

Le package `lymath` : des formules plus sémantiques

Code source disponible sur <https://github.com/bc-latex/ly-math>.

Version 0.6.3-beta développée et testée sur Mac OS X.

Christophe BAL

2019-10-21

Table des matières

1	Introduction	4
2	Comment lire cette documentation ?	4
3	A propos des macros	4
3.1	Règles de nommage	4
3.1.1	Les macros de même « type »	4
3.1.2	Les macros en mode <code>displaystyle</code>	4
3.2	Versions étoilées	5
3.3	Les arguments, deux conventions à connaître	5
3.3.1	Nombre fixé d'arguments	5
3.3.2	Nombre variable d'arguments	5
4	Deux séparateurs d'arguments par défaut	5
5	Quelques gestions d'espaces	5
5.1	Espace et fraction	5
5.2	Espace et racines n-ièmes d'un réel	6
5.3	Sommes et produits en mode ligne	6
5.4	Espace et point-virgule avec l'option <code>french</code> de <code>babel</code>	6
6	Logique et fondements	6
6.1	Différents types d'égalités « standard »	6
6.1.1	Définir quelque chose	7
6.1.2	Indiquer une identité	7
6.1.3	Une égalité à vérifier ou non, une hypothèse, une condition	7
6.1.4	Une égalité indiquant le choix d'une valeur	7
6.1.5	Une égalité indiquant l'équation d'une courbe que l'on utilise	8
6.1.6	Différents types d'inéquations	8
6.1.7	Une table récapitulative	8

6.1.8	Textes utilisés	8
6.1.9	Fiches techniques	9
6.2	Équivalences et implications	9
6.2.1	Des symboles supplémentaires	9
6.2.2	Équivalences et implications verticales	11
6.3	Tables des décorations possibles des opérateurs	12
6.4	Détailler un raisonnement	12
6.5	Des versions alternatives du quantificateur \exists	14
7	Ensembles et applications	15
7.1	Différents types d'ensembles	15
7.1.1	Ensembles versus accolades	15
7.1.2	Ensembles pour la géométrie	15
7.1.3	Ensembles probabilistes	16
7.1.4	Ensembles pour l'algèbre générale	16
7.2	Ensembles classiques en mathématiques et en informatique théorique	17
7.2.1	La liste complète	17
7.2.2	Ensembles classiques suffixés	17
7.3	Des suffixes à la carte	18
7.4	Intervalles	19
7.4.1	Intervalles réels - Notation française (?)	19
7.4.2	Intervalles réels - Notation américaine	20
7.4.3	Intervalles discrets d'entiers	20
7.5	Application totale, partielle, injective, surjective et/ou bijective	21
8	Géométrie	22
8.1	Points et lignes	22
8.1.1	Points	22
8.1.2	Droites parallèles ou non	23
8.2	Vecteurs	23
8.2.1	Les écrire	23
8.2.2	Norme	24
8.2.3	Produit scalaire – Écriture minimaliste	24
8.2.4	Produit scalaire – Écriture « physicienne »	25
8.2.5	Produit vectoriel	25
8.3	Coordonnées	26
8.4	Nommer un repère	27
8.5	Arcs circulaires	29
8.6	Angles	29
8.6.1	Angles géométriques intérieurs	29
8.6.2	Angles orientés de vecteurs	30
9	Analyse	31
9.1	Constantes	31
9.1.1	Constantes classiques	31
9.1.2	Constantes latines personnelles	31
9.2	La fonction valeur absolue	32
9.3	Fonctions nommées spéciales	32
9.3.1	Sans paramètre	32
9.3.2	Avec un paramètre	33
9.4	Des notations complémentaires pour des suites spéciales	33

9.5	Calcul différentiel	34
9.5.1	Les opérateurs ∂ et d	34
9.5.2	Dérivation totale	34
9.5.3	Dérivation partielle	35
9.6	Calcul intégral	36
9.6.1	L'opérateur crochet – 1 ^{ère} version	36
9.6.2	L'opérateur crochet – 2 ^{nde} version	37
9.6.3	Intégrales multiples	37
9.7	Tableaux de variation et de signe	37
9.8	Comparaison asymptotique de suites et de fonctions	39
9.8.1	Les notations \mathcal{O} et \mathcal{o}	39
9.8.2	La notation Ω	39
9.8.3	La notation Θ	40
10	Probabilité	40
10.1	Probabilité conditionnelle	40
10.2	Arbres pondérés	41
11	Arithmétique	44
11.1	Opérateurs de base	44
11.2	Fractions continuées	45
11.2.1	Fractions continuées standard	45
11.2.2	Fractions continuées généralisées	45
11.2.3	Comme une fraction continuée isolée	45
11.2.4	L'opérateur \mathcal{K}	46
12	Algèbre	47
12.1	Polynômes, séries formelles et compagnie	47
12.1.1	Polynômes et fractions polynômiales	47
12.1.2	Séries formelles et leurs corps de fractions	47
12.1.3	Polynômes de Laurent et séries formelles de Laurent	48
12.1.4	Toutes les fiches techniques	48
12.2	Matrices	48
13	Historique	50

1 Introduction

L^AT_EX est un excellent langage, pour ne pas dire le meilleur, pour rédiger des documents contenant des formules mathématiques. Malheureusement toute la puissance de L^AT_EX permet d'écrire des codes très peu sémantiques. Le modeste but du package `lymath` est de fournir quelques macros sémantiques pour la rédaction de formules mathématiques élémentaires. Considérons le code L^AT_EX suivant.

```
Sachant que  $\frac{df}{dx}(x) = 4 \cos(x^2)$  sur  $[a ; b]$  , nous avons :  
 $\int_a^b \cos(x^2) dx = \left[ \frac{1}{4} f(x) \right]_a^b$ .
```

Avec `lymath`, vous pouvez écrire le code suivant.

```
Sachant que  $\frac{d}{dx}f(x) = 4 \cos(x^2)$  sur  $\text{intervalC}\{a\}\{b\}$ , nous avons :  
 $\int_a^b \cos(x^2) \, dd{x} = \hook{\frac{1}{4} f(x)}{a}\{b\}$ .
```

Même si certaines commandes sont plus longues à écrire que ce que permet L^AT_EX, il y a trois avantages à utiliser des commandes sémantiques.

1. La mise en forme dans votre document sera consistante.
2. Il est facile de changer une mise en forme sur l'ensemble d'un document.
3. `lymath` résout certains problèmes "complexes" pour vous.

2 Comment lire cette documentation ?

Le choix a été fait de fournir des exemples comme documentation du package suivis de fiches techniques des macros-commandes. Les exemples se présentent comme ci-dessous (*un code L^AT_EX suivi de sa mise en forme*).

```
Sachant que  $\displaystyle \frac{df}{dx}(x) = 4 \cos(x^2)$  sur  $[a ; b]$  , nous avons :  
 $\displaystyle \int_a^b \cos(x^2) dx = \left[ \frac{1}{4} f(x) \right]_a^b$ .
```

Sachant que $\frac{df}{dx}(x) = 4\cos(x^2)$ sur $[a ; b]$, nous avons : $\int_a^b \cos(x^2)dx = \left[\frac{1}{4} f(x) \right]_a^b$.

3 A propos des macros

3.1 Règles de nommage

3.1.1 Les macros de même « type »

Les macros partageant une même fonctionnalité mathématique auront toute le même préfixe comme par exemple pour `\derpow` , `\derfrac` ... utilisables pour rédiger des dérivées de fonctions. Ce choix est assumé même si pour les macros du type `\set...` on obtient un nom pouvant faire penser à « régler ... » au lieu de « ensemble du type ... ».

3.1.2 Les macros en mode `displaystyle`

Les macros évitant d'avoir à taper `\displaystyle` auront un nom commençant par la lettre d.

3.2 Versions étoilées

Les versions étoilées proposent des mises en forme correspondant aux cas les moins usuels : par exemple une macro utilisant des parenthèses rendra ces dernières extensibles sauf dans sa version étoilée.

3.3 Les arguments, deux conventions à connaître

3.3.1 Nombre fixé d’arguments

Dans ce cas, c’est la syntaxe L^AT_EX usuelle qui sera à utiliser comme dans `\derfrac{f}{x}`.

3.3.2 Nombre variable d’arguments

Certaines macros offrent la possibilité de fournir un nombre variable d’arguments comme dans `\coord{x | y | z | t}` et `\coord{x | y}`. Ceci se fait en utilisant un seul argument, au sens de L^AT_EX, dont le contenu est formé de morceaux séparés par des traits verticaux `|`. Ainsi dans `\coord{x | y | z | t}`, l’unique argument `x | y | z | t`, au sens de L^AT_EX, sera analysé par `lyxam` comme étant formé des quatre arguments `x`, `y`, `z` et `t`.

4 Deux séparateurs d’arguments par défaut

La macro `\lymathsep` définit le séparateur d’arguments de premier niveau, et `\lymathsubsep` celui des arguments de deuxième niveau. Cette documentation utilisant l’option `french` de `babel`, la valeur de `\lymathsep` est `;` et celle de `\lymathsubsep` est `,`. Sans ce choix, les valeurs de `\lymathsep` et `\lymathsubsep` seront `,` et `;` respectivement.

Fiches techniques

`\lymathsep` <macro> (Sans argument)

`\lymathsubsep` <macro> (Sans argument)

5 Quelques gestions d’espaces

5.1 Espace et fraction

Quand on utilise `\frac` ou `\dfrac`, de petits espaces sont automatiquement ajoutés pour éviter d’avoir des traits de fraction trop petits. Le comportement par défaut se retrouve en utilisant les macros `\stdfrac` et `\stdfrac`. Voici un exemple.

Vous avez `$(\frac{2}{3}) = (\stdfrac{2}{3})$` et `$(\dfrac{2}{3}) = (\stdfrac{2}{3})$`.

Vous avez $\frac{2}{3} = \frac{2}{3}$ et $\frac{2}{3} = \frac{2}{3}$.

Fiches techniques

`\frac` <macro> (Sans argument)

`\dfrac` <macro> (Sans argument)

`\stdfrac` <macro> (Sans argument)

`\stdfrac` <macro> (Sans argument)

5.2 Espace et racines n-ièmes d'un réel

`\sqrt` a été redéfini pour ajouter un peu d'espaces. Le comportement par défaut se retrouve en utilisant la macro `\stdsqrt`. Voici un exemple.

`\$ \sqrt{2} = \stdsqrt{2} \$` et `\$ \sqrt[n]{45} = \stdsqrt[n]{45} \$` .

Vous avez $\sqrt{2} = \sqrt{2}$ et $\sqrt[n]{45} = \sqrt[n]{45}$.

Fiches techniques

`\sqrt` <macro> (Sans argument)

`\stdsqrt` <macro> (Sans argument)

5.3 Sommes et produits en mode ligne

Pour limiter l'espace, L^AT_EX affiche $\sum_{k=0}^n$ et non $\sum_{k=0}^n$ sauf si l'on utilise la commande `\displaystyle`.

Les macros `\dsum` et `\dprod` permettent de se passer de `\displaystyle`. Voici un exemple.

`\$ \dsum_{k=0}^n 2^k = \sum_{k=0}^n 2^k = 2^{n+1} - 1 \$`

`\$ \dprod_{k=1}^n k = \prod_{k=1}^n k = k ! \$`

$$\sum_{k=0}^n 2^k = \sum_{k=0}^n 2^k = 2^{n+1} - 1$$

$$\prod_{k=1}^n k = \prod_{k=1}^n k = k !$$

Fiches techniques

`\dprod` <macro> (Sans argument)

`\dsum` <macro> (Sans argument)

5.4 Espace et point-virgule avec l'option french de babel

Seulement si vous utilisez `babel` avec l'option `french`, comme c'est le cas dans cette documentation, alors vous verrez le même espacement autour du point-virgule dans $A(x;y)$. Que c'est beau!

6 Logique et fondements

6.1 Différents types d'égalités « standard »

D'un point de vue pédagogique, il peut être intéressant de disposer de différentes façon d'écrire une égalité, une non égalité ou une inégalité. Bien entendu on tord les règles de typographie avec ce type de pratique mais c'est pour le bien de la communauté.

6.1.1 Définir quelque chose

L'exemple suivant montre trois façons de rédiger une égalité signifiant une définition¹ (la section 6.1.8 explique comment est défini le texte « déf »).

La fonction f est définie sur \mathbb{R} par $f(x) \stackrel{\text{def}}{=} x^3 + 1$ ou avec l'écriture symbolique $f(x) \stackrel{\text{def}}{=} x^3 + 1$.

La fonction f est définie sur \mathbb{R} par $f(x) \stackrel{\text{def}}{=} x^3 + 1$ ou avec l'écriture symbolique $f(x) := x^3 + 1$.

6.1.2 Indiquer une identité

L'exemple suivant montre deux façons de rédiger des identités, la notation symbolique n'étant pas standard (la section 6.1.8 explique comment est défini le texte « id »).

$\forall (a ; b) \in \mathbb{R}^2$, nous avons : $(a + b)^2 \stackrel{\text{id}}{=} a^2 + b^2 + 2 a b$.
On peut utiliser une écriture plus symbolique : $(a + b)^2 \stackrel{\text{id}}{=} a^2 + b^2 + 2 a b$.

$\forall (a ; b) \in \mathbb{R}^2$, nous avons : $(a + b)^2 \stackrel{\text{id}}{=} a^2 + b^2 + 2ab$. On peut utiliser une écriture plus symbolique : $(a + b)^2 \stackrel{\text{id}}{=} a^2 + b^2 + 2ab$.

6.1.3 Une égalité à vérifier ou non, une hypothèse, une condition

Se reporter à la section 6.1.8 pour savoir comment sont définis les textes « cond » et « hyp ».

Est-il vrai que $(a + b)^3 \stackrel{?}{=} a^3 + b^3 + 3 a b$?
Non car $(a + b)^3 \stackrel{\text{id}}{=} a^3 + b^3 + 3 a b$

A-t-on $x \neq 0$ pour pouvoir écrire $\frac{1}{x}$?

Comme x doit être non nul, je sais que $x \neq 0$.

Une autre condition $x \neq 0$ ou une autre hypothèse $x \neq 0$?

