卒業論文

opmlファイルをmwファイルに変換するパーサの開発

関西学院大学 理工学部 情報科学科
5611　森永 祥

2010年3月

指導教員　西谷 滋人 教授

概 要

Mapleは数式処理，数値計算を行うソフトウェアである．グラフ作成に強みをもつため教育現場で利用されるだけでなく，制御設計や回路設計，光学設計などのあらゆる分野の研究に用いられている．しかし，日本語で書かれたMapleの教科書はほとんど存在せず，外国語の教科書を直訳したものしか見当たらない．また，日本と諸外国では数学の学習項目やその順序が大きく異なるため，外国の教科書に沿って学習することは非常に困難である．そういった背景の中，西谷滋人教授がMapleの演習書を執筆することになった．原稿を作成する際，エディタとしてTAOというソフトウェアを用いる．TAOは階層構造の作成，管理に優れており，必要な箇所だけ見やすく文書作成できる利点がある．しかし，計算の動作確認はMaple上でしか行えないため，TAOファイルをMapleのファイル形式であるmwファイルに変換する必要がある．そこで本研究では，演習書の執筆を円滑に行えるようにするため，TAOで書かれた原稿をmwファイルに変換するパーサの構築を行った．

システムの構築にあたり，開発言語としてRubyを用いた．Rubyは文字列のマッチング機能に優れているため，ファイル形式の変換には非常に有用であるからである．またパーサの基本構造として，Pukiwiki記法をHTMLに変換するパーサであるpukipa.rbを参考にした．その上で，数式を扱うために必要な機能を実装し，不要な箇所を取り除いてメソッド間の構造を簡略化した．

これに必要な機能として，数式処理コマンドの追加，動作確認の必要がない箇所の非表示機能，使用用途に応じた出力方法の切り替えなどが挙げられる．数式処理コマンドの追加は，文字列のマッチングによって通常テキストか数式テキストかを判断する規約を作成することにより実現した．非表示機能は，表示を望まない箇所の階層を閉じることにより，出力しない仕様にした．出力方法は，コマンドライン上での入力によって切り替えが出来るようにした．

目 次

第1章　序論

1.1　研究の背景

1.1.1　Maple

Maple は 1980 年代前半にカナダのウォータールー大学で開発され，日本のサイバネットシステム株式会社が販売，翻訳を行っている数式処理，数値計算ソフトウェアである．[1] 本来は紙と鉛筆を用いて行う必要がある計算や作図を，Maple では初歩的なプログラムを用いることによって簡単に解答を出力することができる．また，Maple は手計算では困難な問題を確実に解くことを可能とするだけでなく，紙の上に描くことが不可能である 3 次元データを持つグラフを作成することができ，あらゆる角度からそれを見ることができる．利用者層には学生，小学校や中学校，高等学校の数学教員，大学や企業などの研究者など幅広く普及している．

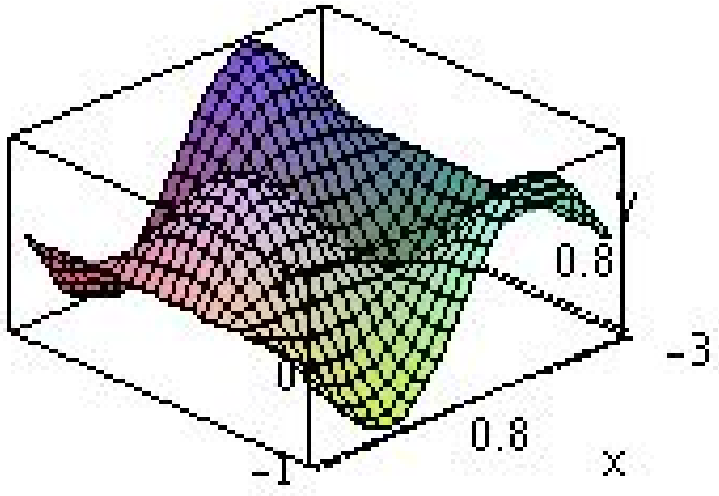

図 1.1: Maple の 3D プロット

Mapleはグラフ描画機能において特に優れており，また必要とするメモリとハードディスク要領が少なく済むといった利点がある．その一方，膨大な計算量を扱うことには向いておらず，計算速度は他の類似ソフトに劣りがちである．また，一部のMapleテキスト用コマンドを除き，基本的にヘルプの内容をコピー＆ペーストすることが出来ないために手作業での文書作成を強いられることが多いだけでなく，図1.2のように日本語に和訳されていない項目が存在するため非常に使い辛い．それに加え，第三者によって手掛けられた日本語のMaple解説書の不足が深刻な問題となっており，初学者の置かれた状況は非常に悪いものとなっている．

