社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2004－ＦＩ－74 (7)
2004－ＤＤ－43 (7)
2004／3／26

素性論理に基づくXML文書ルール記述言語DRDL

今村 誠　増塩 智宏　伊藤 山彦
三菱電機（株）情報技術総合研究所 音声・言語処理技術部

本稿では、インターネット上の組織間XML文書交換システムにおけるクライアント側のXML入力チェック機能や、XML変換やRDB連携などのサーバ側のXML処理機能を効率的に開発するための基盤となるXML文書ルール記述言語 DRDL を提案する。DRDL の技術特徴は、自然言語処理分野のユニフィケーション文法で用いられる素性論理を主に以下の3点で拡張した論理に基づく点にある。(1)素性の値としてXML要素のノードリストを許し、ノード毎の制約やノード数の制約を限量子を用いて表現する。(2)素性構造における入れ子をたどる素性の列からなるパスとして、W3C の Xpath を用いる。(3)ソートとして、XML Schema のデータ型を扱う。 得られた素性論理における制約式を手続き的に解釈実行するDRDLプロセッサにより、XML文書の内容検証、内容代入、およびRDB登録などのXML文書処理を実現できる。そして、電子申請、設計支援、および設備管理などの実用アプリケーションへの適用を通じて、DRDLプロセッサ、DRDLエディタ、およびDRDLルールトランスレータからなるDRDLツールが、文書ルールの編集・再利用・差し替え、および他機能連携の容易さなどの従来のXML文書処理ツールの課題を解決していることを確認した。

An XML Document Rules Description Language DRDL Based on a Feature Logic

Makoto IMAMURA　Tomohiro MASUSHIO　Takahiro ITO
Mitsubishi Electric Corporation Information Technology R & D Center Human Media Technology Dept.

We present an XML Document Rules Description Language DRDL, which is a basis for developing XML processing functions such as XML validation, transformation and RDB connection in the internet-based document exchange systems. The main technological feature of DRDL is that it is based on the extended feature logic by the following three points. (1) XML element node list is permitted as a value of feature. (2) W3C's Xpath is used as a feature path which accesses a value in a feature structure. (3)XML schema is used as a sort in a feature logic. XML document functions such as XML validation, value assignment and RDB connection, are realized by a DRDL processor which interprets constraint formulas in the extended feature logic. We show that a DRDL tool, which consists of editor, rule translator and processor, solves problems about the edition, reuse and replacement of document rules and the connection of other functions, by applying real application systems, such as electronic application, design support and facility management.

1 はじめに

インターネットを用いて業務プロセスを自動化するために、企業間での電子商取引(B2B EC)や行政・企業間での電子申請を実現するXML(eXtensible Markup Language)文書交換システムの開発がさかんになってきた。B2B ECや電子申請用のシステム開発では、XML文書の表示や変換だけでなく、業務での文書データの自動処理を促進するために、文書内容が満たすべき規約(文書ルール)を検証する機能が、文書作成支援や文書受付を実現する上で重要になる。しかし、W3C(World Wide Web Consortium)の内容検証用規格であるXML Schemaでは、DTD(Document Type Definition)の問題点であったデータ型、継承、および名前空間対応の機能追加を主な目的としており、「要素内容間の制約(内容間制約)」、「要素内容と、要素の存在有無や反復回数などの文書構造との関係制約(内容・構造間制約)」、および「あるXML文書中の要素内容と、別のXML文書の存在有無や要素内容との関係制約(複数文書間制約)」を表現できない。そのため、これらの制約の充足性を検証する機能開発では、DOM(Document Object Model)などのXML文書構造木を直接操作するプログラミングが必要になり、「アプリケーションプログラム中への検証ロジック埋め込みによる開発効率低下」、および「文書規約変更時のコードの修正・コンパイル作業による保守性低下」が生じるという問題点があった。

そこで、データ型の制約に関してはXML Schemaを併用しながら、内容間制約、内容・構造間制約、および複数文書間制約を表現・検証するしくみを提供するために、XML文書ルール記述言語DRDL(Document Rules Description Language)とその処理系(DRDLプロセッサ)を開発してきた[1][2]。DRDLの技術特徴は、自然言語処理分野のユニフィケーション文法[3]で用いられる素性論理を理論的な基盤とする点にある。

本稿の構成は、以下の通りである。2章では、組織間でXML文書を交換する受発注や設計など業務支援システムのXML文書内容処理に対する要求とその技術課題

について述べる。3 章では、2 章の問題を解決するために提案する XML 文書ルール記述言語 DRDL の構文と意味について述べる。4 章では、2 章で述べた XML 文書処理機能の開発保守からの要求に応えるために開発した DRDL ツールについて述べる。5 章では、4 章の DRDL ツールを電子申請、設計支援、および設備情報管理などの実システムへの適用結果を報告することにより、DRDL の実用性を示す。6 章では考察について述べ、最後の 7 章ではまとめを述べる。

2 課題

本章では、組織間で XML 文書を交換する受発注や設計など業務支援システム開発における XML 文書内容処理の課題について述べる。具体的には、2.1 節では、XML 文書内容処理機能への要求を整理し、2.2 節ではその要求を実現する際に生じる現状技術の問題点について述べる。2.3 節では、2.2 節であげた問題点を解決する方針について述べる。

