MathML 数式組版入門

2017 年 7 月 28 日　ver. 1.1

道廣勇司[1)]

本文書は数式記述言語 MathML を使って数式組版を行うためのガイドです。内容の大半は特定のアプリケーションソフトに依存しないものですが，MathML に対応した組版ソフト Antenna House Formatter（以下，AH Formatter）の使用を前提とし，そのために有用な知識を随所に盛り込みました。

執筆にあたっては，AH Formatter の開発元であるアンテナハウス株式会社の方々より多大なご支援をいただきました。ただし内容についての一切の責任は筆者にあります。筆者は MathML の専門家ではありません。誤りなどがあればぜひご指摘ください。

[1)] 連絡先は http://moji.gr.jp/mich/

第1章 はじめに

AH Formatter の動かし方や，XSL-FO，HTML，CSS の書き方については他の資料に譲る。

また「要素」「子要素」「空要素」「属性」「(CSS の）プロパティー」「文字実体参照」「文書型」「DTD」「XHTML」といった基本的な用語は既知とする。

本文書では，丸括弧 ( ) のことを印刷業界の慣習に従い，「パーレン[2)]」と呼ぶことにする。一方，パーレンや [ ] や { } のような括り記号の総称として「括弧類」を使用することにする。英語の brackets に対応する。

1.1 MathML

MathML（マスエムエル）は XML 形式で数式を記述するためのマークアップ言語だ。XML 形式ゆえ，機械（コンピューター）に伝えるのに適しており，人間が見た場合の可読性もソレナリにある。

ところで数式を記述するとはどういうことか。例えば x^2 という数式を考えてみよう。

これを機械に伝えるのに，「x の右肩に 2 という添字（そえじ）が付いている」のような見た目を伝えたい場合と，「x を 2 乗したもの」という意味内容を伝えたい場合があるだろう。

MathML はどちらの用途にも対応できるよう，二通りの語彙体系を用意している。

見た目を記述するほうを「プレゼンテーション MathML」，意味内容を記述するほうを「コンテント MathML」と呼ぶ。また，それらによるマークアップをそれぞれ「プレゼンテーションマークアップ」，「コンテントマークアップ」と呼ぶ。

さきほどの x^2 は，プレゼンテーションマークアップでは以下のように書く（簡単のため，名前空間指定は略す）。

```
<math>
  <msup>
    <mi>x</mi>
    <mn>2</mn>
  </msup>
</math>
```

それぞれのタグの詳細は次章以降で説明するが，ここでは以下の点だけ分かれば十分だ。

- 全体を `math` 要素に入れる。
- `msup` 要素は上付き（superscript）を組むためのもの。
- `msup` は第一子要素を本体（base）とし，第二子要素をその右肩に配置する。

一方，コンテントマークアップでは，以下のように書く。

```
<math>
  <apply>
    <power />
    <ci>x</ci>
    <cn>2</cn>
  </apply>
</math>
```

`apply` は演算の適用を意味し，`power` は累乗を意味する。

コンテントマークアップされた数式は数式処理（数式をプログラムで変形したり合成したりすること）にも使うことができる。しかし，コンテントマークアップはどんな数式でも書けるわけではない。

AH Formatter はコンテントマークアップの MathML を読み込むと，いったんプレゼンテーションマークアップに変換してレンダリングする。しかし，本文書ではコンテントマークアップについてこれ以上は触れない。

MathML の仕様はバージョン 2.0 から 3.0 にかけて大きく飛躍した。本文書では MathML 3.0 のみを念頭に置く。

---

2) 英語の parenthesis（複数形は parentheses）に由来する。

対象とする AH Formatter のバージョンは，本文書執筆時点の最新版である V6.4 以降とする。AH Formatter の MathML 組版機能の改良は継続的に行われているので，古いバージョンで試すと本文書と食い違うこともあるだろう。

1.2 FO 組版と CSS 組版

AH Formatter は XSL-FO による組版と，HTML+CSS による組版の二つの組版機能を持っている。本文書ではそれぞれ FO 組版，CSS 組版と呼ぶことにする。

FO 組版と CSS 組版とで，基本的には MathML の書き方に違いは無いが，文書中への埋め込み方や名前空間については少し事情が異なる。

FO では，文書中に `math` 要素を埋め込む際，`fo:instream-foreign-object` 要素に入れる必要がある。また，MathML の名前空間の指定が必須となる。よって，「変数 x は…」を組ませる書き方は以下のようになる。

```
変数
<fo:instream-foreign-object>
  <math xmlns="http://www.w3.org/1998/Math/MathML">
   <mi>x</mi>
  </math>
</fo:instream-foreign-object>
は…
```

一方，CSS 組版では，HTML において同じものを以下のように書くことができる。

```
変数 <math><mi>x</mi></math> は…
```

本文書では簡単のため後者の書き方で例を示す。また，とくに必要がなければ `<math>` タグをも略すことにする。

1.3 Unicode のコード番号

Unicode のコード番号は，「U+」の後に 4〜6 桁の 16 進数を続けた「U+222B」のような形で表示する。これが正式な表記である。

番号が 3 桁以下の場合は，U+0009（タブ），U+0041（‘A’），U+0259（‘ə’）のように，「0」で埋めることになる。

基本多言語面（BMP: Basic Multilingual Plane）の文字は全て 4 桁表示になり，それ以外の文字は，U+29E15（「鯵」）のように 5 桁または 6 桁となる。

1.4 文字実体参照

数式に現れうる記号は極めて多数ある。それらのうち，英数字や「+」や「=」などのように容易に入力できるものはほんの一部に過ぎない。

幸い，容易に入力できない記号でも，そのほとんどは Unicode で定義されているため，数値文字参照を用いて表すことができる。例えば ≃ という記号は U+2243 として定義されているので，`&#x2243;` と入力すればよい。

しかし，多数の記号のコード番号を記憶するのは容易ではなく，入力間違いも起こしやすい。そこで `&rarr;` のような文字実体参照がもしあれば，それを使うことになる。ただ，文字実体参照の利用には少し注意が必要なので，XSL-FO の場合と HTML の場合に分けて説明する。

なお，数学記号も含め，HTML5 で使える文字実体参照の一覧は下記を参照されたい。2000 以上の文字実体参照が掲載されている。AH Formatter ではこれら全てが使える。

https://www.w3.org/TR/html5/syntax.html#named-character-references

1.4.1 XSL-FO の場合

XSL-FO は XML だ。XML である以上，黙っていても使える文字実体参照は以下の五つしかない。

&	<	>	"	'
&amp;	&lt;	&gt;	&quot;	&apos;

これ以上の文字実体参照を使うのであれば，そのための DTD を指定することになる。

例えば $X \to Y$ という式を埋め込む `fo:instream-foreign-object` 要素は以下のような記述になる。

```
<fo:instream-foreign-object>
<![CDATA[<!DOCTYPE math PUBLIC "-//W3C//DTD MathML 3.0//EN"
  "http://www.w3.org/Math/DTD/mathml3/mathml3.dtd">
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <mi>X</mi><mo>&rarr;</mo><mi>Y</mi>
  </math>
]]>
</fo:instream-foreign-object>
```

本文書では分かりやすさのため組版例に文字実体参照を多用するが，FO 組版の場合，そういった文字実体参照が使えるような記述を行うか，数値文字参照に読み替えるなどしていただきたい。

1.4.2 HTML の場合

本文書では，HTML の文書型としては HTML5 だけを念頭に置く。HTML 4.01 など他の文書型を使うメリットはとくに無いだろう。

HTML5 には，いわゆる HTML 形式と XHTML 形式の二つがある。XHTML 形式の場合，これもやはり XML であるので，本来であれば最初から使える文字実体参照は五つしか無いはずである。しかし，AH Formatter はどちらの形式でも，先に URL を掲げた参照先に挙げられている全ての文字実体参照が使えるようになっている。

したがって，以下は期待どおり ⊗ が組まれる（この例では `math` 要素を使っていないが，使った場合も同様）。

```
<!DOCTYPE html>
<meta charset="UTF-8">
<title>sample</title>
<div>&otimes;</div>
```

1.5 トークン要素とレイアウトスキーマ

プレゼンテーションマークアップのための MathML 要素を「プレゼンテーション要素」と呼ぶ。プレゼンテーション要素は，トークン要素（token elements）とレイアウトスキーマ（layout schemata[3]）に大別される。

トークン要素は個々の記号・識別子・数字といったものを表す。`mi`，`mn`，`mo`，`mtext`，`mspace`，`ms` が該当する。

レイアウトスキーマは添字類，分数，累乗根，表組などを表す。分数における水平線や累乗根における根号（という記号）そのものを表す要素は無い。

1.5.1 空白文字の扱い

XML 文書では，人間に見やすくしようとして闇雲に改行やインデントを入れると，結果に影響を及ぼすことがあるため注意が必要だ。MathML ではどうだろう。

第一に，トークン要素の外側の空白文字[4] は無視される。

---

[3] schemata は schema の複数形。

[4] ASCII のスペース（U+0020），タブ（U+0009），CR（U+000D），LF（U+000F）を指す。

第二に，トークン要素の内容の中身の先頭・末尾にある空白文字は無視される。そして，中身の中途に現れる空白文字列は，HTML におけると同じように，単一の ASCII のスペースに置き換えられる。

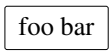
したがって，以下の三つは MathML 文書として完全に等価だ。

```
<math><mo>+</mo><mi>x</mi></math>
```

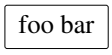
```
<math>  <mo>+</mo>    <mi>x</mi>  </math>
```

```
<math>
  <mo>  +  </mo>
  <mi>  x  </mi>
</math>
```

以下の二つも等価である。



```
<mtext>foo bar</mtext>
```



```
<mtext>
  foo
  bar
</mtext>
```

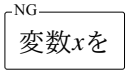
MathML 3 仕様書

2.1.7 Collapsing Whitespace in Input
3.1.2.1 Types of presentation elements

1.6　前後との空き

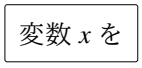
和文中に math 要素を入れるとき気をつけなければならないのは，和欧間の空きが自動的には入らないことである[5]。数式は欧文と同じ扱いになるべきなので，和文との間は空きを入れなければならない。以下はまずい例である。

NG



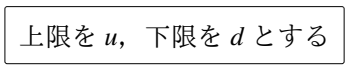
```
変数<math><mi>x</mi></math>を
```

いまの場合，math 要素の前後にあらわにスペースを入れてやらなくてはならない。以下のようにする[6]。



```
変数 <math><mi>x</mi></math> を
```

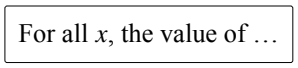
しかし，どんな場合でも math の前後にスペースを入れればよいかというと，そうではない。以下の例に見るように，スペースを入れてはいけない箇所もある。



```
上限を <math><mi>u</mi></math>，
下限を <math><mi>d</mi></math> とする
```

つまり，句読点の前や括弧類の内側などにはスペースを入れてはいけない。

ところで欧文中に数式が入る場合，数式を一つの単語のように考えてワードスペースを入れればよい，つまり，



```
For all <math><mi>x</mi></math>, the value of …
```

のように書けばよい。

なお，英文組版では，物理量を示すとき，数と単位記号の間にはワードスペースを入れる。

よって，以下のようにすればよい。

---

[5] AH Formatter の現在の仕様がそうなっている。

[6] 欧文スペースの幅はフォントに依り，和欧文間の空き（AH Formatter の既定値は 0.25 em）とは違いうる。また，ジャスティファイに伴う調整の量も AH Formatter では異なる。しかしそこに拘泥すると大変なので，本文書ではこれでよいことにする。

The length is 25.4 mm.

```
The length is 25.4 mm.
```

1.7 ディスプレイスタイルとインラインスタイル

数式は，それが置かれた場所によって「ディスプレイ数式」または「インライン数式」と呼ばれる。ディスプレイ数式は別行立て数式とも呼ばれ，

$$x = 0$$

のように本文から独立した行をなす[7]。中央揃えにしたり，行末に数式番号を付けることもある。一方，インライン数式は $x = 0$ のように，文字どおり本文の行中に埋め込まれた数式だ。

ディスプレイ数式とインライン数式とでは，置かれる場所だけでなく組版体裁（スタイル）にも違いがある。

MathML では math 要素ごとにディスプレイ数式とインライン数式が切り替えられるようになっている。

CSS 組版では，math 要素の display 属性を block にすると，ディスプレイスタイルで組まれ，独立の行となる。inline（既定値）とすれば行内にインラインスタイルで組まれる[8]。FO 組版の場合，display 属性を block にすると，ディスプレイスタイルにはなるが，それだけでは独立の行にならないので注意されたい。独立の行にするには，fo:instream-foreign-object を fo:block で囲む必要がある。

スタイルの違いについては，次の例を見てみよう。

ディスプレイスタイル <math display="block">	インラインスタイル <math display="inline">
$\frac{\pi}{2}\sum_{k=0}^{\infty}\int_0^1 f_k(x)dx$	$\frac{\pi}{2}\sum_{k=0}^{\infty}\int_0^1 f_k(x)dx$

ディスプレイスタイルに比べ，インラインスタイルは天地がコンパクトになっていることが直ちに分かる。

文字サイズが全体的に小さくなったように見えるかもしれないが，そうではない。f や x などのサイズ（これが基本のサイズ）を比べてみれば，違っていないことが分かる。

文字サイズが小さくなったのは，分数の分子・分母と，総和記号 $\sum$，積分記号 $\int$ である。

また，$\sum$ の上と下にあった記号が上付き・下付きの位置に変わった。

1.8 部分的にスタイルを変える

前節において，math 要素の display 属性でディスプレイスタイルとインラインスタイルが切り替わることを見た。

それとは別に，数式の一部分をディスプレイスタイルにしたり，インラインスタイルにしたりする手段がある。それには mstyle 要素の displaystyle 属性を使う。

mstyle 要素は，文字どおりスタイル（体裁）を制御する MathML 要素だが，詳細は第 11 章「スタイル」で扱う。

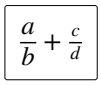
displaystyle 属性の値は true か false で，true だとディスプレイスタイルになり，false だとインラインスタイルになる。

次の例は display="block" で組まれた式の一部分に displaystyle="false" を適用したものである（実際的な例ではない）。

---

[7] ここでいう「ディスプレイ」は，見出しなどでの目立たせる組み方を意味する英語の組版用語 display をそのまま外来語として日本語に取り込んだものだ。

[8] display という属性名と「ディスプレイスタイル」が紛らわしいことや，block という属性値が「ディスプレイスタイル」と結びつきにくいことについては後で触れる。



```
<math display="block">
  <mfrac>
    <mi>a</mi>
    <mi>b</mi>
  </mfrac>
  <mo>+</mo>
  <mstyle displaystyle="false">
    <mfrac>
      <mi>c</mi>
      <mi>d</mi>
    </mfrac>
  </mstyle>
</math>
```

display 属性と displaystyle 属性とその属性値の関係が覚えにくいので，以下に整理する。

適用する要素	属性名	属性値
math	display	block/inline
mstyle	displaystyle	true/false

混乱しやすさのもとは，「display」という言葉が多義語であることによる。

既に述べたように，「ディスプレイスタイル」とか「ディスプレイ数式」といったときの display は組版用語であり，見出しなどの目立たせる組み方のことを指している[9]。

一方，math 要素の display 属性の名は，CSS の display プロパティーに倣っている。こちらは「表示（display）の仕方」から来ている。そしてその属性値 block，inline も CSS のそれと同じだ。

MathML 3 仕様書

3.3.4 Style Change <mstyle>
3.1.6 Displaystyle and Scriptlevel

1.9　引数

1.1 節で見た x^2 をもう一度取り上げよう。

```
<msup>
  <mi>x</mi>
  <mn>2</mn>
</msup>
```

添字の組み方については第 8 章で詳しく説明する。

msup は上付きを組むための要素だ。mi は識別子を，mn は数値を表す。

このように，msup 要素は二つの子要素をもつ。MathML の用語では，これらの子要素は msup 要素の「引数（ひきすう）（arguments）」と呼ばれる[10]。

「引数」は MathML で重要な用語で，本文書でも積極的に使っていく。この用語を使えば，

> msup 要素は第一引数の右肩に添字として第二引数を置く。

というような表現が可能になる。

1.10　寸法単位

寸法を指定する箇所で使う単位を説明する。

MathML 仕様書によれば em，ex，px，in，cm，mm，pt，pc，% が使える。順に見ていこう。

---

[9] 見出し用活字のことを display types と言ったりもする。

[10]「引数」は「いんすう」と読んでもよいが，「因数（factor）」と紛らわしいこともあって，ふつうは「ひきすう」と湯桶読みする。

em は「エム」と読み，文字サイズに等しい長さを意味する。つまり，文字サイズが 10 pt の場合，0.25 em は 2.5 pt である。

ex はいわゆる x-height（エックスハイト）で，文字 'x' の高さだ。アルファベットの小文字のデザインの基準の一つである。同じフォントサイズでも書体によって ex はけっこう違っている。垂直方向の位置決めやスペーシングにこの単位を使うと便利なこともある。

px はピクセルだが，印刷物を作るのに使うことはあるまい。

in はインチで，厳密に 1 in = 25.4 mm である。

pt はポイントである。ポイントにはいろいろな種類があるが，MathML のポイントは DTP ポイント，つまり 1 pt = $\frac{1}{72}$ in = $\frac{25.4}{72}$ mm である。

pc はパイカで，1 pc = 12 pt である。

% は基準値に対するパーセンテージを指定するときに使う。

AH Formatter は以上に加え，Q という単位も使うことができる[11]。1 Q = $\frac{1}{4}$ mm である。

なお，紙面では「12 Q」「8 pt」「0.2 em」のように単位記号の前に適切な空きを入れるのが通常の組版ルールだが，MathML の属性値や CSS のプロパティー値の指定では「12Q」「8pt」「0.2em」のようにスペース無しで記述しなければならない。

MathML 3 仕様書

2.1.5.2 Length Valued Attributes

第 2 章 mn，mi，mo

まず，最も基本的な三つの MathML 要素，mn，mi，mo を紹介しよう。

プレゼンテーションマークアップで使う MathML 要素は**全て**頭に「m」が付いている。math の m だ。

2.1 数値：mn

数値は mn 要素で表す。n は number に由来する[12]。

3.14 `<mn>3.14</mn>`

−1 `<mn>-1</mn>`

2 番目の例で，1 の前に書いた記号 - はマイナス記号ではなくハイフン（U+002D）である[13]。しかし組版結果を見るとまっとうなマイナス記号になっている。これは AH Formatter が気を利かせてハイフンをマイナス記号（U+2212）に変えてくれているのだ。この簡易記法については 13.1 節で再び触れる。

3 桁区切りのカンマを入れたり，指数表記の一種である 2.81E3（2.81×10^3 の意）のようなものも単一の mn 要素で以下のように書いてよい：

---

[11] Q は日本の出版・印刷界では非常によく使われている。元来は**文字サイズ**にのみ用いる単位であって，空き量などの**長さ**を表す H（定義は 1 H = $\frac{1}{4}$ mm）と使い分けなければならなかったが，次第にどちらにも Q が用いられるようになってきている。AH Formatter では H は使えない。

[12] numeral かもしれない。

[13] 紛らわしいが，U+002D の Unicode 名は HYPHEN-MINUS である。タイプライター，テレタイプの世界ではハイフンとマイナスは共用であり，それが ASCII，そして Unicode に受け継がれた。プログラミング言語でもマイナス記号として用いられる。しかし，普通のローマン体フォントでは U+002D はハイフンとしてデザインされており，印刷物でこれをマイナス記号として用いてはならない。

1,800 `<mn>1,800</mn>`

2.81E3 `<mn>2.81E3</mn>`

ローマ数字も当然 mn で表示する。数式中にローマ数字が出てくることは少ないかもしれないが，例えば 2 価，3 価の鉄イオンをそれぞれ Fe^{II}，Fe^{III} のように書くことがある。

しかし，複素数 $3+2i$ のようなものは，単一の mn 要素で記述できない。

MathML 3 仕様書

3.2.4 Number <mn>

2.2 識別名：mi

変数名，定数名，関数名などの識別名は mi 要素で表す。i は identifier に由来する。

x `<mi>x</mi>`

π `<mi>π</mi>`

f `<mi>f</mi>`

mi 要素の中身が 1 文字の欧文アルファベット（ラテン文字，ギリシア文字，キリル文字）のときはイタリック体に，それ以外のときはローマン体になるという規則がある[14]。よって，cos（コサイン）のような関数は何も指定しなくても期待どおりローマン体になる。

cos `<mi>cos</mi>`

しかし，場合によっては 1 文字でもローマン体にしたり，2 文字以上でもイタリック体にしたりしたいことがある。その方法は 11.1 節「mathvariant 属性」で述べる。

MathML 3 仕様書

3.2.3 Identifier <mi>

2.3 演算子：mo

+，− や × などの演算子は mo 要素で表す。o は operator に由来する。

$x-y$
```
<mi>x</mi>
<mo>-</mo>
<mi>y</mi>
```

$-y$
```
<mo>-</mo>
<mi>y</mi>
```

上の二つの例で，マイナス記号と y の間の空きがわずかに違っていることに注意されたい。二項演算子のマイナス記号なのか単項演算子のマイナス記号なのかを自動的に判別して，適切なスペーシングが実現されているのだ。MathML の言葉で言えば，中置演算子か前置演算子かが自動的に決定され，それによって決まるスペースが前後に配される，というわけだ。

MathML 演算子の前置・中置などについては，第 17 章「演算子辞書」で詳述する。

---

[14] mi の中身が 1 文字のとき，どの範囲の文字がイタリックになるべきかは，実は MathML 仕様書では規定されていない。ここで述べたのは AH Formatter のもの。必要ならオプション設定ファイル（8.12.1 節参照）で変更することもできるが，ラテン，ギリシア，キリルに限ったのは妥当だろう。

自動判別がうまくいかない場合もある。これについては第 4 章「グループ化：mrow」で述べる。

なお，ここでも，マイナス記号を表すために <mo>-</mo> のようにただのハイフンを用いたが，正しくマイナス記号として組まれている。

ところで，−1 を組むために，わざわざ

-1

```
<mo>-</mo>
<mn>1</mn>
```

などとする必要は無い（しても間違いではないが）。「−1」全体が一つの数値表記なので，既に見たように

-1

```
<mn>-1</mn>
```

でよい。

演算子には後置するものもある。次に示すのは n の階乗である。

$n!$

```
<mi>n</mi>
<mo>!</mo>
```

mo の o は operator だと言ったが，mo 要素は通常の意味の演算子よりも遙かに広い用途で用いられる[15]。イコール（=），不等号（> など），比例（∝），垂直（⊥）のような関係演算子はもちろん，積分記号（∫），偏微分記号（∂）の類や，( ) [ ] { } などの括弧類も mo で表す。

$(2+3)\times 4 = 20$

```
<mo>(</mo>
<mn>2</mn>
<mo>+</mo>
<mn>3</mn>
<mo>)</mo>
<mo>&times;</mo>
<mn>4</mn>
<mo>=</mo>
<mn>20</mn>
```

その他，x' のプライムや $\dot{x}$ のドット，$\bar{x}$ のバーのようなアクセント類，(x, y) のカンマのような区切り記号，$n \to \infty$ の矢印，といったものはみな MathML でいう演算子である。

演算子は 1 文字とは限らない。次の式は，右辺でもって左辺を定義するという意味だが，:= の 2 文字で一つの演算子となっている。

$$\exp z := \sum_{n=0}^{\infty} \frac{1}{n!} z^n$$

```
<mi>exp</mi>
<mo>&af;</mo>
<mi>z</mi>
<mo>:=</mo>
<munderover>
  <mo>&Sum;</mo>
  <mrow><mi>n</mi><mo>=</mo><mn>0</mn></mrow>
  <mi>&infin;</mi>
</munderover>
<mfrac>
  <mn>1</mn>
  <mrow><mi>n</mi><mo>!</mo></mrow>
</mfrac>
<msup><mi>z</mi><mi>n</mi></msup>
```

MathML 3 仕様書

3.2.5 Operator, Fence, Separator or Accent <mo>

2.4 見えない演算子

MathML には目に見えない演算子がいくつかある。ここでは以下の四つを取り上げる。いずれも Unicode に（特殊な）文字として定義されているので，コード番号と文字実体参照，Unicode 名も挙げておこう。

15) 数学でいう演算子とはもはや別の概念と考えたほうがよさそうだ。

Unicode 名	コード番号	文字実体参照	同（短縮形）
FUNCTION APPLICATION	U+2061	&ApplyFuntion;	⁡
INVISIBLE TIMES	U+2062	⁢	⁢
INVISIBLE SEPARATOR	U+2063	⁣	⁣
INVISIBLE PLUS	U+2064		

