AZmath パッケージ

@monaqa

目次

1. AZmath パッケージの概要

AZmath パッケージは，SATYSFI に豊富な math command を提供するパッケージです．現在は以下のようなコマンドを提供しています．

- 数式環境 (`+gather`, `+align`, etc.)
- アクセント (`\hat`, `\tilde`, etc.)
- 行列 (`\matrix`, `\pmatrix`, etc.)
- 括弧 (`\p`, `\pb`, etc.)

2. 数式環境

LaTeX では、equation や amsmath パッケージの gather, align といった環境 (environment) を用いて別行立て数式を記述します。SATySFi は LaTeX と異なり、本来「環境」と呼ぶべき概念はありませんが、ここでは \alpha や \sum など数式内で用いるコマンドと区別するため、インラインテキストまたはブロックテキストから別行立て数式に入るためのコマンド及び、それに類するコマンドのことを便宜上「数式環境」と呼ぶことにします。

SATySFi に標準で用意されている math パッケージでは、別行立ての数式を実現するために以下のコマンドが用意されています。

- +math: 通常の別行立て数式
- +math-list: 複数の数式を別行立てで横に並べる（長くなると改行）
- +align: 複数の数式を揃えながら並べる (LaTeX の align 環境と alignat 環境を兼ねたようなコマンド)
- \eqn: +eqn のインライン版
- \math-list: +math-list のインライン版
- \align: +align のインライン版

それに対し、AZmath パッケージでは以下のコマンドが定義されています。

- +eqn: 通常の別行立て数式
- +gather: 複数の数式を縦に並べる。LaTeX の gather 環境に近い。
- +align: 複数の数式を、 所定の位置で揃えて縦に並べる。LaTeX の align 環境に近い。
- +alignat: 複数の数式を、 所定の位置で揃えて縦に並べる。LaTeX の alignat 環境に近い。
- \eqn: +eqn のインライン版
- \gather: +gather のインライン版
- \align: +align のインライン版
- \alignat: +alignat のインライン版
- \aligned: +align と似たような揃え方を数式の中で行う。LaTeX の aligned 環境に近い。
- \cases: 数式の中で場合分けを作る。LaTeX の cases 環境に近い。

LaTeX に馴染みのある人に馴染みやすいよう、 命名規則は LaTeX のものを踏襲しています。

AZmath パッケージでは、標準の math パッケージと比較して以下のような機能が追加されています。

- 自動連番のタグを付与することができる。
- タグにラベルを付与し、\ref コマンドなどを用いて参照することが出来る。
- 複数行からなる別行立て数式（+gather, +align など）の 1 行 1 行にタグを付けることができる。タグ付けを手動で抑制することも出来る。
- 複数行からなる別行立て数式の途中で改ページを行うことができる。改ページを抑制することもできる。
- 数式環境前後の余白や、複数行数式がある場合の数式行同士の余白を変更することができる。

以下、具体的に使い方を説明します。

2.1. 数式環境の使い方

最も単純な別行立て数式用のコマンドは +eqn であり、 標準パッケージの +math コマンドと同様に 1 行の別行立て数式を組むことができます。 ただし、+eqn を用いた場合はデフォルトで連番の数式番号が付与されます。

```
+eqn(${
    x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
  });
```

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{1}$$

数式前後で改段落を行いたくない場合は、インラインテキストの中に \eqn を挿入することでも同様の環境を実現することができます。

```
+p{
  したがって、2 次方程式の解の公式は
  \eqn(${
      x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
    });
  で与えられる。
```

```
}
```