Est-il vrai que $(a + b)^3 \stackrel{?}{=} a^3 + b^3 + 3ab$? Non car $(a + b)^3 \stackrel{\text{id}}{\neq} a^3 + b^3 + 3ab$

A-t-on $x \stackrel{\text{hyp}}{\neq} 0$ pour pouvoir écrire $\frac{1}{x}$?

Comme x doit être non nul, je sais que $x \stackrel{\text{cond}}{\neq} 0$.

Une autre condition $x \stackrel{\text{cond}}{=} 0$ ou une autre hypothèse $x \stackrel{\text{hyp}}{=} 0$?

6.1.4 Une égalité indiquant le choix d'une valeur

La section 6.1.8 permet de savoir comment le texte « choix » est défini.

1. Le symbole peu courant $:=$ est utilisé par le langage B qui permet de spécifier et prouver certains programmes.

Sachant que si $x \geq 4$ alors $x^2 \geq 16$ alors $x = 123$ nous donne $123^2 \geq 16$.

Sachant que si $x \overset{\text{cond}}{\geq} 4$ alors $x^2 \overset{\text{cons}}{\geq} 16$ alors $x \overset{\text{choix}}{=} 123$ nous donne $123^2 \overset{\text{appli}}{\geq} 16$.

6.1.5 Une égalité indiquant l'équation d'une courbe que l'on utilise

La section 6.1.8 permet de savoir comment les textes « *graph* » et « *appli* » sont définis (la macro *setgeo* est définie dans la section 7.1.2).

Si $M(4; y_M) \in \mathcal{C} : y \overset{\text{graph}}{=} x^2 + 3$ alors $y_M \overset{\text{appli}}{=} 4^2 + 3 = 19$.

Si $M(4; y_M) \in \mathcal{C} : y \overset{\text{graph}}{=} x^2 + 3$ alors $y_M \overset{\text{appli}}{=} 4^2 + 3 = 19$.

6.1.6 Différents types d'inéquations

Le principe reste le même pour les symboles d'équations excepté qu'il n'y a ici aucune écriture purement symbolique. Voici un code « fourre-tout » montrant quelques exemples.

A-t-on $x \leq x^2$ ou $x < x^2$?

A moins que ce ne soit $x \geq x^2$ ou $x > x^2$ qu'il faille vérifier.

On peut supposer $x \leq 1$ ou avoir la condition $x > 2$.

A-t-on $x \overset{?}{\leq} x^2$ ou $x \overset{?}{<} x^2$?

A moins que ce ne soit $x \overset{?}{\geq} x^2$ ou $x \overset{?}{>} x^2$ qu'il faille vérifier.

On peut supposer $x \overset{\text{hyp}}{\leq} 1$ ou avoir la condition $x \overset{\text{cond}}{>} 2$.

6.1.7 Une table récapitulative

La table 1 page 12 fournit toutes les associations autorisées entre opérateurs de comparaison et décorations.

6.1.8 Textes utilisés

Voici les macros définissant les textes utilisés qui tiennent compte de l'utilisation ou non de l'option *french* de *babel*. Nous ne donnons que les versions françaises.

`\textopappli` donne « *appli* »

`\textopchoix` donne « *choix* »

`\textopcond` donne « *cond* »

`\textopcons` donne « *cons* »

`\textopdef` donne « *déf* »

`\textophyp` donne « *hyp* »

`\textopid` donne « *id* »

`\textopplot` donne « *graph* »

`\textoptest` donne « *?* »

6.1.9 Fiches techniques

Les opérateurs disponibles

<code>\eqappli <macro> (Sans argument)</code>	<code>\lhyp <macro> (Sans argument)</code>
<code>\eqchoice <macro> (Sans argument)</code>	<code>\ltest <macro> (Sans argument)</code>
<code>\eqcond <macro> (Sans argument)</code>	<code>\gappli <macro> (Sans argument)</code>
<code>\eqcons <macro> (Sans argument)</code>	<code>\gchoice <macro> (Sans argument)</code>
<code>\eqdef <macro> (Sans argument)</code>	<code>\gcond <macro> (Sans argument)</code>
<code>\eqdef* <macro> (Sans argument)</code>	<code>\gcons <macro> (Sans argument)</code>
<code>\eqhyp <macro> (Sans argument)</code>	<code>\ghyp <macro> (Sans argument)</code>
<code>\eqid <macro> (Sans argument)</code>	<code>\gtest <macro> (Sans argument)</code>
<code>\eqid* <macro> (Sans argument)</code>	<code>\leqappli <macro> (Sans argument)</code>
<code>\eqplot <macro> (Sans argument)</code>	<code>\leqchoice <macro> (Sans argument)</code>
<code>\eqtest <macro> (Sans argument)</code>	<code>\leqcond <macro> (Sans argument)</code>
<code>\neqappli <macro> (Sans argument)</code>	<code>\leqcons <macro> (Sans argument)</code>
<code>\neqid <macro> (Sans argument)</code>	<code>\leqhyp <macro> (Sans argument)</code>
<code>\neqchoice <macro> (Sans argument)</code>	<code>\leqtest <macro> (Sans argument)</code>
<code>\neqcond <macro> (Sans argument)</code>	<code>\geqappli <macro> (Sans argument)</code>
<code>\neqcons <macro> (Sans argument)</code>	<code>\geqchoice <macro> (Sans argument)</code>
<code>\neqhyp <macro> (Sans argument)</code>	<code>\geqcond <macro> (Sans argument)</code>
<code>\neqtest <macro> (Sans argument)</code>	<code>\geqcons <macro> (Sans argument)</code>
<code>\lappli <macro> (Sans argument)</code>	<code>\geqhyp <macro> (Sans argument)</code>
<code>\lchoice <macro> (Sans argument)</code>	<code>\geqtest <macro> (Sans argument)</code>
<code>\lcond <macro> (Sans argument)</code>	
<code>\lcons <macro> (Sans argument)</code>	

Les textes disponibles

<code>\textopappli <macro> (Sans argument)</code>	<code>\textophyp <macro> (Sans argument)</code>
<code>\textopchoice <macro> (Sans argument)</code>	<code>\textopid <macro> (Sans argument)</code>
<code>\textopcond <macro> (Sans argument)</code>	<code>\textopplot <macro> (Sans argument)</code>
<code>\textopcons <macro> (Sans argument)</code>	<code>\textoptest <macro> (Sans argument)</code>
<code>\textopdef <macro> (Sans argument)</code>	

6.2 Équivalences et implications

6.2.1 Des symboles supplémentaires

Un premier exemple – Implication réciproque

En plus des opérateurs `\iff` et `\implies` proposés par L^AT_EX, il a été ajouté l'opérateur `\liesimp`, où l'on a inversé les groupes syllabiques de `\implies`, un opérateur pour obtenir \Longleftarrow ², ainsi que des versions négatives. Voici un exemple d'utilisation.

2. Penser aussi aux preuves d'équivalence par double implication.

Un théorème de la logique : $(A \implies B) \iff (B \limesimp A)$

Un autre théorème de la logique : $(A \implies B) \not\iff (A \notimplies B)$

Un théorème de la logique : $(A \implies B) \iff (B \lleftarrow A)$

Un autre théorème de la logique : $(A \implies B) \not\iff (A \not\Rightarrow B)$

Un deuxième exemple – Des opérateurs décorés

Tout comme pour les égalités, il existe des versions décorées de type test, hypothèse, condition ... Elles sont toutes présentes dans l'exemple suivant.

Équivalences :

$\$A \backslash\text{iffappli } B \backslash\text{iffchoice } C \backslash\text{iffcond } D \backslash\text{iffcons } E \backslash\text{iffhyp } F \backslash\text{ifftest } G\$$

Équivalences - Formes négatives :

$\$A \backslash\text{notiffappli } B \backslash\text{notiffchoice } C \backslash\text{notiffcond } D$
 $\backslash\text{notiffcons } E \backslash\text{notiffhyp } F \backslash\text{notifftest } G\$$

Implications directes :

$\$A \backslash\text{impliesappli } B \backslash\text{implieschoice } C \backslash\text{impliescond } D$
 $\backslash\text{impliescons } E \backslash\text{implieshyp } F \backslash\text{impliestest } G\$$

Implications directes - Formes négatives :

$\$A \backslash\text{notimpliesappli } B \backslash\text{notimplieschoice } C \backslash\text{notimpliescond } D$
 $\backslash\text{notimpliescons } E \backslash\text{notimplieshyp } F \backslash\text{notimpliestest } G\$$

Implications indirectes :

$\$A \backslash\text{liesimpappli } B \backslash\text{liesimpchoice } C \backslash\text{liesimpcond } D$
 $\backslash\text{liesimpcons } E \backslash\text{liesimphyp } F \backslash\text{liesimptest } G\$$

Implications indirectes - Formes négatives :

$\$A \backslash\text{notliesimpappli } B \backslash\text{notliesimpchoice } C \backslash\text{notliesimpcond } D$
 $\backslash\text{notliesimpcons } E \backslash\text{notliesimphyp } F \backslash\text{notliesimptest } G\$$

Équivalences : $A \overset{\text{appli}}{\iff} B \overset{\text{choix}}{\iff} C \overset{\text{cond}}{\iff} D \overset{\text{cons}}{\iff} E \overset{\text{hyp}}{\iff} F \overset{?}{\iff} G$

Équivalences - Formes négatives : $A \overset{\text{appli}}{\not\iff} B \overset{\text{choix}}{\not\iff} C \overset{\text{cond}}{\not\iff} D \overset{\text{cons}}{\not\iff} E \overset{\text{hyp}}{\not\iff} F \overset{?}{\not\iff} G$

Implications directes : $A \overset{\text{appli}}{\implies} B \overset{\text{choix}}{\implies} C \overset{\text{cond}}{\implies} D \overset{\text{cons}}{\implies} E \overset{\text{hyp}}{\implies} F \overset{?}{\implies} G$

Implications directes - Formes négatives : $A \overset{\text{appli}}{\not\implies} B \overset{\text{choix}}{\not\implies} C \overset{\text{cond}}{\not\implies} D \overset{\text{cons}}{\not\implies} E \overset{\text{hyp}}{\not\implies} F \overset{?}{\not\implies} G$

Implications indirectes : $A \overset{\text{appli}}{\lleftarrow} B \overset{\text{choix}}{\lleftarrow} C \overset{\text{cond}}{\lleftarrow} D \overset{\text{cons}}{\lleftarrow} E \overset{\text{hyp}}{\lleftarrow} F \overset{?}{\lleftarrow} G$

Implications indirectes - Formes négatives : $A \overset{\text{appli}}{\not\lleftarrow} B \overset{\text{choix}}{\not\lleftarrow} C \overset{\text{cond}}{\not\lleftarrow} D \overset{\text{cons}}{\not\lleftarrow} E \overset{\text{hyp}}{\not\lleftarrow} F \overset{?}{\not\lleftarrow} G$

Remarque. La table 1 page 12 montre toutes les associations autorisées entre opérateurs logiques et décorations.

Fiches techniques

<code>\iff <macro> (Sans argument)</code>	<code>\notimplieschoice <macro> (Sans argument)</code>
<code>\iffappli <macro> (Sans argument)</code>	<code>\notimpliescond <macro> (Sans argument)</code>
<code>\iffchoice <macro> (Sans argument)</code>	<code>\notimpliescons <macro> (Sans argument)</code>
<code>\iffcond <macro> (Sans argument)</code>	<code>\notimplieshyp <macro> (Sans argument)</code>
<code>\iffcons <macro> (Sans argument)</code>	<code>\notimpliestest <macro> (Sans argument)</code>
<code>\iffhyp <macro> (Sans argument)</code>	<code>\liesimp <macro> (Sans argument)</code>
<code>\ifftest <macro> (Sans argument)</code>	<code>\liesimpappli <macro> (Sans argument)</code>
<code>\notiff <macro> (Sans argument)</code>	<code>\liesimpchoice <macro> (Sans argument)</code>
<code>\notiffappli <macro> (Sans argument)</code>	<code>\liesimpcond <macro> (Sans argument)</code>
<code>\notiffchoice <macro> (Sans argument)</code>	<code>\liesimpcons <macro> (Sans argument)</code>
<code>\notiffcond <macro> (Sans argument)</code>	<code>\liesimphyp <macro> (Sans argument)</code>
<code>\notiffcons <macro> (Sans argument)</code>	<code>\liesimptest <macro> (Sans argument)</code>
<code>\notiffhyp <macro> (Sans argument)</code>	<code>\notliesimp <macro> (Sans argument)</code>
<code>\notifftest <macro> (Sans argument)</code>	<code>\notliesimpappli <macro> (Sans argument)</code>
<code>\implies <macro> (Sans argument)</code>	<code>\notliesimpchoice <macro> (Sans argument)</code>
<code>\impliesappli <macro> (Sans argument)</code>	<code>\notliesimpcond <macro> (Sans argument)</code>
<code>\implieschoice <macro> (Sans argument)</code>	<code>\notliesimpcons <macro> (Sans argument)</code>
<code>\impliescond <macro> (Sans argument)</code>	<code>\notliesimphyp <macro> (Sans argument)</code>
<code>\impliescons <macro> (Sans argument)</code>	<code>\notliesimptest <macro> (Sans argument)</code>
<code>\implieshyp <macro> (Sans argument)</code>	
<code>\impliestest <macro> (Sans argument)</code>	
<code>\notimplies <macro> (Sans argument)</code>	
<code>\notimpliesappli <macro> (Sans argument)</code>	

6.2.2 Équivalences et implications verticales

À quoi cela sert-il ?

Dans la section 6.4 est expliqué comment détailler les étapes d'un raisonnement. Avec cet outil, il devient utile d'avoir des versions verticales non décorées des symboles d'équivalence et d'implication. Voici comment les obtenir.

```
\begin{tabular}{ccc}
  $A$      & & $B$      & & $C$      & \\
  $\viff$   & & $\vimplies$ & & $\vliesimp$ & \\
  $D$      & & $E$      & & $F$      & 
\end{tabular}
```

A	B	C
\Downarrow	\Downarrow	\Uparrow
D	E	F

Fiches techniques

```
\viff <macro> (Sans argument)
\vimplies <macro> (Sans argument)
\vliesimp <macro> (Sans argument)
```

6.3 Tables des décorations possibles des opérateurs

La table 1 de la présente page donne toutes les associations autorisées entre opérateurs et décorations. Il faut retenir que les versions étoilées produisent des écritures symboliques.

