UDC 371.279.7

【展　望】

統計メタデータ記述の技術動向とXML文書の利用

大津　起夫*

要　約†

大学入試センターの業務および研究によって得られたデータの蓄積とともに，それらの組織的な管理も次第に重要となっていく．各種のデータと資料の管理情報を統一された形式で電子化し検索可能とするための技術的な基盤として，XML（eXtensible Markup Language）を用いる手法が広がりつつある．

ここでは，まず統計メタデータ（データの管理情報）にかかわるプロジェクトの動向をサーベイする．特に欧米での社会科学データアーカイブにおけるメタデータ記述，および電子メディアを用いた学習とテストにかかわる教材や受験者情報データの標準化の動向，および人文系資料全般についての標準記法の提案であるTEI（Text Encoding Initiative）について報告する．ついでそれらのプロジェクトにおいて基盤技術として採用されているマークアップ言語であるXMLの特徴と関連技術の動向を紹介し，XML文書の構造，およびアプリケーションプログラムインタフェース（API），XSLとよばれるXML変換プログラム仕様，XML文書の構造の定義を記述するためのスキーマ定義言語などの動向を報告する．

最後に，Prologと呼ばれる論理型記号処理言語がXML文書の処理，特に研究目的での利用に有効であることを指摘し，処理例を示す．XML文書の読み込みにおいては，文字コードリストをDCG（Definite Clause Grammer）とよばれる文法記述にしたがって構文解析し，木構造を持つPrologのデータへと変換する．このようにして得られた構造は，パターンマッチングと非決定的実行メカニズム（バックトラック）を用いることにより柔軟に検索可能となる．さらに，Prologによって表現されたデータをテンプレートとして用い，パーザの入出力を逆方向に用いることにより，XML文書の生成が容易となることを示す．

キーワード：メタデータ，XML，DDI，Prolog

1　統計メタデータの必要性

大学入試センター試験は毎年度数十万人の志願者を対象として実施されている．試験内容，実施方法などの改良を行うためには，過去の試験の実施によって得られたデータに含まれる情報を十分に把握し利用することが必要である．しかし取り扱うべきデータが膨大でありまた採点基準など多くの付加的な情報が分析において必要とされる．こ

* 大学入試センター 研究開発部試験作成支援研究部門，
〒153–8501 東京都目黒区駒場 2–19–23
2003年11月28日　受理
† 本研究は一部，大津起夫・斎藤大輔・中島晃（2003）「データの属性記述とソフトウェア」2003年度統計関連学会連合大会名城大学（名古屋市），および大津起夫（2003）「統計メタデータの記述と利用」日本心理学会第67回大会 東京大学（文京区）の発表に基づく．

のため各種のマスターファイルの内容を即座に利用することが難しい．また，センター試験業務の他に，研究開発部では各種関連分野の調査が積極的に行われているが，そこで得られたデータは個別の研究者によって管理されており，また分析結果は紙媒体の報告書中に表現されている．今後，これらのデータと資料が蓄積されて大量となり，また研究者の入れ替わりなどが生じると，次第に過去の業務と研究によって得られた情報の全体像を把握することが難しくなる可能性がある．

この問題に対応するためには，各種のデータと資料の管理情報を統一された形式で電子化し検索可能とする必要がある．このような技術基盤への要求は，企業や教育研究機関において共通に生じている．データベース技術はこのような要求に応える一つの手段であったが，従来は定型的な情報のみを主たる管理対象としていた．1990 年代前半まで構造化文書管理について中心的であった技術は SGML（Standard Generalized Markup Language）である．SGML は現在広く利用されている Web ページ記述言語である HTML や XML の前身となったものであり，広範な文書記述能力を持つように設計されている．ISO による規格は 1986 年に制定され（ISO8879），1992 年には JIS 規格が制定されている（JISX4151）．しかしながら，仕様が大きいこともあり，検索や印刷のための処理システムが高価なため，大企業内の技術文書の管理システムなどを除いてはあまり広くは普及しなかった．当時の SGML の応用例については，根岸・石塚（1994）に後述の TEI プロジェクトの例を含め紹介されている．現在普及しているマークアップ言語は HTML と XML であるが，これらは SGML の拡張ではなく，むしろ機能を制約することによって普及の機会を得たものである．

現在では，インターネット上の Web の普及を背景に，これらマークアップ言語を利用する環境が整備され，各種のソフトウェアツールが無償または低コストで利用できる．このため，各種の XML で記述された情報をデータ管理に用いようとする試みが広がりつつある．ここでは，統計データ（特に調査データ）のコードブック（実施要領，調査票，コード表などの記録）の記述を中心に，XML による統計メタデータ記述の研究と利用の動向を紹介する．マークアップ言語は複雑な入れ子の構造を持っており，文書を読み込みこの構造を理解するソフトウェアを自作するのは容易ではない．XML 文書を読み込むためのソフトウェアは，一般的に「XML パーザ」と呼ばれ，このための C++や Java 言語用の関数ライブラリが整備されている．しかし，これらのライブラリを用いて研究目的でプログラムを作成するのは，一般的にはかなり煩雑になる．ここでは論理型記号処理言語（本稿は Prolog と呼ばれる計算機言語を扱う）上の XML パーザを用いた XML 文書の利用と生成法を加えて紹介する．この方法は，処理効率（特にメモリ利用）の面で幾分性能が悪く，また現在のところ広く普及はしていないが，論理型言語の持つパターンマッチング機能を用いることにより，対話的に柔軟な XML 文書の操作が可能である．この特徴は研究目的での利用に極めて有益と思われる．

なお，ここでは詳しくは触れられなかったが，Web 上での資料利用に関係する技術動向として重要なものにセマンティックウェブ関連技術がある．これはオンラインで参照される資料（HTML 文書など）に意味的な記述を付加することにより，ソフトウェアにより多くの情報を与え検索と利用を高度化しようとするものである（萩野他，2002；浦本，2003）．このような要求は幅広く存在するため，今後の急速な技術的展開が予想され，テスト関連データの分析と管理にも影響を与えると思われる．

2 マークアップ言語を用いた統計メタデータ記述

社会調査や試験データの分析など現実のデータ分析作業においては，データ本体以外の関連情報をどれだけ有効に用いるかが，分析の質を左右する．現状の統計分析用のソフトウェアにおいては，簡単な変数ラベルやコードの記述は可能であるものの，データに関連する広範な情報を十分に記述することは，一般的にはいまだ実現されていない．

また社会科学分野の研究においては，各種の調査データが電子的に可読な形態で蓄積されつつあるが，調査仕様（サンプルリング計画，インタビューの手順，質問票，集計分析結果）などは，報告書または未整理の文書として保存されている場合が多い．

調査データ処理の実務においては，データのクリーニング（欠測値や異常値への対応），ファイル名の管理，調査票とコードブックの取り扱いなどが労力の大きな部分を占めている．先進的な研究機関によって行われる分析を除いては，平均や分位点，クロス集計の計算などが実際に行われる統計処理のほとんどを占めている．しかし，アカデミックな視点から見て先進的な手法が用いられていないとしても，それは現実の作業が簡単であることを意味しない．

現状での多くの場合，統計的データ解析の作業に商用の統計ソフトウェアが用いられており，変数名や変数に与えられるラベルの指定を行うことが可能である．しかし現実の研究や業務におけるデータの関係を表現するためには機能が十分とはいえない．

現在主流である商用統計ソフトウェアの多くは，大型計算機がデータ分析に用いられていた時代に開発が始まったものであり，関係型データモデル（Codd, 1970；Date, 1981）を基盤に置いている．関係型データモデルは，データベース管理システムの研究から生まれた概念であるが，柔軟な表現力を持ちビジネス等の業務処理に優れた基盤を提供する．しかし複雑な業務システムの構築においては，データベース管理システム本体だけではなく，入り組んだデータの関係を管理する専用の方法論の助けを用いてデータの論理構造の設計が行われている（Fowler, 2000）．また，関係型データモデルが提供する基本機能は，分析対象データセットの表現と管理には向いているが，分析結果を表現するためのモデルとしては適していない．

業務処理のためのデータの論理構造を取り扱う技術は，データモデリングと呼ばれ，主に情報処理の実務にたずさわる技術者によって担われてきた．1980 年代には関係型データベースを基本構造とし，さらにその構造上に企業の業務や研究活動によって生じるデータの意味を付与しようとする研究が行われた．これらの研究が実現しようとしたシステムは，一般的に「意味データベース」と呼ばれている．意味データベースの研究の中には，統計データの分析支援を主眼に置くものも提案されていた（Su, 1983；佐藤，1988）．これらの技術は現状では統計研究の主要な分野とは認識されていないが，社会科学や行動科学等の研究においては今後大規模な国際比較や経時的調査が増えると予想され，大規模なデータハンドリングを組織的に支援する方法が必要とされている．

電子的に蓄積されたデータの可用性を高めるには，データの利用者が必要とする情報を付与することが必要になる．記述すべき情報の種類とその実現の方針にはいくつか異なる研究の流れがある．

(1) 統計ソフトウェアなどによって扱うデータ構造を豊富なものとする．対話的な統計分析用言語である S（Becker, Chambers, & Wilks, 1988; Chambers, 1998）や R（Ihaka & Gentleman, 1996）ではリストや多重配列を用いることによって複雑な構造を持つデータを表現し，また入力データだけでなく，分析結果についてもオブジェクトとして適切に表現できるようになった．また数式処理システムにおけるデータ交換の必要性から，W3C（World Wide Web Consortium）による MathML（Carlisle, Ion, Miner, & Poppelier, 2003）や OpenMath Society による OpenMath（Buswell, Caprotti, Carlisle, Dewar, Gaetano, & Kohlhase, 2003）など各種の数式（ベクトルや行列を含む）の共通表現の仕様が提案されている．また，国内では慶応大学理工学部において柴田里程教授の研究グループによる統計データの標準的記述方法のプロジェクト（DandD/DandDII）が行われている（柴田，2000；横内・柴田，2001）．