本文書では，短縮形の文字実体参照を使うことにする。

2.4.1 関数適用⁡

$\sin\theta$ という式は，変数 θ に関数 sin を適用したもの，と見て以下のようにマークアップする。

$\sin\theta$

```
<mi>sin</mi>
<mo>&af;</mo>
<mi>θ</mi>
```

sin と θ の間に，関数を適用するという演算子 <mo>⁡</mo> が挟まっているのだ。

もし <mo>⁡</mo> を忘れると，

NG $\sin\theta$

```
<mi>sin</mi>
<mi>θ</mi>
```

のように，適切な空きが確保されない。

この例では，⁡ は「見えない演算子」とは言っても，それ自身は見えないが空きを確保するという見える効果をもたらす。

次に，関数 $f(x)$ を考える。これも，変数 x に対して f という関数を適用すると考え，以下のようにマークアップする。

$f(x)$

```
<mi>f</mi>
<mo>&af;</mo>
<mo>(</mo>
<mi>x</mi>
<mo>)</mo>
```

このケースでは，f と (の間に空きは入らない。よって，<mo>⁡</mo> を略しても，以下のように見た目は変わらない。

$f(x)$

```
<mi>f</mi>
<mo>(</mo>
<mi>x</mi>
<mo>)</mo>
```

しかし，このような場合でも ⁡ は略さないのがタテマエとなっている。

プレゼンテーションマークアップは，数式の意味を捨象してカタチを記述するもののはずだ。なぜ，何の視覚的効果も及ぼさないものをわざわざ入れる必要があるのか？

一つには，数式の読み上げに配慮してのことらしい。英語では $f(x)$ を ef of ex と読む。そう解釈できるよう，関数適用ということを明示する。MathML はテキスト読み上げエンジン[16] が数式を読み上げることをも想定しているのだ。

筆者の知識不足により，この問題にはこれ以上立ち入らない。以後も必要に応じてタテマエを記述するに留める。

2.4.2 見えない乗算演算子⁢

$2a$ とか xy といった式には何の演算子も書かれていないが，それぞれ〈2 と a の積〉，〈x と y の積〉という意味で

[16] 視覚障害者はテキスト読み上げエンジンの助けを借りてウェブを閲覧している。ウェブ制作においてテキストの自動読み上げに配慮すべきことは，アクセシビリティー上の常識となっている。

あって，本来書かれるべき乗算の演算子 × が略されている[17]。

演算子 <mo>⁢</mo> はこの見えない乗算の演算子である。既に見たように it は invisible times に由来する。

したがって，$2a$ は以下のようなマークアップになる。

$2a$

```
<mn>2</mn>
<mo>&it;</mo>
<mi>a</mi>
```

省略しても組版結果は変わらないが，これを入れるのがタテマエである。

2.4.3 見えないカンマ⁣

行列の成分を表すのに a_{23} というような表記が使われる。この添字に注目してほしい。「23」はニジュウサンではない。第 2 行第 3 列の成分であることを示している。添字を変数にして a_{nm} と書いた場合の nm も n と m の積ではない。

この「2」「3」や「n」「m」の間には区切り記号が略されていると考え，ここには見えないカンマを置く。したがって，a_{23} の「23」の部分のマークアップは次のようになる。

23

```
<mn>2</mn><mo>&ic;</mo><mn>3</mn>
```

なお，第 i 行第 $j+1$ 列の成分といった場合は，$a_{i\,j+1}$ のように空きを入れるか，$a_{i,j+1}$ のようにカンマを入れる。

2.4.4 見えない加算演算子 U+2064

U+2064 INVISIBLE PLUS は ⁢ の加算版である。

小学校の算数では帯分数（たいぶんすう）というものがある。$2\frac{1}{3}$ などと書き「ニ**カ**サンブンノイチ」とか「ニ**ト**サンブンノイチ」などと読む[18]。表す値は $2+\frac{1}{3}$ と同じだが，整数部と 1 未満の分数をくっ付けて一つの数として見せるもので，量的に把握しやすいという利点がある。中学数学になると積の × を略す記法が出てくるため，$2\times\frac{1}{3}$ と紛らわしく，帯分数は基本的に使わない。

帯分数 $2\frac{1}{3}$ の 2 と $\frac{1}{3}$ の間には見えない加算記号があると考えるべきである。

よって，以下のようなマークアップになる（分数を表す mfrac については第 6 章「分数：mfrac」で述べる）。

$2\frac{1}{3}$

```
<mn>2</mn>
<mo>&#x2064;</mo>
<mfrac>
  <mn>1</mn>
  <mn>3</mn>
</mfrac>
```

MathML 3 仕様書

3.2.5.5 Invisible operators

第 3 章　テキスト：mtext

数学記号ではないテキストには mtext を用いる。

下付きは，詳しくは第 8 章「添字類」で説明するが，I の最大値 $I_{\max}$ は msub 要素を用いて以下のように書ける。

---

17) 中学一年でこのルールを教わったときの衝撃を覚えているだろうか！

18) 習った時代による。

```
<msub>
  <mi>I</mi>
  <mtext>max</mtext>
</msub>
```

MathML 3 仕様書

3.2.6 Text <mtext>

第 4 章　グループ化：mrow

mrow 要素は部分式のグループ化のためにある。

例えば，e^{-x} という式を考えよう。上付きを実現するための msup 要素は引数を二つ取るのだった。一つ目の引数は本体，二つ目の引数は添字部分である。しかしながら，添字部分は <mo>-</mo><mi>x</mi> のように二つの要素の並びで表現される。これをそのまま msup 要素に入れるわけにはいかない。そこで，部分式 $-x$ を mrow 要素の中に入れ，以下のようにマークアップする。

```
<msup>
  <mi>e</mi>
  <mrow>
    <mo>-</mo>
    <mi>x</mi>
  </mrow>
</msup>
```

4.1　推定された mrow

平方根については第 7 章「累乗根：msqrt，mroot」で説明するが，msqrt 要素を用いた以下の例を見ていただきたい。

```
<msqrt>
  <mo>-</mo>
  <mi>x</mi>
</msqrt>
```

msqrt は引数を根号で包む要素だが，MathML の処理系はこれを

```
<msqrt>
  <mrow>
    <mo>-</mo>
    <mi>x</mi>
  </mrow>
</msqrt>
```

と解釈することになっている。msqrt は引数を一つしか取らないが，子要素が複数あっても mrow で包まれることで一つの引数になるというわけだ。

このようにして補われた mrow を「推定された mrow」（inferred mrow）と呼ぶ。

実は，<math><mi>x</mi></math> のようなマークアップも，<math><mrow><mi>x</mi></mrow></math> の <mrow>/</mrow> タグを略しているのだ。

詳しくは第 17 章「演算子辞書」で述べるが，演算子の前後の空きを決定するプロセスには，その演算子の mrow 要素の中での位置が大きく関わっているので，推定された mrow の理解が必要となる。

4.2　部分式と括弧類の大きさ

パーレン ( ) のような括弧類は，中身に応じて大きさが変わるべきである。

まず，次の式を見てみよう。

$(x+1)$

```
<mo>(</mo>
<mi>x</mi>
<mo>+</mo>
<mn>1</mn>
<mo>)</mo>
```

とくに面白みは無い。では次の式はどうか。

$\left(x+\frac{1}{2}\right)$

```
<mo>(</mo>
<mi>x</mi>
<mo>+</mo>
<mfrac>
  <mn>1</mn>
  <mn>2</mn>
</mfrac>
<mo>)</mo>
```

何の細工も無しにパーレンが大きくなってくれた。しかし，実はパーレンのウチとソトが区別できているわけではなく，単にパーレンを含む部分式（部分といっても，いまの場合は式全体）の中身で天地（垂直方向の大きさ）の一番大きいものに合わせて大きくなっただけなのだ。

次のマークアップが失敗していることは明らかだろう。

NG

$\left(x+1\right)\left(x+\frac{1}{2}\right)$

```
<mo>(</mo>
<mi>x</mi><mo>+</mo><mn>1</mn>
<mo>)</mo>
<mo>&it;</mo>
<mo>(</mo>
<mi>x</mi>
<mo>+</mo>
<mfrac><mn>1</mn><mn>2</mn></mfrac>
<mo>)</mo>
```

括弧類を適切な大きさにするためには，以下のように mrow を使う。

$(x+1)\left(x+\frac{1}{2}\right)$

```
<mrow>
  <mo>(</mo>
  <mi>x</mi><mo>+</mo><mn>1</mn>
  <mo>)</mo>
</mrow>
<mo>&it;</mo>
<mrow>
  <mo>(</mo>
  <mi>x</mi>
  <mo>+</mo>
  <mfrac><mn>1</mn><mn>2</mn></mfrac>
  <mo>)</mo>
</mrow>
```

マークアップの意味も明瞭になった。

MathML 3 仕様書

3.3.1 Horizontally Group Sub-Expressions <mrow>

4.3　適切なスペーシングのために

4.3.1　絶対値の左右の空き

mrow に入れないとマズいケースをもう一つ挙げる。

以下の組版例をよく見てみよう。

NG

$|n|+|m|$

```
<mo>|</mo>
<mi>n</mi>
<mo>|</mo>
<mo>+</mo>
<mo>|</mo>
<mi>m</mi>
```

```
<mo>|</mo>
```

n，m がイタリックなのでちょっと分かりづらいが，n は左よりも右が空いており，m は逆に右よりも左が空いている。左右の空きが不均等なのだ。

これらの原因は，後置演算子であるべき n の右側の | や，前置演算子であるべき m の左側の | が，いずれも中置演算子とみなされてしまったためである。

正しくは以下のように絶対値の部分を mrow で囲まなくてはならない。

$|n| + |m|$

```
<mrow>
  <mo>|</mo>
  <mi>n</mi>
  <mo>|</mo>
</mrow>
<mo>+</mo>
<mrow>
  <mo>|</mo>
  <mi>m</mi>
  <mo>|</mo>
</mrow>
```

4.3.2 単項演算子の左右の空き

さらに例を挙げる。+ や – などの符号は，単項演算子として用いられる場合と二項演算子として用いられる場合で左右の空きが違うことは既に述べた。以下のマイナス記号は二つとも単項演算子なのだが，パーレン内のほうは二項演算子のように前後に不自然な空きができてしまった。

NG
$-(-x)$

```
<mo>-</mo>
<mo>(</mo>
<mo>-</mo>
<mi>x</mi>
<mo>)</mo>
```

$-x$ の部分を mrow に入れてやることで適切な空きとなる：

$-(-x)$

```
<mo>-</mo>
<mo>(</mo>
<mrow>
  <mo>-</mo>
  <mi>x</mi>
</mrow>
<mo>)</mo>
```

単項演算子か二項演算子かを決定するメカニズムについては第 17 章「演算子辞書」で説明する。

第 5 章　区切って囲む：mfenced

第 4 章では，括弧類で囲まれた部分式を mrow でグループ化してやらないと適切なスペーシングが得られない例を挙げた。

括弧類で囲むために mfenced が使える（fenced はまさに「囲われた」の意）。mfenced 要素は，引数が一つの場合，それを括弧類（既定はパーレン）で囲む。

(x)

```
<mfenced>
  <mi>x</mi>
</mfenced>
```

5.1 囲み記号を変える

mfenced の囲み記号は open，close 属性で自由に変えることができる。

$[x]$

```
<mfenced open="[" close="]">
  <mi>x</mi>
</mfenced>
```

$\langle x \rangle$

```
<mfenced open="&lang;" close="&rang;">
  <mi>x</mi>
</mfenced>
```

この例のように，open，close 属性には文字実体参照を書くことができる。

よく使われる囲み記号の一覧は 13.4 節「囲み記号」を参照のこと。

5.2 座標・集合

座標値 (x, y) は以下のようにして組むこともできる。

(x, y)

```
<mo>(</mo>
<mi>x</mi>
<mo>,</mo>
<mi>y</mi>
<mo>)</mo>
```

しかし，mfenced 要素を使ったほうが（階層は深くなるが）スマートだろう。以下のようにする。

(x, y)

```
<mfenced>
  <mi>x</mi>
  <mi>y</mi>
</mfenced>
```

不思議だ。カンマはどこから湧いてきたのか？

実は mfenced 要素は，引数が二つ以上ある場合，それらの間に区切り記号をはさみ，全体を囲み記号で囲む。既定の区切り記号はカンマだ。

区切りや囲みを変えるには以下のようにする。

次に示すのは集合の外延表記の例。

$\{2, 3, 5\}$

```
<mfenced open="{" close="}">
  <mn>2</mn>
  <mn>3</mn>
  <mn>5</mn>
</mfenced>
```

〈0 以上 1 未満〉は次のようになる。つまり，変更したい側の囲み記号のみをあらわに指定すればよい。

$[0, 1)$

```
<mfenced open="[">
  <mn>0</mn>
  <mn>1</mn>
</mfenced>
```

次の三つは量子力学に出てくるブラケット表記の例である。

$|\alpha\rangle$

```
<mfenced open="|" close="&rang;">
  <mi>α</mi>
</mfenced>
```

$\langle \psi | \varphi \rangle$

```
<mfenced open="&lang;" close="&rang;" separators="|">
  <mi>ψ</mi>
  <mi>ϕ</mi>
</mfenced>
```

$\langle \psi | H | \varphi \rangle$

```
<mfenced open="&lang;" close="&rang;" separators="|">
  <mi>ψ</mi>
  <mi>H</mi>
  <mi>ϕ</mi>
</mfenced>
```

区切り記号の属性名が separators と複数形になっていることでも分かるとおり，区切り記号は複数指定することもできる。それらは出現順に使われる。例えば以下のように。

$f(x, y; t)$

```
<mi>f</mi>
<mo>&af;</mo>
<mfenced separators=", ;">
  <mi>x</mi>
  <mi>y</mi>
  <mi>t</mi>
</mfenced>
```

この例では，見やすさのためカンマとセミコロンの間にスペースを入れているが，スペースは無くても構わない。

指定した区切り記号の数より区切り箇所のほうが多い場合は，次の例のように最後の区切り記号が繰り返される。

$f(t; x, y, z)$

```
<mi>f</mi>
<mo>&af;</mo>
<mfenced separators="; ,">
  <mi>t</mi>
  <mi>x</mi>
  <mi>y</mi>
  <mi>z</mi>
</mfenced>
```

$x - y$ の絶対値を組む際，`<mi>x</mi><mo>-</mo><mi>y</mi>` をそのまま mfenced の中に入れたのでは，次のように余計なカンマが入ってしまう。

NG

$|x, -, y|$

```
<mfenced open="|" close="|">
  <mi>x</mi><mo>-</mo><mi>y</mi>
</mfenced>
```

mfenced は引数ごとに区切り記号を入れるからだ。正しくは次のように mrow に入れる。

$|x - y|$

```
<mfenced open="|" close="|">
  <mrow>
    <mi>x</mi><mo>-</mo><mi>y</mi>
  </mrow>
</mfenced>
```

区切り記号を空にすることでカンマを消すこともできるが，これは正しいやり方ではない。

NG

$|x - y|$

```
<mfenced open="|" close="|" separators="">
  <mi>x</mi><mo>-</mo><mi>y</mi>
</mfenced>
```

MathML 3 仕様書

3.3.8 Expression Inside Pair of Fences <mfenced>

5.3 数列

数列のように多数の要素が区切り記号を挟んで連なっているものを表すのに，囲み記号を無くした mfenced が使える。繰り返しを表す省略記号として，ピリオドの高さに点が並ぶ …（…; U+2026 HORIZONTAL ELLIPSIS）を用いる。この省略記号は項として扱われるべきなので，mo でなく mi を使う（MathML 仕様書 3.2.3.3）。

$a_1, a_2, \ldots, a_n, \ldots$

```
<mfenced open="" close="">
  <msub><mi>a</mi><mn>1</mn></msub>
  <msub><mi>a</mi><mn>2</mn></msub>
  <mi>&hellip;</mi>
  <msub><mi>a</mi><mi>n</mi></msub>
  <mi>&hellip;</mi>
</mfenced>
```

この例を見ると和や積にも応用できそうに思えるが，MathML 仕様書は以下のように加算演算子などを区切り記号とするマークアップを「正しくないだろう」（would be incorrect）としている。

NG

$a_1 + a_2 + \cdots + a_n$

```
<mfenced open="" close="" separators="+">
  <msub><mi>a</mi><mn>1</mn></msub>
  <msub><mi>a</mi><mn>2</mn></msub>
  <mi>&ctdot;</mi>
  <msub><mi>a</mi><mi>n</mi></msub>
</mfenced>
```

NG
$a_1a_2\cdots a_n$

```
<mfenced open="" close="" separators="&it;">
  <msub><mi>a</mi><mn>1</mn></msub>
  <msub><mi>a</mi><mn>2</mn></msub>
  <mi>&ctdot;</mi>
  <msub><mi>a</mi><mi>n</mi></msub>
</mfenced>
```

その理由は以下の二つが等価でないことによる。

$x+y$

```
<mrow>
  <mi>x</mi>
  <mo>+</mo>
  <mi>y</mi>
</mrow>
```

$x+y$

```
<mfenced open="" close="" separators="+">
  <mi>x</mi>
  <mi>y</mi>
</mfenced>
```

後者は以下と等価である。

$x+y$

```
<mrow>
  <mi>x</mi>
  <mo separator="true">+</mo>
  <mi>y</mi>
</mrow>
```

つまり，mfenced による区切り記号の演算子では separators 属性が true となっているのだ。

MathML 仕様書は，「したがって正しく表示されないかもしれない」（might therefore render inappropriately）としているが，具体的にどのような不具合があり得るかは述べていない。AH Formatter による上記の組版結果を見ると，とくに変わりは無いように見える。

mfenced の区切り記号も中置演算子としては扱われる。したがって，第 17 章「演算子辞書」で述べるような演算子辞書によるスペーシングの制御は可能だ。

ともかく，MathML 仕様書に基づいた適切なマークアップは以下のようになる。

$a_1+a_2+\cdots+a_n$

```
<msub><mi>a</mi><mn>1</mn></msub>
<mo>+</mo>
<msub><mi>a</mi><mn>2</mn></msub>
<mo>+</mo>
<mi>&ctdot;</mi>
<mo>+</mo>
<msub><mi>a</mi><mi>n</mi></msub>
```

$a_1a_2\cdots a_n$

```
<msub><mi>a</mi><mn>1</mn></msub>
<mo>&it;</mo>
<msub><mi>a</mi><mn>2</mn></msub>
<mo>&it;</mo>
<mi>&ctdot;</mi>
<mo>&it;</mo>
<msub><mi>a</mi><mi>n</mi></msub>
```

mfenced の話題から外れるが，ここで ⋯ について述べておこう。

カンマに挟まれた省略記号には，ピリオドの高さに点が並ぶ …（…; U+2026 HORIZONTAL ELLIPSIS）を用い，二項演算子や変数などに挟まれた省略記号には，中心線のあたりに点が並ぶ ⋯（⋯; U+22EF MIDLINE HORIZONTAL ELLIPSIS）を用いるのがいいだろう。

なお，U+2026 がピリオドの高さに点が並ぶのは欧文中・数式中の話である。和文中では「……。」のようにいわゆる〈三点リーダー〉となる。つまり，Unicode では欧文ものの ellipsis と和文約物の三点リーダーを区別しておらず，AH Formatter が言語情報などから自動的に判定してグリフを切り替える。

一方，U+22EF は中心に点が並ぶ。しかしこちらは数学記号なのでこれを和文の三点リーダーとして用いることは適当でない。

第 6 章　分数：mfrac

分数（fraction）には mfrac 要素を使う。

$\boxed{\frac{1}{x}}$

```
<mfrac>
  <mn>1</mn>
  <mi>x</mi>
</mfrac>
```

第一引数が分子，第二引数が分母となる。分子や分母が単一の要素で表せないときは，次のように mrow 要素に入れる。

$\boxed{\frac{1}{x+1}}$

```
<mfrac>
  <mn>1</mn>
  <mrow>
    <mi>x</mi>
    <mo>+</mo>
    <mn>1</mn>
  </mrow>
</mfrac>
```

上の二つの例はディスプレイスタイルで組まれているが，インラインスタイルの場合，$\frac{1}{x}$ のように，分子・分母の文字サイズが一段階下がる[19]。

ディスプレイスタイルの mfrac では，分子・分母がインラインスタイルになる。

$\boxed{\frac{1}{\frac{1}{2}\sum_{k=1}^{n} k}}$

```
<mfrac>
  <mn>1</mn>
  <mrow>
    <mfrac><mn>1</mn><mn>2</mn></mfrac>
    <mo>&it;</mo>
    <munderover>
      <mo>&Sum;</mo>
      <mrow><mi>k</mi><mo>=</mo><mn>1</mn></mrow>
      <mi>n</mi>
    </munderover>
    <mi>k</mi>
  </mrow>
</mfrac>
```

上の例では，この数式の全体はディスプレイスタイルだが，分子・分母はインラインスタイルとなっている。$\frac{1}{2}$ の数字が小さいのは，インラインスタイルで分数の分子・分母が小さくなるルールに従ったものであり，総和記号が

$$\sum_{k=1}^{n}$$

でなく

$$\sum_{k=1}^{n}$$

となったのはそれがインラインスタイルだからである。

bevelled 属性を true にすれば分数線が水平ではなく斜線になり，分子が左上，分母が右下に配置される。

$\boxed{a/b}$

```
<mfrac bevelled="true">
  <mi>a</mi>
  <mi>b</mi>
</mfrac>
```

bevelled の場合も，ディスプレイスタイルでは分子・分母はインラインスタイルになるが文字サイズは変わらず，インラインスタイルでは分子・分母が a/b のように小さくなる。

次のような場合，分数には mfrac 要素でなく <mo>/</mo> を使うのがいいだろう。

$\boxed{\sqrt{1+x^{1/2}}}$

```
<msqrt>
  <mn>1</mn>
  <mo>+</mo>
  <msup>
```

[19] これには「スクリプトレベル」というものが関わっている。インラインスタイルの mfrac の分子・分母はスクリプトレベルが 1 上がる。スクリプトレベルについては 8.12 節「添字のサイズ」を参照。

```
      <mi>x</mi>
      <mrow>
        <mn>1</mn>
        <mo>/</mo>
        <mn>2</mn>
      </mrow>
    </msup>
  </msqrt>
```

MathML 3 仕様書

3.3.2 Fractions <mfrac>

第 7 章　累乗根：msqrt，mroot

平方根には msqrt 要素を使う。sqrt は square root より。

$\sqrt{2}$

```
<msqrt>
  <mn>2</mn>
</msqrt>
```

$\sqrt{\frac{1}{2}}$

```
<msqrt>
  <mfrac>
    <mn>1</mn>
    <mn>2</mn>
  </mfrac>
</msqrt>
```

一般の累乗根には mroot 要素を使う。第一引数が中身，第二引数が指数（左肩に付くもの）となる。

$\sqrt[3]{7}$

```
<mroot>
  <mn>7</mn>
  <mn>3</mn>
</mroot>
```

入れ子にすると，こうなる：

$\sqrt{\sqrt{16}}$

```
<msqrt>
  <msqrt>
    <mn>16</mn>
  </msqrt>
</msqrt>
```

MathML 3 仕様書

3.3.3 Radicals <msqrt>, <mroot>

第 8 章　添字類

この章で扱う添字類は，右上に添える「上付き」（superscript），右下に添える「下付き」（subscript）に留まらない，幅広い概念だ。添える位置は真上や真下もあるし，$^{4}\mathrm{He}$ のように本体の左側もある。

なお，日本語では「添字」と呼ぶが，もちろん単一の字だけでなく任意の式を添えることができる。

ここで英語の「script」という多義的な単語もしくは形態素[20]について触れておこう。本文書には，①sub-，sup-などの接頭辞を付けて，何らかの意味での添字を意味する語を形成する形態素，②それら添字類の総称，③手書き

[20] 形態素（morpheme）とは単語を構成する部品であって，意味を持つ最小の単位だ。カザムキ（風向き）という単語はカゼという形態素とムキという形態素からなる（このように音が少し変わることもある）。また，カゼ，ムキはそれぞれ一つの形態素からなる単語でもある。

系書体の総称（スクリプト体），の三つだけが出てくる。このほか，文字学用語としては④italic script（イタリック体）におけるような「書体」，⑤Arabic script（アラビア文字）におけるような「文字体系（用字系ともいう）」の意もある。⑥台本，⑦手稿・原稿の意もある。いずれも「書かれたもの」という中心的語義から来ている。

なお，②の語義は普通の辞書には見当たらないが，MathML 仕様書で scripts とあればほぼこの意味である。①から作られた新しい語義かもしれない。

8.1 上付き・下付き：msup，msub

累乗など右肩に付けるのは msup 要素を使う。sup は superscript に由来する。第一引数が本体，第二引数が肩に付けるものとなる。

```
<msup>
  <mi>x</mi>
  <mn>2</mn>
</msup>
```

同様に，下付きは msub 要素を使う。sub は subscript に由来する。

```
<msub>
  <mi>a</mi>
  <mi>n</mi>
</msub>
```

上付きや下付きのものが単一の要素で表せないときは mrow 要素に入れる必要がある。

```
<msup>
  <mi>e</mi>
  <mrow>
    <mo>-</mo>
    <mi>θ</mi>
  </mrow>
</msup>
```

ここまでの例では，上付きや下付きにするのはそれ自体が式であったが，式ではない記号が付くこともある。例えば，行列 A の転置行列（transpose）は次のように書く。

```
<msup>
  <mi>A</mi>
  <mo>T</mo>
</msup>
```