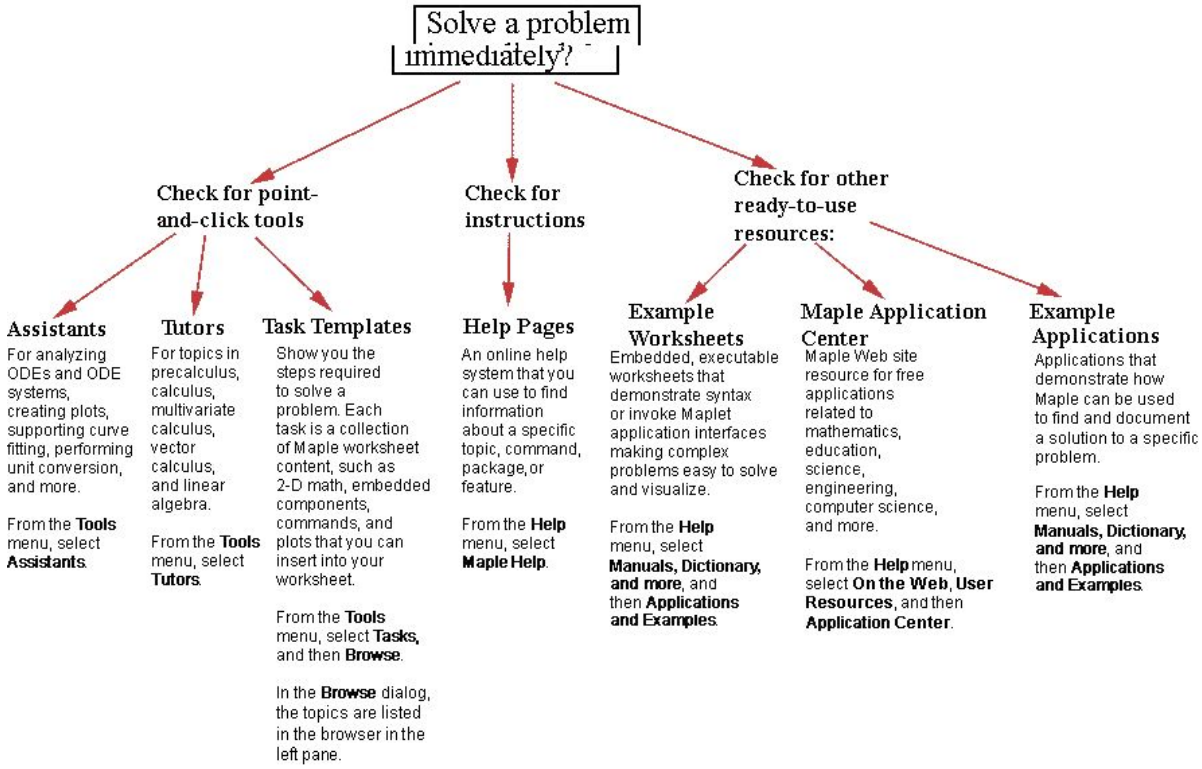

図 1.2: Mapleヘルプ「新規ユーザのためのロードマップ

1.1.2 Maple演習書の執筆

Mapleの解説書が不足しているという背景がある中，このたび関西学院大学理工学部情報科学科の西谷滋人教授がMaple演習書を執筆することになった．その上で，Mapleの入力及び出力結果のフォーマットは演習書として出版するには見栄えが悪いため，LATEX形式で原稿を執筆することとなった．LaTeXとはレスリー・ランポートが開発した組版処理システムであり，その高品質な組版処理能

力によって出版物の作成も可能としている．また，LaTeX には数式専用のコマンドが用意されているため数式組版の質も高く，Maple 演習書の原稿として適していると考えた．

しかしながら，LaTeX で原稿を作成するためには多くのコマンドを入力する必要があり，非常に手間がかかるため，他のアウトラインエディタを用いて最初の原稿を執筆し，それを LaTeX 形式に変換して最終稿とすることにした．また，Maple の演習書を執筆するためには，実際に Maple 上で原稿の数式が正しく動作するかを確認する作業が必要となり，その都度 Maple を開いて数式を入力し実行する，という作業を取っていては大幅な時間と手間がかかってしまう．Maple のテキストを編集するためのソフトウェアも存在しないため Maple に関する本を執筆するための環境は非常に悪いと言える．そのため，西谷研究室の 3 人が役割を分担して原稿を加工し，演習書の執筆を援助することを試みた．