TABLE 1 – Décorations

	appli	choice	cond	cons	def	def*	hyp	id	id*	plot	test
<code>\eq</code>	×	×	×	×	×	×	×	×	×	×	×
<code>\neq</code>	×	×	×	×			×	×			×
<code>\l</code>	×	×	×	×			×				×
<code>\g</code>	×	×	×	×			×				×
<code>\leq</code>	×	×	×	×			×				×
<code>\geq</code>	×	×	×	×			×				×
<code>\iff</code>	×	×	×	×			×				×
<code>\notiff</code>	×	×	×	×			×				×
<code>\implies</code>	×	×	×	×			×				×
<code>\notimplies</code>	×	×	×	×			×				×
<code>\liesimp</code>	×	×	×	×			×				×
<code>\notliesimp</code>	×	×	×	×			×				×

6.4 Détailler un raisonnement

Un exemple sur une petite étape

La macro `\explain` prend deux arguments obligatoires : le premier est un symbole et le second une courte explication. Commençons par une simple étape de raisonnement détaillée comme suit.

```
$0 \leq a < b$
```

```
$\explain{\vimplies}{Par croissance de la fonction carrée sur $\RRp$.$}$
```

```
$a^2 \leq b^2$
```

$$0 \leq a < b$$

$$\Downarrow \quad \{ \text{Par croissance de la fonction carrée sur } \mathbb{R}_+. \}$$

$$a^2 \leq b^2$$

Si vous souhaitez un espace devant le symbole, il suffit d'indiquer l'espace via une distance grâce à l'argument optionnel comme ci-après.

```
$0 \leq a < b$
```

```
$\explain[1.5em]{\vimplies}{Par croissance de la fonction carrée sur $\RRp$.$}$
```

```
$a^2 \leq b^2$
```

$$0 \leq a < b$$

$$\Downarrow \quad \{ \text{Par croissance de la fonction carrée sur } \mathbb{R}_+. \}$$

$$a^2 \leq b^2$$

`\explain` utilise les macros constantes suivantes.

- `\textexplainleft` et `\textexplainright` qui donnent `{` et `}` respectivement par défaut.
- `\textexplainspacein` est l'espace entre le symbole et la courte explication. Par défaut, cette macro vaut `2em`.

Détailler des calculs

Pour finir, voici un exemple avec l'environnement `flalign` du package `amsmath`, qui est automatiquement chargé par `lymath`, où l'on constate qu'il est relativement rapide de détailler un calcul.

```
\begin{flalign*}
& (a + b)^2
&& \backslash
& \explain{=}{On utilise  $x^2 = x \cdot x$ .}
&& \backslash
& (a + b) (a + b)
&& \backslash
& \explain{=}{Double développement depuis la parenthèse gauche.}
&& \backslash
& a^2 + a b + b a + b^2
&& \backslash
& \explain{=}{Commutativité du produit.}
&& \backslash
& a^2 + 2 a b + b^2
&& \backslash
\end{flalign*}
```

$$\begin{aligned} & (a + b)^2 \\ &= \quad \{ \text{On utilise } x^2 = x \cdot x. \} \\ & (a + b)(a + b) \\ &= \quad \{ \text{Double développement depuis la parenthèse gauche.} \} \\ & a^2 + ab + ba + b^2 \\ &= \quad \{ \text{Commutativité du produit.} \} \\ & a^2 + 2ab + b^2 \end{aligned}$$

Fiches techniques

`\explain <macro> [1 Option] (2 Arguments)`

— Option: espacement avant le symbole. Valeur par défaut : `0em`.

— Argument 1: un symbole.

— Argument 2: une courte explication.

`\textexplainleft <macro> (Sans argument)`

`\textexplainright <macro> (Sans argument)`

`\textexplainspacein <macro> (Sans argument)`

6.5 Des versions alternatives du quantificateur \exists

Quantifier l'existence

Voici deux versions, l'une classique, et l'autre beaucoup moins, permettant de préciser le quantificateur \exists (concernant les applications injectives, surjectives, ... etc. se reporter à la section 7.5).

`$f: E \to F$` est une application bijective si `$\forall y \in F$`,
`$\exists! x \in E$` tel que `$y = f(x)$`.

`$f: E \to F$` est une fonction, ou application partielle, si `$\forall x \in E$`,
`$\exists \leq 1 y \in F$` tel que `$y = f(x)$` où `$\exists \leq 1$` signifie
`\emph{\og il existe au plus un \fg}`.

On pourrait poser `$\exists = 1 \eqdef \exists!$` pour ne pas multiplier
le symbolisme. Quoique\dots

$f : E \rightarrow F$ est une application bijective si $\forall y \in F, \exists! x \in E$ tel que $y = f(x)$.

$f : E \rightarrow F$ est une fonction, ou application partielle, si $\forall x \in E, \exists_{\leq 1} y \in F$ tel que $y = f(x)$ où $\exists_{\leq 1}$ signifie « il existe au plus un ».

On pourrait poser $\exists_{=1} \stackrel{\text{def}}{=} \exists!$ pour ne pas multiplier le symbolisme. Quoique...

Versions négatives

On peut utiliser `$\nexists!$` soit `\emph{\og il n'existe pas un unique \fg}`,
et `$\nexists > 4$` soit `\emph{\og il n'existe pas plus de quatre \fg}`, en plus
de `\nexists` proposé par `\verb+amssymb+`.

On peut utiliser $\nexists!$ soit « il n'existe pas un unique », et $\nexists_{>4}$ soit « il n'existe pas plus de quatre »,
en plus de \nexists proposé par `amssymb`.

Fiches techniques

`\existmulti <macro> (1 Argument)`

`\nexistmulti <macro> (1 Argument)`

— Argument 1: une écriture mathématique servant à préciser la portée du quantificateur.

`\existssone <macro> (Sans argument)`

`\nexistsone <macro> (Sans argument)`

7 Ensembles et applications

7.1 Différents types d'ensembles

7.1.1 Ensembles versus accolades

Exemple d'utilisation 1

Un ensemble de beaux nombres : $\{1 ; 3 ; 5\}$.

Un ensemble de beaux nombres : $\{1;3;5\}$.

Exemple d'utilisation 2

Choisissez votre camp :
 $\displaystyle \{ \frac{1}{3} ; \frac{5}{7} ; \frac{9}{11} \}$
ou
 $\displaystyle \{ \frac{1}{3} ; \frac{5}{7} ; \frac{9}{11} \}$.

Choisissez votre camp : $\left\{ \frac{1}{3} ; \frac{5}{7} ; \frac{9}{11} \right\}$ ou $\left\{ \frac{1}{3} ; \frac{5}{7} ; \frac{9}{11} \right\}$.

Fiches techniques

`\setgene <macro> (1 Argument)`

`\setgene* <macro> (1 Argument)`

— Argument: la définition de l'ensemble.

7.1.2 Ensembles pour la géométrie

Exemple d'utilisation 1

Vous pouvez écrire sémantiquement \mathcal{C} , \mathcal{D} et \mathcal{d} mais pas taper `\verb+\setgeo{ABC}` .

Vous pouvez écrire sémantiquement \mathcal{C} , \mathcal{D} et \mathcal{d} mais pas taper `\setgeo{ABC}` .

Exemple d'utilisation 2

Pour les indices, utilisez \mathcal{C}_1 , \mathcal{C}_2 ...

Pour les indices, utilisez \mathcal{C}_1 , \mathcal{C}_2 ...

Fiches techniques

`\setgeo <macro> (1 Argument)`

— Argument: un seul caractère ASCII indiquant un ensemble géométrique.

`\setgeo* <macro> (2 Arguments)`

- Argument 1: un seul caractère ASCII indiquant \mathcal{U} dans le nom \mathcal{U}_d d'un ensemble géométrique.
- Argument 2: un texte donnant d dans le nom \mathcal{U}_d d'un ensemble géométrique.

7.1.3 Ensembles probabilistes

Exemple d'utilisation 1

Vous pouvez écrire sémantiquement `\setproba{E}` et `\setproba{G}` mais pas taper `\verb+\setproba{ABC}+`.

Vous pouvez écrire sémantiquement \mathcal{E} et \mathcal{G} mais pas taper `\setproba{ABC}`.

Exemple d'utilisation 2

Pour les indices, utilisez `\setproba*{E}{1}`, `\setproba*{E}{2}` `\dots`

Pour les indices, utilisez $\mathcal{E}_1, \mathcal{E}_2 \dots$

Fiches techniques

`\setproba <macro> (1 Argument)`

- Argument: un seul caractère ASCII majuscule indiquant un ensemble probabiliste.

`\setproba* <macro> (2 Arguments)`

- Argument 1: un seul caractère ASCII majuscule indiquant \mathcal{U} dans le nom \mathcal{U}_d d'un ensemble probabiliste.
- Argument 2: un texte donnant d dans le nom \mathcal{U}_d d'un ensemble probabiliste.

7.1.4 Ensembles pour l'algèbre générale

Exemple d'utilisation 1

Vous pouvez écrire sémantiquement `\setalge{A}`, `\setalge{K}`, `\setalge{h}` et `\setalge{k}` mais pas taper `\verb+\setalge{ABC}+`.

Vous pouvez écrire sémantiquement A, K, h et k mais pas taper `\setalge{ABC}`.

Exemple d'utilisation 2

Pour les indices, utilisez `\setalge*{k}{1}`, `\setalge*{k}{2}` `\dots`

Pour les indices, utilisez $k_1, k_2 \dots$

Fiches techniques

`\setalge <macro> (1 Argument)`

— **Argument**: soit l'une des lettres `h` et `k`, soit un seul caractère ASCII majuscule indiquant un ensemble de type anneau ou corps.

`\setalge* <macro> (2 Arguments)`

— **Argument 1**: un seul caractère ASCII indiquant \mathbb{U} dans le nom \mathbb{U}_d d'un ensemble de type anneau ou corps.

— **Argument 2**: un texte donnant d dans le nom \mathbb{U}_d d'un ensemble de type anneau ou corps.

7.2 Ensembles classiques en mathématiques et en informatique théorique

7.2.1 La liste complète

Vous pouvez utiliser directement `\nullset`, `\NN`, `\ZZ`, `\DD`, `\QQ`, `\RR`, `\CC`, mais aussi `\PP` pour l'ensemble des nombres premiers, `\HH` pour les quaternions, `\OO` pour les octonions, et aussi `\FF` pour des nombres flottants (*notation à préciser suivant le contexte*).

Vous pouvez utiliser directement \emptyset , \mathbb{N} , \mathbb{Z} , \mathbb{D} , \mathbb{Q} , \mathbb{R} , \mathbb{C} , mais aussi \mathbb{P} pour l'ensemble des nombres premiers, \mathbb{H} pour les quaternions, \mathbb{O} pour les octonions, et aussi \mathbb{F} pour des nombres flottants (*notation à préciser suivant le contexte*).

7.2.2 Ensembles classiques suffixés

Il est facile de taper `\RRn`, `\RRp`, `\RRs`, `\RRsn` et `\RRsp`.

Il est facile de taper \mathbb{R}_- , \mathbb{R}_+ , \mathbb{R}^* , \mathbb{R}_-^* et \mathbb{R}_+^* .

Nous avons utilisé les suffixes `n` pour **N**egatif, `p` pour **P**ositif, et `s` pour **s**tar, soit "étoile" en anglais. Il y a aussi les suffixes composites `sn` et `sp`.

Notez qu'il est interdit d'utiliser `\CCn` pour \mathbb{C}_- car l'ensemble \mathbb{C} ne possède pas de structure ordonnée standard. Jetez un oeil à la section suivante pour apprendre à taper \mathbb{C}_- si vous en avez besoin. L'interdiction est ici purement sémantique!

Remarque. La table 2 page suivante montre les associations autorisées entre ensembles classiques et suffixes.

Fiches techniques

`\NN <macro> (Sans argument)`

`\NNs <macro> (Sans argument)`

`\PP <macro> (Sans argument)`

`\ZZ <macro> (Sans argument)`

`\ZZn <macro> (Sans argument)`

`\ZZp <macro> (Sans argument)`

`\ZZs <macro> (Sans argument)`

`\ZZsn <macro> (Sans argument)`

`\ZZsp <macro> (Sans argument)`

`\DD <macro> (Sans argument)`

`\DDn <macro> (Sans argument)`

`\DDp <macro> (Sans argument)`

TABLE 2 – Suffixes

	n	p	s	sn	sp
\NN			×		
\PP					
\ZZ	×	×	×	×	×
\DD	×	×	×	×	×
\QQ	×	×	×	×	×
\RR	×	×	×	×	×
\CC			×		
\HH			×		
\OO			×		
\FF	×	×	×	×	×

`\DDs <macro> (Sans argument)`
`\DDsn <macro> (Sans argument)`
`\DDsp <macro> (Sans argument)`
`\QQ <macro> (Sans argument)`
`\QQn <macro> (Sans argument)`
`\QQp <macro> (Sans argument)`
`\QQs <macro> (Sans argument)`
`\QQsn <macro> (Sans argument)`
`\QQsp <macro> (Sans argument)`
`\RR <macro> (Sans argument)`
`\RRn <macro> (Sans argument)`
`\RRp <macro> (Sans argument)`
`\RRs <macro> (Sans argument)`
`\RRsn <macro> (Sans argument)`

`\RRsp <macro> (Sans argument)`
`\CC <macro> (Sans argument)`
`\CCs <macro> (Sans argument)`
`\HH <macro> (Sans argument)`
`\HHs <macro> (Sans argument)`
`\OO <macro> (Sans argument)`
`\OOs <macro> (Sans argument)`
`\FF <macro> (Sans argument)`
`\FFn <macro> (Sans argument)`
`\FFp <macro> (Sans argument)`
`\FFs <macro> (Sans argument)`
`\FFsn <macro> (Sans argument)`
`\FFsp <macro> (Sans argument)`

7.3 Des suffixes à la carte

Exemple d'utilisation

Il est tout de même possible d'écrire \mathcal{C}_n ou \mathcal{H}_{sp} .
Il y a aussi \mathcal{P}_n avec une autre mise en forme.