したがって、2 次方程式の解の公式は

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{2}$$

で与えられる。

複数の数式を縦に並べたい場合は複数の +eqn をそのまま並べてもよいですが、+gather コマンドを使えばよりスマートに複数の数式を表示することができます. +gather コマンドでは、単一の数式 (math) の代わりに数式のリスト (math list) を指定します。

```
+gather(${
  | \p{x + y}^2 = x^2 + 2xy + y^2,
  | \p{x + a}\p{x + b} = x^2 + \p{a + b} x + ab,
  | \p{x + y}\p{x - y} = x^2 - y^2.
  |});
```

$$(x + y)^2 = x^2 + 2xy + y^2, \tag{3}$$

$$(x + a)(x + b) = x^2 + (a + b)x + ab, \tag{4}$$

$$(x + y)(x - y) = x^2 - y^2. \tag{5}$$

上の例にあるように、+gather を用いて並べた式にもデフォルトで連番の数式番号が付与されます。また、+eqn と同様にインラインテキスト版の \eqn も用意されています。

なお、デフォルトでは +gather 環境は中央揃えとなっています。後述するパラメータの値を変更することにより、式を左揃えにすることもできます。以下の例と上の例を見比べてみてください。

```
+with-param(AZMathEquation.eqn-gather-align-coef)(0.0)<
  +gather(${
    | \p{x + y}^2 = x^2 + 2xy + y^2,
    | \p{x + a}\p{x + b} = x^2 + \p{a + b} x + ab,
    | \p{x + y}\p{x - y} = x^2 - y^2.
    |});
  >
```

$$(x+y)^2 = x^2 + 2xy + y^2, \tag{6}$$
$$(x+a)(x+b) = x^2 + (a+b)x + ab, \tag{7}$$
$$(x+y)(x-y) = x^2 - y^2. \tag{8}$$

長い式を途中で折り返すときや式変形の経過を見せたいときなど、式を特定の箇所で揃えたい場合があります。そのような用途には +align を用いることができます。

```
+align(${
  | \p{x + y}\p{x - y} |= x^2 -xy + xy - y^2 \notag
  |                    |= x^2 - y^2.
  |});
```

$$\begin{aligned}(x+y)(x-y) &= x^2 - xy + xy - y^2 \\ &= x^2 - y^2.\end{aligned} \tag{9}$$

なお、\notag はその行における数式番号の付与を抑制するコマンドです（タグの仕様についての詳細は後述します）。先程の gather と異なり、イコールの記号 = の前にバー | が付いています。数式はこのバーの位置を基準として、左にあるものは右揃えで、右にあるものは左揃えで整列されます。上の例ではコードの見栄えのために縦棒の位置を揃えて書きましたが、コード上の縦棒の位置を揃える必要はもちろんありません。なお、この縦棒はいずれも数式の区切りを意味するものであり、以下のように数式のリストを渡した場合と同じ動作となります。

```
+align[
  ${ \p{x + y}\p{x - y} };
  ${= x^2 -xy + xy - y^2 \notag};
  ${};
  ${= x^2 - y^2.};
];
```

つまり、+align コマンドも結局は +gather と同様に数式のリストを与えているにすぎないのですが、+align コマンドでは 2 つのペアごとに数式を並べる点が異なっており、結果としてイコールの前などの位置で数式を揃えることができるようになっています。

+alignat は複数の数式を縦横に並べるためのコマンドです。並べる数式の列数と、数式のリストを指定します。

```
+alignat(2)(${
```

```
  | \p{x + y}^2         |= x^2 + 2xy + y^2,
  | \p{x - y}^2         |= x^2 - 2xy + y^2,
  | \p{x + a}\p{x + b} |= x^2 + \p{a + b} x + ab,
  | \p{x + y}\p{x - y} |= x^2 - y^2.
  |});
+alignat(3)($${
  | a |= x, | b =| y,  | c      |= z,
  | d |= w, | e  |= s, | f = t. |
  |});
```

$$(x+y)^2 = x^2 + 2xy + y^2, \qquad (x-y)^2 = x^2 - 2xy + y^2,$$
$$(x+a)(x+b) = x^2 + (a+b)x + ab, \qquad (x+y)(x-y) = x^2 - y^2.$$
$$a = x, \qquad b = y, \qquad c = z,$$
$$d = w, \qquad e = s, \qquad f = t.$$

