特別研究報告

題目

項目間の対応関係を用いたXBRL財務報告書自動変換ツールの試作

指導教官
井上 克郎 教授

報告者
渡辺 貴史

平成 15 年 2 月 19 日

大阪大学 基礎工学部 情報科学科

平成 14 年度 特別研究報告

項目間の対応関係を用いた XBRL 財務報告書自動変換ツールの試作

渡辺 貴史

内容梗概

企業の財務情報を社外に広報する際には，財務報告書が使われるのが一般的である．財務報告は，年度ごと，あるいは業種や企業ごとに文書構造や項目，計算式などが異なるといった特徴がある．近年，XBRL と呼ばれる XML を基盤とした財務報告記述言語が提案されており，計算機上で財務情報を扱う標準的な枠組に関する研究が進められている．財務情報を XBRL データ化することで，データの交換や分析作業のコストを削減することが可能となる．しかし，XBRL 文書の作成作業は依然として人手に依存する要素が多いため，誤記や編集ミスによって財務報告書の信頼性が低下する問題がある．

そこで本研究では，文書作成の効率化と信頼性の向上を目指して XBRL 文書間の自動変換手法を考案した．同じ財務事実を記述した報告書であれば，文書形式が異なっていてもその内容には共通の項目が含まれている．そして，既存の報告書と作成する報告書の間での共通項目の対応がわかれば，その対応関係を用いて既存の報告書から項目値を取得することができる．このことから，対応関係を定義することができれば，その定義を用いて各項目の値を取得していくことで，XBRL 文書を自動的に変換することができると考えられる．

さらに本研究では，考案した手法を用いて XBRL 文書間の自動変換を行うツールの試作を行った．本ツールは，XBRL 財務報告書の文書形式が定義されている XBRL タクソノミ文書から，相互に関連するデータ項目に関する情報を取得し，データ項目間の対応関係を記述したデータを用いて，変換元の XBRL 財務報告書から変換先の XBRL 財務報告書を生成する．また，対応関係を記述する際には，条件節やデータ加工式をあわせて記述することにより，元の文書の一部を抽出することや，いくつかのデータを統合した結果を生成する文書に挿入することが可能である．本ツールを用いることにより，信頼性の高い財務報告文書作成を効率的に行うことが可能になると期待できる．

主な用語

XBRL(eXtensible Business Reporting Language)
財務報告書 (financial report)

タクソノミ (taxonomy)
自動変換 (automatic conversion)

目 次

1　まえがき

企業の財務情報を社外に広報する際には，財務報告書が使われるのが一般的である．財務報告は，年度ごと，あるいは業種や企業ごとに文書構造や項目，計算式などが異なるといった特徴がある．近年，XBRL(eXtensible Business Reporting Language)[13][8][6] と呼ばれる XML(eXtensible Markup Language) を基盤とした財務報告記述言語が提案されており，計算機上で財務情報を扱う標準的な枠組に関する研究が進められている．財務情報を XBRL データ化することで，データの交換や分析作業のコストを削減することが可能となり，投資家やアナリストなどへの情報の提供速度も向上する．しかし，XBRL 文書の作成作業は以前として人手に依存する要素が多いため，誤記や編集ミスによって財務報告書の信頼性が低下する問題がある．

そこで本研究では，文書作成の効率化と信頼性の向上を目指して XBRL 文書間の自動変換手法を考案した．同じ財務事実を記述した報告書であれば，文書形式が異なっていてもその内容には共通の項目が含まれている．そして，既存の報告書と作成する報告書の間での共通項目の対応がわかれば，その対応関係を用いて既存の報告書から項目値を取得することができる．このことから，対応関係を定義することができれば，その定義を用いて各項目の値を取得していくことで，既存の XBRL 文書からの自動変換が可能となると考えられる．

さらに本研究では，考案した手法を用いて XBRL 文書間の自動変換を行うツールの試作を行った．本ツールは，XBRL 財務報告書の文書形式が定義されている XBRL タクソノミ文書から，相互に関連するデータ項目に関する情報を取得し，XBRL 文書間の共通するデータ項目の対応関係を記述したデータを用いて，変換元の XBRL 財務報告書から変換先の XBRL 財務報告書を生成する．また，対応関係を記述する際には，条件節を用いることで元の文書の一部を抽出することが可能である．さらに，データ加工式の記述により，データ項目間の単位変換などが可能である．これらの記述を用いることで，文書形式が異なる XBRL 文書間でも適切な変換処理を行うことができる．本ツールを用いることにより，信頼性の高い財務報告文書作成を効率的に行うことが可能になると期待できる．

以下，2 節では XBRL について，3 節では自動変換手法について説明する．4 節で今回試作した XBRL 財務報告書変換ツールの機能と処理手順を述べ，5 節で考察を述べる．6 節でまとめと今後の課題について述べる．

2　財務報告記述言語 XBRL

XBRL(eXtensible Business Reporting Language) は，各種財務報告用の情報を作成・流通・利用できるように標準化された XML(eXtensible Markup Language) をベースとする言語である．財務情報を XBRL データ化することにより，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・再利用が可能となる．XBRL を使用することにより，コストの削減や財務情報の提供速度の向上が期待される．XBRL 文書はタクソノミ文書とインスタンス文書から成る．タクソノミ文書はタクソノミ本体 (以下，タクソノミ）とリンクベースからなり，財務報告書の文書形式を定義する．タクソノミ文書によって定義された文書形式に従ってインスタンス文書に財務情報を記述する．[5]

2.1　財務報告

表 1: 貸借対照表

会社名	貸借対照表 （平成×年×月×日現在） (単位：○円)		
(資産の部)		(負債の部)	××××
流動資産	××××	流動負債	××××
現金・預金	×××	支払手形	×××
受取手形	×××	買掛金	×××
売掛金	×××	短期借入金	×××
有価証券	×××	短期償還社債	×××
製品	×××	未払金・諸税金	×××
半製品・仕掛品	×××	前受金	×××
原材料・貯蔵品	×××	その他	×××
その他	×××	固定負債	××××
貸倒引当金	△×××	社債	×××
固定資産	××××	長期借入金	×××
有形固定資産	××××	その他	×××
建物・構築物	×××	(資本の部)	××××
機械・装置	×××	資本金	××××
工具・器具・備品	×××	法定準備金	××××
土地	×××	資本準備金	×××
建設仮勘定	×××	利益準備金	×××
無形固定資産	××××	剰余金	××××
工業所有権	×××	○○準備金	×××
その他	×××	○○積立金	×××
投資等	××××	別途積立金	×××
投資有価証券	×××	当期未処分利益 (損失)	×××
子会社株式・出資金	×××	〔当期利益 (損失)〕	×××
長期貸付金	×××		
その他	×××		
貸倒引当金	△×××		
繰延資産	××××		
開発費	×××		
合　計	××××	合　計	××××

財務報告とは，企業が定期的に公表する貸借対照表，損益計算書など財務諸表を含む報告のことである．例として，商法による貸借対照表を表 1 に示す．貸借対照表は資産・負債・資本の残高を一覧表示することで企業の財政状況を表している．各項目に値を記入すること

で企業が調達した資金とその運用形態がわかるのである．

財務報告の作成においては，多くの会計データ，企業組織等に関する説明文書，監査人の報告などの様々な素材情報をもとに，いくつかの種類の報告を作成しなければならない．財務報告の会計基準は国，業種，企業によって異なり，内容や形式にも，企業グループでの連結会計の導入，国際関係基準への段階的移行などにより変更が加えられている．さらに，事業別，地域別，セクション別の財務報告や四半期ごとの報告を求められることもある．このように財務報告の作成は複雑であり，その作成作業の効率化は非常に重要となっている．

2.2 財務報告の XBRL データ化

アナリストや投資家は，財務報告を比較分析することで投資対象を決定していく．そのため，鮮度がよく，精度が高く，扱いやすい財務情報の提供が重要となる．財務情報の提供が複雑になっていることに加えて，その作成作業は人手に依存する部分が多く非効率的であり，誤記や編集ミスが発生しやすい．また，会計基準が異なる財務報告を比較することも困難であった．