Il est tout de même possible d'écrire \mathbb{C}_- ou \mathbb{H}_+^* . Il y a aussi $\mathcal{P}_{\leq 0}$ avec une autre mise en forme.

Fiches techniques

`\setspecial <macro> (2 Arguments)`

`\setspecial* <macro> (2 Arguments)`

— Argument 1: l'ensemble à "suffixer".

— Argument 2: l'un des suffixes n, p, s, sn ou sp.

7.4 Intervalles

7.4.1 Intervalles réels - Notation française (?)

Exemple d'utilisation 1

Dans cet exemple, la syntaxe fait référence à **O**-pened et **C**-losed pour "ouvert" et "fermé" en anglais. Nous verrons que **CC** et **OO** sont contractés en **C** et **O**.

Dans $I =]a ; b] = \text{\intervalOC{a}{b}}$, vous constatez que la macro utilisée résout un problème d'espacement vis à vis du signe $=$.

Dans $I =]a ; b] =]a ; b]$, vous constatez que la macro utilisée résout un problème d'espacement vis à vis du signe $=$.

Exemple d'utilisation 2

Les crochets s'étendent verticalement automatiquement. Pour empêcher cela, il suffit d'utiliser la version étoilée de la macro. Dans ce cas, les crochets restent tout de même un peu plus grands que des crochets utilisés directement. Voici un exemple.

$$\begin{aligned} \text{\displaystyle \intervalC{ \frac{1}{2} }{ 1^{2^3} } } \\ &= [\frac{1}{2} ; 1^{2^3}] \\ &= \text{\intervalC*{ \frac{1}{2} }{ 1^{2^3} } } \end{aligned}$$

$$\left[\frac{1}{2} ; 1^{2^3} \right] = \left[\frac{1}{2} ; 1^{2^3} \right] = \left[\frac{1}{2} ; 1^{2^3} \right]$$

Fiches techniques

Pour toutes les macros ci-dessous, la version non étoilée produit des délimiteurs qui s'étirent si besoin verticalement, tandis que la version étoilée ne le fait pas.

`\intervalCO <macro> (2 Arguments)`

`\intervalCO* <macro> (2 Arguments)`

— Argument 1: borne inférieure a de l'intervalle $[a ; b[$.

— Argument 2: borne supérieure b de l'intervalle $[a ; b[$.

`\intervalC <macro> (2 Arguments)`

`\intervalC* <macro> (2 Arguments)`

— Argument 1: borne inférieure a de l'intervalle $[a ; b]$.

— Argument 2: borne supérieure b de l'intervalle $[a ; b]$.

`\intervalO <macro> (2 Arguments)`

`\intervalO* <macro> (2 Arguments)`

— Argument 1: borne inférieure a de l'intervalle $]a ; b[$.

— Argument 2: borne supérieure b de l'intervalle $]a ; b[$.

`\intervalOC <macro> (2 Arguments)`

`\intervalOC* <macro> (2 Arguments)`

- Argument 1: borne inférieure a de l'intervalle $]a; b]$.
- Argument 2: borne supérieure b de l'intervalle $]a; b]$.

7.4.2 Intervalles réels - Notation américaine

Exemple d'utilisation

Dans cet exemple, la syntaxe fait référence à **P**-arenthèse.

Aux États Unis, un intervalle semi-fermé s'écrit $\text{\textbackslash intervalPC\{a\}\{b\}} = (a ; b]$ et un intervalle ouvert se tape $\text{\textbackslash intervalP\{a\}\{b\}} = (a ; b)$.

Aux États Unis, un intervalle semi-fermé s'écrit $(a; b] = (a; b]$ et un intervalle ouvert se tape $(a; b) = (a; b)$.

Fiches techniques

Pour toutes les macros ci-dessous, la version non étoilée produit des délimiteurs qui s'étirent si besoin verticalement, tandis que la version étoilée ne le fait pas.

$\text{\textbackslash intervalCP}$ <macro> (2 Arguments)

$\text{\textbackslash intervalCP*}$ <macro> (2 Arguments)

— Argument 1: borne inférieure a de l'intervalle $[a; b)$.

— Argument 2: borne supérieure b de l'intervalle $[a; b)$.

$\text{\textbackslash intervalP}$ <macro> (2 Arguments)

$\text{\textbackslash intervalP*}$ <macro> (2 Arguments)

— Argument 1: borne inférieure a de l'intervalle $(a; b)$.

— Argument 2: borne supérieure b de l'intervalle $(a; b)$.

$\text{\textbackslash intervalPC}$ <macro> (2 Arguments)

$\text{\textbackslash intervalPC*}$ <macro> (2 Arguments)

— Argument 1: borne inférieure a de l'intervalle $(a; b]$.

— Argument 2: borne supérieure b de l'intervalle $(a; b]$.

7.4.3 Intervalles discrets d'entiers

Exemple d'utilisation

Dans l'exemple, la syntaxe fait référence à \mathbb{Z} l'ensemble des entiers relatifs.

Par définition, $\text{\textbackslash ZintervalC\{-1\}\{4\}} = \{-1 ; 0 ; 1 ; 2 ; 3 ; 4 \}$. Donc nous avons $\text{\textbackslash ZintervalC\{-1\}\{4\}} = \text{\textbackslash ZintervalO\{-2\}\{5\}}$.

Par définition, $\llbracket -1 ; 4 \rrbracket = \{-1 ; 0 ; 1 ; 2 ; 3 ; 4\}$. Donc nous avons $\llbracket -1 ; 4 \rrbracket = \llbracket -2 ; 5 \rrbracket$.

Fiches techniques

Pour toutes les macros ci-dessous, la version non étoilée produit des délimiteurs qui s'étirent si besoin verticalement, tandis que la version étoilée ne le fait pas.

`\ZintervalC0 <macro> (2 Arguments)`

`\ZintervalC0* <macro> (2 Arguments)`

— Argument 1: borne inférieure a de l'intervalle $\llbracket a; b \rrbracket$.

— Argument 2: borne supérieure b de l'intervalle $\llbracket a; b \rrbracket$.

`\ZintervalC <macro> (2 Arguments)`

`\ZintervalC* <macro> (2 Arguments)`

— Argument 1: borne inférieure a de l'intervalle $\llbracket a; b \rrbracket$.

— Argument 2: borne supérieure b de l'intervalle $\llbracket a; b \rrbracket$.

`\Zinterval0 <macro> (2 Arguments)`

`\Zinterval0* <macro> (2 Arguments)`

— Argument 1: borne inférieure a de l'intervalle $\llbracket a; b \rrbracket$.

— Argument 2: borne supérieure b de l'intervalle $\llbracket a; b \rrbracket$.

`\Zinterval0C <macro> (2 Arguments)`

`\Zinterval0C* <macro> (2 Arguments)`

— Argument 1: borne inférieure a de l'intervalle $\llbracket a; b \rrbracket$.

— Argument 2: borne supérieure b de l'intervalle $\llbracket a; b \rrbracket$.

7.5 Application totale, partielle, injective, surjective et/ou bijective

Voici des symboles qui, bien que très techniques, facilitent la rédaction de documents à propos des applications totales ou partielles³ (*on parle aussi d'applications, sans qualificatif, et de fonctions*).

Applications totales – Un exemple complet

`$f: A \to B$` est une application totale, c'est à dire définie sur `A` tout entier.

`$i: C \toone D$` est une application totale injective.

`$s: E \onto F$` est une application totale surjective.

`$b: G \bijet H$` est une application totale bijective.

 $f : A \rightarrow B$ est une application totale, c'est à dire définie sur A tout entier.

$i : C \rightarrowtail D$ est une application totale injective.

$s : E \twoheadrightarrow F$ est une application totale surjective.

$b : G \rightarrowtailtail H$ est une application totale bijective.

3. $a : E \rightarrow F$ est une application si $\forall x \in E, \exists ! y \in F$ tel que $y = a(x)$. Plus généralement, $f : E \rightarrow F$ est une fonction si $\forall x \in E, \exists_{\leq 1} y \in F$ tel que $y = f(x)$, autrement dit soit $f(x)$ existe dans F , soit f n'est pas définie en x .

Applications partielles – Un exemple complet

`$f: A \pto B$` est une application partielle, c'est à dire définie sur un sous-ensemble de A .

`$i: C \ponetoone D$` est une application partielle injective.

`$s: E \ponto F$` est une application partielle surjective.

`$b: G \pbijet H$` est une application partielle bijective.

$f : A \rightarrowtail B$ est une application partielle, c'est à dire définie sur un sous-ensemble de A .

$i : C \rightarrowtail D$ est une application partielle injective.

$s : E \twoheadrightarrow F$ est une application partielle surjective.

$b : G \twoheadrightarrow H$ est une application partielle bijective.

Fiches techniques

`\to` <macro> (Sans argument)

`\onetoone` <macro> (Sans argument)

`\onto` <macro> (Sans argument)

`\bijet` <macro> (Sans argument)

`\pto` <macro> (Sans argument)

`\ponetoone` <macro> (Sans argument)

`\ponto` <macro> (Sans argument)

`\pbijet` <macro> (Sans argument)

8 Géométrie

8.1 Points et lignes

8.1.1 Points

Exemple d'utilisation 1

`\pt{I}` indique un point nommé "I".

I indique un point nommé "I".

Exemple d'utilisation 2

Une liste de points : `$\pt*{I}{1}$, $\pt*{I}{2}$ \dots`

Une liste de points : $I_1, I_2 \dots$

Exemple d'utilisation 3

Une droite `(\pts{AB})` au lieu de `$(\pt{A}\pt{B})$`.

Une droite (AB) au lieu de (AB).

Fiches techniques

`\pt <macro> (1 Argument)`

— Argument: un texte donnant le nom d'un point.

`\pt* <macro> (2 Arguments)`

— Argument 1: un texte indiquant UP dans le nom UP_{down} d'un point.

— Argument 2: un texte indiquant *down* dans le nom UP_{down} d'un point.

`\pts <macro> (1 Argument)`

— Argument: un texte indiquant des noms de points non indicés.

8.1.2 Droites parallèles ou non

Les opérateurs `\parallel` et `\nparallel` utilisent des obliques au lieu de barres verticales comme le montre l'exemple qui suit où `\stdnparallel` est un alias de `\nparallel` fourni par le package `amssymb`, et `\stdparallel` est un alias de la version standard de `\parallel` proposée par L^AT_EX.

`$(\pts{AB}) \parallel (\pts{CD})$` et `$(\pts{EF}) \nparallel (\pts{GH})$` au lieu de
`$(\pts{AB}) \stdparallel (\pts{CD})$` et de `$(\pts{EF}) \stdnparallel (\pts{GH})$`.

(AB) // (CD) et (EF) \nparallel (GH) au lieu de (AB) || (CD) et de (EF) \nparallel (GH).

Fiches techniques

`\parallel <macro> (Sans argument)`

`\nparallel <macro> (Sans argument)`

`\stdparallel <macro> (Sans argument)`

`\stdnparallel <macro> (Sans argument)`

8.2 Vecteurs

8.2.1 Les écrire

Exemple d'utilisation 1

Voici un vecteur `$(\vect{ABCDEFG})$` avec beaucoup de lettres et vous pouvez écrire
`$(\vect*{e}{rot})$` au lieu de `$(\vect{e_{rot}})$`.

Voici un vecteur $\overrightarrow{ABCDEFG}$ avec beaucoup de lettres et vous pouvez écrire \vec{e}_{rot} au lieu de $\overrightarrow{e_{rot}}$.

Exemple d'utilisation 2

Vous pouvez écrire `(\vect{i})` and `$(\vect*{j}{2})$` sans point.

Vous pouvez écrire \vec{i} and \vec{j}_2 sans point.

Fiches techniques

`\vect <macro> (1 Argument)`

— Argument: un texte donnant le nom d'un vecteur.

`\vect* <macro> (2 Arguments)`

— Argument 1: un texte indiquant *up* dans le nom $\overrightarrow{up}_{down}$ d'un vecteur.

— Argument 2: un texte indiquant *down* dans le nom $\overrightarrow{up}_{down}$ d'un vecteur.

8.2.2 Norme

Exemple d'utilisation

Nous pouvons écrire `\norm{\vect{i}}`, `\displaystyle \norm{\frac{2}{7} \vect*{e}{k}}`, ou `\displaystyle \norm*{\frac{2}{7} \vect*{e}{k}}` avec de petites barres verticales.

Nous pouvons écrire $\|\vec{i}\|$, $\left\|\frac{2}{7}\vec{e}_k\right\|$, ou $\|\frac{2}{7}\vec{e}_k\|$ avec de petites barres verticales.

Remarque. Le code L^AT_EX vient directement de ce message : <https://tex.stackexchange.com/a/43009/6880>.

Fiches techniques

`\norm <macro> (1 Argument)`

`\norm* <macro> (1 Argument)`

— Argument: le vecteur sur lequel appliquer la norme.

8.2.3 Produit scalaire – Écriture minimaliste

Exemple d'utilisation - Version longue

En mathématique, il est usage d'écrire un produit scalaire avec un point via `\dotprod{\dfrac{1}{2} \vect{i}}{\vect{j}}` .

En mathématique, il est usage d'écrire un produit scalaire avec un point via $\frac{1}{2}\vec{i} \cdot \vec{j}$.

Exemple d'utilisation - Version courte mais restrictive

Dans l'exemple suivant, le préfixe *v* est pour **v**-ector.

On peut aussi parfois juste taper `\vdotprod{i}{j}` .

On peut aussi parfois juste taper $\vec{i} \cdot \vec{j}$.

Fiches techniques

`\dotprod <macro> (2 Arguments)`

— Argument 1: le premier vecteur qu'il faut taper via la macro `\vect`.

— Argument 2: le second vecteur qu'il faut taper via la macro `\vect`.

`\vdotprod <macro> (2 Arguments)` où `v = v-ector`

— Argument 1: le nom du premier vecteur sans utiliser la macro `\vect`.

— Argument 2: le nom du second vecteur sans utiliser la macro `\vect`.