世界的には，ソフトウェア開発会社による統計データとモデルの交換のための標準仕様を定めようとする動きがいくつかある．回帰式やニューラルネットなど統計的予測のためのモデルの標準記述を XML によって行おうとするものとしては，ソフトウェアベンダーを構成員とする Data Minig Group（DMG）によって PMML（Predictive Model Markup Language）が提案されている（Data Mining Group, 2003）．また，データの蓄積と分析を担うサーバとこれに問い合わせを行う PC との間のデータ授受の仕様を XML を用いて記述する試み（XML for Analysis）があり，かなり詳細な仕様書が発表されている（Microsoft Corp. & Hyperion Solutions Corp., 2002）．

また，特に教育とテストに関連する分野においては，教材，テストおよび受験者の情報を標準化して記述しようとする試みが盛んに行われている．こ

れはコンピュータを用いた教育への需要が高まっていることを背景にしており，後述のIMSやISOなどの組織が仕様を提案している．

(2) 意味データベースモデリング（Semantic Database Modeling）．特に1980年代に研究が行われていた（Hull & King, 1986）．これらの中で代表的なものは関係型データモデルを基盤とし，さらに現実のビジネスデータを扱うための表現力を増そうとする試みである．これらのデータベースシステムは概念や実体をノードで表現し，それらの関係を矢線などによって表現する．現在の情報システム設計技法であるUML（Unified Modeling Language）はこの流れを汲んでいる．より最近ではネットワーク上で，データ間の意味的な関連を組織化する方法（セマンティックWeb）や，データベースの構造を統計的な処理に適応させたシステム（データウェアハウス）が実用化されている（Inmon, Rudin, Buss, & Sousa 2002；Kimball, 1998）．

(3) 統計データの書誌情報とコードブックの電子化．統計データ（社会科学分野）の利用者側での研究が行われている．過去1, 2年ほどの間に，サーベイデータアーカイブの検索システムがネットワークを通じて次々と（試験的ではあるが）利用可能となった．ICPSRによるDDIプロジェクト（Data Documentation Initiative, 2003），ヨーロッパの社会科学データアーカイブによるNESSTAR（Musgrave, Littlewood, & Ryssevik, 2000；Ryssevik & Musgrave, 2001），U.S. Census Bureauと the Centers for Disease Controlによる DataWeb（U.S. Census Bureau & the Centers for Disease Contorol, 2003）などはいずれも蓄積された調査データを利用可能とするためのメタデータを電子的に記述している．特に書誌関係情報とコードブックを検索可能とすることに努力が払われている．

これらは，いずれもデータに意味を付与することを目的としているが，設計の意図および具体的な内容は互いにかなり異なる．データに付加された情報は，1や2の例ではソフトウェアの動作を規定したり，許容可能な動作を制約する．また，1では単体のオブジェクトの構造によって意味を表現しようとするが，2ではむしろオブジェクト間の関係が意味表現の中心となる．一方3の例では，社会調査コードブックの記述は，データとデータの外の情報（他のデータ，文書，人間の知識）との関連を表し，解釈システム（の多く）が人間であることを想定している．

海外でのこれらの分野のプロジェクトの内，大学入試センターにおける業務と研究とに関連の深いもののひとつはサーベイデータのコードブック電子化であり，もう一つはテスト教材と回答データの共通交換仕様の作成である．

2.1 サーベイコードブックの電子化プロジェクト

欧米においては，社会科学分野での調査データが，調査実施者以外によって利用されることを前提として容易に利用可能とするための技術的開発（上記3にあたる分野）が進展している．調査データ（特に社会学，政治学関連分野）のデータを管理し研究者に提供するための活動を行う機関は「データアーカイブ」と呼ばれている．米国におけるICPSR（Inter-university Consortium for Political and Social Research，1962年設立）や，Roper Center（1947年設立）は電子化された調査データの蓄積に早い時期から取り組んでおり，現在ではオンラインで多くのデータの提供が行われている．国内での社会調査データの電子化と共有への組織的な取り組みはかなり遅れ，1996年に東京大学の社会科学研究所に設置された日本社会研究情報センターが日本社会に関する調査データのデータアーカイブ（SSJデータアーカイブ）としての活動を開始した．

なかでも，注目に値するのは調査データのコードブック（調査票とファイルフォーマットの記述）を電子化するための技術的方法が整備され，実用化の段階を迎えていることである．この分野で重要と思われるものは，ICPSRにおけるDDI（Data Documentation Initiative）と呼ばれる電子コードブックの仕様策定のためのプロジェクトである．

ICPSRでは蓄積された調査データとコードブックを管理するためにOSIRISと呼ばれるシステムを用いていたが，インターネットの普及とWeb技術の進展を考慮して，次世代のデータ管理システムのためのデータ管理情報の記述方式を作成するプ

表 1　ダブリンコアによる要素タイプの定義

		要素タイプ	定義
1	タイトル	title	資料（resource）に与えられた名前
2	著者あるいは作者	creator	内容の作成者（個人，組織，サービス）
3	主題およびキーワード	subject	トピックまたはキーワード
4	内容記述	description	内容の説明，要約
5	公開者（出版者）	publisher	資料を利用可能とした者
6	寄与者（他の関与者）	contributor	寄与した他の者
7	日付	date	資料の履歴に関わる日付
8	資源タイプ	type	性質，ジャンル，カテゴリー
9	形式（フォーマット）	format	物理的または電子的なデータフォーマット
10	資源識別子	identifier	資料を識別するための一意的な文字列または番号（URL など）
11	情報源（出処）	source	資料が基づいている資料への参照
12	言語	language	資料の知的内容の言語
13	関係	relation	他の資料への関係
14	対象範囲（空間的・時間的）	coverage	資料の内容の範囲（地理，時間，行政など）
15	権利管理	rights	資料にかかわる知的権利

ロジェクトが1995年に開始された．この時点での構造化文書の記述方式として主流であったSGMLを用いたフォーマットによる策定を目指していた．1996年に構造化文書の規格としてXMLがW3Cによって提案されると，DDIプロジェクトもXMLの採用を決定し，2000年3月には第1版の仕様を発表している．2003年7月には多次元配列データの記述などが強化された第2版が発表された．

DDIは多くの社会科学データアーカイブに影響を与え，ヨーロッパの複数の機関による仮想データアーカイブ構築を目指すNESSTAR，Harvard大学とMITによるVDC（Virtual Data Center）(Altman et al., 2001)，カリフォルニア大学バークレイ校における社会調査データのサービスシステム（SDA，Survey Documentation and Analysis）などのプロジェクトによってコードブックの記述方法として採用されている．

このDDIの定めるコードブックのフォーマットは，データの書誌情報とデータファイルについての詳細情報の両者を含み，データ収集者以外の研究者による2次分析が容易に行えるよう配慮されている．書誌情報については，ダブリンコアDublin Coreと呼ばれる書誌情報の規格化グループによって提案され，国際規格（ISO15836）となった標準に基づいている．コードブックの全体は(1)文書情報（コードブック全体についての情報，作成者，著作権など），(2)研究情報（研究の主体，目的，研究方法等），(3)ファイル情報（データファイルの所在，利用権限等，ファイルフォーマット）(4)変数情報（変数ラベル，カテゴリー定義，欠測の定義，頻度等），(5)その他の関連情報，の5つのセクションから構成されており，最初の2つのセクションが書誌情報，残りがデータ処理に関係する内容の記述となっている．ファイル中での変数のカラム位置の情報は第1版では変数情報のセクションにおかれていたが，第2版ではファイル情報のセクションに移動している．

ダブリンコアの規定は，メタデータの標準的記述法として広く用いられつつあり，国際ワークショップも行われている．これはかなり抽象的な水準での書誌情報の記述を指定している．提案されているもので最も基本となるのは，表1に示した15の要素タイプからなる要素集合（element set）である．表中の要素タイプの日本語訳は杉本(2003)による．さらに，より詳細な記述が必要とされる場合のために，追加要素と修飾子（refinement）が用意されている．またダブリンコアの要素定義それ自体は抽象的なものであるが，より具体的にXMLにおいて記述する場合の推奨方法（RDFとよばれるメタデータ宣言として定義する）も提案されている．

また，DDIでは規定するタグ（項目）とダブリンコアの要素セットの対応関係が示されている（表2）．DDIコードブックの仕様には幾分冗長なところがあるためもあって，記述されたコードブックは長大なものとなりやすい．DDIによってサンプ

表 2 DDI とダブリンコア（DC）要素との対応

番号	DC 要素	DDI セクション	DDI タグ名	
1	title	1.1.1.1	titl	文書のタイトル
2	creator	1.1.2.1	AuthEnty	著作者名（組織名）
3	subject	2.2.1.1	keyword	キーワード
		2.2.1.2	topcClas	トピックの分類
4	description	2.2.2	abstract	要旨
5	publisher	1.1.3.1	producer	DDI の現在の規定では電子版の作成者
6	contributor	1.1.2.2	othId	他の関与者
7	date	1.1.3.3	prodDate	作成日付（電子版の）
8	type	対応なし		DC type の Text.x-Codebook を示唆
9	format	対応なし		DC Format の “text/xml” を示唆.
10	identifier			DC Identifier による URL 指定を示唆.
				または Document Description citation element の IDNo
11	source	対応なし		DDI コードブックが原典の場合は対応なし.
				他の資料から作成された場合はその書誌情報
12	language		xml:lang 属性	
13	relation	(1.4)	docSrc	文書ソース（部分的に）
14	coverage	2.2.3.1	timePrd	時間的期間
		2.2.3.2	collDate	収集日付
		2.2.3.3	nation	国
		2.2.3.4	geogCover	地理的範囲
15	rights	1.1.3.2	copyright	著作権

Jerome McDonough（U.C. Berkeley Library Systems Office）による 1998 年 7 月の提案
`http://www.icpsr.umich.edu/DDI/users/dtd/dc.html`