これらの問題を解決するため，2000 年 7 月に米国公認会計士協会によって XBRL が公開された．XBRL は，XML をベースとする財務報告記述のための標準化言語である．XBRL は XML ベースであることから，異なる形式の財務報告であってもスキーマによりその形式を定義でき，財務情報の柔軟な記述が可能である．財務情報を XBRL データ化することで，財務報告書の作成を正確に，かつ効率的に行うことができるようになった．さらに，財務データを共通の XBRL データに変換することにより，財務報告の比較が容易に行えるようになった．

2.3 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，属性など) を定義する．定義は次のように記述される．

```
<element name="assets" type="xbrli:decimalItemType" xbrli:balance=""
 id="assets" substitutionGroup="xbrli:item"/>
```

name 属性で要素名を，type 属性で要素の型を指定する．要素の型は以下の 6 種類である．

- 数値を値として取る型
  - xbrli:decimalItemType
  - xbrli:monetaryItemType

- xbrli:sharesItemType

- 文字列を値として取る型
  - xbrli:stringItemType
  - xbrli:uriItemType
  - xbrli:dateTimeItemType

また，id 属性でこの要素の ID を指定し，xbrli:balance 属性でその要素の表す財務情報が資本情報なのか負債情報なのかを指定する．

2.4 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定義する．リンクベースには定義リンク，計算リンク，プレゼンテーションリンク，リファレンスリンク，ラベルリンク，脚注リンクがある．

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

以下，定義リンクと計算リンクについての説明を行う．

2.4.1 定義リンク

定義リンクではロケータと定義アーク要素により要素間の関係が定義される．
ロケータは次のように記述される．

```
<loc xlink:type="locator" xlink:href="tax.xsd\#assets"
 xlink:label="tax_assets" xlink:title="assets"
 xlink:role="http://www.xbrl.org/linkprops/locator/root"/>
```

- xlink:href 属性

  タクソノミで定義されている要素を参照するための属性である．

- xlink:label 属性

  xlink:href 属性で参照される要素に対するリンクベース内でのラベルを指定する．要素間の関係定義はすべてこのラベルを用いて記述される．

- xlink:role 属性

  要素の役割を指定する．上記の例の場合はこの要素が要素の木のルートであることを示している．

また，定義アーク要素は次のように記述される．

```
<definitionArc xlink:type="arc" xlink:show="replace"
 xlink:actuate="onRequest" xlink:from="tax_assets"
 xlink:to="tax_currentAssets" xlink:title="Go Up to tax_currentAssets"
 xlink:arcrole="http://www.xbrl.org/linkprops/arc/parent-child"/>
```

- xlink:from 属性， xlink:to 属性

  定義アーク要素によって関連付けられる要素を指定する．

- xlink:arcrole 属性

  データ項目間の関係を定義する．属性値には次の 4 つがある．

  http://www.xbrl.org/linkprops/arc/child-parent ：子から親へのアーク
  http://www.xbrl.org/linkprops/arc/parent-child ：親から子へのアーク
  http://www.xbrl.org/linkprops/arc/dimension-element ：概念等価な要素間のアーク
  http://www.xbrl.org/linkprops/arc/element-dimension ：概念等価な要素間のアーク

要素の値は関連付けられた子要素や概念等価な要素からも計算できるため，これらの関連付けは重要である．

2.4.2 計算リンク

計算リンクではロケータと計算アーク要素により計算に必要な情報が記述される．ロケータの記述は定義リンクと同様である．計算アーク要素は定義アーク要素の属性に重みを表す weight 属性が加わる．計算アーク要素は次のように記述される．

```
<calculationArc xlink:type="arc" xlink:show="replace"
 xlink:actuate="onRequest" xlink:from="tax_currentAssets"
 xlink:to="tax_assets" xlink:title="Go Up to tax_assets"
 xlink:arcrole="http://www.xbrl.org/linkprops/arc/child-parent"
 weight="1"/>
```

weight 属性は子要素の値から親要素の値を計算するときの子の値の重みを指定するための属性である．

2.4.3 アークの上書き

リンクベースのアークは上書きすることが可能である．上書きを可能とするため，以下の2つの属性が用意されている．

- priority 属性

  priority 属性は整数値を取る属性で，アークの優先度を表す．規定値は 0 である．同じ要素間のアークが複数存在する場合には，priority 属性の値を比較し，より大きい値を持つアークが優先される．仮に同じ要素間のアークが複数存在し，その priority 属性が等しい場合には，その振る舞いはアプリケーションに依存する．

- use 属性

  use 属性は required，optional，prohibited の 3 種類の値をとる属性で，そのアークの利用に関する情報を表す．

  - required

    そのアークに関連付けられたデータ項目のうち一方の要素が出現すると，もう一方の要素も必ず出現しなければならない．

  - optional

    リンクがトラバーサルの候補であることを示す．use 属性の規定値である．

  - prohibited

    リンクがトラバーサルすべきでないことを表す．

アークの上書きの例

```
(1)<definitionArc xlink:type="arc" xlink:show="replace"
    xlink:actuate="onRequest" xlink:title="Go Up to tax_currentAssets"
    xlink:from="tax_assets" xlink:to="tax_currentAssets"
    xlink:arcrole="http://www.xbrl.org/linkprops/arc/parent-child"/>
(2)<definitionArc xlink:type="arc" xlink:show="replace"
    xlink:actuate="onRequest" xlink:title="Go Up to tax_currentAssets"
    xlink:from="tax_assets" xlink:to="tax_currentAssets"
    xlink:arcrole="http://www.xbrl.org/linkprops/arc/parent-child"
    priority="1" use="prohibited"/>
```

(1) のアークのトラバーサルを禁止するためには (2) のアークを追加する．priority 属性の規定値は 0 なので，priority 属性に 1 を，use 属性に prohibited を指定したアーク型要素を記述することで上書きできる．

```
<?xml version="1.0" ?>
<xbrli:group xmlns:xbrli="http://www.xbrl.org/2001/instance"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://tax1 tax1.xsd"
       xmlns:tax1="http://tax1" xmlns:xhtml="http://www.w3.org/1999/xhtml"
       xmlns:xlink="http://www.w3.org/1999/xlink"
       xmlns:link="http://www.xbrl.org/2001/XLink/xbrllinkbase">
  <tax1:balanceSheet id="balanceSheet" numericContext="c1"/>
  <tax1:assets id="assets" numericContext="c1">8000</tax1:assets>
  <tax1:currentAssets id="currentAssets" numericContext="c1">1000</tax1:currentAssets>
  <tax1:noncurrentAssets id="noncurrentAssets" numericContext="c1">5000
  </tax1:noncurrentAssets>
  <tax1:longTermInvestments id="longTermInvestments" numericContext="c1">2000
  </tax1:longTermInvestments>
  <tax1:assetsHeldForSale id="assetsHeldForSale" numericContext="c1">3000
  </tax1:assetsHeldForSale>
  <tax1:liabilitiesAndstockholdersEquity id="liabilitiesAndstockholdersEquity"
    numericContext="c1">8000</tax1:liabilitiesAndstockholdersEquity>
  <tax1:liabilities id="liabilities" numericContext="c1">2000</tax1:liabilities>
  <tax1:stockholdersEquity id="stockholdersEquity" numericContext="c1">6000
  </tax1:stockholdersEquity>
  <tax1:preferredStock id="preferredStock" numericContext="c1">3500</tax1:preferredStock>
  <tax1:commonStock id="commonStock" numericContext="c1">2500</tax1:commonStock>
```

図 1: インスタンス文書の例

2.5 インスタンス文書

タクソノミ文書で定義された要素で財務情報を記述する．各要素はコンテキスト属性によってコンテキスト要素と関連付けられていなければならない．コンテキスト要素とはインスタンス文書内でデータ項目に関する財務情報を記述したもので，データ項目の値の単位や精度などの情報を付加する．要素は，どのコンテキスト要素と関連付けられているかによって同じ名前の要素であっても識別が可能である．要素間の関係はリンクベースを参照することで得られるので，図 1 のようにすべての要素を同じレベルで記述することができる．

