IMPLIED
        width %Length #IMPLIED
        >
<!--==================== Paragraphs========================================-->

<!ELEMENT P     - O (%text)*>
<!ATTLIST P
        align  (left|center|right) #IMPLIED
        >

<!--==================== Headings ==========================================-->

<!--
  There are six levels of headers from H1 (the most important)
  to H6 (the least important).
-->

<!ELEMENT ( %heading )  - -  (%text;)*>
<!ATTLIST ( %heading )
        align  (left|center|right) #IMPLIED
        >

<!--==================== Preformatted Text =================================-->

<!-- excludes images and changes in font size -->

<!ENTITY % pre.exclusion "IMG|BIG|SMALL|SUB|SUP|FONT">

<!ELEMENT PRE - - (%text)* -(%pre.exclusion)>
<!ATTLIST PRE
        width NUMBER #implied -- is this widely supported? --
        >

<![ %HTML.Deprecated [

<!ENTITY % literal "CDATA"
        -- historical, non-conforming parsing mode where
           the only markup signal is the end tag
           in full
        -->
```

```
<!ELEMENT (XMP|LISTING) - -  %literal>
<!ELEMENT PLAINTEXT - O %literal>

]]>

<!--==================== Block-like Quotes ==================================-->

<!ELEMENT BLOCKQUOTE - - %body.content>

<!--==================== Lists ==============================================-->

<!--
    HTML 3.2 allows you to control the sequence number for ordered lists.
    You can set the sequence number with the START and VALUE attributes.
    The TYPE attribute may be used to specify the rendering of ordered
    and unordered lists.
-->

<!-- definition lists - DT for term, DD for its definition -->

<!ELEMENT DL    - -  (DT|DD)*>
<!ATTLIST DL
        compact (compact) #IMPLIED -- more compact style --
        >

<!ELEMENT DT - O  (%text)*>
<!ELEMENT DD - O  %flow;>

<!-- Ordered lists OL, and unordered lists UL -->
<!ELEMENT (OL|UL) - -  (LI)*>

<!--
       Numbering style
    1   arablic numbers     1, 2, 3, ...
    a   lower alpha         a, b, c, ...
    A   upper alpha         A, B, C, ...
    i   lower roman         i, ii, iii, ...
    I   upper roman         I, II, III, ...

    The style is applied to the sequence number which by default
    is reset to 1 for the first list item in an ordered list.

    This can't be expressed directly in SGML due to case folding.
-->

<!ENTITY % OLStyle "CDATA" -- constrained to: [1|a|A|i|I] -->

<!ATTLIST OL -- ordered lists --
        type      %OLStyle   #IMPLIED   -- numbering style --
        start     NUMBER     #IMPLIED   -- starting sequence number --
        compact  (compact)   #IMPLIED   -- reduced interitem spacing --
        >

<!-- bullet styles -->

<!ENTITY % ULStyle "disc|square|circle">
```

```
<!ATTLIST UL -- unordered lists --
        type    (%ULStyle)   #IMPLIED   -- bullet style --
        compact (compact)    #IMPLIED   -- reduced interitem spacing --
        >

<!ELEMENT (DIR|MENU) - -  (LI)* -(%block)>
<!ATTLIST DIR
        compact (compact) #IMPLIED
        >
<!ATTLIST MENU
        compact (compact) #IMPLIED
        >

<!-- <DIR>              Directory list                  -->
<!-- <DIR COMPACT>      Compact list style              -->
<!-- <MENU>             Menu list                       -->
<!-- <MENU COMPACT>     Compact list style              -->

<!-- The type attribute can be used to change the bullet style
     in unordered lists and the numbering style in ordered lists -->

<!ENTITY % LIStyle "CDATA" -- constrained to: "(%ULStyle|%OLStyle)" -->

<!ELEMENT LI - O %flow -- list item -->
<!ATTLIST LI
        type    %LIStyle     #IMPLIED   -- list item style --
        value    NUMBER      #IMPLIED   -- reset sequence number --
        >

<!----------------- Forms ------------------------------------------------->

<!ELEMENT FORM - - %body.content -(FORM)>
<!ATTLIST FORM
        action %URL #IMPLIED  -- server-side form handler --
        method (%HTTP-Method) GET -- see HTTP specification --
        enctype %Content-Type; "application/x-www-form-urlencoded"
        >

<!ENTITY % InputType
        "(TEXT | PASSWORD | CHECKBOX | RADIO | SUBMIT
            | RESET | FILE | HIDDEN | IMAGE)">

<!ELEMENT INPUT - O EMPTY>
<!ATTLIST INPUT
        type %InputType TEXT     -- what kind of widget is needed --
        name  CDATA #IMPLIED     -- required for all but submit and reset --
        value CDATA #IMPLIED     -- required for radio and checkboxes --
        checked (checked) #IMPLIED -- for radio buttons and check boxes --
        size CDATA  #IMPLIED     -- specific to each type of field --
        maxlength NUMBER #IMPLIED
        src   %URL  #IMPLIED     -- for fields with background images --
        align  (top|middle|bottom|left|right) top -- image alignment --
        >

<!ELEMENT SELECT - - (OPTION+)>
<!ATTLIST SELECT
        name CDATA #REQUIRED
        size NUMBER #IMPLIED
        multiple (multiple) #IMPLIED
        >
```