ルとして公開されていた大規模な社会調査のコードブック記述例は，カテゴリー集計値を含んでいることもあり数百 KB から 1MB 程度の大きさとなっている．図 1 に DDI フォーマットによって記述されたコードブックの先頭部分を示す．ここで用いられている XML の文法については後述する．

2.2 電子メディアを用いた学習にかかわる標準仕様プロジェクト

入試データの管理にとってもう一つ関係の深いものは，教材と学習者にかかわる電子情報の標準化のプロジェクトである．このようなプロジェクトは，コンピュータを教育のツールとして用いることが重要となったために，複数の組織によって盛んに行われつつある．このなかで，IMS（Instructional Management Systems）は主に米国の大手コンピュータメーカ，ソフトウェアベンダー，大学，政府機関，テスト実施機関などが参加する非営利の組織であり，コンピュータを用いた教育のための各種のデータ表現の仕様を提案している．教材についてのメタデータ，教材とテストの記述，学習者の記述などについての詳細な XML ベースでのフォーマットがすでに作成されている．ETS の Russel Almond は 2001 年の IMPS（国際計量心理学会）大会（大阪）で，IMS の活動，特に設問とテストの仕様策定プロジェクトの現況について報告している．この仕様の詳細については，IMS のウェブサイトから資料を得ることができる．また，コンファレンス論文集（Yanai, Okada, Shigemasu, Kano, & Meulman, 2003）においては，学習とテスト実施のためのシステムモデルについての提案を示している．このモデルはテストの実施を 4 つのサブプロセス（出題選択，問題呈示，反応，スコア集計）からなるとみなし，様々な種類のテストの枠組みをサブプロセスの特徴の違いによって記述しようとしている （Almond, Steinberg, & Mislevy, 2003）．

ISO（International Organization for Standardization）や IEEE Computer Society などの組織において，学習システムについて検討する委員会が設定されている．ISO の委員会は ISO/ETC JTC1 SC36 という名称であり，各種の規格について活

```
<?xml version="1.0"?>
<?xml-stylesheet href="../XSL/codebook.xsl" type="text/xsl"?>
<?cocoon-process type="xslt"?>
<DOCTYPE codeBook SYSTEM
   "http://www.icpsr.umich.edu/DDI/codebook/samples/Version1.dtd">
<codeBook>
  <docDscr>
    <citation>
      <titlStmt>
        <titl>EURO-BAROMETER 10 -- OCTOBER - NOVEMBER, 1978 </titl>
        <subTitl>NATIONAL PRIORITIES AND THE INSTITUTIONS OF EUROPE </subTitl>
        <IDNo agency="ICPSR">7728</IDNo>
      </titlStmt>
      <rspStmt>
         <AuthEnty  affiliation=
           "special adviser to the Commission of the European Communities">
           RABIER, JACQUES-RENE </AuthEnty>
         <AuthEnty affiliation=
           "Center for Political Studies, the University of Michigan">
           INGLEHART, RONALD</AuthEnty>
         <othId>  </othId>
      </rspStmt>
        ...
```

図 1　DDI フォーマットによるコードブック記述例
「ユーロバロメータ 1978」の先頭部分（DDI サイトの公開例，一部表示のために改行と空白を追加）

動を行っている．分野としては協調学習支援技術，学習者情報，学習オブジェクトのメタデータ，学習管理システムなどが検討の対象となっている．これらの情報にかかわるリンクは国内の ALIC（先進学習基盤協議会）のサイトからたどることができる．IEEE では Learning Techonology Standards Committee（LTSC）が関係する組織である．

これらのいずれの仕様の記述においても XML は情報記述の基盤として重要な役割を果たしている．

2.3 Text Encoding Initiative（TEI）プロジェクト

TEI は人文系の文書の電子化にかかわる研究者，図書館，関連学会などによって，SGML を用いた文書の構造化のための標準を作成するプロジェクトとして開始された．いくつかの人文系の学会の支援を受けてヨーロッパ，北アメリカ，アジアからの参加者を得て，最初の会合が 1987 年に Vassar College で行われた．1989 年には 50 名余りの研究者が参加するプロジェクトとなり，その後プロジェクトの規模が拡大した．1994 年には，SGML を用いた最初の公式なガイドライン（P3）が，文書で出版された．現在では TEI のガイドラインは 1300 ページ，600 要素を超える膨大な仕様に成長している．1998 年には DTD のモジュラー化が図られた．1999 年にはバージニア大学とノルウェーのベルゲン大学が TEI の継続的なサポートのために国際的な組織（TEI コンソーシアム）を設立することを提案し，TEI，ブラウン大，オックスフォード大に加えてこれら 2 つの大学がホストとなった．現在の TEI のガイドラインは TEI P4（Sperberg-McQueen & Burnard, 2002）となっており，XML に対応したものとなっている．文書構造の定義が DTD および実験的ではあるが Relax NG（後述）を用いた記述によって TEI のウェブサイトから提供されている．TEI の仕様は，散文や詩などのテキストのみならず，図，表，式，木構造やグラフ構造などの表現法をも定め極めて広範なものとなっている．

TEI プロジェクトは極めて野心的なものであるが，その評価には批判的なものもある．電子図書館連盟（Digital Libraries Federation）の Daniel Greenstein（2001）は上述の DDI についての NSF の評価レポートのなかで，TEI はそのサポート範

囲を広げすぎたため，あまりに一般的で拡張可能な記述法となってしまい，本来の目的である電子的な文書の交換を有意味に行えなくなってしまっており，DDI は同じ轍を踏まぬようデータアーカイブの管理に有益な仕様に徹するべきだと指摘している．

3 XML の文法と特徴

XML は Web ページの記述言語である HTML と同様に SGML（Standard Generalized Markup Language）から派生したマークアップ言語である．HTML と同様に，SGML よりはるかに簡約化された仕様を持っており処理が容易になった．HTML が文書の視覚化を指定するものであるのに対し，XML は文書とデータをより厳密な文法によって構造化することにより，ソフトウェア間でのデータ交換に用いようと意図されている．最も特徴的なことは，HTML では利用可能なタグにはあらかじめその役割（強調，改行など）が指定されているが，XML では利用可能な文字と記述文法が設定されているだけであり，特定のタグの役割が定まっていないことにある．特定のタグをどのように解釈するかは，利用者（のグループ）に委ねられている．

このためにどのようなタグセットを設定するかについて，様々な提案がなされている．データ解析とその周辺に関連するものをとっても，数式を表現するための MathML（W3C）と OpenMath（OpenMath Society），また統計データと分析結果全般については上述の DandD やソフトウェアベンダーのグループによる XMLA（XML for Analysis），データマイニング系ベンダーの組織（Data Mining Group）による予測モデル記述のための仕様 PMML（Prediction Model Markup Language）などがある．

これらのうち XMLA（xmla.org, 2002）は，SOAP とよばれる XML によるデータ交換プロトコル（XML 自体にはデータ型の指定の機能はないが，タグによって整数，浮動小数点数などのデータ型を明示的に指定する）を用いることにより，データ分析用サーバと分析者が利用するクライアントコンピュータとの間の通信に必要と考えられるデータ交換を記述するための広範な仕様を定めており，今後影響力を持つと予想される．

行番号

```
1    <?xml version="1.0"?>
2    <xmlsample>
3    <tag1 attr1a="1A" attr1b="2B">
4    Sample text1<tag2 attr2a="2A" />
5    Another text
6    </tag1>
7    </xmlsample>
```

図 2 簡単な XML 文書

上に紹介したように，XML を用いた様々な仕様がそれぞれの分野で提案されておりどれが広く用いられるようになるか，現在の段階では判然としないものも多い．しかし，XML の基本的な文法は既に決定されており，コンピュータによる読み込みは共通に行うことができる．問題は，タグの意味を適切に解釈して行うべき処理が仕様によって異なることであり，それぞれに応じたソフトウェアを用意する必要がある．

ここで，まず簡単な XML 文書の例を示して基本的文法についての説明を行う．図 2 は簡単な XML 文書の例である．先頭部分の数字は表示のための行番号であり，実際の XML 文書には含まれない．

図 2 の 1 行目は XML 宣言と呼ばれるものであり，この文書が XML の規約（第 1 版）に従って記述されていることを示す．ここでは特殊なタグ<?と?>が用いられているが，この記号を用いる記述が XML 宣言のほかにもあり，処理命令（PI:processing instruction）と呼ばれる．これは XML 文書を処理するプログラムへの指示を記述する．上述の例では処理命令はないが，

```
<?xml-stylesheet href="./XSL/codebook.xsl"
      type="text/xsl"?>
```

のような行が 1 行目の後に記述されていると，スタイルシート（表示用の処理プログラム）への指示を表すことになる．XSL に対応したプログラム（ブラウザ）でこの文書を処理すると，href に指定したスタイルシート（XSL）のプログラムを利用して表示用 HTML への変換を行う．

2 行目以降が，データの中身を表す．不等号で開始されるタグは必ず終了タグと対になっていなければならない．上の例では 2 行目の<xmlsample>が一番外側の開始タグであり，これに対応する終了

タグは7行目にある．この子となる要素は<tag1>以降6行目の</tag1>にいたる部分となる．ただしタグを用いる際，XML，xml などの文字で始まるもの（全ての大文字・小文字の組み合わせを含む）は，システムが予約しており，利用者が固有の意味を持たせることはできない．4行目の Sample text1 は解析対象文字データ（PCDATA，parsed character data）と呼ばれる部分で，HTML での表示文字の部分に相当する．解析対象文字や属性値の部分において，不等号・引用符など特殊な機能を持つ文字を表示などのために記述する際には，文法上の混乱をさけるため>，' などのように&（アンパサンド）記号で始まる表記法（実体参照 entity reference）を用いなければならない．厳密には改行や空白の文字コードが前後にはいるが，ここでは無視する．3行目の開始タグ内に attr1a="1A" attr1b="2B" と指定されているものは，属性（attribute）と呼ばれ，ノードの性質を記述するために用いられる．ここで attr1a が属性であり，"1A"がそれに対応する属性値である．