なお，転置行列の表記には，${}^{\mathrm{t}}A$ などいくつかのバリエーションがある[21]。

同じような例だが，線型変換や行列のエルミート共役(きょうやく)を以下のように表記する。

```
<msup>
  <mi>A</mi>
  <mo>&dagger;</mo>
</msup>
```

MathML 3 仕様書

3.4.1 Subscript <msub>
3.4.2 Superscript <msup>

8.2 上付きおよび下付き：msubsup

定積分の上端・下端も上付き・下付きとして扱う。

下付きと上付きを同時に付けるため，msubsup という要素を使う。

第一引数が本体，第二引数が下端，第三引数が上端である。

[21] 添字が本体より前に付く場合の書き方は 8.8 節「テンソルと前置き添字」で説明する。

以下のように書く[22]。

$$\int_0^1 xdx$$

```
<msubsup>
  <mo>&int;</mo>
  <mn>0</mn>
  <mn>1</mn>
</msubsup>
<mi>x</mi>
<mi>d</mi>
<mi>x</mi>
```

関数の境界値での値の差を表す $[\ \ ]_a^b$ も次のように書ける。

$$\int_0^\pi \sin x dx = [-\cos x]_0^\pi$$

```
<msubsup>
  <mo>&int;</mo>
  <mn>0</mn>
  <mi>π</mi>
</msubsup>
<mi>sin</mi><mo>&af;</mo><mi>x</mi>
<mi>d</mi><mi>x</mi>
<mo>=</mo>
<msubsup>
  <mfenced open="[" close="]">
    <mrow>
      <mo>-</mo><mi>cos</mi><mo>&af;</mo><mi>x</mi>
    </mrow>
  </mfenced>
  <mn>0</mn>
  <mi>π</mi>
</msubsup>
```

なお，積分領域が上端・下端でなく単一の記号で与えられるときは当然 msub を使う。

$$\oint_C fds$$

```
<msub>
  <mo>&conint;</mo>
  <mi>C</mi>
</msub>
<mi>f</mi>
<mi>d</mi>
<mi>s</mi>
```

これは周回積分の例で，閉曲線 C に沿って一周積分することを表している。

MathML 3 仕様書

3.4.3 Subscript-superscript Pair <msubsup>

8.3 化学式

化学式の例をいくつか見てみる。

CO_2 は以下のどちらでマークアップしても同じ結果になる。

CO_2

```
<mi mathvariant="normal">C</mi>
<msub>
  <mi mathvariant="normal">O</mi>
  <mn>2</mn>
</msub>
```

NG
CO_2

```
<msub>
  <mi>CO</mi>
  <mn>2</mn>
</msub>
```

後者のほうが明らかに楽ではある。しかし C と O は別の元素を表しており，一つの識別子としてマークアップするのは適当でないし，添字 2 は O（酸素原子）の個数を表しているので O の添字とすべきだろう。

なお，1 文字の元素記号を

[22] この式に出てくる，いわゆる「微分の d」について，どのようなマークアップが適切かはここでは立ち入らない。17.8.2 節「微分の d」参照のこと。

C

```
<mi mathvariant="normal">C</mi>
```

などと書くのはいかにも面倒ではあるが，mtext 要素で書くのは適切でないだろう。

もう少し複雑な例として，硫酸イオン SO_4^{2-} を取り上げよう。以下のように書ける。

SO_4^{2-}

```
<msup>
  <mrow>
    <mi mathvariant="normal">S</mi>
    <msub>
      <mi mathvariant="normal">O</mi>
      <mn>4</mn>
    </msub>
  </mrow>
  <mrow>
    <mn>2</mn>
    <mi>-</mi>
  </mrow>
</msup>
```

ここで，マイナス記号が mo 要素でなく mi 要素で表されている点に注目していただきたい。このマイナス記号は，−1 のような単項演算子でも，1 − 2 のような二項演算子でもなく，陰イオンであることを表す印である。前に 2 が付いているのは，2 価であることを表しているのであって，2 から何かを引こうとしているわけではない。

もし，このマイナス記号を mo 要素で表すと，

NG

SO_4^{2-}

```
<msup>
  <mrow>
    <mi mathvariant="normal">S</mi>
    <msub>
      <mi mathvariant="normal">O</mi>
      <mn>4</mn>
    </msub>
  </mrow>
  <mrow>
    <mn>2</mn>
    <mo>-</mo>
  </mrow>
</msup>
```

のように 2 とマイナス記号の間に余計な空きができてしまう。

ところで，硫酸イオンは SO_4^{2-} のように，2− と 4 が上下に並ぶ表現もある[23]。これは msubsup 要素を用いて書ける。しかし，以下のどちらのマークアップも，化学式の意味とはズレており，素直とは言いがたい。

SO_4^{2-}

```
<mi mathvariant="normal">S</mi>
<msubsup>
  <mi mathvariant="normal">O</mi>
  <mn>4</mn>
  <mrow>
    <mn>2</mn>
    <mi>-</mi>
  </mrow>
</msubsup>
```

SO_4^{2-}

```
<msubsup>
  <mi>SO</mi>
  <mn>4</mn>
  <mrow>
    <mn>2</mn>
    <mi>-</mi>
  </mrow>
</msubsup>
```

この問題のすっきりした解決法はおそらく無いだろう。そもそも MathML は化学式など想定していないのかもしれない。

[23] どちらがより妥当なのか，普通なのかは筆者には分らない。

8.4　上下に付く添字：mover，munder，munderover

添字には上付き（superscript），下付き（subscript）のほかに，overscript，underscript というものもある。定着した訳語は見当たらなかったので，本文書では外来語として「オーバースクリプト」「アンダースクリプト」と呼ぶことにする。

これらは，mover，munder，munderover 要素を使う。使い方は msup，msub，msubsup と同じだ。

例えば，定積分の下端・上端を，下付き・上付きの位置ではなくアンダースクリプト，オーバースクリプトの位置に置く流儀があるが，以下のようにマークアップする。msubsup を munderover に変えただけだ。

$$\int_{a}^{b}$$

```
<munderover>
  <mo>&int;</mo>
  <mi>a</mi>
  <mi>b</mi>
</munderover>
```

極限には munder 要素を使えばよい。

$$\lim_{n \to \infty} f(x) = 0$$

```
<munder>
  <mo>lim</mo>
  <mrow>
    <mi>n</mi><mo>&rarr;</mo><mi>&infin;</mi>
  </mrow>
</munder>
<mi>f</mi>
<mo>&af;</mo>
<mfenced><mi>x</mi></mfenced>
<mo>=</mo>
<mn>0</mn>
```

MathML 3 仕様書

3.4.4 Underscript <munder>
3.4.5 Overscript <mover>
3.4.6 Underscript-overscript Pair <munderover>

8.5　総和・総積

総和記号 $\sum$ には ∑ を使えばよい。項のインデックスの範囲を表す式を上下に付けるために munderover 要素に入れる。第一引数が本体，第二引数がアンダースクリプト，第三引数がオーバースクリプトである。

$$\sum_{k=0}^{\infty} a_k$$

```
<munderover>
  <mo>&Sum;</mo>
  <mrow>
    <mi>k</mi><mo>=</mo><mn>0</mn>
  </mrow>
  <mi>&infin;</mi>
</munderover>
<msub>
  <mi>a</mi>
  <mi>k</mi>
</msub>
```

同様に，総積記号 $\prod$ には ∏ を使う。

オーバースクリプトが無いときは munder 要素でよい。次に掲げるのは，M の全ての元 k にわたる，集合 S_k の和集合（合併集合とも）である。

$$\bigcup_{k \in M} S_k$$

```
<munder>
  <mo>&bigcup;</mo>
  <mrow>
    <mi>k</mi>
    <mo>&isin;</mo>
    <mi>M</mi>
  </mrow>
</munder>
<msub>
  <mi>S</mi>
  <mi>k</mi>
```

```
</msub>
```

この手の記号をいくつか挙げておく。

組版例	入力	UCS4	参考		
			記号	入力	UCS4
$\sum_{i=0}^{n} X_i$	`&Sum;`	U+2211			
$\prod_{i=0}^{n} X_i$	`&Product;`	U+220F			
$\bigwedge_{i=0}^{n} X_i$	`&bigwedge;`	U+22C0	∧	`&wedge;`	U+2227
$\bigvee_{i=0}^{n} X_i$	`&bigvee;`	U+22C1	∨	`&vee;`	U+2228
$\bigcap_{i=0}^{n} X_i$	`&bigcap;`	U+22C2	∩	`&cap;`	U+2229
$\bigcup_{i=0}^{n} X_i$	`&bigcup;`	U+22C3	∪	`&cup;`	U+222A
$\bigodot_{i=0}^{n} X_i$	`&bigodot;`	U+2A00	⊙	`&odot;`	U+2299
$\bigoplus_{i=0}^{n} X_i$	`&bigoplus;`	U+2A01	⊕	`&oplus;`	U+2295
$\bigotimes_{i=0}^{n} X_i$	`&bigotimes;`	U+2A02	⊗	`&otimes;`	U+2297

和・積を除けば，二項演算子の big 版を使うわけだ。

8.6 上線・下線

式の上や下に水平の線を引くことがある。とりわけ上線は論理否定，平均，複素共役（きょうやく）など，幅広い目的で使われている[24]。

例えば，論理式 P の否定を次のように書く[25]。

$\boxed{\overline{P}}$

```
<mover>
  <mi>P</mi>
  <mo>_</mo>
</mover>
```

見てのとおり，上線を引くための MathML 要素があるわけではなく，線の演算子を `mover` で上に載せるのである。線の長さは本体に合わせて自動的に伸びてくれるが，このような性質を持った演算子は stretchy であるという[26]。

線の演算子 `<mo>_</mo>` の _ は ASCII に含まれるアンダースコア（U+005F；アンダーバーとも）であり，下線の演算子として使うものだ。Unicode には上線に使うための U+203E OVERLINE という文字（‾）がある（文字実体参照は `&oline;` および `&OverBar;`）のだが，上線にも下線にもアンダースコアを使うことをお勧めしたい。

まず，アンダースコアを上線に使っ**てもよい**理由だが，もともとアンダースコアが下方に位置するようデザインされていても，`mover` で式の上に載せたときは AH Formatter がそのグリフを適当な位置に配置してくれることだ。つまり，オーバーラインを使おうがアンダースコアを使おうが，線の見た目の位置は変わらない。

---

[24] 言語学でも $\bar{X}$（エックスバー）理論なんてのがある。

[25] P の否定には $\neg P$ という書き方もある。

[26] 括弧類もまた中身に合わせて大きくなるので stretchy だ。

次に，アンダースコアを上線にも使った**ほうがよい**理由は，デザインの統一が取れることである。本文書で用いている STIX フォントファミリー（16.1 節参照）のようにオーバーラインとアンダースコアが同じデザイン（位置だけが異なる）であれば気にしなくてよいが，フォントによっては線の太さや端部の形状が異なることがあり，上線と下線の統一感が無くなる恐れがある。そして使う記号を同じにするのであれば，ASCII に含まれ，覚えやすく，キーボードで容易に入力できるアンダースコアを選んだほうがよい，というわけだ。

ド＝モルガンの法則は以下のように書ける。

$\overline{P \wedge Q} = \bar{P} \vee \bar{Q}$

```
<mover>
  <mrow>
    <mi>P</mi><mo>&and;</mo><mi>Q</mi>
  </mrow>
  <mo>_</mo>
</mover>
<mo>=</mo>
<mover>
  <mi>P</mi>
  <mo>_</mo>
</mover>
<mo>&or;</mo>
<mover>
  <mi>Q</mi>
  <mo>_</mo>
</mover>
```

上線は複素共役を表すのにも用いられる。例えば z の複素共役は以下のように書く[27]。

$\overline{z}$

```
<mover>
  <mi>z</mi>
  <mo>_</mo>
</mover>
```

複素共役の複素共役は上線を $\overline{\overline{z}}$ のように重ねることになる[28]。しかし，単純に $\overline{z}$ に上線を引くやり方だと線の長さが揃わない。分かりやすいよう，z よりも幅の広い w で示すと，次のようになる。

NG $\overline{\overline{w}}$

```
<mover>
  <mover>
    <mi>w</mi>
    <mo>_</mo>
  </mover>
  <mo>_</mo>
</mover>
```

一本目（下側）の線は w がイタリック体であることを考慮して長さと水平位置が決定されるが，二本目（上側）の線は本体 $\overline{w}$ の幅全体にかかるように引かれるからである。

理屈はそうなのだが格好は良くない。伝統的には同じ長さの線を載せて組まれてきたはずだ。

次のように〈線を載せた線〉を載せるやり方なら長さが揃う。

$\overline{\overline{w}}$

```
<mover>
  <mi>w</mi>
  <mover>
    <mo>_</mo>
    <mo>_</mo>
  </mover>
</mover>
```

8.7　上極限・下極限

上極限（limit superior）および下極限（limit inferior）という概念がある[29]。

---

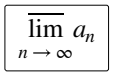
[27] z の複素共役には z^* という書き方もある。

[28] 複素共役の複素共役は元の数に等しいわけだが，二重線を載せたこの記号は，複素共役のこの基本的な性質を表した恒等式 $\overline{\overline{z}} = z$ に出てくる。

[29] 定義については述べないが，両者が一致したとき，単に極限という。

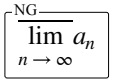
それぞれに二通りの表記があるが，まず水平線を用いた表記について述べる。

水平線を用いた上極限は以下のようになる。



```
<munder>
  <mover>
    <mi>lim</mi>
    <mo>_</mo>
  </mover>
  <mrow><mi>n</mi><mo>&rarr;</mo><mi>&infin;</mi></mrow>
</munder>
<msub>
  <mi>a</mi>
  <mi>n</mi>
</msub>
```

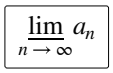
アンダースクリプトとオーバースクリプトだからといって，munderover では，以下のようにうまくいかない。



```
<munderover>
  <mi>lim</mi>
  <mrow><mi>n</mi><mo>&rarr;</mo><mi>&infin;</mi></mrow>
  <mo>_</mo>
</munderover>
<msub>
  <mi>a</mi>
  <mi>n</mi>
</msub>
```

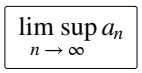
このように，上線の長さが $n \to \infty$ に揃ってしまい，適切な長さにならない。

下極限も同様で，以下のようにする。

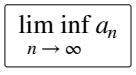


```
<munder>
  <munder>
    <mi>lim</mi>
    <mo>_</mo>
  </munder>
  <mrow><mi>n</mi><mo>&rarr;</mo><mi>&infin;</mi></mrow>
</munder>
<msub>
  <mi>a</mi>
  <mi>n</mi>
</msub>
```

上線・下線というテーマからは外れてしまうが，上極限・下極限のもう一つの表記である lim sup，lim inf を用いたものも掲げておこう。



```
<munder>
  <mi>lim sup</mi>
  <mrow><mi>n</mi><mo>&rarr;</mo><mi>&infin;</mi></mrow>
</munder>
<mspace width="thinmathspace" />
<msub>
  <mi>a</mi>
  <mi>n</mi>
</msub>
```



```
<munder>
  <mi>lim inf</mi>
  <mrow><mi>n</mi><mo>&rarr;</mo><mi>&infin;</mi></mrow>
</munder>
<mspace width="thinmathspace" />
<msub>
  <mi>a</mi>
  <mi>n</mi>
</msub>
```

ここで初めて mspace 要素が出てきたが，これは第 14 章「スペーシング」で述べる。

8.8 テンソルと前置き添字：mmultiscripts

テンソルの成分表示にはもっとややこしい添字が付く。これは mmultiscripts 要素を使って表す。multiscripts は多重添字という意味だ。

テンソルは数ベクトルや行列の概念を拡張したものだ。数ベクトルは 1 階のテンソルであり，その成分は a_i のように一つの添字を付けて表す。同様に行列は 2 階のテンソルであり，その成分は a_{ij} のように二つの添字を付けて表す。0 階のテンソルというものもあって，これは単なる数のこと。成分は一つしか持たず，したがって添字は付かない。もちろん 3 階以上のテンソルもある。微分幾何学やその応用である一般相対性理論，そして弾性体力学などで大活躍する。

このように，n 階のテンソルの成分は n 個の添字を付けて表す。

詳細は省くが，この添字は，実は上に付くこともある。つまり，数ベクトルの成分は a^i のようになることもある[30]し，行列の成分は a^{ij} とか，はたまた $a^i{}_j$ のようになることもある[31]。

mmultiscripts 要素は，テンソルの成分表示だけでなく，添字が前側に付く場合も含め，あらゆる上付き・下付きのパターンに対応している。一般形は ${}^{bd\cdots}_{ac\cdots}X^{qs\cdots}_{pr\cdots}$ のようになる。

ではその書き方を見ていこう。

まずは，右側の上下に添字が付くパターン。以下のようになる。添字は下付き・上付きの順に記述する，ということに注意しよう。この順序は msubsup 要素のときと同じだ。

X_a^b

```
<mmultiscripts>
  <mi>X</mi>
  <mi>a</mi>
  <mi>b</mi>
</mmultiscripts>
```

これはもちろん msubsup を使って書くこともできる。

下付き・上付きのペアは，以下のように複数書くことができる。引数の順序に注目されたい。

X_{ac}^{bd}

```
<mmultiscripts>
  <mi>X</mi>
  <mi>a</mi>
  <mi>b</mi>
  <mi>c</mi>
  <mi>d</mi>
</mmultiscripts>
```

次に，上付き添字の下や，下付き添字の上が空いているパターン。空けたい箇所に none という空要素を使い，以下のように書く。

$X_a{}^b$

```
<mmultiscripts>
  <mi>X</mi>
  <mi>a</mi>
  <none />
  <none />
  <mi>b</mi>
</mmultiscripts>
```

ここで， X_a^b と $X_a{}^b$ とでは数式としての意味が違う場合があることに注意しよう。

前側にも添字が付く場合，以下のように，前側の添字群を空要素 mprescripts の後に書く。

${}_a^b X_c^d$

```
<mmultiscripts>
  <mi>X</mi>
  <mi>c</mi>
  <mi>d</mi>
  <mprescripts />
  <mi>a</mi>
  <mi>b</mi>
</mmultiscripts>
```

mprescripts 要素は 1 回しか書けない。

これを使えば，陽子 2 個，中性子 2 個からなるヘリウムの同位体が次のように書ける。

---

30) この場合，見た目は累乗と区別がつかない。読者は文脈から判断する。テンソルの成分の累乗は，$(a_i)^2$ のように括弧付きで表したりする。

31) ここで，i と j の水平方向の位置がズレている（i の次に j が現れている）ことに注目。

```
<mmultiscripts>
  <mi>He</mi>
  <mprescripts />
  <mn>2</mn>
  <mn>4</mn>
</mmultiscripts>
```

元素記号の左下に原子番号（陽子の数），左上に質量数（陽子と中性子の数の和）を書く。

なお，原子番号は元素記号ごとに決まっている。例えば He（ヘリウム）なら原子番号は 2 に決まっているので，原子番号は略していい。この場合は none 要素を使って

```
<mmultiscripts>
  <mi>He</mi>
  <mprescripts />
  <none />
  <mn>4</mn>
</mmultiscripts>
```

と書く。

次に，順列・組合せの例を見よう。

n 個の物の中から，順序を考慮して m 個の物を取り出す方法の数を ${}_nP_m$ と書く[32]。以下のように記述する。

```
<mmultiscripts>
  <mo>P</mo>
  <mi>m</mi>
  <none />
  <mprescripts />
  <mi>n</mi>
  <none />
</mmultiscripts>
```

n 個の物の中から，順序無関係に m 個の物を取り出す方法の数[33] ${}_nC_m$ も書き方は全く同じなので，マークアップ例は略す。

MathML 3 仕様書

3.4.7 Prescripts and Tensor Indices <mmultiscripts>, <mprescripts/>, <none/>

8.9　アクセント記号

$\dot{x}$，$\bar{z}$，$\tilde{f}$，$\hat{H}$ に見られるような修飾記号を MathML 用語で「アクセント」と呼ぶ。

8.9.1　accent 属性

アクセントは mover で表現するのだが，以下の二つのマークアップを見てほしい[34]。分かりやすいよう見本を大きくした。

```
<mover accent="true">
  <mi>x</mi>
  <mo>.</mo>
</mover>
```

NG

```
<mover accent="false">
  <mi>x</mi>
  <mo>.</mo>
</mover>
```

mover 要素の accent 属性は，第二引数をアクセントとして扱うか否かを決定するものだ。

---

32) P は permutation（順列）より。なお，${}_nP_m$ のほかに $\binom{n}{m}$ などいろいろな書き方がある。

33) C は combination（組合せ）より。

34) $\dot{x}$ はアイザック・ニュートンによる微分のドット表記で，$\frac{dx}{dt}$ を意味する。$\ddot{x}$ は二階微分 $\frac{d^2x}{dt^2}$ である。上に載せるドットとしてピリオドが妥当かどうかについてここでは立ち入らない。17.6 節「accent」を参照のこと。

この例をよく見ると，上に載せる記号がアクセントとして扱われると，以下のようになることが分かる。

- 本体がイタリック体なら，その傾斜を考慮した水平位置に配置される。
- フォントサイズが小さくならない[35]。

では次のマークアップを見てみよう。

```
<mover>
  <mi>x</mi>
  <mo accent="true">.</mo>
</mover>
```

さきほどとは accent="true" を付ける要素が違っている。この書き方でも期待どおりの結果が得られた。これはどういうことか。

mo 要素の accent 属性は，自身をアクセントとして扱うか否かを決定する。mover 要素は第二引数が accent="true" な mo 要素である場合，自身の accent 属性の既定値が true アクセントとなる[36]。

次のマークアップはどうだろう。

```
<mover>
  <mi>H</mi>
  <mo>_</mo>
</mover>
```

どこにも accent 属性が書かれていないが，上線の位置からして期待どおりアクセントとして扱われていることが分かる。8.6 節「上線・下線」において，上線のマークアップに accent 属性を書かなかったのは既に見たとおりだ。

実はこの _（U+005F）という記号は，演算子辞書で「アクセント」と定義されているため，あらわに accent 属性を書かなくてもアクセントとして扱われるのだ。より正確で詳細な説明は第 17 章「演算子辞書」を参照されたい。

8.9.2 伸びるアクセント

上線・下線は本体に合わせて伸びる。このような性質を持ったアクセントをいくつか紹介しよう。

```
<mover>
  <mrow>
    <mi>B</mi>
    <mi>C</mi>
  </mrow>
  <mo>_</mo>
</mover>
```

```
<mover>
  <mrow>
    <mi>V</mi>
    <mi>W</mi>
  </mrow>
  <mo>&rarr;</mo>
</mover>
```

```
<mover>
  <mrow>
    <mi>V</mi>
    <mi>W</mi>
  </mrow>
  <mo>&#x21C0;</mo>
</mover>
```

点 C と点 D を結ぶ弧 $\overset{\frown}{CD}$ は以下のようにする[37]。

---

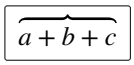
35) オーバースクリプトのサイズが変わるかどうかは「スクリプトレベル」というものが関わっているのだが，ここでは立ち入らない。8.12 節「添字のサイズ」を参照のこと。

36) 単一の mo のほか，「装飾された演算子」（embellished operator）の場合も同様だが，本文書では装飾された演算子には触れない。MathML 仕様書を参照のこと。

37) 点を表す文字は「C」「D」のようにローマン体とする流儀もあるが，ここではマークアップが簡素なイタリック体で示した。

```
<mover>
  <mrow>
    <mi>C</mi>
    <mi>D</mi>
  </mrow>
  <mo>&OverParenthesis;</mo>
</mover>
```

カーリーブラケット（curly brackets）とかブレースと呼ばれる記号は，上に被せるためのものが Unicode に U+23DE TOP CURLY BRACKET として定義されている。これは演算子辞書に伸びるアクセント記号として定義されている。



```
<mover>
  <mrow>
    <mi>a</mi><mo>+</mo><mi>b</mi><mo>+</mo><mi>c</mi>
  </mrow>
  <mo>&OverBrace;</mo>
</mover>
```

MathML 3 仕様書

3.4.5 Overscript <mover>

8.9.3 アンダースクリプト

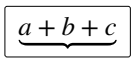
アクセントは munder 要素を使って下に置くこともできる。この場合，accent 属性ではなく accentunder 属性を使う。次の例は実際的なものではないが，使い方を示したものだ。

```
<munder accentunder="true">
  <mi>x</mi>
  <mo>.</mo>
</munder>
```

munder に accentunder を書かなかった場合の扱いは，mover 要素における accent 属性の扱いと似ている（8.9.1 節参照）。

下に置く記号が単一の mo 要素もしくは「装飾された演算子」であって，かつその mo 要素がアクセントである（accent="true" とあらわに指定されていたり，演算子辞書でアクセントと定義されている）場合，accentunder 属性の既定値は true となる。そうでない場合は false である。

カーリーブラケットを下に置く例も示しておこう。これには U+23DF BOTTOM CURLY BRACKET を用いる。

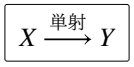


```
<munder>
  <mrow>
    <mi>a</mi><mo>+</mo><mi>b</mi><mo>+</mo><mi>c</mi>
  </mrow>
  <mo>&UnderBrace;</mo>
</munder>
```