```
<!ELEMENT OPTION - O (#PCDATA)*>
<!ATTLIST OPTION
        selected (selected) #IMPLIED
        value  CDATA  #IMPLIED -- defaults to element content --
        >

<!-- Multi-line text input field. -->

<!ELEMENT TEXTAREA - - (#PCDATA)*>
<!ATTLIST TEXTAREA
        name CDATA #REQUIRED
        rows NUMBER #REQUIRED
        cols NUMBER #REQUIRED
        >

<!--======================== Tables ========================================-->

<!-- Widely deployed subset of the full table standard, see RFC 1942
     e.g. at http://www.ics.uci.edu/pub/ietf/html/rfc1942.txt -->

<!-- horizontal placement of table relative to window -->
<!ENTITY % Where "(left|center|right)">

<!-- horizontal alignment attributes for cell contents -->
<!ENTITY % cell.halign
        "align  (left|center|right) #IMPLIED"
        >

<!-- vertical alignment attributes for cell contents -->
<!ENTITY % cell.valign
        "valign  (top|middle|bottom|baseline)  #IMPLIED"
        >

<!ELEMENT table - - (caption?, tr+)>
<!ELEMENT tr - O (th|td)*>
<!ELEMENT (th|td) - O %body.content>

<!ATTLIST table                          -- table element --
        align     %Where;   #IMPLIED  -- table position relative to window --
        width     %Length   #IMPLIED  -- table width relative to window --
        border    %Pixels   #IMPLIED  -- controls frame width around table --
        dummy     (border)  #IMPLIED  -- fixes SGML error for border w/o value --
        cellspacing %Pixels #IMPLIED  -- spacing between cells --
        cellpadding %Pixels #IMPLIED  -- spacing within cells --
        >

<!ELEMENT CAPTION - - (%text;)* -- table or figure caption -->
<!ATTLIST CAPTION
        align (top|bottom) #IMPLIED
        >

<!ATTLIST tr                        -- table row --
        %cell.halign;               -- horizontal alignment in cells --
        %cell.valign;               -- vertical alignment in cells --
        >
```

```
<!ATTLIST (th|td)                    -- header or data cell --
        nowrap (nowrap)  #IMPLIED  -- suppress word wrap --
        rowspan NUMBER   1         -- number of rows spanned by cell --
        colspan NUMBER   1         -- number of cols spanned by cell --
        %cell.halign;              -- horizontal alignment in cell --
        %cell.valign;              -- vertical alignment in cell --
        width   %Pixels  #IMPLIED  -- suggested width for cell --
        height  %Pixels  #IMPLIED  -- suggested height for cell --
        >

<!-----------------= Document Head =----------------------------------------->

<!-- %head.misc defined earlier on as "SCRIPT|STYLE|META|LINK" -->

<!ENTITY % head.content "TITLE & ISINDEX? & BASE?">

<!ELEMENT HEAD O O  (%head.content) +(%head.misc)>

<!ELEMENT TITLE - -  (#PCDATA)* -(%head.misc)
          -- The TITLE element is not considered part of the flow of text.
             It should be displayed, for example as the page header or
             window title.
          -->

<!ELEMENT ISINDEX - O EMPTY>
<!ATTLIST ISINDEX
        prompt CDATA #IMPLIED -- prompt message -->

<!--
    The BASE element gives an absolute URL for dereferencing relative
    URLs, e.g.

         <BASE href="http://foo.com/index.html">
         ...
         <IMG SRC="images/bar.gif">

    The image is deferenced to

         http://foo.com/images/bar.gif

   In the absence of a BASE element the document URL should be used.
   Note that this is not necessarily the same as the URL used to
   request the document, as the base URL may be overridden by an HTTP
   header accompanying the document.
-->

<!ELEMENT BASE - O EMPTY>
<!ATTLIST BASE
        href %URL  #REQUIRED
        >

<!ELEMENT META - O EMPTY -- Generic Metainformation -->
<!ATTLIST META
        http-equiv  NAME    #IMPLIED  -- HTTP response header name  --
        name        NAME    #IMPLIED  -- metainformation name       --
        content     CDATA   #REQUIRED -- associated information     --
        >
```

```
<!-- SCRIPT/STYLE are place holders for transition to next version of HTML -->

<!ELEMENT STYLE  - - (#PCDATA)*  -(%head.misc) -- style info -->
<!ELEMENT SCRIPT - - (#PCDATA)*  -(%head.misc) -- script statements -->