原則としてタグは開始と終了が対となっているが，子要素を持たないタグは，4行目の tag2 のように開始タグの末尾の不等号の直前に/をおくことによって，開始タグと終了タグを兼ねていることを示せる．

XML の文法において最も複雑な部分は，SGML から引き継いだ文書型定義の記述（data type definition，DTD）である．文書型定義においては，XML 文書がどのような要素（タグ）を木構造として持ちうるかおよび持たなければならないか，また各要素がどのような属性を持ちうるかおよび持たねばならないかなどを指定する．ここでは詳細についてはふれないが，文書型定義の文法は他の XML の部分とかなり異なり，処理プログラムを自ら作成する際には煩雑さを増やすことになる．最近では XML Scheme や Relax NG などの XML の文法を採用した構造定義の仕様が提案されている．DTD は文書の要素関係の構造と属性についてのみを指定し，子要素のデータ型（整数，浮動小数点数，文字）の指定についての規約はない．より最近に提案された文書構造の定義法では，データ型についての情報も詳細に指定することが可能になっており，XML 文書を利用する C++や Java などのプログラムを効率的に自動生成しうるように工夫されている．

現在，複雑な動作指定や入力パラメータを必要とする各種の応用プログラム（統計ソフトウェアや表計算ソフトウェア等）は，多くの場合独自の入力のための文法を持ち，それに従って動作命令やパラメータを指定している．XML それ自体には，繰り返しなどの動的な機能が備わっているわけではないが，これらの処理を指定する言語の文法を記述するのに十分な構文上の表現力を持っている．XML を用いて処理言語を記述することにより，応用ソフトウェアの入力部分を汎用的な XML パーザに置き換えることができるので，動作命令の記述やパラーメタ指定の標準的方法として XML の利用が増えると思われる．既に GNU プロジェクトにおけるいくつかの応用プログラム（表計算，データ視覚化ソフトウェアなど）では，入力データを XML によって指定することが標準となっている．また，PC 上の各種のオフィスソフトウェアなどでも，データ保存と交換のための標準形式として採用される方向にある．

4 XML 関連技術

高度に構造化されたデータの標準的な記法への要求が広く存在しているために，XML は提案されてより急速に応用範囲を広げている．それとともに，各種の拡張および関連技術の仕様も次々と現れている．技術動向の将来を正確に予見することは難しいが，これらのうち統計データとメタデータの記述と利用に関係が深く今後も重要性を増すと思われる技術には次のようなものがある．

(1) 応用プログラムインタフェース：Java や C などで記述された応用プログラムから XML 文書を利用するための標準的なアプリケーションプログラムインタフェース（呼び出し機能の仕様）が複数種類定められている．現在のところ，DOM および SAX と呼ばれる2種類のものが標準的である．DOM（Document Object Model）は XML 文書全体を読み込んだ上で検索，編集などを行うための仕様であり，W3C によって規約が定められている（W3C DOM Working Group，2003）．1998年にレベル1，2000年にレベル2が示され，2003年9月現在のところ，レベル3の仕様の決定作業

が行われつつある．

もう一方のSAX（Simple API for XML）は，W3Cではなく利用者のグループによって自発的に設定された仕様である（SAXプロジェクト；Brownell, 2002）．最初の版はDavid Megginsonらの専門家によって1998年に提案された．XML文書を全て読むのではなく，先頭から逐次読み込み，指定された条件が発生した場合に，それを応用プログラムに教えるという方式をとる．このため，文書全体を参照する操作には不適切であるが，大量の文書を高速に処理することが可能になる．DOMと同様に現在ではJava，C++，および多くのスクリプト言語に取り入れられている．

(2) XML変換プログラム：上述のDOMやSAXは汎用のプログラミング言語からXML文書を操作するための仕様である．このほかにもXML文書を操作，変形，表示するための専用の言語仕様が提案され，またそれを実行するソフトウェアが実現している．このなかで最も一般的であるのはXSL（The Extensible Stylesheet Language Family）である．これはXML文書を入力し，変形あるいは表示するための仕様である．XSLは次の3つの部分から構成されている．ひとつはXML文書を変形するための言語であるXSLT（XSL Transformation）（Clark ed., 1999），2番目はXMLの特定の部分を指定するための表現言語XPath（XML Path Language）であり（Clark & DeRose, 1999），これはXMLTによって用いられる．最後のものはXSL-FO（XSL Formatting Objects）と呼ばれ，XML文書を表示するための文法規約である（Adler et al., 2001）．

現在では，XSLTを実現する多くのソフトウェアが流通しており，一部のウェブブラウザではその機能を内蔵しているものもあるが，単体のプログラムとして利用されることも多い．XSLTが最もよく利用されるのは，XML文書のなかから特定の部分を選択して新たなXML文書を作成する場合，もう一つはXML文書の内容を理解しやすい形式で表示するために，HTMLあるいはXHTML文書に変換する場合である．

本稿の後の部分ではPrologによるXML文書の変換例を紹介するが，これはXSLTの機能をほぼ代替し，さらにより多くの操作上の自由度持つ．現在のところXSLTの処理システムは基本的機能の実現を完成し，実行効率の向上など関連機能を整備する段階にあると思われる．商用の製品ではXSLTのための逐次実行を行うデバッガなども開発されており，開発環境の整備が今後進むと思われる．

(3) スキーマ定義言語：XML文書の構造を特定するためには，伝統的にはSGMLより引き継がれたDTDが用いられてきたが，より新しく定められた構造定義用の言語（スキーマ言語とよばれる）が利用されるようになっている．このための代表的なものは，W3CによるXMLスキーマ（XML Scheme）である．これはDTDと異なり，XML本体と同一の文法規約に従って記述される．すなわちXMLスキーマ定義自体が一つのXML文書となる．XMLスキーマは詳細な構造定義を行うことのできる仕様であり，整数型，浮動小数点型，などの通常の基本データ型のほかSQLとの対応を考慮して，日付型，時間型などまでをも基本データ型として指定し，仕様はかなり大規模なものとなっている．XML Schemeに関する解説書としてはvan der Vlist（2000）が詳しい．

より簡潔なスキーマ定義言語としてはRelax NGと呼ばれるものがある．これは2002年にXML利用団体の組織であるOASISによって推奨仕様として採用になった．このRelax NGは日本国内で村田真らによって開発されたRelaxとよばれるスキーマ定義言語と，XML仕様制定の中心人物の一人であるJames Clarkによって提案されたTREXと呼ばれるスキーマ定義言語の両者に起源を持つ（Clark & Murata, 2001）．XML Schemeより小さな仕様であり，各種のサポートソフトウェア（他スキーマとの変換，XML文書の検証などを行うもの）が多く流布している．

これらのスキーマ定義言語によってXML文書の構造を定義することにより，生成されるXML文書が特定の要求を満たしているか否かを検証することが可能になる．またXML文書を処理するプログラムのデータ型を導出することができる．

5 Prologを用いたXML文書の構文解析とコード生成

5.1 XML処理言語としてのProlog

すでで述べたように現状ではXML文書の処理を行うには，Java，C++などの言語によってライブラリ関数を利用して応用プログラムを作成するか，XSLを利用するのが一般的である．しかし，前者の方法では少なくとも現在のところライブラリ関数の機能がプログラミング言語のデータ構造の細部に密着したものであるために，抽象度の高い操作を簡潔に記述することが難しく，またプログラミングが煩雑なものになりやすい．また，プログラムはコンパイラによって実行コードに変換される必要があるため，XML文書を対話的な処理の対象とし，構造についての問い合わせを行い，その結果を見て次の処理を行うという方法をとりにくい．一方，XSLを用いる方法はより簡単であるが，汎用プログラミング言語ではないために少なくとも現状では機能が制限される．

定型的な業務から得られるデータの場合には，高速にデータを処理しうるプログラムをコンパイラ言語によってあらかじめ作成しておく方法は有効である．しかし，研究活動で得られるデータなど，XML文書の構造が多様であり，また要求される処理も様々の場合には，固定したプログラムによって処理を完結することが難しく，対話的に結果を得て次の処理に進むことが望ましい．このような方向に向けての発展は，XML処理に限ったことではなく，データ解析ソフトウェア全般についての過去20年あまりの発展動向にも見て取れる．実際，Perl，Python，Rubyなどの多くのユーザを持つスクリプト言語にはXML用パーザライブラリの機能が組み込まれ，スクリプトからそれらの機能を利用することができる．またオメガプロジェクト（Temple Lang, 2001；Meyer, Leisch, Hothorn, & Hornik, 2002），は英語圏とヨーロッパの計算機統計の研究者によるデータ解析ソフトウェア（特にR，S，およびLisp-Statシステム）とWeb技術を連結するソフトウェア群を開発しようとする試みであるが，統計ソフトウェアからXML文書を利用可能とするモジュールの開発が行われている．

Prologはすでに30年近い歴史のあるコンピュータ言語であるが，一階の述語論理に論理的基盤を置く特殊な性質をもっている．1980年代に日本の通産省（当時）の第5世代コンピュータ計画の核言語として採用され，1990年代には多くの優れた教科書と処理系が現れたものの，実用言語として一般的にはならなかった．しかしながら，木構造やグラフ構造の探索と，集合論的な操作，関係型データモデルの取り扱いなどに優れた特徴をもっており，特にXML処理においてはその意味論的な面での近さから処理言語としての長所を多く持っている．またXSLTと比較しても，汎用言語としての強さを持ちながら，XSLT以上に簡潔な処理命令の記述が可能である．XSLTについては開発環境が急速に整備されつつあり，HTMLなどの出力を対話的にデザインできるソフトウェアも販売されているが，データについての計算やXML文書以外の入出力を多用する処理の場合には，汎用言語を用いる有利さがある．またPrologの処理効率はCなどを用いる場合と比べ高いとはいえないが，長年の研究の蓄積により処理系の巧妙な最適化が実現されており，初期のものと比べると処理速度の改善が著しい．さらに処理系の多くにはデバッガが整備されており，逐次的に実行結果を対応するソースプログラムの箇所を追いながら追跡することも可能である．Prologの実行においては，大量のメモリを必要とすることが多いが，最近のPC性能の急速な発展によって，少なくとも中規模の問題については実行効率上の困難は除かれた．今後64ビットアドレスを可能とするPCが一般化すれば，メモリ需要によるソフトウェア上の制約はコストを別とすれば技術的には取り除かれる．