\aligned は、数式の中で \align のような揃え方を実現したい場合に用いられます。特に場合分けを行う際には \cases が便利です。

```
+eqn($${
  y = \cases{
  | x      | \p{x \geq 0}
  | \neg x | \p{x < 0}
  |}
  });
```

$$y = \begin{cases} x & (x \geq 0) \\ -x & (x < 0) \end{cases} \tag{10}$$

2.2. 数式のタグ

数式には自動で数式番号がつくという話をしました。番号のつく数式には、以下のようにラベルを付与することができます。

```
+gather($${
  | \p{x + y}^2 = x^2 + 2xy + y^2, \label!(`formula1`)
  | \p{x + a}\p{x + b} = x^2 + \p{a + b} x + ab, \label!(`formula2`)
  | \p{x + y}\p{x - y} = x^2 - y^2. \label!(`formula3`)
```

```
  |});
```

$$(x+y)^2 = x^2 + 2xy + y^2, \tag{11}$$
$$(x+a)(x+b) = x^2 + (a+b)x + ab, \tag{12}$$
$$(x+y)(x-y) = x^2 - y^2. \tag{13}$$

付与した数式は `\ref(`eq:formula1`);` とすることで参照することができます。たとえば先程の例の一番上の式は (11) 式でした。

さらに、表示するタグを変更することもできます。

```
+gather(${
  | \p{x + y}^2 = x^2 + 2xy + y^2,  \label?:!(`1-a`)!(`formula4`)
  | \p{x + y}\p{x - y} = x^2 - y^2.
    \label?:!(`very long tag`)!(`formula5`)
  |});
```

$$(x+y)^2 = x^2 + 2xy + y^2, \tag{1-a}$$
$$(x+y)(x-y) = x^2 - y^2. \tag{very long tag}$$

式変形の途中など、数式番号やタグを表示したくない場合もあるでしょう。タグを非表示にする方法は 2 通りあります。1 つ目は `\notag` コマンドを用いる方法です。`\label` の代わりに `\notag` を用いると、タグが付かなくなります。式変形や長い数式の途中など、特定の行のみ無効化したい場合に有効です。

```
+gather(${
  | \p{x + y}^2 = x^2 + 2xy + y^2, \notag
  | \p{x + a}\p{x + b} = x^2 + \p{a + b} x + ab,
  | \p{x + y}\p{x - y} = x^2 - y^2.
  |});
```

$$(x+y)^2 = x^2 + 2xy + y^2,$$
$$(x+a)(x+b) = x^2 + (a+b)x + ab, \tag{14}$$
$$(x+y)(x-y) = x^2 - y^2. \tag{15}$$

2 つ目は、数式環境の引数に `AZMathEquation.notag` を与える方法です。`AZMathEquation.notag` を指定した数式環境ではデフォルトでタグが付かなくなり、`\label` コマンド

を用いた箇所にのみタグが付くようになります。特定の箇所に限りタグを付与したい場合には有効な手段と言えます。

```
+gather?:(AZMathEquation.notag)($${
  | \p{x + y}^2 = x^2 + 2xy + y^2,
  | \p{x + a}\p{x + b} = x^2 + \p{a + b} x + ab,
  | \p{x + y}\p{x - y} = x^2 - y^2. \label!(`formula-notag`)
  |});
```

$$
\begin{aligned}
(x+y)^2 &= x^2 + 2xy + y^2, \\
(x+a)(x+b) &= x^2 + (a+b)x + ab, \\
(x+y)(x-y) &= x^2 - y^2.
\end{aligned} \tag{16}
$$

数式環境のオプション引数に指定する `notag` などの関数は、本来数式のタグの体裁を指定するのに利用されます。したがって、関数定義によって数式のラベルを別の体裁にすることもできます。