<!--================ Document Structure ===================================-->

<!ENTITY % version.attr "VERSION CDATA #FIXED '%HTML.Version;'">

<![ %HTML.Deprecated [
    <!ENTITY % html.content "HEAD, BODY, PLAINTEXT?">
]]>

<!ELEMENT HTML O O  (%html.content)>
<!ATTLIST HTML
        %version.attr;
        >
```

付録 D

実験での HTML 出力

実験データの HTML 出力

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Draft 19960821//EN">
<HTML><HEAD><H1>「インターネットに適応した全文データベース検索システムの構成」 </H1>
<H1>"Configuration of a Full Text Database Retrieval System for Internet Environment" </H1>
<P> 大山　敬三　(学術情報センター) </P>
<P>本論文では現在のインターネット上での情報サービス環境を最大限に利用しつつ、高度で使用しやすい
全文データベース検索サービスを提供できるシステムの構成方法を述べている。WWW のクライアントである
Mosaic とサーバである HTTPD を用い、既存の高度な検索エンジンとの間を独自のゲートウェイにより結合す
る。階層構造を持つ文書データを対象に柔軟で効率的な検索を可能とするインタフェースを実現している。
<HR></P>
<P>全文データベース , 情報検索 , 文書構造 , インターネット , ゲートウェイ , グラフィカルユーザ
インタフェース</P>
<P>full text database , information retrieval , document structure , Internet , gateway ,
graphical user interface </P>
<TITLE>インターネットに適応した全文データベース検索システムの構成</TITLE>
</HEAD>
<BODY><H1>1. はじめに </H1>
<P>インターネットの整備とワークステーションの普及に伴い、インターネットを通じて様々な情報サービ
スが提供されており、その数は急激に増えつつある。これらの多くは情報収集・提供のための組織的な支援
体制を伴わないため、内容やサービスの質に問題を抱えている例も多く、情報を探索し利用する立場からは
必ずしも望ましくない状況も発生している。このため、インターネットを通じた情報サービスに批判的な意
見も散見される。 </P>
<P>しかし、一方で、従来から独自のネットワークや方式で提供してきた情報サービスを、インターネット
経由でも利用可能にし、さらにはインターネットをサービスの中心に据える例も少なくない。これらの多く
は従来からの膨大な蓄積を、より効率的に、より利用しやすく、できるだけ多くのユーザに提供しようとす
るものである。このような努力は、従来からのユーザにとってのみでなく、潜在的なユーザにとっても歓迎
すべきものと言えるであろう。 </P>
<P>筆者が所属する学術情報センターでも、従来からいわゆる情報検索サービスを提供している <I> 「情報
検索サービス NACSIS-IR」　"http://www.nacsis.ac.jp/ir/ir-j.html"　(, , ) </I>
が、これまであまりインターネットの機能を有効に利用できないでいた。その原因の一つとしては、サービ
ス提供のためのホストシステムがいわゆるメインフレームコンピュータであり、インターネットで標準的に
利用されている環境が適用できなかったことがあげられる。また、センターのユーザの多くが自由にインター
ネットを使える状態になかったためにセンターが積極的にインターネット対応を推進する動機を欠いたこと
も一つの理由としてあげることができよう。 </P>
<P>しかし、ここ数年間で大学や研究機関における状況は大きく変わり、希望するものは誰でもインターネッ
トを利用できるようになった。このような状況の変化に対し、情報サービスの提供機関は学術情報センター
に限らず、ユーザの利便性を第一に考え、できるだけ迅速に適応していくことがきわめて重要であると筆者
は考えている。本論文では、このような目的を満たすために筆者が中心になって研究開発を進めているイン
ターネット上の情報サービスシステムの構成方法について述べる。 </P>
<H1>2. 情報サービスのツールの現状と利用方針 </H1>
<P>筆者はインターネットを介したサービスを、既存のサービスの代替ではなく、今後のサービスの中心的
役割を担うものと考えている。そのため、既存のサービスが備えている機能の見直しを含め、新しい環境に
適応した高度な機能を実現することが重要である。一方で、インターネットはさまざまなボランティアが提
供してきた多くのパブリックドメインのソフトウェアと、それを元にして開発された商用の製品群によって
大きく発展してきた。このような発展形態は今後も継続するものと考えられる。 </P>
<P>本研究においては、インターネット上で一般的に利用されているツールを有効に組み合わせながら必要
な機能を実現していくことを基本的な考え方とする。特にユーザ環境としては、広く利用可能な一般的なツー
ルを用いてサービスを利用できることを最重要の課題と考えている。そこで、まず本章では、インターネッ
ト上の情報サービスのためのツールの現状を整理し、本システムでの利用方針を検討する。 </P>
<P>インターネット上の情報サーバとしては archie、WWW (World Wide Web)、gopher、WAIS 等が一般的であ
る <I>　"「インターネットの情報サービス」"　(斎藤正史, 山口英 ,　情報処理 ,　Vol.34, No.12 ,
pp.1415-1421 .　1993 ) </I>
。これらはいずれもパブリックドメインのソフトウェアとし提供されており、さらに WWW と WAIS については
機能や性能を強化した商用版も提供されている。これらはそれぞれ目的を異にし、実現可能なサービスにも
それぞれ特徴がある。全文データベース検索機能の実現という観点からこれらを整理すると以下のようになる。 </P>
<P><OL><LI>archie は FTP で提供されているファイル名に対する文字列検索のみを提供する <P><I>
"‘‘Archie Reference Manual Pages"" () </I>
</P>
```

