Easytable パッケージ

@monaqa

目次

1. Easytable パッケージの概要

Easytable は、SATYSFI で汎用的な表を楽に組めるようにするためのものです。SATYSFI には既に表を組むためのパッケージがいくつか標準で用意されていますが、いずれも単純な表を描くには少々複雑な構文を要求されるものでした。本パッケージのコマンドでは、ユーザが設定しなければならない必須引数を最小限に抑え、 よりシンプルで組みやすいインターフェースを実現しています。

2. Gallery

具体例から説明したほうが理解が進みやすいと考え、まず具体的なコード例とともに出来上がりの表をいくつか紹介します。以下では easytable パッケージが正常にインストールされており、かつ以下のようなコードで正常にインポートされているものとします。

```
@require: easytable
open EasyTableAlias
```

なお、以降のコードをより簡潔にするため、easytable のインポートだけでなく、パッケージに入っている EasyTableAlias モジュールを open して中身に直接アクセスできる状態になっているものとします。

2.1. 単純な表

最も単純に表を組むには +easytable コマンドを用います。ブロックテキスト中で用いることで、簡単に表を組むことができます。

```
+easytable[l;c;r]{
  | header1    | header2      | header3
  | align left | align center | align right
  | a          | b            | c
  |}
```

header1	header2	header3
align left	align center	align right
a	b	c

+easytable の API は上で示すように非常に単純です。+easytable は 2 つの必須引数を取り、1 番目で各列の体裁、2 番目で表の中身を指定します。1 番目は cellfmt と呼ばれる型のリストであり、n 番目の要素が「左から数えて n 番目の列をどのように揃えるか」を表しています。上の例では [l; c; r] を指定することで、左の列から順にそれぞれ左揃え (left)、中央揃え (center)、右揃え (right) となっています。2 番目はインラインテキスト型のリストです。ユーザが引数として指定するのは 1 次元配列のようなデータ構造ですが、内部で 2 次元の構造（リストのリスト）へと変換されて表となります。インラインテキストのリストは普

通に書くならば `[{a}; {b}; {c}]` と表すことになりますが、SATYSFI にはこれを `{|a|b|c|}` と表記する糖衣構文があります。`+easytable` コマンドはこの構文を積極的に活用し、Markdown や AsciiDoc のような軽量マークアップ言語に近い書き易さを実現しています。

なお、改行を含めた余剰なスペースは単にコードの見やすさのために入れているだけであり、たとえば以下のように書いても結果は変わりません。読みやすさ、編集のしやすさなどを考えて調整するとよいでしょう。

```
+easytable[l;c;r]{| header1 | header2 | header3 | align left
| align center | align right | a | b | c |}
```

「結果は変わらない」ということから分かる通り、上のコマンドから得られるのは 3 行 3 列の表です。すなわち、列の数は第 2 引数の改行の位置ではなく、第 1 引数のリストの長さによって決定されます。この仕様を知らなければ非直観的な結果を生んでしまう可能性があるため、注意が必要です。

`\easytable` は `+easytable` のインライン版であり、任意のインラインテキスト中に表を埋め込むことができます。引数は（後述するオプション引数も含めて）`+easytable` と一切変わりません。

```
+p{
  インライン中に埋め込まれた表：
  \easytable[l;c;r]{
    | header1    | header2      | header3
    | align left | align center | align right
    | a          | b            | c
    |}
}
```

インライン中に埋め込まれた表：

header1	header2	header3
align left	align center	align right
a	b	c

`+easytable` や `\easytable` の第 2 引数に渡すのはインラインテキストのリストですから、インラインコマンドを用いた装飾や、数式の挿入も可能です。（使う機会があるかはともかく）表の入れ子もできます。

```
+easytable[r; c]{
  |        | 例
  | Emph   | \emph{強調されたテキスト}
  | Code   | \code(`some code`);
  | Math   | ${x = \frac{\minus b \pm \sqrt{b^2 - 4ac}}{2a}}
  | Inner table |
   \easytable[l; r]{
     | header 1 | header 2
     | a        | b
     |}
  |}
```