8.2.4 Produit scalaire – Écriture « physicienne »

Dans l'exemple suivant, le préfixe `a` est pour `a-ngle`, et `v` pour `v-ector`.

Les physiciens pourront utiliser
`\displaystyle \dotprod{\frac{1}{2} \vect{i}}{\vect{j}}` ,
`\displaystyle \dotprod*{\frac{1}{2} \vect{i}}{\vect{j}}` ,
`\displaystyle \vdotprod{i}{j}`
ou
`\displaystyle \vdotprod*{i}{j}` .

Les physiciens pourront utiliser $\left\langle \frac{1}{2} \vec{i} \mid \vec{j} \right\rangle$, $\left\langle \frac{1}{2} \vec{i} \mid \vec{j} \right\rangle$, $\left\langle \vec{i} \mid \vec{j} \right\rangle$ ou $\left\langle \vec{i} \mid \vec{j} \right\rangle$.

Fiches techniques

`\adotprod <macro> (2 Arguments)` où `a = a-ngle`

`\adotprod* <macro> (2 Arguments)`

— Argument 1: le premier vecteur qu'il faut taper via la macro `\vect`.

— Argument 2: le second vecteur qu'il faut taper via la macro `\vect`.

`\vadotprod <macro> (2 Arguments)` où `a = a-ngle` et `v = v-ector`

— Argument 1: le nom du premier vecteur sans utiliser la macro `\vect`.

— Argument 2: le nom du second vecteur sans utiliser la macro `\vect`.

8.2.5 Produit vectoriel

Exemple d'utilisation - Version longue

Un produit vectoriel peut s'écrire via `\crossprod{\dfrac{1}{2} \vect{i}}{\vect{j}}` .

Un produit vectoriel peut s'écrire via $\frac{1}{2} \vec{i} \wedge \vec{j}$.

Exemple d'utilisation - Version courte mais restrictive

Dans certain cas, un produit vectoriel s'écrit vite via `\vcrossprod{i}{j}` .

Dans certain cas, un produit vectoriel s'écrit vite via $\vec{i} \wedge \vec{j}$.

Fiche technique

`\crossprod` <macro> (2 Arguments)

— Argument 1: le premier vecteur qu'il faut taper via la macro `\vect`.

— Argument 2: le second vecteur qu'il faut taper via la macro `\vect`.

`\vcrossprod` <macro> (2 Arguments) où `v = v-ector`

— Argument 1: le nom du premier vecteur sans utiliser la macro `\vect`.

— Argument 2: le nom du second vecteur sans utiliser la macro `\vect`.

8.3 Coordonnées

Exemple d'utilisation 1

On peut choisir d'écrire

`\displaystyle \pt{I} \coord{\frac{1}{3} | -4 | 0}`

ou bien

`\displaystyle \pt{I} \coord*{\frac{1}{3} | -4 | 0}` .

On peut choisir d'écrire $I\left(\frac{1}{3}; -4; 0\right)$ ou bien $I(\frac{1}{3}; -4; 0)$.

Exemple d'utilisation 2

Dans l'exemple suivant, le préfixe `v` est pour **v**-ertical.

Pour les vecteurs, on peut préférer `\vect{i} \vcoord{3 | -4 | 0}`, voire `\vect{i} \vcoord*{3 | -4 | 0}`, à la place de `\vect{i} \coord{3 | -4 | 0}` afin de bien différencier les coordonnées de points de celles de vecteurs.

Pour les vecteurs, on peut préférer $\vec{i}\begin{pmatrix} 3 \\ -4 \\ 0 \end{pmatrix}$, voire $\vec{i}\begin{bmatrix} 3 \\ -4 \\ 0 \end{bmatrix}$, à la place de $\vec{i}(3; -4; 0)$ afin de bien différencier les coordonnées de points de celles de vecteurs.

Fiches techniques

`\coord` <macro> (1 Argument)

`\coord*` <macro> (1 Argument)

— Argument: l'argument est une suite de "morceaux" séparés par des barres `|`, chaque morceau étant une coordonnée. Il peut n'y avoir qu'un seul morceau.

`\vcoord <macro> (1 Argument)` où `v = v-ertical`

`\vcoord* <macro> (1 Argument)` pour des crochets à la place de parenthèses

— **Argument**: l'argument est une suite de "morceaux" séparés par des barres | , chaque morceau étant une coordonnée. Il peut n'y avoir qu'un seul morceau.

8.4 Nommer un repère

Exemple d'utilisation 1 – La méthode basique

Commençons par la manière la plus basique d'écrire un repère (*nous verrons d'autres méthodes qui peuvent être plus efficaces*).

Dans le plan, trois points $\text{\pt{0}}$, $\text{\pt{I}}$ and $\text{\pt{J}}$ non alignés définissent un repère cartésien $\text{\axes{\pt{0}} | \pt{I} | \pt{J}}$.

Dans le plan, trois points O, I and J non alignés définissent un repère cartésien (O ; I, J).

Exemple d'utilisation 2 – La méthode basique en version étoilée

Dans l'exemple ci-dessous, on voit que la version étoilée produit des petites parenthèses.

$\text{\displaystyle \axes{\pt{0}} | \frac{7}{3} \text{\vect{i}} | \text{\vect{j}}}$

ou

$\text{\displaystyle \axes*{\pt{0}} | \frac{7}{3} \text{\vect{i}} | \text{\vect{j}}}$

$\left(O; \frac{7}{3} \vec{i}, \vec{j}\right)$ ou $(O; \frac{7}{3} \vec{i}, \vec{j})$

Exemple d'utilisation 3 – La méthode basique en dimension quelconque

Il faut au minimum deux "morceaux" séparés par des barres |, cas de la dimension 1, mais il n'y a pas de maximum, cas d'une dimension quelconque $n > 0$.

$\text{\axes{\pt{0}} | \text{\vect*{i}{1}} | \text{\vect*{i}{2}} | \text{\vect*{i}{3}} | \dots | \text{\vect*{i}{9}} | \text{\vect*{i}{10}} | \text{\vect*{i}{11}} | \text{\vect*{i}{12}}}$

$(O; \vec{i}_1, \vec{i}_2, \vec{i}_3, \dots, \vec{i}_9, \vec{i}_{10}, \vec{i}_{11}, \vec{i}_{12})$

Exemple d'utilisation 4 – Repère affine

Dans l'exemple suivant, le préfixe **p** est pour **p**-oint.

$\text{\paxes{0} | I | J | K}$ évite de taper $\text{\axes{\pt{0}} | \pt{I} | \pt{J} | \pt{K}}$.

(O ; I, J, K) évite de taper (O ; I, J, K).

Exemple d'utilisation 5 – Repère vectoriel (méthode 1)

Dans l'exemple suivant, le préfixe **v** est pour **v**-ecteur.

`$\vaxes{\pt{0} | i | j}$` est un raccourci de `$\axes{\pt{0} | \vect{i} | \vect{j}}$`.

$(O; \vec{i}, \vec{j})$ est un raccourci de $(O; \vec{i}, \vec{j})$.

Exemple d'utilisation 6 – Repère vectoriel (méthode 2)

Dans l'exemple suivant, le préfixe `pv` permet de combiner à la fois les fonctionnalités proposés par les préfixes `p` et `v`.

`$\pvaxes{0 | i | j}$` donne rapidement `$\axes{\pt{0} | \vect{i} | \vect{j}}$`.

$(O; \vec{i}, \vec{j})$ donne rapidement $(O; \vec{i}, \vec{j})$.

Fiches techniques

`\axes <macro> (1 Argument)`

`\axes* <macro> (1 Argument)`

— **Argument** : l'argument est une suite de "morceaux" séparés par des barres `|`.

- Le premier morceau est l'origine du repère.
- Les morceaux suivants sont des points ou des vecteurs qui "définissent" chaque axe.

`\paxes <macro> (1 Argument)` où `p` = p-oint

— **Argument** : l'argument est une suite de "morceaux" séparés par des barres `|`.

- Le premier morceau est le nom de l'origine du repère sur laquelle la macro-commande `\pt` sera automatiquement appliquée.
- Viennent ensuite les noms des points "définissant" chaque axe. Pour chacun de ces points la macro-commande `\pt` sera automatiquement appliquée.

`\vaxes <macro> (1 Argument)` où `v` = v-ector

— **Argument** : l'argument est une suite de "morceaux" séparés par des barres `|`.

- Le premier morceau est l'origine du repère.
- Viennent ensuite les noms des vecteurs "définissant" chaque axe. Pour chacun de ces vecteurs la macro-commande `\vect` sera automatiquement appliquée.

`\pvaxes <macro> (3 Arguments)` où `pv` = p + v

— **Argument** : l'argument est une suite de "morceaux" séparés par des barres `|`.

- Le premier morceau est le nom de l'origine du repère sur laquelle la macro-commande `\pt` sera automatiquement appliquée.
- Viennent ensuite les noms des vecteurs "définissant" chaque axe. Pour chacun de ces vecteurs la macro-commande `\vect` sera automatiquement appliquée.

8.5 Arcs circulaires

Exemple d'utilisation 1

Voici un arc $\text{\arc{ABCDEF}}$ utilisant beaucoup de lettres, et vous pouvez écrire $\text{\arc*{A}{rot}}$ à la place de $\text{\arc{A_{rot}}}$.

Voici un arc \widehat{ABCDEF} utilisant beaucoup de lettres, et vous pouvez écrire $\widehat{A_{rot}}$ à la place de $\widehat{A_{rot}}$.

Exemple d'utilisation 2

$\text{\arc{i}}$ et $\text{\arc*{j}{2}}$ n'affiche pas de point sous l'arc.

\widehat{i} et $\widehat{j_2}$ n'affiche pas de point sous l'arc.

Fiches techniques

\arc <macro> (1 Argument)

— Argument: un texte donnant le nom d'un arc circulaire.

\arc* <macro> (2 Arguments)

— Argument 1: un texte indiquant *up* dans le nom $\widehat{up_{down}}$ d'un arc circulaire.

— Argument 2: un texte indiquant *down* dans le nom $\widehat{up_{down}}$ d'un arc circulaire.

8.6 Angles

8.6.1 Angles géométriques intérieurs

Exemple d'utilisation 1

Voici un angle géométrique intérieur $\text{\anglein{ABCDEF}}$ avec un long nom, et vous pouvez écrire $\text{\anglein*{A}{rot}}$ au lieu de $\text{\anglein{A_{rot}}}$.

Voici un angle géométrique intérieur \widehat{ABCDEF} avec un long nom, et vous pouvez écrire $\widehat{A_{rot}}$ au lieu de $\widehat{A_{rot}}$.

Exemple d'utilisation 2

Vous pouvez aussi écrire $\text{\anglein{i}}$ and $\text{\anglein*{j}{2}}$ sans point.

Vous pouvez aussi écrire \widehat{i} and $\widehat{j_2}$ sans point.

Fiches techniques

\anglein <macro> (1 Argument)

— Argument: un texte donnant le nom d'un angle intérieur.

`\anglein*` <macro> (2 Arguments)

— Argument 1: un texte indiquant *up* dans le nom \widehat{up}_{down} d'un angle intérieur.

— Argument 2: un texte indiquant *down* dans le nom \widehat{up}_{down} d'un angle intérieur.

8.6.2 Angles orientés de vecteurs

Sans chapeau - Version longue

En mathématique, il est d'usage de noter les angles orientés via
`\displaystyle \angleorient{\frac{1}{2} \vect{i}}{\vect{j}}`
ou
`\displaystyle \angleorient*{\frac{1}{2} \vect{i}}{\vect{j}}` .

En mathématique, il est d'usage de noter les angles orientés via $\left(\frac{1}{2}\vec{i}; \vec{j}\right)$ ou $\left(\frac{1}{2}\vec{i}; \vec{j}\right)$.

Sans chapeau - Version courte mais restrictive

Dans l'exemple suivant, le préfixe **v** est pour **v**-ector.

Avec des noms de vecteurs utilisant juste des lettres, on peut juste taper
`\displaystyle \vangleorient{i}{j}`
ou
`\displaystyle \vangleorient*{i}{j}` (la seconde écriture n'apporte rien de nouveau).

Avec des noms de vecteurs utilisant juste des lettres, on peut juste taper $(\vec{i}; \vec{j})$ ou $(\vec{i}; \vec{j})$ (la seconde écriture n'apporte rien de nouveau).

Avec un chapeau

Dans l'exemple suivant, le préfixe **h** est pour **h**-at, et **v** pour **v**-ector.

Si vous préférez les angles orientés avec un chapeau, tapez les alors via
`\displaystyle \hangleorient{\frac{1}{2} \vect{i}}{\vect{j}}` ,
`\displaystyle \hangleorient*{\frac{1}{2} \vect{i}}{\vect{j}}` ,
`\displaystyle \hvangleorient{i}{j}`
ou
`\displaystyle \hvangleorient*{i}{j}` (la dernière écriture n'apportant rien de neuf).

Si vous préférez les angles orientés avec un chapeau, tapez les alors via $\widehat{\left(\frac{1}{2}\vec{i}; \vec{j}\right)}$, $\widehat{\left(\frac{1}{2}\vec{i}; \vec{j}\right)}$,
 $\widehat{(\vec{i}; \vec{j})}$ ou $\widehat{(\vec{i}; \vec{j})}$ (la dernière écriture n'apportant rien de neuf).

Fiche technique

`\angleorient` <macro> (2 Arguments)

`\hangleorient` <macro> (2 Arguments) où **h** = **h**-at

— Argument 1: le premier vecteur qu'il faut taper via la macro `\vect`.

— Argument 2: le second vecteur qu'il faut taper via la macro `\vect`.

`\vangleorient <macro> (2 Arguments)` où $v = v\text{-ector}$
`\hvangleorient <macro> (2 Arguments)` où $h = h\text{-at}$ et $v = v\text{-ector}$

— Argument 1: le nom du premier vecteur sans utiliser la macro `\vect`.

— Argument 2: le nom du second vecteur sans utiliser la macro `\vect`.

9 Analyse

9.1 Constantes

9.1.1 Constantes classiques

La liste complète

Voici la liste des constantes classiques où $\tau = 2\pi$ est la benjamine :
`\ggamma`, `\ppi`, `\tttau`, `\ee`, `\ii`, `\jj` and `\kk`.

