表変換ソフトの開発について

平成 21 年 2 月 12 日

情報電子工学科 4 年
星野 喬

目 次

概要

パソコンやワークステーションで作表をするには専用のソフトを用いることが多い。しかし、そのようなソフトの場合、作表したデータを確認するにはそのソフト上でしか編集や確認することはできない。逆に、LaTeX や HTML ではテキストベースで、ソフトに縛られることなく作表が可能であるが、複雑な構成の表、例えば結合が多用されているような表を作成する場合、データ構造と最終的な表の形との関連がわかりにくくなるため面倒な作業となる。そこで、本研究ではソフトに依存しない保存形式をとり、単純な構成で表を表現できる構造を考える。その構造を Perl のプログラムで実装し、他の形式に変換できるソフトについて検討した。

1　はじめに

現在私達はデータの整理などの多くの場面で作表ソフトを使用している。しかし、それらのソフトでデータの編集をするには専用ソフト上で行わなければいけないことも多い。当研究室では LaTeX と HTML というソフトで作表を行うことができる。LaTeX はテキストベースの文章処理ソフトであり、HTML は Web 上のドキュメントを記述するマークアップ言語である。どちらもテキストベースなので、編集はソフトに縛られることはない。しかし、これらを用いての作表はデータ構造やコマンドを知らないと作表することは難しい。本研究では、テキストベースのデータ構造を提案し、作成したデータを LaTeX や HTML に変換し、表をそれらの形式で利用できる変換ソフトを目標とする。

また、LaTeX から HTML への変換には LaTeX2HTML というソフトがあるが、これは表の変換では、テキスト形式のままの変換ではなく画像形式として出力する形式をとっている。よって、LaTeX と HTML の相互変換も考えた変換方式を考える。

2　LaTeX の作表構造

2.1　LaTeX とは

LaTeX とは、コンピュータ科学者のレスリー・ランポート（Leslie Lamport）によって機能強化された TeX である。TeX とは、コンピュータ科学者であるドナルド・クヌースにより作られた電子組版ソフトウェアである。LaTeX では文書の論理的な構造と視覚的なレイアウトを分けて考えることができることが特徴である。長所として、

- 章・節・図・表・数式などの番号を自動で付け、参照箇所には番号やページを自動挿入でき、目次・索引・引用文献の処理も自動でおこなえる。
- 文章の他に数式や作表のコマンドがあり多用途での使用が可能である。
- テキストベースなのでデータの確認が簡単である。

などが挙げられる。しかし、コマンドが多数ありレイアウトが自由であることは熟練者には扱いやすいが、慣れていない者には難しい作業となるという欠点がある。

2.2　作表構造

LaTeX での作表は、比較的線と面との定義分けができているため、複雑な表の作成もすることができる。しかし、線と面との分離も完全でなく、列の要素に依存している部分も

ある。また、コマンド操作と構造に特徴があり理解していないと作りづらい。
以下に作表の要素を示す。下 Table 1 とその作表コマンドを例として解説をする。

a	b	c
d	e	f
g	h	i

Table 1 LaTeX での作表例

下記は表１の作表コマンドである。

```
\begin{tabular}{|c|c|c|}
\hline
a & b & c\\
\hline
d & e & f\\
\hline
g & h & i\\
\hline
\end{tabular}
```

以降に表１でのコマンドの解説をする。
表作成の宣言は以下のようになっている。

```
\begin{tabular}{|c|c|c|}
表データ
\end{tablar}
```

ここで{|c|c|c|}は表の列要素を示している。c はセル内データの中央寄せ (center) を表す。これは、

```
l：left（左寄せ）
c：center(中央)
r：right(右寄せ)
```

のように配置指定することができる。今回はすべて中央寄せで統一している。 | は縦罫線を示している。縦罫線をなくす場合にはこの記号を無記入の状態にすることで表示されることはなくなる。この例で示した状態は、３列の要素が用意され、セル内データは全て中央寄せとなる。罫線を引かなかったからといってセルがつながるわけではなく、セル内

データの配置指定により列数は決定される。また、列数はここで定義するとこれ以降増やすことはできない。
横罫線や行の定義は表データ部で記述していくことになる。横罫線は、表範囲内の左端から右端までを引く場合

```
\hline
```

を使用する。任意の位置に横罫線を引く場合には

```
\cline{x-y}
```

が使われる。ここでｘとｙは横罫線が引かれる範囲を指定する。指定はセルの単位で行われ、1 列目から 2 列目の部分だけに引く場合は、

```
\cline{1-2}
```

のように記述できる。横罫線も縦罫線と同様に、不要な場合には記述しなければ表示されることは無い。セル内データの入力はデータ間を & 記号を入力することでセルを区切る。データ入力終了には改行記号を入力し、次の行のデータへと移行する。
セルの結合では、データ構造の特徴からコマンドの使用に注意が必要である。列を挟んだ横の結合（以降、列結合という）と行を挟んだ縦の結合（以降、行結合という）は呼び出しに違いがある。列結合では、以下のコマンドを利用する。

```
\multicolumn{結合セル数}{縦罫線と配置}{セル内データ}
```