一般的にある課題についてどのような処理方法が優れているか，特に使いやすさを議論することは分析者の知識に依存するため難しいが，少なくとも現在の時点では，Prolog言語はXMLで記述された情報を研究活動で利用するための有力な方法の一つであり，十分な性能を実用的に持つ．ここではSICStus Prolog（Swedish Institute of Computer Science, 2003），と呼ばれるProlog処理系を用いる．他にも無償で公開されているものを含め多くのProlog処理系が利用可能であるが，製品としての信頼性が高く処理が速いこと，メモリ管理が柔軟

で大規模なデータに適応しうること，多バイトコードへの対応が整っており，EUC（Enhanced Unix Code）と UTF8 で記述されたファイルを処理できるなどの特徴がある．日本語 EUC は Unix ベンダーの規約として定義されており（UNIX インターナショナル, 1992），UNIX や Linux システムで用いられている．UTF8 は Unicode の表現方法の一つである（Unicode Consortium & Aliprand, 2003；JIS X0221-1:2001；ISO/IEC 10646-1:2000）．多くの日本語の文字はプログラム上では 2 バイトの符号なし整数として表現されるが UTF8 を用いるとファイル上では文字の先頭バイトなどの認識のために冗長になり，3 バイトで表現される．しかし，SICStus Prolog では利用者が入出力関数を改造することにより，他の多バイトコードにも対応が容易になっており，ここでの処理例は Windows 系 OS 上でシフト JIS コードで記述されたファイルを用いて行った．

文字コードの取り扱いについては，すべての日本語表現を Unicode へ統一するほうが一貫性はとれる．しかし，現状では Unicode だけで実務的な処理を行うことは他の既存のソフトウェアとの関連から困難であり，それぞれのシステムの固有の多バイトコードの表現方法（Windows ならばシフト JIS，Unix/Linux 系であれば EUC）を入出力に用いる必要がある．シフト JIS コードへの対応モジュールは，Windows のライブラリ関数を用いて入出力関数に変更を加えることにより自作した．Prolog 処理系の内部では Unicode 文字列として表現している．ただし EUC を入出力用のコードとして設定した場合には，SICStus Prolog の現在の版（Ver. 3.11.0）では内部表現は EUC コードのままであり，Unicode への変換を行っていない．

各種コードブックの国際的な交換を考えるなら，現状で多言語に対応した文字コード体系として主流になりそうなものは Unicode でありこれに基づくことが望ましいと思われる．また公式の規約として，XML 処理系がサポートすることが義務付けられている文字コードは Unicode（UTF8 および UTF16 とよばれる符号化方式）である．日本国内の情報科学の研究者には，漢字コードの国際共有化などをめぐって Unicode に批判的な論調があったが（和田他, 1998），商用システムにおいては複数の言語の混在する文書の表現方法として Unicode が主流となっている．日本語コードの既存の文書を取り扱う際に問題になるであろうと思われることは，シフト JIS コードから Unicode への変換が一通りではないため，同一のコードが変換後に異なるコードとして保存されてしまう可能性のあることである．Unicode を含む東アジア系言語のための文字コードについては Lunde（2002）が包括的な解説を行っている．

5.2 Prolog の基本機能

Prolog 言語の理論的な背景とは別に，実用プログラミング言語としての最大の特徴は，パターンマッチング（論理プログラミングの用語で単一化（unification））の機能とバックトラッキング（後戻り）の機能が言語自体に組み込まれていることにある．現在，商用の汎用言語で，これらの機能があるものは他にない．これらを用いることにより，集合論的計算（和集合，積集合の計算，部分集合の生成）や，組合せ論計算（順列や組合せの生成），対称群の計算（群の積，部分群の生成など）を柔軟に行なえる．現在利用可能な Prolog によるプログラミングの標準的な教科書としては，Bratko (2001), Clocksin & Mellish (2003), Flach (1994), Sterling & Shapiro (1994)，などがある．また，特色のあるプログラミング例を示しているものとしては，有限集合上の位相構造や有限群についての計算法を示している 飯高 (1990)，回路設計や離散フーリエ変換なども扱っている Clocksin (1997) がある．ただし，最近の Prolog 処理系はコーディングスタイルに依存する巧妙な最適化を行うため，古い教科書に示されているプログラム例では十分な性能を実現できない場合がある．

Prolog は論理的推論メカニズムによる記号処理のために設計された言語であるため，多くの文献に示されている応用は定理の自動証明や自然言語の構文解析，エキスパートシステムの設計などに関係するものが多い．1980 年代の終わりから 1990 年代にかけては，日本の計算機統計の研究者によってデータ解析を支援するためのエキスパートシステムがいくつか提案されているが，データ解析を念頭においたプログラムの解説はほとんど存在しない．しかし，その記号処理能力は複雑な統計モデル操作を支援し，データにまつわる様々な関連情報を統合するために有益であり，特に社会科学

における研究ツールとしての大きな潜在的能力を持っている．

Prolog（を含む論理型言語）が他の計算機言語と著しく異なっている点の1つは，「変数」の概念である．Prologにおける「変数」とは，どのようなものにも変形し得るものを意味し，本来の数学における「変数」の概念にむしろ近い．通常利用されているPrologの文法（Edinburgh syntax）では，大文字または下線で始まる記号は変数を表す．

Prologにおける述語（真偽を値とする関数）は

```
a(p1).  a(p2).
a(p,q1). a(p,q2). a(p,q,r).
```

のような形式（事実 fact）か，あるいは

```
b(X) :- c(X),d(p,X).
b(X) :- e(X),d(q,X).
```

などの形式（規則，rule）をとる．事実は真である述語を指定するものである．引数の異なる述語はPrologにおいては別の述語とみなされるため，述語はその名称と引数の数の組によってa/1（引数が1個のaという名称の述語），a/2（引数が2個のもの）などの様に識別する．上の例は述語a/1は，引数がp1またはp2であるとき真であることを示している．また，a/2は引数の組p,q1およびp,q2について真となる．

一方，規則は:-の右辺が，左辺が成立するための前提条件であることを示している．上の例における1行目の規則は，Xがc/1を真とする引数であり，しかもd/2がp,Xの組について真であるならb/1は真となることを表す．

事実または規則が複数記述されているならば，それらのうちいずれかが成立すれば，述語は真と判断される．ピリオドで終わる記述の単位（事実または規則）は，一般的に節（clause）と呼ばれる．事実は前提なしに成立している規則とみなすこともできる．すなわち述語a(p1).は

```
a(p1) :- true.
```

と同じ意味を持つ．ここでtrueは常に真となる組み込み述語である．また，規則の右辺で述語がカンマ(,)で結ばれているのは，それらの全てが成立する場合に，述語が真となることを示している．

Prologプログラムの実行は，問い合わせの真偽判定のための推論演算として処理される．ここで，述語a/1についての問い合わせを行うと，次のような解が得られる．下の実行例において，先頭の表示| ?-はPrologシステムが問い合わせ述語の入力を待っている状態を示すプロンプトである．1行目は利用者がa(A).と入力し，リターンキーを押した場合を示している．2行目以降にあるセミコロンは，利用者が他の解をシステムに要求するためにタイプするものである．2行目，3行目などに表示されている?は別解を求めるか否かの判断をPrologが求めていることを表すプロンプトであり，この記号の左側（A = p1など）はPrologシステムによる解の表示である．ここでセミコロンを入力せず，リターンのみを押すと解の探索は中止される．以下のプログラム実行の応答例では%記号の右側はコメントをあらわす．

```
| ?- a(A).
A = p1 ? ;  % A=p1 が一つの解．セミコロンを入
              力すると他の解を探す
A = p2 ? ;  % A=p2 が別解．再びセミコロンを入
              力すると別解を探す
no          % 別解がないのでシステムが no と表示

| ?- a(A,B). % a/2 についての問い合わせ
A = p,       % Prolog システムは A=p，B=q1 が
               解であることを見つけた
B = q1 ? ;   % セミコロンを入力し別解を求める
A = p,       % A=p，B=q2 が別解である
B = q2 ? ;
no
```

さらに次の事実の宣言

```
c(r1).  d(p,r1).  d(q,u1).  e(u1).
```

をプログラムに追加して記述し，b/1についての問い合わせを行うと，以下の2つの解が得られる．最初の解はc(r1)とd(p,r1)がともに成立するため，bの一番目の規則の定義から導かれる．2番目の解はe(u1)とd(q,u1)がともに成立することから，2番目のルールによって導かれる．

```
| ?- b(A).  % b/1 の問い合わせ
A = r1 ? ;  % A=r1 が一つの解である．次の解を
              探す．
A = u1 ? ;  % A=u1 がもう一つの解である．
no          % 他に解はない
```

```
%  述語 member の定義

member(H,[H|_]).
member(I,[H|T]):- member(I,T).

------------------------------
%  リストの分割表現.

| ?-  [a,b,c,d] = [X|Y].
X = a,
Y = [b,c,d] ?
yes
| ?- member(a,[a,b,c]).    %   要素に含まれているかの判定
yes
| ?- member(a,[p,q,r]).
no
| ?- member(X,[p,q]).      %   具体化された値を次々と返す.
X = p ? ;
X = q ? ;
no
```

図 3　member/2 の定義と実行例

5.3　リストの表現

Prologに限らず, 複雑な対象を計算機で取り扱う場合に利用されるデータ構造にリストと呼ばれるものがある．Prologの文法では, リストは[a,b,c,d]の様に表現され，下に示す内的構造を持つ．

```
. --- . --- . --- . --- []
|     |     |     |
a     b     c     d
```