数式が長いことによって通常の位置にタグを打つと数式とタグが重なってしまう場合、タグの位置を自動で下にずらしてくれるのは AZmath パッケージの数式環境の強みです。

```
+gather($${
  | \p{a + b + c + d + e}\p{x + y}
    = ax + ay + bx + by + cx + cy + dx + dy + ex + ey
  | x^3 + y^3 + z^3 - 3xyz =
    \p{x + y + z}\p{x^2 + y^2 + z^2 - xy - yz - zx}
    \label?:!(`long tag`)!(`long-tag-formula`)
  |});
```

$$
(a+b+c+d+e)(x+y) = ax + ay + bx + by + cx + cy + dx + dy + ex + ey \tag{17}
$$

$$
x^3 + y^3 + z^3 - 3xyz = (x+y+z)\big(x^2 + y^2 + z^2 - xy - yz - zx\big) \tag{long tag}
$$

2.3. 数式環境途中での改ページ

標準パッケージの `+align` では数式環境の途中で改ページを行うことができなかったのに対し、AZmath パッケージの `+gather`, `+align`, `+alignat` ではデフォルトで数式環境の途中で改ページを行うことができます。LaTeX の `\allowdisplaybreaks` オプションを付けた

場合と同じような挙動と言えます。しかし、改ページを抑制したい場合もあるでしょう。数式の中で \keeppage コマンドを付けることで、その数式の直後の位置で改ページすることを抑制できます。

数式の途中だからタグ付けも改ページもしたくない、という場合には \notag\keeppage としてもよいですが、同じ効果を持つ \tbc（To Be Continued の略）というより短いコマンドも用意しています。

2.4. 数式環境のパラメータ

AZmath パッケージで用意されている数式環境では、パラメータを指定してその振る舞いを変更することができます。

`allow-display-break`
: bool 型のパラメータ。デフォルトは `true`。
別行立て数式の途中で改行することを許すかどうか。false にすると別行立て数式に入る直前及び、別行立て数式の行の間では改行ができない。

`vmargin-between-eqn`
: `context -> length` 型のパラメータ。デフォルトは `(fun ctx -> (get-font-size ctx) *' 0.6)`。
数式同士の間の余白。 テキスト処理文脈を引数に取る関数の形で指定する。 たとえば `(fun ctx -> (get-font-size ctx) *' 1.5)` とすると現在のフォントサイズの 1.5 倍の高さとなり、`(fun _ -> 20pt)` とすると現在のフォントサイズなどによらず 20pt となる。

`vmargin-between-eqn`, `vmargin-after-eqn`
: `context -> length` 型のパラメータ。デフォルトはともに `(fun ctx -> (get-font-size ctx) *' 1.0)`。
別行立て数式前後の余白（段落間空白）。

`min-gap-between-eqn-and-tag`
: `context -> length` 型のパラメータ。デフォルトは `(fun ctx -> 2pt)`。
数式やタグがどこまで近づいてよいか。数式を整列させたとき、数式の右端とタグの左端がこのパラメータで示す値以上無かった場合、タグは数式の下に回り込んで組まれる。テキスト処理文脈を引数に取る関数の形で指定する。

`vmargin-between-eqn-and-tag`
: `context -> length` 型のパラメータ。デフォルトは `(fun ctx -> (get-font-size ctx) *' 0.3)`。
数式やタグが長いためにタグが数式の下に回り込んだ際に、数式とタグの間に入る縦方向の余白。テキスト処理文脈を引数に取る関数の形で指定する。

パラメータはプリアンブルで以下のように指定することができます。

```
let () = AZMathEquation.allow-display-break
  |> AZMathParam.set true
let () = AZMathEquation.vmargin-between-eqn
  |> AZMathParam.set (fun _ -> 20pt)
```