2.1 業務システム実現に必要な XML 文書処理機能

本節では、業務支援システム実現に必要とされる XML 文書内容処理機能と、それらの機能の開発や保守を効率化するために必要な要求について述べる。

2.1.1 XML 文書の内容処理への要求機能

受発注や設計など企業業務システムにおいて組織間で交換される XML 文書は、企業内の業務システム内で処理する際にエラーがでないように、種々の内容チェックや内容代入処理を実現するプログラムを開発する必要がある。XML の用語で述べると、以下のような「内容間検証/代入」、「内容・構造間検証/代入」、および、「RDB 整合性検証/登録復元」の機能を実現するプログラム開発が必要になる。

(1) 内容間制約の処理(内容間検証/代入)

XML の要素内容間制約をチェックする検証処理。また、反復出現するある要素の合計値を計算して別の要素に挿入するといった代入処理。

(2) 内容・構造間制約の処理(内容・構造間検証/代入)

ある要素の内容と、別の要素の内容の出現有無や反復数との関係制約を検証する処理。また、それらの制約を満たすように内容を補完/代入する処理。

(3) XML-RDB 対応制約の処理(整合性検証/XML 登録復元)

・XML 文書中の要素内容が RDB 中に格納される値と整合性がとれていることをチェックする処理。また、入力 XML 文書中の選択肢を RDB 中から抽出・代入する処理。

・一つの XML 文書中の要素内容を複数の RDB のテーブルに切り分けて登録する処理。また、RDB 中の複数のテーブル中に登録されているデータを一つの XML 文書に復元する処理。

2.1.2 XML 文書処理機能の開発・保守効率化への要求

2.1.1 項で述べた XML 文書処理機能の開発や保守を効率化するには、「(1)内容処理機能の開発・保守の容易さ」と、「(2)他機能との連携の容易さ」を満たす必要がある。以下、順に、要求内容について説明する。

(1) 内容処理機能の開発・保守の容易さ

組織間で交換される XML 文書の項目変更や、検証/代入用文書ルールの変更が生じた場合に簡単に対応できるように、以下の 3 点を満たす必要がある。

(i) 文書ルール編集の容易さ

文書ルールの作成や変更が、情報システム部門の担当者だけでなく、業務部門の担当者でも可能であること。

(ii) 文書ルール再利用の容易さ

文書ルールは、文書を交換する組織間で共有し、組織毎の業務システム用に利用しやすいものであること。

(iii) 文書ルール差し替えの容易さ

分散環境下において、業務内容の変更に応じて追加/削除が容易にするために、文書ルールは簡単に差し替え可能であること。

(2) 他機能との連携の容易さ

文書ルールを処理する機能は、アプリケーション中のいくつかのプログラムと非常に密に呼びあいながら動作できるようにすること。例えば、クライアント側の XML 文書入力ツールでは、入力画面を表示・制御するプログラムや、業務特有の複雑な計算を実現するプログラムと密に連携する必要がある。

2.2 要素技術の現状レベルと課題

本節では、W3C の標準である DOM, XML Schema, Xpath(XML Path Language)、および XSLT をとりあげ、2.1 節の要求を実現する際に生じる現状の XML 技術の課題を整理する。

(1) DOM

Java や C++などのプログラミング言語により、DOM-API を用いれば、2.1.1 節であげた XML 文書処理機能を実現することができる。しかし、低レベルの記述のため表現能力が非常に大きいという特徴がある反面、2.1.2 項で述べた「文書ルール編集の容易さ」や「文書ルール再利用の容易さ」を実現することは困難であるという問題がある。

(2) XML Schema

XMLSchema では、豊富なデータ型の記述の能力を用いた単項目のチェックは可能であるが、2.1.1 項で述べた「内容間検証代入」や「内容・構造間検証/代入」はそのスコープ範囲外である。

(3) Xpath

Xpath は、階層構造をたどる(トラバースする)ための式(以下では、Xpath 式と呼ぶ)を提供しており、「要素

A の内容が要素 B の内容に等しい」、「要素 A の内容が、反復出現する要素 B の内容の合計値に等しい」といった内容間検証を記述することができる。しかし、「if 文に相当する条件分岐に関する制約が記述できない」、『「反復出現する要素 A の内容は、すべてがある値以上である」といった反復要素毎の制約が記述できない』という問題があった。また、XML 文書中のある一部分を指し示す(アドレッシング)するための言語を提供することが目的であるので、検証処理は実現できるが、補完/代入処理は Xpath 仕様のスコープ外なので、別途のプログラミングが必要になる。

(4) XSLT

XSLT は、Xpath を用いて XML 文書の変換規則を記述する言語である。XSLT は、Xpath では記述できない if 文や反復要素毎の制約を表現する for-each 文をもち、さらに、XSLT プロセッサの拡張機能を用いることにより複数 XML 文書間の処理を記述できるので、2.1.1 項にあげた XML 文書内容処理をほぼ実現するだけの表現能力を持っているといえる。