2.6 問題点

財務情報を XBRL データ化することにより，データの比較・分析作業が効率化された．また，情報の電子化とインターネット XML 基盤の利用によって情報の提供速度が向上した．さらに XBRL データからの財務報告書の自動生成などにより効率と信頼性も向上した．

XBRL データにはこのような様々な利点があるが，財務情報を XBRL データ化する作業や XBRL データから別の XBRL データへと変換する作業には手作業の部分が多い．そのた

め，XBRL データの生成は非効率的で信頼性も低くなっている．

そこで，この問題を解決するために XBRL 文書間の自動変換を考える．変換により XBRL 文書の生成を自動的に行えるようになれば，財務報告の作成作業がより効率化され，信頼性も向上すると考えることができる．

3　自動変換手法

本節では，XBRL データの作成作業の問題点を解決するために考案した自動変換手法について述べる．この手法を用いることで XBRL データの作成を自動的に行い，作成作業の効率化と信頼性の向上を実現する．

3.1　変換手法の概要

財務報告書には様々な形式が存在するが，同じ財務活動についての報告であれば，その内容には同じ項目が含まれているはずである．財務情報として等価な項目であれば，その値も等価か，あるいは定式による加工で相互に変換可能であると考えられる．また，必要なデータと不必要なデータを識別ができれば，必要なデータの値のみをもつ文書を生成することができる．これらのことから，要素間の対応関係，必要なデータを識別するための条件，値の変換式の 3 点についての情報が得られれば，自動変換が可能となる．

この考え方を XBRL 文書に適用することで，XBRL 文書間の自動変換が可能となる．自動変換の流れは次のようになる．

1. 作成する XBRL 文書の要素と対応関係にある変換元の XBRL 文書の要素を探して値を取得する
2. 条件判定を行うことで必要なデータであるかを識別する
3. 値の加工が必要であれば加工式を用いて加工する

これらの変換処理を行うには，対応関係の定義を記述した文書が必要となるが，XBRL 文書には他の XBRL 文書の要素との対応関係に関する情報は記述されていない．そこで，対応関係の定義を別の文書に記述することとしてその記述方法を定めた．

3.2　データ項目間の対応

データ項目間の対応はタクソノミで定義された要素を関連付けることで定義する．対応関係はロケータとアーク要素に加えて，条件指定を行うための要素を用いて記述する．ロケータは定義リンクや計算リンクと同様である．アーク要素は次のように記述する．

```
<equivalentArc xlink:actuate="onRequest" xlink:arcrole="element-element"
 xlink:from="tax_assets" xlink:show="replace" xlink:title="equivalent"
 xlink:to="tax2_assets2" xlink:type="arc" calculation="true"
  value="true" cond="cs1"/>
```

- xlink:arcrole 属性

  アークで対応付けられる要素の種類を表す．属性値は次の 4 種類である．

  element-element　：要素と要素を対応付ける
  element-attribute　：要素と属性を対応付ける
  attribute-element　：属性と要素を対応付ける
  attribute-attribute　：属性と属性を対応付ける

- calculation 属性

  子要素の値を利用して親要素の値の計算を行うかを指定する．値には true か false をとり，true であれば値を計算する．規定値は false である．変換元のインスタンス文書のデータ項目値をそのまま記述できない場合にはこの属性により値を求める．

- value 属性

  変換先のインスタンス文書の要素に値が必要であるかを指定する．値としては true か false をとり，true であれば値を記述する．規定値は true である．変換元のインスタンス文書のデータ項目値をそのまま記述できず，さらに計算によっても正しい値が求められない場合にはこの属性により値を記述しないように指定することができる．

- cond 属性

  対応に条件を設定する．条件と判定結果後の処理については conditionalStatement 要素で定義されるので cond 属性値で ID を指定する．

3.3　条件付き対応

対応付けに設定できる条件は，以下の 4 種類の要素を用いて定義する．

- conditionalStatement 要素

  condition 要素，formula 要素，str 要素を利用して条件と値の加工を指定する．加工には，数値の場合は四則演算，文字列の場合は結合を用いる．

- condition 要素

  条件の定義をする．

- formula 要素

  数値の加工に用いる定式を定義する．

- str 要素

  文字列の加工に用いる文字列を定義する．

以下，各要素の説明を行う．

3.3.1 conditionalStatement 要素

アークによる対応付けの条件とその判定結果による値の加工を指定するための要素で，以下のように記述する．

```
<conditionalStatement if="exp1" then="n * f1" else="n * 100" id="cs1"
 type="numeric"/>
```

- id

  アーク型要素から参照される ID である．

- if 属性

  condition 要素を用いた条件式の指定を行う．指定は condition 要素の id 属性と and, or，(，) を用いて行う．条件が真であれば対応関係が成立し，偽であれば対応関係は不成立となる．対応が不成立となるとその要素は不必要であると判断されるため，その要素は生成しない．

  値を加工する場合には条件が真ならば then 属性で指定する式で値を加工し，偽ならば else 属性で指定する式で値を加工する． この属性は省略可能で，省略された場合には条件が真であるとして then 属性で指定された値の加工を行う．

- then 属性

  条件判定が真の場合の値の加工式を指定する．指定は値が数値ならば formula 要素の id 属性と+，-，*，/，(，)，n を用いて行う．また，値が文字列ならば str 要素の id 属性と+，n を用いて行う．n は加工する前の値を表す．

  formula 要素と str 要素を用いずに指定することも可能だが，文字列の加工で文字としての”+”や”n”を使用するには str 要素で定義する必要がある．この属性は省略可能で，省略されて条件判定が真となった場合には加工は行わない．

- else 属性

  条件判定が偽の場合の値の加工式を指定する．指定方法は then 属性と同様である．この属性も省略可能であり，省略されて条件判定が偽となった場合には対応が不成立となる．

条件処理の流れを図 2 に示す．条件判定の結果と各属性の有無により最終的に得られる値は「加工しない」,「then 属性の式で加工」,「else 属性の式で加工」,「対応する要素の記述無し」の 4 パターンが考えられる．

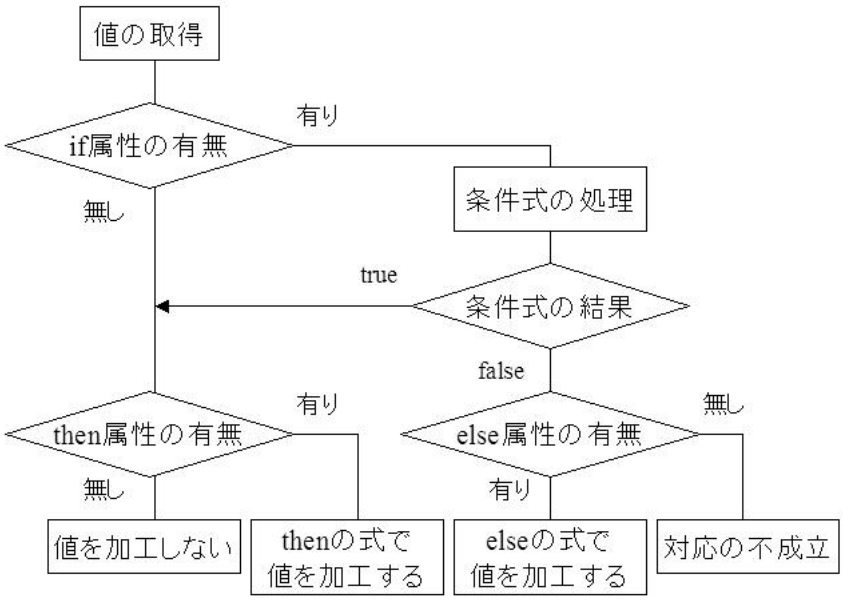

図 2: 条件処理の流れ

3.3.2 condition 要素

条件の具体的な内容を指定するための要素で，以下のように記述する．

```
<condition id="exp1" xlink:href="C:\test\testData\tax.xsd#assets"
 value="10" operator="greater than" type="element"/>
```