	例
Emph	強調されたテキスト
Code	`some code`
Math	$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
Inner table	header 1 | header 2 a | b

1 番目に渡す `cellfmt` 型のリストに `l` ではなく `lw (length)` を渡すことによって、表の横幅を指定することができます。このとき、表の横幅よりも長いインラインテキストが表の中にあれば、SATYSFI の本体で用意されている行分割アルゴリズムに従って行分割されます。表の中に長い文章を書く場合に便利といえるでしょう。

```
+easytable[l; lw 120pt; lw 120pt;]{
  | Column 1 | Column 2   | Column 3
  | 通常の列 | 横幅 120pt
  | 横幅 120pt で、なおかつ横幅よりも長いテキストが入っている場合
  |}
```

Column 1	Column 2	Column 3
通常の列	横幅 120pt	横幅 120pt で、なおかつ横幅よりも長いテキストが入っている場合

同様に列幅を指定できるオプションとして、中央揃えの `cw` と右揃えの `lw` も用意されています。ただし、これらのオプションを指定しても、長いテキストを入れたときに段落が中央揃えや右揃えになるわけではなく、段落内は全て左揃えで組まれます[*1]。その点には注意が必要です。

```
+easytable[lw 120pt; cw 120pt; rw 120pt;]{
  | Align left | Align center | Align right
  | short text | short text   | short text
  |横幅 120pt で、なおかつ横幅よりも長いテキストが入っている場合
  |横幅 120pt で、なおかつ横幅よりも長いテキストが入っている場合
  |横幅 120pt で、なおかつ横幅よりも長いテキストが入っている場合
  |}
```

Align left	Align center	Align right
short text	short text	short text
横幅 120pt で、なおかつ横幅よりも長いテキストが入っている場合	横幅 120pt で、なおかつ横幅よりも長いテキストが入っている場合	横幅 120pt で、なおかつ横幅よりも長いテキストが入っている場合

2.2. 罫線の指定

表組版において、罫線は一般にデザイン上重要な意味を持つと考えられます。今までの例で見た通り、デフォルトでは表の上下に太い線を、そして 1 行目の下に細い線を引く、という挙動でした。これは 1 行目がヘッダ行を、それ以降の列がコンテンツを表すような表に対しては自然なレイアウトとなりますが、場合によってはヘッダ行がなかったり、ヘッダ列があったりするような表も当然考えられます。本パッケージでは、そのような需要にもできるだけ

1 これはどちらかというと、SATYSFI で単なるインラインテキストを右揃えや中央揃えの段落にする汎用的な手段が現状用意されていないのが原因です。

簡単なインターフェースで対応できるようにしています。

本パッケージで罫線を制御するには、+easytable コマンドにオプション引数を指定します。この引数は ruleptn と呼ばれる型のリストを取り、ここに引きたい罫線を羅列することで幅広い種類の罫線を引くことができます。まず最も単純な場合として、オプション引数に空のリストを与えた場合を考えましょう。この場合、以下のように罫線が全く引かれずに表が組まれます。

```
+easytable?:[][r; c; l]{
  | How       | I       | want
  | a         | drink   | alcoholic
  | of        | course  | after
  | the       | heavy   | lectures
  | involving | quantum | mechanics
  |}
```

How	I	want
a	drink	alcoholic
of	course	after
the	heavy	lectures
involving	quantum	mechanics

これではやはり少し寂しいので、上下に罫線を引くことにしましょう。オプション引数のリストに t を追加すると上に、b を追加すると下に、それぞれ太い罫線が引かれます。

```
+easytable?:[t][r; c; l]{
  | How | I | want | a | drink | alcoholic | of | course
  | after | the | heavy | lectures | involving | quantum | mechanics
  |}
+easytable?:[t; b][r; c; l]{
  | How | I | want | a | drink | alcoholic | of | course
  | after | the | heavy | lectures | involving | quantum | mechanics
  |}
```

