社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.
SS2003-21, KBSE2003-24 (2003-11)

XBRLで記述された財務データを扱う言語処理系の提案

高尾　祐治†　　松下　　誠†　　井上　克郎†　　湯浦　克彦††

† 大阪大学大学院情報科学研究科　〒560–8531 大阪府豊中市待兼山町1番3号
†† 株式会社 日立製作所 ビジネスソリューション事業部　〒212–8567 神奈川県川崎市幸区鹿島田890
E-mail: †{y-takao,matusita,inoue}@ist.osaka-u.ac.jp, ††kyuura@itg.hitachi.co.jp

あらまし　XMLの構文を用いて財務報告書を記述するための言語，XBRLが策定されている．XBRLで記述された既存の財務報告書を変換し新しい財務報告書を作成するといった，XBRL文書変換の需要は非常に高い．しかし，XBRLの仕様は複雑であるため，既存のXML処理系を利用した場合の作業効率が問題となっている．そこで我々は，XBRL文書の構造を言語に組み込むことで，XBRL文書の変換を容易に行うための言語処理系を提案する．本稿では，提案するプログラミング言語の概要と，その処理系について説明する．さらに，提案するプログラミング言語を使用してXBRL文書変換プログラムを作成した場合の作業効率について考察する．

キーワード　XBRL，変換，言語処理系

A Programming Language for Finalcial Data Written in XBRL

Yuji TAKAO†, Makoto MATSUSHITA†, Katsuro INOUE†, and Katsuhiko YUURA††

† Graduate School of Information Science and Technology, Osaka University
1–3, Machikaneyama–cho, Toyonaka–shi, Osaka, 560–8351 Japan
†† Business Solution Systems Division, Hitachi, Ltd.
890, Kashimada, Saiwai–ku, Kawasaki–shi, Kanagawa, 212–8567 Japan
E-mail: †{y-takao,matusita,inoue}@ist.osaka-u.ac.jp, ††kyuura@itg.hitachi.co.jp

Abstract　XBRL, a XML based language to describe financial information, has been defined. The conversion of XBRL documents is highly requested to generate a new financial report. However, there is a problem about efficiency, since the XBRL specifications are very complicated. We propose a programming language for XBRL document conversion which has XBRL document structure. In this paper, we explain the programming language, and, the framework to manipulate XBRL documents. We also discuss about the efficiency of XBRL document conversion using our language.

Key words　XBRL, conversion, programming language

1. まえがき

企業は，自らの財務状態を公表するために財務報告書を作成し公開する．近年は，web上で財務報告書を公開する企業も増えてきている．しかし，財務報告書の形式が統一されていないために，企業により使用する用語や，その意味が異なっていたり，異なる企業間の財務報告書を比較することが困難であるといった問題があった．そこで，XMLの構文を用いて財務情報を記述する言語である，XBRL(eXtensible Business Reporting Language)がXBRL Internationalによって策定された[7].

XBRLを用いて財務報告書を記述することで，既存データの容易な再利用や，異なる組織間でのデータの比較ができるようになる．正確，かつ，迅速に財務報告書を作成できるため，XBRL文書を変換するという需要は非常に高い．しかし，既存のXML処理系を用いてXBRL文書の変換処理を行おうとすると，変換処理とは直接関係のないXBRL文書自体の解析に関するプログラムを大量に書かなければならない．このため，XBRL文書の変換処理は非常に困難な作業となっている．

一般に，財務報告書の編集に携わる人にプログラミングの経験が豊富にあることは珍しい．このとき，XBRL文書を変換するために複雑なプログラミングを要求することは現実的ではない．そこで本研究では，XBRL文書の構造を言語機能に組み込むことでXBRL文書の変換を容易に行うことを目的としたプログラミング言語，及び，処理系を提案する．本プログラミング言語，及び，処理系を使用することで，XBRL文書の変換を効率よく行うことが期待できる．