1.2 研究の目的

原稿の執筆には，アウトラインエディタ TAO を用いて作成される．TAO は階層構造の文章作成を円滑に行えるというメリットがあるため，初期の原稿作成においては Maple よりも適していると考えられるためである．TAO は項目の作成やそれらの並べ替え，階層の作成等をサポートしているが，原稿にかかれた数式に誤りが無いかを確認するには，一度その数式を Maple に入力し，計算させてみる必要がある．

そこで本研究では，TAO で書かれた原稿を Maple のファイル形式である mw ファイルに変換することにより，数式処理，数値計算の実行を可能とするパーサの開発を目的とする．

第2章　開発環境

2.1　ruby

2.1.1　ruby とは

本研究には，まつもとゆきひろ氏によって開発されたオブジェクト指向言語，ruby を用いた．ruby は 1993 年に考案され，1995 年にネットニュースのニュースグループである fj で発表された言語で，クラス定義や正規表現処理に優れている等の長所がある．正規表現とは文字列の集合をある一つの文字列で表現する方法であり，テキストエディタにおけるパターンマッチを検索する際などに使用される．本研究ではファイル形式の変換を目的としているため，パターンマッチ文字列を表す正規表現処理は非常に有用であるため，ruby が本研究に適していると考えた．

2.1.2　pukipa.rb

本研究でパーサを作成する際，MIT ライセンスのソフトウェアである pukipa.rb を参考にした．pukipa.rb とは，PHP 言語で実装された Wiki の一種である PukiWiki 文法を HTML に変換するシステムである．pukipa.rb のスタティックコールグラフは次の通り．[2]

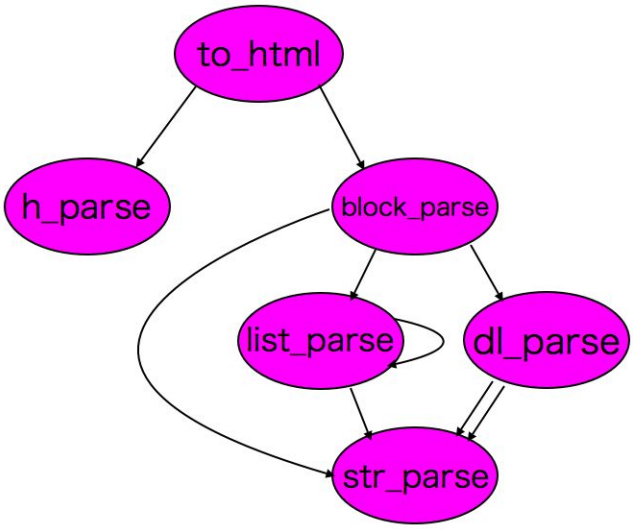

図 2.1: pukipa.rb のスタティックコールグラフ

関数 to_html がプログラムを制御しており，Pukiwiki 記法において文頭にアスタリスク「*」が付いている行の変換を扱う際に関数 h_parse を呼び出し，タグで囲まれた行の変換を行う際に block_parse を呼び出す仕組みになっている．

本研究ではタグによって記述される xml 表記のファイルを扱うため，主に pukipa.rb の block_parse を参考にプログラムを作成した．

第3章　本論

3.1　手法

3.1.1　ファイル変換方法

TAO は TAO 形式という独自のファイル形式を取るため，このファイルそのものの形式を mw ファイルに変換することが困難である．そこで，TAO の機能を用いてファイルを一度 opml ファイルとして書き出し，その後パーサにかけることによって mw ファイルに変換するようにした．

表 3.1: opml と mw の相違点

	opml	mw
本文エリア	<body>..</body>	<Worksheet>..</Worksheet>
1行の文章	<outline../>	<Text-field>..</Text-field>
複数行をグループにまとめる	<outline>..</outline>	<Group> <Text-field>..</Text-field> ... </Group>
Maple上で数式として扱う	×	<Text-field prompt=">" style="Maple Input">
階層構造の生成	<expansionState>(行番号) </expansionState>	<Section collapsed="false">.. </Section>

これは opml と mw の主な相違点を示した対照表である．opml と mw はどちらも xml(Extensible Markup Language) の一種であるため，タグにおける文法上の類似点が多く存在する．このタグの中身を書き換えることにより，ファイル形式の変換を行っていく．なお，xml 形式同士の変換パーサとして，REXML というシステムが存在するが，[3] このシステムは mw ファイルの数式処理，数値計算コマンドに対応していないため，mw ファイルへの変換後に Maple で数値計算を行うことに特化した新しいシステムを開発した．