How	I	want
a	drink	alcoholic
of	course	after
the	heavy	lectures
involving	quantum	mechanics

How	I	want
a	drink	alcoholic
of	course	after
the	heavy	lectures
involving	quantum	mechanics

表の途中に水平線を引きたい、というのもよくあることです。その場合には `m (int)` を用いることができます。`m` の引数は整数であり、正の数 n を指定すれば n 行目の真下に線が引かれます。また、負の数 $-n$ を指定したときは、最後から数えて n 行目の真下に線が引かれます。ちなみに `+easytable` のオプション引数を省略した場合は、`[t; b; m 1]` が設定されたときと等価な挙動を示します。

```
+easytable?:[t; b; m 2][r; c; l]{
  | How | I | want | a | drink | alcoholic | of | course
  | after | the | heavy | lectures | involving | quantum | mechanics
  |}
+easytable?:[t; b; m (-2)][r; c; l]{
  | How | I | want | a | drink | alcoholic | of | course
  | after | the | heavy | lectures | involving | quantum | mechanics
  |}
```

How	I	want
a	drink	alcoholic
of	course	after
the	heavy	lectures
involving	quantum	mechanics

How	I	want
a	drink	alcoholic
of	course	after
the	heavy	lectures
involving	quantum	mechanics

列を分けるために縦に線を引くスタイルも、特に日本でよく見かけるといえるでしょう。`v (int)` で列の境目に鉛直方向の線を引くことができます。引数の意味は `m` のときと同様であり、左から数えて何番目に線を引くかを表します。

もう少し自由度の高い罫線が引きたい場合は、`d (int, int) (int, int)` を指定します。格子点の座標をインデックスで指定し、2 点を結ぶ直線を引くことができます。

```
+easytable?:[t; b; d (0, 0) (1, 1); d (3, 1) (1, -1)][r; c; l]{
  | How | I | want | a | drink | alcoholic | of | course
  | after | the | heavy | lectures | involving | quantum | mechanics
  |}
```

How	I	want
a	drink	alcoholic
of	course	after
the	heavy	lectures
involving	quantum	mechanics

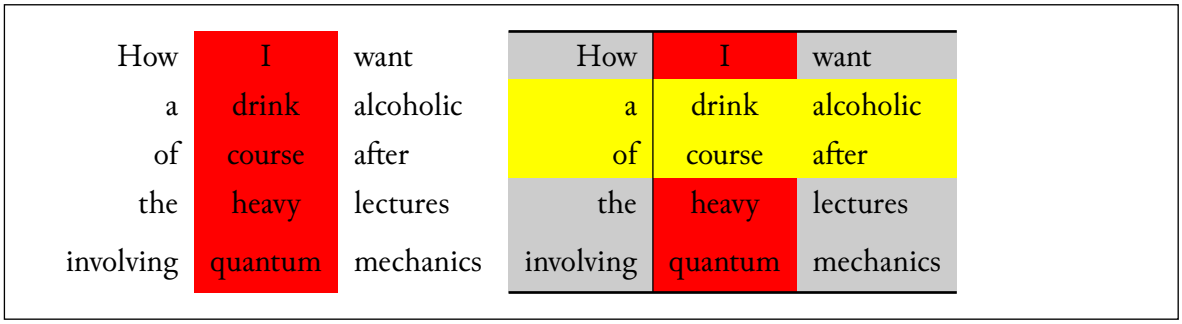
表全体を太い外枠で囲むときは `rect` を使うことができます。

```
+easytable?:[rect][r; c; l]{
  | How | I | want | a | drink | alcoholic | of | course
  | after | the | heavy | lectures | involving | quantum | mechanics
  |}
```

How	I	want
a	drink	alcoholic
of	course	after
the	heavy	lectures
involving	quantum	mechanics