2. 財務報告書記述言語 XBRL

2.1 財務報告書

財務報告書とは，企業が定期的に公表する貸借対照表，損益計算書，キャッシュフロー計算書などの財務諸表を記載した文書である [1]．例として，商法による貸借対照表を表 1 に示す．貸借対照表とは，資産，負債，及び，資本を対照表示することにより，企業の財政状態を明らかにする報告書である．投資機関や銀行などは，貸借対照表を始めとする財務報告書を使って企業の安全性を判断し，投資を行うなど，財務報告書は経済活動にとって極めて重要な文書として使われている．

表 1　貸借対照表

株式会社 ○○　　貸借対照表
（平成×年×月×日現在）
（単位　百万円）

(資産の部)		(負債の部)	400
流動資産	470	流動負債	300
現金・預金	150	支払手形	40
受取手形	50	買掛金	30
売掛金	30	短期借入金	30
有価証券	100	短期償還社債	50
製品	120	未払金・諸税金	50
半製品・仕掛品	15	前受金	50
原材料・貯蔵品	35	その他	50
その他	100	固定負債	100
貸倒引当金	△ 130	社債	50
固定資産	600	長期借入金	20
有形固定資産	300	その他	30
建物・構築物	100	(資本の部)	700
機械・装置	50	資本金	400
工具・器具・備品	50	法定準備金	200
土地	50	資本準備金	100
建設仮勘定	50	利益準備金	100
無形固定資産	100	剰余金	100
工業所有権	75	○○準備金	35
その他	25	○○積立金	15
投資等	200	別途積立金	20
投資有価証券	80	当期未処分利益（損失）	15
子会社株式・出資金	20	〔当期利益（損失）〕	15
長期貸付金	50		
その他	70		
貸倒引当金	△ 20		
繰延資産	30		
開発費	30		
合　計	1,100	合　計	1,100

2.2 財務報告書の XBRL 化

財務情報を迅速かつ効率的にに公表するため，財務報告書が web 上で公開されるようになった．しかし，財務報告書の形式が統一されていないために，企業により使用する用語や，その意味が異なっていたり，異なる企業間の財務報告書を比較しづらいといった問題があった．これらの問題を解決するため，2000 年 7 月に XBRL 1.0 [8] が XBRL International により策定された．XBRL とは，各種財務報告書用の情報を作成・流通・利用できるように標準化された，XML の構文を用いた言語である．財務情報を XBRL 文書化することにより，ソフトウェアやプラットフォームに依存しない電子的な財務情報の作成や流通・再利用が期待できる．

我々が提案する処理系は，2001 年 12 月に公開された XBRL 2.0 [8] を対象としている　以下では，XBRL 2.0 について簡単に説明する．

2.3 XBRL 文書の構成

XBRL 文書はタクソノミ文書とインスタンス文書から構成される．タクソノミ文書はタクソノミ本体 (以下，タクソノミ) と 5 種類のリンクベースから構成され，XBRL 文書の語彙と，財務報告書に記される項目間の関係を定義する．インスタンス文書にはタクソノミ文書によって定義された語彙で財務事実が記述される．図 1 に，XBRL 文書の構成図を示す．

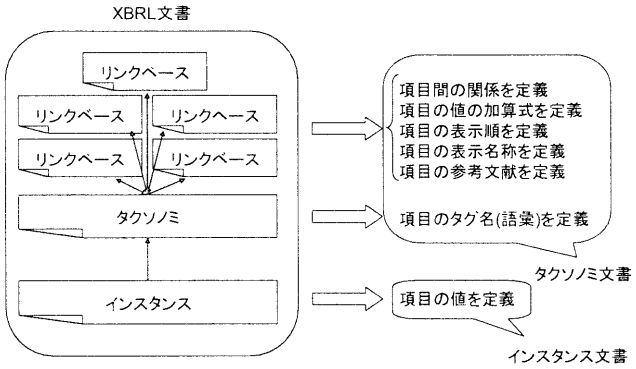

図 1　XBRL 文書の構成

2.4 タクソノミ

タクソノミは，インスタンス文書の語彙 (要素名，属性，データ型) を定義する XML スキーマ [5] 文書である．タクソノミでは，インスタンス文書に記述される要素名の定義や，財務情報を補足するための情報であるタプル要素の定義を行う．さらに，インスタンス文書に記述される項目間の関係を定める，リンクベースを指定する．