。内容に対するサーチ機能はなく、情報へのアクセスも別途 FTP を用いて行う必要がある。 </LI>
</OL>
<OL><LI>WWW はネットワーク上に分散した情報資源をハイパーテキストの概念に基づいて利用可能にするもので、HTML という形式で文書間に予め張られたリンクをたどることによってアクセスを行う <P><I> "‘‘A Beginner's Guide to HTML"" ("http://www.ncsa.uiuc.edu/demoweb/html-primer.html") </I>
</P>
。他の情報資源に対するリンクの設定も可能であるが、WWW サーバ自体には内容に対する検索機能はない。
</LI>
</OL>
<OL><LI>gopher はネットワーク上に分散した情報資源をメニュー形式で利用可能にするもので、予め階層構造に分類整理されたメニューをたどることによってアクセスを行う <P><I> "‘‘Gopher Reference Manual Pages"" ("http://www.ncsa.uiuc.edu/demoweb/html-primer.html") </I>
</P>
。情報資源の一つの形態として内容に対する簡単な文字列サーチ機能を持つが、情報検索機能としてはきわめて不十分である。 </LI>
</OL>
<OL><LI>WAIS はサーバ中に格納されている文書に対し、文書中に現れる自由語を用いて頻度や類似度に応じた情報検索機能を提供する 。情報検索の知識がないユーザでも比較的容易に使用できるが、履歴や集合演算の機能がないため、複雑な検索を試行錯誤的に行うには不十分である。 </LI>
<LI><I>‘‘WAIS 2.0 Technical Documentation", http://www.wais.com/newhomepages/techtalk.html
</I>
</LI>
</OL>
</P>
<P>また、これらの情報サービスを利用するためのユーザインタフェースとなるクライアントには、GUI を用いて統合的に対話が可能な Mosaic がパブリックドメインソフトウェアとして普及しており <I> ‘‘NCSA Mosaic Home Page", <BR> http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html
</I>
<I> "「社内情報共有に新時代ーインターネットの WWW が急浮上」" (斎藤正史, 山口英 , 日経コミュニケーション , Vol.34, No.12 , pp.76-97 . 1995.1.6) </I>
、商用版も各種のものが利用可能となってきている (以下、これらを総称して Mosaic と呼ぶ)。Mosaic は元々 WWW のビューワとして開発されたが、現在では WWW サーバ (HTTP プロトコルによる) ばかりでなく、gopher サーバや WAIS サーバ (Z39.50 プロトコルによる) へのアクセスが可能であり、また、HTTP プロトコルによりゲートウェイを介して archie サーバなど、他の情報サーバへもアクセス可能である。gopher や WAIS には専用のクライアントもあるが、普及の度合いや使いやすさを考慮すると、Mosaic をクライアントとして利用するのがユーザにとって最も有効であろう。 </P>
<P>一方、全文データベースの検索には、単に自由語による内容のサーチを可能とするだけでなく、文書の階層構造に即した検索と取り出しが可能であることが必須である。既存の情報検索システムの中では、カナダ OpenText 社製の Pat <I> ‘‘Pat Reference Manual" "http://www.nacsis.ac.jp/ir/ir-j.html" (, OpenText Corp ,) </I>
が機能的に必要条件をもっとも良く満たしている。また、今後、より高度なシステムが利用可能になることも期待される。本研究の目的は Mosaic を通じてそれらを適時に利用可能にするための方式を確立することにある。 </P>
<P>Mosaic から利用可能な標準的なプロトコルとしては Z39.50 プロトコルのサブセットである WAIS プロトコルと、WWW で用いられている HTTP プロトコルがある。Z39.50 は情報検索用の汎用の標準通信プロトコルであり、セッションの保持機能があって比較的高度な検索が可能であるが、WAIS プロトコルはこのセッション保持機能を省略している <I> ‘‘WAIS(tm) Network Publishing using Z39.50" "http://www.wais.com/newhomepages/z3950.html" (, OpenText Corp ,) </I>
。また、Z39.50 では検索対象のデータの階層構造を利用した検索・表示要求に対応できない。 </P>
<P>HTTP は HTML 文書などのハイパーテキストを転送するためのプロトコルであるが、HTML の FORM の機能を用いるとユーザに提示する検索用の GUI を送ることができる <I> ‘‘The Common Gateway Interface" "http://hoohoo.ncsa.uiuc.edu/cgi/overview.html" (, OpenText Corp ,) </I>
。HTML ではセマンティクスは何も規定しないので、FORM を設計するときにユーザに容易に理解可能なように考慮することが重要である。FORM に入力されたデータはゲートウェイを通じて情報サーバに届けられ、情報サーバからの出力を FORM にフィードバックすることも可能であるので、Mosaic を通じたユーザとの対話処理用に適している。 </P>
<P>HTTP 自体はステートレスでありセッションの保持機能がないが、HTTP のゲートウェイの高度な構成方式によりこの機能が実現できれば、Pat などの強力な全文検索エンジンと組み合わせることにより、Mosaic を通じて試行錯誤を含む高度な対話処理を可能とする全文データベース検索システムを構成することができる。
</P>
<H1>3. 全文データベース検索システムの要件 <OL><LI><P>節タイトルに「実験方法」が含まれ、節の本文中に「真空蒸着」が含まれるような節を持つ論文を表示する。 </P>
</LI>