Voici la liste des constantes classiques où $\tau = 2\pi$ est la benjamine : γ , π , τ , e , i , j and k .

Remarque. Faites attention car `\Large $\ppi \neq \pi$` produit $\pi \neq \pi$. Comme vous le constatez, les symboles ne sont pas identiques. Ceci est vraie pour toutes les constantes grecques.

Fiches techniques

<code>\ggamma <macro> (Sans argument)</code>	<code>\ii <macro> (Sans argument)</code>
<code>\ppi <macro> (Sans argument)</code>	<code>\jj <macro> (Sans argument)</code>
<code>\tttau <macro> (Sans argument)</code>	<code>\kk <macro> (Sans argument)</code>
<code>\ee <macro> (Sans argument)</code>	

9.1.2 Constantes latines personnelles

Exemple d'utilisation

Il est aisé d'écrire `\ct{a} x^2 + \ct{b} x + \ct{c}` au lieu de `$a x^2 + b x + c$` afin d'indiquer que `\ct{a}`, `\ct{b}` et `\ct{c}` sont des constantes.

Il est aisé d'écrire $\mathbf{a}x^2 + \mathbf{b}x + \mathbf{c}$ au lieu de $ax^2 + bx + c$ afin d'indiquer que \mathbf{a} , \mathbf{b} et \mathbf{c} sont des constantes.

Fiche technique

`\ct <macro> (1 Argument)`

— Argument: un texte utilisant l'alphabet latin.

9.2 La fonction valeur absolue

Un exemple d'utilisation

Il est facile d'écrire $\abs{2}$ ou $\displaystyle \abs{\frac{3}{5}}$ voire aussi $\abs*{\frac{3}{5}}$ si vous préférez des petits traits verticaux.

Il est facile d'écrire $|2|$ ou $\left|\frac{3}{5}\right|$ voire aussi $|\frac{3}{5}|$ si vous préférez des petits traits verticaux.

Remarque. Le code L^AT_EX vient directement de ce poste : <https://tex.stackexchange.com/a/43009/6880>.

Fiches techniques

`\abs <macro>` (1 Argument)

`\abs* <macro>` (1 Argument)

— Argument : l'expression à laquelle on applique la fonction valeur absolue.

9.3 Fonctions nommées spéciales

9.3.1 Sans paramètre

Un exemple d'utilisation

Quelques fonctions nommées supplémentaires (voir la liste complète ci-dessous) : $\ch x \neq ch\ x$, $\ppcm(x;y)$, $\lg x$ le logarithme binaire.

Quelques fonctions nommées supplémentaires (voir la liste complète ci-dessous) : $ch\ x \neq chx$, $\ppcm(x;y)$, $\lg x$ le logarithme binaire.

Fiches techniques

`\ch <macro>` (Sans argument)

`\sh <macro>` (Sans argument)

`\th <macro>` (Sans argument)

`\ach <macro>` (Sans argument)

`\ash <macro>` (Sans argument)

`\ath <macro>` (Sans argument)

`\arccosh <macro>` (Sans argument)

`\arcsinh <macro>` (Sans argument)

`\arctanh <macro>` (Sans argument)

`\acos <macro>` (Sans argument)

`\asin <macro>` (Sans argument)

`\atan <macro>` (Sans argument)

`\pgcd <macro>` (Sans argument)

`\ppcm <macro>` (Sans argument)

9.3.2 Avec un paramètre

Un exemple d'utilisation

D'autres fonctions nommées supplémentaires avec un paramètre (voir la liste complète ci-dessous) : $\log_2 x = \lg x$ et $\exp_6 y = 6^y$.

D'autres fonctions nommées supplémentaires avec un paramètre (voir la liste complète ci-dessous) : $\log_2 x = \lg x$ et $\exp_6 y = 6^y$.

Fiches techniques

`\expb <macro>` (1 Argument)

— Argument : la base de l'exponentielle

`\logb <macro>` (1 Argument)

— Argument : la base du logarithme

9.4 Des notations complémentaires pour des suites spéciales

Exemple d'utilisation

Parfois nous avons besoin d'écrire $\text{seqplus}\{F\}_{1\{2\}}$ ou $\text{seqhypergeo}\{F\}_{1\{2\}}$ et le fou (?,) aime vraiment $\text{seqsuprgeo}\{F\}_{1\{2\}\{3\}\{4\}}$.

Parfois nous avons besoin d'écrire F_1^2 ou ${}_1F_2$ et le fou (?) aime vraiment ${}_1F_2^3$.

Fiches techniques

`\seqplus <macro>` (2 Arguments)

— Argument 1 : l'exposant à droite.

— Argument 2 : l'indice à droite.

`\seqhypergeo <macro>` (2 Arguments)

— Argument 1 : l'indice à gauche.

— Argument 2 : l'indice à droite.

`\seqsuprgeo <macro>` (4 Arguments)

— Argument 1 : l'indice à gauche.

— Argument 2 : l'indice à droite.

— Argument 3 : l'exposant à droite.

— Argument 4 : l'exposant à gauche.

9.5 Calcul différentiel

9.5.1 Les opérateurs ∂ et d

Exemple d'utilisation

Vous pouvez écrire `\dd{f}` et `\pp{t}` et aussi `\dd[5]{x}` ou `\pp[n]{x}`.

Vous pouvez écrire df et ∂t et aussi d^5x ou $\partial^n x$.

Fiches techniques

`\dd <macro> [1 Option] (1 Argument)`

`\pp <macro> [1 Option] (1 Argument)`

— Option: utilisée, cette option sera mise en exposant du symbole ∂ ou d .

— Argument: la variable de différentiation à droite du symbole ∂ ou d .

9.5.2 Dérivation totale

Exemple d'utilisation 1

`\displaystyle \derpow*{f} (a) = \derpow{f} (a)`
`= \derfrac{f}{x} (a)`
`= \dersub{f}{x} (a)`

$$f'(a) = f^{(1)}(a) = \frac{df}{dx}(a) = d_x f(a)$$

Exemple d'utilisation 2

`\displaystyle \derpow*[3]{f}(a) = \derpow[3]{f} (a)`
`= \derfrac[3]{f}{x} (a)`
`= \dersub[3]{f}{x} (a)`
et `\displaystyle \derpow*[10]{\cos} a = \derfrac[10]{\cos}{x} (a)` avec beaucoup trop de primes à gauche (mais le compte y est).

$$f'''(a) = f^{(3)}(a) = \frac{d^3 f}{dx^3}(a) = d_x^3 f(a) \text{ et } \cos^{''''''''} a = \frac{d^{10} \cos}{dx^{10}}(a) \text{ avec beaucoup trop de primes à gauche (mais le compte y est).}$$

Exemple d'utilisation 3

Si `\displaystyle f(x) = \frac{1}{x^2+3}` alors nous avons :
`\displaystyle \derpow[3]{f} (a)`
`= \derfrac*[3]{\left(\frac{1}{x^2+3} \right)}{x} (a)`.

$$\text{Si } f(x) = \frac{1}{x^2+3} \text{ alors nous avons : } f^{(3)}(a) = \frac{d^3}{dx^3} \left(\frac{1}{x^2+3} \right) (a).$$

Fiches techniques

`\derpow <macro> [1 Option] (1 Argument)`
`\derpow* <macro> [1 Option] (1 Argument)`

— **Option**: utilisée, cette option sera l'exposant de dérivation mis entre des parenthèses pour la version non étoilée, et le nombre de primes pour la version étoilée.

— **Argument**: la fonction à différencier.

`\derfrac <macro> [1 Option] (2 Arguments)`
`\derfrac* <macro> [1 Option] (2 Arguments)`
`\dersub <macro> [1 Option] (2 Arguments)`

— **Option**: utilisée, cette option sera l'exposant de dérivation.

— **Argument 1**: la fonction à dériver.

— **Argument 2**: la variable.

9.5.3 Dérivation partielle

Exemple d'utilisation 1

```
\displaystyle \partialfrac{f}{x} (a;b)  
= \partialsub{f}{x} (a;b)  
= \partialprime{f}{x} (a;b)
```

$$\frac{\partial f}{\partial x}(a;b) = \partial_x f(a;b) = f'_x(a;b)$$

Exemple d'utilisation 2

```
\displaystyle \partialfrac[3]{G}{f^2 | v} (a;b)  
= \partialfrac{G}{f^2 | v} (a;b)  
= \partialsub{G}{f^2 | v} (a;b)  
= \partialprime{G}{f^2 | v} (a;b)
```

$$\frac{\partial^3 G}{\partial f^2 \partial v}(a;b) = \frac{\partial G}{\partial f^2 \partial v}(a;b) = \partial_{f(2)v} G(a;b) = G'_{f(2)v}(a;b)$$

Exemple d'utilisation 3

```
Si \displaystyle f(x;y) = \frac{\cos(x y)}{x^2+y^2} alors nous avons  
\displaystyle \partialfrac[2]{f}{x | y}  
= \partialfrac*[2]{\left( \frac{\cos(x y)}{x^2 + y^2} \right)}{x | y}.
```

$$\text{Si } f(x;y) = \frac{\cos(xy)}{x^2 + y^2} \text{ alors nous avons } \frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2}{\partial x \partial y} \left(\frac{\cos(xy)}{x^2 + y^2} \right).$$

Fiches techniques

`\partialfrac <macro> [1 Option] (2 Arguments)`

`\partialfrac* <macro> [1 Option] (2 Arguments)`

— **Option**: utilisée, cette option sera l'exposant total de dérivation mis en exposant de ∂ .

— **Argument 1**: la fonction à dériver partiellement.

— **Argument 2**: les variables utilisées pour la dérivation partielle en utilisant la syntaxe suivante :
par exemple, `x | y^3 | ...` indique de dériver suivant x une fois, puis suivant y trois fois... etc.

`\partialsub <macro> (2 Arguments)`

`\partialprime <macro> (2 Arguments)`

— **Argument 1**: la fonction à dériver partiellement.

— **Argument 2**: les variables utilisées pour la dérivation partielle en utilisant la syntaxe suivante :
par exemple, `x | y^3 | ...` indique de dériver suivant x une fois, puis suivant y trois fois... etc.

9.6 Calcul intégral

9.6.1 L'opérateur crochet – 1^{ère} version

Exemple d'utilisation 1

Par définition, $\displaystyle \int_a^b f(x) \, dx = [F(x)]_a^b$ où $[F(x)]_a^b = F(b) - F(a)$.

Par définition, $\int_a^b f(x) \, dx = [F(x)]_a^b$ où $[F(x)]_a^b = F(b) - F(a)$.

Exemple d'utilisation 2

Par défaut, les crochets s'étirent verticalement si besoin, mais si cela vous dérange, vous pouvez faire appel à la version étoilée de la macro comme dans l'exemple suivant

$$\displaystyle \left[\frac{x-1}{5+x^2} \right]_a^b = \left[\frac{x-1}{5+x^2} \right]_a^b.$$

$$\left[\frac{x-1}{5+x^2} \right]_a^b = \left[\frac{x-1}{5+x^2} \right]_a^b.$$

Fiches techniques

`\hook <macro> (3 Arguments)`

`\hook* <macro> (3 Arguments)`

— **Argument 1**: le contenu entre les crochets.

— **Argument 2**: la borne inférieure affichée en indice.

— **Argument 3**: la borne supérieure affichée en exposant.

9.6.2 L'opérateur crochet – 2^{de} version

Exemple d'utilisation 1

Vous pouvez utiliser `\vhook{F(x)}{a}{b}` au lieu de `\hook{F(x)}{a}{b}`.

Vous pouvez utiliser $F(x)|_a^b$ au lieu de $[F(x)]_a^b$.

Exemple d'utilisation 2

Tout comme avec la première version de l'opérateur crochet, vous pouvez utiliser une version étoilée pour empêcher l'étirement verticalement du trait vertical. Voici un exemple.

`\displaystyle \vhook{\frac{x - 1}{5 + x^2}}{a}{b}`
`= \vhook*{\frac{x - 1}{5 + x^2}}{a}{b}`.

$$\left. \frac{x - 1}{5 + x^2} \right|_a^b = \left. \frac{x - 1}{5 + x^2} \right|_a^b.$$

Fiches techniques

`\vhook <macro> (3 Arguments)`

`\vhook* <macro> (3 Arguments)`

- Argument 1 : le contenu avant le trait vertical.
- Argument 2 : la borne inférieure affichée en indice.
- Argument 3 : la borne supérieure affichée en exposant.

9.6.3 Intégrales multiples

Le package réduit les espacements entres des symboles \int successifs. Voici un exemple.

`\displaystyle`
`\int \int \int F(x;y;z) \dd{x} \dd{y} \dd{z}`
`= \int_{a}^{b} \int_{c}^{d} \int_{e}^{f} F(x;y;z) \dd{x} \dd{y} \dd{z}`

$$\int \int \int F(x;y;z) dx dy dz = \int_a^b \int_c^d \int_e^f F(x;y;z) dx dy dz$$

Remarque. Par défaut, L^AT_EX affiche $\int \int \int F(x;y;z) dx dy dz = \int_a^b \int_c^d \int_e^f F(x;y;z) dx dy dz$.

9.7 Tableaux de variation et de signe

Comment ça marche ?

Tout le boulot est fait par le package `tkz-tab` auquel on impose le choix d'une pointe de flèche plus visible. Nous vous demandons donc de vous reporter à la documentation de `tkz-tab` pour savoir comment s'y prendre.