- id 属性

  conditionalStatement 要素から参照される ID を設定する．

- xlink:href 属性

  条件の対象となる要素あるいは属性を指定する．対象が要素の場合にはこの属性は省略可能で，省略されると条件が指定されたアークの xlink:to 属性の要素を指定したことになる．

- value 属性

  条件の判定基準となる値を設定する．

- operator 属性

  条件判定が真になるための，xlink:href 属性で指定された要素あるいは属性の値と value 属性値との関係を設定する. 値は表 2 の 6 種類である．

表 2: operator 属性値の候補

演算子	真となるための条件
greater than	value の値よりも href で指定された要素 (属性) の値のほうが大きい
greater or equal	value の値よりも href で指定された要素 (属性) の値のほうが大きいあるいは値が等しい
less than	value の値よりも href で指定された要素 (属性) の値のほうが小さい
less or equal	value の値よりも href で指定された要素 (属性) の値のほうが小さいあるいは値が等しい
equal	value の値と href で指定された要素 (属性) の値が等しい
not equal	value の値と href で指定された要素 (属性) の値が異なる

- type

  xlink:href 属性の参照が要素へのものならば element を，属性へのものならば attribute を値として持つ．この属性は省略可能で，規定値は element である．

上記の例では「tax.xsd というタクソノミのデータ項目 assets の値が 10 よりも大きい場合に真」という条件になる．

3.3.3 formula 要素

数値の値を加工するときに利用する定式を定義するための要素で，以下のように記述する．

```
<formula id="f1" exp="n + 10"/>
```

- id 属性

  conditionalStatement 要素から参照される ID である．

- exp 属性

  値の加工に使用する定式を設定する．式には+，-，*，/，(，)， n を使用できる．ここで，変数 n はアークの対応付けによって取得した変換元のインスタンス文書の要素の値を表す．上記の例の場合，”n + 10”は取得した値に 10 を加えるという意味になる．

3.3.4 str 要素

文字列の値を加工するときに利用できる文字列を定義するための要素で，以下のように記述する．

```
<str id="s1" exp="START + n"/>
```

- id 属性

  conditionalStatement 要素から参照するための ID である．

- exp 属性

  値の加工に使用できる文字列を設定する．formula 要素とは異なり，属性値の文字列はそのままの形で使用される．上記の例の場合，アークによって取得した値が”ABC”であったとしても str 要素が表す文字列は”START ABC”ではなく”START + n”になる．

3.4 変換の例

図 3 は対応定義を用いた変換の例である．この例では assets 要素と assets2 要素の対応に条件が指定されている．条件の内容は「assets の値が 3000 以上である場合のみ対応が成立する」である．図 3 の上の図は assets の値が 5000 なので条件判定が真となり条件指定がない場合と同様に対応が成立する．一方，下の図では assets の値が 2000 なので条件判定が偽となり対応が不成立となる．対応が不成立となるとそのデータは必要ないものであると判断されるので変換先では記述されなくなる．このため，assets2 以下の項目はすべて記述されないことになる．

また，図 4 は値の加工の指定を行った変換の例である．この例では図 3 と同じ条件の下で判定を行い，判定結果に対して値の加工を行っている．

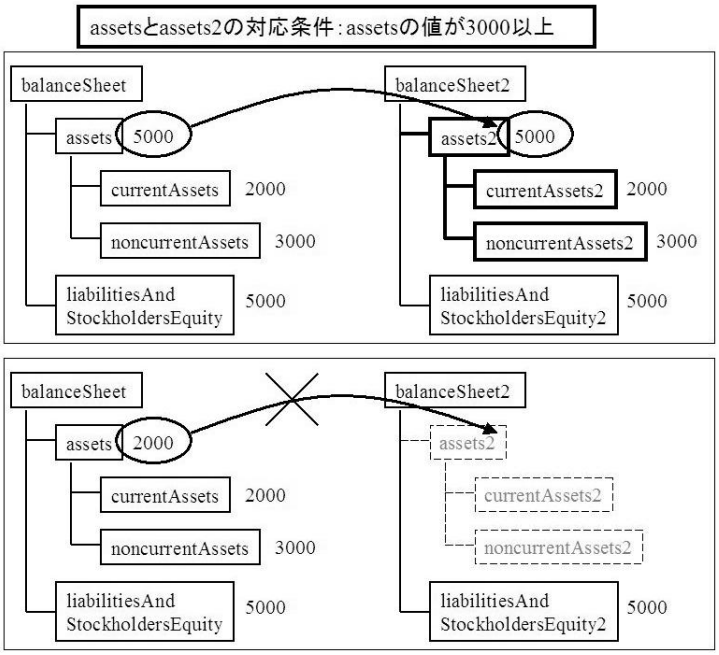

図 3: 条件指定による変換の違い

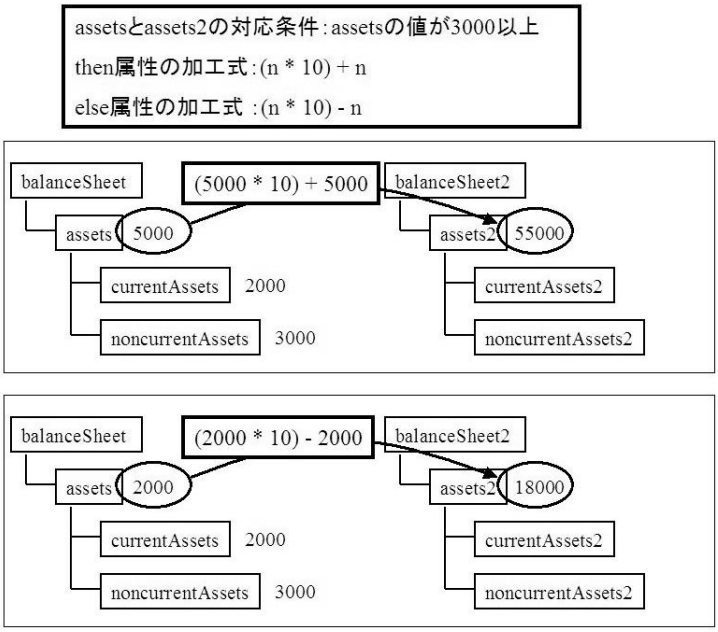

図 4: 値の加工の例

4　XBRL 財務報告書自動変換ツール

本研究では，XBRL 文書のデータ項目間の対応関係を利用して既存の財務報告書から別の財務報告書への自動変換を行うツールの試作を行った．本節では，試作した財務報告書自動変換ツールの説明を行う．

4.1　ツールの概要

本ツールの実装には，W3C が勧告している XML 文書処理のための API である DOM から言語バインディングが提供されていることもあり，Java を選択した．XML パーサとしては Apache[4] の Xerces を利用している．

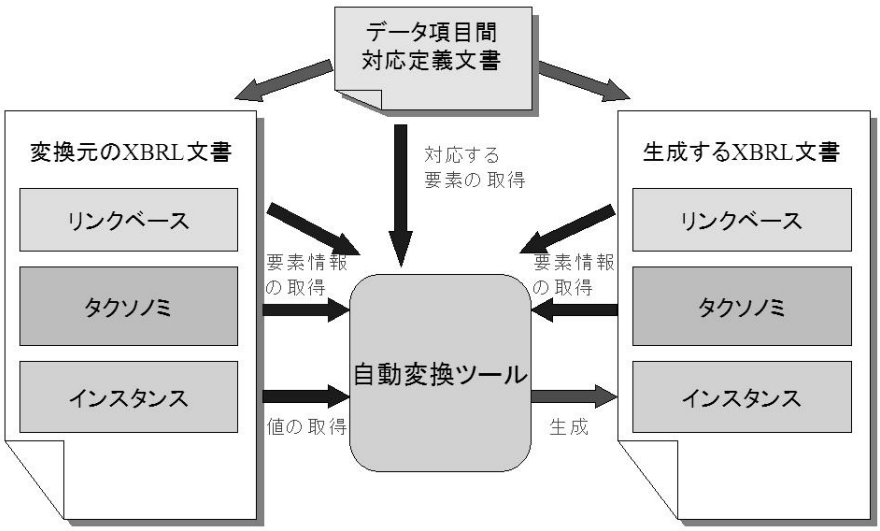

図 5: ツールの構成

本ツールの構成を図 5 に示す．本ツールの処理には変換元と変換先のタクソノミ文書，変換元のインスタンス文書，対応定義文書が必要である．タクソノミのパスはインスタンス文書から，リンクベースのパスはタクソノミから得られるので，パスの入力が必要なのは変換元のインスタンス文書，変換先のタクソノミ，変換先のインスタンス文書 (出力先)，対応定義文書の 4 つである．