ここで注意すべきは、縦罫線と配置の指定である。これは結合したセルに入るデータの配置と結合されたセルの端の縦罫線の指定である。記述は表作成の宣言と同様の方法で書かれることになる。この指定は結合したセルの端の縦罫線の指定が解除され罫線が無いものと判断されるため必要となる。しかし、この指定は結合したセルの位置によって異なってくる。下 Fig.1 に示すように左端のセルが結合対象となった場合、結合後のセルの両端の縦罫線の情報が指定範囲となる。その他のセルに関しては、結合後のセルの右端の縦罫線の指定だけとなり、左端は表作成の宣言で縦罫線を定義していた場合、自動的に表示されることになる。

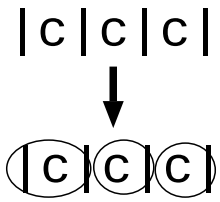

Fig. 1 LaTeX の構造

行結合は、結合部分でコマンドを宣言するだけでは使用することができない。これは、

表作成の宣言より前にあるドキュメント開始の宣言の位置で、

```
\usepackage{multirow}
```

という宣言が必要となることに注意する。この宣言をした上で、

```
\mulltirow{結合セル数}{結合したセルの長さ}{セル内データ}
```

とする。結合したセルの長さは、 * であれば文字列の長さを使用することとなる。ここでは配置の指定はなく、自動的に結合した中央に配置されることになる。結合後は、行結合では結合を宣言した下のセルは結合前と同様に存在するかのように扱われるので、データを入れてはいけないことに注意する。また、行結合をした場合、結合セルのがかかる横罫線部分は\cline{x-y}によって線を非表示状態にしなければならない。
行結合は、横罫線の指定によって擬似的に結合を表現することも可能である。この場合、セルは結合されていないのでそれぞれのセルの入力が可能である。
これらの作表データは、以下のような環境で囲むことで空きすぎのページができないように、最適な位置に表が配置される。

```
\begin{table}
作表データ
\end{table}
```

また、\usepackage{}宣言によって行結合だけでなく様々なコマンドが使用可能になる。コマンドの使用にはそれに対応するオプションを{}内に記述する必要がある。

今回の表の変換で対象となるコマンドを以下に示す。

\begin{table}〜\end{table}	表の配置
\begin{tabular}{l|l}〜\end{tablar}	作表開始宣言
\hline	横罫線指定
\cline{x-y}	指定位置に横罫線をひく
\multicolumn{結合セル数}{縦罫線と配置}{セル内データ}	列結合

3 HTML の作表構造

3.1 HTML とは

HTML(HyperText Markup Language) とは、Web ページを記述するためのマークアップ言語である。文書の中に画像や音声、動画、他の文書へのハイパーリンクなどを

埋め込むこともできる。HTML で記述された文書を閲覧するには通常 Web ブラウザを使用する。しかし、HTML 文書はテキスト文書の一種であるため、テキストエディタで HTML 文書を開き、テキスト文書として読み書きすることも可能である。

3.2 作表構造

HTML での作表は、面の情報を記述し、その周辺に線を引いていくというような構造である。また、行数や列数の指定をしないので、自由にセルの増減ができるが、列数はセルの最も多い行位置に合わせるので、その他の少ないところには空白セルができることになる。そのため作表されたものは下 Table 2 のように全セルが枠線で囲われる。面の情報に依存しており、部分ごとの線の指定はできない構造となっている。このため複雑な表の作成には適していない。面のデータを中心とした構造のため、結合などによっては作表が難しくなる場合がある。以下に作表の要素を示す。

a	b	
c	d	e
f	g	

Table 2 HTML で作表した場合の表例

以下は表 2 の作表コマンドである。

```
<table border>
<tr>
<td> a </td><td> b </td>
</tr>
<tr>
<td> c </td><td> d </td><td> e </td>
</tr>
<tr>
<td> f </td><td> g </td>
</tr>
</table>
```

以降に表 2 でのコマンドを解説する。
表作成の宣言は以下のようになっている。

```
<table border>
```

表データ
</table>

ここでは作表タグで開始を宣言し、border オプションで枠線の表示の指定をしている。border オプションを指定しない場合には、この表は枠線のない形で表示することになる。この他で罫線の指定をする場所はないので、枠線のあるなしのどちらしか選択することはできない。
HTML では、行ごとにデータを区切り表を作成していく。表データの記述を以下に示す。

<tr>列データ</tr>

このタグは上下に境界線を設ける行コマンドである。これは行ごとに定義しなければならない。<tr>と</tr>ではさまれた列データ部分には以下に示す列に分けるタグが入る。

<td>セル内データ</td>

これにより行分けされたデータが列ごとに分けられセルができることになる。このタグは列コマンドと同様の働きをする。行コマンド内でこの列コマンドはいくつも定義することができる。また、これと同様に、