リストは複雑な入れ子構造を柔軟に表現できるが, 要素の入れ換えや末尾への新たな要素の付加操作が繁雑になる．PrologによるXML処理にはいくつか方法があるが，最も明快なものは文書の文字列を文字コードのリストとして表現し，これを構文解析の対象とする．図3にPrologによる簡単なリスト操作の例を示す．図の上部には，第1引数が第2引数に指定されたリストの要素と単一化可能か否かを判定する述語memberの定義を示す．縦棒を用いた表記[X|Y]は，リストの最も上位のノードの左の枝の要素をXと置き，右の枝をYと置くことを表す．リストの構造上の特徴からYは常にリストとなるが，Xは一般的にはリストとはならない．図3の下段の最初には，リストについての簡単な問い合わせの例を示した．組み込み述語 = は単一化が可能であるか否かの判別を行う．また引数は通常の数学記号の表記と同様に記号の左右に置く．つまりX = Yは, =(X,Y)と同等である．同一の記号は単一化可能であるが，異なる記号は単一化可能ではないのでa = aは真であり，a = bは偽である．また変数は任意の記号と単一化可能であるので, a = Xは真である．単一化（パターンマッチング）が可能であれば, 引数中に現れる変数は，パターンマッチングの条件を満たしつつ最も一般性を保った形式に変形され, 具体的に特定されない部分は変数のまま残される．a = Xが評価された後は，変数Xはaと具体化される．また, a(X,Y) = a(Z,Z)が評価された後には，X,Y,Zは同一の変数を表すよう拘束される．

第2引数のリスト中に現れる下線は無名変数（述語中の他の箇所で引用されないため，名前をつける必要のない変数）を表す．2行目は再帰的な定義であり，第2引数から第1要素を除いた部分リストTにおいて，述語memberが真となるならば，先頭要素を加えたリスト[H|T]においても述語が真となることを示す．これらを組み合わせることにより，リスト要素に第1引数と単一化可能な要素があるか否かの判定を行える．

また，述語memberの機能は双方向的である．第1引数に具体的な値が指定されると，それが実際に第2引数のリストに含まれているかを判断し，ま

```
file_to_codes(Codes) :-
        get_code(X0),
        (X0 == -1 -> Codes=[]
        ; Codes=[X0|Codes1], file_to_codes(Codes1)
        ).
```

図 4 文字コードの読み込み
（構文 X -> Y;Z は if X then Y else Z の意味を示す.）

たリスト要素が変数を含んでいるとその変数が第1要素と単一化される．一方，第1引数に変数が指定されると，リスト要素の値と単一化される．

5.4 Prolog 上の XML パーザ

Prolog が XML 処理に適していることは，Prolog 言語の研究者によっても認識されており，いくつかの処理系では XML パーザのライブラリが整備されている．また無償の XML パーザもいくつか Web 上で公開されている．無償で公開されているもののなかで良く整備されているものには，Jan Wielemaker によって開発された SWI-Prolog 上の SGML パーザ（Wielemaker, 2003）と Binding Time Ltd. の John Fletcher によるパーザ（Fletcher, 2003）がある．また，Prolog 上での HTTP による通信と HTML/XML 処理をサポートするソフトウェアとしては，マドリッド工科大で開発された PiLLoW ライブラリ（Cabeza & Hermenegildo, 2001）がある．Fletcher による XML パーザと PiLLoW ライブラリとはともに SICStus Prolog の非サポートライブラリとして採用されている．また，前述の TEI プロジェクトにおいて中心的な役割を果たし，現在 W3C のメンバーである Sperberg-McQueen（2003）は，XML Scheme で定義された構造と XML 文書との整合性の検証に Prolog が有効であることを指摘している．

前述の RDF はセマンティック Web 実現に関わる技術の一つであり，文書の意味の要約を3つの要素（主語 subject，述語 predicate，目的語 object）の組として記述する．主語や述語となりうるのは URI（Universal Resource Identifier，インターネット上のリソースの識別子）であり，目的語は URI またはリテラル（文字列）である．このような形式のデータは Prolog による取り扱いに向いており，柔軟な推論を簡潔に記述することができる．SWI-Prolog には RDF パーザがパッケージとして整備されており，またブラウザ開発のための mozilla プロジェクトでは，RDF 推論のためのバックエンドとして Prolog を用いる試みがある．

SWI-Prolog の SGML パーザは極めて高速であり，XML のみでなく SGML 文書を一般的に処理することができる．さらに DTD による文書構造の検証もサポートしている．このプログラムは Prolog による処理部分と C 言語によって記述された XML 文書の入力関数から構成されており，テキストは長いアトム（Prolog の記号識別子）として表現されるため，文書を文字コードとして表現する方法にくらべより少ないメモリ消費で処理が可能である．しかしながら現在のところ多バイトの文字コードテキストへの対応が考慮されておらず，かなりプログラムが複雑であるため改造が難しい．ここではより単純な構造を持っている Fletcher によるパーザを用いた．

Fletcher のパーザはすべて Prolog で記述されており，文字コードのリストとして表現された XML 文書を構文解析し，Prolog の内部データ構造を変換する．ただしこれも多バイトの文字コードの入力を許さないが，比較的簡単なプログラムの変更によって日本語への対応を実現できる．テキストから文字コードリストへの変換はパーザの機能とは別に，利用者が行う．図4は標準入力を文字コードのリストへ変換し引数 Codes に返す簡単なプログラム例である．

述語 get_code/1 は標準入力から1文字分のデータを入力し，文字コードを整数として引数 X0 に返す．ファイルの終わりが検出された場合には，−1 が引数の値となる．上記のプログラムは，再帰的にこの文字コードをリストの末尾に順次付加する．一般的に再帰を用いたプログラミングは効率が悪く，明示的な大規模なデータ対象とする場合には繰り返しを用いるべきとされているが，最近の Pro-

```
xml( [version="1.0"],
        [
        element( xmlsample, [],
                [
                element( tag1,  [attr1a="1A", attr1b="2B"],
                        [
                        pcdata("\nSample text1"),
                        element( tag2,  [attr2a="2A"],
                                [] ),
                        pcdata("\nAnother text\n")
                        ] )
                ] )
        ] ).
```

図 5 XML 文書を変換して得られた Prolog データ

log 処理系においては単純な末尾再帰はコンパイラによって効率よく処理される．上記のプログラムでは，リスト構造を逐次作成するため，プログラム領域の消費は避けられないが，再帰処理自体はかなり効率的であり，単純な算術計算では通常の配列を用いた場合と遜色ない速度を実現している．むしろ，上述のプログラムを実行すると作業領域をメモリ上に確保するための処理に時間が消費されている．

文字コードは整数として表現されるため，最低4バイトの領域が必要である．さらに上述の方法ではリストとしての構造を作成するための領域が必要であり，さらに再帰処理を行うためのプログラム領域も必要とされる．Windows 上の SICStus Prolog により，ASCII コードで記述された 717KB の XML データ（DDI のコードブック）を入力し Fletcher のパーザによって変換するためには，約 18MB のプログラム領域（ヒープとスタックと呼ばれる部分）を必要とし，処理の経過時間は Pentium3（866 MHz）の PC で約 6 秒であった．ただし，この値は作業を実行するために必要な領域であり，構文解析後のデータを保持するために消費されるメモリ領域は約 2.5MB であった．

Fletcher による XML パーザと PiLLow ライブラリは，いずれも XML の構文解析を行うために，Prolog に用意されている DCG（Definite Clause Grammer，確定節文法）と呼ばれる文法の記述法を用いている．これは文法要素（XML の場合にはタグ要素など）がどのような下位要素から構成されるかを定義すると，それに従った構文解析のプログラムが自動的に生成されるものである．また，文法規則とともにその際に行うべき計算や判断などの処理も付随して記述することができる．これを用いて XML の文法を定義し，XML 文書の木構造を Prolog の内部データへと変換する．図 5 は図 2 に示した XML 文書を Fletcher のパーザによって Prolog の内部構造へ変換したものである．SWI-Prolog のパーザや PiLLoW ライブラリによっても，細部は多少異なるが類似の構造が生成される．

ひとたび XML 文書が Prolog のデータとして表現されたならば，言語に備えられたパターンマッチングの機能を再帰的に適用することにより，柔軟な情報の検索が可能になる．例えば図 6 に示すのは Fletcher によって提供されている `xml_subterm/2` という述語である．第 1 引数に XML 文書から変換された Prolog のデータ構造を定義すると，第 2 引数に部分要素を逐次列挙する．述語が再帰的に定義されているため，列挙はトップレベルの直接の子要素だけでなく，それらの子要素の子孫にあたる要素を全て列挙することができる．

このように Prolog を用いることにより，かなり複雑な処理を簡単なプログラムによって実現することができる．

上と類似のプログラミング方法によって，様々な XML 処理のためのプログラムを作成できる．前述の DDI コードブックについては，次のような応用プログラムを作成することができる．

1. XML 文書のタグに表示用のラベルをそれぞれ定義し，それを用いて表示用の HTML 文書に変換する．
2. 特定のセクションのみを選択して表示する．

```
% xml_subterm/2 の定義

xml_subterm( Term, Term ).
xml_subterm( xml(_Attributes, Content), Term ) :-
        xml_subterm( Content, Term ).
xml_subterm( [H|T], Term ) :-
        ( xml_subterm( H, Term )
        ; xml_subterm( T, Term )
        ).
xml_subterm( element(_Name,_Attributes,Content), Term ) :-
        xml_subterm( Content, Term ).
xml_subterm( namespace(_URI,_Prefix,Content), Term ) :-
        xml_subterm( Content, Term ).

-----------------------------------------
% 利用例

| ?- xmltree(sample1,_T),xml_subterm(_T,element(tag2,A,Sub)).
A = [attr2a=[50,65]],     % [50,65] は"2A"の文字コード
Sub = [] ?                % 下位要素はない
yes
```

図 6 Fletcher による部分要素探索述語とその利用
%より右はコメントを表す．xmltree/2 に XML 文書から得られた木構造を保存しており，_T に木構造を得る．xml_subterm/2 で tag2 というタグ名の部分要素を求める．A は該当要素の属性，Sub はさらに下位の要素となる．