しかし、XML 文書変換用言語なので、以下に示すように、XML 文書内容検証や代入処理用のルール記述言語として必ずしも適していない点がある。

(i)入出力共に XML ファイルを想定しており、共有メモリ上に展開された XML 文書に対して代入処理や検証処理を実施する文書内容処理を実現するのには適さない。

(ii) 「表現能力は大きいが、低レベルの雑多な構文を含む大きな言語仕様である」、また「Xpath が指し示すカレントノード毎に変換規則を適用するので、比較対象となる要素内容のコンテキストを指定する記述が煩雑になる」ため、2.1.2 で述べた「文書ルール編集の容易さ」や「文書ルール再利用の容易さ」の実現が困難である。

2.3 課題の解決方針

本稿では、2.1 節の要求を満たす XML ツールを実現するために、XML の専門知識が不要な編集ルールを作ることができ、かつ XML 入力機能における内容チェックや内容代入処理を記述できるような簡潔な XML 文書ルール記述言語 DRDL を提案する(3 章)。また、2.1.2 節に述べた XML 文書処理機能の開発・保守への要求に対応するために開発した DRDL ツールについて述べる(4 章)。

3 XML 文書ルール記述言語 DRDL

本章では、2.1 節で述べた要求実現に十分な記述力をもつと共に、簡潔な構文と意味をもつ XML 文書ルール記述言語として、DRDL を定義する。以下、順に、DRDL の設計方針、DRDL の構文、DRDL の操作的意味、そして、DRDL の記述例について説明する。

3.1 言語の設計方針

設計方針は、素性論理がもつ簡明さをもちかつ実用的な言語を定義することである。以下、言語の構文と意味毎の設計方針を述べる。

(1) 構文

2.1 で述べた XML 文書内容処理を実現できる表現能力を得るために、Smolka[4]や Jhonson[5]らの素性制約式を以下の点で拡張した Xpath 制約式を導入する(Smolka の素性論理における素性制約式の構文を通常の一階言語の項との相違点に注目した解説は[6]に譲る)。

・素性制約言語の項に相当するものとして Xpath 式を対応させる (本項とでは XPath 項と呼ぶ)。

・素性の値として、XML のノードの集合を扱えるようにする。また、「束縛変数がとりうる値の集合を Xpath 式が指し示すノードの集合に限定した限量子付制約式」と「ノードの集合の要素数に関する制約式」を導入する。

・ソートに相当するもとして XMLSchema のデータ型を対応させる。

(2) 意味

ユニフィケーション文法の文解析で用いられる素性構造のユニフィケーションでは、素性論理における素性制約式の充足可能性の判定手続きを用いていた。しかし、素性制約式の充足可能性判定を 2.1 節で述べた課題解決に用いるには、「アルゴリズムが実装上遅いこと」と「双方向性をもつ計算が文書ルールの記述者にとってわかりにくい」という問題がある。そこで、実用上要求されているのは、2.1 節で述べた内容検証と内容代入の処理であるという特性を生かして、素性制約式の操作的な意味として、与えられた XML 文書の内容をチェックする「チェック解釈」と、与えられた XML 文書の内容の書き換えに利用する「破壊的代入解釈」の二つを導入する。

チェック解釈では、Xpath 制約式の意味は、XML 文書を入力として、真偽値を出力として返す関数である。チェック解釈では、限量子の束縛変数がとり得る値の集合が決まってしまうので、Xpath 制約式の充足可能性判定は、命題論理の充足可能性に帰着されるため、通常の手続き型言語の論理演算と同様の処理となる。

破壊的代入解釈では、Xpath 制約式の意味は、XML 文書を入力として XML 文書を出力として返す関数である。具体的には、通常の手続き型言語の代入文と同様に、等式を左辺の変数に対する右辺の値の代入と解釈することにより、与えられた Xpath 制約式を満たすように XML のノードの値を書き換える処理を実現する。

3.2 DRDL のコア言語 XPath 制約言語の構文

本節では、DRDL のコアとなる言語である XPath 制約式の構文について述べる。通常の一階言語の場合にならって、項と式を定義する。

(1) Xpath 項の構文

(a) 変数は、XPath 項である。
(b) 定数は、XPath 項である。ここで、定数とは、XPath が定めるオブジェクトの基本型である"ノード集合"、"ブール値"、"数値"、"文字列"に属する値とする。
(c) Xpath 式は、XPath 項である。