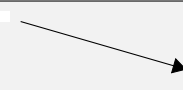
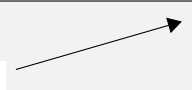
Un exemple de tableaux de signes

```
\begin{tikzpicture}
\tkzTabInit{$x$ / 1 , $\cos(x)$ / 1}{$0$, $\frac{\pi}{2}$, $\pi$}
\tkzTabLine{, +, z, -, }
\end{tikzpicture}
```

x	0	$\frac{\pi}{2}$	π
$\cos(x)$	+	0	−

Un exemple de tableaux de variation

```
\begin{tikzpicture}
\tkzTabInit{$x$ / 1 , $f(x)$ / 1.5}{$-\infty$, $p$, $+\infty$}
\tkzTabVar{+/, -/ $f(p)$, +/ }
\end{tikzpicture}
```

x	$-\infty$	p	$+\infty$
$f(x)$		$f(p)$	

Un exemple de tableaux de variation avec une dérivée

```
\begin{tikzpicture}
\tkzTabInit{$x$ / 1 , $\cos(x)$ / 1, $\sin(x)$ / 1.5}{$0$, $\frac{\pi}{2}$, $\pi$}
\tkzTabLine{, +, z, -, }
\tkzTabVar{-/ 0, +/ 1, -/ 0}
\end{tikzpicture}
```

x	0	$\frac{\pi}{2}$	π
$\cos(x)$	+	0	−
$\sin(x)$	0	1	0

9.8 Comparaison asymptotique de suites et de fonctions

9.8.1 Les notations \mathcal{O} et \mathcal{o}

Exemple d'utilisation 1

Vous pouvez utiliser les symboles `\bigO{}` et `\smallO{}` créés par Landau.

Vous pouvez utiliser les symboles \mathcal{O} et \mathcal{o} créés par Landau.

Exemple d'utilisation 2

Vous pouvez écrire `\bigO{x} \neq \smallO{x}` et `e^{t + \smallO{t}} = e^{\bigO{t}}`.

Vous pouvez écrire $\mathcal{O}(x) \neq \mathcal{o}(x)$ et $e^{t+\mathcal{o}(t)} = e^{\mathcal{O}(t)}$.

Fiches techniques

`\bigO <macro> (1 Argument)`

`\smallO <macro> (1 Argument)`

— Argument: si l'argument vide, il est ignoré, sinon il est mis entre des parenthèses après \mathcal{O} ou \mathcal{o} .

9.8.2 La notation Ω

Exemple d'utilisation 1

Vous pouvez utiliser le symbole `\bigomega{}` créé par Hardy et Littlewood.

Vous pouvez utiliser le symbole Ω créé par Hardy et Littlewood.

Exemple d'utilisation 2

`$f(n) = \bigomega{g(n)}` signifie :
`\exists (m, n_0)` tel que `$n \geqslant n_0$` implique `$f(n) \geqslant m g(n)$`.

$f(n) = \Omega(g(n))$ signifie : $\exists(m, n_0)$ tel que $n \geqslant n_0$ implique $f(n) \geqslant mg(n)$.

Fiche technique

`\bigomega <macro> (1 Argument)`

— Argument: si l'argument vide, il est ignoré, sinon il est mis entre des parenthèses après Ω .

9.8.3 La notation Θ

Exemple d'utilisation 1

Voici le dernier symbole `\bigtheta{}` qui peut rendre service.

Voici le dernier symbole Θ qui peut rendre service.

Exemple d'utilisation 2

`$f(n) = \bigtheta{g(n)}` signifie : $\exists (m, M, n_0)$ tel que $n \geq n_0$ implique $m g(n) \leq f(n) \leq M g(n)$.

$f(n) = \Theta(g(n))$ signifie : $\exists (m, M, n_0)$ tel que $n \geq n_0$ implique $mg(n) \leq f(n) \leq Mg(n)$.

Fiche technique

`\bigtheta` <macro> (1 Argument)

— Argument : si l'argument vide, il est ignoré, sinon il est mis entre des parenthèses après Θ .

10 Probabilité

10.1 Probabilité conditionnelle

Un exemple type

Écrire des probabilités conditionnelles :

```
$\probacond{p}{A}{B} = \probacond*{p}{A}{B}$  
                    = \probacond**{p}{A}{B}$  
                    = \dprobacond**{p}{A}{B}$.
```

Écrire des probabilités conditionnelles : $p_B(A) = p(A | B) = \frac{p(A \cap B)}{p(B)} = \frac{p(A \cap B)}{p(B)}$.

Fiche technique

`\probacond` <macro> (3 Arguments)

`\probacond*` <macro> (3 Arguments)

`\probacond**` <macro> (3 Arguments)

`\dprobacond**` <macro> (3 Arguments)

— Argument 1 : le nom de la probabilité.

— Argument 2 : l'ensemble dont on veut calculer la probabilité.

— Argument 3 : l'ensemble qui donne la condition.

10.2 Arbres pondérés

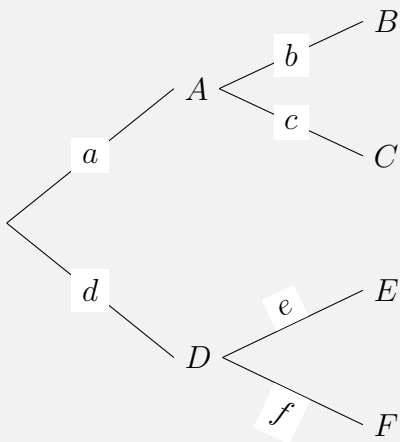
Que se passe-t-il en coulisse ?

Le gros du travail est fait par le package `forest` qui utilise `TiKz`. Ceci permet de faire des choses sympathiques comme dans le 2^e exemple ci-dessous.

Un exemple type

Dans le code suivant l'environnement `probatree` utilise en coulisse celui nommé `forest` du package `forest`. Des réglages spécifiques sont faits pour obtenir le résultat ci-après. A cela s'ajoute les styles spéciaux `pweight`, `apweight` et `bpweight` qui facilitent l'écriture des pondérations sur les branches⁴.

```
\begin{probatree}
[
  [A$, pweight = $a$
    [B$, pweight = $b$]
    [C$, pweight = $c$]
  ]
  [D$, pweight = $d$
    [E$, apweight = $e$]
    [F$, bpweight = $f$]
  ]
]
\end{probatree}
```



Un exemple décoré facilement

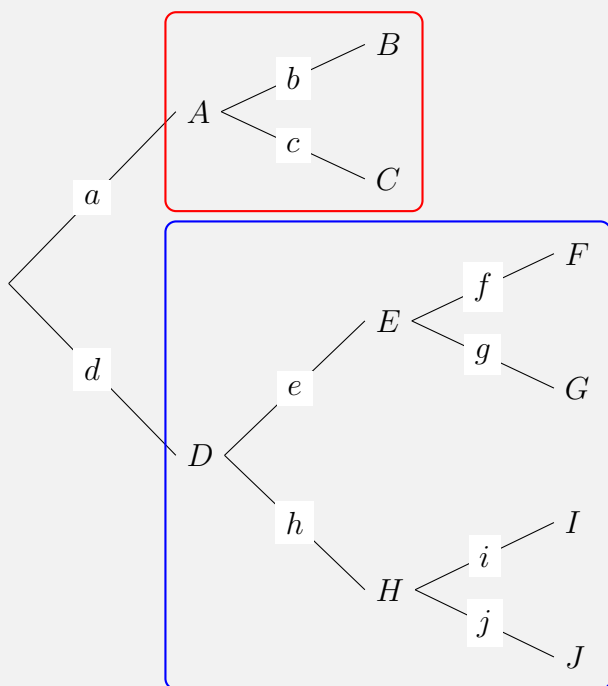
Via la clé `frame`, il est très aisé d'encadrer un sous-arbre comme le montre l'exemple suivant. Dans l'exemple ci-après nous utilisons la bidouille `{},s sep = 1.3cm` qui évite que les cadres se superposent.

4. `pweight` vient de « *probability* » et « *weight* » soit « *probabilité* » et « *poids* » en anglais. Quant à `a` et `b` au début de `apweight` et `bpweight` respectivement, ils viennent de « *above* » et « *below* » soit « *dessus* » et « *dessous* » en anglais.

```

\begin{probatree}
[{} ,s sep = 1.3cm
  [A$, pweight=$a$, frame=red
    [B$, pweight=$b$]
    [C$, pweight=$c$]
  ]
  [D$, pweight=$d$, frame=blue
    [E$, pweight=$e$
      [F$, pweight=$f$]
      [G$, pweight=$g$]
    ]
    [H$, pweight=$h$
      [I$, pweight=$i$]
      [J$, pweight=$j$]
    ]
  ]
]
\end{probatree}

```



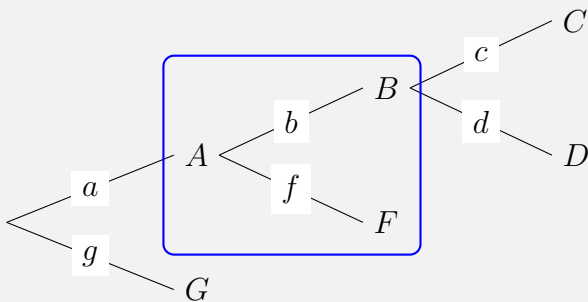
Un exemple décoré à la main

En utilisant la machinerie de `TikZ` il est facile de décorer un arbre de probabilité comme ci-dessous où le cadre s'appuie sur trois noeuds nommés. Notons que cet exemple n'est pas faisable avec la clé `frame`.

```

\begin{probatree}
[
  [$A$, pweight = $a$, name = left
    [$B$, pweight = $b$, name = topright
      [$C$, pweight = $c$]
      [$D$, pweight = $d$]
    ]
  [$F$, pweight = $f$, name = bottomright]
]
[$G$, pweight = $g$]
]
\node[draw = blue, thick, rounded corners, fit = (left)(topright)(bottomright)] {};
\end{probatree}

```



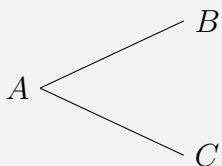
Un exemple de poids cachés partout

On peut cacher tous les poids via l'environnement étoilé `probatree*` sans avoir à retaper un arbre où les pondérations ont déjà été indiquées.

```

\begin{probatree*}
[$A$, pweight = $a$
  [$B$, pweight = $b$]
  [$C$, pweight = $c$]
]
\end{probatree*}

```



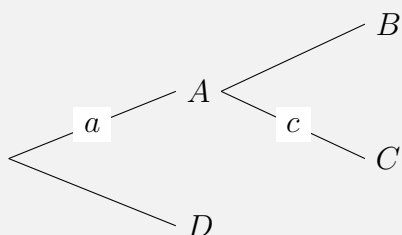
Un exemple de poids cachés localement

Pour ne cacher que certains poids, il faudra utiliser, à la main, le style `pweight*` comme dans l'exemple ci-dessous.

```

\begin{probatree}
[
  [$A$, pweight = $a$
    [$B$, pweight* = $b$]
    [$C$, pweight = $c$]
  ]
  [$D$, pweight* = $d$]
]
\end{probatree}

```



Fiches techniques

`probatree` <env>

`probatree*` <env>

- Contenu: un arbre codé en utilisant la syntaxe supportée par le package `forest`.
- Option "*pweight*": pour écrire un poids sur le milieu d'une branche.
- Option "*apweight*": pour écrire un poids au-dessus le milieu d'une branche.
- Option "*bpweight*": pour écrire un poids en-dessous du milieu d'une branche.
- Option "*frame*": pour encadrer un sous-arbre se terminant par des feuilles.

11 Arithmétique

11.1 Opérateurs de base

Pour des raisons d'expressivité des codes L^AT_EX, les opérateurs binaires `\divides`, `\notdivides` et `\modulo` ont été ajoutés comme alias de `\mid`, `\nmid` et `\bmod` respectivement qui sont proposés par le package `amssymb`.

`$10 \divides 150$` et `$10 \notdivides 154$` c'est mieux que `$10 \mid 150$` et `$10 \not\mid 154$`.

De plus, il est facile d'écrire que `$a \equiv b \modulo p \iff p \divides (a - b)$`.

`10 \mid 150` et `10 \nmid 154` c'est mieux que `10\mid150` et `10 \not\mid154`.

De plus, il est facile d'écrire que $a \equiv b \bmod p \iff p \mid (a - b)$.

Fiches techniques

`\divides` <macro> (Sans argument)

`\notdivides` <macro> (Sans argument)

`\modulo` <macro> (Sans argument)

11.2 Fractions continuées

11.2.1 Fractions continuées standard

Exemple d'utilisation

Dans l'exemple suivant, la notation en ligne semble être due à Alfred Pringsheim. La notation à gauche utilise toujours le maximum d'espace pour améliorer la lisibilité.

```
$ \contfrac{u_0 | u_1 | u_2 | \dots | u_n}
= \contfrac*{u_0 | u_1 | u_2 | \dots | u_n}$
```

$$u_0 + \frac{1}{u_1 + \frac{1}{u_2 + \frac{1}{\dots + \frac{1}{u_n}}}} = u_0 + \left| \frac{1}{u_1} \right| + \left| \frac{1}{u_2} \right| + \left| \frac{1}{\dots} \right| + \left| \frac{1}{u_n} \right|$$

Fiches techniques

`\contfrac <macro> (1 Argument)`

`\contfrac* <macro> (1 Argument)`

— **Argument**: tous les éléments de la fraction continuée séparés par des `|`.

11.2.2 Fractions continuées généralisées

Exemple d'utilisation

Voici comment écrire une fraction continuée généralisée.

```
$\displaystyle \contfracgene{a | b | c | d | e | f | \dots | y | z}
= \contfracgene*{a | b | c | d | e | f | \dots | y | z}$
```

$$a + \frac{b}{c + \frac{d}{e + \frac{f}{\dots + \frac{y}{z}}}} = a + \left| \frac{b}{c} \right| + \left| \frac{d}{e} \right| + \left| \frac{f}{\dots} \right| + \left| \frac{y}{z} \right|$$

Fiches techniques

`\contfracgene <macro> (1 Argument)`

`\contfracgene* <macro> (1 Argument)`

— **Argument**: tous les éléments de la fraction continuée généralisée séparés par des `|`.

11.2.3 Comme une fraction continuée isolée

Exemple d'utilisation

La raison d'être de la macro ci-dessous vient juste de son usage en interne.

Les fous (? \,) adorent vraiment écrire des choses comme $\frac{a}{b}$.

Les fous (?) adorent vraiment écrire des choses comme $\frac{a}{b}$.

Fiche technique

`\singlecontfrac` <macro> (2 Arguments)

— Argument 1: le pseudo numérateur.

— Argument 2: le pseudo dénominateur.