3. 特定の属性を持つタグ要素を列挙する．
4. コードブック中に記述されたカラム情報とラベルを参照し，SAS 用のデータステップを生成する．

5.5 XML パーザを用いたコード生成

Prolog の一つの特徴は，述語が実現しているのは原則としてパターンマッチであり，入力と出力が固定されないことにある．実際に複雑なプログラムを記述する場合には，各種の組み込み関数や手続き的な処理などを用いるため，上の性質は多くの場合成立しないが，簡単な述語ではこの性質を利用できる．

次のような述語 f/2 が定義されているとしよう．

```
f([X,Y],p(X,q(Y))).
```

ここで大文字で記述されているのは変数であり，どのような値にも単一化される．この述語の第 2 引数に，変数を持たない次のようなデータを指定すると，構造中の該当する部分が第 1 引数として得られる．

```
| ?- f(Z, p(a,q(b))).
Z = [a,b] ?
yes
```

逆に，第 1 引数に具体的な値を持つリストを指定すると，第 2 引数の構造中の変数に値が代入された結果が得られる．

```
| ?- f([c,d],Z).
Z = p(c,q(d)) ?
yes
| ?- f([c,d(e)],Z).
Z = p(c,q(d(e))) ?
yes
```

上の例において，第 2 引数にあたる部分が XML 文書から得られたデータ構造であると仮定する．このようなデータを用いて XML 文書を生成するには，DCG によって定義された構文解析プログラムが原則として逆方向にも動作しうることを利用する．つまり，規約に適合した Prolog のデータ構造を指定すると，逆にその構造に対応した XML 文書を生成することができる．これらの機能を利用すると，上のプログラムに類似した方法によって，XML 文書のテンプレートを実現できる．つまり

```
xmltemplates(simple_table,
             [PageTitleString,TableCaptionString,
              HeaderElements,FooterElements,
              TableBodyElements],
 xml( [version="1.0", encoding="Shift_JIS"],
   [
   doctype( html, public( "-//W3C//DTD XHTML 1.1//EN",
           "http://www.w3.org/TR/xhtml11/DTD/xhtml1.dtd" ) ),
   namespace( 'http://www.w3.org/1999/xhtml', "",
     element( html,  ['xml:lang'="ja"],
     [
      element( head, [],
      [
        element( meta,
          ['http-equiv'="Content-Type",
            content="text/html; charset=Shift_JIS"],
          [] ),
        element( title, [], [ pcdata(PageTitleString) ] ),
        element( link,  [rel="stylesheet",
          href="file:///C:/stat/prolog/DNCfiles/default.css",
          type="text/css"],
          [] )
      ] ),
      element( body, [],
        [
          element( table, [],
            [
              element( caption,[],[pcdata(TableCaptionString)]),
              element( thead, [], HeaderElements),
              element( tfoot, [], FooterElements),
              element( tbody, [], TableBodyElements  )
            ] )
        ] )
      ] )
  )
 ] )
).
```

図 7 XHTML 生成用の Prolog テンプレート

XML 文書によって得られた構造の一部を変数に置き換え，さらにそれらの変数リストを別の引数として定義する．このように定義された述語を用いて，XML 文書の一部分の値を任意に変更することができる．

変数にはアトムや数値だけでなく，XML の部分木の構造を指定することも可能であり，構造の合成を簡潔に実現できる．また，この方法によって XML 文書を生成すると，個別にタグを書き出す方法と比較しプログラム上の誤りが生じにくい．大津（2003）は，XHTML 文書のテンプレートを作成し，この機能を用いて得点統計量の一覧表を生成する方法を示している．図 7 は統計量の表を生成するための Prolog によって記述されたテンプレートの例である．ここで PageTitleString 等の変数と，出力すべき内容（文字列や数値データのリスト等）を単一化し，それを DCG パーザの機能を用いて XHTML の文字コードに変換する．図 8 に，これによって得られた XHTML 文書の一部を示す．

```
<?xml version="1.0" encoding="shift_jis"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ja">
 <head>
  <meta http-equiv="Content-Type" content="text/html;
    charset=Shift_JIS" />
  <title>平成 16 年モニター　本試験</title>
  <link rel="stylesheet" href="./default.css" type="text/css" />
 </head>
 <body>
  <table>
   <caption>平成 16 年モニター　本試験</caption>
   <thead>
    <tr>
     <th class="str">科目コード</th>
     <th class="str">総数</th>
     <th class="str">有効件数</th>
     <th class="str">平均</th>
     <th class="str">標本標準偏差</th>
     <th class="str">最小値</th>
     <th class="str">25\%</th>
     <th class="str">中位数</th>
     <th class="str">75\%</th>
     <th class="str">最大値</th>
     <th class="str">得点分布図</th>
    </tr>
   </thead>
   <tfoot />
   <tbody>
    <tr>
     <td class="str">世界史 B</td>
     <td class="int">112</td>
     <td class="int">112</td>
     <td class="flt">61.96</td>
     <td class="flt">19.68</td>
     <td class="int">11</td>
     <td class="int">47</td>
     <td class="int">65</td>
     <td class="int">77</td>
     <td class="int">98</td>
     <td class="other">
      <a href="./pngfiles/file00101.png">得点分布図</a>
     </td>
    </tr>
    ...
```

図 8　生成された XHTML 文書（部分）

6　おわりに

電子化された情報をどのように管理するかは，多くの組織にとって重要性な問題である．現在，電子化される情報は定型的な業務の数値情報から，より多様な文書，視聴覚メディア，抽象化された規則などへと次第に広がりつつある．人間を対象

とする評価は多様な情報がかかわる分野である．情報技術の進展は著しいが，業務や研究等によって得られるデータが必要とされる期間は，ハードウェアや処理ソフトウェアの寿命よりしばしば長い．XMLへの需要が大きいのは，情報技術にかかわる者によってこのような認識が共有されているからである．

NESSTARなどのプロジェクトにおいては，既にオンラインでのデータ配信を可能にするためのメタデータ編集ソフトウェア，サーバソフトウェア，およびクライアントソフトウェアなどが作成され，一般に利用可能となっている．このようなデータ配信のための標準的な基盤技術が整備されると，個別のプロジェクトのために専用のソフトウェアを作成しなくとも，標準仕様に適応したツールを用いることによってプロジェクトを実現することが可能となる．これらのツールは，広く一般にデータを公開する場合だけでなく，組織内部や特定の閉じたグループ内で限定されたデータ共有を行う場合にも有効である．

テストデータの管理は，多くの教育機関などで必要とされており，またこれらのデータに基づく知見を研究者が共有することは，適切な施策を行うためにも重要である．現状で利用可能なデータ配信システムでは，日本語を含む多言語の素材の取り扱いを考慮しているわけではないため，今後，日本語対応等の機能を実現するためには，かなりの努力を払う必要がある．しかし，低コストで利用可能なソフトウェアシステムとデータに基づく知見を共有することには重要な意義がある．

伝統的に「テスト理論」とはテストによって得られたデータの統計的特性についての研究を意味してきたが，本稿で示したテストにまつわる各種の情報の組織化ための方法論も，これに劣らない重要性と技術的な広がりを持っている．今後の技術動向にも注意を払うことが，業務によって得られた情報を十分活用するために必要と思われる．

参考文献

Adler, S., Berglund, A. Caruso, J. Deach, S. Graham, T., Grosso, P., Gutentag, E., Milowski, A., Parnell, S., Richman, J., & Zilles, S. (2001). *Extensible Stylesheet Language (XSL) Version 1.0*, http://www.w3.org/TR/xsl.

ALIC（先進学習基盤協議会） http://www.alic.gr.jp/

Almond, R.G., Steinberg, L.S., and Mislevy, R.J. (2003) A framework for reusing assessment components. In Yanai, H., Okada, A., Shigemasu, K., Kano, Y. and Meulman, J.J. (eds.) (2003). *New Developments in Psychometrics*, Tokyo:Springer-Verlag, pp. 281–288.

Altman, M., Andreev, L., Diggory, M., King, G., Sone, A., Verba, S. Kiskis, D.L. & Krot, M. (2001). A digital library for the dissemination and replication of quantitative social science research, *Social Science Computer Review*, **19**(4), pp. 458–470.

Becker, R.A., Chambers, J.M., Wilks, A.R. (1988). *The New S Language—A Programming Environment for Data Analysis and Graphics*, California: Wadsworth & Brooks.

Bratko, I. (2001). *PROLOG: Programming for Artificial Intelligence 3rd ed.* Boston: Addison-Wesley. 安部憲広訳 (1990). Prologへの入門—PrologとAI, 近代科学社.（原著旧版前半の訳）. 安部憲広，田中和明訳 (1996). AIプログラミング—PrologとAI, 近代科学社.（原著旧版後半の訳）.

Brownell, D. (2002). *SAX2*, O'Reilly.

Buswell, S., Caprotti, O., Carlisle, D.P., Dewar, M.C., Gaetano, M., & Kohlhase, M. (eds.) (2003). *The OpenMath Standard, Version 2.0*, http://www.openmath.org/.

Cabeza, D. & Hermenegild, M. (2001). *The PiLLow Web Programming Library: Reference Manual.* School of Computer Science, Technical University of Madrid.

Carlisle, D., Ion, P., Miner, R., & Poppelier, N. (eds.) (2003). Mathemtical Markup Language (MathML) Version 2.0, http://www.w3.org/TR/MathML2/.

Chambers, J.M. (1998) *Programming with Data: A Guide to the S language*, New York: Springer-Verlag.

Clark, J.(ed.) (1999). *XSL Transformations (XSLT) Version 1.0*, http://www.w3.org/TR/xslt .

Clark, J. & DeRose, S. (eds.) (1999). *XML Path Language (XPath)* http://www.w3.org/TR/xpath.

Clark, J. & Murata, M. (2001). *Relax NG speci-*

fication, Committee Specification 3 December 2001, OASIS.