3.2 研究の説明

3.2.1 expansion

演習書作成の原稿となる TAO ファイルは極めて冗長なものとなる．そのため，TAO の機能として原稿を書く際は作業中の箇所以外の階層を閉じておくことによりファイルを見やすくすることが出来る．しかし，TAO ファイルを opml に書き出すと，階層を閉じて非表示にしておいた階層もすべて記述されてしまっているため，mw ファイルに変換後 Maple 上で開くと，莫大なファイルの内容が全て表示されてしまう．それでは原稿の動作確認が困難になってしまうため，mw ファイル変換時には必要な箇所のみを表示し，見やすくすることが望ましい．そのため，動作の確認が不要な箇所の階層をあらかじめ閉じておくことによって，mw ファイルに出力しない機能を実装した．その要となっている関数がこの expansion である．

opml ファイルは始めに

```
<?xml version="1.0" encoding="UTF-8"?>
<opml version="1.0">
<head>
<title>test1.tao1</title>
<expansionState>0,2,6,7,10,13,15,22</expansionState>
</head>
```

といった風に記述される．<expansionState></expansionState>で囲まれている数字が，階層のユニットの内，開いている部分の行番号を表している．本システムではまず関数 expansion にて，階層が開いているユニットの行番号を配列@expansion に格納している．

3.2.2 block_parse

関数 `block_parse` では，`opml` 形式のタグを `mw` 形式に合ったタグに書き換えており，本文エリア以外の変換作業は全てこの関数内で行っている．また，`</outline>` は本文エリアにおける現在開いている階層の最深部の末尾を表している `opml` 形式の終了タグであり，この行にマッチした場合は `mw` 形式の終了タグである`</Group>` に書き換える他，階層の深度を表す変数`@status` の数値を一つ下げ，次の行から一つ浅い階層に出る指示を与える．

3.2.3 outline

opml ファイルは<head></head>で囲まれた初期定義の次に，<body></body>で囲まれた本文エリアが書かれる．opml 形式では本文エリア内のタグは全て<outline...>

となっており，その中身一つ一つがテキストとなっている．関数 outline では本文エリア内の構造を調べ，mw 形式に合ったタグ付けをしている．

例えば，今

```
<outline text="等号" _created="2009-09-19 05:45:52 +0900"
_modified="2009-12-02 15:36:10 +0900">
```

と書かれた行がある．このように，行末の「>」の前にスラッシュ「/」が入っていない場合，階層的にこの行を親ユニットとする子ユニットが次に続くことを表している．この場合，階層の深度を表す変数@status の数値を一つ上げ，次の行から一つ深い階層に入る指示を与える．

次に，

```
<outline text="とする． " _created="2009-12-02 15:41:52 +0900"
_modified="2009-12-02 15:41:52 +0900"/>
```

といったように，文末の「<」の前にスラッシュ「/」が入っている場合を見る．これは現時点での階層がこの 1 行のみで終わっている尚かつこれより下位の階層を持たないことを表すため，@status の値の変更はせずテキストのタグ変換のみを行う．

最後に，本研究の要である数式処理を扱う部位を紹介する．まず，Maple に変換した際に通常テキストとして表示するか，あるいは Maple の数式処理コマンドを付けることによって計算可能なテキストとして表示するかを区別する必要がある．そのため，TAO で原稿を書く際，数式処理を実行したい行には，行頭に「>」を付けることによってその行を計算可能な行として出力させるという規約を用いた．実際に行頭に「>」を付けて opml ファイルに書き出すと

```
<outline text="> a:=3; b:=2; a+b;" _created="2009-12-02
15:41:50 +0900"_modified="2009-12-02 15:41:50 +0900"/>
```

といった風になる．<outline text= ”の後に>という表記があるが，本システムはこれにマッチした場合，変換後の mw ファイルに style=”Maple Input” というコマンドを加えるようにした．これにより，mw ファイルを Maple 上で開いた際に計算可能な行として出力する機能を実現した．

これらを組み合わせることにより，opml ファイルの階層構造を読み取り，mw 形式で同等の階層構造を作り上げることに成功した．

3.2.4 to_mw

関数 to_mw は，元の opml ファイルを 1 行毎に分割し，正規表現のマッチングによってその行のデータを expansion，block_parse，outline のどの関数に渡すかを決定し，全ての行の変換が終了したら変換後のデータを mw ファイルに出力する役割を担っている．また，関数 expansion によって配列@expansion に格納された階層が開いている行の行番号と，現在の行がマッチしているかどうかの判定を行っている．

このプログラムのスタティックコールグラフは次のようになる．

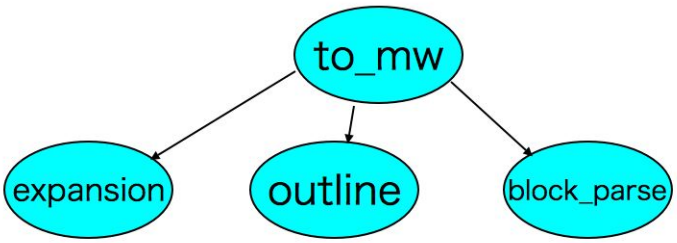

図 3.1: 本システムのスタティックコールグラフ

数式処理に対応するために関数 outline が複雑になった一方，opml 形式と mw 形式の類似点の多さやハイパーリンクが存在しないことから，pukipa.rb と比較してメソッド間の構造が極めて簡易になった．