罫線とは異なりますが、 オプション引数では表の背景色を指定することもできます。 表全体の背景色を指定するには `bg-a (color)` を、 特定の範囲の列の背景色を指定するには `bg-c (color) (int) (int)` を、特定の範囲の行の背景色を指定するには `bg-r (color) (int) (int)` を用います。なお、グラフィックスの重ね順は罫線・背景色関係なく指定された順です。すなわち、リストで並んだ順にグラフィックスが上書きされてしまいます。従って、背景色に塗り潰されないように罫線を引きたければ、背景色を指定した後に罫線を指定しなければなりません。

```
\easytable?:[bg-c (Color.red) 1 2][r; c; l]{
  | How | I | want | a | drink | alcoholic | of | course
  | after | the | heavy | lectures | involving | quantum | mechanics
  |}
\easytable?:[
  bg-a (Color.gray 0.8);
  bg-c (Color.red) 1 2;
  bg-r (Color.yellow) 1 3;
  t; b; v 1;
][r; c; l]{
  | How | I | want | a | drink | alcoholic | of | course
  | after | the | heavy | lectures | involving | quantum | mechanics
  |}
```



How	I	want	How	I	want
a	drink	alcoholic	a	drink	alcoholic
of	course	after	of	course	after
the	heavy	lectures	the	heavy	lectures
involving	quantum	mechanics	involving	quantum	mechanics

2.3. 複雑な表の作成

今までの例で挙げた表組みには以下の共通点があります。

- 左揃え、中央揃えなどのセルの整列条件は列ごとに定まっており、同一列の中で揃え方が異なることはない。
- セルの結合がない。

実際、上で挙げた機能が無くとも多くの表を組むことができます。しかし、中にはセルの結合や特定箇所の書式変更を必要とするような、複雑な表を組みたい場合もあるでしょう。easytable は、そのような複雑な表にも対応しています。

上で示したような複雑な構造を持つ表を作成するには、`+easytable` コマンドの 2 番目のオプション引数を用います。

```
+easytable?:[
  t; b; m 2; m 4;
  m ?:(1, 3) 1; m ?:(3, 5) 1;
]?:[
  set-fmt-(3, 0) r;
  merge (0, 1) (0, 2) c;
  merge (0, 3) (0, 4) c;
  merge (4, 1) (5, 2) c;
  merge (4, 4) (5, 4) r;
convert-text-range (0, 0) (5, 0) (fun it -> {\emph{#it;}});
] [l; c; c; r; r] {
  | Program | Answer          || Time [ms]      |
  |         | case A | case B | case A | case B
  | foldn   | Yes    | Yes    | ${4}   | ${16}
```

```
  | mc91    | No     | -       | ${24}  | -
  | rev     | Yes    |         | ${4}   | ${78}
  | max     |        |         | ${8}   |
  |}
```

Program	Answer		Time [ms]	
	case A	case B	case A	case B
foldn	Yes	Yes	4	16
mc91	No	–	24	–
rev	Yes		4	78
max			8	

引数がだいぶややこしくなってきました。上の例では、`+easytable` コマンドに `?:` から始まる 2 つのオプション引数を渡しています。`?:[t; b; ...]` という箇所は 1 番目のオプション引数であり、前の節で紹介したように罫線を指定します。その次の `?:[set-fmt-(3, 0) r; ...]` という箇所は「セルの書式設定をデフォルトから変更する」ための設定を並べたものです。

`set-fmt- (r, c) fmt` という関数を指定すると、`r` 行 `c` 列 (0-based) のセルの書式を `fmt` へと変更することができます。上の例では 4 行目の 1 列目にある `mc91` というテキストに `r` 形式を指定することで、本来は左揃えで表示されるはずのセルが右揃えで表示されています。

`merge (rs, cs) (re, ce) fmt` という関数を指定すると、`rs` 行 `cs` 列 (0-based) のセル及び `re` 行 `ce` 列 (0-based) のセルを対角線に持つような長方形状のブロックを結合し、`fmt` 書式で表示することができます。上の例の `merge (5, 2) (2, 2) c` では、5 行目の 2 列目にある `Yes` と書かれたセルが 2 行 2 列へと拡張され、さらに中央揃えにして表示されています。