Clocksin, W.F. (1997). *Clause and Effect: Prolog Programming for the Working Programmer*. London: Springer-Verlag.

Clocksin, W.F. & Mellish, C.S. (2003). *Programming in Prolog: Using the ISO Standard*. London: Springer-Verlag.

Codd, E.F. (1970). A relational model for large shared data banks, *Communication in ACM*, **13**(6), pp. 377–387.

Data Documentation Initiative (2003). The Data Documentation Initiative Codebook DTD Version 2.0. `http://www.icpsr.umich.edu/DDI/users/dtd/`.

Data Mining Group (2003). The Predictive Model Markup Language (PMML) Version 2.1. (`http://www.dmg.org`).

Date, C.J. (1981). *An Introduction to Database Systems, 3rd. ed.*, Addison-Wesley.

Fletcher, J. (2003) *XML in Prolog Ver.1.6*, `http://homepages.tesco.net/binding-time/`.

Fowler, M., 羽生田栄一監訳 (2000). *UML* モデリングのエッセンス *(UML Distilled 2nd ed.)*, 東京：翔泳社.

Greenstein, D. (2001). DDI evaluation report, in *Addendum to Final Report, "Electronic Preservation of Data Documentation: Complement SGML and Image Capture, SBR-9617813, Result of the Evaluation of the Data Documentation Initiative (DDI)"*.

萩野達也他 (2002). 特集 セマンティック Web, 情報処理, **43**(7), pp. 707–750.

Hassan, S, Sylthe, G., de Vries, R. (1996). The CESSDA integrated data catalogue, *IASSIT '96 Conference, Minneapolis May 1996*.

Hull, R. & King, R. (1986). Semantic database modeling: Survey, application, and research issues, *ACM Computing Surveys*, **19**(3), pp. 201–259.

Ihaka, R., & Gentleman, R. (1996) R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, **5**, pp. 299–314.

飯高茂 (1990). *Prolog* で作る数学の世界, 東京：朝倉書店.

IMS Global Learning Consortium, Inc., (2000). IMS Question & Test Interoperability Specification: A Review, *A QTI White Paper from IMS*, IMSWP-1 Version A., `http://www.imsglobal.org/question/whitepaper.pdf`.

Inmon, W.H., Rudin, K., Buss, C.K. & Sousa, R. (鈴木健司, 三船洋一, 室住正晴 訳) (2002). データウェアハウスパフォーマンス, —システム構築・管理技法—, 東京：共立出版. (原著 1999, John-Wiley & Sons, Inc).

ISO TC 46/SC4 (2003). *The Dublin Core metadata element set*, ISO Technical Committee 46 (Information and Documentation)/ Subcommittee 4 (Technical Interoperability) N515.

Kimball, R. 藤本康秀 監訳 (1998). データウェアハウスツールキット：多次元データウェアハウス構築の実践手法. 日経 BP. (原著 1996, John-Wiley & Sons, Inc.).

Lunde, K. 小松章・逆井克己訳 (2002). *CJKV* 日中韓越情報処理, オライリー・ジャパン. (原著 1999, O'Reilly & Associates Inc.)

Meyer, D., Leisch, F., Hothorn, T. & Hornik, K. (2002). StatDataML—An XML format for statistical data. In Härdle, W., Rönz, B. (eds.) *Compstat 2002—Proceedings in Computational Statistics*, Heidelberg: Physika Verlag. pp. 545–550.

Microsoft Corp. & Hyperion Solution Corp. (2002). XML for Analysis Specification Ver.1.1, XMLA 仕様書サイト `http://xmla.org`.

Musgrave, S., Littlewood, P., & Ryssevik, J. (2000). *NESSTAR (Networked European Social Science Tools and Resources) Final Report*, `http://www.nesstar.org/`.

根岸正光・石塚英弘 編 (1994). *SGML* の活用, 東京：オーム社.

日本規格協会編 (2003). 国際符号化文字集合 (UCS)—第 1 部：体系および基本多言語面 (抜粋) X0221-1:2001, (ISO/IEC 10646-1:2000), JIS ハンドブック 64, 情報技術 I, 東京：日本規格協会.

オメガプロジェクトサイト, `http://www.omegahat.org`

OpenMath サイト, `http://monet.nag.co.uk/cocoon/openmath/`

大津起夫 (2003). マークアップ言語を用いた得点統計量一覧表の生成, 大学入試センター研究開発部 リ

サーチノート, RN-03-14.

Ryssevik, J., & Musgrave, S. (2001). The social science dream machine, *Social Science Computer Review*, **19**(2), pp. 163–174.

SAX project, `http://www.saxproject.org/`.

佐藤英人 (1988) 統計データベースの設計と開発：データモデルと知識ベースの応用, 東京：オーム社.

柴田里程 (2001). データリテラシー, 東京：共立出版.

Sperberg-McQueen, C.M. (2003). Logic grammars and XML Scheme, *Extreme Markup Languages 2003 Proceedings*, Montreal Canada. `http://www.idealliance.org/papers/extreme03/`.

Sperberg-McQueen, C.M. & Burnard, L. (eds.) (2002). *TEI P4: Guidelines for Electronic Text Encoding and Interchange*. Text Encoding Initiative Consortium. XML Version: Oxford, Providence, Charlottesville, Bergen.

Sterling, L. & Shapiro, E. (1994). *The Art of Prolog, 2nd ed.*, MIT Press.

Su, S.Y.W. (1983). SAM* : A semantic association model for corporate and scientific-statistical databases. *Information Sciences*, **29**, pp. 151–199.

杉本重雄 (2003) 日本語による Dublin Core Element Set, `http://www.dl.ulis.ac.jp/DC/index_J.html`（ダブリンコアのホームから指定されている日本語訳）.

Swedish Institute of Computer Science (SICS). (2003). *SICStus Prolog 3.11.0 Manual*, Kista Sweden:SICS.

Temple Lang, D. (2001). Embedding S in other languages and environments. In Hornik, K. & Leisch, F. (eds.) *Proceedings of the 2nd International Workshop on Distributed Statistical Computing*. `http://www.omegahat.org/Papers/`.

Unicode Consortium & Aliprand, J. (2003). *The Unicode Standard, Version 4.0*, Boston: Addison-Wesley.

UNIX インターナショナル (1992), *UNIX SYSTEM V* 日本語環境共通規約, 東京：トッパン.

浦本直彦 (2003). Web における情報統合—セマンティック Web と Web サービス—, 情報処理, **44**(7), pp. 707–712.

U.S. Census Bureau & the Centers for Disease Control (2003) *DataFerrett Ver.1.1.8 Release Note*, `http://dataferrett.census.gov/TheDataWeb/`.

van der Vlist, E. 田村健人他訳 (2003). *XML Scheme*, 東京：オライリー・ジャパン. (原著 2002, O'Reilly & Associates Inc.).

W3C DOM Working Group, (2003), *Document Object Model (DOM) Technical Reports*, `http://www.w3.org/DOM/DOMTR`.

和田英一, 戸村哲, 萩谷昌巳 (1998). インタラクティブエッセイ Unicode は好きですか？, 情報処理, **39**(4), pp. 321–323.

Wielemaker, J. (2003). *SWI-Prolog 5.2.9 Reference Manual*, Department of Social Science Informatics (SWI), University of Amsterdam.

Yanai, H., Okada, A., Shigemasu, K., Kano, Y. and Meulman, J.J. (eds.) (2003). *New Developments in Psychometrics*, Tokyo:Sprinver-Verlag.

横内大介, 柴田里程 (2001). インターデータベース—DandD インスタンスのエージェント化—, 統計数理, **49**, pp. 317–331.

Technical Trends in Statistical Metadata Description and the use of XML Documents

OTSU Tatsuo*

Abstract

Increased archived data relating to university entrance examinations require well organized management. An effective method for the organized management of a wide range of data resources and documents is needed. The growing ubiquitous use of XML (eXtensible Markup Language), a descendant of SGML, is enforced by the technical necessity for managing electrical data and documents. We will review recent important developments in related technologies.

First, we surveyed some statistical metadata description projects. The word 'metadata' means information for data management. We focused on three topics. The first was a survey codebook description mainly of American and European social science oriented data archives. One promising specifications in this area was a proposal by DDI (Data Documentation Initiative). The second topic was an information description standardization related to e-Learning and computer-based testing. And the third was TEI (Text Encoding Initiative) project, the largest activity to construct a standard markup system for documents in liberal arts and humanities.

In the next part, we introduce basic properties of XML language. Specifications of XML are simpler than SGML, and this simplicity has made it popular for practical information systems in technological trends. HTML, another popular markup language, is primarily aimed at the visual representation of texts and other resources on Web browsers. The purpose of XML is more system oriented. The assumed primary 'consumers' of XML documents are computer systems rather than humans. Many of the introduced projects adopted XML as their information representation language.

We then reviewed the technical developments that support construction of an XML based information system. We reviewed related topics. Application Program Interface (API) is the most basic technology for constructing an application system. An XML transformation language specification, XSL (eXtensible Stylesheet Language family), is a rapidly growing technology for processing XML documents, and is easier to use than programming with APIs. Another important topic is XML scheme languages. The W3C authorised XML Scheme is replacing traditional DTD (Data Type Definition), which originated with SGML. Another influential XML scheme language is Relax NG. It has simpler specifications than XML Scheme, and was experimentally adopted in the TEI project.

In the final part, XML processing by Prolog, a logic-based language, was introduced. There are some XML parser on Prolog. Prominent properties of Prolog are its pattern

* Department of Applied Statistics and Measurement, Research Division,
The National Center for University Entrance Examinations,
2–19–23 Komaba, Meguro-ku, Tokyo 153–8501 Japan

matching ability and indeterministic execution mechanism. At first, simple examples were given of the basic mechanisms of Prolog language. Then XML parsers on Prolog systems were introduced. Examples were given of the parsing of a small XML document and a method to retrieve information by pattern-matching. Finally, XML document generation with a template code was explained.

Key words: metadata, XML, DDI, Prolog