2.5 リンクベース

リンクベースは，タクソノミで定義された項目間の関係や，各項目に対する追加情報を XLink [4] の外部リンク機能を利用して定義した文書である．リンクベースには次の 5 種類がある．

- 定義リンク – 項目間の関係を定義
- 計算リンク – 項目の値の加算式を定義
- プレゼンテーションリンク – 項目の表示順を定義
- ラベルリンク – 項目の表示名称を定義
- リファレンスリンク – 項目の参考文献を定義

定義リンク，計算リンク，プレゼンテーションリンクは関係リンクとも呼ばれ，インスタンス文書に記された要素間の関係を定義する．我々の提案する処理系は，定義リンク，計算リンクが定義する関係を利用するため，以下では，これら 2 種類のリンクについて簡単に説明する．

2.5.1 定義リンク

インスタンス文書に記された項目間の，概念上の親子関係や，異なる要素として記されているが，異なる観点から見た同一の概念であることを指定する，等価概念の関係を定義する．

2.5.2 計算リンク

財務報告書では，親項目の値が子項目の値の和として計算される項目がある．計算リンクでは項目間の親子関係と，計算によって値を決定するという関係を定義する．さらに，親項目の値を計算する際に，子項目の値に掛ける重みの値を指定することもできる．

2.6 インスタンス文書

インスタンス文書は，タクソノミ文書で定義された要素で財務事実を記述した XML 文書である．インスタンス文書は，以

```
<?xml version="1 0" encoding="UTF-8"?>
<xbrli.group
 xmlns xbrli="http://www xbrl.org/2001/instance"
 xmlns.xsi="http.//www w3 org/2001/XMLSchema-instance"
 xmlns.iso4217="http://www iso org/4217"
 xmlns:sample="http://sample"
 xsi schemaLocation="http //sample sample xsd">

 <!-- Item-Element starts here -->
 <sample:assets
   numericContext="c1">1100000000</sample.assets>
 <sample:currentAssets
   numericContext="c1">500000000</sample:currentAssets>
 <sample:noncurrentAssets
   numericContext="c1">600000000</sample:noncurrentAssets>
 <sample liabilitiesAndStockholdersEquity
   numericContext="c1">1100000000</sample liabilitiesAndStockholdersEquity>
 <sample liabilities
   numericContext="c1">400000000</sample·liabilities>
 <sample.stockholdersEquity
   numericContext="c1">700000000</sample.stockholdersEquity>

 <!-- Context-Element starts here -->
 <xbrli numericContext id="c1" precision="18" cwa="true">
  <xbrli entity>
   <xbrli.identifier
     schema="www nasdaq.com">SAMP</xbrli:identifier>
   <xbrli segmant/>
  </xbrli entity>
  <xbrli period>
   <xbrli instant>2001-08-16</xbrli.instant>
  </xbrli period>
  <xbrli.unit>
   <xbrli operator id="divide">
    <xbrli measure>ISO4217 JPY</xbrli·measure>
    <xbrli·measure>xbrli:shares</xbrli.measure>
   </xbrli:operator>
  </xbrli.unit>
 </xbrli.numericContext>
</xbrli group>
```

例1　インスタンス文書 sampleInstance.xml

下の5種の要素から構成される.

2.6.1　グループ要素

グループ要素 (group) は，インスタンス文書のルート要素である．他の要素を含む汎用のコンテナとして使うこともできる.