また、パラメータをブロックテキスト・インラインテキスト中で指定するコマンドも存在します。

3. Unicode を用いた数式の記述

最新版の SATYSFI（未リリース）では、数式モード内で直接 Unicode 文字を記述できるようになりました。ただし、現時点では `set-math-char` という関数を用いてテキスト処理文脈を変更しない限り、数式文字間のスペーシングが正しくならない場合があります。たとえば除算記号である「÷」は多くのケースで二項演算子として用いられる記号ですが、何も設定せず標準の `math` パッケージの `+math` コマンドを用いるだけでは以下のように二項演算子前後のスペースが詰まって表示されます。

```
+math(${ x ÷ y });  % SATySFi 標準の math パッケージにある数式環境
```

$$x÷y$$

AZMath パッケージの数式環境では自動でスペーシングに関する設定を行い、Unicode を用いた数式の記述であっても正しいスペーシングで組まれるよう努力しています。

```
+eqn(${ x ÷ y });  % azmath パッケージの数式環境
```

$$x \div y \tag{18}$$

現時点では全ての記号を網羅できていない可能性があります。意図しないスペーシングで組まれてしまうものを発見された場合は、GitHub での Issue 報告[*1]にご協力よろしくお願いします。

4. アクセント

`\hat{x}` などとすることによって、数式にアクセントを追加することができます.

1 https://github.com/monaqa/satysfi-azmath/issues

$$\hat{a},\hat{b},\hat{c},\hat{d},\hat{e},\hat{f},\hat{g},\hat{h},\hat{i},\hat{j},\hat{k},\hat{l},\hat{m},\hat{n},\hat{o},\hat{p},\hat{q},\hat{r},\hat{s},\hat{t},\hat{u},\hat{v},\hat{w},\hat{x},\hat{y},\hat{z}, \tag{19}$$

$$\hat{A},\hat{B},\hat{C},\hat{D},\hat{E},\hat{F},\hat{G},\hat{H},\hat{I},\hat{J},\hat{K},\hat{L},\hat{M},\hat{N},\hat{O},\hat{P},\hat{Q},\hat{R},\hat{S},\hat{T},\hat{U},\hat{V},\hat{W},\hat{X},\hat{Y},\hat{Z}, \tag{20}$$

$$\hat{\alpha},\hat{\beta},\hat{\gamma},\hat{\delta},\hat{\varepsilon},\hat{\zeta},\hat{\eta},\hat{\theta},\hat{\iota},\hat{\kappa},\hat{\lambda},\hat{\mu},\hat{\nu},\hat{\xi},\hat{o},\hat{\pi},\hat{\rho},\hat{\sigma},\hat{\tau},\hat{\upsilon},\hat{\varphi},\hat{\chi},\hat{\psi},\hat{\omega}, \tag{21}$$

$$\hat{A},\hat{B},\hat{\Gamma},\hat{\Delta},\hat{E},\hat{Z},\hat{H},\hat{\Theta},\hat{I},\hat{K},\hat{\Lambda},\hat{M},\hat{N},\hat{\Xi},\hat{O},\hat{\Pi},\hat{P},\hat{\Sigma},\hat{T},\hat{\Upsilon},\hat{\Phi},\hat{X},\hat{\Psi},\hat{\Omega}. \tag{22}$$

他のアクセントについても同様に付けることができます．現時点で使用可能なコマンドは `\hat{}`, `\tilde{}`, `\bar{}`, `\adot{}`, `\ddot{}`, `\breve{}`, `\vec{}` です。

$$\tilde{a},\tilde{b},\tilde{c},\tilde{d},\tilde{e},\tilde{f},\tilde{g},\tilde{h},\tilde{i},\tilde{j},\tilde{k},\tilde{l},\tilde{m},\tilde{n},\tilde{o},\tilde{p},\tilde{q},\tilde{r},\tilde{s},\tilde{t},\tilde{u},\tilde{v},\tilde{w},\tilde{x},\tilde{y},\tilde{z}, \tag{23}$$

$$\bar{a},\bar{b},\bar{c},\bar{d},\bar{e},\bar{f},\bar{g},\bar{h},\bar{i},\bar{j},\bar{k},\bar{l},\bar{m},\bar{n},\bar{o},\bar{p},\bar{q},\bar{r},\bar{s},\bar{t},\bar{u},\bar{v},\bar{w},\bar{x},\bar{y},\bar{z}, \tag{24}$$