ファイルパス入力用インターフェースを図 6 に示す．テキストフィールドに

1. 変換元のインスタンス文書
2. 変換先のタクソノミ
3. 変換先のインスタンス文書 (出力先)

図 6: ファイルパス入力画面

4. 対応定義文書

の 4 つのファイルを順に入力する．入力後は自動的に変換処理を行い，インスタンス文書を生成して出力する．

4.2 ツールの動作

ツール内部での処理の流れは以下のとおりである．

1. 入力されたファイルからインスタンス文書で記述する要素に関する情報を取得し登録する
2. 対応する要素の取得に利用するための変換元の XBRL 文書の木を生成する
3. データ項目間の対応関係の情報を取得し登録する
4. 以下の要素の生成処理を再帰的に繰り返して木を生成する

   4-1．対応する要素の値を対応関係を利用して取得する
   4-2．条件の判定して必要ならば値を加工する
   4-3．子要素を追加して必要ならば値の計算処理を行う
   4-4．要素に追加する
   4-5．属性の処理を行う

5. 生成された木を出力する

以下，各処理について説明する．

4.2.1 タクソノミ，リンクベースから得られる情報の取得

インスタンス文書の変換を行うためには，まず変換元の XBRL 文書と作成する XBRL 文書の情報をタクソノミ文書から取得する必要がある．以下はどちらの XBRL 文書にも共通の処理である．

- 名前空間接頭辞の取得
  タクソノミと定義リンク，計算リンクの名前空間に関する情報を取得する．取得された名前空間接頭辞は要素や属性の値を取得する際に必要となる．

- タクソノミからの情報の取得
  タクソノミで定義されている要素の情報を取得する．

- 定義リンクからの情報の取得
  定義リンクから各データ項目の関係情報を取得する．すべての定義アーク要素を調べ，アークの上書きがあればその情報によって上書きする．同じ要素間で priority 属性の値が等しいアークが複数存在する場合には，先に取得したアークの情報を優先する．

作成する XBRL 文書の情報の取得では，計算リンクの情報の取得も行う必要がある．

- 計算リンクからの重み情報の取得
  計算リンクから各データ項目の重みを取得する．すべての計算アーク要素を調べ，アークの上書きがあれば，その情報によって上書きする．同じ要素間で priority 属性の値が等しいアークが複数存在する場合には，先に取得したアークの情報を優先する．最後に定義リンクで取得したアークに計算リンクから取得した重みの情報を追加する．

4.2.2 変換元の XBRL 文書の木の構築

取得した情報を元に，変換元の XBRL 文書の木を構築する．この木は値の取得のために変換元のインスタンス文書の要素を探すときに，要素間の関係を調べるのに利用される．

木の構築を行うにはまずルート要素を特定しなければならない．ここでいうルート要素とは，タクソノミやインスタンス文書全体のルート要素のことではなく，タクソノミで定義された要素で構成される木のルート要素のことである．定義リンクでは要素間の親子関係が定義されているが，ルート要素は親要素を持たないため，親を持たない要素をすべてルート要素として取得する．

定義リンクから得た情報で要素間の親子関係がわかるので，子の要素の生成と追加をルート要素からの再帰処理により木を構築する．

4.2.3 データ項目間の対応付け情報の取得

まず名前空間情報の取得を行い，次に要素間の対応情報を取得する．項目間の対応に指定する条件があればその情報も取得する．

4.2.4 変換先のインスタンス文書の木の構築

取得した情報を元に，変換先のインスタンス文書の木を構築する．処理は木のルート要素から開始する．以下，各処理について説明する．

1. 値の取得

   インスタンス文書に記述する要素について，対応する要素の値を変換元のインスタンス文書から取得する．要素の値を取得するには，対応する要素の特定，要素の値の取得，条件付き対応の場合の条件の判定，値の加工が必要である．それぞれの処理について，以下で説明する．

   - 対応する要素の特定と値の取得

     処理する要素の名前を元に対応付け情報を調べ，その要素と対応付けられた変換元のインスタンス文書の要素の名前を取得する．名前がわかったら変換元のインスタンス文書から要素を探して値を取得する．

   - 条件付き対応の場合の条件の判定

     アーク要素で条件が設定されていれば conditionStatement 要素の情報から条件式を取得し，判定を行う．

   - 判定結果に基づく値の加工処理の決定

     条件式の判定結果により値の加工を行う．条件式の結果が真であれば then 属性の式に従い値を加工する．then 属性が省略されている場合には加工は行わない．また，条件式の結果が偽であれば else 属性の式に従い値を加工する．else 属性が省略されている場合には，対応が不成立となり要素は生成されずに次の要素の処理に移る．

   - 値の加工

     conditinalStatement 要素の情報から加工する値の型と加工の式を取得して加工する．

2. 子要素の追加

   子要素は定義リンクの情報から得られるので，すべての子要素について処理を行う．また，要素の値を子要素の値から計算しなければならない場合には，子要素を追加す

る際に子要素が持つ値と重みを乗算する．この値をすべての子要素について計算して和をとり，要素の値とする．子要素が概念等価な要素である場合には別の処理が必要となる．

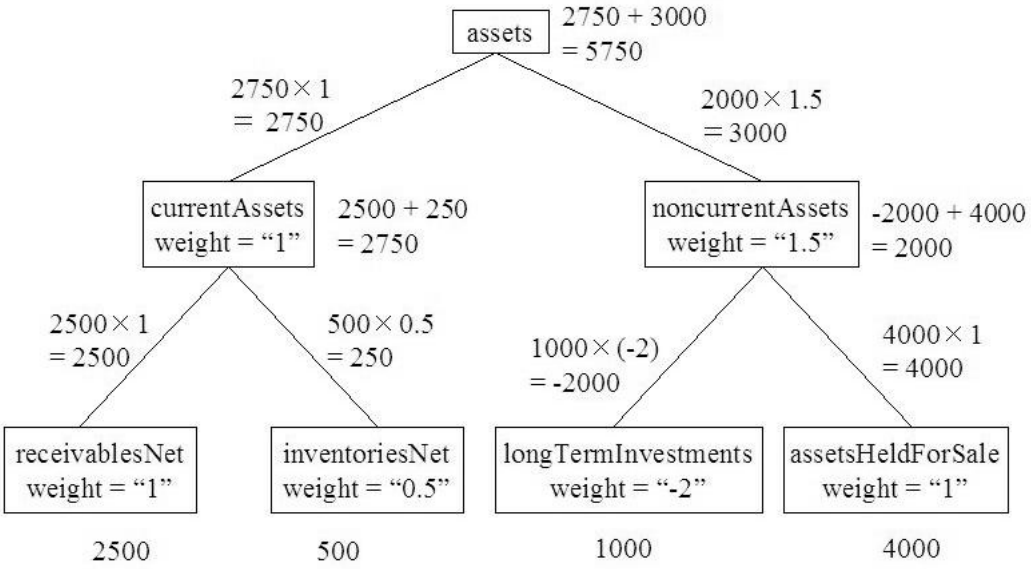

図 7: 計算処理の例

- 子要素の値を用いた親要素の値の計算

  図 7 は子要素の値を用いた親要素の値の計算の例である．assets，currentAssets，noncurrentAssets の値はそれぞれの子要素の値に重みをかけたものの和になる．