(2) Xpath 制約式の構文

一階言語にならって、下記で定義される Xpath 制約式の集合を Xpath 制約言語と呼ぶ。

(a) s と t が XPath 項ならば、**s=t**、s!=t、**s<=t**、**s>=t**、**s>t**、および **s<t** は、Xpath 制約式である(**Xpath** パス比較式と呼ぶ)。s=t を **Xpath** パス等式と呼ぶ。
(b) s が Xpath 項であり、t が XML Schema で定義されたデータ型ならば、**s: t** は、Xpath 制約式である。s: t を **Xpath** 型式と呼ぶ。
(c)Φ と Ψ が Xpath 制約式ならば、**Φ∧Ψ**、**Φ∨Ψ**、¬**Φ**、および**Φ⇒Ψ**は、Xpath 制約式である。これらの制約式を順に、**連言 Xpath 制約式**、**選言 Xpath 制約式**、**否定 Xpath 制約式**、および**含意 Xpath 制約式**と呼ぶ(総称して、**Xpath 論理式**と呼ぶ)。
(d) x が変数、s が Xpath 項、そして Φ が Xpath 制約式ならば、**∀ x ∈ s . Φ**は、Xpath 制約式である(**全称 Xpath 制約式**と呼ぶ)。ここで、∈の右辺の s は、Xpath の仕様にしたがって、Xpath 項を評価して得られるノード集合を意味する。
(e) x が変数、s が Xpath 項、n は自然数、そして Φ が Xpath 制約式ならば、**∃(= n) x ∈ s s.t.(count(.)=n) . Φ**は、Xpath 制約式である(**存在 Xpath 制約式**と呼ぶ)。ただし、n は評価結果が自然数となる式でもよい(例えば、n は、3+5 や、Xpath 式 t が指し示すノードの数である count(t)などでもよい)。また、=は、!=、<=、>=、または、>などの比較記号でもよい。また、**∃x ∈ s s.t.(count(.)=n)**の部分を出現数 **Xpath** 制約式と呼ぶ。

3.3 Xpath 制約式の操作的意味と従来技術との差異

以下、3.2 節で述べた構文毎に操作的意味を説明する。

3.3.1 XPath 項の意味

「チェック解釈」と「破壊的代入解釈」のいずれの場合も、XPath 項の意味は、Xpath の意味に準ずる。すなわち、与えられた XML 文書において Xpath 項が指し示すノード集合、ブール値、数値、または、文字列とする。

(a) Xpath 比較式

チェック解釈では、左辺と右辺の Xpath 項が指し示す値の対が比較演算子(=,<など)を満たす場合は真を返し、そうでない場合は、偽を返すものとする。但し、ノード集合の要素の XML 文書中の出現順序も含めて等しい場合とする(Xpath の等式とは意味が異なる)。

代入解釈では、パス等式で左辺が Xpath 項の場合にだけ意味をもち、左辺の Xpath 項が指し示す XML 文書中のノードに対して、右辺が指し示す値を代入する。DOM-API の値の代入(Node クラスの setNodeValue)に相当する。

(b) Xpath 型式

チェック解釈では、左辺の Xpath 項が指し示す値が、右辺のデータ型に属する場合には真を返し、そうでない場合は、偽を返すものとする。Xpath 型式は、Xpath 論理式や限定子付 Xpath 制約式中に出現することができるので、XMLSchema による型チェックよりも表現能力が大きい。

破壊的代入解釈は、定義しない。

(c) Xpath 論理式

チェック解釈では、連言 Xpath 制約式、選言 Xpath 制約式、否定 Xpath 制約式、および含意 Xpath 制約式は、通常の論理式と同様の解釈で、真偽値を返すものとする。

破壊的代入解釈では、連言 Xpath 制約式Φ∧Ψは、Xpath 制約式Φを実行した後に、Xpath 制約式Ψを実行することと定義する。XSLT の逐次文に対応する。選言 Xpath 制約式Φ∨Ψは、最初の Xpath 制約式Φのみを実行することと定義する。否定 Xpath 制約式は、破壊的代入解釈の対象外とする。含意制約式Φ⇒Ψは、Xpath 制約式Φをチェック解釈したときに真であるときのみ、Xpath 制約式Ψを実行することと定義する。XSLT の if 文に対応する。

(c) 全称 Xpath 制約式

チェック解釈では、∀ x ∈ s. Φは、Xpath 制約式Φが、s が指し示すノード集合の要素ノードを入力とした場合にすべて真を返す場合に真を返し、それ以外の場合に偽を返すものとする。

破壊的代入解釈では、s が指し示すノード集合の要素の値を入力として、XML 文書中の出現順に、Φを実行する操作と定義する。XSLT の for-each 文に相当する。

(d) 存在 Xpath 制約式

チェック解釈では、∃x ∈ s s.t(count(.)=n) . Φは、s が指し示すノード集合の要素数が n に等しく、かつパス制約式Φが真であるときのみ真を返し、それ以外の場合は偽を返すものとする。出現数 Xpath 制約式(count(.)=n)は、XML Schema における要素出現数をチェックする属性 minOccurs や maxOccurs に対応するが、右辺の n が変数、計算式、Xpath 式などを記述できるので、Xpath 制約式の方が記述能力が大きい。

破壊的代入解釈では、存在 Xpath 制約式は、∃ x ∈ s s.t(count(.)= n) . Φは、count(s) = n を満たすように XML 文書を操作した後で、Xpath 制約式Φを実行することと定義する。出現数 Xpath 制約式の部分は、その制約を満たすように、XML 文書中のノードを追加・削除する操作であり、DOM-API のノード操作メソッド(Node クラスの appendChild や removeChild)に相当する。

3.4 XML 文書ルール記述言語 DRDL

XML 文書ルール記述言語 DRDL では、3.3 節で述べた Xpath 制約式をベースとして、応用上有用な「ベクトル演算」、「RDB とのマッピング定義」、「XML 文書中

のコンテキスト指定」、および「Xpathの計算関数の併用」などの機能拡張を実施した。以下、順に説明する。

(1)ベクトル演算(ベクトル和・差、スカラー積、内積)