この記号は既定の演算子辞書でアクセントと定義されているので，munder 要素に accentunder 属性を書かなくてよい。

8.10 矢印などに言葉を載せる

mover は矢印などの上に言葉を載せるためにも使うことができる。



```
<mi>X</mi>
<mover>
  <mo>&rarr;</mo>
  <mtext> 単射 </mtext>
</mover>
<mi>Y</mi>
```

矢印 → の長さが「単射」に合わせて伸びていることが分かる。このように mover の本体に → を置くと，オーバースクリプトの長さに応じて伸ばしてくれる。

「単射」の前後に   を置いているのは，空きを確保するためである。これがないと矢印の長さが「単射」とぴ

ったり同じになり，体裁が悪い。

なお，accent="true" は付けていない。もし付けると，以下のように「単射」の文字サイズが X などと同じになってしまう。

NG
$X \xrightarrow{\text{単射}} Y$

```
<mi>X</mi>
<mover accent="true">
  <mo>&rarr;</mo>
  <mtext> 単射 </mtext>
</mover>
<mi>Y</mi>
```

次のような場合にも使う。この例では mover を入れ子にしている。

$$n^m = \overbrace{n \times n \times \cdots \times n}^{m\text{ 回}}$$

```
<msup><mi>n</mi><mi>m</mi></msup>
<mo>=</mo>
<mover>
  <mrow>
    <mi>n</mi>
    <mo>&times;</mo>
    <mi>n</mi>
    <mo>&times;</mo>
    <mi>&ctdot;</mi>
    <mo>&times;</mo>
    <mi>n</mi>
  </mrow>
  <mover>
    <mo>&OverBrace;</mo>
    <mrow><mi>m</mi><mtext> 回</mtext></mrow>
  </mover>
</mover>
```

このような場合でも，⏞ が期待どおり本体に合わせて伸びていることに注目されたい。

8.11 疑似添字

この節では，プライム（′）のような右上に付く記号について説明する。

13.1 節「記号を表す簡易記法」で述べるが，AH Formatter では ASCII のアポストロフィー（U+0027）を mo 要素に入れれば正しいプライム（U+2032）に置き換えてくれる。したがって，プライムのマークアップは <mo>'</mo> でよい。

さて，プライムの付いた式，例えば x' のマークアップは，MathML 仕様書によれば以下のように書くことになっている。

x'

```
<msup>
  <mi>x</mi>
  <mo>'</mo>
</msup>
```

つまり，プライムの部分を上付きにするのである。

しかし，プライムという記号はもともと横に並べるだけでよいようにデザインされているはずだ。上付きにしてしてしまうと，余計に小さくなって上にせり上がってしまうのではないだろうか？

実はそうならない。次の，プライムを横に並べるだけのマークアップと比較してみよう。

x'

```
<mi>x</mi>
<mo>'</mo>
```

組版結果は同じだ[38)]。

これは，MathML の仕様でプライムが**疑似添字**（pseudo-scripts）とされているからだ。疑似添字と定義された文字は，上付きや下付きにされてもスクリプトレベル（8.12 節「添字のサイズ」参照）や垂直位置は変わらないことに

38) AH Formatter 以外のソフトウエアでこうなるかどうかは分からない。

なっている[39]。

疑似添字にはプライム類のほか，度記号（°），ASCII のアスタリスク（*）などがある。詳細は MathML 仕様書を参照されたい。

結果が同じならなぜわざわざ上付きにマークアップしなければならないのか。理由の一つは，x'_1 のような下付きと組み合わさったものは msubsup を使わなければうまく組めない，ということである[40]。

x'_1

```
<msubsup>
  <mi>x</mi>
  <mn>1</mn>
  <mo>'</mo>
</msubsup>
```

MathML 3 仕様書

7.7.2 Pseudo-scripts

8.12　添字のサイズ

e^{x_1} などのように，添字にさらに添字が付いていることがある。それどころか，

$$n^{n^{n^{n^{n^{n}}}}}$$

のような多重の入れ子になった添字もありうる[41]。

何段階目の添字か，を表す整数を「スクリプトレベル」という。先の例で言えば e^{x_1} はスクリプトレベル 0 であり，この式の x_1 の部分はスクリプトレベル 1，そしてその中の $_1$ の部分はスクリプトレベル 2，という具合だ。

スクリプトレベルが一つ上がれば文字サイズは一段階小さくなる。倍率の規定値は 0.71 だ[42]。

しかし，実は添字の文字サイズに下限があり，0.71 倍して算出したサイズがこの下限よりも小さかった場合はこの下限値が実際の文字サイズになる。

さきほどの n の肩に多重にn が載っている例をよく見てみよう。スクリプトレベル 2 までは確かに漸次小さくなっているが，そこから先は文字サイズは変わらない。垂直位置が上がっているだけである。

なぜ下限があるのかは明らかだろう。小さすぎる字は読みづらいからだ。

AH Formatter ではこの下限の既定値は 8 pt となっている。以下に述べるように，この値は大きすぎる[43]。

日本の書籍本文によく使われる 9 pt というサイズを考えてみよう。これがスクリプトレベル 0 だとすると，その添字は 9 pt × 0.71 = 6.39 pt となるはずだが，8 pt に達しないため，実際には 8 pt で組まれる。本文が 11 pt でさえ「0.71 倍」という値にはまったく意味が無い。

書体や印刷条件にもよるので一概には言えないが，漢字などを含んでいても 6 pt 程度ならまあ読める。英数字や簡単な記号なら 5 pt もあれば十分だ。

幸い，この下限値はオプション設定ファイルによって簡単に変えることができる。

8.12.1　添字サイズ下限値の変更

オプション設定ファイルは AH Formatter の組版動作などの設定を記述した XML ファイルだ。コマンドラインで AH Formatter を動かすとき，-i オプションでこのファイルのパスを指定してやればよい。

---

39) AH Formatter は，この仕様どおりに組むかどうかがオプション設定で変えられるようになっているが，既定値は true なので，仕様どおりとなる。

40) MathML 仕様書にはもう一つの根拠が書かれているが，そちらは私には今ひとつピンと来なかった。

41) これは荒唐無稽な例ではなく「指数タワー」と呼ばれるもので，巨大数の研究に出てくる。

42) なぜこんな半端な値かというと，$1/\sqrt{2} = 0.7071\cdots$ の概数なのである。よって，添字の添字は親文字の半分のサイズになる。

43) しかし，0.71 倍にしろ，8 pt にしろ，これは MathML 仕様書で規定されているとおりである。

このファイルには設定の全てを書く必要は無く，変更したい設定だけを書けばよい。

添字サイズ下限値は scriptminsize という設定である。pt や Q などの単位の付いた値を与える。

以下に，そのような設定を行うためのオプション設定ファイルを示す。この例では下限を 6 pt とした。

```
<?xml version="1.0" encoding="utf-8" ?>
<formatter-config>
  <mathml-settings scriptminsize="6pt" />
</formatter-config>
```

見てのとおり非常に単純だ。ルート要素は formatter-config であり，その直下に，MathML 関係のさまざまな設定を書くための mathml-settings 要素がある。そこに scriptminsize 属性を書けばいいだけである。

オプション設定ファイルの MathML の設定については下記を参照されたい。

http://www.antenna.co.jp/AHF/help/v64/ahf-optset.html#mathml-settings

第 9 章　数ベクトルと行列ほか：mtable，mtr，mtd

数ベクトルや行列は，要素を縦横に並べる。このために使うのが，HTML の table，tr，td によく似た mtable，mtr，mtd 要素だ。

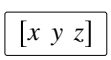
9.1 数ベクトル

縦ベクトルは mtable で要素を n 行 1 列に配列し，全体を ( ) か [ ] で囲む。

```
<mfenced>
  <mtable>
    <mtr>
      <mtd><mi>x</mi></mtd>
    </mtr>
    <mtr>
      <mtd><mi>y</mi></mtd>
    </mtr>
  </mtable>
</mfenced>
```

```
<mfenced open="[" close="]">
  <mtable>
    <mtr>
      <mtd><mi>x</mi></mtd>
    </mtr>
    <mtr>
      <mtd><mi>y</mi></mtd>
    </mtr>
  </mtable>
</mfenced>
```

横ベクトルなら，要素を 1 行 n 列に並べればよい。



```
<mfenced open="[" close="]">
  <mtable>
    <mtr>
      <mtd><mi>x</mi></mtd>
      <mtd><mi>y</mi></mtd>
      <mtd><mi>z</mi></mtd>
    </mtr>
  </mtable>
</mfenced>
```

なお，$\boldsymbol{v}$ のようなボールドイタリックのベクトル表記については第 11 章「スタイル」を，$\vec{p}$ や $\overrightarrow{PQ}$ のような矢印載せによるベクトル表記については 8.9 節「アクセント記号」を参照されたい。

MathML 3 仕様書

3.5.1 Table or Matrix <mtable>
3.5.2 Row in Table or Matrix <mtr>
3.5.4 Entry in Table or Matrix <mtd>

9.2 行列と行列式

行列は前節で見た組み方を n 行 m 列にしただけである。

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

```
<mfenced open="[" close="]">
  <mtable>
    <mtr>
      <mtd><mn>1</mn></mtd>
      <mtd><mn>0</mn></mtd>
    </mtr>
    <mtr>
      <mtd><mn>0</mn></mtd>
      <mtd><mn>1</mn></mtd>
    </mtr>
  </mtable>
</mfenced>
```

行間が空き過ぎと感じないだろうか。その場合，rowspacing 属性で行間をあらわに指定してやることができる。既定値は MathML 仕様書で規定されている 1.0ex である。

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

```
<mfenced open="[" close="]">
  <mtable rowspacing="0.2em">
    <mtr>
      <mtd><mn>1</mn></mtd>
      <mtd><mn>0</mn></mtd>
    </mtr>
    <mtr>
      <mtd><mn>0</mn></mtd>
      <mtd><mn>1</mn></mtd>
    </mtr>
  </mtable>
</mfenced>
```

rowspacing は垂直方向の空きなので，既定値はフォント依存の ex を単位としているが，もちろん上のように em を使ってもよい。

列間の空きは columnspacing 属性で調整する。既定値は MathML 仕様書では 0.8em となっているが，AH Formatter では 0.4em を採用している。

行列式は括弧の代わりに縦棒 | を使えばよい。

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix}$$

```
<mfenced open="|" close="|">
  <mtable rowspacing="0.2em">
    <mtr>
      <mtd><mi>a</mi></mtd>
      <mtd><mi>b</mi></mtd>
    </mtr>
    <mtr>
      <mtd><mi>c</mi></mtd>
      <mtd><mi>d</mi></mtd>
    </mtr>
  </mtable>
</mfenced>
```

縦棒と中身が接近し過ぎではないだろうか。また，上で見た数ベクトルや行列の例でも，括弧類の内側が窮屈に感じられるかもしれない。この問題は mtable の framespacing 属性で解決できる。

framespacing は，本来は

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

```
<mtable frame="dashed" framespacing="0.5em 0.25em">
  <mtr><mtd><mi>A</mi></mtd><mtd><mi>B</mi></mtd></mtr>
  <mtr><mtd><mi>C</mi></mtd><mtd><mi>D</mi></mtd></mtr>
</mtable>
```

のように frame 属性（値は none，solid，dashed）で枠線を付ける場合に，枠線との空きを左右・天地の順に指定す

るものだ。この例では，左と右にそれぞれ 0.5 em の空きが，上と下にそれぞれ 0.25 em の空きが出来ている。MathML 仕様書によれば，frame が none（無し）のときには天地・左右の空きがゼロになる。

しかし，AH Formatter では，framespacing の仕様を拡張し，三つ目の値を指定できるようにした。三つ目の値は，frame が none（既定値）のとき，つまり枠線を付けないときに左右に設ける空きを指定する。

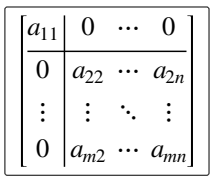
これを使えば行列式の見栄えを次のように改善することができる。

```
<mfenced open="|" close="|">
  <mtable rowspacing="0.2em" framespacing="0 0 0.15em">
    <mtr>
      <mtd><mi>a</mi></mtd>
      <mtd><mi>b</mi></mtd>
    </mtr>
    <mtr>
      <mtd><mi>c</mi></mtd>
      <mtd><mi>d</mi></mtd>
    </mtr>
  </mtable>
</mfenced>
```

なお，| | の内側の空きの問題については，17.8.3 節「絶対値記号」で再び触れることにする。

さて，mtable では，行間・列間に罫線を入れることもできる。それぞれ，rowlines，columnlines 属性に線の種類を並べて書く。線の種類は solid（実線），none（無し），dashed（破線）である。順に適用され，数が足りないと最後の値が繰り返される。

成分の表記を省略するには ⋮（⋮），⋯（⋯），⋱（⋱）などを使えばよい（13.5 節「省略記号」参照）。



```
<mfenced open="[" close="]">
  <mtable rowlines="solid none" columnlines="solid none">
    <mtr>
      <mtd><msub><mi>a</mi><mn>11</mn></msub></mtd>
      <mtd><mn>0</mn></mtd>
      <mtd><mo>&ctdot;</mo></mtd>
      <mtd><mn>0</mn></mtd>
    </mtr>
    <mtr>
      <mtd><mn>0</mn></mtd>
      <mtd><msub><mi>a</mi><mn>22</mn></msub></mtd>
      <mtd><mo>&ctdot;</mo></mtd>
      <mtd>
        <msub><mi>a</mi><mrow><mn>2</mn><mi>n</mi></mrow></msub>
      </mtd>
    </mtr>
    <mtr>
      <mtd><mo>&vellip;</mo></mtd>
      <mtd><mo>&vellip;</mo></mtd>
      <mtd><mo>&dtdot;</mo></mtd>
      <mtd><mo>&vellip;</mo></mtd>
    </mtr>
    <mtr>
      <mtd><mn>0</mn></mtd>
      <mtd>
        <msub><mi>a</mi><mrow><mi>m</mi><mn>2</mn></mrow></msub>
      </mtd>
      <mtd><mo>&ctdot;</mo></mtd>
      <mtd>
        <msub><mi>a</mi><mrow><mi>m</mi><mi>n</mi></mrow></msub>
      </mtd>
    </mtr>
  </mtable>
</mfenced>
```

なお，上の例ではできるだけ簡略に mtable の使い方を示すため，a_{11} や a_{mn} のマークアップで ⁣ を略したが，2.4 節で見たように，本来は

```
<msub>
  <mi>a</mi>
  <mrow>
    <mn>1</mn>
```

```
      <mo>&ic;</mo>
      <mn>1</mn>
    </mrow>
  </msub>
```

のように書くことになっている。

次に，mtable で作った大きな塊が，その周囲とどのように揃うのか見てみよう。

$$\begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix} = -2$$

```
<mfenced open="|" close="|">
  <mtable rowspacing="0.2em" framespacing="0 0 0.15em">
    <mtr><mtd><mn>1</mn></mtd><mtd><mn>2</mn></mtd></mtr>
    <mtr><mtd><mn>3</mn></mtd><mtd><mn>4</mn></mtd></mtr>
  </mtable>
</mfenced>
<mo>=</mo>
<mn>-2</mn>
```

何も考えなくても期待どおりの揃い方になった。

mtable には上下方向の揃え方を指定する align という属性があり，既定値が axis となっている。これは，mtable の天地中央を数式の軸（axis）に揃えることを意味する。数式の軸というのは，マイナス記号の高さの水平線だと思えばよい。上の組版例で確認されたい。

align には様々な指定ができる。MathML 仕様書を参照のこと。

9.3 rowspacing，columnspacingの初期値を変える

rowspacing も columnspacing も，オプション設定ファイルで初期値を変えることができる。そうすれば個々の mtable でいちいち指定しなくて済む。

以下のように指定すればよい（この値を推奨しているわけではない）。

```
<?xml version="1.0" encoding="utf-8" ?>
<formatter-config>
  <mathml-settings rowspacing="0.1em" columnspacing="0.25em" />
</formatter-config>
```

オプション設定ファイル自体の説明は 8.12.1 節「添字サイズ下限値の変更」を参照されたい。

9.4 セルをまとめる：rowspan，columnspan属性

HTML の table と同様，縦横のセルをひとまとめにすることができる。セルを縦につなぐのが rowspan，横につなぐのが columnspan である[44]。

次の例は対角行列（対角要素以外が 0 の正方行列）の表示である。0 は fontsize="2em" で通常の 2 倍の大きさにした。

$$\begin{bmatrix} 1 & & 0 \\ & 1 & \\ 0 & & \ddots \\ & & & 1 \end{bmatrix}$$

```
<mfenced open="[" close="]">
  <mtable rowspacing="0.2em">
    <mtr>
      <mtd><mn>1</mn></mtd>
      <mtd />
      <mtd columnspan=2 rowspan=2><mn fontsize="2em">0</mn></mtd>
    </mtr>
    <mtr>
      <mtd />
      <mtd><mn>1</mn></mtd>
    </mtr>
    <mtr>
      <mtd columnspan=2 rowspan=2><mn fontsize="2em">0</mn></mtd>
      <mtd><mo>&dtdot;</mo></mtd>
      <mtd />
    </mtr>
    <mtr>
      <mtd />
```

[44] HTML の場合は rowspan と **colspan**。混同せぬよう。

```
        <mtd><mn>1</mn></mtd>
      </mtr>
    </mtable>
  </mfenced>
```

9.5 水平位置を揃える

9.5.1 場合分け

mtable は，数ベクトル，行列だけでなく，縦横に要素を並べるさまざまな局面で使える。

ここでは場合分けの例を挙げよう。

$$|x| = \begin{cases} x & \text{if } x \geqq 0 \\ -x & \text{other} \end{cases}$$

```
<mfenced open="|" close="|"><mi>x</mi></mfenced>
<mo>=</mo>
<mrow>
  <mo rspace="0.25em">{</mo>
  <mtable columnalign="left" columnspacing="0.5em">
    <mtr>
      <mtd><mi>x</mi></mtd>
      <mtd>
        <mtext>if </mtext>
        <mi>x</mi><mo>&gE;</mo><mn>0</mn>
      </mtd>
    </mtr>
    <mtr>
      <mtd><mo>-</mo><mi>x</mi></mtd>
      <mtd><mtext>other</mtext></mtd>
    </mtr>
  </mtable>
</mrow>
```

ここで，mtable の columnalign 属性は，各セル内の水平方向の揃えを指定するものだ。既定値は center（中央揃え）なので，上のような場合は left で左揃えにしてやらなければならない。

なお，{ の右側を空けるために rspace 属性を使っているが，これについては 14.2 節「lspace，rspace 属性」で説明する。

9.5.2 等号の位置を揃える

式の変形の過程を = で縦に書き連ねる場合にも mtable が使える。

マークアップを分かりやすくするため左辺・右辺の式を単に L，R1，R2 で表すと以下のようになる。

$$\begin{aligned} \mathrm{L} &= \mathrm{R1} \\ &= \mathrm{R2} \end{aligned}$$

```
<mtable columnalign="left" columnspacing="thickmathspace">
  <mtr>
    <mtd><mtext>L</mtext></mtd>
    <mtd><mo>=</mo></mtd>
    <mtd><mtext>R1</mtext></mtd>
  </mtr>
  <mtr>
    <mtd />
    <mtd><mo>=</mo></mtd>
    <mtd><mtext>R2</mtext></mtd>
  </mtr>
</mtable>
```

columnspacing に指定した値，thickmathspace は 5/18 em に等しい。このような空き量の名称については 14.1 節「mspace」で説明する。

等号などの演算子の前後の空き量については第 17 章「演算子辞書」で説明する。

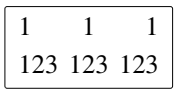
15.2 節「式変形の等号を揃える」では，等号を揃える別のやり方を述べる。

9.5.3 columnalign 属性

columnalign 属性については 9.5.1 節で述べたが，ここでもう少し詳しく見てみよう。

規定値は center（中央揃え）で，この他に left（左揃え），right（右揃え）も指定できる。

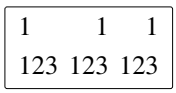
列ごとに揃え方を変えることもできる。columnalign に複数の値をスペースで区切って与えれば左から順に適用される。実際的な例ではないが，以下のようになる。




```
<mtable columnalign="left center right">
  <mtr>
    <mtd><mn>1</mn></mtd>
    <mtd><mn>1</mn></mtd>
    <mtd><mn>1</mn></mtd>
  </mtr>
  <mtr>
    <mtd><mn>123</mn></mtd>
    <mtd><mn>123</mn></mtd>
    <mtd><mn>123</mn></mtd>
  </mtr>
</mtable>
```

指定した値の数より実際の列の数が少なければ余分な指定値は無視される。

逆に指定した値の数が実際の列の数より少ない場合，最後の値が繰り返し適用される。




```
<mtable columnalign="left right">
  <mtr>
    <mtd><mn>1</mn></mtd>
    <mtd><mn>1</mn></mtd>
    <mtd><mn>1</mn></mtd>
  </mtr>
  <mtr>
    <mtd><mn>123</mn></mtd>
    <mtd><mn>123</mn></mtd>
    <mtd><mn>123</mn></mtd>
  </mtr>
</mtable>
```

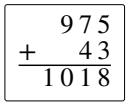
第 10 章　筆算

この章では，筆算の組版に用いる mstack，msrow，msline，mscarries，mscarry，mlongdiv，msgroup の使用例を挙げていく。これらは MathML 3.0 で追加されたものである。

10.1 mstack，msrow，msline

筆算の最大の特徴は，数字の位（くらい）が縦に揃っていることである。

簡単な例を見てみよう。




```
<mstack>
  <mn>975</mn>
  <msrow>
    <mo>+</mo>
    <none />
    <none />
    <mn>43</mn>
  </msrow>
  <msline />
  <mn>1018</mn>
</mstack>
```

このような組版には mstack 要素を用いる。mstack の引数一つ一つが一つの行となる。上の例では，水平罫線を含め，計四つの行でできている。

各行を検討しよう。第 1 行と第 4 行が最も単純で，それぞれ単一の mn 要素で出来ている。その数字一つ一つが一つの桁として組まれる。桁の間には適当な空きが出来るため，字間が空いている。この空きは mstack 要素の charspacing 属性で変えることができる。規定値は medium だが，loose にすればゆるく（広く）なり，tight にすれ

ばきつく（狭く）なる[45]。0.2em のような「長さ」を指定することもできる。

第 3 行は <msline /> で，これは見てのとおり水平罫線を表す空要素である。

やや複雑なのは第 2 行で，これは複数の要素で一つの行を構成するため，全体が msrow に入れられている。実を言えば他の行も全て**暗黙の** msrow の中に入っている。つまり，<mn>975</mn> の行は，実は <msrow><mn>975</mn></msrow> の省略記法なのだ。mstack の引数は全て msrow である。

「+」の部分はとくに説明を要しないだろう。

その次の <none /> はその桁が空であることを表している。8.8 節「テンソルと前置き添字」で同様の使用例を見た。2 桁分の空きを確保するため二つの <none /> が入れられている。

なお，

$$\begin{array}{r} 975 \\ +)\ \ 43 \\ \hline 1018 \end{array}$$

のように演算記号の後に閉じパーレンを付ける流儀もある。

次に，小数点がどうなるかを見てみよう。

$$\begin{array}{r} 101 \\ -\ 92.4 \\ \hline 8.6 \end{array}$$

```
<mstack>
  <mn>101</mn>
  <msrow>
    <mo>-</mo>
    <none />
    <mn>92.4</mn>
  </msrow>
  <msline />
  <mn>8.6</mn>
</mstack>
```

何も特別な指定をしていないのに，期待どおりの揃い方になった。ちょっと意外な感じがしないだろうか。さきほどの例では，各行は右揃えになるように見えたが，今回は右揃えではない。

実は mstack には stackalign という，行の水平配置を制御する属性があり，その既定値が decimalpoint となっている。小数点の位置を揃えるということだ。

整数の場合は小数点が存在しないが，1 の位の右に暗黙の小数点があるものとして扱われる。

なお，小数点としてピリオドでなくカンマを使う地域・言語圏も少なくないが，math 要素の decimalpoint 属性で任意の記号を小数点に指定することができる。その場合，指定した記号の位置で桁が揃うことになる。

MathML 3 仕様書

3.6.1 Stacks of Characters <mstack>
3.6.4 Rows in Elementary Math <msrow>
3.6.7 Horizontal Line <msline/>

10.2 mscarries, mscarry

ところで，筆算の計算過程を表示したいことがある。繰り上がりのメモを付した例を挙げる。

$$\begin{array}{r} {\scriptstyle 1} \\ 281 \\ +\ \ 46 \\ \hline 327 \end{array}$$

```
<mstack>
 <mscarries>
  <mn>1</mn>
  <none />
  <none />
 </mscarries>
 <mn>281</mn>
 <msrow>
  <mo>+</mo>
  <none />
```