<th>セル内データ</th>

というタグも列コマンドと同様の働きをする。このコマンドは<td>〜</td>に代わって使われることで、<th>〜</th>ではさまれたデータを見出しとして判断する。見出しとなった部分は、太字で表示される。<tr>〜</tr>が作表上の親となり、<td>〜</td>や<th>〜</th>はその内部に作られる子となる。

セルの結合では、HTML の作表特性から複雑化する場合がある。列結合では問題は発生しないが、行結合では作表が複雑化する。解説は以降に示す。列結合、は<td>または<th>にオプション指定を colspan とする。

<td colspan="X">〜</td>

この場合、X には結合する要素数を指定する。<th>で指定した場合、通常の作表と同じように見出し文字となり、太字となる。行結合も列結合と同じように<td>または<th>にオプション指定をする形となる。オプションは rowspan とする。

<td rowspan="X">〜</td>

X には結合する要素数を指定する。宣言は上のセルで行われ、それ以下の行の領域に影響を及ぼす。行結合を宣言されると、結合範囲は大きな 1 つの面と判断されるので、範囲にかかる行は結合された部分の入力がなくなるため、ずれが生じてくる。そのために、完成形を想像しながら作っていくことになり難しくなってしまう。

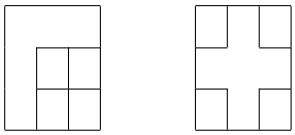

Table 3 HTML では表現できない表例

HTML では Table 3 のような複雑な表は表現できない。左の表例の形は、行結合と列結合を同じ位置で宣言すると、X 行．Y 列と結合をしたとすると，X と Y の対角頂点の長方形状に結合が行われ大きな１つのセルとなる。そのため表４のように１行１列目で列結合を宣言し、２行１列目で行結合をするか、１行１列目で行結合を宣言し、１行２列目で列結合をするようにして分けて考えるしかない。
右の表例では、結合列と結合行が交差する形になっているが、HTML で表現しようとすると、２つの面が重なってしまい正しい出力を得られない。
複雑な表は色指定を行い周りの色と合わせたり変えたりして表現することはできるが、構成が繁雑になってしまうために適応しないブラウザでは内容を理解できないものとなってしまう。また、２つのつながれていない表を横に並べて表示することもできない。よってそのような場合には、大きな１つの表として作るか、上下に分けて表示させなければいけない。

今回の表の変換で対象となるコマンドを以下に示す。

`<table border>〜</table>`	作表開始宣言
`<tr>〜</tr>`	行指定
`<td>〜</td>`	列指定
`<td colspan="X">〜</td>`	列結合
`<td rowspan="X">〜</td>`	行結合

4 開発ソフトの構造

4.1 作表構造

作表は LaTeX や HTML と同様にテキストエディタ上で行っていく事を前提とする。
HTML の作表は面を中心とした形式のため線の自由がない。LaTeX では線の記述はできるものの、構造や作表自体が複雑なため作りづらい。それに完全な分離ができているわけではないので、依存が生じてしまう。
今回の提案する形式では、面と線の情報を分離して記述することができる構造を考える。横罫線の情報と縦罫線の情報、そして記入されるデータの情報を分離することで、それぞ

れに依存せず作表を進めることができるので任意の形を条件を考えず構成できる。はじめに、考えられた形式を提示する。

4.1.1 形式１

この形式は、コマンドで記述していく方法である。コマンドが複雑なものではLaTeXやHTMLとかわらず作表が難しくなってしまうので、結合など基本の格子状の表形状から変わる部分での使用を考えた。基本的な格子状の表 (Table 4) を作成する場合の記述例を以下に示す。次に結合の含まれる表 (Table 5) の記述例を示していく。

a	b	c
d	e	f
g	h	i

Table 4 基本表

```
$$$
&a&b&c&
$$$
&d&e&f&
$$$
&g&h&i&
$$$
```

以下に解説をする。
ここで作表の開始・終了を宣言するコマンドは不要である。
まず、最初に記述されている `$` はセルごとの横罫線を表している。基本表は３列となっているので、 `$` は３つ記述されることになる。次に `&` だが、これは縦罫線を指定している。セル中に配置されるデータは、ここで `&` ではさまれた形で記述される。
複雑な表を作成する場合は、 `$` や `&` の代わりに他のコマンドを用いることで表現していく。結合の存在する場合の表例とコマンドをを以下 Table 5 に示し、コマンドの解説を行う。

```
$$$
&row{2}a&col{2}b&
*$$
&&c&d&
```

a	b	
	c	d
e		f

Table 5 結合のある表例

```
$$$
&col{2}e&f&
$$$
```

この表例の場合、まず1行目の横罫線の情報は基本表と変わらない。2行目に `row{2}a` とあるが、これは行結合を表すコマンドである。このとき、`{}` 内には結合行数が指定される。その後、セルに入るデータを記述していくようになっている。次に、同行に `col{2}b` とある。これは列結合を表すコマンドである。このコマンドも行結合のと同じように、`{}` 内には結合列数が指定される。その後にデータを記述していく。3行目を見ていくと、始めに `*` がある。このコマンドは、横罫線の指定であり、表示させない部分に記述していく。これは、2行目で行結合が行われているため、ここで横罫線の指定を行わないと結合セルに線が入ってしまう。なので、この指定は行結合とセットで使われることになる。4行目は、1列目のセルデータが入力されていない。これも2行目の行結合の影響で、ここにはデータを入力することは許されない。5行目は、基本表と同じである。6行目にある `col{2}e` も2行目にあるコマンドと同様である。以降基本表と同じである。