XML 文書中の内容間制約検証でよく用いられる反復要素全体に対する演算として、ベクトル和・差、スカラー積、内積を追加した。

s + t は、Xpath 式 s と t が指し示すノード集合の要素を、XML の出現順に並べたベクトルとみなして、ベクトルの和と差を計算した結果を返す。ベクトル和や差は、反復要素同士で同じ順序の値ごとに和や差をとる場合に用いる。同様に、Xpath 式を評価した結果をベクトルとみなして、スカラー積と内積をとる演算を追加した。

(2)RDB とのマッピング定義

「XML 文書の RDB 登録」や「RDB の検索結果の XML 文書への取り込み」を記述するための構文を追加した。

(3)XML 文書中のコンテキスト指定

Xpath 式のカレントノードを、Xpointer で指定する構文を追加することにより、XPath 制約式を簡潔に記述できるようにした。Xpointer を用いるので、複数 XML 文書間の内容制約を記述することもできる。

(4)Xpath や XSLT の組み込み関数の利用

反復要素の合計値を計算する sum や、文字列を連結する concat などの Xpath や XSLT の組み込み関数を併用利用できるようにした。

3.5 DRDL の記述例

本節では、2.1.1 節で述べた「内容間制約」、「構造・内容間制約」、および「XML-RDB 対応制約」の DRDL 記述例を示す。

(1) 内容間制約の例

図 1に示す Xpath 制約式は、図 2のような XML 文書を想定して、『「小計」は、「購入品」の「価格」の合計値と等しい』という制約を表現している。チェック解釈を用いれば内容チェック機能を実現できる。また、破壊的代入解釈を用いれば、表計算ソフトウェアのセル値間計算に相当する 編集支援機能を実現できる。

```
∀ x ∈ //購入品リスト.    x/小計 = sum(x/購入品/価格)
```

図 1 内容制約制約の記述例

```
<購入品リスト 購入日＝"2002-10-01">
 <購入品><名称>PC</名称><価格>95000</価格></購入品>
 <購入品><名称>Disk</名称><価格>50000</価格></購入品>
 <小計>        </小計>
</商品リスト>
<購入品リスト 購入日＝"2002-11-01">
 <購入品><名称>note</名称><価格>200</価格></購入品>
 <購入品><名称>pen </名称><価格>100</価格></購入品>
 <小計>        </小計>
</商品リスト>
```

図 2 処理対象 XML 文書の例

Xpath 型制約式を用いることにより、XMLSchema では表現できない「ある要素内容に依存したデータ型」を定義できる。図 3は、『要素「商品」の子要素「カテゴリ」が「通信」の場合には、「番号」の型は「通信機器コード」でなければならない』という制約を表現している。但し、ここで「通信機器コード」は、XML Schema のデータ型として定義されているものとする。

```
∀x ∈ //商品.
  ( x/カテゴリ ="通信") ⇒ (x/番号 : 通信機器コード)
```

図 3 型制約式の例

ベクトル演算を用いれば、反復出現する複数要素間の計算を簡潔に表現できる。図 4の(a)式と(b)式は、いずれも要素 a,b,c が反復出現する XML 文書において、n 番目に出現する要素 a の値が、n 番目の b 要素の値と n 番目の c 要素の和に等しいことを示している。(a)はベクトル演算を用いており、(b)は全称 Xpath 制約式で表現している。ここで、position(x)は、全称記号で束縛されるノード x を与えると、Xpath 式(本例では、//a)の評価結果として得られるノードリスト中での出現順序数を返す関数である。

```
(a)  //a  =  //b  +  //c
(b)∀x ∈ //a.   n=position(x) ∧ ( //a[n] = //b[n] + //c[n] )
```

図 4 ベクトル演算を用いた制約式の例

(2) 構造・内容間制約の例

図 5に示す Xpath 制約式は、図 6のような XML 文書を想定して、『要素「装置」の出現数は、要素「装置数」の内容に等しい』という制約を表現している。

チェック解釈を用いれば内容チェック機能を実現できる。また、破壊的代入解釈を用いれば、DOM の API を用いて実現するような要素の追加や削除を実現できる。図 6の XML 文書を入力として破壊的代入解釈により図 5に示す Xpath 制約式を実行すれば、『要素「装置」を 3 つ追加して、要素「装置」の出現数を要素「装置数」の内容である 5 と等しくなるようにする』という処理が実行される。この処理を XML 文書入力支援ソフトウェアから呼び出すことにより、XML 文書中のある記載項目の内容に応じて、他の記載項目の入力枠数を可変にするような動的な入力フォームを実現することができる。

```
∃ x  ∈ //装置 s.t.( count(.)=  //装置数 )
```

図 5 構造・内容間制約の記述例

```
<装置数> 5 </装置数>
<装置リスト>
  <装置>  ... </装置>
  <装置>  ... </装置>
</装置リスト>
```