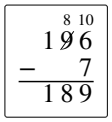
[45] AH Formatter は medium を 0.2 em に，tight を 0 em に，loose を 0.4 em にしているが，これらの値はオプション設定ファイル（8.12.1 節参照）で変えられる。

```
  <mn>46</mn>
 </msrow>
 <msline />
 <mn>327</mn>
</mstack>
```

このように，mscarries を用いる。carry とは繰り上がった数のことだ[46]。mscarries 要素の引数一つ一つが，対応する桁の位置に小さい字で組まれる。今の場合，「8」や「1」の上には何も置かないので，<none /> を二つ引数に追加している。

実は mscarries の引数は，mscarry という要素もしくは <none /> ということになっている。上の例のように直接 <mn>1</mn> と書いた場合は，**暗黙の** mscarry に含まれているものとして扱われる。

crossout という属性で本体のほうの数字を斜線で消すことができる[47]。



```
<mstack>
 <mscarries crossout="updiagonalstrike">
  <mn>8</mn>
  <mscarry crossout="none">
    <mn>10</mn>
  </mscarry>
 </mscarries>
 <mn>196</mn>
 <msrow>
  <mo>-</mo>
  <none />
  <none />
  <mn>7</mn>
 </msrow>
 <msline></msline>
 <mn>189</mn>
</mstack>
```

mscarries に crossout 属性を指定した。updiagonalstrike は右上がりの斜線だ。これにより，mscarries の各引数の下の数字に斜線が引かれることになる。

しかし，今の場合，1 の位には斜線を引きたくない。そこでこの桁だけ crossout="none" を指定して斜線を無しにするわけだが，mn 要素に直接このような属性を与えることはできないので，mscarry 要素をあらわに書いたのである。

なお，斜線の向きを右下がりにしたければ，crossout を downdiagonalstrike にすればよい。

斜線の向きだけでなく，繰り上がり・繰り下がりをどのように表示するかはさまざまな流儀がある。mscarries は次行に対する注記（annotation）の汎用的な仕組みなので，それぞれの流儀に合わせて自由に使えばいい。

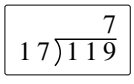
MathML 3 仕様書

3.6.5 Carries, Borrows, and Crossouts <mscarries>
3.6.6 A Single Carry <mscarry>

10.3 mlongdiv, msgroup

mlongdiv は除算（割り算）の筆算を組むための要素だ[48]。除算の筆算の書き方は地域・文化圏による違いが大きいが，ここで取り上げるスタイルは日本の初等教育を念頭に置いた例である。

まずはごく単純な例を見てみよう。



```
<mlongdiv>
  <mn>17</mn>
  <mn>7</mn>
  <mn>119</mn>
</mlongdiv>
```

---

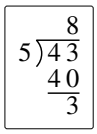
[46] 要素名の由来は繰り上がりなのだが，繰り下がりにも使う。

[47] 英語の cross out には「線を引いて消す」という意味がある。

[48] 英語の long division はほぼ日本語の除算の筆算にあたる。正確には部分積を略さないのを long division（長除法）と呼び，部分積とそれによる減算を暗算で済ませるのを short division（短除法）と呼ぶらしい。

このように，mlongdiv の最初の三つの引数は，順に①除数（割る数），②商（割った結果），③被除数（割られる数）となる。

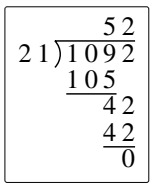
次に剰余（余り）が出る場合を見てみよう。



```
<mlongdiv>
  <mn>5</mn>
  <mn>8</mn>
  <mn>43</mn>
  <mn>40</mn>
  <msline />
  <mn>3</mn>
</mlongdiv>
```

mlongdiv の第三引数以降が mstack と同じようなルールで組まれることが伺える。

では商が 2 桁になる例を見よう。この場合，桁を正しい位置に持ってくるための指定が必要となる。



```
<mlongdiv>
  <mn>21</mn>
  <mn>52</mn>
  <mn>1092</mn>
  <msgroup position="1" shift="-1">
    <msgroup>
      <mn>105</mn>
      <msline length="3" />
    </msgroup>
    <msgroup>
      <mn>42</mn>
      <mn>42</mn>
      <msline length="2" />
      <mn>0</mn>
    </msgroup>
  </msgroup>
</mlongdiv>
```

msgroup 要素が初めて出てきたが，これは複数の行（msline が入っていても構わない）をひとまとめにするものだ。

一つの msgroup の中に二つの msgroup が入っている。これを一つ一つ検討しよう。

まず，〈42，42，罫線，0〉が一つの msgroup に入っている。この中で，1 の位（2，2，0）の位置が揃っていることと，罫線が正しい位置・長さになっていることに注目されたい。

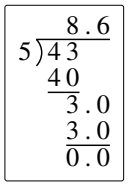
次に，〈105，罫線〉の msgroup を見よう。これも罫線が「105」に合っている。

そして，この二つの msgroup が一つの msgroup の引数となっている。ここに付された属性 position と shift が桁合わせするうえでのポイントだ。

shift 属性は，引数間の桁合わせ位置のずらし量を表している。今の場合，「105」の 1 の位に対し，「42」の 1 の位を 1 桁だけ低いほうにずらすので，-1 と指定している。引数が 3 個以上あっても，それらが順に 1 桁ずつ右へズレていくことになる。

一方，position 属性は，何も指定しなかった場合の位置から見てどこに位置を合わせるかを指定している。今の場合，指定がなければ「1092」の 1 の位と「105」の 1 の位が揃うはずである。高い位へ 1 だけズレた位置に配置するので，1 を指定している。

では次に，小数の商を得るようにしてみよう。



```
<mlongdiv>
  <mn>5</mn>
  <mn>8.6</mn>
  <mn>43</mn>
  <msgroup>
    <msgroup>
      <mn>40</mn>
      <msline length="2" />
    </msgroup>
    <msgroup>
      <mn>3.0</mn>
      <mn>3.0</mn>
      <msline position="-2" length="3" />
      <mn>0.0</mn>
```

```
        </msgroup>
      </msgroup>
    </mlongdiv>
```

まず，「40」の下に引かれた罫線を見よう。length 属性は文字どおり線の長さ（何桁に渡って線を引くか）を指定するものだ。これを指定しなかった場合（あるいは既定値である 0 を指定した場合），左右端一杯まで罫線が引かれてしまう。

位置がちょうど「40」の下に来たのは，msline の位置合わせのルールが数字列と同じだからだ。つまり，罫線の右端が（暗黙の）小数点に揃うということである。

次に「3.0」の下の罫線を見よう。ここでは position と length が指定されている。position 属性が無かった場合，罫線の右端は小数点の手前に揃う。これを低い位に向かって 2 列ぶんずらすために -2 を指定するのだが，この「列」は「桁」とは違って小数点も 1 列と数えることになっている。length も同様で，ここでは小数点も含めて 3 列ぶんの長さとしたいため 3 を指定した。

最後に，とくに説明は付さないが，乗算の筆算の例を挙げておこう。

$$\begin{array}{r} 26 \\ \times\ 17 \\ \hline {\scriptstyle 4} \\ 142 \\ 26 \\ \hline 442 \end{array}$$

```
<mstack>
  <mn>26</mn>
  <msrow>
    <mo>&times;</mo>
    <none />
    <mn>17</mn>
  </msrow>
  <msline />
  <mscarries><mn>4</mn><none /></mscarries>
  <mn>142</mn>
  <msgroup position="1">
    <mn>26</mn>
  </msgroup>
  <msline length="3" />
  <mn>442</mn>
</mstack>
```

MathML 3 仕様書

3.6.2 Long Division <mlongdiv>
3.6.3 Group Rows with Similiar Positions <msgroup>

本章で取り上げた mstack，mlongdiv，msgroup などの要素にはさまざまな属性が定義されており，奥が深いのでぜひ MathML 仕様書に当たっていただきたい。

第 11 章　スタイル

本章では，フォントスタイルや文字色といったスタイルに関するいくつかの属性を紹介する。

これらは，mi などの要素に付与することもできるし，スタイル変更のための要素 mstyle に付与することもできる。

11.1　mathvariant 属性

mathvariant 属性は，mi などの要素に付けて，イタリックやボールドなどの指定を行うものだ。

例えば，ベクトルなどの表記のためにボールドイタリックを指定するには mathvariant 属性に bold-italic を指定し，以下のようにする。

$$\boldsymbol{v}$$

```
<mi mathvariant="bold-italic">v</mi>
```

mathvariant 属性は mi，mn，mo，ms，mtext 要素に付けることができる。

値は次の表のとおり。

値	例	値	例
normal	R	bold	**R**
italic	*R*	bold-italic	***R***
fraktur	$\mathfrak{R}$	bold-fraktur	$\mathfrak{R}$
script	$\mathscr{R}$	bold-script	$\mathscr{R}$
double-struck	$\mathbb{R}$		
sans-serif	R	bold-sans-serif	**R**
sans-serif-italic	*R*	sans-serif-bold-italic	***R***
monospace	R		

実はこの他にも initial，tailed，looped，stretched があるのだが，いずれもアラビア文字に適用するものであり，AH Formatter V6.4 ではそもそも対応していないので割愛する。

2.2 節「識別名：mi」で少し触れたが，mathvariant の既定値は基本的には normal である。例外は mi 要素の中身が 1 文字の欧文アルファベット（ラテン文字，ギリシア文字，キリル文字）のとき。この場合に限り，既定値は italic となる。数字や記号などは 1 文字でもイタリックにはならない。

自然数，整数，有理数，実数，複素数全体の集合を表す記号にはいろいろな流儀があるが，まずボールドイタリックの ***N***，***Z***，***Q***，***R***，***C*** とするやり方がある。**R** など，ただのボールドにすることもある[49]。後述する double-struck による $\mathbb{R}$ が使われることもある。

零ベクトル，零行列（全ての要素が 0 である数ベクトルや行列）を表すのに数字「0」のボールドを用いることがある。

0
```
<mi mathvariant="bold">0</mi>
```

11.1.1 double-struck とは

double-struck というのは，日本語で黒板太字とか黒板ボールドなどと呼ばれているものに由来するスタイルだ。数学の授業で黒板に ***R*** のような文字を手書きしたいとき，ストロークの太い部分を二重線で表現する書き方がある。最小の手間で太字らしくする方法だ。これを黒板太字とか黒板ボールドと呼ぶ。

double-struck は，この手書き文字を逆に活字書体化したものだ[50]。印刷物でボールド（やボールドイタリック）の代わりに double-struck を使用することについては根強い反対論もある。

11.1.2 フラクトゥール

フラクトゥール（Fraktur）はドイツ文字とか亀甲（きっこう）文字などとも呼ばれる，ドイツで使われてきた書体だ[51]。

若干の使用例を挙げる。

リー代数[52]を表す変数を $\mathfrak{g}$ と書いたり，イデアルというものを $\mathfrak{a}$ と書いたりする。

リー代数には，何らかのリー群に付随するものがあるが，例えば n 次の特殊ユニタリー群 SU(n) に付随するリー

[49] これはボールドイタリックを持たないフォントファミリーが使われた場合の代替手段の名残かもしれない。活版時代からデジタルフォント時代に至るまで，欧文フォントのファミリーがボールドイタリックを持たないことはごく普通にあった。

[50] double-struck という言葉の直接の意味はタイプライターの二度打ちであるらしい。擬似的に太字にするという点は共通しているが，なぜこれがこの意味で使われるのかは分らない。

[51] フラクトゥーアとも。ナチス政権下で禁止令が出されたことがある。戦後あまり使われなくなった。

[52]「リー」は人名（Lie）。名称が紛らわしいが，リー代数は代数**学**の名前ではない。「リー代数」の「代数」は，「多元環」とも呼ばれる数学的構造の別名である。英語でも Lie algebra などと呼ぶ。

代数を，同じ綴りを用いて $\mathfrak{su}(n)$ と書く習慣がある。

また，複素数 z の実部，虚部をそれぞれ $\mathfrak{R}\,z$，$\mathfrak{I}\,z$ と書くことがある。現在は，それぞれ $\mathrm{Re}\,z$，$\mathrm{Im}\,z$ と書くことが多くなってきたようだ。

11.1.3 スクリプト体

「スクリプト体」は（広義には）手書き風にデザインされた活字書体全般を指す言葉である。したがって非常に幅が広い。

例えば $\mathscr{o}$ と $\mathcal{D}$ とでは同じ分類に入れるのも憚られるほど違うが，いずれも広義のスクリプト体である。後者はいわゆるブラッシュ・スクリプト，つまり，カリグラフィーペンでなくブラシ（筆）で書いたようにデザインされた書体の一種である。

11.1.4 サンセリフ

サンセリフ体（sans serif）は，`mathvariant` 属性の値としては，ウエイト（太さ）の違いと立体／斜体の違いに応じて `sans-serif`，`bold-sans-serif`，`sans-serif-italic`，`sans-serif-bold-italic` の四つがある。

『日本物理学会誌』の投稿規定では，テンソルに立体のサンセリフを使うことになっている。

http://www.jps.or.jp/books/files/toukou-kitei.pdf

11.1.5 Unicode 文字との関係

実は Unicode には，基本的な英数字についてボールドイタリック版，フラクトゥール版，スクリプト体版，ダブルストラック版のコードポイントがある。

例えば，`<mi mathvariant="script">H</mi>` として表現できるスクリプト体の H（$\mathscr{H}$）は，U+210B SCRIPT CAPITAL H として Unicode に定義されているので，`<mi>&#x210B;</mi>` と書くこともできる。これが収録されているエリアは Letterlike Symbols（文字様記号）だが，他のスクリプト体アルファベット，例えばスクリプト体の A（$\mathscr{A}$）は U+1D49C MATHEMATICAL SCRIPT CAPITAL A として Mathematical Alphanumeric Symbols に収録されている。このような乱れは Unicode が発展してきた歴史的事情によるのだろう。

11.2 `mathcolor` 属性

全ての MathML プレゼンテーション要素に `mathcolor` 属性を付けることができる。これは文字どおり文字色を指定するものだ[53]。

p `<mi mathcolor="red">p</mi>`

色指定には，`orange` のような HTML 色名や `#F399AC` のような 16 進 RGB が使えるほか，AH Formatter 拡張である `cmyk` 関数を使った `cmyk(0, 0.7, 1, 0)` のような CMYK 指定も可能だ[54]。

印刷物ではあまりカラフルな数式は考えないが，色が効果的に使える場面もある。例えば，次のような正誤表はいかがだろう。複雑な数式を含む正誤表は，どこが違っているのか一見して分かりにくい。そこで，相違点に色づけするのである（Σ の前の符号と，括弧内第二項の偏微分の分子が違っている）。

---

53) 本章ではここ以降，色を使った組見本が出てくる。モノクロ印刷ではグレイになって分かりづらいがご了承いただきたい。

54) `cmyk` 関数は V6.3 で導入されたもので，それまでは AH Formatter 拡張である `rgb-icc` 関数を使って，`rgb-icc(#CMYK, 0, 0.7, 1, 0)` と書いていた。

箇所	誤	正
p. 214, l. 8	$\frac{d\rho}{dt} = \frac{\partial\rho}{\partial t} - \sum_{i=1}^{n}\left(\frac{\partial\rho}{\partial q_i}\dot{q}_i + \frac{\partial p}{\partial p_i}\dot{p}_i\right) = 0$	$\frac{d\rho}{dt} = \frac{\partial\rho}{\partial t} + \sum_{i=1}^{n}\left(\frac{\partial\rho}{\partial q_i}\dot{q}_i + \frac{\partial\rho}{\partial p_i}\dot{p}_i\right) = 0$

モノクロ印刷物にする場合は使えないが，PDF で公開する正誤表なら使えるだろう。

なお，色分けはうまく使えば効果的だが，アクセシビリティーの観点からすると，色が区別できないと意味が読み取れないような使い方は避けるべきだ。

MathML 3 仕様書

3.1.10 Mathematics style attributes common to presentation elements

11.3 mathbackground 属性

mathcolor 属性と同様，全ての MathML 要素に mathbackground 属性を付けることができる。これは文字どおり背景色を指定するものだ。

$x + y$

```
<mi>x</mi>
<mo>+</mo>
<mi mathbackground="orange">y</mi>
```

先ほどの正誤表のような使い方もできるだろう。文字の色を変えるよりも背景色を付けるほうが色弱者には分かりやすいかもしれない。

MathML 3 仕様書

3.1.10 Mathematics style attributes common to presentation elements

11.4 mstyle 要素

mstyle はさまざまな体裁を制御する要素で，MathML プレゼンテーション要素のほとんどの属性を付与することができる。

それに加え，以下のような属性を持たせることもできる。

- displaystyle（true または false を指定）　1.8 節参照
- scriptlevel（+，- または非負整数で指定）　8.12 節参照
- scriptsizemultiplier（添字サイズの倍率；0.65 のような数で指定）　8.12 節参照
- scriptminsize（添字サイズの下限；6pt のような長さで指定）　8.12 節参照
- infixlinebreakstyle（中置演算子の行分割方法；before，after，duplicate）
- decimalpoint（桁揃えに使う小数点の記号；既定値は .〔ピリオド〕）

mstyle で体裁を変えるにあたって，一つ注意しなければならない点を挙げる。次の例を検討しよう。

NG
$x+y$

```
<mi>x</mi>
<mstyle mathcolor="red">
  <mo>+</mo>
  <mi>y</mi>
</mstyle>
```

これは，$x + y$ の $+ y$ の部分の色を変えたものだが，+ の前のあるはずの空きが無くなり，後ろの空きが非常に狭くなった。

これは本来 form="infix"（中置演算子）であるはずの <mo>+</mo> が form="prefix"（前置演算子）とみなされたためである。mstyle の中に入れることによって，<mo>+</mo><mi>y</mi> の部分が「推定された mrow」（4.1 節）で包まれる。すると <mo>+</mo> は mrow 要素内の位置から前置演算子と判定される。こういう理屈である。

さきほどの例で，mstyle を生かしたまま解決するには，mo 要素にあらわに form 属性を付与して，

```
<mi>x</mi>
<mstyle mathcolor="red">
  <mo form="infix">+</mo>
  <mi>y</mi>
</mstyle>
```

と書く。form 属性については第 17 章「演算子辞書」で解説する。

MathML 3 仕様書

3.3.4 Style Change <mstyle>

第 12 章　非ラテン文字

12.1　ギリシア文字

ギリシア文字には全て文字実体参照が定義されている。表のごとくに全て小文字の文字名称で小文字のギリシア文字になり，先頭を大文字にすると大文字のギリシア文字になる。

しかし日本語環境では文字実体参照を使わなくとも，例えば IME で「あるふぁ」や「ぱい」として変換される「α」「π」をそのまま使えばよい。いわゆる全角／半角の区別は無い。

小文字	入力	大文字	入力	名称
α	α	Α	Α	アルファ
β	β	Β	Β	ベータ
γ	γ	Γ	Γ	ガンマ
δ	δ	Δ	Δ	デルタ
ε	ε	Ε	Ε	エプシロン
ζ	ζ	Ζ	Ζ	ゼータ
η	η	Η	Η	エータ
θ	θ	Θ	Θ	シータ
ι	ι	Ι	Ι	イオタ
κ	κ	Κ	Κ	カッパ
λ	λ	Λ	Λ	ラムダ
μ	μ	Μ	Μ	ミュー
ν	ν	Ν	Ν	ニュー
ξ	ξ	Ξ	Ξ	クシー
ο	ο	Ο	Ο	オミクロン
π	π	Π	Π	パイ
ρ	ρ	Ρ	Ρ	ロー
σ	σ	Σ	Σ	シグマ
τ	τ	Τ	Τ	タウ
υ	υ	Υ	Υ	ユプシロン
φ	φ	Φ	Φ	ファイ
χ	χ	Χ	Χ	カイ

小文字	入力	大文字	入力	名称
ψ	ψ	Ψ	Ψ	プシー
ω	ω	Ω	Ω	オメガ

日本語名称は，日本の印刷物制作現場で文字を識別するための便宜的なものだ。日本における慣習をベースにしたが，ε か υ か紛らわしい「イプシロン」を避けるなどした[55]。

ギリシア文字の小文字には

ϕ　ϕ

のような異体字がいくつかある。これらの文字実体参照を次に掲げる。

記号	入力	UCS4	参考		
			記号	入力	UCS4
ϵ	ϵ	U+03F5	ε	ε	U+03B5
ϑ	ϑ	U+03D1	θ	θ	U+03B8
ϖ	ϖ	U+03D6	π	π	U+03C0
ϱ	ϱ	U+03F1	ρ	ρ	U+03C1
ς	ς	U+03C2	σ	σ	U+03C3
ϕ	ϕ	U+03D5	φ	φ	U+03C6

ギリシア文字にもローマン体とイタリック体があるが，MathML における指定方法はラテン文字と変らない。すなわち，mi 要素に 1 文字だけ含まれた場合にイタリック体，その他の場合にローマン体になり[56]，変更したければ mathvariant 属性を使う（11.1 節「mathvariant 属性」）。

なお，総和記号 ∑，総積記号 ∏ はそれぞれギリシア文字の Σ（シグマ），Π（パイ）に由来するが，Unicode では別のコードポイントが与えられており，そちらを使わなければならない（8.5 節参照）。

Unicode には，ギリシア文字の小文字 μ とは別に，マイクロ記号 µ（10^{-3} を意味する，単位記号の接頭語）のコードポイント U+00B5 MICRO SIGN が用意してある。

12.2 キリル文字

キリル文字はブルガリア語，ロシア語，セルビア語，モンゴル語などに用いられている文字である。

英語などのラテン文字を使う言語や日本語などの文書において，数式にキリル文字を使う機会はあまり無いかもしれないが，ここではローマン体とイタリック体[57]についてだけ述べる。

キリル文字も，ラテン文字やギリシア文字と同様，mi 要素に 1 文字だけ含まれた場合にイタリック体，その他の場合にローマン体になり，変更したければ mathvariant 属性を使う。

12.3 その他の文字

集合の濃度を表すのに使う ℵ（アレフ）はヘブライ文字の第 1 字であり，文字実体参照は ℵ である。

ヘブライ文字には本来はイタリック体という概念はないが，もし mathvariant="italic" を指定したら斜体にはなる。

しかし，AH Formatter が，1 文字の mi 要素をイタリック体にするのはラテン文字，ギリシア文字，キリル文字に限られているので，敢えて指定しない限りヘブライ文字が斜体にされる恐れはない。

---

55) 結果，英語風読み（例：ファイ）と現代ギリシア語風読み（例：ベータ）のチャンポンになっている。

56) 繰り返しになるが，ギリシア文字にこのルールが適用されるのは MathML の規定ではなく AH Formatter の仕様である。

57) ロシア語ではイタリック体と言わず，英語の cursive と同源のクルシフ（курсив）という用語を使う。

$\aleph_0$

```
<msub>
  <mi>&aleph;</mi>
  <mn>0</mn>
</msub>
```

代数学では $\mathfrak{a}$，$\mathfrak{g}$ といった文字が使われる。「ドイツ文字」とも呼ばれるが，ラテン文字とは異なる文字体系というよりも，ラテン文字のフラクトゥールという書体（11.1.2 節）だ。

第 13 章　記号

13.1　記号を表す簡易記法

2.1 節「数値：mn」や 2.3 節「演算子」で述べたように，mn，mo 要素の中では ASCII のハイフン（U+002D）がマイナス記号（U+2212）に置き換えられる。mi 要素でも同じだ。

同様に，ASCII のアポストロフィー[58]（', U+0027）を打てば，プライム（U+2032）に置き換えられる。

よって，以下のように書くことができる。

$x - x'$

```
<mi>x</mi>
<mo>-</mo>
<msup>
  <mi>x</mi>
  <mo>'</mo>
</msup>
```

2〜4 重プライムも，ASCII のアポストロフィーを 2〜4 個書けばよい。

x'', x''', x''''

```
<msup><mi>x</mi><mo>''</mo></msup>
<mo>,</mo>
<msup><mi>x</mi><mo>'''</mo></msup>
<mo>,</mo>
<msup><mi>x</mi><mo>''''</mo></msup>
```

このような簡易記法は MathML 仕様書で言及されているが，どのような記号の置き換えを行うかは処理系に依存するとしている。つまり，AH Formatter が対応していても，他のソフトウエアが対応しているとは限らない。したがって，書いたものが他に流用される可能性があるなら，このような簡易記法を用いるかどうか慎重に判断すべきだろう。

ところで，上記の置き換えが起こるのは，mn，mi，mo 要素のみである。これとは別に，mtext 要素（第 3 章）などにおいては，U+002D HYPHEN-MINUS が U+2010 HYPHEN に置き換えられる。これは平たく言えば，ASCII に含まれる，キーボードで簡単に打てる〈いわゆるハイフン〉が〈れっきとしたハイフン〉に置換されることを意味する。多くのフォントで両者は同じデザインになっているので，利用者は目で見て気づくことは無いだろう。2.1 節「数値：mn」の脚注も参照されたい。