以下にこの作表形式で用いられる全コマンドをまとめる。

`$`	横罫線表示指定
`&`	縦罫線表示指定
`*`	横罫線非表示指定
`#`	縦罫線非表示指定
`col{x}`	列結合指定 (`{x}` は結合列数)
`row{x}`	行結合指定 (`{x}` は結合行数)

この形式での長所は、

- 作表のコマンドが単純なので分かりやすい。
- 変換時に対応がとりやすい。

また、短所は、

- LaTeX の形式に類似している
- LaTeX と HTML の中間形式にはなっていない
- 構造の理解が必要

などが挙げられる。この形式では、1行だけセルが飛び出している表や、セルの少ない表の表現なども可能である。しかし、この形式では横の作表構造は表現できるが、縦の作表構造は表現することは難しい。

4.1.2 形式2

この形式では、表を縦罫線要素と横罫線要素とセル内データを分けて記述していく。記述方式は、縦罫線、横罫線を記号で表記していく。まず Table 4 の基本表を表す場合の記述法を以下に示す

```
||||
||||
||||

---
---
---
---

0.0:a
0.1:b
0.2:c
1.0:d
1.1:e
1.2:f
2.0:g
2.1:h
2.2:i
```

以下に解説をする。
ここで作表の開始・終了を宣言するコマンドは不要である。
まず、 | は縦罫線の表示指定をしている。ここで表全体の縦罫線の指定をしている。こ

の表記は、行ごとに記述していく。次に、 - は横罫線の指定をしている。ここでも縦罫線の指定と同じように、表全体の横罫線の指定をする。セル内データの記述は、

```
x.y:データ
```

のようになる。ここで x は行番号を示すが、行のカウントは 0 からとなっている。y は列番号を示している。列番号も行番号と同じようにカウントは 0 からとなっている。また、セル内データの記述順序は記述例のように行ずつの記述の仕方だけではなく、順不同に宣言することが可能である。
縦罫線・横罫線・セル内データはそれぞれ空行１行で区切り記述する。結合の存在する表を作成する場合は、非表示罫線の部分をこれらとは違う記号で表現していく。以下に表５そ作成する場合の記述例を示していく。

```
||#|
||||
|#||

---
#--
---
---

0.0+1.0:a
0.1+0.2:b
1.1:c
1.2:d
2.0+2.1:e
2.2:f
```

以下に解説をする。
１行目の縦罫線の宣言部分で # とあるが、これは結合表現である。この部分は Table 5 で分かるように、列結合があるため # となっている。３行目にあるものも同様である。次に、横罫線の宣言部分でも # となっている。ここでは、行結合が行われているためこのような指定となっている。結合部分のセル内データの記述部分では、

```
0.0+1.0:a
```

のようにセル座標を + でつなぐことで宣言されている。これは、0.0 のセルと 0.1 のセルの結合を行い、結合部分に a を入力することを示している。この宣言は、

```
x0.y0+x1.y1+x2.y2+...:z
```

のように、結合するセルの数だけセル指定をする必要がある。
列結合だけではなく、行結合も同様に宣言される。
また、 $ を用いることで、指定位置の罫線の非表示指定ができるため、疑似的な結合表現も可能である。この場合、セルが結合されないので、データ入力部分は格子状に存在したままである。

以下にこの作表形式で用いるコマンドをまとめる。

-	横罫線表示指定
|	縦罫線表示指定
$	罫線非表示指定
#	結合表現
x.y:データ	セルデータ入力 (x は行番号，y は列番号)
x0.y0+x1.y1+x2.y2+...:データ	結合指定

この形式の長所は、

- 表の完成形が分かりやすいので作表が簡単
- 作表データが見やすい
- データの入力が自由である

また、短所は

- 表の形が決った形 (長方形の格子状) である
- 変換時の対応が難しい

などが挙げられる。この形式での作表では、長方形の形の中に格子状にセルが配置されているので、１行だけセルが多い、もしくは少ないなどは認められない。しかし、罫線の表示・非表示を組み合わせることで表現することは可能である。縦構造の表を作成する場合には、

```
---
---
```

a	d	g
b	e	h
c	f	i

Table 6 縦構造での表例

```
---
---

||||
||||
||||

0.0:a
0.1:b
0.2:c
1.0:d
1.1:e
1.2:f
2.0:g
2.1:h
2.2:i
```