- 子要素の追加生成

  図 8 は子要素の生成処理において対応する要素が変換元のインスタンス文書に複数記述されている場合の処理の例である．balanceSheet2 要素の子要素である assets2 要素の生成処理において，対応する assets 要素が 3 つ記述されている．しかし，これら 3 つの assets 要素は会計年度などが異なっており，それぞれ別々の情報を表している．

  このような場合には，変換先のインスタンス文書でも 3 つの要素を記述する．変換元のインスタンス文書で記述されている数 (この場合は 3 つ) だけ子要素を追加生成し，それぞれ別々に値を取得する．この例の場合は 2 つの子要素を追加生成して処理することになる．そして処理が終了した子要素を追加する．

- 概念等価な要素の処理

  概念等価とは，ある財務事実を別の観点から見た要素間の関係が概念的に等

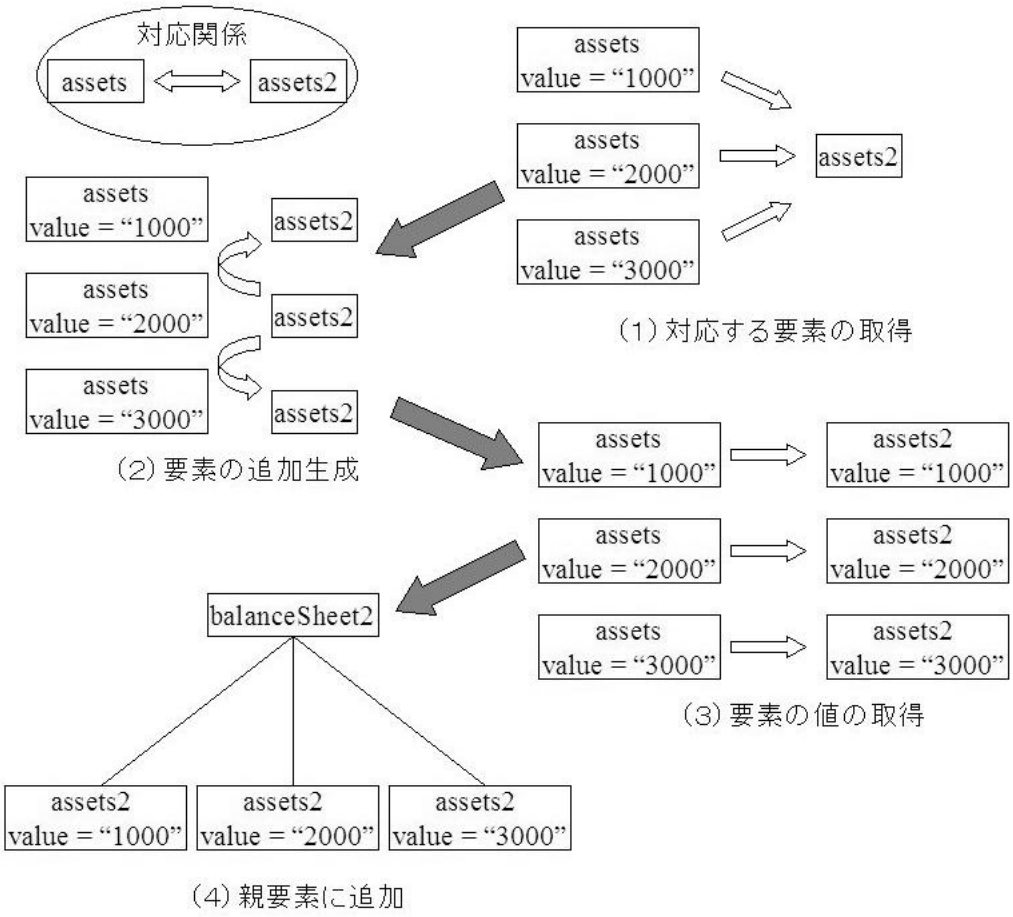

図 8: 子要素の追加生成の例

図 9: 概念等価の例

価であるというものである．図 9 に概念等価な要素を含む XBRL 文書の要素間の関係の例を示す．この例では cash(現預金) 要素の異なる観点として cashByBranchLocation(支店所在地別現預金) 要素，cashByAccountType(口座型別預金) 要素，cashByAvailability(利用可能性別現預金) 要素が存在している．

概念等価な要素はそれぞれの値が親要素の異なる観点を表しているので，子要素の値を合計すると値が多重に計算されることになる．よって親に値を渡す要素は等価な要素のうちの 1 つのみでなければならない．そのため，通常の処理とは別にこれらの要素の処理を行う必要がある．

等価な要素の間には値を与える側と受け取る側が存在する．図 9 では要素 CashByBranchLocation が与える側，CashByAccountType と CashByAvailability が受け取る側であり，CashByBranchLocation の値が他の概念等価な要素に与えられる．このため，他の概念等価な要素に値を与えている要素の値だけを親要素の値の計算に使用し，それ以外の要素は計算には使用しないようにすることで多重計算を防ぐことができる．図 9 の場合は CashByBranchLocation の値だけを Cash の値の計算に使用し，CashByAccountType，CashByAvailability の値は計算には使用しない．

3. 要素の値の追加と属性処理

値の取得，あるいは子要素の値からの計算によって得られた値を要素に追加する．

また，属性の値は変換先のタクソノミでの要素定義でデフォルト値が設定されている．この属性値に付いても対応付けを定義すれば値の変更が可能である．属性値の取得に関しては要素とほぼ同じ処理で行うことができる．

4.3 ツールの実行例

ツールの実行例を示す．図 10 のインスタンスから図 11 の対応関係を元に値を取得して，図 12 のインスタンスを生成した．図 10 と図 12 を表形式にしたものが表 3 と表 4 である．対応関係に条件指定のない要素には変換元の文書の値がそのまま記述され，noncurrentAssets の条件判定および値の加工処理，assets，liabilitiesAndStockholdersEquity，StockholdersEquity の子要素からの値の計算処理が正しく行われていることがわかる．