図 6 処理対象 XML 文書の例

(3) XML-RDB 対応制約

図 7の Xpath 制約式は、破壊的代入解釈により、図 8

の設備情報を記載したXML文書を入力として、該当する機器IDの点検日と電圧をRDBから得てXML文書に書き込み出力することができる。具体的には、機器ID毎に02行目と03行目にあるSQL文を実行し(結果は図9)、04行目から06行目にて該当する箇所に点検日と電圧の値を挿入する操作を実行する(結果は図 10)。

```
01: ∀ x ∈outputfile.xml!設備情報/機器ID .
02: ( y =  select DATE, DENATSU from kikitenken
03:              where KIKINO = '$x'")
04: (∃z (=count(y/Row))∈x/点検リスト/点検情報 .
05:     (  z/点検日 =  y/ROW[position(z)]/DATE                ∧
06:        z/電圧  =  y/ ROW[position(z)]/DENATSU )  )
```

図 7 XML-RDB 対応制約の例

```
<設備情報>
  <機器ID>0001</機器ID>
  <点検情報リスト>
    <点検情報>     </点検情報>
  </点検情報リスト>
</設備情報>
<設備情報>
  <機器ID>0002</機器ID>
  ……
</設備情報>
```

図 8 入力XML文書の例

```
<ROW>
  <DATA >04-02-01</DATA>
  <DENATSU> 2000</DENATSU>
</ROW>
<ROW>
  <DATA >04-02-02</DATA>
  <DENATSU>3000</DENATSU>
</ROW>
```

図 9 中間結果として変数yに格納されるXML文書例

```
<設備情報>
  <機器ID>0001</機器ID>
  <点検情報リスト>
  <点検情報>
    <点検日>04-02-01</点検日>
    <電圧>2000 </電圧>
  </点検情報>
  <点検情報>
    <点検日>04-02-02</点検日>
    <電圧> 3000</電圧>
  </点検情報>
  </点検情報リスト>
</設備情報>
<設備情報>
  ……
</設備情報>
```

図 10 出力の中間状態(図 9の最初のRowを処理した時点)

4 DRDLの実装

本章では、2.1.2 項で述べた「XML 文書処理機能の開発・保守効率化への要求」に応えるために開発したDRDLツールについて述べる。

4.1 DRDLツールの全体構成

DRDLツールは、XML文書ルールを記述したXML文書であるDRDLファイル、DRDLファイルを作成するためのDRDLエディタ、DRDLファイルを解釈実行することによりXML文書内容処理を実行するDRDLプロセッサ、およびXML文書ルールを既存のXMLツール用の記述形式に変換するDRDLルールトランスレータからなる。

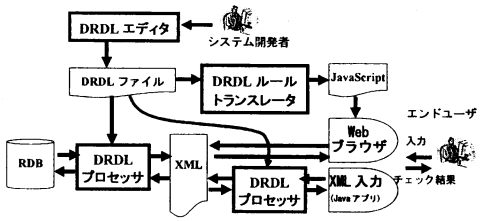

図 11 DRDLツールのシステム構成図

以下、順に説明する。

4.2 DRDLファイル

2.1.2 節(1)(iii)「文書ルール差し替えの容易さ」を実現するために、アプリケーションプログラムから独立したファイル(DRDLファイルと呼ぶ)として文書ルールを記述できるようにした。この DRDL ファイルを伝送・差し替えするだけで、クライアント側のXML文書入力ツールにおける内容処理機能を更新したり、また、サーバ側のXML文書変換機能を更新できる。

DRDLファイルは、3章で述べたDRDLの個々のXpath制約式をXMLの構文で表現したものである。図 12に、図 1のXpath制約式のXML構文での表現例を示す。

```
<all-path bind-val="x" path="//購入品リスト">
  <path l-path="{$x}/小計" op="eq" r-path="sum(./購入品/価格)">
</all-path>
```

図 12 XML構文によるXpath制約式の記述

4.3 DRDLエディタ

2.1.2 項(1)(i)「文書ルール編集の容易さ」を実現するために、XML、Java、Webなどの特別な専門知識を持たない業務部門担当者でも Xpath 制約式を作成できるDRDLエディタを開発した。DRDLエディタは、図 13に示すようなテーブル形式の GUI をもち、左部分の文書スキーマ記述部を常時表示させながら、右部分の文書内容制約記述部やXML-RDB関係部を切り替えながら、XML文書ルールを編集できるようにしている。

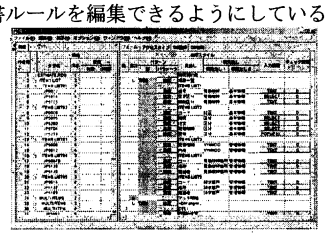

図 13 DRDLエディタ

以下、文書スキーマ記述部、文書内容制約記述部、

XML-RDB 関係部について説明する。

(1)文書スキーマ記述部

DTD や XMLSchema で記述できる XML 文書の構造やデータ型を指定する部分である。市販の DTD や XMLSchema 設計支援ツールとの併用を可能にするために、文書スキーマ記述部は、XMLSchema をインポートできるようにしている。

(2)文書内容制約記述部

2.1.1 項で述べた「内容間制約」と「内容・構造間制約」を記述する(単項目制約は、内容間制約に含まれる)。これらの制約を簡単に記述できるように、簡易 DRDL 言語を開発した。簡易 DRDL 言語における主な工夫点は以下である。