`convert-text-range (rs, cs) (re, ce) fmt` という関数を指定すると、`rs` 行 `cs` 列 (0-based) のセル及び `re` 行 `ce` 列 (0-based) のセルを対角線に持つような長方形状のブロックのテキストをまとめて変換することができます。上の例では、*Program* の列にあるセルがまとめて選択され、`(fun it -> {\emph{#it;}})` という変換が一括で行われた結果、該当するセルがすべて強調表示されています。

2.4. その他の機能

+list-table コマンドを用いると、箇条書きの構文を用いて表を組むことができます。特に表の中に長い文章を入れたいときに重宝します。

```
+list-table[r; lw 140pt; lw 120pt;]{
  *
    ** 言語
    ** 代表的なパングラム
    ** 代表的な回文
  *
    ** 日本語
    ** いろはにほへとちりぬるを
       わかよたれそつねならむ
       うゐのおくやまけふこえて
       あさきゆめみしゑひもせす
    ** 長き夜の遠の睡りの皆目醒め波乗り船の音の良きかな
  *
    ** 英語
    ** The quick brown fox jumps over the lazy dog.
    ** Some men interpret nine memos.
}
```

言語	代表的なパングラム	代表的な回文
日本語	いろはにほへとちりぬるをわかよたれそつねならむうゐのおくやまけふこえてあさきゆめみしゑひもせす	長き夜の遠の睡りの皆目醒め波乗り船の音の良きかな
英語	The quick brown fox jumps over the lazy dog.	Some men interpret nine memos.

+list-table も +easytable と同様に 2 種類のオプション引数をとることができます。引数の型や意味は +easytable と変わりません。また、インライン版の \list-table コマンドも存在し、同様の引数を取ることができます。

列の書式設定についてまだ説明していない機能が 2 つあるので、ここで紹介します。以下の例は、左右のマージンの調整及び均等割り付けを行ったものです。

```
+easytable?:[t; b; m 1; v 2]
  ?:[ merge (0, 0) (0, 1) c ]
  [r |> hmgn 10pt 0pt; l |> hmgn 0pt 10pt; e]{
    |日付 || 2020 年 の 祝 日
    |  1/|1  |元 日
    |  1/|13 |成 人 の 日
    |  2/|11 |建 国 記 念 の 日
    |    |   |(中略)
    |  9/|22 |秋 分 の 日
    | 11/|3  |文 化 の 日
    | 11/|23 |勤 労 感 謝 の 日
    |}
```

日付	2020 年 の 祝 日
1/1	元 日
1/13	成 人 の 日
2/11	建 国 記 念 の 日
	(中略)
9/22	秋 分 の 日
11/3	文 化 の 日
11/23	勤 労 感 謝 の 日

1 列目では列の設定を `r` ではなく `r |> hmgn 10pt 0pt` とすることで、単に右揃えなだけではなく、追加で 1 列目の左側のマージンを 10pt に、右側のマージンを 0pt にするという設定を加えています。2 列目のマージンも同様に設定して左側のマージンを 0pt とすることにより、1 列目と 2 列目の間に余白が無くなり、スラッシュの位置で日付が揃うようなレイアウトが実現されています。他の応用として、小数を小数点の位置で揃えて表記したいときなどにも便利といえるでしょう。

また 3 列目では均等割り付けを行っています。`e` を用いると、該当する列を均等割り付けで表示することができます。ただし均等割り付けのときに間に伸縮する余白が挟まるのは原則

としてインラインテキスト記述時に半角空白などの空白が挟まっているときです。現に「（中略）」と書かれたセルは均等割り付けの列にあるにもかかわらず、余白は挟まっていません。