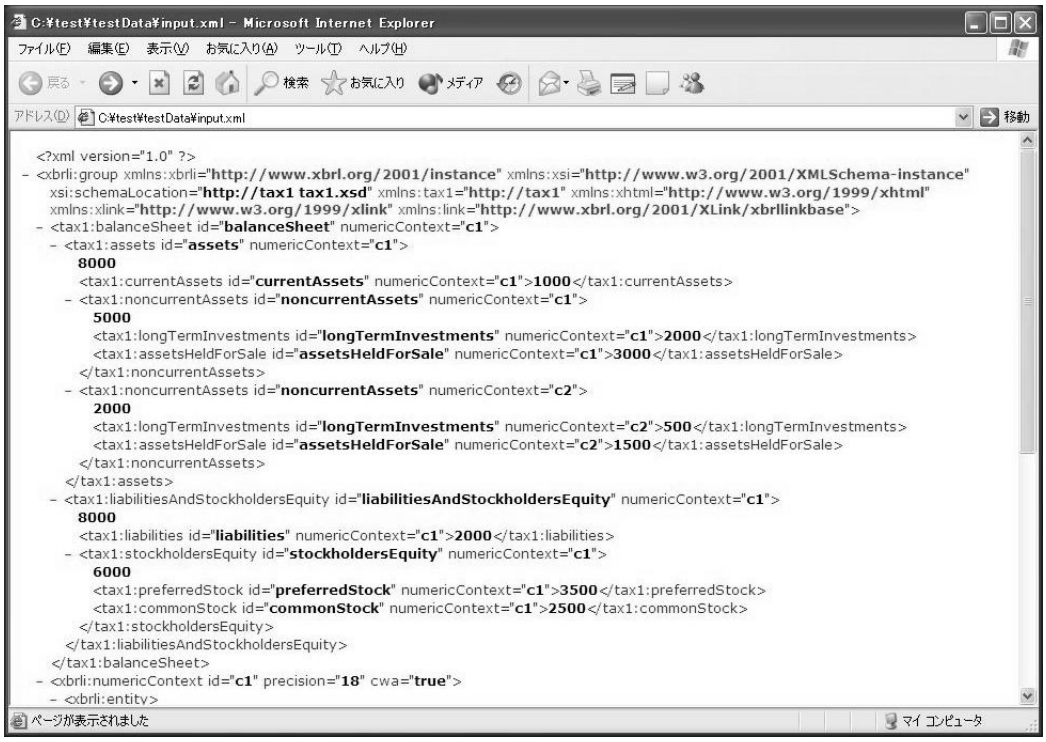

図 10: 変換元のインスタンス文書

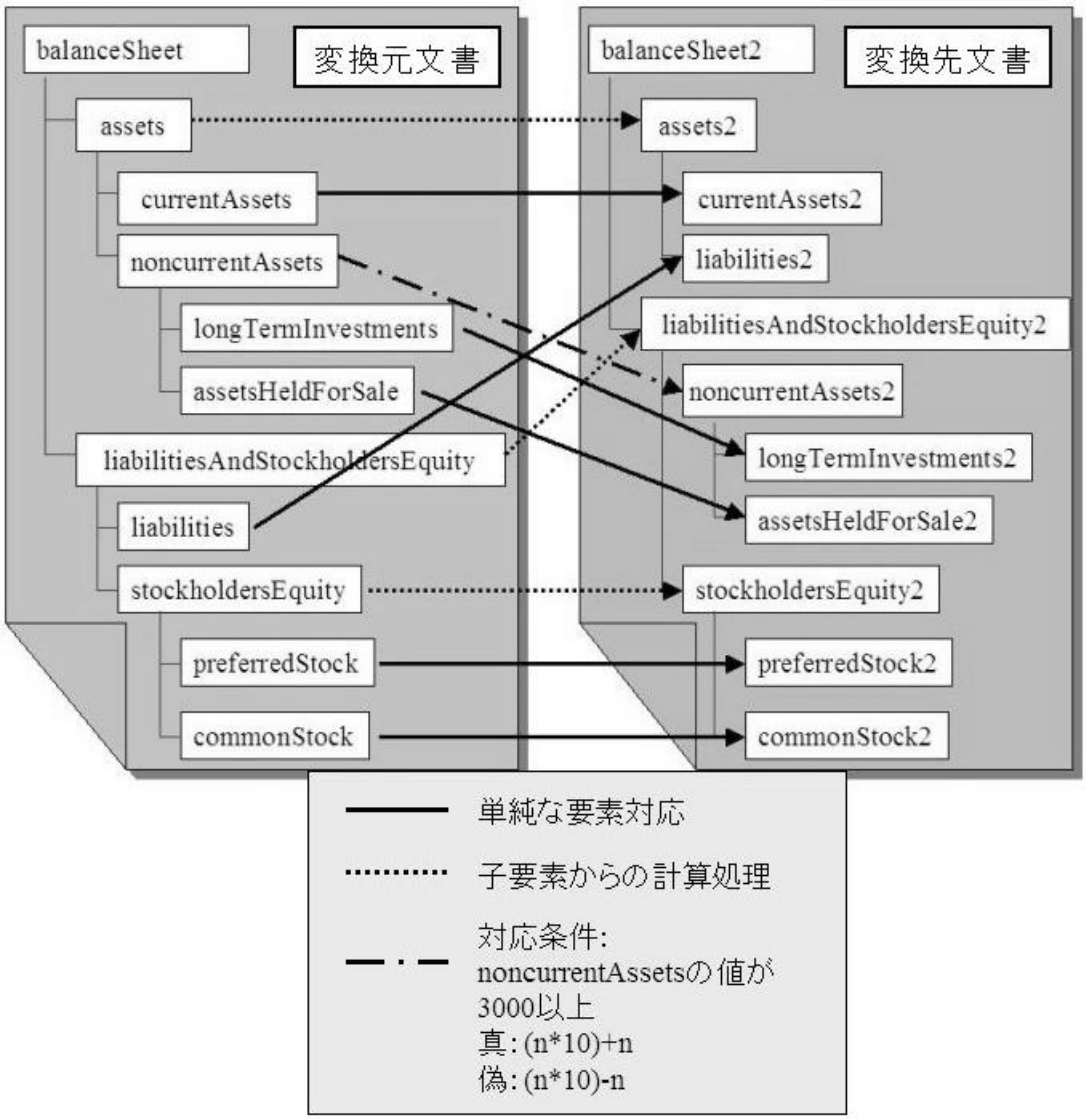

図 11: データ項目間の対応

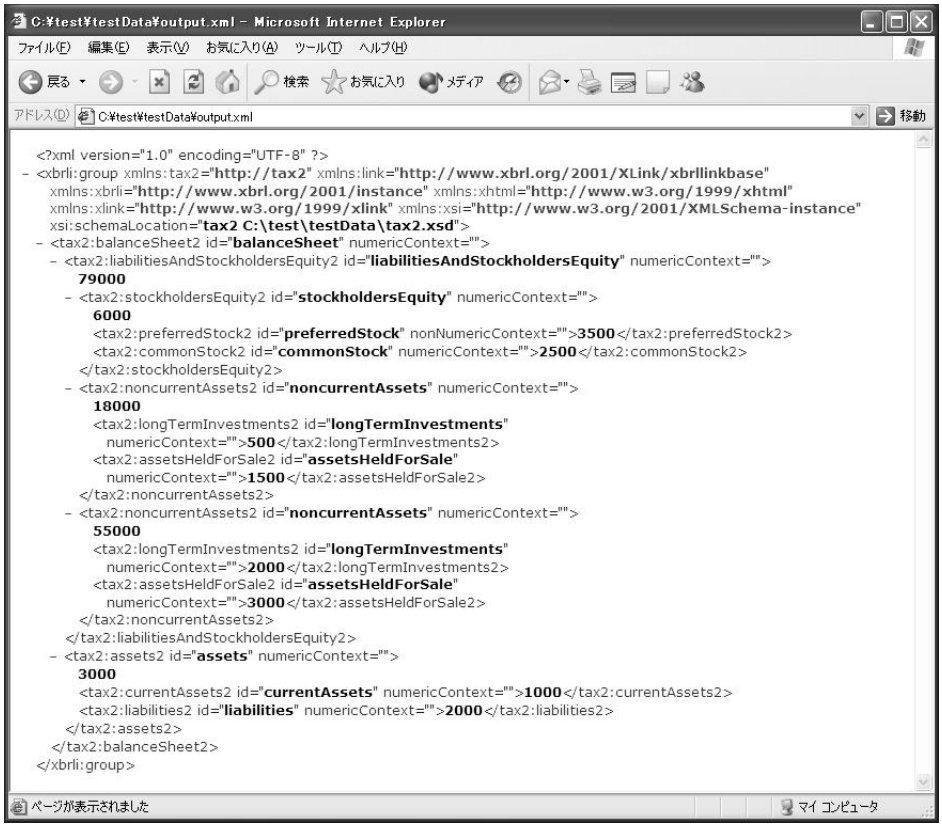
C:¥test¥testData¥output.xml - Microsoft Internet Explorer
ファイル(F) 編集(E) 表示(V) お気に入り(A) ツール(T) ヘルプ(H)
戻る 検索 お気に入り メディア
アドレス(D) C:¥test¥testData¥output.xml
移動
<?xml version="1.0" encoding="UTF-8" ?>
<xbrli:group xmlns:tax2="http://tax2" xmlns:link="http://www.xbrl.org/2001/XLink/xbrllinkbase"
xmlns:xbrli="http://www.xbrl.org/2001/instance" xmlns:xhtml="http://www.w3.org/1999/xhtml"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="tax2 C:\test\testData\tax2.xsd">
<tax2:balanceSheet2 id="balanceSheet" numericContext="">
<tax2:liabilitiesAndStockholdersEquity2 id="liabilitiesAndStockholdersEquity" numericContext="">
79000
<tax2:stockholdersEquity2 id="stockholdersEquity" numericContext="">
6000
<tax2:preferredStock2 id="preferredStock" nonNumericContext="">3500</tax2:preferredStock2>
<tax2:commonStock2 id="commonStock" numericContext="">2500</tax2:commonStock2>
</tax2:stockholdersEquity2>
<tax2:noncurrentAssets2 id="noncurrentAssets" numericContext="">
18000
<tax2:longTermInvestments2 id="longTermInvestments"
numericContext="">500</tax2:longTermInvestments2>
<tax2:assetsHeldForSale2 id="assetsHeldForSale"
numericContext="">1500</tax2:assetsHeldForSale2>
</tax2:noncurrentAssets2>
<tax2:noncurrentAssets2 id="noncurrentAssets" numericContext="">
55000
<tax2:longTermInvestments2 id="longTermInvestments"
numericContext="">2000</tax2:longTermInvestments2>
<tax2:assetsHeldForSale2 id="assetsHeldForSale"
numericContext="">3000</tax2:assetsHeldForSale2>
</tax2:noncurrentAssets2>
</tax2:liabilitiesAndStockholdersEquity2>
<tax2:assets2 id="assets" numericContext="">
3000
<tax2:currentAssets2 id="currentAssets" numericContext="">1000</tax2:currentAssets2>
<tax2:liabilities2 id="liabilities" numericContext="">2000</tax2:liabilities2>
</tax2:assets2>
</tax2:balanceSheet2>
</xbrli:group>
ページが表示されました
マイ コンピュータ