3.2.5 出力方法の切り替え

TAO から opml ファイルに書き出した際，日本語テキストは文字コードでの表記ではなく，日本語表記のまま出力される．しかしパーサを通して mw 形式に変換する際，日本語表記のままで出力すると Maple 上でうまく開かない．そのため，本システムでは日本語テキストは UTF-8 表記に変換する仕様にした．その際，Ruby には日本語を UTF-8 形式に変換することは出来るが [4]，その構造の中身そのものを出力する方法が基本的に存在しない．唯一存在する出力方法が「p」メソッドを用いる手法であるが，このメソッドは主にデバッグ用に用意されているものであるため，文字列をダブルクォーテーション「”」で囲む形式で出力されるというデメリットや，改行データもその構造そのものを出力されるためファイル上で改行が行われないというデメリットが存在する．そのため，p メソッドを用いて出力した mw ファイルは，Maple がファイルの内容を認識することが出来ない． そこ

で本研究では，次の図で示すようなステップを踏み，p メソッドで出力したファイルを中間ファイルとして書き出し，そのファイルの「”」を取り除く変換や，改行の命令を正しく与えるための変換を再度行うようにした．

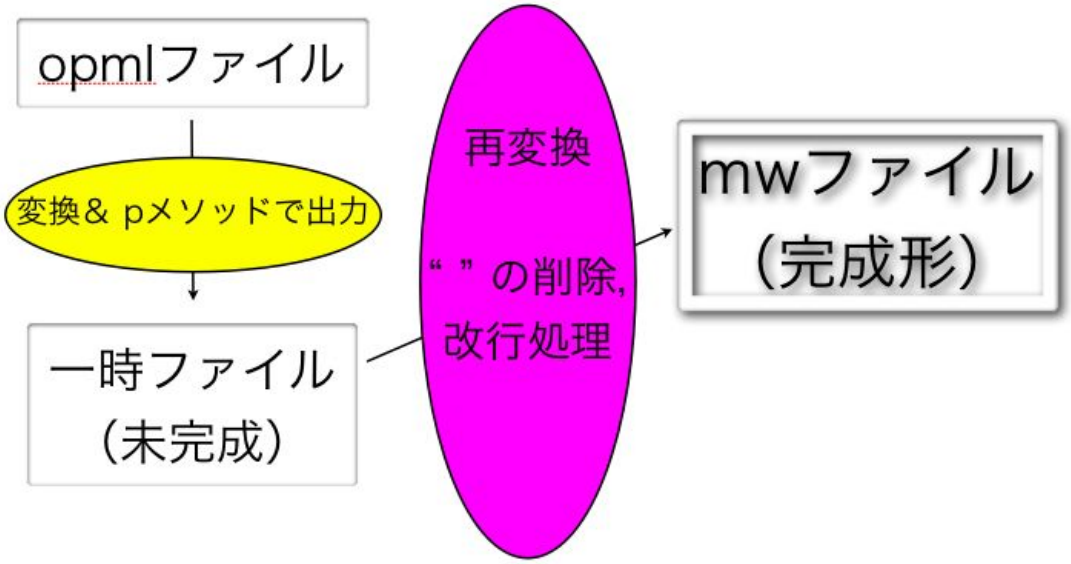

図 3.2: ファイル変換のステップ

これによって日本語テキストが UTF-8 表記で記述された mw ファイルを作成することができ，Maple 上で開くことが可能となった．しかし，このファイルを xml 表記で開いた場合，日本語テキストがどのように書かれているかが非常に分かりづらいため，mw ファイルの中身がどのようになっているかを確認することが極めて困難である．そのため，コマンドライン上での入力によって，日本語テキストを日本語表記のまま出力するか，UTF-8 表記に変換して出力するかを切り替えられるようにした．

これらの機能によって，mw ファイルを Maple 上で開くことを可能とし，かつ xml 表記で開いて構造を調べやすくすることも実現した．

3.2.6 その他の機能

TAO で同じユニット内において改行すると，opml ファイルに書き出した際に改行データとして扱われず，コード表記で出力される．そのため，初期のシステムでは数式を 2 行以上続けて記述したい場合は複数のユニットに分けて記述しなければならず，原稿を作成するにあたり手間がかかってしまった．そこで，数式テ