のように記述することで、表6のような形で出力することも可能と考えられる。

4.2 変換構造

今回の変換ソフトでは、形式2を採用することにする。これは、作表時の簡単さと線の構造の自由度、縦構造の対応を考え、この形式とした。まず、この変換ソフトの全体の流れを Fig.2 のフローチャートに示す。

目標の変換ソフトは、LaTeX や HTML の文章部分などを無視し、表情報だけを取り出し、目的の形式に変換するものを最終的な目標とする。
変換方式は3種類に分けられる。1つは、テキストの形式で作表され LaTeX や HTML に変換する。2つ目は、LaTeX の形式で作られ、その表を HTML 形式に変換する。3つ目は、HTML 形式で作られ、LaTeX の形式に変換する方法である。現在考えている変換対象は LaTeX と HTML の2つの形式なのでこのようになっているが、変換する対象が増え

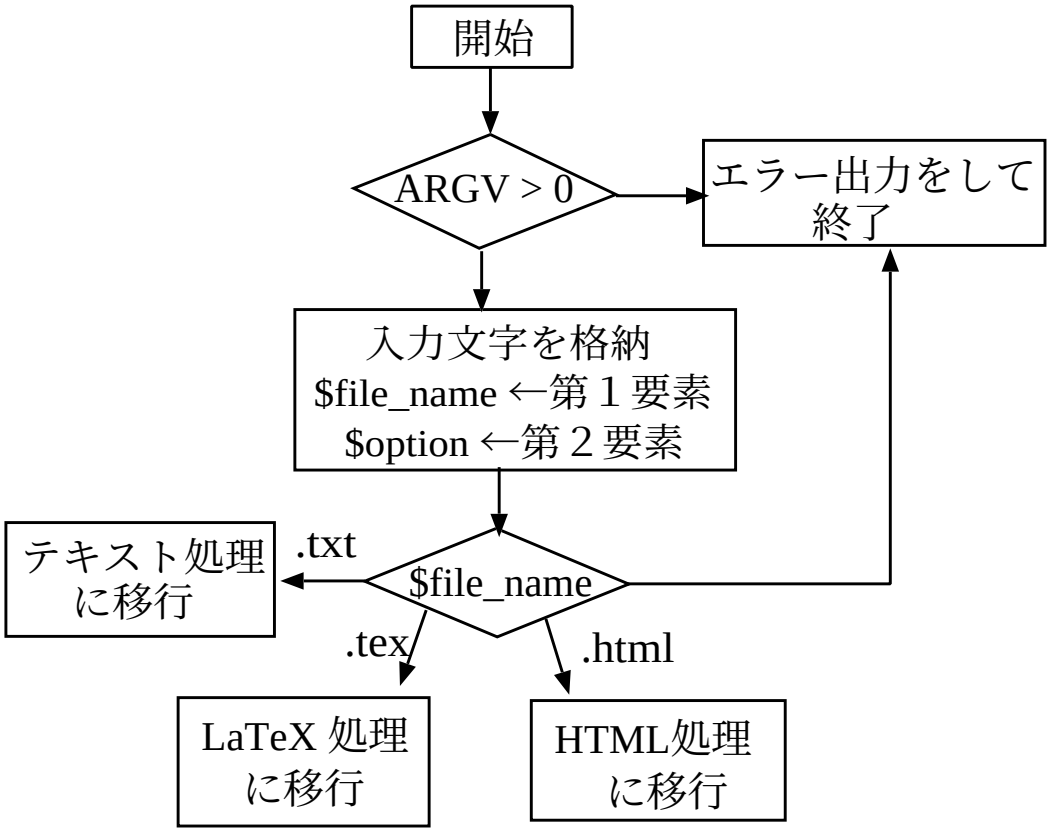

Fig. 2 全体の流れ

ることによりこの形は変わっていく。このプログラムは、実行時に以下のように変換するファイル名と変換先の指定が必要となる。

```
perl henkan.pl test.txt html
```

これは、`henkan.pl` という変換プログラムに `test.txt` というテキストファイルを渡し、`html` の形式に変換することを意味している。

全体の共通の処理として、起動時にファイルの名前と変換先の指定が無い場合は、エラー出力をして終了する。次に、ファイルの拡張子を判断し、`.html .tex .txt` 以外の場合にはエラー出力をして終了する。変換先の指定も同様に、`html latex` 以外であった場合には終了する。ファイルの拡張子の判断が終了すると、それぞれの変換作業に移る。以下にそれぞれの変換の流れを解説していく

4.2.1 テキスト変換処理

テキストの形式で作表して、そのファイルを変換する場合、変換する前に作表したデータを確認する必要がある。この確認は、作表された表の範囲に正しくデータが配置されているのか、範囲外にデータが配置されていないかを確認する作業である。テキストファイル読み込みから変換までの流れを Fig.3 のフローチャートに示し、解説する。

1. 変換ファイルを読み込み、行ごとに格納する。行ごとに格納したデータの表を表現する部分 (| や - で構成されている部分) だけを１字ごとにばらし、２次元配列に格納していく。このとき格納されたデータは、目標の表の形と同じように格納されている。格納例を以下に示す。