2.6.2　項目要素

財務事実，すなわち，所定のビジネス実体に対して，所定の期間内に報告された事実を記述する要素である．要素名はタクソノミで定義される．財務事実が数値型である項目要素には numericContext 属性が，文字列型である項目要素には nonNumericContext 属性が必ず含まれ，次節で説明するコンテキスト要素への参照を指定する.

2.6.3　コンテキスト要素

コンテキスト要素は，数値コンテキスト要素 (numericContext)，非数値コンテキスト要素 (nonNumericContext) の2種類に分けられる．数値コンテキストは，数値として記述された財務事実に関するメタデータや属性（期間や単位など）を提供する．非数値コンテキストは，文章として記述された財務事実に関して同様の情報を提供する.

2.6.4　タプル要素

タプル要素 (tuple) は，経営者の名前とその肩書きといった，相互に依存している財務事実の集合を記述する．タプル要素では，情報は構造化して記述される.

2.6.5　脚　　注

タプル要素は，財務事実に関連する構造化された情報を記述する．しかし，財務事実との間に不規則に構造化された情報も存在する．この情報を表すためには，タプル要素ではなく，脚注を使う．脚注は，XLink で表現される.

2.7　インスタンス文書の構造

インスタンス文書の構造を規定するスキーマでは，項目要素が子要素を含むことを許していない．グループ要素を使って要素が複雑にネストした構造を書くことができるが，一般的にインスタンス文書は，単純に列挙された XML 文書となっている．例1に，ごく簡単なインスタンス文書の例を示す．例1では，6個の項目要素により財務事実が記述されている．これらの項目要素は，c1 という id によって数値コンテキストに関係付けられている.

3.　XBRL のモデルと操作方法

3.1　インスタンス文書のモデル

本処理系は，インスタンス文書に記述された値の変更や，新しいインスタンス文書の値の設定といった，インスタンス文書に対する操作を扱う．そのため，インスタンス文書を簡単に操作するためのモデルと，そのモデルに対する操作を定義する.

インスタンス文書は XML 文書である．XML 文書のモデルとしては，W3C により Document Object Model(DOM) [3] が公開されており，広く使われている．DOM は，XML 文書の木構造による表現である，DOM ツリーの操作を通じて，任意の XML 文書操作が可能であるという大変優れたモデルである．しかし，DOM をインスタンス文書のモデルとして使うと，複雑すぎて扱いづらいため，DOM を基に新しいモデルを考える.

一般の XML 文書では，マークアップされた文字列データが子要素により分割されうる．分割された文字列データや，その順番も保持するため，DOM ツリーでは文字列データを，テキストノードという子要素として表現する．しかし，インスタンス文書に記された項目要素は，子要素を持たないために，マークアップされた文字列データは分割されない．そこで，要素ノードに文字列データを持たせたモデルを作ることが可能となる．要素ノードに文字列データを持たせると，項目要素が持つ値を参照する際に，子要素の解析をすること無しに値を参照できるため，大幅な省力化が可能となる.

また，インスタンス文書に記された要素間には，次に示す関係がある.

- 要素間の入れ子構造による親子関係
- リンクベースで定義された項目要素間の親子関係
- 項目要素からコンテキスト要素間への参照関係

DOM は，要素間の入れ子構造による親子関係のみを使い木構造を形成するが，XBRL のインスタンス文書を表すためには，上に挙げた3種類の関係全てを使ったモデルを考える必要がある.

本処理系では，要素を頂点とし，要素間の関係を辺とする有向グラフをインスタンス文書のモデルとする．有向グラフの頂点は，要素の値（マークアップされた文字列データ）と，属性を持つ.