キストに対し，opml ファイル内をパースした際にこの改行コードを探索し，マッチングを見つけた場合は mw ファイルに変換時に複数行に分ける仕様にした．この機能によって全ての計算式を同ユニット内に記述して数式処理コマンドを付加できるようになった．その際の規約として，TAO での文書作成の際，ユニット内先頭の行頭に「>」をつけ，2 行目以降は行頭に何も付けず，改行のみを行うことによってそれぞれの行数式処理を扱う部位として認識させる仕様にした．

第4章　システム

4.1　TAO

次のように，TAO を用いて原稿のサンプルを作成した．

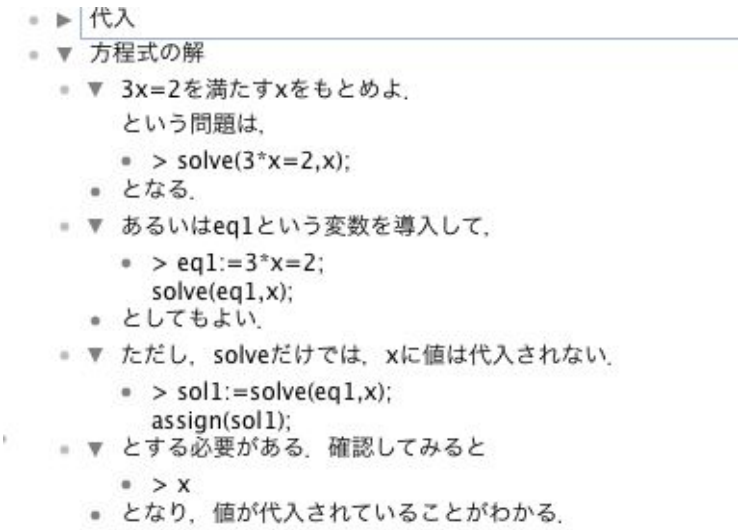

図 4.1: TAO による原稿

「代入」の項目の階層を閉じることにより，以下の階層に含まれるユニットを非表示にする．また，計算を実行したいユニットには，行頭に「>」を付けておく．

4.2 opmlファイル

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <opml version="1.0">
3  <head>
4    <title>final.tao1</title>
5    <expansionState>1,4,5,8,11,13</expansionState>
6  </head>
7  <body>
8   <outline text="代入" _created="2009-09-19 05:46:00 +0900" _modified="2010-02-22 05:57:36 +0900">
9    <outline text="変数に値や式を代入する時には:=(colonequal)を使う．&#10;a=3, b=2のとき，a+bはいくらか？&#10;という問題は，Mapleに教える時には，&#'
10     <outline text="&gt; a:=3; b:=2; a+b;" _created="2009-12-02 15:41:50 +0900" _modified="2010-02-22 05:57:36 +0900"/>
11 </outline>
12   <outline text="とする．" _created="2009-12-02 15:41:52 +0900" _modified="2010-02-22 05:57:36 +0900"/>
13 </outline>
14  <outline text="方程式の解" _created="2009-09-19 05:52:41 +0900" _modified="2010-02-22 05:58:35 +0900">
15   <outline text="3x=2を満たすxをもとめよ．&#10;という問題は，" _created="2009-09-22 15:24:17 +0900" _modified="2010-02-22 05:58:35 +0900">
16    <outline text="&gt; solve(3*x=2,x);" _created="2009-12-02 15:41:32 +0900" _modified="2010-02-22 05:58:35 +0900"/>
17 </outline>
18   <outline text="となる．" _created="2009-12-02 15:41:34 +0900" _modified="2010-02-22 05:58:35 +0900"/>
19   <outline text="あるいはeq1という変数を導入して，" _created="2009-09-22 15:26:17 +0900" _modified="2010-02-22 05:58:35 +0900">
20    <outline text="&gt; eq1:=3*x=2;&#10;solve(eq1,x);" _created="2009-12-02 15:41:27 +0900" _modified="2010-02-22 05:58:35 +0900"/>
21 </outline>
22   <outline text="としてもよい．" _created="2009-12-02 15:41:28 +0900" _modified="2010-02-22 05:58:35 +0900"/>
23   <outline text="ただし，solveだけでは，xに値は代入されない．" _created="2009-09-22 15:26:11 +0900" _modified="2010-02-22 05:58:35 +0900">
24    <outline text="&gt; sol1:=solve(eq1,x);&#10;assign(sol1);" _created="2009-09-22 15:27:22 +0900" _modified="2010-02-22 05:58:35 +0900"/>
25 </outline>
26   <outline text="とする必要がある．確認してみると" _created="2009-09-22 15:28:26 +0900" _modified="2010-02-22 05:58:35 +0900">
27    <outline text="&gt; x" _created="2009-09-23 06:36:46 +0900" _modified="2010-02-22 05:58:35 +0900"/>
28 </outline>
29   <outline text="となり，値が代入されていることがわかる．" _created="2009-09-23 06:36:48 +0900" _modified="2010-02-22 05:58:35 +0900"/>
30 </outline>
31 </body>
32 </opml>
```

図 4.2: opml 形式に書き出したファイル

TAO を opml 形式に書き出すと，本文エリアに作成日時等の余計な情報が表記されていることや，開いている階層の行番号が，opml ファイルの 5 行目に記されていることが確認できる．

4.3 mw ファイルの xml 表記