   ```
   ---      基本表表記の５行目
   ||||              １行目
   ---               ６行目
   ||||              ２行目
   ---               ７行目
   ||||              ３行目
   ---               ８行目
   ```

   これは表４の基本表の記述を読み込み格納した場合である。このように格納することで、変換時に線の情報が簡単に参照できるのでこのように格納している。

2. データ情報の格納をしていく。データ情報は、`x.y:`データと記述されているので、まずセルの座標データを取り出すために一度データをばらし、座標データを入手する。その後データ用の２次元配列にばらしたデータを再びつなぎ合わせ、座標の位置に格納する。このとき、座標が表の外にあった場合、エラー出力をする。

ここまでがファイルのチェックである。
次に、テキストから HTML への変換手順を解説する。
テキストから HTML への形式へ変換する場合、線情報を見ていくよりセルの中に入るデータ情報を見ていくと効率が良い。これは、HTML の特性によるものであり、HTML ではセルデータの周辺に線を自動で引いていくので、セルのデータを読み取りそのデータを書き込んでいく。しかし、これは結合の無い場合であり、結合がある場合には線の情報とデータ情報を照らし合わせて変換していかなければならない。
Table 3 で示したような HTML では表現ができない表がテキストで作成されていた場合、これは１つのセルに複数の結合命令がある状態なので、このような場合には、エラー出力をして終了する。LaTeX の作表でも１つのセルに対しての２つ以上の結合命令などは受け付けていないのでこのように処理をしている。その他の線の表示・非表示の細かい指定はできないので、セルの周辺に線を引く形となる。