3. Easytable パッケージが提供する機能

3.1. コマンド

本パッケージが提供するコマンドは 2 つです。`ruleptn` 及び `cellfmt` 型の説明は後述します。

`+easytable?:[rule-pattern]?:[cell-convert][cell-format]{| contents |}`

`{| contents |}` の内容を表にします。

引数	型	種類	内容
`rule-pattern`	`ruleptn list`	Optional	罫線の設定
`cell-convert`	`(builder -> builder) list`	Optional	セルの書式設定の変更内容
`cell-format`	`cellfmt list`	Needed	n 番目の列の体裁（揃え方）
`contents`	`inline-text list`	Needed	表の中身

`+list-table?:[rule-pattern]?:[cell-convert][cell-format]{* contents }`

`{* contents }` の内容を表にします。

引数	型	種類	内容
`rule-pattern`	`ruleptn list`	Optional	罫線の設定
`cell-convert`	`(builder -> builder) list`	Optional	セルの書式設定の変更内容
`cell-format`	`cellfmt list`	Needed	n 番目の列の体裁（揃え方）
`contents`	`itemize`	Needed	表の中身

`\easytable?:[rule-pattern]?:[cell-convert][cell-format]{| contents |}`

`+easytable` のインラインコマンド版。

`\list-table?:[rule-pattern]?:[cell-convert][cell-format]{* contents }`

`\list-table` のインラインコマンド版。

3.2. セルの書式指定

既に見たとおり、1 番目の必須引数である cell-format（cellfmt list 型） を用いることでセルの書式を列ごとに指定することができます。 ここで cellfmt とは context -> inline-text -> cell 型のシノニムであり、「テキスト処理文脈と表のコンテンツが与えられたとき、それをセルへと変換する」ための関数です。

cell-format の要素には自分で定義した関数を指定することもできるものの、 主要と思われるものについては EasyTable モジュールにて用意されています。 さらにそれらのほとんどには EasyTableAlias モジュールにてより短い名前の関数が当てがわれているため、長い名前をいちいち指定する必要がありません。 ユーザが好みに応じて EasyTableAlias モジュールを open することで、 あたかも LaTeX の tabular 環境のように簡便に書式を指定できる仕組みとなっています。

EasyTable.align-left *(alias:* EasyTableAlias.l*)*

cellfmt 型。入力されたテキストをそのまま組み、左揃えにして表示します。

EasyTable.align-center *(alias:* EasyTableAlias.c*)*

cellfmt 型。入力されたテキストをそのまま組み、中央揃えにして表示します。

EasyTable.align-right *(alias:* EasyTableAlias.r*)*

cellfmt 型。入力されたテキストをそのまま組み、右揃えにして表示します。

EasyTable.align-left-with-width *(alias:* EasyTableAlias.lw*)*

length -> cellfmt 型。 入力されたテキストをそのまま組み、 得られた横幅が指定した長さより短ければそのまま左揃えにして表示し、長ければ行を折り返します。

SATySFI の行分割アルゴリズムを使用しているため、 コストの関係で折り返しが最適解と判断されなければ、折り返されない場合もあります。

EasyTable.align-center-with-width *(alias:* EasyTableAlias.cw*)*

length -> cellfmt 型。 入力されたテキストをそのまま組み、 得られた横幅が指定した長さより短ければそのまま中央揃えにして表示し、長ければ行を折り返します。

EasyTable.align-right-with-width *(alias:* EasyTableAlias.rw*)*

length -> cellfmt 型。 入力されたテキストをそのまま組み、 得られた横幅が指定した長さより短ければそのまま右揃えにして表示し、長ければ行を折り返します。