```
<?xml version="1.0" encoding="UTF-8"?>
<Worksheet>

<Section collapsed="false"><Title>
 <Text-field><Font encoding="UTF-8">方程式の解</Font></Text-field></Title>
<Section collapsed="false"><Title>
  <Text-field><Font encoding="UTF-8">3x=2を満たすxをもとめよ．&#10;という問題は，</Font></Text-field></Title>
<Group>
<Text-field prompt="&gt;" style="Maple Input" linebreak="space">solve(3*x=2,x);</Text-field>
</Group>
</Section>
<Group>
  <Text-field><Font encoding="UTF-8">となる．</Font></Text-field>
</Group>
<Section collapsed="false"><Title>
  <Text-field><Font encoding="UTF-8">あるいはeq1という変数を導入して，</Font></Text-field></Title>
<Group>
<Text-field prompt="&gt;" style="Maple Input" linebreak="space">eq1:=3*x=2;</Text-field>
<Text-field prompt="&gt;" style="Maple Input" linebreak="space">solve(eq1,x);</Text-field>
</Group>
</Section>
<Group>
  <Text-field><Font encoding="UTF-8">としてもよい．</Font></Text-field>
</Group>
<Section collapsed="false"><Title>
  <Text-field><Font encoding="UTF-8">ただし，solveだけでは，xに値は代入されない．</Font></Text-field></Title>
<Group>
<Text-field prompt="&gt;" style="Maple Input" linebreak="space">sol1:=solve(eq1,x);</Text-field>
<Text-field prompt="&gt;" style="Maple Input" linebreak="space">assign(sol1);</Text-field>
</Group>
</Section>
<Section collapsed="false"><Title>
  <Text-field><Font encoding="UTF-8">とする必要がある．確認してみると</Font></Text-field></Title>
<Group>
<Text-field prompt="&gt;" style="Maple Input" linebreak="space">x</Text-field>
</Group>
</Section>
<Group>
  <Text-field><Font encoding="UTF-8">となり，値が代入されていることがわかる．</Font></Text-field>
</Group>
</Section>