図 12: 出力結果

表 3: 変換元文書

balanceSheet	
assets	8000
currentAssets	1000
noncurrentAssets	5000
longTermInvestments	2000
assetsHeldForSale	3000
noncurrentAssets	2000
longTermInvestments	500
assetsHeldForSale	1500
liabilitiesAndStockholdersEquity	8000
liabilities	2000
stockholdersEquity	6000
preferredStock	3500
commonStock	2500

表 4: 変換先文書

balanceSheet2	
assets2	3000
currentAssets2	1000
liabilities2	2000
liabilitiesAndStockholdersEquity2	79000
noncurrentAssets2	55000
longTermInvestments2	2000
assetsHeldForSale2	3000
noncurrentAssets2	18000
longTermInvestments2	500
assetsHeldForSale2	1500
stockholdersEquity2	6000
preferredStock2	3500
commonStock2	2500

5 考察

5.1 ツールについての考察

本ツールによりこれまで手作業で行われる部分の多かった XBRL 文書間の変換が自動的に行えるようになった．項目数 2000 程度の XBRL 文書間の変換は 2 分程度で完了する．手作業で変換する場合に比べて効率が大幅に向上している．また，値の計算などを自動的に行うことにより計算ミスや編集ミスが減り，文書の信頼性を向上させることができた．

しかし，本ツールは変換後のインスタンス文書をファイルに出力するだけで，ユーザに対して変換に関する情報を提示することができない．使いやすいツールにするためには，変換後のデータや修正のための情報の表示などを行うインターフェースの提供が必要である．

5.2 変換手法についての考察

本研究の変換手法には，変換元のインスタンス文書に記述されていないデータ項目で，値の計算や加工でも取得できない項目については変換先のインスタンス文書に記述することはできないという問題点がある．そのような未変換部分の問題の解決手段としては，変換元の XBRL 文書にすべての財務活動のデータを記述してしまい，その文書から必要な XBRL 文

書へと変換する方式が考えられる．あるいは，未変換部分についてのリストを作成し，ユーザに通知するなどの機能をもたせることで修正作業の負担を軽減することができる．

また，本研究の変換手法は項目間の対応関係の定義とタクソノミ文書により定義される文書形式を用いて処理を行うため，それらの定義に含まれないコンテキストについては処理を行えないという問題点もある．コンテキストを記述していなければ同じ名前の要素の識別が困難であるため，識別を可能とするために本研究ではインスタンス文書を木構造で記述することとした．木構造によるグループ化を行うことで，それぞれの要素の関係を保持したまま処理を行うことができる．コンテキストについて必要な情報を文書として用意しておけば自動処理は可能となるが，作成する XBRL 文書のコンテキストの内容は作成時点などにより異なるので変換処理のたびに新たな文書を用意する必要があり，あまり効率はよくないと考えられる．未変換部分の問題と同様，ユーザがコンテキストを追加する時の作業を支援する方法を提供することで修正作業の負担を軽減することができる．

対応定義の記述方法には，対応付けに設定する条件に細かい指定ができないことや，値の加工で可能な処理が簡単なものに限られることなどの改善点がある．ただし，より細かい条件指定やより複雑な計算式の記述ができるようにすることは可能だが，どの程度まで記述できるようにするかが問題となる．関数ライブラリの提供などを行い指定を簡単に行えるようにすることも必要である．

6　まとめ

本研究では，XBRL 文書のデータ項目間の対応関係を利用した財務報告書自動変換ツールの試作を行った．自動変換により文書作成が容易に行えるようになり，信頼性も向上した．書式や XBRL 文書間の対応関係の変更にも対応定義文書内の一部の要素の変更だけで対応できた．

今後の課題を以下に述べる．

ユーザは自動変換後のデータに対して，コンテキストや未変換部分のデータを追加しなければならない．ユーザの修正作業の負担を減らすため，変換後のデータと照らし合わせながら修正箇所を表示できるようにする．

データ項目間の対応の条件指定で正規表現を使用できるようにするなどの改良を行い，より具体的に条件を指定できるようにする．

値の加工でより複雑な処理を行えるようにする．特に文字列の加工は現状では結合のみなので，置換なども行えるようにする．また，複雑な処理を簡単に記述するための関数ライブラリの提供する．

謝辞

本研究の全過程を通して，常に適切なご指導および御助言を賜りました 大阪大学大学院情報科学研究科コンピュータサイエンス専攻 井上 克郎 教授に心より深く感謝致します.

本論文を作成するに当たり，逐次適切なご指導およびご助言を賜りました 大阪大学大学院情報科学研究科コンピュータサイエンス専攻 楠本 真二 助教授に心から感謝致します.

本論文を作成するに当たり，適切なご指導，ご助言を賜りました 大阪大学大学院情報科学研究科コンピュータサイエンス専攻 松下 誠 助手に心から感謝致します.

本論文の作成において，適切なご助言を頂きました　日立製作所　湯浦　克彦　氏に心から感謝致します.

本論文の作成において，適切なご助言を頂きました　日立製作所　鈴木　文音　氏に心から感謝致します.

最後に，その他様々なご指導，御助言等を頂いた 大阪大学大学院情報科学研究科コンピュータサイエンス専攻　井上研究室の皆様に深く感謝致します.

参考文献

[1] atmarkIT Corp., @IT:XML eXpert eXchange, http://www.atmarkit.co.jp/fxml/. http://www.hitachi.co.jp/Div/bisd/.

[2] ChuoAoyama Audit Corp., e-Business TrendWatch XBRL 最新動向, http://www.chuoaoyama.or.jp/ebusiness/trend/020204_0101.html.

[3] Sun Microsystems, Inc., The Source for Java(TM), http://java.sun.com/.

[4] The Apache Software Foundation., xml.apache.org, http://xml.apache.org/.

[5] XBRL Japan マーケット・アンド・コミュニケーション (”マーコム ”) 委員会, “ XBRL FACT BOOK ”, XBRL Japan, 2002.

[6] XBRL International, XBRL, http://www.xbrl.org/.

[7] チェルシー・ヴァレンタイン, ルシンダ・ダイクス, エド・ティテル, “ XML スキーマ詳解 ”, コンピュータ・エージ社, 2002.

[8] 三分一信行, “ XBRL Specification 2.0 の概要 ”, 2001.

[9] 中山幹敏, 奥井康弘, “ 改訂版 標準 XML 完全解説 (上) ”, 技術評論社, 2001.

[10] 中山幹敏, 奥井康弘, “ 改訂版 標準 XML 完全解説 (下) ”, 技術評論社, 2001.

[11] 湯浦克彦, 竹内成明, 佐々木達也, “ xml.trend XBRL ”, 2002.

[12] 横井与次郎, “ Java/XML プログラミング入門 ” ソフト・リサーチ・センター, 2001.

[13] ルーサー・ハムプトン, デヴィッド・ヴァンカノン, “ 拡張可能なビジネス報告言語 (XBRL) 仕様書 ”, 2001.