例1で示したインスタンス文書の，DOM ツリーによる表現

を図2に，有向グラフによる表現を図3に記す．ただし，簡単のために名前空間接頭辞と，xbrli:numericContext の子要素は省略してある．また，表1の様に次の親子関係がリンクベースで定義されているとする．

- assets(資産の部) は currentAsset(流動資産) と，noncurrentAssets(固定資産) の親
- liabilitiesAndStockholdersEquity (負債の部) は liabilities(負債) と stockholdersEquity(資本) の親

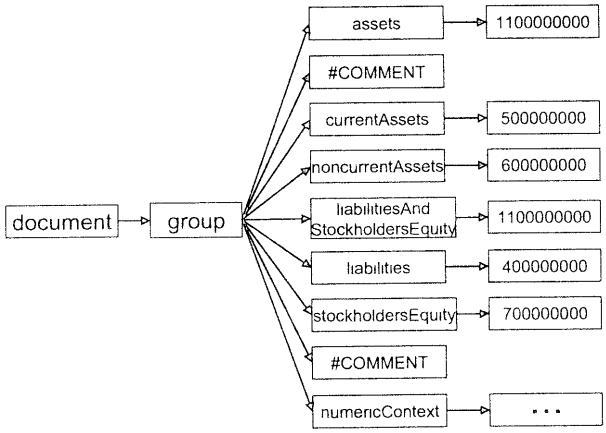

図2　DOM ツリーによる表現

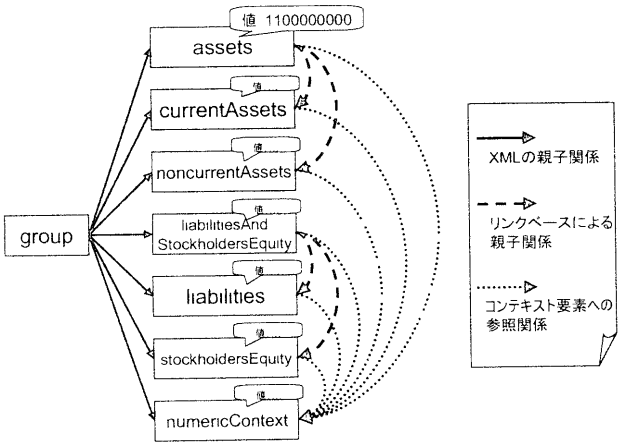

図3　有向グラフによる表現

有向グラフによる表現では，要素のみが頂点として現れるため，頂点の数が DOM ツリーよりも少なくなり，簡単なグラフとなる．さらに，XBRL 文書特有の関係で要点間に辺を引くため，要素間の関係を利用した要素の参照を簡単に行うことができる．有向グラフによるモデルは，DOM ツリーより理解しやすく，操作しやすいモデルであると考えられる．

さらに本研究では，有向グラフの頂点と，DOM ツリー中の要素ノードを同一の頂点で表すことで，インスタンス文書のモデルと DOM ツリーを結合する事を考える．すると，XBRL 文書の操作は，インスタンス文書のモデルのみを使って，一般の XML 文書の操作は，DOM ツリーを使って行うことができる．すなわち，汎用性を持たせたまま，XBRL 文書変換を簡易に行うことができる．

3.2　モデルに対する操作

インスタンス文書への操作は次の3種類がある．

- 値の操作
- 属性の操作
- 構造の操作

値の操作とは，数値，又は文字列として表された財務情報に何らかの計算処理，文字列処理を施すこと，属性の操作とは，属性の追加，削除，及び，値を変更すること，構造の操作とは，要素間の入れ子構造を操作することである．

インスタンス文書のモデルでは，値と属性は有向グラフの頂点が持っているため，値の操作と，属性の操作は，頂点が持っているデータの変更に対応する．構造の操作は，頂点の追加，削除，及び，XML の親子関係による辺（実線の矢印）をつなぎ変えることに対応する．言語処理系では，これらの処理を行う機構を提供する．

4.　言語処理系

4.1　言語の概要

提案する処理系は，XBRL 文書の変換を容易に行うことを第一の目標としている．そのため，変換のためのプログラミング言語は，言語仕様が小さく，かつ，習得しやすいことが望ましい．本研究では，XBRL 文書変換のための言語を Java の構文からクラスを除いた言語として定義する．その理由は次の通りである．