EasyTable.equal-spacing *(alias:* EasyTableAlias.eq-sp*)*

cellfmt 型。 入力されたテキストを均等割り付けにします。 単語間空白の伸長方向の

ペナルティを極端に下げるという単純な実装のため、インラインテキストの時点で半角空白などの空白がないものには恐らく効果がありません。

`EasyTable.equal-spacing-with-width` *(alias:* `EasyTableAlias.eq-wd`*)*

`length -> cellfmt` 型。横幅を指定して、入力されたテキストを均等割り付けにします。

さらに、`cellfmt` 型の値を変換するための以下の関数を用意しています。

`EasyTable.hmargin` *(alias:* `EasyTableAlias.hmgn`*)*

`length -> length -> cellfmt -> cellfmt` 型。`cfmt |> EasyTable.hmargin len1 len2` とすることで、`cfmt` の書式のマージンを左側が `len1`、右側が `len2` となるように変更します。なお、特に `EasyTable.hmargin` を用いなかったときに定まるデフォルトのマージンは左右ともにフォントサイズの 0.5 倍です。

`EasyTable.hmargin-ratio` *(alias:* `EasyTableAlias.hmgnr`*)*

`float -> float -> cellfmt -> cellfmt` 型。`cfmt |> EasyTable.hmargin-ratio r1 r2` とすることで、`cfmt` の書式のマージンを左側が元の `r1` 倍、右側が `r2` 倍となるように変更します。

`EasyTable.vmargin` *(alias:* `EasyTableAlias.vmgn`*)*

`length -> length -> cellfmt -> cellfmt` 型。`cfmt |> EasyTable.vmargin len1 len2` とすることで、`cfmt` の書式のマージンを上が `len1`、下が `len2` となるように変更します。なお、特に `EasyTable.vmargin` を用いなかったときに定まるデフォルトのマージンは上下ともにフォントサイズの 0.4 倍です。

`EasyTable.vmargin-ratio` *(alias:* `EasyTableAlias.vmgnr`*)*

`float -> float -> cellfmt -> cellfmt` 型。`cfmt |> EasyTable.vmargin-ratio r1 r2` とすることで、`cfmt` の書式のマージンを上が元の `r1` 倍、下が `r2` 倍となるように変更します。

3.3. 罫線の設定

`+easytable` などにおける 1 番目のオプション引数では、`ruleptn list` 型の値を指定することで罫線の引き方を指定することができます。`ruleptn` は `length list -> length list -> graphics list` 型のシノニムであり、この型の意味は tabular プリミティヴの第 2 引数とほぼ変わりません。すなわち、表の格子点に相当する点の x 座標及び y 座標のリストが与えられたとき、（イメージとしては、表の内容が決まって「セルの境界線」が定まった

とき）線をどこにどのように引くべきかを指定したグラフィックスのリストを返す、というものです。こちらもやはり自分で定義した関数を指定できますが、主要なものは予め用意されています。

`EasyTable.toprule` *(alias:* `EasyTableAlias.t`*)*

`ruleptn` 型。表の一番上に太い黒線を引きます。

`EasyTable.bottomrule` *(alias:* `EasyTableAlias.b`*)*

`ruleptn` 型。表の一番下に太い黒線を引きます。

`EasyTable.vertrule` *(alias:* `EasyTableAlias.v`*)*

`int -> ruleptn` 型。`vertrule n` とすると、左から数えて n 番目のインデックスを持つところに細い黒の鉛直線を引きます。

`EasyTable.horizrule` *(alias:* `EasyTableAlias.h`*)*

`int -> ruleptn` 型。`horzrule n` とすると、上から数えて n 番目のインデックスを持つところに細い黒の水平線を引きます。

`EasyTable.midrule` *(alias:* `EasyTableAlias.m`*)*

`int -> ruleptn` 型。`horzrule` と似ていますが、こちらは両端が少し短くなります。

`EasyTable.diagrule` *(alias:* `EasyTableAlias.d`*)*

`(int * int) -> (int * int) -> ruleptn` 型。`diagrule (i1, j1) (i2, j2)` とすると、右上から数えて `(i1, j1)` 番目にある格子点から `(i2, j2)` 番目にある格子点にかけて細い黒の直線を引きます。