$$\dot{a},\dot{b},\dot{c},\dot{d},\dot{e},\dot{f},\dot{g},\dot{h},\dot{i},\dot{j},\dot{k},\dot{l},\dot{m},\dot{n},\dot{o},\dot{p},\dot{q},\dot{r},\dot{s},\dot{t},\dot{u},\dot{v},\dot{w},\dot{x},\dot{y},\dot{z}, \tag{25}$$

$$\ddot{a},\ddot{b},\ddot{c},\ddot{d},\ddot{e},\ddot{f},\ddot{g},\ddot{h},\ddot{i},\ddot{j},\ddot{k},\ddot{l},\ddot{m},\ddot{n},\ddot{o},\ddot{p},\ddot{q},\ddot{r},\ddot{s},\ddot{t},\ddot{u},\ddot{v},\ddot{w},\ddot{x},\ddot{y},\ddot{z}. \tag{26}$$

$$\breve{a},\breve{b},\breve{c},\breve{d},\breve{e},\breve{f},\breve{g},\breve{h},\breve{i},\breve{j},\breve{k},\breve{l},\breve{m},\breve{n},\breve{o},\breve{p},\breve{q},\breve{r},\breve{s},\breve{t},\breve{u},\breve{v},\breve{w},\breve{x},\breve{y},\breve{z}. \tag{27}$$

$$\vec{a},\vec{b},\vec{c},\vec{d},\vec{e},\vec{f},\vec{g},\vec{h},\vec{i},\vec{j},\vec{k},\vec{l},\vec{m},\vec{n},\vec{o},\vec{p},\vec{q},\vec{r},\vec{s},\vec{t},\vec{u},\vec{v},\vec{w},\vec{x},\vec{y},\vec{z}. \tag{28}$$

$\hat{f}$ や $\hat{J}$ など，中にはアクセントを付けるときに x 方向の位置がズレているように見えるものがあります．これは，引数の文字に関わらず x 方向の位置を一律に定めて文字を付けているためであり，アクセントの適切な位置をフォントから取得する機構が SATYSFI にまだ備わっていないことに起因します．場当たり的な解決策として，オプション引数に補正係数を指定して `\hat?:!(0.18){f}` などとすれば補正することができます．

```
+alignat(3)(${
  | \hat{f} |= \hat?:!(0.18){f},
  | \hat{J} |= \hat?:!(0.20){J},
  | \hat{m} |= \hat?:!(0.-.0.15){m}.
  |});
```

$$\hat{f} = \hat{f}, \qquad \hat{J} = \hat{J}, \qquad \hat{m} = \hat{m}.$$

現時点での実用的な運用方法は，文書で使いそうなアクセント変数をプリアンブル部分で以下のように定義してしまうことでしょう．

```
let-math \hat-f = ${\hat?:!(0.18){f}}
let-math \hat-J = ${\hat?:!(0.20){J}}
```

複数文字にまたがる数式の装飾は、上で示したコマンドでは対応できません。代わりに `\overline`, `\widehat`, `\overrightarrow` といったコマンドが用意されています。

```
+gather(${
  | \overline{abc} = \overline{ab} + \overline{ac} - \overline{bc}
```

```
  | \widehat{abc} = \widehat{ab} + \widehat{ac} - \widehat{bc}
  | \overrightarrow{abc} =
    \overrightarrow{ab} + \overrightarrow{ac} - \overrightarrow{bc}
  |});
```

$$\overline{abc} = \overline{ab} + \overline{ac} - \overline{bc} \tag{29}$$

$$\widehat{abc} = \widehat{ab} + \widehat{ac} - \widehat{bc} \tag{30}$$

$$\overrightarrow{abc} = \overrightarrow{ab} + \overrightarrow{ac} - \overrightarrow{bc} \tag{31}$$