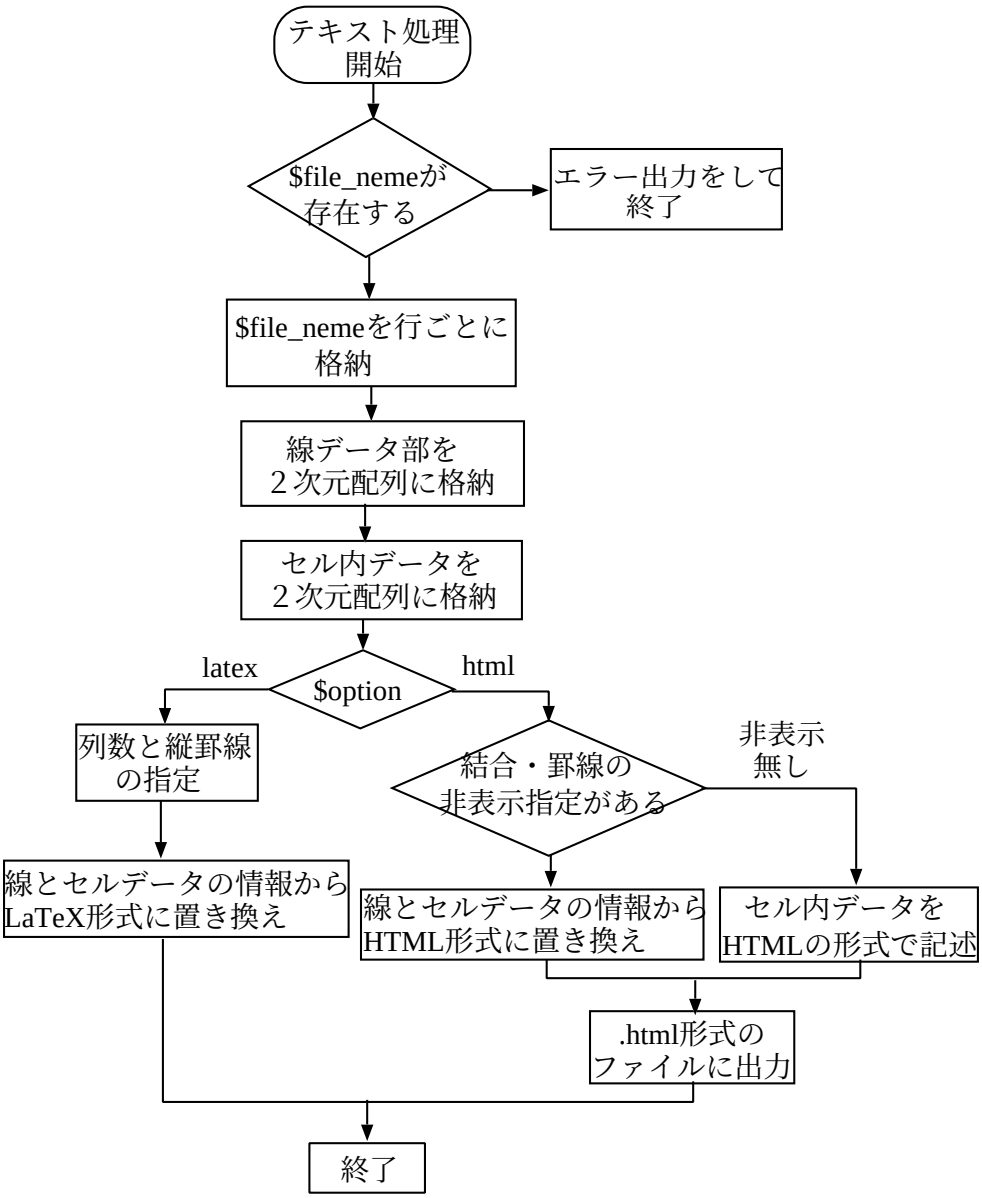

Fig. 3 テキストからの変換の流れ

LaTeX への変換を解説していく。
テキストから LaTeX の形式へ変換する場合、まず列数を調べる必要がある。これは \begin{tabular}の宣言で、{|c|c|c|}のように列の指定があるために必要となる。このとき、線データを確認し、縦罫線情報が全て非表示の状態であった場合には、縦罫線の表示指定をなくさなければいけない。次に、横罫線情報を見ていき、LaTeX の形式に置き換えていく。置き換え時、横罫線情報が全て非表示を示しているならば、その行の横罫線の指定はない。セル内データを入力するとき、そのセルの右の罫線情報を同時に確認する。このとき、結合表現が存在した場合、結合セル数を数え LaTeX の結合コマンドと置き換える。縦罫線が表示の状態で、セルの右の線情報が罫線非表示であった場合も、LaTeX の結合コマンドを用い罫線を非表示にしておく。

4.2.2 HTML 変換処理

ここでは、ファイルの拡張子が.html であった場合の処理の解説をする。.html から変換する場合、一度.txt 形式に変換し、そこから他の形式に変換する。.txt からそれぞれの形式への変換は 4.2.1 節で行っているのでここでは、.html から.txt へ変換する流れを Fig.4 のフローチャートに示し解説する。

HTML 形式のファイルをテキストの形式に変換する場合、まず、HTML のファイルを読み込み、行ごとに格納する。格納したデータから<table>部分を確認する。ここで <table border>となっていた場合、線データは全て表示指定となる。<table>となっていたとき、線データは全て非表示の状態で保存されることになる。次に、<tr>から</tr>で区切られる行データ内のセル内データを格納していく。これは、<td>セル内データ</td>部分を見ていき、格納していくことになる。全ての行データの格納が終了すると、縦罫線データ・横罫線データ・セル内データに分割し、テキストファイルとして出力をして、次の目標の形式への変換に移行する。

4.2.3 LaTeX 変換処理

こでは、ファイルの拡張子が.tex であった場合の処理の解説をする。.tex から変換する場合、一度.txt 形式に変換し、そこから他の形式に変換する。.txt からそれぞれの形式への変換は 4.2.1 節で行っているのでここでは、.tex から.txt へ変換する流れを Fig.5 のフローチャートに示し解説する。

LaTeX 形式のファイルをテキストの形式に変換する場合、まず、LaTeX のファイルを読み込み、行ごとに格納する。格納したデータから、\begin{tabular}部分を確認する。ここで、縦罫線と列数の表示指定から、縦罫線の表示・非表示部分指定を確認し格納する。次に、横罫線の情報を見ていく。\hline となっていたら、１セル目から最終セルま

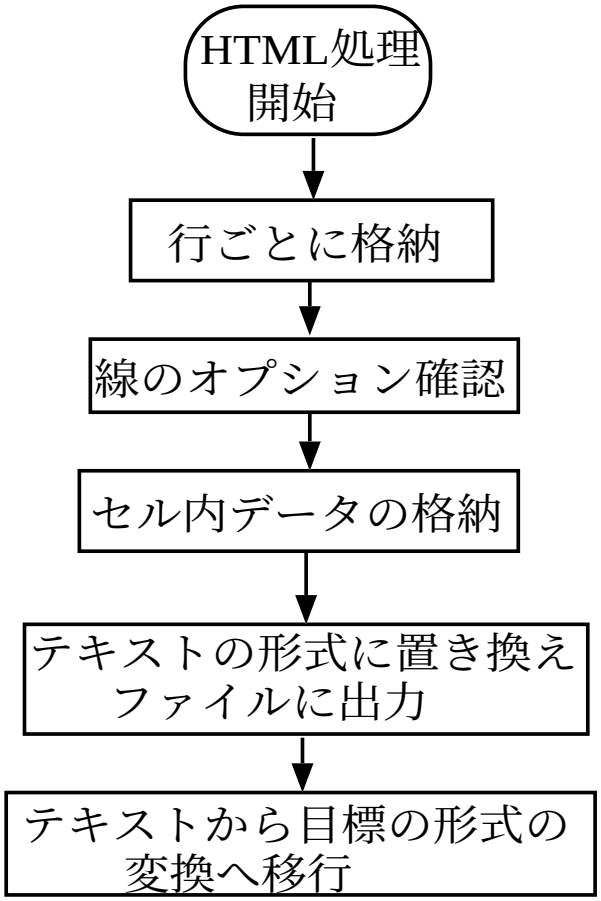

Fig. 4 全体の流れ

で全てに横罫線を表示するので、その情報を格納する。`\cline{x-y}`となっていた場合`{x-y}`で指定されている範囲にだけ線を表示させるので、その情報を格納していく。ここで、`\end{tablar}`が宣言されていた場合格納を終了する。宣言されていなかった場合には、セル内データの格納を行う。セル内データは `&` で分けられているので、これで区切り格納する。この横罫線の格納からデータの格納までの流れを`\end{tablar}`まで繰り返す。全ての格納が終了すると、縦罫線データ・横罫線データ・セル内データに分割し、テキストファイルとして出力をして、次の目標の形式への変換に移行する。