(i)if 文、代入文、×や＋等の算術演算などの既存のプログラミング言語で使い慣れた構文で XML 文書ルールを入力できるようにした。

(ii)反復出現する要素は、自動的に全称記号やベクトル演算などを補って解釈するマクロを導入した。

(iii)対象 XML 文書の構造を自由に定義できる場合は、末端の要素名を一意とすることにより、内容間制約を記述する際に、パスの文脈を指定しなくてよいようにした。

(iv)「要素の反復数に関する制約」や「要素への値の代入規則」の指定用に列を設けることにより、Xpath 制約式を意識させずに、行の左端に示す XML 要素に対する設定と自然に解釈できるようにした。

(v)ある要素の内容に依存して、別の要素がとりうる値の選択肢が可変となる制約式を編集する際は、アプリケーション毎のパターンに応じた GUI を用意した。

(3)XML-RDB 関係制約記述部

XML 文書のタグ毎に、複数の RDB のフィールド名を指定できる。一つの対応指定ごとに、対応するテーブル名、フィールド名、データ型、サイズ、主キー、外部キー、検索条件、および一覧表示時のラベル名などを指定する。通常、外部キーになっているタグには、複数のフィールド名指定がなされる。

4.4 DRDL プロセッサ

2.1.2 項(2)「他機能との連携の容易さ」を実現するために、DRDL プロセッサは、他のプログラムと DOM 木を共有メモリとして共有しながら協調的に XML 文書を処理できるような DRDL 処理系(DRDL プロセッサ)を開発した。DRDL プロセッサは、アプリケーションから、API を介して呼び出され、処理対象 XML 文書に対して、DRDL ファイルを解釈実行した結果を返す。また、解釈実行の方式は、呼び出し API の引数、または、DRDL ファイル中の Xpath 制約式毎の設定により、「チェック解釈」か「破壊的代入解釈」かを選択できる。

また、数値計算処理、テキスト処理、バイナリデータ処理など DRDL では記述できない処理、あるいは無理に DRDL を用いて記述すると生産性が落ちる処理を実現する場合には、Java 等で記述した文書内容処理プログラムを呼び出しながら、アプリケーション特化の文書内容処理を実現できるようにしている。

4.5 DRDL ルールトランスレータ

2.1.2 項(1)(i) 「文書ルール再利用の容易さ」を実現するために、XML 文書ルールを既存の XML ツール用の記述形式に変換する DRDL ルールトランスレータを開発した。本トランスレータにより、Xpath 制約式を Web ブラウザからの XML 文書入力画面用の内容チェック用 JavaScript を自動生成することができる。自動生成された JavaScript は、[7]で述べた方式で生成された XML 入力用 XSLT スタイルシートと併用することで、Web ブラウザでの XML 文書入力用チェックに利用できる。

5 評価

本章では、4 章で述べた DRDL ツールを、官庁向け電子申請システム[1] [8]、昇降機設計支援システム[9]、および設備情報管理システム[10]に適用した結果を報告することにより、DRDL の実用性を示す。以下、2.1 節で述べた課題毎の評価結果について述べる。

(1) XML 文書の内容処理機能の表現能力

2.1.1 項の「(1)内容間制約」と「(2)内容・構造間制約」の記述能力については、「電子申請システムにおける申請書作成時の入力チェック機能の実用化」と「設計支援システムにおける入力チェック機能と工期自動判定機能の実用化」を通じて確認した。「(3)XML-RDB 対応制約」については、「設備情報管理システム用のサンプルデータに対する試作」を通じて確認した。

(2) 文書ルール編集の容易さ

2.1.2 項(1)(i)の「文書ルール編集の容易さ」については、設計支援システムにおいて、情シ部門でなく機種設計部門の担当者による実利用を通じて確認した。

また、内容間制約と内容構造間制約の開発工数を、DRDL と Java での実装とで比較したところ、約 3.5 倍から 5.1 倍の向上がみられた[11]。

(3) 文書ルール再利用の容易さ

2.1.2 項(1)(ii)の「文書ルール編集の容易さ」については、設計支援システムにおいて、内容チェック用の JavaScript を自動生成することにより、DRDL エディタで作成した同一の記述を用いて、Java アプリケーション用のチェック機能と、Web アプリケーション用のチェック機能を実現することができた。

(3) 文書ルール差し替えの容易さ

2.1.2 項(1)(iii)の「文書ルール差し替えの容易さ」については、電子申請システムと設計支援システムにおいて、

DRDL ファイルを差し替えるだけで、XML 文書処理機能を差し替え可能であることを確認した。

(4)他機能との連携の容易さ

2.1.2 項(2)の「他機能との連携の容易さ」については、電子申請システムと設計支援システムにおいて、Java アプリケーションとして実装された XML 入力画面ソフトウェアと DOM 木を共有しながら、文書編集や検証を実現できることを確認した。また、設計支援システムにおいて、業務に依存したバイナリーデータを処理する Java プログラムと連携することにより、後工程の工事設計支援システム用のデータ変換を実現できることを確認した。

6 考察

本章では、DRDL の開発と実用化を通じて得た考察として、「ルールの表現能力と作成容易さのトレードオフ」と「素性構造ユニフィケーションの XML 文書処理への応用」について述べる。