<LI><P>同一の段落に「バブル経済」と「経済摩擦」が含まれるような論文の書誌情報を表示する。 </P>
</LI>
<LI><P>同一の段落に「バブル経済」と「経済摩擦」が含まれるような論文について、書誌情報と章・節のタイトル、およびその段落を、出現順に表示する。 </P>
</LI>
</OL>
<TT>(1) サーチ機能 </TT>
<DL><DT>(a) 文字列サーチ </DT>
<DD>ユーザが利用目的に応じてさまざまな角度から検索できるよう、テキスト中に出現する任意文字列でのサーチ、範囲検索、近接演算 (順序指定、順序非指定) などが可能でなければならない。特に日本語の学術情報では、辞書にない単語や複合語でのサーチができることも重要である。 </DD>
<DT>(b) 論理演算 </DT>
<DD>検索対象が階層構造を持っているため、階層に即した論理演算が必要であり、各階層における任意の文書要素および一致文字列に対して、論理積、論理和、論理差、否定などの論理演算が可能でなければならない。 </DD>
<DT>(c) 集合演算 </DT>
<DD>ユーザにとって、複雑な検索を最初から単一の論理式で記述することは困難であり、試行錯誤を繰り返す上でも過去の検索結果集合を組み合わせた集合演算が可能であることが重要である。論理演算同様、文書要素集合および一致文字列集合に対する共通集合、合併集合、排他集合、補集合などの集合演算が可能でなければならない。 </DD>
<DT>(d) 階層演算 </DT>
<DD>階層構造中の任意の文書要素を基点にしてサーチを行い、さらに異なる階層の文書要素に基点を移してサーチを行うためには、階層構造にある文書要素・文書要素間および文書要素・一致文字列間の包含と包摂の演算が、論理演算および集合演算の双方において可能でなければならない。 </DD>
</DL>
<TT>(2) 表示機能 </TT>
<DL><DT>(a) 出現位置の周辺表示 </DT>
<DD>文中にある任意の文字列に対するキーワード検索を行う場合、その語の使われ方をユーザが予めすべて予想することは困難な場合が多いため、キーワードに一致した文字列がどのような文脈で現れたかを簡略に表示する、いわゆる KWIC 表示が有効である。 </DD>
<DT>(b) 文書要素を指定した表示 </DT>
<DD>キーワードによるサーチやその結果の組み合わせにより得られた文書が適切であるかどうか判断し、また、それらの中から有用と思われるものを選択するためには、そのために必要かつ十分な部分を一覧できるような表示機能が必要である。たとえば文献情報部分のみを一覧表示させたり、文書ごとの目次 (文書タイトルと章節の見出しなど) を表示させたり、キーワードが見つかった段落を表示させたりできることが要求される。また、マニュアルなどでは該当する節を表示させるだけでユーザの目的を達せられることもある。
</DD>
<DT>(c) 全文を簡易整形して表示する機能 </DT>
<DD>選択した文書を通覧するためには、ある程度読みやすい形でオンライン表示できる必要がある。最終的には SGML で記述された原データを加工して厳密な整形出力をオフライン印刷できればよいであろうが、ユーザの利用性を高めるためには対話処理のなかで簡易整形表示できる機能が重要である。 </DD>
</DL>
</H1>
<P>本システムでは、検索の対象を、学術文献 (数ページ) から技術マニュアル (数千ページ) 程度の、階層構造を持つ文書の全文データとする。必ずしも必要な条件ではないが、SGML(Standard Generalized Markup Language) で記述されているものと仮定する。データベースにはこのような相互に比較的独立な全文データが多数格納されている。 </P>
<P>本章では、このような環境において全文データベース検索システムが扱うべき文書データの構造を分析し、多数の文書の中から所望の文書を探し出すためにユーザが必要とする機能を検討することにより、システムに要求される要件を明らかにする。 </P>
<H2>3.1 文書データの構造と検索要求 </H2>
<P>本システムが対象とする文書データは一般に図 のような階層構造を持つ。この他にも参考文献とその参照のようなリンク構造もあるが、多数の文書データに対して一括して検索する場合にこれが有効な手がかりとなることはあまり無いと考えられるため、検索対象は階層構造のみとする。 </P>
<P>(図文書の階層構造) </P>
<P>SGML で階層構造定義に関連して定義可能なものには、出現順などを指定する結合子と、繰り返しなどを指定する出現指示子がある <I> ‘‘The SGML Handbook" , Clarendon Press (Goldfarb, Charles F. , 1990 , 663pp) </I>
。これらの中で、出現順などは多数の文書の中からキーワードを頼りに検索する環境ではあまり有効な利用方法は考えられない。一方、繰り返しは適切に取り扱えないと精度の高い検索が期待できない。 </P>
<P>ユーザが必要とするであろうと考えられる検索条件の例としては、以下のようなものが考えられる。
</P>