- Java は，広く使われているプログラミング言語であるため，Java の構文を使うことで習得の手間が省かれる
- 想定している利用者層や，変換プログラムの規模を考慮すると，オブジェクト指向という複雑な概念を盛り込むことは好ましくない

これより，変換言語について簡単に説明する．

4.1.1　変数と型

変数は型，及び，変数名の明示的な宣言無しで使用可能とする．変数には型がないが，処理系内部では，整数型，浮動小数点型，文字列型，配列型，ハッシュ型，4.2.1節で述べる Node 構造体型がある．変数に値が代入された時，右辺の型が左辺の変数の型として設定される．本言語は財務報告書を扱う言語であるため，財務情報を扱う整数型は桁あふれを起こしてはならない．そのため，整数型は任意精度の演算をサポートする．

4.1.2　演算子

算術演算子，代入演算子ともに Java で定義されている演算子を持つ．例えば，整数型と文字列型を加算演算子 “+” で演算しようとしたときなど，異なる型同士に対して演算を行う場合には，処理系内部で自動型変換を行う．この場合は，整数型が文字列型に変換され，演算結果は2個の文字列が結合された文字列型となる．ただし，整数型と文字列型を乗算演算子 “*” で演算しようとしたときなど，自動型変換を行えないときはエラーとする．

4.1.3　制御構造

Java で定義されている制御構造のうち，例外（try-catch 文）を除く，if 文，while 文，do-while 文，for 文，switch 文を持

つ．さらに，break 文による，ループ脱出と，continue 文によるループの再開をサポートする．

4.1.4 関　　数

オブジェクト指向言語 Java には，関数という概念はないが，Java のメソッドを本研究では関数として定義する．ただし，型による関数の区別ができないため，プログラム中で関数名が重複してはならない．関数では，任意個の引数を受け取り，指定された計算を行う．return 文で計算結果を返すことができるが，必ずしも値を返す必要はない．

4.1.5 入　出　力

本言語では，XML ファイルとの入出力のみを考える．指定されたインスタンス文書を解析して，XML 文書のルートノードを返す機能と，指定したノードを持つ木構造全体をファイルに保存する機能を関数として提供する．

4.2 XBRL/XML 文書への参照機能

4.2.1 Node 構造体

3.1 節で述べたモデルで，頂点を表現する構造体を定義する．この構造体を Node 構造体と呼ぶ．Node 構造体は，DOM のノードも表現するため，DOM で定められた Node インタフェースを基に定義する．表 2 に，Node 構造体が持つメンバ一覧とその意味を示す．

表 2　Node 構造体のメンバ一覧

型	メンバ名	意味
string	name	要素名
int | string | float	value	要素の値
Hashtable	attributes	属性
Node	parent	親の要素ノード
Node 配列	children	子の要素ノードの配列
string	dom_nodeName	DOM による要素名
string	dom_nodeValue	DOM による要素の値
int	dom_type	Node の種類
Node	dom_parentNode	親ノード
Node 配列	dom_childNodes	子ノード
Node	dom_firstChild	最初の子ノード
Node	dom_lastChild	最後の子ノード
Node	dom_previousSibling	直前の兄弟ノード
Node	dom_nextSibling	直後の兄弟ノード

name，value，attributes は，DOM ツリーのデータにアクセスする読み書き可能なメンバである．構造体自身が，要素ノードを表している時のみアクセス可能で，それぞれ，要素名，要素の値，属性を DOM ツリーを解析して動的に計算する．ノードに複数のテキストノードが存在するときは文字列型として結合した結果を value の値とする．XBRL の項目要素では，財務事実は単一のテキストノードとして表現されるため，value メンバの読み書きで一番需要の多い XBRL 文書の変換である値の操作を実現できる．

parent，children は，DOM ツリーの構造にアクセスする読み取り専用のメンバである．インスタンス文書のモデルで，親子関係の辺を表現する．動的に DOM ツリーを解析して parent は，親要素となるノードを返し，children は子要素の配列を返す．テキストノード，属性ノードなど，DOM ツリーに現れる要素を表すノード以外のノードを返す可能性のある dom_parentNode，dom_childNodes と異なり，parent, children は，XML の要素を表すノードのみを返す．