5. 括弧

標準の math パッケージにも括弧は定義されていますが，azmath パッケージでも新たな括弧を定義しています．標準のアプローチと同様に括弧はグラフィックスで定義しており，フォントに入っている括弧は用いていません．

```
+gather($${
  | \p{\int_0^\infty \frac{\sin x}{\sqrt{x}} \ordd x}^2
    = \sum_{k=0}^\infty
      \frac{\p{2k}\!}{2^{2k}\p{k\!}^2} \frac{1}{2k + 1}
    = \prod_{k=1}^\infty \frac{4k^2}{4k^2 - 1}
    = \frac{\pi}{2}
  | \lim_{x \to 0}\p{\frac{1}{2} \p{\p{\p{x + 1} + x}^2}}
  |});
```

$$\left(\int_0^\infty \frac{\sin x}{\sqrt{x}}\mathrm{d}x\right)^2 = \sum_{k=0}^\infty \frac{(2k)!}{2^{2k}(k!)^2}\frac{1}{2k+1} = \prod_{k=1}^\infty \frac{4k^2}{4k^2-1} = \frac{\pi}{2} \tag{32}$$

$$\lim_{x\to 0}\left(\frac{1}{2}\left(((x+1)+x)^2\right)\right) \tag{33}$$

azmath で定義された括弧にはいくつかの特徴があります．

- 括弧の高さは必要に応じて伸縮する．
- 複数の括弧を入れても，括弧の高さは変わらない．ただし例外として，絶対値に用いられる \pabs のように開き括弧と閉じ括弧の区別が見かけ上つかない括弧については，括弧のネストをわかりやすくするため中身より一段階高くする．

2 番目については，以下のような例を見るとよりはっきりと違いが分かるでしょう．左側が標準で用意されている \paren 及び \sqbracket、右側が AZmath で定義されている \p 及び \pB です。

$$\Big(\Big(\Big(\Big(\Big(\Big(\big((x)\big)\Big)\Big)\Big)\Big)\Big), \qquad (((((((x))))))).$$

$$\Big[\Big[\Big[\Big[\Big[\Big[\big[[x]\big]\Big]\Big]\Big]\Big]\Big], \qquad [[[[[[[x]]]]]]].$$

括弧の種類はいくつか用意されています．

```
+gather($${
  | \pb{\frac{1}{2} \pb{\pb{\pb{x + 1} + x}^2}}
  | \pB{\frac{1}{2} \pB{\pB{\pB{x + 1} + x}^2}}
  | \pabs{\frac{1}{2} \pabs{\pabs{\pabs{x + 1} + x}^2}}
  | \pnorm{\frac{1}{2} \pnorm{\pnorm{\pnorm{x + 1} + x}^2}}
  | \pangle{\frac{1}{2} \pangle{\pangle{\pangle{x + 1} + x}^2}}^2
  |});
```

$$\left\{\frac{1}{2}\left\{\left\{x+1\right\}+x\right\}^2\right\} \tag{34}$$

$$\left[\frac{1}{2}\left[\left[x+1\right]+x\right]^2\right] \tag{35}$$

$$\left|\frac{1}{2}\left|\left|x+1\right|+x\right|^2\right| \tag{36}$$

$$\left\|\frac{1}{2}\left\|\left\|x+1\right\|+x\right\|^2\right\| \tag{37}$$

$$\left\langle\frac{1}{2}\left\langle\left\langle x+1\right\rangle+x\right\rangle^2\right\rangle^2 \tag{38}$$

時には括弧の大きさを自動ではなく手動で調整したい場合もあるかもしれません．括弧の大きさはオプション引数によって調整できます．\p?:!(20pt){xxx} で，あたかも中身が 20pt の高さを持っているかのように括弧を組むことができます．

```
+gather($${
  | \p?:!(10pt){xxx} = \p?:!(15pt){xxx} = \p?:!(30pt){xxx}
  | \pangle?:!(12pt){ \frac{1}{2} \pangle?:!(8pt){
      \pangle?:!(16pt){\pangle?:!(12pt){x + 1} + x}^2
      }}
```

```
  |});
```

$$\big(xxx\big) = \Big(xxx\Big) = \left(xxx\right) \tag{39}$$

$$\Big\langle \frac{1}{2} \langle\!\Big\langle \Big\langle x+1 \Big\rangle + x \Big\rangle^2 \rangle\!\Big\rangle \tag{40}$$