<P>これらのうち、(1)、(2) は検索対象と表示対象の関係が比較的単純であるため条件の記述も容易であるが、(3) はやや複雑となるため Mosaic を通じてユーザが条件の指定をできるような簡潔な HTML の FORM を作成することは困難である。本システムでは、検索対象と表示対象の関係が単純な包含関係にある場合のみを想定する。 </P>
<H2>3.2 システムの要件 </H2>
<P>情報検索システムの一般的な機能には大きく分けてサーチ機能と表示機能がある。全文データベース検索では通常の書誌型、索引型、あるいは抄録型の情報検索と比較して、以下のような機能が特に重要である。
</P>
<P>これらの機能のうち、サーチ機能はサーバで実現する必要がある。表示機能はサーバ、クライアントのいずれでも実現可能であるが、本システムでクライアントに想定している Mosaic にはこれらを実現する機能がないため、サーバで実現する必要がある。また、仮にクライアントがこれらを実現する機能をもっていたとしても、通信の負荷を考慮すると多数ユーザの環境では現実的ではない。 全文データベース検索では特に文書がもつ階層構造を適切に処理する必要があるため、上記のような複雑な機能を持つシステムが必要であるが、一般のユーザがこのような機能を理解し、対象とする文書の具体的な構造を覚えて検索に有効に利用するのは極めて難しい。このため、メニューなどを用いることにより特別の知識を持たないユーザでも上記のような高度な機能を容易に利用できるようなユーザインタフェースを実現することが重要である。
</P>
<H1>4. システム構成 <TT>(1)HTTP クライアント </TT>
<TT>(2)WWW サーバ </TT>
<TT>(3) ゲートウェイ </TT>
<TT>(4) フロントエンド </TT>
<TT>(5) 全文検索エンジン </TT>
<TT>(1) セッション管理機能 </TT>
<TT>(2) コマンド解釈機能 </TT>
<TT>(3) フォーム表示機能 </TT>
<TT>(4) 文書表示機能 </TT>
HTTP クライアントはユーザがシステムと対話するための直接のインタフェースを提供する。ユーザがインターネット上の情報サーバにアクセスするときに随時起動する。現在利用可能なものには Mosaic 以外にもいくつかの種類が存在し、フォームを扱う機能を持つものであれば本システムのクライアントとして利用可能である。しかし、ユーザインタフェースの質や利用可能なプラットホームの多様性から、現在では Mosaic が最も利用しやすい。 WWW サーバはクライアントからのアクセスに対して要求された情報を提供する。通常は予め用意された HTML 文書や画像ファイルを送信するだけであるが、本システムではフロントエンドで動的に作成される HTML 文書をクライアントに送信する。全文データベース検索サーバの起動時に 1 回だけ起動され、システム停止まで存続する。クライアントとは HTTP プロトコルにより TCP/IP を介して通信する。ゲートウェイ側のインタフェースは CGI(Common Gateway Interface) として規格が定められている。WWW サーバにはパブリックドメインや市販製品として利用可能な既製のものがいくつか存在するが、今回は CERN 版の HTTPD を用いることとした。 ゲートウェイは WWW サーバとフロントエンドの間のデータの仲介を行う。フォームが送られてくるごとに WWW サーバから起動され、標準入出力を通して WWW サーバと情報の授受を行う。フロントエンドとの通信は各フロントエンドと 1 対 1 に対応した FIFO ファイルを通じて行う。結果の情報を WWW サーバに返すと消滅する。頻繁に生成と消滅を繰り返すのでシステムの性能を律する可能性があるため、可能な限り機能を簡素化してある。 フロントエンドはユーザの要求を全文検索エンジンのコマンドに解釈して実行させるとともに、結果やデータの表示に必要な HTML のフォームや文書を作成する。セッションの維持や検索の履歴の管理に必要な処理も行う。個々のセッションに対応して 1 つずつ動的に生成され、セッションが終了すると消滅する。全文検索エンジンとはパイプを通じて通信する。既存の構成要素で提供されていない機能の大部分はここで実現され、本システムのプログラム開発の大部分が含まれている。 全文検索エンジンには、各種のパブリックドメインソフトウェアや市販製品を比較検討した結果、カナダ OpenText 社製の Pat サーバを採用することとした。ただしこれは全文データベースに対する検索機能を評価した結果の選択であって、システムの構成上からの必要性によるものではない。従って、今後、より適当なものが利用可能になればそれを採用することも可能である。全文検索エンジンはフロントエンドにより管理されており、セッションが開設されるたびに起動され、セッションが終了すると消滅する。 セッションの開始時には、ログインフォームに記入された情報からセッション ID を生成し、指定されたデータベースに対して Pat を起動するとともに、以後のセッションの中で GWP からのアクセスに用いる FIFO ポートを作成してからフォーム表示機能を呼び出す。セッション ID はユーザ名、クライアントホスト名、サーバホスト名、FEP のプロセス ID、セッション開始時刻などから容易に想像できないように生成する。課金などのためにユーザ管理が必要な場合は、上記の処理を開始する前に、ログインフォームに記入されたユーザ ID やパスワード、および HTTPD から渡されるクライアントのホスト ID などを用いてユーザ認証を行う。 セッションの終了処理はユーザからのセッション終了要求を受けて行う。Pat を終了させ、利用統計などを記入した終了通知を作成して返す。ユーザ管理が必要な場合は各種統計情報をファイルに書き出す。使用済みの FIFO ポートを削除してプロセスを終了する。 HTTP にはセッションの概念がないため、クライアントとサーバは 1 回のアクセスごとに接続と切断を繰り返す。従って、クライアントがログアウト要求を送らないまま終了してしまうと、サーバはこれを検知できない。このため、FEP はセッションの保留時間を監視し、ユーザから要求がない状態が一定時間継続した場合は中断処理を行う。現在は中断処理では強制的に終了処理を実行しているが、サービスの内