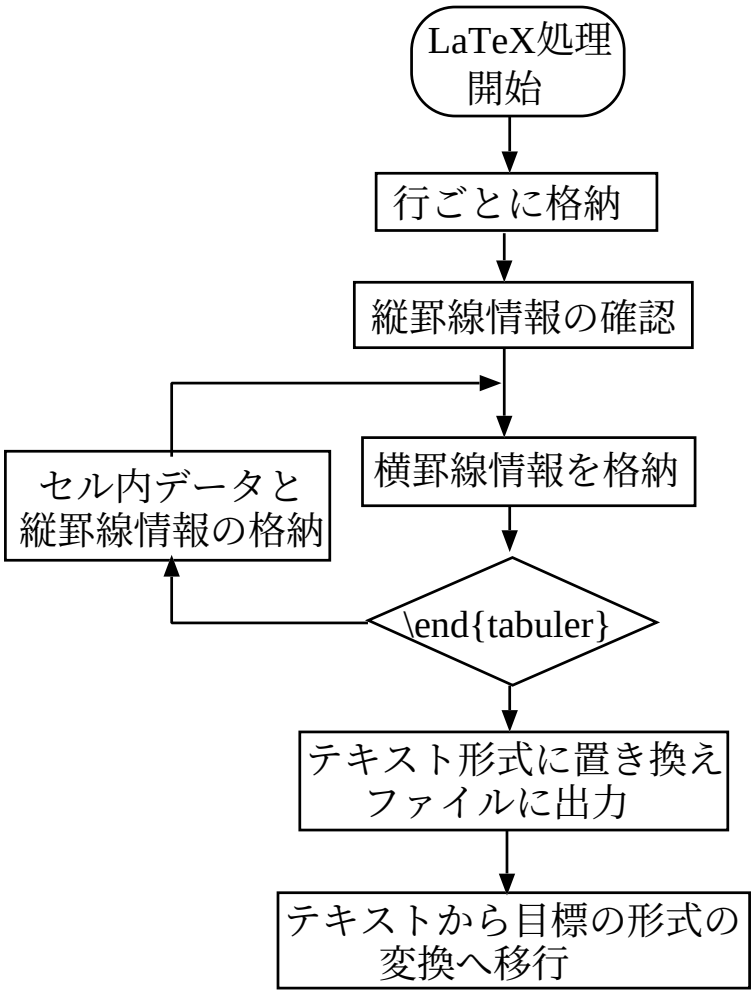
LaTeX処理
開始
行ごとに格納
縦罫線情報の確認
横罫線情報を格納
セル内データと
縦罫線情報の格納
\end{tabuler}
テキスト形式に置き換え
ファイルに出力
テキストから目標の形式の
変換へ移行

Fig. 5 全体の流れ

5　まとめ

本稿では、LaTeX と HTML の作表構造の解説し、変換ソフトの基本構造の提案、変換時の全体の流れについて考えてきた。調べてみると、LaTeX と HTML とを変換するソフトはいくつか確認できた。だが、それらのほとんどが特殊な形式での保存であるため、内容の確認が簡単にはできないこともわかった。そのため今回の研究では、テキストベースでの作表を提案した。

それぞれの作表構造を調べてみると、その作表の特性が見えてきた。HTML では、面の情報に依存しているため線の定義が不要なので、単純な表は簡単に作ることができるが、線の指定ができないため作れる表が限られてしまう。一方 LaTeX では線と面の情報が記述できるが、構造の依存と作表の複雑さが難点であった。

本研究では、新たなデータ構造の提案と変換の流れを考えたが、実際に変換ソフトを完成させることはできなかった。

今後の課題として以下のようなことが挙げられる。
今回の構造を基に変換ソフトを完成させる。提案した構造は、基本的な作表の対応を考えただけなので、今回報告したコマンド以外が使われていた場合、表現できない。よって、報告したコマンド以外の対応も考える必要がある。その他に、変換先の拡張を行い、LaTeX や HTML 以外の形式に対応させることも考えられる。

参考文献

[1] 奥村 晴彦: “LaTeX 2_ε美文書作成入門”, (技術評論社,1997)

[2] シーズ: “HTML ポケットリファレンス”, (技術評論社,1997)

[3] 東京情報大学 Mizutani Laboratory: “オンラインコンピュータ講座”
`http://www.rsch.tuis.ac.jp/~mizutani/online/index.html`