VarTable

VarTable est un paquet pour rendre la réalisation
des tableaux de signe plus simple
Ce paquet est construit sur Cetz
(version : 0.2.1)

Table des Matières

1 - Introduction

Ce paquet a été réalisé pour rendre la création de tableau de signe plus simple. Pour cela, ce paquet fournis la fonction « `tabvar` », dont les arguments sont décrit dans cette documentation.

Si vous rencontrez un bug, merci de me prévenir via mon GitHub.

2 - Tabvar

2.1 - déscription générale

tabvar

Retourne un tableau de variation

Parameters

```
tabvar(
  variable: content,
  label: array,
  domain: array,
  contents: array,
  table-style: style,
  nocadre: bool,
  arrow-mark: mark,
  arrow-style: style,
  line-0: bool,
  line-style: style,
  hatching-style: tiling,
  first-column-width: length,
  first-line-height: length,
  element-distance: length,
  values: array,
  add: content
)
```

variable `content`

`variable` est la variable qui contient la variable du tableau (comme x ou t)
Exemple : si la variable de la fonction est t, alors :

```
variable : $ t $
```

Default: `$ x $`

label `array`

`label` est un array qui contient des array de longueur 2, une pour chaque ligne du tableau, dont le premier élément est le titre de la ligne et le second est le type de la ligne : signe (`s`) ou variation (`v`)
Exemple : pour le tableau de variation de la fonction f, vous devriez écrire :

```
label : (
  ([Signe de $f$], "s"), // la première ligne est un tableau de signe
  ([Variation de $f$], "v") // la seconde ligne est un tableas de variation
)
```

Default: `()`

domain `array`

les valeurs prises par la variable
par exemple, si votre fonction change de signe ou atteind un extremum pour $x \in \{0, 1, 2, 3\}$
vous devriez écrire :

```
domain: ($0$, $1$, $2$, $3$)
```

Default: `()`

contents `array`

le contenu de la table
voir 2.2 pour plus de détaille

Default: `((),)`

table-style `style`

Optionelle
Le style de la table
le type style est définis par Cetz, ainsi je vous recommande de vous référer au manuelle de Cetz.
Attention : Si vous ne mettez pas le paramètre de style : `mark` a `none`, alors toute les lignes du tableau aurons une tête en flèche

Default: `(stroke: 1pt + black, mark: (symbol: none))`

nocadre `bool`

Pour cacher le cadre externe du tableau

Default: `false`

arrow-mark `mark`

Le style de la tête de flèche.
N.B. le type `mark` est définis par Cetz

Default: `(end: "straight")`

arrow-style `style`

Optionelle :
Le style des flèches.
Attention : le paramètre `mark` est supplenté par le paramètre `arrow-mark`

Default: `(stroke: black + 1pt)`

line-0 `bool`

Optionelle
si vous voulez changer la bar par défaut dans les tableaux de signe, pour une bar avec un zéro en sont centre

Default: `false`

line-style `style`

Optionelle
Si vous voulez le style de toutes les bars de séparation entre les signes

Default: `(stroke: black + 1pt)`

hatching-style `tiling`

Optionelle
le style des hachures s'il y a des zones hachurées

Default: `tiling(size: (30pt, 30pt))[`

```
    #place(line(start: (0%, 100%), end: (100%, 0%), stroke: 2pt))
    #place(line(start: (-100%, 100%), end: (100%, -100%), stroke: 2pt))
    #place(line(start: (0%, 200%), end: (200%, 0%), stroke: 2pt))
  ]
```

first-column-width `length`

Optionelle
change la largeur de la première colonne

Default: `none`

first-line-height `length`

change la hauteur de la première ligne (celle du domaine et de la variable)

Default: `none`

element-distance `length`

Optionelle
change la distance entre deux éléments

Default: `none`

values `array`

pour ajouter des valeurs entre deux valeurs prè-définis

Default: `((),)`

add `content`

Pour ajouter plus d'éléments via Cetz

Default: `()`

2.2 - Le paramètre de contenus

Le paramètre contenu est un array avec un élément par ligne (par label).

Chaque éléments sont eux même des array avec un élément pour chaque colonne, avec un format différents pour les signes et les variations qui seront détaillés ci-dessous.

2.2.1 - Le format pour les Signes

Il doit être possisioné au même index dans l'array `contents` que un label possèdant le string `"s"`, ce qui indique que la ligne doit être considéré comme un tableau de signe

De plus, il doit contenir autemps d'éléments que le domaine moins un (un par interval), plus un argument optionelle pour pour le sytle de la bar

Chaque éléments doits être d'une de ces forme, différentes formes peuvent être utilisées sur une même ligne :

`()` - Vide : pour étendre le dernier signe en partant de la gauche sur les intervals marqués vides
`body` - Le cas basique, constitué du type body de typst, comme `$ + $` ou `$ - $`
`(style de la bar, body)` - Pour spécifier un style particulier à la bar de **devant** le signe, ce style peut être : `"|"` la bar simple, `"||"` une double bar ou `"0"` pour une bar avec un zéro en sont centre
NB : le paramètre `line-0` change la bar par défaut pour la bar avec un zéro `"0"`.