容に応じは、後で回復可能なように中断時の FEP と Pat の状態を保存してセッションを解放するというような機能の導入も考えられる。 検索フォームに埋め込まれた検索要求を取り出し、Pat のコマンドに解釈して処理を実行させる。検索要求にはサーチ要求と表示要求がある。サーチ要求に対しては検索条件を Pat のサーチコマンドに翻訳して実行させる。要求内容と Pat が作成した検索集合を履歴として保持し、実行結果をフォーム表示機能に渡す。表示要求に対しては、選択された文書の中から指定された文書要素を取り出すための Pat のコマンドに翻訳して実行させる。取り出した文書データは文書表示機能に渡す。 あらかじめ対象データベースごとに用意された検索フォームのテンプレートに、サーチの履歴と検索集合の内容の簡略表示を埋め込んで HTTPD に渡す。履歴には最近の指定回数分の検索集合の番号、その件数、およびサーチ要求の内容を表示する。簡略表示には最新の検索集合に含まれる文書データの先頭部分を指定件数分だけ表示する。検索フォームのテンプレートにはサーチ文字列の入力フィールド、サーチ範囲を限定するための文書要素のメニュー、集合演算のための演算子のメニュー、階層演算のための文書要素のメニュー、表示要求で文書要素を指定するための文書要素のメニューなどが予め準備されている。履歴にはチェックボックスを付加し、集合演算のオペランドとして指定できるようにする。簡略表示にもチェックボックスを付加し、表示要求の文書指定をできるようにする。 表示モードとして整形出力とソース出力を持つ。整形出力モードでは与えられた SGML 文書を HTML 形式に変換してクライアントに送る。変換のためのフィルタはデータベースごとに独立したプログラムとして用意しておく。ソース出力モードでは元の SGML 文書をそのままプレインテキストとしてクライアントに送る。 </H1>

<P>本章では、上記のような諸々の条件を満たすために採用した全体構成とその動作の概要を述べ、本システムの開発の中心となるフロントエンドについて詳述する。 </P>

<H2>4.1 全体構成 </H2>

<P>本システムでは前述の通り、クライアントとして Mosaic を利用し、WWW サーバを基礎として全文データベース検索サーバを構成する。全体構成を図 (a) に示す。各部分の機能と相互間の関係を簡単に述べる。 </P>

<P>(図システムの概要) </P>

<H2>4.2 動作の概要 </H2>

<P>全体の動作の概要を図 (b) に示す。ユーザはまず HTTP クライアントである Mosaic を起動し、ログインフォームを WWW サーバである HTTPD から取り出す。フォームにユーザ ID などの必要な情報を記入し、使用するデータベースを指定して HTTPD に送る (ログイン要求)。HTTPD はゲートウェイプロセス (GWP) を起動してフォームに記入されたデータを渡す。GWP はセッションの環境を作成してからフロントエンドプロセス (FEP) を生成する。FEP は全文検索エンジン (Pat) を起動し初期化してから、検索フォームを GWP、HTTPD を経由して転送して Mosaic に表示させる。 </P>

<P>ユーザは、検索フォームに検索条件を記入して HTTPD に送り返す (検索要求)。HTTPD は新たな GWP を起動してフォームに記入された検索条件を渡す。GWP は対応する FEP を見つけて検索条件を渡す。FEP はこれを Pat のコマンドに解釈して Pat に渡す。Pat から結果を受け取ると FEP は検索フォームに検索集合の一覧と内容の簡略表示を埋め込んで結果フォームを作成し、GWP、HTTPD を経由して Mosaic に送り返し、表示させる。 </P>

<P>結果フォームに簡略表示されたもののなかで詳細な表示をさせたいものがある場合は、ユーザは文書を選択して HTTPD に送り返す (表示要求)。HTTPD は GWP を起動して要求を FEP に渡し、FEP はそれをコマンドに解釈して Pat に処理を行わせる。取り出した文書データは FEP が HTML 形式に簡易整形して GWP、HTTPD を通じて Mosaic に転送する。 </P>

<P>一連の検索を終えると、ユーザは結果フォーム中に終了の指示を記入して HTTPD に送り返す (ログアウト要求)。GWP を通じてログアウト要求を受け取った FEP は、Pat を終了させ利用統計などを記した終了通知の HTML 文書を作成し、これを GWP、HTTPD を経由して Mosaic に転送した後に終了する。 </P>

<H2>4.3 フロントエンドの構成 </H2>

<P>フロントエンドは本システムの開発の中心であり、機能的にも最も複雑である。機能には大きく分けて、セッション管理機能、コマンド解釈機能、フォーム表示機能、文書表示機能がある。以下でこれらについて説明する。 </P>

<H1>5. ユーザインタフェース <TT>(図ログインフォーム) </TT>

<TT>(図検索フォーム (その 1)) </TT>

<TT>(図検索フォーム (その 2)) </TT>

<TT>(図詳細表示) </TT>

<DL><DT>(1) キーワードによる文字列サーチ </DT>

<DD>検索に用いる任意の文字列をフォームの「検索文字列」フィールドに入力する。検索範囲を限定する場合は「サーチ範囲」から対象とする文書要素を選択する。履歴表示部分に結果の集合番号、件数、および検索内容が表示され、簡略表示部分に該当部分のリストが表示される。表示する履歴の数と簡略表示の数はパラメータ設定部分の「履歴表示件数」と「簡略表示件数」で指定する。なお、この検索フォームでは論理演算はサポートしておらず、単一文字列によるサーチで集合を作っておいてから集合演算を行うことで代替する。 </DD>