4.2.2 親子関係によるノードの取得

インスタンス文書のモデルで，リンクベースで定義された親子関係の辺を参照するための関数を提供する．

`Node xbrl_getParentNode(Node node)`
node の親要素を返す．

`Node xbrl_getChildNodeByNum(Node node, int index)`
node の子要素で，index 番目の要素を返す．

`Node[] xbrl_getChildNodes(Node node)`
node の子要素の配列を返す．

これらの関数を使うことで，実際の財務諸表に現れる項目間の親子関係で項目要素にアクセスすることができる．

4.2.3 コンテキスト要素の取得

インスタンス文書のモデルで，項目要素から，それに関連付けられたコンテキスト要素を参照する辺を辿る関数を提供する．

`Node xbrl_getContext(Node node)`
node に関連付けられたコンテキスト要素を返す．

4.2.4 要素名によるノードの取得

構造体のメンバや，ノードを参照する関数を使うと，要素間，ノード間の関係からノードを得ることができる．DOM では，このほかに指定したノードを直接取得する関数が 2 個定義されている．本処理系でも同じ機能を持つ関数を提供する．

`Node[] getElementByTagName(Node node, string name)`
node の子要素で，name という要素名のノードの配列を返す．

`Node getElementById(Node node, string id)`
node の子要素で指定された id を持つノードを返す．

4.3 構造操作のための機能

本処理系では，DOM で定義されている関数を用いて，XML 文書の構造を操作する．関数名と機能は次の通りである．

`Node insertBefore(Node newNode, Node refNode)`
refNode の直前に newNode を挿入し，newNode を返す．

`Node replaceNode(Node newNode, Node oldNode)`
oldNode を newNode で置き換え，oldNode を返す．

`Node removeNode(Node oldNode, Node parent)`
parent から子ノードである oldNode を削除し，oldNode を返す．

`Node appendNode(Node newChild, Node parent)`
parent に newChild を子ノードとして追加し，newChild を返す．

`bool hasChildNodes(Node parent)`
parent に子ノードがあれば真，なければ偽を返す．

`Node closeNode(Node node, bool deep)`
deep が真の時は，node の子要素も再帰的に複製して返す．偽の時は，node のみを複製して返す．

`Node createNode()`
新しいノードを作成する．

Node 構造体を使うことで DOM で定義されたツリー構造を

形成することができ，上記の関数を使うことで，ツリー構造を変形し，任意のツリーを作ることができる．すなわち，DOMで可能なXML操作が本処理系で可能となる．

4.4 XBRL文書特有の計算処理

2.5.2節で述べたように，XBRL文書ではある項目要素の値が，その子要素の値の重み付き合計として計算される場合があるが，この計算を行う関数として，次の2個を提供する．

`void xbrl_calculateNode(Node node)`

引数として与えたnodeの値を計算する．

`void xbrl_calculateAllNodes(Node node)`

nodeが含まれるインスタンス文書が持つ項目要素の値を全て計算する．戻り値はない．

既存の処理系で子要素からの値の計算を行う場合は，まず，どの要素と親子関係にあるかを計算リンクから調べる．次に，合計する際の重みを取得し，最後に，親子関係にある要素の値を取得して計算する．この手間のかかる計算を行う機能を提供することで，大幅な省力化が可能となる．

4.5 実行環境

本処理系は，簡単なプログラムを書き，トライ&エラーを繰り返し目的のXBRL文書を作成する，という作業を想定している．使い勝手の良さや，利用者へのフィードバック，デバッグの容易さから，処理系はインタープリタで実装する．