\genparen を用いれば、デフォルトでは用意されていないペアの括弧を定義することもできます。

```
+eqn(
  open AZMathParens in
  let-math \bra m =
    ${ \genparen!(angle-bracket-l)!(abs-bracket-r)!(m) }
  in
  let-math \ket m =
    ${ \genparen!(abs-bracket-l)!(angle-bracket-r)!(m) }
  in
  ${ \bra{\phi}, \ket{\psi} }
```

$$\langle \varphi |, | \psi \rangle \tag{41}$$

AZMathParens.paren-scheme は SATySFi の paren 型を生成するための関数であり、これを用いればさらに自由度の高い括弧を組むことができます。

数式を囲む括弧とは多少異なるものの、\overbrace{m}, \underbrace{m} で数式の上下に波括弧を描く事ができます．数式の一部に注釈を入れたいときなどに使えます．

```
+gather(
  let m = ${\frac{\neg b \pm \sqrt{b^2 - 4ac}}{2a}} in
  ${
    |x = \overbrace{#m} + \overbrace{\pb{#m} + #m}^{\text!{大事}}
    |x = \underbrace{#m + \pb{#m}} + \underbrace{#m}_{\text!{大事}}
    |});
```

$$x = \overbrace{\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}} + \overbrace{\left\{\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}\right\}}^{\text{大事}} + \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{42}$$

$$x = \underbrace{\frac{-b \pm \sqrt{b^2 - 4ac}}{2a} + \left\{\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}\right\}} + \underbrace{\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}}_{\text{大事}} \tag{43}$$

片方が片方に内包される形であれば，入れ子にして使うことも可能です．

```
+gather($${
  | x = \underbrace{
      \overbrace{ a + \underbrace{b + c}_{=0} }^{\text!{foo}}
      + d + \overbrace{e + f}
    }_{=0}
  |});
```

$$x = \underbrace{\overbrace{a + \underbrace{b + c}_{=0}}^{\text{foo}} + d + \overbrace{e + f}}_{=0} \tag{44}$$

6. 行列

行列を描くことも出来ます．`\pmatrix!(n){| ... elements ... |}` とすることで、n 個の列を持つ行列を組むことができます。なお、行の数は要素数から自動で判断されます。

```
+gather($${
  | A = \pmatrix!(2){| a | b | c | d |}
  | A^{-1} = \frac{1}{ad - bc} \pmatrix!(2){|d|\neg b|\neg c|a|}
    = \pmatrix!(2){
    |      \frac{d}{ad - bc} | \neg \frac{b}{ad - bc}
    | \neg \frac{c}{ad - bc} |      \frac{a}{ad - bc}
    |}
  |});
```

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \tag{45}$$

$$A^{-1} = \frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix} = \begin{pmatrix} \dfrac{d}{ad-bc} & -\dfrac{b}{ad-bc} \\ -\dfrac{c}{ad-bc} & \dfrac{a}{ad-bc} \end{pmatrix} \tag{46}$$

行列の括弧は色々変えられます.

```
+align(${
  | A |= \matrix!(2){| a | b | c | d |}
  | A |= \bmatrix!(2){| a | b | c | d |}
  | A |= \vmatrix!(2){| a | b | c | d |}
  | A |= \dmatrix!(2){| a | b | c | d |}
  |});
```

$$A = \begin{matrix} a & b \\ c & d \end{matrix} \tag{47}$$

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{48}$$

$$A = \begin{vmatrix} a & b \\ c & d \end{vmatrix} \tag{49}$$

$$A = \begin{Vmatrix} a & b \\ c & d \end{Vmatrix} \tag{50}$$