Vous pouvez mettre en plus à la fin le string `"||"`, pour rajouter un double bar à la toute fin

2.2.1.1 - Un array classique pour les signes

Un tableau de signe classique :

```
#tabvar(
  init: (
    variable: $t$,
    label: (([signe], "s"),),
  ),
  domain: ($2$, $4$, $6$, $8$),
  contents: (($+$, $-$, $ + $),),
)
```

t	2	4	6	8
signe	+	−	+	

Un example plus complexe :

```
#tabvar(
  variable: $t$,
  label: (
    ([signe], "s"),
  ),
  domain: ($ 2 $, $4$, $6$, $8$),
  contents: (
    ("Hello world !", $-$, $ 3 / 2 $),
  ),
)
```

t	2	4	6	8
signe	Hello world !	−	$\frac{3}{2}$	

Note : Sur le second example, le tableau est comprimé à l'aide de la fonction scale

2.2.1.2 - Une bar de séparation customisé

2.2.1.2.1 - Le style de la bar

Vous pouvez modifier le style de la bar

Le style de la bar est un dictionary, du type `"style"` définis par Cetz.
Pour fair simple, si vous voulez changer uniquement le stroke des bars, vous avez juste a mettre `(stroke: votre stroke)`.
Pour des usages plus complexe référer vous au manuel de Cetz.

Example :

```
#tabvar(
  line-style: (
    stroke: (paint: red, dash: "dashed")
  ),
  variable: $t$,
  label: (([signe], "s"),),
  domain: ($2$, $4$, $6$,),
  contents: (
    ($+$, $-$),
  ),
)
```

t	2	4	6
signe	+	−	

2.2.1.2.2 - Le type de la bar

Pour tout les signes sauf le premier, au lieux de placé directement un signe, vous pouvez mettre un couple, dont le premier éléments définis le type de la bar placée avant le signe.
Il y a trois type différents de bar :

- `"|"` : une bar simple
- `0` : une bar avec un zéro en sont centre
- `||` une double bar, pour les valeurs non-définis

Exemple

```
#tabvar(
  variable: $t$,
  label: (([signe], "s"),),
  domain: ($2$, $4$, $6$, $8$, $10$),
  contents: (
    (
      $+$,
      ("|", $-$),
      ("0", $-$),
      ("||", $+$)
    ),
  ),
)
```

t	2	4	6	8	10
signe	+	−	0	−	+

Si vous voulez avoir une double bar avant le premier signe, vous pouvez utilisez le couple avec en permier éléments `"||"`, à la place du premier signe ; pour mettre une double bar à la fin, ajoutez à la fin de l'array le string `"||"`.

Example :

```
#tabvar(

  variable: $t$,
  label: (([signe], "s"),),

  domain: ($2$, $4$, $6$),
  contents: (
    (
      ("||", $+$),
      $-$,
      "||"
    ),
  ),
)
```

t	2	4	6
sign	+	−	

2.2.1.3 - Un même signe pour plus d'une seul valeur

Quand votre tableau de signe possède plus d'un sous tableau, alors vous seriez tanté de vouloir mettre un même signe pour plusieurs valeurs du domaine.
Pour celà c'est assez simple, au lieux de mettre un signe directement, mettez simplement un couple vide `()`

Example :

```
#tabvar(
  line-0: true,
  variable: $t$,
  label: (
    ([signe], "s"),
  ),
  domain: ($2$, $4$, $6$, $8$),
  contents: (
    ($+$, (), $-$),
  ),
)
```

t	2	4	6	8
signe		+		−

2.2.1.4 - Hachurage pour une zone non définis pour les sous tableaux de signes

Il se peut que vos fonctions ne soient pas définis sur un ou plusieurs interval malheuresement présent dans le domaine du tableau de signe, pour celà la convention veut que l'on hache la zone en question.
Étant donnée que les signes porte sur les intervals du domaine, il en résulte une syntaxe ralativement simple d'usage, dont on pourait distinguer 4 cas :

- le premier cas et le plus courant, celuis où les deux bornes de l'interval indéfini le sont également, ainsi à la place où vous auriez mis votre signe (ou tout autres éléments), vous renseignerez l'élément suivant : `"|h|"`
- le second cas, également relativement présent, est celuis où les deux bornes elle définits contrairement cette fois à l'interval, ainsi vous omettrez les deux bar « | » de l'élément présenté ci-dessus, i.e. vous renseignerez `"h"`

- les deux autres cas, moins courant mais pouvant tout de même apparaitre, est celuis où seul l'une des deux bornes est définis, ainsi, comme vous l'auriez sans doute compris, retirer (resp. rajouter) la bare pour le côter où l'élément est défini (resp. indéfinis), soit : pour une valeurs définis à gauches `"h|"`; pour une valeur définis à droite `"|h"`

Remarque : Vous avez sans doute compris que la bare « | » symbolise les doubles bares indéfini, de même que le « h » représente le « h » de hachurage, ainis il est naturel de mettre ou non les bares au besoins

Pour étandre le hachurage sur plus d'un des intervals du domaine, il vous suffie de sauté l'élément suivant avec toujour la même notation, à savoir `()`

Example :

```
#tabvar(
  line-0: true,
  variable: $t$,
  label: (
    ([signe], "s"),
  ),
  domain: ($2$, $4$, $6$, $8$),
  contents: (
    ($+$, (), $-$),
  ),
)
```