Klein-Gordon equation

```
<mtext>Klein-Gordon equation</mtext>
```

MathML 3 仕様書

7.7.1 Keyboard Characters

以下の「文字の置換」も参照されたい。

---

[58] ASCII のアポストロフィーは，少なくとも①アポストロフィー，②左シングルクオート，③右シングルクオートの三つの働きを兼ねた記号である。その起源はタイプライター，テレタイプにまで遡る。印刷物で普通の文章に使うことは無い。使うのは，プログラムなどコンピューターの文字を表すときだけである。通常のタイポグラフィーにおいては左シングルクオート（‘）に U+2018 を，右シングルクオートとアポストロフィー（’）には U+2019 を用いる。

http://www.antenna.co.jp/AHF/help/v64/ahf-mathml3.html#MathML-char-subst

13.2　二項演算子

二項演算子は Unicode に極めて多数のものが定義されているが，ここにはその一部を挙げた。

記号	入力	UCS4
＋	+	U+002B
−	−	U+2212
±	±	U+00B1
∓	∓	U+2213
×	×	U+00D7
÷	÷	U+00F7
·	·	U+00B7

記号	入力	UCS4
∖	∖	U+2216
∗	∗	U+2217
∘	∘	U+2218
∶	∶	U+2236
∧	∧	U+2227
∨	∨	U+2228

記号	入力	UCS4
∩	∩	U+2229
∪	∪	U+222A
⊕	⊕	U+2295
⊖	⊖	U+2296
⊗	⊗	U+2297
⊙	⊙	U+2299

アスタリスク演算子（∗）に使う記号（U+2217 ASTERISK OPERATOR）は ASCII のアスタリスク（*, U+002A）とは別物である。

$f * g$

```
<mi>f</mi>
<mo>&lowast;</mo>
<mi>g</mi>
```

NG
$f * g$

```
<mi>f</mi>
<mo>*</mo>
<mi>g</mi>
```

複素共役などの意味で肩に付けるアスタリスクも，U+2217 を上付きにするのがいいだろう。

z^*

```
<msup>
  <mi>z</mi>
  <mo>&lowast;</mo>
</msup>
```

3：2 のような比を表すには，ASCII のコロン（:, U+003A）でなく ∶（U+2236）を用いる。

$6 : 4 = 3 : 2$

```
<mn>6</mn>
<mo>&ratio;</mo>
<mn>4</mn>
<mo>=</mo>
<mn>3</mn>
<mo>&ratio;</mo>
<mn>2</mn>
```

ASCII のコロンの場合，左右の空きが狭くかつ不均等（左より右のほうが大きく空く）になるので，比を表すのには使えない。ASCII のコロンは $f:A \to B$ のようなところで使う。しかし，この例ではコロンのあとの空きが十分でないと感じるかもしれない。

一方，比に ∶ を用いた場合，逆に左右の空きが大きすぎると感じるかもしれない。

ASCII のコロンにせよ，∶にせよ，演算子の左右にどれだけの空きを確保するかは，演算子辞書で定義されている。オプション設定ファイルで演算子辞書をカスタマイズすれば，左右の空きを変えることも可能だ。具体的な方法は 17.8 節「演算子辞書のカスタマイズ」を参照されたい。

13.2.1　二項演算子の使用例

複数の二項演算子が出てくる数式の例を挙げよう。

$f \circ g : x \mapsto f(g(x))$

```
<mi>f</mi>
<mo>&compfn;</mo>
<mi>g</mi>
<mo>:</mo>
```

```
<mi>x</mi>
<mo>&map;</mo>
<mi>f</mi>
<mo>&af;</mo>
<mfenced>
  <mrow>
    <mi>g</mi>
    <mo>&af;</mo>
    <mfenced><mi>x</mi></mfenced>
  </mrow>
</mfenced>
```

見てのとおりスペーシングがあまりよくないが，既定の演算子辞書ではこのようになる。演算子辞書をカスタマイズすれば改善されるだろう。

13.3 関係演算子

関係演算子も Unicode に多数定義されているが，よく使うものを挙げよう。

記号	入力	UCS4
∼	∼	U+223C
≃	≃	U+2243
≅	≅	U+2245
≈	≈	U+2248
≒	≒	U+2252
≠	≠	U+2260
≡	≡	U+2261
≢	≢	U+2262
<	<	U+003C
≤	≤	U+2264
≦	≦	U+2266
>	>	U+003E
≥	≥	U+2265
≧	≧	U+2267

記号	入力	UCS4
≪	≪	U+226A
≫	≫	U+226B
⊂	⊂	U+2282
⊃	⊃	U+2283
⊄	⊄	U+2284
⊅	⊅	U+2285
⊆	⊆	U+2286
⊇	⊇	U+2287
⊈	⊈	U+2288
⊉	⊉	U+2289
⊊	⊊	U+228A
⊋	⊋	U+228B

記号	入力	UCS4
∝	∝	U+221D
∈	∈	U+2208
∉	∉	U+2209
∋	∋	U+220B
∌	∌	U+220C
∣	∣	U+2223
∤	∤	U+2224
∥	∥	U+2225
∦	∦	U+2226
⊥	⊥	U+22A5

集合 A が集合 B の部分集合であるとは，A の元全てが B の元であることを言う。そのため A は A 自身の部分集合である。自分以外の部分集合を真部分集合と呼ぶ。

ここで，部分集合と真部分集合を表す記号に複数の流儀があって統一されていないことに注意されたい。基本的には以下の三つのパターンがある。

	A は B の部分集合	**A は B の真部分集合**
流儀 1	$A \subseteq B$	$A \subset B$
流儀 2	$A \subseteq B$	$A \subsetneq B$
流儀 3	$A \subset B$	$A \subsetneq B$

この表を見ると，記号単体で見たときに曖昧（多義的）なのは $\subset$ だけであることが分かる。よって，これを含まない流儀 2 は誤解の余地が無い。いずれにせよ，同一著作内で統一することと，必要に応じて記号の意味を明らかにすることが重要である。

n が m を割り切る（つまり n が m の約数である）ことを $n \mid m$ と書く。その否定は $n \nmid m$ である。

平行の記号（∥）はフォントによっては // のように斜めにデザインされている。

垂直の記号は $AB \perp CD$（線分 AB と線分 CD は垂直の意）のように使う。U+22A5 の Unicode 名は UP TACK（上向き画鋲？）である。これとは別に U+27C2 PERPENDICULAR という文字もある。

13.3.1 関係演算子の使用例

$$\forall\varepsilon > 0; \exists\delta > 0; |f(\delta)| < \varepsilon$$

```
<mo>&forall;</mo>
<mi>&epsilon;</mi>
<mo>&gt;</mo>
<mn>0</mn>
<mo>;</mo>
<mo>&exist;</mo>
<mi>&delta;</mi>
<mo>&gt;</mo>
<mn>0</mn>
<mo>;</mo>
<mfenced open="|" close="|">
  <mrow>
    <mi>f</mi>
    <mo>&af;</mo>
    <mfenced>
      <mi>&delta;</mi>
    </mfenced>
  </mrow>
</mfenced>
<mo>&lt;</mo>
<mi>&epsilon;</mi>
```

この例もスペーシングがあまりよくないが，演算子辞書のカスタマイズで改善されるだろう。

13.4 囲み記号

よく使われる囲み記号の例を挙げる。

組版例	open 属性	UCS4	close 属性	UCS4
(x)	(	U+0028	)	U+0029
$[x]$	[	U+005B	]	U+005D
$\{x\}$	{	U+007B	}	U+007D
$\lvert x\rvert$	|	U+007C	|	U+007C
$\lVert x\rVert$	||		||	
$\langle x\rangle$	⟨	U+2329	⟩	U+232A
$\lfloor x\rfloor$	⌊	U+230A	⌋	U+230B
$\lceil x\rceil$	⌈	U+2308	⌉	U+2309
$[\![x]\!]$	⟦	U+27E6	⟧	U+27E7

ベクトル $\boldsymbol{v}$ のノルムを $\|\boldsymbol{v}\|$ などと書くが，この || は，ASCII の縦棒 | を二つ並べて書けばよい。この記号については 17.8.3「絶対値記号」で再び触れる。

13.5 省略記号

記号	入力	UCS4
⋮	⋮	U+22EE
…	…	U+2026
⋯	⋯	U+22EF

記号	入力	UCS4
⋰	⋰	U+22F0
⋱	⋱	U+22F1

この表のグリフは全て STIX Regular フォント（16.1 節参照）であるが，… は他よりドットが小さく，また

ドットの間隔が ⋯ より広い。そういうフォントなのだと割り切るしか無いだろう。

13.6 矢印

矢印類は U+2190–21FF に定義されている。網羅的な一覧は Unicode コードチャートを参照のこと。

フォントによって，同じ種類の矢印であってもグリフがあったりなかったりする。この場合，デザインの異なる矢印が混在することになりかねない。矢印に限ったことではないが，どのフォントにどの記号が含まれているかをあらかじめ把握するようにしたい。

記号	入力	UCS4
←	←	U+2190
↑	↑	U+2191
→	→	U+2192
↓	↓	U+2193
↔	↔	U+2194
↕	↕	U+2195
↖	↖	U+2196
↗	↗	U+2197
↘	↘	U+2198
↙	↙	U+2199

記号	入力	UCS4
⇐	⇐	U+21D0
⇑	⇑	U+21D1
⇒	⇒	U+21D2
⇓	⇓	U+21D3
⇔	⇔	U+21D4
⇕	⇕	U+21D5
⇖	⇖	U+21D6
⇗	⇗	U+21D7
⇘	⇘	U+21D8
⇙	⇙	U+21D9

記号	入力	UCS4
⇄	⇄	U+21C4
⇆	⇆	U+21C6
↦	↦	U+21A6
⟼	⟼	U+27FC

13.7 スペース

Unicode には ASCII のスペース（U+0020）の他にもさまざまなスペースが定義されている。

名称	入力	UCS4	幅	見本
EN SPACE		U+2002	1/2 em（半角）	x x 0 0
EM SPACE		U+2003	1 em（全角）	x x 0 0
THREE-PER-EM SPACE		U+2004	1/3 em（三分）	x x 0 0
FOUR-PER-EM SPACE		U+2005	1/4 em（四分）	x x 0 0
SIX-PER-EM SPACE		U+2006	1/6 em（六分）	x x 0 0
FIGURE SPACE		U+2007	数字幅	x x 0 0
PUNCTUATION SPACE		U+2008	句読点幅	x x 0 0
THIN SPACE		U+2009		x x 0 0
HAIR SPACE		U+200A		xx00
ZERO WIDTH SPACE		U+200B	無し	xx00

四分空き（FOUR-PER-EM SPACE）は和欧混植で単位記号の前の空きに使うことができる。

消費電力量は 3.6 kWh だった。

```
消費電力量は3.6 kWhだった。
```

THIN SPACE と HAIR SPACE の幅がいくらかは，Unicode の仕様としては決まっていない。AH Formatter では，既定値としてそれぞれ 0.2 em，0.1 em となっているが，この値はオプション設定ファイルで変えることができる。

ここで取り上げた空白類は，使用フォントの実装に関わらず，仕様どおりの固定幅空白として扱われるのが AH Formatter の既定の動作である。詳細はオンラインマニュアルの次のページを参照されたい。

http://www.antenna.co.jp/AHF/help/v64/ahf-optset.html#fixed-width-space-treatment

13.8　アクセント記号

ここでは，8.9 節で述べた mover，munder などのアクセントに用いられる記号の例を挙げる。これらの中には，アクセント記号としてだけでなく演算子として用いるものなどもあるので，他の節と一部重複している。

この表の「プロパティー」欄に書かれた stretchy，accent などの意味は第 17 章「演算子辞書」を参照されたい。表には，演算子のフォームが postfix ないし infix の場合の値を掲載している。

記号	入力	UCS4	プロパティー	組版例
_	_	U+005F	stretchy, accent	$\overline{x}$
‾	‾	U+203E	stretchy, accent	$\overline{x}$
^	^	U+005E	stretchy, accent	$\hat{x}$
→	→	U+2192	stretchy, accent	$\overrightarrow{x}$
˙	.	U+002E		$\dot{x}$
¨	..			$\ddot{x}$
⃛	...			$\dddot{x}$
˙	˙	U+02D9	accent	$\dot{x}$
¨	¨	U+00A8	accent	$\ddot{x}$
⃛	⃛	U+20DB	accent	$\dddot{x}$
⃜	⃜	U+20DC	accent	$\ddddot{x}$
~	~	U+007E	stretchy, accent	$\widetilde{x}$
˜	˜	U+02DC	stretchy, accent	$\tilde{x}$
⏜	⏜	U+23DC	stretchy, accent	$\overparen{PQ}$
⏝	⏝	U+23DD	stretchy, accent	$\underparen{PQ}$
⏞	⏞	U+23DE	stretchy, accent	$x+\overbrace{y+z}$
⏟	⏟	U+23DF	stretchy, accent	$\underbrace{x+y}+z$

この表で，入力方法として実体参照にしていないものは ASCII に含まれる記号である。したがって，キーボードで容易に入力できる。

8.6 節「上線・下線」で述べたように，上線・下線とも U+005F を使えばよい。

$\hat{x}$ のハットも，ASCII にあるサーカムフレックスでよい。

ピリオドを 1〜3 個連ねたものは既定の演算子辞書にいちおうあるが，アクセントとしての使用は想定されていないので，使うなら演算子辞書をカスタマイズする必要がある。具体的な方法は 17.8.4 節「微分のドット表記」を参照されたい。

チルダにも二つ候補がある。使用するフォントで，気に入ったデザインのほうを使えばいいと思う。

U+23DD（下パーレン）の用例を見つけることはできなかった。「組版例」欄には，仮に U+23DC（上パーレン）と同じように（ただしアンダースクリプトで）使ってみたらどうなるかを示した。

13.9　その他の MathML 演算子

ここでは，本章の他の節で取り上げていない MathML 演算子をいくらか掲載する。

記号	入力	UCS4
′	′	U+2032
″	″	U+2033
‴	‴	U+2034
⁗	⁗	U+2057
°	°	U+00B0
†	†	U+2020
∗	∗	U+2217
¬	¬	U+00AC
∂	∂	U+2202
ⅆ	ⅆ	U+2146

記号	入力	UCS4
∇	∇	U+2207
∫	∫	U+222B
∬	∬	U+222C
∭	∭	U+222D
∮	∮	U+222E
∯	∯	U+222F
∰	∰	U+2230
∀	∀	U+2200
∃	∃	U+2203

度記号，プライム，二重プライムは，角度を表す度・分・秒に用いることができる。

23°12′40″

```
<mn>23</mn>
<mo>&deg;</mo>
<mn>12</mn>
<mo>&prime;</mo>
<mn>40</mn>
<mo>&Prime;</mo>
```

ⅆ（微分記号）は参考のため挙げた。Unicode 名は DOUBLE-STRUCK ITALIC SMALL D である。double struck については既に 11.1 節「mathvariant 属性」で述べた。

MathML について書かれたウェブサイトなどで，この記号を用いたサンプルをしばしば見かける。また，Unicode のコード表にも，この文字について「sometimes used for the differential」と書かれている。

これに従うなら，例えば以下のようになる。

$$\frac{ⅆ}{ⅆx}\int_a^x f(t)\,ⅆt = f(x)$$

しかし，筆者の狭い経験では，小文字アルファベットのディーを用いた微分記号としてはイタリック体の d とローマン体の d[59] しか見たことがない。

文字実体参照 ′（′）と ″（″）の命名の関係は，†（†）と ‡（‡, double dagger）に似ている。

3 重プライムの tprime は triple prime に，4 重プライムの qprime は quadruple prime に由来する。

13.10　その他の記号

数学記号は極めて多数あるが，本節では mi 要素に入れて使う若干の記号を取り挙げるにとどめる。

記号	入力	UCS4
∅	∅	U+2205
∞	∞	U+221E
ℏ	ℏ	U+210F

記号	入力	UCS4
ⅇ	ⅇ	U+2147
ⅈ	ⅈ	U+2148
ⅉ	ⅉ	U+2149

U+2205（∅）は空集合の記号である。Unicode のコードチャート（18.2 節）の例示字形ではこの表に掲げたごとく円に斜線の入ったデザインであり，直径記号（⌀ U+2300）と区別しづらい。もう一つの字形としてゼロに斜線の入

59) どちらがよいかはそれぞれに支持者がいる。

った ∅ もある[60]。いずれにせよ，ギリシア文字の ϕ を用いるのは適切ではない。

U+210F（$\hbar$）は換算プランク定数と呼ばれるもので，物理定数の一つであるプランク定数 h を 2π で割ったもの[61]。ちなみに，Unicode にはプランク定数のためのコードポイント U+210E PLANCK CONSTANT（h）が用意してあるが，使う必要はあるまい。

U+2147（e），U+2148（i），U+2149（j）は参考のため挙げた。

U+2147 は「ネイピア数」とか「自然対数の底（てい）」と呼ばれる数学定数を表すもので，MathML 仕様書にも `mi` 要素で表すものの例として出てくる。Unicode 名は「DOUBLE-STRUCK ITALIC SMALL E」である。

しかし，筆者の狭い経験では，小文字アルファベットのイーを用いたネイピア数の表記としてイタリック体の e とローマン体の e しか見たことがない[62]。

U+2148（i），U+2149（j）はいずれも虚数単位を表すもので，Unicode 名はそれぞれ「DOUBLE-STRUCK ITALIC SMALL I」「DOUBLE-STRUCK ITALIC SMALL J」である。U+2148（i）のほうは MathML 仕様書にも，先に挙げた e とともに `mi` 要素で表すものの例として出てくる。

恐らく数学・物理で最も普及している虚数単位の表記は普通の小文字イタリック体の i だろう。電気工学では，i が電流を表す変数に用いられることもあり，小文字イタリック体の j が好まれるようだ。小文字ローマン体の i が用いられることもある。筆者の狭い経験では，U+2148（i），U+2149（j）が実際に使われているところを見たことがない。

第 14 章　スペーシング

誤読を招かず，読みやすく美しい組版のためには適切なスペーシングが必要だ。多くの場合，スペーシングは自動的に行われるが，あらわに指定してやらなければならない場合もある。

14.1 `mspace`

その名のとおりスペースを空けるための MathML 要素が `mspace` だ。

以下のように使う。「p は 2 の n 乗（ただし n は整数）」という意味だ。

$$p = 2^n \quad (n \in \mathbf{Z})$$

```
<mi>p</mi>
<mo>=</mo>
<msup>
  <mn>2</mn>
  <mi>n</mi>
</msup>
<mspace width="1em" />
<mfenced>
  <mrow>
    <mi>n</mi>
    <mo>&in;</mo>
    <mi mathvariant="bold-italic">Z</mi>
  </mrow>
</mfenced>
```

これで 1 em の空きが出来る。

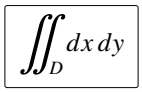
もう一つ例を挙げよう：

---

60) Cambria Math や Latin Modern Math といったフォントでは「ゼロに斜線」にデザインされている。本文書で用いている STIX Regular というフォント（16.1 節参照）は「円に斜線」である。STIX Two Math というフォントは両方のグリフを持っているが，既定のグリフは「円に斜線」である。

61) 日本語では「エイチ・バー」とか（ドイツ語風に）「ハー・バール」と読まれているようだ。

62) 定数をローマン体とするルールなら，ネイピア数 e も円周率 π も，次に出てくる虚数単位 i もローマン体となる。



```
<msub>
  <mo>&Int;</mo>
  <mi>D</mi>
</msub>
<mi>d</mi>
<mi>x</mi>
<mspace width="thinmathspace" />
<mi>d</mi>
<mi>y</mi>
```

「$dxdy$」が dx と dy からなることを，間に空きを入れてはっきりさせているわけである。

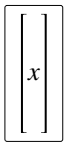
thinmathspace という値については後述する。

mspace には，幅だけでなく高さ（height）と深さ（depth）も指定できる。高さとはベースラインより上の大きさ，深さとはベースラインより下の大きさのことである。ベースラインは欧文書体のデザインの最も基本的な水平基準線であり，「A」などの文字はこの線に載るようデザインされる。

実用的な例ではないが，効果が分かりやすいよう根号に入れて示すと次のようになる。

```
<msqrt>
  <mn>2</mn>
  <mspace width="0.5em" height="3ex" depth="4ex" />
</msqrt>
```

width="0" の mspace を使えば，括弧類の大きさの制御に使える。



```
<mfenced open="[" close="]">
  <mrow>
    <mspace height="2.5em" />
    <mi>x</mi>
  </mrow>
</mfenced>
```

width は規定値が 0em なので書いていない。この例で，height しか指定していないにも関わらず深さ方向にも大きくなっているのは，17.5 節「symmetric」で述べるとおり，[ ] が symmetric という性質を持っているからだ。

width，height，depth に指定できるのは「長さ」であって，この例のような単位付きの数値のほか，以下の定数が書ける[63]。

定数名	値	定数名	値
veryverythinmathspace	1/18 em	negativeveryverythinmathspace	−1/18 em
verythinmathspace	2/18 em	negativeverythinmathspace	−2/18 em
thinmathspace	3/18 em	negativethinmathspace	−3/18 em
mediummathspace	4/18 em	negativemediummathspace	−4/18 em
thickmathspace	5/18 em	negativethickmathspace	−5/18 em
verythickmathspace	6/18 em	negativeverythickmathspace	−6/18 em
veryverythickmathspace	7/18 em	negativeveryverythickmathspace	−7/18 em

この表からも分かるとおり，負の値も指定できる。

指定に使える単位は 1.10 節「寸法単位」で述べた。

MathML 3 仕様書

3.2.7 Space <mspace/>

[63] 単位が 1/18 em になっているのが半端に感じられるかもしれないが，欧文タイポグラフィーではわりとポピュラーだ。2，3，6 などで割り切れ，活版印刷での作業性が良い。同一サイズの全ての活字の幅を 1/18 em の整数倍に設計することも行われた。

14.2　lspace，rspace 属性

演算子には lspace，rspace 属性を付けることができる[64]。これはその演算子の左側・右側の空き量をあらわに指定するものである。値には任意の長さが指定できる。

例えば整数論で出てくる「n を法として」という意味の $(\mathrm{mod}\ n)$ を組むには，

$(\mathrm{mod}\ n)$

```
<mfenced>
  <mrow>
    <mo lspace="0">mod</mo>
    <mi>n</mi>
  </mrow>
</mfenced>
```

とする。lspace 属性を付けないと以下のように，mod の前に空きが出来てしまう。

NG
$(\ \mathrm{mod}\ n)$

```
<mfenced>
  <mrow>
    <mo>mod</mo>
    <mi>n</mi>
  </mrow>
</mfenced>
```

「n を法として a と b は合同」は以下のように書ける。

$a \equiv b\ \ (\mathrm{mod}\ n)$

```
<mi>a</mi>
<mo>&equiv;</mo>
<mi>b</mi>
<mspace width="0.5em" />
<mfenced>
  <mrow>
    <mo lspace="0">mod</mo>
    <mi>n</mi>
  </mrow>
</mfenced>
```

しかし，この問題は lspace よりも演算子辞書のカスタマイズで解決するほうがよい。具体的な方法は 17.8.1 節「mod」で述べる。

$\sqrt{2}i$ は間違いではないが，i が根号の中に入った $\sqrt{2i}$ といくぶん紛らわしい。そこで以下のように空きを入れることもある。

$\sqrt{2}\,i$

```
<msqrt>
  <mn>2</mn>
</msqrt>
<mo rspace="thinmathspace">&it;</mo>
<mi>i</mi>
```

間に見えない乗算の演算子が入っているのでそれに右余白を付けた。もちろん左余白でも構わない。

空きの量は，こうでなければならないという決まりは無いが，文書全体で統一的に決めるべきだろう。

第 15 章　行分割

次のような場合，数式を複数行にわたって組むことになる。

(1)本文中でインラインの数式が行末にさしかかり，追い込み調整もできず，追い出し調整だと字間が開きすぎる場合[65]。

(2)ディスプレイ数式で，行長に収まらない場合。

---

64) mi，mrow，mfenced の類には付けられないことに注意。

65) 追い込み調整とは行末の半端物を行内に押し込め，欧文単語間や句読点・括弧類まわりなどの空きを詰めてジャスティファイを実現することを指す。追い出し調整は行末の半端物を次行に追い出し，文字間を空けてジャスティファイを実現することを指す。

(3)式の変形過程を縦に連ねたり，意図して式を分割して縦に配置するなどの場合。

このうち，(1) は math 要素を分割する（内容を複数の math 要素に分ける）ほか無い。AH Formatter は一つの math 要素を行末で自動的に分割したりしないからだ。行長が変われば分割を元に戻したり，別の位置で分割したりしなければならないことにもなる。このような事態の発生を抑えるためには，インラインであまり長い数式を書かぬほうがよい。

(2) は，何も指定しなければ AH Formatter が箇所を決めて自動分割することになるが，以下で述べるように，分割位置を指定したり，分割しやすさを指定したり，また分割後のインデントを指定することもできる。

(3) は，分割位置を指定して分割させるほか，目的に応じて行分割とは異なる方法で縦に並べていく方法もとりうる。

15.1 強制的に分割

この節では，一つの math 要素について，成り行きで行分割するのではなく，特定の位置で分割させる方法を取り上げる。

mo 要素に linebreak="newline" を指定すれば，その演算子で分割される。

$$a+b+c\\+d+e$$

```
<mi>a</mi>
<mo>+</mo>
<mi>b</mi>
<mo>+</mo>
<mi>c</mi>
<mo linebreak="newline">+</mo>
<mi>d</mi>
<mo>+</mo>
<mi>e</mi>
```