<DT>(2) 集合演算と階層演算 </DT>

<DD>演算の種類を「演算」から選択し、オペランドを検索集合か文書要素の中から指定して「実行」する。演算の種類には集合演算、階層演算、近接演算がある。集合演算は 2 つのオペランドの文書要素の種類が一致しなければあまり意味がない。階層演算は 2 つのオペランドの文書要素の種類の間の上下関係が適切でないと意味の無い結果となる。近接演算は 2 つのオペランドが文字列サーチの結果でないと出現位置の間の距離を反映しなくなる。なお、距離 (文字数) はパラメータ設定部分の「近接演算距離」で指定する。 </DD>

<DT>(3) 簡略表示 </DT>
<DD>文字列サーチや集合・階層演算を行うと最新の検索集合の簡略表示が行われるが、以前の検索集合を見たいときには履歴部分の表示項目をクリックする。また、表示している部分の前後を見たい場合は上下の矢印のリンクをクリックする。 </DD>
<DT>(4) 詳細表示 </DT>
<DD>詳細表示には 2 つの方法がある。表示したい文書の簡略表示を直接クリックすると、その文書が詳細表示される。また、簡略表示の頭にあるチェックボックスをチェックしてから「表示」を行うと複数の文書を一括して詳細表示することができる。「全部表示」をチェックすると全ての文書を一括表示できる。表示の形式はパラメータ設定部分の「詳細表示形式」で選択する。 </DD>
<DT>(5) パラメータの設定 </DT>
<DD>各種のパラメータの設定はパラメータ設定部分の項目を指定して「設定」を行う。 </DD>
<DT>(6) その他 </DT>
<DD>セッションの終了は「ログアウト」で行う。また、データベースの内容に関する情報と検索フォームの使用方法に関する情報はマニュアルのリンクをたどることで参照可能である。 </DD>
</DL>
</H1>
<P>階層構造を持つ文書を対象とする全文データベース検索システムのユーザインタフェースでは、文書の階層構造の処理を含む検索要求を入力できる必要があり、指定すべきパラメータの種類が多い。ユーザがこれらをすべて記憶し、適切に誤りなく直接入力することはきわめて困難である。そこで、本システムではこれらのパラメータをユーザに提示し、ユーザはそれらの中から選択してゆくだけで検索要求を入力できるようなユーザインタフェースを実現する。HTML のフォームで Mosaic に表示できるグラフィカルユーザインタフェースの構成要素には限界があり、ユーザの操作に対応したきめ細かい動作はできないが、基本的な検索の対話をするための要求は満たせると考えている。 </P>
<P>図 、 、 および図 にユーザインタフェースの例を示す。図 はログインフォームの表示である。ユーザはユーザ ID、パスワードを入力し、データベースの一覧の中から 1 つを選択する。図 (a)、図 (b) は検索フォームの表示例である。サーチ要求の記入部分、履歴表示部分、簡略表示部分から構成されている。ユーザが直接入力するのはサーチ文字列だけである。図 は簡易整形した文書表示の例である。 </P>
<P>図 (a)、図 (b) に示した検索フォームの操作について次に説明する。 </P>
<H1>6. 終わりに </H1>
<P>本論文ではインターネット上でもっとも広く用いられているユーザ環境から利用可能な全文データベース検索システムの構成方法について述べた。このシステムは、データベースに収録されている文書の構造についての特別な知識なしに GUI を通じて容易に検索できることを特長とし、本格的で先進的な情報検索サービスの利用を可能とする。 </P>
<P>本論文で述べた検索フォームは基本的なものであり、データベースの特性に応じて予めアクセス形態のモデル化が可能なものについては、それに沿った検索をより効率的に行うための検索フォームの作成を行うのがよいであろう。複数の検索フォームの中から、適切なものを選んで使用することも容易に実現できる。
</P>
<P>本構成方法は、より一般的な既存の情報検索サービスにも適用可能であり、利用方法が難しいためにユーザ層が限定されていたり有効な機能が十分に活用されていなかったような分野のサービスにも、より広く有効に利用される可能性を開くものである。 </P>
<P>本システムは現在開発中であり、本論文で述べた内容は若干変更される可能性がある。完成後は利用実験による評価、改良を行った後にサービスを一般に公開する予定である。また、同様のサービスを行おうとする者のために、開発したソースコード、HTML フォームなども公開する予定である。 </P>
<P>ただし、HTML の記述能力や Mosaic の表示能力の制約から、不満足な点が残っていることは否めない。履歴や簡略表示がスクロールできないのもそのひとつである。これらは HTML の拡張や Mosaic の機能強化により徐々に解決するであろうが、より専門的なユーザのためには専用のクライアントと通信プロトコルを用意すべきであろう。 </P>
<P>学術情報センターでは現在、本システムの他に、文字型ユーザインタフェースの全文データベース検索システムを開発中であり、また、電子図書館の一部として機能する全文データベース検索サーバも開発中である。これらについては別の機会に紹介する。 </P>
</BODY>
</HTML>