(1) ルールの表現能力と作成容易さのトレードオフ

テキスト処理や数値処理を含む複雑なロジックの実装では、Java 等の通常のプログラミング言語を用いるのが普通である。そこで、DRDL の設計にあたっては、「別のソフトウェアモジュールとの併用利用を想定して、XML 特有の内容チェックや内容代入操作を表現可能」と「他のルールへの変換が容易な単純な構文と意味をもつ制約記述言語」という特徴をもつ XML スクリプト言語の実現を目標としておいた。そのため、Xpath 制約式から自然に導きだせない While 文は、我々の応用用上必要がなかったので、あえて追加しなかった。

(2)素性構造ユニフィケーションの XML 文書処理への応用

本稿では、「できだけ多くの人が使えるようにする」と「実用的な速度性能を得る」という目標を達成するために、DRDL の意味としては、Java や C といった手続き型言語と類推が利くように「チェック解釈」と「破壊的代入解釈」を扱った。しかし、理論的には、Xpath 制約式の充足可能性の判定手続きを与えることにより、Xpath 制約式間のユニフィケーションを定義することができる。ここで、ユニフィケーションとは、二つの Xpath 制約式が与えられたとき、その式に含まれる変数に対して適切な代入を行い、二つの式を同一の表現にする演算である。Xpath 制約式のユニフィケーションを扱うことは、理論的には以下の２点で興味がある。

(i) XML 文書が XML Schema を満たすかどうかの妥当性判定は、Xpath 制約式の充足可能性の判定により実現できる。上記の性質は、インスタンスである XML 文書も、その XML 文書を受理する文法定義に相当する XMLSchema も共に、Xpath 制約式で表現できることを用いている。また、スキーマの上位互換性の判定(前版の XMLSchema を満たす XML 文書が次版の XMLSchema を満たすかどうかの判定)は、Xpath 制約式の包摂関係を判定することにより実現できる。

(ii)XML 文書を項とするような論理型言語を定義できる。つまり、Prolog における項を Xpath 制約式で置き換えることにより得られる論理型言語を定義できる。この言語は、XSLT とは別のパターン記述に基づく変換言語が定義できることを意味している。

7 おわりに

W3C 勧告である Xpath 式を項として、論理演算として連言(∧)、選言(∨)、否定(¬)をもち、限量子と等式をもつ一階論理の部分言語である XML 文書ルール記述言語 DRDL を提案した。DRDL のチェック解釈により、XMLSchema では記述しきれない内容間制約や内容・構造間制約を検証できる。また、DRDL の破壊的代入解釈により、DOM-API を用いたノード追加削除やノードへの値代入や、XSLT の if 文や for-each 文に相当する制御構造を実現できる。さらに、電子申請、設計支援、および設備管理などの現実のインターネットアプリケーションへの適用を通じて、DRDL エディタ、DRDL プロセッサ、および DRDL ルールトランスレータからなる DRDL ツールが、文書ルールの編集・再利用・差し替え、および他機能連携の容易さなどの従来の XML 文書処理ツールの課題を解決していることを確認した。

【参考文献】

[1] 今村 誠, 長浜 隆次, 鈴木 克志, 渡部 明洋：電子申請における XML 文書利用技術, 情報処理学会 デジタルドキュメントシンポジウム 2000 (2000)

[2] 今村誠, 増塩智宏, 伊藤山彦：XML 文書ルール記述言語 DRDL とその EC システムへの応用, 情報処理学会 デジタルドキュメント研究会(DD)39-5, pp23-30 (2003)

[3] Stuart M.Shiever : An Introduction to Unification-based Approaches to Grammer, CSLI Lecture Notes Number4, Stanford University (1986)

[4]Gert Smolka:A Feature Logic with Subsorts,LILOG Report33, IBM Deutschland, Stuttgart.F.R.G. (1988)

[5]Mark Johnson: Attribute-Value Logic and the Theory of Grammar, CSLI Lecture Notes Number16, Stanford University (1988)

[6]今村誠： 素性構造の単一化, 情報処理 Vol. 32 No. 10 pp1070-1078 (1991)

[7] 今村誠, 増塩智宏, 伊藤山彦：木・表構造間写像モデルに基づく XML-HTML 変換用スタイルシート自動生成方式, 情報処理学会 デジタルドキュメント研究(DD)42-5, pp23-30 (2003)

[8]今村誠、富川直毅：電子政府における XML 利用技術の動向、情報処理 Vol42, N1. 7, pp654-662(2001)

[9]今村 誠, 森口 修, 鈴木 克志： XML 文書ワークフロー構築方式, 情報処理学会 デジタルドキュメント研究会資 27-1, pp1-8 (2001)

[10]外崎道夫, 道行泰代, 南部雅彦, 今村誠：上下水道維持管理支援 ASP 情報サービス, 三菱電機技報 Vol. 76, No. 10, pp19-22 (2002)

[11]増塩 智宏, 伊藤 山彦, 今村 誠：XML 文書規約記述言語 DRDL と設計支援システムへの適用, 情報処理学会, FIT-2003, E-027(2003)