とくに指定しなければ当該演算子の〈前〉で分割するようになっているので，このような結果になる[66]。linebreakstyle 属性の値を before にすれば，演算子の前で分割され，after にすれば演算子の後で分割される（このほか，属性値には duplicate と infixlinebreakstyle もあるが，割愛）。

ところで，上に挙げた例で 2 行目以降を字下げ（インデント）したい場合，以下のように indentshift 属性を使う。

$$a+b+c\\\quad+d+e$$

```
<mi>a</mi>
<mo>+</mo>
<mi>b</mi>
<mo>+</mo>
<mi>c</mi>
<mo linebreak="newline" indentshift="1em">+</mo>
<mi>d</mi>
<mo>+</mo>
<mi>e</mi>
```

しかし，3 行以上にわたる場合，改行する mo 要素ごとに indentshift を付けるのは煩わしい。

実は indentshift などの属性は，mstyle 要素や math 要素に指定することもできる。その場合，その中に含まれる mo の indentshift の既定値となる。

$$a+b+c\\\quad+d+e\\\quad+f+g$$

```
<mstyle indentshift="1em">
  <mi>a</mi>
  <mo>+</mo>
  <mi>b</mi>
  <mo>+</mo>
  <mi>c</mi>
  <mo linebreak="newline">+</mo>
  <mi>d</mi>
  <mo>+</mo>
  <mi>e</mi>
  <mo linebreak="newline">+</mo>
  <mi>f</mi>
  <mo>+</mo>
  <mi>g</mi>
```

66) 演算子の前後どちらで分割するかは演算子辞書（第 17 章）によって既定値が決まっており，カンマ（,）など一部の演算子は〈後〉で分割することになっている。

```
</mstyle>
```

この例では，mstyle に指定したが，math 要素内で同一の値を用いるなら math に指定すればよい。

MathML 3 仕様書

3.2.5.2.2 Linebreaking attributes

15.2 式変形の等号を揃える

以下のような式変形で等号の位置を揃えるにはどうすればよいだろう。

$$\begin{aligned}|z|^2 &= z\overline{z} \\ &= (x+yi)(x-yi) \\ &= x^2+y^2\end{aligned}$$

mtable を使って 3 列の表の形で位置合わせする方法は，9.5.2 節で既に述べたが，ここでは別の方法を示す。

15.2.1 indentalign="id" による方法

一つの方法は，1 個目の <mo>=</mo> に id 属性を付けておき，2 個目，3 個目の <mo>=</mo> で行分割を指定する際，indenttarget 属性にその id を指定する方法だ。

簡単な数式で指定方法を見てみよう。

$$\begin{aligned}A &= B \\ &= C\end{aligned}$$

```
<mi>A</mi>
<mo id="equal_sign">=</mo>
<mi>B</mi>
<mo linebreak="newline" indentalign="id" indenttarget="equal_sign">=</mo>
<mi>C</mi>
```

indentalign="id" は，揃える先を id で指定するということを意味する。

この方法は非常に簡素で良いが，インデントだけのために**文書全体で**一意の id を割り振らなければならないという難点がある。

id を組織的に設計しなければならない。例えば全ての math 要素に一意の名前を割り振り，それに 1 番目のインデント位置という意味で _i1 といった文字列を付加する，といった工夫が必要になる。

いずれにしても，この問題は数式の使い回しを困難にする。MathML 表現を他の箇所からコピー＆ペーストしてきたり，別の場所で定義した MathML 表現をプログラム処理で埋め込むなどの場合に，id をどうにかしなければならない。

プログラム処理で id を自動的に付加する方法も検討に値しよう。

15.2.2 mphantom による方法

以下のような式展開を考えよう。

$$\begin{aligned}&(x+y)(x-y) \\ =\ &x^2-y^2\end{aligned}$$

2 行目の等号を 1 行目に揃えようとしても 1 行目には等号がないので indentalign="id" による方法は使えない。

このような場合に mphantom が使える。mphantom は〈その中身が実際には存在しないのにあたかも存在しているかのように空間を占有する〉ものだ[67]。以下のように使う。

$$\begin{aligned}A \\ =\ B\end{aligned}$$

```
<mphantom><mo>=</mo></mphantom>
<mi>A</mi>
<mo linebreak="newline">=</mo>
<mi>B</mi>
```

この例では，1 行目の mphantom の部分に，等号が存在しているかのように空間ができている。そのため，A と B

67) phantom [fǽntəm] は「幻影」「霊」などという意味。

の左端が揃うのだ。

mphantom には任意の部分式を入れることができる。しかし，演算子まわりの空きに関して少し注意しなければならない点がある。次の失敗例を見よう。

NG

$$\frac{x+y-z}{x\quad -z}$$

```
<mfrac>
  <mrow>
    <mi>x</mi> <mo>+</mo> <mi>y</mi> <mo>-</mo> <mi>z</mi>
  </mrow>
  <mrow>
    <mi>x</mi>
    <mphantom> <mo>+</mo> <mi>y</mi> </mphantom>
    <mo>-</mo> <mi>z</mi>
  </mrow>
</mfrac>
```

分母は分子の $+y$ の部分をファントム化しただけである。なのに分子・分母が揃っていない。なぜか？

それは mphantom の内容が「推定された mrow」（4.1 節）で包まれているからだ。mphantom の代わりに mrow を入れてみるとよく分かる。

$$\frac{x+y-z}{x+y-z}$$

```
<mfrac>
  <mrow>
    <mi>x</mi> <mo>+</mo> <mi>y</mi> <mo>-</mo> <mi>z</mi>
  </mrow>
  <mrow>
    <mi>x</mi>
    <mrow> <mo>+</mo> <mi>y</mi> </mrow>
    <mo>-</mo> <mi>z</mi>
  </mrow>
</mfrac>
```

y の前の + が，mrow の先頭にあることによって中置演算子でなく前置演算子とされてしまったのだ。

これを回避するにはあらわに form を与えて以下のようにすればよい。

$$\frac{x+y-z}{x\quad -z}$$

```
<mfrac>
  <mrow>
    <mi>x</mi> <mo>+</mo> <mi>y</mi> <mo>-</mo> <mi>z</mi>
  </mrow>
  <mrow>
    <mi>x</mi>
    <mphantom> <mo form="infix">+</mo> <mi>y</mi> </mphantom>
    <mo>-</mo> <mi>z</mi>
  </mrow>
</mfrac>
```

第 16 章　フォント

この章では，数式用フォントをどのように選ぶか，どのように指定するか，ということについて述べる。

16.1　STIX

一般論を述べる前に STIX というプロジェクトおよびその成果物であるフォントについて述べておきたい。STIX は Scientific and Technical Information Exchange の頭文字を取ったもので，科学技術文書のためのフォントを作ることを目的としている。

成果物であるフォントにはバージョン 1 系統と 2 系統があり，それぞれが以下のように二つのフォントファミリーからなる[68]。

[68] このほか，STIXGeneral というものがあったが，本文書では触れない。

系統	フォントファミリー	フォント
バージョン 1	STIX	STIX Regular
		STIX Bold
		STIX Italic
		STIX Bold Italic
	STIX Math	STIX Math Regular
バージョン 2	STIX Two	STIX Two Text Regular
		STIX Two Text Bold
		STIX Two Text Italic
		STIX Two Text Bold Italic
	STIX Two Math	STIX Two Math Regular

フォントファミリー（font family）とは，統一的にデザインされたフォントの一群であり，一つのフォントファミリーがウエイト（太さ）やスタイル（ローマン／イタリックなど）などの異なる複数のフォントで構成されていたりする。

STIX Math フォントファミリーには一つのフォントしかないが，実はこの STIX Math Regular の中にボールド，イタリック，ボールドイタリックのグリフ（字形データ）も含まれており，AH Formatter は MathML のレンダリングにおいてウエイト，スタイルに応じたグリフを呼び出す。STIX Two Math フォントファミリーについても同様である。

なお，単に「STIX」と書くと，プロジェクト名を指しているのか，その成果物であるフォントないしフォントファミリーの全般を指しているのか，上の表で「STIX」と書かれたフォントファミリーを指しているのか，STIX Regular フォントの略称なのか非常に紛らわしい。そこで，本文書ではそれぞれ「STIX プロジェクト」「STIX プロジェクトのフォント」「STIX プロジェクトのフォントファミリー」「STIX フォントファミリー」「STIX Regular フォント」と書いて区別することにする。

バージョン 1 系統の最新版は 1.1.1 で，2 系統の最新版は 2016 年 12 月にリリースされた 2.0.0 である（2017 年 7 月現在）。

AH Formatter V6.4 は，STIX フォントファミリーに最もよく調整されているので，現時点ではこれを用いるのが無難である。

STIX プロジェクトのフォントについてはこれ以上踏み込まない。以下のプロジェクトサイトやフォントに添付されている README，Release Documentation を参照されたい。

http://www.stixfonts.org/

16.2 フォント選び

前節では AH Formatter V6.4 で組むことを前提に STIX フォントファミリーを勧めておいたが，本節では一般論を述べる。

数式用フォントに何を選ぶかは非常に難しいが，極めて重要でもある。ここでは，以下の観点で考えてみる。

- 本文書体と整合するか。
- ファミリーが揃っているか。
- 識別性が高いか。
- 文字・記号が十分にあるか。

16.2.1 本文書体との整合性

数式部分とそうでない部分とでデザインの異なるフォントが使われていては，やはりおかしい。合わせるようにしたい。

16.2.2 フォントファミリー

数式用フォントファミリーにローマン体とイタリック体が両方必要なことは言うまでもないだろう。

ウエイト（太さ）も，通常のものとボールドとの二つが必要だ。零行列 **0** や複素数体 **C** を表すのにボールドを使ったり，ベクトルを表すのにボールドイタリックを使ったりする。

つまり，ファミリーに最低限必要なバリエーションは以下のようになる。

	ローマン体	**イタリック体**
通常ウエイト	Sample	*Sample*
ボールド	**Sample**	***Sample***

さきほどはウエイトの違いで記号の意味が変わる例を挙げたが，学習参考書などでは強調のためにウエイトの高いフォントを数式に使うことがある。また，見出しのようなボールドを用いた文字列中の数式はやはりそれに合わせたウエイトがふさわしい。

16.2.3 識別性

書体デザインにおける識別性とは，文字の見分けやすさだ。

例えば Gill Sans という書体は，大文字のアイ（I），小文字のエル（l），数字のイチ（1）が見分けづらい。

通常の文章であれば，わざと綴りを間違えて（アイを数字に，エルをアイに）

```
1llustration
```

などとしても，文脈の力で支障なく読める（多少違和感を感じるとしても）。

しかし，数式は冗長性が低いので，文脈では判断できないことも多く，また 1 文字の違いで別の内容になってしまう。例えば

```
2lm
```

は「2 ルーメン」だろうか「21 メートル」だろうか[69]。

一般的に言って，サンセリフ体は数式組版には向かない[70]。とりわけ，ジオメトリック・サンセリフに分類される，幾何学的なエレメントを持つ書体は向かない。

ではセリフ書体である Times New Roman を用いた次の例はどうだろう。

$$\lambda = \frac{v}{\nu}$$

これは屈折率 n の媒質中の光速（$v = c/n$）と光の振動数 ν と波長 λ の関係式である。

右辺の分子は媒質中の光速 v でラテン文字のブイ，分母は振動数 ν でギリシア文字のニューだが，全く区別がつかない。

---

69) 本来，2 lm のように，数値と単位記号の間には空きを入れるべきではある。なお，l（リットル）については「固有名詞に由来しない単位記号は小文字」の原則を破って L の表記も認められており，広く使われている

70) 20 世紀のサンセリフ体隆盛の一つの動機は「無駄な装飾の排除」であった。これは文字デザインの冗長性の低減を意味する。冗長性の低減は識別性の低下を招きうる。サンセリフ体がいけないのではなく，数式の組版に向かないのだ。なお，ここで問題にしているのは数式全体をサンセリフ体とすることであって，個々の記号（例えばテンソル記号）にサンセリフ体を用いることではない。

Times New Roman は数字のイチ（1）と小文字エル（l）の区別もつきにくい。

二つの書体を比較してみよう。

フォントファミリー	ブイ	ニュー	イチ	エル
Times New Roman	ν	ν	1	l
STIX	ν	ν	1	l

STIX フォントファミリーはラテン文字・数字については Times New Roman を元にしており，イチとエルの識別性はやはり低い[71]。しかし，ギリシア文字については全く異なるデザインとなっている。

16.2.4 文字・記号の豊富さ

数式には極めて多数の文字・記号を用いるので，文字・記号の豊富なフォントが必要になる。

まずギリシア文字について述べる。DTP の黎明期には，ギリシア文字はファミリーの中の別フォントの形で提供されていたが，OpenType フォントになって，一つのフォントにギリシア文字・キリル文字が含まれるようになった。

ギリシア語の組版には ὐ のようなアクセント記号の付いたギリシア文字も必要になるが（古典ギリシア語はとくに多数のグリフを要す），数式に使うギリシア文字は基本のアルファベットがあれば十分だ。ただし，同じファイでも φ と ϕ のように異体字のグリフが必要になることもある。

記号については，やはりよほど簡単な式でない限り，数式組版用に作られていないフォントでは足りない。STIX プロジェクトのフォントは科学技術文書用だけあって非常に多くの記号を含んでいる。

本節の要件により，フォントの選択肢は一気に減ることになる。

16.3 フォントの指定方法

この節では AH Formatter における `math` 要素のフォントの指定方法について述べる。AH Formatter は STIX フォントファミリーを既定の数式フォントとしているが，これを採用する場合でも指定方法は知っておきたい。

16.3.1 CSS によるフォント指定

スタイルシートで

```
body {
  font-family: "Some Font";
}
```

などとフォントを指定しても，これは `math` 要素には適用されない。`math` 要素のフォントはこれとは別に指定する必要がある。つまり

```
math {
  font-family: "Some Font";
}
```

などとする。

なお，数式フォントに既定の STIX フォントファミリーを採用する場合でも，数式中に日本語が出てくる可能性を考えて日本語フォントも指定しておくとよいだろう。日本語部分に IPAmjMincho（IPAmj 明朝）を使うなら，以下のように書く。

```
math {
```

[71] STIX Two Text および STIX Two Math ではこの点は改善されている。

```
  font-family: STIX, IPAmjMincho, serif;
}
```

このようにフォントファミリー名を複数列挙して指定した場合，文字ごとに先頭のフォントファミリーから順に調べてその文字のグリフを持つものが採用される。例えば「3」であれば STIX がグリフを持っているので STIX で組まれ，「個」であれば STIX はグリフを持たないが IPAmjMincho はグリフを持つので，IPAmjMincho で組まれる，といった具合だ。最後の serif は「総称ファミリー」（generic family）と呼ばれるものの一つで，実際のフォントの選択は組版ソフトに委ねられる。詳細は下記の「フォントの選択」を参照されたい。

http://www.antenna.co.jp/AHF/help/v64/ahf-tech.html#font-selection

フォントファミリー指定におけるこのルールは math 要素に限らないし，CSS 組版だけでなく XSL-FO 組版の場合にも適用される。

16.3.2 オプション設定ファイルによるフォント指定

math 要素のフォントについて，もう少し手の込んだ指定を行いたいときはオプション設定ファイルを用いる。このファイルについては既に 8.12.1 節「添字サイズ下限値の変更」で紹介した。

例えば，double-struck のみ他のフォントに差し替えたり，特定の文字だけ他のフォントに差し替えたり，といったことができる。

この節では，STIX フォントファミリーを基本としつつ，字形の気に入らない一部の記号について自作フォントで置き換える，といった状況を仮定する[72]。

仮に，STIX フォントファミリーの乗算記号（×）を取り替えたいとする。コードポイントは U+00D7 である。そして，好みのデザインの乗算記号を含んだフォント MyMath が出来たとする。オプション設定ファイルは以下のように書けばよい。

```
<?xml version="1.0" encoding="utf-8" ?>
<formatter-config>
  <mathml-settings>
    <variant-font
      fontfamily="STIX, IPAmjMincho, serif" mathvariant="normal">
      <font-entry fontfamily="MyMath" unicode-range="U+00D7" />
    </variant-font>
  </mathml-settings>
</formatter-config>
```

ルート要素は formatter-config であり，その中で，MathML に関する設定は mathml-settings 要素に書く。

MathML のフォントについては variant-font 要素に書く。この要素は，mathvariant ごとに用意する。上の例では normal，つまりイタリックやボールドなどではない普通の太さ（ウエイト）のローマン体についてのみの設定となっている。

variant-font には fontfamily 属性[73]で基本となるフォントファミリーを指定する。このように併記した場合に，どのフォントが適用されるかのルールは前節で述べた CSS の font-family プロパティーの場合と同じである。

さて，ここでのテーマである，文字ごとにフォントを指定するのは，その中に記述する font-entry である。この要素では，どの文字をどのフォントにするかを指定する。〈どの文字〉は unicode-range で指定する。属性値としては U+2057 のような形式で Unicode のコード番号（UCS4）を指定する。U+2032-2034 のようにハイフンでつないで範囲を指定することができ，またこれらをカンマで区切って複数併記することもできる。どのフォントにするかは

---

72) そのようなフォントをきちんと作るのは容易ではない。グリフのデザインだけでなく，メトリクスなど目に見えない情報も適切に設定されなければならない。しかし，本文書ではそこに立ち入らない。

73) この属性名は，CSS のプロパティー名 font-family と違ってハイフンが入らないことに注意。fontsize，fontweight，fontstyle 属性も同様。

fontfamily 属性で指定する。

以上の設定で，乗算記号のグリフが MyMath になる。

このようにして任意の文字に任意のフォントを適用させることができる。

なお，CSS で math 要素に対して font-family を指定してしまうと，オプション設定ファイルで行ったフォントの設定が上書きされてしまうので注意されたい。

また，mathml-settings の中で，同じ mathvariant 属性を持つ variant-font を二つ書くと，最後の指定だけが有効になる。よって，variant-font は mathvarint ごとにまとめなければならない。

16.3.3 STIX の「の」問題

STIX プロジェクトのフォントには一つ困った特徴がある。いくつかのフォントが，どういうわけか平仮名の「の」のグリフを持っているのだ[74]。字形は「の」である。

したがって，数式中に「の」が出てくると以下のようにおかしなことになる。

$$\text{登録率} = \frac{\text{登録者の数}}{\text{サービス人口}}$$

この問題に対処するには，前節で述べたのと同じようにすればよい。オプション設定ファイルの mathml-settings の部分だけを抜き出すと，次のようになる。

```
<mathml-settings>
  <variant-font
    fontfamily="STIX, IPAmjMincho, serif"
    mathvariant="normal">
    <font-entry fontfamily='IPAmjMincho' script="Hiragana" />
  </variant-font>
</mathml-settings>
```

unicode-range 属性で「の」のコードポイント（U+306E）を指定してもよいが，ここでは script 属性に Hiragana を指定して，すべての平仮名を IPAmjMincho にした。

script 属性は適用すべきスクリプト（用字系）を指定するもので，その値には ISO 15924 Codes for the representation of names of scripts で定められた文字列を使う。この規格ではすべてのスクリプトを 4 文字のアルファベットで表すが，平仮名の「Hira」に対する「Hiragana」のように通称が定義されているものもある。

いくつか例を挙げると，ラテン文字が Latn（Latin），ギリシア文字が Grek（Greek），キリル文字が Cyrl（Cyrillic），片仮名が Kana（Katakana），両仮名（平仮名＋片仮名）が Hrkt（Katakana_Or_Hiragana），漢字が Hani（Han），日本語（漢字＋平仮名＋片仮名）が Jpan，といった具合である（いずれも括弧内は通称）。「Kana」が両仮名でなく片仮名であることに注意されたい。

第 17 章　演算子辞書

この章では，これまで詳しくは述べなかった演算子の振る舞いについて説明する。

2.3 節「mo」などで見てきたとおり，MathML の演算子は mo 要素で表す。この要素の中には様々な記号類を入れることができるが，記号の種類によって望ましい振る舞いは異なるはずである。

いくらか例を挙げよう。

---

[74]「の」のグリフを持っているフォントは STIX Regular，STIX Math Regular，STIX Two Math Regular。

- = のような関係演算子や × のような二項演算子の前後は少し空きを入れるが，プライム ′ の前や括弧類の内側・外側には空きを入れない。
- 同じプラス記号 + でも，二項演算子としてのプラスは前後にちょっとした空きを入れるが，単項演算子としてのプラス記号の場合は空きを抑える。
- 積分記号 $\int$ や総和記号 $\sum$ はディスプレイスタイルで組まれると通常より大きくなる。
- 総和記号はインラインで組まれたとき，$\sum_{i=1}^{n}$ のようにアンダースクリプト，オーバースクリプトがそれぞれ下付き，上付きの位置に変わる。
- 式の上に載せた矢印 → は本体に合わせて長さがいくらでも伸びるし，括弧類も中身に合わせて大きさが変わる。

各演算子がどのような振る舞いをするかは，演算子辞書（operator dictionary）というものに定義されている。

MathML 仕様書の付録 C.4「Operator dictionary entries」には参考として演算子辞書の表が掲載されている。ただし，この表は non-normative である。つまり仕様として規定されたものではない。

AH Formatter は付録 C.4 の表に若干の記号を追加したものを既定の演算子辞書としている。オプション設定ファイルを用いれば，これを部分的に上書きする形でカスタマイズできる。

以下では上に挙げた例を元に演算子の振る舞いについて説明していこう。

MathML 3 仕様書

C Operator Dictionary (Non-Normative)

17.1　フォーム：前置・中置・後置

プラス記号の例は，同じ演算子が文脈によって異なる扱いを受けることを示している。演算子は位置関係により，前置演算子，中置演算子，後置演算子のいずれかとみなされる。この違いをフォーム（form）という。

フォームは自動的に決定されるが，mo 要素の form 属性であらわに指定することもできる。値は，前置が prefix，中置が infix，後置が postfix である。

form 属性が与えられていない場合は mrow 内の位置関係で決まる。次の二つの例を見よう。

$+x$　`<mo>+</mo><mi>x</mi>`

$x+y$　`<mi>x</mi><mo>+</mo><mi>y</mi>`

いずれも，全体が「推定された mrow」で包まれていることに注意してほしい。

最初の例では，`<mo>+</mo>` が mrow 内の先頭にあるため，前置演算子（prefix）であると判断される。したがって，`<mo form="prefix">+</mo>` と指定したのと同じ結果になる。

既定の演算子辞書では，前置演算子の + は以下のようになっている[75]。

演算子	フォーム	lspace	rspace	プロパティー
+	prefix	0	1	

この表で，真ん中の罫線が太線になっているのは，ここより左の値（演算子の種類とフォーム）の組合せで右の値が決まることを意味している。

lspace が 0 で，rspace が 1 となっているが，単位は 1/18 em である。結果として，$+x$ の + と x の間には 1/18 em の空きが出来ることになる。

一方，2 番目の例では，`<mo>+</mo>` は中置演算子（infix）であると判断される。

既定の演算子辞書では以下のようになっている。

---

[75] 演算子辞書で定義される演算子の属性として priority というものもあるのだが，AH Formatter はこの値を参照していないので，本文書では触れない。

演算子	フォーム	lspace	rspace	プロパティー
+	infix	4	4	

よって，$x+y$ の + の前後には 4/18 em の空きが出来る。

後置演算子には二通りある。一つは受け（閉じ）の括弧類や，階乗

$n!$

```
<mi>n</mi><mo>!</mo>
```

の「!」のようにまさしく後置されるもの。

もう一つは単一の演算子が添字（下付き，上付き，アンダースクリプト，オーバースクリプト）の位置にあるものである。普通の意味で「後置」ではないが，演算子辞書を参照するときは postfix として扱われる。

これら以外の場合は基本的に infix である。例外の一つは

$U \otimes' V$

```
<mi>U</mi>
<msup>
  <mo>&otimes;</mo>
  <mo>&prime;</mo>
</msup>
<mi>V</mi>
```

における $\otimes'$ のような装飾された演算子（embellished operator）の場合だ。（この例では）マークアップ上は mo 要素でなく msub 要素なのだが，第一引数が $\otimes$ であり，**期待どおり** $\otimes'$ の全体が infix な演算子として扱われる。しかし，本文書では装飾された演算子についてこれ以上は述べない。詳細は MathML 仕様書を参照されたい。

演算子が演算子辞書に無かったり，載ってはいても別のフォームしかなかったりした場合については 17.4 節で述べる。

演算子前後の空きの決定について，一つ注意しなければならないことがある。

例として偏微分記号 ∂ を取り上げよう。既定の演算子辞書では以下のように定義されている。

演算子	フォーム	lspace	rspace	プロパティー
∂	prefix	2	1	

これによると，prefix のときに左に空きがあるのだ[76]。ところが，以下の組版例を見てみよう。

$\frac{\partial x}{\partial u}$

```
<mfrac>
  <mrow><mo>&part;</mo><mi>x</mi></mrow>
  <mrow><mo>&part;</mo><mi>u</mi></mrow>
</mfrac>
```