4.6 サンプルプログラム

図4にサンプルプログラムを示す．このプログラムでは，例1のインスタンス文書を読み込み，円からドルへ，項目要素の単位の換算を行っている．

各文が行っている具体的な処理はコメントとして付加している．プログラムを見て分かるように，XBRL文書の解析を行う処理を書くことなく，行いたい処理，この場合は単位の換算に関わる処理のみを簡潔に書くことができる．

```
// ファイルを開く
rootnode = open("sampleInstance.xml");

// 要素の取得
assets = getElementByTagName(rootnode, "sample:assets")[0];
LandS = getElementByTagName(rootNode,
            "sample:liabilitiesAndStockHoldersEquity")[0];

// 子要素の配列を取得
assets_child = xbrl_getChildNodes(asset);
LandS_child = xbrl_getChildNodes(LandS);

// 子要素全てに対して，計算をする
// rateは為替レート
for (i=0; i<length(assets_child); i++) {
    assets_child[i].value *= rate;
}

for (i=0; i<length(LandS_child); i++) {
    LandS_child[i].value *= rate;
}

// 子要素からの値の計算
xbrl_calculateNode(assets);
xbrl_calculateNode(LandS);

// 同一ファイルに保存する
save(rootnode);
```

図4 サンプルプログラム

5. 考　察

XSLT[6]をはじめとして，XML文書の形式を変換するための枠組みに関する研究が広く行われている．XSLTは制御構造や算術演算をサポートし，本研究で想定している財務報告書の変換も行うことができる．しかし，XSLTにはXBRLの持つ複雑な構造を簡単に扱う手段がない．すなわち，本来行いたい変換処理のプログラムに加え，タクソノミ文書を解析処理するためのプログラムも書かなければならない．そのため，XSLTを使ったXBRLのためのプログラムは大変複雑になる．また，最近注目されている変換言語にXDuce[2]がある．XDuceはXMLのスキーマをデータ型と見なし，生成するXML文書の妥当性を静的に確かめる．XDuceもXSLTと同じ問題を含み，変換のためのプログラムは複雑で難しいものとなる．

本稿で提案した処理系は，XBRL文書に対する簡単なアクセス方法を提供する．利用者は，タクソノミ文書の解析処理など意識せず，目的の変換処理の記述のみに集中できる．そのため，既存の言語に比べると変換に必要なプログラム量は大幅に減少し，高い生産性をもたらすと考えられる．さらに，DOMツリーにアクセス可能な構造体，関数群を使うと，DOMツリーに対する操作も可能のため，XBRL文書の変換にとどまらず，一般のXML文書の変換にも用いることができると考えられる．

本処理系は，XDuceのようなXML文書の妥当性を検証する機能を持っていない．生成した文書を外部のValidatorで検証するという方式では，プログラム中のどの部分に誤りがあるのかが分からない．妥当性の検証機能を組み込むことでプログラムのデバグが容易になることが考えられる．

6. まとめ

XBRL文書の構造を利用した，XBRL文書の変換処理を簡単に記述するための処理系を提案した．また，この処理系はXBRL文書の変換にとどまらず，一般のXML文書の変換にも使えることを示した．今後の課題として，妥当性の検証機能の組み込み，処理系の実装，及び，評価が挙げられる

文　献

[1] 安平昭二，“入門　企業会計　(三訂版)”，東京経済情報出版

[2] Haruo Hosoya and Benjamin C. Pierce “XDuce: A typed XML processing language” ACM Transactions on Internet Technology, 3(2):117-148, 2003

[3] The World Wide Web Consortium, “Document Object Model (DOM)”, http://www.w3.org/DOM/

[4] The World Wide Web Consortium, “XML Linking Language (XLink)”, http://www.w3.org/TR/xlink/

[5] The World Wide Web Consortium, “XML Schema”, http://www.w3.org/XML/Schema

[6] The World Wide Web Consortium, “XSL Transformations (XSLT)”, http://www.w3.org/TR/xslt

[7] XBRL International, “XBRL”, http://www.xbrl.org/

[8] XBRL International, “XBRL Specifications”, http://www.xbrl.org/resourcecenter/specifications.asp