</Worksheet>

```

図 4.3: xml 表記で開いた mw ファイル

パーサによって変換したファイルを xml 表記で開くと，この様になっている．TAO で原稿を作成した際に閉じておいた階層が表記されていないことが分かる．

4.4 Maple 上で開いた際の mw ファイル

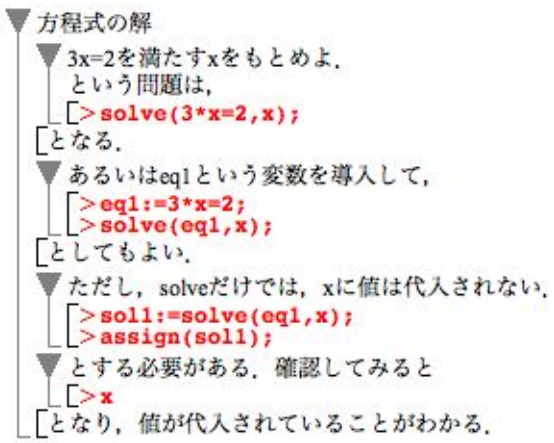

図 4.4: Maple 上で開いた際の mw ファイル

mw ファイルを Maple 上で開くと，この様に TAO で作成した原稿と同等の階層構造が組まれており，なおかつ階層を開いておいた項目のみが記載されている．また，赤字で表記されている部分は計算可能な数式と認識され，Maple 上の「ワークシート全体を実行」のコマンドをクリックすることにより，ファイル上部の項目から順に計算されていく．

第5章　総括

TAO ファイルを mw ファイルに変換するに際し，次のような機能を実装した．
・opml 形式のタグを mw 形式のタグに書き換え
・ユニットの行頭が「>」で書き始められていた場合，そのユニット内のテキストを Maple 上で計算可能なテキストに変換
・パーサの起動時，開いている階層をチェックすることにより，その箇所のみを出力
・Maple 上での動作確認をしたい場合は日本語を UTF-8 文字コード表記で出力し，xml 表記での出力確認をしたい場合は日本語をそのまま出力するための切り替えを行うスイッチ

また，本研究では TAO から Maple への変換に成功したが，Maple から TAO への逆変換を行うパーサがあれば，さらに原稿の作成を容易に行えるようになる．しかし，そのためには Maple で計算した結果をパースする必要があり，特に 3 次元データを持つグラフはデータの構造が複雑でなおかつ膨大なデータ量となる．そのため，冗長な箇所を限りなく削減した本システムを参考に逆変換システムを作成することは，極めて困難であると考えられる．　さらに，作成した原稿を mw 形式に変換するだけでなく LaTeX 形式に変換する機能をつけると，さらに便利なものとなることが考えられる．これらを機に，Maple がより使いやすくなり，より利用者に分かりやすい環境ができれば，更に数学の学習や研究活動が容易になるだろう．

参考文献

[1] 『Maple』, (http://ja.wikipedia.org/wiki/Maple　2010).

[2] 須藤功平　著,『Ruby de XML』, (オーム社　2004).

[3] 青木峰郎　著,『Rubyist Magazine 出張版　正しい Ruby コードの書き方講座』, (株式会社毎日コミュニケーションズ　2007).

[4] るびきち　著,『Ruby 逆引きハンドブック』, (シーアンドエー研究所　2009).

付 録A　パーサの使い方

- tao→Mapleへの変換パーサ
  - このパーサは，opml形式のファイルをMapleのファイルであるmw形式のファイルに変換するためのものである．
    taoで作成したファイルをopml形式で書き出した後にこのパーサを用いることによってMapleでファイルを開くことが可能となり，数式がMaple上で正しく実行されるかを確認することができる．
  - taoファイルの作成に関して
    - 階層構造，日本語文章の作成方法に関しては特に制限はない．ただし，文頭を「>」で開始すると，その行は数式部分として認識されるため注意．つまり
      - >日本語文章
    - という風にしてはダメ．（行間に「>」を入れることは問題ない．行の始めに入れることがダメなだけ．）
    - 数式の作成
      - Maple上で実行可能な数式を作成する際は，文頭に「>」をつけて開始する．
        ２行以上に渡る数式を作成する場合は，改行すれば次の行も数式部分として認識される．
        ただし，日本語文章と混合して数式を作成することは出来ない．
      - 実行可能な例
        - `> solve(3*x=2,x);`
        - ```
          > eq1:=3*x=2;
          solve(eq1,x);
          ```
      - 実行不可能な例
        - 日本語文章と数式が混合している場合．（分離して記述することにより解消）
          - ```
            3x=2を満たすxをもとめよ．
            という問題は，
            > eq1:=3*x=2;
            solve(eq1,x);
            となる．
            ```
          - 上の場合では，「文章と数式」としては認識されず，「グループ全体が文章」として認識される．
        - 2行目以降も「>」から開始している場合．（taoでの数式記述作業負担を軽減するため，改行のみで次の行も数式としてあつかうようにしてある）
          - ```
            > eq1:=3*x=2;
            > solve(eq1,x);
            ```
          - 上の場合ではグループ全体が数式として認識されるものの，２行目がMaple上で正しく動作しない．
  - パーサの使用法
    - taoファイルにおいて，Maple上での動作を確認する必要の無い箇所は，階層を閉じておく．
      例えばこのマニュアルをパーサにかける際，上記の「実行不可能な例」の部分をMaple上で確認する必要がないと感じたときは，この部分の階層を閉じておく．すると，下位下層に記述された部分とともに，Maple上で表示されなくなる．
    - taoで作成したファイルを，opml形式のファイルに書き出す．
      「ファイル」→「書き出し」→名称と保存場所を指定して「保存」
    - "change.rb" "moripo.rb" "mwchange2.rb" の３つのプログラムを必要とするため，変換したいopmlファイルとこれらのプログラムを同一箇所に入れておく．
    - ターミナル上でパーサを実行
      - test1.opmlというファイルを変換してtest1.mwというファイルを作りたい場合は，
        - ruby change.rb test1.opml > test1.mw
      - という風にする．
  - 最新版のプログラムはsubversion管理下にあるため，管理サーバにアクセスし
    /Users/Shared/svn1/Maple3/morisho
    から確認することが出来る．
            ```

謝辞

本研究を遂行するにあたり，終始熱心な御指導を頂いた関西学院大学理工学部情報科学科教授の西谷滋人先生に厚く御礼を申し上げます.

また，それぞれ研究内容は異なりましたが，様々な知識とご協力を頂いた西谷研究室の皆様に，深く感謝の意を表します.