この偏微分記号は二つとも prefix だが，左に空きができているようには見えない。いや，もちろん空きは出来てほしくないのだが。

MathML 仕様書には書かれていない動作だが，AH Formatter は以下のような場合の空きを抑制する（ゼロにする）ようになっている。

- math 内の先頭の lspace，末尾の rspace
- mtd 内の先頭の lspace，末尾の rspace
- msrow 内の先頭の lspace，末尾の rspace
- munder の第二引数（添字）内の先頭の lspace，末尾の rspace
- mover の第二引数（添字）内の先頭の lspace，末尾の rspace
- mfrac の第一引数（分子）・第二引数（分母）内の先頭の lspace，末尾の rspace
- mfenced 内の先頭の lspace，末尾の rspace

次節以降，いくつかの重要なプロパティーについて見ていく。取り上げなかったプロパティー fence と separator

[76] なぜそのようになっているのか，よく分からない。

については MathML 仕様書を参照されたい。

MathML 3 仕様書

3.2.5.7 Detailed rendering rules for <mo> elements
3.2.5.7.2 Default value of the form attribute
3.2.5.7.3 Exception for embellished operators

17.2 largeop

総和記号 ∑ などは，ディスプレイスタイルのときに通常よりも大きく描画される。これは，largeop 属性の既定値が演算子辞書において true と定義されているからである。

largeop 属性は mo 要素にあらわに書くこともできる。false にすればディスプレイスタイルでも大きくならない。

17.3 movablelimits

既に 1.7 節「ディスプレイスタイルとインラインスタイル」で見たように，総和記号などの上下に置かれるオーバースクリプト，アンダースクリプトは，インラインスタイルにしたときに上付き，下付きの位置に移る。こうなるのは演算子辞書で movablelimits が true と定義された演算子である。

17.4 stretchy

括弧類の大きさが中身に合わせて変わることは既に見た。

$\left(\frac{1}{2}\right)$

```
<mfenced>
  <mfrac><mn>1</mn><mn>2</mn></mfrac>
</mfenced>
```

また，矢印（→）や上に被せるパーレン（⏜）をオーバースクリプトにすると，本体に合わせて長さが伸びる。

$\overparen{AB}$

```
<mover>
  <mrow><mi>A</mi><mi>B</mi></mrow>
  <mo>&OverParenthesis;</mo>
</mover>
```

$\overrightarrow{PQ}$

```
<mover>
  <mrow><mi>P</mi><mi>Q</mi></mrow>
  <mo>&rarr;</mo>
</mover>
```

このような性質を持つことを，stretchy[77] である，という。これらの演算子は

演算子	フォーム	lspace	rspace	プロパティー
(	prefix	0	0	fence, stretchy, symmetric
)	postfix	0	0	fence, stretchy, symmetric
⏜	postfix	0	0	stretchy, accent
→	infix	5	5	stretchy, accent

と定義されている。

普通のパーレンや上に被せるパーレンが stretchy と定義されていることが分かる。

では矢印はどうか。実は postfix の矢印は既定の辞書に無い。載っているのは前掲のごとく infix だけである。この場合，仕様書 3.2.5.7.2 節によれば他のフォームの値が使われることになっている。他のフォームを探索する優先順位は infix，postfix，prefix となっている。よって infix の値が適用される。だから stretchy かつ accent というわけだ。

77) 伸縮性がある，の意。

演算子辞書にどのフォームも載っていない演算子の振る舞いについては後述する。

MathML 3 仕様書

3.2.5.8 Stretching of operators, fences and accents
3.2.5.7.2 Default value of the form attribute

17.5 symmetric

次の式の | | がどのように伸びているかに注目していただきたい。

$$\left| x + \frac{1}{y + \frac{1}{2}} \right|$$

```
<mfenced open="|" close="|">
  <mrow>
    <mi>x</mi>
    <mo>+</mo>
    <mfrac>
      <mn>1</mn>
      <mrow>
        <mi>y</mi>
        <mo>+</mo>
        <mfrac><mn>1</mn><mn>2</mn></mfrac>
      </mrow>
    </mfrac>
  </mrow>
</mfenced>
```

よく見ると上側に少し空きがある。これは，分母に合わせて下に伸びたぶん，上にも伸びたからである。このように上下方向に対称的に伸びる演算子は symmetric であるという。絶対値記号や括弧類などの囲み記号は symmetric が true と定義されている。

17.6 accent

オーバースクリプトやアンダースクリプトがアクセントであることがどういうことか，については 8.9 節「アクセント記号」で述べた。すなわち，文字サイズは小さくならず，本体の傾斜に応じた水平位置に配置されるのであった。

MathML 仕様書では垂直位置についても触れられていて，アクセントにしないと $\hat{x}$ のようにハット記号が離れてしまうという組版例が掲載されているが，AH Formatter ではこのように離れたりはしない。

演算子辞書で accent と定義された演算子は，accent 属性をあらわに書かなくともアクセントとして扱われる。

17.7 演算子の各種属性の既定値

演算子辞書にどのフォームも載っていない場合は仕様書 3.2.5.2.1 節に示された既定値が適用される。

lspace と rspace の既定値はいずれも thickmathspace（5/18 em）である。この値は infix の = などと同じで，infix の + など（4/18 em）よりわずかに大きい。

試しに辞書に無い「★」を演算子にしてみると，以下のようになる。

$$F \star G$$

```
<mi>F</mi>
<mo>★</mo>
<mi>G</mi>
```

fence，separator，stretchy，symmetric，largeop，movablelimits，accent の既定値は全て false である。

accent が false であるため，辞書に無い「★」をオーバースクリプトにすると以下のように小さくなる。

$$\overset{\star}{X}$$

```
<mover>
  <mi>X</mi>
  <mo>★</mo>
</mover>
```

★の位置が左に寄って見えるのは，本体（X）がイタリック体で，右に傾斜していることを無視して配置されるからである。

あらわに accent="true" を指定した場合と比べてみよう。

```
<mover>
  <mi>X</mi>
  <mo accent="true">★</mo>
</mover>
```

MathML 3 仕様書

3.2.5.2.1 Dictionary-based attributes

17.8　演算子辞書のカスタマイズ

既定の演算子辞書に不備や不足があった場合，オプション設定ファイルを用いて簡単に辞書を修正することができる。

17.8.1　mod

手始めに，14.2 節で述べた，「n を法として」という意味の「(mod n)」に出てくる mod を演算子辞書に追加してみよう。

問題は，mod が演算子辞書にないため，あらわに `lspace="0"` を指定しなければ以下のように mod の前に余計な空きが出来ることであった。

NG

$(\mathrm{mod}\ n)$

```
<mfenced>
  <mrow>
    <mo>mod</mo>
    <mi>n</mi>
  </mrow>
</mfenced>
```

`lspace` の既定値 `thickmathspace`（5/18 em）が適用されるためである。

そこで，mod が prefix のときの `lspace` を `0` とし，`rspace` は `4` としてみよう。オプション設定ファイルは以下のようになる。

```
<?xml version="1.0" encoding="utf-8" ?>
<formatter-config>
  <mathml-settings>
    <operator-dictionary>
      <entry operator="mod" form="prefix" lspace="0" rspace="4" />
    </operator-dictionary>
  </mathml-settings>
</formatter-config>
```

簡単だ。これはオプション設定ファイルの抜粋ではなく全体である。要するに `mathml-settings` 要素の中に `operator-dictionary` 要素を設け，その中に `entry` 要素を好きなだけ並べればいい。

既定の演算子辞書に無い演算子を追加する場合も，既定の演算子辞書に登録されている演算子の定義を上書きする場合も，書き方は変わらない。

ところで，演算子 mod は n mod m のような中置演算子の形でも使われる。n を m で割った余りの意だ。しかし，上記のようなカスタマイズを行うと，以下のように組まれてしまう。

$n\mathrm{mod}\ m$

```
<mi>n</mi> <mo>mod</mo> <mi>m</mi>
```

なぜなら，`infix` の mod が演算子辞書に無いため，`prefix` の設定値が適用されてしまうためだ。

これを防ぐには，さきほどの定義に

```
<entry operator="mod" form="infix" lspace="5" rspace="5" />
```

を付け加えればよい。このように，演算子辞書をカスタマイズする際は，あるフォームについての設定値が他のフ

ォームにも適用されてしまう可能性を常に考えておかなければならない。

なお，マイナス記号やプライムをカスタマイズしたい場合，operator 属性にハイフンやアポストロフィーを入れてもダメである。オプション設定ファイルではそのような置き換えはなされない。また，− とか ′ といった文字実体参照も使えないことに注意しよう。オプション設定ファイルはただの XML ファイルなので，基本の五つの文字実体参照しか使えないのだ。結局，文字そのものを書くか，− とか ′ といった数値文字参照を書くしかない。

17.8.2 微分の d

ここでは，以下のような微分積分学の数式に現れる，いわゆる「微分の d」を取り上げる。

$$\frac{dy}{dx}, \quad \int f(x)dx$$

微分の d はイタリック体にする流儀とローマン体にする流儀があるが，本節ではそこに頓着しない。必要に応じて mathvariant 属性で制御すればいい。

最初の問題は，果たして微分の *d* は MathML の演算子なのか，ということである。仕様書にはなぜか微分・積分の例が一切出てこないため例から判断することができない。

数学的意味においても，実は *d* には複数の捉え方がある。

高校数学では *dx* における *d* と *x* は不可分であり，〈*d* と *x* による何か〉ではない。しかも *dx* 単体では数学的実体を表してはおらず，上記のような微分や積分の式を構成して初めて意味をなす。

ところが微分幾何学では，*dx* は座標関数 *x* に微分作用素 *d* を作用させたもので，1 次微分形式（1-form）と呼ばれる数学的実体である。

後者の立場では，*d* は正弦関数 sin のような識別名であると考えられるかもしれない。

しかし，数学的意味をどう捉えるにせよ，ここ以降，*d* は MathML の演算子であると考えることにする。理由の一つは，*d* に役割の似た偏微分記号 ∂ やナブラ ∇，さらには，ずばり微分記号に使うためと思われる「double-struck italic small d」（ⅆ）までもが演算子辞書に登録されていること。もう一つは，演算子として扱うことで，スペーシングの制御がやりやすいことである。

前置きが長くなったが，カスタマイズを始めよう。「d」は既定の演算子辞書に無い。

ここ以降，マークアップ例を簡素に見せたいのでローマン体の d を採用することにする。

まず <mo>d</mo><mi>x</mi> というマークアップにおいて，d の前後に空きがあってはならないから，prefix の場合の lspace，rspace は 0 である。

次に，以下のような式において，d の左は少し空いてほしい（空けない流儀もある）。

$$\int x^2 \mathrm{d}x$$

これは infix で lspace="2" などとすればいいと思う。

したがって，オプション設定ファイルの operator-dictionary 要素の下に以下のように書けばよい。

```
<entry operator="d" form="prefix" lspace="0" rspace="0" />
<entry operator="d" form="infix"  lspace="2" rspace="0" />
```

このようにすれば，以下のようなマークアップで期待どおりのスペーシングになる。

$$\mathrm{d}x \wedge \mathrm{d}y = -\mathrm{d}y \wedge \mathrm{d}x$$

```
<mrow><mo>d</mo><mi>x</mi></mrow>
<mo>&and;</mo>
<mrow><mo>d</mo><mi>y</mi></mrow>
<mo>=</mo>
<mrow>
  <mo>-</mo>
  <mrow><mo>d</mo><mi>y</mi></mrow>
  <mo>&and;</mo>
```

```
    <mrow><mo>d</mo><mi>x</mi></mrow>
  </mrow>
```

$\mathrm{d}f = f_x \mathrm{d}x + f_y \mathrm{d}y$

```
<mo>d</mo><mi>f</mi>
<mo>=</mo>
<msub><mi>f</mi><mi>x</mi></msub>
<mo>d</mo><mi>x</mi>
<mo>+</mo>
<msub><mi>f</mi><mi>y</mi></msub>
<mo>d</mo><mi>y</mi>
```

偏微分記号 ∂，差分記号 Δ，変分記号 δ なども同じように考えればいいだろう。

17.8.3 絶対値記号

既定の演算子辞書によれば，prefix や postfix の「|」の前後に空きはない。

既定の定義を見てみよう。ついでにノルム || || の表記に使う記号も併記する。

演算子	フォーム	lspace	rspace	プロパティー
|	prefix	0	0	fence, stretchy, symmetric
|	postfix	0	0	fence, stretchy, symmetric
‖	prefix	0	0	fence, stretchy, symmetric
‖	postfix	0	0	fence, stretchy, symmetric

そのため，書体デザインにもよるが，$|X|$ や $\|\boldsymbol{v}\|$ の見た目はややキツい。

ここでは，それぞれについて，内側に 2/18 em（verythinmathspace）の空きを入れてみよう。以下のように定義すればよい。

```
<entry operator="|" form="prefix"  lspace="0" rspace="2" />
<entry operator="|" form="postfix" lspace="2" rspace="0" />
<entry operator="||" form="prefix"  lspace="0" rspace="2" />
<entry operator="||" form="postfix" lspace="2" rspace="0" />
```

このように定義すれば，以下のように内側にほどほどの空きが出来る。

$|X|$

```
<mfenced open="|" close="|">
  <mi>X</mi>
</mfenced>
```

$\|\boldsymbol{v}\|$

```
<mfenced open="||" close="||">
  <mi mathvariant="bold-italic">v</mi>
</mfenced>
```

17.8.4 微分のドット表記

8.9 節「アクセント記号」において，$\dot{x}$ のような記号を取り上げた。以下のようなマークアップであった。

$\dot{x}$

```
<mover accent="true">
  <mi>x</mi>
  <mo>.</mo>
</mover>
```

$\ddot{x}$

```
<mover accent="true">
  <mi>x</mi>
  <mo>..</mo>
</mover>
```

mover 要素に付けられた accent="true" は，上に載せる記号をアクセントとして扱うために必要であった。

演算子辞書で . や .. が accent と定義されていれば，accent="true" は不要になる[78]。

[78] 既定の演算子辞書には . も .. もあるが，アクセントとして使うことは想定されていないようだ。

以下のように定義すればいい。

```
<entry operator="." form="postfix" lspace="0" rspace="0" accent="true" />
<entry operator=".." form="postfix" lspace="0" rspace="0" accent="true" />
```

なお，使う記号としてはピリオドよりも U+02D9 DOT ABOVE と U+00A8 DIAERESIS を用いるほうがよいのかもしれない。これらは既定の演算子辞書において，postfix で accent となるよう定義されている。

$\dot{x}$

```
<mover>
  <mi>x</mi>
  <mo>&#x02D9;</mo>
</mover>
```

$\ddot{x}$

```
<mover>
  <mi>x</mi>
  <mo>&#x00A8;</mo>
</mover>
```

この場合，ドットの大きさなどデザインが 1 点と 2 点とで釣り合うかどうか，使用するフォントで確認しておかなくてはならない。

ピリオドを用いる利点は，少なくともドットの大きさは揃う，ということと，何より覚えやすいことである。

17.8.5 倍率の×

光学系の倍率を表記するために「10×」とか「×10」といった表記が用いられる。

しかし，既定の演算子辞書では，×（U+00D7）は infix しか定義されていないため，素直にマークアップすると以下のように空きが出来すぎてしまう。

NG $10\times$

```
<mn>10</mn><mo>&times;</mo>
```

NG $\times 10$

```
<mo>&times;</mo><mn>10</mn>
```

以下のような定義を追加すればいいだろう。

```
<entry operator="&#x00D7;" form="prefix"  lspace="0" rspace="1" />
<entry operator="&#x00D7;" form="postfix" lspace="1" rspace="0" />
```

第 18 章　参考文献・参考サイト

18.1 MathML

MathML についての日本語情報源は少ない。

[1] MathML 3 仕様書（英語）
https://www.w3.org/TR/MathML3/
当然ながら最も確実・詳細かつ網羅的な資料。しかし図版や例が少ない。

[2] Mozilla Developer Network (MDN)
https://developer.mozilla.org/ja/
Mozilla 財団が提供するウェブ技術に関する多言語情報源。MathML についてのページもある。日本語への翻訳はまだ途上。Firefox が実装していない MathML 要素については記事が無い場合もある。アカウントを作ってログインすれば誰でも編集に参加できるので，多少とも英語の読める方は翻訳に参加してほしい。

18.2　Unicode

数式を組むためには多数の記号を知らなければならない。本文書に掲載した記号は，そのほんの一部に過ぎない。

[3] Unicode 8.0 Character Code Charts
http://www.unicode.org/charts/
Unicode コンソーシアムによるコードチャート。文字・記号を調べるにはまずここから。「General Punctuation」といった分類から辿ったり，検索窓にコード番号を入力してチャートを検索したりできる。
[4] HTML5 > 8 The HTML syntax > 8.5 Named character references
https://www.w3.org/TR/html5/syntax.html#named-character-references
文字実体参照一覧。

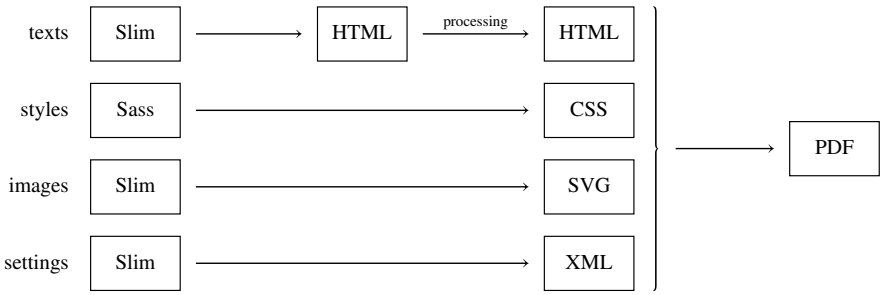
18.3　AH Formatter

[5] MathML 仕様の実装状況
http://www.antenna.co.jp/AHF/help/v64/ahf-mathml3.html
[6] グラフィクス → MathML
http://www.antenna.co.jp/AHF/help/v64/ahf-gra.html#MathML
[7] オプション設定ファイル → MathML の設定
http://www.antenna.co.jp/AHF/help/v64/ahf-optset.html#mathml-settings

第 19 章　制作ノート

最後に，本文書の制作にあたっての技術的側面について触れておきたい。

この文書は，AH Formatter の CSS 組版機能を用い，以下のようなフローで制作されている[79]。



筆者が実際に記述したのは左端に描かれた Slim 形式や Sass 形式のファイルである。これを Ruby スクリプトで HTML，CSS，SVG，XML ファイルに変換する。HTML に関しては，さらに Ruby スクリプトで加工を施している（後述）。

出来上がった HTML，CSS，SVG，XML から PDF を作成するのはもちろん AH Formatter の役割だ。

以下，それぞれについて簡単に説明しておこう。

19.1　テキスト

テキストは Slim という形式で書いた。Slim はもともとウェブアプリケーションの HTML を生成するためのテンプレートシステムとして開発されたもので，HTML や XML を簡潔に記述することができる。

Slim の最大の特徴は，要素の階層構造をインデントで表すことにある。例えば

---

[79] お気づきのとおり，この図は MathML のみで記述されている。

```
header
  h1#title MathML
  div.subtitle writing math
```

と書けば

```
<header><h1 id="title">MathML</h1><div class="subtitle">writing math</div></header>
```

に変換される。属性のうち，id と class だけはこのように特別な記法が用意されている。

Slim を使えば，例えば $\sqrt{2x+1}$ といった式は

```
math
  msqrt
    mn 2
    mo &it;
    mi x
    mo +
    mn 1
```

と書ける。縦長になることを嫌うなら，部分的にタグをあらわに書いて

```
math
  msqrt <mn>2</mn><mo>&it;</mo><mi>x</mi><mo>+</mo><mn>1</mn>
```

とすることもできる。

Slim は，ある要素の子要素が一つだけの場合，コロン＋スペースで区切って同一行に記述することもできるので，例えば $(x+y)$ といった式は

```
math
  mfenced
    mrow
      mi x
      mo +
      mi y
```

と書くほかに

```
math: mfenced: mrow
  mi x
  mo +
  mi y
```

とも書ける。

HTML を記述するのに Slim を用いたのは，もちろんマークアップを楽にするためだ。しかし，Markdown，Textile や各種 Wiki 記法など，簡易マークアップ言語が星の数ほどあるのに，どうして Slim なのか。理由は次のとおり[80]。

- どんな HTML も書ける
- Ruby コードの埋め込みができる
- シンプル
- 難しくない

どんな HTML も書けるということは大きい。箇条書きが入れ子にできなかったり，表組のセル結合や罫線が自由にならなかったり，好きなクラス属性を付けることができなかったり，MathML が書けなかったり（！）しては困るのだ。

---

[80] あくまで筆者にとっても利点である。読者に押しつけるつもりは無い。

Ruby のコードが埋込めるので，手で書けばうんざりするような記号の表も，ループで簡潔に書くことができた。組版見本とその MathML マークアップを対で表示した箇所が多数あったが，これも MathML だけ与えれば自動生成する。ケアレスミスで組版見本とマークアップが食い違っているなんてことにならないのだ[81]。

Slim は簡単だろうか？　基本は驚くほど簡単だ。本格的に使い始めると，ちょっとつまづくところも出てくるが，難しすぎると言うことはない。

一方，単語を   で囲むのに，Markdown なら

```
MathML is *not* difficult.
```

と書けるところが，Slim だと

```
p MathML is <em>not</em> difficult.
```

のようにタグをそのまま書くか，

```
p
  | MathML is
  em<> not
  | difficult.
```

のように煩雑な書き方をしなければならない[82]。

インライン要素は楽にならないのだ。しかし，まとまった分量の文書を Slim で書いてみて，それでいいのだと確信した。それより任意の HTML が書ける利点のほうが大きい。

19.2 スタイルシート

CSS は Sass という形式で書いた。インデントでセレクターの階層構造を表し，簡潔に記述できる。変数や計算式なども使えるので，うまく書けば保守性のよいスタイルシートが出来る。

Sass ももともとはウェブアプリケーションの世界で使われ，広まったものだ。印刷物のスタイルを書くのにも向いていると思う。

なお，広い意味の Sass フォーマットは，第一世代のフォーマット（狭義の Sass）と，第二世代のフォーマット（SCSS: Sassy CSS）の二つがある。前者の利点はより簡潔であること，後者の利点は CSS の完全上位互換となっていることだ。筆者は前者を用いている。

19.3 オプション設定ファイル

本文書では，AH Formatter 用のオプション設定ファイルも Slim で書いた。概ねこんな具合だ。

```
doctype xml
formatter-config
  mathml-settings scriptminsize="8Q"
    variant-font mathvariant="normal" fontfamily="STIX, IPAmjMincho, serif"
      font-entry fontfamily='IPAmjMincho' script="Hiragana"
```

19.4 画像

本文書には，画像はほとんど出てこないが，箇条書きの記号は SVG 画像である。SVG は XML なので，これも

---

[81] 実は若干，自動生成でない箇所がある。演算子辞書をカスタマイズしたらこうなるという組版例だ。演算子辞書を局所的に切り替えることはできないので，マークアップと組版結果を別々に記述した。

[82] em の後ろに付いている < と > は，それぞれ em 要素の前と後ろにスペースを設けることを意味している。

Slim で記述することができる。

例えば，「○」という記号は以下のように記述している。

```
svg[xmlns="http://www.w3.org/2000/svg" viewBox="0 0 1em 1em"
  width="1em" height="1em" preserveAspectRatio="xMinYMin meet"]
  circle[cx="0.5em" cy="0.5em" r="0.18em"
  stroke="black" fill="none" stroke-width="0.05em"]
```

19.5 HTML の加工

Slim から変換して出来た HTML は，組版に回す前に Ruby スクリプトで加工している。

具体的には，節番号などの相互参照の処理，目次の生成，柱の生成などである。

例えば章番号を参照するところは，Slim の原稿に

```
…<ref name="sec_summation" />節…
```

などと書いている。これは HTML に変換してもそのまま生きる。しかし，もちろん HTML に `ref` などという要素は無い。

一方，参照される節のほうは，Slim の原稿に

```
h3#sec_summation 総和・総積
```

と書いている。これが HTML になると

```
<h3 id="sec_summation">総和・総積</h3>
```

となる。

出来た HTML の加工は次のように行われる。

まず，`h2`，`h3` などの見出し要素が抽出され，それらの章・節番号が決定される。そして，先ほどの見出しは

```
<h3 id="sec_summation"><span class="sec-number">8.5</span>総和・総積</h3>
```

のように変換される。

一方，参照側は

```
…<span class="ref ref-sec" data-name="sec_summation">8.5</span>節…
```

と変換される。

このような HTML の後加工は極めて有用で，例えば原稿では単一の要素で記述したものを，複雑な要素の組合せに変換する，といった使い方もできるだろう。

19.6 PDF 作成

端末でコマンドを打てば上に述べた全ての変換が行われ，最後に AH Formatter に渡されて PDF が生成され，それが開くようになっている。

WYSIWYG でない編集環境では，修正結果がなるべく少ない手間で確認できることが重要だ。