`EasyTable.outerframerule` *(alias:* `EasyTableAlias.rect`*)*

`ruleptn` 型。表に太い黒の外枠を付けます。

罫線だけでなく、表の背景色も指定できます。

`EasyTable.whole-bgcolor` *(alias:* `EasyTableAlias.bg-a`*)*

`color -> ruleptn` 型。表全体の背景色を指定した色に設定します。

`EasyTable.column-bgcolor` *(alias:* `EasyTableAlias.bg-c`*)*

`color -> int -> int -> ruleptn` 型。`column-bgcolor clr n m` で、`n` 番目から `m` 番目にかけての列を指定した色で塗り潰します。

`EasyTable.row-bgcolor` *(alias:* `EasyTableAlias.bg-r`*)*

`color -> int -> int -> ruleptn` 型。`row-bgcolor clr n m` で、`n` 番目から `m` 番

目にかけての行を指定した色で塗り潰します。

3.4. セル単位での書式設定

+easytable などにおける 2 番目のオプション引数では、(builder -> builder) list 型の値を指定することで特定のセルの書式を変更したり、セルを結合したりといった操作ができるようになります。builder は正確には TableBuilder.builder という型であり、表を組むときの cellfmt や ruleptn を保持したデータ構造です。TableBuilder モジュールのメソッドを用いて cellfmt 等の中身を「強制的に組み替える」ことで、表に関するおよそあらゆる類の編集を可能にします。このように、2 番目のオプション引数は自由度の高さという意味では最も強力であるものの、指定方法が複雑であり多用しすぎるとコードの可読性を損ねるおそれがあります。必須引数の範囲で可能な限りレイアウトを指定し、セルの結合などどうしても必要な箇所だけこのオプションで微調整する、という使い方をお勧めします。

TableBuilder.merge *(alias:* EasyTableAlias.merge*)*

int * int -> int * int -> cellfmt -> builder -> builder 型。

merge (rs, cs) (re, ce) fmt で、(rs, cs) 番地のセル（0-based index で上から数えて rs 番目、左から数えて cs 番目にあるセル）及び (re, ce) 番地のセルを対角線に持つ長方形状のブロックをまとめて結合します。結合後のテキストは最も左上のセル、すなわちの (rs, cs) 番地のセルの中身が引き継がれ、それ以外のセルに格納されていたテキストは全て無視されます[*2]。結合後のセルの書式には fmt で指定したものが使用されます。

TableBuilder.set-fmt- *(alias:* EasyTableAlias.set-fmt-*)*

int * int -> cellfmt -> builder -> builder 型。

set-fmt- (r, c) fmt で、(r, c) 番地のセルの書式を fmt に変更します。特定のセルのみ書式を変更したい場合に便利です。複数のセルの書式を塊で変更したい場合は後述の set-fmt-range を用いることができます。

TableBuilder.set-fmt-range *(alias:* EasyTableAlias.set-fmt-range*)*

int * int -> int * int -> cellfmt -> builder -> builder 型。

set-fmt- の範囲指定版。set-fmt-range (rs, cs) (re, ce) fmt で、(rs, cs) 番地のセル及び (re, ce) 番地のセルを対角線に持つ長方形状のブロックの書式をまとめ

2 可読性を可能な限り損ねないよう、それ以外のセルには空白や空行のみを入れたほうが良いでしょう。

て fmt に変更します。

TableBuilder.convert-text-range *(alias:* EasyTableAlias.convert-text-range*)*

int * int -> int * int -> (inline-text -> inline-text) -> builder -> builder 型。

convert-text-range (rs, cs) (re, ce) convertf で、(rs, cs) 番地のセル及び (re, ce) 番地のセルを対角線に持つ長方形状のブロックのテキストを convertf でまとめて変換します。 たとえば convertf に (fun it -> {\emph{#it;}}) を指定すれば、指定したセルのテキストがまるまる強調表示されるようになります。「レイアウトの都合上ヘッダ部分には太字のフォントを用いたいが、いちいち同じコマンドをセル毎に噛ませるのは面倒だ」という場合に便利です。