11.2.4 L'opérateur \mathcal{K}

Exemple d'utilisation 1

La notation suivante est proche de celle qu'utilisait Carl Friedrich Gauss.

```

$$\frac{\displaystyle \prod_{k=1}^n (b_k : c_k)}{c_1 + \frac{b_2}{c_2 + \frac{b_3}{\dots + \frac{b_n}{c_n}}}}$$

```

$$\mathcal{K}_{k=1}^n(b_k : c_k) = \frac{b_1}{c_1 + \frac{b_2}{c_2 + \frac{b_3}{\dots + \frac{b_n}{c_n}}}}$$

Remarque. La lettre \mathcal{K} vient de "kettenbruch" qui signifie "fraction continuée" en allemand.

Exemple d'utilisation 2

```

$$u_0 + \frac{1}{u_1 + \frac{1}{u_2 + \frac{1}{\dots + \frac{1}{u_n}}}}$$

```

$$u_0 + \mathcal{K}_{k=1}^n(1 : u_k) = u_0 + \frac{1}{u_1 + \frac{1}{u_2 + \frac{1}{\dots + \frac{1}{u_n}}}}$$

Fiche technique

`\contfracope` <macro> (Sans argument)

12 Algèbre

12.1 Polynômes, séries formelles et compagnie

12.1.1 Polynômes et fractions polynômiales

Exemple d'utilisation 1 : Polynômes

`\setpoly{\RR}{X}` est l'ensemble des polynômes à coefficients réels en la variable X , et `\setpoly{\RR}{X | Y | Z}` est l'ensemble des polynômes à coefficients réels en les variables X , Y et Z .

$\mathbb{R}[X]$ est l'ensemble des polynômes à coefficients réels en la variable X , et $\mathbb{R}[X;Y;Z]$ est l'ensemble des polynômes à coefficients réels en les variables X , Y et Z .

Exemple d'utilisation 2 : Fractions polynômiales

`\setpolyfrac{\QQ}{T}` et `\setpolyfrac{\QQ}{S_1 | S_2 | \dots | S_k}` permettent d'indiquer des ensemble de fractions polynomiales à coefficients rationnels.

$\mathbb{Q}(T)$ et $\mathbb{Q}(S_1; S_2; \dots; S_k)$ permettent d'indiquer des ensemble de fractions polynomiales à coefficients rationnels.

12.1.2 Séries formelles et leurs corps de fractions

Exemple d'utilisation 1 : Séries formelles

`\setserie{\CC}{X}` et `\setserie{\CC}{T | O | P}` permettent de travailler avec des séries formelles à coefficients complexes.

$\mathbb{C}[[X]]$ et $\mathbb{C}[[T; O; P]]$ permettent de travailler avec des séries formelles à coefficients complexes.

Exemple d'utilisation 2 : Corps des fractions de séries formelles

`\setseriefrac{\ZZ}{X}` et `\setseriefrac{\ZZ}{Z | T | O | P}` permettent de travailler avec des fractions de séries formelles à coefficients entiers.

$\mathbb{Z}((X))$ et $\mathbb{Z}((Z; T; O; P))$ permettent de travailler avec des fractions de séries formelles à coefficients entiers.

12.1.3 Polynômes de Laurent et séries formelles de Laurent

Exemple d'utilisation 1 : Polynômes de Laurent

`\setpolylaurent{\RR}{X} = \setpoly{\RR}{X | X^{-1}}` est l'ensemble des polynômes réels de Laurent en X . On propose de généraliser comme suit (notation non standard) :
`\setpolylaurent{\RR}{X_1 | X_2} = \setpoly{\RR}{X_1 | X_1^{-1} | X_2 | X_2^{-1}}`

$\mathbb{R}\{X\} = \mathbb{R}[X; X^{-1}]$ est l'ensemble des polynômes réels de Laurent en X . On propose de généraliser comme suit (notation non standard) : $\mathbb{R}\{X_1; X_2\} = \mathbb{R}[X_1; X_1^{-1}; X_2; X_2^{-1}]$

Exemple d'utilisation 2 : Séries formelles de Laurent

`\setserielaurent{\QQ}{X} = \setserie{\QQ}{X | X^{-1}}` est l'ensemble des séries formelles rationnelles de Laurent en X . On généralise via, notation non standard,
`\setserielaurent{\QQ}{X_1 | X_2} = \setserie{\QQ}{X_1 | X_1^{-1} | X_2 | X_2^{-1}}`

$\mathbb{Q}\{\{X\}\} = \mathbb{Q}[[X; X^{-1}]]$ est l'ensemble des séries formelles rationnelles de Laurent en X . On généralise via, notation non standard, $\mathbb{Q}\{\{X_1; X_2\}\} = \mathbb{Q}[[X_1; X_1^{-1}; X_2; X_2^{-1}]]$

12.1.4 Toutes les fiches techniques

`\setpoly` <macro> (2 Arguments)
`\setpolyfrac` <macro> (2 Arguments)
`\setserie` <macro> (2 Arguments)
`\setseriefrac` <macro> (2 Arguments)
`\setpolylaurent` <macro> (2 Arguments)
`\setserielaurent` <macro> (2 Arguments)

— Argument 1: l'ensemble auquel les coefficients appartiennent.

— Argument 2: cet argument est une suite de "morceaux" séparés par des barres |, chaque morceau étant une variable formelle.

12.2 Matrices

Comment ça marche ?

Tout le boulot est fait par le package `nicematrix` auquel on impose l'option `transparent`. Veuillez vous reporter à la documentation de `nicematrix` pour savoir comment s'y prendre.

Exemple 1 tiré de la documentation de nicematrix

```

 $\begin{pmatrix} 1 & \cdots & \cdots & 1 & \\ 0 & \ddots & & \vdots & \\ \vdots & \ddots & \ddots & \vdots & \\ 0 & \cdots & 0 & & 1 \end{pmatrix}$ 

```

$$\begin{pmatrix} 1 & \cdots & \cdots & 1 & \\ 0 & \ddots & & \vdots & \\ \vdots & \ddots & \ddots & \vdots & \\ 0 & \cdots & 0 & & 1 \end{pmatrix}$$

Exemple 2 tiré de la documentation de nicematrix

```

 $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ 

```

```

\tikz[remember picture,overlay]
\draw (mymatrix-2-2) circle (2mm) ;

```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & \textcircled{5} & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Exemple 3 tiré de la documentation de nicematrix

```

 $\left( \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{array} \right)$ 

```

$$\left(\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{array} \right)$$

13 Historique

Nous ne donnons ici qu'un très bref historique de `lymath` côté utilisateur principalement. Tous les changements sont disponibles uniquement en anglais dans le dossier `change-log` : voir le code source de `lymath` sur [github](https://github.com).

2019-10-21 Nouvelle version sous-mineure 0.6.3-beta.

- Pour les intervalles, `\CSinterval` a été déplacée vers le package `\lyalgo` disponible à l'adresse <https://github.com/bc-latex/ly-algo>.
- En logique, il y a eu les modifications suivantes.
 - `\eqdef**` a été supprimé. Voir la macro `\Store*` du package `\lyalgo`.
 - Différentes versions de l'opérateur \exists via `\existssone` et `\existmulti` avec leurs versions négatives `\nexistsone` et `\nexistmulti`.
 - Deux nouvelles macros `\eqplot` et `\eqappli` pour indiquer une équation de courbe et l'application d'une identité à des variables. Ceci s'accompagne de l'ajout des macros `\textopplot` et `\textopappli`.
 - Ajout des formes négatives `\niff`, `\nimplies` et `\nliesimp`.
 - Les décorations `cons`, `appli` et `choice` sont utilisables avec les opérateurs `\iff`, `\implies` et `\liesimp` et leurs formes négatives.
 - Une macro `\textoptest` a été ajoutée afin de rendre personnalisable tous les textes décorant les symboles.
- En analyse, il y a eu les renommages suivants.
 - `\hypergeo` est devenu `\seqhypergeo`.
 - `\suprgeo` est devenu `\seqsuprgeo`.
- En géométrie, `\notparallel` est devenu `\nparallel`.

2019-10-14 Nouvelle version sous-mineure 0.6.2-beta.

- En algèbre, il y a eu les renommages ci-dessous qui avaient été oubliés.
 - `\polyset` est devenu `\setpoly`.
 - `\polyfracsetp` est devenu `\setpolyfrac`.
 - `\serieset` est devenu `\setserie`.
 - `\seriefracset` est devenu `\setseriefrac`.
 - `\polylaurentset` est devenu `\setpolylaurent`.
 - `\serielaurentset` est devenu `\setserielaurent`.

2019-10-13 Nouvelle version sous-mineure 0.6.1-beta.

- En logique, la macro `\explain` possède maintenant un argument optionnel pour indiquer l'espacement avant le symbole. Les macros obsolètes `\explain*` et `\textexplainspacebefore` ont été supprimées.
- En probabilité, voici ce qui a évolué.
 - Les macros `\probacond` et `\probacond*` n'ont plus d'argument optionnel. Pour obtenir l'écriture fractionnaire, il faut utiliser `\probacond**` ou `\dprobacond**`.
 - Les environnements `probatree` et `probatree*` ont trois nouvelles clés. La clé `frame` permet d'encadrer un sous-arbre, et les clés `apweight` et `bpweight` permettent d'écrire des poids dessus/dessous une branche.
- Pour les ensembles, il y a eu les renommages suivants par souci de cohérence.
 - `\algeset` est devenu `\setalge`.
 - `\geoset` est devenu `\setgeo`.

- `\geneset` est devenu `\setgene`.
- `\probaset` est devenu `\setproba`.
- `\specialset` est devenu `\setspecial`.

2019-10-10 Nouvelle version mineure 0.6.0-beta.

- Des nouveaux outils spécifiques aux probabilités.
 - Les macros `\probacond` et `\probacond*` servent à écrire des probabilités conditionnelles.
 - Les environnements `probatree` et `probatree*` simplifient la production d'arbres probabilistes pondérés ou non.
- En géométrie, la macro `\notparallel` a été rajoutée.
- Un nouveau type d'intervalle pour l'informatique théorique via la macro `\CSinterval` afin d'obtenir quelque chose comme `a..b`.
- En logique, il y a deux nouvelles macros sémantiques `\neqid` et `\eqchoice`.

2019-09-27 Nouvelle version mineure 0.5.0-beta.

- Ajout des macros `\dsum` et `\dprod` qui sont vis à vis de `\sum` et `\prod` des équivalents de `\dfrac` pour `\frac`.
- En arithmétique, ajout des opérateurs `\divides`, `\notdivides` et `\modulo`.
- En géométrie, une nouvelle macro et un opérateur modifié.
 - `\pts` permet d'indiquer plusieurs points.
 - `\parallel` utilise des obliques pour symboliser le parallélisme au lieu de barres verticales.
- En logique, il y a les nouveautés suivantes.
 - La version doublement étoilée `\eqdef**` donne une deuxième écriture symbolique d'un symbole égal de type définition (*cette notation vient du langage B*).
 - Ajout de `\liesimp` comme alias de `\Longleftarrow`.
 - Les macros `\vimplies`, `\viff` et `\vliesimp` sont des versions verticales de `\implies`, `\iff` et `\liesimp`.
 - Comme pour les égalités, il existe les macros `\impliestest`, `\iffhyp` ... etc.

2019-09-06 Nouvelle version mineure 0.4.0-beta.

- Dans « *Logique et fondements* », différents types de signes d'inéquation et de non égalité pour des cas de test, d'hypothèse faite et de condition à vérifier.
- Intégration du package `tkz-tab` pour rédiger des tableaux de variations et de signes.
- Intégration du package `nicematrix` pour écrire des matrices.

2019-07-23 Nouvelle version mineure 0.3.0-beta.

- Une nouvelle section « *Logique et fondements* » a été ajoutée.
 - Trois types de signes = décorés sont proposés : voir les macros `\eqdef`, `\eqid` et `\eqtest`.
 - Via la macro `\explain`, il devient facile d'expliquer des étapes de raisonnement ou des calculs.
- Pour les ensembles, la macro `\fieldset` a été renommé `\algeset` et la macro `\PP` permet d'indiquer l'ensemble des nombres premiers.
- En géométrie, il y a quelques nouveautés.
 - La macro `\hangleorient` permet l'écriture d'angles orientés avec un chapeau en plus.
 - Les macros `\vangleorient` et `\vhangleorient` évite d'avoir à utiliser `\vect` lorsque l'on a juste des vecteurs simples nommés et non coefficientés.
 - De même pour les macros `\vdotprod`, `\vadotprod` et `\vcroosprod`.
- Ajout de `\lymathsubsep` qui définit le séparateur des arguments de second niveau.

2019-02-21 Nouvelle version mineure 0.2.0-beta.

- L'usage de `//` pour les macros-commandes avec un nombre quelconque d'arguments a été remplacé par celui de `|`.
- En géométrie, il y a diverses nouveautés.
 - Ajout de l'écriture de coordonnées, de produits scalaires et de produits vectoriels.
 - `\axis` a été correctement traduit en `\axes`.
 - Les macros `\gpaxis` et `\gpvaxis` deviennent `\paxes` et `\pvaxes` pour être cohérent avec `\pt` qui a remplacé l'ancien `\gpt`.
- En analyse, ajout de la macro commande étoilée `\derpow*` pour la gestion automatique des primes d'une dérivée.
- Une nouvelle section "algèbre" propose des macros pour écrire des ensembles de polynômes, de fractions polynomiales, de séries formelles, de fractions de séries formelles, et aussi de polynômes et de séries formelles de Laurent.
- Redéfinition de `\frac` et `\dfrac` pour obtenir des traits de fraction un peu plus longs.
- Ajout de `\lymathsep` qui définit le séparateur d'arguments.

2017-11-01 Nouvelle version mineure 0.1.0-beta : pour les ensembles, les fonctions et la géométrie, il y a eu des changements et l'ajout de nouveaux outils.

2017-10-21 Historique court de `lymath` ajouté au présent document.

2017-10-18 Nouvelle version "patchée" 0.0.2-beta : de nouveaux outils pour le calcul différentiel.

2017-10-06 Nouvelle version "patchée" 0.0.1-beta : de nouveaux outils pour l'arithmétique, la géométrie, le calcul intégral et le calcul différentiel.

2017-10-02 Première version 0.0.0-beta du package.